

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Воронежский государственный технический университет»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет Информационных технологий и компьютерной безопасности  
Кафедра: Систем автоматизированного проектирования и информационных систем  
Направление подготовки (специальность): 09.04.02 Информационные системы и технологии  
(код и наименование направления подготовки)  
Профиль/программа/направленность Программа «Разработка Web-ориентированных информационных систем»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**  
магистерская диссертация

Сушилов Антон Александрович

(фамилия, имя, отчество обучающегося)

Тема: Разработка веб-ориентированной информационной системы по организации спортивных соревнований

**Состав выпускной квалификационной работы:**

Расчетно-пояснительная записка на 83 страницах

Графическая часть на \_\_\_\_\_ листах

**Расчетно-пояснительная записка к выпускной квалификационной работе:**

И.о. заведующего кафедрой

Гусев П. Ю.

(подпись)

(инициалы, фамилия)

Руководитель

Рындин Н. А.

(подпись)

(инициалы, фамилия)

**Консультанты:**

по

(

)

(наименование раздела, подпись)

(инициалы, фамилия)

по

(

)

(наименование раздела, подпись)

(инициалы, фамилия)

по

(

)

(наименование раздела, подпись)

(инициалы, фамилия)

по

(

)

(наименование раздела, подпись)

(инициалы, фамилия)

по

(

)

(наименование раздела, подпись)

(инициалы, фамилия)

Обучающийся

Сушилов А. А.

(подпись)

(инициалы, фамилия)

«20»

12

2023 г.

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет Информационных технологий и компьютерной безопасности  
Кафедра Систем автоматизированного проектирования и информационных систем  
Направление подготовки/специальность 09.04.02 Информационные системы и технологии  
(код, наименование)  
Профиль/программа/направленность Программа «Разработка Web-ориентированных  
информационных систем»

Утверждаю

20.12.2023

(дата)

Зав.кафедрой

(подпись)

**ЗАДАНИЕ**  
**ПО ПОДГОТОВКЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**  
магистерская диссертация

Обучающемуся Сушилову Антону Александровичу

фамилия имя, отчество

1. Тема: Разработка веб-ориентированной информационной системы по организации  
спортивных соревнований

утверждена приказом по университету

2. Срок сдачи обучающимся выпускной квалификационной работы:

3. Краткое содержание выпускной квалификационной работы: Во введении обосновывается актуальность выбранной темы исследования, определяется объект, предмет и цель исследования. В первой части работы проводится анализ популярных веб-сервисов для организации спортивных мероприятий, определяются основные модули разрабатываемого сервиса. Во второй части рассматриваются технологии составления расписаний соревнований, в том числе круговая система и олимпийская систем. В третьей части производится проектирование и разработка системы, в частности функциональные характеристики системы и общая архитектура приложения

4. Перечень графического материала (с точным указанием обязательных чертежей по разделам:

5. Консультанты (с указанием относящихся к ним разделов)

График выполнения выпускной квалификационной работы

№ п/п	Наименование разделов выпускной квалификационной работы	Срок выполнения этапов работы
	Введение	21.08.2023
	Основной раздел	21.10.2023
	Заключение	20.12.2023

Руководитель

(подпись)

Дата выдачи задания 16.06.2023

Задание принял к исполнению 25.07.2023

(дата)

Подпись обучающегося



**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Воронежский государственный технический университет»**

**АННОТАЦИЯ  
выпускной квалификационной работы**

**Обучающегося Сушилова Антона Александровича**

**Тема ВКР: Разработка веб-ориентированной информационной системы по организации спортивных соревнований**


**Характеристика выпускной квалификационной работы**

Данная выпускная квалификационная работа состоит из введения, трех разделов, заключения, списка используемых источников. Содержание работы изложено на 83 страницах, иллюстрировано 43 рисунками и 8 таблицами. Список использованных источников насчитывает 15 используемых источников.

Целью выпускной квалификационной работы является разработка веб-ориентированной информационной системы по организации спортивных соревнований.

В выпускной квалификационной работе рассмотрены следующие вопросы: обоснование выбора темы выпускной квалификационной работы, анализ проблемной области, сравнение и анализ существующих решений, выбор среды разработки, обоснование выбора среды разработки, разработка веб-приложения.

Итогом выполнения выпускной квалификационной работы стало разработанное приложение для организации спортивных соревнований.

Автор ВКР  Ф.И.О. (полностью)  
(подпись)

**Ministry of Science and Higher Education of the Russian Federation  
Federal State Budgetary Educational Institution higher education  
Voronezh State Technical University**

**ANNOTATION  
final qualifying work**

**Of the training (s) Sushilov Anton Aleksandrovich**

**Final qualifying work topic: Development of a web-based information system for organizing sports competitions.**


**Characteristics of the final qualifying work**

This final qualifying work consists of an introduction, three sections, a conclusion, and a list of sources used. The contents of the work are presented on 83 pages, illustrated with 43 figures and 8 tables. The list of sources used contains 15 sources used.

The goal of the final qualifying work is to develop a web application for organizing and hosting sports competitions.

The final qualifying work covers the following issues: justification for choosing the topic of final qualifying work, analysis of the problem area, comparison and analysis of existing solutions, choice of development environment, justification for choosing a development environment, development of a web application.

The result of the final qualifying work was the developed application for organizing sports competitions.

Автор ВКР  \_\_\_\_\_ Ф.И.О. (полностью)  
(подпись)

## РЕФЕРАТ

Объем ВКР 83 с., 3 ч., 43 рис., 8 табл., 15 источников, 1 прил.

ВЕБ-СЕРВИС, ВЕБ-ПРИЛОЖЕНИЕ, REACT, DJANGO, API, ROUND-ROBIN, PYTHON, JAVASCRIPT

Целью данной выпускной квалификационной работы являлась веб-ориентированной информационной системы по организации спортивных соревнований.

В результате исследования были описаны цели и задачи исследуемой области информационных систем, а также набор инструментария, который будет применяться для достижения этих целей.

В рамках выполнения работы был проведен анализ аналогичных сервисов. Был проведен сравнительный анализ преимуществ и недостатков каждого сервиса, а также были рассмотрены основные концепции, применяемые в каждом из них.

Полученные результаты:

- Рассмотрены и разработаны ключевые аспекты информационной системы для управления сервисом организации спортивных соревнований.
- Была определена архитектура системы и выделены основные модули, необходимые для ее правильной работы.
- Разработан модуль составления расписания, использующий различные алгоритмы, в частности round-robin.
- Разработана веб-ориентированная информационная система, включающая в себя разработку backend и frontend.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	9
1 Обзор существующих решений .....	11
1.1 Анализ популярных веб-сервисов для организации спортивных мероприятий .....	11
1.2.Определение модулей сервиса.....	16
2. Технологии составления расписаний соревнований .....	23
2.1.Круговая Система [7, 8, 9] .....	23
2.2 Алгоритм планирования .....	24
2.3.Олимпийская система или плей-офф [10] .....	29
2.4.Смешанная система.....	35
2.5.Выбор технологий для разработки .....	37
3. Проектирование и реализация системы .....	44
3.1.Функциональные характеристики системы.....	44
3.2.Требования к пользовательскому интерфейсу приложения .....	44
3.3.Общая архитектура приложения [15].....	45
3.4.Реализация.....	46
ЗАКЛЮЧЕНИЕ.....	79
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	80
ПРИЛОЖЕНИЕ А .....	82

## ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

ВКР - выпускная квалификационная работа.

REST (Representational State Transfer) - это программная архитектура, которая определяет условия работы API.

API (Application programming interface) – программный интерфейс приложения.

БД - база данных.

СУБД - система управления базами данных.

HTML (от англ. HyperText Markup Language) – стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере.

OAuth - стандарт (схема) авторизации, обеспечивающий предоставление третьей стороне ограниченного доступа к защищённым ресурсам пользователя без передачи ей (третьей стороне) логина и пароля



## ВВЕДЕНИЕ

Спорт играет важную роль в нашей жизни, объединяя людей, способствуя здоровому образу жизни и развитию командного духа. Организация спортивных соревнований требует тщательной планировки, координации и управления различными аспектами, чтобы обеспечить их эффективное проведение. В связи с этим возникает потребность в эффективной информационной системе, которая способна упростить и автоматизировать процессы организации спортивных соревнований.

Целью данной дипломной работы является разработка информационной системы для управления сервисом. Основным преимуществом разработанной системы является использование современных технологий веб-разработки, таких как веб-фреймворк React [1] для клиентской стороны и фреймворк разработки Django [2] для серверной стороны. Комбинация этих технологий позволяет создать полноценную и интуитивно понятную информационную систему, которая удовлетворит потребности организаторов и участников спортивных соревнований.

В рамках разработки информационной системы будут рассмотрены следующие задачи:

1) Разработка пользовательского интерфейса с использованием веб-фреймворка React. Это позволит создать привлекательный и удобный интерфейс, который позволит пользователям легко ориентироваться и выполнять необходимые действия в системе.

2) Разработка серверной части приложения на базе фреймворка Django. Здесь будут реализованы бизнес-логика, функциональность и взаимодействие с базой данных, необходимые для управления спортивными мероприятиями.

3) Взаимодействие между клиентской и серверной частями системы. Система будет основана на RESTful API [3], что обеспечит обмен данными между клиентом и сервером и позволит эффективно обновлять информацию в режиме реального времени.

В ходе работы будут рассмотрены современные методы разработки веб-приложений и подходы для организации спортивных соревнований. Будет проведен

анализ существующих решений в данной области, а также реализована разработанная информационная система и проведено ее тестирование.

В результате выполнения данной работы ожидается получить полнофункциональную информационную систему, которая Система позволит участникам и организаторам эффективно обмениваться информацией, управлять расписанием соревнований, регистрировать участников и получать актуальную информацию о проводимых событиях.

В следующих разделах работы будет представлен детальный обзор существующих решений в области управления спортивными мероприятиями, выделены требования к разрабатываемой системе, описана разработка и реализация приложения, а также выполнено тестирование системы. В заключении будет приведен обзор полученных результатов и указаны возможные направления дальнейшего развития системы.

## 1 Обзор существующих решений

### 1.1 Анализ популярных веб-сервисов для организации спортивных мероприятий

#### 1.1.1 Challonge [4]

Challonge - популярная платформа для организации спортивных мероприятий и соревнований. Ее основные и ключевые особенности включают в себя:

1) Простота использования: Challonge предлагает очень простой и понятный интерфейс, что делает его доступным даже для новичков в организации турниров. Создание и управление турнирами становится легким и быстрым процессом. Все нужные функции находятся на виду, и пользователи могут легко настраивать правила, добавлять участников, а также следить за прогрессом соревнования.

2) Расширенные функциональные возможности: Challonge предоставляет множество полезных инструментов для организации турниров. Организаторы имеют возможность выбирать различные форматы турниров, устанавливать правила, настраивать время и дату матчей, отслеживать прогресс участников, а также автоматически генерировать сетку турнира. Это упрощает процесс организации и облегчает работу организаторам.

3) Интеграция с другими платформами: Challonge имеет интеграцию с популярными платформами для стриминга игр, такими как Twitch. Это позволяет организаторам турниров привлекать больше зрителей к своим мероприятиям и увеличивать их видимость. Стриминг матчей и трансляции соревнований через платформу Twitch обеспечивают более широкую аудиторию и помогают привлечь спонсоров и рекламодателей.

4) Гибкая настройка турниров: Challonge предлагает организаторам возможность гибко настраивать правила и структуру турниров. Вы можете создавать разные типы турниров, выбирать систему сетки (одноэтапную или двухэтапную), настраивать раунды, время и дату матчей, а также устанавливать дополнительные правила в зависимости от ваших потребностей и требований.

5) Бесплатная версия: Challonge предлагает бесплатную версию платформы, которая включает в себя основные функции и подходит для мероприятий с небольшим числом участников. Это делает Challonge доступным для широкой аудитории и позволяет многим организаторам начать свою деятельность без затрат. Платная версия Challonge также предоставляет расширенный набор возможностей, которые могут быть полезны при организации крупных мероприятий или для организаторов, которые нуждаются в дополнительных функциях и инструментах. В платной версии есть дополнительные функции, такие как кастомизация веб-страницы турнира, расширенная статистика, уведомления об обновлениях и прочие дополнительные возможности, которые позволяют более полноценно настроить и управлять турнирами.

Challonge является хорошим выбором для организации спортивных мероприятий и турниров. Его простота использования, функциональность и бесплатная версия делают платформу привлекательной для широкого спектра организаторов.

Однако важно учесть недостаток веб-сервиса Challonge. Недостатком является ограничение количества участников. Challonge имеет ограничение на количество участников, что может создать ограничения для организаторов, планирующих проведение крупных соревнований или мероприятий с большим числом участников.

### 1.1.2 Tournament Scheduler [5]

Tournament Scheduler — это веб-сервис, предоставляющий организаторам спортивных мероприятий возможность удобно планировать и организовывать турниры. Рассмотрим его основные и ключевые особенности подробнее:

1) Удобный интерфейс: Tournament Scheduler имеет интуитивно понятный и легкий в использовании интерфейс. Организаторы турниров могут легко создавать новые турниры, настраивать их параметры и правила, а также добавлять участников без лишних сложностей. Удобная навигация и простота работы делают Tournament Scheduler доступным для организаторов различного уровня опыта.

2) Гибкие возможности настройки: С помощью Tournament Scheduler организаторы могут гибко настраивать параметры турниров в соответствии с применимыми правилами и требованиями. Они имеют возможность выбрать желаемый формат турнира, установить правила игры, определить количество участников и раундов, а также назначить время и даты матчей. Это позволяет организаторам полностью адаптировать турнир к своим потребностям и требованиям.

3) Автоматизация составления расписания: Tournament Scheduler предоставляет функцию автоматического составления расписания матчей. Организаторы могут ввести необходимые ограничения и предпочтения, и инструмент будет генерировать оптимальное расписание, чтобы минимизировать конфликты и обеспечить удобство для участников. Это сокращает время и усилия, затрачиваемые на разработку расписания вручную.

4) Расширенные функциональные возможности: Tournament Scheduler предлагает ряд расширенных функций, чтобы упростить организацию и управление турнирами. Организаторы могут создавать несколько дивизионов или категорий, учитывать победы и поражения для автоматического определения лидеров, выводить результаты и таблицы в понятном формате, а также добавлять комментарии и уведомления для участников. Это помогает обеспечить более полное и информативное управление турниром.

5) Расширение и экспорт данных: Tournament Scheduler позволяет импортировать и экспортировать данные, что облегчает интеграцию с другими инструментами. Организаторы могут импортировать данные участников из файлов Excel или CSV, а также экспортировать результаты и расписание в форматах PDF, Excel или HTML. Это дает возможность обмена данными и совместного использования информации с другими системами или участниками турнира.

В целом, Tournament Scheduler предлагает широкий набор функций и возможностей для организации и планирования спортивных мероприятий и турниров. Удобный интерфейс, гибкие настройки турниров, автоматическое составление расписания, расширенные функции управления и возможность экспорта данных делают его привлекательным инструментом для организаторов турниров.



Однако, следует учесть его ограничение в использовании на мобильных устройствах. Несмотря на то, что Tournament Scheduler имеет удобный интерфейс для настольных компьютеров, он может быть не совсем удобным в использовании на мобильных устройствах. Интерфейс может не оптимизироваться для мобильных экранов и использования сенсорного ввода, что может создать неудобства для организаторов, которые хотят работать в пути.

### 1.1.3 GoodGame [6]

Веб-сервис GoodGame предоставляет широкие возможности для организации и создания сеток для спортивных мероприятий, таких как турниры или кубки. Его основные особенности и функциональность делают его привлекательным инструментом для организаторов турниров и участников. Ниже мы рассмотрим его особенности более подробно:

1) Создание пользовательских сеток турниров: Этот веб-сервис позволяет пользователям создавать собственные сетки для турниров и кубков. Пользователь может выбрать подходящий формат сетки, как, например, одиночная элиминация, двойная элиминация, групповая стадия с плей-офф и многое другое. Это позволяет организаторам адаптировать формат турнира к нуждам и требованиям конкретного мероприятия.

2) Удобный интерфейс и простота использования: GoodGame имеет простой и интуитивно понятный интерфейс, что делает его легким в использовании для организаторов и участников турниров. Пользователи могут легко заполнять информацию о командах или участниках и настраивать параметры сетки. Это позволяет быстро и удобно создавать сетки для турниров.

3) Возможность делиться и экспортировать сетки: GoodGame позволяет пользователям легко делиться сетками турниров с другими участниками или зрителями. Организаторы могут создавать публичные ссылки или коды доступа, чтобы поделиться сеткой с другими. Кроме того, сетки могут быть экспортированы в

различных форматах, таких как PDF или изображение, для использования на других платформах или публикации.

4) Комментарии и уведомления: Веб-сервис предлагает функциональность комментирования и уведомлений, что обеспечивает более активную коммуникацию между организаторами и участниками турниров. Участники могут оставлять комментарии, обсуждать результаты матчей или задавать вопросы. Уведомления помогают пользователям быть в курсе последних обновлений, таких как следующий матч или изменения в расписании.

GoodGame предлагает удобный и простой способ создания сеток для турниров и кубков. Его удобный интерфейс, возможность делиться и экспортировать сетки, а также комментарии и уведомления делают его полезным инструментом для организаторов и участников спортивных мероприятий.

Однако, несмотря на свои преимущества, у GoodGame есть один преимущественный недостаток, который стоит учесть. Ограниченные возможности настройки. Веб-сервис может иметь ограниченный набор возможностей для настройки сеток турнира. Некоторые более специализированные форматы или опции настройки могут быть недоступны, что может быть недостаточно для организаторов турниров с особыми требованиями или форматами.

## 1.2. Определение модулей сервиса

Основные модули разрабатываемой системы могут включать:

- 1) Модуль аутентификации и авторизации: обеспечение безопасного входа для организаторов и участников.
- 2) Модуль управления соревнованиями: функции для создания, редактирования и удаления соревнований.
- 3) Модуль регистрации участников.
- 4) Модуль управления расписанием: функции для создания и управления расписанием спортивных соревнований.

Такая модульная архитектура позволит эффективно управлять процессами организации спортивных соревнований и обеспечить удобство использования для пользователей.

В результате постановки задачи будут определены требования к разрабатываемому веб-сервису и создана архитектура системы, учитывающая основные модули и функциональность. Такой подход обеспечит успешное и целенаправленное выполнение дипломной работы.

### 1.2.1 Описание основных модулей

#### 1.2.1.1 Модуль аутентификации и авторизации

Модуль аутентификации и авторизации в веб-приложении является ключевым компонентом безопасности любого веб-сайта или приложения. Он отвечает за проверку подлинности пользователей и управление доступом к ресурсам сайта.

Аутентификация — это процесс проверки подлинности пользователя, который делает запрос к веб-приложению. Целью аутентификации является установление, что пользователь, который пытается получить доступ к приложению, является тем, за кого он себя выдает. Для этого пользователь предоставляет идентифицирующую информацию, такую как логин и пароль.

Процесс аутентификации может включать следующие шаги:

- 1) Пользователь вводит свои учетные данные (логин и пароль) на странице входа в приложение.
- 2) Веб-сервер получает учетные данные и проверяет их достоверность.
- 3) Если данные являются верными, пользователь считается успешно аутентифицированным и получает доступ к приложению.

Авторизация — это процесс определения разрешений и прав доступа, которые пользователь имеет после аутентификации. Он определяет, какие действия и ресурсы в приложении доступны для каждого отдельного пользователя.

Процесс авторизации обычно включает следующие шаги:

- 1) После успешной аутентификации сервер проверяет роли и права доступа пользователя.
- 2) На основе этих данных сервер принимает решение о том, какие функции и данные пользователь может использовать.

Для реализации модуля аутентификации и авторизации можно использовать различные технологии, такие как OAuth, JWT, SAML и другие. Важно выбрать подходящую технологию для вашего проекта, учитывая его особенности и требования к безопасности.

Чаще всего система авторизации основана на ролях пользователей. Каждому пользователю назначается роль (например, администратор, модератор, обычный пользователь и т. д.) с определенными правами. При аутентификации сервер проверяет роль пользователя и позволяет доступ только к разрешенным функциям и данным.

Важной частью модуля аутентификации и авторизации является хранение и защита паролей пользователей. Хранение паролей в открытом виде недопустимо из-за возможности злоумышленников получить доступ к пользовательским аккаунтам. Поэтому пароли обычно хешируются - превращаются в непонятную последовательность символов, полученную с использованием специальных алгоритмов хеширования.

При процессе аутентификации веб-сервер сравнивает хэшированный пароль с сохраненным хэшем в базе данных. Если они совпадают, то пароль считается верным.

Важно использовать надежные алгоритмы хэширования паролей, а также добавлять соль (случайные данные), чтобы усложнить процесс взлома паролей злоумышленниками.

Вся эта функциональность модуля аутентификации и авторизации обеспечивает безопасность и контроль доступа к веб-приложению, позволяет управлять пользователями и их правами, а также защищает конфиденциальные данные от несанкционированного доступа.

#### 1.2.1.2 Модуль управления соревнованиями

Модуль управления соревнованиями — это часть веб-приложения, которая предоставляет администраторам или определенным пользователям функциональные возможности для создания, редактирования и удаления соревнований. Этот модуль обычно включает следующие функции:

##### 1) Создание соревнования:

Функция создания соревнования позволяет администратору или соответствующему пользователю создать новое соревнование и заполнить необходимую информацию о нем. В форме создания соревнования обычно присутствуют следующие поля:

- Название соревнования: имя, которое будет идентифицировать соревнование.

- Дата и время начала и окончания соревнования: указывается время и дата начала и окончания соревнования.

- Место проведения: информация о месте или площадке, где будет проходить соревнование.

- Правила и описание: детальное описание правил и особенностей соревнования.



- Дополнительные параметры: дополнительные настройки или параметры, специфичные для данного соревнования.

После заполнения всех необходимых данных и подтверждения, соревнование создается и становится доступным для участников или зрителей.

## 2) Редактирование соревнования:

Функция редактирования соревнования позволяет администратору или пользователю с соответствующими правами вносить изменения в информацию о существующем соревновании. Это может быть полезно, если есть необходимость обновить дату, время или другие детали соревнования.

Возможности редактирования соревнования могут включать:

- Изменение названия соревнования.
- Изменение даты и времени начала и окончания соревнования.
- Обновление информации о месте проведения или добавление новых деталей.
- Изменение правил и описания соревнования.
- Обновление дополнительных параметров или настроек.

После внесения изменений, обновленная информация о соревновании сохраняется и отображается для пользователей.

## 3) Удаление соревнования:

Функция удаления соревнования позволяет администратору или пользователю с соответствующими правами удалить соревнование из системы. При удалении соревнования все связанные с ним данные, такие как результаты, расписания, участники и другая информация, также могут быть удалены или архивированы.

Важно отметить, что удаление соревнования может быть необратимым действием, поэтому обычно требуется подтверждение удаления для предотвращения случайных удалений.

После удаления соревнования, связанные данные становятся недоступными для пользователя, и соревнование больше не отображается в списке активных соревнований.

Функциональность модуля управления соревнованиями обеспечивает удобство и эффективность в организации и управлении соревнованиями. Создание,

редактирование и удаление соревнований позволяют администраторам и пользователям поддерживать актуальную информацию о соревнованиях и настраивать их параметры в соответствии с требованиями и изменениями.

### 1.2.1.3 Модуль регистрации участников

Модуль регистрации участников соревнований в веб-приложении — это ключевой компонент системы, предназначенный для управления процессом регистрации и участия в соревнованиях. Этот модуль обеспечивает участникам и организаторам соревнований удобные средства для регистрации, сбора необходимой информации и обеспечивает эффективное управление данными участников.

Основные характеристики и функциональность модуля регистрации участников соревнований в веб-приложении включают в себя:

1) Регистрация участников: Модуль предоставляет возможность участникам соревнований зарегистрироваться, заполнив необходимую информацию о себе, такую как имя, контактные данные, данные о платеже и другие.

2) Управление доступностью мест: Модуль позволяет устанавливать ограничения на количество доступных мест в соревнованиях и предоставляет участникам информацию о доступности мест.

3) Уведомления и подтверждения: Модуль отправляет уведомления участникам о статусе и подтверждении их регистрации. Также он предоставляет возможность администраторам соревнований управлять статусами регистрации.

4) Административный интерфейс: Администраторы соревнований могут просматривать, редактировать и управлять данными зарегистрированных участников через специальный административный интерфейс.

5) Документация и инструкции: Предоставление документации и инструкций как для участников, так и для администраторов, чтобы облегчить понимание и использование модуля.

б) Интеграция с другими модулями: Модуль регистрации должен интегрироваться с другими компонентами веб-приложения, такими как модуль расписания, уведомлений и результатов.

Этот модуль играет важную роль в соревновательной среде, облегчая организацию и участие в спортивных мероприятиях. Он повышает эффективность процесса регистрации, управления данными и взаимодействия с участниками, способствуя успешному проведению соревнований.

#### 1.2.1.4 Модуль управления расписанием

Модуль управления расписанием соревнований является важной частью веб-приложения для организации спортивных мероприятий. Он отвечает за создание и управление расписанием соревнований, а также предоставляет функцию выбора метода составления расписания, включая Круговую систему, Двойную круговую систему, олимпийскую систему и смешанную систему.

Описание модуля управления расписанием:

1) Создание расписания: Пользователь, являющийся организатором соревнования, может использовать модуль управления расписанием для создания расписания соревнований. Он может указать дату и время проведения каждого матча или игры, а также определить условия и правила проведения соревнования.

2) Выбор метода составления расписания: Модуль управления расписанием предоставляет пользователю возможность выбрать метод составления расписания из нескольких доступных вариантов. Эти варианты включают Круговую систему, Двойную круговую систему, олимпийскую систему и смешанную систему:

– Круговая система: Этот метод состоит из раундов, где каждая команда или участник встречается с каждым другим участником по очереди. Это позволяет обеспечить равные условия для всех участников и справедливое решение о победителях. В модуле можно задать количество команд или участников, и система автоматически сгенерирует расписание матчей между ними.

– Двойная круговая система: В этом методе каждая команда или участник соревнуется с каждой другой командой или участником дважды - один раз дома, другой раз на чужом поле. Это помогает сбалансировать условия и проверить силу команды или участника тщательнее. Модуль позволяет выбирать количество команд или участников и автоматически составляет пары для каждого тура соревнований.

– Олимпийская система: В этом методе команды или участники соревнуются друг с другом в различных турах, применяя систему единоборств. Участники с высокими результатами двигаются дальше в соревнованиях, а те, кто проигрывает, выбывают из турнира. Модуль позволяет указать количество команд или участников и автоматически генерирует сценарии соревнований для каждого тура.

– Смешанная система: Этот метод комбинирует различные методы для создания разнообразных расписаний соревнований. Например, первый тур может быть проведен по Круговой системе, а затем олимпийская система может использоваться для дальнейших этапов. Модуль позволяет настроить приоритеты методов и генерирует соответствующие расписания на основе указанных параметров:

1) Автоматическое составление расписания: Модуль управления расписанием также может предложить возможность автоматического составления расписания на основе выбранного метода. С помощью алгоритмов и логики веб-приложения можно разработать автоматический генератор расписания, который учитывает все условия и ограничения выбранного метода составления расписания.

2) Модификация расписания: Пользователь может внести изменения в расписание в любое время. Модуль управления расписанием предоставляет возможность добавлять новые матчи, изменять время и даты проведения, а также менять команды или участников, если это необходимо.

Модуль управления расписанием позволяет организаторам спортивных мероприятий эффективно создавать и управлять расписанием, выбирая подходящий метод составления расписания в соответствии с требованиями и целями соревнования. Участники соревнований смогут легко получить доступ к актуальному расписанию и быть в курсе времени и места проведения своих матчей.

## 2. Технологии составления расписаний соревнований

### 2.1. Круговая Система [7, 8, 9]

В круговой системе, которая широко применяется в спортивных соревнованиях, каждый участник турнира соревнуется с каждым из остальных участников в рамках определенного тура или раунда. Эта система популярна в игровых видах спорта, таких как футбол, волейбол, баскетбол и киберспорт, особенно в национальных чемпионатах и при отборочных турнирах к чемпионатам мира или континентов. Она считается наиболее справедливой, поскольку каждый участник имеет возможность сыграть со всеми другими участниками. Однако для определения мест требуется проведение большего числа игр, чем при использовании других турнирных систем.

В круговой системе каждый участник соревнования сталкивается с каждым из остальных участников, и это позволяет более четко определить уровень мастерства и способности каждого участника. Также важно отметить, что в круговой системе требуется провести больше матчей, чтобы определить победителей и распределить места среди участников. Это может потребовать значительных усилий в планировании и проведении турнира, особенно если в нем принимает участие большое количество команд или игроков.

Однако, несмотря на дополнительные трудности, круговая система считается наиболее справедливой, поскольку каждая команда или игрок имеет возможность сыграть с каждым другим и показать свои способности в полной мере. Это особенно важно в крупных турнирах, где команды или игроки из разных регионов или стран должны быть справедливо оценены и сравнены друг с другом.

Для определения количества игр при использовании круговой системы используется определенная формула. Эта формула позволяет вычислить количество встреч в одном круге турнира, основываясь на количестве команд.



Формула (1) выглядит следующим образом:

$$X = \frac{n(n-1)}{2}, \quad (1)$$

где  $x$  - количество встреч;

$n$  - количество команд;

$(n-1)$  - это количество игр, которые каждая команда должна провести;

2 - показатель того, что одна игра засчитывается двум командам.

Эта формула исходит из того, что каждая команда должна сыграть с каждой другой командой один раз. При использовании этой формулы можно определить, сколько игр потребуется провести в рамках одного круга турнира.

Например, участвуют 4 команды, тогда количество встреч равно:

$$\frac{4(4-1)}{2} = 6,$$

При составлении календаря игр в круговой системе применяются различные алгоритмы и методы, которые позволяют справедливо распределить матчи между командами. Один из наиболее распространенных методов — это метод "Round Robin" (круговая система).

## 2.2 Алгоритм планирования

Если число команд  $n$  в турнире является четным, то в каждом из  $n - 1$  раундов можно проводить одновременно  $\frac{n}{2}$  игр, при условии наличия достаточных ресурсов, таких как корты для теннисного турнира.

Однако, если число команд  $n$  не является четным, то количество раундов будет равно  $n$ , и общее количество игр будет равно  $\frac{n-1}{2}$ . В этом случае, в каждом раунде у одной из команд не будет игры, так как она не сможет сыграть со всеми остальными командами.

Примеры:

– При 8 командах (четное число): в каждом из  $(8-1) = 7$  раундах можно проводить по  $8/2 = 4$  игры одновременно, что обеспечивает более эффективное использование ресурсов.

– При 7 командах (нечетное число): будет 7 раундов и в каждом раунде у одной из команд не будет игры, так как она не может сыграть с самой собой.

Календарь игр по круговой системе составляется путем распределения матчей между командами в соответствии с определенными правилами. Результатом является календарь игр, который указывает дату и время проведения каждого матча между командами.

### 2.2.1 Метод круга

Метод круга является простым алгоритмом для создания расписания кругового турнира. В этом методе каждому участнику присваивается номер, а затем в первом туре формируются пары для проведения матчей или игр (Таблица 1)

Таблица 1 - Раунд 1 (1 играет с 14, 2 играет с 13 и т.д)

1	2	3	4	5	6	7
14	13	12	11	10	9	8

Далее фиксируется один из участников в первом или последнем столбце таблицы (в данном примере номер один), а остальные поворачиваются по часовой стрелке на одну позицию (Таблица 2, Таблица 3)

Таблица 2 - Раунд 2 (1 играет с 13, 14 играет с 12 и т.д)

1	14	2	3	4	5	6
13	12	11	10	9	8	7

Таблица 3 - Раунд 3 (1 играет с 12, 13 играет с 11 и т.д)

1	13	14	2	3	4	5
12	11	10	9	8	7	6

Это повторяется до тех пор, пока вы не вернетесь почти в исходное положение (Таблица 4)

Таблица 4 - Раунд 13 (1 играет с 2, 3 играет с 14 и т.д)

1	3	4	5	6	7	8
2	14	13	12	11	10	9

Чтобы убедиться в том, что данный алгоритм реализует все возможные комбинации (то есть, что все пары участников различны), когда количество участников  $n$  является четным числом, мы рассуждаем следующим образом.

Во-первых, алгоритм очевидно реализует каждую пару участников, если один из них равен 1 (неподвижный участник).

Затем, для пар участников, не равных 1, пусть их расстояние будет числом  $k < \frac{n}{2}$ , которое показывает, сколько раз нужно выполнить поворот, чтобы один участник оказался на позиции другого.

В данном примере ( $n = 14$ ) участник 2 имеет расстояние 1 до 3 и 14, а расстояние 6 до 8 и 9.

В каждом раунде, на не крайней позиции (не включая 1), может находиться только участник с определенным расстоянием. В первом раунде примера, на второй позиции участник 2 играет против 13, их расстояние равно 2. Во втором раунде эта позиция занимает участники 14 и 12, которые также имеют расстояние 2, и так далее. Точно так же, следующая позиция (3 против 12 в первом раунде, 2 против 11 во втором раунде и т. д.) может содержать только участников с расстоянием 4.

Для каждого значения  $k < \frac{n}{2}$ , существует ровно  $n-1$  пара расстояния  $k$ . Существует  $n-1$  раунд, и все они реализуют пару расстояния  $k$  на одной и той же позиции. Очевидно, что эти пары попарно различны. Вывод заключается в том, что каждая пара расстояния  $k$  реализуется.

Это справедливо для каждого значения  $k$ , следовательно, каждая пара реализуется.

Если количество участников нечетное, можно добавить фиктивного участника, который в данном раунде не будет играть и у его оппонента в этом раунде не будет игры. Комбинации участников могут быть вычислены так, как если бы фиктивный участник был обычным игроком.

### 2.2.2 Таблицы Бергера

В качестве альтернативы при планировании турниров широко используются таблицы Бергера (Таблица 5), названные в честь австрийского шахматного мастера Иоганна Бергера. Бергер опубликовал таблицы жеребьевки в двух своих книгах Schach-Jahrbücher (Шахматные летописи), со ссылкой на ее изобретателя Рихарда Шурига.

Таблица 5 – Пример таблицы Бергера

Раунд 1	1-14	2-13	3-12	4-11	5-10	6-9	7-8
Раунд 2	14-8	9-7	10-6	11-5	12-4	13-3	1-2
Раунд 3	2-14	3-1	4-13	5-12	6-11	7-10	8-9
...	...						
Раунд 13	7-14	8-6	9-5	10-4	11-3	12-2	13-1

Созданное расписание имеет определенную структуру, при которой игрок с номером 14 занимает постоянную позицию, а все остальные игроки поворачиваются против часовой стрелки на  $n/2$  позиций. Это расписание можно легко сгенерировать вручную. Для составления следующего раунда последний игрок, с номером 8 в первом раунде, перемещается на первую позицию за столом. Затем следующий игрок, с номером 9, играет против игрока с номером 7, игрок с номером 10 играет против игрока с номером 6 и так далее, пока игрок с номером 1 не играет против игрока с номером 2. Это можно выразить арифметически, добавляя  $n/2$  к предыдущему раунду,

за исключением игрока с номером  $n$ . Если результат превышает  $(n-1)$ , то из этой суммы вычитается  $(n-1)$ .

Также можно представить это расписание в виде таблицы  $(n-1, n-1)$ , которая показывает, в каком раунде игроки встречаются друг с другом. Например, игрок с номером 7 играет против игрока с номером 11 в четвертом раунде. Если игрок встречается самого себя, это означает, что он получает без игры очки или играет против игрока с номером  $n$ . Все игры в раунде образуют диагональные элементы (Таблица 6)

Таблица 6 – Диагональная схема

X	2	3	4	5	6	7	8	9	10	11	12	13	1	2	3	4	5	6	7	8	9	10	11	12	13
1													1	2	3	4	5	6	7	8	9	10	11	12	13
2												1	2	3	4	5	6	7	8	9	10	11	12	13	
3											1	2	3	4	5	6	7	8	9	10	11	12	13		
4										1	2	3	4	5	6	7	8	9	10	11	12	13			
5									1	2	3	4	5	6	7	8	9	10	11	12	13				
6								1	2	3	4	5	6	7	8	9	10	11	12	13					
7							1	2	3	4	5	6	7	8	9	10	11	12	13						
8						1	2	3	4	5	6	7	8	9	10	11	12	13							
9					1	2	3	4	5	6	7	8	9	10	11	12	13								
10				1	2	3	4	5	6	7	8	9	10	11	12	13									
11			1	2	3	4	5	6	7	8	9	10	11	12	13										
12		1	2	3	4	5	6	7	8	9	10	11	12	13											
13	1	2	3	4	5	6	7	8	9	10	11	12	13												

Таблица 7– Расписание кругового турнира

x	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4	5	6	7	8	9	10	11	12	13
2	2	3	4	5	6	7	8	9	10	11	12	13	1
3	3	4	5	6	7	8	9	10	11	12	13	1	2
4	4	5	6	7	8	9	10	11	12	13	1	2	3
5	5	6	7	8	9	10	11	12	13	1	2	3	4
6	6	7	8	9	10	11	12	13	1	2	3	4	5
7	7	8	9	10	11	12	13	1	2	3	4	5	6
8	8	9	10	11	12	13	1	2	3	4	5	6	7
9	9	10	11	12	13	1	2	3	4	5	6	7	8
10	10	11	12	13	1	2	3	4	5	6	7	8	9
11	11	12	13	1	2	3	4	5	6	7	8	9	10
12	12	13	1	2	3	4	5	6	7	8	9	10	11
13	13	1	2	3	4	5	6	7	8	9	10	11	12



Для определения порядка проведения матчей можно использовать различные стратегии, например, случайное назначение или определенные правила, учитывающие силу команд или предпочтения организаторов.

Важно отметить, что в круговой системе (Таблица 7) обычно предусматривается график, в котором каждая команда имеет свой уникальный номер или идентификатор, а матчи отображены в виде пар команд или команд с указанием даты и времени проведения. Это позволяет участникам и зрителям легко ориентироваться и следить за ходом турнира.

Пример календаря игр в круговой системе (Таблица 8):

Таблица 8 – календарь игр в круговой системе

		Команда1	Команда2	Команда3	Команда4	Очки	Место
1	Команда1		5:1	2:1	0:0	7	1
2	Команда2	1:5		3:0	2:1	6	2
3	Команда3	1:2	0:3		2:0	3	3
4	Команда4	0:0	1:2	0:2		1	4

Как видно из примера, каждая команда играет с каждой другой командой ровно один раз. Также важно учесть, что при наличии большого числа команд или ограниченном количестве дней для проведения турнира может потребоваться проведение нескольких туров для завершения всех игр.

### 2.3. Олимпийская система или плей-офф [10]

Олимпийская система, также известная как плей-офф или схема Single Elimination, является форматом соревнований, который применяется во многих видов спорта. В этой системе участник турнира выбывает после первого поражения. Это может быть как в результате одной отдельной игры, так и после серии игр между двумя участниками, которая позволяет однозначно определить победителя. Главная цель олимпийской системы - выявить победителя за минимальное количество туров и создать максимальную напряженность в ходе турнира.

Интересно, что идея олимпийской системы впервые была внедрена на Олимпийских играх. Сегодня она широко применяется не только в Олимпийских соревнованиях, но и во многих других видов спорта и турниров.

Олимпийская система стимулирует команды или участников использовать все свои навыки и ресурсы, чтобы избежать поражения и преодолеть каждый этап турнира. Также она увеличивает напряжение и волнение участников и зрителей, поскольку каждая игра может стать последней для команды или игрока.

В турнире число участников должно быть вычислено по формуле (2) степенью двойки (8, 16, 32, 64, 128 и т.д.):

$$2^n \quad (2)$$

где  $n > 3$  – число кругов.

Если количество участников не соответствует этому правилу, можно использовать эту систему с добавлением фиктивных («пустых») участников, чтобы получить ближайшее правильное количество. Участник, которому выпадает игра с «пустым» участником (как правило, это спортсмен с более высоким рейтингом), автоматически побеждает в данном раунде. Обычно данная система удобна при числе участников, точно равном 8, 16 или более 20. Если числом участников является менее 8, то лучше использовать круговую систему проведения турнира (чтобы было достаточное количество матчей).

Чтобы определить число кругов или туров в турнире по олимпийской системе, мы можем использовать формулу (3) двоичный логарифм числа участников:

$$\log_2 n \quad (3)$$

где  $n$  - число участников.

Например, для 2 участников будет один круг, для 4 участников — два круга, для 8 участников — три круга, и так далее. Общее количество игр в турнире будет на единицу меньше числа участников.

Круги розыгрышей часто называются в соответствии с количеством пар участников. Например, для 1 пары участников у нас будет "финал" (и этот матч определит победителя), для 2 пар — "полуфиналы", для 4 пар — "четвертьфиналы",

для 8 пар — "одна восьмая финала", для 16 пар — "одна шестнадцатая финала" и так далее.

В каждом круге участники составляют пары и играют друг с другом. Это может быть одна игра или серия игр, в которой победитель определяется на основе очков или других правил. Важно отметить, что результат каждого тура всегда явно определен, и ничьих быть не может.

При использовании олимпийской системы составление расписания матчей осуществляется в соответствии с предварительно созданной сеткой игр, которая соответствует числу участвующих команд. Затем происходит жеребьевка, которая проводится судейской комиссией. В результате жеребьевки каждый коллектив получает определенный номер, который определяет его позицию в сетке турнира.

Сетка турнира определяет, какие команды будут играть между собой в каждом раунде. Номер каждой команды в сетке определяет ее пару или противника в конкретном этапе турнира.

Жеребьевка является важным шагом в организации турнира и делает процесс составления расписания справедливым и случайным. Она обеспечивает равные условия для всех участников и позволяет каждой команде иметь возможность встретиться с разными соперниками:

1) При 4 командах (Рисунок 1):



Рисунок 1 – Схема турнирной таблицы для 4 команд

2) При 8 командах (Рисунок 2):



Рисунок 2 – Схема турнирной таблицы для 8 команд

Если число команд равно 6 или 10, то сетка должна быть такой, чтобы ко второму туру осталось количество команд, равное степени двойки. В данном случае 4 или 8. Следовательно, согласно жребию, часть команд приступит к играм во втором туре, а часть – в первом.

3) При 6 командах (Рисунок 3):



Рисунок 3 – Схема турнирной таблицы для 6 команд

4) При 10 командах (Рисунок 4):

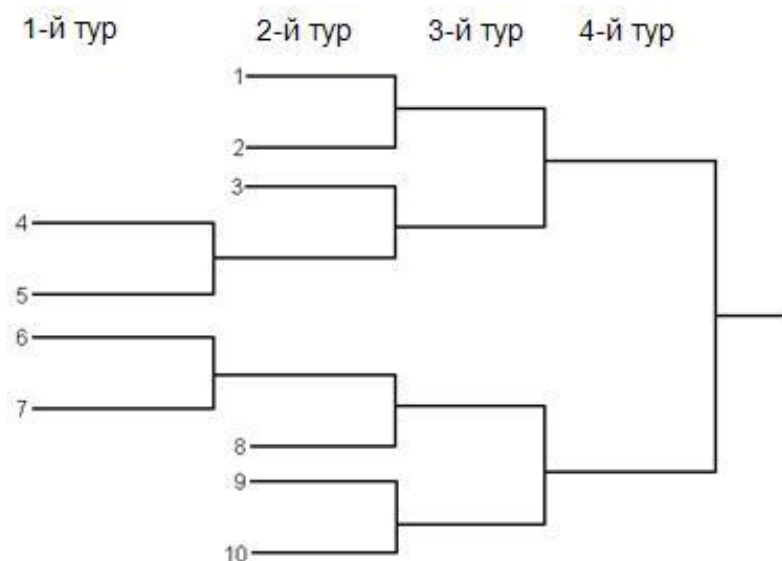


Рисунок 4 – Схема турнирной таблицы для 10 команд

При нечетном количестве команд-участниц коллективы, вступающие в игру со второго тура, распределяются таким образом, чтобы внизу сетки было на один больше.

5) При 5 командах (Рисунок 5):

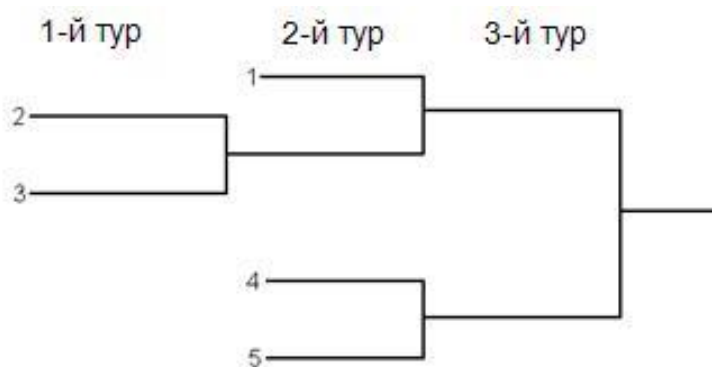


Рисунок 5 – Схема турнирной таблицы для 5 команд

6) При 7 командах (Рисунок 6):



Рисунок 6 – Схема турнирной таблицы для 7 команд

7) При 9 командах (Рисунок 7):



Рисунок 7 – Схема турнирной таблицы для 9 команд

8) При 19 командах (Рисунок 8):

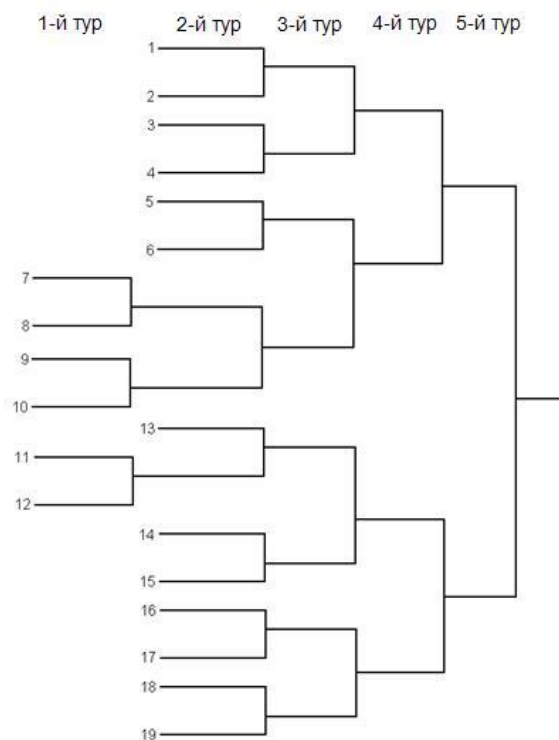


Рисунок 8– Схема турнирной таблицы для 19 команд

#### 2.4. Смешанная система

Смешанная система проведения соревнований объединяет две основные системы розыгрыша - круговую и олимпийскую. В этом формате часть соревнований (обычно предварительная или финальная часть) проводится по олимпийской системе с выбыванием участников, а другая часть - по круговой системе.

В некоторых случаях система с выбыванием может быть применена на предварительных этапах соревнований, когда команды разделены на подгруппы, а оставшиеся команды выходят в одну группу сильнейших, где проводятся матчи по круговой системе.

Другой вариант смешанной системы включает в себя проведение предварительных этапов в подгруппах, где команды играют между собой по круговой системе, а затем проводится финальная часть соревнований по системе с выбыванием.

Применение смешанной системы часто встречается на крупных международных соревнованиях, таких как чемпионаты мира, Европы или олимпийские футбольные турниры. Этот формат позволяет проводить соревнования в относительно короткие сроки и более точно определить уровень игры и силу каждой участвующей команды.

Преимуществом смешанной системы является возможность проведения соревнований в сравнительно небольшой срок при значительном числе участвующих команд. Это позволяет более динамично и увлекательно протекать соревнования, а также более точно определить лучших команд-участниц.

Приведем пример на 16 командах, разделенных на 4 группы по 4 команды. После проведения группового этапа по круговой системе выявляются лидеры каждой группы (Рисунок 9):

	Место
Команда 1	1
Команда 2	2
Команда 3	3
Команда 4	4

	Место
Команда 5	1
Команда 6	2
Команда 7	3
Команда 8	4

	Место
Команда 9	1
Команда 10	2
Команда 11	3
Команда 12	4

	Место
Команда 13	1
Команда 14	2
Команда 15	3
Команда 16	4

Рисунок 9 – Расположение представленных команд по местам в группах в виде таблиц



В следующий этап из каждой группы выходят по две команды занявшие первые места и образуют турнирную сетку для плей-оффа (Рисунок 10):

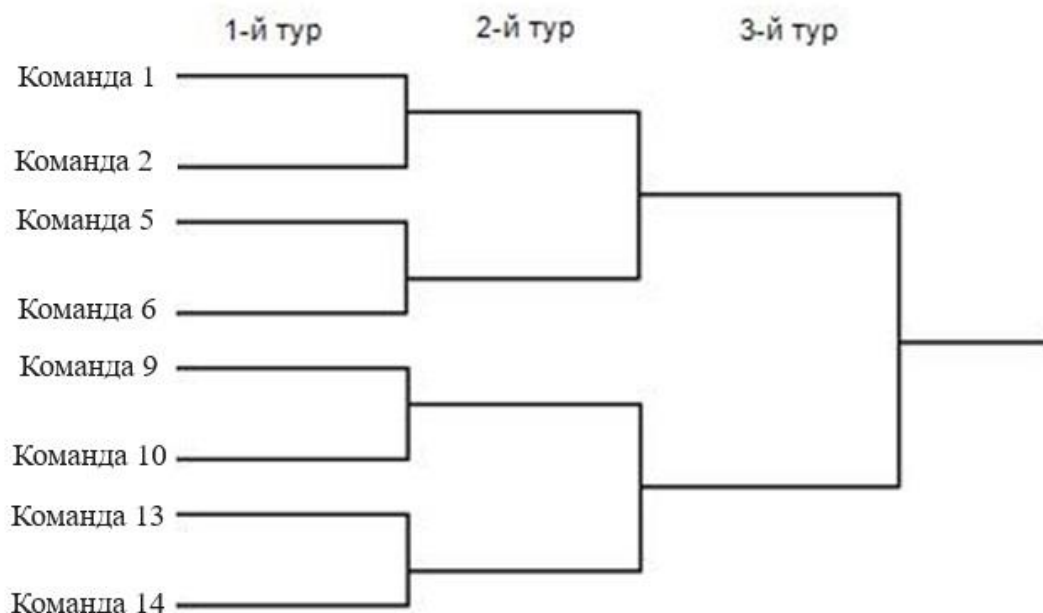


Рисунок 10 – Схема турнирной таблицы для лидеров команд из 4 групп

## 2.5. Выбор технологий для разработки

Выбор технологий играет важную роль в определении архитектуры, функциональности и масштабирования проекта. При выборе технологий для клиентской и серверной частей веб-приложения были установлены следующие основные критерии:

- Популярность технологии: Приоритет отдавался технологиям, которые пользуются широкой популярностью в индустрии разработки веб-приложений. Популярные технологии обычно поддерживаются активным сообществом разработчиков, что предоставляет доступ к большому объему документации, библиотек и ресурсов, упрощая разработку и поддержку проекта.

- Большое сообщество: Наличие активного и развитого сообщества пользователей и разработчиков является важным фактором при выборе технологий. Распространенные и широко используемые технологии обычно имеют обширные

сообщества, где можно найти помощь, решения проблем и поддержку от опытных разработчиков.

- Зрелость технологии: был установлен критерий, по которому технология должна быть проверена временем и существовать на протяжении не менее года. Это позволяет разработчикам использовать надежные и стабильные технологии, которые имеют широкое применение и подтвержденные практикой успехи.

- Стабильность API: также был учтен фактор стабильности API. Технологии, у которых API меняется слишком часто или сильно, могут привести к проблемам в будущем, связанным с обновлениями и совместимостью. Поэтому было предпочтительно выбрать технологии с более стабильными и предсказуемыми API.

Перед началом работы, были изучены различные источники литературы и электронных ресурсов. Это позволило провести исследование и получить необходимое понимание о различных технологиях и инструментах, которые могут быть использованы для разработки веб-приложения.

В результате проведенного исследования и анализа были определены необходимые технологии и инструменты для реализации веб-приложения. Выбор технологий основывался на описанных критериях, чтобы обеспечить оптимальные результаты и эффективность разработки проекта.

### 2.5.1 Выбор технологии для клиентской части

Популярными технологиями для разработки клиентской части веб-приложения являются следующие:

- jQuery: Известная как одна из самых популярных библиотек JavaScript, но в последние годы ее популярность снизилась, и разработчики все чаще применяют другие библиотеки или фреймворки для создания SPA-приложений.

- Angular: Этот фреймворк часто выпускает новые несовместимые версии, что может привести к сложностям в поддержке и обновлении кода.

- Vue.js: Этот фреймворк был выпущен в 2014 году и с тех пор быстро набирает популярность. Создатели Vue.js опирались на опыт разработчиков Angular и ReactJS,

но также, как и Angular, часто выпускают обновления, которые могут быть несовместимы со старыми версиями.

– ReactJS: Эта библиотека была создана компанией Facebook в 2013 году. ReactJS является простой, но мощной библиотекой с большим сообществом разработчиков. Интерфейс новых версий не меняется сильно, поэтому код легко поддерживать и обновлять. Кроме того, ReactJS позволяет относительно легко реализовывать SSR (серверный рендеринг).

В результатах исследования было решено использовать ReactJS для разработки клиентской части веб-приложения.

В качестве библиотеки для визуальных элементов (кнопок, таблиц и т.д.) была выбрана Ant Design.

Преимущества Ant Design:

- Адаптивность;
- Большой набор графических элементов;
- Удобная работа с формами;
- Хорошая документация.

Однако у Ant Design есть и некоторые недостатки, включая большой размер библиотеки, что может замедлять загрузку страницы.

## 2.5.2 Выбор технологии для серверной части

Для разработки серверной части приложения был выбран язык Python версии 3.8. Python широко применяется в различных областях, включая написание desktop-приложений и использование нейронных сетей. Он также входит в ТОП-5 языков программирования по версии ТЮВЕ.

Для написания веб-приложений на Python часто выбирают фреймворки Flask или Django. Существуют и другие фреймворки и библиотеки, такие как Pyramid, TurboGears, Web2py, Bottle, Sanic, CherryPy, Tornado и Dash.

Однако Django обладает большим сообществом разработчиков и широким набором функций. Он также предоставляет множество дополнительных библиотек,

которые были написаны сторонними разработчиками. Django следует шаблону проектирования MVC, хотя применяется другая номенклатура. Модель в Django строится с помощью Django ORM, представление осуществляется с помощью Django шаблонизатора, а контроллеры в Django называются представлениями (Views).

Одной из ключевых особенностей Django является удобный ORM (объектно-реляционное отображение), который облегчает работу с базами данных.

### 2.5.3 Выбор технологии для базы данных

Хранение данных в нашем приложении будет осуществляться с использованием СУБД PostgreSQL [11]. Мы выбрали эту базу данных по нескольким причинам:

1) Совместимость с Django ORM: PostgreSQL полностью совместима с Django ORM, что обеспечивает высокую эффективность и удобство работы с данными. Django ORM предоставляет мощные инструменты для создания, чтения, обновления и удаления записей в базе данных PostgreSQL, а также обеспечивает защиту от SQL-инъекций и облегчает переносимость кода между различными СУБД.

2) Поддержка JSON: PostgreSQL предоставляет расширенную поддержку для работы с данными в формате JSON. Это позволяет хранить данные в структурированном формате JSON и выполнять поиск по ним. Такая гибкость и функциональность JSON-поддержки в PostgreSQL может быть полезна для обработки и хранения разнообразной семантической информации.

3) Рекомендация от официальной документации Django: Django рекомендует использовать PostgreSQL в качестве первого варианта для развития приложений на его основе. Официальная документация Django активно поддерживает и документирует возможности взаимодействия Django ORM с PostgreSQL, что делает их взаимодействие естественным и удобным для разработчика.

Перед проектированием приложения, важно определить модель предметной области. Построение модели предметной области позволяет оценить сложность приложения и выбрать подходящие архитектурные решения для его реализации.

Модель предметной области определяет структуру данных, их взаимосвязь и взаимодействие, а также помогает понять требования к функциональности приложения. Такой подход способствует эффективной разработке и обеспечивает согласованность и понятность системы.

#### 2.5.4 Вспомогательные инструменты для разработки

Для разработки был выбрана многофункциональный редактор Visual Studio Code [12].

Visual Studio Code (VS Code) — это бесплатный и открытый исходный код редактор, разработанный Microsoft. Он является одним из самых популярных инструментов для разработки веб-приложений и предлагает широкий набор функциональности для повышения производительности разработчика.

Основные преимущества использования Visual Studio Code в разработке веб-приложений:

- **Расширяемость:** VS Code имеет мощную систему расширений, которая позволяет разработчикам настраивать и расширять функциональность редактора под свои нужды. Существует огромное количество расширений, доступных из магазина VS Code, которые позволяют добавить поддержку различных языков программирования, интеграцию с системами контроля версий, отладчики и многое другое.

- **Интеграция с Git:** VS Code предоставляет удобный интерфейс для работы с системой контроля версий Git. Он позволяет легко просматривать изменения, сравнивать файлы и коммитить изменения прямо из редактора.

- **Отладка:** Встроенный отладчик в VS Code позволяет удобно отлаживать веб-приложения. Он поддерживает различные языки программирования и позволяет устанавливать точки останова, анализировать переменные и выполнение кода пошагово.

– Интеграция с терминалом: VS Code имеет встроенный терминал, который позволяет запускать команды прямо из редактора. Это упрощает выполнение различных задач, таких как установка зависимостей, запуск сервера и другие.

– Быстрота и производительность: VS Code известен своей скоростью и отзывчивостью. Он быстро открывает большие проекты и обеспечивает плавную работу даже при работе с большим количеством файлов.

– Однако, как и любой другой инструмент, у Visual Studio Code есть и некоторые недостатки:

– Ресурсоемкость: при работе с очень большими проектами или при одновременном использовании нескольких расширений, VS Code может потреблять большое количество оперативной памяти и процессорного времени.

– Отсутствие полноценной интегрированной среды разработки (IDE): в отличие от некоторых других решений, таких как JetBrains WebStorm или Microsoft Visual Studio, VS Code является редактором кода с некоторыми функциями IDE. Он не предлагает такую полную интеграцию и поддержку для определенных языков программирования или фреймворков.

– Некоторые функции требуют установки дополнительных расширений: хотя VS Code имеет множество встроенной функциональности, некоторые продвинутые возможности могут требовать установки дополнительных расширений, что может занять время и привести к необходимости настройки редактора.

В итоге использование VS Code в разработке веб-приложений предлагает значительное количество преимуществ, таких как расширяемость, удобная навигация по коду, поддержка систем контроля версий и производительность. Несмотря на некоторые ограничения, VS Code остается популярным выбором для многих разработчиков веб-приложений за свою гибкость, легкость работы и богатый набор функциональности.

## Пример интерфейса VS Code (Рисунок 11):

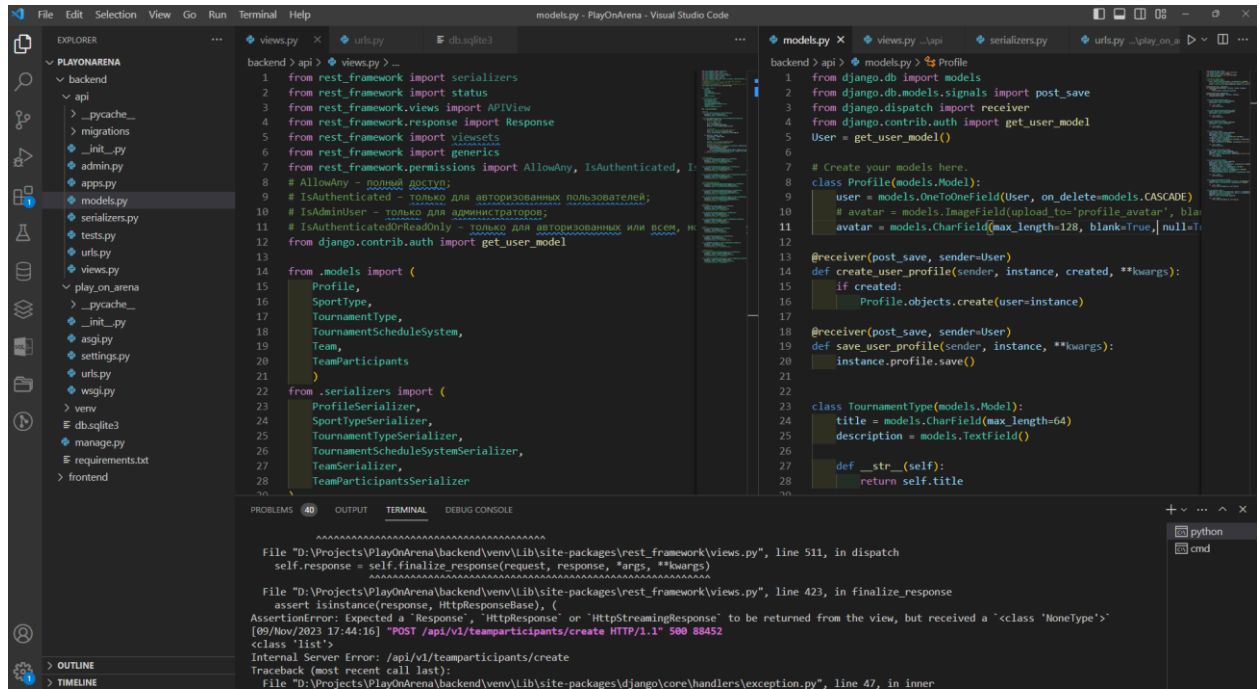


Рисунок 11 - Пример интерфейса VS Code

### 3. Проектирование и реализация системы

#### 3.1. Функциональные характеристики системы

При разработке сформулированы следующие обязательные требования к функционалу разрабатываемого web-приложения:

- 1) Регистрация новых или ввод данных существующих пользователей;
- 2) Создание соревнований, команд;
- 3) Возможность отправки заявки на участие в соревновании;
- 4) Реализация основных таблиц базы данных: таблица с соревнованиями, таблица с командами; таблица с расписанием;
- 5) Визуализация данных в виде схем, диаграмм, таблиц.

#### 3.2. Требования к пользовательскому интерфейсу приложения

Сформулируем основные требования пользовательского интерфейса, которые необходимо обеспечить для продуктивной работы пользователя:

- 1) Поддержка русского языка;
- 2) Интерфейс должен быть спроектирован с учетом целей, мотивов и потребностей целевой аудитории при использовании системы;
- 3) Интерфейс системы должен обеспечивать наглядное, интуитивно понятное представление структуры размещенной информации, быстрый и логичный переход к соответствующим разделам системы;
- 4) Навигационные элементы интерфейса системы должны обеспечивать однозначное понимание пользователем их смысла и обеспечивать навигацию по всем доступным пользователю разделам системы и отображать соответствующую информацию.



### 3.3. Общая архитектура приложения [15]

Архитектура играет ключевую роль в разработке любого приложения. При проектировании нашего приложения мы выбрали отдельный подход frontend и backend, который является популярным и широко применяемым решением в индустрии разработки.

Отдельный подход frontend и backend позволяет нам эффективно разделить ответственности между клиентской и серверной частями приложения. Фронтенд будет построен с использованием SPA (Single-Page Application) технологии, что позволяет создать более динамичный и отзывчивый пользовательский интерфейс. Запросы от фронтенда к серверу будут осуществляться с использованием REST API, что обеспечивает гибкость и удобство взаимодействия между клиентом и сервером.

Для обеспечения безопасности и авторизации пользователей мы будем использовать токены. Токен-аутентификация является распространенным и безопасным методом аутентификации, позволяющим передавать токен в каждом запросе и осуществлять проверку доступа на серверной стороне.

Серверная часть будет построена с использованием архитектурного шаблона MVC (Model-View-Controller). Эта архитектура позволяет явно отделить данные, логику и представление в приложении. Представление будет возвращать данные в формате JSON, что облегчает обмен данными между клиентом и сервером (Рисунок 12).

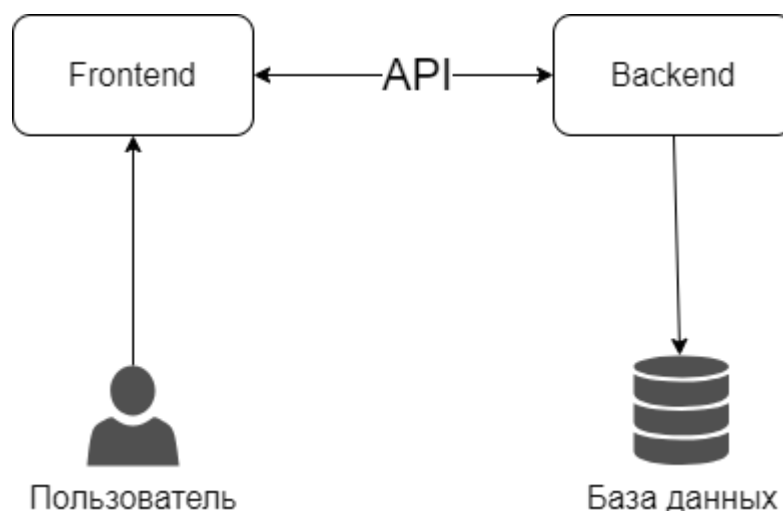


Рисунок 12 – Схема архитектуры веб приложения

### 3.3.1 Проектирование REST API

Для связи между клиентской и серверной частями приложения будет использоваться API. В данном случае будет использован REST API, так как это наиболее популярный подход. Кроме того, для Django существует отличная библиотека для работы с REST API — Django Rest Framework (DRF).

Для аутентификации пользователей будет использоваться токен-аутентификация. Аутентификация на основе токена работает следующим образом: пользователь проходит процедуру аутентификации на сервере, после чего ему выдается токен. Этот токен содержит информацию о пользователе и используется для проверки подлинности запросов от клиента. При каждом запросе к API сервер проверяет токен клиента и, если токен действителен, обрабатывает запрос. Если токен недействителен, сервер возвращает сообщение об ошибке.

## 3.4. Реализация

В этой главе будет описываться реализация клиентской и серверной части разрабатываемого веб-приложения.

### 3.4.1 Требования к аппаратным средам

Требования к аппаратным средам выполнения данного программного решения следует разбить на требования к клиентским машинам, где будет лишь отображаться система, и требования к серверной части, где будет выполняться вся логика приложения.

Для клиентской машины минимальными требованиями будут являться требования, при которых может быть запущен web-браузер или его аналоги.

Для серверной части минимальными требованиями, при которых гарантируется стабильная работа приложения, являются:

- процессор: Intel Core или его аналоги с тактовой частотой не ниже 2,11 ГГц;

- оперативная память: 4 Гб ОЗУ DDR4+ или более;
- дисковое пространство: не менее 40Гб дискового пространства для приложения и базы данных.

Главной отличительной чертой веб-приложений и веб-сайтов является их кросс-платформенность, то есть отсутствие привязки к конкретной операционной системе.

Таким образом, веб-сайт может быть запущен в любой ОС, в которой имеется браузер.

Требования к ОС для пользователей сервиса:

- ОС: Windows, \*nix-подобные системы (Linux, Unix, Ubuntu, Debian, FreeBSD), MAC OS;
- Microsoft Edge или его аналоги.
- Для работы сервиса на стороне сервера необходимы следующие ресурсы:
- ОС: Windows Server 2008 и выше, либо аналогичные \*nix-системы (CentOS 7, Debian server и т.д.);
- Docker версии 20.10.16 или выше;
- Docker compose версии 1.92 или выше;
- WinSCP;
- Postman;
- PUTTy;

При разработке сервиса по созданию спортивных соревнований использовались следующие приложения и программные продукты:

- языки программирования: Python >3.8, JavaScript, TypeScript;
- СУБД: Postgres 15;
- среда разработки: VSCode;
- В рамках магистерской диссертации используются данные параметры:
- Windows 10 OS x64;
- Intel Core i5-1135G7 2.4 ГГц;
- оперативная память 16 Гб;

- жесткий диск 120 Гб;
- видеокарта Intel(R) Iris(R) Xe Graphics.

Настройка локального стенда

### 3.4.2 Управление версиями проекта на основе веб-сервиса GitHub.

GitHub - это веб-сервис для хранения и управления исходным кодом проектов, основанный на системе контроля версий Git. С его помощью можно создавать репозитории, в которых хранятся все версии проекта, а также отслеживать изменения в коде, комментировать их и работать над проектом в команде.

В GitHub можно создавать ветки проекта, что позволяет работать над разными версиями проекта параллельно. Также можно создавать запросы на слияние изменений из разных веток, что упрощает работу в команде.

GitHub также предоставляет инструменты для отслеживания ошибок и задач проекта, а также для автоматической сборки и тестирования кода (Рисунок 13).

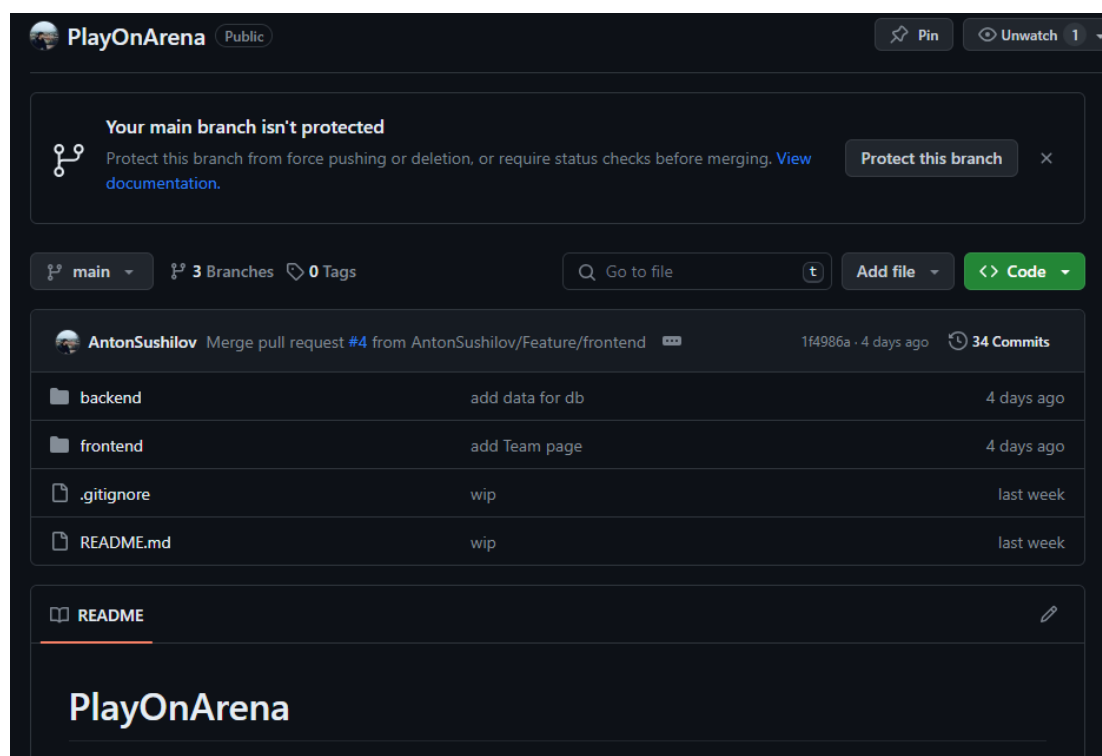


Рисунок 13 – Репозиторий на GitHub

### 3.4.3 Реализация клиентской части

Клиентская часть веб-сервиса будет состоять из 8 основных страниц:

- 1) Главная страница;
- 2) Страница регистрации;
- 3) Страница авторизации;
- 4) Страница профиля пользователя;
- 5) Страница раздела “Турниры”;
- 6) Страница раздела “Команды”;
- 7) Страницы конкретного турнира;
- 8) Страница конкретной команды.

Для разработки клиентской части развернем базовый шаблон веб-приложения на основе React.js.

Структура клиентской части проекта в VSCode изображена на рисунке 14.

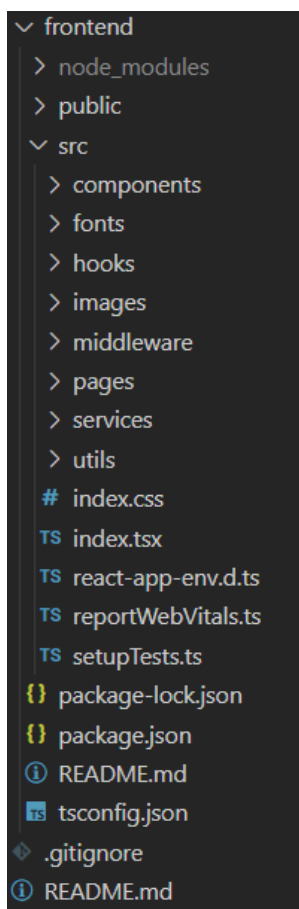


Рисунок 14 – Структура клиентской части проекта

– node\_modules – сторонние пакеты, которые требуются для разработки веб-приложения.

– public – статичный контент веб-приложения.

– src – корень приложения.

– components – компоненты, из которых строится интерфейс приложения.

– fonts – шрифты используемые в приложении.

– hooks – папка с хуками, которые позволяют использовать состояние и другие возможности React без написания классов.

– images – папка для картинок.

– middleware – папка для функций, которые добавляют дополнительную логику в функции.

– pages – папка для компонентов основных страниц приложения.

– utils – папка для вспомогательных функций.

– index.tsx – точка входа в приложение.

После получения макетов основных страниц веб-приложения приступаем к их разработке.

Страница регистрации представлена на рисунке 15.

PlayOnArena Турниры Команды Регистрация Вход

### Регистрация

\* Логин:

\* Пароль:

\* Подтверждение пароля:

[Зарегистрироваться](#)

Уже зарегистрированы? [Войти](#)

Рисунок 15 – Страница регистрации

Логика работы страницы:

- 1) Пользователь вводит свои данные в соответствующие поля форм;
- 2) Нажимает на кнопку «Зарегистрироваться»;
- 3) Из клиентского кода веб-приложения уходит HTTP-запрос в backend;
- 4) Backend возвращает токен, который далее используется для аутентификации запросов получения контента на остальных страницах;
- 5) Токен сохраняется в локальное хранилище браузера;
- 6) В случае возникновения ошибки регистрации, например логин пользователя уже существует в системе, пользователь увидит страницу регистрации с уведомлением об ошибке;
- 7) Пользователь переадресуется на страницу профиля пользователя.

Страница авторизации представлена на рисунке 16.

PlayOnArena Турниры Команды Регистрация Вход

**Вход**

\* Логин: Придумайте логин

\* Пароль: Придумайте пароль

Вход

Вы - новый пользователь? [Зарегистрироваться](#)

Рисунок 16 – Страница авторизации

Логика работы страницы:

- 1) Пользователь вводит логин и пароль в соответствующие поля форм;
- 2) Нажимает на кнопку «Войти»;
- 3) Из клиентского кода веб-приложения уходит HTTP-запрос в backend;

- 4) Backend возвращает токен, который далее используется для аутентификации запросов получения контента на остальных страницах;
- 5) Токен сохраняется в локальное хранилище браузера;
- 6) В случае возникновения ошибки аутентификации, например логин или пароль пользователя указаны неверно, пользователь увидит страницу аутентификации с уведомлением об ошибке;
- 7) Пользователь переадресуется на страницу профиля пользователя.
- Страница с турнирами представлена на рисунке 17.

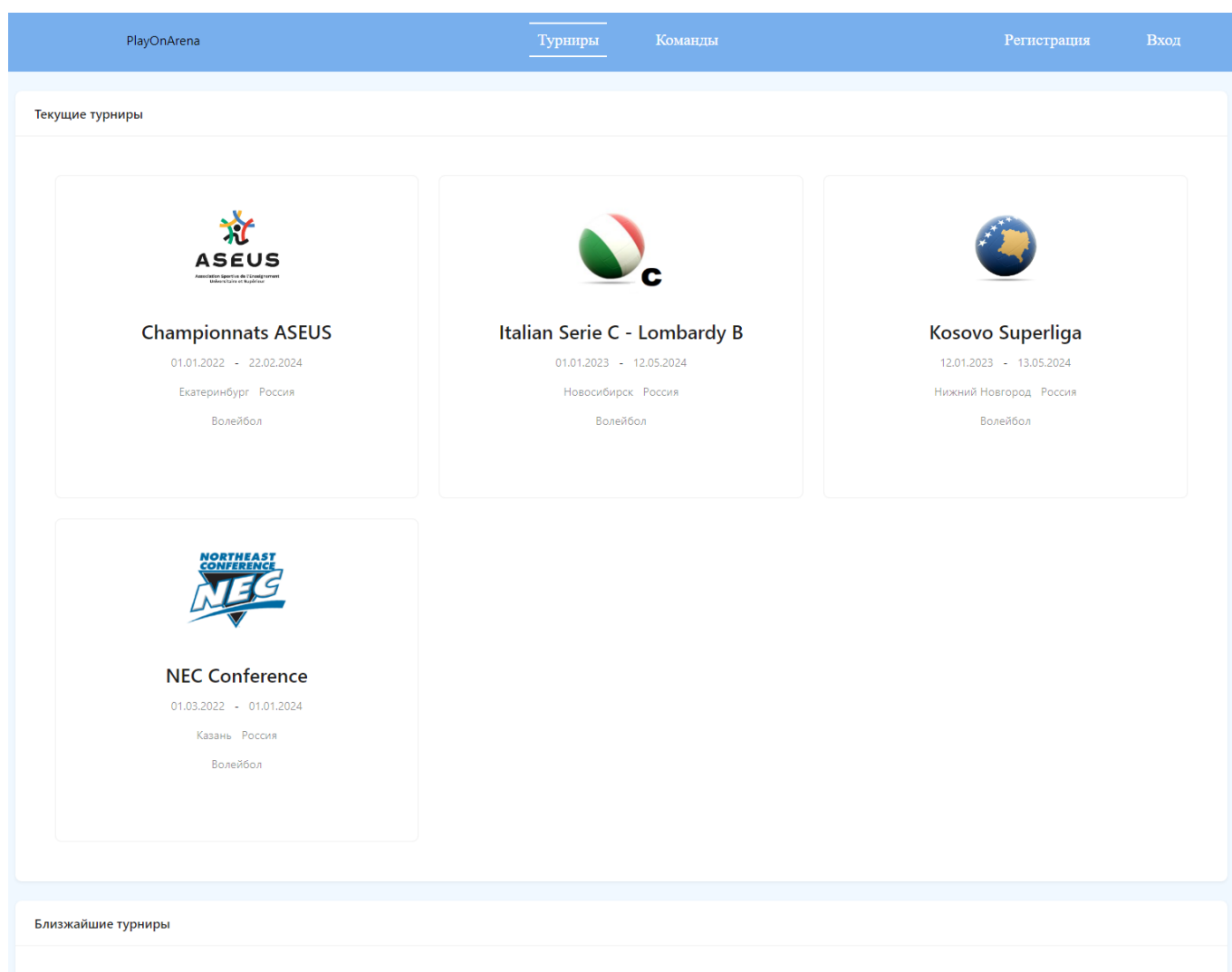


Рисунок 17 – Страница с турнирами



Логика работы страницы:

1) Пользователь может просмотреть турниры, разделенные по группам: Текущие турниры, Ближайшие турниры, Завершенные турниры;

2) Пользователь может нажать на карточку турнира и перейти на страницу детального просмотра информации о нем.

Страница с информацией о турнире представлена на рисунке 18.

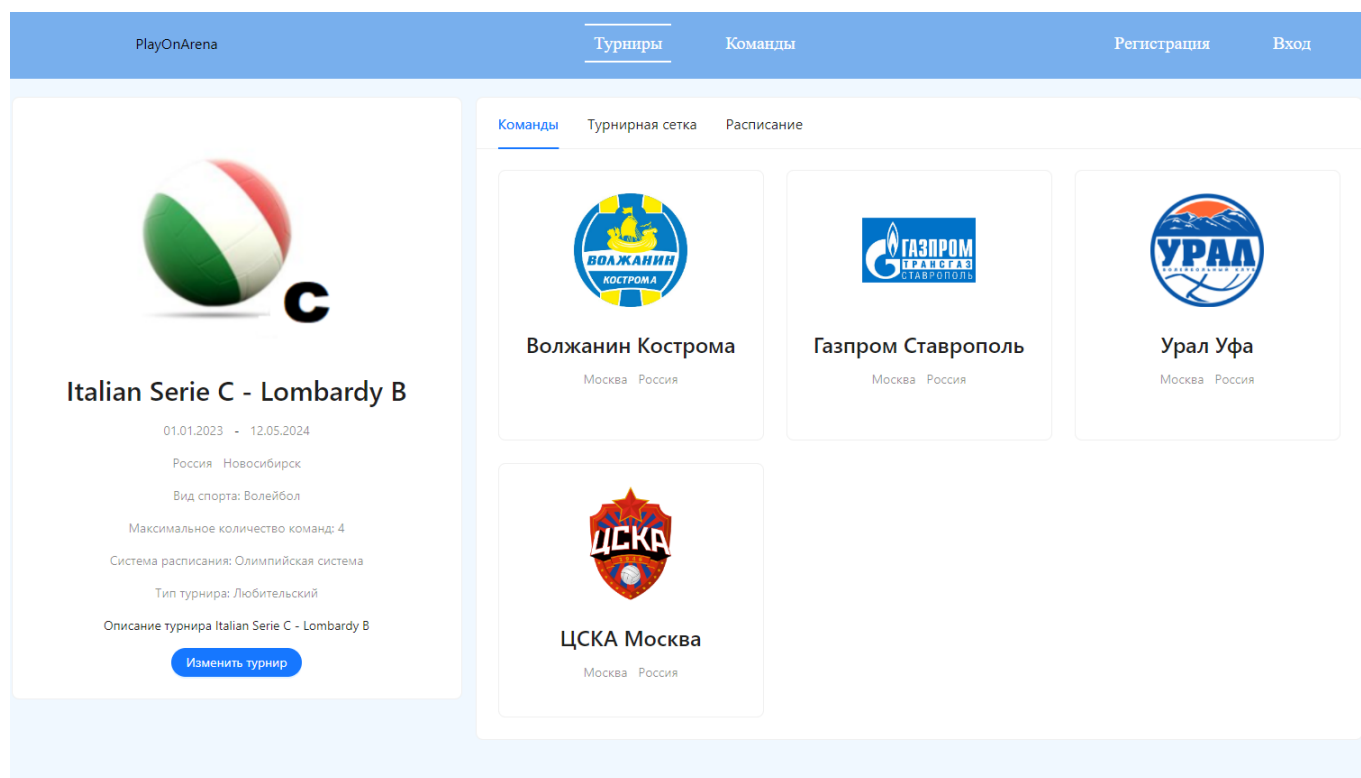


Рисунок 18– Страница конкретного турнира

Логика работы страницы:

1) Пользователь может просмотреть информацию о турнире, командах, которые участвуют в нем, посмотреть турнирную сетку и расписание матчей;

2) Пользователь, который является организатором турнира, может редактировать информацию о турнире, настраивать турнирную сетку и расписание турнира.

Страница с командами представлена на рисунке 19.

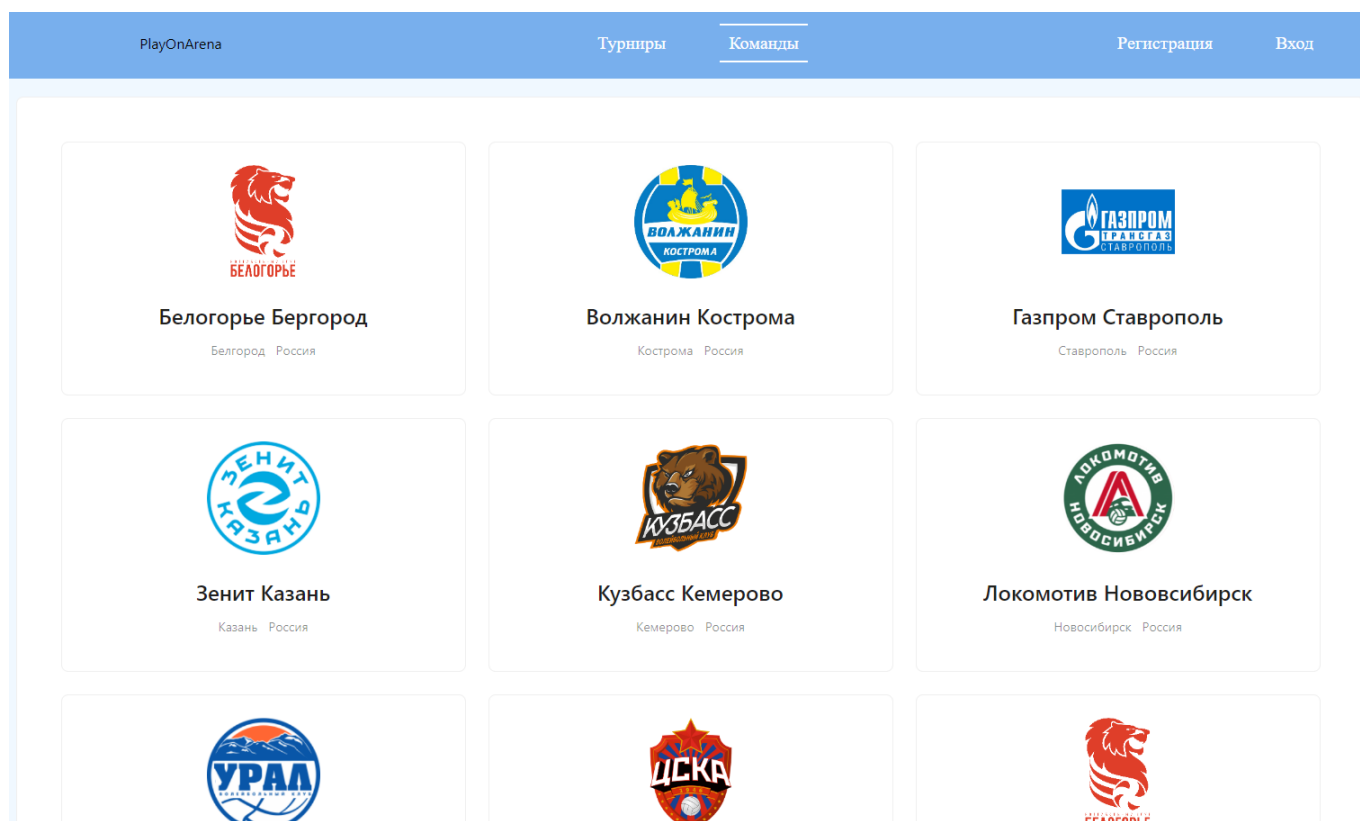


Рисунок 19 – Страница с командами

Логика работы страницы:

- 1) Пользователь может просмотреть команды;
- 2) Пользователь может нажать на карточку команды и перейти на страницу детального просмотра информации о ней.

Страница с информацией о команде представлена на рисунке 20.

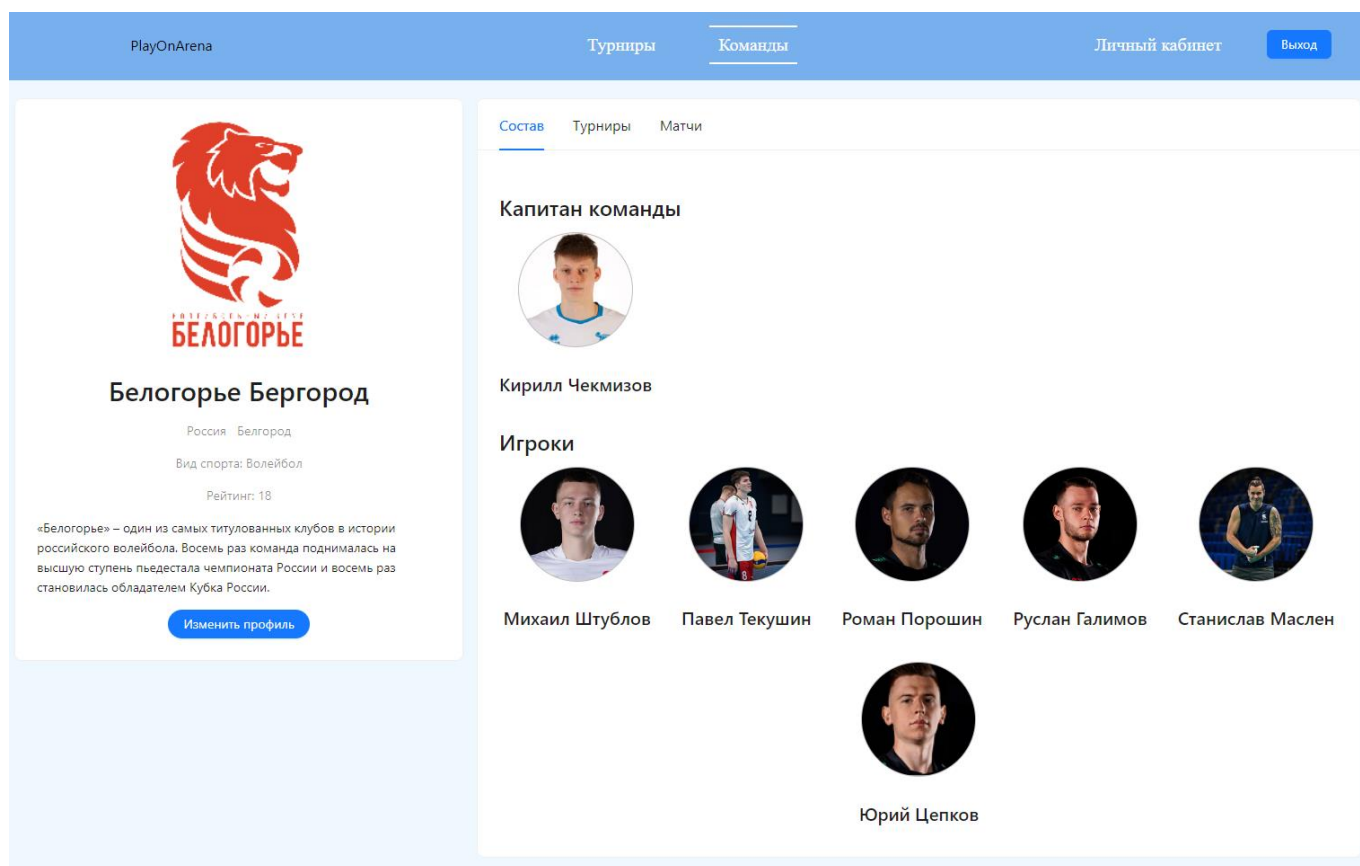


Рисунок 20 – Страница с конкретной командой

Логика работы страницы:

- 1) Пользователь может просмотреть информацию о команде, ее составе, о турнирах, в которых она участвует и список матчей, в которых участвует эта команда;
- 2) Пользователь, который является создателем команды, может редактировать информацию о команде.

Страница профиль пользователя представлена на рисунке 21.

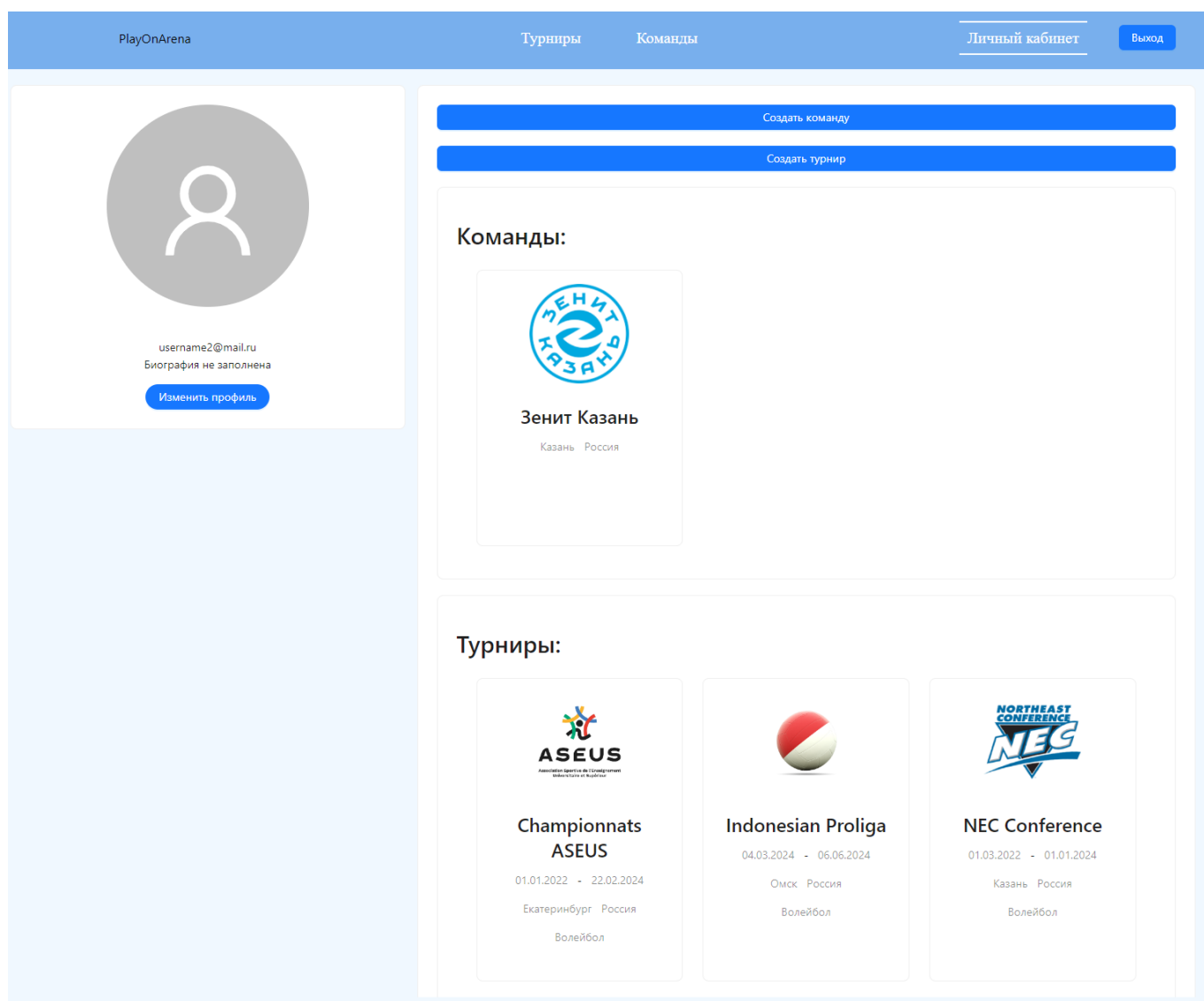


Рисунок 21– Страница профиля пользователя

Логика работы страницы:

- 1) Пользователь может просмотреть информацию о себе, список команд в которых состоит пользователь и список турниров, в которых участвует пользователь;
- 2) Пользователь может изменить информацию о себе;
- 3) Пользователь может создать новую команду или новый турнир.

### 3.4.4 Реализация серверной части

Проект на Django состоит из отдельных приложений. Вместе они образуют полноценное веб-приложение. Каждое приложение представляет какую-то определенную функциональность или группу функциональностей. Один проект может включать множество приложений. Это позволяет выделить группу задач в отдельный модуль и разрабатывать их относительно независимо от других.

Серверная часть веб-сервиса будет состоять из 3 основных приложений django:

1) play\_on\_arena – основное приложение django;

2) users – приложение django, отвечающее за логику связанную с пользователем;

3) tournaments – приложение django, отвечающее за логику связанную с турнирами, командами и расписанием;

4) api – приложение django, отвечающее за endpoint's веб-приложения.

Для разработки клиентской части развернем базовый шаблон веб-приложения на основе Django.

Структура серверной части проекта в VSCode изображена на рисунке 22.

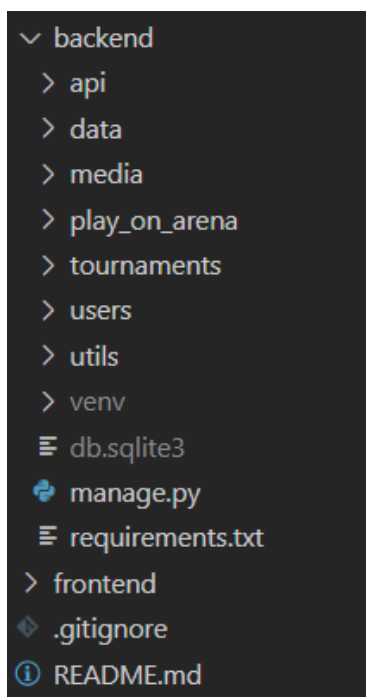


Рисунок 22 - Структура серверной части проекта

Структура серверной части проекта:

– api – директория с приложением api;

– data – директория для тестовых данных в формате json;

- `media` – директория для сохранения на сервере загружаемых файлов;
- `play_on_arena` – директория с приложением `play_on_arena`;
- `tournaments` – директория с приложением `tournaments`;
- `users` – директория с приложением `users`;
- `utils` – директория для вспомогательных функций;
- `venv` – папка с виртуальным окружением `python`.

### 3.4.5 Проектирование базы данных

В приложении используется реляционная база данных PostgreSQL. Особенности работы с Django таковы, что проектирование таблицы происходит в программном коде, в модуле `models.py`. Django фреймворк использует ORM для описания формата БД в виде кода.

Создаётся новый класс, унаследованный от `Django.models`, и каждый атрибут этого класса является полем в БД. Каждый класс — отдельная таблица. Фреймворк позволяет использовать все возможности реляционной базы данных — связывать таблицы отношениями многие ко многим, один к одному, один ко многим, определять свойства поля. Django предоставляет такие виды полей как:

- `CharField`, поле для небольшого текста;
- `IntegerField`, для целочисленных значений;
- `DecimalField`, для десятичных чисел заданной точности;
- `FileField`, поле для отображения файлов, имеет возможность генерировать ссылку на файл;
- `ImageField`, похож на `FileField`, но более заточен на картинки;
- `BooleanField`, для отображения булевых значений.

Это не все поля, которые предоставляет Django, так же существует множество других, в том числе и сторонних библиотек.

Так как для Django требуется создать отдельную схему БД, была спроектирована UML диаграмма классов и результат ее генерации в PostgreSQL представлен на рисунке 23.

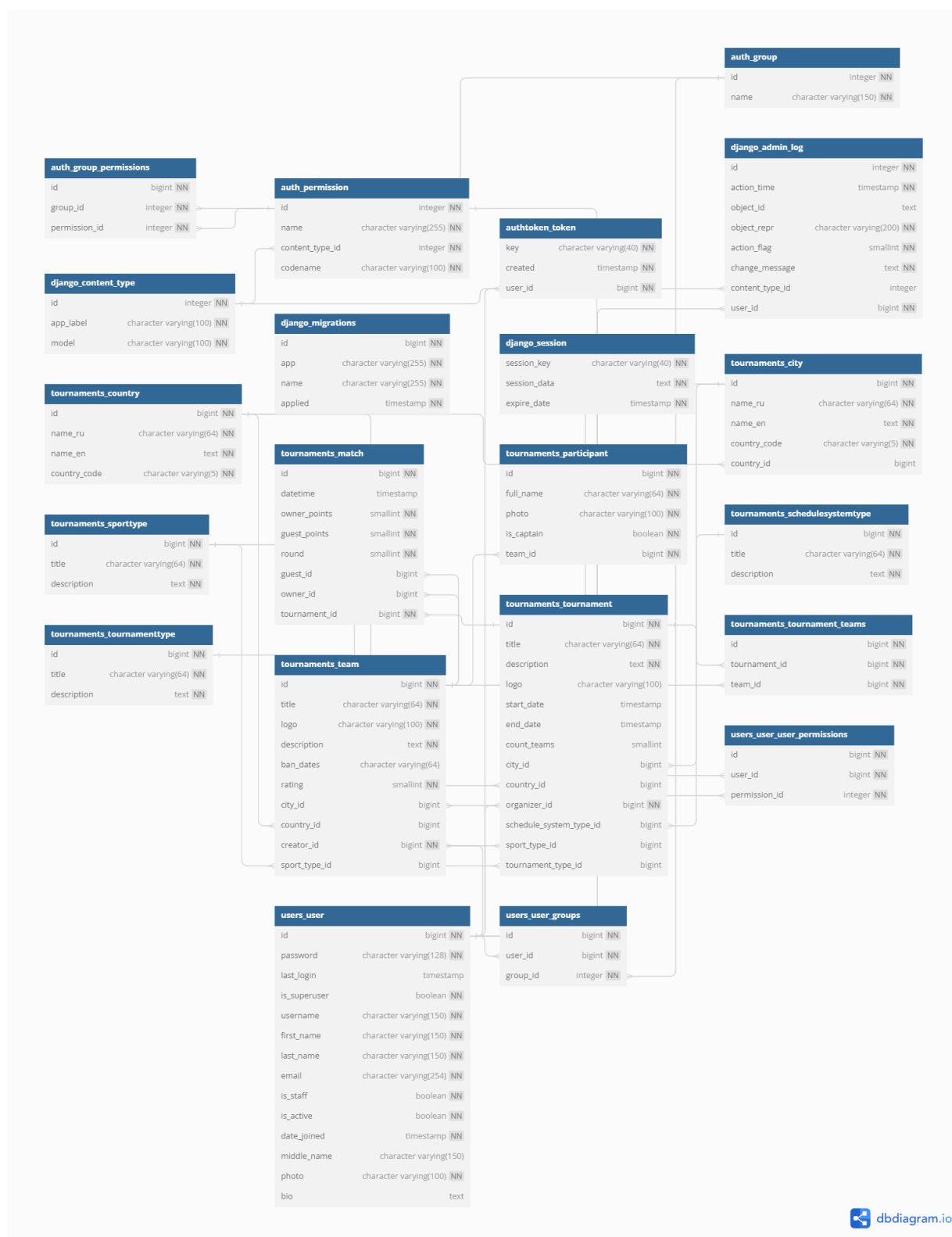


Рисунок 23 – Схема БД

По спроектированной диаграмме классов были созданы модели в приложении Django.

Для данного приложения в качестве основных объектов предметной области используются:

- User, сущность пользователя данной системы;
- Country, сущность страны;
- City, сущность города;
- SportType, сущность вида спорта;
- TournamentType, сущность типа турнира;
- ScheduleSystemType, сущность типа системы расписания;
- Team, сущность команды;
- Participant, сущность участника команды;
- Tournament, сущность турнира;
- Match, сущность матча;

Были установлены следующие связи:

- Для связи «Country - City» используется связь «один-ко-многим» на основании того, что у города может быть только одна страна, а у страны может быть много городов;

- Для связи «SportType - Team» используется связь «один-ко-многим» на основании того, что у команды может быть только один тип спорта, а у типа спорта может быть много команд;

- Для связи «Country - Team» используется связь «один-ко-многим» на основании того, что у команды может быть только одна страна, а у страны может быть много команд;

- Для связи «City - Team» используется связь «один-ко-многим» на основании того, что у команды может быть только один город, а у города может быть много команд;

- Для связи «User - Team» используется связь «один-ко-многим» на основании того, что у команды может быть только один пользователь (создатель), а у пользователя (создателя) может быть много команд;

- Для связи «Team - Participants» используется связь «один-ко-многим» на основании того, что у участника может быть только одна команда, а у команды может быть много участников;



– Для связи «SportType - Tournament» используется связь «один-ко-многим» на основании того, что у турнира может быть только один тип спорта, а у типа спорта может быть много турниров;

– Для связи «Country - Tournament» используется связь «один-ко-многим» на основании того, что у турнира может быть только одна страна, а у страны может быть много турниров;

– Для связи «City - Tournament» используется связь «один-ко-многим» на основании того, что у турнира может быть только один город, а у города может быть много турниров;

– Для связи «ScheduleSystemType - Tournament» используется связь «один-ко-многим» на основании того, что у турнира может быть только одна система расписания, а у системы расписания может быть много турниров;

– Для связи «TournamentType - Tournament» используется связь «один-ко-многим» на основании того, что у турнира может быть только один тип турнира, а у типа турнира может быть много турниров;

– Для связи «Match - Team» используется связь «один-ко-многим» на основании того, что у матча может быть только одна команда (“хозяева”), а у команды (“хозяева”) может быть много матчей;

– Для связи «Match - Team» используется связь «один-ко-многим» на основании того, что у матча может быть только одна команда (“гости”), а у команды (“гости”) может быть много матчей;

– Для связи «Match - Tournament» используется связь «один-ко-многим» на основании того, что у матча может быть только один турнир, а у турнира может быть много матчей;

– Для связи «Tournament - Team» используется связь «многие-ко-многим» на основании того, что у турнира может быть много команд, а у команды может быть много турниров;

Модели из диаграммы классов были воссозданы в моделях Django (Рис.25, Рис.26, Рис.27, Рис.28, Рис.29, Рис.30, Ри.31), была проведена миграция с помощью

команды «python manage.py makemigrations» и «python manage.py migrate», были созданы новые таблицы в базе данных PostgreSQL рисунок 24.

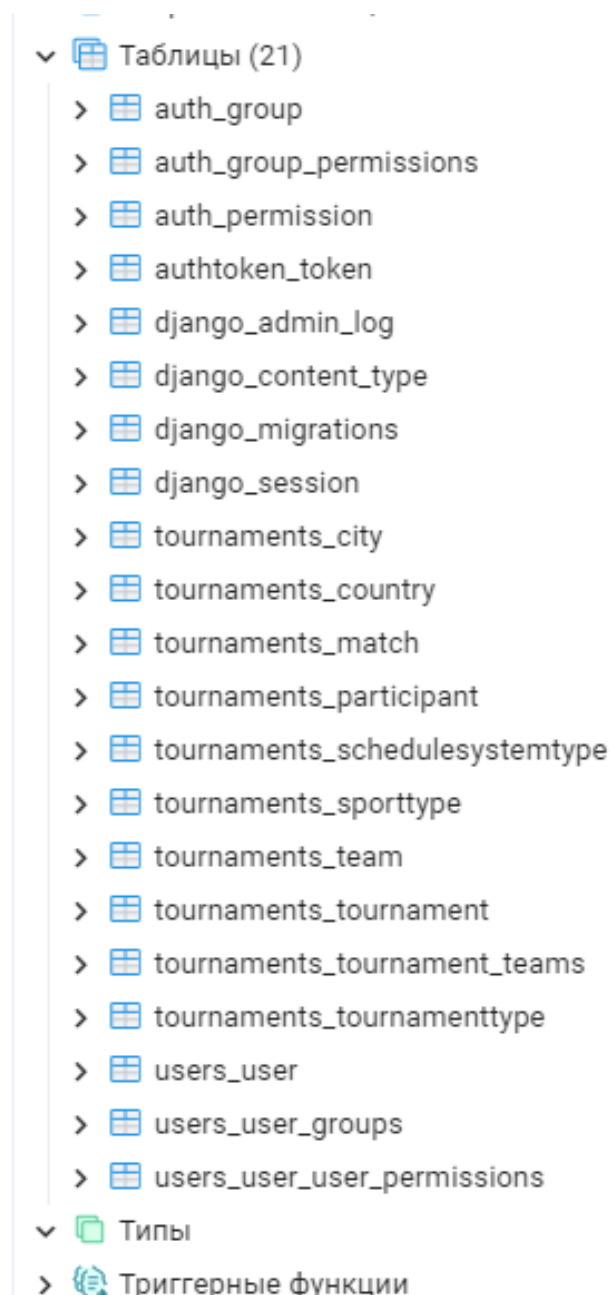


Рисунок 24 – Таблицы в базе данных PostgreSQL

```

class Country(models.Model):
    name_ru = models.CharField(
        'Название на русском',
        max_length=enums.CountryEnums.NAME_MAX_LENGTH.value
    )
    name_en = models.TextField(
        'Название на английском',
        max_length=enums.CountryEnums.NAME_MAX_LENGTH.value
    )
    country_code = models.CharField(
        'Код страны',
        max_length=enums.CountryEnums.COUNTRY_CODE_MAX_LENGTH.value
    )

```

Рисунок 25 – Модель Country

```

class City(models.Model):
    name_ru = models.CharField(
        'Название на русском',
        max_length=enums.CityEnums.NAME_MAX_LENGTH.value
    )
    name_en = models.TextField(
        'Название на английском',
        max_length=enums.CityEnums.NAME_MAX_LENGTH.value
    )
    country_code = models.CharField(
        'Код страны',
        max_length=enums.CityEnums.COUNTRY_CODE_MAX_LENGTH.value
    )
    country = models.ForeignKey(
        Country,
        models.SET_NULL,
        related_name='cities',
        verbose_name='Страна',
        null=True
    )

```

Рисунок 26 – Модель City

```

class TypeModel(models.Model):
    title = models.CharField(
        'Название',
        max_length=enums.TypeEnums.TITLE_MAX_LENGTH.value
    )
    description = models.TextField(
        'Описание',
        max_length=enums.TypeEnums.DESRIPTION_MAX_LENGTH.value
    )

    class Meta:
        abstract = True

class SportType(TypeModel):
    class Meta:
        verbose_name = 'Вид спорта'
        verbose_name_plural = 'Виды спорта'
        ordering = ('title',)

    def __str__(self):
        return self.title

class TournamentType(TypeModel):
    class Meta:
        verbose_name = 'Тип турнира'
        verbose_name_plural = 'Типы турнира'
        ordering = ('title',)

    def __str__(self):
        return self.title

class ScheduleSystemType(TypeModel):
    class Meta:
        verbose_name = 'Система расписания'
        verbose_name_plural = 'Системы расписания'
        ordering = ('title',)

    def __str__(self):
        return self.title

```

Рисунок 27 – Модель Type Model

```

class Team(models.Model):
    title = models.CharField(
        'Название',
        max_length=enums.TeamEnums.TITLE_MAX_LENGTH.value
    )
    logo = models.ImageField(
        'Логотип',
        upload_to='images/teams/',
    )
    description = models.TextField(
        'Описание',
        max_length=enums.TeamEnums.DESCRPTION_MAX_LENGTH.value
    )
    country = models.ForeignKey(
        Country,
        models.SET_NULL,
        related_name='teams',
        verbose_name='Страна',
        null=True
    )
    city = models.ForeignKey(
        City,
        models.SET_NULL,
        related_name='teams',
        verbose_name='Город',
        null=True
    )
    ban_dates = models.CharField(
        'Запрещенные дни',
        max_length=enums.TeamEnums.BAN_DATES_MAX_LENGTH.value,
        null=True,
        blank=True
    )
    sport_type = models.ForeignKey(
        SportType,
        models.SET_NULL,
        related_name='teams',
        verbose_name='Вид спорта',
        null=True
    )
    creator = models.ForeignKey(
        User,
        models.CASCADE,
        related_name='created_teams',
        verbose_name='Создатель'
    )
    rating = models.PositiveSmallIntegerField('Рейтинг')

```

Рисунок 28 – Модель Team

```

class Participant(models.Model):
    full_name = models.CharField(
        'ФИО',
        max_length=enums.ParticipantEnums.FULL_NAME_MAX_LENGTH.value
    )
    photo = models.ImageField(
        'Фотография',
        upload_to='images/participants/',
    )
    team = models.ForeignKey(
        Team,
        models.CASCADE,
        related_name='participants'
    )
    is_captain = models.BooleanField(
        'Капитан',
        default=False
    )

```

Рисунок 29 – Модель Participant

```

class Tournament(models.Model):
    title = models.CharField(
        'Название',
        max_length=enums.TournamentEnums.TITLE_MAX_LENGTH.value
    )
    description = models.TextField(
        'Описание',
        max_length=enums.TournamentEnums.DESCRPTION_MAX_LENGTH.value
    )
    logo = models.ImageField(
        'Фотография',
        upload_to='images/tournaments/',
        blank=True,
        null=True
    )
    start_date = models.DateTimeField(null=True)
    end_date = models.DateTimeField(null=True)
    count_teams = models.PositiveSmallIntegerField(
        'Количество команд', null=True)
    organizer = models.ForeignKey(User, models.CASCADE, related_name='tournaments', verbose_name='Организатор')
    country = models.ForeignKey(Country, models.SET_NULL, related_name='country', verbose_name='Страна', null=True)
    city = models.ForeignKey(City, models.SET_NULL, related_name='city', verbose_name='Город', null=True)
    sport_type = models.ForeignKey(SportType, models.SET_NULL, related_name='tournaments', verbose_name='Вид спорта', null=True)
    tournament_type = models.ForeignKey(TournamentType, models.SET_NULL, related_name='tournaments', verbose_name='Вид турнира', null=True)
    schedule_system_type = models.ForeignKey(
        ScheduleSystemType,
        models.SET_NULL,
        related_name='tournaments',
        verbose_name='Система расписания',
        null=True
    )
    teams = models.ManyToManyField(
        Team,
        related_name='tournaments',
        verbose_name='Команды'
    )
    start_date = models.DateTimeField(null=True)
    end_date = models.DateTimeField(null=True)
    count_teams = models.PositiveSmallIntegerField(
        'Количество команд', null=True)

```

Рисунок 30 – Модель Tournament

```

class Match(models.Model):
    owner = models.ForeignKey(
        Team,
        models.SET_NULL,
        related_name='owned_matches',
        verbose_name='Хозяева',
        null=True
    )
    guest = models.ForeignKey(
        Team,
        models.SET_NULL,
        related_name='guested_matches',
        verbose_name='Гости',
        null=True
    )
    tournament = models.ForeignKey(
        Tournament,
        models.CASCADE,
        related_name='matches',
        verbose_name='Турнир',
    )
    datetime = models.DateTimeField(null=True)
    owner_points = models.PositiveSmallIntegerField(
        'Очки хозяев',
        default=enums.MatchEnums.POINTS_DEFAULT_VALUE.value
    )
    guest_points = models.PositiveSmallIntegerField(
        'Очки гостей',
        default=enums.MatchEnums.POINTS_DEFAULT_VALUE.value
    )
    round = models.PositiveSmallIntegerField(
        'Раунд',
        default=enums.MatchEnums.POINTS_DEFAULT_VALUE.value
    )

```

Рисунок 31 – Модель Match

### 3.4.6 Административный интерфейс

После определения модели, Django может автоматически создать удобный интерфейс администратора, позволяющий авторизованным пользователям добавлять, изменять и удалять объекты. Для этого следует зарегистрировать свою модель с помощью декоратора `@admin.register(Model)`. Пример регистрации моделей представлен ниже (Рисунок 32):

```
@admin.register(models.Team)
class TeamAdmin(admin.ModelAdmin):
    list_display = (
        'id',
        'title',
        'description',
        'sport_type',
        'creator'
    )
    list_filter = (
        'title',
        'sport_type',
        'creator'
    )

@admin.register(models.Participant)
class ParticipantAdmin(admin.ModelAdmin):
    list_display = ('id', 'full_name', 'photo', 'team', 'is_captain')
    list_filter = ('full_name', 'team', 'is_captain')
    readonly_fields = ('id',)

@admin.register(models.Tournament)
class TournamentAdmin(admin.ModelAdmin):
    list_display = (
        'title',
        'id',
        'description',
        'organizer',
        'sport_type',
        'tournament_type',
        'schedule_system_type',
    )
    list_filter = (
        'title',
        'organizer',
```

Рисунок 32 – Регистрация моделей



Особенностью Django Admin (Рисунок) является то, что редактировать разделы могут только администраторы, то есть суперпользователи. Суперпользователь (англ. superuser) – пользователь, который имеет полный доступ к управлению сайтом. Чтобы создать суперпользователя, нужно выполнить команду «python manage.py createsuperuser», и ввести необходимые данные (Рисунок 33).

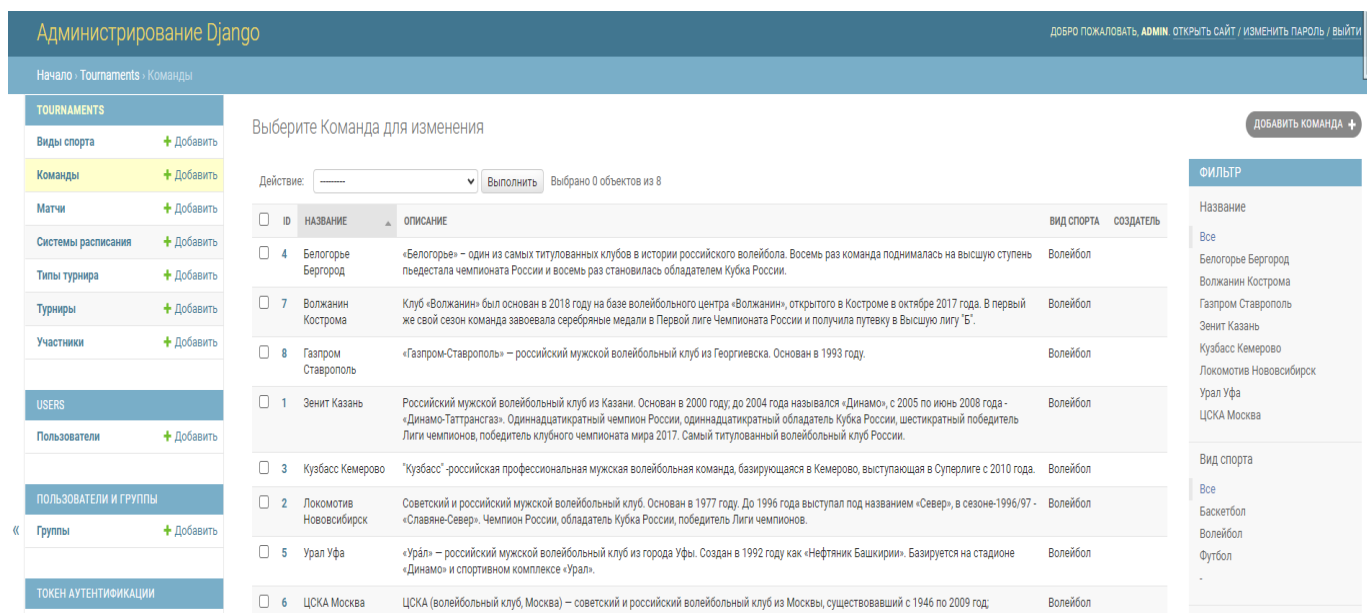


Рисунок 33 – Административный интерфейс для Team

### 3.4.7 Разработка авторизации

Авторизация была разработана с помощью библиотеки «djoser».

Эта библиотека парой строк кода позволяет добавить необходимый функционал авторизации, в том числе с использованием токена. Чтобы использовать эту библиотеку, достаточно добавить новые URL в файл urls.py, где находятся главные endpoints приложения (Рисунок 34)

```
urlpatterns = (  
    path('', include(router.urls)),  
    path('auth/', include('djoser.urls')),  
    path('auth/', include('djoser.urls.authtoken')),  
    path('round_robin/', views.RoundRobin.as_view()),  
    path('create_tournament/', views.CreateTournament.as_view())  
)
```

Рисунок 34 – URL API

### 3.4.8 Разработка API

API – Application Programming Interface. API нужен для того чтобы приложение могло взаимодействовать с другими сервисами, а так же позволяет другим разработчикам пользоваться сервисом.

API было спроектировано с использованием библиотеки «django-rest-framework». Это библиотека позволяет быстро проектировать API, при этом сохраняя качество кода. Для каждой модели нужны модули `serializer.py`, `view.py` и `urls.py`. Ниже на (рис. 35) представлен API для модуля Tournament.

```

class TournamentListSerializer(serializers.ModelSerializer):
    logo = Base64ImageField()
    country = ShortCountrySerializer(read_only=True)
    city = ShortCitySerializer(read_only=True)
    sport_type = ShortSportTypeSerializer(read_only=True)
    tournament_type = ShortTournamentTypeSerializer(read_only=True)
    schedule_system_type = ShortScheduleSystemTypeSerializer(read_only=True)
    organizer = ShortUserSerializer(read_only=True)
    teams_amount = serializers.SerializerMethodField()

    class Meta:
        model = models.Tournament
        fields = (
            'id',
            'title',
            'description',
            'logo',
            'organizer',
            'country',
            'city',
            'sport_type',
            'tournament_type',
            'schedule_system_type',
            'count_teams',
            'teams_amount',
            'start_date',
            'end_date'
        )

    def get_teams_amount(self, tournament):
        return tournament.teams.count()

```

Рисунок 35 – API для модуля Tournament

Были спроектированы API для каждой модели аналогичным образом. На (рис. 36) представлен веб интерфейс, позволяющий увидеть API.

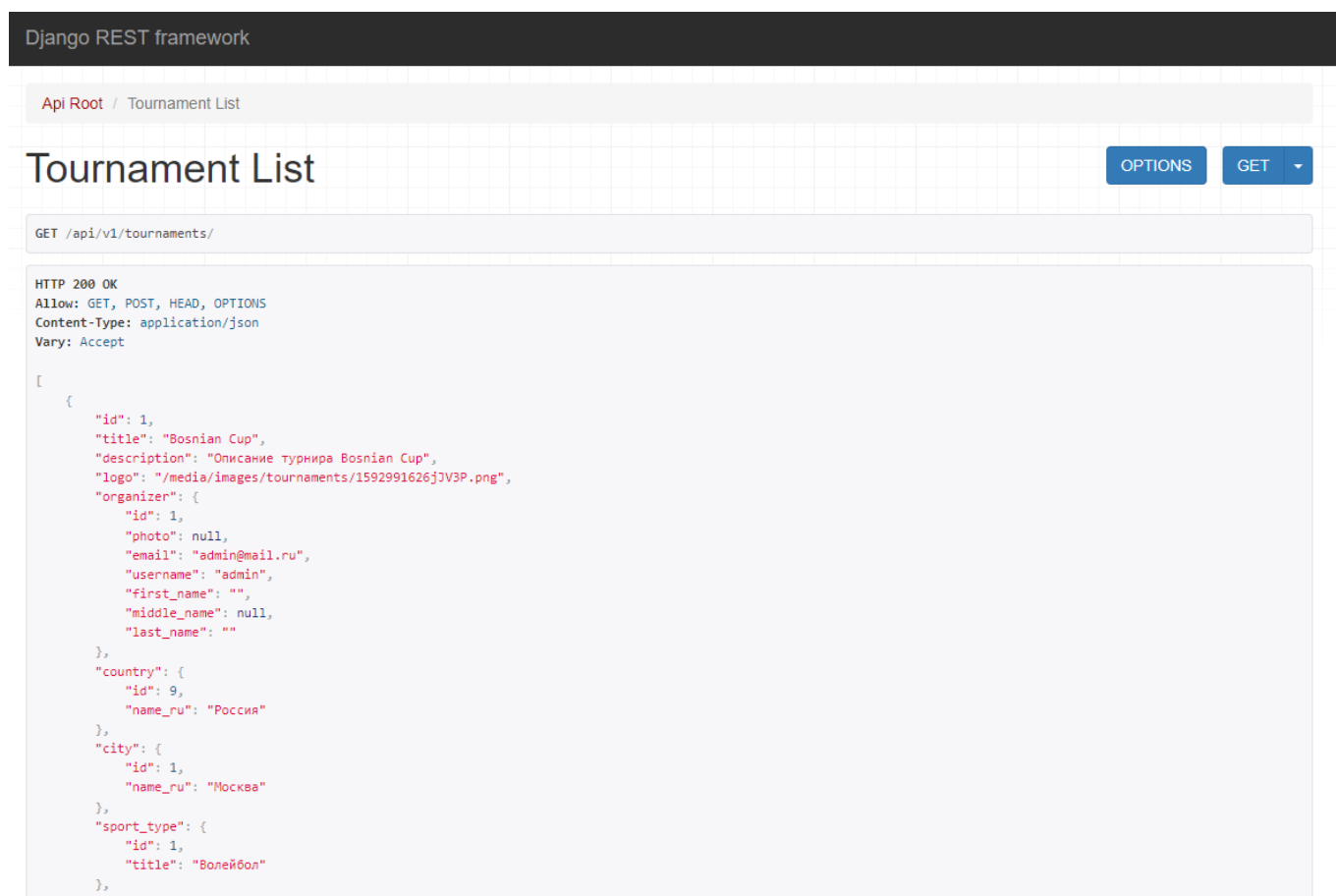


Рисунок 36 – Веб интерфейс API, для модели Tournament Lis

### 3.4.9 Разработка модуля генерации расписания

Модуль генерации расписания является одной из основных возможностей приложения.

Для генерации расписания была создана форма, представленная на рисунке 37.

Формирование турнирной сетки

	Волжанин Кострома
	Газпром Ставрополь
	Урал Уфа
	ЦСКА Москва

Сформировать сетку

Закрыть Сохранить

Рисунок 37 – Форма для генерации расписания

После нажатия на кнопку «Сформировать сетку» происходит отправка запроса на backend по endpoint'у «/api/v1/round\_robin/» с отправкой списка идентификаторов команд, после чего в ответном сообщении возвращаются сгенерированные объекты матчей с помощью алгоритма round\_robin (Рис. 38, Рис. 39, Рис. 40, Рис. 41)

```

@staticmethod
def __round_robin(team_ids, day_off='Day off', double=False):
    if len(team_ids) % 2:
        team_ids.append(day_off)

    n = len(team_ids)
    matches = []
    fixtures = []
    for fixture in range(1, n):
        for i in range(int(n / 2)):
            matches.append([team_ids[i], team_ids[n - 1 - i]])
            team_ids.insert(1, team_ids.pop())
            fixtures.insert(int(len(fixtures) / 2), matches)
            matches = []
    if double:
        count_matches = len(fixtures)
        for i in range(count_matches):
            fixtures.append([m[::-1] for m in fixtures[i]])
    result_matches = []
    for i, matches in enumerate(fixtures):
        for match in matches:
            template = {
                "owner": match[0],
                "guest": match[-1],
                "datetime": None,
                "owner_points": 0,
                "guest_points": 0,
                "round": i
            }
            result_matches.append(template)
    return fixtures

```

Рисунок 38 – Код алгоритма round\_robin

```

template = {
    "datetime": None,
    "guest": None,
    "guest_points": 0,
    "owner": None,
    "owner_points": 0,
    "round": 0,
    "tournament": request.data.get('tournament_id'),
}
result = []
for i, matches in enumerate(
    self.__round_robin(
        request.data.get('team_ids'),
        request.data.get('day_off', 'Dat off'),
        request.data.get('double', False)
    )
):
    for j, match in enumerate(matches):
        result.append({
            **template,
            "guest": match[0],
            "owner": match[-1],
            "round": i + 1
        })
return Response(result, status=status.HTTP_200_OK)

```

Рисунок 39 – Генерация объектов матчей

POST
⌵
http://127.0.0.1:8000/api/v1/round\_robin/

Params
Authorization
Headers (9)
Body ●
Pre-request

● none
● form-data
● x-www-form-urlencoded
● raw

```

1  {
2    "team_ids": [1,2,3,4],
3    "tournament_id": 1
4  }

```

Body
Cookies
Headers (11)
Test Results

Pretty
Raw
Preview
Visualize
JSON ⌵
⌵

```

1  [
2    {
3      "datetime": null,
4      "guest": 1,
5      "guest_points": 0,
6      "owner": 3,
7      "owner_points": 0,
8      "round": 1,
9      "tournament": 1
10   },
11   {
12     "datetime": null,
13     "guest": 4,
14     "guest_points": 0,
15     "owner": 2,
16     "owner_points": 0,

```





Рисунок 40 – Пример ответа по endpoint'у «/api/v1/round\_robin/»







Сформировать сетку

№	Команда	1	2	3	4
1	Волжанин Кострома	-	0 - 0	0 - 0	0 - 0
2	Газпром Ставрополь	0 - 0	-	0 - 0	0 - 0
3	Урал Уфа	0 - 0	0 - 0	-	0 - 0
4	ЦСКА Москва	0 - 0	0 - 0	0 - 0	-

#### Раунд 1

 Урал Уфа	VS	 Волжанин Кострома
 Газпром Ставрополь	VS	 ЦСКА Москва

#### Раунд 2


 Газпром Ставрополь	VS	 Волжанин Кострома
 ЦСКА Москва	VS	 Урал Уфа

#### Раунд 3

Рисунок 41 – Вывод сетки расписания пользователей

После того, как пользователь посмотрит сформированную сетку турнира, он может отправить ее на сохранение в базу данных.

Результат выполненных действий можно увидеть на странице турнира в соответствующих вкладках (Рис. 42, Рис. 43)



Italian Serie C - Lombardy B

01.01.2023 - 12.05.2024

Регламент: Чемпионат


Команды

Турнирная сетка

Расписание

№	Команда	1	2	3	4
1	Волжанин Кострома	-	0 - 0	0 - 0	0 - 0
2	Газпром Ставрополь	0 - 0	-	0 - 0	0 - 0
3	Урал Уфа	0 - 0	0 - 0	-	0 - 0
4	ЦСКА Москва	0 - 0	0 - 0	0 - 0	-

Рисунок 42 – Турнирная сетка



**Italian Serie C - Lombardy B**





01.01.2023 - 12.05.2024

Россия Новосибирск

Вид спорта: Волейбол

Команды
Турнирная сетка
Расписание

**Раунд 1**

 Урал Уфа	VS	 Волжанин Кострома
 Газпром Ставрополь	VS	 ЦСКА Москва

**Раунд 2**



 Газпром Ставрополь	VS	 Волжанин Кострома
--	----	---

Рисунок 43 – Расписание турнирной сетки

## ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были рассмотрены и разработаны ключевые аспекты информационной системы для управления сервисом организации спортивных мероприятий. Была определена архитектура системы и выделены основные модули, необходимые для ее правильной работы.

Один из основных модулей сервиса - модуль управления расписанием, был подробно описан. Он включает в себя системы составления расписаний соревнований, в том числе круговую систему. Был представлен алгоритм планирования, основанный на методе круга и использовании таблиц Бергера. Этот подход позволяет эффективно планировать спортивные мероприятия, обеспечивая справедливость и равные возможности для всех участников.

В целом, проведенное исследование дает понимание о методах и подходах к разработке веб-сервиса для организации спортивных мероприятий, а также предоставляет базовые принципы и инструменты для управления расписанием соревнований.

Дальнейшие исследования в этой области могут включать более глубокий анализ и сравнение различных методов составления расписаний, а также разработку новых алгоритмов учета сложных ограничений и условий проведения соревнований.

Результатом выпускной квалификационной работы является разработанный веб-сервис по управлению спортивными соревнованиями с использованием различных алгоритмов планирования расписаний.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) The library for web and native user interfaces: официальный сайт. – URL: <https://react.dev/> (дата обращения 08.11.2023).
- 2) The web framework for perfectionists with deadlines / Django: официальный сайт. – URL: <https://www.djangoproject.com/> (дата обращения 05.11.2023).
- 3) Проектирование веб-API / сост. Арно Лоре. - ДМК-Пресс, 2020 г. – 440 с. – ISBN 978-5-97060-861-6
- 4) Challonge - Турнирные сетки: официальный сайт. – URL: <https://challonge.com/ru/about> (дата обращения 29.10.2023).
- 5) Tournament Scheduler - Easily create a round robin tournament schedule: официальный сайт. – URL: <https://challonge.com/ru/about> (дата обращения 08.11.2023).
- 6) Генератор турнирных сеток — турниры: официальный сайт. – URL: <https://goodgame.ru/cups/bracket/generator> (дата обращения 01.11.2023).
- 7) Tournament (graph theory) - Wikipedia: официальный сайт. – URL: [https://en.wikipedia.org/wiki/Main\\_Page](https://en.wikipedia.org/wiki/Main_Page) (дата обращения 03.11.2023).
- 8) Round-robin tournament – Wikipedia: официальный сайт. – URL: [https://en.wikipedia.org/wiki/Round-robin\\_tournament](https://en.wikipedia.org/wiki/Round-robin_tournament) (дата обращения 03.11.2023).
- 9) Круговой турнир - Round-robin tournament: официальный сайт. – URL: [https://ru.wikibrief.org/wiki/Round-robin\\_tournament](https://ru.wikibrief.org/wiki/Round-robin_tournament) (дата обращения 04.11.2023).
- 10) Tournament Service: официальный сайт. – URL: <https://tournamentservice.net/charts.php?article=311> (дата обращения 30.10.2023).
- 11) Postgres. Первое знакомство. Версия 15 / сост. П. Лузанов, Е. Рогов, И. Лёвшин - ДМК-Пресс, 2023 г. – 178 с. – ISBN 978-5-6045970-1-9
- 12) Visual Studio Code - Code Editing: официальный сайт. – URL: <https://code.visualstudio.com> (дата обращения 04.11.2023).
- 13) RAMUS / ramus: официальный сайт. – URL: <https://ramussoftware.com/> (дата обращения 03.11.2023).
- 14) Моделирование и анализ систем. I DEF-технологии: практикум / С. В. Черемных, И.О. Семенов, В.С. Ручкин. - М.: Финансы и статистика, 2006. - 192 с: ил. - (Прикладные информационные технологии). ISBN 5-279-02564-X

15) Чистая архитектура. Искусство разработки программного обеспечения /  
сост. Мартин Роберт - ДМК-Пресс, 2022 г. – 352 с. – ISBN 978-5-4461-0772-8

## ПРИЛОЖЕНИЕ А

Исходный код алгоритма round\_robin

```
def __round_robin(team_ids, day_off='Day off', double=False):
    if len(team_ids) % 2:
        team_ids.append(day_off)

    n = len(team_ids)
    matches = []
    fixtures = []
    for fixture in range(1, n):
        for i in range(int(n / 2)):
            matches.append([team_ids[i], team_ids[n - 1 - i]])
            team_ids.insert(1, team_ids.pop())
            fixtures.insert(int(len(fixtures) / 2), matches)
            matches = []

    if double:
        count_matches = len(fixtures)
        for i in range(count_matches):
            fixtures.append([m[::-1] for m in fixtures[i]])

    result_matches = []
    for i, matches in enumerate(fixtures):
        for match in matches:
            template = {
                "owner": match[0],
                "guest": match[-1],
                "datetime": None,
                "owner_points": 0,
                "guest_points": 0,
```

```
        "round": i
    }
    result_matches.append(template)

return fixtures
```