МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Сервис поиска спортивной команды

Курсовой проект

09.03.02 Информационные системы и технологии

Программная инженерия в информационных системах

Допущен к защите

Зав. Кафедрой _____*С.Д. Махортов, к.ф.- м.н., доцент* _._.__.20__

Обучающийся _____*А.Г. Волков 3 курс, д/о*

Обучающийся _____*С.А. Маликов 3 курс, д/о*

Обучающийся _____*И.И. Антоненко 3 курс, д/о*

Руководитель _____*В.С. Тарасов, преподаватель*

Воронеж 2020

Содержание

Введение

В современном мире информационные технологии участвуют практически во всех сферах человеческой деятельности. Но по мере их популяризации общество столкнулось с проблемами нового характера. Это снижение физической активности людей и уменьшение роли общения в реальной жизни.

Учитывая темп и образ жизни современного человека, все более явной становится и проблема поиска людей с общими интересами. Но тут на помощь пришли различные сервисы онлайн-поиска. Они позволяют сделать процесс получения информации о многих вещах быстрее и проще.

Данный курсовой проект посвящен разработке приложения, которое позволит упростить процесс поиска людей для совместных занятий спортом.

1 Постановка задачи

Целью курсового проекта является создание web - приложения, с помощью которого пользователи смогут найти себе сокомандников для игр разных видов спорта, откликаясь на объявления о наборе, создаваемых другими пользователями.

Основную функциональность разрабатываемого приложения отражает диаграмма прецедентов, изображенная на рисунке 1.

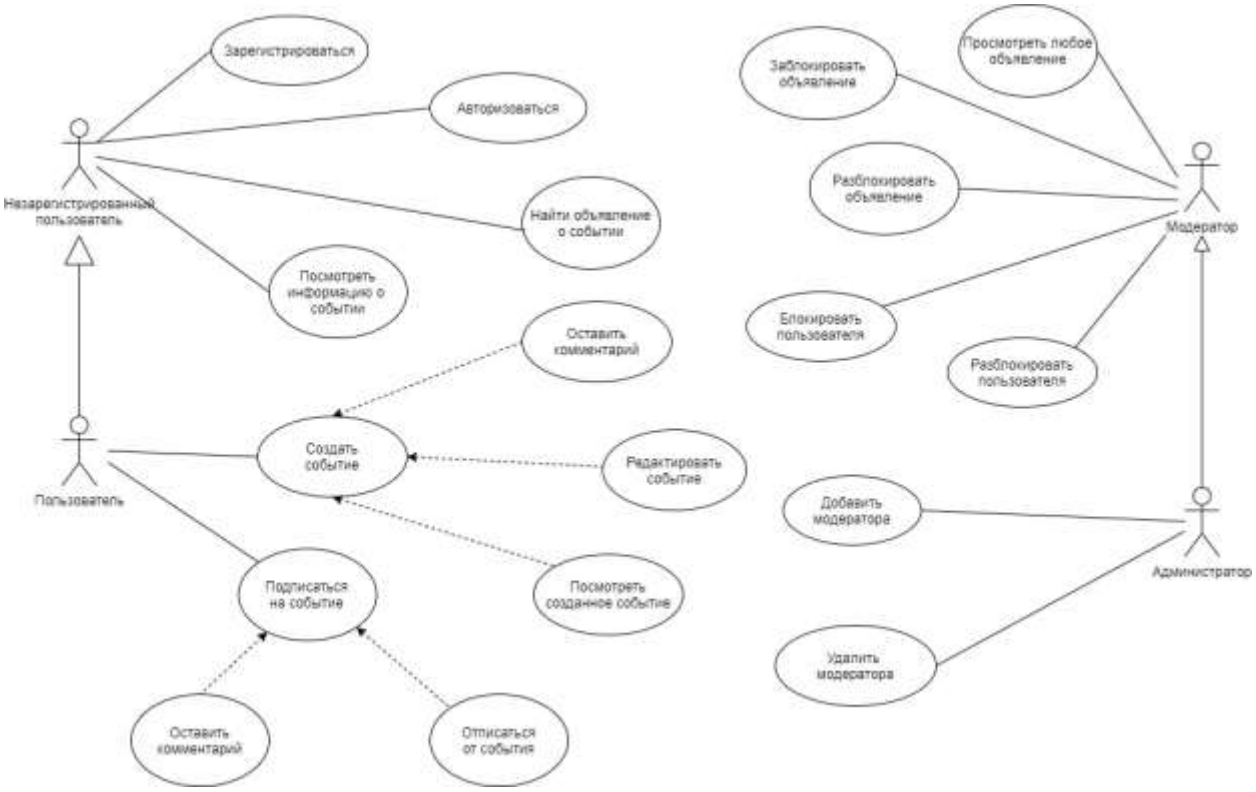

Рисунок 1 – Диаграмма прецедентов

Неавторизованный пользователь обладает следующими возможностями:

— авторизация;
— регистрация
— найти объявление о событии;
— посмотреть объявление о событии;

Авторизованный пользователь обладает следующими возможностями:

— подписаться на событие;
— найти объявление о событии;
— посмотреть объявление о событии;
— прокомментировать событие;
— создать новое событие;
— редактировать созданное событие;

Для мониторинга и контроля системы предусмотрена роль модераторов со следующими возможностями:

— посмотреть любое объявление
— заблокировать объявление
— разблокировать объявление

Модераторов может добавлять или удалять администратор.

В системе должны быть реализованы все вышеописанные возможности пользователя.

Завершенный проект представляет собой полностью функционирующее web-приложение.

2 Анализ предметной области

2.1 Актуальность

С проблемой поиска команды для игры сталкивался любой человек, занимающийся командными видами спорта как профессионально, так и любительски. Обычно люди организовывают постоянный коллектив, которым они собираются и играют. Но что делать тем, кто не имеет такого коллектива или кто-то из команды не может прийти на очередную игру?

Чтобы узнать интерес пользователей к потенциальному приложению был проведен опрос. В нем поучаствовало 93 человека разных возрастов. Исходя из данных рисунка 2 видно, что проблема, обозначенная в предыдущем абзаце довольно актуальна, и большинство людей сталкивались с проблемой нехватки игроков для спортивных игр.

Рисунок 2 – Результаты опроса

Давно известно, что человек – существо биосоциальное, поэтому занимаясь спортом с другими людьми, он еще и удовлетворяет потребность в общении. И отсутствие компании может его сдерживать от занятия спортом. Это подтверждают данные рисунка 3.

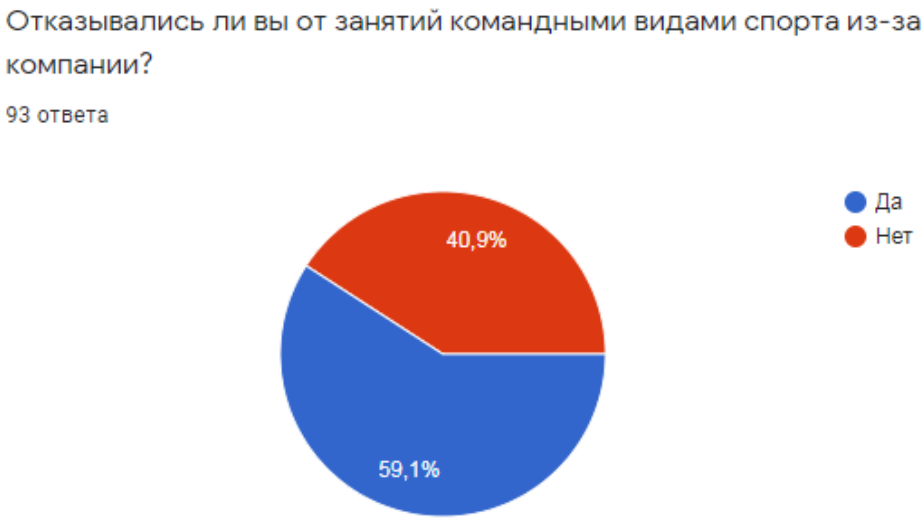

Рисунок 3 – Результаты опроса

Мы решили узнать, как именно люди собираются для игр. И согласно данным рисунка 4 можно сделать вывод, что в основном они играют вместе со своими друзьями. Но больше всего удивляет, что около 34% респондентов в принципе не занимаются командными видами спорта, потому что им не с кем.

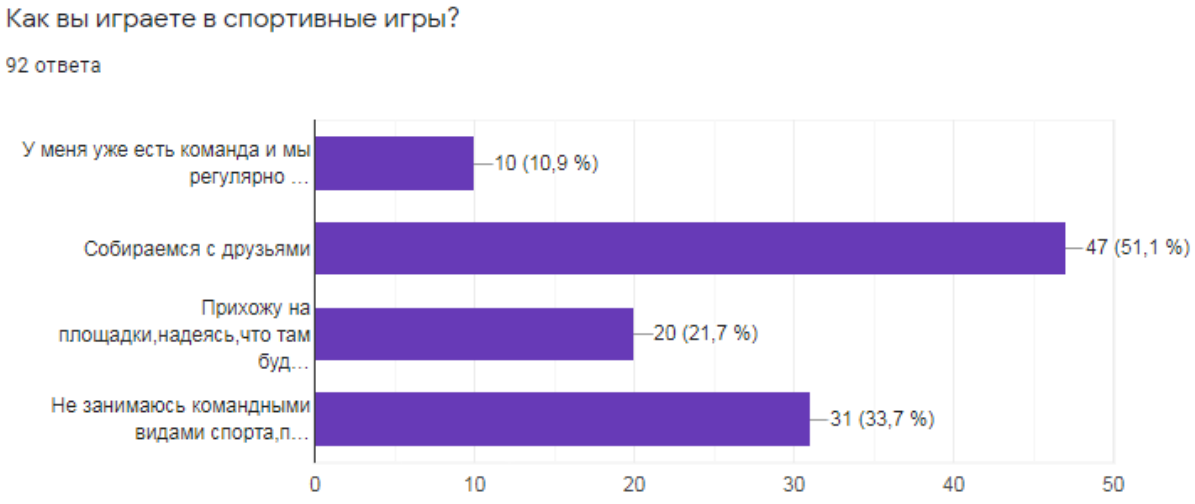

Рисунок 4 – Результаты опроса

Так как сейчас информационные технологии являются неотъемлемой частью жизнью людей, возникает вопрос: пробовали ли наши респонденты

применять их для поиска сокомандников? И глядя на рисунок 5, очевидно, что нет.

Рисунок 5 – Результаты опроса

Поэтому мы решили спросить у наших потенциальных пользователей, хотели бы они воспользоваться приложением, которое позволить упростить процесс поиска людей для совместных занятий спортом. И абсолютное большинство выбрало ответ «ДА».

Рисунок 6 – Результаты опроса

Таким образом, очевидна потребность в сервисе, который бы позволял пользователям создавать объявления о наборе игроков, на которые будут подписываться другие пользователи и собираться для игр. Для успешного функционирования необходимо автоматизировать процесс поиска команды, что и позволит реализация этого курсового проекта.

2.2 Сравнение с аналогами

На данный момент существует не так много сервисов для поиска команд, что отражает актуальность нашего проекта. Главным потенциальным конкурентом является сайт findysport.com - сервис поиска партнера или компании для совместных занятий спортом.

Достоинства:

- Удобный интерфейс
- Возможность поиска события при помощи карты
- Авторизация через facebook
- Возможность поиска по ключевым словам
- Возможность поиска в территориальном радиусе

Недостатки:

- Сервис ориентирован на пользователей, профессионально занимающихся спортом
- Нельзя находить события по дате игры
- Нестабильность работы русского интерфейса
- Нельзя обмениваться информацией при просмотре объявления

2.3 Анализ задач

Для функционирования системы необходимо реализовать 3 основных задачи:

- Создание события (объявления об игре);
- Поиск события и подписка на него;

— Модерация системы

2.3.1 Задача создания события

При создании события должна указываться следующая информация:

- Название события
- Описание
- Дата проведения
- Необходимое количество участников
- Категория
- Место проведения
- Время проведения

Таким образом, задача создания события включает в себя следующие этапы:

1.Введение пользователем необходимой информации в личном кабинете

2.Передача введенных данных на сервер

3.Запись полученной информации в базу данных

Таким образом, пользователи будут создавать новые события, которые будут записываться в базу данных в ожидании подписки на них других пользователей.

2.3.2 Задача поиска события и подписки на него

Для работы приложения необходимо реализовать поиск объявлений об играх с возможностью подписки (подтверждения участия в них). Для этого надо реализовать работу следующих этапов:

1. Поиск пользователем игр по фильтрам;

2. Отправка данных с формы на сервер;

3. Формирование запроса к базе данных;

4. Обработка сервером ответа со списком открытых для подписки событий;

5. Отображение результатов поиска пользователю;

6. Если пользователь хочет подписаться на найденное событие (при условии авторизации), обработка подписки пользователя: передача информации о подписке на сервер

2.3.3 Задача модерации системы

Для контроля действий пользователей необходимо реализовать работу модерации системы. Для этого предусмотрено 2 роли: модератор и администратор. Модераторы осуществляют непосредственный контроль работы системы и наделены полномочиями блокировать (и разблокировать) пользователей и объявления. Модераторы назначаются администраторами. Список администраторов же вручную корректируется через базу данных пользователей.

2.4 Анализ продуктовых воронок

При проектировании системы важно не упускать из внимания психологию пользователя. Можно иметь хорошее приложение, сделать приятный дизайн и развернуть широкую рекламную компанию, но все равно иметь недостаточную аудиторию пользователей. Поэтому вовлечение пользователя было построено следующим образом.

Изначально мы даем пользователю попробовать воспользоваться продуктом, осуществив поиск событий без авторизации. Это намного эффективнее, чем изначально принудительно заставлять регистрироваться. Если пользователь поймет, что приложение ему подходит, он все равно пройдет регистрацию, чтобы подписываться на события и создавать свои.

Построение диалога с пользователем таким образом ,на наш взгляд, принесет максимальную пользу продукту.

2.5 Анализ архитектуры

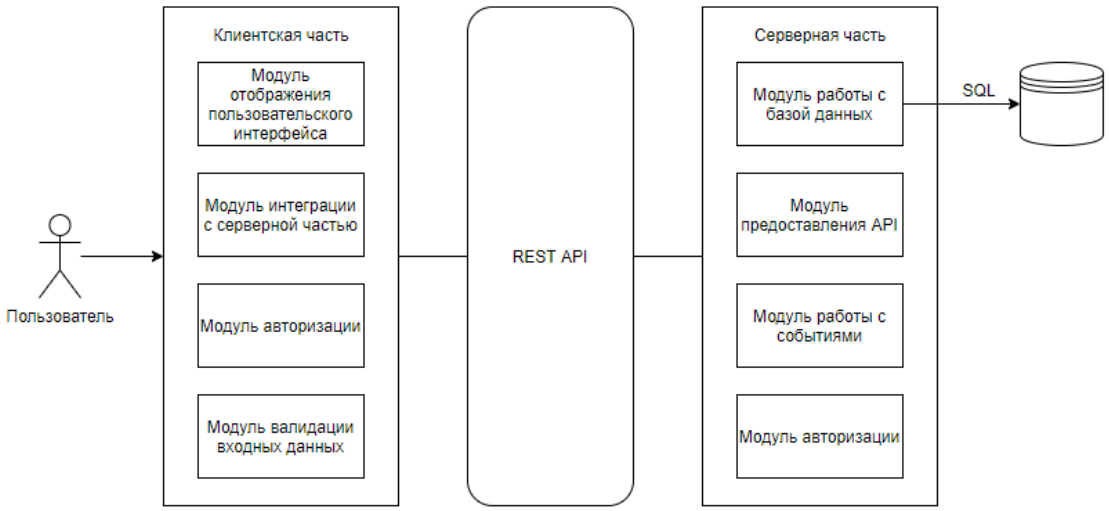

Рисунок 7 - Архитектура приложения

Приложение должно иметь архитектуру, основанную на шаблоне проектирования Model-View-Controller, а также иметь разделение на front-end и back-end.

Базы данных (Model) представляет собой фундаментальные данные, необходимые для работы приложения.

Клиентская часть (View) является подписчиком на событие изменения значений свойств, запросов или команд, предоставляемых серверной частью. Если пользователь воздействует на какой-либо элемент интерфейса, вызывается соответствующая команда, предоставляемая серверной частью.

Серверная часть (Controller) предоставляет обёртку данных из базы данных, которые подлежат связыванию. Таким образом, она является посредником между клиентской частью и базой данных, предоставляя пользователю возможность для обработки вводимой им информации, извлекая данные из базы данных и форматируя их для отображения в графическом интерфейсе.

Модуль отображения пользовательского интерфейса – генерирует HTML страницу, которую видит конечный пользователь.

Модуль интеграции с серверной частью – отвечает за отправку запросов на Back-end сервер и обработку ответов.

Модуль авторизации клиентской части – отвечает за получение и хранение токена при авторизации и регистрации пользователя. А также за обновление токена.

Модуль валидации входных данных – отвечает за проверку на корректность вводимых пользователем данных на формах сайта.

Модуль работы с базой данных – отвечает за установление соединения с базой данных, маппинг таблиц из базы данных в сущности Java классов, CRUD операции.

Модуль предоставления API – набор эндпоинтов, через которые сторонние приложения обращаются к этому приложению.

Модуль работы с событиями – отвечает за выполнение операций с событиями, таких как сортировка, изменение, удаление, оформление подписок и отписок пользователями, работы с комментариями, содержащимися в событиях.

Модуль авторизации в серверной части – отвечает за выдачу токенов при авторизации пользователя, регистрацию новых пользователей.

2.6 Графическое описание работы системы

Для описания работы системы был использован язык графического описания системы UML. В данном разделе представлены все необходимые диаграммы.

2.6.1 IDEF0 диаграмма

Для лучшего понимания работы системы и выделения ключевых сценариев составлена IDEF0 диаграмма. На рисунках 8 и 9 приводятся контекстная диаграмма и диаграмма работы системы соответственно.

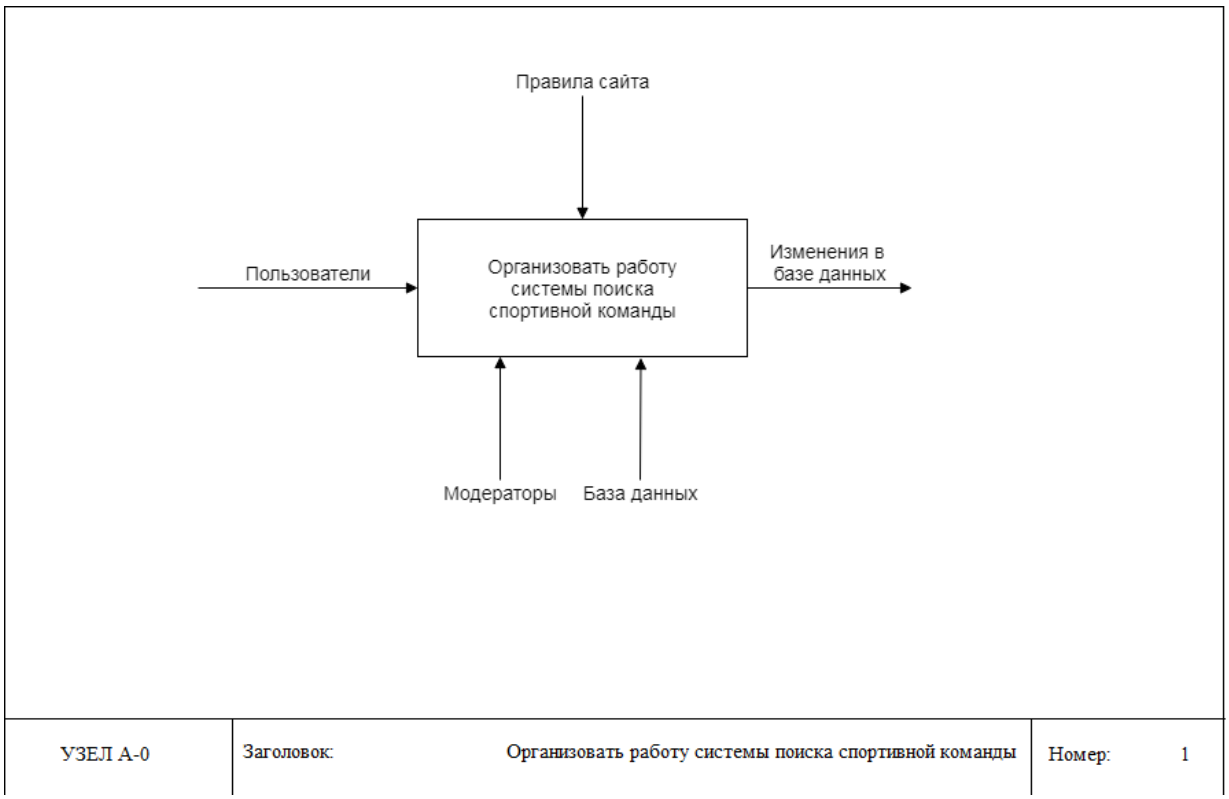

Рисунок 8. Контекстная IDEF0 диаграмма.

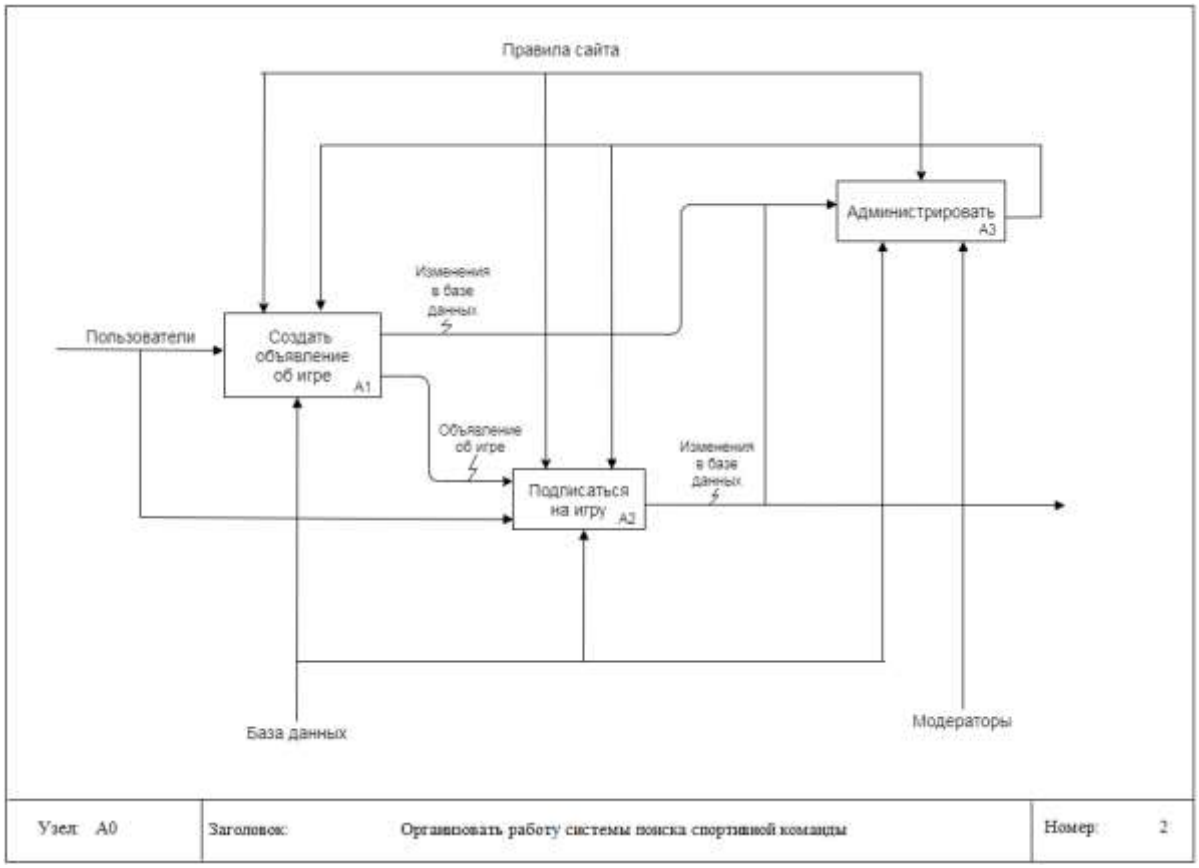

Рисунок 9. IDEF0 диаграмма работы системы.

2.6.2 Диаграмма классов

В качестве паттерна проектирования была выбран шаблонный метод – поведенческий паттерн, определяющий общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов. В нашем проекте таким суперклассом является «Базовая сущность». Диаграмма классов представлена на рисунке 10.

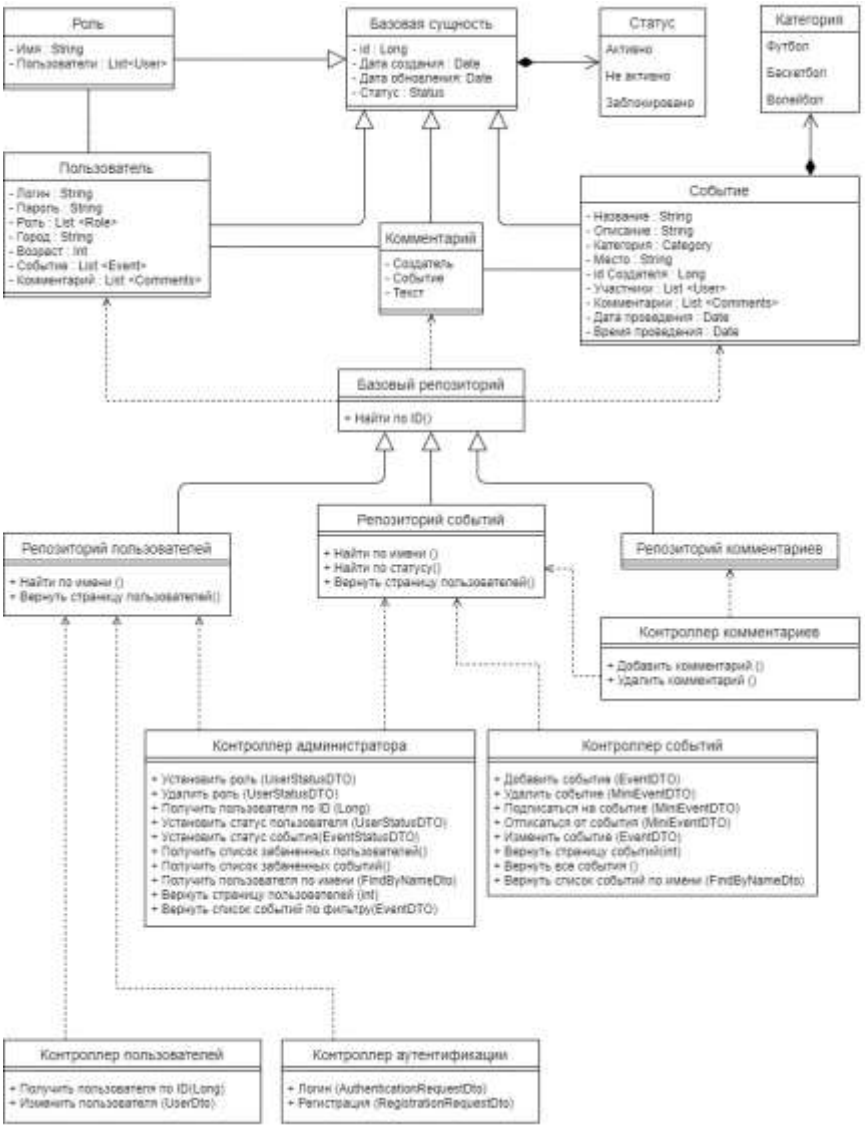

Рисунок 10 – Диаграмма классов

Классы Модели. Используются для создания таблиц в базе данных.

Перечисление Статус. Содержит в себе список состояний класса (Активно, Не активно, Заблокировано).

Класс Базовая сущность. Является классом – родителем для всех классов – моделей базы данных. Содержит в себе данные о идентификаторе экземпляра класса (id), дате создания, дате обновления и статусе экземпляра класса(Status). Не используется для создания таблицы в базе данных.

Класс Пользователь. Является классом – наследником класса Базовая сущность. Содержит в себе данные о уникальном имени пользователя в системе(Логин), пароле пользователя, списке ролей, которые определяют доступный функционал сайта для конкретного пользователя, городе, возрасте, списке событий, на которые подписан пользователь и комментарии, которые когда-либо были написаны им. Также наследует все поля класса Базовая сущность.

Класс Роль. Является классом – наследником класса Базовая сущность. Содержит в себе данные о имени роли и списке пользователей, у которых есть данная роль. Также наследует все поля класса Базовая сущность.

Перечисление Категория. Содержит в себе список видов спорта (Футбол, Волейбол, Баскетбол).

Класс Событие. Является классом – наследником класса Базовая сущность. Содержит в себе данные о названии события, описании конкретного события, в котором записана дополнительная информация о событии, категории, которая является перечислением множества видов спорта (Футбол, Волейбол, Баскетбол), месте – в этом поле создатель события указывает место, в котором будет проведено событие, дате и времени встречи, идентификаторе создателя (id создателя) и комментариях, которые были добавлены данному событию. Также наследует все поля класса Базовая сущность.

Класс Комментарий. Является классом – наследником класса Базовая сущность. Содержит в себе данные о создателе комментария, о событии, в

котором комментарий был оставлен и тексте комментария. Также наследует все поля класса Базовая сущность.

Классы Хранилища. Используются для взаимодействия коде программы с таблицей в базе данных.

Класс Базовый репозиторий (org.springframework.data.jpa.repository). Является классом – родителем для всех классов – хранилищ базы данных. Содержит метод поиска по идентификатору, а также другие методы с которыми можно ознакомиться в документации.

Класс Репозиторий событий. Является классом – наследником класса Базовый репозиторий. Наследует все методы класса Базовый репозиторий.

Класс Репозиторий событий. Является классом – наследником класса Базовый репозиторий. Содержит в себе методы поиска по имени и статусу, метод возврата страницы пользователей. Также наследует все методы класса Базовый репозиторий.

Класс Репозиторий пользователей. Является классом – наследником класса Базовый репозиторий. Содержит в себе метод поиска по имени и метод возврата страницы пользователей. Также наследует все методы класса Базовый репозиторий.

Классы Контроллеры. Используются для вызова кода программы как из сторонних приложений, так и внутри самого приложения при помощи ссылок по протоколу http или https.

Класс Контроллер комментариев. Содержит в себе методы добавления комментария событию и удаления комментария из события.

Класс Контроллер событий. Содержит в себе методы добавления события, удаления события, подписки на событие, отписки от события, изменения события пользователем, который является его создателем,

возврата страницы, которая состоит из списка с указанным количеством событий, возврата списка всех событий и возврата списка событий по их названию из базы данных.

Класс Контроллер администратора. Содержит в себе методы добавления роли (установить роль), удаления роли, получения пользователя по идентификатору, задания статуса пользователю и событию, получения списка заблокированных пользователей и событий, получение пользователя по его уникальному имени, получения страницы в виде списка с заданным кол-вом пользователей, получения списка событий по фильтру.

Класс Контроллер аутентификации. Содержит в себе метод возврата токена, логина, id и списка ролей пользователя после аутентификации (метод – Логин), метод регистрации пользователя.

Класс Контроллер пользователей. Содержит в себе методы возврата пользователя по идентификатору и изменения данных пользователя.

Для лучшего понимания работы системы приводится диаграмма объектов на рисунке 11.

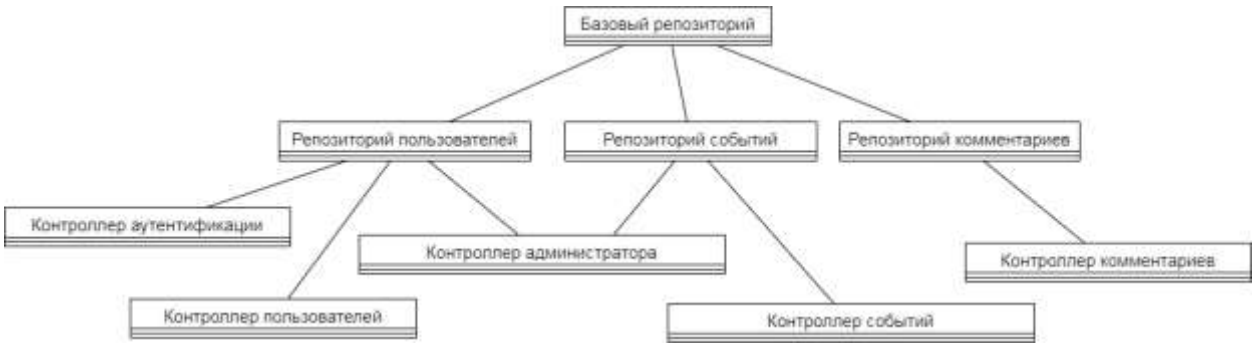

Рисунок 11 – Диаграмма объектов

2.6.3 Диаграммы состояния

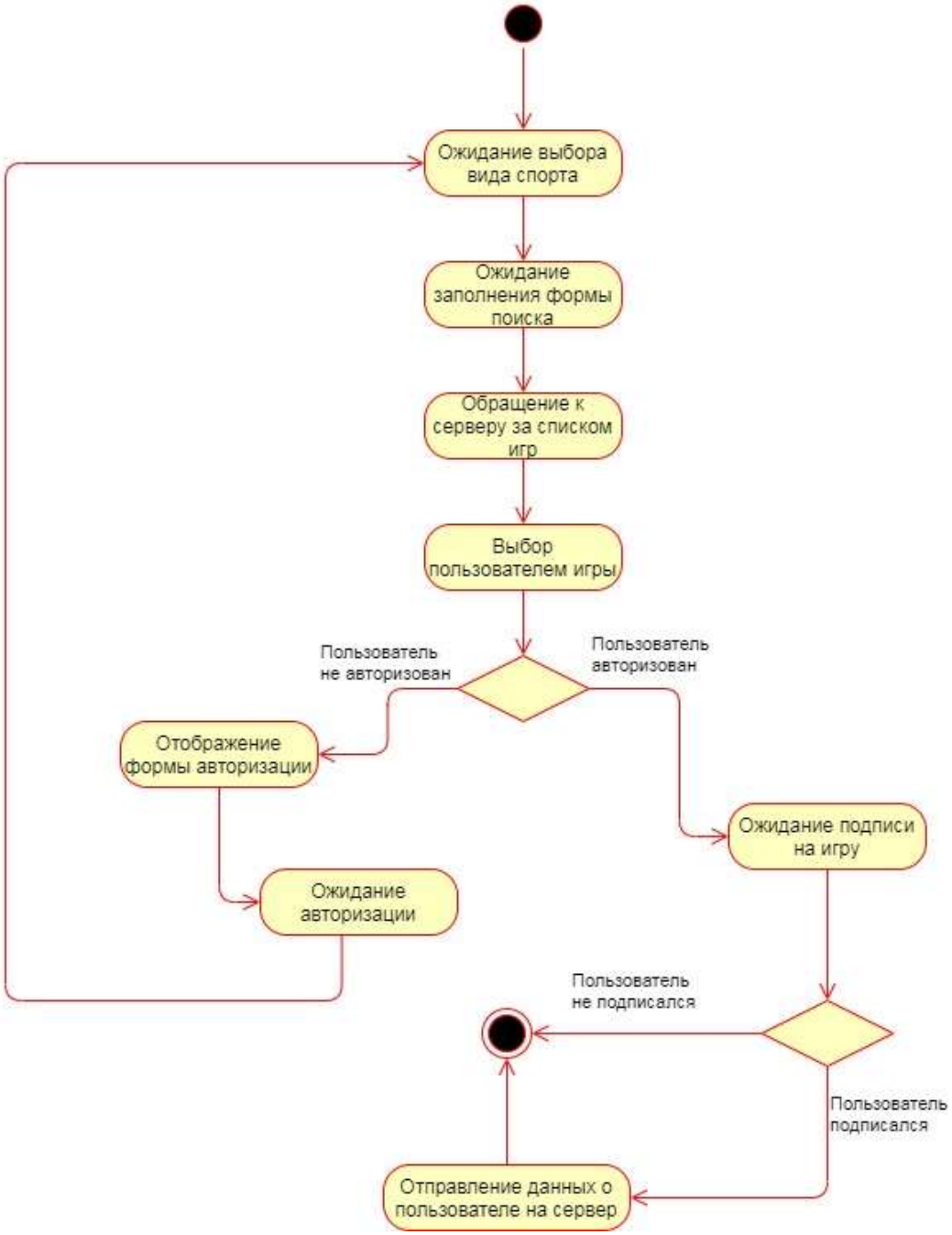

Рисунок 12. Диаграмма состояний поиска событий и подписки на них

Для описания состояний, в которых находится система при сценарии поиска событий и подписки на них, составлена диаграмма, изображенная на рисунке 12. Изначально на экране пользователь видит карточки видов спорта и система ожидает выбор интересующего вида спорта. Система отображает форму поиска событий по фильтрам и ожидает ее заполнения. После этого она обращается к серверу за списком подходящих игр. И после его

отображения ожидает выбора пользователя интересующего его игры. Подписка доступна только авторизованным пользователям, поэтому если пользователь желает подписаться на событие, система отображает на сервер данные о пользователе.

Для описания состояний, в которых находится система при сценарии создании нового события, составлена диаграмма, изображенная на рисунке 13.

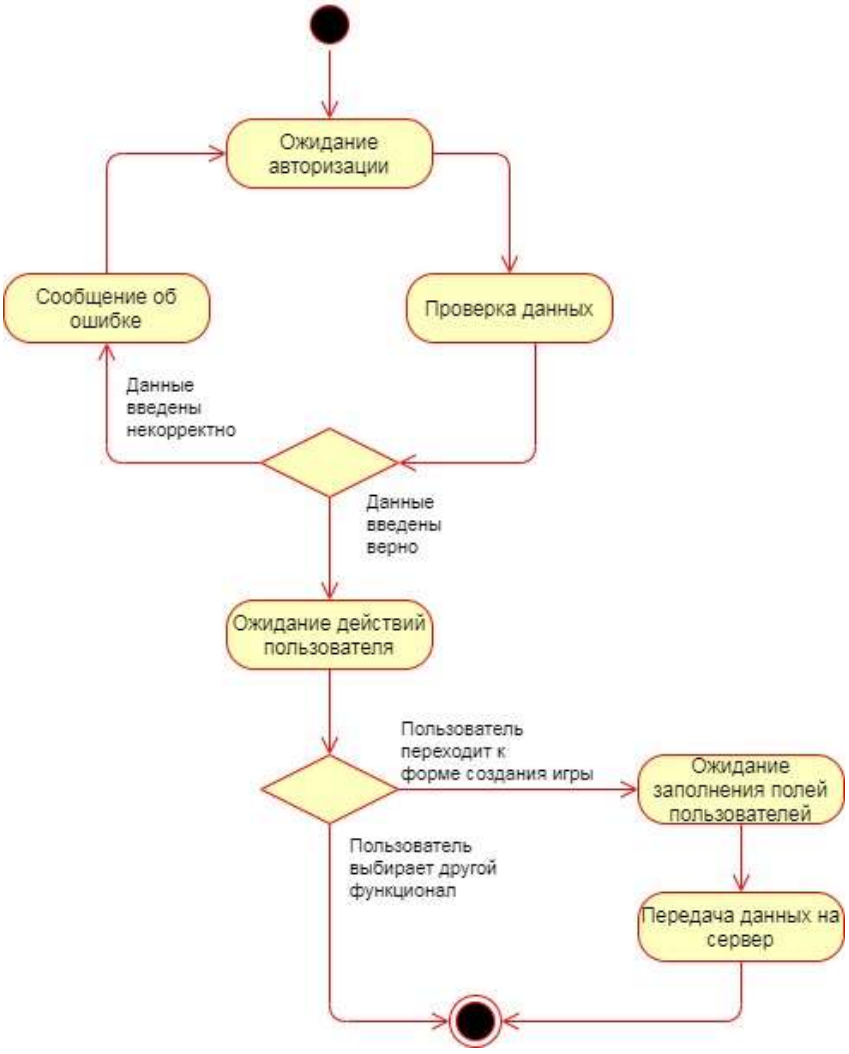

Рисунок 13. Диаграмма состояний создания нового события

Создание новых событий доступно только авторизованным пользователям, поэтому сценарий начинается с авторизации пользователя.

После этого пользователь заполняет необходимые поля при создании и система передает данные о созданном событии на сервер.

Для описания состояний, в которых находится система при сценарии модерации, составлена диаграмма, изображенная на рисунке 14.

Рисунок 14. Диаграмма состояний модерации системы.

Система ожидает действий модератора, после чего передает на сервер id забаненного элемента.

2.6.4 Диаграммы активностей

Диаграммы активности являются расширениями диаграмм состояний, находящихся в предыдущем разделе.

Диаграмма активности сценария поиска событий и подписки на них изображена на рисунке 15. На данной диаграмме присутствуют 4 дорожки: пользователь, приложение, сервер и база данных

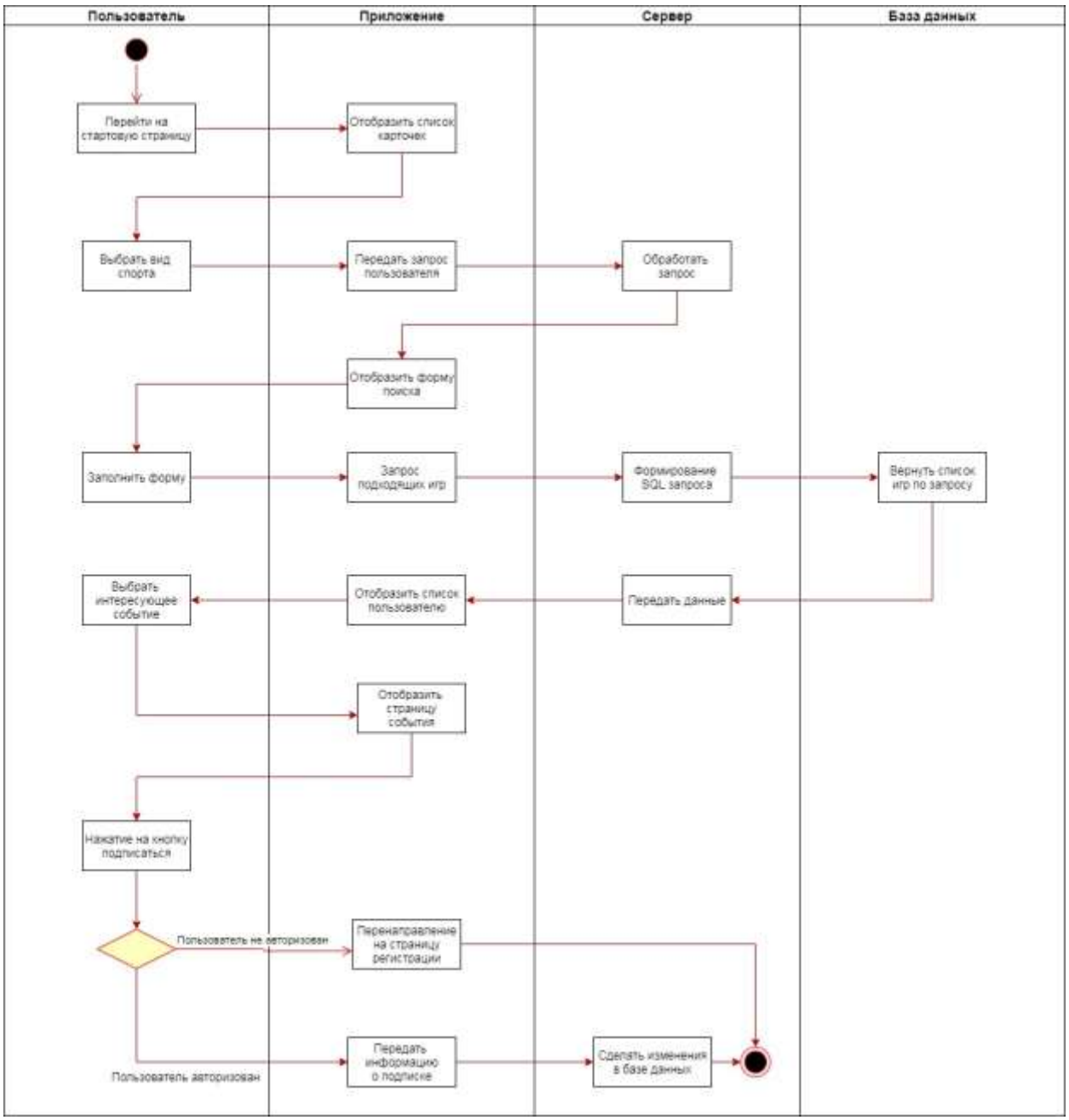

Рисунок 15 - Диаграмма активности сценария поиска событий и подписки на них

Диаграмма активности сценария создания нового события изображена на рисунке 16. На данной диаграмме присутствуют 3 дорожки: пользователь, приложение и сервер.

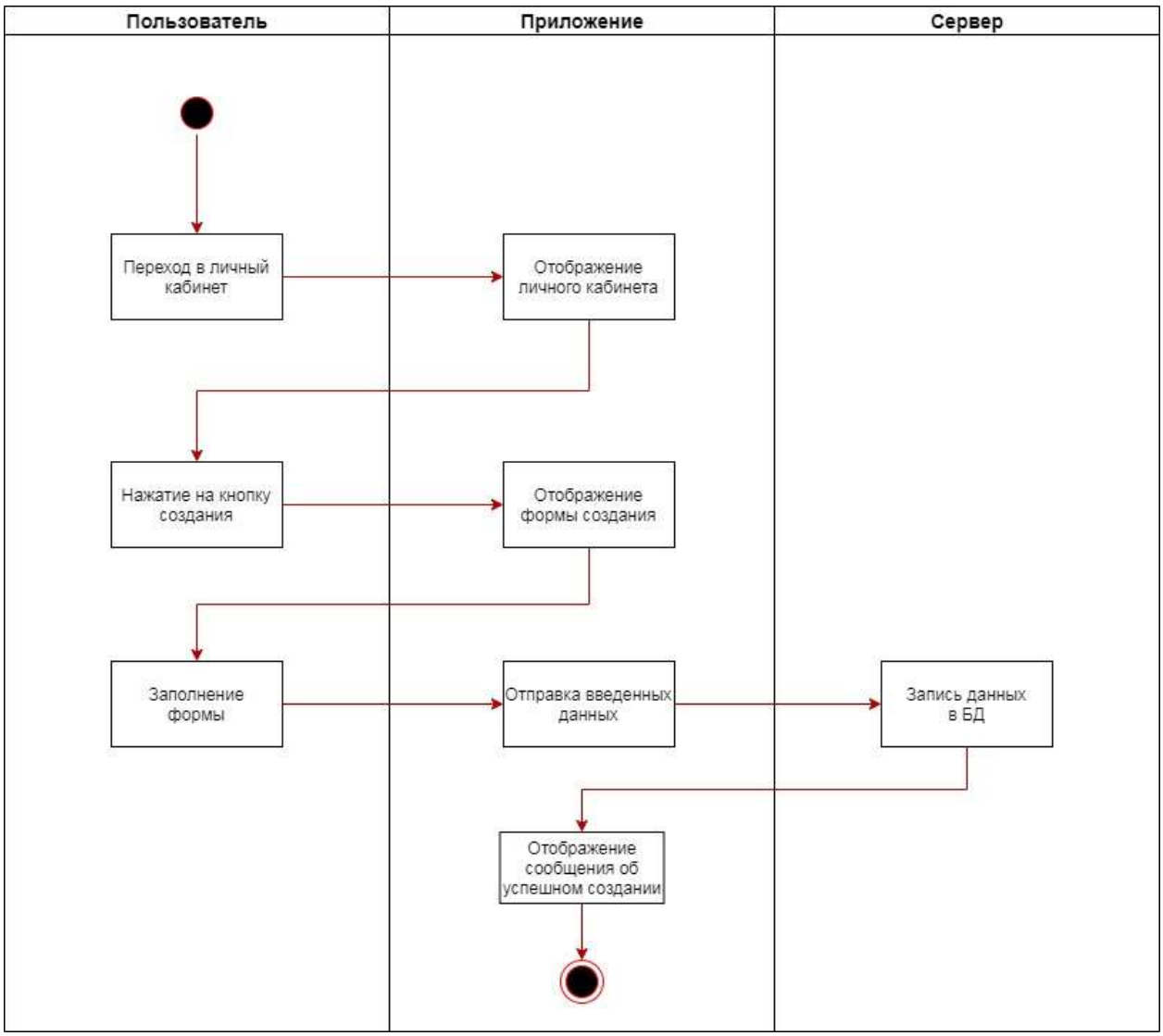

Рисунок 16 – Диаграмма активности сценария создания нового события

2.6.5 Взаимодействие компонентов системы

На рисунке 18 показана диаграмма взаимодействий, на которой указываются отношения между компонентами системы при поиске и подписки на событие, а на рисунке 17 представлена диаграмма последовательностей, на которой изображено упорядоченное во времени взаимодействие компонентов системы.

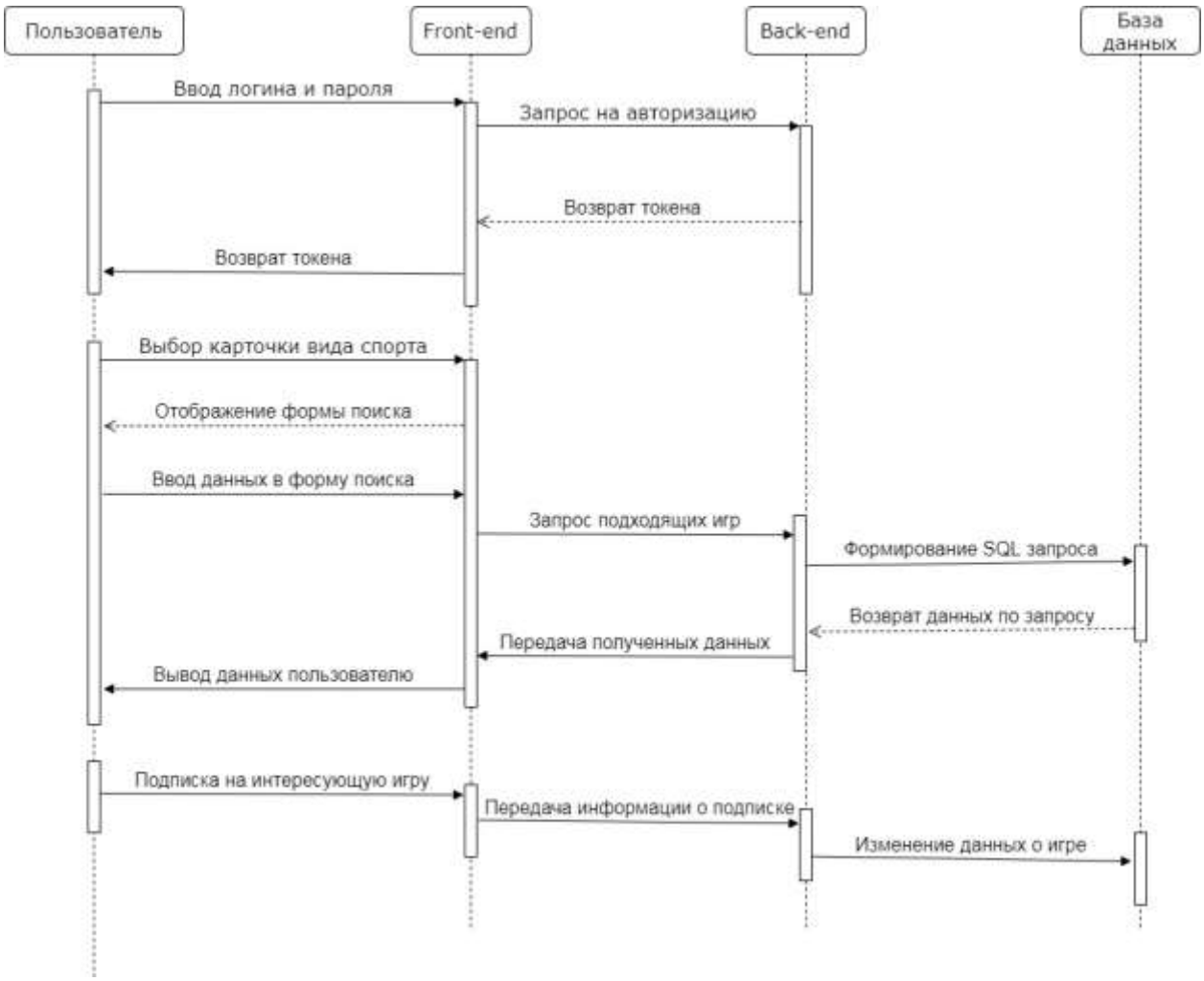

Рисунок 17 – Диаграмма последовательностей поиска и подписки на событие

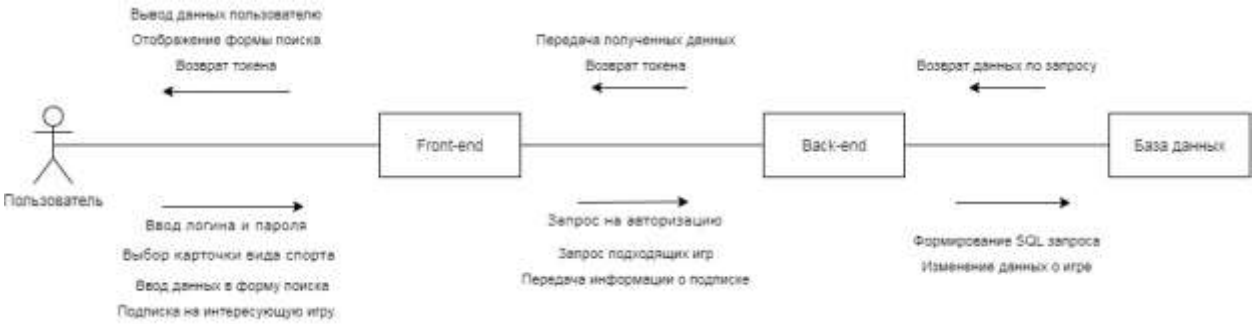

Рисунок 18 – Диаграмма взаимодействий поиска и подписки на событие

Пользователь вводит логин и пароль. Front-end отправляет на сервер запрос на авторизацию. Если пользователь успешно проходит авторизацию,

ему возвращается токен. Пользователь нажимает на карточку вида спорта для отображения формы поиска. Он вводит данные для поиска по фильтрам, по которым Front-end делает запрос подходящих игр на сервер. Тот ,в свою очередь, формирует SQL запрос к базе данных для возврата по нему данных. Далее данные передаются на Front-end и отображаются пользователю. Если пользователь хочет подписаться на интересующее событие, он нажимает на соответствующую кнопку. Front-end передает информацию о подписке и Back-end корректирует данные о событии в базе данных.

На рисунке 20 показана диаграмма взаимодействий, на которой указываются отношения между компонентами системы при создании нового события, а на рисунке 19 представлена диаграмма последовательностей, на которой изображено упорядоченное во времени взаимодействие компонентов системы.

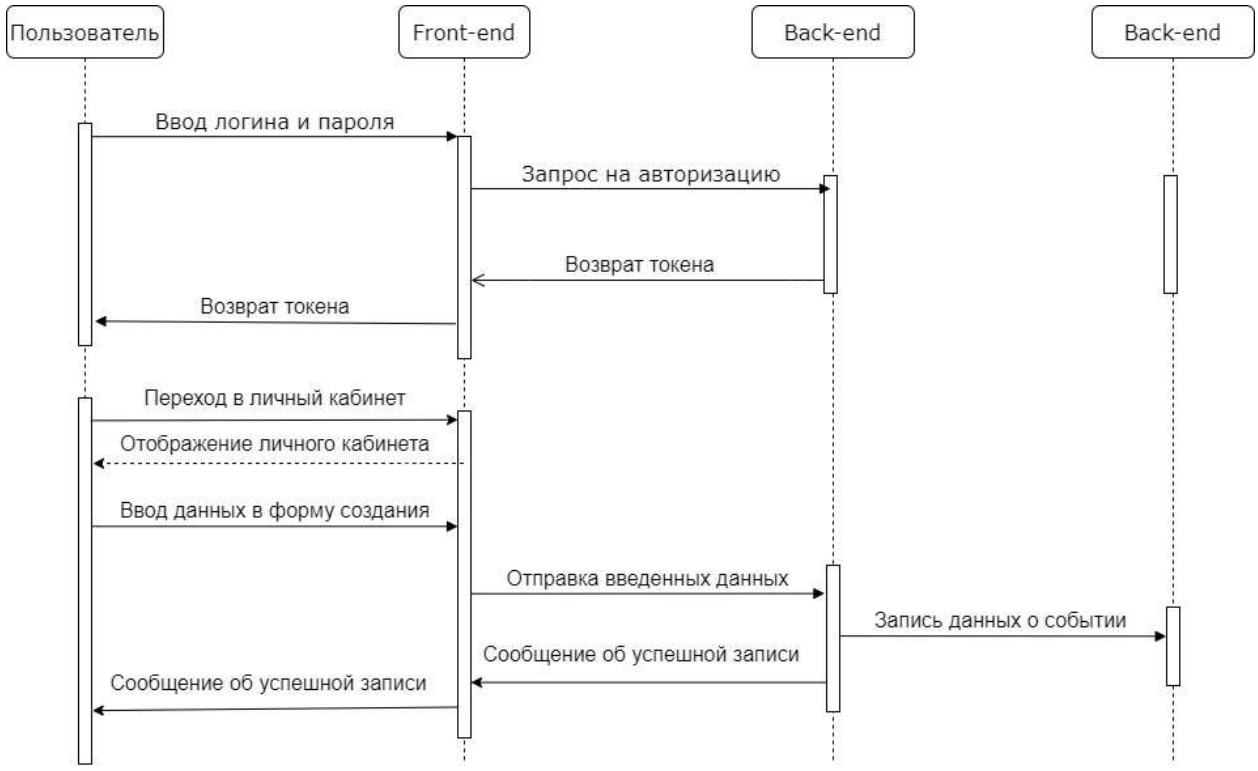

Рисунок 19 – Диаграмма последовательностей создания нового события

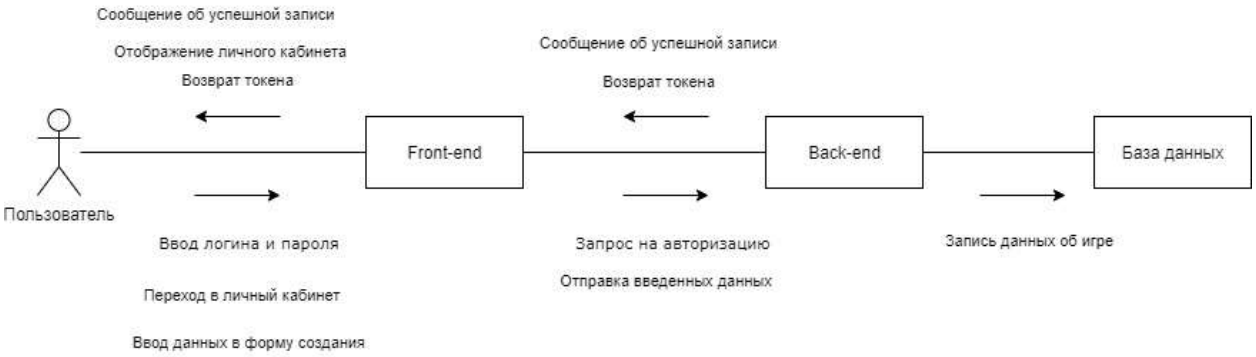

Рисунок 20 – Диаграмма взаимодействий создания нового события

Для создания нового события пользователь обязательно должен пройти авторизацию. После этого он переходит в личный кабинет. Там он вводит требуемые данные в форму создания. Front-end отправляет на сервер введенные данные, которые записываются в базу данных. Пользователю выводится сообщение об успешной записи.

На рисунке 22 показана диаграмма взаимодействий, где указаны отношения между компонентами системы при процессе модерации. На рисунке 21 представлена диаграмма последовательностей, на которой изображено упорядоченное во времени взаимодействие компонентов системы.

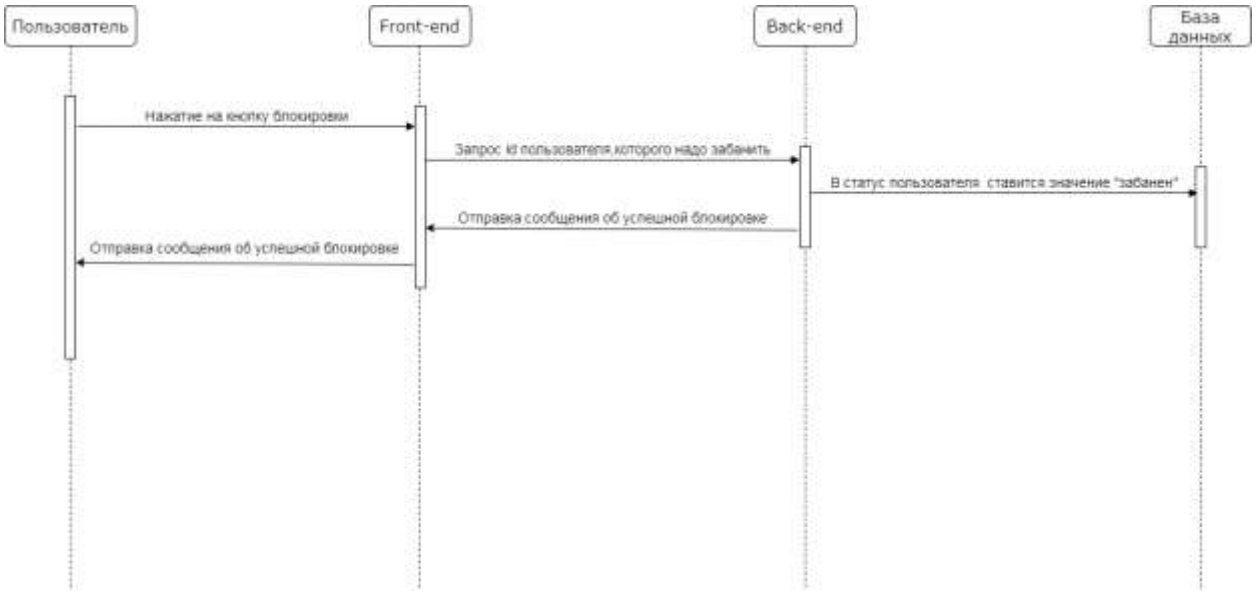

Рисунок 21 – Диаграмма последовательностей модерации системы

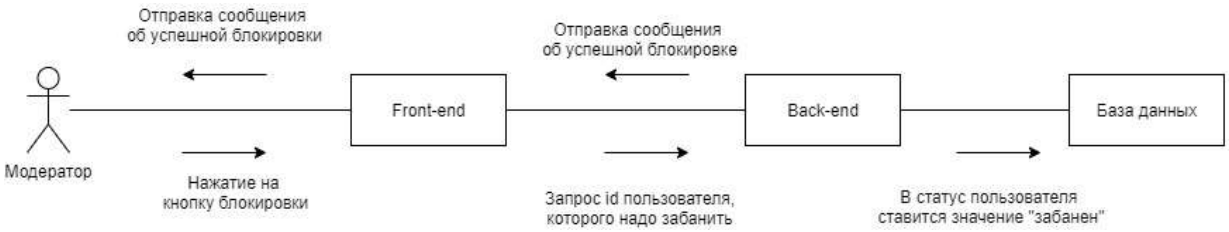

Рисунок 22 – Диаграмма взаимодействий модерации системы

Модератор нажимает на кнопку блокировки. С Front-end’а на Back-end делается запрос id пользователя, которого надо забанить. Далее в базе данных в статусе пользователя ставится соответствующее значение. И модератору отправляется сообщение об успешной блокировке.

2.6.6 Диаграмма развертывания

На рисунке 23 приведена диаграмма развертывания, визуализирующая программные и аппаратные компоненты, существующие на этапе исполнения.

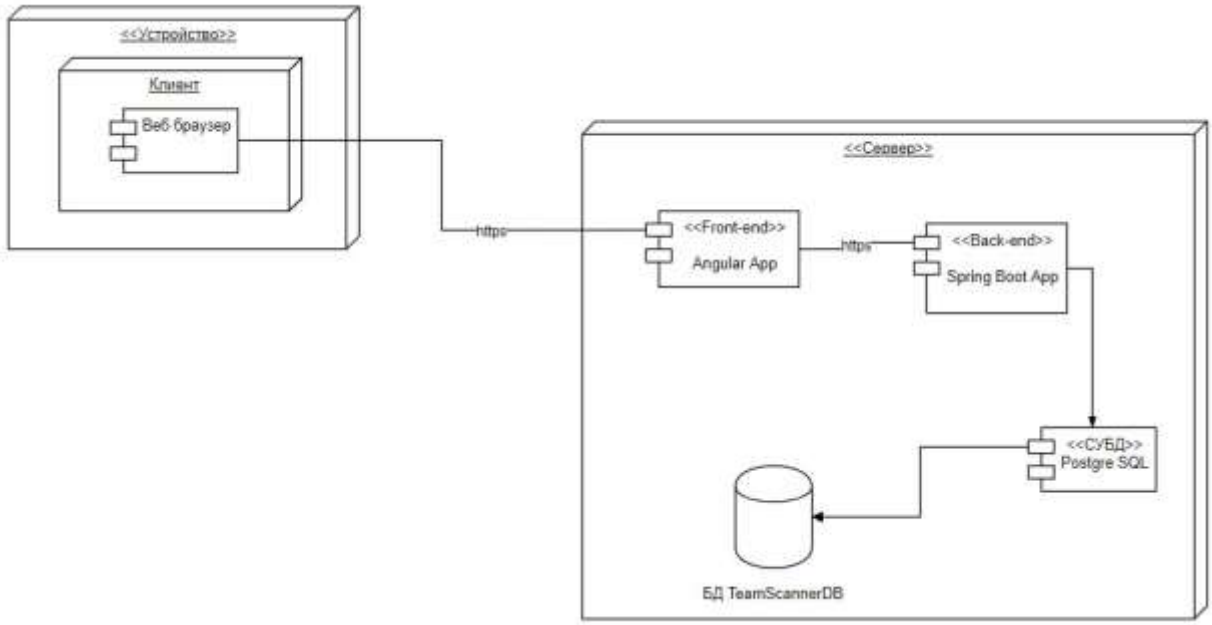

Рисунок 23 – Диаграмма развертывания

Для разрабатываемой системы узлами является рабочая станция клиента и сервер. В качестве узла среды выполнения выступает веб браузер.

Через него клиент заходит на адрес сайта в сети Интернет. На сервере разворачивается Front-end и Back-end, а также СУБД Postgre SQL. Общение между клиентом и сервером, осуществляется посредством отправки клиентом https запросов на Front-end и получения ответов. Общение между Back-end и Front-end осуществляет также посредством https протокола.

2.7 Анализ средств реализации

Для реализации нашей системы были выбраны следующие технологии:

1. В качестве языка программирования для Back-end части нашего приложения был выбран строго типизированный объектно-ориентированный язык программирования Java. Особенностями языка, повлиявшими на его выбор, были: ООП стиль программирования, платформонезависимость яз
2. В качестве дополнительных программных средств был использован для Back-end части был выбран фрейворк Spring Boot. Это обусловлено тем, что Spring Boot предоставляет каркас приложения. При этом фреймворк диктует правила построения приложения – есть определенная архитектура приложения, в которую вам нужно встроить свою функциональность. Эта функциональность, собственно, и будет бизнес логикой нашего приложения. Также в состав Spring Boot входит много подпроектов, «заточенных» под определенную функциональность (SpringMVC, Spring Security, SpringData и др.), некоторые из которых мы тоже использовали при разработке нашего приложения.
3. В качестве СУБД была выбрана PostgreSQL, так как оно является наиболее популярным в данный момент и регулярно обновляемым, а также просто интегрируется со Spring Boot фреймворком.
4. Для облегчения документации предоставляемых API использован SWAGGER.

5. Фреймворк для фронта – angular 8, так как это гибкий фреймворк, в котором испульзуется схема MVC, разделяющая логику, представление и данные приложения. Что как раз подходит для выбранной архитектуры проекта.
6. Данные передаются с помощью протокола HTTPS для повышения безопасности. Также этот протокол позволяет получить более точные данные о переходах на сайт, что поможет для составления метрик.

3 Реализация

3.1 Серверная часть приложения

3.1.1 Сущности

В приложении присутствуют 6 сущностей: «Базовая сущность», «Пользователь», «Роль», «Событие», «Статус», «Категория», «Комментарий». Диаграмма классов, отражающих их отношения изображена на рисунке 24.

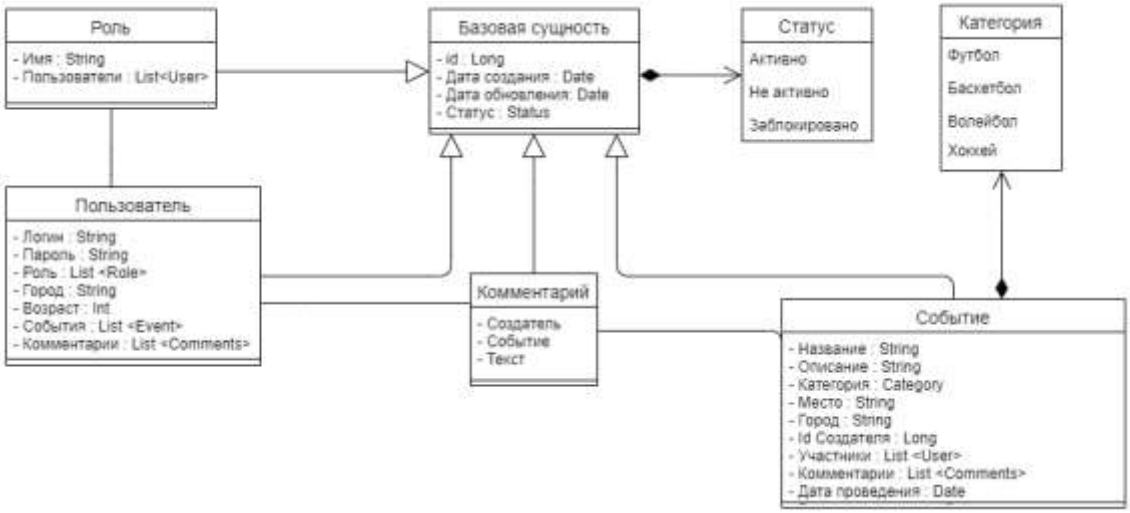

Рисунок 24 – Диаграмма классов сущностей

1. Класс «Базовая сущность» – является предком для классов – моделей базы данных. Содержит обязательные свойства, которые должны быть у сущности, которая хранится в базе данных. Класс имеет следующие свойства:
    - «id» – уникальные идентификатор.
    - «Дата создания» – дата, когда сущность была создана.
    - «Дата обновления» – дата, последнего изменения свойств сущности.
    - «Статус» – текущее состояние сущности.

2. Класс «Пользователь» – является наследником класса «Базовая сущность». Имеет свойства, характеризующие пользователя разрабатываемой системы, такие как:
    - «Логин» – уникальное свойство, используется для определения конкретного пользователя.
    - «Пароль» – свойство, используемое для защиты доступа к аккаунту пользователя. Хранится в базе данных в зашифрованном виде.
    - «Роль» – свойство, определяющее права пользователя на доступ к функциям системы.
    - «Город» – место, определяющее местоположение пользователя, используется для более точного подбора событий.
    - «Возраст» – дата рождения пользователя, необходимо для сортировки событий по среднему возрасту участников.
    - «События» – свойство, которое реализует связь многие ко многим с классом «Событие».
    - «Комментарии» – свойство, которое реализует связь один ко многим с классом «Комментарий».
3. Класс «Роль» – является наследником класса «Базовая сущность». Необходим для определения прав пользователя, согласно которым осуществляется доступ к функциям приложения. Класс имеет следующие свойства:
    - «Имя» – определяет название роли.
    - «Пользователи» – свойство, которое реализует связь многие ко многим с классом «Пользователь».
4. Класс «Событие» – является наследником класса «Базовая сущность». Содержит свойства, полностью описывающие содержание отдельного события, такие как:
    - «Название» – определяет название события.

- «Описание» – определяет описание события, необходимо чтобы описать цели создания события.
- «Категория» – определяет категорию вида спорта, к которому относится событие.
- «Место» – описание места, где участники и создатели события договариваются встретиться.
- «Город» – определяет город, в котором будет происходит событие.
- «Id Создателя» – внешний ключ к таблице «Пользователь» в базе данных.
- «Участники» – свойство, которое реализует связь многие ко многим с классом «Пользователь».
- «Комментарии» – свойство, которое реализует связь один ко многим с классом «Комментарий».
- «Дата проведения» – свойство, определяющее дату проведения события.

5. Перечисление «Статус» – определяет текущее состояние, в котором может находиться событие или пользователь.
6. Перечисление «Категория» – определяет категорию вида спорта события.
7. Класс «Комментарий» – является наследником класса «Базовая сущность» – определят свойства комментариев, которые пользователи оставляют в событиях, такие как:
    - «Создатель» – пользователь, создавший комментарий. Свойство, реализует связь многие к одному с классом «Пользователь».
    - «Событие» – определяет принадлежность к событию, реализует связь многие к одному с классом «Событие».
    - «Текст» – текстовое содержание комментария.

3.1.2 Хранение данных и бизнес логика

Для работы с данным в серверном приложении используется такой механизм как Spring Data. Он осуществляет взаимодействие с сущностями базы данных, организует их в репозитории, занимается изменением и извлечением данных.

Основное понятие в Spring Data — это репозиторий. Это несколько интерфейсов которые используют JPA Entity для взаимодействия с ней. Так, например интерфейс JpaRepository обеспечивает основные операции по поиску, сохранения, удалению данных (CRUD операции) и др. операции.

Рассмотрим, используемые в приложении репозитории:

«CommentRepository» – репозиторий, интерфейс которого представлен на рисунке 25, используется для:

1. Сохранения комментариев.
2. Удаления комментариев.
3. Извлечения комментариев.

```
public interface CommentRepository extends JpaRepository<Comment, Long> {}
```

Рисунок 25 – Интерфейс репозитория «CommentRepository»

«EventRepository» – репозиторий, интерфейс которого представлен на рисунке 26, используется для:

1. Сохранения событий.
2. Удаления событий.
3. Извлечения событий.
4. Поиска событий по названию.
5. Поиска событий по статусу.
6. Поиска событий по идентификатору создателя события.

```
public interface EventRepository extends JpaRepository<Event, Long>, JpaSpecificationExecutor<Event> {
    List<Event> findByName(String name);

    List<Event> findByStatus(Status status);

    List<Event> findByCreatorId(Long id);
}
```

Рисунок 26 – Интерфейс репозитория «EventRepository»

«RoleRepository» – репозиторий, интерфейс которого представлен на рисунке 27, используется для:

1. Сохранения ролей.
2. Удаления ролей.
3. Извлечения ролей.
4. Поиска ролей по названию.

```
public interface RoleRepository extends JpaRepository<Role, Long> {
    Role findByName(String name);
}
```

Рисунок 27 – Интерфейс репозитория «RoleRepository»

«UserRepository» – репозиторий, интерфейс которого представлен на рисунке 28, используется для:

1. Сохранения пользователей.
2. Удаления пользователей.
3. Извлечения пользователей.
4. Поиска пользователей по логину.
5. Поиска пользователей по статусу.

```
public interface UserRepository extends JpaRepository<User, Long> {
    User findByLogin(String name);

    List<User> findByStatus(Status status);
}
```

Рисунок 28 – Интерфейс репозитория «UserRepository»

Бизнес логика серверного приложения состоит из следующих классов:

1. «UserServiceImpl» – сервис, содержит функции по работе с классом «Пользователь» такие как:
    - Изменение ролей пользователя.
    - Удаление ролей пользователя.
    - Шифрование пароля пользователя.
    - Проверка пароля пользователя.
    - Регистрация пользователя.
    - Получение списка пользователей.
    - Поиск пользователя по идентификатору.
    - Поиск пользователя по логину.
    - Удаление пользователя.
2. «EntityManagerService» – сервис, который содержит следующие функции:
    - Получение списка событий, на которые подписан пользователь.
    - Выполнение произвольных SQL запросов к базе данных.
    - Получение логина пользователя по идентификатору пользователя

3.1.3 Контроллеры

За реализацию REST API для сообщения с пользовательской частью приложения отвечают шесть REST контроллеров.

1. **«UserRestControllerV1»** – контроллер, отвечающий за действия с пользователями, предоставляет следующие возможности:
    - Получение пользователя по идентификатору.
    - Изменение пользователя.
    - Получение списка городов.

2. **«AuthenticationRestControllerV1»** – контроллер, отвечающий за авторизацию и регистрацию, предоставляет следующие возможности:

- Получение токена авторизации.
- Регистрация пользователя в системе и получение токена авторизации.

3. **«EventController»** – контроллер, отвечающий за работу с событиями, предоставляет следующие возможности:
    - Проверка подписки пользователя на событие.
    - Создание нового события.
    - Удаление события.
    - Подписка пользователя на событие.
    - Отписка пользователя от события.
    - Изменение события.
    - Получение списка событий.
    - Получение списка событий, на которые подписан пользователь.
    - Получение списка событий, которые создал пользователь.
    - Получение логина создателя события по идентификатору.
    - Поиск событий по имени.
    - Сортировка событий по городу, дате проведения и категории.

4. **«CommentController»** – контроллер, отвечающий за работу с комментариями, предоставляет следующие возможности:
    - Получение списка комментариев по идентификатору события.
    - Добавление комментариев.
    - Удаление комментариев.

5. **«ModerController»** – контроллер, предоставляющий возможности по модерированию системы, такие как:
    - Получение пользователя по идентификатору.

- Изменение статуса пользователя.
- Изменение статуса события.
- Получение списка заблокированных пользователей.
- Получение списка заблокированных событий.
- Поиск событий по названию события.
- Поиск пользователя по логину пользователя.
- Получение списка пользователей.

6. **«AdminRestControllerV1»** – контроллер, предоставляющий возможности по администрированию системы, такие как:
    - Поиск пользователя с правами модератора по логину пользователя.
    - Получение списка пользователей с правами модератора.
    - Получение списка пользователей.
    - Поиск пользователя по логину пользователя.
    - Изменение прав модератора у пользователей.

3.5 Клиентская часть приложения

Клиентская часть приложения написана на фреймворке Angular 8. Разберем, что представляет собой архитектура Angular приложения.

Сам фреймворк состоит из нескольких библиотек (или модулей), каждая из которых содержит в себе определенный функционал, а каждый модуль состоит из совокупности классов и их свойств и методов. Каждый класс имеет свое функциональное предназначение. Не все библиотеки обязательны для использования в приложении (англ. Angular App), часть подключается по мере необходимости.

Разберем **модули**, именно с них начинается проектирование архитектуры Angular приложения. Каждый из них имеет собственный набор структурных элементов:

- **component** – отвечает за часть web-страницы и включает в себя HTML–шаблон, CSS-стили и логику поведения;
- **service** – поставщик данных для component;
- **directive** – преобразует определенную часть DOM заданным образом.

Все вышеперечисленное собирается в корневой модуль, который общепринято называется **AppModule**.

Компонент – это часть интерфейса приложения с собственной логикой. Вся видимая часть Angular App реализуется с помощью компонентов. Основные свойства объекта, который принимает декоратор:

- **selector** – название компонента;
- **template** (или templateUrl) – HTML-разметка в виде строки (или путь к HTML-файлу);
- **providers** – список сервисов, поставляющих данные для компонента;
- **styles** – массив путей к CSS-файлам, содержащим стили для создаваемого компонента.

Рассмотрим компоненты, из которых состоит клиентская часть приложения:

1. **«header»** – верхняя часть каждой страницы сайта, которая состоит из картинки логотипа сайта и ссылок навигации по нему.
2. **«homepage»** – домашняя страница сайта.
3. **«login-page»** – страница для авторизации пользователей.
4. **«register-page»** – страница регистрации новых пользователей.
5. **«event-search»** – страница поиска событий с фильтрами сортировки событий по определённым параметрам.
6. **«event-page»** – страница подробного просмотра события.

7. **«create-event»** – страница создания нового события.
8. **«edit-event»** – страница редактирования уже созданного события.
9. **«events»** – страница просмотра событий, созданных пользователем в личном кабинете.
10. **«subscriptions»** – страница просмотра событий, на которые подписался пользователь.
11. **«user-data»** – страница для просмотра и редактирования личных данных.
12. **«moder-page»** – страница для управления статусами обычных пользователей, а также управления статусами событий. Доступна только для пользователей с ролью модератора и администратора.
13. **«admin-page»** – страница управления ролями пользователей. Доступна только для пользователей с ролью администратора.

Сервисы нужны для предоставления данных компонентам. Это могут быть не только запросы к серверу, но и функции, преобразующие исходные данные по заданному алгоритму. Они позволяют архитектуре Angular приложения быть более гибкой и масштабируемой.

Для клиентской части приложения были разработаны следующие сервисы:

«auth.service» – сервис по работе с данными авторизированного пользователя. Выполняет функции по за отправку запросов на серверную часть приложения при регистрации и авторизации пользователей, за хранение информации о пользователе и предоставлении её для компонентов.

«event-service» – сервис по работе с данными, составляющими подробное описание события.

4 Интерфейс

4.1 Задача поиска события и подписка

При открытии приложения пользователь видит страницу, представленную на рисунке 29. Он может пройти авторизацию и регистрации, нажав на «Вход» или попробовать найти существующие события в Поиске событий.

Рисунок 29. Стартовая страница приложения

При переходе в поиск событий пользователь сразу видит все существующие объявления. Он может посмотреть интересующее объявление подробнее. Если же пользователь хочет провести поиск по фильтрам, он выбирает категорию вида спорта, город проведения, дату события. Эти фильтры можно сбросить, нажав на соответствующую кнопку. Интерфейс страницы представлен на рисунке 30.

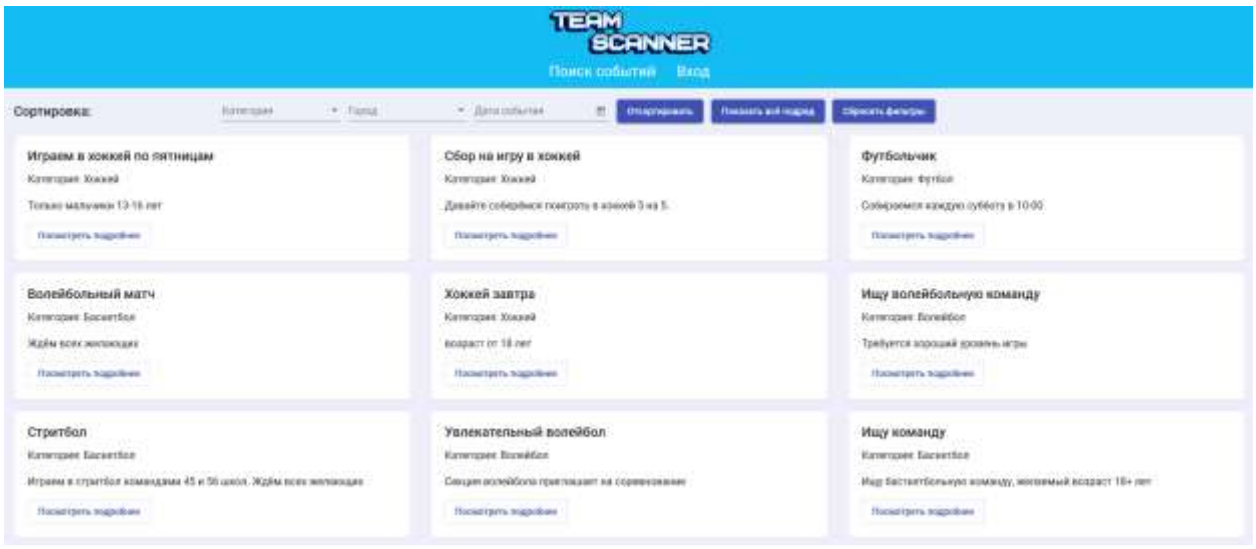

Рисунок 30. Страница поиска

Для подробного просмотра события открывается соответствующая страница, представленная на рисунке 31. Здесь представлена информация о создателе события, дате и времени, категории, количестве участников, адресе и описании события. Пользователи могут подписываться на событие и оставлять комментарии на этой странице. Но эти возможности доступны только авторизованным пользователям.

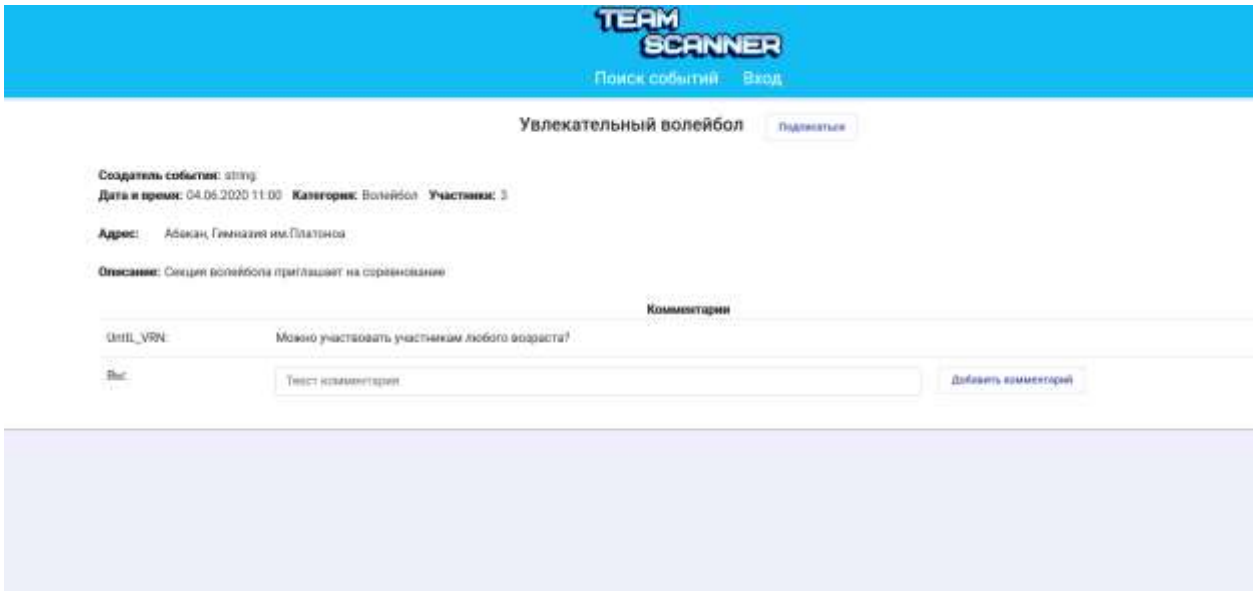

Рисунок 31. Страница подробного просмотра события

Далее по продуктовому сценарию следует авторизация. Окно авторизации представлено на рисунке 32. Пользователь может войти в свою

учетную запись, при наличии, или зарегистрироваться по соответствующей кнопке.

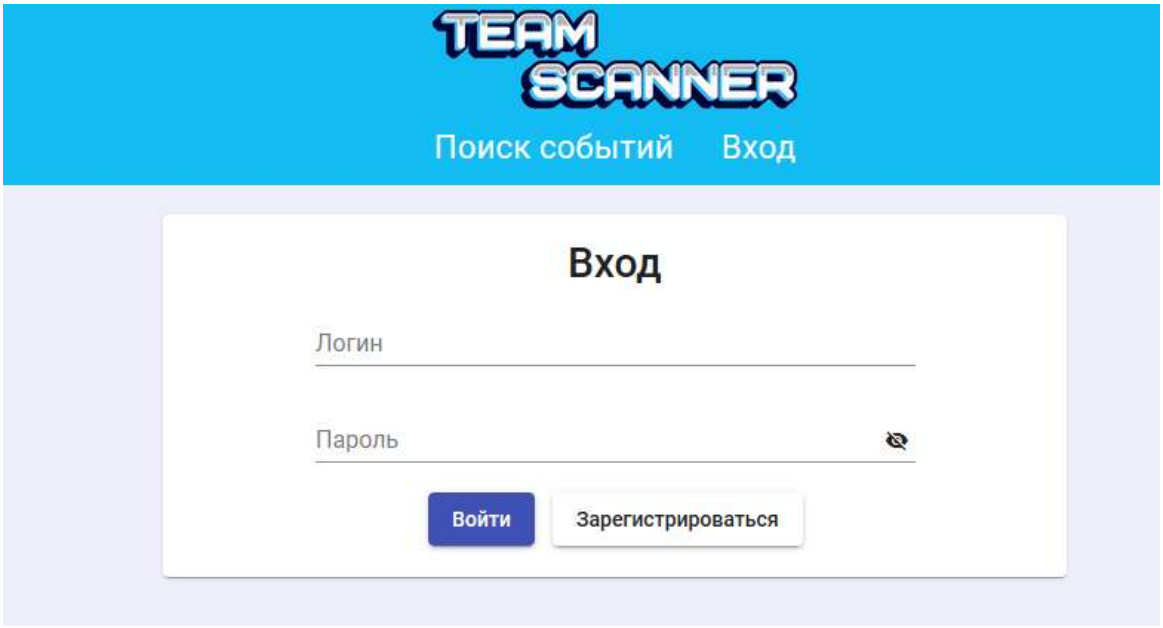

Рисунок 32. Окно авторизации

На рисунке 33 представлено окно регистрации. Пользователь указывает:

- Логин
- Свой населенный пункт
- Дату рождения
- Пароль
- Пароль повторно

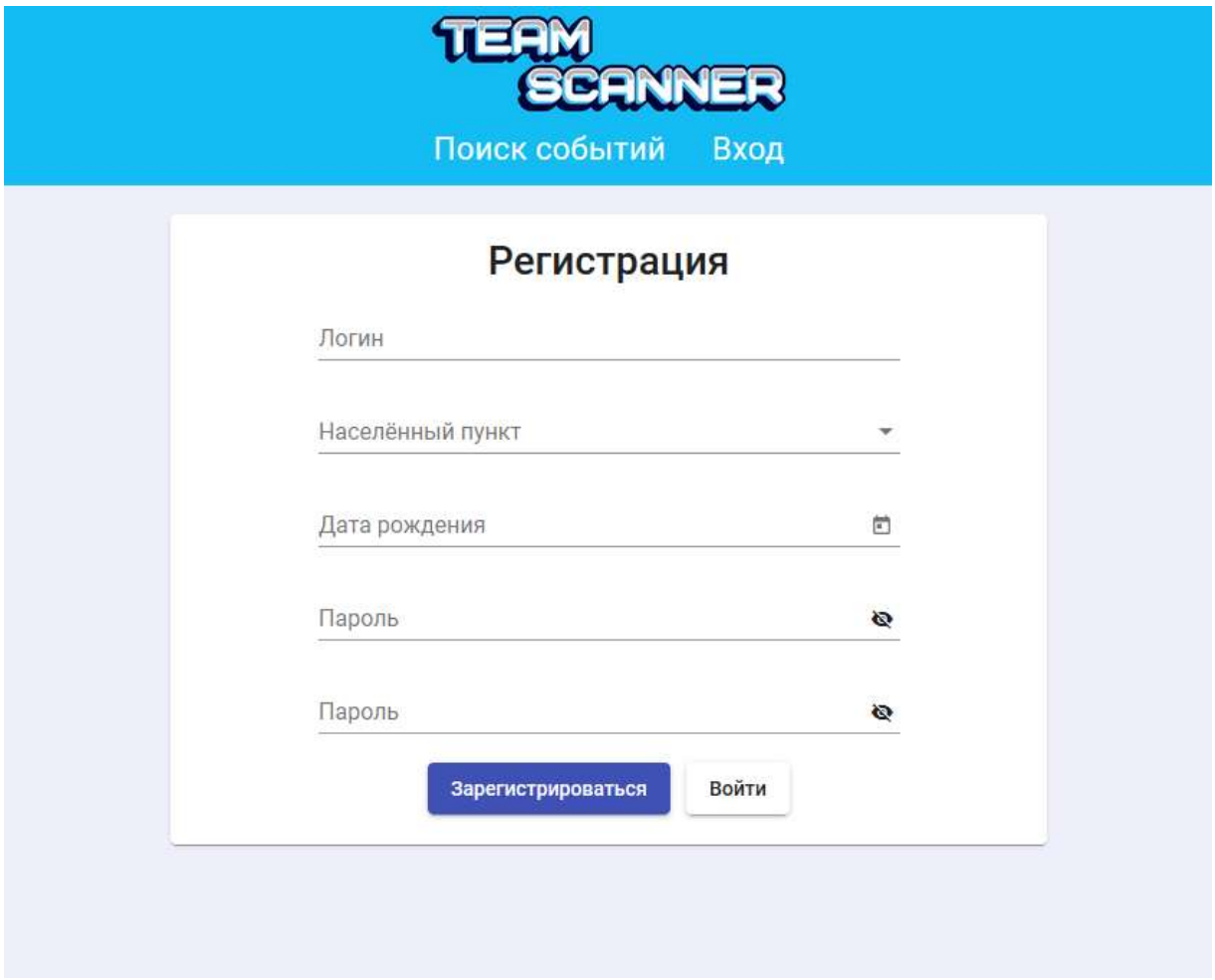

Рисунок 32. Страница регистрации

После успешной регистрации пользователь может вернуться к интересующему событию через формы поиска и подробного просмотра и подписаться на него или оставить комментарий.

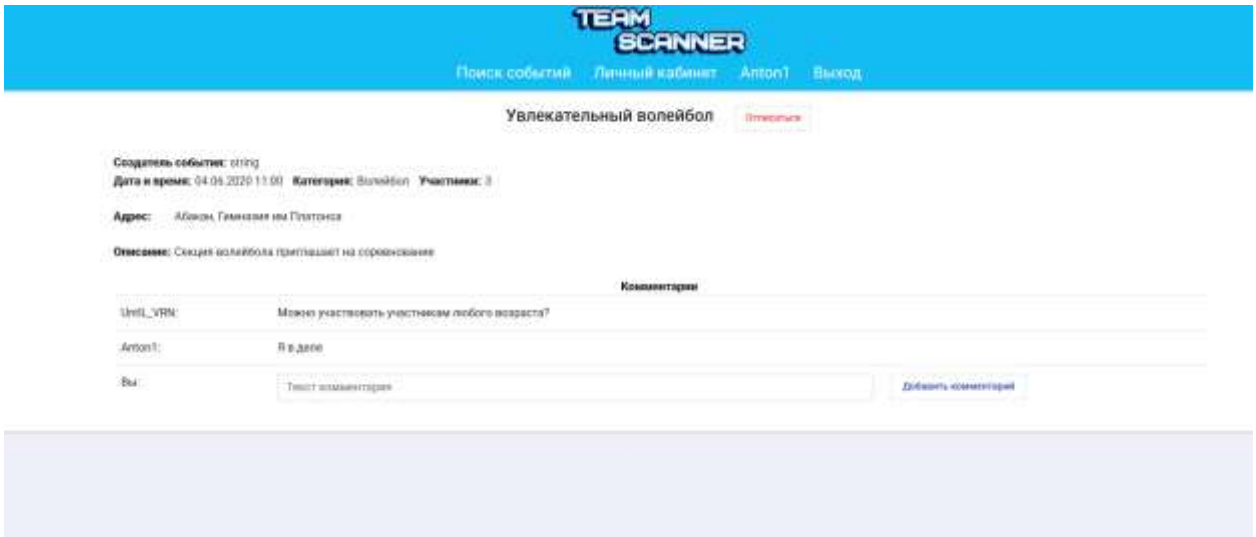

Рисунок 33. Окно подробного просмотра при оформленной подписке

Подписку можно посмотреть перейдя в личный кабинет, что представлено на рисунке 34.

Рисунок 34. Просмотр подписок

4.2 Задача создания события

Создание событий доступно только авторизованным пользователям. Форма создания представлена на рисунке 35. Для создания он заполняет все поля и нажимает на кнопку создания.

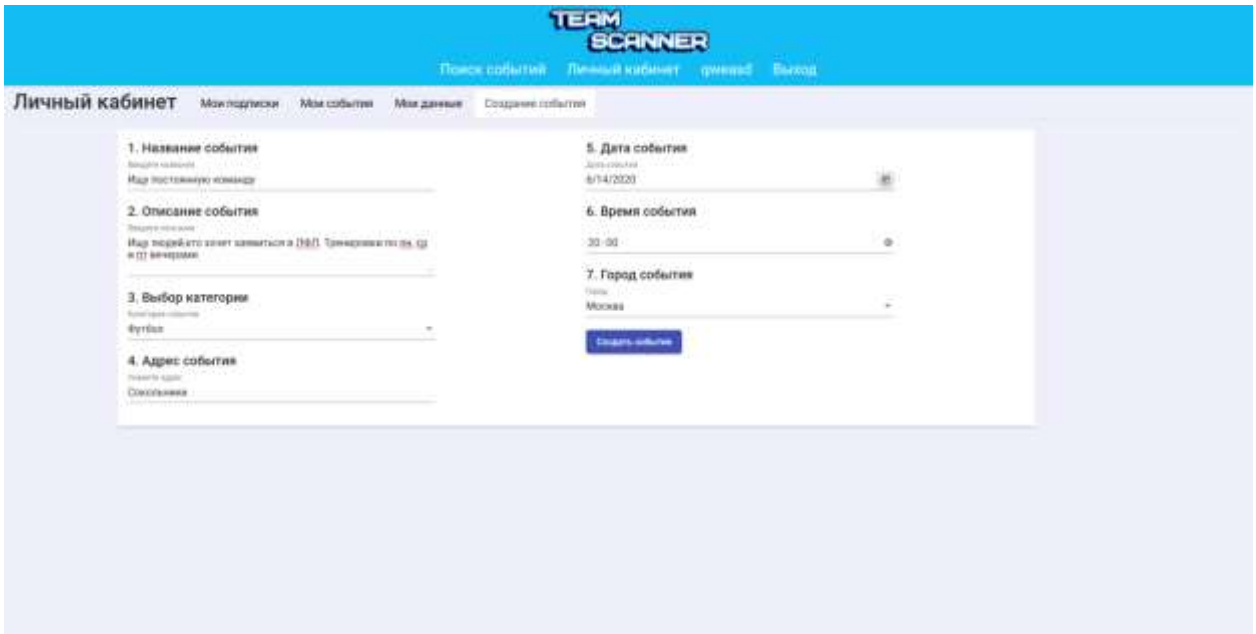

Рисунок 35. Форма создания события

Созданные события пользователь может потом посмотреть на вкладке «Мои события», представленной на рисунке 36. Также он может посмотреть объявление подробнее, редактировать его или удалить.

Рисунок 36. Просмотр списка созданных событий

4.3 Задача модерации системы

Если пользователь является модератором, на его главной странице появляется дополнительный функционал. Модератор может искать пользователей или события и их блокировать. Страница модерации представлена на рисунке 37.

Рисунок 37. Страница модерации

В системе предусмотрена разблокировка пользователей и событий. Форма разблокировки событий представлена на рисунке 38. Для разблокировки пользователей предусмотрена аналогичная страница.

Рисунок 38. Форма разблокировки событий

В системе предусмотрена учетная запись администратора. В его полномочия входит весь функционал системы, в том числе присвоение и лишение пользователя роли модератора. Страница администрирования представлена на рисунке 39.

Рисунок 39. Страница администрирования

5 Тестирование

После реализации всех задач был проведен запланированный набор тестов, а именно:

- Дымовое тестирование
- UI тесты

5.1 Дымовое тестирование

Для данного тестирования необходимо было проверить работоспособность приложения на всех основных сценариях:

- Создание события
- Поиск и подписка на событие
- Модерация системы

Тестирование проводилось вручную и охватило также дополнительные сценарии. Его результаты представлены на таблице 1.

Таблица 1 – Результаты дымового тестирования

№ п/п	Название	Шаги	Ожидаемый результат	Статус теста
1	Регистрация	Нажатие на кнопку "Вход" Нажатие на копку "Зарегистрироваться" Ввод логина Выбор населенного пункта Ввод даты рождения Ввод пароля Подтверждение пароля Нажатие на копку "Зарегистрироваться"	Успешная регистрация и отображение главной страницы	Пройден
2	Авторизация	Нажатие на кнопку "Вход" Ввод логина Ввод пароля Нажатие на копку "Войти"	Успешная авторизация и отображение главной страницы	Пройден

Продолжение таблицы 1

3	Поиск и подписка на событие	Нажатие на кнопку "Вход" Ввод логина Ввод пароля Нажатие на копку "Войти" Нажатие на карточку вида спорта Ввод фильтров поиска Нажатие на кнопку "Отсортировать" Нажатие кнопку "Посмотреть подробнее" у события Нажатие на кнопку "Подписаться"	Пользователь подписывается на событие, кнопка "Подписаться" меняется на "отписаться"	Пройден
4	Создание события	Нажатие на кнопку "Вход" Ввод логина Ввод пароля Нажатие на копку "Войти" Нажатие на кнопку "Личный кабинет" Нажатие на кнопку "Создание события" Ввод названия события Ввод описания события Выбор категории вида спорта Ввод адреса проведения события Ввод даты проведения события Ввод времени проведения события Выбор города проведения события Нажатие на кнопку "Создать событие"	Добавление события в базу данных, появление окна об успешном создании события	Пройден
5	Блокировка пользователя	Авторизация под учетной записью модератора Нажатие на кнопку "Модерация" Ввод логина пользователя Нажатие на кнопку "Найти" Нажатие на кнопку заблокировать	Блокировка пользователя	Пройден
6	Блокировка события	Авторизация под учетной записью модератора Нажатие на кнопку "Модерация" Ввод названия события Нажатие на кнопку "Найти" Нажатие на кнопку заблокировать	Блокировка события	Пройден

Продолжение таблицы 1

7	Разблокировка пользователя	Авторизация под учетной записью модератора Нажатие на кнопку "Модерация" Переход на вкладку "Заблокированные пользователи" Нажатие на кнопку "Разблокировать" рядом с нужным пользователем	Разблокировка пользователя	Пройден
8	Разблокировка события	Авторизация под учетной записью модератора Нажатие на кнопку "Модерация" Переход на вкладку "Заблокированные события Нажатие на кнопку "Разблокировать" рядом с нужным событием	Разблокировка события	Пройден
9	Добавление модератора	Авторизация под учетной записью администратора Нажатие на кнопку "Администрирование" Нажатие на кнопку "Сделать модератором" рядом с нужным пользователем	Присвоение пользователю роли модератора	Пройден
10	Удаление модератора	Авторизация под учетной записью администратора Нажатие на кнопку "Администрирование" Переход на вкладку "Снятие модераторов" Нажатие на кнопку "Снять модераторские полномочия	Пользователь теряет должность модератора	Пройден

По итогам тестирования было установлено, что приложение удовлетворяет требованиям основных сценариев.

5.2 UI тесты

Во время UI тестирования была проверена работоспособность всех возможностей приложения, с которыми может работать пользователь. Его результаты представлены на таблице 2.

Таблица 2 – Результаты UI тестирования

№ п/п	Начальная страница	Название	Шаги	Ожидаемый результат	Статус теста
1	Главная	Выбор карточки вида спорта	Нажатие на карточку на главной странице	Откроется форма поиска с выбраной категорией	Пройден
2	Поиск событий	Поиск по всем фильтрам существующих событий	Переход на форму поиска Ввод всех фильтров для поиска Нажатие на кнопку "Отсортировать"	Отображение нужных событий	Пройден
3	Поиск событий	Показ всех существующих событий	Переход на форму поиска Нажатие на кнопку "Показать все подряд"	Показ всех существующих событий	Пройден
4	Поиск событий	Сброс фильтров	Переход на форму поиска Ввод фильтров для поиска Нажатие на кнопку "Сбросить фильтры"	Очистка категории, города и даты	Пройден
5	Поиск событий	Поиск несуществующих событий	Переход на форму поиска Ввод фильтров, по которым не существуют события Нажатие на кнопку "Отсортировать"	Отображение надписи, что события не найдены	Пройден
6	Любая	Переход на главную страницу	Нажатие на логотип "TeamScanner"	Переход на главную страницу	Пройден
7	Главная	Переход к авторизации	Нажатие на кнопку "Вход"	Переход к форме авторизации	Пройден
8	Авторизации	Переход в учетную запись	Ввод верного логина Ввод верного пароля Нажатие на кнопку "Войти"	Отображение главной страницы авторизованного пользователя	Пройден
9	Авторизации	Неверный ввод данных при авторизации	Ввод верного логина Ввод неверного пароля	Отображение окна неверного ввода	Пройден
10	Авторизации	Приватное отображение пароля	Ввод пароля пользователем	Отображение символов "*"	Пройден

Продолжение таблицы 2

11	Авторизации	Ввод слишком короткого пароля	Ввод пароля менее 6 символов	Отображение окна, что длина должна быть более 6 символов	Пройден
12	Авторизации	Переход к форме регистрации	Нажатие на кнопку "Зарегистрироваться"	Отображение формы регистрации	Пройден
13	Регистрации	Приватное отображение пароля	Ввод пароля пользователем	Отображение символов "*"	Пройден
14	Регистрации	Некорректный ввод даты	Ввод текста в поле даты рождения	Подсветка поля красным цветом	Пройден
15	Регистрации	Переход в учетную запись	Заполнение формы регистрации Нажатие на кнопку "Зарегистрироваться"	Отображение главной страницы авторизованного пользователя	Пройден
16	Просмотра события	Проверка кнопки подписки	Нажатие на кнопку "Подписаться"	На кнопке меняется надпись на "Отписаться", кнопка становтся красной	Пройден
17	Просмотра события	Проверка отображения комментария	Набор текста комментария Нажатие на кнопку "Добавить комментарий"	На страницу добавляется написанный комментарий	Пройден
18	Просмотра события	Проверка кнопки отписки	Нажатие на кнопку "Отписаться"	Отображается синяя кнопка "Подписаться"	Пройден
19	Поиск событий	Переход к найденному событию	Нажатие на кнопку "Посмотреть подробнее"	Открытие окна просмотра события	Пройден
20	Личный кабинет	Переход на поиск игр при отсутствии подписок	Нажатие на ссылку по слову "здесь"	Отображение страницы поиска	Пройден
21	Личный кабинет	Переход к созданию события при отсутствии текущих	Нажатие на вкладку "Мои события" Нажатие на ссылку по слову "здесь"	Отображение формы создания события	Пройден
22	Личный кабинет	Отображение личных данных	Нажатие на вкладку "Мои данные"	Отображение данных пользователя	Пройден
23	Личный кабинет	Отображение формы создания события	Нажатие на вкладку "Создание события"	Отображение формы создания события	Пройден

Продолжение таблицы 2

24	Любая	Выход из учетной записи и отображение главной страницы	Нажатие на кнопку "Выход"	Отображение главной страницы неавторизованного пользователя	Пройден
25	Авторизации	Отображение учетной записи модератора	Ввод логина ,пароля модератора, нажатие на кнопку "Войти"	Отображение главной страницы учетной записи модератора	Пройден
26	Главная модератора	Переход к форме модерации	Нажатие на кнопку "Модерация"	Отображение поиска пользователей и событий	Пройден
27	Модерации	Отображение искомого пользователя	Ввод логина пользователя Нажатие на кнопку "Найти"	Отображение пользователя и кнопки "Заблокировать"	Пройден
28	Модерации	Проверка кнопки блокировки пользователя	Нажатие на кнопку "Заблокировать"	Пользователь пропадает из списка	Пройден
29	Модерации	Отображение искомого события	Ввод названия события Нажатие на кнопку "Найти"	Отображение события и кнопки "Заблокировать"	Пройден
30	Модерации	Проверка кнопки блокировки события	Нажатие на кнопку "Заблокировать"	Событие пропадает из списка	Пройден
31	Модерации	Отображение заблокированных пользователей	Нажатие на вкладку "Заблокированные пользователи"	Отображается список заблокированных пользователей	Пройден
32	Модерации	Проверка кнопки разблокировки пользователя	Нажатие на вкладку "Заблокированные пользователи" Нажатие на кнопку "Разблокировать"	Пользователь пропадает из списка	Пройден
33	Модерации	Отображение заблокированных событий	Нажатие на вкладку "Заблокированные события"	Отображение списка заблокированных событий	Пройден
34	Модерации	Проверка кнопки разблокировки события	Нажатие на вкладку "Заблокированные события" Нажатие на кнопку "Разблокировать"	Событие пропадает из списка	Пройден

Продолжение таблицы 2

35	Авторизации	Отображение главной страницы администратора	Ввод логина,пароля администратора нажатие на кнопку "Войти"	Отображение главной страницы администратора	Пройден
36	Главная администратора	Переход к форме администрирования	Нажатие на кнопку "Администрирование"	Отображение формы администрирования	Пройден
37	Администрирование	Проверка кнопки назначения модератором	Ввод логина пользователя Нажатие на кнопку "Сделать модератором"	Пользователь пропадает из списка	Пройден
38	Администрирование	Проверка кнопки удаления модератора	Ввод логина пользователя Нажатие на кнопку "Снять модераторские полномочия	Пользователь пропадает из списка	Пройден

По итогу, приложение успешно UI тестирование.

Заключение

В ходе выполнения курсового проекта была выявлена необходимость в создании приложения по поиску команды для игр разных видов спорта. Был проведен анализ этой предметной области и установлен список ключевых сценариев, а именно:

- Создание события
- Поиск события и подписка
- Модерация системы

Для этих ключевых сценариев были составлены диаграммы прецедентов, последовательностей, коммуникаций, состояния, активности и развертывания.

В результате, реализовано приложение, позволившее решить поставленные задачи:

- Регистрацию, авторизацию, просмотр личного кабинета, выход из Учетной записи
- Создание объявления о поиске игроков, обмен комментариями
- Поиск объявлений по нескольким критериям
- Возможность откликнуться на объявление о поиске игроков и оставить комментарий
- Возможность модерации системы

Это приложение удовлетворяет изначальным требованиям:

- Разделение приложения на Front-end и Back-end
- Взаимодействие между ними с помощью REST API
- Наличие базы данных, содержащей не менее трех таблиц

Список использованных источников

1 Мэтт Вайсфельд. Объектно-ориентированное мышление – Санкт-Петербург: Питер, 2014. – 304 с.

2 Карл И. Вигерс. Разработка требований к программному обеспечению -

3 Spring Framework Documentation [сайт] – URL: https://docs.spring.io/spring/docs/current/spring-framework-reference/ (дата обращения: 20.05 – 31.05.2019)

4 Чистый код. Создание, анализ и рефакторинг – Robert C. Martin 2010.

5 Spring в действии – Крейг Уоллс 2015.

6 Введение в Spring Boot: создание простого REST API на Java – URL: https://habr.com/ru/post/435144/

7 Учебник AngularJS: Всеобъемлющее руководство, часть 1 – URL: https://habr.com/ru/post/246881/

8 Учебник AngularJS: Всеобъемлющее руководство, часть 2 – URL: https://habr.com/ru/post/247283/