**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.

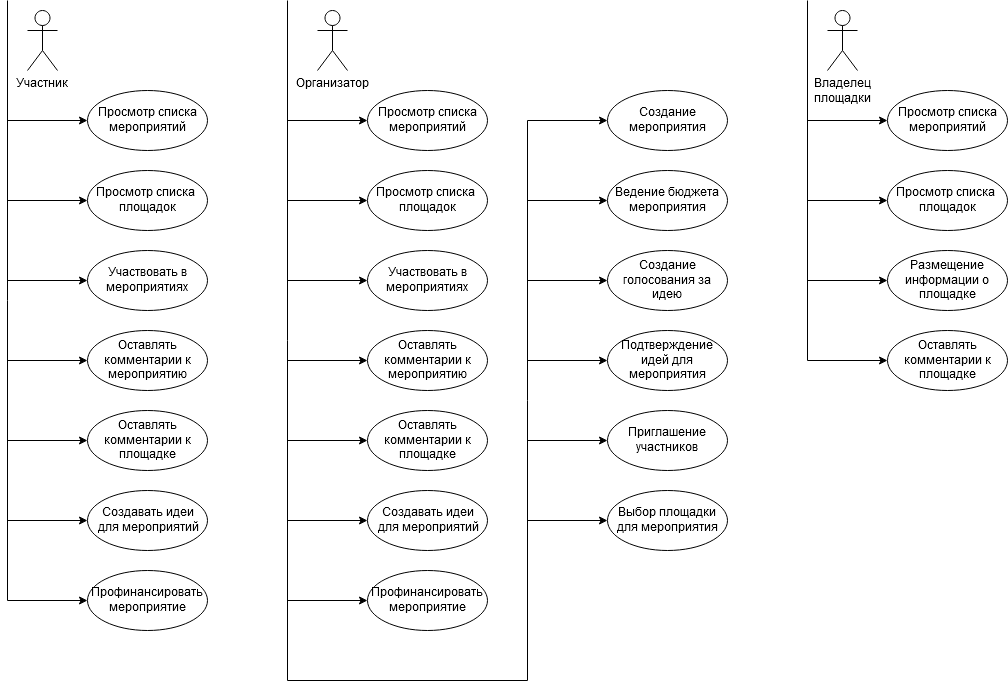


Рисунок 2 – UML диаграмма пользовательских сценариев для покупателя.

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.

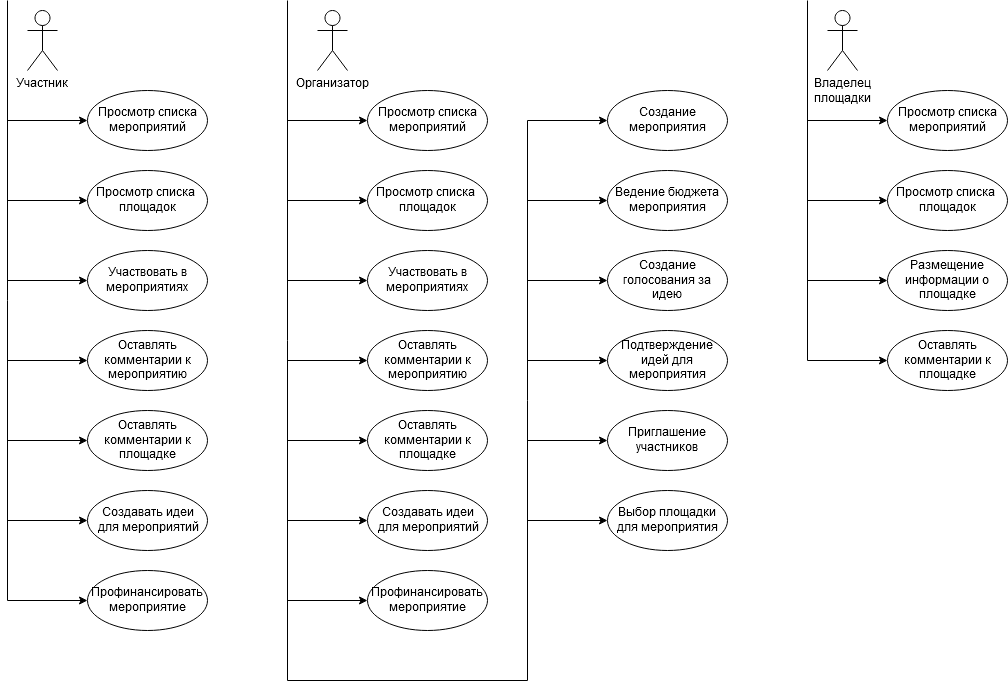


Рисунок 3 – UML диаграмма пользовательских сценариев для организатора.

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.

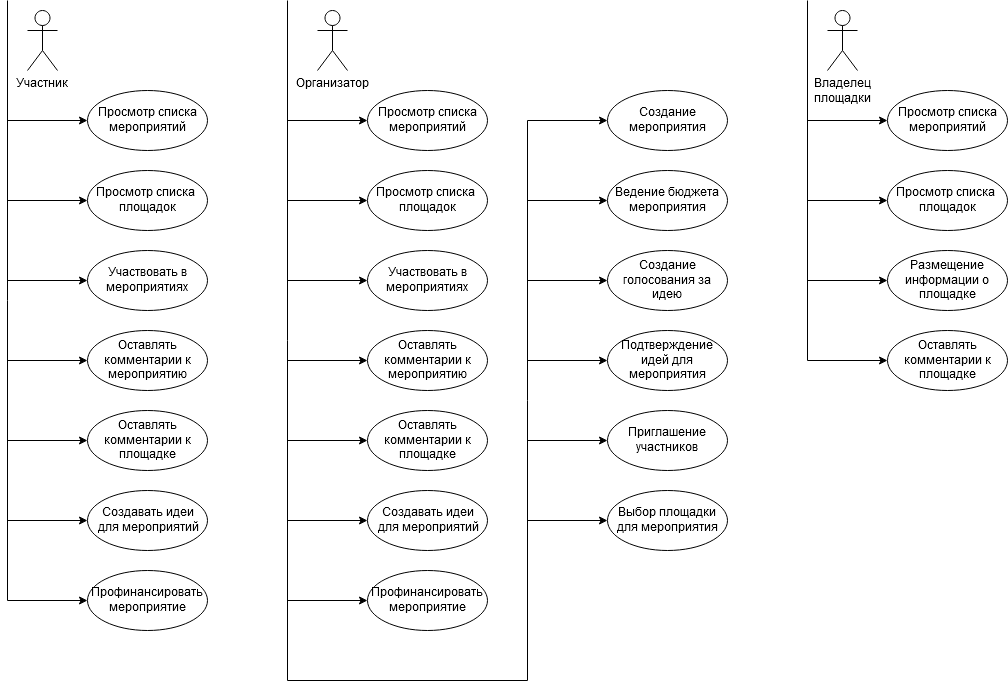


Рисунок 3 – UML диаграмма пользовательских сценариев для владельца площадок.

# 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.