## **Содержание**

[**1. Техническое задание** 3](#_Toc28399285)

[**2. Исследовательская часть** 9](#_Toc28399305)

[2.1. Постановка задачи проектирования 9](#_Toc28399306)

[2.2 Пользовательские сценарии 10](#_Toc28399307)

[2.2.2 Сценарии использования. 14](#_Toc28399308)

[2.4. Обоснование выбора инструментов и платформы для разработки 18](#_Toc28399309)

[**3. Проектно-конструкторская часть** 21](#_Toc28399310)

[3.1 Разработка структуры программной системы 21](#_Toc28399311)

[3.2. Разработка пользовательского интерфейса 22](#_Toc28399312)

[3.2.1 Создание макета интерфейса. 22](#_Toc28399313)

[3.2.2 Создание React приложения. 24](#_Toc28399314)

[**4. Проектно-технологическая часть** 27](#_Toc28399315)

[4.1 Технологические решения, поддерживающие эксплуатационный цикл программы 27](#_Toc28399316)

[**Выводы** 30](#_Toc28399317)

[**Список источников** 31](#_Toc28399318)

**1. Техническое задание**

**1.1. Введение**

**1.1.1. Наименование программы**

“Web-приложение для организации частных и общественных мероприятий посредством краудфандинга”.

**1.1.2. Краткая характеристика области применения**

Приложение следует реализовать в качестве web-сервиса, предоставляющего возможность пользователям коллективной организации мероприятий, а также в качестве платформы для размещения рекламы помещений, участков, площадок и других объектов предназначенных для различного рода событий.

**1.2. Основания для разработки**

Разработка ведется на основании данного технического задания. Этим документом подтверждается процесс разработки программного обеспечения на тему “Web-приложение для организации частных и общественных мероприятий посредством краудфандинга”.

**1.3. Назначение разработки**

**1.3.1. Наименование темы разработки**

Разработка графического интерфейса web-приложения для организации частных и общественных мероприятий посредством краудфандинга.

**1.3.2. Функциональное назначение программы**

Функциональным назначением разработки указанного программного обеспечения является повышение эффективности работы, облегчения рабочего процесса, помощь в реализации товаров и услуг для самозанятых лиц, а также удобная торговая площадка для покупателей с возможностью поиска. Функциональным назначением программы является создание странички с информацией о возможном частном либо общественном мероприятии, где вероятные участники смогут голосовать за различные детали проведения данного мероприятия, в том числе посредством сбора средств. Также возможность размещения информации об объектах предназначенных для проведения мероприятий.

Критериями повышения эффективности будут служить:

* уменьшение затрат времени на поиск информации о площадках для проведения мероприятий;
* удобный интерфейс для формирования всей необходимой информации о готовящемся мероприятии;
* возможность общения организатора с участниками мероприятия;
* отслеживание формирования бюджета предстоящего мероприятия;
* возможность предварительного сбора средств на организацию мероприятия;

**1.3.3 Эксплуатационное назначение**

Подразумевается эксплуатация системы пользователями, организаторами мероприятий и арендодателями площадок для проведения мероприятий.

* 1. **Требования к программному изделию**

**1.4.1. Требования к функциональным характеристикам**

Программная система должна представлять собой работоспособный сайт, реализующее следующие функции:

* функция регистрации для пользователей, организаторов мероприятий и арендодателей;
* функция создания странички мероприятия;
* функция размещения информации о площадках для мероприятий;
* функция комментирования мероприятий и площадок;
* функция формирования идей для предстоящего мероприятия;
* поиск мероприятий и площадок;

**1.4.2. Требования к надежности**

Система должна работать стабильно, не допускается прерываний работы системы, вызванных ее внутренними ошибками. Интерфейс системы не должен содержать элементов, назначение которых неочевидно, которые вводят пользователя в заблуждение или не несут функционального значения.

**1.4.3. Условия эксплуатации**

Система предназначена для использования в нормальных условиях. Специальных требований по эксплуатации системе не предъявляется.

**1.4.4. Требования к составу и параметрам технических средств**

Специальных требований к составу и параметрам технических средств не предъявляется.

**1.4.5. Требования к информационной и программной совместимости**

Сайт должен быть совместим со всеми актуальными браузерами, в качестве системы управления базой данных будет использоваться PostgreSQL. Исходные коды (back-end) должны быть реализованы на языке JavaScript(ECMAScript) программная платформа Node.js. Визуальное представление (front-end) должно быть реализовано на JavaScript-библиотеке React для разработки пользовательских интерфейсов. В качестве среды разработки программы должна быть использована среда Microsoft Visual Studio Code.

**1.5. Требования к программной документации**

Разрабатываемое программное обеспечение не требует разработки руководства пользователя. Техническое задание должно быть оформлено в соответствии с требованиями ГОСТ.

**1.6. Стадии и этапы разработки**

Содержимое разделов настоящего технического задания может быть изменено и дополнено по согласованию с руководителем.

**1.6.1. Стадии разработки**

Разработка должна быть проведена в пять стадий стадии:

* Утверждение технического задания;
* Эскизный проект;
* Технический проект;
* Рабочий проект;
* Ввод в эксплуатацию

**1.6.2 Этапы разработки**

Порядок разработки программной системы разбит на следующие стадии:

**Этап утверждения технического задания**

* Обоснование необходимости разработки ПО;
* Исследование предметной области
* Разработка и утверждение технического задания

Результатом работы на данном этапе является согласованное и утвержденное техническое задание на разработку системы и описание предметной области.

Срок – 11.09.2019.

**Этап проектирования:**

* Разработка концептуальной схемы данных, прототипа или эскиза интерфейса;
* Разработка утверждение с заказчиком эскизного проекта;

Результатом работы на данном этапе является оформленная исследовательская часть, часть разработанных и реализованных алгоритмов функционирования приложения, прототип интерфейса, концептуальная и логическая схемы базы данных.

Срок – 24.09.2019.

**Этап технического проекта:**

* Оформление проектно-конструкторской части
* Разработка технической версии проекта
* Демонстрация разработанной версии приложения

Результатом работы на данном этапе будет являться оформленная проектно-конструкторская часть и рабочая техническая версия ПО

Срок - 03.11.2019.

**Этап рабочего проекта:**

* Разработка оставшегося функционала приложения
* Отладка и тестирование рабочей версии проекта
* Разработка программной документации

Результатом работы на данном этапе является оформленная проекто-технологическая часть, рабочая версия ПО и завершенная программная документация

Срок – 01.12.2019.

**Этап ввода в эксплуатацию:**

* Подготовка системы к развертыванию
* Первоначальная настройка системы

Результатом работы на данном этапе является спроектированный базовый функционал разрабатываемой системы.

**1.7. Порядок контроля и приемки**

Система в процессе разработки должна быть протестирована. Приёмка осуществляется преподавательским составом кафедры СОИ КФ МГТУ им. Н.Э.Баумана.

**2. Исследовательская часть**

**2.1. Постановка задачи проектирования**

Разрабатываемое приложение, в качестве основной функции, должно предоставлять возможность коллективного формирования информации и программы предстоящего мероприятия, с возможностью выбора места проведения на основании данных предоставленными арендодателями, а также возможностью коллективного сбора средств для его реализации.

Для хранения информации о предоставляемых площадках, мероприятиях, личных данных и комментариях пользователей будет использоваться реляционная база данных PostgreSQL. В качестве серверной части будет использоваться платформа Node.js, а в качестве клиентской – библиотека для разработки пользовательского интерфейса React.js.

Структурное проектирование системы осуществляется с помощью составления концептуальной и логической моделей хранимых данных. Описание этапов проектирования архитектуры приложения сопровождается описанием на естественном языке.

Необходимо разработать web-платформу целью которой является возможность взаимодействия организаторов с будущими участниками мероприятий, что поможет учесть пожелания целевой аудитории и по возможности реализовать самые популярные из них, в том числе основываясь на общей сумме взносов пользователей. Система должна обеспечивать удобный интерфейс организаторам для полноты формирования информации о предстоящем мероприятии, с возможностью поиска сдаваемых в аренду площадок, и также предоставлять участникам интерфейс предложения идей для мероприятия.

**2.2 Пользовательские сценарии**

В таблицах 1 – 11 представлены основные пользовательские сценарии поведения системы.

Таблица 1 – Сценарий регистрации пользователя

|  |  |
| --- | --- |
| **Название** | Регистрация |
| **Краткое описание** | Пользователь регистрируется в системе для дальнейшего взаимодействия с ней. |
| **Участники** | Пользователь, система |
| 1. Базовый сценарий 2. Пользователь открывает сайт. 3. Пользователь нажимает кнопку «Регистрация» 4. Система отправляет пользователю форму регистрации. 5. Пользователь вводит данные во все поля формы. 6. Система отправляет данные на сервер для занесения пользователя в базу данных. | |
| **Постусловие** | Пользователь зарегистрирован в системе. |
| **Расширения:** | |
| **1** | Пользователь заполнил не все обязательные поля.  Система сообщит об ошибке и попросит их заполнить. |
| **2** | Имя пользователя занято  Система сообщит об ошибке и попросит выбрать другое. |
| **3** | Пароль не соответствует требованиям.  Система сообщит об ошибке и попросит выбрать другой. |

Таблица 2 – Сценарий входа в систему

|  |  |
| --- | --- |
| **Название** | Вход в систему |
| **Краткое описание** | Пользователь входит в систему под своим логином и паролем для дальнейшего взаимодействия с ней |
| **Участники** | Пользователь, система |
| 1. Базовый сценарий 2. Пользователь открывает сайт и нажимает на кнопку «Войти в систему». 3. Система отправляет пользователю форму авторизации. 4. Пользователь вводит имя пользователя и пароль. 5. Система проверяет корректность ввода логина и пароля. 6. Пользователь авторизован в системе. | |
| **Постусловие** | Пользователь авторизовался в системе |
| **Расширения:** | |
| **1** | Неверный логин или пароль  Система уведомит об ошибке и попросит ввести корректные данные. |

Таблица 3 – Сценарий создания страницы мероприятия

|  |  |
| --- | --- |
| **Название** | Создание страницы мероприятия |
| **Краткое описание** | Организатор заполняет информацию о запланированном им мероприятии. |
| **Участники** | Организатор, система |
| 1. Организатор на странице личного кабинета, в разделе «мои мероприятия» нажимает на кнопку «создать мероприятие». 2. Система отправляет организатору форму заполнения информации о мероприятии. 3. Организатор заполняет необходимую информацию о запланированном мероприятии. 4. Система отправляет данные на сервер для занесения информации о мероприятии в базу данных. | |
| **Постусловие** | Организатор создал мероприятие |
| **Расширения:** | |
| **1** | Организатор заполнил не все обязательные поля.  Система сообщит об ошибке и попросит их заполнить. |
| **2** | Организатор установил дату проведения ниже текущей.  Система сообщит об ошибке и предложит установить корректную дату. |

Таблица 4 – Сценарий размещения информации о сдаваемой в аренду площадке

|  |  |
| --- | --- |
| **Название** | Размещение информации о площадке |
| **Краткое описание** | Арендодатель заполняет информацию о площадке для мероприятий. |
| **Участники** | Арендодатель, система |
| 1. Арендодатель на странице личного кабинета, в разделе «мои площадки» нажимает на кнопку «создать объявление». 2. Система отправляет арендодателю форму заполнения информации о площадке. 3. Арендодатель заполняет необходимую информацию о запланированном мероприятии. 4. Система отправляет данные на сервер для занесения информации о площадке в базу данных. | |
| **Постусловие** | Арендодатель разместил объявление о сдаваемой в аренду площадке |
| **Расширения:** | |
| **1** | Арендодатель заполнил не все обязательные поля.  Система сообщит об ошибке и попросит их заполнить. |

Таблица 5 – Сценарий добавления участниками идеи

|  |  |
| --- | --- |
| **Название** | Участники добавляют идею для мероприятия |
| **Краткое описание** | На странице мероприятия, будущие участники размещают свои пожелания для мероприятия. |
| **Участники** | Участник мероприятия, система |
| 1. Участник на странице мероприятия, в котором участвует, нажимает на кнопку «Добавить идею» 2. Система отправляет участнику форму заполнения информации о идею. 3. Участник заполняет необходимую информацию о своей идеи. 4. Система отправляет данные на сервер для занесения информации об идеи в базу данных. | |
| **Постусловие** | Участник добавил идею |
| **Расширения:** | |
| **1** | Участник заполнил не все обязательные поля.  Система сообщит об ошибке и попросит их заполнить. |

Таблица 5 – Сценарий голосования за идею.

|  |  |
| --- | --- |
| **Название** | Участники мероприятия голосуют за идею |
| **Краткое описание** | На странице мероприятия, будущие участники голосуют за понравившуюся им идею. |
| **Участники** | Участник мероприятия, организатор, система |
| 1. Участник на странице мероприятия, в котором участвует, в разделе «идеи», на понравившейся идеи нажимает кнопку «мне нравится». 2. Система сохраняет количество оценок для каждой идеи в базе данных. 3. Организатор выбирает наиболее популярные из них и нажимает на кнопку «Добавить на реализацию». 4. Система отправляет организатору форму заполнения дополнения информации об идеи. 5. Организатор проводит оценивает идею и заполняет графу ее стоимости. 6. Система переносит информацию об идеи в раздел расходов на мероприятие в базе данных. | |
| **Постусловие** | Организатор подтвердил идею |

# 2.2.2 Сценарии использования.

После регистрации на сайте пользователь может принимать участие в общественных мероприятиях или быть приглашенным на частное мероприятие. У пользователей есть возможность просматривать списки общественных мероприятий и сдаваемых в аренду площадках. Пользователь в роли участника может оставлять комментарии о мероприятиях, в которых он принимает непосредственное участие, так же предлагать идеи для расширения программы будущего мероприятия и совершить необходимый взнос средств если таковой требуется.

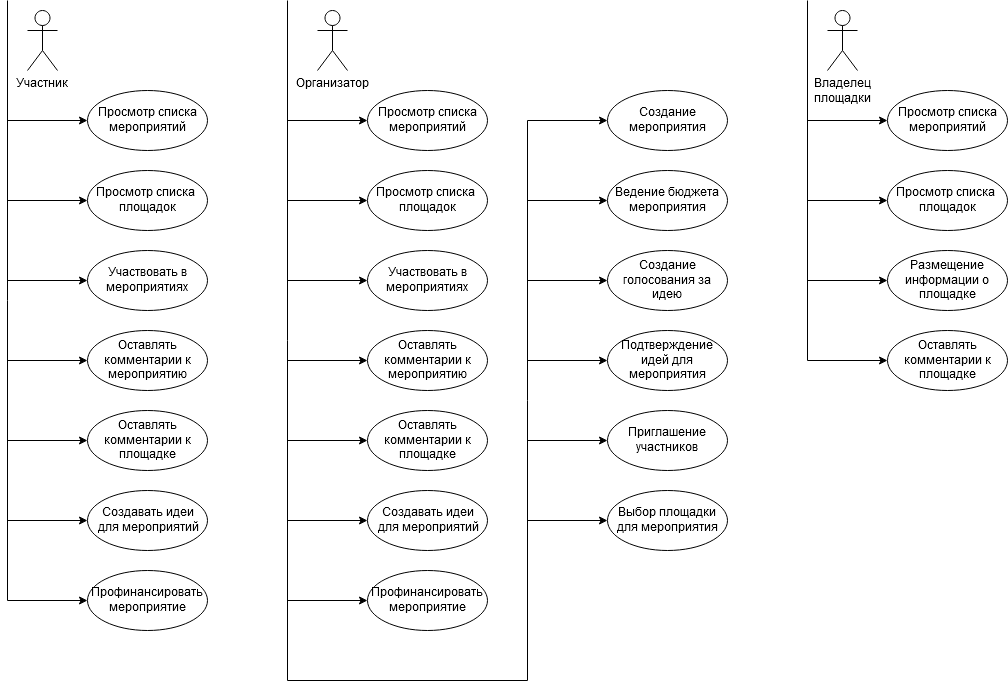


Рисунок 2 – UML диаграмма пользовательских сценариев для покупателя.

У пользователя есть возможность заполнить всю необходимою личную информацию профиля для того, чтобы в последствии как организатор создавать страницы предстоящих мероприятий и приглашать бедующих участников. Если для реализации мероприятия необходимо привлечение средств участников, то конечный бюджет формируется организатором в виде отдельных статей расходов, просмотр которых будет доступен участникам. Организатор выносит идеи пользователей на голосование, в котором принимают будущие участники мероприятия и проводит анализ стоимости наиболее востребованных идей.

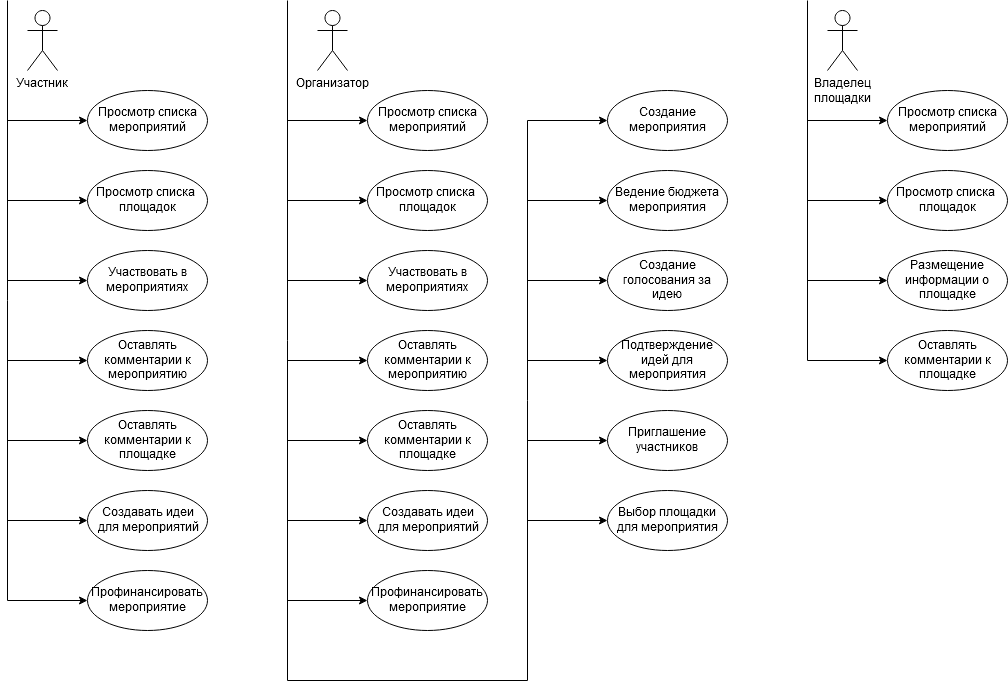


Рисунок 3 – UML диаграмма пользовательских сценариев для организатора.

Как владелец площадок, пользователь может размещать информацию о собственных сдаваемых в аренду площадках. Также комментировать мероприятия в том числе для предоставления информации об опыте работы с конкретными организаторами другим арендодателям.

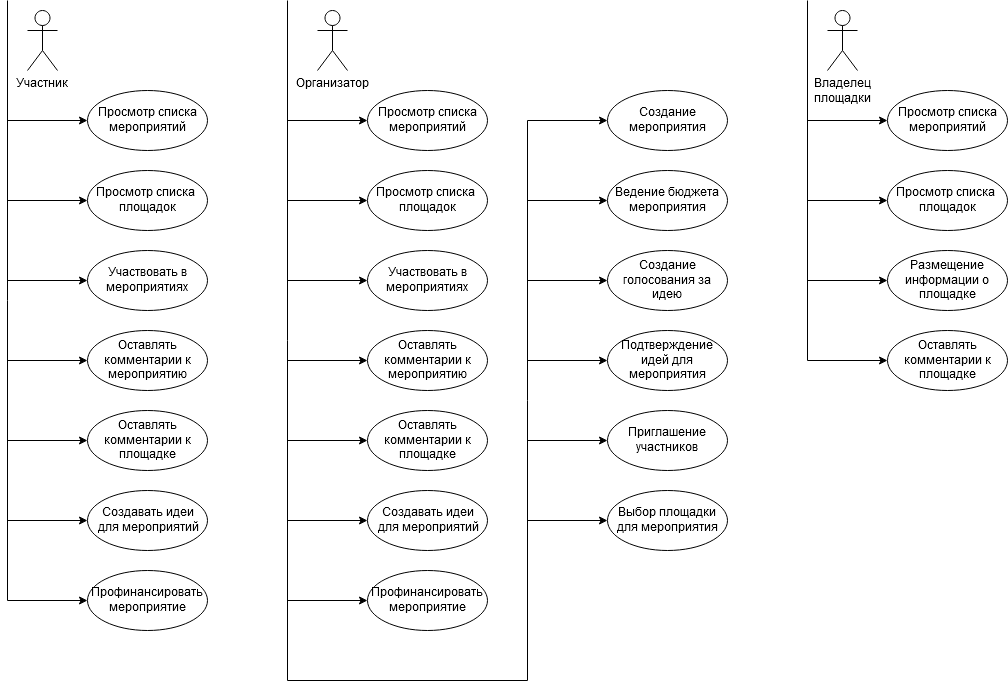


Рисунок 3 – UML диаграмма пользовательских сценариев для владельца площадок.

# 2.4. Обоснование выбора инструментов и платформы для разработки

Figma — кроссплатформенный онлайн-сервис для дизайнеров интерфейсов и веб-разработчиков.

Figma позволяет разрабатывать интерфейсы в онлайн-приложении. У Figma две ключевые особенности: доступ к макету прямо из окна браузера и возможность совместной работы над документами.

Одной из ключевых возможностей Figma являются компоненты — это элементы пользовательского интерфейса, которым можно задавать общие стили — и затем быстро менять их во всем макете сразу.

React.js — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.

React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость.

Однонаправленная передача данных — свойства передаются от родительских компонентов к дочерним. Компоненты получают свойства как множество неизменяемых значений, поэтому компонент не может напрямую изменять свойства, но может вызывать изменения через callback функции. Такой механизм называют «свойства вниз, события наверх».

React использует виртуальный DOM. React создает кэш структуру в памяти, что позволяет вычислять разницу между предыдущим и текущим состояниями интерфейса для оптимального обновления DOM браузера. Таким образом программист может работать со страницей, считая, что она обновляется вся, но библиотека самостоятельно решает, какие компоненты страницы необходимо обновить.

JavaScript XML (JSX) — это расширение синтаксиса JavaScript, которое позволяет использовать похожий на HTML синтаксис для описания структуры интерфейса. Как правило, компоненты написаны с использованием JSX, но также есть возможность использования обычного JavaScript. JSX напоминает другой язык, созданный в компании Фейсбук для расширения PHP, XHP.

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера.

В качестве основного модуля используется Express — это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Имеет в своем распоряжении множество служебных методов HTTP и промежуточных обработчиков, для создания надёжного API. Предоставляет тонкий слой фундаментальных функций веб-приложений, которые не мешают работать с любыми существующими функциями Node.js.

В качестве связующего звена между сервером и базой данных используется node-postgres (pg.js) — это коллекция модулей node.js для взаимодействия с вашей базой данных PostgreSQL. Он поддерживает обратные вызовы, обещания(promises), асинхронные вызовы(async/await), пул соединений, связываемые переменные, курсоры, результаты потоковой передачи, привязки C/C++, преобразование типов и многое другое.

PostgreSQL — это свободно распространяемая объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

* Надежность PostgreSQL.
* Производительность PostgreSQL основывается на использовании индексов, интеллектуальном планировщике запросов, тонкой системы блокировок, системе управления буферами памяти и кэширования, превосходной масштабируемости при конкурентной работе.
* Поддержка SQL
* PostgreSQL имеет очень богатый набор встроенных функций и операторов для работы с данными, полный список которых можно посмотреть в документации.
* Простота использования всегда являлась важным фактором для разработчиков.
* pgAdmin (GNU Artistic license) предоставляет удобный интерфейс для работы с базами данных PostgreSQL.

Для тестирования запросов к API используется Postman — приложение, которое упрощает разработку, ускорение и улучшение разработки API. Доступные для загрузки в Windows, MacOS, Chrome и Linux. Основные фукции — это история сохраненных API-запросов, подробная документация по API-интерфейсам и веб-видимости, гибкий мониторинг API, макеты серверов, поддерживающие разработку разделенных стеков, и т.д.

**3. Проектно-конструкторская часть**

**3.1 Разработка структуры программной системы**

Приложение представляет собой клиент-серверное приложение, которое является трехуровневым. Клиент-серверное приложение — это вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение.

Трехуровневое клиент-серверное (рис. 5) приложение содержит в себе:

* Представление данных (клиент). В данной курсовой работе клиентом будет являться веб-браузер, который будет передавать веб-серверу запросы на получение HTML-страниц, изображений, файлов и других данных, обозначенных URL-адресами.
* Сервер приложений располагается на втором уровне, на нём сосредоточена большая часть бизнес-логики. Вне его остаются только фрагменты, экспортируемые на клиента, а также элементы логики, погруженные в базу данных. Реализация данного компонента обеспечивается связующим программным обеспечением.
* Сервер базы данных, выполняет обслуживание и управление базой данных и СУБД, а также отвечает за целостность и сохранность данных и обеспечивает операции ввода-вывода при доступе клиента к информации.

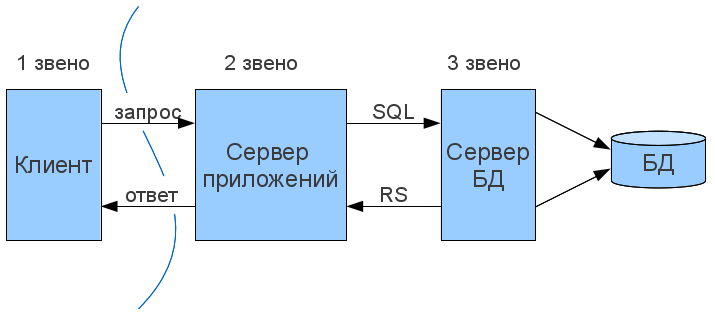


Рисунок 5 – Компоненты системы

Последовательность разработки системы:

1) Проектирование базы данных.

2) Создание логики серверной части.

3) Создание интерфейса.

4) Заполнение базы данных.

5) Тестирование.

**3.2. Разработка пользовательского интерфейса**

Разработку пользовательского интерфейса можно разделить на несколько этапов:

• Создание формы регистрации

• Создание формы для входа в аккаунт (профиль на сайте)

• Создание главной страницы сайта

• Создание страницы для товаров

• Создание адаптивного интерфейса

**3.2.1 Создание макета интерфейса.**

При проектировании макета сайта с помощью Figma активно использовалась возможность создавать элементы интерфейсов в качестве компонентов, такие как: AppBar (меню приложения), кнопки, карточки с информацией и т.д. Данный подход позволяет сократить время при повторном создании одинаковых компонентов, а так же вносить, при необходимости, изменения только в родительский компонент, тем самым изменив стиль всех его наследников.

В качестве демострации работы с компонентами (рис. 6, 7), продемострированна реализация темной темы для списка карточек с информацией о сдаваемых в аренду площадках. Слева отображено дерево элементов компонентов, родительский элемент отмечен заполненным ромбом.

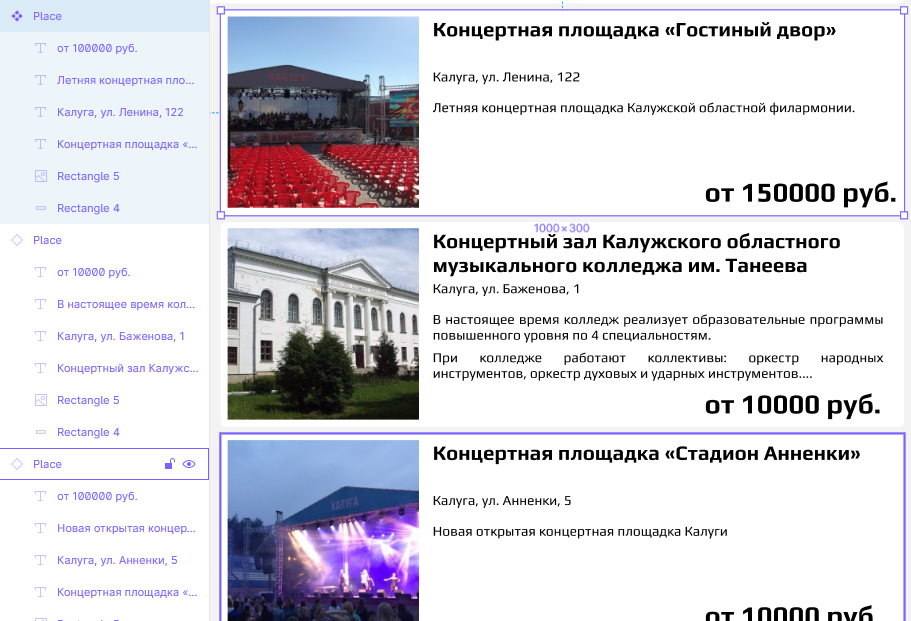


Рисунок 6 – Макет списка сдаваемых в аренду помещений.

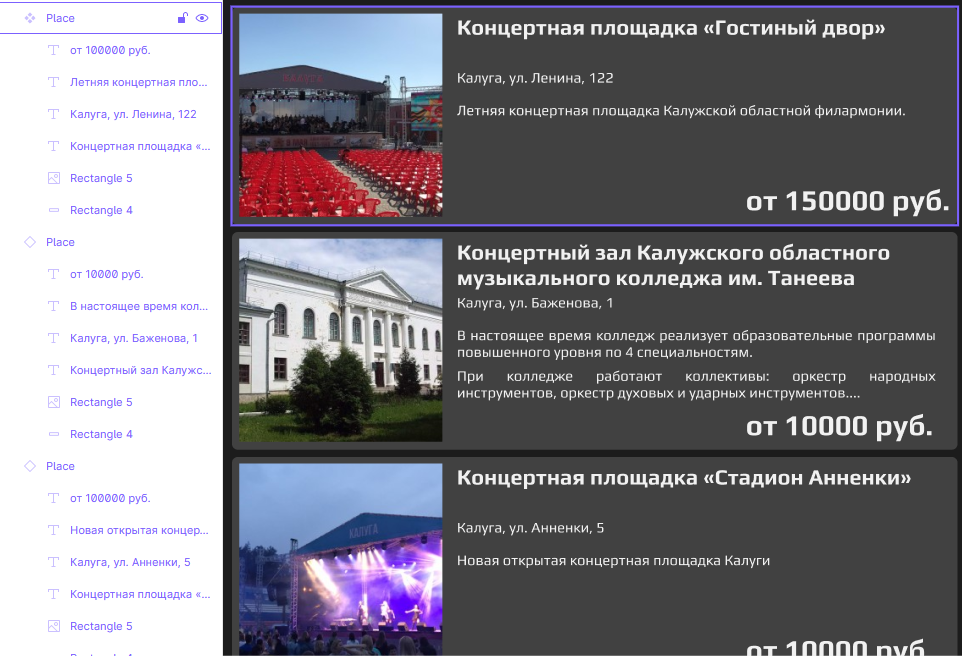


Рисунок 7 – Макет списка сдаваемых в аренду помещений (темная тема).

Как видно из приведенно выше примера компоненты наследники могут иметь отличное от родителей содержимое, но при этом наследуют его стили. В данном случае для создания темной версии карточек с информацией потребовалось всего два изменения у родительского элемента — изменение цвета текста и цвета фона.

**3.2.2 Создание React приложения.**

На основе разработанного макета были сформированы основные React-компоненты приложения. Для задания CSS стилей компонентов использовалась библиотека styled-components, что позволяет создать стилизованный компонент готовый к интеграции в DOM-модель, на основе уже существующих компонентов, либо простых HTML элементов.

Листинг 1 – описания стиля для меню приложения при помощи styled-components.

const AppBar = styled.div`

  background-color: #9e2020;

  height: 60px;

  width: 100%;

  margin: 0;

  display: -webkit-flex;

  display: flex;

`;



Рисунок 8 – Компонет AppBar.

На примере стартовой страницы приложения можно выделить следующие компоненты (рис. 9, 10):

* Header (AppBar) – содержит в себе основную навигацию и кнопку регистрации. Присутствует на всех страницах приложения.
* SearchArea – превью проводимых общественных мероприятий со строкой поиска.
* Description – краткое описание работы приложение.
* DescriptionCard – карточка содержащая в себе расширенную информацию об одной из возможности приложения.
* Footer – блок в нижней части страницы, куда вынесена полезная, но не первостепенная информация.

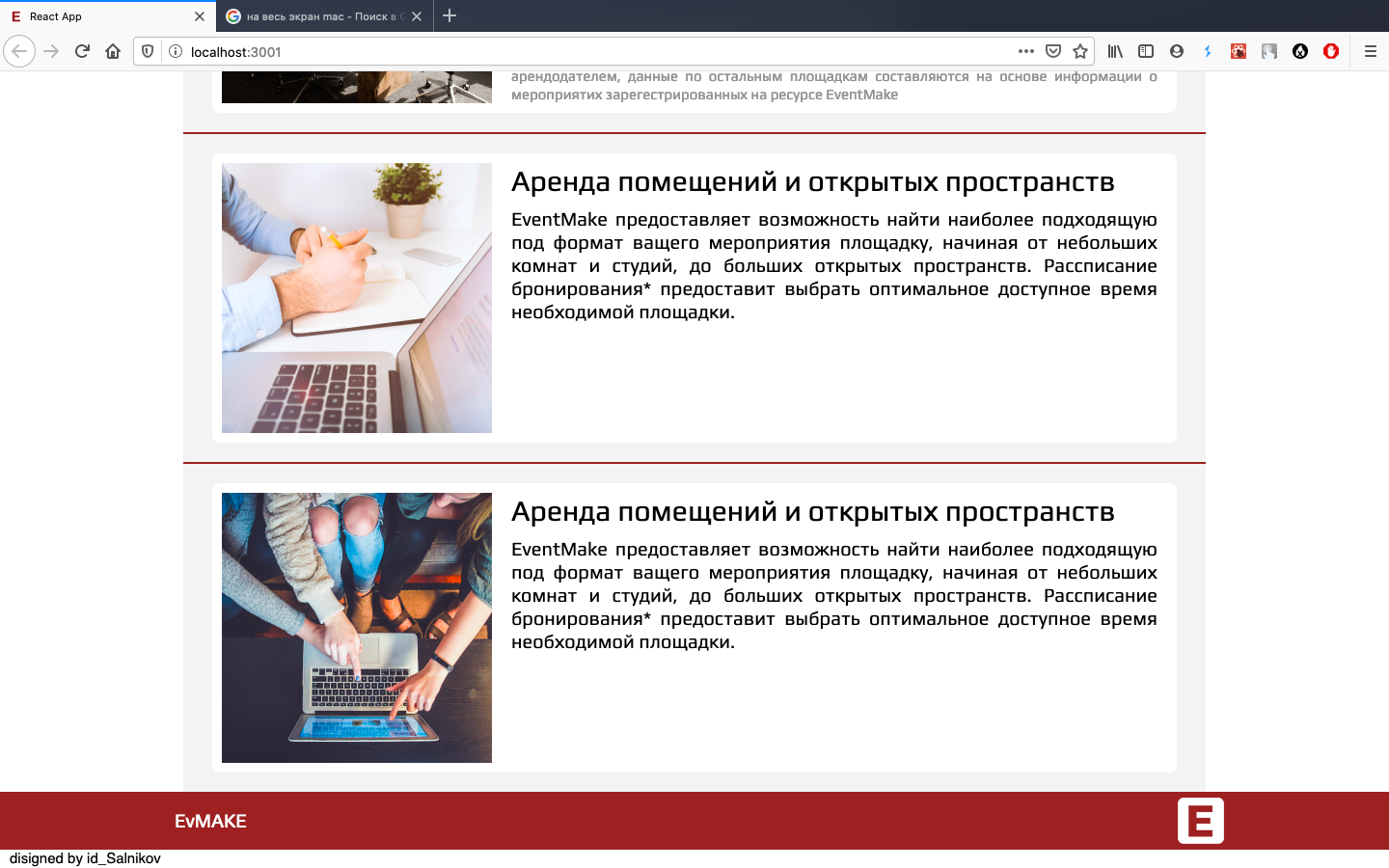


Рисунок 9 – Стартовая страницца сайта.

Листинг 2 – Основной компонент начальной страницы.

class Header extends Component {

  render() {

    return (

      <>

        <AppBar>

          <LogoP>EvMAKE</LogoP>

          <WrapBtn>

            <BtnText>Площадки</BtnText>

            <BtnText>Мероприятия</BtnText>

            <BtnText>О нас</BtnText>

          </WrapBtn>

          <WrapBtn end>

            <BtnText rounded>Войти</BtnText>

            <BtnText>

              <img src={accountImg} alt="AccountImg" />

            </BtnText>

          </WrapBtn>

        </AppBar>

        <SearchArea></SearchArea>

        <GridWithBottomRedLine></GridWithBottomRedLine>

        <Footer />

      </>

    );

  }

}

В качестве взаимодействия с веб-сервером использовался интерфейс JavaScript Fetch API, для работы с запросами и ответами HTTP. Он также предоставляет глобальный метод fetch(), который позволяет легко и логично получать ресурсы по сети асинхронно. При работе React-приложения данные необходимо получить до того как начнется построение интерфейса, для этого есть специальный метод componentDidMount(). В данном приложении данные передаются в формате JSON и записываются в состояние компонента (листинг 3).

Листинг 3 – Запрос данных у веб-сервера Node.

state ={

    data: []

  }

  componentDidMount() {

    fetch('http://localhost:4000/events/')

    .then(response => response.json())

    .then(jsonData => this.setState({data: jsonData}))

  }

**4.** **Проектно-технологическая часть**

**4.1 Технологические решения, поддерживающие эксплуатационный цикл программы**

При проектировании системы с использованием библиотеки React.js и платформы Node.js был использован свободный фреймворк для веб-API Express. С их помошью можно реализовать архитектурный паттерн Model-View-Controller.

Архитектура Model-View-Controller – это схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

В данной работе паттерн MVC включает ряд компонентов:

* Модель пре доступ к данным, обрабатывается слоем работы с базой данных. Он осуществляет запросы в базу данных и проверку на корректность. Модель не зависит от представления, а следовательно, не имеет возможности визуализировать данные и не имеет точек взаимодействия с пользователем. Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом уведомляя «наблюдателей».
* Представление (View) отвечает за получение необходимых данных из модели и отправляет их пользователю.
* Контроллер обеспечивает общение между пользователем и системой. Он контролирует и направляет данные от пользователя к системе и наоборот.
* Система маршрутизация как дополнительный компонент сопоставляет запросы с маршрутами и выбирает для обработки запросов определенный контроллер.

В общем случае, когда к приложению приходит запрос, система маршрутизации выбирает нужный контроллер для обработки запроса. Контроллер обрабатывает запрос. В процессе обработки он может обращать к данным через модели и для рендеринга ответа использовать представления. Результат обработки контроллера отправляет в ответ клиенту. В данной работе ответом от веб-API будут являться JSON объекты при запросе данных или код ответа HTTP при их загрузке на сервер.

Модель данного паттерна изображена на рисунке 7.

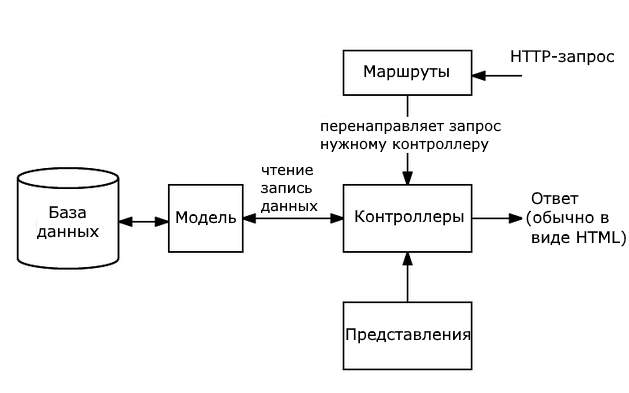


Рисунок 11 – Паттерн Model-View-Controller для веб-приложений Node.js

При проектировании клиентской части необходимо учитывать жизненный цикл React-компонентов. React позволяет определять компоненты как классы или функции. Единственный обязательный метод в подклассе React.Component — render(). Все остальные методы, являются необязательными. Каждый компонент имеет несколько «методов жизненного цикла» (рис. 12). Переопределение такого метода позволяет выполнять код на конкретном этапе этого процесса.

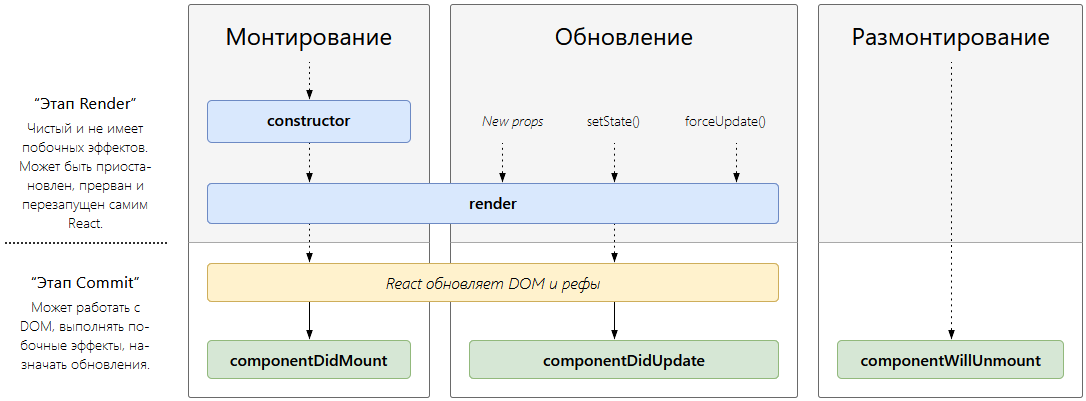


Рисунок 12 – Жизненный цикл компонентов React

* componentDidMount() вызывается сразу после монтирования (то есть, вставки компонента в DOM). В этом методе должны происходить действия, которые требуют наличия DOM-узлов. Это хорошее место для создания сетевых запросов.
* componentDidUpdate() вызывается сразу после обновления. Не вызывается при первом рендере.
* componentWillUnmount() вызывается непосредственно перед размонтированием и удалением компонента. В этом методе выполняется необходимый сброс: отмена таймеров, сетевых запросов и подписок, созданных в componentDidMount().

**Выводы**

В ходе выполнения данной курсовой работы была проанализированная выбранная предметная область, выбран инструментарий для реализации функционала веб-приложения, а также выведены UML-диаграммы сценариев работы пользователя.

Была реализована базовая часть веб-приложения, в которой была заложена функциональность для ее дальнейшего расширения. В ходе реализации веб-приложения были получены практические навыки работы с приложением для создания макетов Figma и библиотекой для веб-разработки React.js.

Таким образом цель курсовой работы достигнута, а все поставленные задачи выполнены.

**Список источников**

1. Абрамов Г.В. Проектирование информационных систем : учебное пособие / Г.В. Абрамов, И.Е. Медведкова, Л.А. Коробова. - Воронеж : Воронежский государственный университет инженерных технологий, 2012,172 с. URL://biblioclub.ru/index.php?page=book&id=141626
2. Диков, А.В. Веб-технологии HTML и CSS / А.В. Диков. – 2-е изд. – Москва: Директ-Медиа, 2012. – 78 с.: ил.,табл., схем. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=96968, по подписке
3. Золотов С.Ю. Проектирование информационных систем: учебное пособие/ ЗолотовС.Ю.— Томск: Томский государственный уни-верситет систем управления и радиоэлектроники, Эль Контент, 2013.— 88 c.— URL: <http://www.iprbookshop.ru/13965>
4. Рудинский И.Д. Технология проектирования автоматизированных систем обработки информации и управления. — М. : Горячая линия-Телеком, 2011. — 304 с. — URL: <http://e.lanbook.com/book/5191>
5. Сидоров В.Н., Сломинская Е.Н., Полникова Т.В., Макарова О.Ю. Оформление графической части выпускной квалификацион-ной работы. Учебное пособие. М.: МГТУ им. Н.Э. Баумана, 2016.
6. Сырецкий Г.А. Проектирование автоматизированных систем. Часть 1: учебное пособие/Сырецкий Г.А.— Новосибирск: Новоси-бирский государственный технический университет, 2014.— 156 c.— URL: <http://www.iprbookshop.ru/47714>
7. Стефанов СтоянС79 React.js. Быстрый старт. — СПб.: Питер, 2017. — 304 с.: ил. — (Серия «Бестселлеры O’Reilly»).
8. Форд Н., Хайгард М., де Ора Б. - 97 этюдов для архитекторов программных систем. - Пер. с англ. - СПб. Символ-Плюс, 2010. - 224 с., ил. ISBN 978-5-93286-176-9
9. Документация React.js [Электронный ресурс] / reactjs.org – Электрон. текстовые дан. – Режим доступа: https://ru.reactjs.org/docs/getting-started.html, свободный.
10. Документация приложения Figma [Электронный ресурс] / figma.com – Электрон. текстовые дан. – Режим доступа: https://help.figma.com/, свободный.