и простыеМинистерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

**РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ АВТОМАТИЗАЦИИ ОРГАНИЗАЦИИ И ПРОВЕДЕНИЯ СПОРТИВНЫХ СОРЕВНОВАНИЙ**

**ОТЧЁТ**

ПО РЕЗУЛЬТАТАМ

производственной практики: научно-исследовательская работа

Студент гр. 429-3

\_\_\_\_\_\_\_\_\_\_\_ Кутольвас В.Б.

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2022 г.

Руководитель практики

канд. техн. наук., доцент каф. АОИ

\_\_\_\_\_\_\_\_\_\_\_ Потахова И.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2022 г.

Томск 2022

# **Содержание**

Введение.......................................................................................................................3

1 Выбор стека технологий..........................................................................................4

1.1 Аналоги системы........................................................................................4

1.2 Бэкенд фреймворк.......................................................................................5

1.2.1 Критерии.........................................................................................5

1.2.2 Обзор...............................................................................................6

1.3 Фронтенд фреймворк................................................................................10

1.4 Сервис для развёртывания.......................................................................10

2 Реализация приложения.........................................................................................11

2.1 Функционал приложения.........................................................................11

2.2 UX/UI дизайн.............................................................................................11

2.3 Backend.......................................................................................................15

2.4 Frontend......................................................................................................17

3 Тестирование..........................................................................................................19

4 Развёртывание в сети интернет.............................................................................20

Заключение................................................................................................................21

Список использованных источников.......................................................................22

Приложение А...........................................................................................................23

**Введение**

В данной работе целью ставится создание программного комплекса для организации спортивных турниров. В процессе создания подразумевается исследование возможностей различных фреймворков и библиотек, позволяющих реализовать данный комплекс, сравнение их качественных и количественных характеристик с последующий выбором тех из них, чей функционал наиболее удовлетворяет поставленным требованиям, а также исследование возможностей выбранного стека технологий.

Реализуемый сервис должен предоставлять функционал просмотра и администрирования соревнований, а также автоматической генерации турнирной сетки. Для определения оптимального типа системы был проведён обзор предоставляющих схожий функционал сервисов, коими являются веб-ресурсы challonge.com и tlab.pro. В результате обзора было сделано заключение, что наилучшим вариантом является создание веб-ресурса, так как к нему возможен доступ с любого устройства и нет необходимости предварительной установки программы.

После изучения стека технологий следует перейти к выполнению описанных ниже задач:

* изучение предметной области, определение моделей данных и отношений между ними;
* проектирование дизайна и интерфейса ресурса на основе предметной области;
* проектирование и реализация серверной части приложения;
* вёрстка страниц;
* тестирование;
* развёртывание системы в сети интернет.

**1 Выбор стека технологий**

**1.1 Аналоги системы**

Первым делом был проведён обзор существующих аналогов создаваемой системы. На данном этапе были выделены следующие ресурсы, предоставляющие возможность создать турнирную сетку для самых разнообразных видов спорта, реальных или виртуальных: challonge.com и tlab.pro. Все перечисленные программные системы являются веб-приложениями. Такой формат является удобнее нежели мобильное приложение в случае несистемного участия, организации или просто наблюдении пользователем за соревнованиями за счёт отсутствия необходимости устанавливать приложение на устройство и возможности получить доступ к ресурсу с любого устройства, на котором есть браузер.

**challonge.com.** В сгенерированной турнирной сетке возможно проведение матча за третье место, но нет возможности внесения результатов матча, а также дополнительной информации по матчу.

**tlab.pro.** Минималистичный интуитивно понятный интерфейс, две системы проведения турниров.

В вышеприведённых системах имена участников и названия команд вносятся организатором. В разрабатываемой же системе заявку на участие будут оставлять сами участники, после чего она автоматически распределяются на матчи.

**1.2 Backend фреймворк**

**1.2.1 Критерии**

После того как был определён формат приложения, необходимо выбрать фреймворк для бэкенд-разработки. Для определения подходящего фреймворка был проведён анализ по следующим критериям:

* Безопасность;
* Архитектура;
* Администрирование;
* Работа с базами данных;
* Документированность.

**Безопасность.** Интернет-пространство полнится мошенниками, желающими использовать любую уязвимость в приложении для достижения своих корыстных или же развлекательных целях. Это может вылиться в последствия совершенно разного уровня - начиная с того, аккаунт пользователя в соцсети приобретёт новую нежелательную подписку или рассылку, и заканчивая тем, что конфиденциальные данные пользователя попадут в руки злоумышленников, приведя к тому что с его банковского счёта могут быть списаны весомые денежные средства. Во избежание подобных ситуаций, при выборе фреймворка важным критерием будет наличие в нём встроенных защитных механизмов, таких как защита от подделки межсайтовых запросов, защита от кликджекинга, от SQL-инъекций и других атак.

**Архитектура.** Удобство проектирования и разработки системы зависит от возможности разграничения в структуре проекта основополагающих компонента системы. Для этого в используемом веб-фреймворке желательно применение паттерна проектирования Model-View-Controller.

**Администрирование.** Любой интернет ресурс это в первую очередь контент на нём размещаемый. Возможность управления этим контентом и данными пользователей - необходимая его часть, и наличие администраторской панели является важным фактором при выборе фреймворка.

**Работа с базами данных.** Работа с базами данными через написание SQL-запросов является часто используемым, но с моей точки зрения не самым удобным способом взаимодействия с данными. Для упрощения этого процесса существует такая технология как ORM, представляющая дополнительный уровень абстракции для работы с информацией за счёт перехода от терминов реляционных баз данных к терминам классов и их экземпляров. Наличие данной технологии в инструментарии делает разработку проще и быстрее.

**Документированность.** Фреймворк, изучение которого предстоит в рамках данной работы, должен иметь исчерпывающую документация для возможности изучения его возможностей.

**1.2.2 Обзор**

Для выбора фреймворка также немаловажным фактором является язык программирования, на котором будет производиться разработка. Язык python имеет простой синтаксис, обширную документацию и большое сообщество, а также является единственным языком изученным мной в достаточной для разработки мере. Исходя из всех вышеперечисленных причин, далее будут рассматриваться фреймворки для языка python из статьи[1].

**Web2Py**. Данный фреймворк разработан Массимо Де Пьеро в 2007 году и является масштабируемой платформой с открытым исходным кодом[2]. Его отличительными особенностями являются наличие собственной IDE, имеющей инструменты для развёртывания и отладки, а также отсутствие поддержки python3. Рассмотрим критерии оценивания:

* Для защиты от межсайтового скриптинга Web2Py использует экранирование переменных предоставляемых в шаблон. SQL инъекции предотвращаются за счёт использования дополнительного слоя абстракции для доступа к данным, а CSRF атаки предупреждаются за счёт хранения всех данных о сессии на стороне сервера, в браузере же храниться только её идентификатор.
* Web2Py следует архитектурному шаблону MVC.
* Web2Py имеет администраторскую панель, дающую возможность создания и редактирования приложений, а также предоставляющую доступ к модулям каждого приложения, но имеющую не слишком современный дизайн.

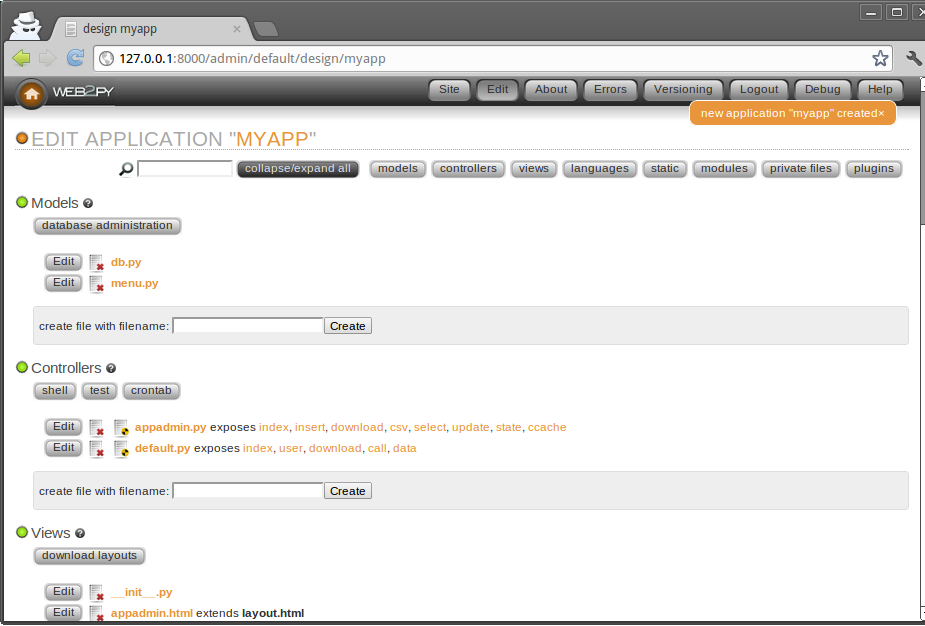


Рисунок 1.1 - Панель администратора Web2Py

* Работа с базами данных в данном фреймворке реализована созданием уровня абстракции, представленным классом соединения с базой данных, реализующим поля таблиц и методы различных обращений к БД.
* Данный фреймворк имеет хорошо структурированную документацию на русском языке.

**Django.** Популярный python веб-фреймворк, работающий что называется, из коробки, при этом имеет большой перечень библиотек, предоставляющих разработчику самый разнообразный функционал при низком пороге вхождения и большом сообществе[3].

* Защитные механизмы фреймворка представляют защиту от CSRF за счёт проверки токена, ORM предоставляет защиту от sql-инъекций, также Django содержит защиту от кликджекинга в виде промежуточного ПО (middleware), которое поддерживается браузерами и может запретить отображение страницы внутри <iframe>.
* Django разделяет логику, отображение и данные путём применения MVC-MTV паттернов.
* Администрирование происходит через встроенную настраиваемую панель django-admin (рисунок 1.2)

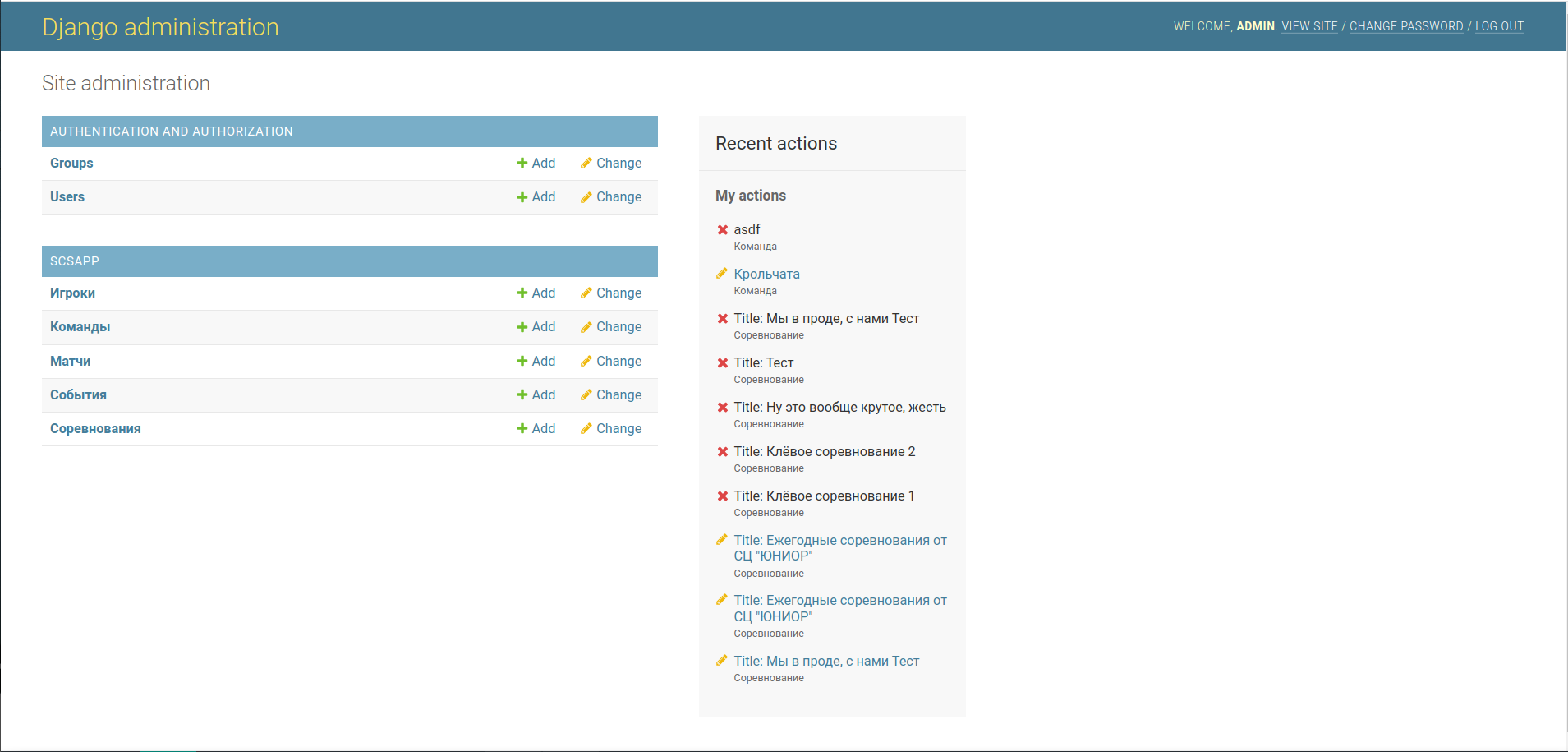


Рисунок 1.2 - панель Django-admin

* Django имеет встроенную систему ORM, представляющую данные приложения в терминологии классов.
* Данный фреймворк имеет широкую документацию на английском языке.

**Flask.** Ещё один популярный python бэкенд-фреймворк. Его отличие заключается в его модульности, он является лишь базовым каркасом для создания веб-приложений, что делает его более гибким инструментом для разработки и понижает уровень вхождения[4].

* Для защиты от CSRF существует подключаемое отдельно расширение flask-WTF, шаблонизатор Jinja2 автоматически экранирует все значения, что представляет защиту от межсайтового скриптинга, а также браузеру даётся указание преобразовать все HTTP-запросы в HTTPS, предотвращая атаки типа "злоумышленник в середине" (MITM).
* во Flask не заложено никакого строгого шаблона, как и строгой структуры проекта, что предоставляет разработчику свободу выбора: самому реализовать архитектурный шаблон MVC, MTV или же не следовать шаблону в принципе.
* для администрирования системы на Flask существует подключаемый модуль flask-admin, предоставляющий функционал достаточный для выполнения основных задач администратора (рисунок 1.3).

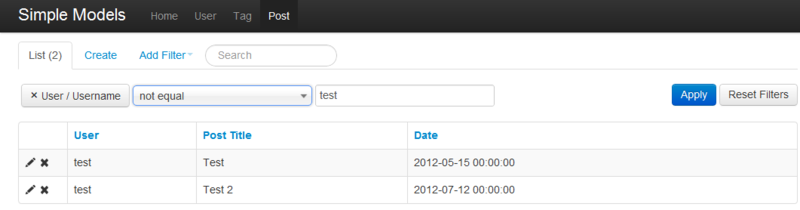


Рисунок 1.3 - flask-admin панель

* для работы с базами данных в Flask чаще всего используется расширение flask-sqlalchemy, представляющие возможности ORM.
* при работе с данным микро-фреймворком возникают сложности при изучении документации, так как для каждого подключаемого модуля документация находится на его ресурсе, в результате чего она становится очень разрозненной.

В результате анализа возможностей каждого из представленных фреймворков, мой выбор пал на Django из-за целостности, большого сообщества, а также и из-за требований знания этого фреймворка большого числа работодателей и следующей из этого перспективы его изучения.

**1.3 Frontend фреймворк**

Для реализации сайта с современным дизайном был выбран уже знакомы мне фреймворк, имеющий разграничивающую страницу сетку и представляющий возможность кроссбраузерной разработки Вootstrap. Он имеет большое количество встроенных классов стилей и разнообразных JS функций, благодаря которым возможно быстрое создание адаптивных страниц[5]. Помимо этого boostrap возможно легко кастомизировать, благодаря препроцессору SASS, облегчающим написание стилей для вёрстки[8].

**1.4 Сервис для развёртывания**

Для развёртывания системы в сети интернет был выбран облачный сервис Heroku, предоставляющий возможность деплоя приложений написанных на разных языках. Процесс развёртывания на Heroku относительно прост и быстр, а сам сервис имеет бесплатные тарифы подходящие для тестирования и первоначального использования приложений.

**2 Реализация приложения**

**2.1 Функционал приложения**

Основным функционалом приложения является анонсирование соревнований, подача заявок на участие, автоматическое составление турнирной сетки и отображение информации о соревнованиях, предстоящих, текущих или прошедших.

Помимо основного функционала в последующих версиях возможно добавление следующих функций:

* рассылка e-mail писем о создании нового соревнования, о предстоящем матче, о результатах завершённого матча;
* Ведение матчей при судейском доступе, что подразумевает ввод в систему данных о событиях матча, во время их проведения;
* онлайн трансляция событий матча;
* прочий функционал.

**2.2 UI/UX дизайн**

Первым делом на основе функциональных требований и предметной области был разработан дизайн веб-страниц. Так как система предназначена для проведения турниров преимущественно по волейболу, то и цвета для оформления были подобраны в соответствии с цветовой гаммой волейбольного мяча, к которой был добавлен чёрный цвет для прошедших соревнований (рисунок 2.1).



Рисунок 2.1 - Цветовая гамма

Шапка сайта выполнена в жёлтом цвете, и содержит ссылку перехода на главную страницу, кнопку входа и регистрации, либо же выхода, а также кнопку создание нового соревнования при судейских правах доступа.

Для удовлетворения требований основного функционала необходимы были страницы со списком соревнований, страница конкретного соревнования, страница создания соревнования, страницы входа и регистрации:

Страница со списком всех соревнований. Для упрощения структуры веб-ресурса было решено поместить анонсированные, текущие и прошедшие соревнования на одну страницу. Так как основной задачей информационной системы является анонсирование соревнований судьями и запись на них командами или их представителями, то основным элементом страницы является список анонсированных соревнований, занимающий всю ширину контент-блока страницы. Ниже идут текущие и прошедшие соревнования, занимающие по ширине по половине ширины страницы (рисунок 2.1);

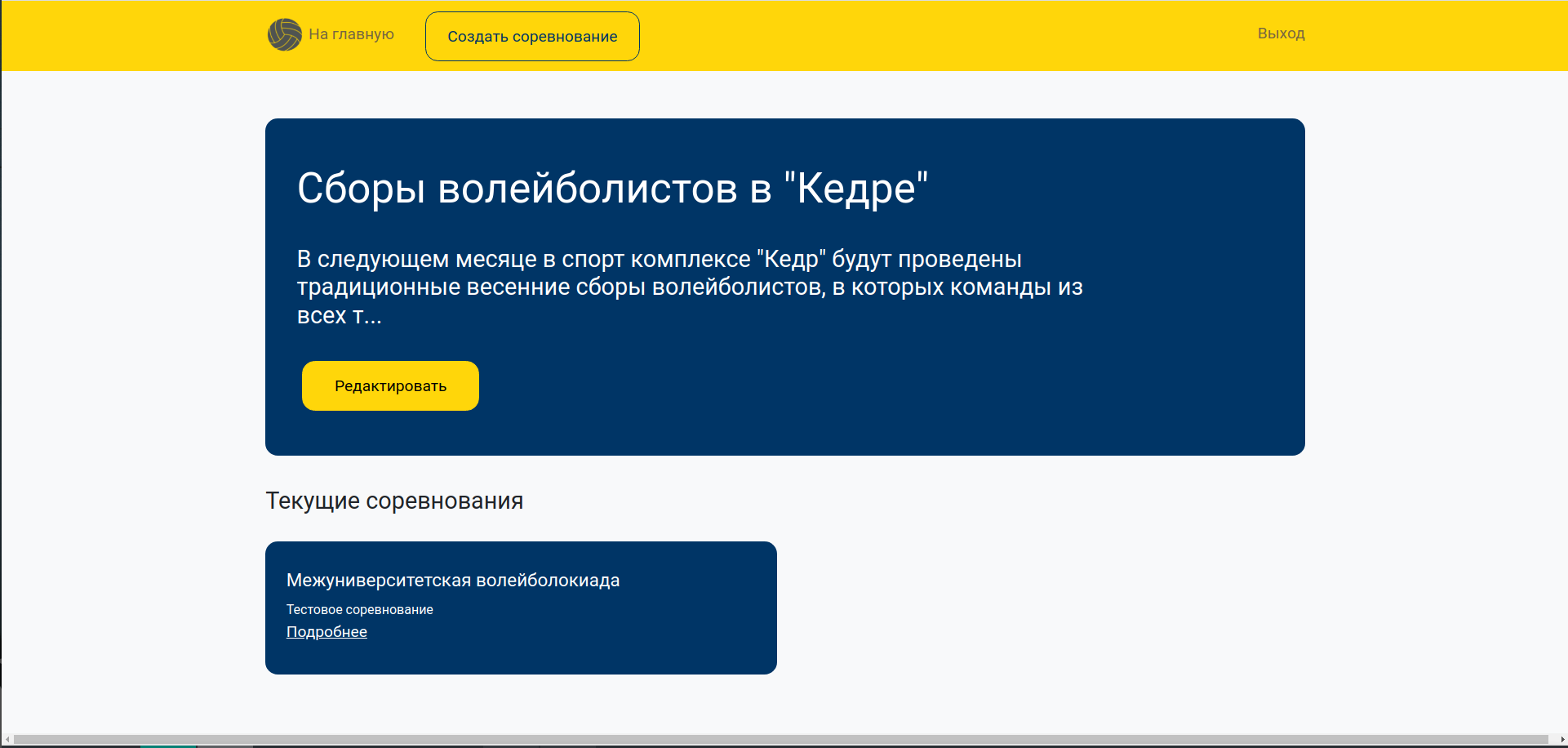


Рисунок 2.1 - Главная страница

Страница со списком прошедших соревнований. Когда прошедших турниров станет слишком много, предусмотрена веб-страница, реализующая их постраничный вывод. В её нижней части находится пейджинатор, позволяющий перемещаться по истории соревнований;

Страница турнира. В зависимости от статуса соревнования и прав доступа пользователя на странице будут присутствовать следующие виджеты:

* Основная информация. На странице находится постоянно, при судейском доступе предоставляет возможность редактирования данных соревнования. В данном виджете находится такая информация как заголовок, описание, наименование организации, проводящей соревнования, а также в зависимости от статуса соревнования время подачи заявок (для анонсированного), время следующего матча (для текущих) или время завершения соревнования (для завершённых). В случае анонсированного соревнования также предоставляет возможность редактирования количества команд-участниц.
* Виджет подачи заявки на участие в соревновании. Содержит форму для регистрации команды (название и описание), а также набор форм для ввода информации о каждом игроке команды. Данный виджет виден только когда соревнование анонсировано, а пользователь не имеет судейских прав доступа и ещё не подавал заявки на участие в данном соревновании (рисунок 2.2).

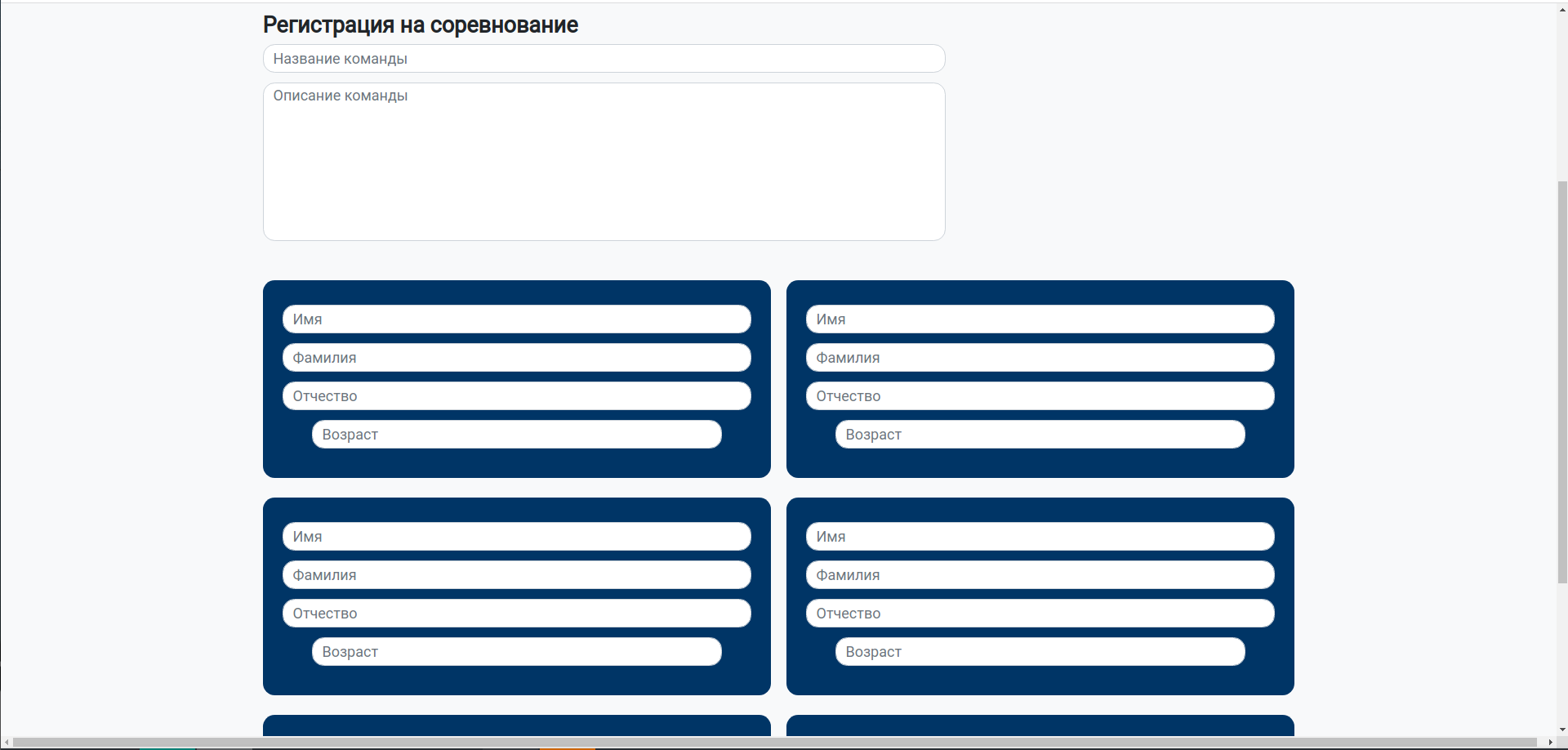


Рисунок 2.2 - Форма подачи заявок на участие

* Виджет со списком матчей данного соревнования. Включает в себя матчи, отсортированные по времени. Каждый матч имеет информация о командах в данном матче участвующих, о времени и месте проведения, а также игровой счёт. Каждый матч имеет возможность редактирования данных при судейских правах доступа, а статус матча помечается фоновым цветом - синим или чёрным (для предстоящих и прошедших матчей соответственно) (рисунок 2.3).

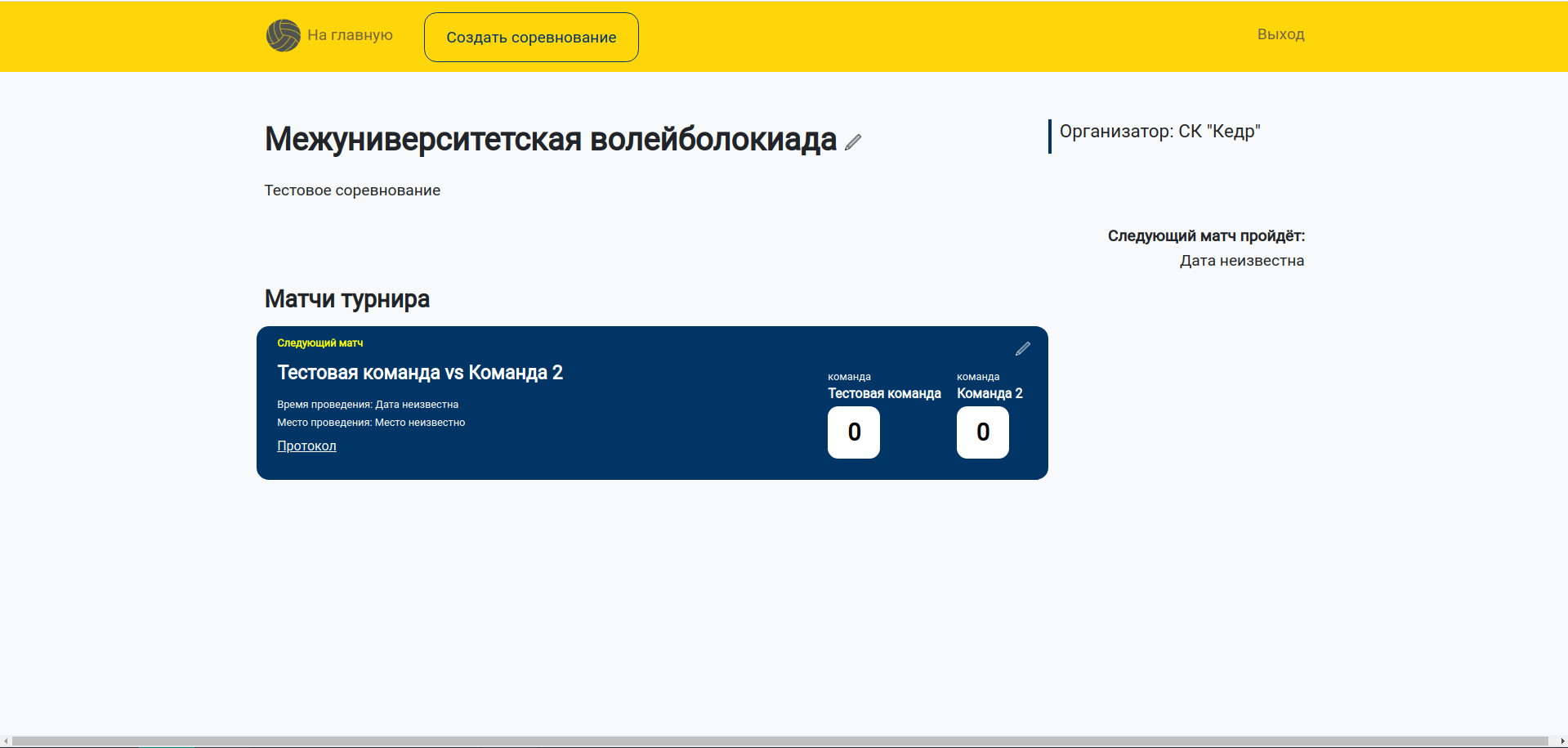


Рисунок 2.3 - страница текущего соревнования при судейском доступе

Страница создания соревнований. Доступна только при судейских правах доступа, имеет форму для создания соревнований, включающую следующие поля: заголовок, описание, время подачи заявок, организатор и минимально необходимое количество команд (рисунок 2.4).

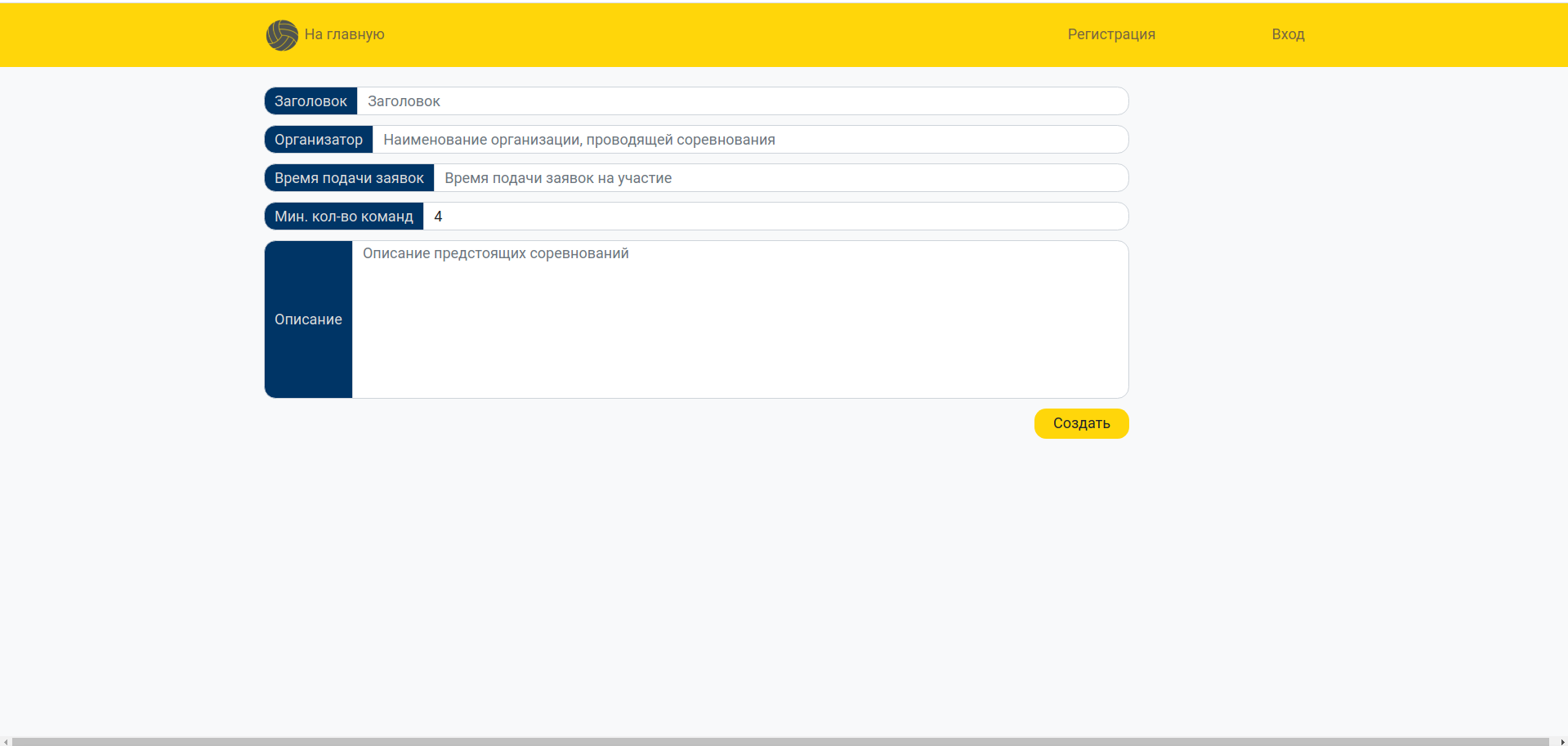


Рисунок 2.4 - Страница создания соревнования

**2.3 Backend**

Бэкенд системы был написан на фреймворке Django, реализующем архитектурное решение MTV. Данная архитектура состоит из следующих слоёв:

* Model. Данный слой подразумевает под собой объектно-реляционное отображение данных, заключающееся в преобразовании реляционных баз данных в виртуальную базу объектов, обеспечивая возможность работы с информацией в терминах классов.
* Template. Слой работы с отображением данных. Включает в себя html-шаблоны и другие документы, отвечающие за позиционирование и внешний вид элементов на веб-страницах. Определяет как отображать информацию предоставленную слоем View.
* View. Слой представления отвечает за бизнес-логику приложения и является связующим звеном между данными и шаблонами. Определяет какую информацию предоставить, как обработать, и в какой шаблон поместить.

Первым делом для реализации приложения необходимо прописать модели таких сущностей как волейбольная команда, игрок, матч, соревнование и игровое событие. Плюсом ORM является то, что за данными предоставленными в виде класса можно закрепить наиболее часто производимые над данными действия. Этим мы и воспользуемся, создав в модели соревнование методы makeStanding() и updateStanding(), первый из которых в последующем будет генерировать в рандомном порядке объекты матчей с внешними ключами к другим матчам из этого же соревнования, тем самым создавая таблицу вида бинарного дерева. Второй метод запускается после завершения одного из матчей, обновляя турнирную таблицу в соответствии с победившей в прошедшем матче командой.

Далее необходимо реализовать слой отображения, прописав представления. В Django представления срабатывают при переходе пользователем по url, прописанному в карте в соответствующих файлах. Таким образом, необходимо создать представление для каждой страницы, разработанной на стадии проектирования UI/UX дизайна.

Так как все представления работают по одной схеме, далее будут описаны лишь представление, отвечающее за вывод прошедших соревнований и представление, отвечающее за страницу конкретного соревнования.

Для постраничного вывода прошедших соревнований через менеджер из модели соревнования извлекаются все прошедшие соревнования, после чего при слишком большой длине у них укорачиваются описания. После этого проверяется длина получившегося querySet, и если она больше установленной величины, список делится на страницы путём создания пейджинатора, после чего представление возвращает метод render(), передающий в шаблон объект страницы и пейджинатор.

Для вывода страницы конкретного соревнования из модели происходит извлечение информации указанного в url соревнования, а также создаются формы, соответствующие режиму доступа и статусу соревнования, после чего это всё передаётся в методе render() в шаблон.

**2.4 Frontend**

Слой Template, отвечающий в Django за отображение данных, представляется html-шаблонами и сопутствующими таблицами стилей и имеет механизм django-тегов, облегчающих передачу данных из слоя View, а также предоставляет возможность наследования шаблонов. Для реализации фронтенда был использован фреймворк Bootstrap, представляющий готовые классы стилей и js-функции, ответственные за анимацию. В данном фреймворке присутствует система строк и колонок, позволяющая легко верстать адаптивные страницы.

При начале вёрстки создадим шаблон base.html, включающий в себя тег <head>, в котором загружаются статичные файлы: css и js-модули, шрифты. Также в данном шаблоне в теге <body> находится разметка шапки сайта, за которой идёт контент-контейнер, так как на всех страницах они идентичны. От этого шаблона будет происходить наследование всех остальных шаблонов.

Далее перечислены наиболее часто используемые в проекте css-классы, входящие в Boostrap:

* row. Создает контейнер, всё содержимое которого выстраивается в одну строку. Строка делится на 12 колонок, и если содержимое строки превышает этот размер, происходит перенос на новую строку.
* col. Создаёт контейнер внутри контейнера row, даёт возможность регулирования относительной ширины элемента за счёт постфиксов контрольных точек ширины экрана устройства, тем самым поддерживая адаптивность страниц.
* collapse. Создаёт выпадающий элемент, будь то список или форма. Снабжен встроенной анимацией.
* justify-content-(end/start/center/between). Отвечает за позиционирование контента внутри элемента.

Кроме того, благодаря препроцессору SASS существует возможность кастомизации Boostrap и создания собственных классов на языке препроцессора, что упрощает написание таблиц стилей. Из собственных классов выделим competition, в котором задаются цвета для текста, фона, а также всевозможные отступы для элементов соревнований на главной странице.

**3 Тестирование**

Тестирование программной системы производилось на протяжении всего процесса разработки.

На этапе проектирования дизайна был привлечён специалист в области UI/UX дизайна, озвучивший свои замечания насчёт первоначального макета после чего в него были внесены соответствующие коррективы.

На этапе кодинга серверной части после реализации каждой функции производился запуск локального сервера и проверка данной функции, после чего в случае несоответствия результатов образцовым происходило выявление багов и их устранение.

В момент развёртывания в сети интернет выявлялись ошибки в настройках сервера, такие как неверно прописанный путь к wsgi-файлу, отсутствие некоторых зависимостей и другое.

Результаты работы приложения будут прикреплены в приложении А.

**4 Развёртывание в сети интернет**

Развёртывание в сети интернет происходило на сервисе Heroku, являющейся облачной платформой предоставляющей память и вычислительные мощности под нужды пользовательских систем. Данные сервис работает на операционных системах Ubuntu и Debian, что способствует облегчению развёртывания, так как разработка также велась под операционной системой семейства Linux[6].

При развёртывании был использован HTTP-сервер gunicorn. Данный сервер был выбран из-за малых требований к ресурсам, довольно большой скорости работы и совместимостью с Django[7].

**Заключение**

В ходе проведённых во время практической работы исследований был проведён обзор существующих фреймворков для бэкенд и фронтенд разработки, из которых были вычленены Django и Bootstrap с последующим углублённым изучением их возможностей, после чего изученные возможности данного инструментария были применены на практике.

Получившийся программный продукт обладает всем необходимым минимальным функционалом, и вместе с тем имеет заданное направление дальнейшего его наращивания и изменения, вследствие чего возможно получение информационной системы полностью удовлетворяющей требованиям заказчика.

**Список использованных источников**

1. Топ-7 python фреймворков для веб-разработки [Электронный ресурс] - Режим доступа: <https://www.monocubed.com/blog/top-python-frameworks/>
2. Документация Web2Py [Электронный ресурс] - Режим доступа: <http://www.web2py.com/>
3. Документация Django [Электронный ресурс] - Режим доступа: <https://docs.djangoproject.com/en/4.0/>
4. Документация Flask [Электронный ресурс] - Режим доступа: <https://flask.palletsprojects.com/en/2.0.x/>
5. Документация Bootstrap [Электронный ресурс] - Режим доступа: <https://bootstrap5.ru/docs/getting-started/introduction>
6. Heroku [Электронный ресурс] - Режим доступа: <https://ru.wikipedia.org/wiki/Heroku>
7. Gunicorn [Электронный ресурс] - Режим доступа: <https://docs.gunicorn.org/en/stable/>
8. SASS [Электронный ресурс] - Режим доступа: <https://sass-scss.ru/>

**Приложение А. Скриншоты работы веб-ресурса**

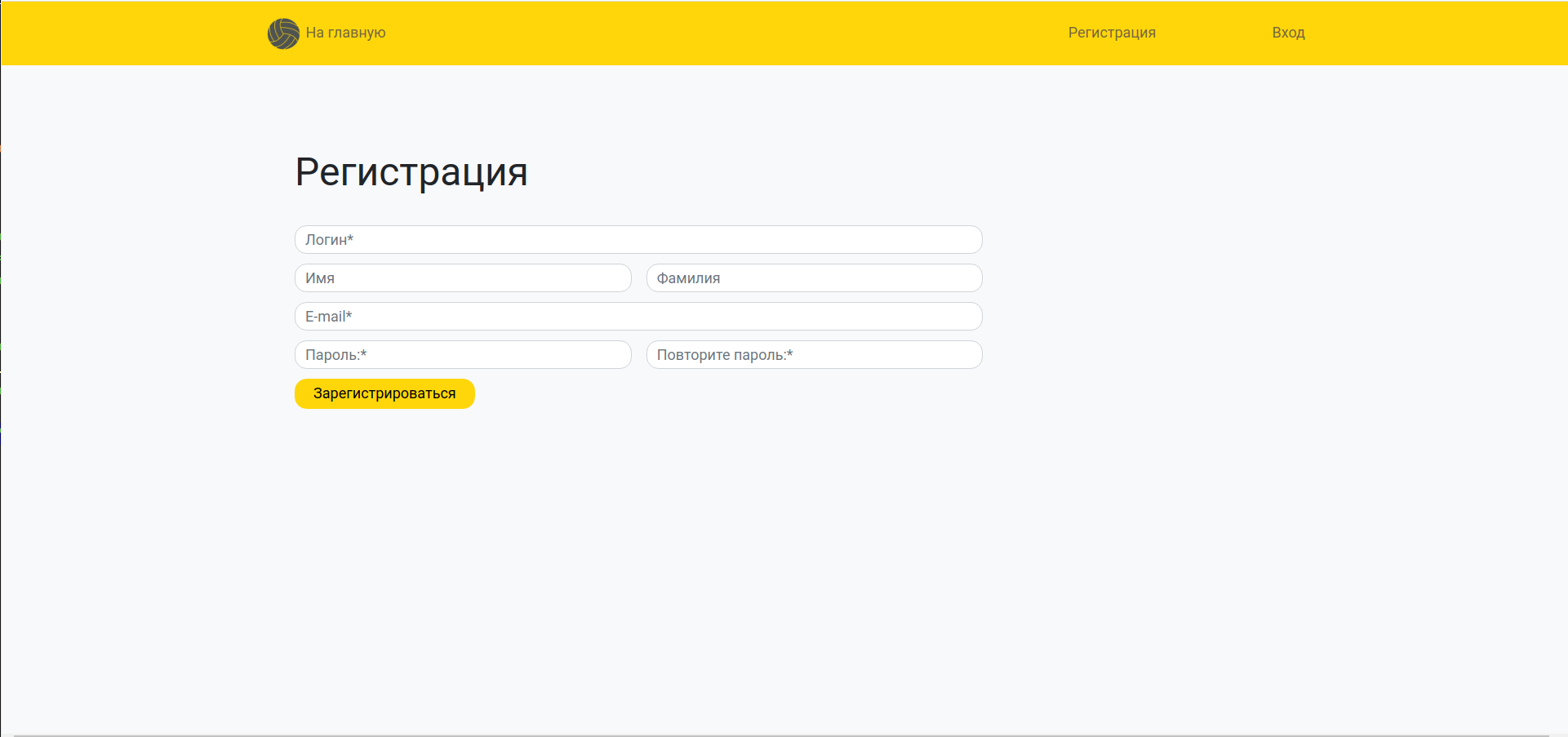
****

Рисунок 1 - форма регистрации пользователя

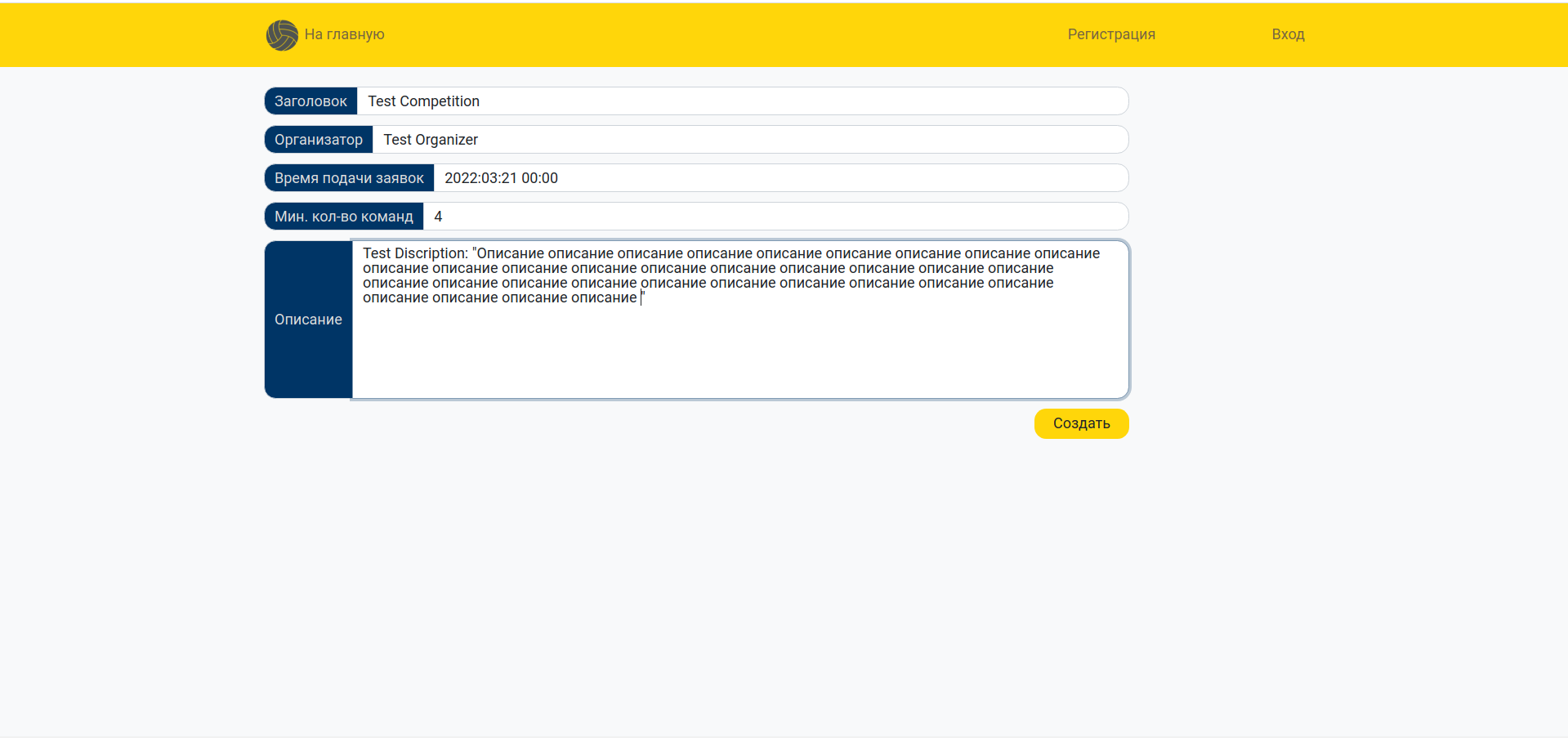


Рисунок 2 - форма создания соревнования

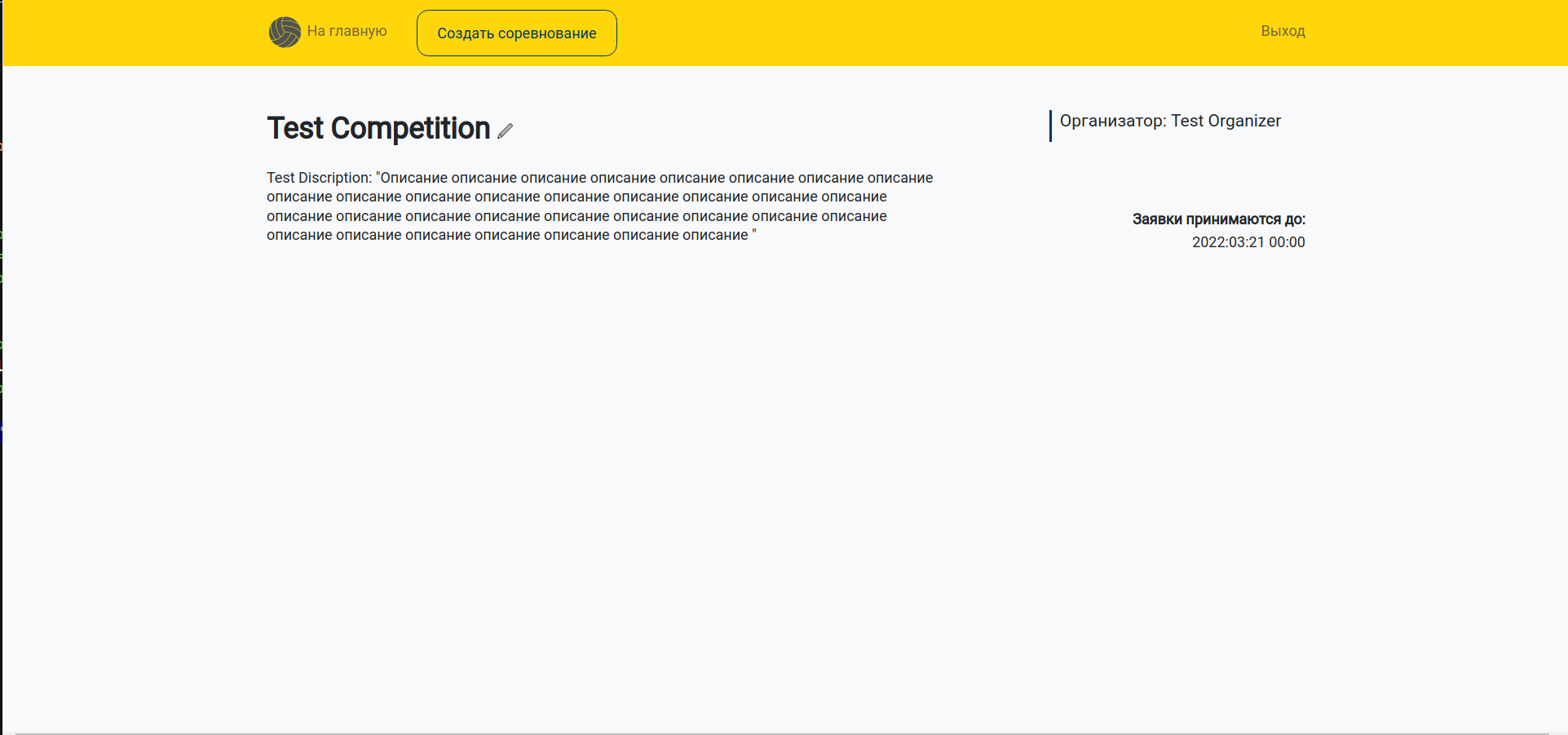


Рисунок 3 - Созданное соревнование

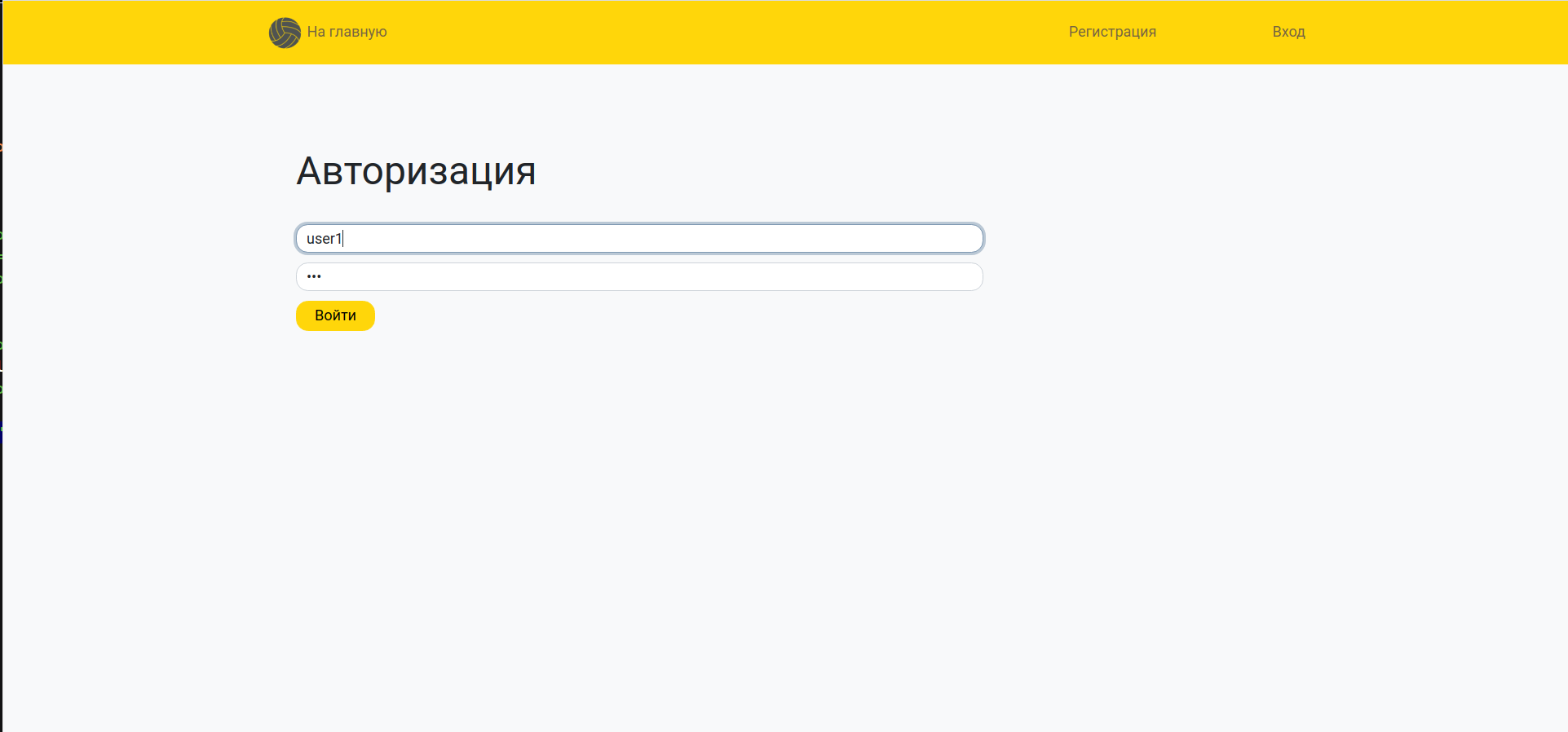


Рисунок 4 - Форма авторизации

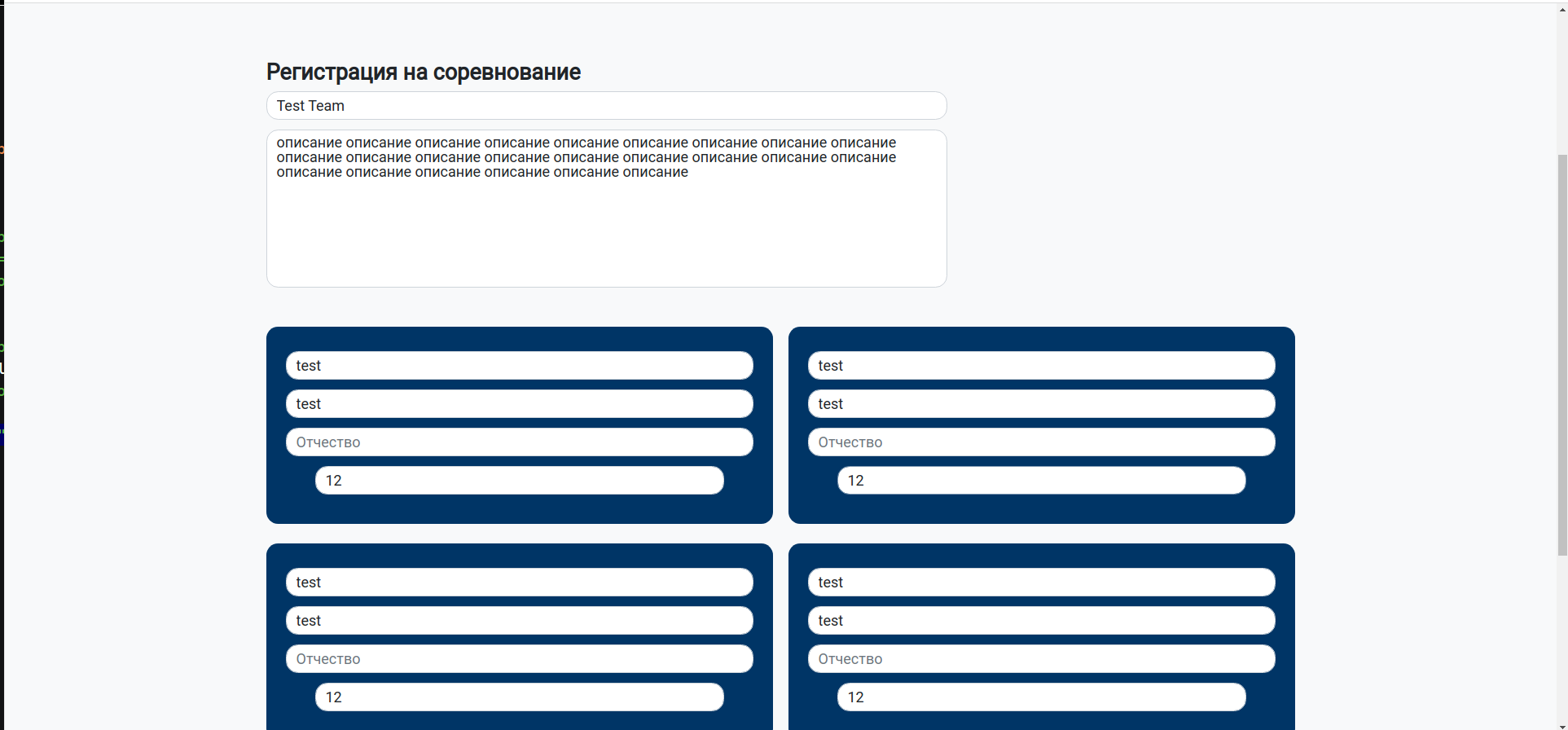


Рисунок 5 - форма регистрации команды

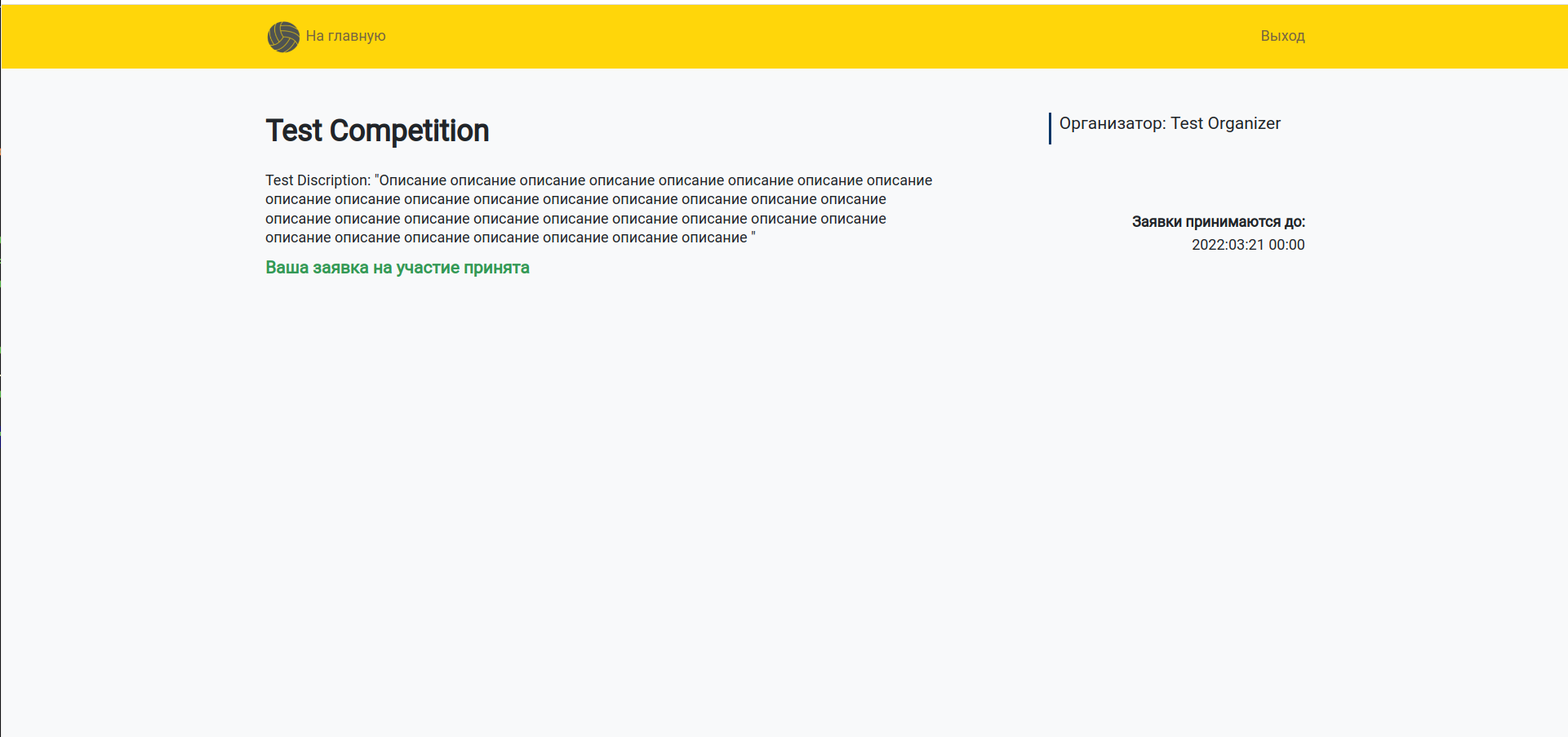


Рисунок 6 - страница соревнования после подачи заявки на участие

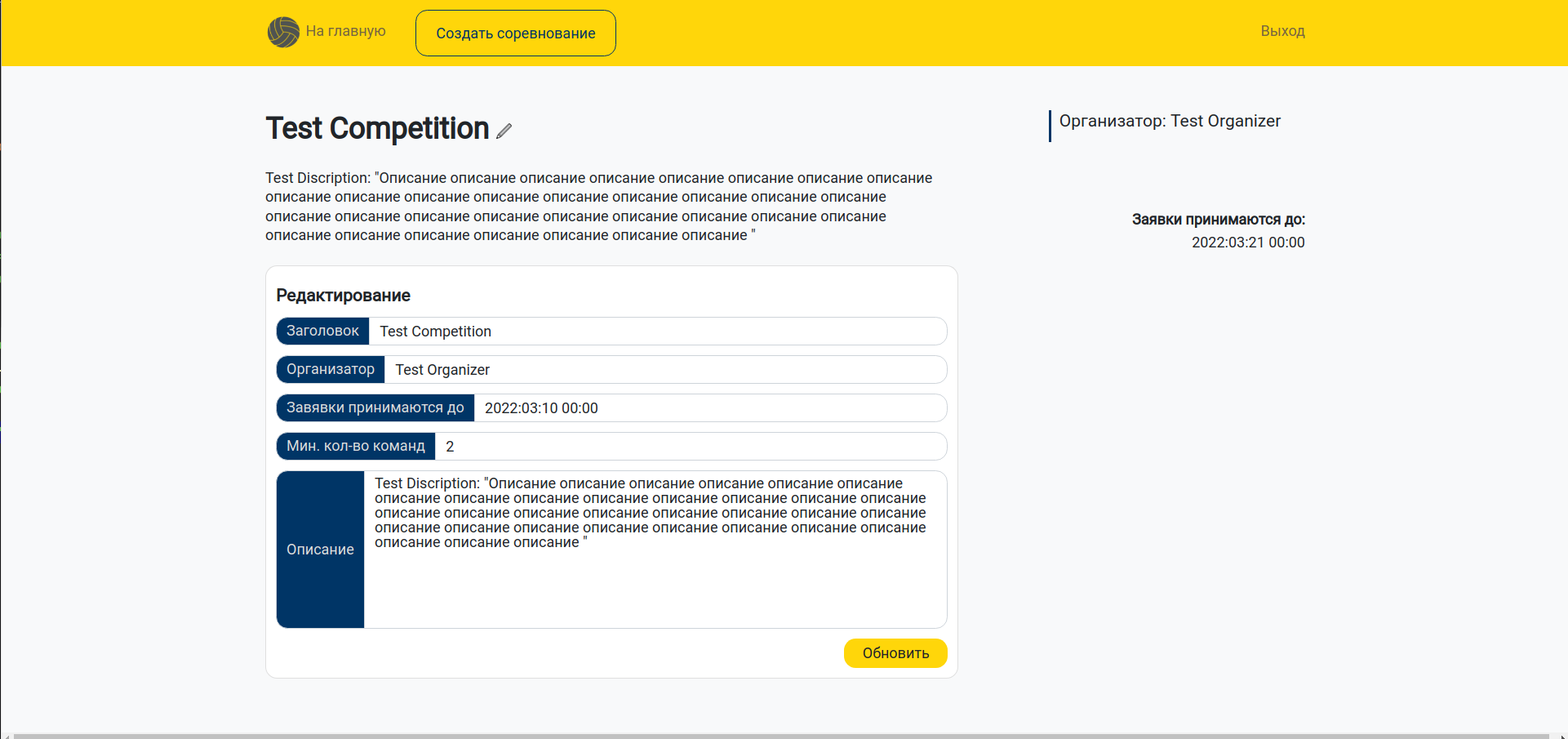


Рисунок 7 - форма редактирования соревнований

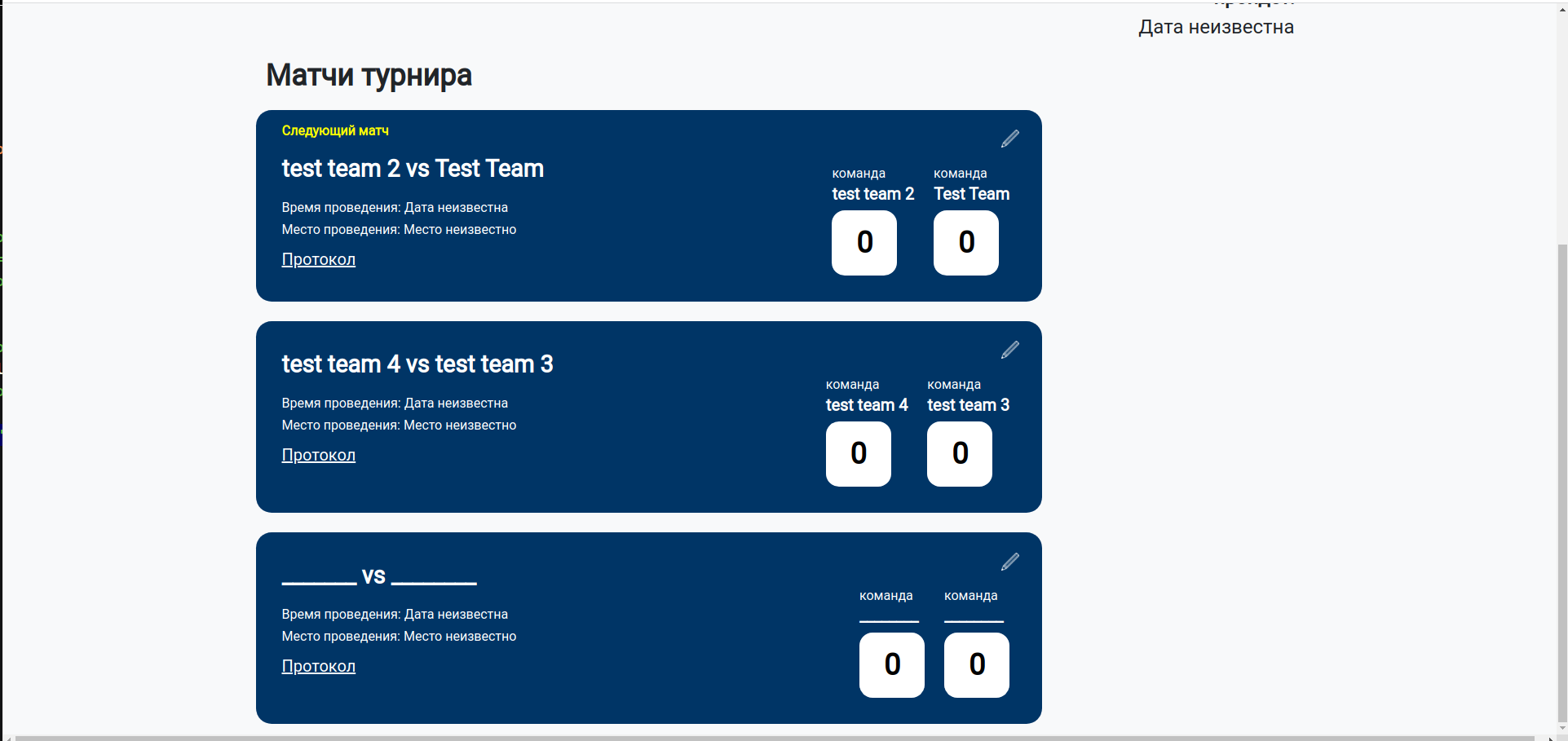


Рисунок 8 - виджеты матчей турнира

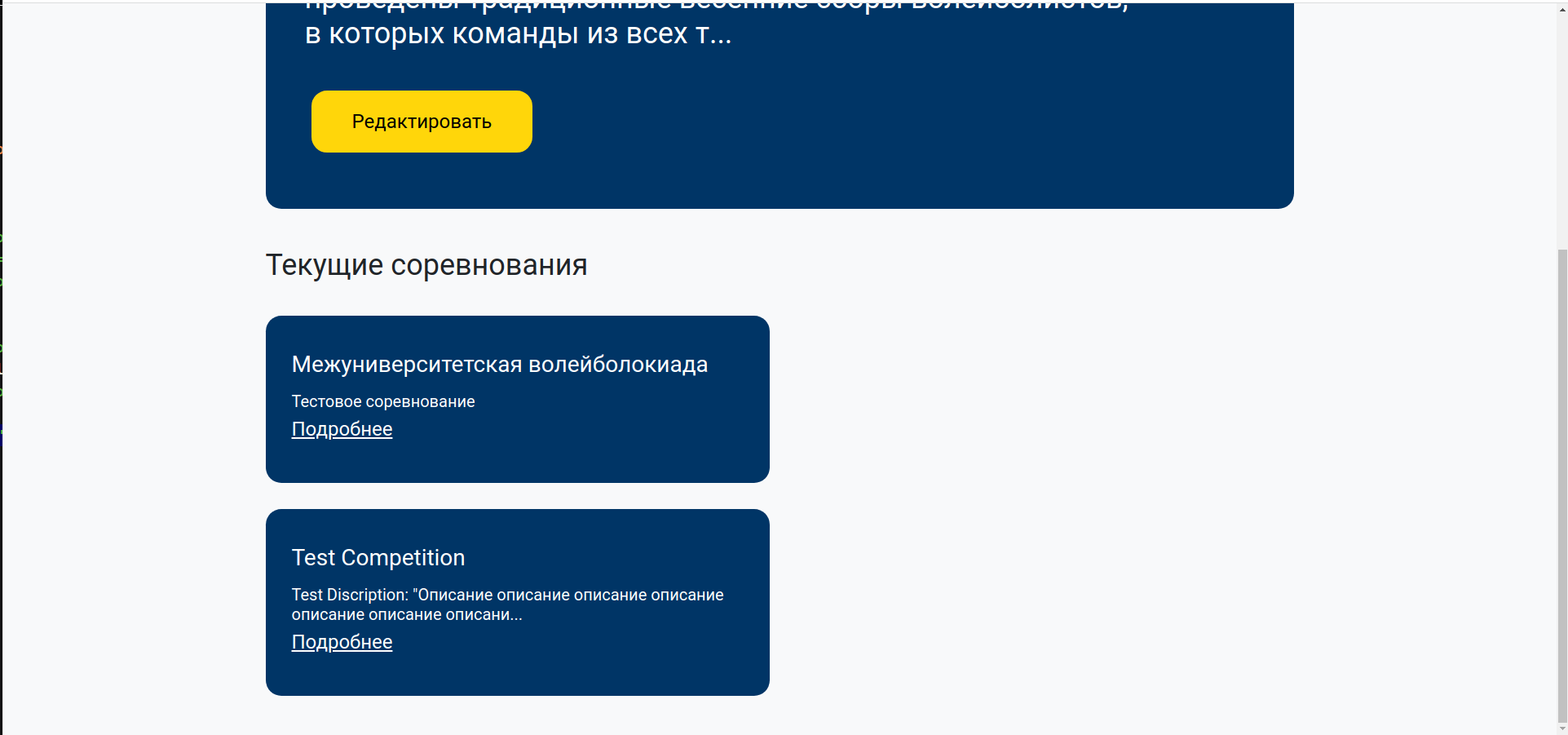


Рисунок 9 - турнир переместился в разряд текущих

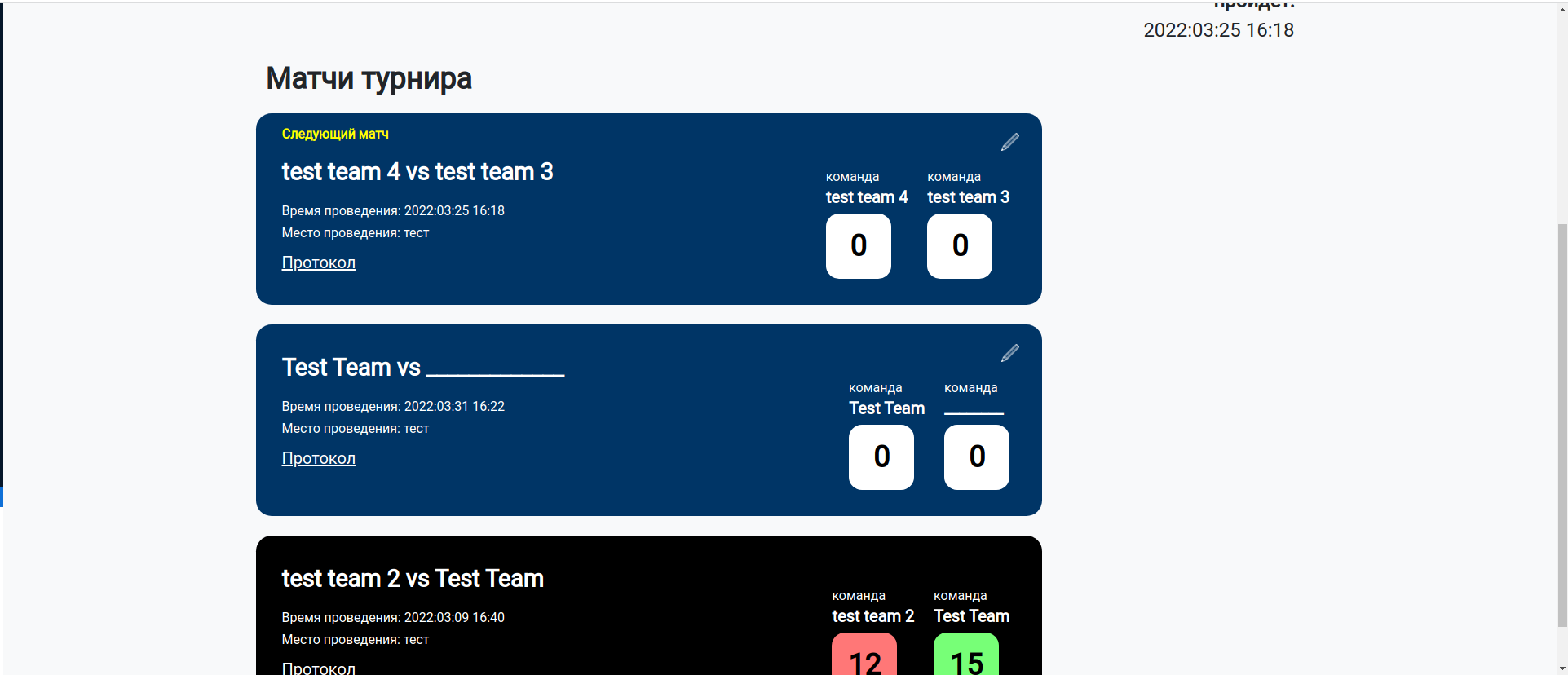


Рисунок 10 - изменение турнирной сетки после завершения матча

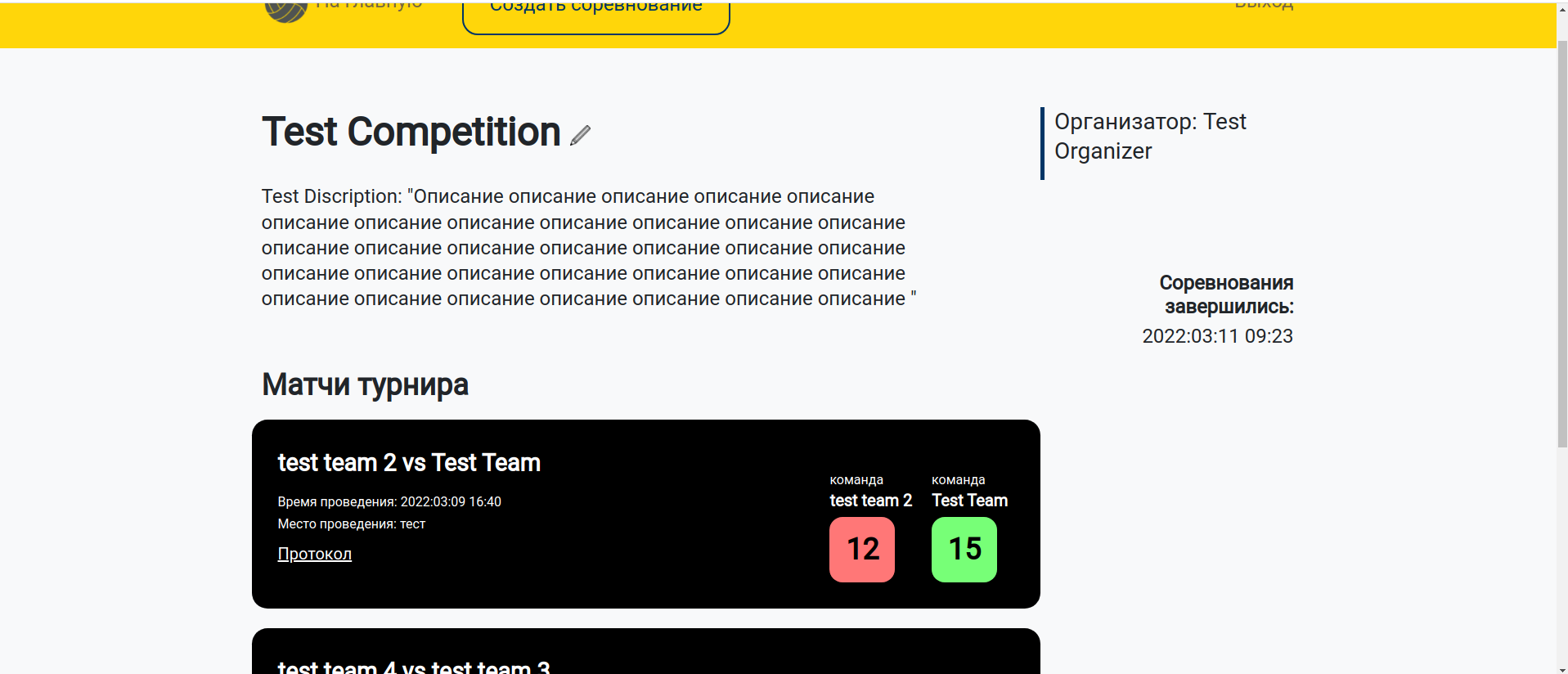


Рисунок 11 - турнир после завершения всех матчей

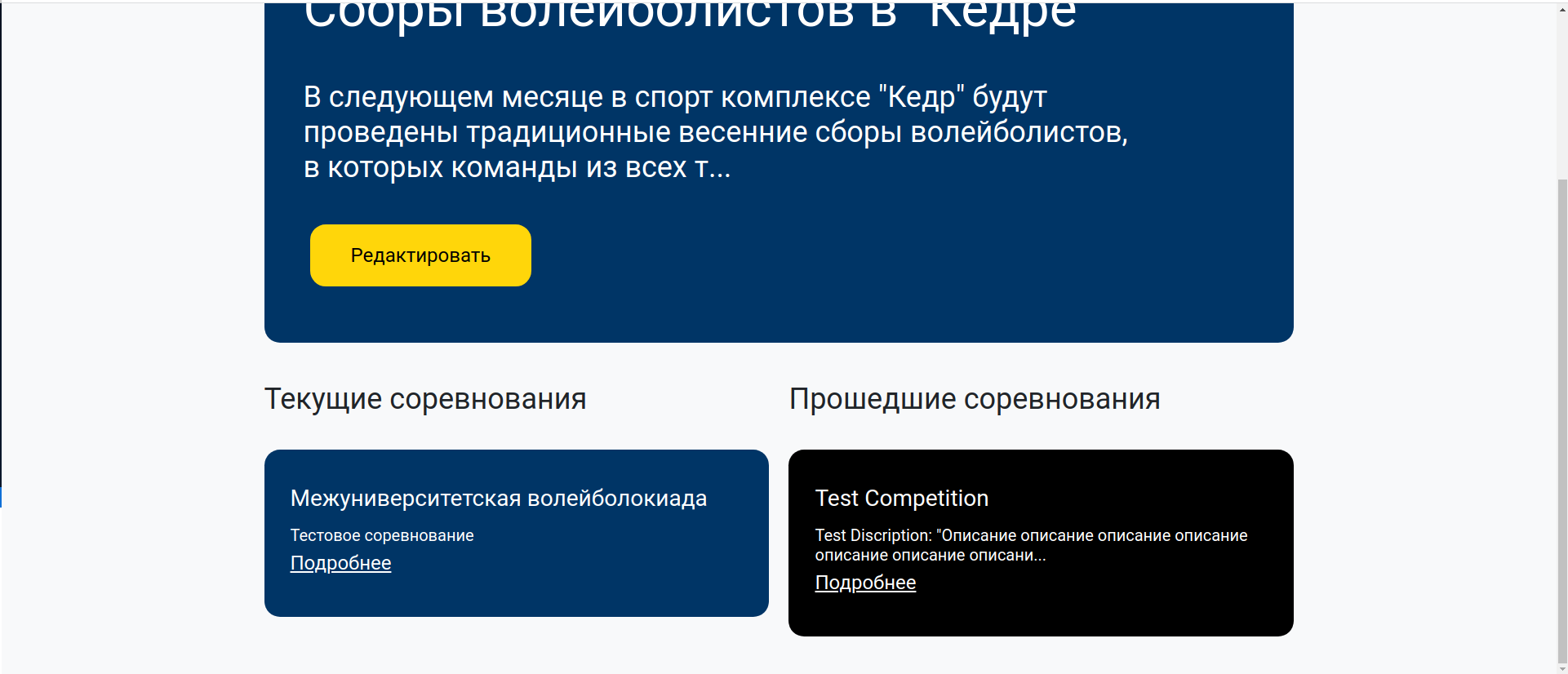


Рисунок 12 - главная страница после завершения матча