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.

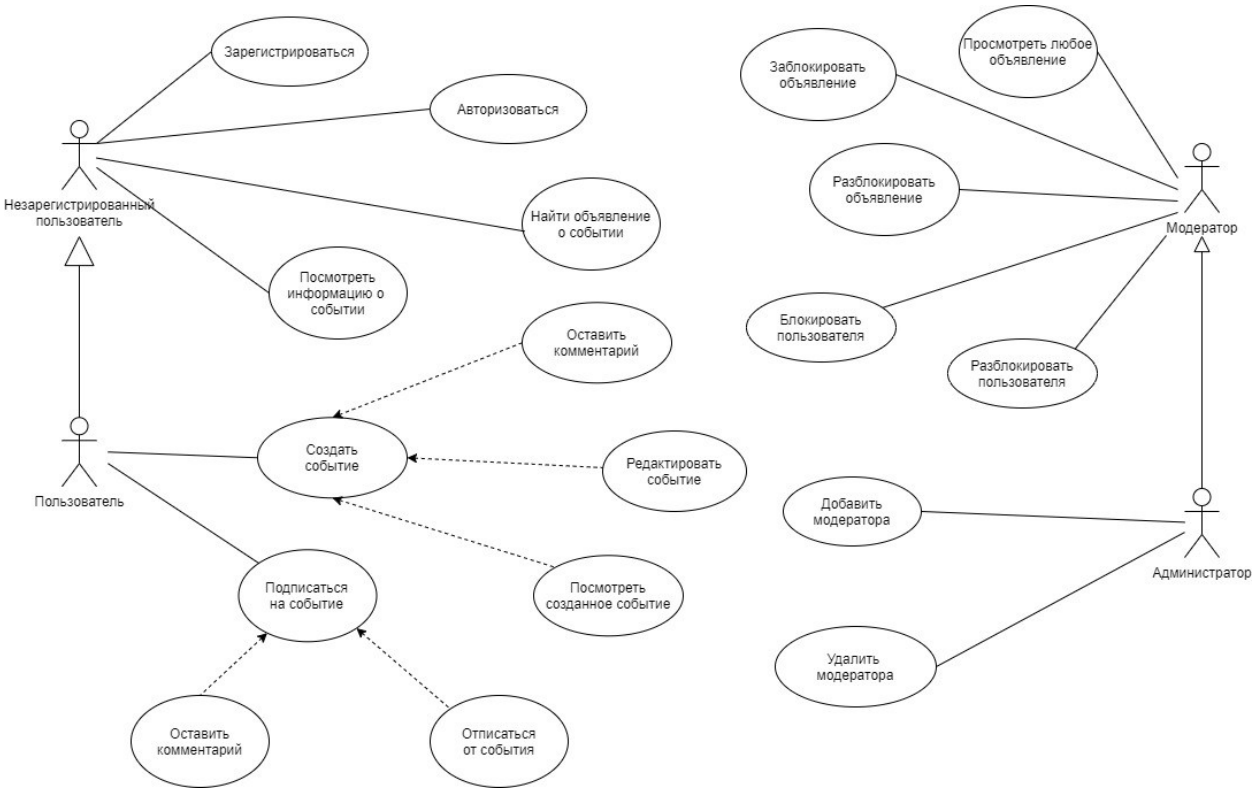

Рисунок 1 – Диаграмма прецедентов

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.

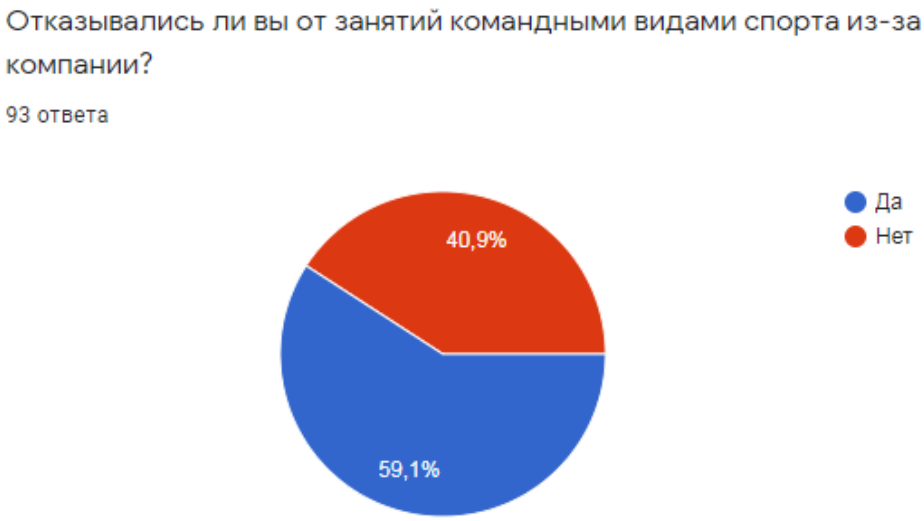

Рисунок 3 – Результаты опроса

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.

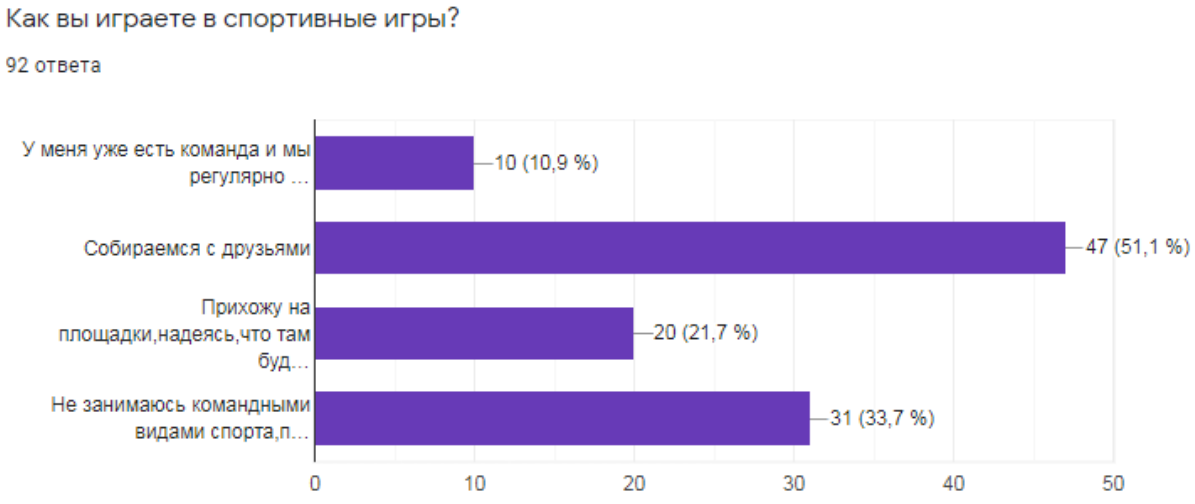

Рисунок 4 – Результаты опроса

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 нео

автоматизировал 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 архитектуры

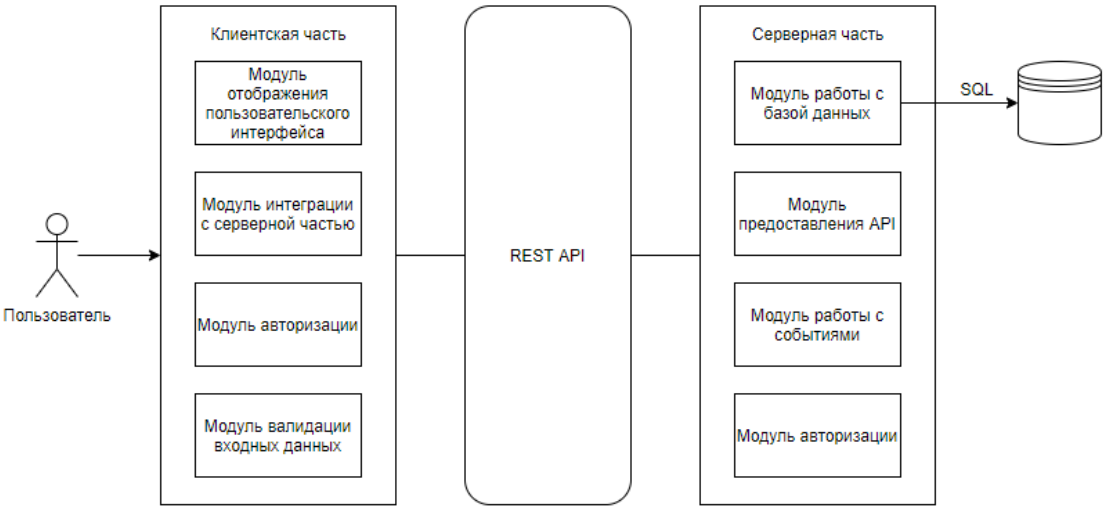

Рисунок 7 - Архитектура приложения

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.

2.5. Графическое описание работы системы

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

2.5.1. 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.

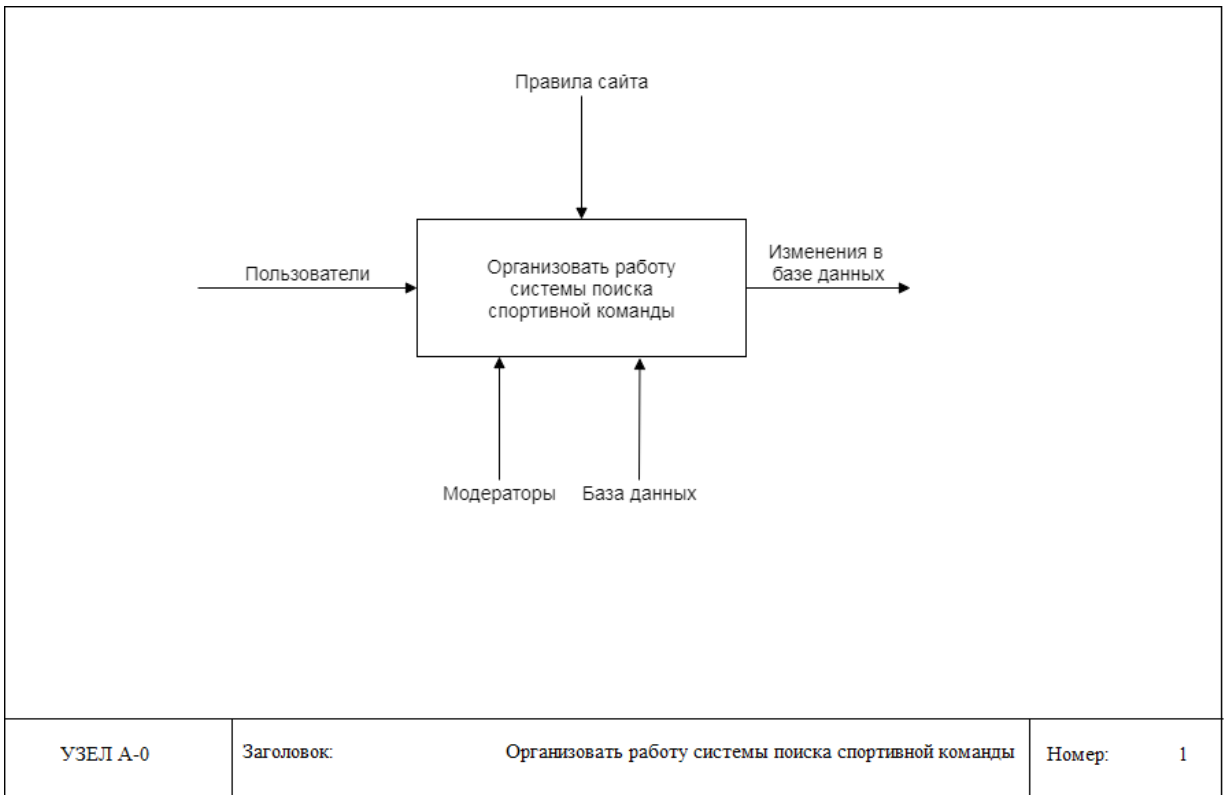

Рисунок 8. Контекстная IDEF0 диаграмма.

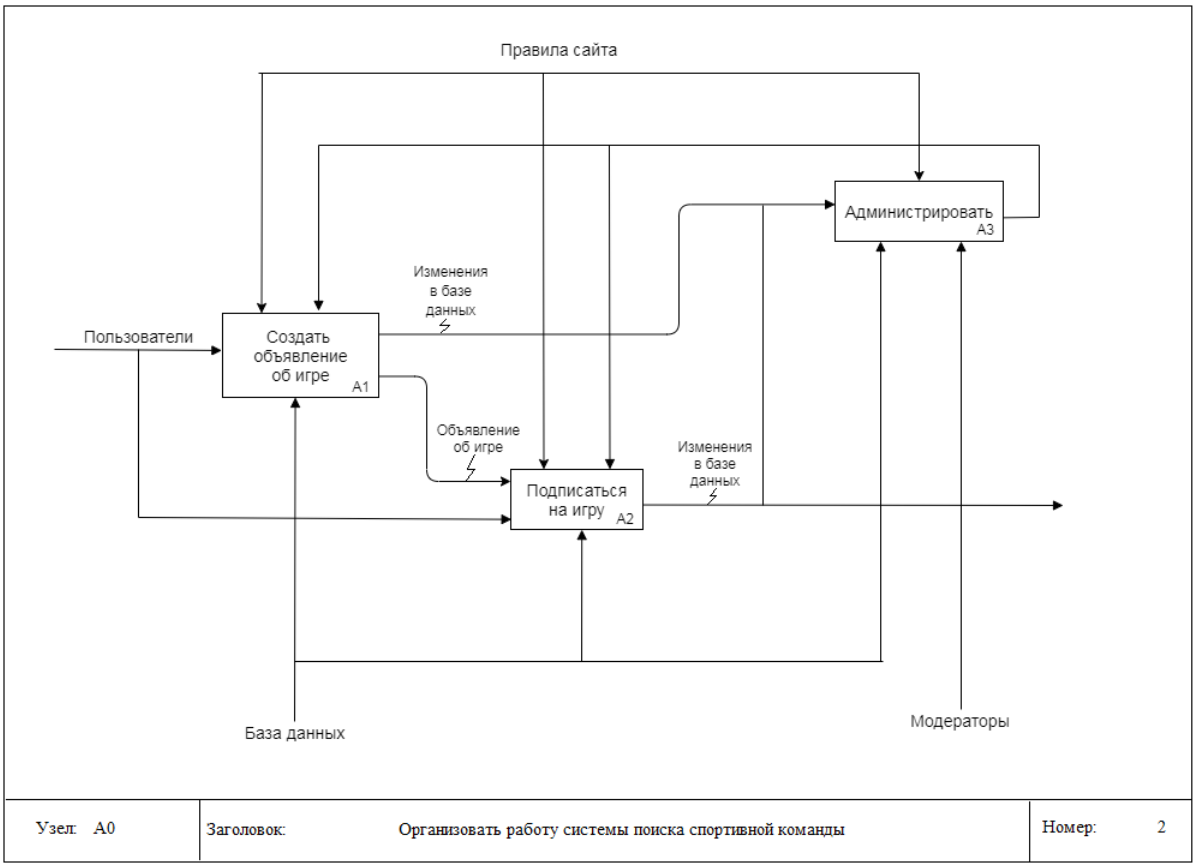

Рисунок 9. IDEF0 диаграмма работы системы.

2.5.2 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.

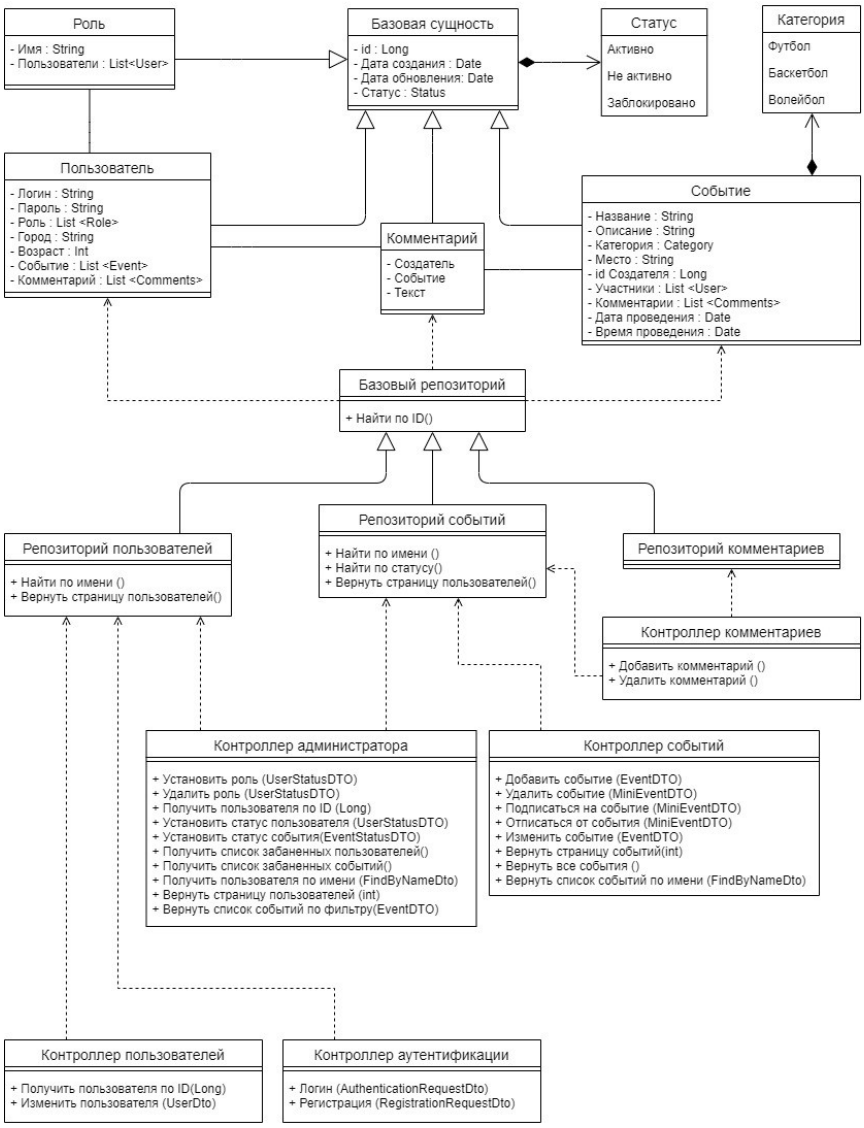

Рисунок 10 – Диаграмма классов

– 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.

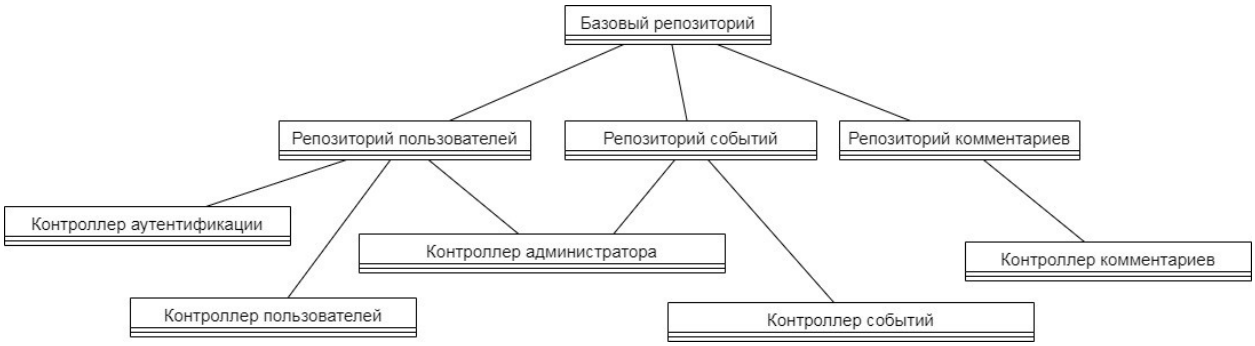

Рисунок 11 – Диаграмма объектов

2.5.3. Диаграммы состояния

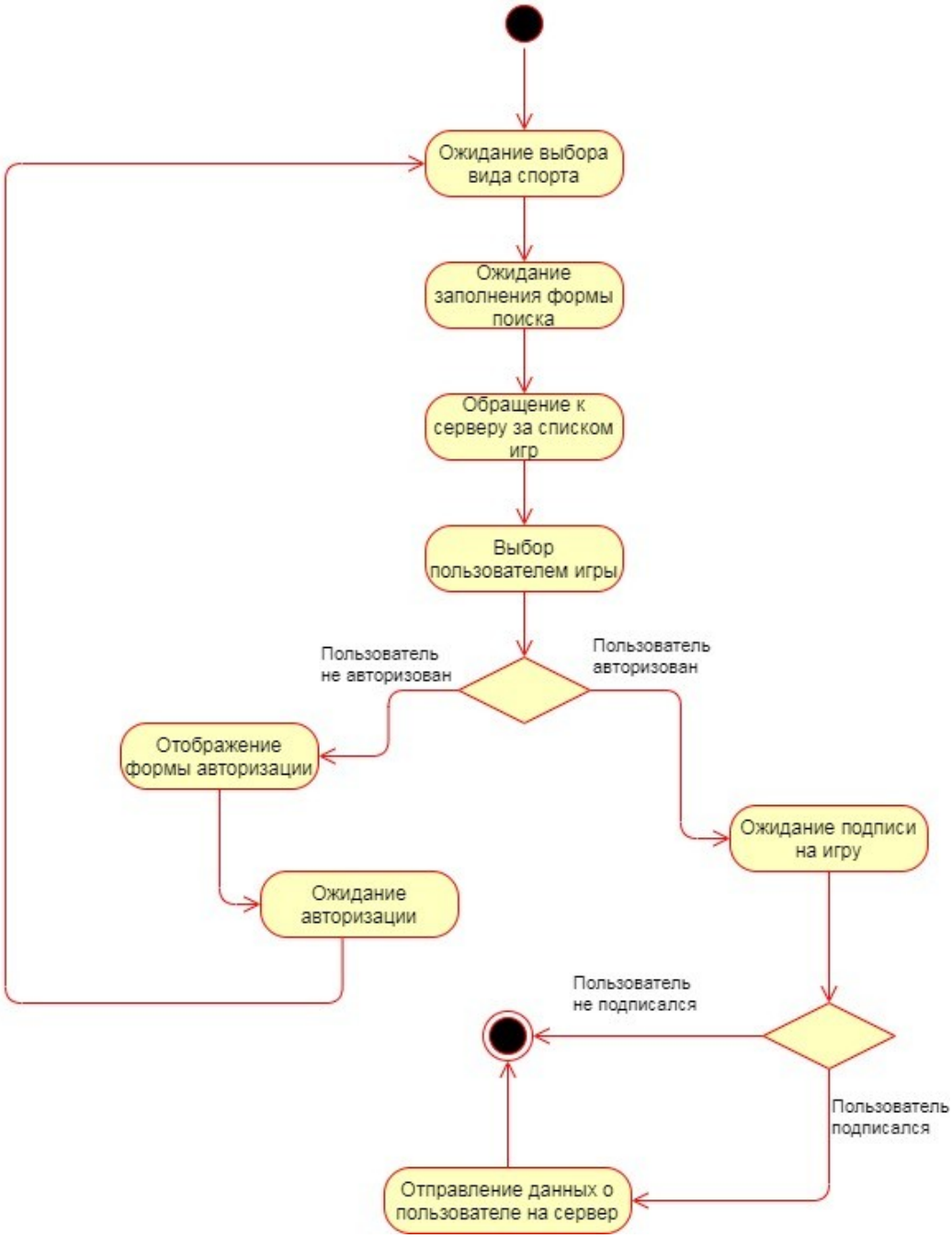

Рисунок 12. Диаграмма состояний поиска событий и подписки на них

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.

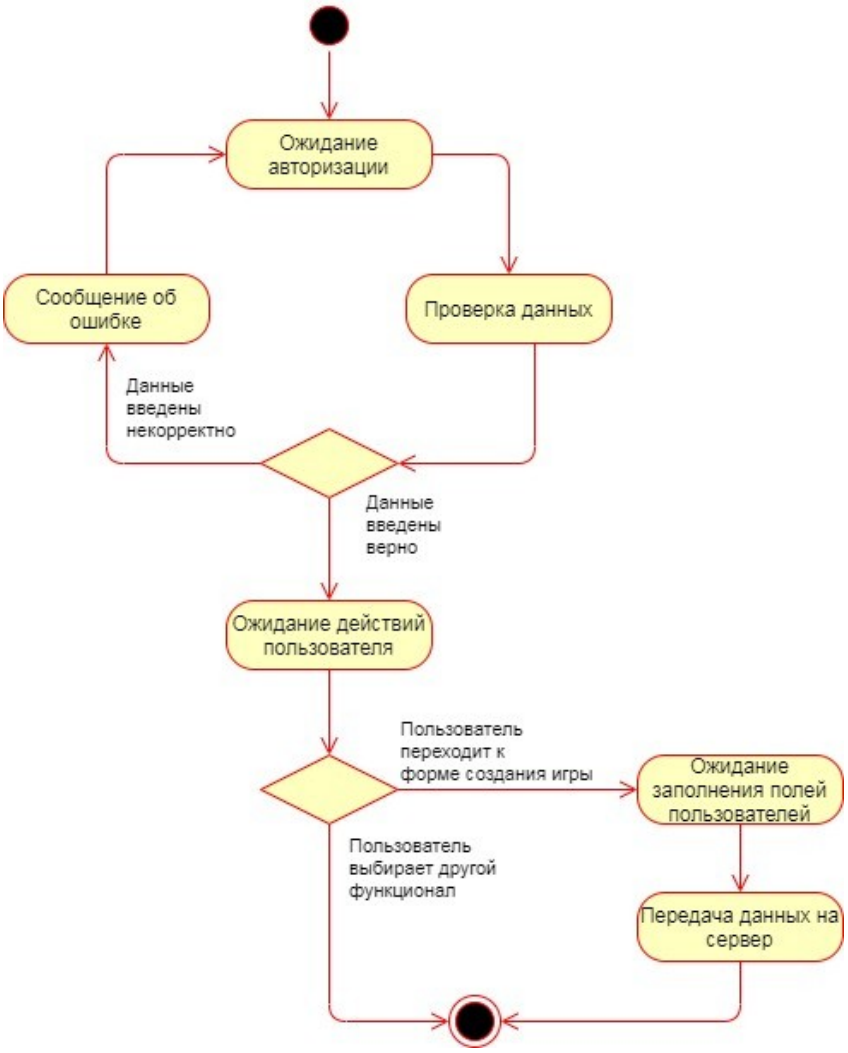

Рисунок 13. Диаграмма состояний создания нового события

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.

2.5.4. 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

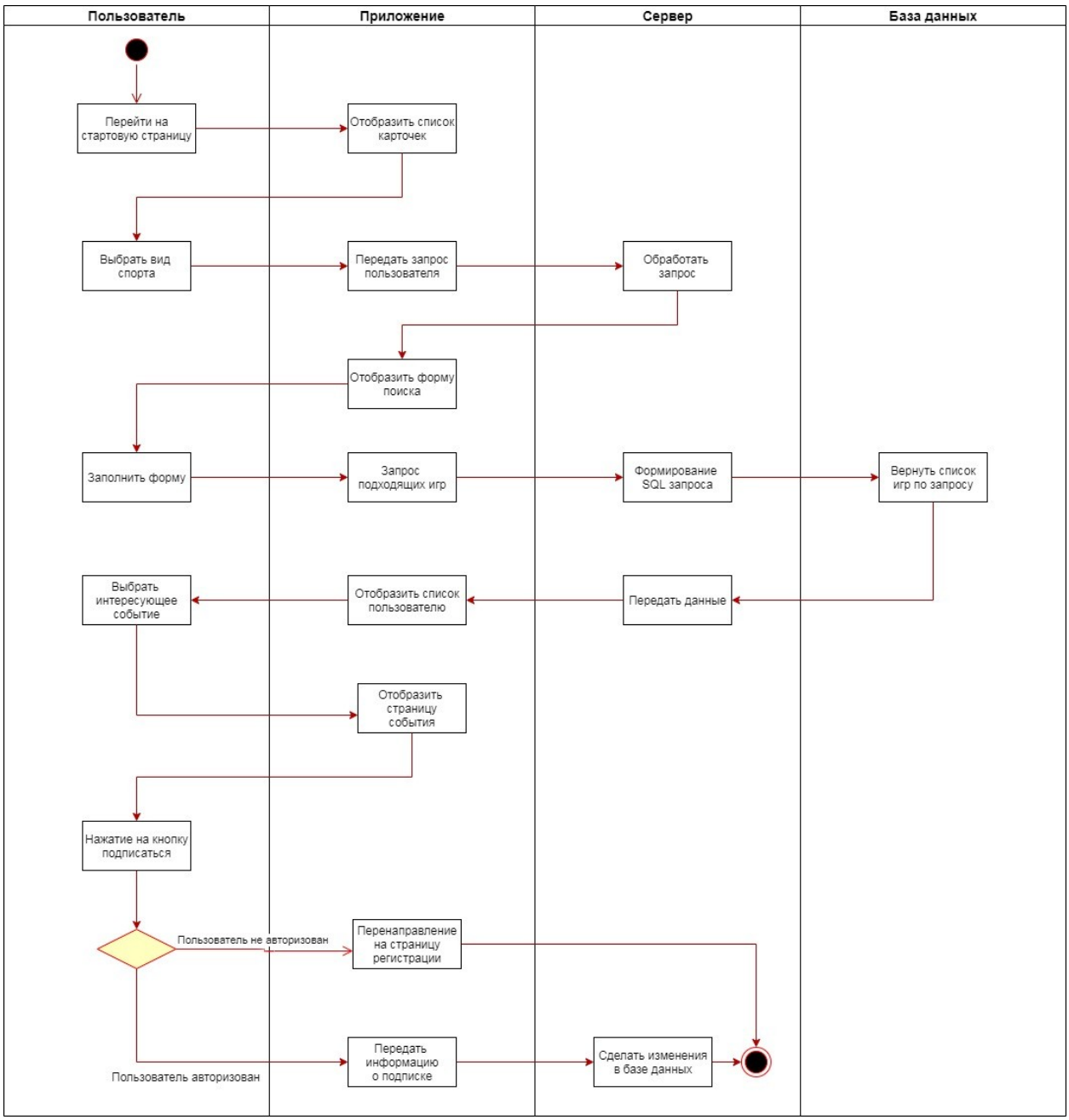

Рисунок 15 - Диаграмма активности сценария поиска событий и подписки на них

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

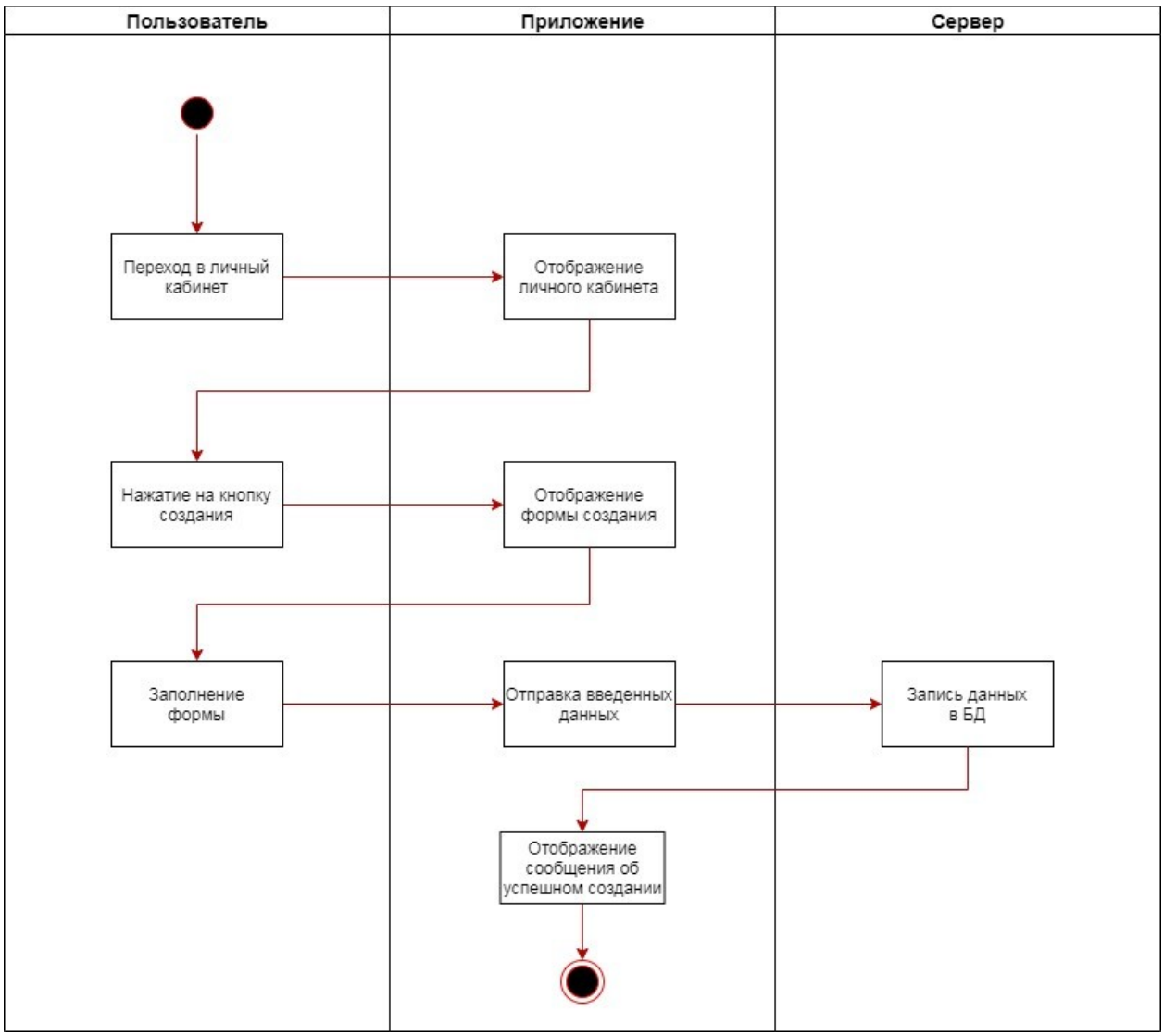

Рисунок 16 – Диаграмма активности сценария создания нового события

2.5.5. 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.

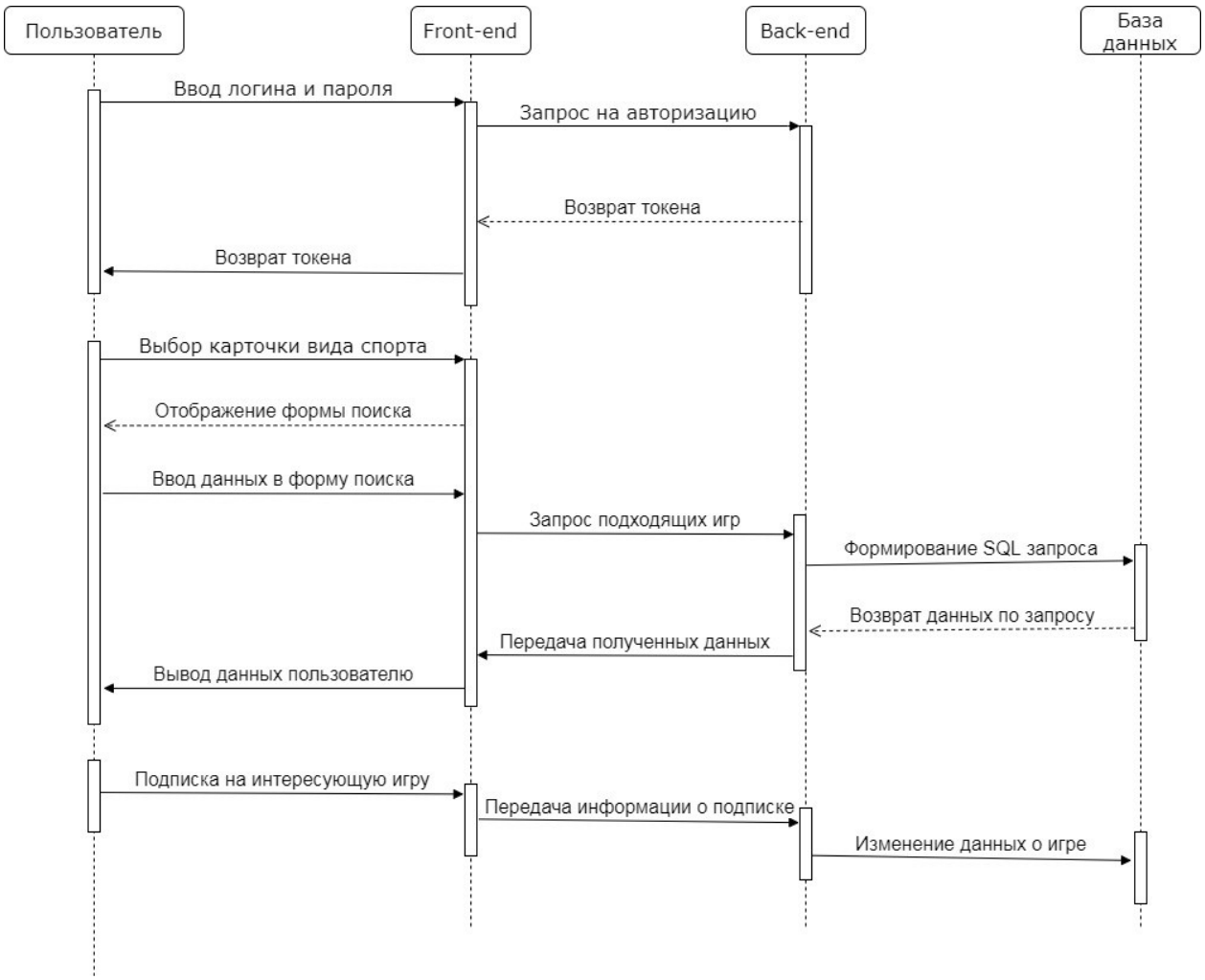

Рисунок 17 – Диаграмма последовательностей поиска и подписки на событие

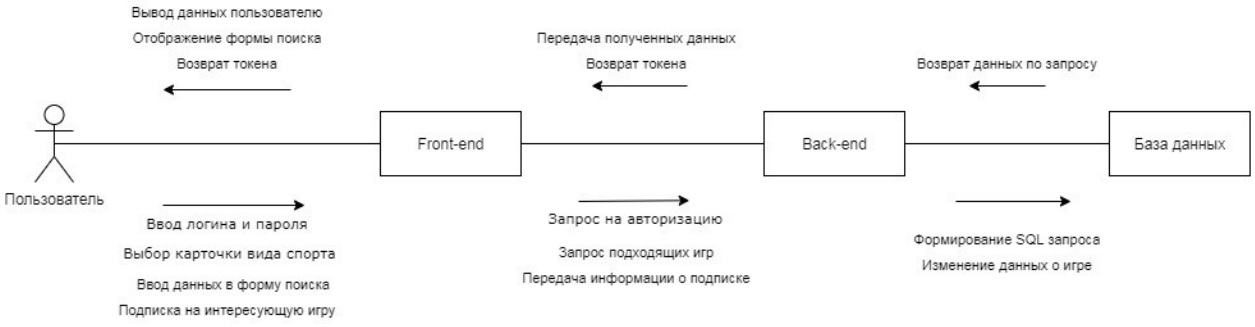

Рисунок 18 – Диаграмма взаимодействий поиска и подписки на событие

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.

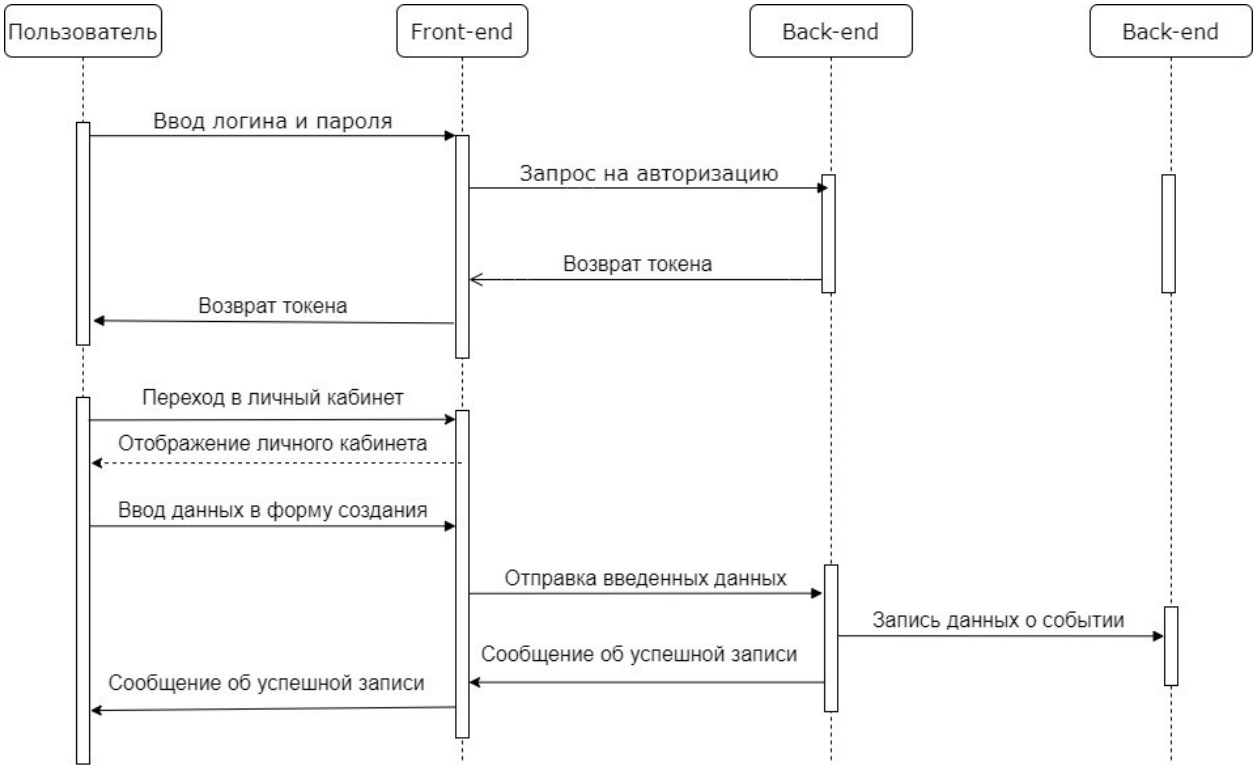

Рисунок 19 – Диаграмма последовательностей создания нового события

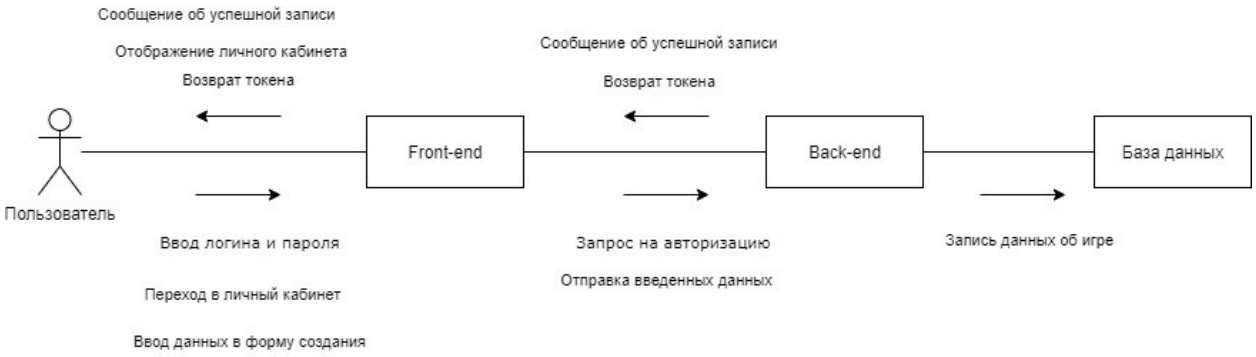

Рисунок 20 – Диаграмма взаимодействий создания нового события

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.

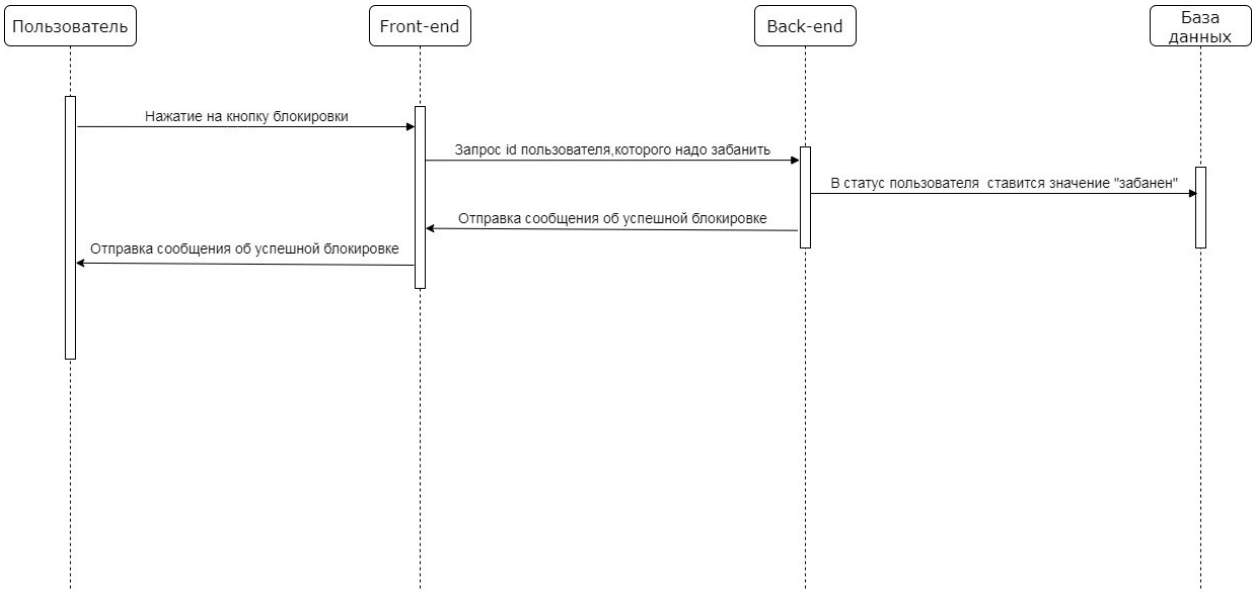

Рисунок 21 – Диаграмма последовательностей модерации системы

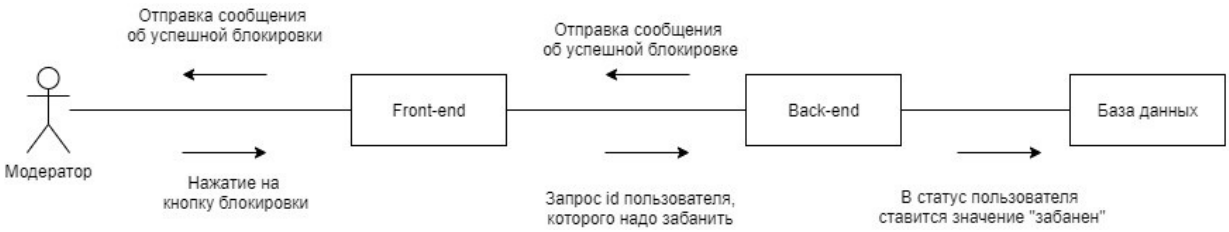

Рисунок 22 – Диаграмма взаимодействий модерации системы

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.

2.5.6. Диаграмма развертывания

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

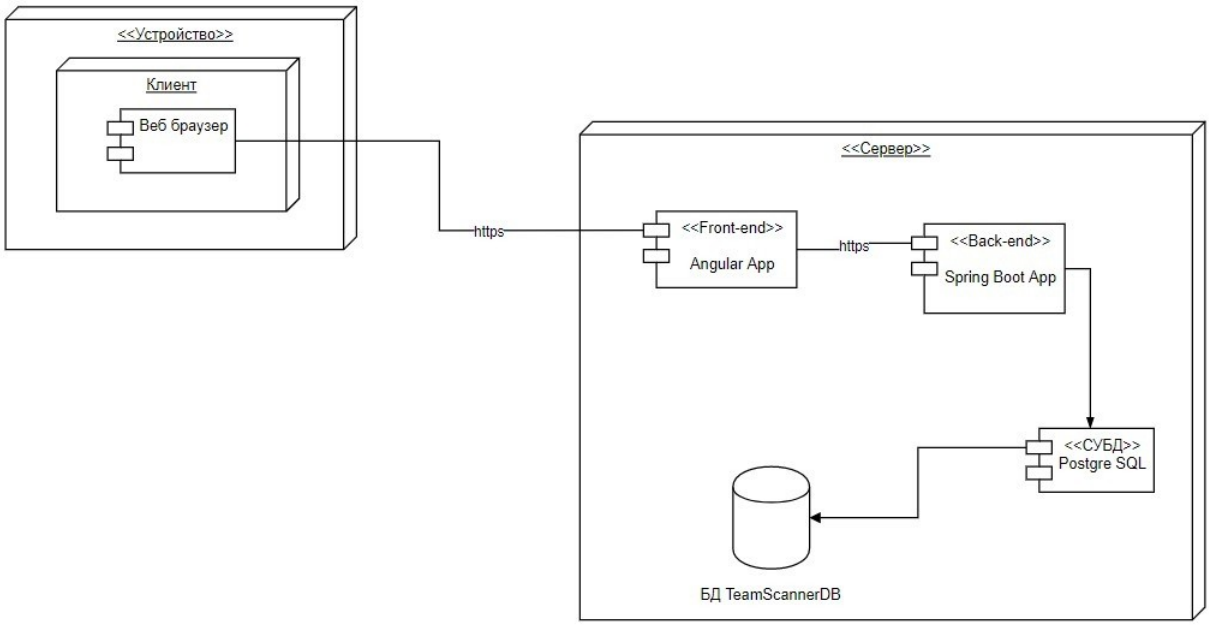

Рисунок 23 – Диаграмма развертывания

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.

2.6. 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

4. Интерфейс

5. Тестирование

Заключение

Список использованных источников

Приложения