**Содержание**

[**1. Техническое задание** 3](#_Toc28243482)

[**2. Исследовательская часть** 9](#_Toc28243502)

[2.1. Постановка задачи проектирования 9](#_Toc28243503)

[2.2. Описание предметной области. 9](#_Toc28243504)

[2.2.1 Концептуальная схема данных. 11](#_Toc28243505)

[2.2.2 Сценарии использования. 12](#_Toc28243506)

[2.3. Перечень задач, подлежащих решению в процессе разработки 16](#_Toc28243507)

[2.4. Обоснование выбора инструментов и платформы для разработки 17](#_Toc28243508)

[**3. Проектно-конструкторская часть** 19](#_Toc28243509)

[3.1 Разработка структуры программной системы 19](#_Toc28243510)

[3.2 Логическая схема данных 20](#_Toc28243511)

[**4. Проектно-технологическая часть** 25](#_Toc28243512)

[4.1 Технологические решения, поддерживающие эксплуатационный цикл программы 25](#_Toc28243513)

**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 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;
* функция голосования за идею для предстоящего мероприятия среди участников;
* функция составления бюджета для 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.

# 2.2. Описание предметной области.

Мероприятие — определенное действо с привлечением широкого круга лиц. В рамках данной работы подразумевает как публичные, посещение которого доступно любому желающему, так и частные, доступ к которым осуществляется по приглашению организатора.

Организатор мероприятия — это человек, который планирует и проводит мероприятие, принимая на себя ответственность за творческие, технические и логистические элементы. Это включает в себя общий дизайн мероприятия, создание бренда, маркетинговую и коммуникационную стратегию, аудиовизуальную продукцию, написание сценариев, логистику, составление бюджета, ведение переговоров и обслуживание клиентов.Процесс планирования и координации мероприятия обычно называется планированием мероприятия и может включать в себя планирование бюджета, планирование, выбор площадки, получение необходимых разрешений, координацию транспортировки и парковки, организацию выступлений или конферансье, организацию обстановки, обеспечение безопасности мероприятия, организацию питания, координацию с сторонними поставщиками и планами на случай чрезвычайных ситуаций.

Арендодатель — физическое или юридическое лицо, которое предоставляет имущество арендатору за плату во временное пользование и владение. Сдавать в аренду имущество может только его собственник или лицо, уполномоченное на это собственником либо законом.

Краудфандинг — это способ привлечь средства на какой-то проект или в свой бизнес с помощью специальных интернет-площадок. Деньги можно взять в долг, обменять на долю вашей будущей прибыли или даже получить безвозмездно. Во всех случаях выдает их не банк и не государство, а другие люди или компании.

Идея для мероприятия — предложение от участников или организатора о расширения программы и используемых средств мероприятия, которое возможно повлечет дополнительные расходы.

Необходимо разработать web-приложение, для предоставления будущим участникам подробной информации о проводимом мероприятии напрямую от непосредственного организатора. Что может стать решением следующих проблем:

* Предварительный сбор средств с участников мероприятия;
* Составление оптимальной программы мероприятия на основе идей участников и информации о предварительно собранной суммы денег;
* Доступ участников к информации о статьях бюджета мероприятия.

# 2.2.1 Концептуальная схема данных.

На основании представленной информации в предметной области можно выделить несколько сущностей (рисунок 1): Пользователь, событие, идея, статья бюджета, платеж, площадка и комментарий.

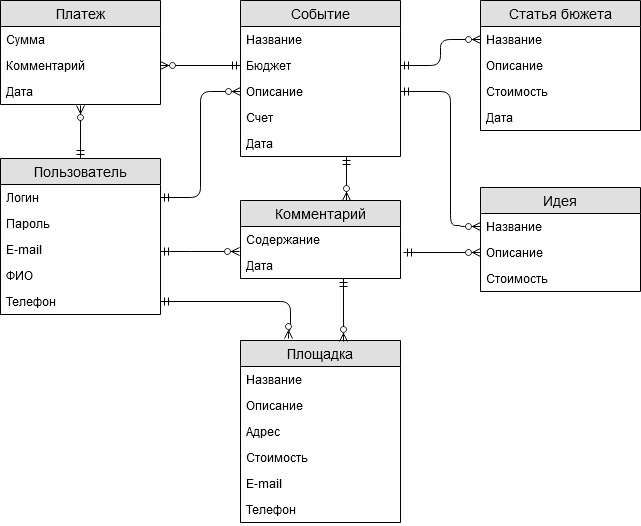


Рисунок 1 – Концептуальная модель данных

На рисунке 1 представлена концептуальная схема данных предметной области. Из данной схемы можно выделить следующие связи: существуют пользователи, которые могут создавать события в качестве организаторов, размещать информации о сдаваемой в аренду площадке в качестве арендодателей, учувствовать в мероприятиях и вносить средства на реализацию запланированного мероприятия. Участники и организаторы мероприятий могут оставлять комментарии к мероприятиям и арендуемым площадкам, создавать идеи для мероприятий. Так же будущим участникам будет доступна информация о статьях расходов на мероприятие. Арендодатели могут размещать информации о собственных сдаваемых в аренду площадках.

# 2.2.2 Сценарии использования.

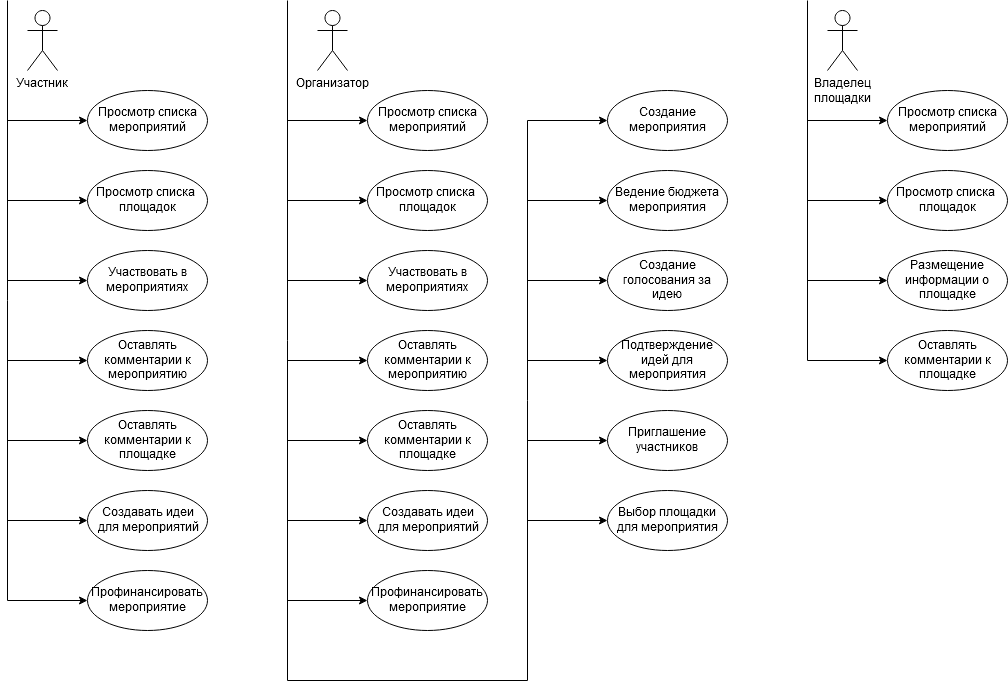


Рисунок 2 – UML диаграмма пользовательских сценариев для покупателя

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.

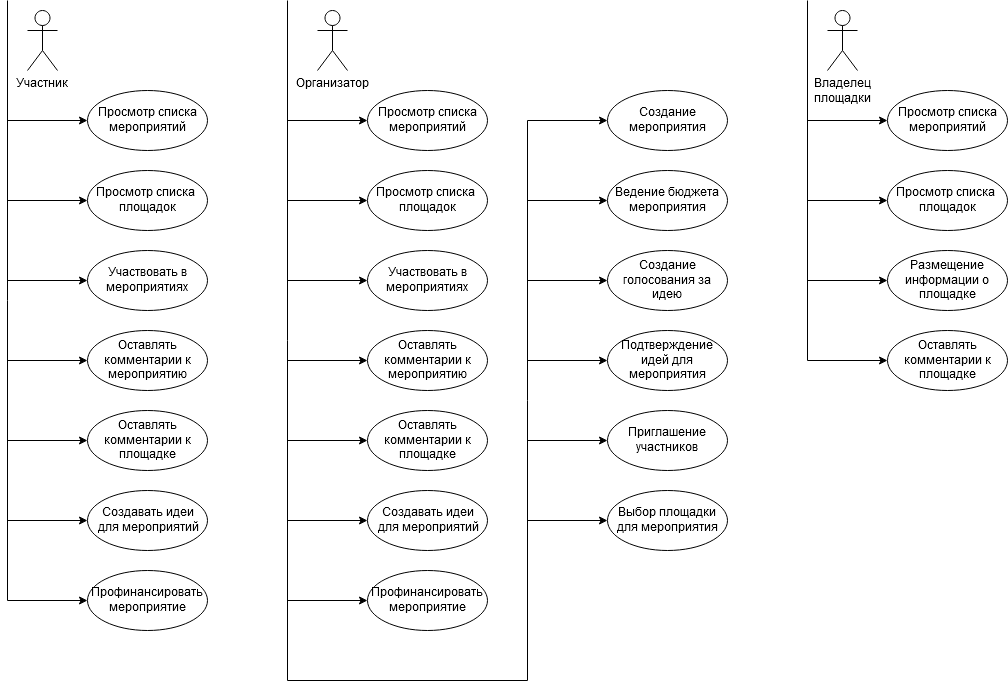


Рисунок 3 – UML диаграмма пользовательских сценариев для организатора.

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.

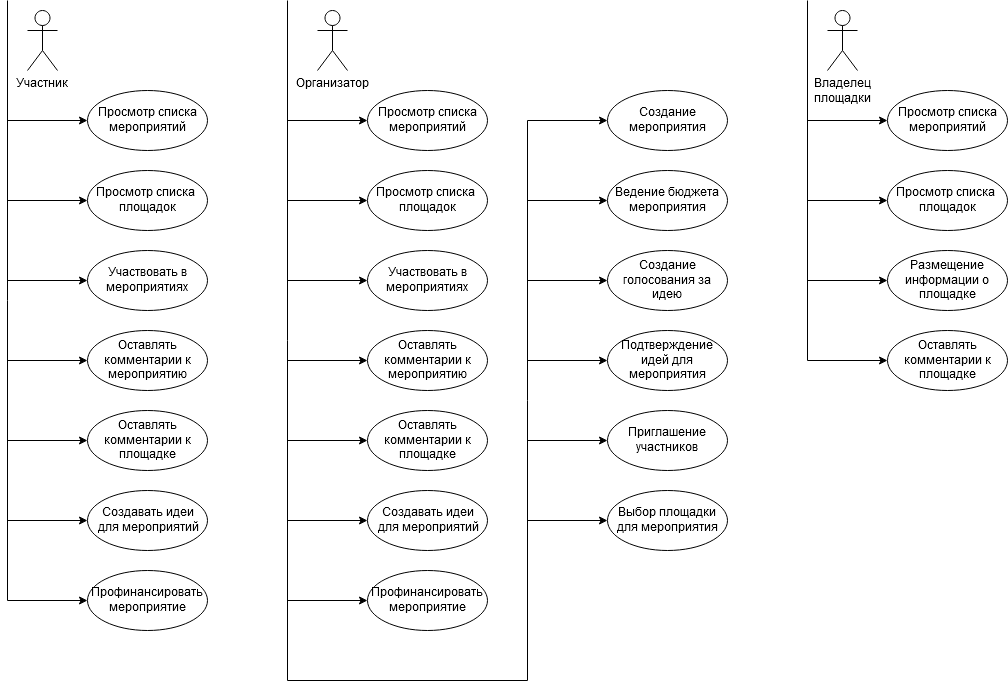


Рисунок 3 – UML диаграмма пользовательских сценариев для владельца площадок.

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.

# 2.3. Перечень задач, подлежащих решению в процессе разработки

Для создания системы необходимо:

* Составить техническое задание.
* Провести исследование предметной области;
* Сформировать концептуальную и логическую модели хранимых данных;
* Подготовить перечень необходимого функционала;
* Оформить описание процессов, протекающих в системе;
* Спроектировать макет сайта;
* Организовать структуру хранения данных и реализовать тестовое наполнение базы данных,
* Реализовать систему:
* Произвести тестирование и отладку;
* Оформить конструкторскую документацию;

Перечень реализуемых функций:

* Реализация поиска мероприятий по категориям и метаположению.
* Реализация поиска площадок по категориям, вместимости и местоположению.
* Реализация ведения статей бюджета.
* Реализация добавления комментариев к мероприятиям и площадкам.
* Предоставлять пользователю информацию о мероприятиях и сдаваемых в аренду площадках.
* Функция голосования за передоложенную пользователем идею среди остальных участников.

# 2.4. Обоснование выбора инструментов и платформы для разработки

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.

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.

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.

Для тестирования запросов к API используется Postman — приложение, которое упрощает разработку, ускорение и улучшение разработки API.

**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 (рисунок 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.

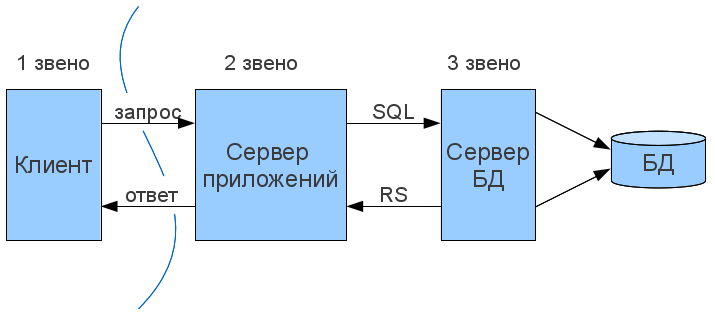


Рисунок 5 – Компоненты системы

Последовательность разработки системы:

1) Проектирование базы данных.

2) Создание логики серверной части.

3) Создание интерфейса.

4) Заполнение базы данных.

5) Тестирование.

**3.2 Логическая схема данных**

Модель «сущность-связь» представляет собой графическую нотацию, которая основывается на блоках и соединительных линиях. С их помощью можно описать объекты модели и отношения между ними. ER имеет несколько основных понятий, таких как сущности, атрибуты и связи. На рисунке 6 представлена логическая схема базы данных разрабатываемой системы.

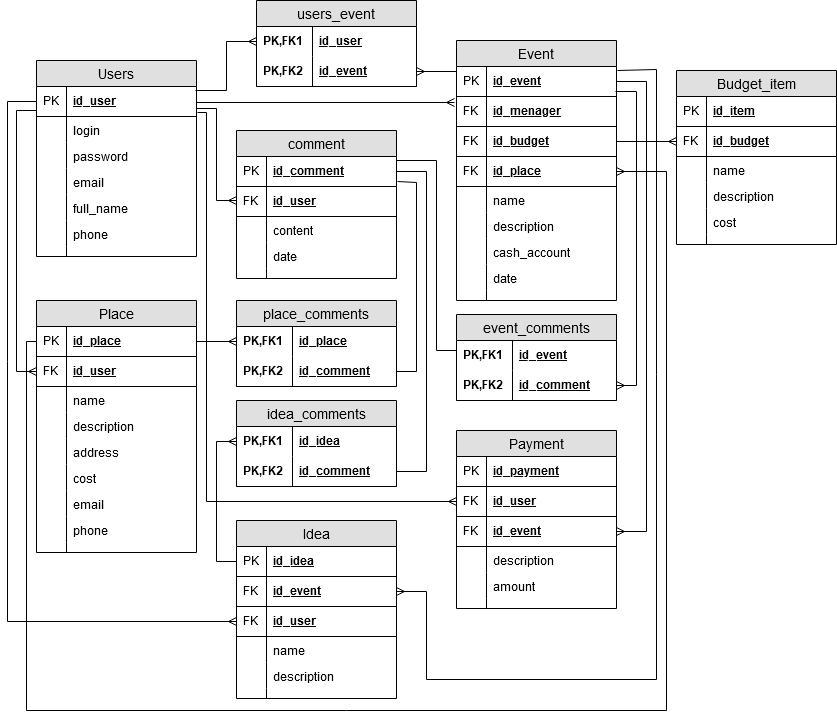


Рисунок 6 – Логическая схема базы данных

В результате анализа предметной области на данном этапе разработки системы были выделены следующие сущности:

Сущностью для хранения данных о участниках, организаторах и арендодателях является Пользователь. Он содержит в себе такие атрибуты как фамилия, имя и отчество, контактный номер и денежный счет. Будучи участником, пользователь может принять участие в мероприятии, оставить комментарий к мероприятиям, идеям, площадкам. Также совершить взнос для участия в мероприятии и создать идею. Поля ФИО, e-mail, контактный телефон необходимо будет заполнить для возможности стать организатором или арендодателем. Как организатор может создать мероприятие, а как арендодатель, может составить информации о сдаваемой в аренду площадке.

Следующей сущностью разрабатываемой системы является Мероприятие. Содержит в себе атрибуты названия, описания, даты проведения и суммы текущих взносов сформированную на основе статей бюджета. Может иметь много участников, только одного организатора и одну площадку. Так как участники принимают участие во многих мероприятиях, а мероприятие содержит много участников, созданная промежуточная таблица пользователи-события.

Сущность статья бюджета имеет такие атрибуты как название, описание и стоимость, которую может изменить организатор при получении новой информации. Одна статься бюджета может принадлежать только одному мероприятию.

Сущностью для хранения информации о сдаваемой в аренду площадке является Место, имеет атрибуты: название, описание, адрес, стоимость и контактные данные. Может иметь только одного владельца, а также предоставляться для многих мероприятий.

Для хранения идей пользователей используется сущность Идея, которая имеет атрибуты: название, описание и стоимость. После окончательного принятия идеи организатором информация о ней переходит в сущность Статья бюджета. Идея может иметь отношение только к одному мероприятию и быть инициализирована только одним пользователем.

Сущность Комментарии хранит в себе информацию о комментариях пользователей, такую как содержание и дату комментария. Комментарии могут быть оставлены к событиям, идеям и площадкам. Для связи с вышеперечисленными сущностями используются промежуточные таблицы. Комментарий может относиться только к одной из сущностей.

Для хранения информации о взносах пользователей используется сущность Платеж с атрибутами: описание, количество. Платеж могут совершать все пользователи и только на счет одного мероприятия.

**3.3 Разработка API веб-приложения**

Поскольку веб-приложение включает в себя множество сценариев использования и предполагает работу с базами данных, в качестве его основы выбран архитектурный паттерн Model-View-Controller (MVC), что позволило ускорить разработку путем повторного использования кода и упростит дальнейшую поддержку.

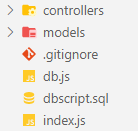


Рисунок 7 – структура веб-сервера.

На листинге 1 изображена инициализация подключения к базе данных в настройках сервера Node.js, посредством модуля pg.js.

Листинг 1 – подключение базы данных и проверка посредством запроса текущего времени.

const { Pool} = require('pg');

const connectionString = 'postgresql://postgres:12344321@localhost:5432/evmake\_test'

exports.pool = new Pool({

connectionString : connectionString,

});

sqlStatus = pool.query('SELECT NOW()', (err, res) => {

if(err){

console.log(err);

}

else{

console.log(res.rows[0]);

}

});

Контроллеры в паттерне MVC позволяют связать представления и модели и выполняют некоторую логику по обработке запроса.

Листинг 2 – обработка маршрутов /event при помощи контроллера EventsController в файле “index.js” .

app.get("/event/:id", EventsContoller.findById);

app.get("/events", EventsContoller.all);

app.get("/event/:id/users", UsersController.allForEvent);

app.post('/event', EventsContoller.create);

app.put('/event/:id', EventsContoller.update);

app.delete('/event/:id', EventsContoller.delete);

Листинг 3 – пример реализации отельного контроллера EventsController.

exports.findById = function (req, res) {

    Events.findById(req.params.id, function (err, result){

        if(err) {

            console.log(err);

            return res.sendStatus(500);

        }

        res.send(result.rows);

    })

}

Модель Events запрашивает информации о мероприятии из базы данных для последующей ее передачи контроллеру.

Листинг 4 – запрос информации о конкретном мероприятии.

exports.findById = function (id, cb) {

    pool.query(`select \* from events where id\_event = ${id};`,

function(err, result){

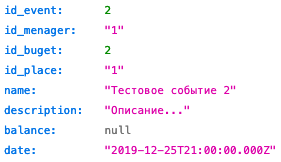
        cb(err, result);

    });

}

Приложение Node.js запускает сервер и слушает соединения на порте 4000. С помощью методов веб-фреймворка express осуществляются маршрутизация, которая определяет как приложение отвечает на клиентский запрос к конкретному адресу (конечной точке), которым является URI (или путь), и определенному методу запроса HTTP (GET, POST и т.д.). На листинге 2-5 показаны функции обработки запросов GET и POST, которые позволяют читать и записывать информацию о мероприятии из базы данных.

Листинг 5 – результат выполнения запроса /event/2 в формате JSON.



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

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

При проектировании системы на платформе Node.js был использован свободный фреймворк для веб-приложений на языке Express. С его 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.

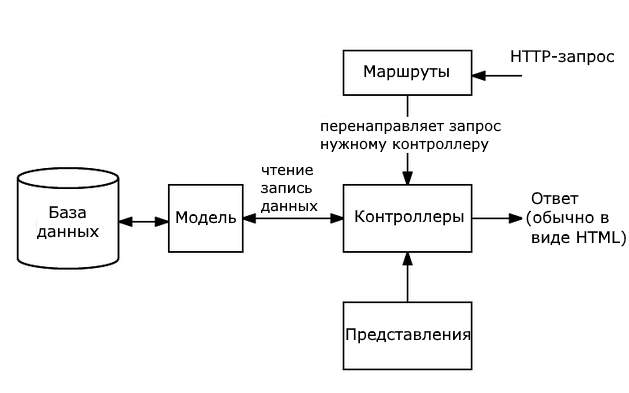


Рисунок 7 – Паттерн Model-View-Controller для веб-приложений Node.js

**Выводы**

В ходе выполнения данной курсовой работы была проанализированная выбранная предметная область, выделены основные сущности системы, разработана концептуальная схема данных, выбран инструментарий для реализации функционала веб-приложения, а также выведены UML-диаграммы сценариев.

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 JavaScritp и Node.js.

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

**Список источников**

1. Каскиаро М., Маммино Л. Шаблоны проектирования Node.JS / М. Каскиаро, Л. Маммино - Шаблоны проектирования Node.js/ пер. с анг. А.Н. Киселева. – Москва: ДМК Пресс, 2017.– 396с.: ил. URL://biblioclub.ru/index.php?page=book&id=141626
2. Пюривал С. Основы разработки веб-приложений. / С. Пюривал Основы разработки веб-приложений. — СПб.: Питер, 2015. — 272 с.: ил. — (Серия «Бестселлеры O’Reilly») – Режим доступа: http://biblioclub.ru/index.php?page=book&id=96968, по подписке
3. Холмс С. Стек MEAN. Mongo, Express, Angular, Node. — СПб.: Питер, 2017. — 496 с.: ил. — (Серия «Библиотека программиста»).ISBN 978-5-496-02459-4
4. Янг А., Мек Б., Кантелон М. Node.js в действии. / А. Янг, Б. Мек, М. Кантелон - Node.js в действии. 2-е изд. — СПб.: Питер, 2018. — 432 с.: ил. — (Серия «Для профессионалов»).ISBN 978-5-496-03212-4
5. Документация модуля node-postgres [Электронный ресурс] / node-postgres.com – Электрон. текстовые дан. – Режим доступа: https://node-postgres.com/features/, свободный.
6. Документация платформы Node.js [Электронный ресурс] / nodeguide.ru – Электрон. текстовые дан. – Режим доступа: https://nodeguide.ru/doc/, свободный.
7. Документация фреймворка Express.js [Электронный ресурс] / expressjs.com – Электрон. текстовые дан. – Режим доступа: https://expressjs.com/ru/starter/installing.html, свободный.
8. Документация языка Html [Электронный ресурс] / w3.org – Электрон. текстовые дан. – Режим доступа: https://www.w3.org/TR/html52/, свободный.
9. Официальная документация приложения Postman [Электронный ресурс] / postman.ru – Электрон. текстовые дан. – Режим доступа: https://postman.ru.softonic.com/#tab-review, свободный.
10. Справочник по html, css, веб-дизайн [Электронный ресурс] / Htmlbook.ru– Электрон. текстовые дан. – Режим доступа: http://htmlbook.ru/html, свободный.