МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Web-приложение для мониторинга и анализа результатов спортивных соревнований

Студент _____ Хе Артур Викторович
(подпись, дата) (ФИО полностью)

Руководитель _____ Князев В.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Макарычев П.П.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2022

Федеральное государственное бюджетное образовательное учреждение высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____П.П. Макарычев
«__»_____20__ г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Хе Артур Викторович гр. 18ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы Web-приложение для мониторинга и анализа результатов спортивных соревнований

Тема утверждена приказом ПГУ № 327/О от "18" апреля 2022

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н. _____

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Разработать web-приложение, обеспечивающее мониторинг результатов спортивных соревнований и анализ по каждому виду спорту.
2) Среда выполнения приложения: операционная система Windows 10
Среда выполнения сервера: операционная система Windows 10
Языковые средства разработки: Python, UML, JavaScript, SQL
Инструментальные средства разработки: PyCharm, PostgreSQL, Microsoft Visual Studio Code , Microsoft Project, ArgoUML, Visual Paradigm, Figma
Технологии разработки: RUP, ООП, Django Rest Framework
3) Основные функции web-приложения:
а) просмотр текущих результатов;
б) просмотр анализа по каждому виду спорта
в) добавление события;
г) формирование результата;
г) изменение результата

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи

2) Проектирование

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	*Анализ предметной области постановка задачи*	15%	13.04.2022–26.04.2022	
2	*Проектирование*	25%	27.04.2022–08.05.2022	
3	*Реализация*	40%	08.05.2022–30.05.2022	
4	*Контроль качества программного обеспечения*	5%	31.05.2022–02.06.2022	
5	*Оформление пояснительной записки*	15%	03.06.2022–17.06.2022	

Дата выдачи задания "22" декабря 2021

Руководитель бакалаврской работы _____

(подпись, дата)

Задание к исполнению принял студент _____

(подпись, дата)

Работу к защите допустить

Декан факультета _____

(подпись, дата)

Реферат

Пояснительная записка содержит 114 листов, 58 рисунков, 33 таблицы, 36 использованных источников, 5 приложений.

WEB-ПРИЛОЖЕНИЕ, СПОРТИВНЫЕ СОРЕВНОВАНИЯ, PYTHON, POSTGRESQL, JAVASCRIPT, RUP, UML

Объектом разработки является web-приложение для мониторинга и анализа результатов спортивных соревнований.

Цель работы – разработка web-приложения для мониторинга и анализа результатов спортивных соревнований, позволяющее добавлять в систему новые события, формировать результаты матчей, а также просматривать результаты отдельных встреч и анализировать статистические данные по отдельным видам спорта.

Разработка серверной части происходила на языке программирования Python с использованием фреймворка Django. В качестве среды разработки использовалась PyCharm Community. В качестве СУБД была использована СУБД PostgreSQL. Для взаимодействия с СУБД использовалось средство Django ORM.

Разработка клиентской части приложения производилась на языке Javascript с использованием библиотеки ReactJS, операционная система Windows 10.

В результате выполнения выпускной квалификационной работы разработано web-приложение для мониторинга и анализа результатов спортивных соревнований, отвечающее всем требованиям технического задания.

Содержание

Введение 7

1 Анализ предметной области и требований web-приложения для мониторинга и анализа результатов спортивных событий 9

1.1 Анализ предметной области и постановка задач 9

1.2 Анализ аналогичных решений 10

1.2.1 Анализ web-приложения «Чемпионат» 11

1.2.2 Анализ web-приложения «Flashscore» 12

1.2.3 Анализ web-приложения «Спорт-экспресс» 13

1.2.4 Сравнение аналогов 14

1.3 Анализ целевой аудитории 14

1.4 Анализ функциональных требований 15

1.5 Анализ нефункциональных требований 20

1.6 Планирование и расчет бюджета для web-приложения для мониторинга и анализа результатов соревнований 21

2 Проектирование web-приложения 29

2.1 Выбор технологий, инструментальных и языковых средств разработки 29

2.2 Проектирование архитектуры программного обеспечения 30

2.3 Проектирование структур базы данных 31

2.3.1 Концептуальная модель базы данных 31

2.3.2 Логическая модель базы данных 33

2.4 Проектирование программных средств 37

2.5 Проектирование пользовательского интерфейса 38

3 Реализация web-приложения 42

3.1 Реализация серверной части web-приложения 42

3.1.1 Реализация базы данных 42
3.1.2 Реализация серверных компонентов 46
3.2 Реализация клиентской части web-приложения 49
4 Тестирование web-приложения 51
4.1 Выбор режима тестирования приложения 51
4.2 Тестирование серверной части приложения информационной системы .. 52
4.3 Тестирование клиентской части приложения информационной системы 53
4.4 Метрики кода 54
4.4.1 Метрика SLOC 54
4.4.2 Метрика Холстеда 55
Заключение 58
Список использованных источников 59
Приложение А. Глоссарий 63
Приложение Б. Исходный код серверной части 65
Приложение В. Исходный код web-приложения 77
Приложение Г. Результаты тестирования web-приложения 100
Приложение Д. Рисунки и диаграммы 110

Введение

С каждым годом информационные технологии все глубже проникают в различные сферы человеческой жизни. Сложно представить какую-либо деятельность без использования программных продуктов. Начиная приложениями для документооборота, заканчивая сложными программно-аппаратными комплексами. В связи с этим все страны миры стараются уделять внимание сфере информационных технологий, наша страна не является исключением. Об этом может свидетельствовать утверждение в 2017 году программы «Цифровая экономика Российской Федерации», основной задачей которой является «развитие цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост и экономический суверенитет» [1]. По данным отчета Digital 2022: The Russian Federation о состоянии социальных сетей и интернета в Российской Федерации на январь 2022 года в России насчитывается 129,8 млн интернет-пользователей. Интернетом пользуются от 89% от общей численности населения. Ежедневно пользователь проводит в сети интернет 7 часов 50 минут, из которых 4 часа 11 минут через компьютеры и планшеты [2]. За последнее десятилетие резко увеличилось web-приложений, интернет-сервисов. «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы» предусматривает широкомасштабное применение в информационном обществе web-сервисов сети «Интернет» [3].

За последнее десятилетие все большую популярность набирает спорт и физическая культура. Так, по данным государственной статистики доля граждан, систематически занимающихся спортом и физической культурой, составляет около 43%. Для сравнения в 2008 году это значение было равно 15,9 % [4]. В связи с этим выросло число спортивных событий. Данные мероприятия в нашей стране проводятся на всех уровнях. Начиная от дворового уровня, заканчивая профессиональным. Профессиональные

спортивные команды, школы, университеты, предприятия соревнуются и выявляют сильнейших в различных спортивных турнирах. Любое спортивное события сопровождается огромным количеством информации. Например, данные участников, календарь игр, результаты отдельных встреч и т.д. На профессиональном уровне сбором данных, протоколированием и публикацией результатов занимаются группа ответственных за это людей, благодаря которым мы всегда в курсе всех результатов состязаний. Но на более низком уровне соревнований, к примеру первенство района, сбором данных и протоколированием занимается один человек, как правило это секретарь или судья, и результаты отдельных матчей нигде не публикуются. Если же участники этих соревнований могут хоть как-то следить за ходом турнира (положение команды в турнире, результаты встреч команд соперниц), то их родственникам, друзьям это делать приходится из уст самих участников или присутствовать на спортивном мероприятии, что не всегда представляется возможным. Информация о прошедших соревнований также необходима для организаторов состязаний. Например, для выявления лидеров и оутсайдеров для составления турнирной сетки следующих соревнований. Все вышеописанные процессы можно представить в другой, более удобной форме в настоящее время – цифровой.

При наличии web-приложения, позволяющего представлять информацию о спортивных турнирах, протоколировать матчи можно упростить процесс проведения спортивных соревнований, сделать его более удобным как для организаторов турниров, так и для самих спортсменов.

Отталкиваясь с вышесказанного, актуальной целью выпускной квалификационной работы является разработка web-приложения для мониторинга и анализа результатов спортивных соревнований.

1 Анализ предметной области и требований web-приложения для мониторинга и анализа результатов спортивных событий

1.1 Анализ предметной области и постановка задач

В настоящее время спорту и физической культуре уделяется большое внимание. Проводятся соревнования по различным видам спорта на всех уровнях. Начиная от соревнований среди дворовых команд, заканчивая чемпионатами мира и олимпиадой. Пропагандируя спорт, мы пропагандируем здоровый образ жизни, тем самым закладываем надежный фундамент для развития здорового общества.

Всего в 2013 – 2020 гг. на развитие спортивной индустрии в рамках государственной программы «Развитие физической культуры и спорта» выделено 1 717 млрд рублей, на рисунке 1 более подробно представлено распределение денежных средств, выделенных на развитие спортивной индустрии в России. Значительная часть этих средств идет на развитие спортивной инфраструктуры. Объем производства отечественной продукции спортивной индустрии в России вырос на 6,6% в 2017 году по отношению к 2016 году и составил 27,32 млрд. рублей [5].

Рисунок 1 – Распределение денежных средств в спортивной индустрии в России.

Можно сделать вывод, что спортивный сектор в России активно развивается и значительная доля средств тратится в организацию и проведение спортивных мероприятий. Это говорит о том, что разработка приложений,

предметной областью которых является спортивная отрасль является актуальной на сегодняшний день [6].

При проведении соревнований секретарь заносит в протокол результаты отдельных матчей. По результатам матчей можно следить за прогрессом участника на турнире. Например, если турнир проводится по олимпийской системе, то можно отслеживать движение участника по турнирной сетке [7]. При проведении многих соревнований протоколы ведутся в бумажном виде, как и вся информация о турнире (турнирная сетка, результаты матчей, расписание матчей). Хранится данная информация у организаторов.

Необходимо разработать приложение, чтобы все участники, а также их болельщики, могли быть в курсе всех событий, происходящих на турнире. Вся информация должна быть представлена в электронной форме, доступна всем посредством интернет-соединения и с любой платформы. Для этого подойдет web-приложение.

Таким образом, необходимо разработать web-приложение для мониторинга и анализа результатов спортивных соревнований, позволяющее предоставлять информацию о мероприятии, результатах матчей, статистику события. Данное приложение будет являться многопользовательским, будут существовать учетные записи организаторов турниров, заполняющих всю информацию о турнире в базу данных, и секретарей матчей – основной функцией которых будет являться создание событий и добавления результатов матчей. Для обычных пользователей, основной возможностью которых будет являться получение информации наличие учетной записи не требуется.

1.2 Анализ аналогичных решений

На сегодняшний день существуют приложения со схожим функционалом, позволяющие получать информацию о результатах матчей. Проведем сравнительный анализ наиболее известных программных продуктов данного сектора.

1.2.1 Анализ web-приложения «Чемпионат»

Web-приложение «Чемпионат» – российский спортивный интернет-портал, публикующий новости, результаты матчей, интервью и аналитические материалы, ведущий онлайн-трансляции, размещающий фото и видеоматериалы [8]. Главный экран приложения представлен на рисунке 2.

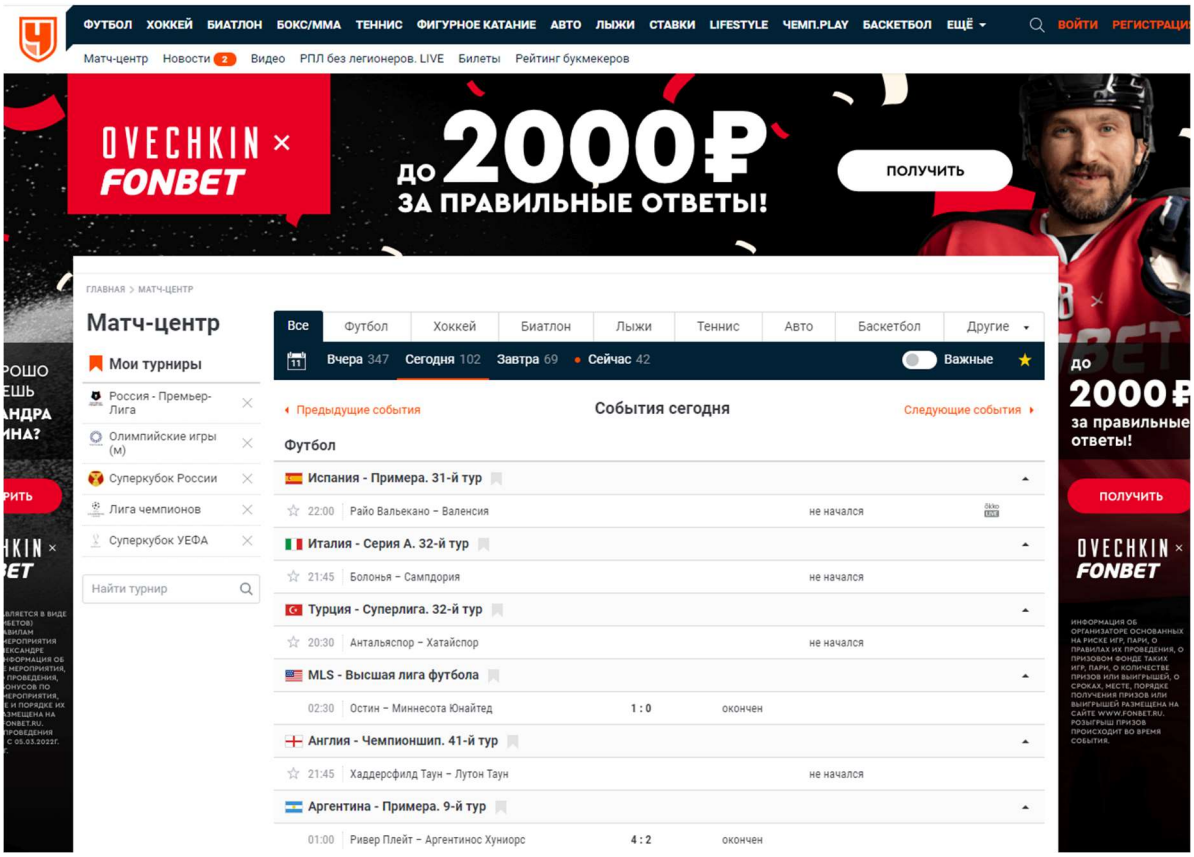


Рисунок 2 – Главный экран приложения «Чемпионат»

Достоинство приложения является то, что он предоставляет информацию не только о результатах матчей, но и новостные сводки, турнирные таблицы, результаты предыдущих встреч команд, наличие трансляций.

Недостатком данного приложения заключается в том, что он публикует данные только крупных, официальных спортивных событий. И как следствие, в нем нельзя создать свой турнир.

1.2.2 Анализ web-приложения «Flashscore»

FlashScore – агрегатор спортивный событий, показывает статистику игр, команд, а также отправляет уведомление о приближающихся играх [9]. Главный экран приложения представлен на рисунке 3.

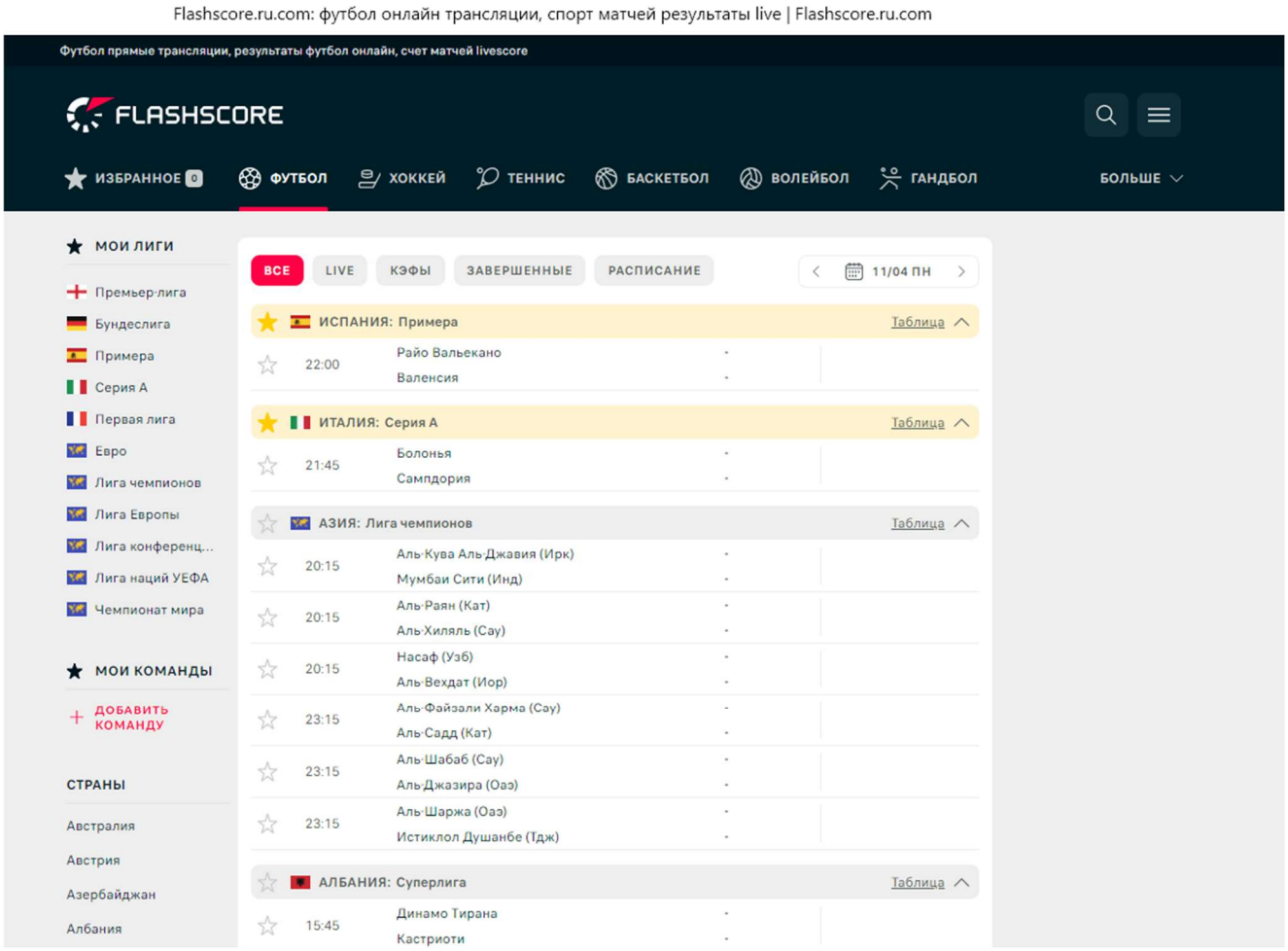


Рисунок 3 – Главный экран приложения «Flashscore»

Достоинство приложения является то, что он предоставляет информацию не только о результатах матчей, но и турнирные таблицы, сетки результаты предыдущих встреч команд, просмотр хода матчей, отправка уведомлений о приближающихся играх.

Недостатком данного приложения, как и с предыдущем приложением, заключается в том, что он публикует данные только крупных, официальных спортивных событий. И как следствие, в нем нельзя создать свой турнир.

1.2.3 Анализ web-приложения «Спорт-экспресс»

Спорт-экспресс – интернет-портал одноименной газеты, публикующий новости, результаты матчей, интервью и аналитические материалы, ведущий онлайн-трансляции [10]. Главный экран приложения представлен на рисунке 4.

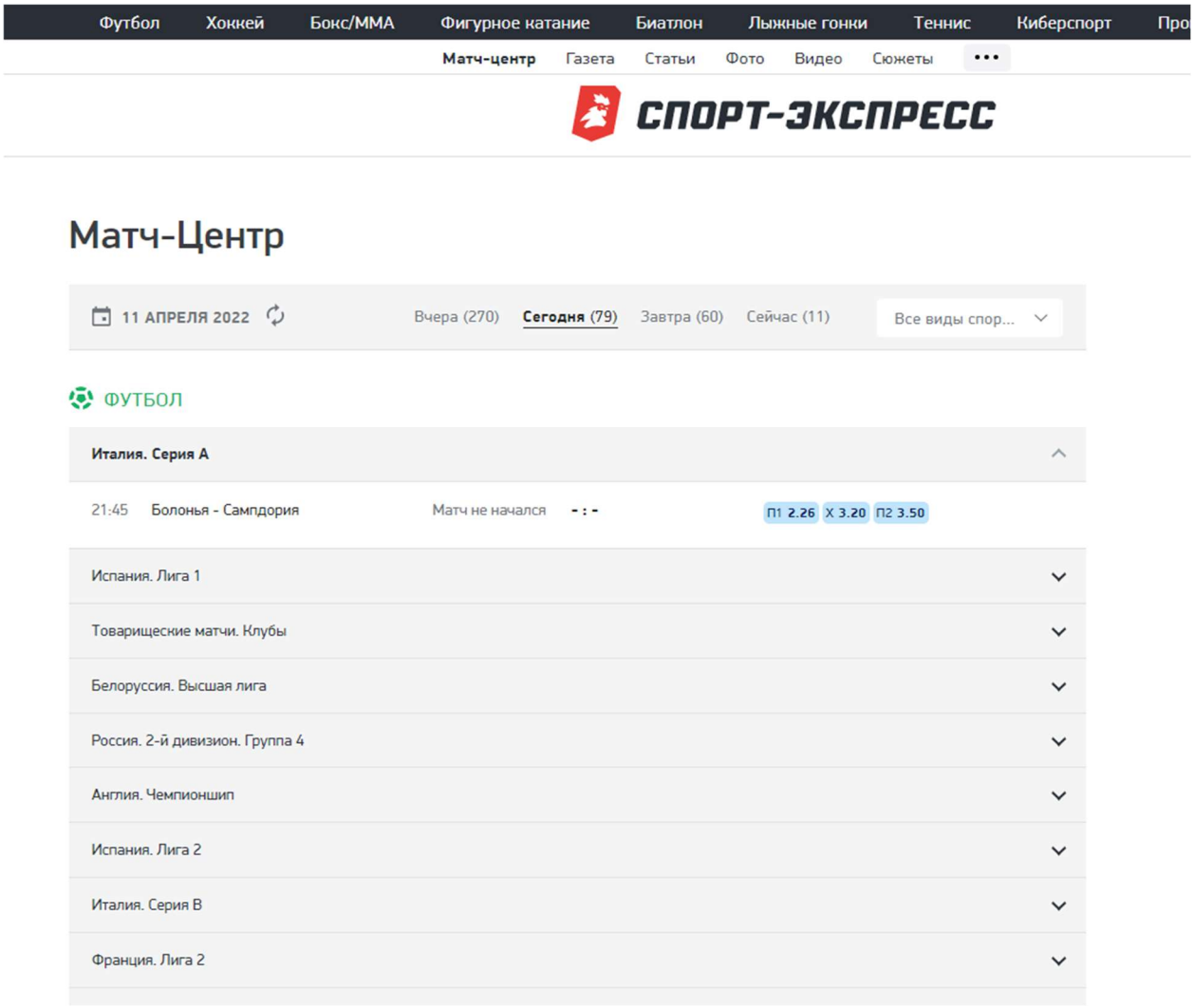


Рисунок 4 – Главный экран приложения «Спорт-экспресс»

Достоинство приложения является то, что он предоставляет информацию не только о результатах матчей, но и новостные сводки, турнирные таблицы, результаты предыдущих встреч команд, наличие трансляций.

Недостатком данного приложения заключается в том, что он публикует данные только крупных, официальных спортивных событий. И как следствие, в нем нельзя создать свой турнир.

1.2.4 Сравнение аналогов

Сравнительный анализ рассмотренных выше интернет-библиотек представлен в таблице 1.

Таблица 1 – Сравнение аналогов

Приложение	Просмотр результатов матчей	Уровень событий	Возможность создания своих событий	Система поиска
Чемпионат	Да	Крупные официальные турниры	Отсутствует	Понятная
Flashscore	Да	Крупные официальные турниры	Отсутствует	Понятная
Спорт-экспресс	Да	Крупные официальные турниры	Отсутствует	Понятная
Разрабатываемое приложение	Да	Все	Существует	Понятная

Таким образом, в результате анализа систем аналогичного назначения было выявлено, что у всех приложений отсутствует возможность создание своих событий, также в них освещаются только события крупного масштаба. В связи с этим было принято решение о разработке собственного приложения, которое будет лишено недостатков, выявленных у рассмотренных выше web-приложений.

1.3 Анализ целевой аудитории

Целевой аудиторией для разрабатываемого приложения является организаторы соревнований, спортсмены, тренера, а также болельщики.

Ниже представлен график использования браузеров в России за период 11 – 17 апреля 2022 по данным веб-сервиса «Яндекс.Радар».

По рисунку 5 можно сделать вывод, что самые популярные браузеры – это Google Chrome, Яндекс.Браузер, Safari.

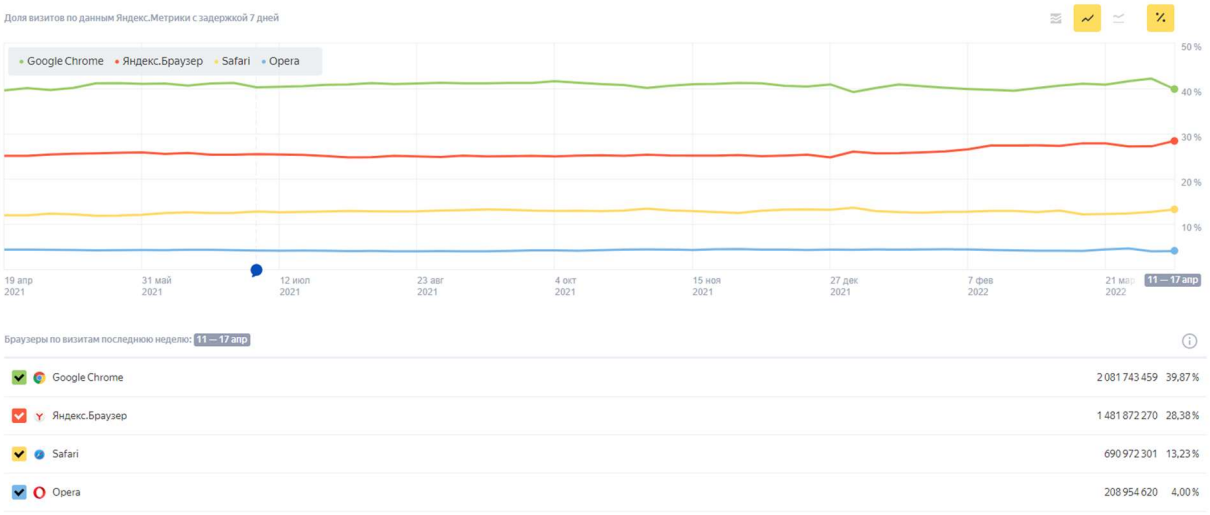


Рисунок 5 – График использования браузеров в России

Таким образом, наше разрабатываемое web-приложение должно иметь поддержку этих браузеров.

1.4 Анализ функциональных требований

Разрабатываемое приложение предназначено для следующих категорий пользователей: администратор и простой пользователь. Основной задачей администратора будет являться создания события и занесения результатов матчей. Простому пользователю будет доступна возможность просмотра различной информации о турнире.

В результате анализа предметной области и аналогичных программных продуктов были выделены функциональные требования [11]. Для администратора были выделены следующие функции:

– добавить событие;
– сформировать результат;
– изменить результат;
– войти в систему;
– завершить сеанс.

Для пользователя были определены следующие функции:

– посмотреть анализ вида спорта;

– посмотреть текущие результаты.

Диаграмма вариантов использования представлена на рисунке 6.

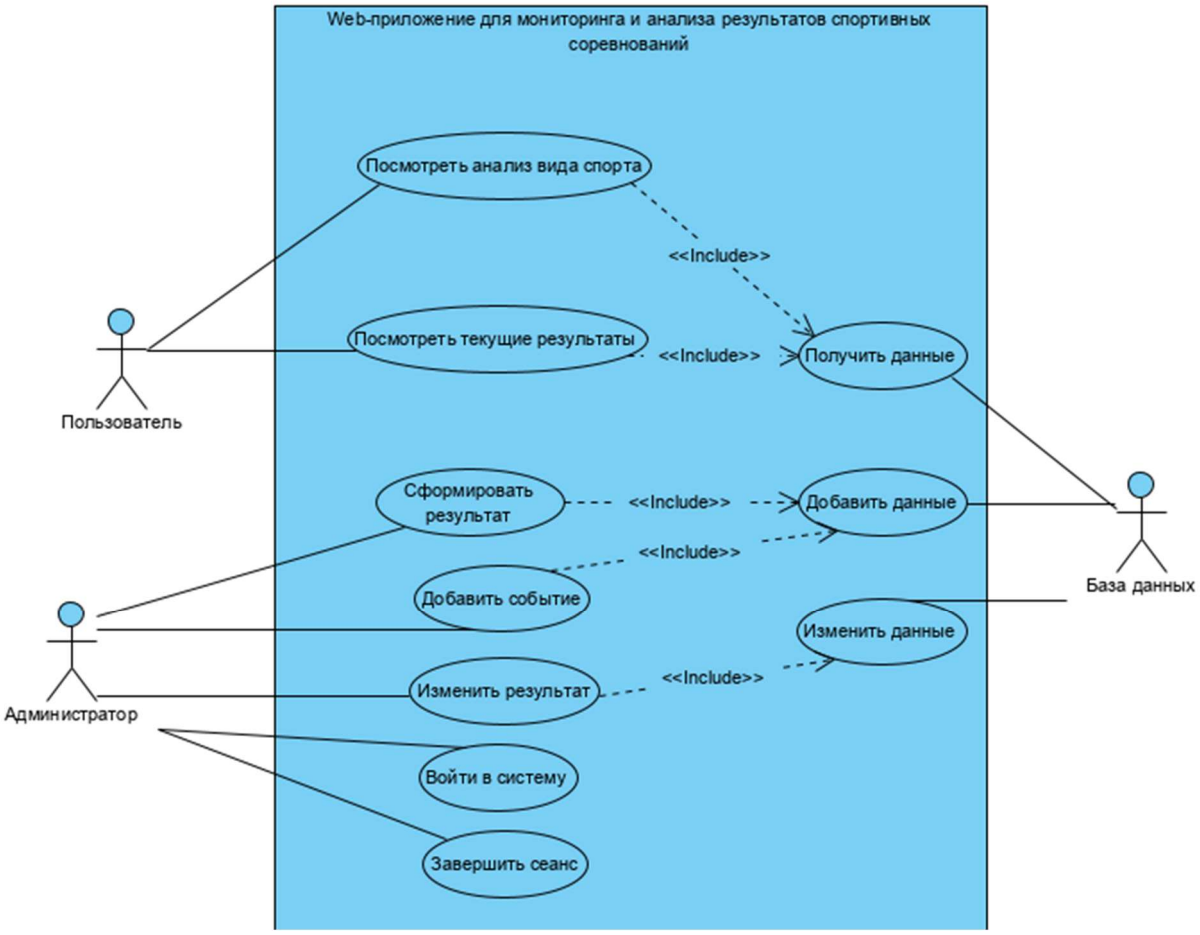


Рисунок 6 – Диаграмма вариантов использования.

Отдельно представим диаграмму варианта использования «Добавить событие» на рисунке 7.

На рисунке 8 описана спецификация для варианта использования «Добавить событие», который предназначен для администратора. При выборе схемы проведения соревнований администратору предоставляется на выбор следующие схемы: олимпийская система, круговой турнир, групповой турнир с последующим плэй-офф.

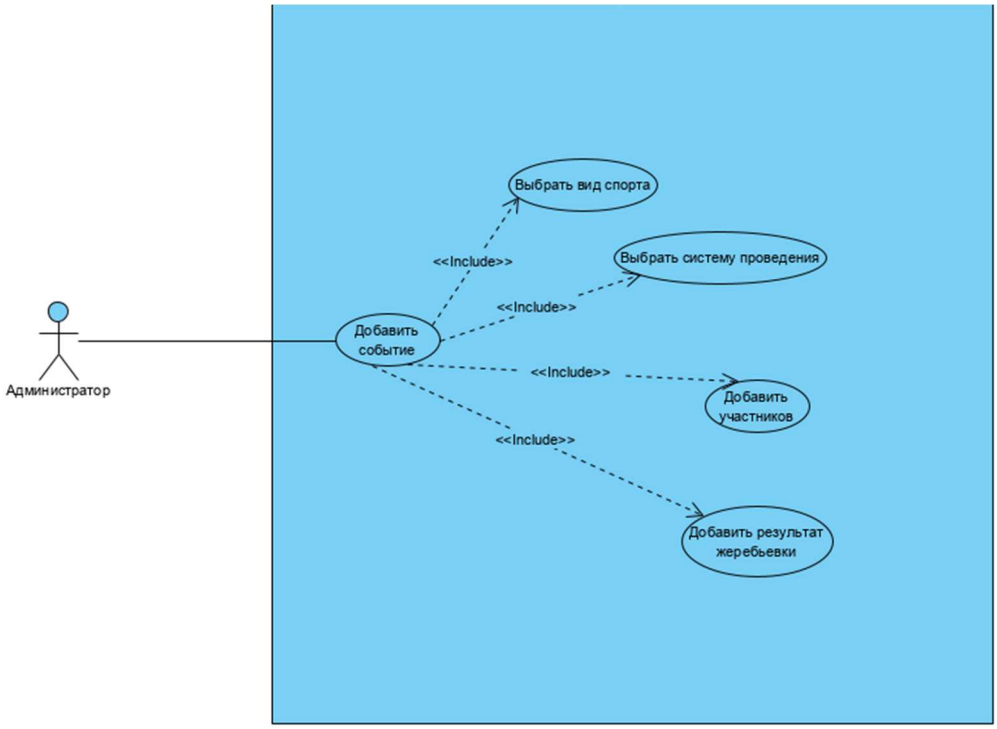


Рисунок 7 – Диаграмма варианта использования «Добавить событие»

Прецедент: Добавить событие
ID:5
Краткое описание: Добавление нового события.
Главные актеры: Администратор.
Второстепенные актеры: Нет.
Предусловия: 1. Администратор вошел в веб-приложение.
Основной поток: 1. Администратор вводит данные о событии (название, вид спорта, описание). 2. Администратор выбирает систему проведения соревнований. 3. Администратор добавляет участников Администратор добавляет результаты жеребьевки
Постусловия: 1. В базу данных добавлено новое событие.
Альтернативные потоки: Нет

Рисунок 8 – Спецификация для варианта использования «Добавить событие»

Построим диаграмму состояний и переходов для объекта «событие». Данная диаграмма представлена на рисунке 9.

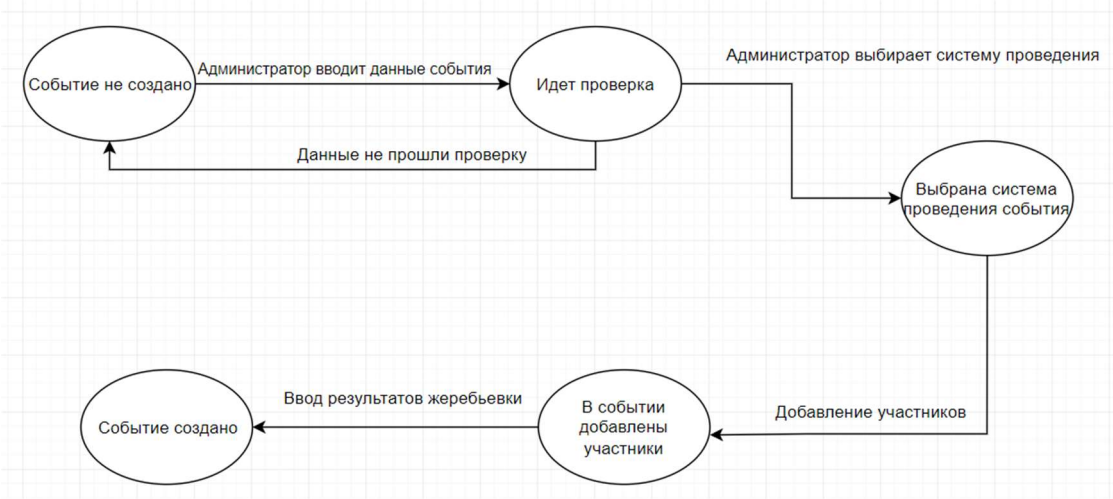


Рисунок 9 – Диаграмма состояний и переходов для объекта «Событие»

На рисунке 10 описана спецификация для варианта использования «Посмотреть текущий результат», предназначенный для простого пользователя. Для поиска результата матча, пользователь должен найти в навигационном меню вид спорта, далее выбрать из списка, который подгружается из базы данных, событие. После чего будет загружен список матчей с результатами.

Прецедент: Посмотреть текущий результат
ID:3
Краткое описание: Пользователь просматривает текущий результат.
Главные актеры: Пользователь
Второстепенные актеры: Нет.
Предусловия: Пользователь открыл веб-приложение.
Основной поток: 1. Пользователь выбрал вид спорта 2. Пользователь выбрал событие 3. Пользователь выбрал категорию
Постусловия: Система показывает турнирную схему и список завершенных матчей.
Альтернативные потоки: Нет

Рисунок 10 – Спецификация для варианта использования «Посмотреть текущий результат»

На рисунке 11 описана спецификация для варианта использования «Посмотреть анализ развития вида спорта», предназначенный для простого пользователя. Для просмотра анализа, пользователь должен найти в навигационном меню пункт «Анализ», далее выбрать из списка, который подгружается из базы данных, вид спорта. После чего будет отображены данные по выбранному виду спорта.

Прецедент: Посмотреть анализ вида спорта.
ID: 1
Краткое описание: Просмотр анализа развития вида спорта.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: пользователь открыл приложение
Основной поток: 1. Пользователь открыл вкладку «Анализ» 2. Пользователь выбрал вид спорта
Постусловия: Система показывает диаграммы с количеством событий и средним числом участников за последние 5 лет
Альтернативные потоки: Нет.

Рисунок 11 – Спецификация для варианта использования «Посмотреть анализ развития вида спорта»

На рисунке 12 описана спецификация для варианта использования «Сформировать результат», предназначенный для администратора. Для формирования результата, администратор создает новый матч, далее должен добавить счет первого и второго участника, а затем сохранить данные, после чего в базе данных добавится новый результат матча.

На рисунке 13 описана спецификация для варианта использования «Изменить результат», предназначенный для администратора. Для изменения результата, администратор выбирает матч, далее должен изменить необходимые данные, а затем сохранить данные, после чего в базе данных изменится результат матча.

Прецедент: Сформировать результат.
ID: 2
Краткое описание: Запись результата матча.
Главные актеры: Администратор.
Второстепенные актеры: Нет.
Предусловия: Администратор вошел в систему.
Основной поток: 1. Администратор добавляет матч 2. Администратор добавляет счет первого участника 3. Администратор добавляет счет второго участника 4. Администратор сохраняет данные
Постусловия: В базу данных добавлен результат матча
Альтернативные потоки: Нет.

Рисунок 12 – Спецификация для варианта использования «Сформировать результат»

Прецедент: Изменить результат.
ID: 4
Краткое описание: Изменени результата матча.
Главные актеры: Администратор.
Второстепенные актеры: Нет.
Предусловия: Администратор вошел в систему.
Основной поток: 1. Администратор выбирает матч 2. Администратор изменяет данные 3. Администратор сохраняет данные
Постусловия: В базу данных изменен результат матча
Альтернативные потоки: Нет.

Рисунок 13 – Спецификация для варианта использования «Изменить результат»

Таким образом для каждого варианта использования были приведены спецификации.

1.5 Анализ нефункциональных требований

Разработка серверной части приложения должна быть выполнена на языке программирования Python с использованием фреймворка Django, в среде программирования PyCharm. Серверная часть должна включать в себя API (интерфейс прикладного программирования). Клиентскую часть необходимо написать на языке Javascript с использованием фреймворка React.js. В качестве СУБД (системы управления базами данных) должна выступать PostgreSQL. Работы с СУБД необходимо организовать при помощи

средств DjangoORM. Приложение должно работать в браузерах: Google Chrome, Яндекс.Браузер, Safari.

1.6 Планирование и расчет бюджета для web-приложения для мониторинга и анализа результатов соревнований

Успешное выполнение проекта, в том числе в сфере информационных технологий, трудно достичь без правильного планирования, процесс которого позволяет грамотно декомпозировать проект на задачи таким образом, чтобы добиться завершение проекта за минимальное возможное время, с минимально возможными затратами [12 – 16].

Для этого были определены задачи и ресурсы, которые необходимы для создания Web-приложения.

В таблице 2 представлены задачи, которые представляют этапы создания проекта.

Таблица 2 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ аналогичных решений	6 дня	1
4	Анализ правил проведения соревнований	3 дня	3
5	Принятие окончательного решения	1 день	4
6	Анализ требований завершен	-	5
7	Выбор средств реализации проекта	-	-
8	Анализ возможных программных средств	6 дня	1
9	Выбор программного средства для реализации	2 дня	8
10	Выбор средств завершен	-	9
11	Реализация проекта	-	-
12	Разработка структуры данных	2 дня	6; 10
13	Разработка серверной части приложения	-	-
14	Программирование моделей данных	3 дня	12
15	Программирование API-методов	7 дней	14
16	Разработка пользовательского интерфейса	-	-
17	Разработка макета интерфейса приложения	4 дня	12
18	Верстка приложения	6 дней	17
19	Реализация проекта завершена	-	15; 18
20	Отладка	-	-
21	Тестирование пользовательского интерфейса	3 дня	19
22	Исправление ошибок	10 дней	21
23	Тестирование завершено	-	22

Продолжение таблицы 2

№	Название задачи	Длительность	Предшественники
24	Внедрение программного средства в систему проведения соревнований города	-	-
25	Внедрение программного средства	10 дней	23
26	Поиск и исправления ошибок при внедрении	5 дней	25
27	Внедрение завершено	-	26
28	Конец проекта	-	27

Из таблицы, представленной выше видно, что для реализации проекта необходимо выполнить 28 задач.

В таблице 3 представлены ресурсы необходимые для реализации проекта.

Таблица 3 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист1	Трудовой	150 Р/ч	200 Р/ч
Программист2	Трудовой	250 Р/ч	300 Р/ч
Дизайнер	Трудовой	200 Р/ч	250 Р/ч
Интернет	Затраты		
Компьютер	Материальный	500 Р	

Распределение ресурсов по задачам представлено в таблице 4.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований		
3	Анализ аналогичных решений	Программист1 Компьютер Интернет	50% 1 шт
4	Анализ правил проведения соревнований	Программист1 Программист2 Компьютер	50% 50% 1 шт
5	Принятие окончательного решения	Программист1 Программист2	25% 25%
6	Анализ требований завершен		
7	Выбор средств реализации проекта		
8	Анализ возможных программных средств	Программист1 Программист2 Компьютер Интернет	50% 50% 1 шт
9	Выбор программного средства для реализации	Программист1 Программист2 Компьютер Интернет	50% 50% 1 шт
10	Выбор средств завершен		
11	Реализация проекта		
12	Разработка структуры данных	Программист1 Компьютер Интернет	100% 1шт

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
13	Разработка серверной части приложения		
14	Программирование классов моделей данных	Программист1 Компьютер Интернет	100% 1 шт
15	Программирование API-методов	Программист1 Компьютер Интернет	100% 1 шт
16	Разработка пользовательского интерфейса	-	-
17	Разработка макета интерфейса приложения	Дизайнер Компьютер Интернет	100% 1 шт
18	Верстка приложения	Программист2 Компьютер Интернет	100% 1 шт
19	Реализация проекта завершена		
20	Отладка		
21	Тестирование пользовательского интерфейса	Программист1 Программист2 Дизайнер Компьютер Интернет	20% 50% 30% 1 шт
22	Исправление ошибок	Программист1 Программист2 Компьютер Интернет	40% 60% 1 шт
23	Тестирование завершено		
24	Внедрение программного средства в систему проведения соревнований города		
25	Внедрение программного средства	Программист1 Программист2 Компьютер Интернет	50% 50% 1 шт
26	Поиск и исправления ошибок при внедрении	Программист1 Программист2 Компьютер Интернет	50% 50% 1 шт
27	Внедрение завершено		
28	Конец проекта		

Из представленных таблиц видно, что реализацией проекта будут заниматься 2 программиста и 1 дизайнер, которым для работы нужны компьютеры и доступ к интернету.

Для расчёта времени, необходимого для реализации проекта, построим сетевой график и определим критический путь.

Сетевой график представлен на рисунке Д.1.

По данному графику видно, что некоторые задачи выполняются параллельно.

Далее для вычисления критического пути необходимо для каждой задачи определить раннее и позднее время начала работ. Обозначим через t_i

время выполнения работы i, G – множество работ, непосредственно предшествующих работе i, H – множество работ, непосредственно следующих за работой i.

Ранее время определяется по формуле:

$$T_{\text{р}(i)} = \max_{j \in G}\left(T_{\text{р}(j)} + t_j\right).$$

Позднее время начала следует вычислять по формуле:

$$T_{\text{п}(i)} = \min_{j \in H}\left(T_{\text{п}(j)} - t_j\right).$$

Расчет раннего времени начала работ представлено в таблице 5.

Таблица 5 – Расчет раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	3	0	6	0+6	6
5	4	3	3	6+3	9
6	5	6	1	9+1	10
8	1	0	0	0+0	0
9	8	0	6	0+6	6
10	9	6	2	6+2	8
12	6 10	10 5	0 0	Max(10+0; 8+0)	10
14	12	10	2	10+2	12
15	14	12	3	12+3	15
17	12	10	2	10+2	12
18	17	12	4	12+4	16
19	15 18	15 16	7 6	Max(15+7; 16+6)	22
21	19	22	0	22+0	22
22	21	22	3	22+3	25
23	22	25	10	25+10	35
25	23	35	0	35+0	35
26	25	35	10	35+10	45
27	26	45	5	45+5	50
28	27	50	0	50+0	50

Расчет позднего времени начала работ представлено в таблице 6.

Таблица 6 – Вычисление позднего времени начала работ

Работа	Последователи		Длительность	Вычисление	Позднее время начала
	№	Позднее время начала			
28	-	-	-	-	50
27	28	50	0	50-0	50
26	27	50	5	50-5	45
25	26	45	10	45-10	35
23	25	35	0	35-0	35

Продолжение таблицы 6

Работа	Последователи		Длительность	Вычисление	Позднее время начала
	№	Позднее время начала			
22	23	35	10	35-10	25
21	22	25	3	25-3	22
19	21	22	0	22-0	22
18	19	22	6	22-6	16
17	18	16	4	16-4	12
15	19	22	7	22-7	15
14	15	15	3	15-3	12
12	14 17	12 12	2 2	Min(12-2;12-2)	10
10	12	10	0	10-0	10
9	10	10	2	10-2	8
8	9	8	6	8-6	2
6	12	10	0	10-0	10
5	6	10	1	10-1	9
4	5	9	3	9-3	6
3	4	6	6	6-6	0
1	3 8	0 2	0 0	Min(0-0;2-0)	0

На базе вычисленных значений раннего и позднего времени начала работ можно найти критический путь. Расчет критического пути представлен в таблице 7.

Таблица 7 – Вычисление критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	0	0
4	6	6	0
5	9	9	0
6	10	10	0
8	0	2	2
9	6	8	2
10	8	10	2
12	10	10	0
14	12	12	0
15	15	15	0
17	12	12	0
18	16	16	0
19	22	22	0
21	22	22	0
22	25	25	0
23	35	35	0
25	35	35	0
26	45	45	0
27	50	50	0
28	50	50	0

Таким образом, на базе полученных данных, приходим к тому, что критический путь проходит через работы с номерами 1, 3, 4, 5, 6, 12, 14, 15, 17, 18, 19, 21, 22, 23, 25, 26, 27, 28, а общее время реализации проекта равно 50 дням. На рисунке Д.2 представлен сетевой график, на котором желтыми стрелками указан критический путь.

Далее проверим правильность расчетов длительности проекта и вычислим его бюджет. Для этого воспользуемся системой управлениями проектами Microsoft Project, которая и рассчитает необходимые показатели [17 – 20].

На рисунке Д.3 представлен список задач из таблицы 23, внесенный в Microsoft Project. Началом проекта была принята дата 01.03.2022.

На рисунке Д.4 представлена диаграмма Ганта, построенная в среде Microsoft Project на основе введенных данных.

На основе построенной диаграммы Ганта можно сделать вывод, что критический путь проходит через задачи 1, 3, 4, 5, 6, 12, 14, 15, 17, 18, 19, 21, 22, 23, 25, 26, 27, 28. Таким образом критический путь, вычисленный программой полностью совпал с критическим путем, рассчитанным в ручную.

На рисунке видно, что все задачи разделены на фазы.

Приведем графики ресурсов. На рисунках 14 – 16 представлены графики трудовых ресурсов «Программист1», «Программист2», «Дизайнер».

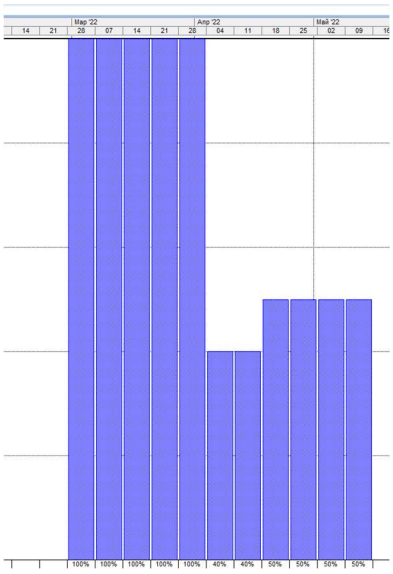


Рисунок 14 – График ресурсов для ресурса «Программист1»

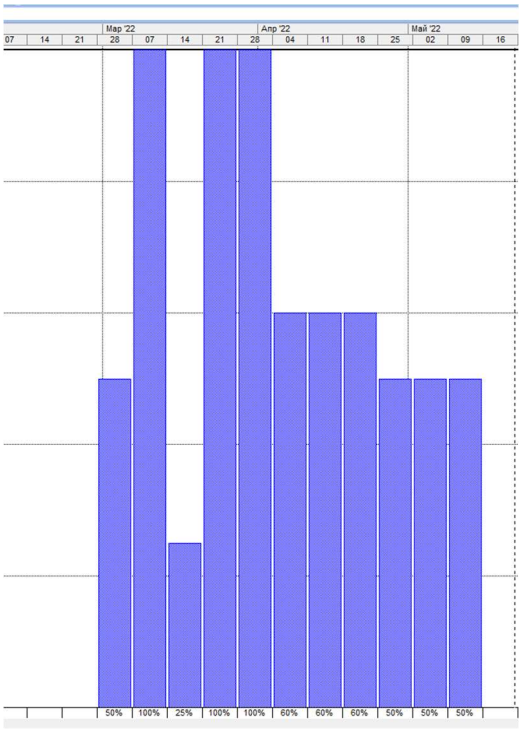


Рисунок 15 – График ресурсов для ресурса «Программист2»

По загруженности программистов 1, 2 видно, что они заняты на протяжении всей длительности проекта.

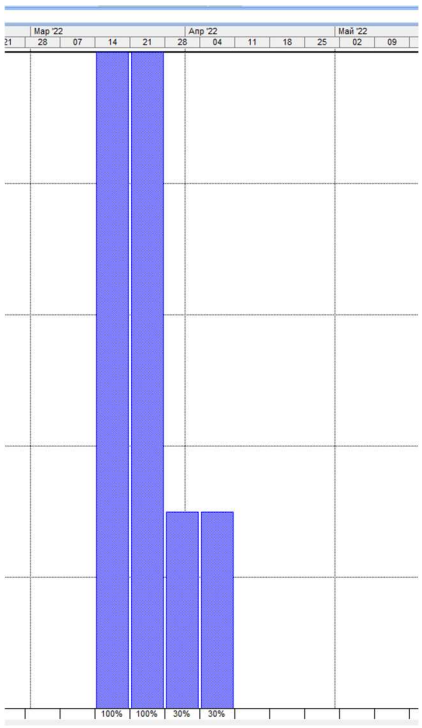
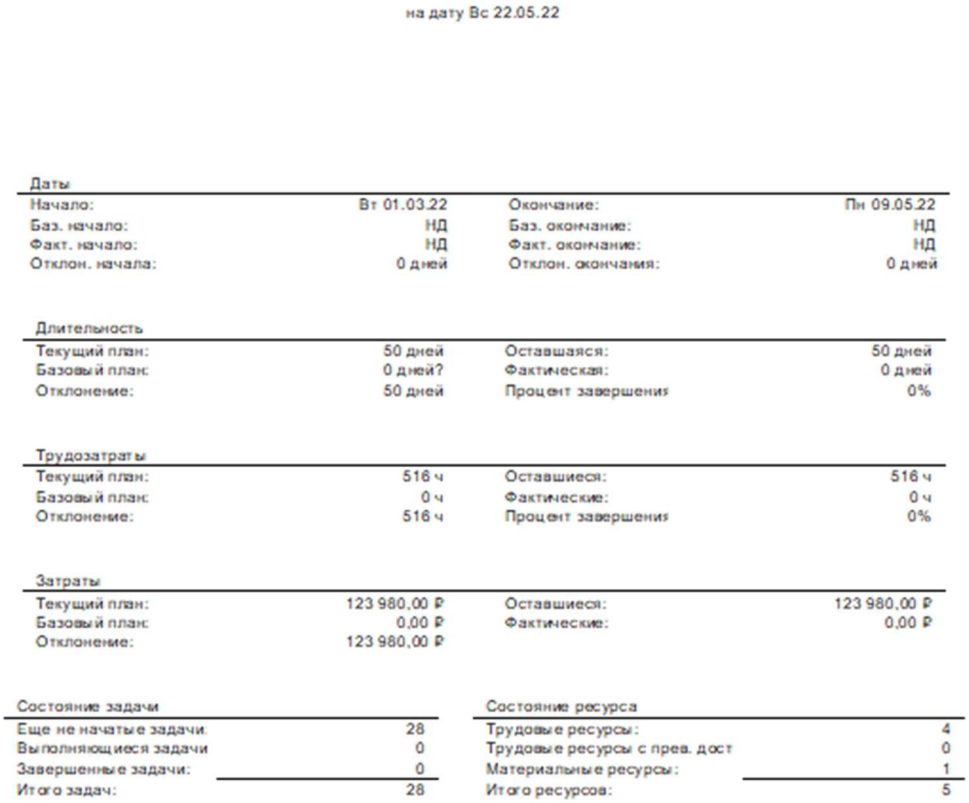


Рисунок 16 – График ресурсов для ресурса «Дизайнер»

По графикам видно, что ни один трудовой ресурс не превышает норму нагрузки.

На рисунке 17 представлен обзор проекта.



на дату Вс 22.05.22

Даты			
Начало:	Вт 01.03.22	Окончание:	Пн 09.05.22
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней

Длительность			
Текущий план:	50 дней	Оставшаяся:	50 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	50 дней	Процент завершения	0%

Трудозатраты			
Текущий план:	516 ч	Оставшиеся:	516 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	516 ч	Процент завершения	0%

Затраты			
Текущий план:	123 980,00 ₽	Оставшиеся:	123 980,00 ₽
Базовый план:	0,00 ₽	Фактические:	0,00 ₽
Отклонение:	123 980,00 ₽		

Состояние задачи	
Еще не начатые задачи:	28
Выполняющиеся задачи:	0
Завершенные задачи:	0
Итого задач:	28

Состояние ресурса	
Трудовые ресурсы:	4
Трудовые ресурсы с прев. дост:	0
Материальные ресурсы:	1
Итого ресурсов:	5

Рисунок 17 – Обзорный отчет проекта

Из рисунка видно, что общая длительность проекта составляет 50 дней, что совпадает длительностью, рассчитанной вручную, а предполагаемые затраты – 123 980 руб.

На основе произведенных оценок стоимости и сроков проекта можно сделать вывод о его экономической целесообразности.

2 Проектирование web-приложения

2.1 Выбор технологий, инструментальных и языковых средств разработки

В основе методологии Rational Unified Process (RUP), выбранной для разработки web-приложения, лежит принцип итеративной модели разработки, что позволяет разбить весь проект на более мелкие проекты, выполняющиеся последовательно. Каждая итерация должна решать свои цели и задачи и в результате давать промежуточную функциональную версию программного обеспечения. Данный принцип устраняет риски на каждой стадии проекта, что позволяет лучше понять проблему и вносить необходимые изменения, пока е будет найдено приемлемое решение, то есть будут решены все задачи проекта [21].

Языком моделирования методологии RUP является Unified Modelling Language (UML). Данный язык используется для объектно-ориентированного анализа и построения диаграмм, моделирующих бизнес-процессы и структуры программного продукта [22].

Таким образом, учитывая достоинства RUP, мы выбираем данную методологию для разработки web-приложения для мониторинга и анализа результатов спортивных соревнований.

Серверная часть приложения разрабатывалось на языке программирования Python с использованием фреймворка Django и Django Rest Framework [23]. Серверная часть представляет собой REST API и выполняет взаимодействие с базой данной [24]. В качестве системы управления базой данных была выбрана система PostgreSQL. Взаимодействия с базой данной производилась с помощью средств ORM от фреймворка Django.

Для тестирования работы серверной части использовалась библиотека модульного тестирования pytest [25].

Клиентская часть разрабатывалась на языке javascript, с помощью библиотеки React.js [26]. Также для стилизации использовалась библиотека MUI [27].

Планирование проекта и оценка бюджета были выполнены при помощи программы Microsoft Project.

2.2 Проектирование архитектуры программного обеспечения

При разработке web-приложения использовалась двухуровневая архитектура «клиент-сервер». Схема архитектуры представлена на рисунке 18.

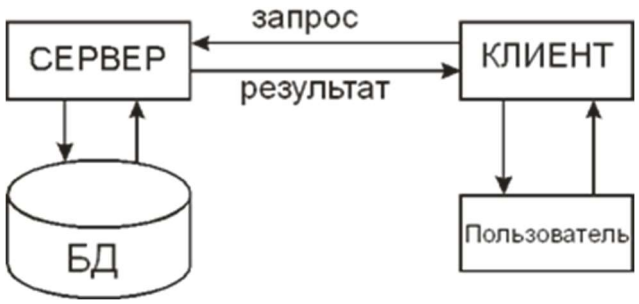


Рисунок 18 – Клиент-серверная архитектура

Двухуровневая архитектура используется в клиент-серверных системах, где сервер отвечает на клиентские запросы напрямую и в полном объеме, при этом используя только собственные ресурсы. Т.е. сервер не вызывает сторонние сетевые приложения и не обращается к сторонним ресурсам для выполнения какой-либо части запроса [28].

Применение такой архитектуры имеет ряд сильных сторон:

– все данные хранятся на сервере, который защищен гораздо лучше, чем клиенты;

– снижается нагрузка на клиентские приложения;

– программный код серверной и клиентской частей приложения разделен, это упрощает дальнейшую работу по доработке или изменению программного продукта.

Взаимодействие между клиентом и сервером осуществляется следующим образом: клиент посылает серверу HTTP-запросы, интерфейс прикладного программирования API обрабатывает этот запрос и посылает запрос на языке SQL базе данных PostgreSQL. В случае успешной обработки запроса клиенту возвращаются данные в формате JSON – текстовый формат

обмена данными [29]. После этого клиентская часть отображает полученные данные в удобной форме.

2.3 Проектирование структур базы данных

2.3.1 Концептуальная модель базы данных

Концептуальная модель представляет собой описание основных сущностей (таблиц) и связей между ними без учета принятой модели БД и синтаксиса целевой СУБД. Часто на такой модели отображаются только имена сущностей (таблиц) без указания их атрибутов. Основная задача концептуальной модели – передать фундаментальные принципы и основные функциональные возможности системы, которую она представляет. Кроме того, концептуальная модель должна быть разработана таким образом, чтобы обеспечить легко понятную системную интерпретацию для пользователей модели.

Графическое представление концептуальной модели базы данных в виде диаграммы «сущность-связь» для предметной области информационной системы представлено на рисунке 19.

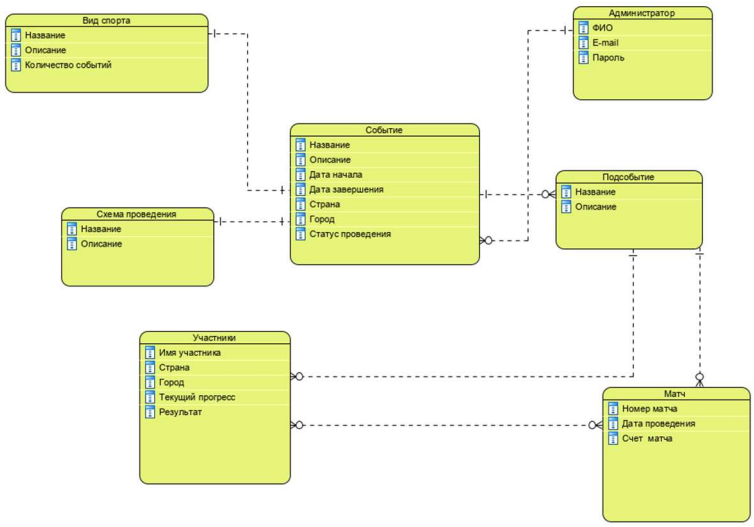


Рисунок 19 – Графическое представление концептуальной модели

Проведено описание сущностей, их атрибутов и связей между ними. В таблице 8 представлено описание сущностей.

Таблица 8 – Описание сущностей

№	Название	Описание	Количество экземпляров	Режим обновления
1	Событие	Событие (Чемпионат, соревнование, турнир и т.д.)	100	Раз в день
2	Вид спорта	Вид спорта, в котором проводятся матчи между двумя участниками	100	Раз в год и реже
3	Схема проведения	Схема проведения события (олимпийская система, круговая схема, групповая схема с плэй-офф и т.д.)	5	Раз в год и реже
4	Администратор	Администратор	100	Раз в день
5	Подсобытие	Подсобытие (Весовая категория, групповой этап)	100	Раз в день
6	Участник	Участник события (отдельный спортсмен или команда)	300	Раз в день
7	Матч	Отдельный матч	1000	Раз в день

В таблице 9 представлено описание атрибутов сущностей.

Таблица 9 – Описание атрибутов сущностей

№	Сущность	Атрибут	Описание
1	Событие	Название	Название события
		Описание	Описание события
		Дата начала	Дата начала
		Дата завершения	Дата завершения
		Страна	Страна проведения события
		Город	Город проведения события
		Статус	Текущий статус (завершено/не завершено)
2	Вид спорта	Название	Название вида спорта
		Описание	Описание вида спорта
		Количество событий	Количество событий по данному виду спорта
3	Схема проведения	Название	Название схемы проведения
		Описание	Описание схемы
4	Администратор	ФИО	ФИО администратора
		E-mail	Электронная почта
		Пароль	Пароль для входа
5	Подсобытие	Название	Название подсобытия
		Описание	Описание подсобытия
6	Участник	Имя участника	Имя участника или название команды
		Страна	Страна
		Город	Город
		Текущий прогресс	Текущий прогресс в событии
		Результат	Итоговый результат участника
7	Матч	Номер матча	Номер матча
		Дата проведения	Дата проведения
		Счет	Счет матча

В таблице 10 представлено описание связей между сущностями.

Таблица 10 – Описание связей

№	Отношение	Описание
1	Событие – Вид спорта	Определяет вид спорта события
2	Событие – Схема проведения	Определяет схему проведения события
3	Событие - Администратор	Определяет администратора события
4	Событие - Подсобытие	Определяет принадлежность подсобытия к событию
5	Подсобытие - Матч	Определяет принадлежность матча к подсобытию
6	Подсобытие - Участники	Определят принадлежность участников к подсобытию
7	Участники - Матч	Определяет принадлежность участников к матчу

Концептуальная модель построена в среде visual paradigm в нотации ERD (Entity Relationship Diagram) [30].

2.3.2 Логическая модель базы данных

Логическая модель – это модель базы данных, выраженная в понятиях модели данных. Логическая модель расширяет концептуальную путем определения для сущностей их атрибутов, описаний и ограничений, уточняет состав сущностей и взаимосвязи между ними.

На основании концептуальной модели была сконструирована логическая модель базы данных (рисунок 20).

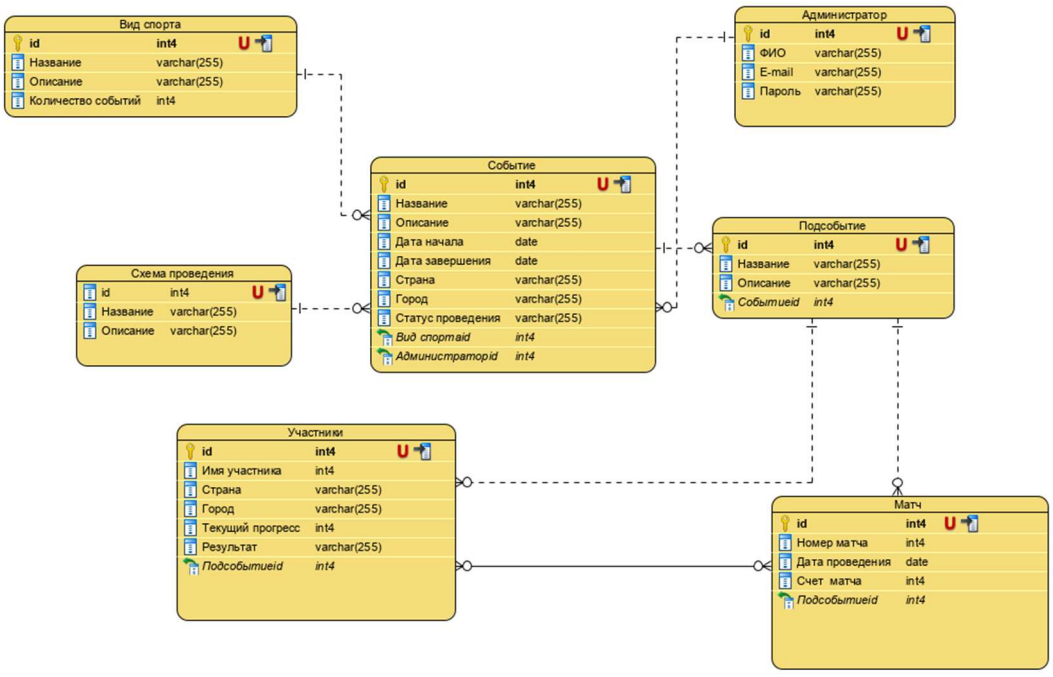


Рисунок 20 – Графическое представление логической модели базы данных

Проведено описание сущностей (таблица 11).

Таблица 11 – Описание сущностей

№	Название	Описание	Количество экземпляров	Режим обновления
1	Событие	Событие (Чемпионат, соревнование, турнир и т.д.)	100	Раз в день
2	Вид спорта	Вид спорта, в котором проводятся матчи между двумя участниками	100	Раз в год и реже
3	Схема проведения	Схема проведения события (олимпийская система, круговая схема, групповая схема с плэй-офф и т.д.)	5	Раз в год и реже
4	Администратор	Администратор	100	Раз в день
5	Подсобытие	Подсобытие (Весовая категория, групповой этап)	100	Раз в день
6	Участник	Участник события (отдельный спортсмен или команда)	300	Раз в день
7	Матч	Отдельный матч	1000	Раз в день

В таблице 12 приведено описание типов связей между сущностями.

Таблица 12 – Описание типов связи между сущностями

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность
2	Вид спорта	Событие	По виду спорту могут проводиться несколько событий	1:M	Обязательная
3	Схема проведения	Событие	По данной схеме проведения могут проводиться несколько событий	1:M	Обязательная
4	Администратор	Событие	Администратор может создать несколько событий	1:M	Обязательная
5	Событие	Подсобытие	События может состоять из нескольких подсобытий	1:M	Обязательное
6	Подсобытие	Матч	В одном подсобытии могут проводиться несколько матчей	1:M	Обязательное
7	Матч	Участники	Участник может участвовать в нескольких матчах	M:M	Обязательное
8	Подсобытие	Участники	В одном подсобытии участвуют несколько участников	1:M	Обязательное

В таблице 13 приведено описание атрибутов сущности «Событие».

Таблица 13 – Атрибуты сущности «Событие»

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер события	INT	6	0 – 100000	-	PK	1
Название	Имя события	VARCHAR	255	А – я A – z 0 – 9	-		Первенство города по тхэквондо
Описание	Описание события	VARCHAR	255	А – я A – z 0 – 9	-		Турнир по тхэквондо среди кадетов
Дата начала события	Дата начала события	DATE	10	0 – 9-	-		21-12-2021
Дата завершения события	Дата завершения события	DATE	10	0 – 9-	-		21-12-2021
Страна	Страна проведения	VARCHAR	255	А – яA – z	-		Россия
Город	Город проведения	VARCHAR	255	А – яA – z	-		Пенза
Статус проведения	Статус	VARCHAR	255	А – яA – z	-		Завершено

В таблице 14 приведено описание атрибутов сущности «Вид спорта».

Таблица 14 – Атрибуты сущности «Вид спорта»

Наименование	Описание	Тип данных	Размерность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер вида спорта	INT	3	1 – 100	-	PK	1
Название	Название вида спорта	VARCHAR	255	А – я A – z	-		Баскетбол
Описание	Описание вида спорта	VARCHAR	255	А – я A – z	-		Командная игра с мячом
Количество событий	Количество событий по данному виду спорта	INT	6	100000	-		100

В таблице 15 приведено описание атрибутов сущности «Администратор».

Таблица 15 – Атрибуты сущности «Администратор»

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер администратора	INT	6	1 – 100000	-	PK	1
Имя администратора	ФИО администратора	VARCHAR	255	А – я A – z	-		Иванов Иван
Email	Электронная почта администратора	VARCHAR	255	А – я A – z@.	-		ivan@example.ru

Продолжение таблицы 15

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
Пароль	Пароль для авторизации	VARCHAR	255	А – я A – z0 – 9	-		12345sad

В таблице 16 приведено описание атрибутов сущности «Подсобытие».

Таблица 16 – Атрибуты сущности «Подсобытие»

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер администратора	INT	6	1 – 100000	-	PK	1
Название	Название подсобытия	VARCHAR	255	А – яA – z	-		Кадеты весовая категория до 41 кг
Описание	Описание подсобытия	VARCHAR	255	А – яA – z	-		Поединки кадетов весовой категории до 41 кг

В таблице 17 приведено описание атрибутов сущности «Матч».

Таблица 17 – Атрибуты сущности «Матч»

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер матча	INT	6	1 – 100000	-	PK	1
Дата проведения	Дата проведения матча	DATE	10	0 – 9-	-		21-12-2021
Счет 1 участника	Счет 1-го участника	INT	4	1 – 1000	-		23
Счет 2 участника	Счет 2-го участника	INT	4	1 – 1000	-		23

В таблице 18 приведено описание атрибутов сущности «Участники».

Таблица 18 – Атрибуты сущности «Участники»

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
id	Уникальный участника	INT	6	1 – 100000	-	PK	1
Имя участника	Имя участника	VARCHAR	255	А – яA – z	-		Ivanov Ivan
Страна	Страна проведения	VARCHAR	255	А – яA – z	-		Россия
Город	Город проведения	VARCHAR	255	А – яA – z	-		Пенза

Продолжение таблицы 18

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
Текущий прогресс	Текущий прогресс участника в соревнованиях	VARCHAR	255	А – яА – z/	-		1/8 финала

В таблице 19 приведено описание атрибутов сущности «Схема проведения».

Таблица 19 – Атрибуты сущности «Схема проведения»

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер схемы проведения	INT	3	1 – 100	-	PK	1
Название	Название схемы	VARCHAR	255	А – я A – z	-		Олимпийская система
Описание	Описание схемы	VARCHAR	255	А – я A – z	-		Участники продвигаются по сетке в зависимости от результата матча

Логическая модель построена в среде visual paradigm в нотации ERD (Entity Relationship Diagram) [30].

2.4 Проектирование программных средств

На основе модели базы данных была спроектированы классы: «Event», «Subevent», «System», «SportTypes», «Match», «Participant». А также класс «Requester», который предназначен для запросов данных.

Диаграмма классов представлена на рисунке 21.

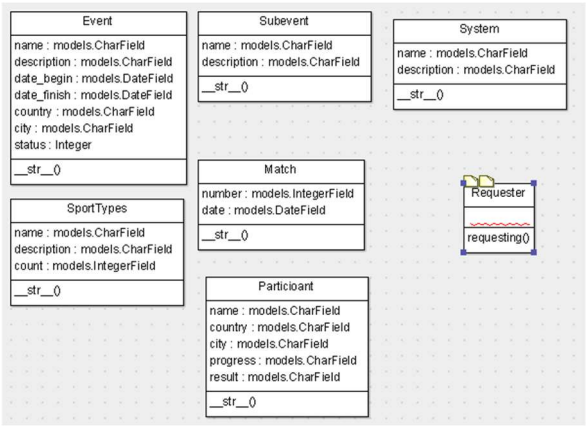


Рисунок 21 – Диаграмма классов

Спецификация классов представлена в таблице 20.

Таблица 20 – Спецификация классов

Класс	Описание	Поля	Описание	Методы	Описание
Event	Класс модели Event	name	Название события	__str__()	Возвращает строку с названием события
		description	Описание события		
		date_begin	Дата начала		
		date_finish	Дата завершения		
		country	Страна		
		city	Город		
		status	Статус		
Match	Класс модели Match	number	Номер матча	__str__()	Возвращает строку с номером матча
		date	Дата проведения		
Subevent	Класс модели Subevent	name	Название подсобытия	__str__()	Возвращает строку с названием подсобытия
		description	Описание подсобытия		
Participant	Класс модели Participant	name	Имя участника	__str__()	Возвращает строку с именем участника
		country	Страна		
		city	Город		
		progress	Прогресс участника		
		result	Результат		
System	Класс модели System	name	Название схемы	__str__()	Возвращает строку с названием схемы
		description	Описание схемы		
Sport_type	Класс модели Sport_type	name	Название вида спорта	__str__()	Возвращает строку с названием вида спорта
		description	Описание вида спорта		
		count	Количество событий по виду спорта		
Requester	Предназначен для запросов данных с сервера	-	-	requesting()	Делает запрос по заданному URL

Классы были спроектированы согласно правилам библиотеки DjangoORM [23].

2.5 Проектирование пользовательского интерфейса

Для проектирования интерфейса построим макет приложения в графическом онлайн редакторе Figma.

Пользовательский интерфейс представляет собой single page application [31]. Главный экран представлен на рисунке 22.

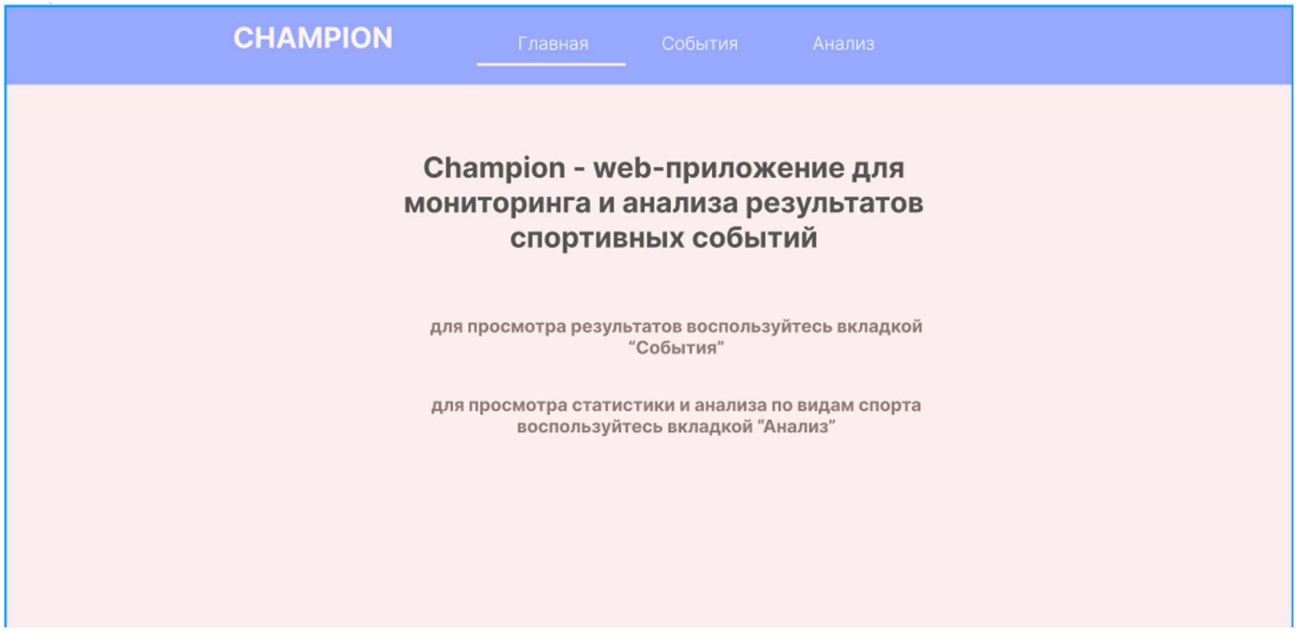


Рисунок 22 – Главный экран приложения

На главном экране присутствует описание приложения. Также присутствует навигационная панель, с помощью которой пользователь сможет перемещаться по сайту. На ней три вкладки: «Главная», «События», «Анализ».

Для просмотра результатов событий, пользователю необходимо выбрать пункт «События», после чего пользователю предложат наиболее популярные виды спорта и поисковую строку для выбора вида спорта (рисунок 23).

При выборе вида спорта, пользователю подгрузятся карточки с событиями. На карточках расположено название соревнования и кнопка «посмотреть» рисунок 24.

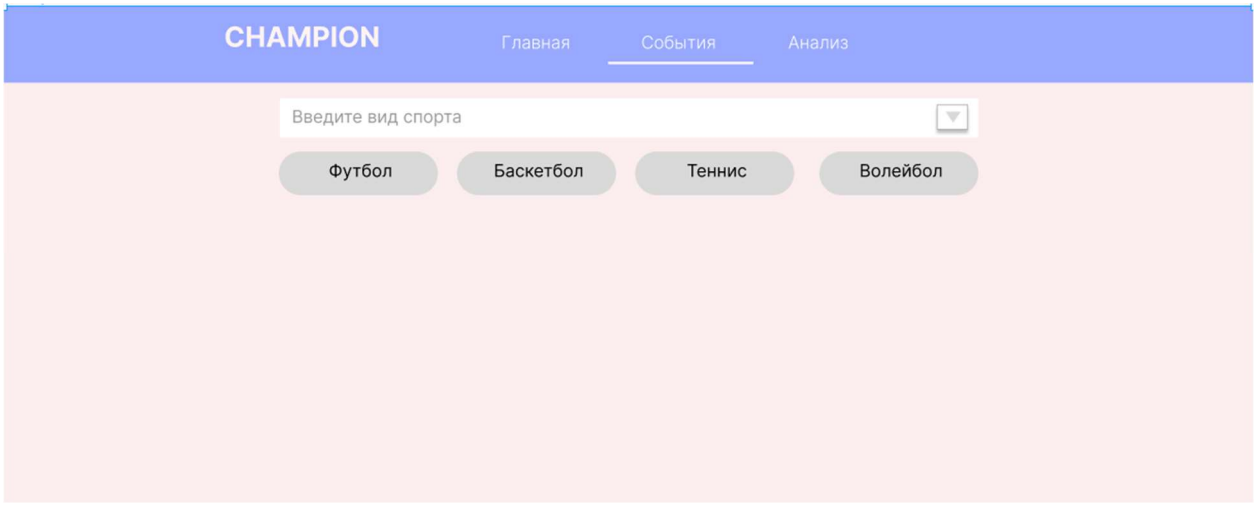


Рисунок 23 – Экран выбора вида спорта

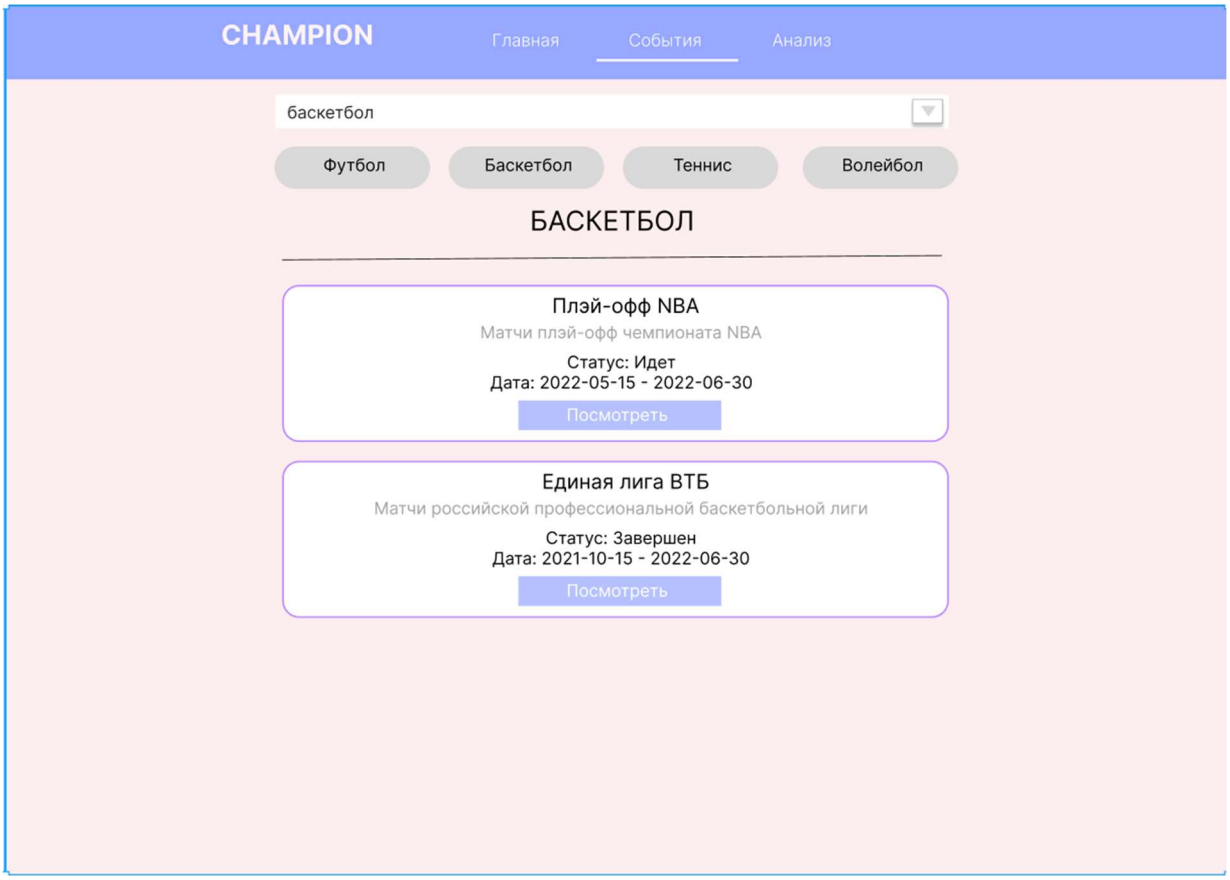


Рисунок 24 – Экран с загруженными событиями по баскетболу

При нажатии на кнопку «Посмотреть» появится список матчей, при нажатии на которые будет всплывать модальное окно с описанием матча. Список матчей показан на рисунке 25, а модальное окно на рисунке 26.

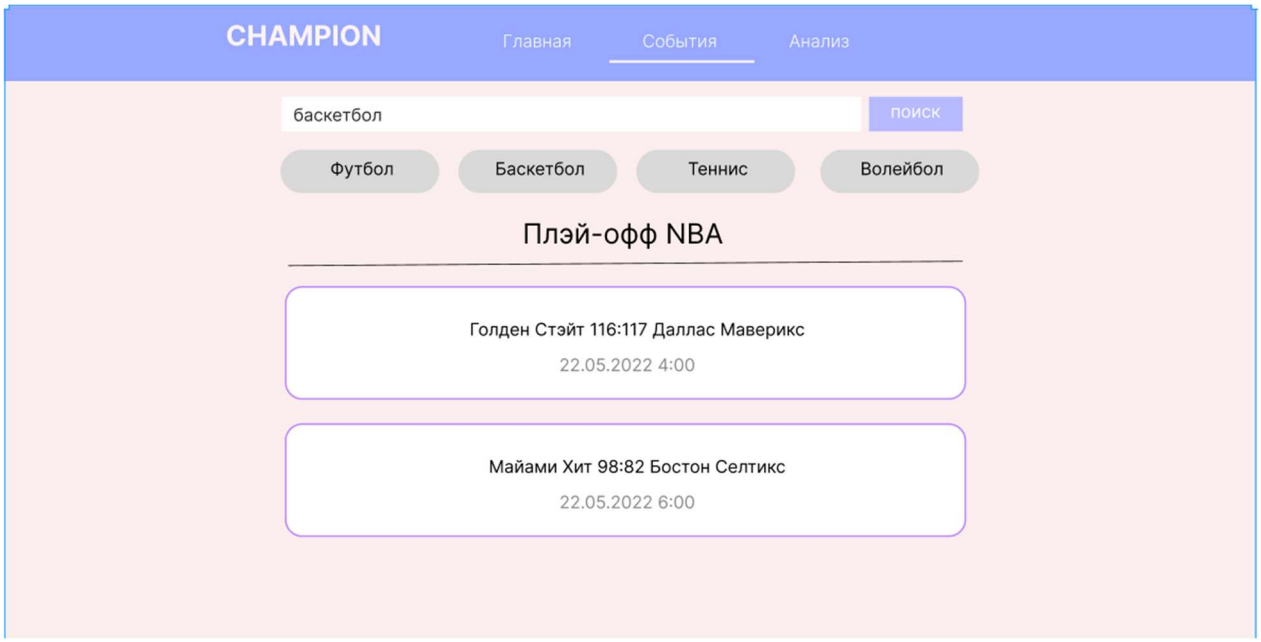


Рисунок 25 – Экран с загруженными матчами по баскетболу

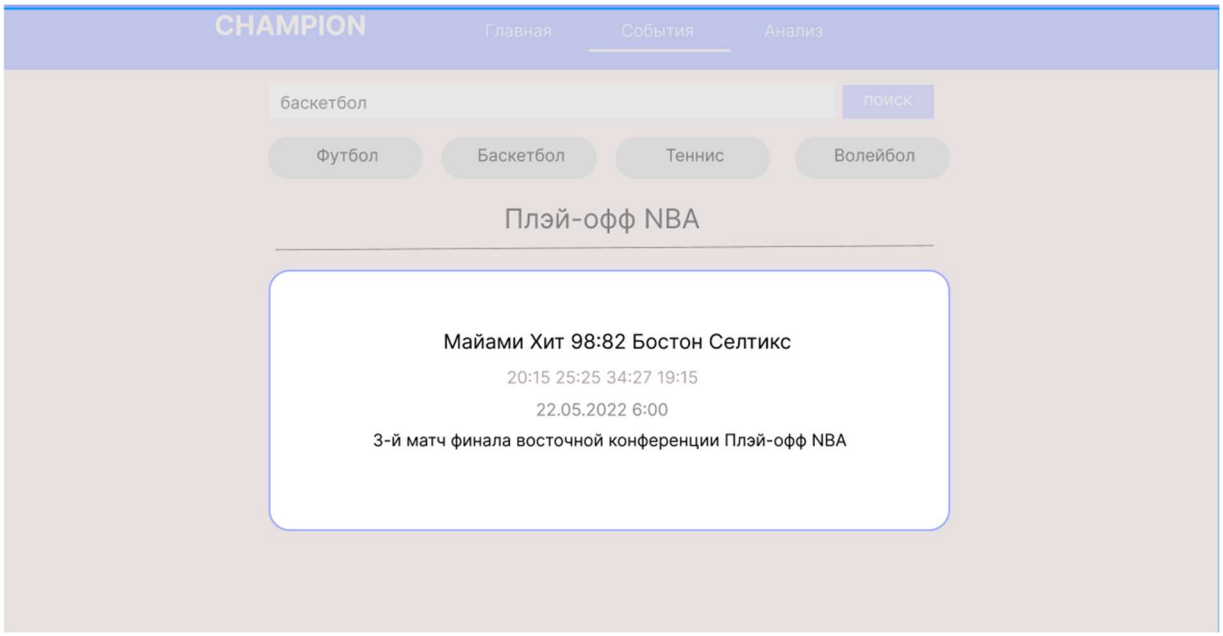


Рисунок 26 – Модальное окно с результатом матча

При выборе пункта меню «Анализ», пользователю предоставляется возможность просмотр сводки по каждому виду спорта, которая содержит статистическую информацию о количестве событий и гистограмму. Экран с примерной сводкой показан на рисунке 27.

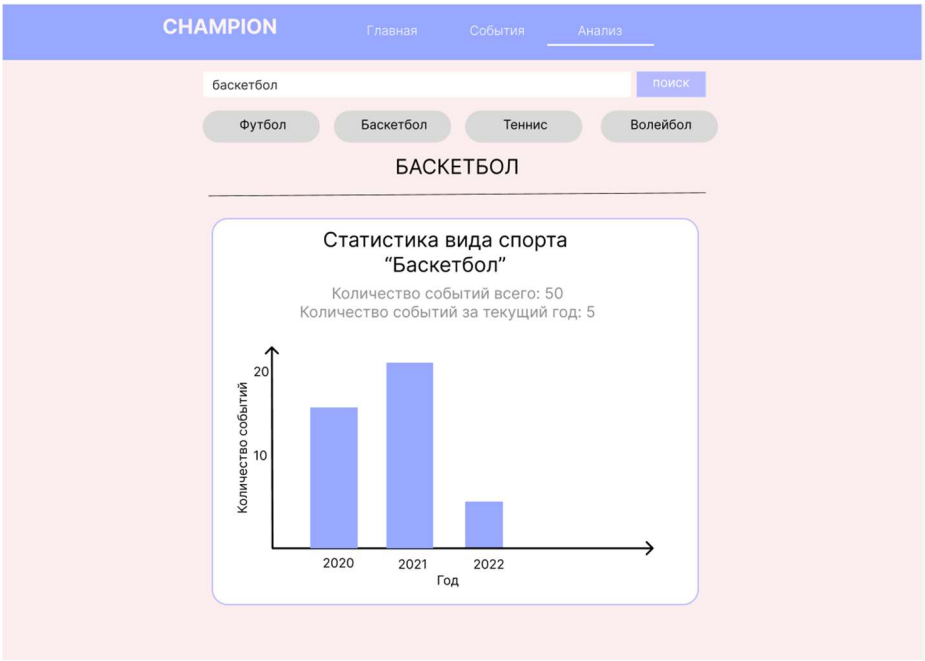


Рисунок 27 – Экран сводки по виду спорта Баскетбол

Таким образом, спроектированный пользовательский интерфейс удовлетворяет функциональным требованиям приложения.

3 Реализация web-приложения

3.1 Реализация серверной части web-приложения

3.1.1 Реализация базы данных

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания конкретной СУБД. В данном проекте – в терминах СУБД PostgreSQL.

На основе концептуальной и логической модели была разработана физическая модель базы данных (рисунок 28).

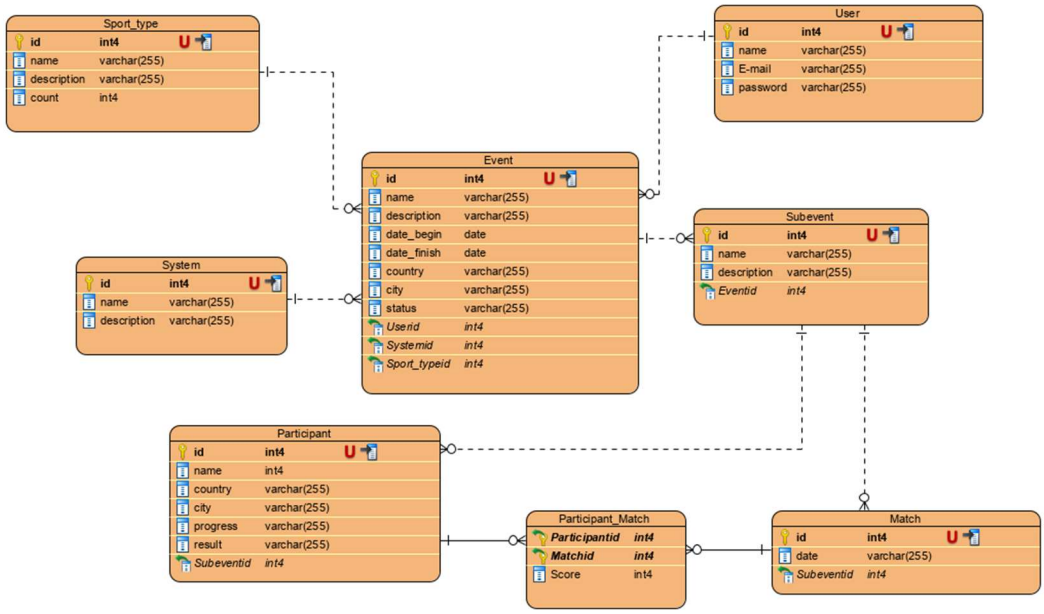


Рисунок 28 – Физическая модель базы данных

Согласно физической модели, была создана база данных в СУБД PostgreSQL.

Описание атрибутов сущности «Event» представлены в таблице 21.

Таблица 21 – Атрибуты сущности «Event»

Наименование		Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
id		Уникальный номер события	INT	6	0 – 100000	-	PK	1

Продолжение таблицы 21

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
name	Имя события	VARCHAR	255	А – я A – z 0 – 9	-		Первенство города по тхэквондо
description	Описание события	VARCHAR	255	А – я A – z 0 – 9	+		Турнир по тхэквондо среди кадетов
date_begin	Дата начала события	DATE	10	0 – 9-	-		21-12-2021
date_finish	Дата завершения события	DATE	10	0 – 9-	+		21-12-2021
country	Страна проведения	VARCHAR	255	А – я A – z	-		Россия
city	Город проведения	VARCHAR	255	А – я A – z	-		Пенза
status	Статус	VARCHAR	255	А – я A – z	-		Завершен
Adminid	Номер администратора, создавшего событие	INT	6	1 – 100000	-	PK	1
Systemid	Номер схемы проведения	INT	3	1 – 100	-	PK	1
Sport_typeid	Номер вида спорта	INT	3	1 – 100	-	PK	1

В таблице 22 приведено описание атрибутов сущности «Sport_type».

Таблица 22 – Атрибуты сущности «Sport_type»

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер вида спорта	INT	3	1 – 100	-	PK	1
name	Название вида спорта	VARCHAR	255	А – яA – z	-		Баскетбол
description	Описание вида спорта	VARCHAR	255	А – яA – z	-		Командная игра с мячом
count	Количество событий по данному виду спорта	INT	3	1 – 10000	-		10

В таблице 23 приведено описание атрибутов сущности «User».

Таблица 23 – Атрибуты сущности «User»

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер администратора	INT	6	1 – 100000	-	PK	1
name	ФИО администратора	VARCHAR	255	А – я A – z	-		Иванов Иван Иванович
Email	Электронная почта администратора	VARCHAR	255	А – я A –z @.	-		ivan@example.ru

Продолжение таблицы 23

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
password	Пароль для авторизации	VARCHAR	255	А – я A – z 0 – 9	-		12345sad

В таблице 24 приведено описание атрибутов сущности «Subevent».

Таблица 24 – Атрибуты сущности «Subevent»

Наименование	Описание	Тип данных	Размерность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер администратора	INT	6	1 – 100000	-	PK	1
name	Название подсобытия	VARCHAR	255	А – я A – z	-		Кадеты весовая категория до 41 кг
description	Описание подсобытия	VARCHAR	255	А – я A – z	-		Поединки кадетов весовой категории до 41 кг
Eventid	Номер события	INT	6	0 – 100000	-	FK	1

В таблице 24 приведено описание атрибутов сущности «Match».

Таблица 25 – Атрибуты сущности «Match»

Наименование	Описание	Тип данных	Размерность	ОДЗ	Nullable	Роль	Пример
Id	Уникальный номер матча	INT	6	1 – 100000	-	PK	1
number	Номер матча	INT	6	1 – 100000	-		1
date	Дата проведения матча	DATE	10	0 – 9-	-		21-12-2021
Subeventid	Номер подсобытия	INT	6	1 – 100000	-	FK	1

В таблице 26 приведено описание атрибутов сущности «Participant».

Таблица 26 – Атрибуты сущности «Participant»

Наименование	Описание	Тип данных	Размерность	ОДЗ	Nullable	Роль	Пример
id	Уникальный участника	INT	6	1 – 100000	-	PK	1
name	Имя участника	VARCHAR	255	А – я A – z	-		Ivanov Ivan

Продолжение таблицы 26

Наименование	Описание	Тип данных	Размер-ность	ОДЗ	Nullable	Роль	Пример
result	Итоговый результат участника в событии	VARCHAR	255	А – я A – z/	-		1/8 финала
Subeventid	Номер подсобытия	INT	6	1 – 100000	-	FK	1
country	Страна проведения	VARCHAR	255	А – я A – z	-		Россия
city	Город проведения	VARCHAR	255	А – я A – z	-		Пенза
progress	Текущий прогресс участника в соревнованиях	VARCHAR	255	А – я A – z/	-		1/8 финала

В таблице 27 приведено описание атрибутов сущности «System».

Таблица 27 – Атрибуты сущности «System»

Наименование	Описание	Тип данных	Размерность	ОДЗ	Nullable	Роль	Пример
id	Уникальный номер схемы проведения	INT	3	1 – 100	-	PK	1
name	Название схемы	VARCHAR	255	А – я A – z	-		Олимпийская система
description	Описание схемы	VARCHAR	255	А – я A – z	-		Участники продвигаются по сетке

В таблице 28 приведено описание атрибутов сущности «Participant_Match».

Таблица 28 – Атрибуты сущности «Participant_Match»

Наименование	Описание	Тип данных	Размерность	ОДЗ	Nullable	Роль	Пример
Participantid	Номер пользователя	INT	6	1 – 100000	-	FK	1
Matchid	Номер события	INT	6	0 – 100000	-	FK	1
Score	Счет участника	INT	4	1 – 1000	-		23

Физическая модель построена в среде visual paradigm в нотации ERD (Entity Relationship Diagram) [30].

3.1.2 Реализация серверных компонентов

Серверная часть приложения реализовывалась в среде программирования PyCharm Community с использованием языка Python и фрэймворка Django. Взаимодействие с базой данной производилось посредством Django ORM.

Серверная часть включает в себя классы моделей, которые соответствуют сущностям базы данных, классы сериализаторы, которые участвуют при передаче данных между клиентом и сервером и классы представления, которые служат для реализации представлений.

Диаграмма классов серверной части представлены на рисунках 29, 30.

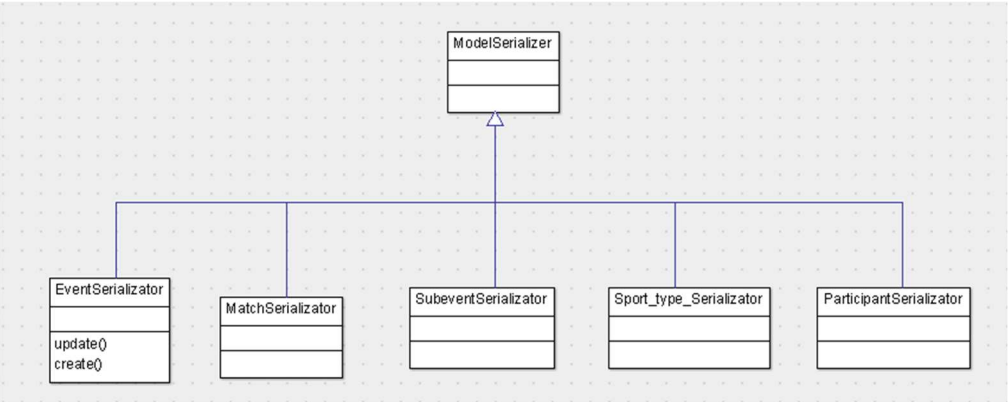


Рисунок 29 – Диаграмма классов сериализаторов

Каждый класс сериализатор наследуется от класса ModelSerializer фреймворка Django.

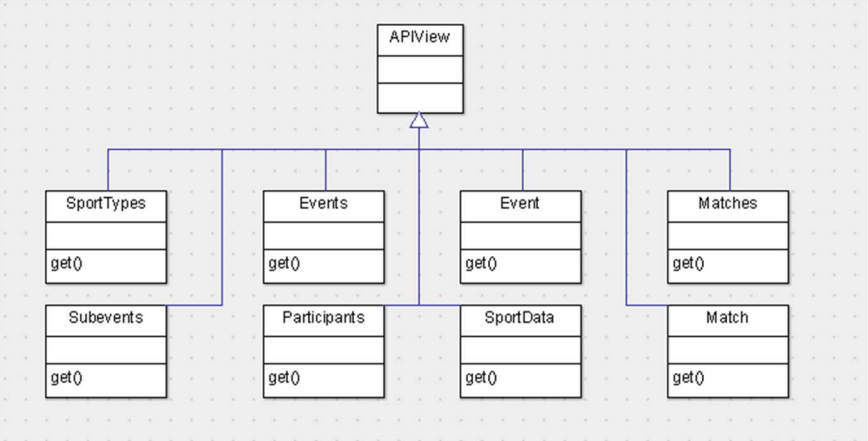


Рисунок 30 – Диаграмма классов представлений

Каждый класс представлений наследуется от класса APIView. Метод get() вызывается при соответствующем GET запросе.

С помощью классов представлений сервер реализует следующие методы API:

– GET /types/ – получение списка видов спорта;
– GET /events/ – получение списка событий;
– GET /event/ – получение события;
– GET /matches/ – получение списка матчей;
– GET /match/ – получение матча;
– GET /participants/ – получение списка участников;
– GET /sport/ – получение статистических данных.

При разработке триггеров был использован механизм сигналов Django ORM. Он позволяет пользовательскому коду получать сигналы – уведомления о действиях с базой данных, таких как сохранение сущности, ее модификация или удаление. Прием сигнала реализуется написанием функции с декоратором receiver в котором указывается обрабатываемый сигнал (сигнал перед модификацией данных, сигнал после удаления и т.д.) и модель, от которой необходимо принимать сигналы [23].

В листинге 1 представлен обработчики сигналов, пересчитывающий количество событий по виду спорта, по которому создается или удаляется событие. В таблице 29 приведены спецификации обработчиков.

Листинг 1 – Обработчики сигналов, пересчитывающие количество событий по виду спорта, по которому создается или удаляется событие

```
@receiver(post_save, sender=Event)
def create_event_handler(sender, instance, created, **kwargs):
    if created:
        sport_id = instance.sport_type_id.id
        s_type = Sport_type.objects.get(pk=sport_id)
        s_type.count = Event.objects.filter(sport_type_id=sport_id).count()
        s_type.save()


@receiver(post_delete, sender=Event)
def delete_event_handler(sender, instance, **kwargs):
    sport_id = instance.sport_type_id.id
    s_type = Sport_type.objects.get(pk=sport_id)
```

```
    s_type.count = Event.objects.filter(sport_type_id=sport_id).count()
    s_type.save()
```

Таблица 29 – Спецификации триггеров

№ п/п	Назначение триггера	Вид триггера	Событие триггера	Таблица триггера	Описание работы триггера
1	Пересчет количества событий по виду спорта, по которому создается событие	AFTER	INSERT	tournament_event	Пересчитывает количество событий при добавления нового события и сохраняет результат
2	Пересчет количества событий по виду спорта, по которому удаляется событие	AFTER	DELETE	tournament_event	Пересчитывает количество событий при удалении события и сохраняет результат

Функции триггеров были реализованы в модуле signals.py.

На рисунке 31 показана диаграмма компонентов серверной части приложения.

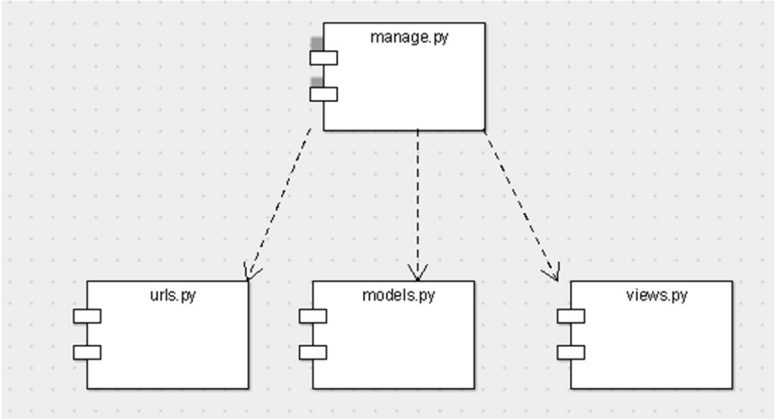


Рисунок 31 – Диаграмма компонентов сервера приложения

Рассмотрим каждый компонент:

– manage.py – предназначен для выполнения административных задач (запуск dev-сервера, создание администратора и т.д.);

– urls.py – содержит URL адреса;

– models.py – предназначен для описания моделей базы данных;

– views.py – компонент, содержащий все функции доступа к базе данных.

Для запуска сервера, необходимо выполнить команду «runserver» из файла manage.py.

3.2 Реализация клиентской части web-приложения

Клиентская часть приложения реализовывалась в текстовом редакторе Visual Studio Code на языке javascript с использованием фреймворка React и библиотеки стилей Material UI.

При разработке пользовательского интерфейса использовался компонентный подход. Весь интерфейс представляет собой Single page application, в котором три вкладки: «Главная», «События» и «Анализ». Навигация по приложению реализовано при помощи компонентов библиотеки Material UI: Grid, Tabs, Tab. При переходе по вкладкам загружаются соответственно страницы: Home, Events, Analysis.

Основным компонентом приложения является App.jsx, который вызывает другие компоненты такие, как Event.jsx, Matches.jsx, Subevent.jsx, Search.jsx. Диаграмма компонентов клиентского приложения представлена на рисунке 32.

Все компоненты выполнены в виде экспортируемых функций, которые возвращают представление.

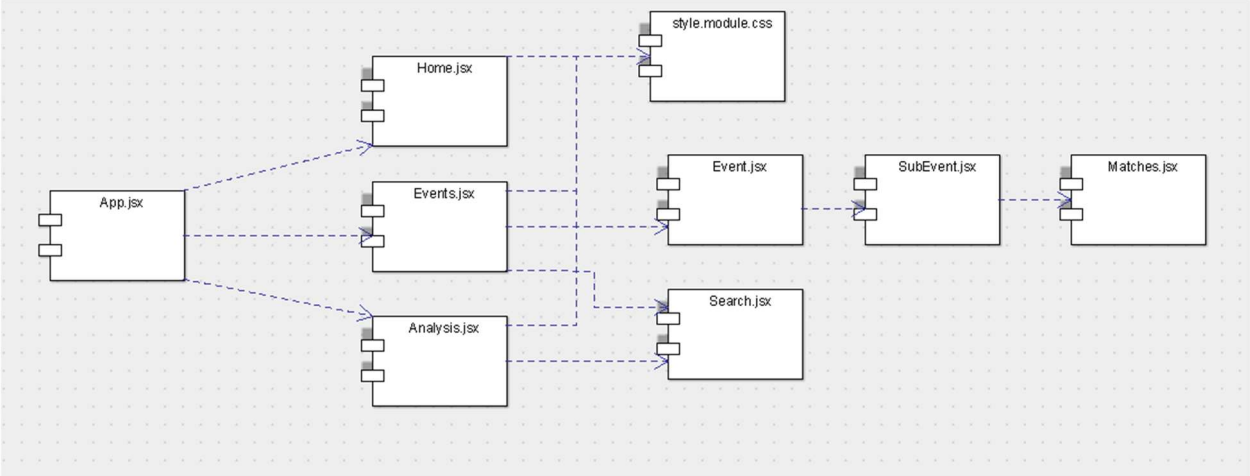


Рисунок 32 – Диаграмма компонентов клиентского приложения

Для выполнения административных функций, использовалась встроенное во фреймворк Django административное приложение, которое позволяет добавлять, изменять и удалять данные из базы, а также имеет свою авторизацию.

На рисунке 33 изображена диаграмма развертывания web-приложения.

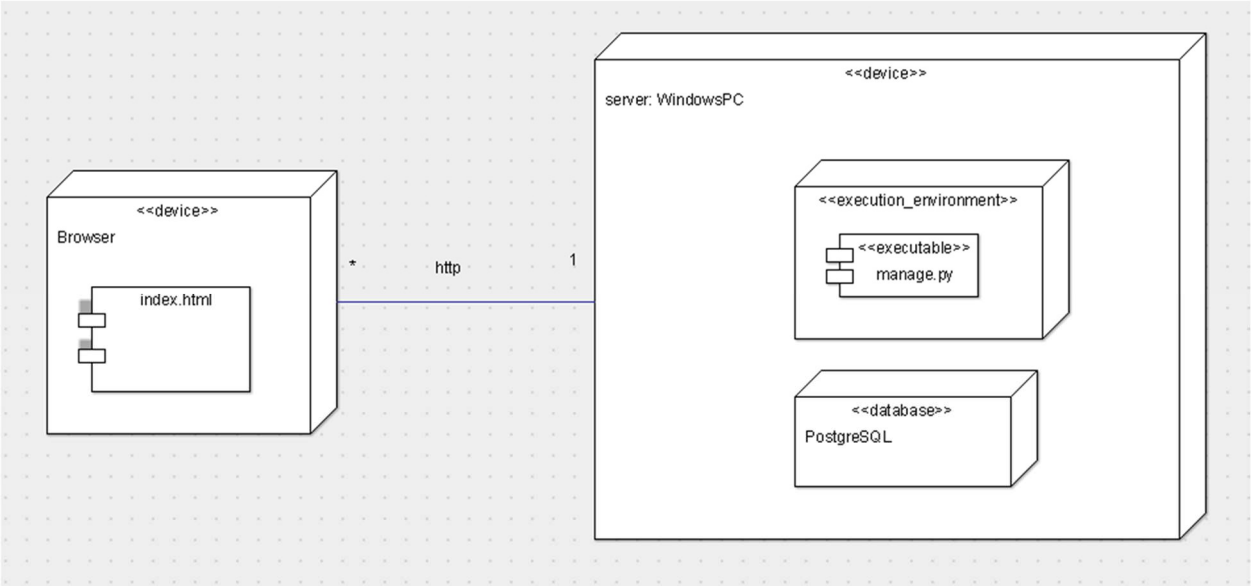


Рисунок 33 – Диаграмма развертывания web-приложения

Диаграмма показывает, что API приложения и база данных располагаются на сервере, а клиентское разворачивается на отдельном устройстве и запускается посредством браузера.

4 Тестирование web-приложения

4.1 Выбор режима тестирования приложения

Существует достаточно много видов тестирования программного продукта. Были рассмотрены основные виды тестирования [32]:

– функциональное тестирование;
– нефункциональное тестирование;
– интеграционное тестирование;
– модульное тестирование.

Функциональное тестирование – направлено на проверку того, какие функции приложения реализованы, и что они работают верным образом. Данный метод тестирования является одним из самых применяемых, так как позволяет наиболее полно проверить программу на соответствие требованиям.

Нефункциональное тестирование – вид тестирования, направленный на проверку нефункциональных особенностей приложения, таких как удобство использования, совместимость, производительность, безопасность и т.д.

Интеграционное тестирование – направлено на проверку взаимодействия между несколькими частями приложения. Этот вид тестирования позволяет проверить корректность взаимодействия частей ПО между собой.

Юнит тестирование или модульное тестирование – заключается в изолированной проверке каждого отдельного элемента путем запуска тестов в искусственной среде.

Было принято решение осуществить проверку клиентской части программы с помощью функционального тестирования, а у серверной части провести модульное тестирование. Такой выбор обусловлен тем, что к программе были предъявлены в основном функциональные требования, выполнение которых нужно проверить.

4.2 Тестирование серверной части приложения информационной системы

В результате разработки программы было проведено модульное тестирование. Модульные тесты были разработаны с помощью фреймворка для unit-тестирования PyTest. Исходный код модульных тестов приведен в приложении. План тестирования приведен в таблице 30.

Таблица 30 – План модульного тестирования

Тест	Описание	Ожидаемый результат	Актуальный результат
test_get_events	Проверка правильности выдачи результата по запросу /events	Пройден	Пройден (рисунок 34)
test_get_event	Проверка правильности выдачи результата по запросу /event/1	Пройден	Пройден (рисунок 34)
test_get_subevents	Проверка правильности выдачи результата по запросу /subevents/1	Пройден	Пройден (рисунок 34)
test_get_participants	Проверка правильности выдачи результата по запросу /participants/1	Пройден	Пройден (рисунок 34)
test_get_participant	Проверка правильности выдачи результата по запросу /participant/1	Пройден	Пройден (рисунок 34)
test_get_matches	Проверка правильности выдачи результата по запросу /matches/1	Пройден	Пройден (рисунок 34)
test_get_match	Проверка правильности выдачи результата по запросу /match/4	Пройден	Пройден (рисунок 34)

Из 7 тестов успешно завершились все 7 тестов.

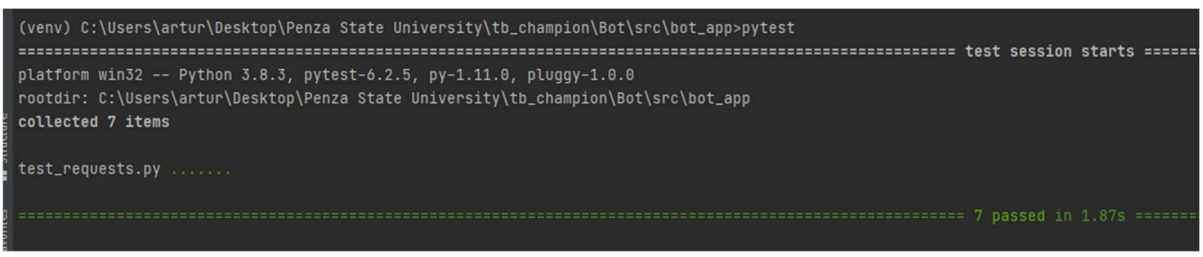


Рисунок 34 – Результаты модульного тестирования

Модульные тесты расположены в модуле test_requests.py

4.3 Тестирование клиентской части приложения информационной системы

Для тестирования основных вариантов использования были выполнены функциональные тесты. План тестирования приведен в таблице 31.

Таблица 31 – План функционального тестирования

№ п/п	Назначение теста	Описание	Результаты теста
1	Тест предназначен для проверки варианта использования «Посмотреть анализ развития вида спорта»	Выбор вкладки «Анализ», выбор вида спорта «Тхэквондо». В результате должно вывестись два графика.	Пройден (Рисунок Г.1)
2	Тест предназначен для проверки варианта использования «Посмотреть текущие результаты»	Выбор вкладки события, затем выбор вида спорта, после чего выбор события, подсобытия и результата. В результате в модальном окне показывается результат.	Пройден (Рисунки Г.2 – 5)
3	Тест предназначен для проверки варианта использования «Посмотреть текущие результаты»	Просмотр результатов по виду спорта, по которому нет событий. В результате должно появиться сообщение о том, что события не найдены.	Пройден (Рисунок Г.6)
4	Тест предназначен для проверки варианта использования «Войти в систему»	Успешная авторизация в приложении для администрации.	Пройден (Рисунки Г.7, 8)
5	Тест предназначен для проверки варианта использования «Войти в систему»	Неуспешная авторизация в приложении для администрации.	Пройден (Рисунки Г.9, 10)
6	Тест предназначен для проверки варианта использования «Завершить сеанс»	Выход из приложения для администратора.	Пройден (Рисунок Г.11)
7	Тест предназначен для проверки варианта использования «Добавить событие»	Ввод данных события и сохранение. В результате должно появится новое событие.	Пройден (Рисунки Г.12 – 14)
8	Тест предназначен для проверки варианта использования «Добавить событие»	Ввод идентичного события. В результате должно появится сообщение о некорректных данных.	Пройден (Рисунок Г.15)
9	Тест предназначен для проверки варианта использования «Сформировать результат»	Добавление матча и результата.	Пройден (Рисунки Г.16 – 19)
10	Тест предназначен для проверки варианта использования «Изменить результат»	Изменение счета у участника.	Пройден (Рисунок Г.20)

Все тесты пройдены, следовательно, по результатам функционального тестирования можно сделать вывод о корректности работы клиентской части web-приложения.

4.4 Метрики кода

Метрика программного обеспечения – мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций [33].

Приведенный в листинге 2 код предназначен для получения данных от сервера.

Листинг 2 – Методы для получения данных по API

```
import axios from 'axios';
import data from './mock.json';
const api = 'http://localhost:8000';
export const getTypes = async () => {
  return axios.get(`${api}/types`);
};
export const getEvents = async (id) => {
  return axios.get(`${api}/events/${id}`);
};
export const getSubEvents = async (id) => {
  return axios.get(`${api}/subevents/${id}`);
};
export const getSport = async (id) => {
  return axios.get(`${api}/sport/${id}`);
};
export const getMatches = async (id) => {
  return axios.get(`${api}/matches/${id}`);
};
export const getMatch = async (id) => {
  return axios.get(`${api}/match/${id}`);
};
export const getSportInfo = async (id) => {
  return axios.get(`${api}/sport/${id}`);
};
```

Приведенный в листинге 2 код используем для вычисления метрик кода.

4.4.1 Метрика SLOC

Количество строк кода (англ. Source Lines of Code – SLOC) – это метрика программного обеспечения, используемая для измерения его объёма с помощью подсчёта количества строк в тексте исходного кода [34].

Физическими строками считаются все непустые строки текстового файла.

Измеряя логические строки кода, предпринимается попытка посчитать количество собственно операторов в программе, но, разумеется, их определение зависит от конкретного языка программирования.

В данной программе имеется:

- 24 физических строк;
- 21 логическая строка.

Таким образом было проведена оценка кода метрикой SLOC.

4.4.2 Метрика Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы [35]. Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

– NUOprtr (Number of Unique Operators) – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);

– NUOprnd (Number of Unique Operands) – число уникальных операндов программы (словарь операндов);

– Noprtr (Number of Operators) – общее число операторов в программе;

– Noprnd (Number of Operands) – общее число операндов в программе.

В таблице 32 перечислим операторы.

Таблица 32 – Операторы

Оператор	Количество вхождений
async	7
=	8
‘’	7
return	7
{}	7
;	7
export	7
get	7

Продолжение таблицы32

Оператор	Количество вхождений
${}	14
Import from	2
const	8
n1 = 11	N1 = 81

В таблице 32 перечислим операторы.

Таблица 33 – Операнды

Операнд	Количество вхождений
getTypes	1
getEvents	1
api	8
getSubEvents	1
getSport	1
getMatches	1
getMatch	1
getSportInfo	1
id	12
axios	7
n2 = 10	N2 = 34

Расчет длины программы:

$$N = N1 + N2 = 81 + 34 = 115,$$

где N1 – количество вхождений операторов;

N2 – количество вхождений операндов.

Расчет словаря:

$$n = n1 + n2 = 11 + 10 = 21,$$

где n1 – количество операторов;

n2 – количество операндов.

Расчет объема программы:

$$V = n \cdot log_2(n) = 115 \cdot 4{,}392 = 505{,}08,$$

где N – длина программы;

n – словарь программы.

Расчет сложности программы:

$$D = \left(\frac{n1}{2}\right) \cdot \left(\frac{N2}{n2}\right) = \left(\frac{11}{2}\right) \cdot \left(\frac{34}{10}\right) = 18{,}7,$$

где n1 – количество операторов;

N2 – количество вхождений операндов;

n2 – количество операндов.

Уровень качества программирования, основанный лишь на параметрах реальной программы без учета теоретических параметров:

$$L' = \frac{2 \cdot n2}{n1 \cdot N2} = \frac{2 \cdot 10}{11 \cdot 34} = 0{,}053,$$

где n1 – количество операторов;

N2 – количество вхождений операндов;

n2 – количество операндов.

Сложность понимания программы:

$$E_c = \frac{V}{L' \cdot 2} = \frac{505{,}08}{0{,}053 \cdot 2} = 4764{,}91,$$

где V – объем программы;

L’ – уровень качества программирования, основанный лишь на параметрах реальной программы без учета теоретических параметров.

Полученные результаты сложности показывают, что разработанный код имеет достаточно высокий уровень качества, что позволит легко поддерживать и изменять его при поддержке проекта.

Заключение

В результате выполнения выпускной квалификационной работы было разработано и протестировано web-приложение для мониторинга и анализа результатов спортивных событий.

На стадии анализа требований был проведен сравнительный анализ аналогичных web-приложений, были выявлены функциональные и нефункциональные требования и проведен анализ целевой аудитории.

Для проекта было осуществлено структурное и календарное планирование, а также оценена стоимость разработки приложения. По итогу на реализацию проекта необходимо 50 дней и 123 980 рублей.

На стадии проектирования приложения была разработана концептуальная и логическая модели базы данных и выстроена архитектура программного продукта. А также был спроектирован макет приложения в редакторе Figma.

Реализация серверной части происходила на в среде разработки PyCharm Community на языке программирования python с использованием фреймворка Django. В качестве системы управления базами данных была выбрана система PostgreSQL, а взаимодействия с ней производились при помощи средств ORM фреймворка Django.

Клиентская часть разрабатывалась в текстовом редакторе Visual Studio Code на языке javascript с использованием библиотеки React и Material UI.

Приложение тестировалось двумя видами тестирования. Серверная часть прошла модульное тестирования, а клиентская часть – функциональное. Все тесты были пройдены.

В дальнейшем планируется разработка мобильного клиента, а также расширение функционала приложения.

Таким образом, задача, поставленная по реализации web-приложения для мониторинга и анализа результатов спортивных событий, была выполнена в полном объеме.

Список использованных источников

1. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. URL: http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf (дата обращения: 23.04.2022).

2. Отчет Digital 2022: The Russian Federation о состоянии социальных сетей и интернета в Российской Федерации [Электронный ресурс]. URL: https://datareportal.com/reports/digital-2022-russian-federation (дата обращения: 23.04.2022).

3. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 23.04.2022).

4. Доля граждан, систематически занимающихся физической культурой и спортом // сайт Федеральной Службы государственной статистики [Электронный ресурс]. URL: https://www.fedstat.ru/indicator/59266 (дата обращения: 23.04.2022).

5. Распоряжение правительства РФ от 03.06.2019 «Об утверждении Стратегии развития спортивной индустрии до 2035 года» № 1188 [Электронный ресурс]. URL: http://www.consultant.ru/document/cons_doc_LAW_326728/ (дата обращения: 23.04.2022).

6. О состоянии спортивной инфраструктуры в России // Сайт спортивной направленности [Электронный ресурс]. URL: http://sportfiction.ru/articles/o-sostoyanii-sportivnoy-infrastruktury-v-rossii/ (дата обращения: 23.04.2022)

7. Олимпийская система // Сайт Академик [Электронный ресурс]. URL: https://dic.academic.ru/dic.nsf/ruwiki/1075016 (дата обращения: 23.04.2022)

8. Чемпионат // Статья Википедии [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Чемпионат_(сайт) (дата обращения: 23.04.2022)

9. Flashscore // Описание приложения [Электронный ресурс]. URL: https://trashbox.ru/link/myscore-android (дата обращения: 23.04.2022)

10. Спорт-экспресс // Сайт газеты [Электронный ресурс]. URL: https://www.sport-express.ru/ (дата обращения: 23.04.2022)

11. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2е издание / Д. Арлоу, И. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.

12. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.

13. Кудрявцев, Е.М. Методы сетевого планирования и управления проектом / Е.М. Кудрявцев. – М.: Книга по Требованию, 2009. – 238 с.

14. Баев, Л.А. Сетевые методы планирования и управления: методические указания к курсовому проекту / составители: Л.А. Баев, С.Ю. Лелекова, Н.С. Дзензелюк. – Челябинск: Издательский центр ЮУрГУ, 2010. – 24 с.

15. Скороход, С.В.Управление проектами средствами Microsoft Project. Курс лекций. / С.В. Скороход. – Издательство: Интуит НОУ – 2016. – 319 стр.

16. Новиков, Ф.А. Учебно-методическое пособие по дисциплине Управление проектами и разработкой программного ПО / Ф.А. Новиков, Э.А. Опалева, Е.О. Степанов. – Спб. : НИУ ИТМО, 2012. – 256 с.

17. Скороход, С.В. Управление проектами средствами Microsoft Project. Курс лекций. / С.В. Скороход. – М.: Интуит НОУ, 2016. – 319 стр.

18. Куперштейн, В.И. Microsoft Project 2013 в управлении проектами / В.И. Куперштейн. – СПб:БХВ-Петербург, 2013. – 432 с.

19. Джонсон, Т., Четфилд К. Microsoft Project 2016. Шаг за шагом / Т.Джонсон, К.Четфилд. – М., Изд-во "КТК Галактика", 2018. – 566 с.

20. Акатова, Н.А. Планирование и контроль выполнения проектов с использованием MS Project / Н.А. Акатова, М.Г. Коротких. – М.: МИСиС, 2019. – 74 с.

21. Унифицированный процесс Rational (RUP) [Электронный ресурс]. URL: https://www.internet-technologies.ru/articles/newbie/unificirovannyy-process-rational-rup.html (дата обращения 21.05.2022)

22. UML – диаграммы классов. [Электронный ресурс]. – URL: https://prog-cpp.ru/uml-classes/ (дата обращения 21.05.2022)

23. Документация фреймворка Django [Электронный ресурс]. URL: https://django.fun/docs/django/ru/3.2/topics/signals/ (дата обращения 21.05.2022)

24. Введение в REST API [Электронный ресурс]. URL: https://habr.com/ru/post/483202/ (дата обращения 21.05.2022)

25. Документация библиотеки PyTest [Электронный ресурс]. URL: https://docs.pytest.org/en/7.1.x/ (дата обращения 20.05.2021)

26. Документация библиотеки React [Электронный ресурс]. URL: https://docs.pytest.org/en/7.1.x/ (дата обращения 20.05.2021)

27. Документация библиотеки MUI [Электронный ресурс]. URL: https://mui.com/ (дата обращения 20.05.2021)

28. Двухуровневая клиент-серверная архитектура [Электронный ресурс]. URL: https://studopedia.ru/7_95326_dvuhurovnevaya-klient-servernaya-arhitektura.html (дата обращения 20.05.2021)

29. Что такое JSON? [Электронный ресурс]. URL: https://habr.com/ru/post/554274/ (дата обращения 20.05.2021)

30. Документация case-средства Visual Paradigm [Электронный ресурс]. URL: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/ (дата обращения 20.05.2021)

31. Single Page Application [Электронный ресурс]. URL: https://thecode.media/spa/ (дата обращения 21.05.2022)

32. Куликов, С.С. Тестирование программного обеспечения. Базовый курс (3-е изд.) / С.С. Куликов – Минск: EPAM Systems, 2021. – 300 с.

33. Метрика программного обеспечения [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Метрика_программного_обеспечения (дата обращения 20.05.2021)

34. Количество строк кода [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Количество_строк_кода (дата обращения 20.05.2021)

35. Метрика сложности Холстеда [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Halstead_complexity_measures (дата обращения 20.05.2021)

Приложение А.

ГЛОССАРИЙ

(справочное)

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

RUP (Rational Unified Process) – методология разработки программного обеспечения, которая использует итеративную модель разработки.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, основанный на технологии «клиент-сервер».

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение, или преобразование) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

СУБД – система управления базами данных.

Приложение Б.
ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

Листинг Б.1 – Файл models,py

```
from django.db import models
from django.contrib.auth.models import User

# Create your models here.
from django.db.models import ForeignKey


class Sport_type(models.Model):
    name = models.CharField(max_length=255)
    description = models.CharField(max_length=255)
    count = models.IntegerField(default=0)

    def __str__(self):
        return self.name


class TB_user(models.Model):
    name = models.CharField(max_length=255)

    def __str__(self):
        return self.name


class System(models.Model):
    name = models.CharField(max_length=255)
    description = models.CharField(max_length=255)

    def __str__(self):
        return self.name


class Event(models.Model):
    name = models.CharField(max_length=255, unique=True)
    description = models.CharField(max_length=255)
    date_begin = models.DateField()
    date_finish = models.DateField()
    country = models.CharField(max_length=255)
    city = models.CharField(max_length=255)
    status = models.CharField(max_length=255)
    admin_id: ForeignKey = models.ForeignKey(User, on_delete=models.CASCADE)
    system_id = models.ForeignKey(System, on_delete= models.CASCADE)
    sport_type_id = models.ForeignKey(Sport_type, on_delete= models.CASCADE)
```

```
    def __str__(self):
        return self.name


class Subevent(models.Model):
    name = models.CharField(max_length=255)
    description = models.CharField(max_length=255)
    event_id = models.ForeignKey(Event, on_delete=models.CASCADE)

    def __str__(self):
        return self.name


class Participant(models.Model):
    name = models.CharField(max_length=255)
    country = models.CharField(max_length=255)
    city = models.CharField(max_length=255)
    progress = models.CharField(max_length=255)
    result = models.CharField(max_length=255)
    subevent_id = models.ForeignKey(Subevent, on_delete=models.CASCADE)

    def __str__(self):
        return self.name


class Match(models.Model):
    number = models.IntegerField(default=0)
    date = models.DateField()
    subevent_id = models.ForeignKey(Subevent, on_delete=models.CASCADE)

    def __str__(self):
        return str(self.number)


class Participant_Match(models.Model):
    participant_id = models.ForeignKey(Participant, on_delete=models.CASCADE)
    match_id = models.ForeignKey(Match, on_delete=models.CASCADE)
    score = models.IntegerField(default=0)


class User_Event(models.Model):
        tb_user_id = models.ForeignKey(TB_user, on_delete=models.CASCADE)
```

```
        event_id = models.ForeignKey(Event, on_delete=models.CASCADE)
```

Листинг Б.2 – Файл views,py

```
from django.shortcuts import render
from rest_framework import serializers
from rest_framework.views import APIView
from rest_framework.response import Response
from . import models
from django.http import HttpResponseNotFound
from datetime import date


class EventSerializator(serializers.ModelSerializer):
    class Meta:
        model = models.Event
        fields = ['pk', 'name', 'description', 'date_finish', 'date_begin',
'country', 'city', 'status', 'admin_id', 'system_id', 'sport_type_id']

    def update(self, instance, validated_data):
        instance.name = validated_data.get('name', instance.name)
        instance.description = validated_data.get('description',
instance.description)
        instance.date_finish = validated_data.get('date_finish',
instance.date_finish)
        instance.date_begin = validated_data.get('date_begin',
instance.date_begin)
        instance.country = validated_data.get('country', instance.country)
        instance.city = validated_data.get('city', instance.city)
        instance.status = validated_data.get('status', instance.status)
        instance.admin_id = validated_data.get('admin_id', instance.admin_id)
        instance.system_id = validated_data.get('system_id',
instance.system_id)
        instance.sport_type_id = validated_data.get('sport_type_id',
instance.sport_type_id)
        instance.save()
        return instance

    def create(self, validated_data):
        instance = models.Event.objects.create(
            name=validated_data.get('name'),
            description=validated_data.get('description'),
            date_finish=validated_data.get('date_finish'),
            date_begin=validated_data.get('date_begin'),
            country=validated_data.get('country'),
            city=validated_data.get('city'),
            status=validated_data.get('status'),
            admin_id=validated_data.get('admin_id'),
            system_id=validated_data.get('system_id'),
            sport_type_id=validated_data.get('sport_type_id')
        )


class SubeventSerializator(serializers.ModelSerializer):
```

```
    class Meta:
        model = models.Subevent
        fields = ['pk', 'name', 'description', 'event_id']


class Sport_type_Serializator(serializers.ModelSerializer):
    class Meta:
        model = models.Sport_type
        fields = ['pk', 'name', 'description', 'count']


class ParticipantSerializator(serializers.ModelSerializer):
    class Meta:
        model = models.Participant
        fields = ['pk', 'name', 'country', 'city', 'progress', 'result',
'subevent_id']


class MatchSerializator(serializers.ModelSerializer):
    class Meta:
        model = models.Match
        fields = ['pk', 'number', 'date', 'subevent_id']


class SportTypes(APIView):
    def get(self, *args, **kwargs):
        all_types = models.Sport_type.objects.all()
        serialized_types = Sport_type_Serializator(all_types, many=True)
        return Response(serialized_types.data)


class FindSports(APIView):
    def get(self, request, name, format=None):
        st = models.Sport_type.objects.filter(name=name)
        if not st:
            return HttpResponseNotFound()
        serialized_types = Sport_type_Serializator(st, many=True)
        return Response(serialized_types.data)


class Events(APIView):
    def get(self, request, st, format=None):
        events = models.Event.objects.filter(sport_type_id=st)
        if not events:
            return HttpResponseNotFound()

        serialized_events = EventSerializator(events, many=True)
        return Response(serialized_events.data)


class Event(APIView):
    def get(self, request, pk, format=None):
        event = models.Event.objects.filter(pk=pk)
        if not event:
            return HttpResponseNotFound()
```

```
        serialized_events = EventSerializator(event, many=True)
        return Response(serialized_events.data)




class Subevents(APIView):
    def get(self, request, ev, format=None):
        subevents = models.Subevent.objects.filter(event_id=ev)
        if not subevents:
            return HttpResponseNotFound
        ser_subevents = SubeventSerializator(subevents, many=True)
        return Response(ser_subevents.data)


class Participants(APIView):
    def get(self, request, sid, format=None):
        participants = models.Participant.objects.filter(subevent_id=sid)
        if not participants:
            return HttpResponseNotFound
        ser_participants = ParticipantSerializator(participants, many=True)
        return Response(ser_participants.data)


class Participant(APIView):
    def get(self, request, pk, format=None):
        participant = models.Participant.objects.filter(pk=pk)
        if not participant:
            return HttpResponseNotFound

        ser_participant = ParticipantSerializator(participant, many=True)
        return Response(ser_participant.data)


class SportData(APIView):
    def get(self, request, pk):
        sport = models.Sport_type.objects.get(pk=pk)
        if not sport:
            return HttpResponseNotFound
        else:
            count_event = sport.count
            today_year = date.today().year
            count_events_in_year = []
            avg_count_participant_in_year = []
            for i in range(5):
                events = models.Event.objects.filter(sport_type_id=pk,

date_begin__gt=date(today_year - i, 1, 1),

date_begin__lt=date(today_year - i + 1, 1, 1)
                                                     )

                count_events_in_year.append(len(events))
                list_counts_participants_in_year = []
                for event in events:
```

```
                    subevents =
models.Subevent.objects.filter(event_id=event.pk)
                    count = 0
                    for subevent in subevents:
                        count +=
models.Participant.objects.filter(subevent_id=subevent.id).count()
                    list_counts_participants_in_year.append(count)

                if (len(events) != 0):
                    avg_count_participant_in_year.append(
                        sum(list_counts_participants_in_year) /
len(list_counts_participants_in_year)
                    )
                else:
                    avg_count_participant_in_year.append(
                        0.0
                    )


            events_cur_year = models.Event.objects.filter(sport_type_id=pk,
date_begin__gt=date(today_year, 1, 1), date_begin__lt=date(today_year + 1, 1,
1))
            count_event_cur_year = len(events_cur_year)
            events_last_year = models.Event.objects.filter(sport_type_id=pk,
date_begin__gt=date(today_year-1, 1, 1), date_begin__lt=date(today_year, 1,
1))
            count_event_last_year = len(events_last_year)
            list_counts_participants_cur_year = []
            for event in events_cur_year:
                subevents = models.Subevent.objects.filter(event_id=event.pk)
                count = 0
                for subevent in subevents:
                    count +=
models.Participant.objects.filter(subevent_id=subevent.id).count()
                list_counts_participants_cur_year.append(count)
            if (len(list_counts_participants_cur_year) != 0):
                avg_count_participant_cur_year =
sum(list_counts_participants_cur_year) /
len(list_counts_participants_cur_year)
            else:
                avg_count_participant_cur_year = 0
            list_counts_participants_last_year = []
            for event in events_last_year:
                subevents = models.Subevent.objects.filter(event_id=event.pk)
                count = 0
                for subevent in subevents:
                    count +=
models.Participant.objects.filter(subevent_id=subevent.id).count()
                list_counts_participants_last_year.append(count)
            if (len(list_counts_participants_last_year) != 0):
                avg_count_participant_last_year =
sum(list_counts_participants_last_year) /
len(list_counts_participants_last_year)
            else:
                avg_count_participant_last_year = 0
            return Response({
```

```
                "name": f"{sport.name}",
                "count_events": f"{count_event}",
                "count_events_cur_year": f"{count_event_cur_year}",
                "count_events_last_year": f"{count_event_last_year}",
                "avg_participants_cur_year":
f"{avg_count_participant_cur_year}",
                "avg_participants_last_year":
f"{avg_count_participant_last_year}",
                "count_events_in_year": count_events_in_year,
                "avg_count_participant_in_year":
avg_count_participant_in_year
            })



class CountSubevents(APIView):
    def get(self, request, evid, format=None):
        count = models.Subevent.objects.filter(event_id=evid).count()
        return Response({"message": f"{count}"})


class CountParticipants(APIView):
    def get(self, request, evid, format=None):
        subevents = models.Subevent.objects.filter(event_id=evid)
        count = 0
        for subevent in subevents:
            count +=
models.Participant.objects.filter(subevent_id=subevent.id).count()
        return Response({"message": f"{count}"})


class Matches(APIView):
    def get(self, request, sid, format=None):
        matches = models.Match.objects.filter(subevent_id=sid)
        matches_lst = []
        for match in matches:
            links =
models.Participant_Match.objects.filter(match_id=match.pk)
            data = {"pk": match.pk,
                    "number": match.number,
                    "date": match.date,
                    "first_participant": links[0].participant_id.name,
                    "second_participant": links[1].participant_id.name,
                    "score": f"{links[0].score}:{links[1].score}"
                    }
            matches_lst.append(data)
        return Response(matches_lst)


class Match(APIView):
    def get(self,request, pk, format=None):
        match = models.Match.objects.get(pk=pk)
        links = models.Participant_Match.objects.filter(match_id=match.pk)
        data = {"pk": match.pk,
                "number": match.number,
```

```
                "date": match.date,
                "first_participant": links[0].participant_id.name,
                "second_participant": links[1].participant_id.name,
                "score": f"{links[0].score}:{links[1].score}"
                }
        return Response(data)


class AvgCounts(APIView):
    def get(self, request, pk, format=None):
        event = models.Event.objects.get(pk=pk)
        subevents = models.Subevent.objects.filter(event_id=event.pk)
        count = []
        for subevent in subevents:

count.append(models.Participant.objects.filter(subevent_id=subevent.id).count
())
        avg = sum(count) / len(count)
        return Response({"avg_count": [f"{avg}", 0]})
```

Листинг Б.3 – Файл signals,py

```
from django.db.models.signals import post_save, post_delete
from django.dispatch import receiver
from .models import Event, Sport_type


@receiver(post_save, sender=Event)
def create_event_handler(sender, instance, created, **kwargs):
    if created:
        sport_id = instance.sport_type_id.id
        s_type = Sport_type.objects.get(pk=sport_id)
        s_type.count = Event.objects.filter(sport_type_id=sport_id).count()
        s_type.save()


@receiver(post_delete, sender=Event)
def delete_event_handler(sender, instance, **kwargs):
    sport_id = instance.sport_type_id.id
    s_type = Sport_type.objects.get(pk=sport_id)
    s_type.count = Event.objects.filter(sport_type_id=sport_id).count()
    s_type.save()
```

Листинг Б.4 – Файл test_requests,py

```
from .data_fetch import Requester
from .local_settings import EVENTS_API_URL, SUBEVENTS_API_URL
from .local_settings import PARTICIPANTS_API_URL, PARTICIPANT_API_URL,
MATCHES_API_URL, MATCH_API_URL
from .local_settings import EVENT_API_URL

requester = Requester()

def test_get_events():
    req = requester.requesting(EVENTS_API_URL, pk="1")
```

```
    assert req[0]['pk'] == 1


def test_get_event():
    req = requester.requesting(EVENT_API_URL, pk="1")
    assert req == [{
        "pk": 1,
        "name": "Первенство Пензенской области по тхэквондо",
        "description": "Первенство Пензенской области по тхэквондо среди
кадетов и юниоров",
        "date_finish": "2022-01-05",
        "date_begin": "2021-12-30",
        "country": "Россия",
        "city": "Пенза",
        "status": "Идет",
        "admin_id": 1,
        "system_id": 1,
        "sport_type_id": 1
        }
    ]


def test_get_subevents():
    req = requester.requesting(SUBEVENTS_API_URL, pk="1")
    assert req[0] == {"pk": 1, "name": "Кадеты. Весовая категория  -41кг",
                      "description": "Поединки среди кадетов весовой
категории -41кг",
                      "event_id": 1}


def test_get_participants():
    req = requester.requesting(PARTICIPANTS_API_URL, pk="1")
    assert req[0] == {"pk": 1,
                      "name": "Ларин Дмитрий Константинович",
                      "country": "Россия",
                      "city": "Пенза",
                      "progress": "1/4",
                      "result": "1/4",
                      "subevent_id": 1}


def test_get_participant():
    req = requester.requesting(PARTICIPANT_API_URL, pk="1")
    assert req[0] == {"pk": 1,
                      "name": "Ларин Дмитрий Константинович",
                      "country": "Россия",
                      "city": "Пенза",
                      "progress": "1/4",
                      "result": "1/4",
                      "subevent_id": 1}


def test_get_matches():
    req = requester.requesting(MATCHES_API_URL, pk="1")
    assert req[0] == {"pk": 4,
                      "number": 1,
```

```
                    "date": "2022-01-03",
                    "first_participant": "Ларин Дмитрий Константинович",
                    "second_participant": "Смирнов Олег Григорьевич",
                    "score": "8:5"}


def test_get_match():
    req = requester.requesting(MATCH_API_URL, pk="4")
    assert req == {"pk": 4,
                   "number": 1,
                   "date": "2022-01-03",
                   "first_participant": "Ларин Дмитрий Константинович",
                   "second_participant": "Смирнов Олег Григорьевич",
                   "score": "8:5"}
```

Листинг Б.5 – Файл urls,py

```
from django.contrib import admin
from django.urls import path
from tournament import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('types/', views.SportTypes.as_view()),
    path('events/<int:st>', views.Events.as_view()),
    path('event/<int:pk>', views.Event.as_view()),
    path('findsports/<str:name>', views.FindSports.as_view()),
    path('subevents/<int:ev>', views.Subevents.as_view()),
    path('participants/<int:sid>', views.Participants.as_view()),
    path('participant/<int:pk>', views.Participant.as_view()),
    path('countsubevent/<int:evid>', views.CountSubevents.as_view()),
    path('countparticipants/<int:evid>', views.CountParticipants.as_view()),
    path('matches/<int:sid>', views.Matches.as_view()),
    path('match/<int:pk>', views.Match.as_view()),
    path('sport/<int:pk>', views.SportData.as_view()),
    path('avg/<int:pk>', views.AvgCounts.as_view()),
]
```

Листинг Б.6 – Файл admin,py

```
from django.contrib import admin
from . import models

class EventsAdmin(admin.ModelAdmin):
    list_display = ['pk', 'name', 'description', 'country', 'city', 'status',
'date_finish', 'date_begin']
    list_editable = ['name', 'description', 'status', 'date_finish']


class ParticipantAdmin(admin.ModelAdmin):
    list_display = ['pk', 'name', 'country', 'city', 'progress', 'result',
'subevent_id']
    list_editable = ['name', 'country', 'city', 'progress', 'result',
'subevent_id']
```

```
class SubeventAdmin(admin.ModelAdmin):
    list_display = ['pk', 'name', 'description', 'event_id']
    list_editable = ['name', 'description', 'event_id']


class Sport_typeAdmin(admin.ModelAdmin):
    list_display = ['pk', 'name', 'description', 'count']
    list_editable = ['name', 'description', 'count']


class SystemAdmin(admin.ModelAdmin):
    list_display = ['pk', 'name', 'description']
    list_editable = ['name', 'description']


class MatchAdmin(admin.ModelAdmin):
    list_display = ['pk', 'number', 'date', 'subevent_id']
    list_editable = ['number', 'date', 'subevent_id']


class Participant_matchAdmin(admin.ModelAdmin):
    list_display = ['pk', 'participant_id', 'match_id', 'score']
    list_editable = ['participant_id', 'match_id', 'score']

# Register your models here.


admin.site.register(models.Event, EventsAdmin)
admin.site.register(models.Participant, ParticipantAdmin)
admin.site.register(models.Subevent, SubeventAdmin)
admin.site.register(models.Sport_type, Sport_typeAdmin)
admin.site.register(models.System, SystemAdmin)
admin.site.register(models.Match, MatchAdmin)
admin.site.register(models.Participant_Match, Participant_matchAdmin)
```

Приложение В.
ИСХОДНЫЙ КОД WEB-ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг В.1 – Файл index.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <link rel="icon" type="image/svg+xml" href="/src/favicon.svg" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Champion</title>

  </head>

  <body>

    <div id="root"></div>

    <script type="module" src="/src/main.jsx"></script>

  </body>

</html>
```

Листинг В.2 – Файл App.jsx

```
import { useEffect, useState } from 'react';

import { Tabs, Tab, Grid, Container, listItemSecondaryActionClasses } from
'@mui/material';

import { Link } from 'react-router-dom';

import { Outlet } from 'react-router-dom';

import './App.css';

import { useLocation } from 'react-router-dom';


function App() {

  const [value, setValue] = useState('main');

  const location = useLocation();

  const handleChange = (event, newValue) => {

    setValue(newValue);

  };


  useEffect(() => {

    if (location.pathname.includes('main')) setValue('main');
```

```
    if (location.pathname.includes('events')) setValue('events');
    if (location.pathname.includes('analysis')) setValue('anal');
  }, [location.pathname]);

  return (<>
    <nav className='nav'>
      <Grid
        container
        direction="row"
        justifyContent="center"
        alignItems="center"
      >
        <Tabs
          value={value}
          onChange={handleChange}
          textColor="primary"
          indicatorColor="primary"
        >
          <Tab value="main" label="Главная" to='/' component={Link} />
          <Tab value="events" label="События" to='/events' component={Link}
/>
          <Tab value="anal" label="Анализ" to='/analysis' component={Link}/>
        </Tabs>
      </Grid>
    </nav>
    <main>
      <Container>
        <Outlet/>
      </Container>
    </main>
  </>
  );
```

```
}


export default App;
```

Листинг В.3 – Файл Home.jsx

```
import { useEffect, useState } from 'react';

import { getTypes } from '../api';

import Search from '../components/Search';


function Home() {

  return (

    <>

      <div>

        <div className='text--primary'>Champion - web-приложение для
мониторинга и анализа результатов спортивных событий</div>

        <div className='text--secondary'>для просмотра результатов
воспользуйтесь вкладкой “События”</div>

        <div className='text--secondary'>для просмотра статистики и анализа
по видам спорта воспользуйтесь вкладкой “Анализ”</div>

      </div>

      {/* <div>{types.length && types.map((el) => <>{el.name}</>)}</div> */}

    </>

  );

}

export default Home;
```

Листинг В.4 – Файл Events.jsx

```
import { Backdrop, Button, CircularProgress, Modal } from '@mui/material';

import { useState } from 'react';

import Event from '../components/Event';

import Search from '../components/Search';

import { useEvents, useMatch, useSubEvent, useMatches } from '../hooks';

import ArrowBackIcon from '@mui/icons-material/ArrowBack';

import styles from './style.module.css';

import SubEvent from '../components/SubEvent';
```

```
import Matches from '../components/Matches';
import { useNavigate } from 'react-router-dom';

const Events = () => {
  const [state, setState] = useState(1);
  const [modalIsOpen, setModalIsOpen] = useState(false);
  const [selectedEventId, setSelectedEventId] = useState(null);
  const [selectedSubEventId, setSelectedSubEventId] = useState(null);
  const [selectedMatchId, setSelectedMatchId] = useState(null);

  const navigator = useNavigate();

  const { types, events, searchValue, handleChange, isEventLoading } =
    useEvents();
  const { subEvents, subEventsIsLoading } = useSubEvent(selectedEventId);
  const { matches, matchesIsLoading, clearMatches } =
    useMatches(selectedSubEventId);
  const { match, matchIsLoading, clearMatch } = useMatch(selectedMatchId);

  const openEvent = (id) => {
    setState(2);
    setSelectedEventId(id);
  };
  const openSubEvent = (id) => {
    setState(3);
    setSelectedSubEventId(id);
  };

  const handleShowResult = (id) => {
    setModalIsOpen(true);
    setSelectedMatchId(id);
  };
```

```
  const handleCloseModal = () => {
    setModalIsOpen(false);
    setSelectedMatchId(null);
    clearMatch();
  };

  const changeType = (type) => {
    setState(1);
    handleChange(type);
  };

  return (
    <>
      {types.length ? (
        <Search
          types={types}
          onClick={changeType}
          value={searchValue}
        />
      ) : (
        <div className={styles.spin}>
          <CircularProgress />
        </div>
      )}
      {searchValue && (
        <div>
          <div className={styles.eventTitle}>
            <div>
              {state !== 1 ? (
                <Button className={styles.iconButton} onClick={() =>
setState(state - 1)}>
                  <ArrowBackIcon />
```

```
        </Button>
      ) : (
        <div></div>
      )}
    </div>
    {state === 1 ?
      searchValue.name.toUpperCase() :
      events.find((el) => el.pk === selectedEventId).name}
  </div>
  <div className={styles.divider} />
  {state === 1 ? (
    !isEventLoading ? (
      events.length ? (
        <div className={styles.eventList}>
          {events.map((event) => (
            <Event key={event.key} event={event} open={openEvent}/>
          ))}
        </div>
      ) : (
        <div className={styles.notFound}>
          <div>Не найдено событий</div>
          <div>Выберете другой вид спорта</div>
        </div>
      )
    ) : (
      <div className={styles.spin}>
        <CircularProgress />
      </div>
    )
  ) : (
    <></>
  )}
```

```
            {state === 2 ? (
              <div>
                {subEventsIsLoading ? (
                  <div className={styles.spin}>
                    <CircularProgress />
                  </div>
                ) : (
                  <div className={styles.subEventList}>
                    {subEvents.map((el) => (
                      <SubEvent key={el.pk} event={el} open={openSubEvent} />
                    ))}
                  </div>
                )}
              </div>
            ) : (
              <></>
            )}

            {state === 3 ? (
              <div>
                {matchesIsLoading ?
                  (<div className={styles.spin}>
                    <CircularProgress />
                  </div>) :
                  (matches.length ? <div className={styles.subEventList}>
                    {matches.map((el) => (
                      <Matches key={el.pk} event={el} open={handleShowResult}
/>

                    ))}
                  </div> : <div className={styles.notFound}>
                    <div>Не найдено событий</div>
```

```
                    <div>Выберете другой матч</div>
                  </div>)
                }
              </div>
            ) : (
              <></>
            )}
          </div>
        )}
        {modalIsOpen && match === null && <Backdrop open={true}>
          <CircularProgress />
        </Backdrop>}
        {modalIsOpen && match !== null ? (
          <Modal open={true} onClose={handleCloseModal}>
            <div className={styles.modalPaper}>
              <div className={styles.modalText1}>
                {match.first_participant} - {match.second_participant}
              </div>
              <div className={styles.modalTextScore}>{match.score}</div>
              <div className={styles.modalText2}>{match.date}</div>
              <div className={styles.modalText3}>{match.description}</div>
            </div>
          </Modal>
        ) : (
          <></>
        )}
      </>
    );
};

export default Events;
```

Листинг В.5 – Файл Analysis.jsx

```
import { CircularProgress } from '@mui/material';
import { useEffect, useState } from 'react';
import Search from '../components/Search';
import { useEvents } from '../hooks';
import styles from './style.module.css';
import { getSportInfo } from '../api';
import {
  BarChart,
  Bar,
  Cell,
  XAxis,
  YAxis,
  CartesianGrid,
  Tooltip,
  Legend,
  ResponsiveContainer,
} from 'recharts';

function Analysis() {
  const [sport, setSport] = useState(null);
  const [chartData, setData] = useState(null);
  const [chartData2, setData2] = useState(null);
  const [isLoading, setIsLoading] = useState(false);

  const { types, events, searchValue, handleChange, isEventLoading } =
    useEvents();

  useEffect(() => {
    if (!searchValue) return;
    setData(null);
    setData2(null);
```

```
  setIsLoading(true);
  getSportInfo(searchValue.pk)
      .then(({ data }) => {
        setSport(data);
      })
      .finally(() => setIsLoading(false));
}, [searchValue]);

useEffect(() => {
  if (sport) {
    setData(
        [
          { name: '2022', value: sport.count_events_in_year[0] },
          { name: '2021', value: sport.count_events_in_year[1] },
          { name: '2020', value: sport.count_events_in_year[2] },
          { name: '2019', value: sport.count_events_in_year[3] },
          { name: '2018', value: sport.count_events_in_year[4] },
        ].reverse());
    setData2(
        [
          { name: '2022', value: sport.avg_count_participant_in_year[0] },
          { name: '2021', value: sport.avg_count_participant_in_year[1] },
          { name: '2020', value: sport.avg_count_participant_in_year[2] },
          { name: '2019', value: sport.avg_count_participant_in_year[3] },
          { name: '2018', value: sport.avg_count_participant_in_year[4] },
        ].reverse());
  }
}, [sport]);

return (
  <>
    {types.length ? (
```

```
        <Search types={types} onClick={handleChange} value={searchValue} />
      ) : (
        <div className={styles.spin}>
          <CircularProgress />
        </div>
      )}
      {searchValue ? (
        <div style={{display: 'flex', flexDirection: 'column', alignContent:
'center', width: '100%',
          position: 'absolute',
          left: 0}}>
          <div className={styles.eventTitle}>
            {searchValue.name.toUpperCase()}
          </div>

          <div className={styles.divider} />
          <div className={styles.analPaper}>
            <div className={styles.analTitle}>Статистика вида спорта</div>
            <div className={styles.analTitle}>"{searchValue.name}"</div>
            {isLoading ? (
              <div className={styles.spin}>
                <CircularProgress />
              </div>
            ) : chartData ? (
              <div className={styles.analGraphs}>
                <div>
                  <div className={styles.analSubTitle}>
                  Количество событий всего: {sport.count_events}
                  </div>
                  <div className={styles.analSubTitle}>
                  Количество событий за текущий год:{' '}
                    {sport.count_events_cur_year}
```

```
                </div>
                <div
                  style={{ width: '700px', height: '500px', margin: 'auto'
}}
                >
                  <ResponsiveContainer width='100%' height='100%'>
                    <BarChart
                      width={500}
                      height={150}
                      data={chartData}
                      margin={{
                        top: 5,
                        right: 30,
                        left: 20,
                        bottom: 5,
                      }}
                    >
                      <CartesianGrid strokeDasharray='3 3' />
                      <XAxis
                        label={{
                          value: 'Год',
                          offset: -2,
                          position: 'insideBottom',
                        }}
                        name='Количество событий'
                        dataKey='name'
                      />
                      <YAxis
                        label={{
                          value: 'Количество событий',
                          angle: -90,
                          position: 'insideLeft',
```

```
                }}
              />
              <Tooltip />
              <Legend />
              <Bar
                offset='100px'
                name='Количество событий'
                dataKey='value'
                fill='#8884d8'
              />
            </BarChart>
          </ResponsiveContainer>
        </div>
      </div>
      <div>
        <div className={styles.analSubTitle}>
        Среднее количество участников за текущий год:{' '}
          {sport.avg_participants_last_year}
        </div>
        <div className={styles.analSubTitle}>
           
        </div>
        <div
          style={{ width: '700px', height: '500px', margin: 'auto'
}}
        >
          <ResponsiveContainer width='100%' height='100%'>
            <BarChart
              width={500}
              height={150}
              data={chartData2}
              margin={{
```

```
      top: 5,
      right: 30,
      left: 20,
      bottom: 5,
    }}
  >
    <CartesianGrid strokeDasharray='3 3' />
    <XAxis
      label={{
        value: 'Год',
        offset: -2,
        position: 'insideBottom',
      }}
      name='Количество участников'
      dataKey='name'
    />
    <YAxis
      label={{
        value: 'Количество участников',
        angle: -90,
        position: 'insideLeft',
      }}
    />
    <Tooltip />
    <Legend />
    <Bar
      offset='100px'
      name='Количество участников'
      dataKey='value'
      fill='#8884d8'
    />
  </BarChart>
```

```
                    </ResponsiveContainer>
                  </div>
                </div>
              </div>
            ) : (
              <div className={styles.analTitle}>Нет информации</div>
            )}
          </div>
        </div>
      ) : (
        <></>
      )}
    </>
  );
}
export default Analysis;
```

Листинг В.6 – Файл Event.jsx

```
import { Button } from '@mui/material';
import styles from './style.module.css';
import cn from 'classnames';

function Event({event, open}) {
  return (
    <div className={styles.eventPaper}>
      <div className={styles.eventTitle}>
        {event.name} ({event.country},{event.city})
        <div className={styles.eventSubTitle}>{event.description}</div>
      </div>
      <div className={styles.eventInfo}>
        <div>Даты: {event.date_begin} - {event.date_finish}</div>
        <div>Статус: {event.status}</div>
      </div>
```

```
      <div className={styles.eventBtns}>
        <Button className={cn(styles.watch, styles.eventBtn)} onClick={() =>
open(event.pk)} >Посмотреть</Button>
      </div>
    </div>
  );
}
export default Event;
```

Листинг В.7 – Файл Matches.jsx

```
import { Button } from '@mui/material';
import styles from './style.module.css';
import cn from 'classnames';

function Matches({event, open}) {
  return (
    <div className={styles.eventPaperMatch}>
      <div className={styles.eventTitle}>
        {event.number}. {event.first_participant} vs
{event.second_participant}
        <div className={styles.eventSubTitle}></div>
      </div>
      <div className={styles.eventBtns}>
        <Button className={cn(styles.stat, styles.eventBtn)}
onClick={()=>open(event.pk)}>Результат</Button>
      </div>
    </div>
  );
}

export default Matches;
```

Листинг В.8 – Файл Search.jsx

```
import { Autocomplete, Button, TextField } from '@mui/material';
import { useEffect, useState } from 'react';
```

```
import styles from './style.module.css';
import PropTypes from 'prop-types';

function Search({ types, onClick, value }) {
  const [fourTypeChips, setFourTypeChips] = useState([]);
  const [valueInput, setValueInput] = useState(null);
  const generateTypes = () => {
    const result = [];

    if (!types.length) {
      return [];
    }

    for (let i = 0; i < 4; i++) {
      result.push(<Button onClick={() => onClick(types[i])}
className={styles.btn}>{types[i].name}</Button>);
    }
    return result;
  };

  useEffect( () => {
    setFourTypeChips(generateTypes());
  }, [types]);
  useEffect( () => {
    setValueInput(value);
  }, [value]);

  return (
    <div className={styles.root}>
      <div className={styles.search}>
        <Autocomplete
          id="country-select-demo"
          getOptionLabel={(option) => option.name}
          value={valueInput}
          className={styles.searchInput}
```

```
          options={types}
          size='small'
          variant='outlined'
          onChange={(_, value) => onClick(value)}
          renderInput={(params) => <TextField {...params}
className={styles.searchInput} />}
        ></Autocomplete>
      </div>
      <div className={styles.btns}>
        {fourTypeChips}
      </div>
    </div>
  );
}

Search.propTypes ={
  types: PropTypes.array,
  onClick: PropTypes.func,
  value: { pk: PropTypes.number, name: PropTypes.string },
};
export default Search;
```

Листинг В.9 – Файл Search.jsx

```
import { Button } from '@mui/material';
import styles from './style.module.css';
import cn from 'classnames';

function SubEvent({event, open}) {
  return (
    <div className={styles.eventPaper}>
      <div className={styles.eventTitle}>
        {event.name}
        <div className={styles.eventSubTitle}>{event.description}</div>
      </div>
      <div className={styles.eventBtns}>
```

```
        <Button className={cn(styles.stat, styles.eventBtn)}
onClick={()=>open(event.pk)}>Результаты</Button>

      </div>

    </div>

  );

}

export default SubEvent;
```

Листинг В.10 – Файл style.module.css

```
.search {

    margin: 18px;

    display: flex;

    flex-direction: row;

    justify-content: center;

    gap: 9px;

}

.search .searchInput {

    height: 40px;

    width: 740px;

    background-color: white;

}


.root {

    display: flex;

    flex-direction: column;

}


.btns {

    margin: auto;

    display: flex;

    flex-direction: row;

    gap: 20px;

}
```

```
.btn {
    background-color: #d9d9d9 !important;
    color: black !important;
    border-radius: 25px !important;
    width: 183px;
    height: 50px;
}

.eventPaper {
    width: 900px;
    height: 190px;

    background: #ffffff;
    border: 2px solid #bd89ff;
    border-radius: 20px;

    display: flex;
    flex-direction: column;
    justify-content: space-between;
    padding-bottom: 20px;
}

.eventPaperMatch {
    width: 900px;
    height: 100px;

    background: #ffffff;
    border: 2px solid #bd89ff;
    border-radius: 20px;

    display: flex;
    flex-direction: column;
```

```
    justify-content: space-between;
    padding-bottom: 20px;
}

.eventTitle {
    font-family: "Inter";
    font-style: normal;
    font-weight: 400;
    font-size: 24px;
    line-height: 29px;
    text-align: center;
    margin-top: 10px;
}

.eventBtns {
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: center;
    gap: 32px;
}

.eventBtn {
    width: 240px;
}

.eventSubTitle {
    font-family: "Inter";
    font-style: normal;
    font-weight: 300;
    font-size: 16px;
    line-height: 20px;
```

```
    color: rgb(159, 159, 159);
}
.eventInfo {
    font-family: "Inter";
    font-style: normal;
    font-weight: 300;
    font-size: 16px;
    line-height: 20px;
    text-align: center;
}
.watch {
    background-color: #b6c1ff !important;
    color: black !important;
}
.stat {
    background-color: #53db9a !important;
    color: black !important;
}
```

Приложение Г.

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ WEB-ПРИЛОЖЕНИЯ

(рекомендуемое)

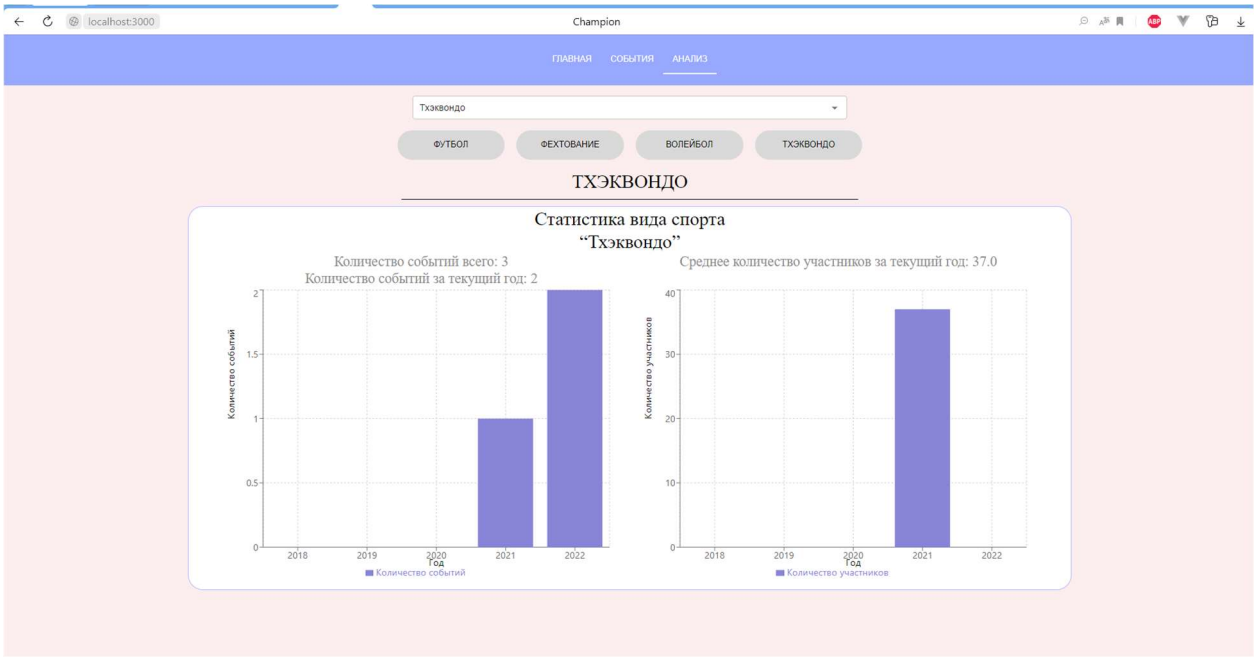


Рисунок Г.1 – Анализ по виду спорта тхэквондо

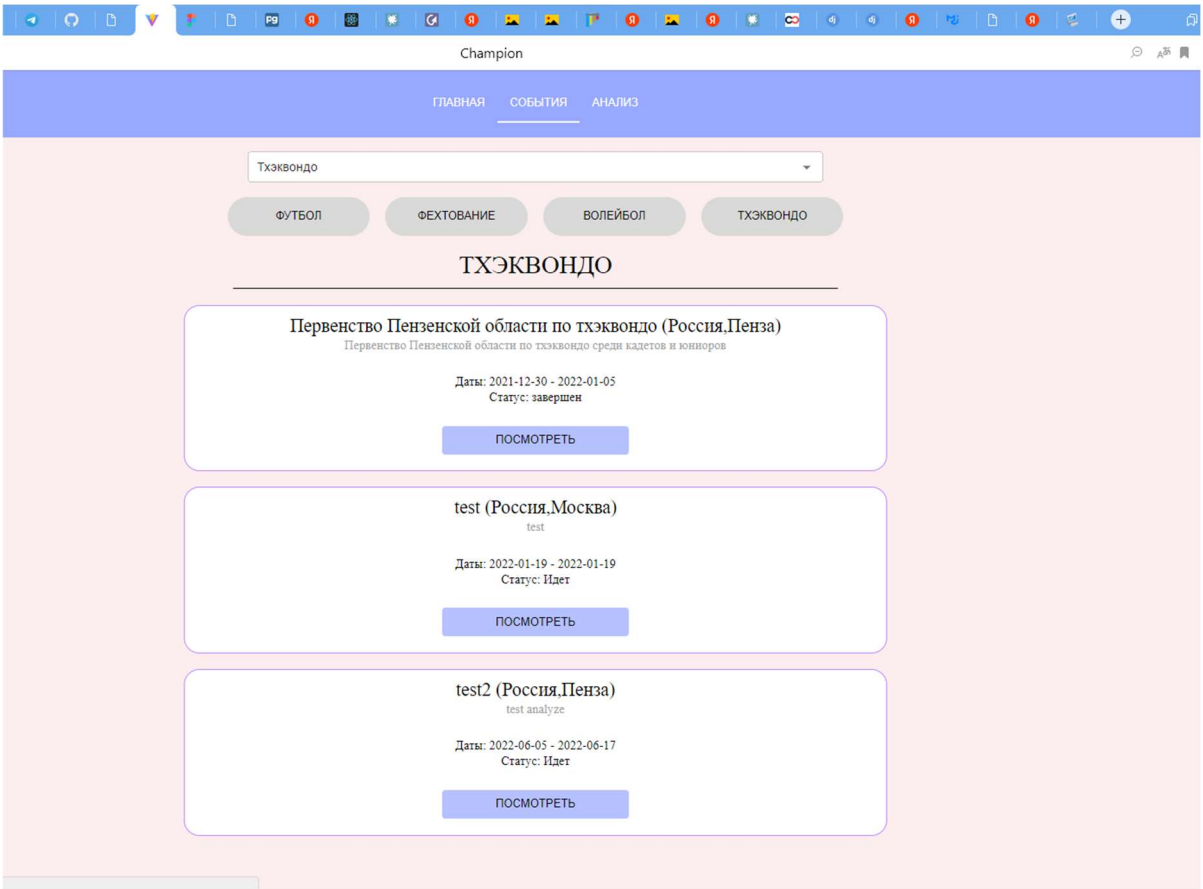


Рисунок Г.2 – Доступные события по виду спорта тхэквондо

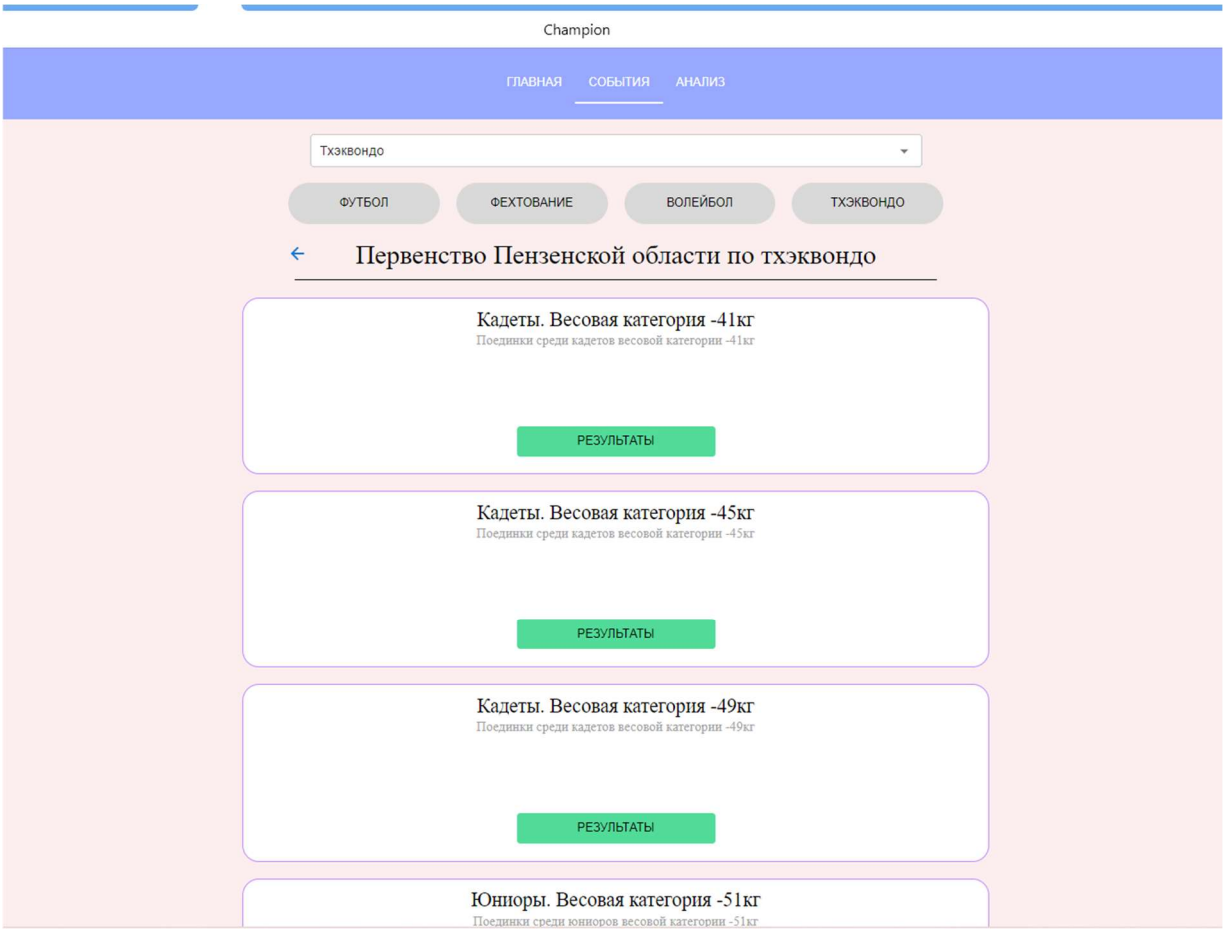


Рисунок Г.3 – Подсобытия Первенства области по тхэквондо

Рисунок Г.4 – Результаты по выбранной категории

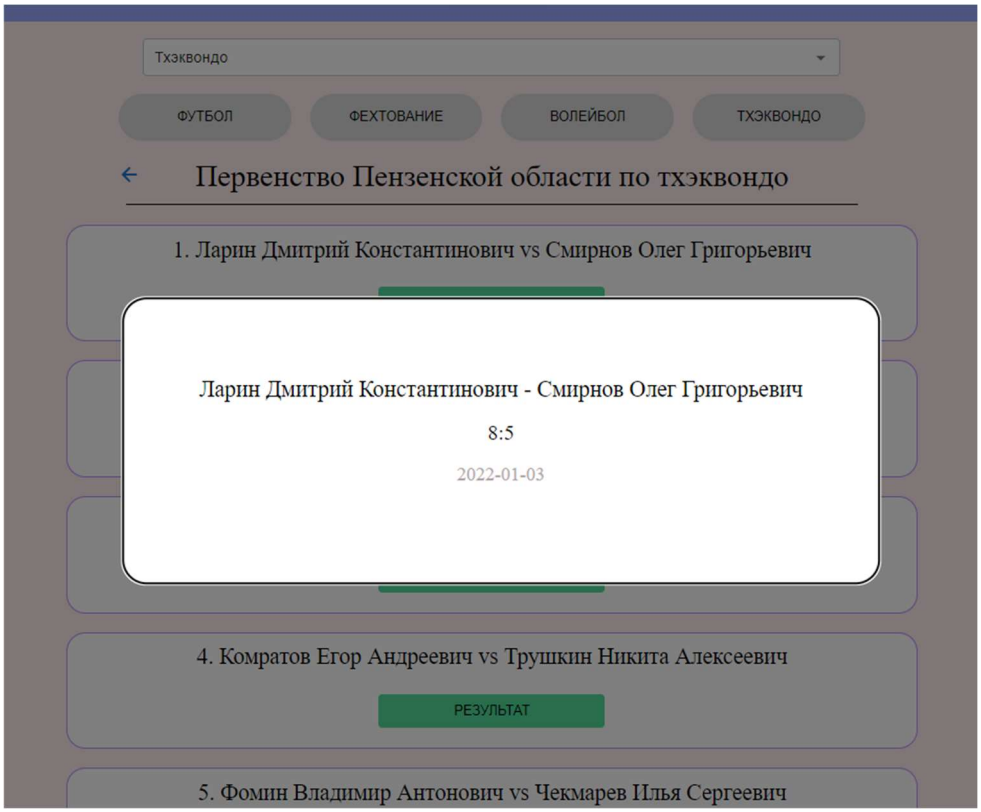


Рисунок Г.5 – Результат выбранного матча

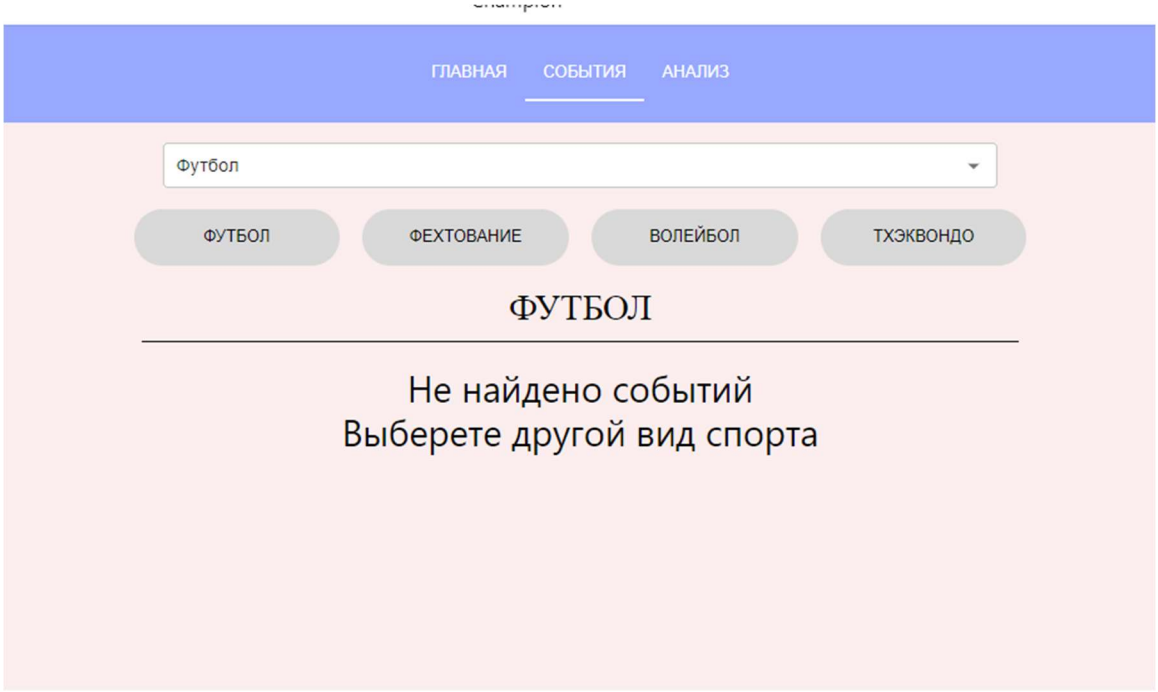


Рисунок Г.6 – Отсутствие событий по виду спорта «футбол»

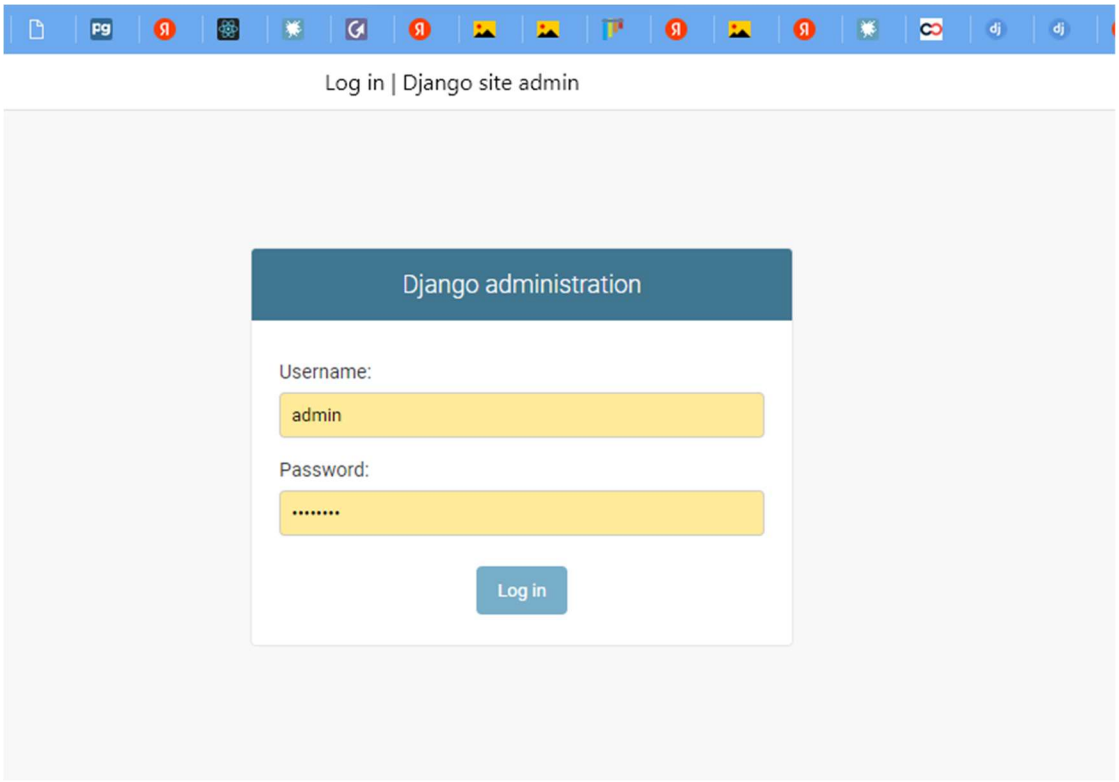


Рисунок Г.7 – Ввод логина и пароля

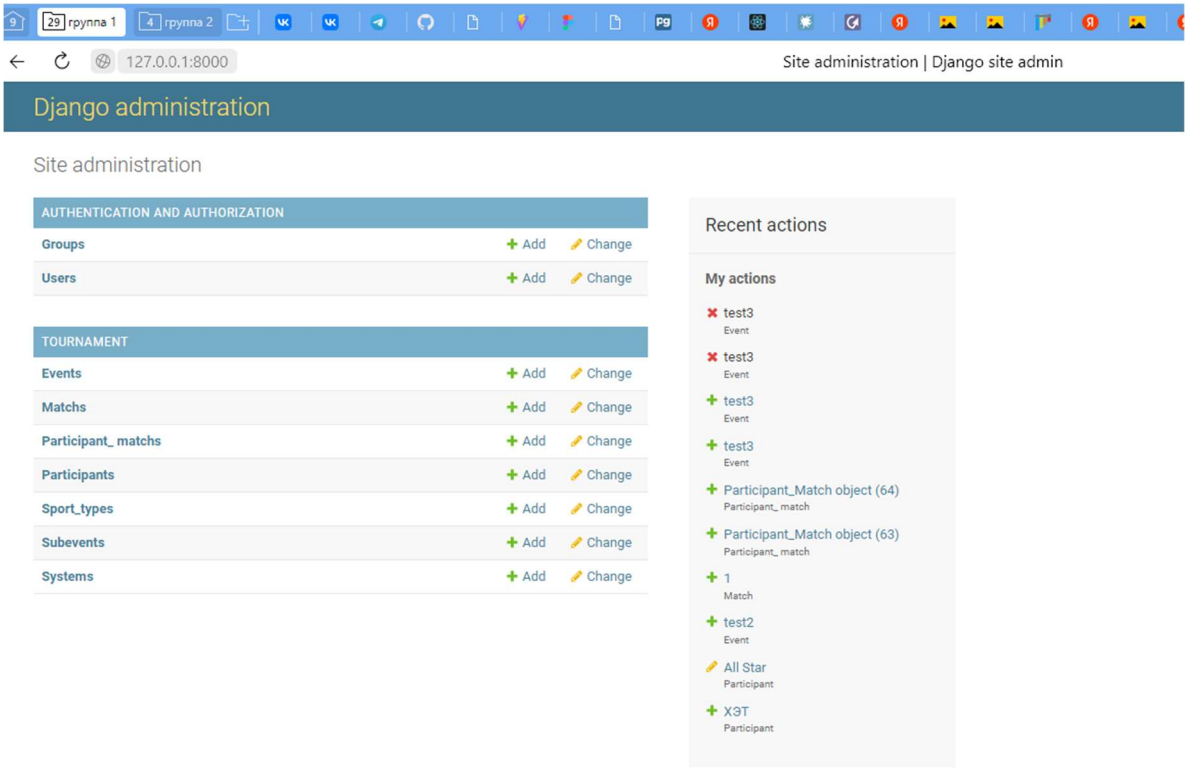


Рисунок Г.8 – Результат после входа

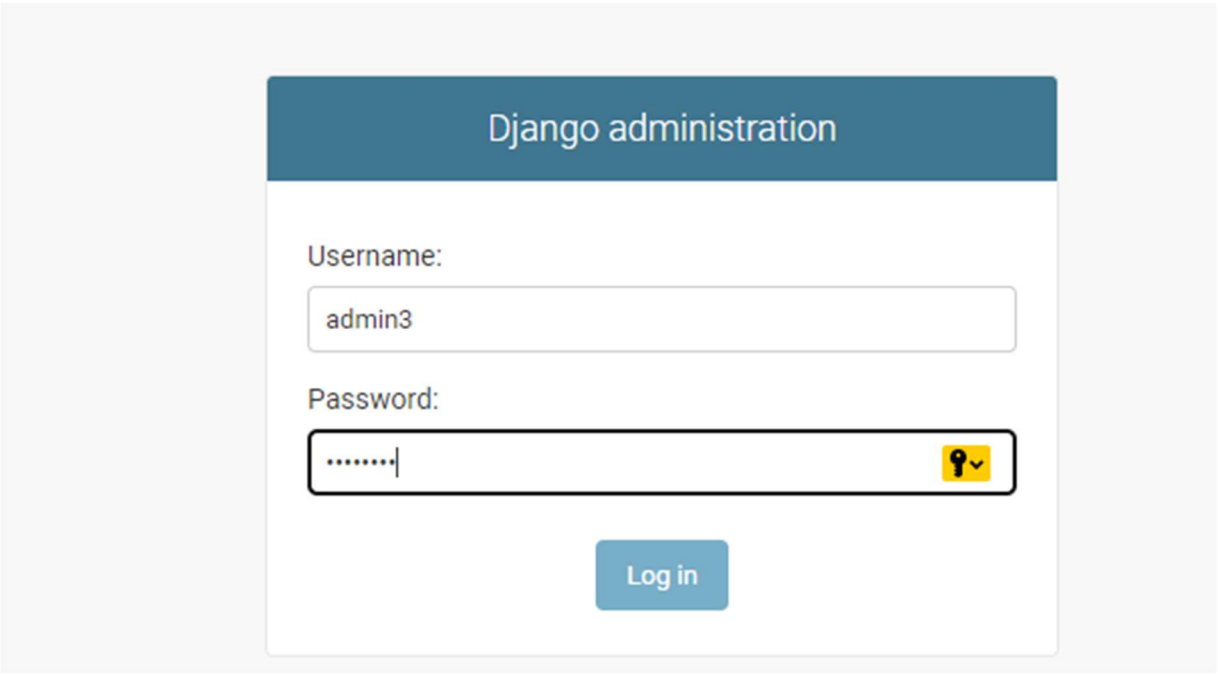


Рисунок Г.9 – Ввод неверных данных

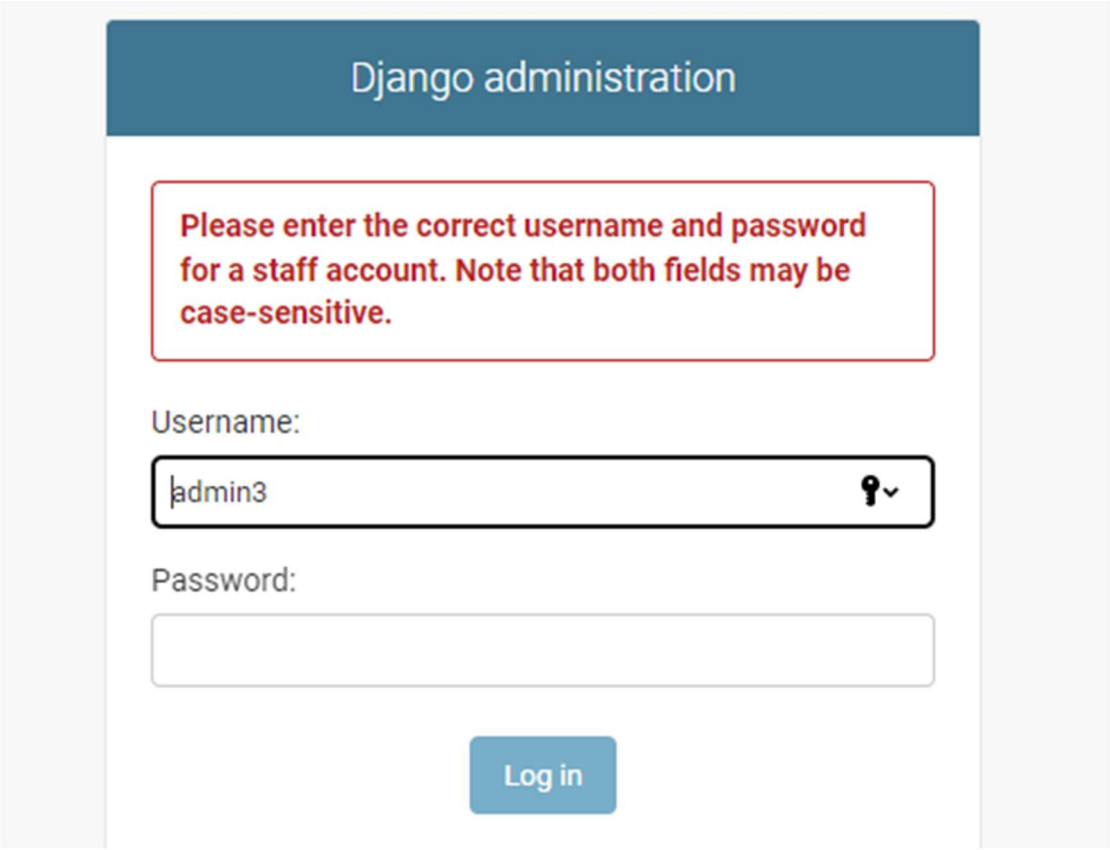


Рисунок Г.10 – Результат после нажатия кнопки «Log In»

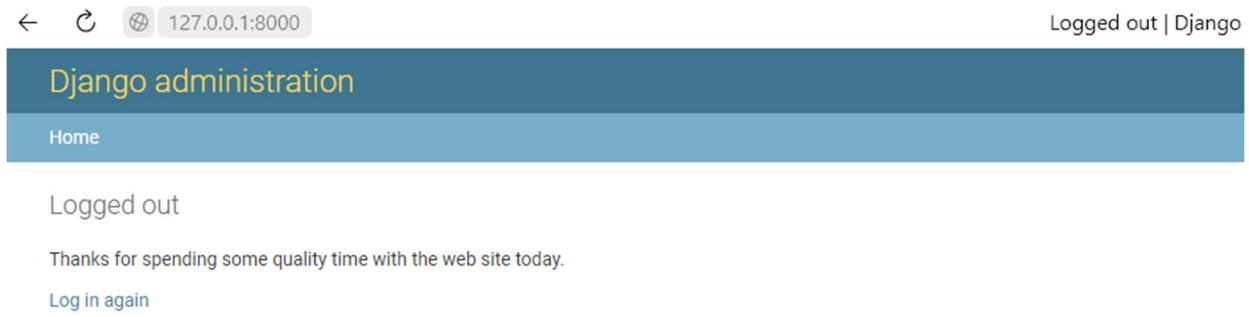


Рисунок Г.11 – Результат выхода из приложения для администратора

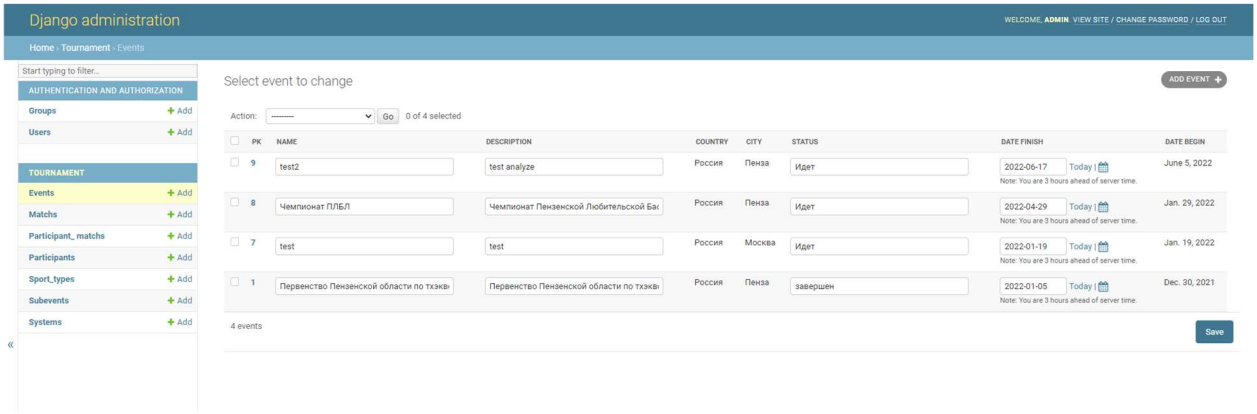


Рисунок Г.12 – Список событий до добавления

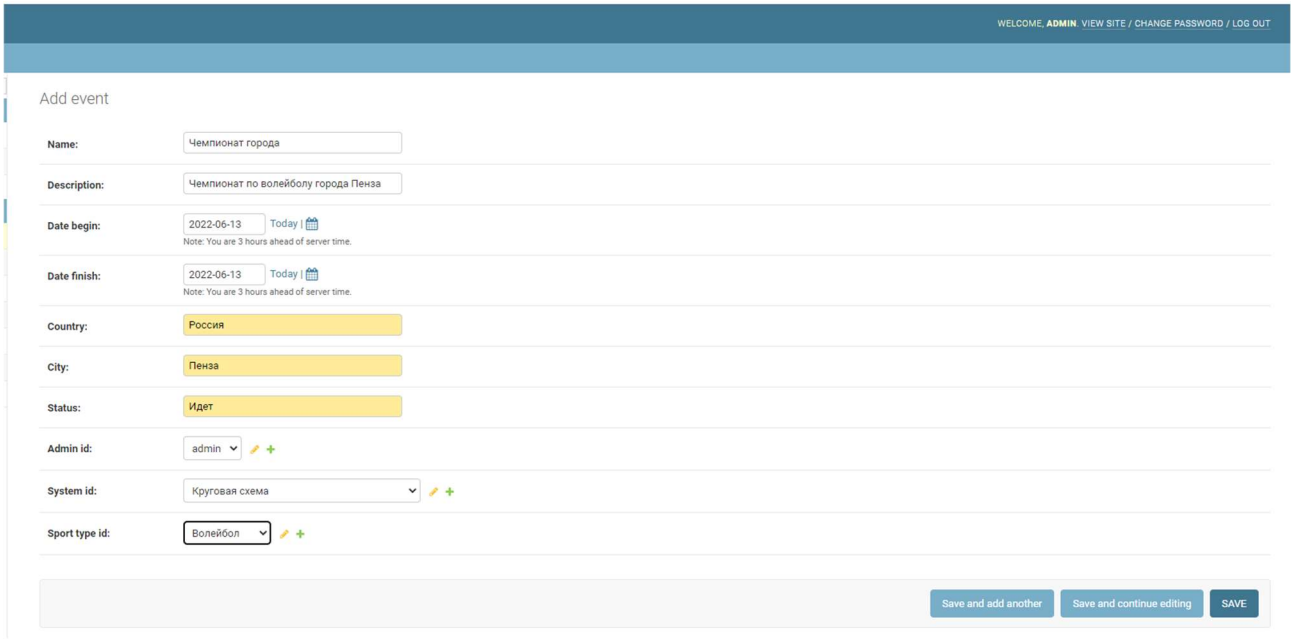


Рисунок Г.13 – Ввод данных

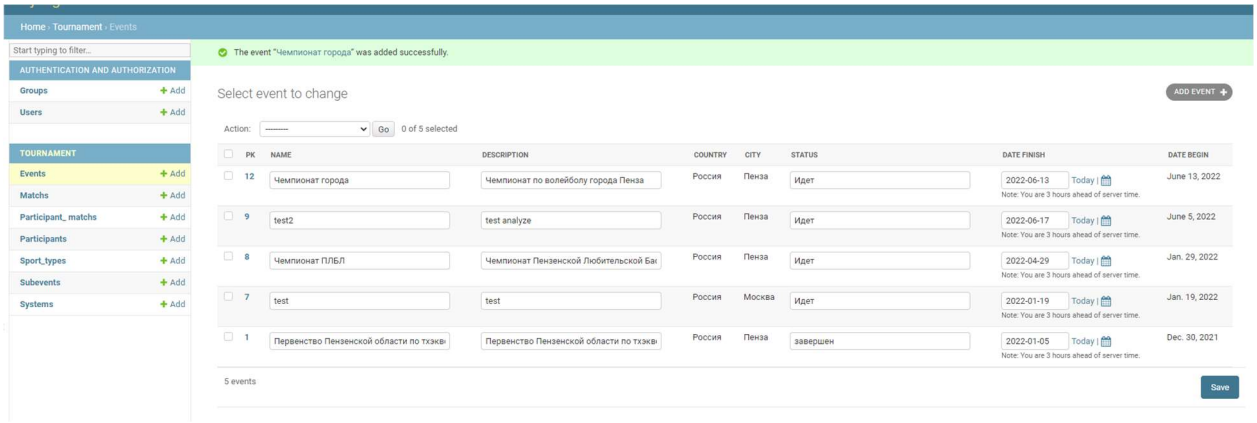


Рисунок Г.14 – Список событий после добавления

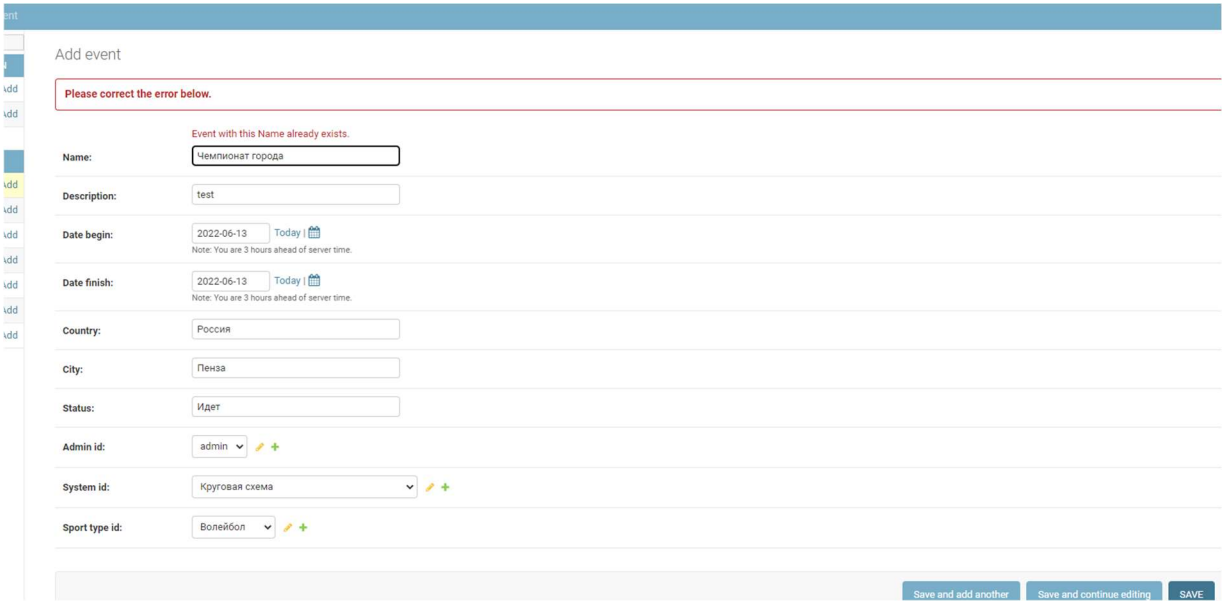


Рисунок Г.15 – Ошибка добавления повторного события

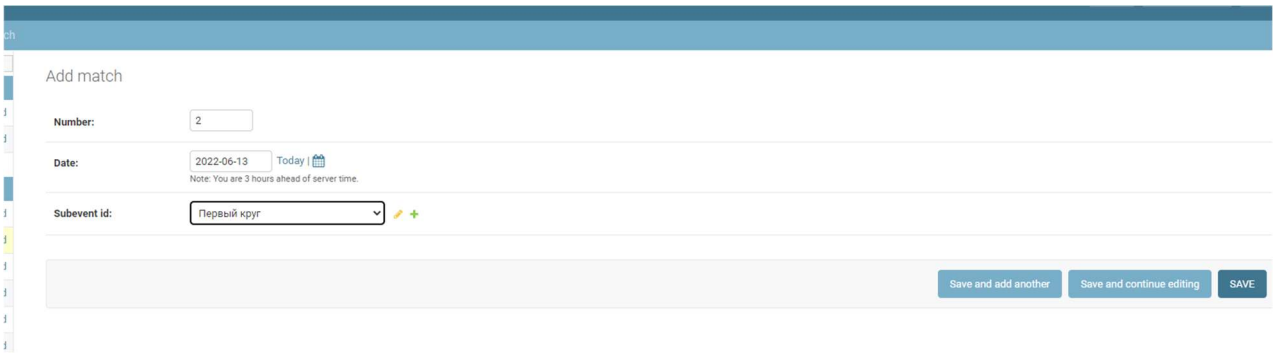


Рисунок Г.16 – Добавление матча

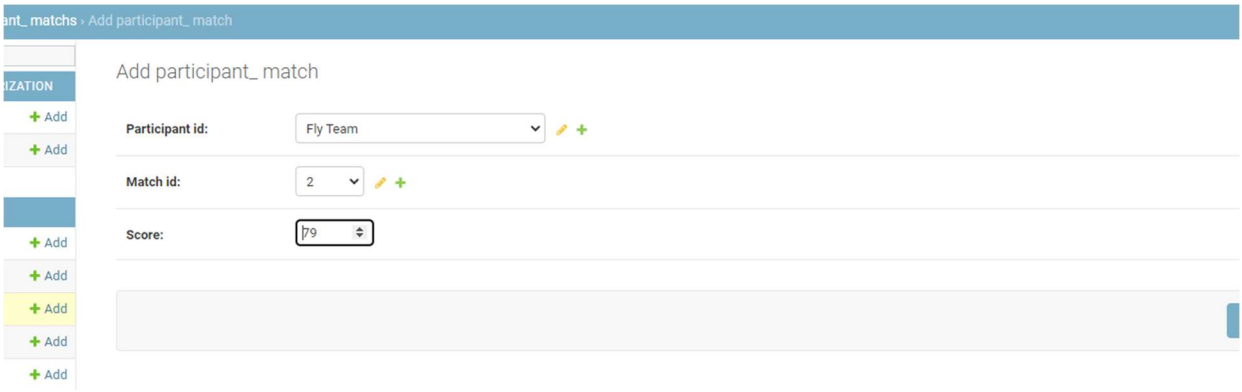


Рисунок Г.17 – Добавление результата первого участника

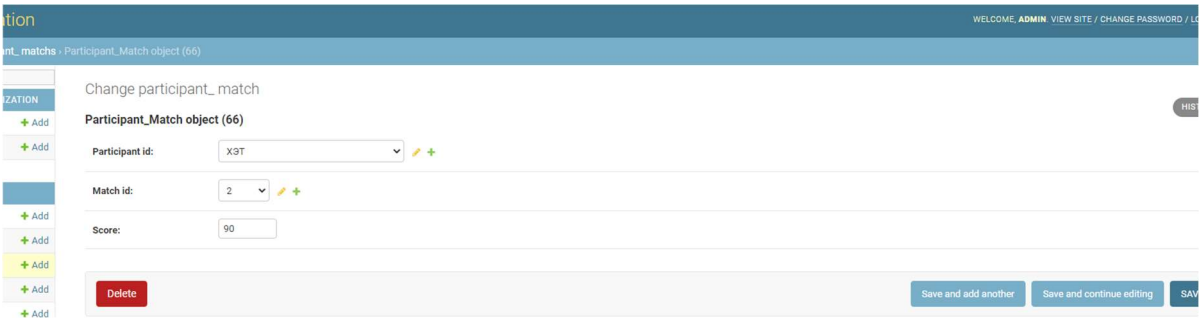


Рисунок Г.18 – Добавление результата второго участника

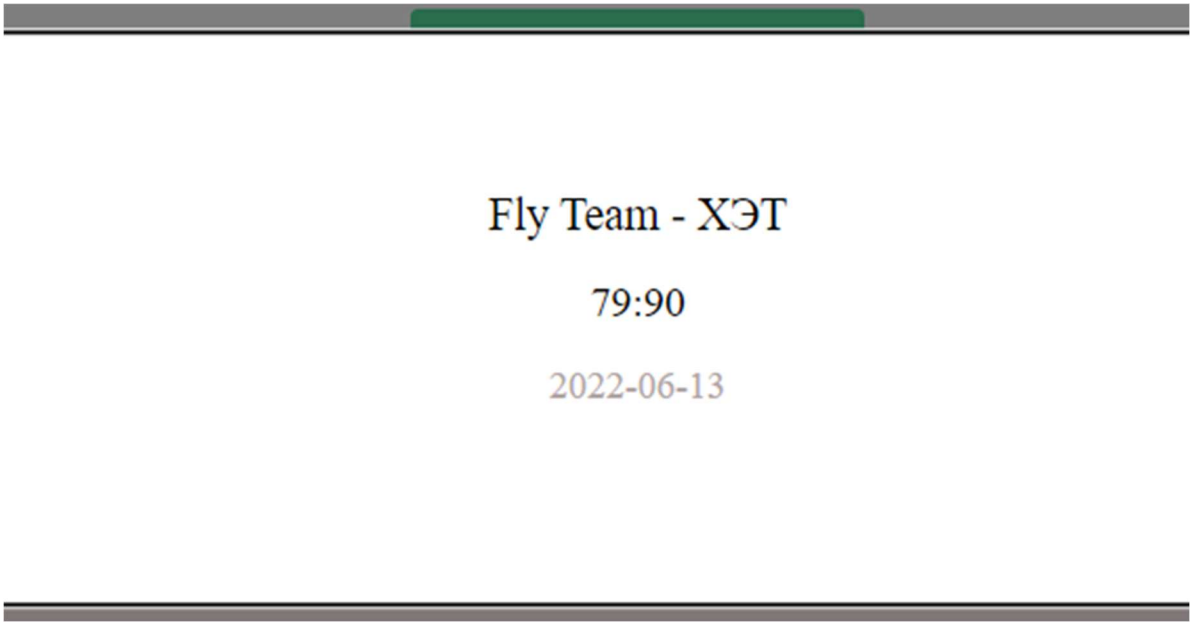


Рисунок Г.19 – Результат формирования результата матча

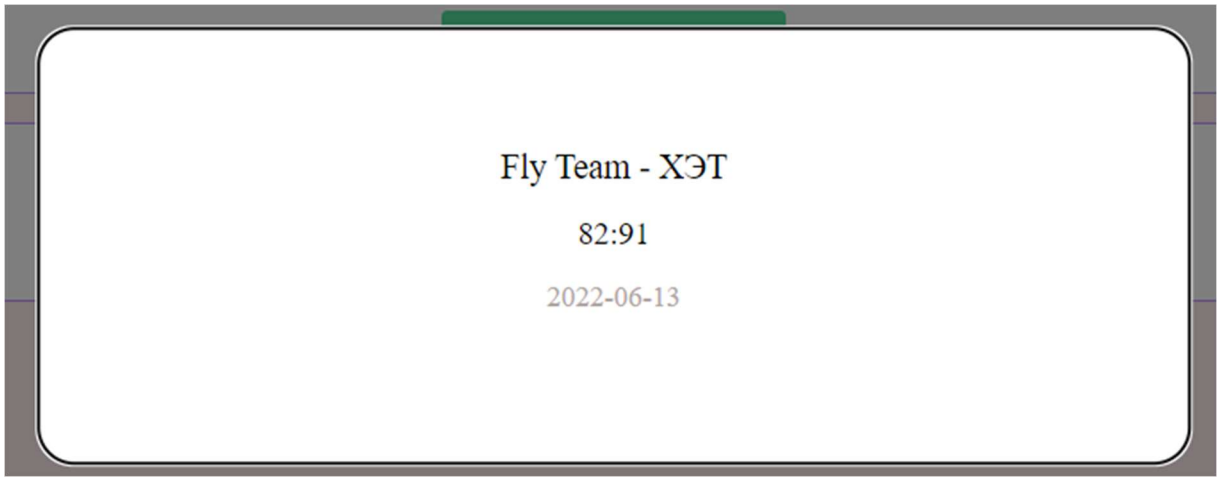


Рисунок Г.20 – Результат изменения счета матча

Приложение Д.
РИСУНКИ И ДИАГРАММЫ
(рекомендуемое)

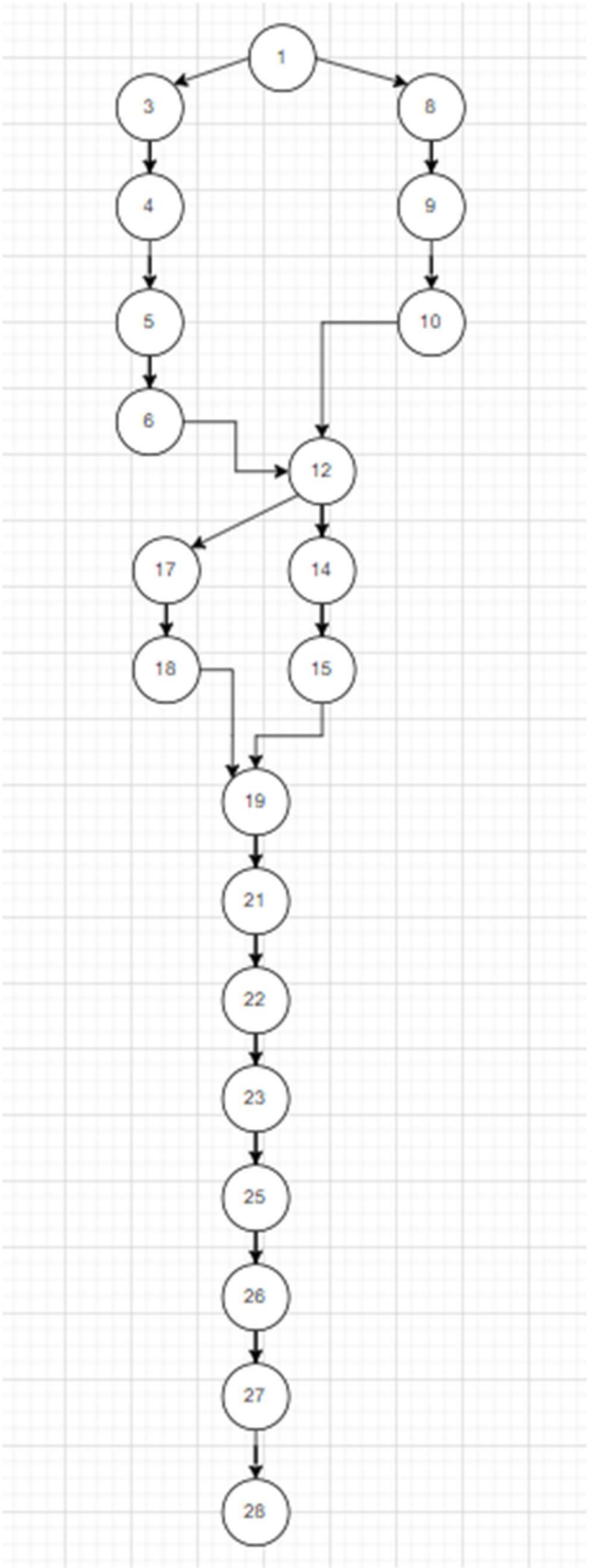


Рисунок Д.1 – Сетевой график

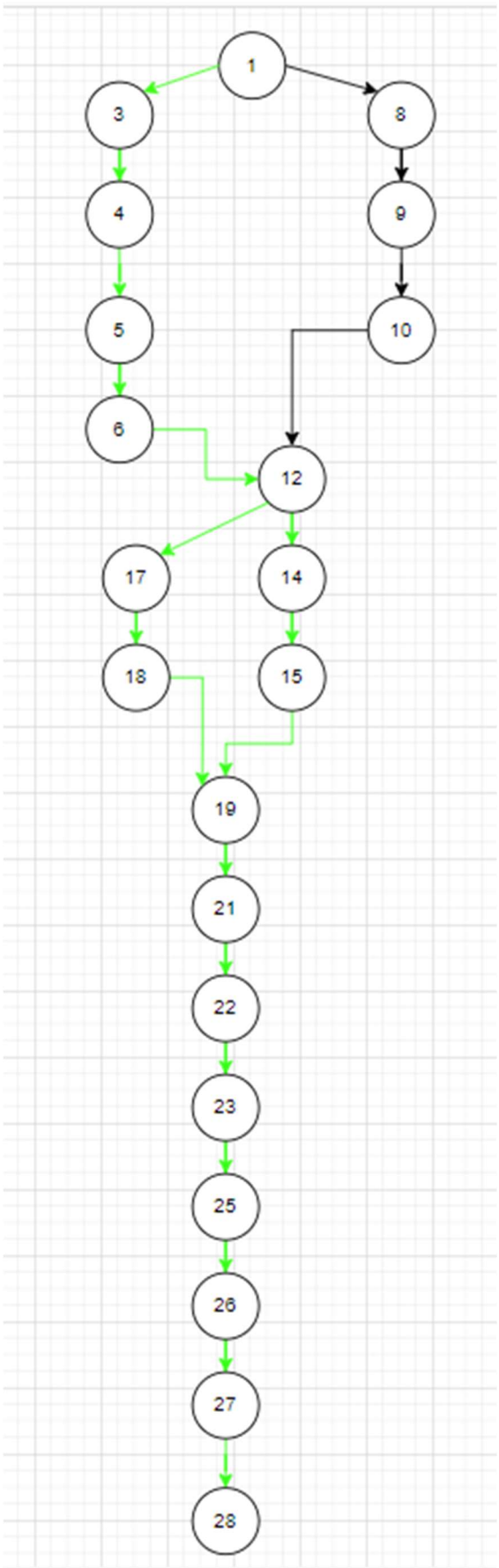


Рисунок Д.2 – Сетевой график с указанным критическим путем

	Название задачи	Длительность	Начало	Окончание	Предш
1	Начало	0 дней	Вт 01.03.22	Вт 01.03.22	
2	**Анализ требований**	**10 дней**	**Вт 01.03.22**	**Пн 14.03.22**	
3	Анализ аналогичных реш	6 дней	Вт 01.03.22	Вт 08.03.22	1
4	Анализ правил проведен	3 дней	Ср 09.03.22	Пт 11.03.22	3
5	Принятие окончательное	1 день	Пн 14.03.22	Пн 14.03.22	4
6	Анализ требований завe	0 дней	Пн 14.03.22	Пн 14.03.22	5
7	**Выбор средств реализации п**	**8 дней**	**Вт 01.03.22**	**Чт 10.03.22**	
8	Анализ возможных прог	6 дней	Вт 01.03.22	Вт 08.03.22	1
9	Выбор программного сре	2 дней	Ср 09.03.22	Чт 10.03.22	8
10	Выбор средств заверше	0 дней	Чт 10.03.22	Чт 10.03.22	9
11	**Реализация проекта**	**12 дней**	**Вт 15.03.22**	**Ср 30.03.22**	
12	Разработка структуры д	2 дней	Вт 15.03.22	Ср 16.03.22	6;10
13	**Разработка серверной**	**10 дней**	**Чт 17.03.22**	**Ср 30.03.22**	
14	Программирован	3 дней	Чт 17.03.22	Пн 21.03.22	12
15	Программирован	7 дней	Вт 22.03.22	Ср 30.03.22	14
16	**Разработка пользовате**	**10 дней**	**Чт 17.03.22**	**Ср 30.03.22**	
17	Разработка макет	4 дней	Чт 17.03.22	Вт 22.03.22	12
18	Верстка приложе	6 дней	Ср 23.03.22	Ср 30.03.22	17
19	Реализация проекта зав	0 дней	Ср 30.03.22	Ср 30.03.22	15;18
20	**Отладка**	**13 дней**	**Чт 31.03.22**	**Пн 18.04.22**	
21	Тестирование пользова	3 дней	Чт 31.03.22	Пн 04.04.22	19
22	Исправление ошибок	10 дней	Вт 05.04.22	Пн 18.04.22	21
23	Тестирование завершен	0 дней	Пн 18.04.22	Пн 18.04.22	22
24	**Внедрение программного ср**	**15 дней**	**Вт 19.04.22**	**Пн 09.05.22**	
25	Внедрение программног	10 дней	Вт 19.04.22	Пн 02.05.22	23
26	Поиск и исправления ош	5 дней	Вт 03.05.22	Пн 09.05.22	25
27	Внедрение завершено	0 дней	Пн 09.05.22	Пн 09.05.22	26
28	Конец проекта	0 дней	Пн 09.05.22	Пн 09.05.22	27

Рисунок Д.3 – Задачи проекта в среде Microsoft Project

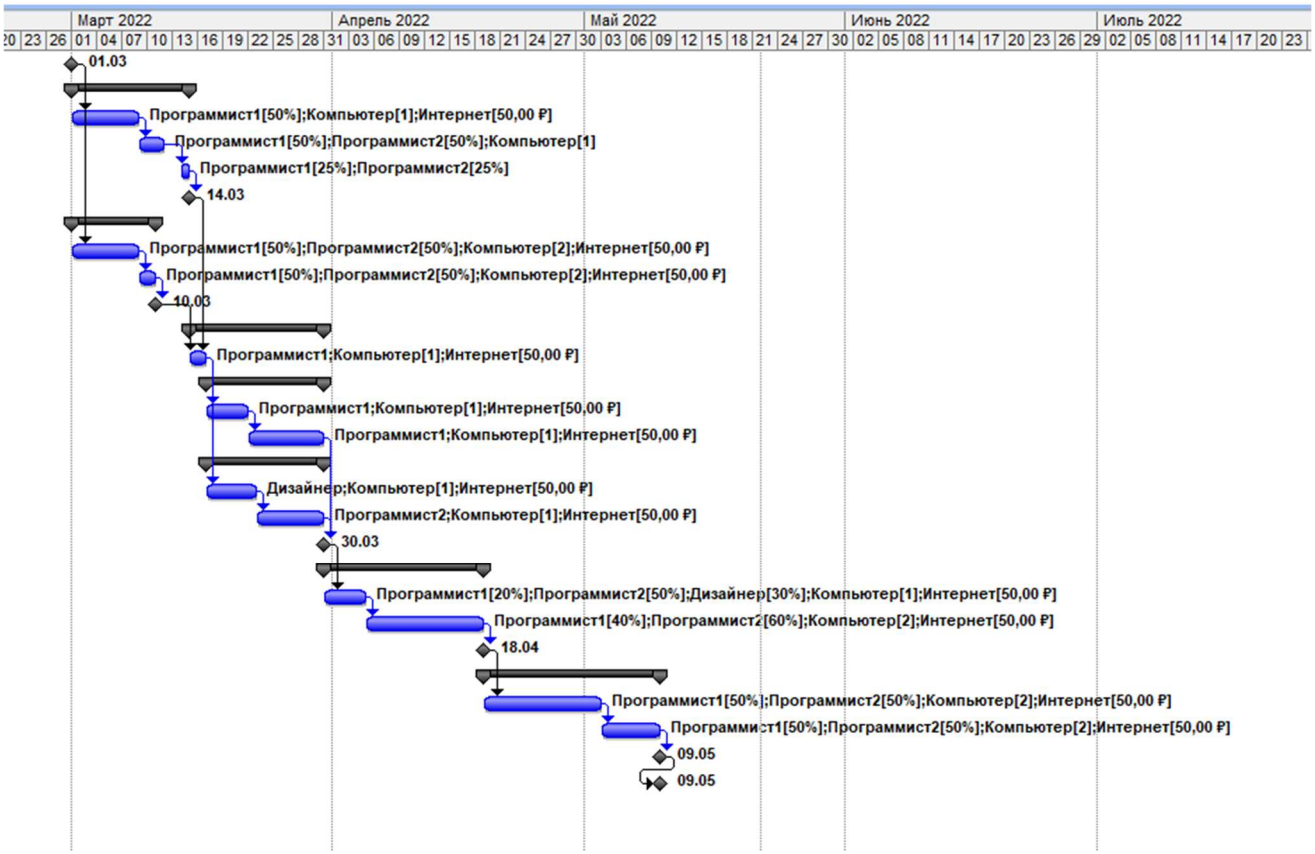


Рисунок Д.4 – Диаграмма Ганта