# АННОТАЦИЯ

Пояснительная записка к дипломному проекту содержит постановку и программу решения задачи «Разработка веб-приложения для проведения соревнований по дзюдо».

Программа JudoArena написана на языке Python 3.11 с использованием фреймворка Django в среде программирования PyCharm Community Edition 2024.1 с использованием системы управления базами данных MySQL. Программа предназначена для работы в операционной системе Microsoft Windows 10 и выше, отлажена на данных контрольного примера.

# СОДЕРЖАНИЕ

|  |  |
| --- | --- |
|  | лист |
| Введение | 4 |
| 1 Постановка задачи | 6 |
| 1.1 Описание предметной области | 6 |
| 1.2 Проектирование бизнес-процессов предметной области | 10 |
| 1.3 Описание входной информации | 11 |
| 1.4 Описание выходной информации | 11 |
| 1.5 Общие требования к программному продукту | 12 |
| 1.6 Описание структуры базы данных | 13 |
| 1.7 Контрольный пример | 16 |
| 2 Экспериментальный раздел | 17 |
| 2.1 Описание программы | 17 |
| 2.2 Протокол тестирования программного продукта | 18 |
| 2.3 Руководство пользователя | 29 |
| Заключение | 33 |
| Приложения А Шаблоны выходных документов | 34 |
| Приложения Б. Схема базы данных | 37 |
| Приложение В. Входные данные контрольного примера | 38 |
| Приложение Г. Выходные данные контрольного примера | 39 |
| Приложение Д. Код программы |  |
| Список сокращений | 40 |
| Список использованных источников | 41 |
|  |  |
|  |  |
|  |  |
|  |  |

# ВВЕДЕНИЕ

Дзюдо зародилось в Японии 19 века. Дисциплина была создана на основе джиу-джитсу. Свое активное развитие в России получило в 60-х годах прошлого века. И развивается по сей день, с каждым готов число занимающихся не уклонно растет. Все больше проводиться соревнований, по различным возрастным и весовым категориям, для обмена опытом, навыками и выявления сильнейшего спортсмена.

Дзюдо - Олимпийский вид спорта, без оружия, в основе которого лежит бросковая техника боя, с применением болевых и удушающих приемов. Есть четкое деление на традиционное дзюдо и спортивное дзюдо. Первое фокусируется на самообороне и философии «Гибкого пути», а спортивное, для которого разработана программа, акцентирует внимание на соревновательной части, для выявления сильнейшего спортсменав каждой весовой и возрастной категории. Весовые категории: спортсмены делятся на весовые категории, чтобы обеспечить более равные условия соревнований. Обычно устанавливаются различные интервалы веса, например, легкий вес, средний вес, полутяжелый вес и тяжелый вес. Возрастные категории: соревнования могут проводиться в различных возрастных группах, чтобы спортсмены могли соревноваться среди своих сверстников.

Как и любая спортивная дисциплина, в дзюдо есть ряд правил. Основаные из них относятся к очерёдности этапов проведения соревнований, обязанностям судей и спортсменов, порядок проведения встреч. Ключевые этапы это – организация соревнования, регистрация спортсменов, жеребьевка, соревновательные бои, церемония награждения. Где происходи документооборот от создания протокола соревнований до таблицы результатов.

Актуальность данной работы обусловлена тем, что на данный момент документация на большинстве соревнованиях по дзюдо представлена в виде бумажного носителя, заполняемого от руки. Это создает проблемы в эффективности и оперативности проведения соревнований. Ручное заполнение бумажной документации требует значительного времени и усилий, что может привести к задержкам и ошибкам при обработке данных. Кроме того, бумажные документы могут быть утеряны или повреждены, что дополнительно затрудняет доступ к информации о результатах соревнований. В связи с этим, разработка приложения, которое упрощает и ускоряет взаимодействие судей и спортсменов с документацией, имеет большую актуальность. Такое приложение может предоставлять электронную форму для заполнения данных об спортсменах и их результатах, что сократит время и усилия, затрачиваемые на обработку информации. Оно также может обеспечить хранение данных в электронном виде, что предотвратит потерю или повреждение документов.

Цель дипломного проекта заключается в разработке приложения, которое будет способствовать ускорению и упрощению взаимодействия судей и спортсменов с документацией на соревнованиях по дзюдо.

Задачами дипломного проекта являются:

* описать предметную область;
* разработать структуру базы данных;
* разработать приложение;
* провести тестирование приложения.

# 1 Постановка задачи

## 1.1 Описание предметной области

Требуется разработать информационную систему для проведений соревнований по дзюдо.

Данная информационная система предполагает наличие четырех групп пользователей: спортсмен, тренер, судья и секретарь соревнования.

У спортсмена есть возможность:

* зарегистрироваться и войти;
* редактировать свой профиль;
* просматривать списки запланированных и проведённых соревнований;
* просматривать отчёты соревнований;
* просматривать подробную информацию о соревновании.

В базе данных должна храниться следующая информация об спортсмене:

* фамилия;
* имя;
* отчество;
* вес;
* дата рождения;
* номер медицинской страховки;
* логин;
* пароль.

Спортсмен должен соответствовать своей возрастной и весовой категории.

У тренера есть возможность:

* зарегистрироваться и войти;
* редактировать свой профиль;
* просматривать списки запланированных и проведённых соревнований;
* просматривать отчёты соревнований;
* просматривать подробную информацию о соревновании;
* подавать заявку на участие.

В базе данных должна храниться следующая информация о тренере:

* фамилия;
* имя;
* отчество;
* номер лицензии;
* логин;
* пароль.

У судии есть возможность:

* войти;
* редактировать свой профиль;
* просматривать списки запланированных и проведённых соревнований;
* просматривать отчёты соревнований;
* просматривать подробную информацию о соревновании;
* сформировать график встреч спортсменов;
* cформировать отчет встреч.

В базе данных должна храниться следующая информация о судье:

* фамилия;
* имя;
* отчество;
* категория;
* логин;
* пароль.

Секретарь соревнования имеет возможность:

* войти;
* редактировать свой профиль;
* просматривать списки запланированных и проведённых соревнований;
* просматривать отчёт соревнований;
* просматривать подробную информацию о соревновании;
* заносить результаты взвешивания;
* формирование отчета спортсменов;
* занесение результатов встреч;
* формирования итогов соревнований;
* регистрация соревнований;
* добавления списка судей на соревнование.

В базе данных должна храниться следующая информация о секретаре соревнования:

* фамилия;
* имя;
* отчество;
* логин;
* пароль.

Авторизация.

При запуске приложения окно авторизации – первое, что видит пользователь. На нем пользователю предлагается ввести свой логин и пароль или есть возможность прейти на окно регистрации тренера и спортсмена. Неавторизованный пользователь не имеет доступа к списку соревнований.

Список соревнований.

После успешной авторизации или регистрации пользователь переходит к странице со списком соревнований, где представлена следующая информация:

* название соревнования;
* дата проведения;
* адрес.

На странице представлен функционал:

* выход к окну авторизации;
* поиск по названию соревнования;
* фильтрация по дате: все статусы, в ожидании, завершенные и в процессе;
* фильтровать по дате;
* секретарь может запланировать новое соревнование.

Подробно о соревновании.

Страница «Подробнее» представлена информация:

* название соревнования;
* дата проведения;
* адрес;
* главный судья;
* весовые и возрастные категории.

Страница «Список участников» представлена информация:

* весовые и возрастные категории;
* список спортсменов: Фамилия И.О., вес, дата рождения, Фамилия И.О. тренера;
* на данной странице тренер может подать заявку.

Все пользователи могут сформировать отчет по соревнованию, где представлены все спортсмены, сгруппированные по категориям.

Планирование нового соревнования производиться секретарём.

Поля ввода:

* название соревнования;
* дата проведения;
* описание;
* главный судья;
* адрес.

После планирования соревнования секретарь назначает судей на данное соревнование, так же это можно сделать на странице «Подробнее».

Далее секретарь вносит весовые и возвратные категорий, при необходимости может создать новые категории, так же это можно сделать на странице «Подробнее».

Подачей заявок на соревнование занимается тренер на страницу «Список спортсменов», там предоставлен список спортсменов прикреплённые к данному тренеру, где он может найти необходимого человека и подать с ним заявку.

## 1.2 Проектирование бизнес-процессов предметной области

Диаграмма прецедентов — это схема, отражающая взаимодействие между системой и её пользователями или другими системами. На ней изображаются актеры и прецеденты. Линии связи показывают, как актеры взаимодействуют с системой. Диаграмма прецедентов показана на рисунке 1.2.1.

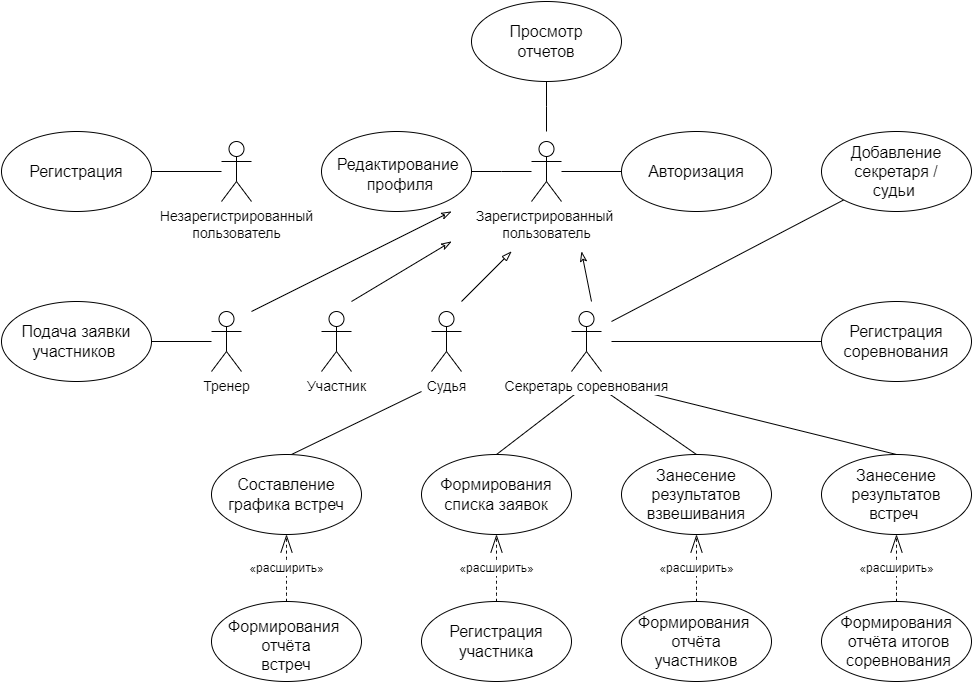


Рисунок 1.2.1 – Диаграмма прецедентов

## 1.3 Описание входной информации

Входной информацией для выполнения задачи являть данные тренеров и спортсменов, данные запланированного соревнования, данные поданных заявок, словари весовых и возрастных категорий.

## 1.4 Описание выходной информации

Выходной информацией являются:

* протокол соревнования:
* протокол жеребьевки;
* итоговый протокол.

Протокол соревнования:

* заголовок: название соревнования, дата, адрес, описание, список судей;
* тело документа: наименование категории, данные спортсмена;
* окончание: подпись организатор, дата формирование документа.

Протокол жеребьевки:

* заголовок: название соревнования;
* тело документа: наименование категории, данные спортсменов, порядок встреч;

окончание: подпись организатор, дата формирование документа.

Итоговый протокол:

* заголовок: название и номер соревнования, дата, адрес;
* тело документа: наименование категории, данные спортсменов, результаты встреч, призеры;
* окончание: подпись организатор, дата формирование документа.

Описание выходных документов представлено в таблице 1.4.1.

Таблица 1.4.1 – Описание выходных документов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование документа | Периодичность выдачи документа | Кол-во экз. | Куда передаются | Поля сортировки | Поля группировки | Итоги |
| Протокол жеребьевка | После окончания жеребьевки | По мере необходимости | Тренерам, судейскому составу | - | Весовая категория | - |
| Протокол соревнования | После окончания взвешивания | По мере необходимости | Тренерам, судейскому составу | - | Весовая категория | - |
| Итоговый протокол | После окончания соревнования | По мере необходимости | Тренерам, судейскому составу | - | Весовая категория | - |

Шаблоны выходных документов представлен в приложении А.

## 1.5 Общие требования к программному продукту

Пользователи должны иметь базовые навыки пользования персональным компьютером.

Минимальные требования к техническому обеспечению программного продукта следующие:

* ОС:  Windows 10 Service Pack1/ Windows 7 64Bit Service Pack1/ Windows 8.1 64Bit / Windows 10 64Bit / Mac OS;
* процессор: 2.40 Ггц (четырехъядерный) / AMD Phenom 9850 (четырехъядерный) @ 2.5 Ггц;
* оперативная память: 512МБ (Win 10 и выше);
* видеокарта: NVIDIA 9800 GT c 512 МБ видеопамяти/ AMD HD 4870 с 1 Гб видеопамяти (DX 9, 10, 10.1);
* жесткий диск: 10 гигабайт свободного места;
* Microsoft DirectX версия 9.0c.

Функциональные возможности программного продукта:

* приложение должно формировать и отображать выходные данные пользователю;
* в приложении должен быть обеспечен просмотр таблиц (справочников) базы данных с возможность добавления, редактирования, удаления данных.

Требования к надежности:

* приложение должно обрабатывать ошибочные действия пользователя и сообщать ему об этом;
* приложение должно обеспечивать контроль входной и выходной информации.

Требования к информационной и программной совместимости: обеспечить работу приложения с таблицами СУБД MySQL.

## 1.6 Описание структуры базы данных

Схема базы данных – это структура базы данных, описанная на формальном языке, поддерживаемом СУБД (системой управления базы данных). В реляционных базах данных схема определяет таблицы, поля в каждой таблице и ограничения целостности, такие как первичный и внешний ключи.

Схема базы данных представлена в приложении Б.

Описание структуры базы данных представлено в таблицах 1.6.1 – 1. 6.11.

Таблица 1.6.1 – Group (Группы пользователей)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID группы | INT(4) | PK[[1]](#footnote-1) |
| name | Название | VARCHAR(150) | Обязательное поле |

Таблица 1.6.2 – TrainerParticipant (Спортсмен и его тренер)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID | INT(4) | PK |
| user\_participant\_id | ID спортсмена | INT(4) | FK[[2]](#footnote-2) |
| user\_trainer\_id | ID тренера | INT(4) | FK |

Таблица 1.6.3 – ExpansionUser (Расширение базового класса пользователе)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID Расширения | INT(4) | PK |
| patronymic | Отчество | VARCHAR(45) | Обязательное поле |
| category\_referee | Категория судьи | INT(4) | Необязательное поле |
| license\_number\_trainer | Номер лицензии тренера | VARCHAR(10) | Необязательное поле |
| medical\_insurance\_participants | Медицинская страховка спортсмена | VARCHAR(10) | Необязательное поле |
| weight\_participants | Вес спортсмена | DECIMAL(6) | Необязательное поле |
| date\_birth\_participants | Дата рождения спортсмена | DATE(8) | Необязательное поле |
| user\_id | ID пользователя | INT(4) | FK |

Таблица 1.6.4 – User (Данные пользователя)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | ­­­­Тип ключа |
| id | ID пользователя | INT(4) | PK |
| password | Пароль | VARCHAR(128) | Обязательное поле |
| last\_login | Дата последнего входа | DATETIME(8) | Необязательное поле |
| is\_superuser | Суперпользователь | TINYINT(1) | Обязательное поле |
| username | Логин | VARCHAR(150) | Обязательное поле |
| first\_name | Имя | VARCHAR(150) | Обязательное поле |
| last\_name | Фамилия | VARCHAR(150) | Обязательное поле |
| email | Email | VARCHAR(254) | Обязательное поле |
| is\_staff | Персонал | TINYINT(1) | Обязательное поле |
| is\_active | Активен | TINYINT(1) | Обязательное поле |
| date\_joined | Дата создания | DATETIME(8) | Обязательное поле |

Таблица 1.6.5 – Application (Заявка)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID заявки | INT(4) | PK |
| participant\_id | ID спортсмена | INT(4) | FK |
| category\_id | ID категории | INT(4) | FK |
| trainer\_id | ID тренера | INT(4) | FK |
| date | Дата заявки | DATE(8) | Обязательное поле |
| status\_application | Статус заявки | TINYINT(1) | Обязательное поле |

Таблица 1.6.6 – GroupPermissions (Групповые разрешения)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID группового разрешения | INT(4) | PK |
| group\_id | ID группы | INT(4) | FK |
| permission\_id | ID разрешения | INT(4) | FK |

Таблица 1.6.7 – Competition (Соревнование)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID соревнования | INT(4) | PK |
| name\_competition | Название соревнования | VARCHAR(100) | Обязательное поле |
| date\_event | Дата проведения | DATE(8) | Обязательное поле |
| address | Адрес | VARCHAR(127) | Обязательное поле |
| status\_competition | Статус соревнования | TINYINT(1) | Обязательное поле |
| slug | Уникальное название | VARCHAR(255) | Обязательное поле |
| organizer\_id | ID организатора | INT(4) | FK |
| date\_end | Дата конца | DATE(8) | Обязательное поле |
| description | Описание соревнования | VARCHAR(150) | Необязательное поле |

Таблица 1.6.8 – Category (Категории соревнований)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID категории | INT(4) | PK |
| id\_competition | ID соревнования | INT(4) | FK |
| id\_weight | ID веса | INT(4) | FK |
| id\_age | ID возраста | INT(4) | FK |

Таблица 1.6.9 – Weight (Вес)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID веса | INT(4) | PK |
| weight\_start | Начала весовой категории | DECIMAL(6) | Обязательное поле |
| weight\_end | Конец весовой категории | DECIMAL(6) | Обязательное поле |

Таблица 1.6.10 – Age (Возраст)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID возраста | INT(4) | PK |
| age\_start | Начала возрастной категории | DATE(8) | Обязательное поле |
| age\_end | Конец возрастной категории | DATE(8) | Обязательное поле |

Таблица 1.6.11 – CompetitorReferee (Список судей)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID заявки | INT(4) | PK |
| competition\_id | ID соревнования | INT(4) | FK |
| referee\_id | ID судьи | INT(4) | FK |

Таблица 1.6.12 – Meet (Встреча)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Тип ключа |
| id | ID заявки | INT(4) | PK |
| sequence\_number | Порядковый номер | INT(4) | Необязательное поле |
| duration | Продолжительность | INT(4) | Необязательное поле |
| assessments | Оценки | LONGTEXT (512) | Необязательное поле |
| result | Результат | TINYINT(1) | Необязательное поле |
| id\_blue\_id | ID Спортсмена в синем | INT(4) | FK |
| id\_category\_id | ID категории | INT(4) | FK |
| id\_judge\_id | ID судьи | INT(4) | FK |
| id\_white\_id | ID Спортсмена в белом | INT(4) | FK |

## 1.7 Контрольный пример

Контрольный пример является ручным подсчётом задачи. По составленной программе обрабатываются исходные данные контрольного примера. Полученные результаты сравниваются с известными результатами контрольного примера. При несовпадении результатов производится поиск, исправление ошибок, и снова производится выполнение программы.

Входная информация контрольных примеров представлена в приложении В.

Выходные данные для контрольных примеров показаны в приложении Г.

# 2 Экспериментальный раздел

## 2.1 Описание программы

Программа имеет модульную структуру. При ее запуске выполняется проект на JudoArena. Схема взаимодействия модулей программы представлена на рисунке 2.1.1. Описание модулей и методов представлено в таблице 2.1.1.

bd\_juda\_arena

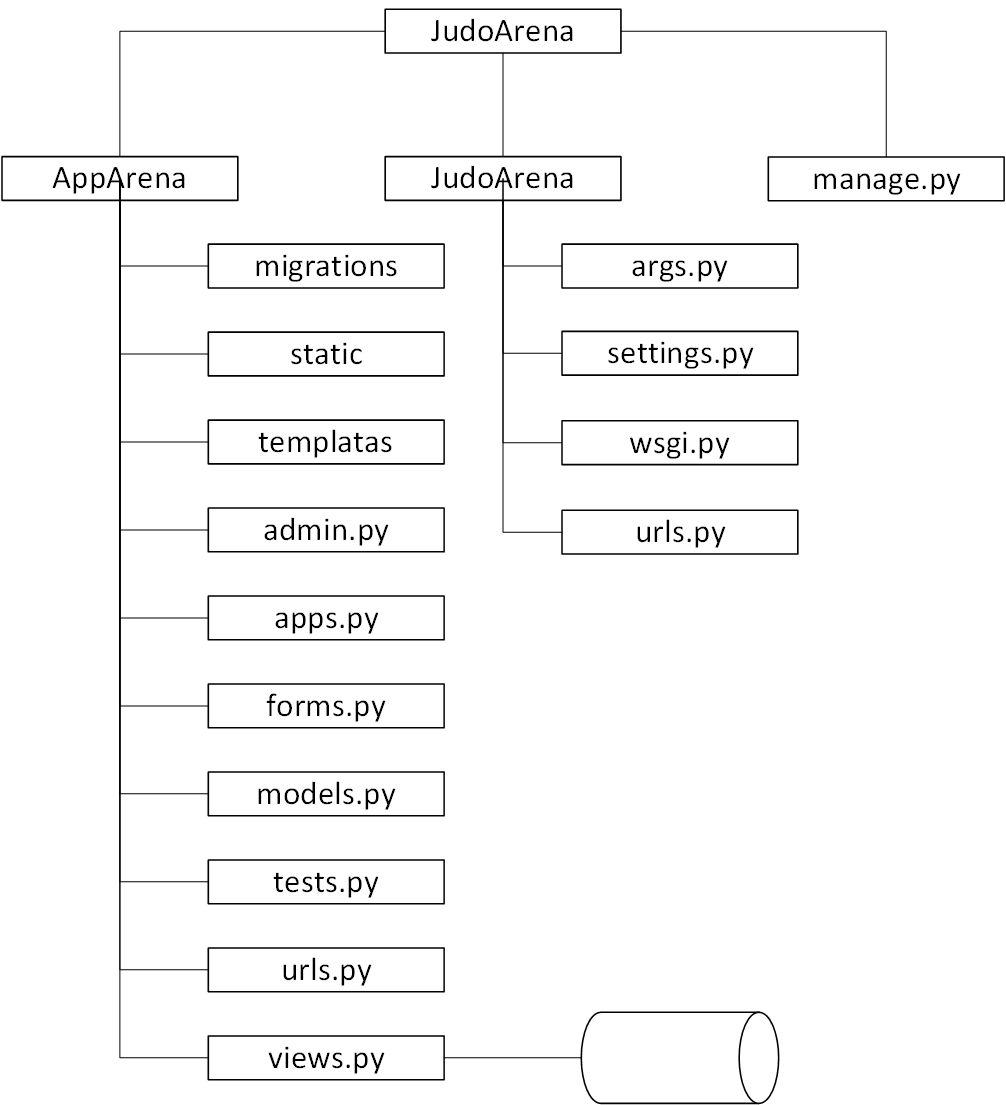
Все алгоритмы и код программы представлены в приложении Д

Рисунок 2.1.1 – Схема взаимодействия модулей

Таблица 2.1.1 - Описание модулей

|  |  |
| --- | --- |
| Элементы | Назначение |
| 1 | 2 |
| models.py | |
| ExpansionUser | Модель пользователя с дополнительными полями |
| TrainerParticipant | Модель списка тренеров и спортсменов |
| Competition | Модель соревнования |
| CompetitorReferee | Модель списка судей |
| Age | Модель возраста |
| Weight | Модель веса |
| Category | Модель категории |
| Application | Модель заявки |
| Meet | Модель встречи |
| forms.py | |
| SignUpForm | Форма регистрации |
| SignInForm | Форма авторизации |
| AddCompetitionForm | Форма добавление соревнования |
| ProfileForm | Форма редактирования профиля |
| ProfileExpansionForm | Форма пользователя с дополнительными полями |
| ProfileTrainerParticipant | Форма прикрепления спортсмена к тренеру |
| JudgesForm | Форма изменения списка судей на соревнование |
| CategoryForm | Форма добавление категории |
| AgeForm | Форма добавление возраста |
| WeightForm | Форма добавление веса |
| ApplicationForm | Форма добавление заявки |
| MeetForm | Форма добавление встречи |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| views.py | |
| register | Представление регистрации |
| user\_login | Представление авторизации |
| show\_competitions | Представление соревнований |
| add\_competition | Представление добавления соревнования |
| user\_logout | Представление выхода пользователя |
| edit\_profile | Представление редактирования профиля |
| reject\_application | Представление изменения статуса заявки |
| show\_competition | Представление соревнования |
| choose\_judges | Представление добавление седей |
| category | Представление категории |
| create\_category | Представление добавление категории |
| suitable\_participants | Представление подачи заявки |
| meets | Представление отображения свтреч |
| next\_power\_of\_two | Функция |
| power\_of\_two\_exponent | Функция |
| draw\_meet | Функция генерации жеребьёвки |
| edit\_meet | Представление проведения встречи |
| meet\_detail | Представление просмотра встречи |
| generate\_docx\_for\_competition | Функция генерации протокола соревнования |
| generate\_meets\_docx\_for\_competition | Функция генерации протокола жеребьевки |
| generate\_meets2\_docx\_for\_competition | Функция генерации итогового протокола |
| populate\_database.py |  |
| create\_trainers | Функция заполнения таблице user (Тренер) |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| create\_participants | Функция заполнения таблице user (Спортсмен) |
| category\_referee | Функция заполнения таблице user (Судья) |
| utils.py | |
| unique\_slugify | Функция генерации итогового протокола |
| group\_required | Функция генерации итогового протокола |

## 2.2 Протокол тестирования программного продукта

В ходе тестирования программного продукта на корректных и некорректных данных не было обнаружено ошибок, которые влияли бы на работу самого программного продукта и всей системы.

Данный программный продукт удовлетворяет всем предъявленным требованиям, имеет комфортный интерфейс и интуитивно понятный функционал, исключает появления системных ошибок.

В таблицах 2.2.1 - 2.2.11 представлены протоколы тестирования.

Таблица 2.2.1 – Тестирование авторизации на корректных данных

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 10.05.2024 |
| Приоритет тестирования | Высокий |
| Название теста | Авторизация на корректных данных |
| Резюме испытания | Каретная робота, безе аварийного завершения |
| Этапы теста | Запуск программы  Ввод данных  Авторизация |
| Тестовые данные | Пароль: your\_password  Логин: trainer\_0 |
| Ожидаемый результат | Переход на страницу с соревнованиями |
| Фактический результат | Переход на страницу с соревнованиями |

Результат тестирования на авторизацию пользователя представлен на рисунках 2.2.1-2.2.2.

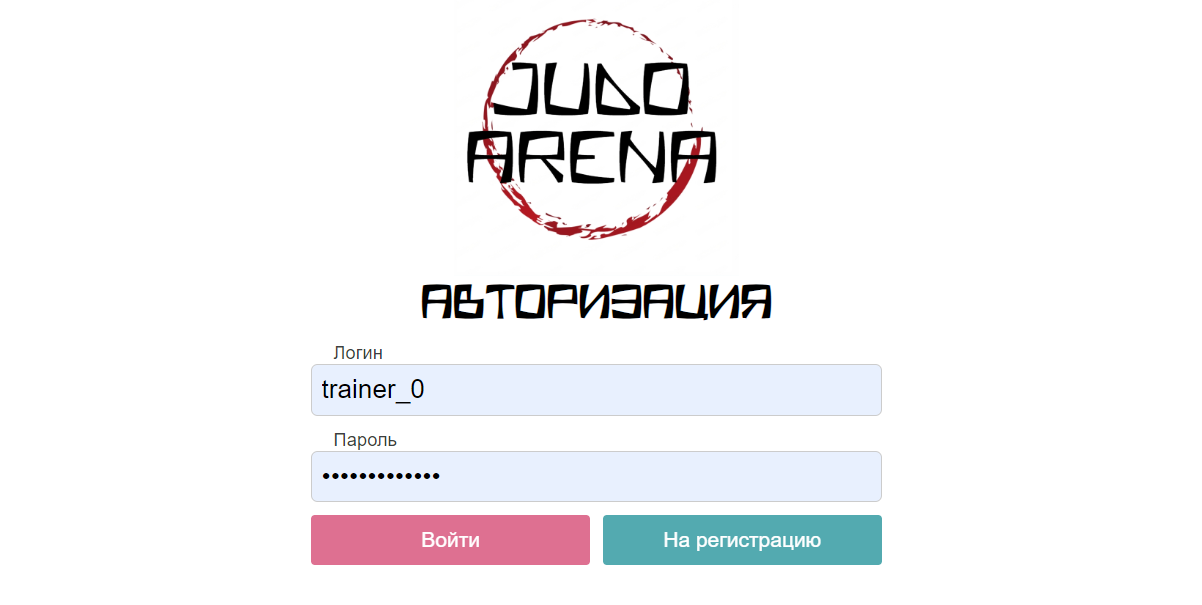


Рисунок 2.2.1 – Авторизация пользователя на корректных данных



Рисунок 2.2.2 – Переход на страницу отображения списка соревнований.

Таблица 2.2.2 – Тестирование добавления соревнования на каретных данных

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 17.06.2024 |
| Приоритет тестирования | Высокий |

Продолжение таблицы 2.2.2

|  |  |
| --- | --- |
| Название теста | Добавление соревнования на каретных данных |
| Резюме испытания | Каретная робота, безе аварийного завершения |
| Этапы теста | Переход с главной страницы на добавление соревнования  Заполнение данных  Добавлении |
| Тестовые данные | Название: Соревнования Мишутка  Адрес: г.Уфа, ул. Кирова, д.65  Дата начала: 17.06.2004  Дата конца: 18.06.2024  Главный судья: Петров П. O.  Описание: Первенство детей до 13 лет. |
| Ожидаемый результат | Переход на страницу добавления судей |
| Фактический результат | Переход на страницу добавления судей |

Результат тестирования на добавления соревнования представлен на рисунках 2.2.3 – 2.2.4.

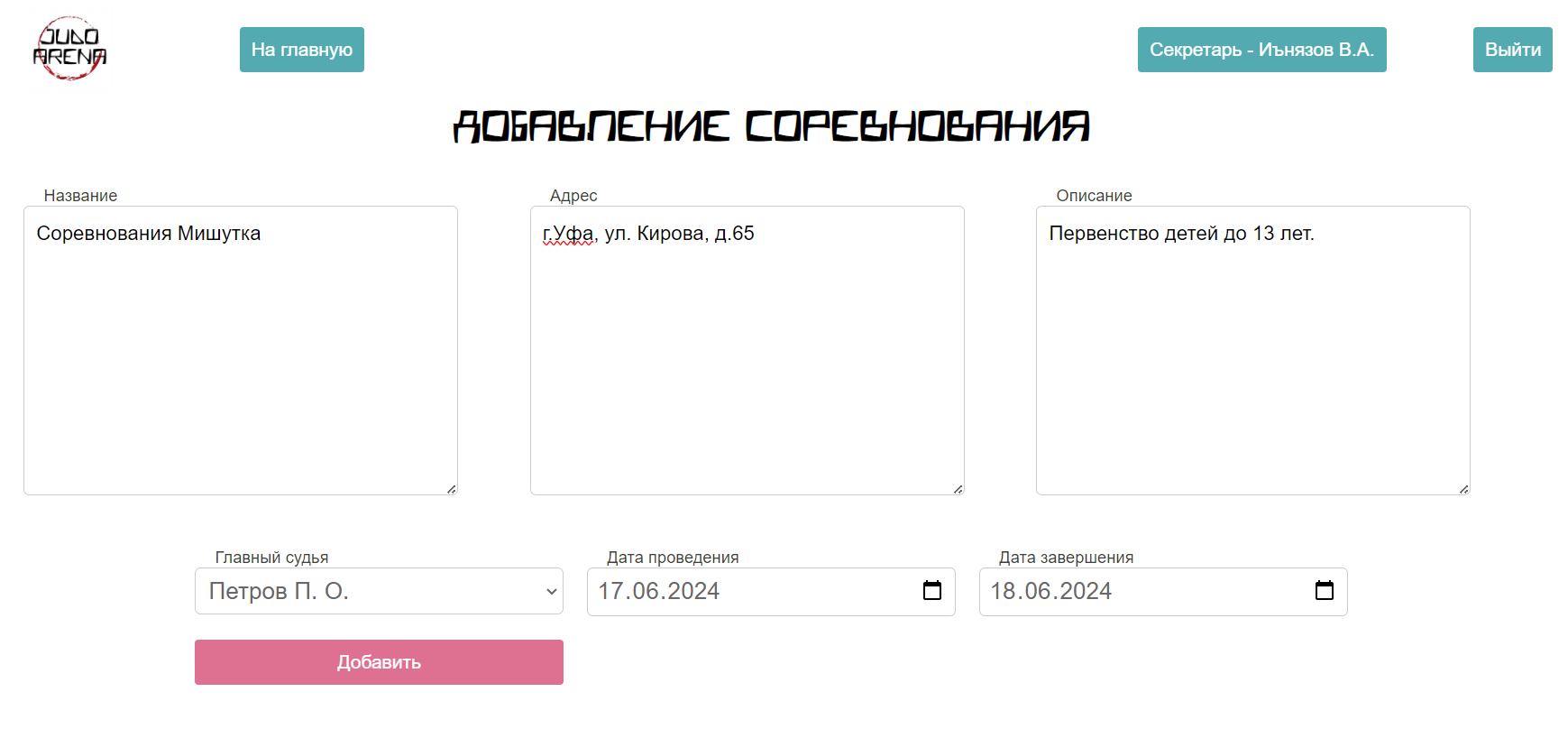


Рисунок 2.2.3 – Переход на страницу добавление соревнования



Рисунок 2.2.4 – Переход на страницу добавление судей.

Таблица 2.2.3 – Тестирование редактирования профиля на корректных данных

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 17.06.2024 |
| Приоритет тестирования | Высокий |
| Название теста | Редактирование профиля на корректных данных |
| Резюме испытания | Каретная робота, безе аварийного завершения |
| Этапы теста | Переход с главной страницы на профиль  Заполнение данных  Сохранение |
| Тестовые данные | Фамилия: Петров |
| Ожидаемый результат | Переход на страницу с соревнованиями и изменения отображения имени |
| Фактический результат | Переход на страницу с соревнованиями и изменения отображения имени |

Результат тестирования на редактирования профиля представлен на рисунках 2.2.5 – 2.2.6.

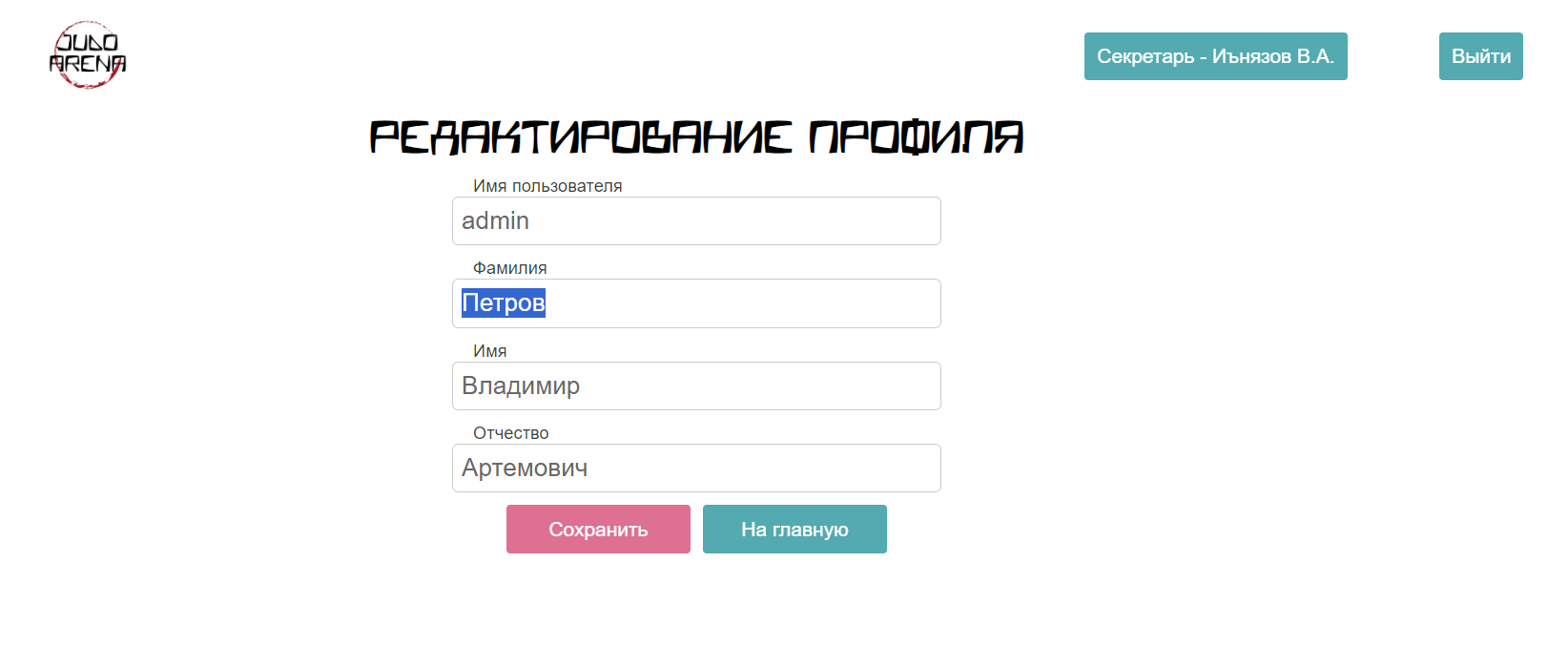


Рисунок 2.2.5 –Переход на страницу профиля

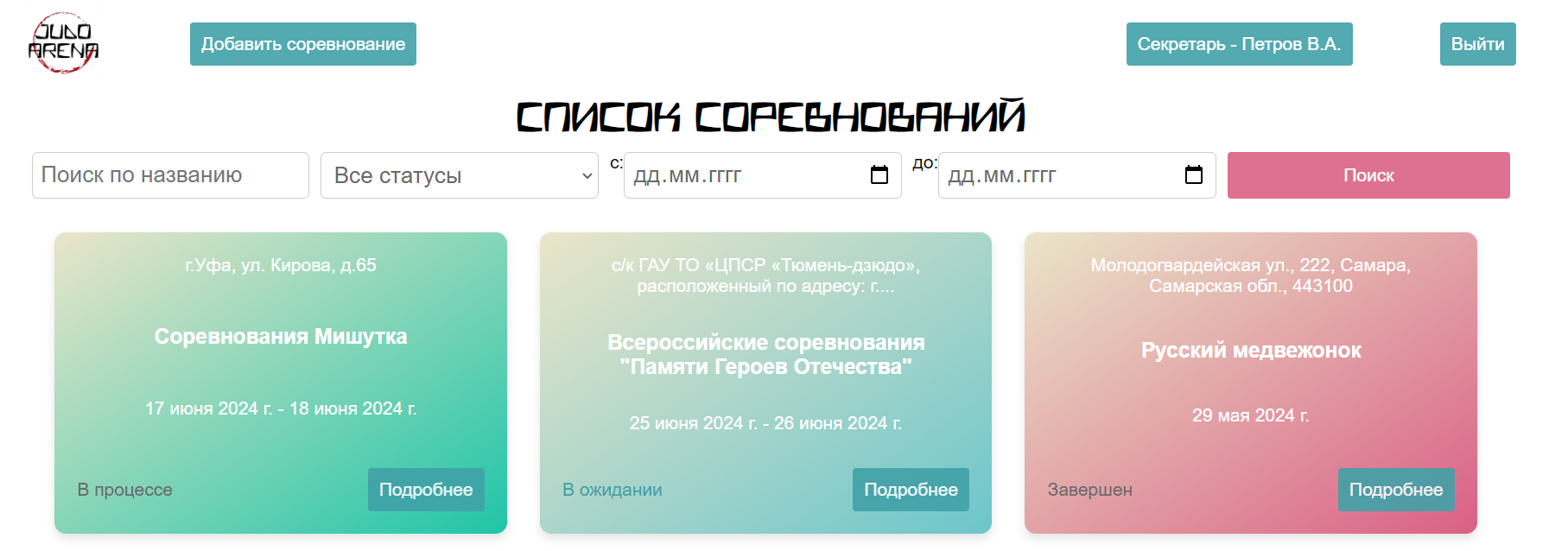


Рисунок 2.2.6 –Смена отображаемой фамилии.

Таблица 2.2.4 – Тестирование добавления судей соревнования

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 17.06.2024 |
| Приоритет тестирования | Средний |
| Название теста | Добавления судей соревнования |
| Резюме испытания | Каретная робота, безе аварийного завершения |

Продолжение таблицы 2.2.4

|  |  |
| --- | --- |
| 1 | 2 |
| Этапы теста | Переход с главной страницы на соревнования  Переход на страницу добавления судей  Изменения данных  Сохранение |
| Тестовые данные | Добавить судью: Graham M. В. |
| Ожидаемый результат | Переход на страницу соревнования |
| Фактический результат | Переход на страницу соревнования |

Результат тестирования на добавления судей представлен на рисунках 2.2.7 – 2.2.8.

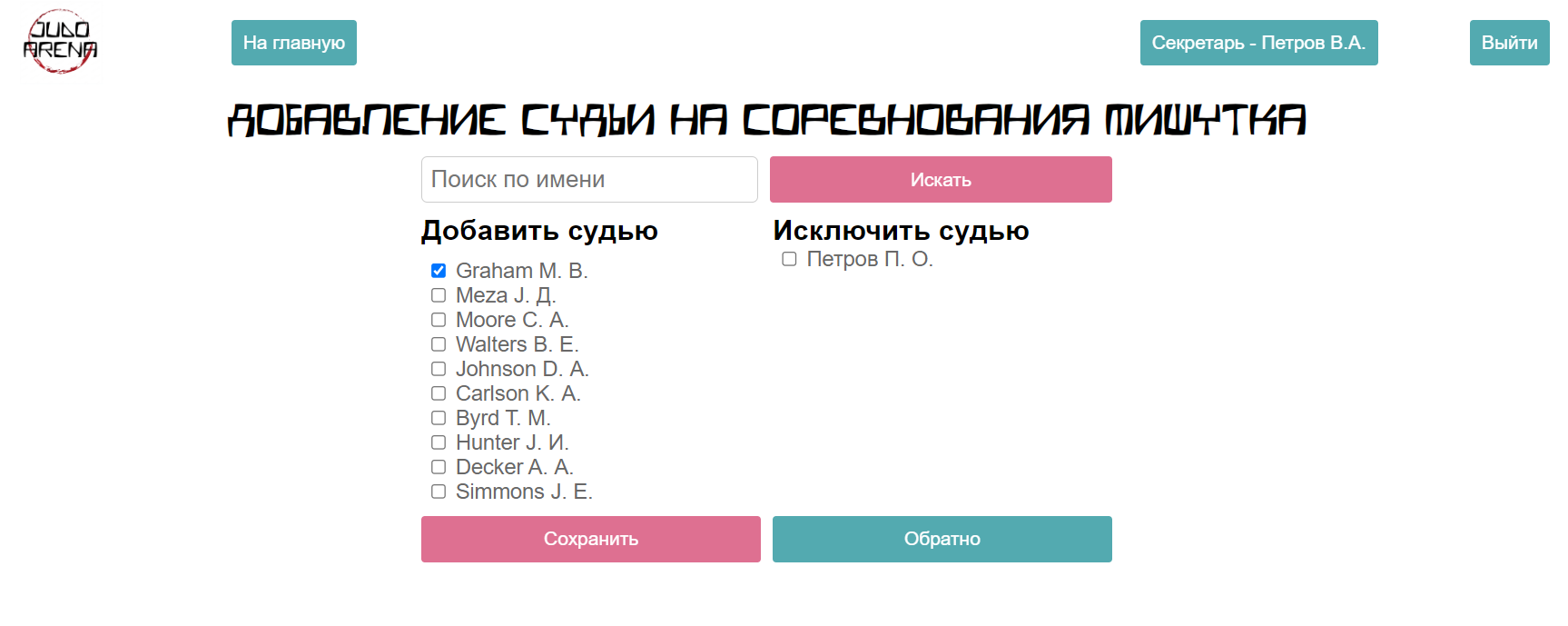


Рисунок 2.2.7 – Добавление судьи на соревнования

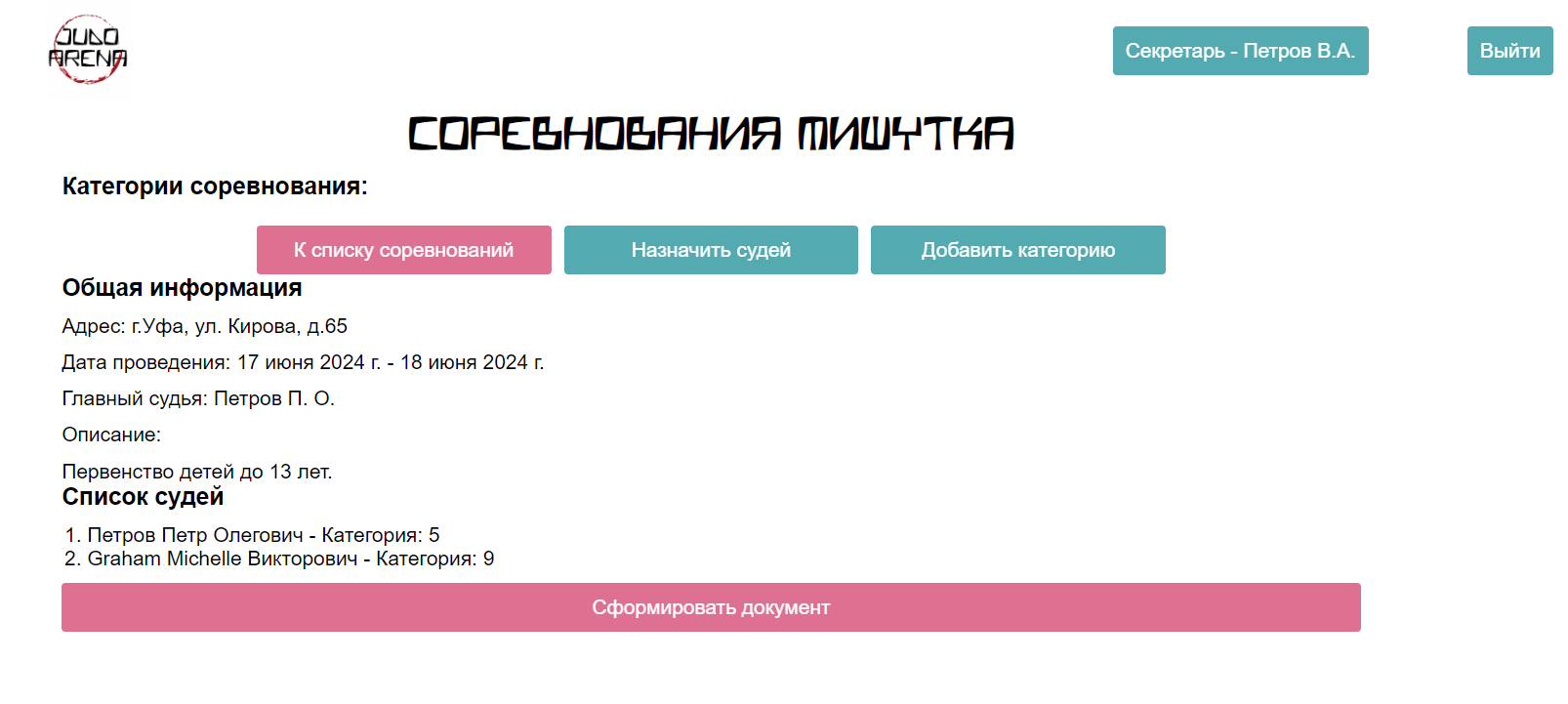


Рисунок 2.2.8 – Отображении соревнования с новым судьей

Таблица 2.2.5 – Тестирование регистрации тренера на некорректных данных

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 17.06.2024 |
| Приоритет тестирования | Высокий |
| Название теста | Регистрация тренера на некорректных данных |
| Резюме испытания | Каретная робота, безе аварийного завершения |
| Этапы тест | Запуск приложения  Переход на регистрацию  Запыления данных  Регистрация |
| Тестовые данные | Логин: user1  Пароль: password1  Фамилия: Кузнецов  Имя: Олег  Отчество: Индирович  Номер лицензии: 1234567890 |
| Ожидаемый результат | Вывод сообщения о том, что пользователь с таким логином существую и что пароль слишком распространённый. |
| Фактический результат | Вывод сообщения о том, что пользователь с таким логином существую и что пароль слишком распространённый. |

Результат тестирования на регистрации тренера представлен на рисунках 2.2.9 – 2.2.10.

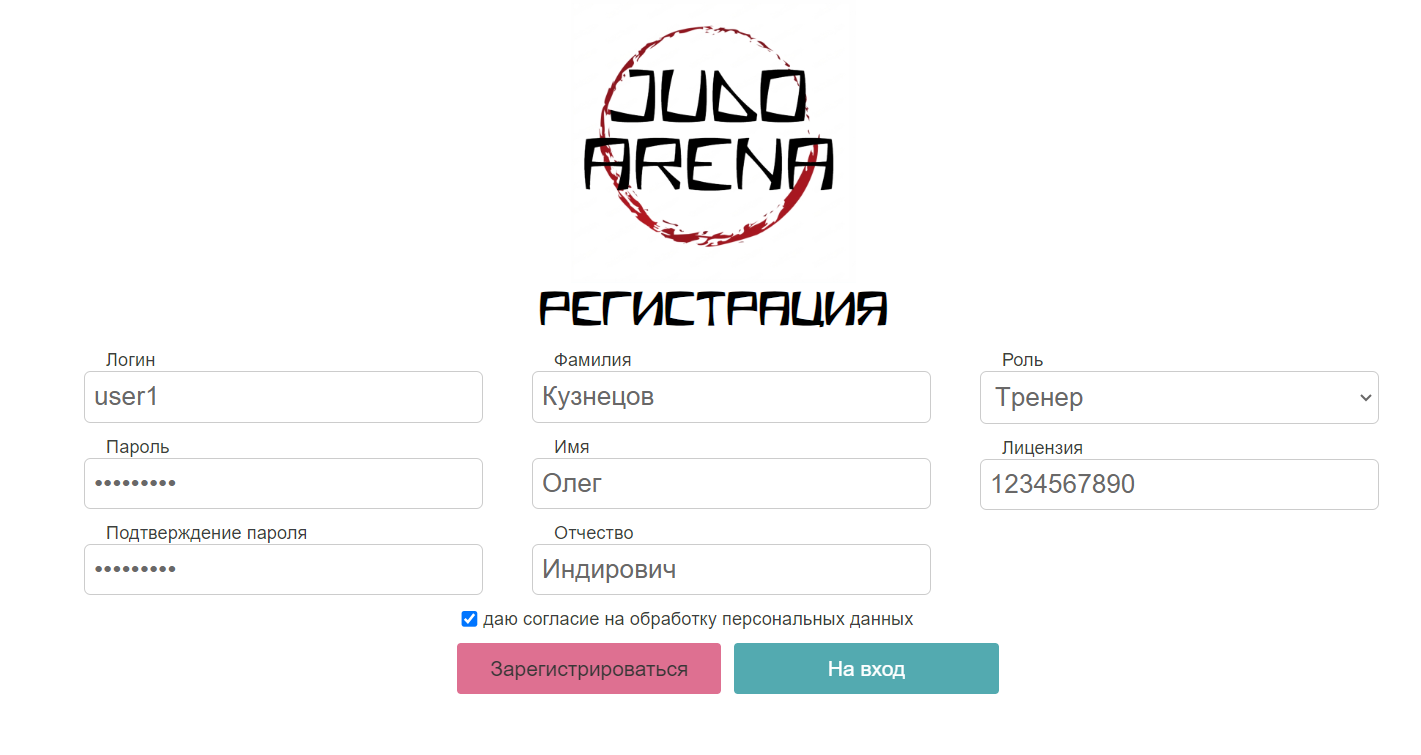


Рисунок 2.2.9 –Заполнения данных

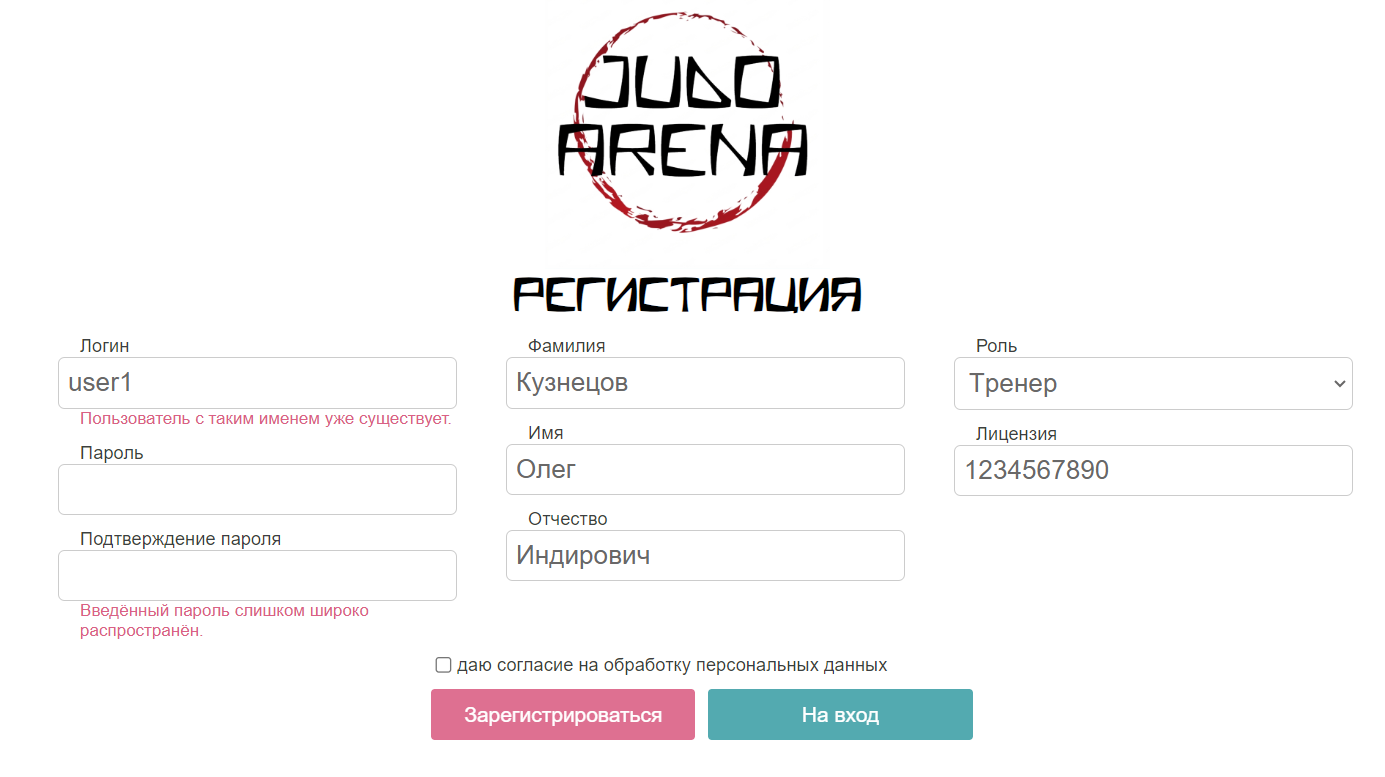


Рисунок 2.2.10 –Отображение ошибки регистрации

Таблица 2.2.5 – Тестирование добавление категории дважды

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |

Продолжение таблицы 2.2.2

|  |  |
| --- | --- |
| 1 | 2 |
| Дата теста | 17.06.2024 |
| Приоритет тестирования | Высокий |
| Название теста | Добавление категории дважды |
| Резюме испытания | Каретная робота, безе аварийного завершения |
| Этапы тест | Переход с главной страницы на соревнования  Переход на страницу добавления категорий  Создание возраста  Создание веса  Добывание категории на соревнование  Повторное добавление категории |
| Тестовые данные | Логин: user1  Пароль: password1  Фамилия: Кузнецов  Имя: Олег  Отчество: Индирович  Номер лицензии: 1234567890 |
| Ожидаемый результат | Первый раз: успешно добавлена категория  Во второй раз: Эта категория уже добавлена |
| Фактический результат | Вывод сообщения о том, что пользователь с таким логином существую и что пароль слишком распространённый. |

Результат тестирования на регистрации тренера представлен на рисунках 2.2.11 – 2.2.16.

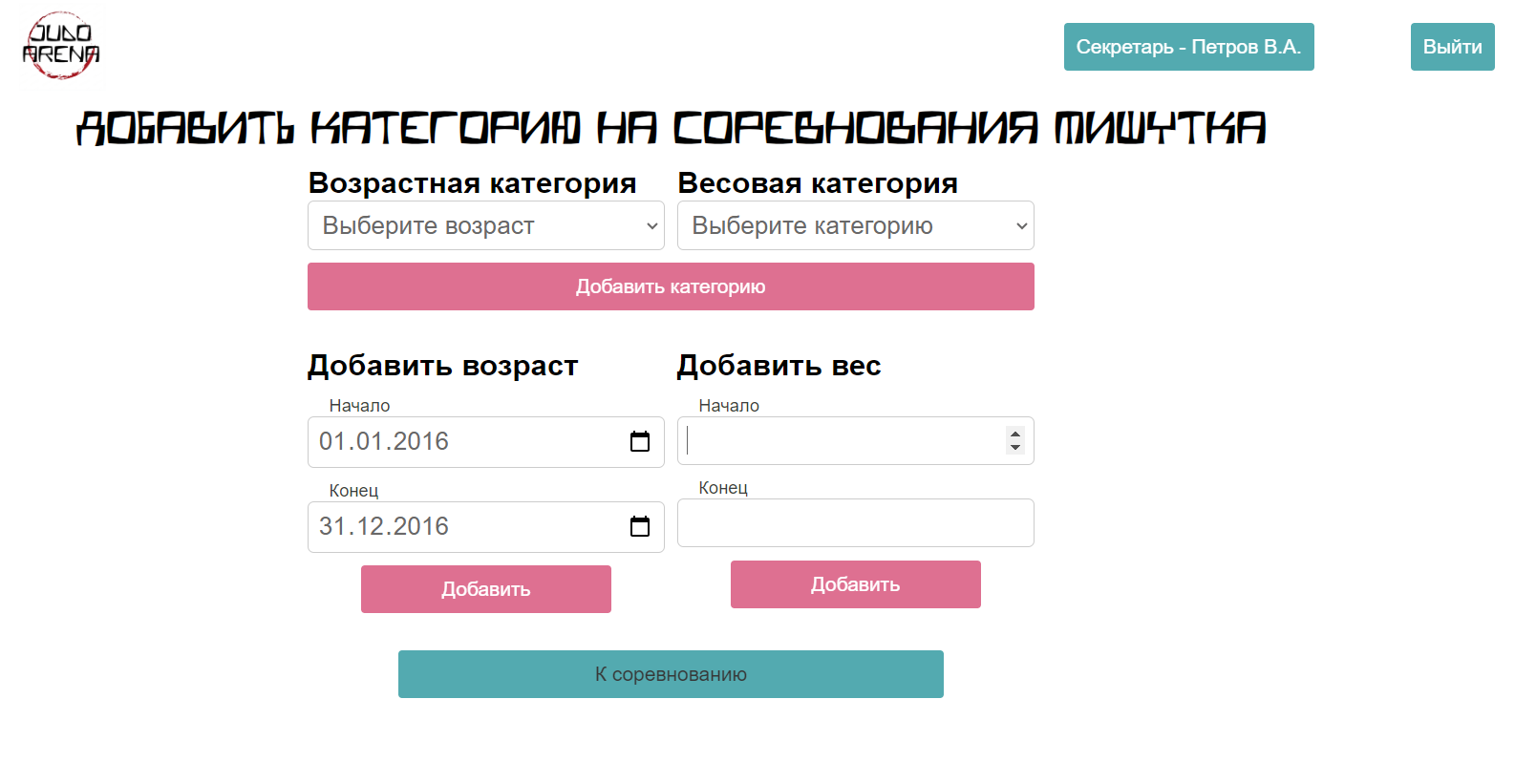


Рисунок 2.2.11 –Добавление возраста

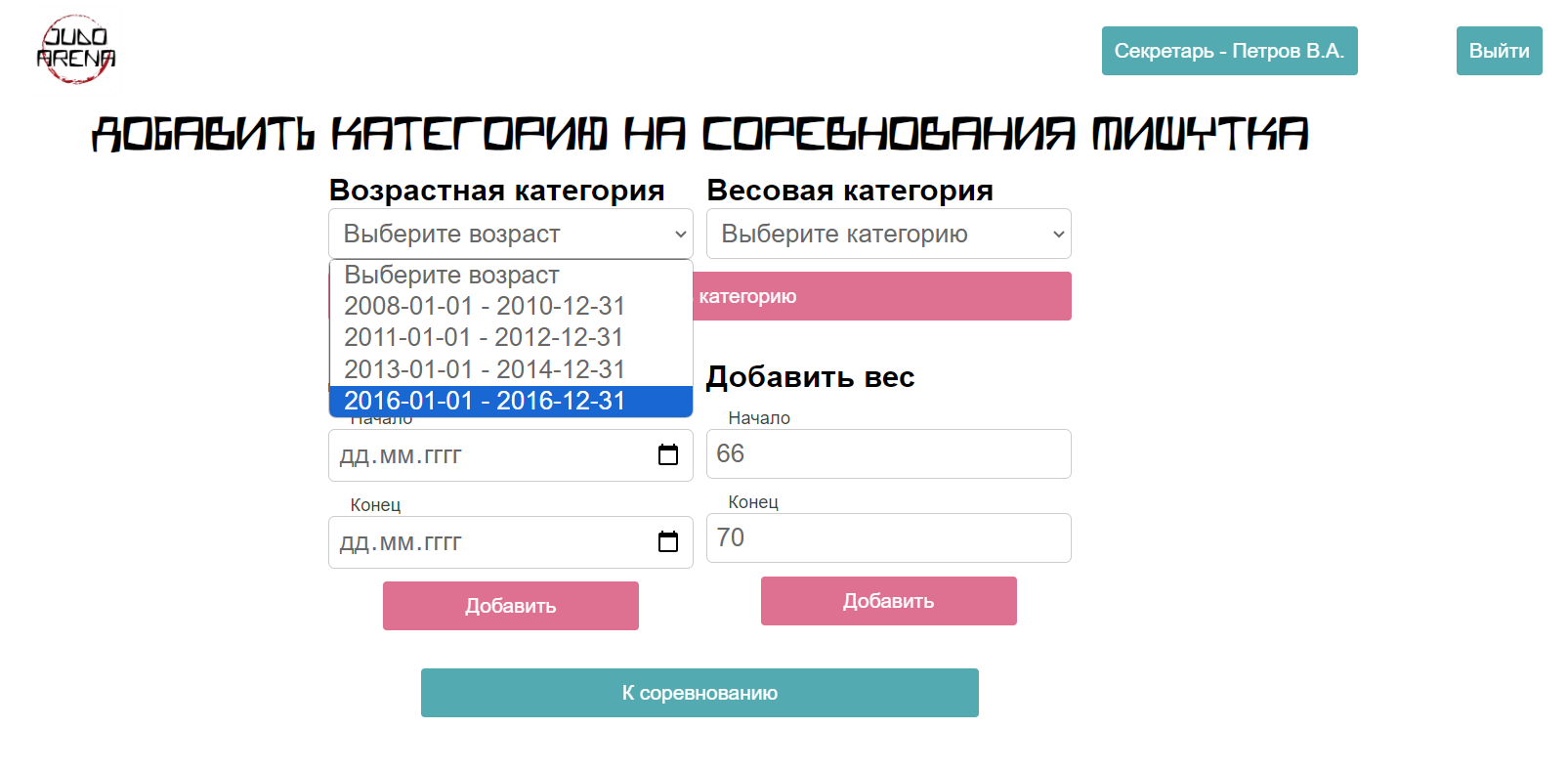


Рисунок 2.2.12 – Отображение добавление возраста

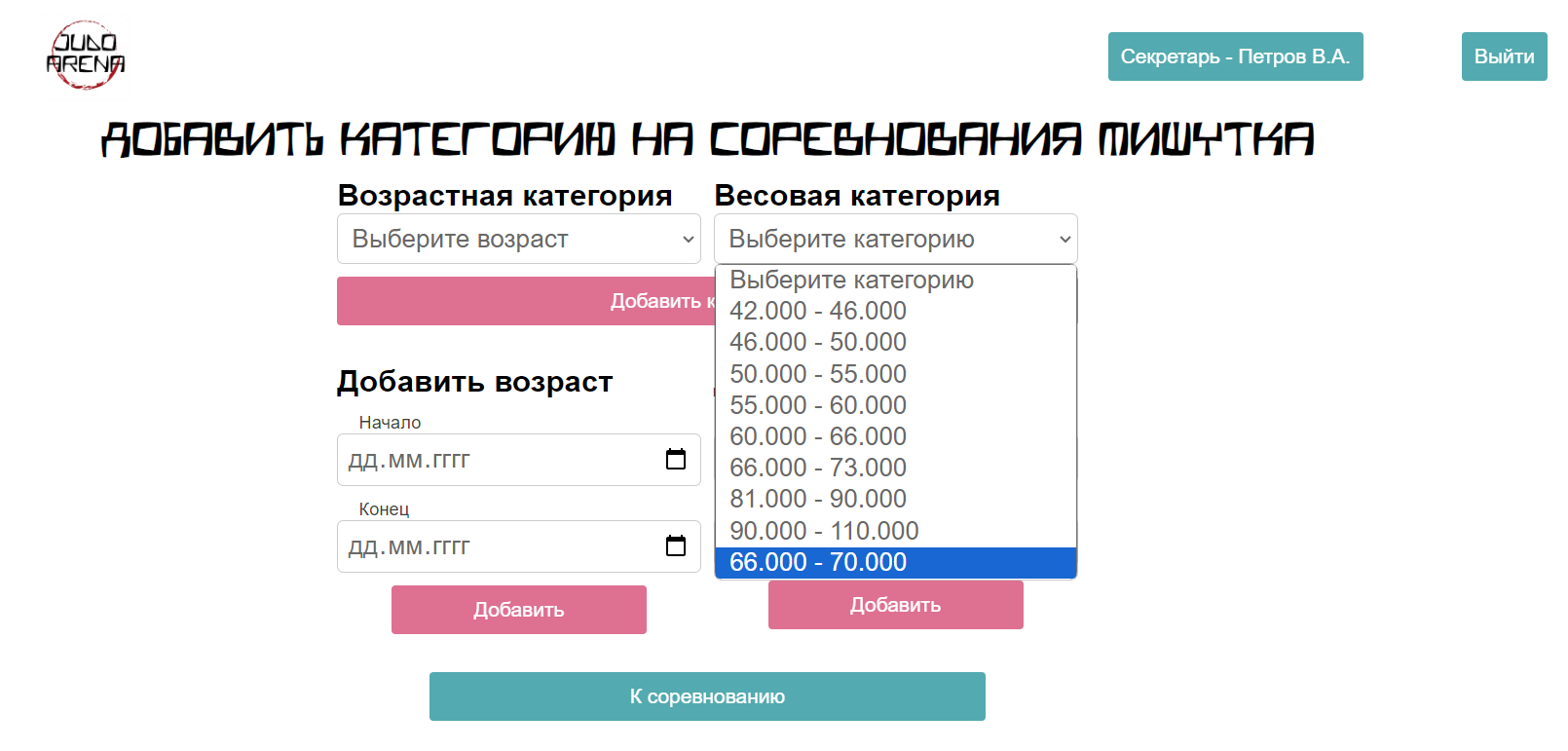


Рисунок 2.2.13 – Отображение добавление веса

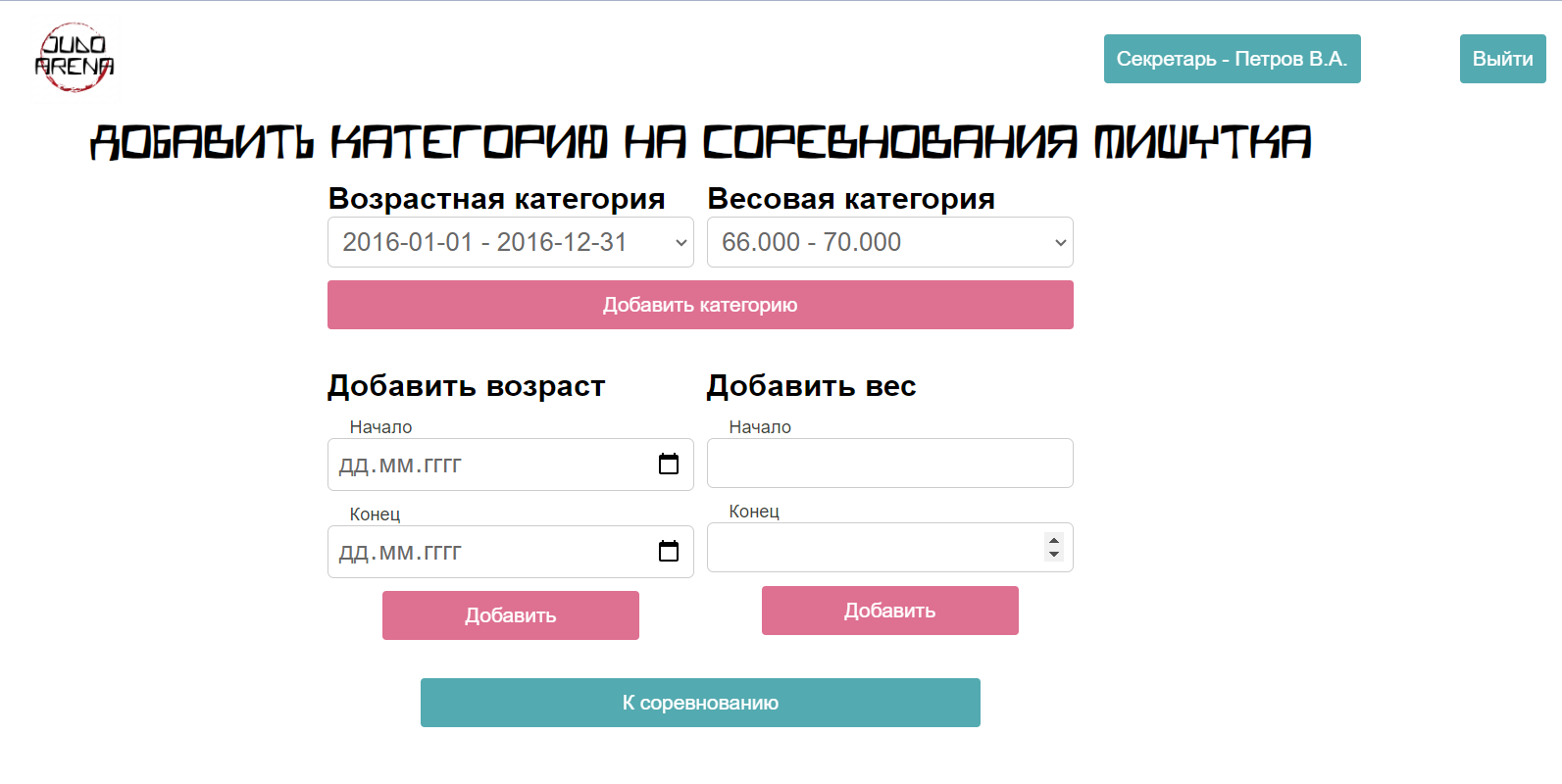


Рисунок 2.2.14 – Запыление данными для добавления категории

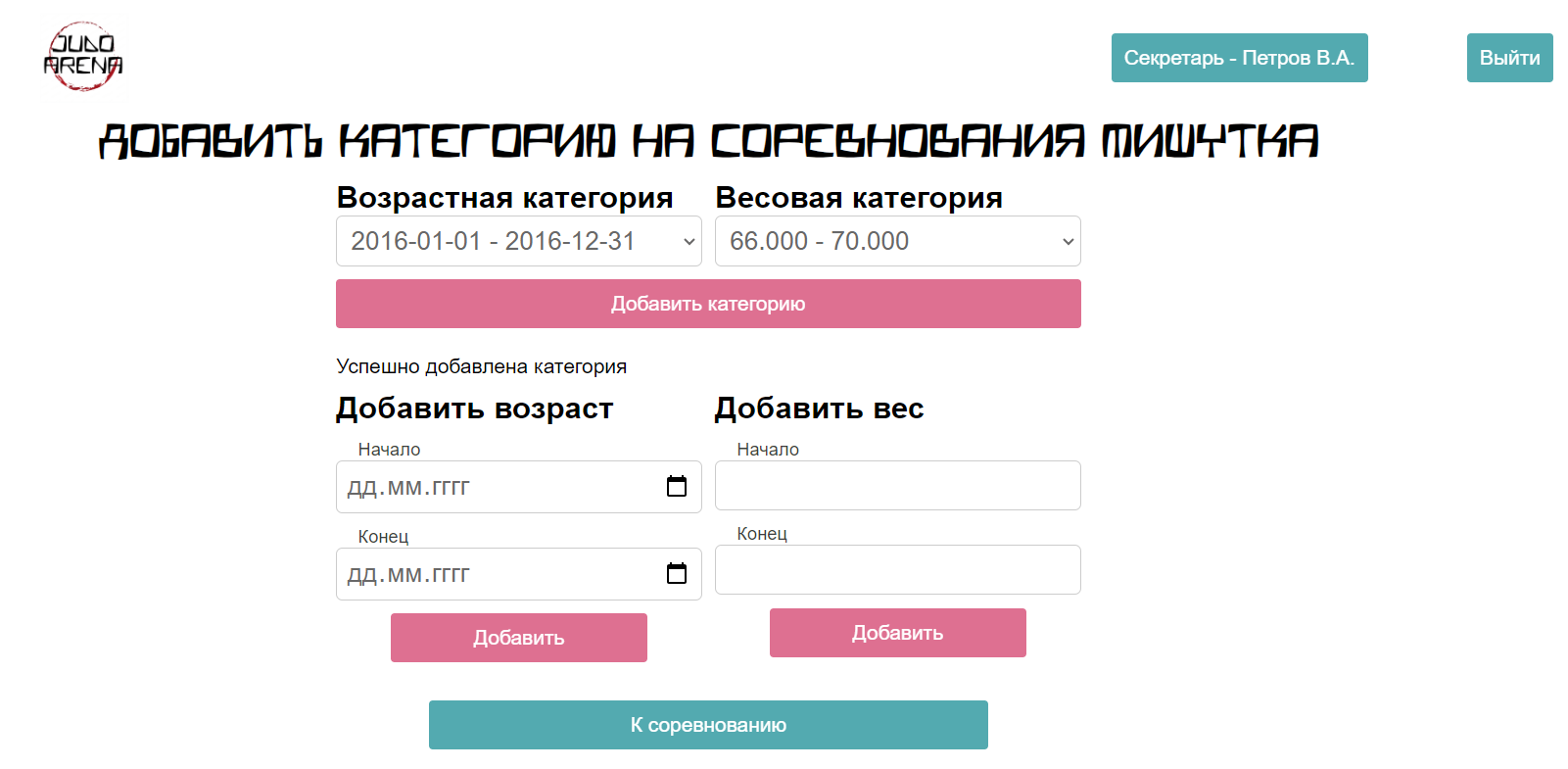


Рисунок 2.2.15 –Успешное добавление категории

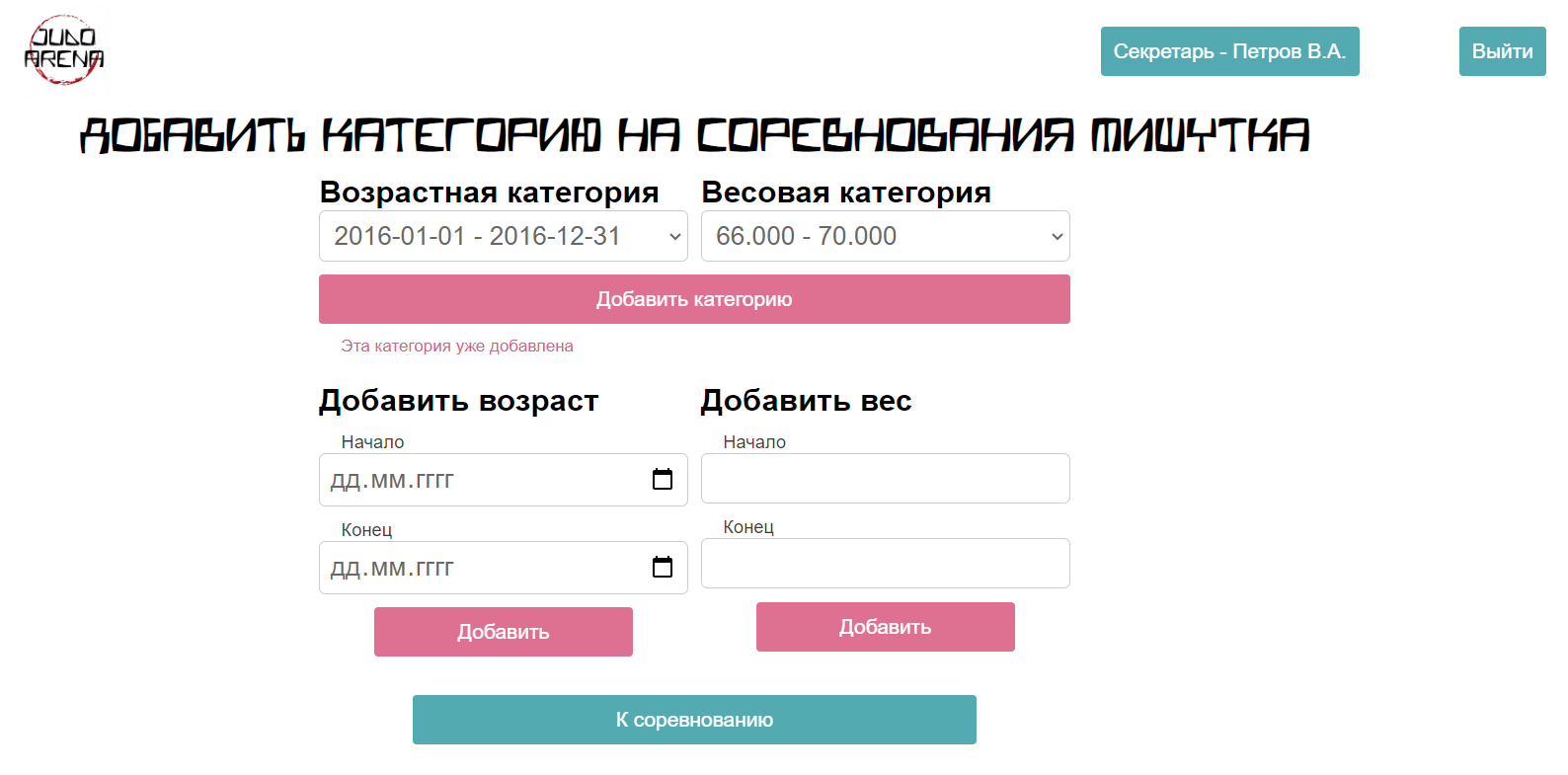


Рисунок 2.2.16 –Не успешное добавление категории

## 2.3 Руководство пользователя

Назначение программы.

Разработанная система предназначена для автоматизации процесса документирования и обработки информации на соревнованиях по дзюдо. Она обеспечивает электронную форму для заполнения данных об спортсменах и их результатах, а также хранение и обработку данных в электронном виде.

Цель системы состоит в упрощении и ускорении процесса взаимодействия судей и спортсменов с документацией, связанной с соревнованиями по дзюдо. Система заменяет ручное заполнение бумажных документов, которое требует значительного времени и усилий. Вместо этого, пользователи могут эффективно заполнять данные об спортсменах и их результатах с использованием электронной формы.

Условия применения системы.

Программное обеспечение разрабатывается для персональной вычислительной техники со следующими характеристиками:

* Microsoft Windows 10;
* процессор 1 ГГц;
* 128 МБ ОЗУ;
* 60 МБ свободного пространства на диске;
* разрешение экрана монитора не менее 1920 × 1080;
* дополнительное оборудование
* принтер.

Программа «Дзюдо арена» предназначена для пользователей, имеющих как минимум первоначальные навыки работы с графической операционной системой, на которой будет запускаться данная программа.

Выполнение программы.

Для запуска программы необходимо запустить приложение JudoArena и прейти ссылки из терминала. После этого открывается окно авторизации для входа в программу (рисунок 2.3.1).

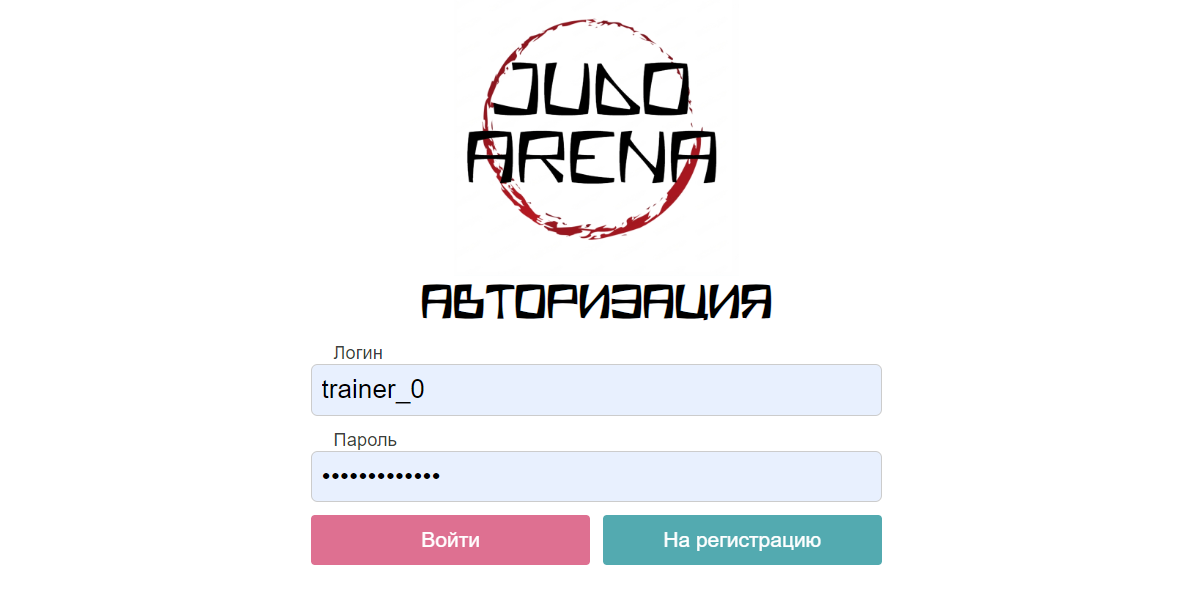


Рисунок 2.3.1– Страница авторизации

Есть возможно прейти на окно регистрации (рисунок 2.3.2 и русинок 2.3.3) тренера и спортсмена.

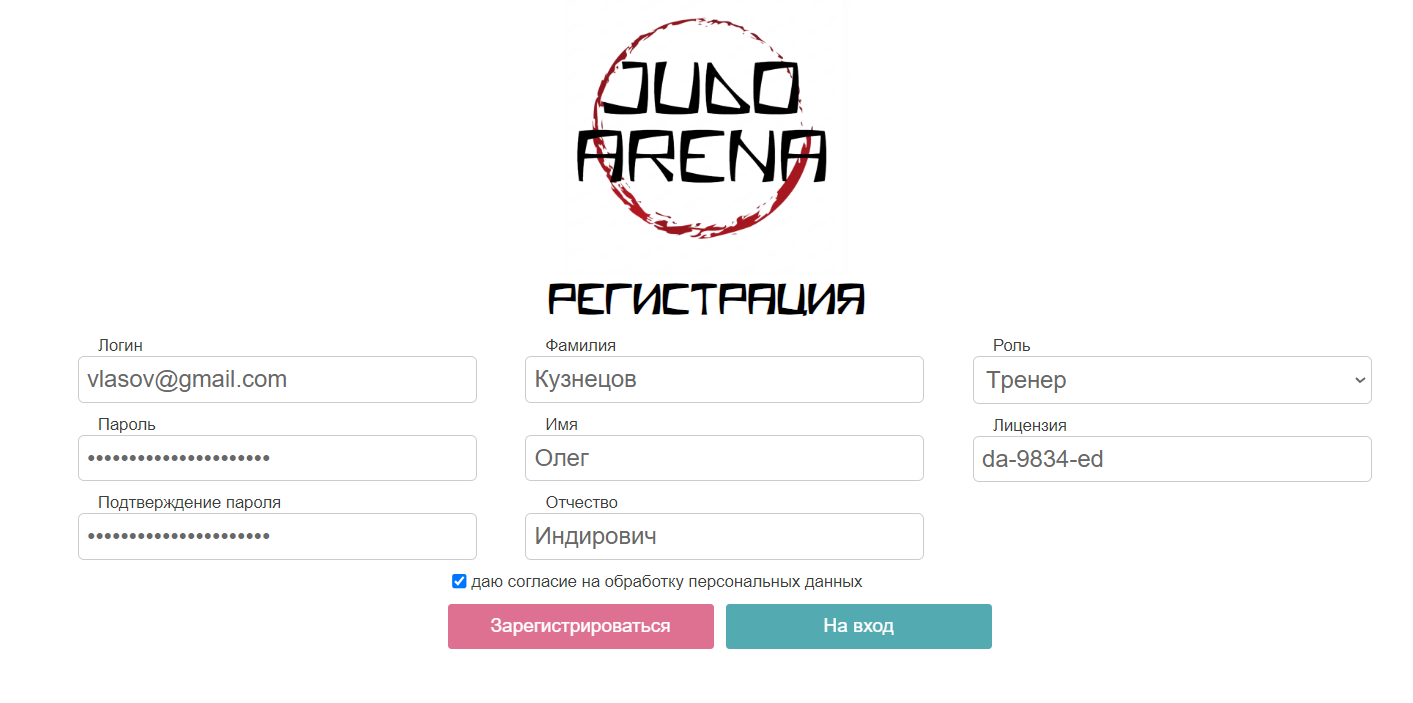


Рисунок 2.3.2– Страница регистрации тренера

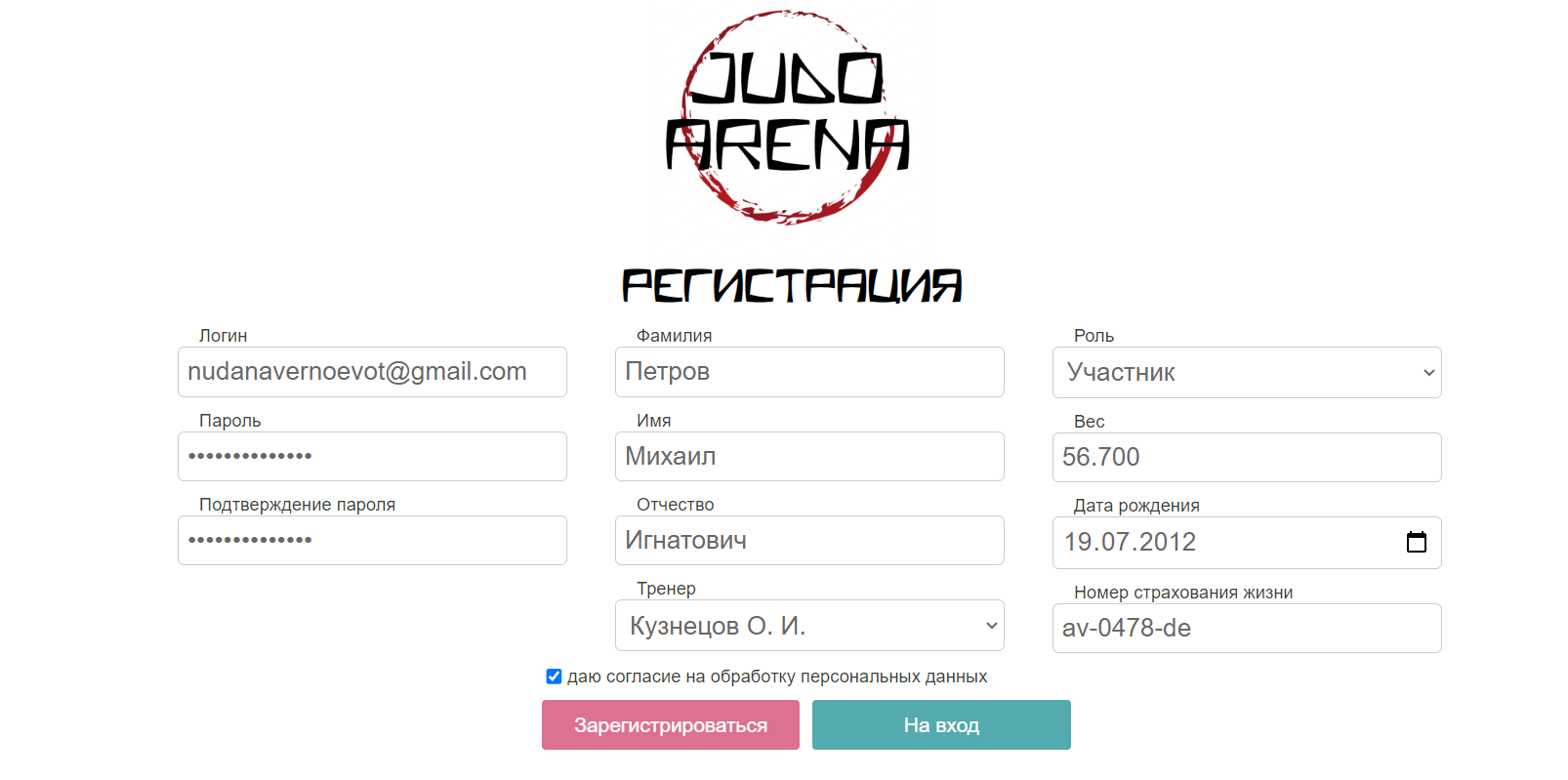


Рисунок 2.3.3– Окно регистрации спортсмена

Для получения доступа основным функциям нужно авторизоваться. После успешной авторизации переходит на страницу списка соревнований (рисунок 2.3.4).

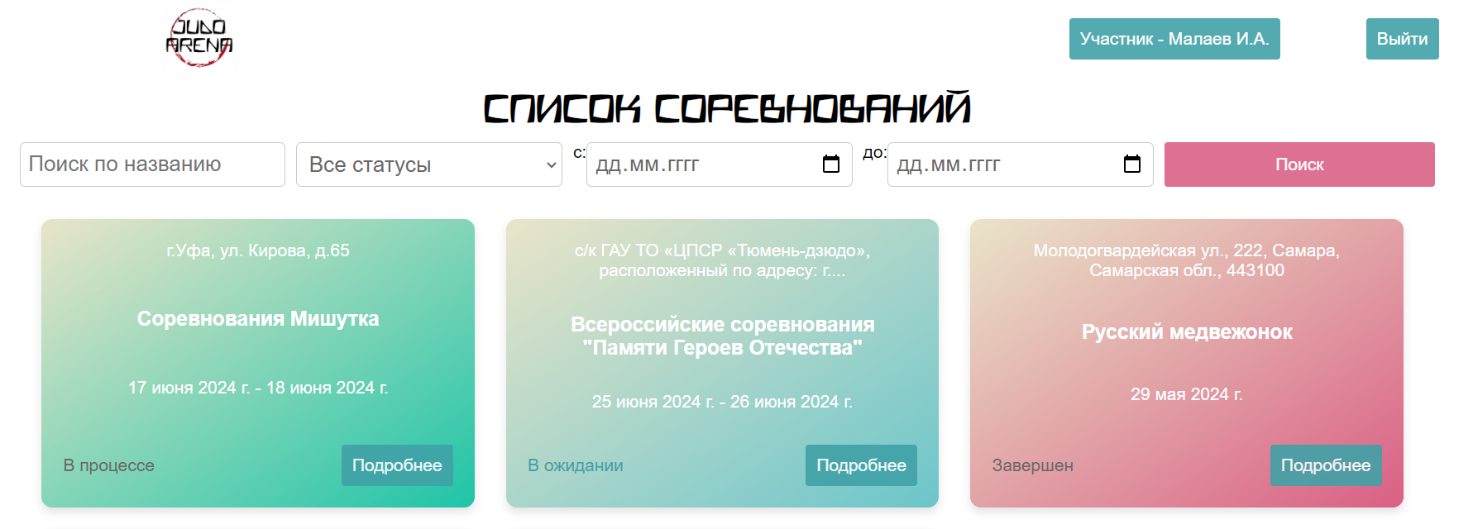


Рисунок 2.3.4– Список соревнований

Есть поиск по наименованию, сортировка по дате, фильтр по дате, возможность выйти на страницу авторизации. Кнопка подробнее переведет на страницу соревнования (рисунок 2.3.5 и рисунок 2.3.6).



Рисунок 2.3.5– Страницу соревнования начало

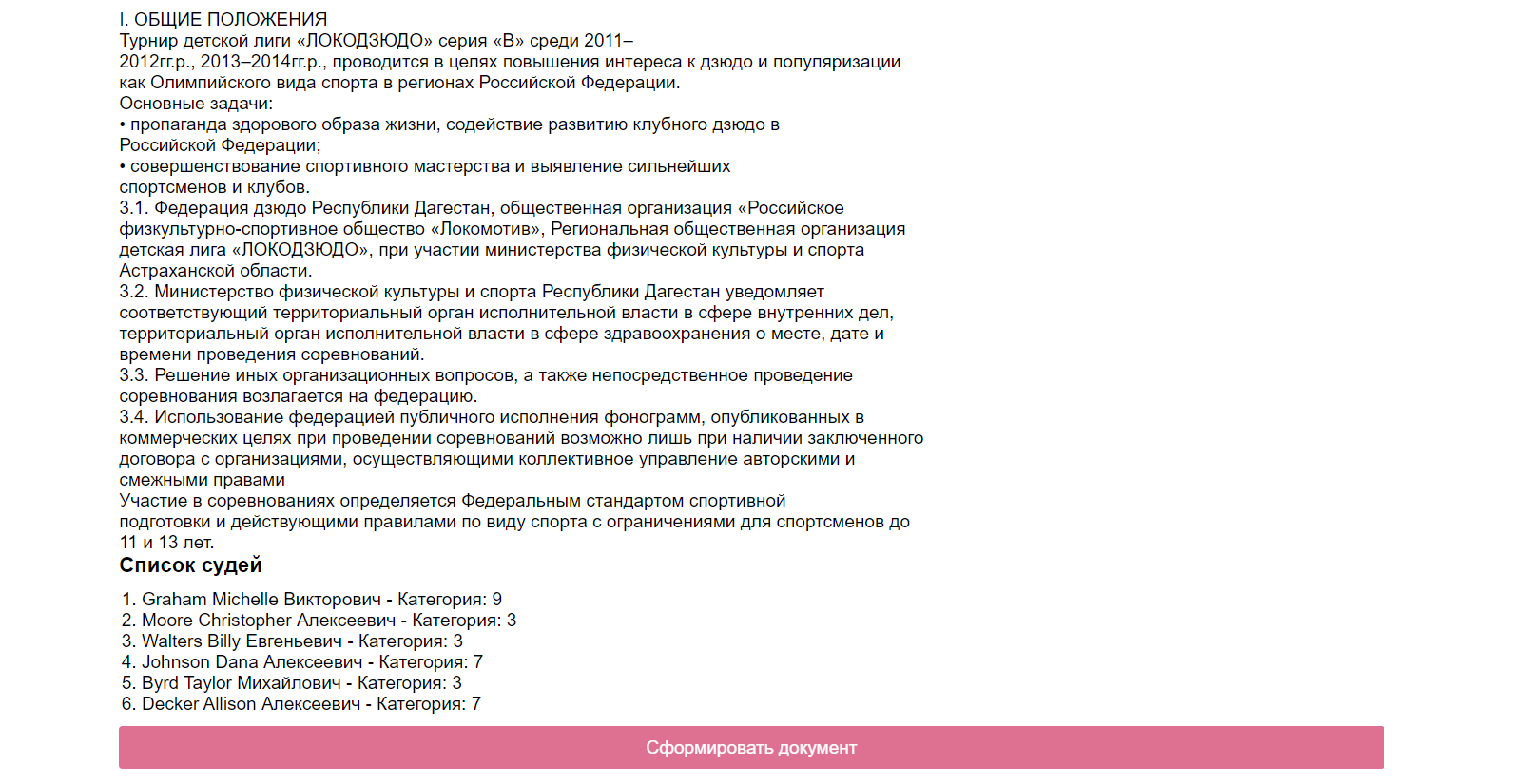


Рисунок 2.3.6– Страницу соревнования конец

Так же кнопка и ролью с ФИО на главной странице переведет на редактирование своего профиля. (рисунок 2.3.7 и рисунок 2.3.8).



Рисунок 2.3.7– Страницу редактирования профиля спортсмена

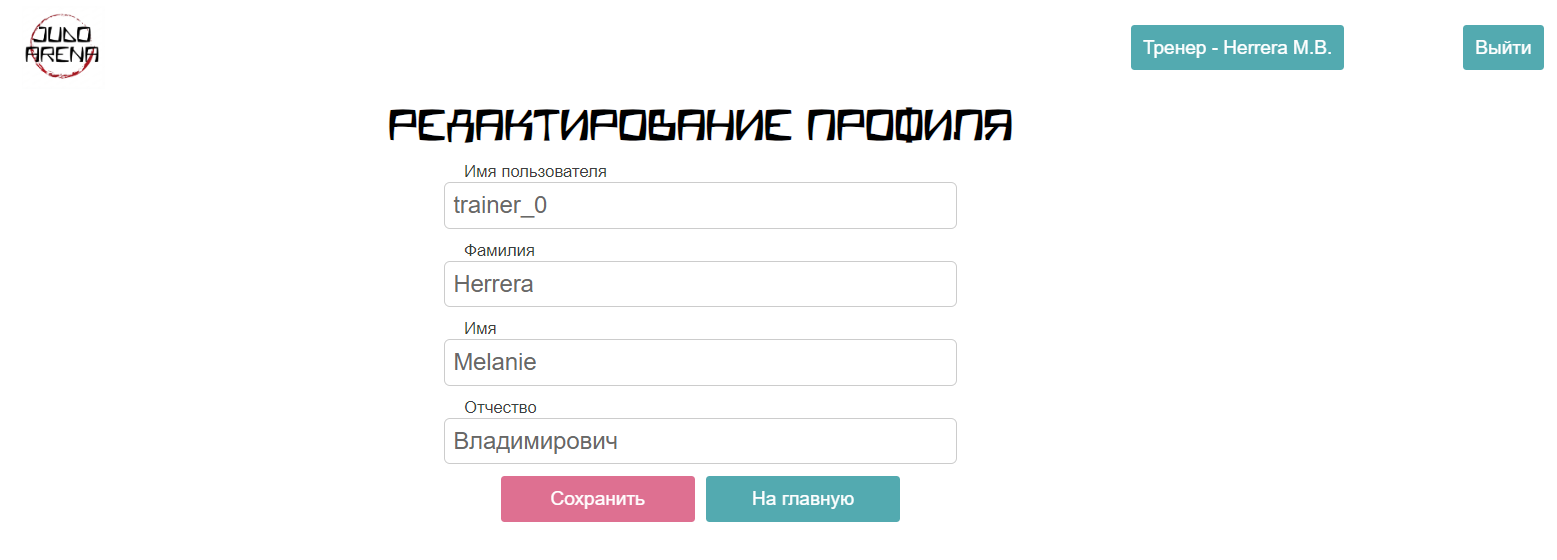


Рисунок 2.3.8– Страницу редактирования профиля тренера, судьи и секретаря

Внутри соревнования нажав на весовую категорию будет осуществлен переход на страницу с заявками на участие (рисунок 2.3.9).

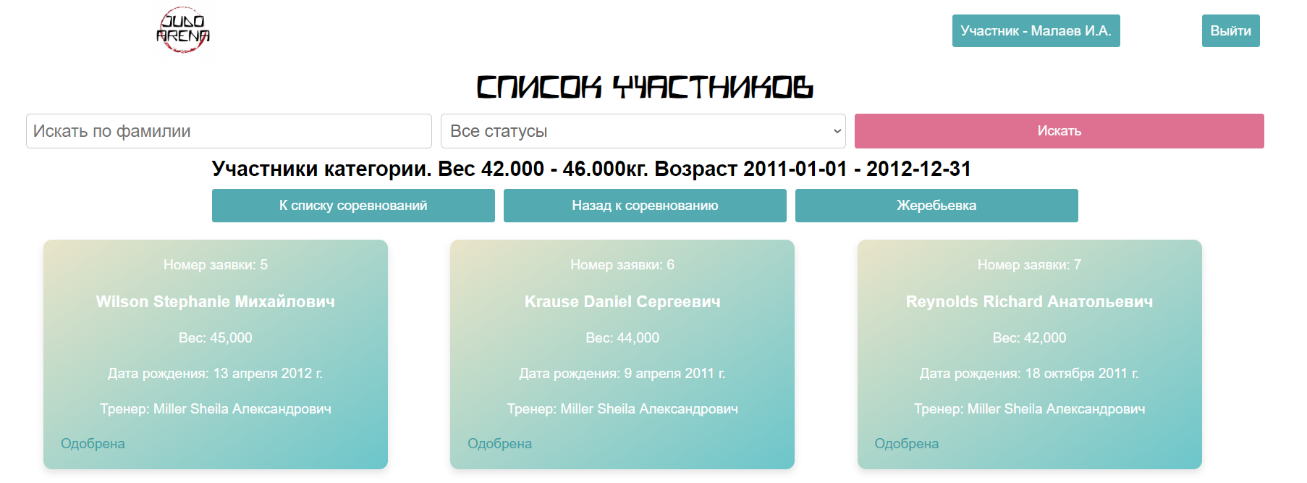


Рисунок 2.3.9– Страницу с заявками на участие

Нажав на кнопку жеребьевка будут показана турнирная сетка

(рисунок 2.3.10 и рисунок 2.3.11).

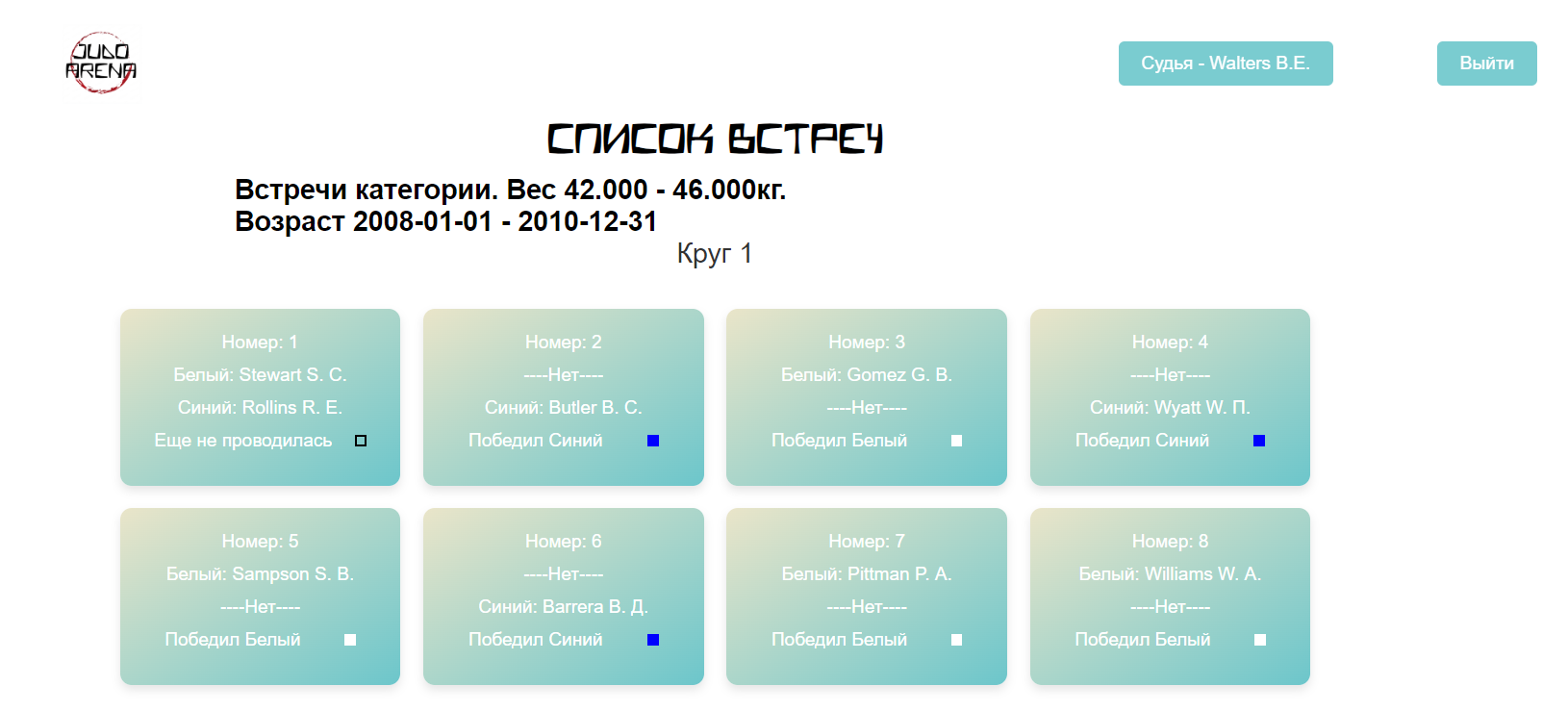


Рисунок 2.3.10 – Отображение турнирной сетки начало



Рисунок 2.3.11 – Отображение турнирной сетки конец

Секретарь

Секретарь может добавить согревание (рисунок 2.3.12 и рисунок 2.3.13).

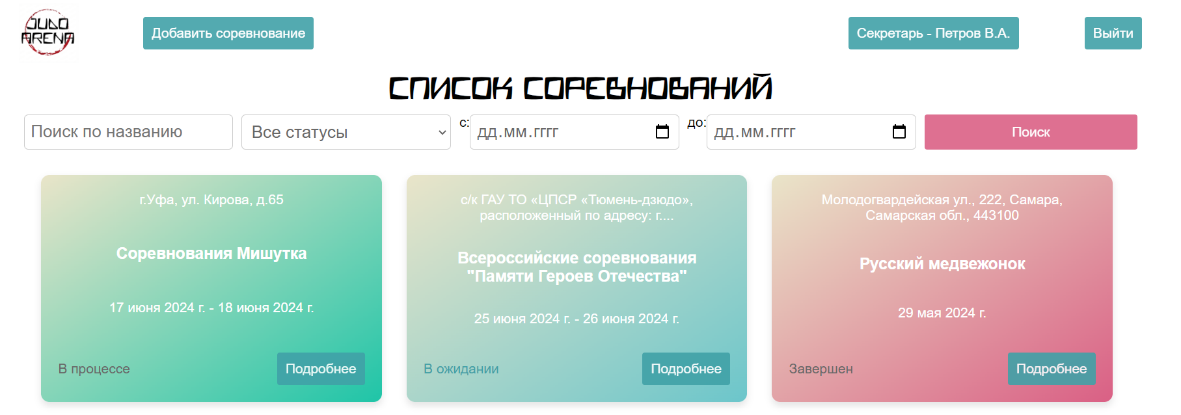


Рисунок 2.3.12 – Отображение кнопки для добавления соревнования

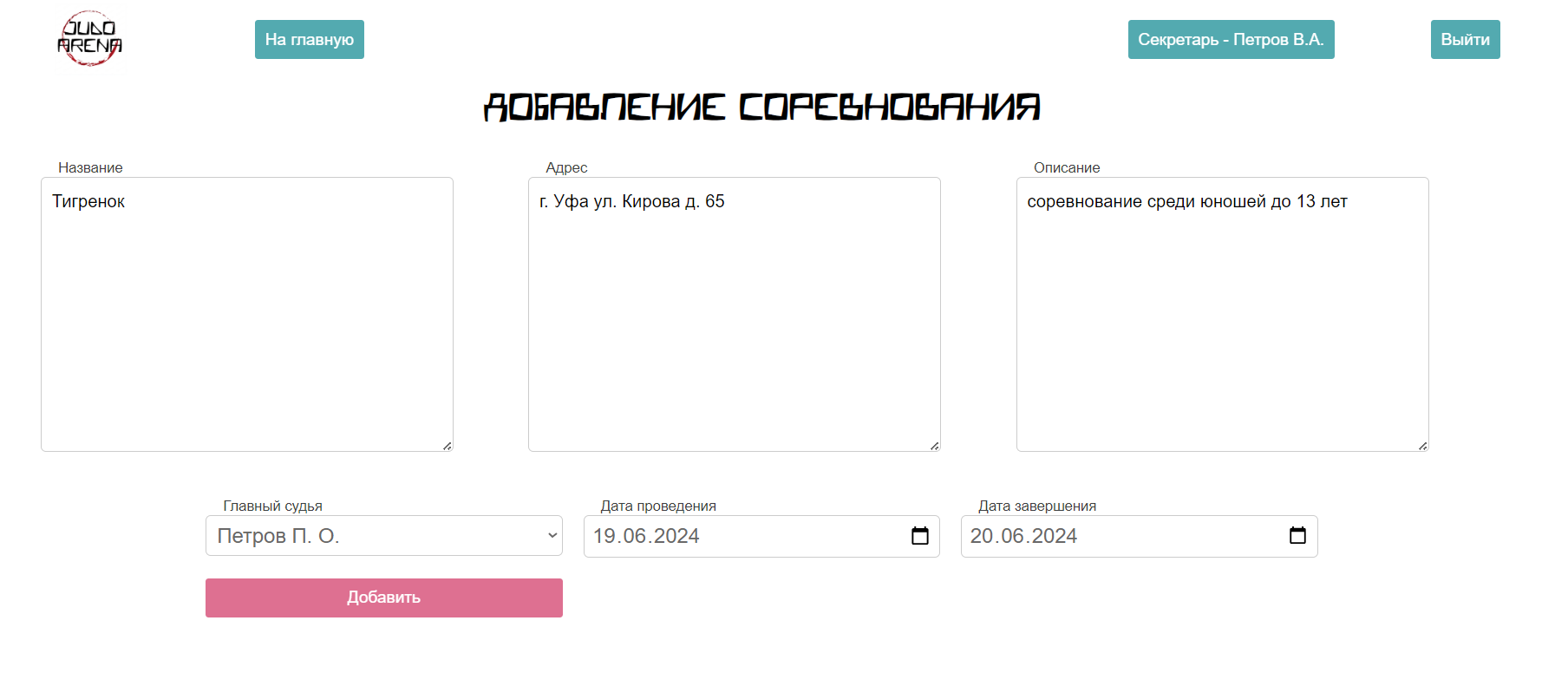


Рисунок 2.3.13 – Форма для добавления соревнования

После добавление соревнования отроиться форма для назначения судей этого согревания (рисунок 2.3.14).

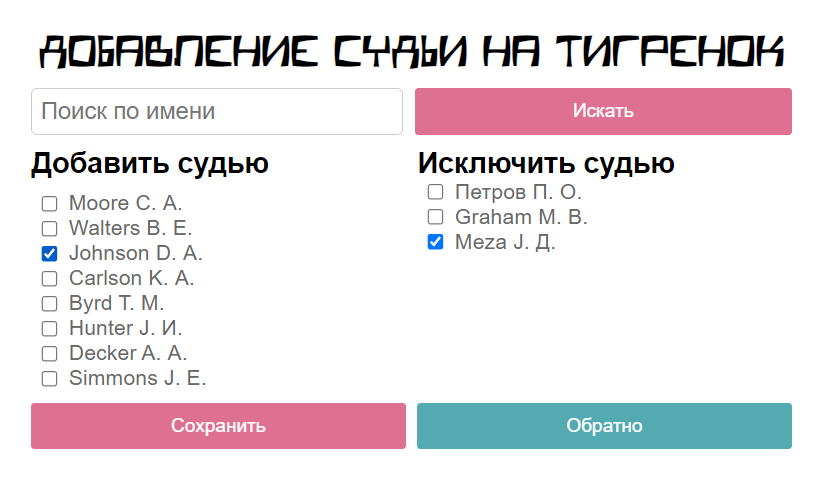


Рисунок 2.3.14 – Форма для добавления судей на соревнование

После назначения судей есть возможность добить весовые категории (рисунок 2.3.15).

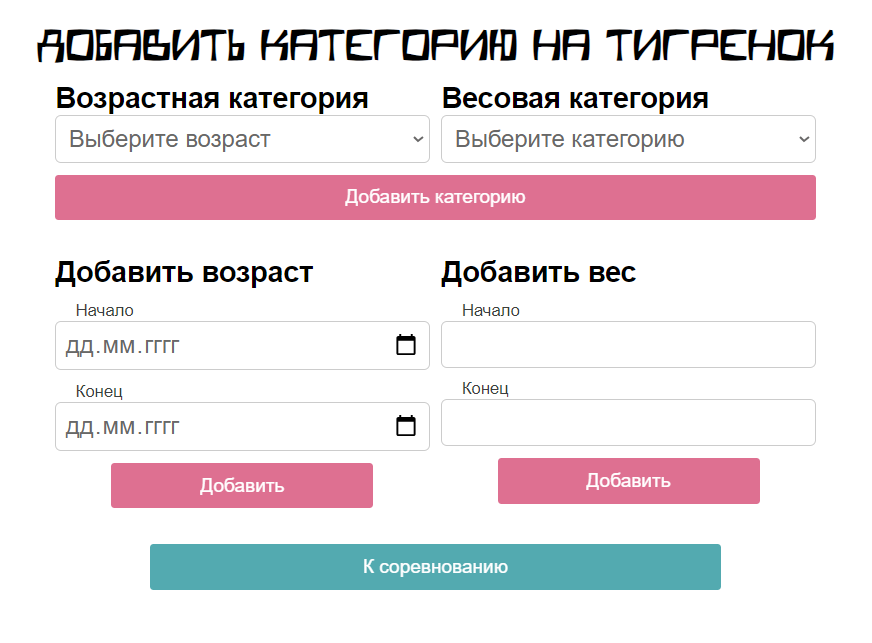


Рисунок 2.3.15 – Форма для добавления весовых категории на соревнование

На странице список участников, секретарь может отклонить или принять заявку (рисунок 2.3.16).

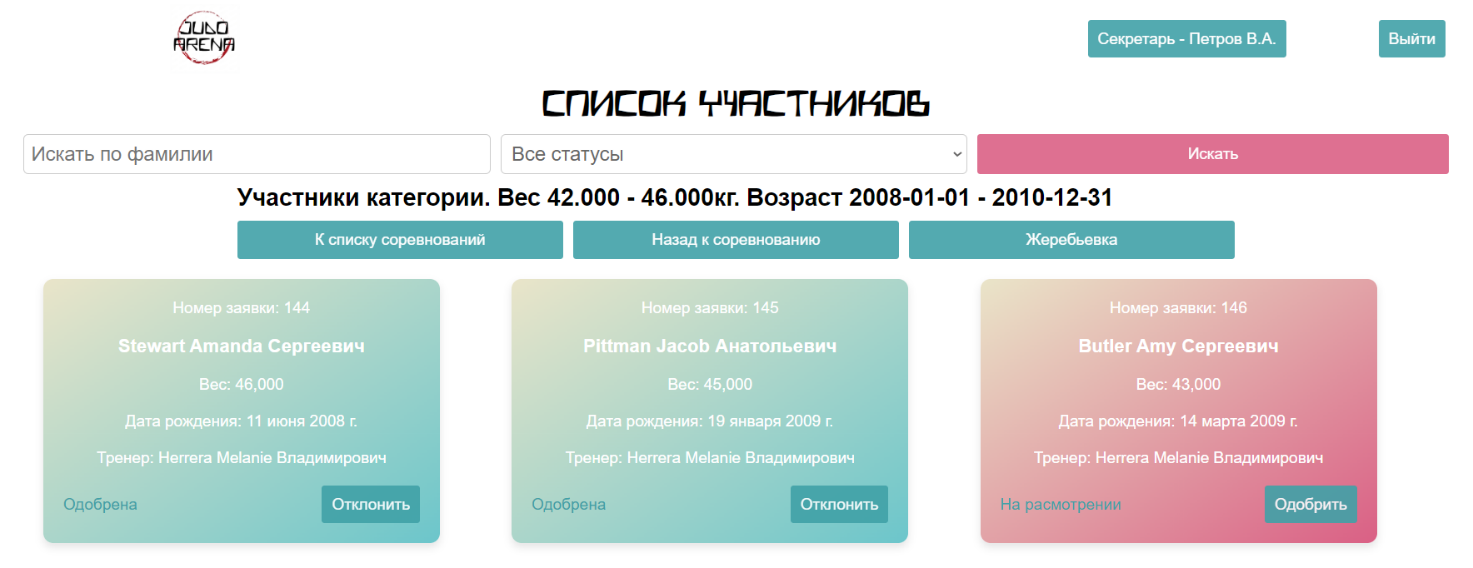


Рисунок 2.3.15 – Страница принятия зевок

На странице жеребьевки, секретарь может провести встречу (рисунок 2.3.17 и рисунок 2.3.18).

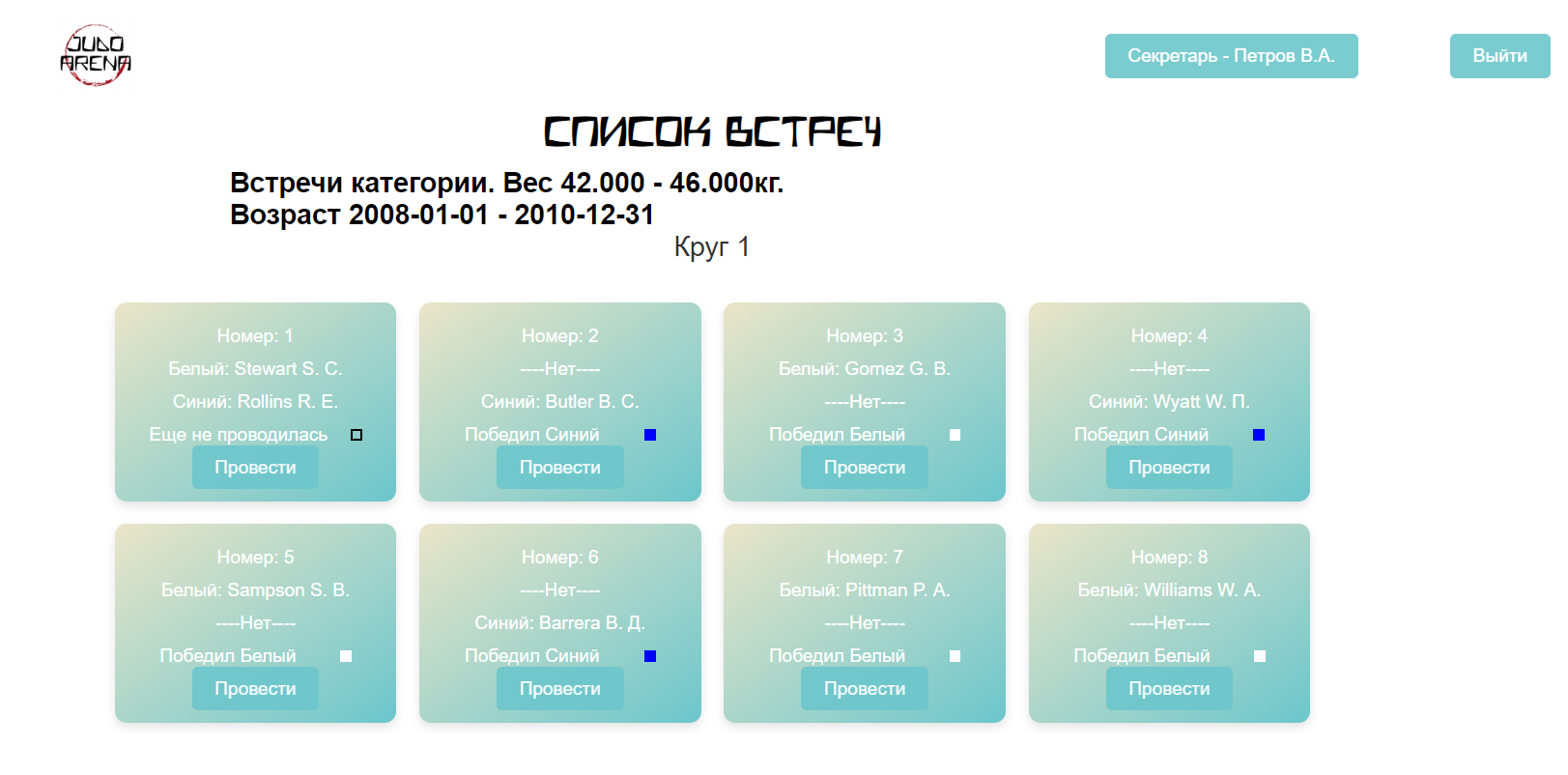


Рисунок 2.3.17 – Отображение встреч (Секретарь)

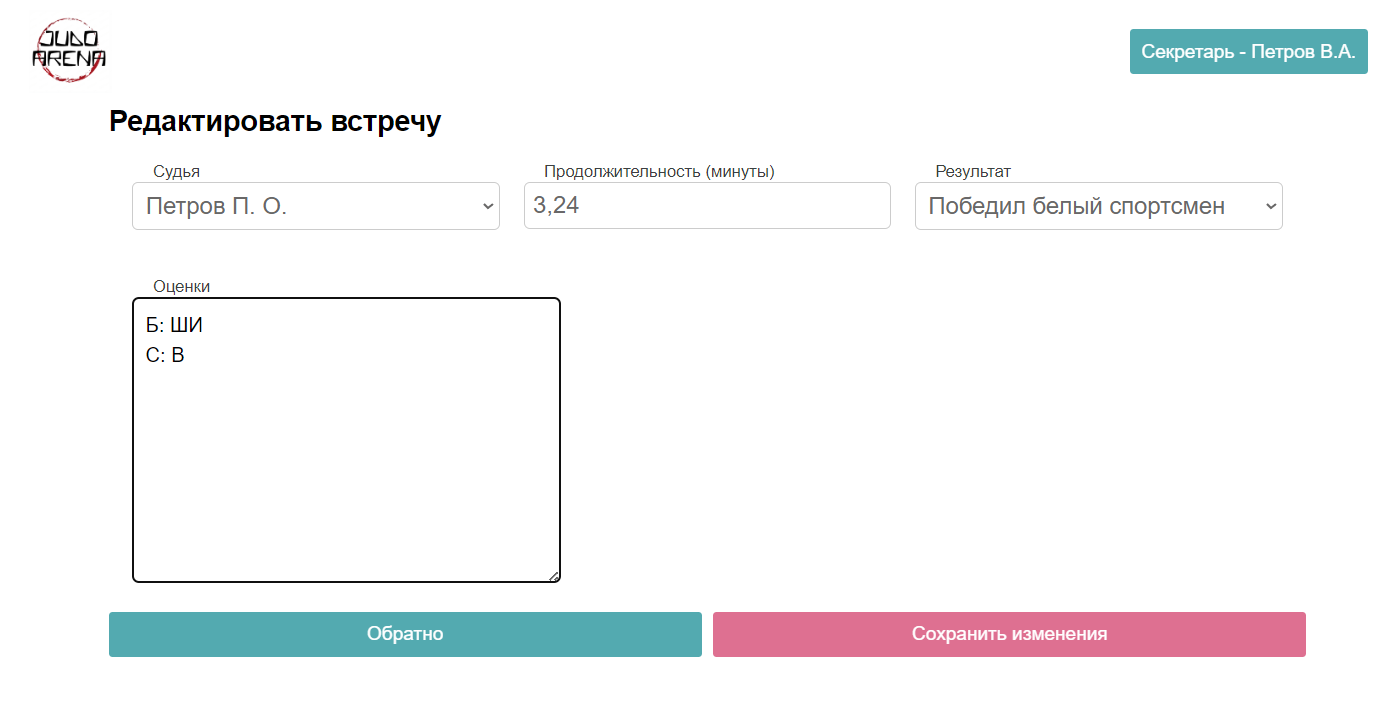


Рисунок 2.3.18 –Редактирование встречи

Тренер

На странице списка участников, тренер может подать завку (рисунок 2.3.19 и рисунок 2.3.20).

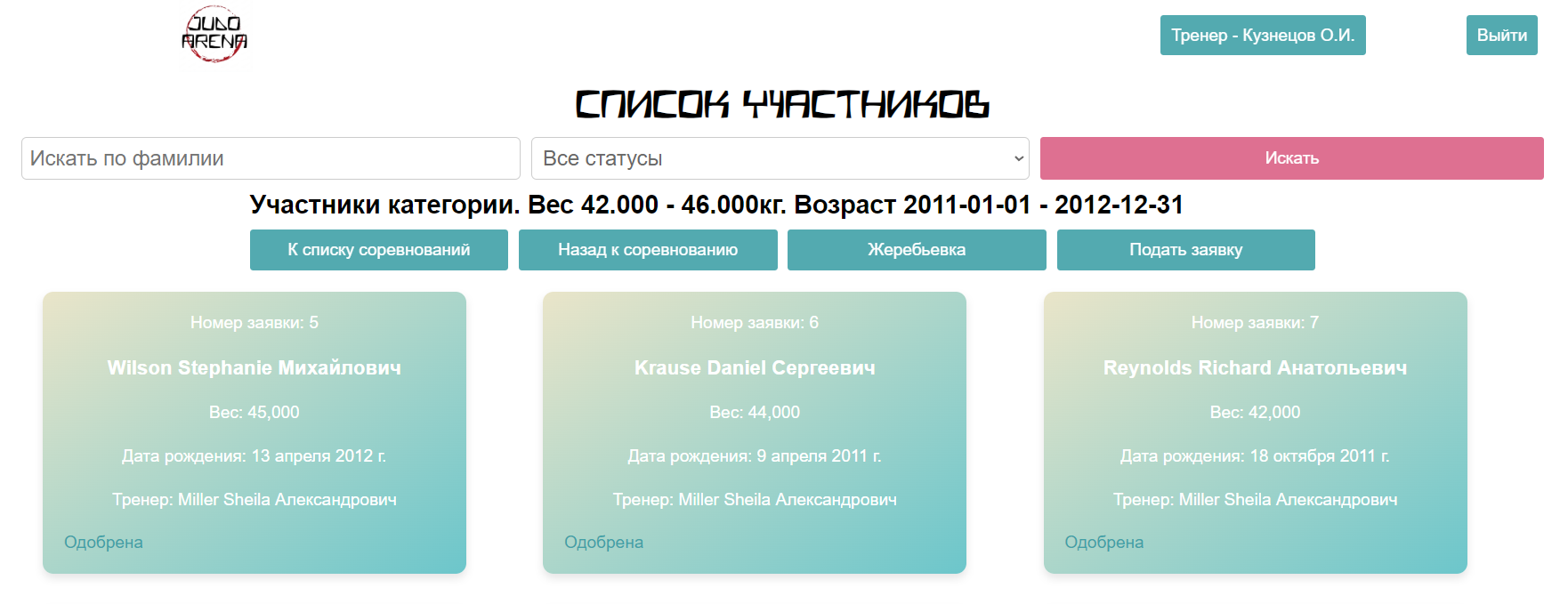


Рисунок 2.3.19 – Отображение кнопки для навигации



Рисунок 2.3.20 – Подача заявки

Судья

На странице списка встреч, судья может привести жеребьевку (рисунок 2.3.21 и рисунок 2.3.22).

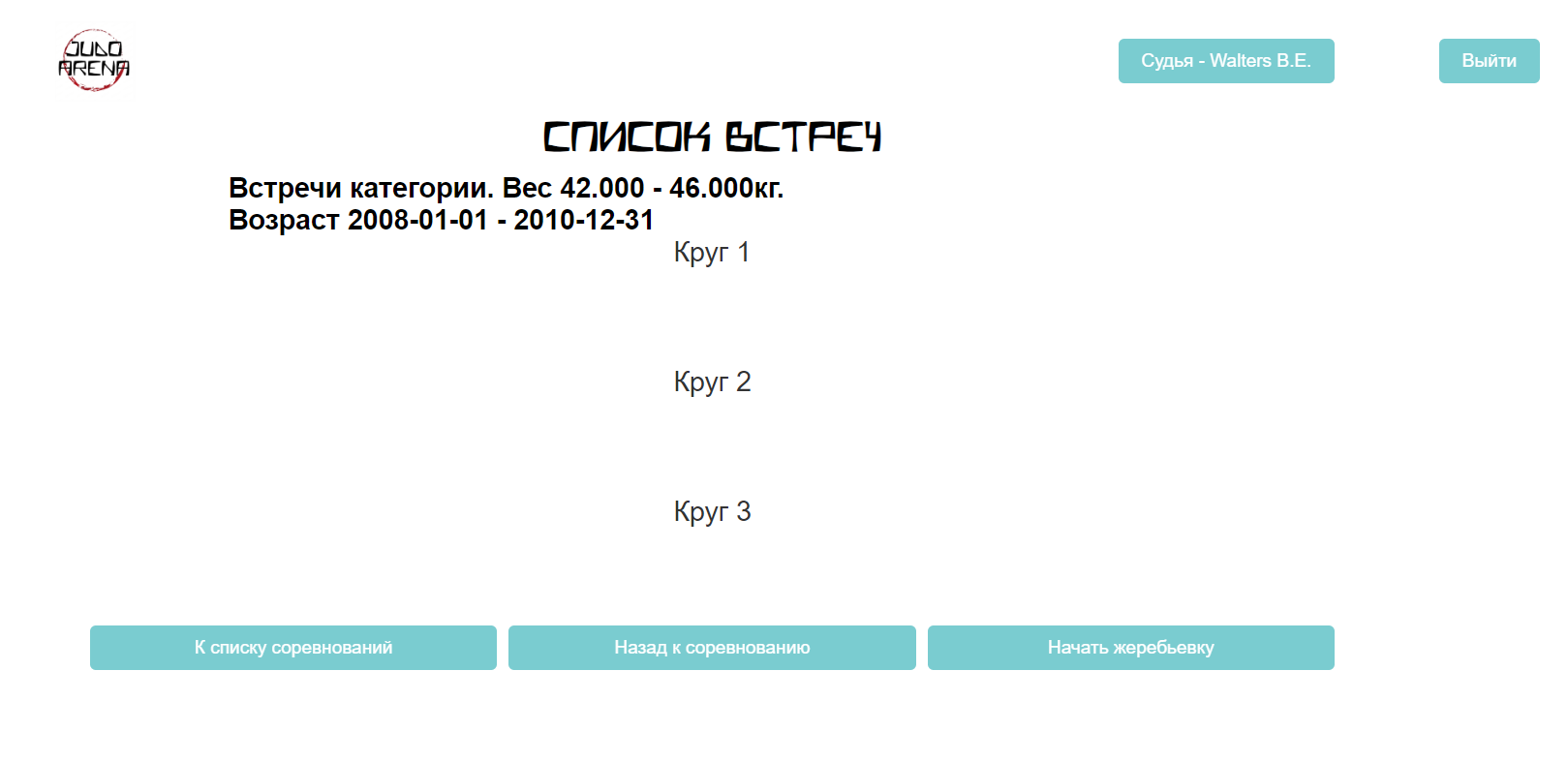


Рисунок 2.3.19 – Отображение не проведенной жеребьевки

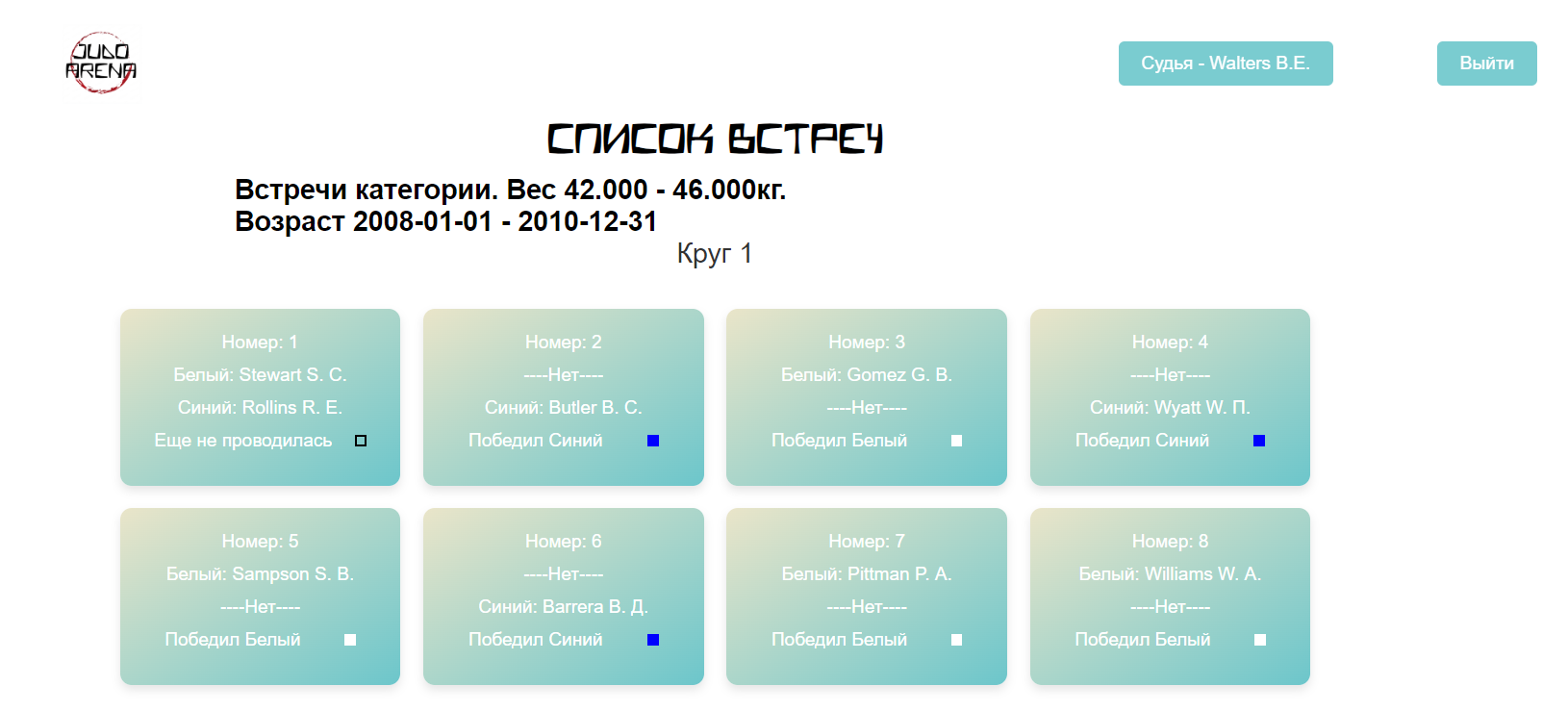


Рисунок 2.3.19 – Отображение проведенной жеребьевки

# ЗАКЛЮЧЕНИЕ

В ходе разработки дипломного проекта было создано приложение JudoArena на языке Python 3.11 с использованием фреймворка Django, в среде программирования PyCharm Community Edition 2024.1, с использованием системы управления базой данных MySQL, предназначена для работы в операционной системе MS Windows 10 и выше, а для управления базой данных использовалась система MySQL Workbench 8.0. Оно было успешно отлажено и протестировано на данных контрольного примера.

Целью данного проекта было разработать приложение, которое упрощает и ускоряет процесс взаимодействия судей и спортсменов с документацией на соревнованиях по дзюдо. Традиционно документация на подобных соревнованиях представляется в виде бумажных носителей, что затрудняет эффективность и оперативность проведения мероприятий. Ручное заполнение бумажных документов требует много времени и может привести к ошибкам и задержкам в обработке данных. Кроме того, бумажные документы подвержены риску утери или повреждения, что создает проблемы доступа к информации о результатах соревнований.

Разработанное приложение JudoArena предлагает электронную форму для заполнения данных об спортсменах и их результатах, что значительно сокращает время и усилия, затрачиваемые на обработку информации. Оно также обеспечивает хранение данных в электронном виде, предотвращая потерю или повреждение документов. Благодаря использованию базы данных и возможностям системы MySQL Workbench, приложение обеспечивает надежное хранение и управление данными о соревнованиях, спортсменах, результатах и других важных аспектах проведения соревнований по дзюдо.

# Шаблоны выходных документов

Жеребьевка {Наименование соревнования}

Категория: {Весовая и возрастная категория}.

|  |  |  |
| --- | --- | --- |
| № | Спортсмен в белый | Спортсмен в синий |
| 1 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} |
| 2 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} |
| 3 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} |
| 4 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} |
| 5 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} |
| 6 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} |
| 7 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} |
| 8 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} |

Встречи

Дата формирования документа: {Дата формирования документа}

Главный судья: \_\_\_\_\_\_\_\_\_\_\_\_ {ФИО главного судьи}

Рисунок А.1 – Протокол жеребьевки

{Наименование соревнования}

Дата соревнования: {Дата соревнования}. Адрес: {Место проведения соревнования}.

Описание { Описание }.

Категория до {Весовая и возрастная категория}.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Фамилия | Имя | Отчество | Вес | Дата рождения | ФИО тренера |
| {Фамилия спортсмена} | {Имя спортсмена} | {Отчество спортсмена} | {Вес спортсмена} | {Дата рождения спортсмена} | {ФИО тренера спортсмена} |
| {Фамилия спортсмена} | {Имя спортсмена} | {Отчество спортсмена} | {Вес спортсмена} | {Дата рождения спортсмена} | {ФИО тренера спортсмена} |
| {Фамилия спортсмена} | {Имя спортсмена} | {Отчество спортсмена} | {Вес спортсмена} | {Дата рождения спортсмена} | {ФИО тренера спортсмена} |

Количество спортсменов: {Количество спортсменов}.

Категория до {Весовая и возрастная категория}.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Фамилия | Имя | Отчество | Вес | Дата рождения | ФИО тренера |
| {Фамилия спортсмена} | {Имя спортсмена} | {Отчество спортсмена} | {Вес спортсмена} | {Дата рождения спортсмена} | {ФИО тренера спортсмена} |
| {Фамилия спортсмена} | {Имя спортсмена} | {Отчество спортсмена} | {Вес спортсмена} | {Дата рождения спортсмена} | {ФИО тренера спортсмена} |
| {Фамилия спортсмена} | {Имя спортсмена} | {Отчество спортсмена} | {Вес спортсмена} | {Дата рождения спортсмена} | {ФИО тренера спортсмена} |

Количество спортсменов: {Количество спортсменов}.

Количество весовых категорий: {Количество весовых категорий}.

{ФИО главного судьи}

{Дата формирования документа}

Рисунок A.2 – Протокол соревнования

Итоги {Наименование соревнования}

Категория: {Весовая и возрастная категория}.

Занимаемые места

|  |  |
| --- | --- |
| Место № | Фио участнника |
| 1 | {ФИО спортсмена соревнования} |
| 2 | {ФИО спортсмена соревнования} |
| 3 | {ФИО спортсмена соревнования} |
| 3 | {ФИО спортсмена соревнования} |

Встречи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Спортсмен в белый | Спортсмен в синий | Оценки и время | Судья | Результат |
| 1 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 2 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 3 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 4 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 5 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 6 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 7 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 8 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 9 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 10 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |
| 11 | {ФИО спортсмена соревнования} | {ФИО спортсмена соревнования} | {Оценки, время} | {ФИО Судья} | {Победа спортсмена} |

Дата формирования документа: {Дата формирования документа}

Главный судья: \_\_\_\_\_\_\_\_\_\_\_\_ {ФИО главного судьи}

Рисунок А.1 – Итоговый протокол

# Схема базы данных

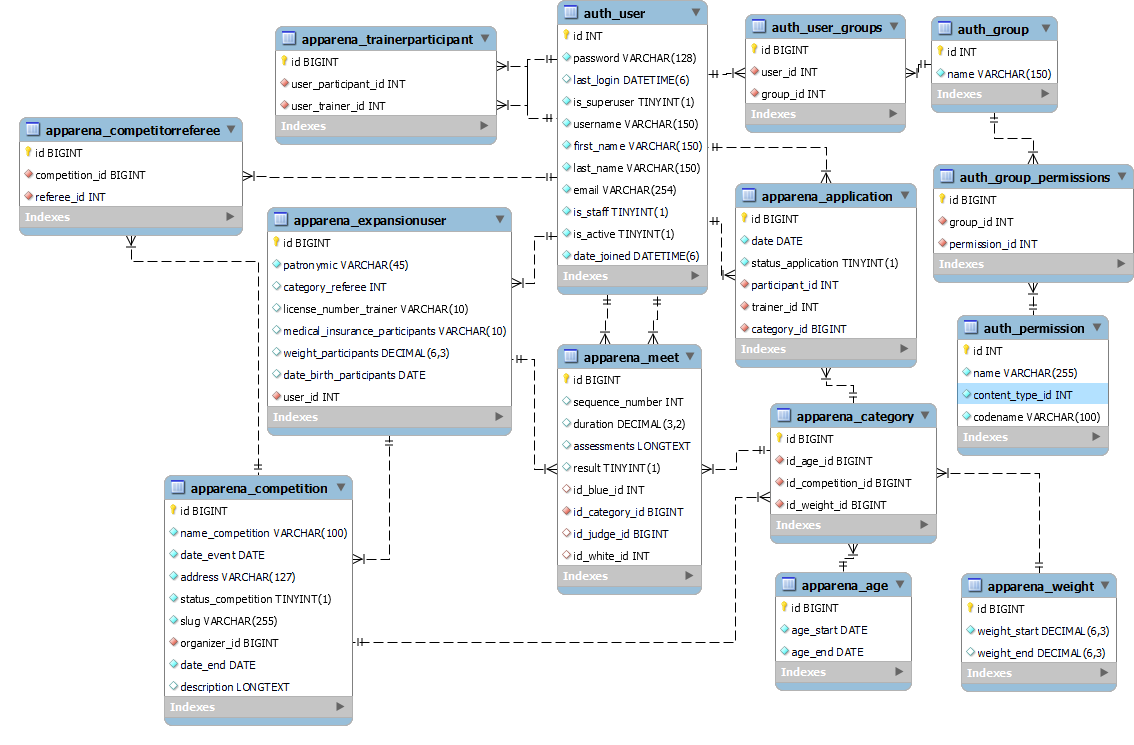


Рисунок Б.1 – Схема базы данных

# Входные данные контрольного примера

Таблица Б.1 – Спортсмен соревнования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ФИО | Вес | Дата рождения | Номер страховки | Логин | Пароль |
| Reynolds Richard Анатольевич | 42 | 18.10.2011 | 7372940541 | participant\_48trainer\_1 | gnidenko456789 |
| Kelley Kimberly Евгеньевич | 42 | 07.03.2011 | 5419179211 | participant\_56trainer\_11314 | participant\_56trainer\_11314 |
| Moody Jerry Дмитриевич | 42 | 02.11.2011 | 7890109909 | participant\_52trainer\_2 | participant\_52trainer\_2 |
| Martin Stephanie Алексеевич | 42 | 20.05.2011 | 1303309210 | participant\_62trainer\_3 | participant\_62trainer\_3 |
| Watson Sharon Евгеньевич | 42 | 03.04.2011 | 1845860285 | participant\_58trainer\_5 | participant\_58trainer\_5 |
| Riley Nathan Алексеевич | 42 | 20.03.2011 | 669898316 | participant\_06trainer\_51314 | participant\_06trainer\_51314 |

Таблица Б.2 –Тренер

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ФИО | Номер лицензии | Номер страховки | Логин | Пароль |
| Петров Иван Николаевич | 4569249701 | 7372940541 | petrovi@gmail.com | Petrovi452751 |

Таблица Б.4 – Данные для добавления соревнования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Дата | Адрес | Весовые категории | Возрастные категории |
| Кано́ Дзи́горо | 30.05.2024 | г.Уфа ул. Ладыгина, д 10 | 38 - 42 | 01.01.2011  31.12.2012 |

# Выходные данные контрольного примера

Протокол Кано́ Дзи́горо

Дата проведения: 2024-05-30

Адрес: г.Уфа ул. Ладыгина, д 10

Описание:

Судьи: Петров Петр Олегович

Категории и Участники

Вес: 38 - 42  
Возраст: 2011-01-01 - 2012-12-31

|  |  |  |  |
| --- | --- | --- | --- |
| Фамилия Имя Отчество | Вес | Дата Рождения | Возраст |
| Reynolds Richard Анатольевич | 42.000 кг | 18.10.2011 | 12 |
| Kelley Kimberly Евгеньевич | 42.000 кг | 07.03.2011 | 13 |
| Moody Jerry Дмитриевич | 42.000 кг | 02.11.2011 | 12 |
| Martin Stephanie Алексеевич | 42.000 кг | 20.05.2011 | 13 |
| Watson Sharon Евгеньевич | 42.000 кг | 03.04.2011 | 13 |
| Riley Nathan Алексеевич | 42.000 кг | 20.03.2011 | 13 |

Количество заявок: 6

Общее количество заявок

Всего заявок: 6

Дата формирования документа: 2024-05-30 16:54:18

Главный судья: \_\_\_\_\_\_\_\_\_\_\_\_ Петров Петр Олегович

Рисунок Г.1 - Протокол соревнования

Жеребьевка Кано́ Дзи́горо

Категория: вес 38.000 - 42.000 кг, возраст 2011-01-01 - 2012-12-31

Встречи

|  |  |  |
| --- | --- | --- |
| № | Участник в белый | Участник в синий |
| 1 | Нет соперника | Watson Sharon Евгеньевич |
| 2 | Нет соперника | Moody Jerry Дмитриевич |
| 3 | Kelley Kimberly Евгеньевич | Riley Nathan Алексеевич |
| 4 | Reynolds Richard Анатольевич | Martin Stephanie Алексеевич |

Дата формирования документа: 30.05.2024

Главный судья: \_\_\_\_\_\_\_\_\_\_\_\_ Петров Петр Олегович

Рисунок Г.2 - Протокол жеребьевки

# Итоги Кано́ Дзи́горо

Категория: вес 38.000 - 42.000 кг, возраст 2011-01-01 - 2012-12-31

Занимаемые места

|  |  |
| --- | --- |
| Место № | Фио участнника |
| 1 | Moody Jerry Дмитриевич |
| 2 | Kelley Kimberly Евгеньевич |
| 3 | Reynolds Richard Анатольевич |
| 3 | Watson Sharon Евгеньевич |

Встречи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Участник в белый | Участник в синий | Оценки и время | Судья | Резулитат |
| 1 | Нет соперника | Watson Sharon Евгеньевич | - | - | Победил синий спортсмен |
| 2 | Нет соперника | Moody Jerry Дмитриевич | - | - | Победил синий спортсмен |
| 3 | Kelley Kimberly Евгеньевич | Riley Nathan Алексеевич | 1.30мин. Б: ШИ С: В | Петров Петр Олегович | Победил белый спортсмен |
| 4 | Reynolds Richard Анатольевич | Martin Stephanie Алексеевич | 4.56мин. Б: В С: \_ | Петров Петр Олегович | Победил белый спортсмен |
| 5 | Watson Sharon Евгеньевич | Moody Jerry Дмитриевич | 3.21мин. Б: Ш С: \_ | Петров Петр Олегович | Победил синий спортсмен |
| 6 | Kelley Kimberly Евгеньевич | Reynolds Richard Анатольевич | 1.23мин. Б:И С: В | Петров Петр Олегович | Победил белый спортсмен |
| 7 | Moody Jerry Дмитриевич | Kelley Kimberly Евгеньевич | 5.45мин. Б: ВВШШ С: ШВШ | Петров Петр Олегович | Победил белый спортсмен |

Дата формирования документа: 19.06.2024

Главный судья: \_\_\_\_\_\_\_\_\_\_\_\_ Петров Владимир Артемович

Рисунок Г.3 - Итоговый протокол

# Код программы

from django.db import models  
from django.urls import reverse  
from django import forms  
from django.contrib.auth.models import User  
from django.utils.text import slugify  
from .utils import unique\_slugify

class ExpansionUser(models.Model):  
 user = models.OneToOneField(User, on\_delete=models.CASCADE)  
 patronymic = models.CharField(max\_length=45, verbose\_name='Отчество')  
  
 category\_referee = models.IntegerField(null=True, blank=True, verbose\_name='Категория')  
  
 license\_number\_trainer = models.CharField(max\_length=10, null=True, blank=True, verbose\_name='Лицензия')  
  
 medical\_insurance\_participants = models.CharField(max\_length=10, null=True, blank=True, verbose\_name='Номер страхования жизни')  
 weight\_participants = models.DecimalField(max\_digits=6, decimal\_places=3, null=True, blank=True, verbose\_name='Вес')  
 date\_birth\_participants = models.DateField(null=True, blank=True, verbose\_name='Дата рождения')  
  
 def \_\_str\_\_(self):  
 if self.date\_birth\_participants is None:  
 return f'{self.user.last\_name} {self.user.first\_name[:1]}. {self.patronymic[:1]}.'  
 else:  
 return f'{self.user.last\_name} {self.user.first\_name[:1]}. {self.patronymic[:1]}. {self.date\_birth\_participants}'  
  
  
class TrainerParticipant(models.Model):  
 user\_trainer = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='trainer')  
 user\_participant = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='participant')

class Competition(models.Model):  
 name\_competition = models.CharField(max\_length=100, verbose\_name='Название')  
 date\_event = models.DateField(verbose\_name='Дата проведения')  
 address = models.CharField(max\_length=127, verbose\_name='Адрес')  
 description = models.TextField(blank=True, null=True, verbose\_name="Описание")  
 status\_competition = models.BooleanField(default=True, verbose\_name='Статус')  
 organizer = models.ForeignKey(ExpansionUser, on\_delete=models.CASCADE, verbose\_name='Организатор')  
 slug = models.SlugField(max\_length=255, unique=True,  
 verbose\_name='Уникальное название')  
 date\_end = models.DateField(verbose\_name='Дата завершения')  
  
 def \_\_str\_\_(self):  
 return self.name\_competition  
 def get\_absolute\_url(self):  
 return reverse('competition', kwargs={'competition\_id': self.pk})  
  
 def save(self, \*args, \*\*kwargs):  
 if not self.slug:  
 self.slug = unique\_slugify(self, self.name\_competition)  
 print(self.slug)  
 super().save(\*args, \*\*kwargs)

class CompetitorReferee(models.Model):  
 competition = models.ForeignKey(Competition, on\_delete=models.CASCADE)  
 referee = models.ForeignKey(User, on\_delete=models.CASCADE)

class Age(models.Model):  
 age\_start = models.DateField()  
 age\_end = models.DateField()  
  
 def \_\_str\_\_(self):  
 return f"{self.age\_start} - {self.age\_end}"  
  
  
class Weight(models.Model):  
 weight\_start = models.DecimalField(max\_digits=6, decimal\_places=3)  
 weight\_end = models.DecimalField(max\_digits=6, decimal\_places=3, null=True, blank=True)  
  
 def \_\_str\_\_(self):  
 return f"{self.weight\_start} - {self.weight\_end}"  
  
  
class Category(models.Model):  
 id\_competition = models.ForeignKey(Competition, on\_delete=models.CASCADE)  
 id\_weight = models.ForeignKey(Weight, on\_delete=models.CASCADE)  
 id\_age = models.ForeignKey(Age, on\_delete=models.CASCADE)  
  
 def \_\_str\_\_(self):  
 return f"Category {self.id} - Competition {self.id\_competition} - Weight {self.id\_weight} - Age {self.id\_age}"  
  
 def get\_absolute\_url(self):  
 return reverse('category', kwargs={'category\_id': self.pk})  
  
  
class Application(models.Model):  
 trainer = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='trainer\_applications')  
 participant = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='participant\_applications')  
 category = models.ForeignKey(Category, on\_delete=models.CASCADE)  
 date = models.DateField(auto\_now\_add=True)  
 status\_application = models.BooleanField(default=False)  
  
 def \_\_str\_\_(self):  
 return (f"Application {self.id} - Participant {self.participant.last\_name} - Category {self.category} - "  
 f"Trainer {self.trainer.last\_name}")  
  
 def get\_absolute\_url(self):  
 return reverse('application', kwargs={'application\_id': self.pk})  
  
  
class Meet(models.Model):  
 id\_white = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='white\_meets', null=True, blank=True)  
 id\_blue = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='blue\_meets', null=True, blank=True)  
 sequence\_number = models.IntegerField(null=True, blank=True)  
 id\_category = models.ForeignKey(Category, on\_delete=models.CASCADE)  
 id\_judge = models.ForeignKey(ExpansionUser, related\_name='judge\_meets', on\_delete=models.CASCADE, null=True, blank=True)  
 duration = models.DecimalField(max\_digits=3, decimal\_places=2, null=True, blank=True)  
 assessments = models.TextField(null=True, blank=True)  
 result = models.BooleanField(null=True, blank=True)  
  
 def \_\_str\_\_(self):  
 return f"Meet sequence\_number {self.sequence\_number} - White {self.id\_white} - Blue {self.id\_blue} - Category {self.id\_category}"  
  
 def get\_absolute\_url(self):  
 return reverse('meet', kwargs={'meet\_id': self.pk})

from decimal import Decimal  
  
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm  
from django.utils import timezone  
from django.core.exceptions import ValidationError  
from django.utils.translation import gettext\_lazy as \_  
from django import forms  
from django.contrib.auth.models import User, Group  
from datetime import timedelta  
  
from .models import \*  
  
from django import forms  
from django.utils import timezone  
from datetime import timedelta  
from decimal import Decimal  
from django.contrib.auth.forms import UserCreationForm  
from django.contrib.auth.models import User  
  
  
class SignUpForm(UserCreationForm):  
 patronymic = forms.CharField(label='Отчество', widget=forms.TextInput(), max\_length=45)  
  
 ROLE\_CHOICES = (  
 (1, 'Спортсмен'),  
 (0, 'Tренер'),  
 )  
  
 role = forms.ChoiceField(label='Роль', choices=ROLE\_CHOICES, widget=forms.Select(), required=False)  
 license\_number\_trainer = forms.CharField(label='Лицензия', max\_length=10, widget=forms.TextInput(), required=False)  
 medical\_insurance\_participants = forms.CharField(label='Номер страхования жизни', max\_length=10,  
 widget=forms.TextInput(), required=False)  
 weight\_participants = forms.DecimalField(label='Вес', max\_digits=6, decimal\_places=3, widget=forms.NumberInput(),  
 required=False)  
 date\_birth\_participants = forms.DateField(label='Дата рождения', widget=forms.DateInput(attrs={'type': 'date'}),  
 required=False)  
 consent = forms.BooleanField(label='Даю согласие на обработку персональных данных', required=True)  
  
 id\_trener = forms.ModelChoiceField(queryset=User.objects.filter(groups\_\_name='Тренер'), label='Тренер', required=False)  
  
 class Meta:  
 model = User  
 fields = [  
 'username', 'last\_name', 'first\_name', 'patronymic', 'password1', 'password2', 'role',  
 'license\_number\_trainer', 'weight\_participants', 'date\_birth\_participants', 'consent', 'id\_trener'  
 ]  
 labels = {  
 'username': 'Логин'  
 }  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['id\_trener'].empty\_label = 'Выберите тренера'  
 trainer\_group = Group.objects.get(name="Тренер")  
 all\_trainer = User.objects.filter(groups=trainer\_group)  
 expansion\_users = ExpansionUser.objects.filter(user\_\_in=all\_trainer)  
 self.fields['id\_trener'].queryset = expansion\_users  
  
 def clean(self):  
 cleaned\_data = super().clean()  
 role = cleaned\_data.get('role')  
 license\_number\_trainer = cleaned\_data.get('license\_number\_trainer')  
 medical\_insurance\_participants = cleaned\_data.get('medical\_insurance\_participants')  
 weight\_participants = cleaned\_data.get('weight\_participants')  
 date\_birth\_participants = cleaned\_data.get('date\_birth\_participants')  
 id\_trener = cleaned\_data.get('id\_trener')  
  
 date\_birth = self.cleaned\_data.get('date\_birth\_participants')  
 if date\_birth and date\_birth > timezone.now().date():  
 self.add\_error('date\_birth\_participants', "Не верная дата рождения")  
 if date\_birth\_participants and date\_birth\_participants > timezone.now().date() - timedelta(days=365 \* 4):  
 self.add\_error('date\_birth\_participants', "Не верная дата рождения")  
  
 if date\_birth\_participants and date\_birth\_participants < timezone.now().date() - timedelta(days=365 \* 120):  
 self.add\_error('date\_birth\_participants', "Не верная дата рождения")  
  
 if weight\_participants is not None:  
 weight\_float = Decimal(weight\_participants)  
 if weight\_float < 25:  
 self.add\_error('weight\_participants', "Не верный вес")  
 if weight\_float > 500:  
 self.add\_error('weight\_participants', "Не верный вес")  
  
 if role == '0' and not license\_number\_trainer:  
 self.add\_error('license\_number\_trainer', 'Поле "Лицензия" обязательно для тренера')  
  
 if role == '1' and not medical\_insurance\_participants:  
 self.add\_error('medical\_insurance\_participants', 'Для спортсмена необходимо Страховку')  
 if role == '1' and not id\_trener:  
 self.add\_error('id\_trener', 'Для спортсмена необходимо выбрать тренера')  
 if role == '1' and not weight\_participants:  
 self.add\_error('weight\_participants', 'Для спортсмена необходимо заполнить Вес')  
 if role == '1' and not date\_birth\_participants:  
 self.add\_error('date\_birth\_participants', 'Для спортсмена необходимо заполнить Дату рождения')  
  
 if role == '1' and medical\_insurance\_participants and len(medical\_insurance\_participants) != 10:  
 self.add\_error('medical\_insurance\_participants', "Страховка имеет размер в 10 символов")  
  
 if role == '0' and license\_number\_trainer and len(license\_number\_trainer) != 10:  
 self.add\_error('license\_number\_trainer', "Лицензия имеет размер в 10 символов")  
  
 consent = cleaned\_data.get('consent')  
 if not consent:  
 self.add\_error('consent', 'Для регистрации необходимо дать согласие на обработку персональных данных')  
  
 return cleaned\_data  
  
 def save(self, commit=True):  
 user = super().save(commit=True)  
 user.first\_name = self.cleaned\_data['first\_name']  
 user.last\_name = self.cleaned\_data['last\_name']  
 user.save()  
  
 expansion\_user = ExpansionUser(  
 user=user,  
 patronymic=self.cleaned\_data['patronymic'],  
 )  
 print(self.cleaned\_data['role'])  
 if self.cleaned\_data['role'] == '1':  
 user.groups.add(2)  
 expansion\_user.medical\_insurance\_participants = self.cleaned\_data.get('medical\_insurance\_participants',  
 None)  
 expansion\_user.weight\_participants = self.cleaned\_data.get('weight\_participants', None)  
 expansion\_user.date\_birth\_participants = self.cleaned\_data.get('date\_birth\_participants', None)  
 else:  
 user.groups.add(1)  
 expansion\_user.license\_number\_trainer = self.cleaned\_data.get('license\_number\_trainer', None)  
  
 expansion\_user.save()  
 user.save()  
 return user  
  
  
class SignInForm(AuthenticationForm):  
 class Meta:  
 model = User  
 fields = ['username', 'password']  
  
 # login.html на 13стр кастыль  
 # labels = {  
 # 'username': 'Логин',  
 # }  
  
  
class AddCompetitionForm(forms.ModelForm):  
  
 class Meta:  
 model = Competition  
 fields = ['name\_competition', 'date\_event', 'address', 'description', 'date\_end', 'organizer']  
 widgets = {  
 'description': forms.Textarea(attrs={'class': 'textarea\_class'}),  
 'address': forms.Textarea(attrs={'class': 'textarea\_class'}),  
 'date\_event': forms.DateInput(attrs={'type': 'date'}),  
 'date\_end': forms.DateInput(attrs={'type': 'date'}),  
 'name\_competition': forms.Textarea(attrs={'class': 'textarea\_class'}),  
  
 }  
  
 def clean(self):  
 cleaned\_data = super().clean()  
 date\_event = cleaned\_data.get('date\_event')  
  
 if date\_event and date\_event < timezone.now().date():  
 self.add\_error('date\_event', "Не верная дата")  
  
 date\_end = cleaned\_data.get('date\_end')  
  
 if date\_end and date\_end < timezone.now().date():  
 self.add\_error('date\_end', "Не верная дата")  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(AddCompetitionForm, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['organizer'].empty\_label = 'Выберите судью'  
 trainer\_group = Group.objects.get(name="Судья")  
 all\_judges = User.objects.filter(groups=trainer\_group)  
 expansion\_users = ExpansionUser.objects.filter(user\_\_in=all\_judges)  
 self.fields['organizer'].queryset = expansion\_users  
  
  
class ProfileForm(forms.ModelForm):  
  
 class Meta:  
 model = User  
 fields = ['username', 'last\_name', 'first\_name']  
  
  
class ProfileExpansionForm(forms.ModelForm):  
  
 class Meta:  
 model = ExpansionUser  
 fields = ['patronymic', 'medical\_insurance\_participants',  
 'weight\_participants']  
  
  
class ProfileTrainerParticipant(forms.ModelForm):  
 user\_trainer = forms.ModelChoiceField(  
 queryset=ExpansionUser.objects.filter(user\_\_in=User.objects.filter(groups=Group.objects.get(name="Тренер"))),  
 label='Тренер',  
 )  
  
 class Meta:  
 model = TrainerParticipant  
 fields = ['user\_trainer']  
  
  
class JudgesForm(forms.Form):  
  
 judges = forms.ModelMultipleChoiceField(  
 queryset=ExpansionUser.objects.select\_related('user').all(),  
 widget=forms.CheckboxSelectMultiple,  
 required=False,  
 label='Добавить судью'  
 )  
 existing\_judges = forms.ModelMultipleChoiceField(  
 queryset=ExpansionUser.objects.select\_related('user').all(),  
 widget=forms.CheckboxSelectMultiple,  
 required=False,  
 label='Исключить судью'  
 )  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 competition = kwargs.pop('competition', None)  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
  
 judges\_group = Group.objects.get(name="Судья")  
 all\_judges = User.objects.filter(groups=judges\_group)  
 expansion\_users = ExpansionUser.objects.filter(user\_\_in=all\_judges)  
  
 if competition:  
 existing\_judges\_qs = User.objects.filter(competitorreferee\_\_competition\_id=competition)  
 existing\_expansion\_users\_qs = ExpansionUser.objects.filter(user\_\_in=existing\_judges\_qs)  
 self.fields['existing\_judges'].queryset = existing\_expansion\_users\_qs  
 self.fields['judges'].queryset = expansion\_users.exclude(user\_\_in=existing\_judges\_qs)  
 else:  
 self.fields['judges'].queryset = expansion\_users  
  
  
class CategoryForm(forms.ModelForm):  
 class Meta:  
 model = Category  
 fields = ['id\_weight', 'id\_age']  
 labels = {  
 'id\_weight': 'Весовая категория',  
 'id\_age': 'Возрастная категория',  
 }  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(CategoryForm, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['id\_weight'].empty\_label = 'Выберите категорию'  
 self.fields['id\_age'].empty\_label = 'Выберите возраст'  
 self.fields['id\_age'].queryset = Age.objects.all()  
 self.fields['id\_weight'].queryset = Weight.objects.all()  
  
  
class AgeForm(forms.ModelForm):  
 class Meta:  
 model = Age  
 fields = ['age\_start', 'age\_end']  
 widgets = {  
 'age\_start': forms.DateInput(attrs={'type': 'date'}),  
 'age\_end': forms.DateInput(attrs={'type': 'date'}),  
 }  
 labels = {  
 'age\_start': 'Начало',  
 'age\_end': 'Конец',  
 }  
  
 def clean\_age\_start(self):  
 age\_start = self.cleaned\_data['age\_start']  
 if age\_start:  
 return age\_start.replace(month=1, day=1)  
 return age\_start  
  
 def clean\_age\_end(self):  
 age\_end = self.cleaned\_data['age\_end']  
 if age\_end:  
 return age\_end.replace(month=12, day=31)  
 return age\_end  
  
  
class WeightForm(forms.ModelForm):  
 class Meta:  
 model = Weight  
 fields = ['weight\_start', 'weight\_end']  
 widgets = {  
 'weight\_start': forms.NumberInput(attrs={'type': 'number'}),  
 'weight\_end': forms.NumberInput(attrs={'type': 'number'}),  
 }  
 labels = {  
 'weight\_start': 'Начало',  
 'weight\_end': 'Конец',  
 }  
  
  
class ApplicationForm(forms.ModelForm):  
 class Meta:  
 model = Application  
 fields = ['trainer', 'participant', 'category']  
  
  
class MeetForm(forms.ModelForm):  
 RESULT\_CHOICES = [  
 (None, 'Не проводилось'),  
 (True, 'Победил белый спортсмен'),  
 (False, 'Победил синий спортсмен')  
 ]  
  
 result = forms.ChoiceField(  
 choices=RESULT\_CHOICES,  
 label='Результат',  
 widget=forms.Select  
 )  
 class Meta:  
 model = Meet  
 fields = ['id\_judge', 'duration', 'assessments', 'result']  
 labels = {  
 'id\_judge': 'Судья',  
 'duration': 'Продолжительность (минуты)',  
 'assessments': 'Оценки',  
 'result': 'Результат',  
 }  
 widgets = {  
 'assessments': forms.Textarea(attrs={'rows': 1}),  
 }  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(MeetForm, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['id\_judge'].empty\_label = 'Выберите судью'  
 trainer\_group = Group.objects.get(name="Судья")  
 all\_judges = User.objects.filter(groups=trainer\_group)  
 competitor\_referees = CompetitorReferee.objects.filter(referee\_\_in=all\_judges)  
 expansion\_users = ExpansionUser.objects.filter(user\_\_in=competitor\_referees.values('referee'))  
 self.fields['id\_judge'].queryset = expansion\_users

from django.http import HttpResponse  
from django.shortcuts import render, redirect, get\_object\_or\_404  
  
from django.contrib.auth import authenticate, login, logout  
from .forms import \*  
from .models import \*  
  
from datetime import date, datetime  
from django.db.models import Case, When, IntegerField, Q  
from django.contrib.auth.decorators import login\_required  
from django.core.exceptions import ObjectDoesNotExist  
from .utils import group\_required  
import random  
import math  
  
from docx import Document  
from docx.shared import Pt, RGBColor  
from docx.oxml.ns import qn  
from docx.enum.text import WD\_ALIGN\_PARAGRAPH  
from docx.enum.table import WD\_TABLE\_ALIGNMENT  
from docx.shared import Inches  
  
  
MENU = [{'title': "Добавить соревнование", 'url\_name': 'add\_competition'},  
 {'title': "На главную", 'url\_name': 'show\_competitions'},]  
  
  
def register(request):  
 if request.method == 'POST':  
 form = SignUpForm(request.POST)  
 if form.is\_valid():  
 user\_form = form.save()  
 id\_trener = form.cleaned\_data['id\_trener']  
 trener, created = TrainerParticipant.objects.get\_or\_create(user\_trainer=id\_trener.user, user\_participant=user\_form)  
 if not created:  
 error\_message = "Age range already exists."  
  
 username = form.cleaned\_data.get('username')  
 password = form.cleaned\_data.get('password1')  
 user = authenticate(request, username=username, password=password)  
 if user is not None:  
 login(request, user)  
 return redirect('show\_competitions')  
 else:  
 logout(request)  
 form = SignUpForm()  
 return render(request, 'AppArena/register.html', {'form': form, 'title': "Регистрация", 'name\_btn': 'Зарегистрироваться'})  
  
  
def user\_login(request):  
 if request.method == 'POST':  
 form = SignInForm(request, request.POST)  
  
 if form.is\_valid():  
 username = form.cleaned\_data.get('username')  
 password = form.cleaned\_data.get('password')  
 user = authenticate(request, username=username, password=password)  
  
 if user is not None:  
 login(request, user)  
 return redirect('show\_competitions')  
 else:  
 logout(request)  
 form = SignInForm()  
 return render(request, 'AppArena/login.html', {'form': form, 'title': "Авторизация"})  
  
  
@login\_required  
def user\_logout(request):  
 logout(request)  
 return redirect('login')  
  
  
@login\_required  
def show\_competitions(request):  
 today = date.today()  
 competitions = Competition.objects.all()  
  
 competitions = competitions.annotate(  
 order=Case(  
 When(date\_event\_\_lte=today, date\_end\_\_gte=today, then=0),  
 When(date\_event\_\_gt=today, then=1),  
 When(date\_event\_\_lt=today, then=2),  
 output\_field=IntegerField(),  
 )  
 ).order\_by('order', 'date\_event')  
 page\_menu = MENU.copy()  
  
 query = request.GET.get('q')  
 if query:  
 competitions = competitions.filter(name\_competition\_\_icontains=query)  
  
 status = request.GET.get('status')  
 if status == 'pending':  
 competitions = competitions.filter(status\_competition=True, date\_event\_\_gt=today)  
 elif status == 'completed':  
 competitions = competitions.filter(status\_competition=True, date\_event\_\_lt=today)  
 elif status == 'ongoing':  
 competitions = competitions.filter(Q(date\_event\_\_gte=today, date\_end\_\_lte=today))  
 elif status == 'not\_scheduled':  
 competitions = competitions.filter(status\_competition=False)  
  
 start\_date = request.GET.get('start\_date')  
 end\_date = request.GET.get('end\_date')  
 if start\_date and end\_date:  
 start\_date = datetime.strptime(start\_date, '%Y-%m-%d').date()  
 end\_date = datetime.strptime(end\_date, '%Y-%m-%d').date()  
 competitions = competitions.filter(date\_event\_\_range=[start\_date, end\_date])  
  
 context = {  
 'title': 'Список соревнований',  
 'competitions': competitions,  
 'menu': page\_menu[:1],  
 'today': today  
 }  
 return render(request, 'AppArena/competitions.html', context)  
  
  
@login\_required  
@group\_required('Секретарь')  
def add\_competition(request):  
 page\_menu = MENU.copy()  
 if request.method == 'POST':  
 form = AddCompetitionForm(request.POST)  
 if form.is\_valid():  
 competition = form.save()  
 judge = form.cleaned\_data['organizer']  
 CompetitorReferee.objects.create(competition=competition, referee=judge.user)  
 return redirect('choose\_judges', competition.slug)  
 else:  
 form = AddCompetitionForm()  
 context = {  
 'title': "Добавление соревнования",  
 'form': form,  
 'menu': page\_menu[1:]  
 }  
 return render(request, 'AppArena/add\_competition.html', context)  
  
  
@login\_required  
def edit\_profile(request, id\_user):  
 user = get\_object\_or\_404(User, pk=id\_user)  
 expansion\_user = get\_object\_or\_404(ExpansionUser, user\_id=id\_user)  
 groups = user.groups.values\_list('name', flat=True)  
 if request.method == 'POST':  
 form\_user = ProfileForm(request.POST, instance=user)  
 form\_expansion\_user = ProfileExpansionForm(request.POST, instance=expansion\_user)  
 if form\_user.is\_valid() and form\_expansion\_user.is\_valid():  
 form\_user.save()  
 form\_expansion\_user.save()  
 return redirect('show\_competitions')  
 else:  
 form\_user = ProfileForm(instance=user)  
 form\_expansion\_user = ProfileExpansionForm(instance=expansion\_user)  
 context = {  
 'title': "Редактирование профиля",  
 'form\_user': form\_user,  
 'form\_expansion\_user': form\_expansion\_user,  
 'role': list(groups),  
 }  
 return render(request, 'AppArena/edit\_profile.html', context)  
  
  
@login\_required  
@group\_required('Секретарь')  
def reject\_application(request, comp\_slug, id\_category, id\_application):  
 application = get\_object\_or\_404(Application, id=id\_application)  
 if request.user.groups.filter(name='Секретарь').exists():  
 application.status\_application = not application.status\_application  
 application.save()  
 return redirect('category', comp\_slug=comp\_slug, id\_category=id\_category)  
 else:  
 return HttpResponse("Вы не имеете права на удаление соревнования.", status=403)  
  
  
@login\_required  
def show\_competition(request, comp\_slug):  
 today = date.today()  
 comp = get\_object\_or\_404(Competition, slug=comp\_slug)  
 categories = Category.objects.filter(id\_competition=comp)  
 referees = CompetitorReferee.objects.filter(competition=comp).select\_related('referee')  
 expansion\_users = ExpansionUser.objects.filter(user\_\_in=[ref.referee for ref in referees])  
 context = {  
 'title': f"{comp.name\_competition}",  
 'comp': comp,  
 'categories': categories,  
 'role': request.user.groups.all(),  
 'today': today,  
 'referees': expansion\_users,  
 }  
 return render(request, 'AppArena/competition\_view.html', context)  
  
  
@login\_required  
@group\_required('Секретарь')  
def choose\_judges(request, comp\_slug):  
 page\_menu = MENU.copy()  
 competition = get\_object\_or\_404(Competition, slug=comp\_slug)  
 previous\_url = request.META.get('HTTP\_REFERER', '')  
 if request.method == 'POST':  
 form = JudgesForm(request.POST, competition=competition)  
 if form.is\_valid():  
 judges = form.cleaned\_data['judges']  
  
 for judge in judges:  
 CompetitorReferee.objects.create(competition=competition, referee=judge.user)  
  
 existing\_judges = form.cleaned\_data['existing\_judges']  
 for judge in existing\_judges:  
 CompetitorReferee.objects.filter(competition=competition, referee=judge.user).delete()  
 if 'competitions/' in previous\_url:  
 return redirect('create\_category', competition.slug)  
 else:  
 return redirect('competition', competition.slug)  
 else:  
 form = JudgesForm(competition=competition)  
 last\_name\_query = request.GET.get('last\_name', '')  
 if last\_name\_query:  
 form.fields['judges'].queryset = form.fields['judges'].queryset.filter(  
 user\_\_last\_name\_\_icontains=last\_name\_query  
 )  
 form.fields['existing\_judges'].queryset = form.fields['existing\_judges'].queryset.filter(  
 user\_\_last\_name\_\_icontains=last\_name\_query  
 )  
 context = {  
 'title': f"Добавление судьи на {competition.name\_competition}",  
 'competition': competition,  
 'form': form,  
 'menu': page\_menu[1:]  
 }  
 return render(request, 'AppArena/choose\_judges.html', context)  
  
  
@login\_required  
def category(request, comp\_slug, id\_category):  
 comp = get\_object\_or\_404(Competition, slug=comp\_slug)  
 category = get\_object\_or\_404(Category, id=id\_category)  
 query = request.GET.get('q')  
 status = request.GET.get('status')  
 filters = Q(category\_id=category.pk)  
  
 if query:  
 filters &= Q(participant\_\_last\_name\_\_icontains=query) | Q(trainer\_\_last\_name\_\_icontains=query)  
  
 if status:  
 filters &= Q(status\_application=status)  
 if status:  
 filters &= Q(status\_application=status)  
  
 applications = Application.objects.filter(filters).order\_by('-status\_application')  
 context = {  
 'title': "Список спортсменов",  
 'applications': applications,  
 'comp\_slug': comp\_slug,  
 'category\_id': category.pk,  
 'comp': comp,  
 'title2': f"Спортсмени категории. Вес {category.id\_weight}кг. Возраст {category.id\_age}",  
 }  
 return render(request, 'AppArena/category\_view.html', context)  
  
  
@login\_required  
@group\_required('Секретарь')  
def create\_category(request, comp\_slug):  
 competition = get\_object\_or\_404(Competition, slug=comp\_slug)  
  
 age\_form = AgeForm()  
 weight\_form = WeightForm()  
 category\_form = CategoryForm()  
  
 error\_message = ''  
 if request.method == 'POST':  
 if 'add\_age' in request.POST:  
 age\_form = AgeForm(request.POST)  
 if age\_form.is\_valid():  
 age\_start = age\_form.cleaned\_data['age\_start']  
 age\_end = age\_form.cleaned\_data['age\_end']  
 if age\_start:  
 age\_start = age\_start.replace(month=1, day=1)  
 if age\_end:  
 age\_end = age\_end.replace(month=12, day=31)  
  
 if age\_start > age\_end:  
 error\_message = "Дата начала не может быть позже даты окончания"  
 elif age\_end and age\_end > timezone.now().date() - timedelta(days=365 \* 4):  
 error\_message = "Дата окончания не может быть позже чем на 4 года в будущем"  
 else:  
 age, created = Age.objects.get\_or\_create(age\_start=age\_start, age\_end=age\_end)  
 if created:  
 age\_form = AgeForm()  
 else:  
 error\_message = "Этот возраст уже существует"  
 elif 'add\_weight' in request.POST:  
 weight\_form = WeightForm(request.POST)  
 if weight\_form.is\_valid():  
 weight\_start = weight\_form.cleaned\_data['weight\_start']  
 weight\_end = weight\_form.cleaned\_data['weight\_end']  
 if weight\_start < 15:  
 error\_message = "Вес начала не может быть менее 15 кг"  
 elif weight\_start > weight\_end:  
 error\_message = "Вес начала не может быть больше веса окончания"  
 else:  
 weight, created = Weight.objects.get\_or\_create(weight\_start=weight\_start, weight\_end=weight\_end)  
 if created:  
 weight\_form = WeightForm()  
 else:  
 error\_message = "Этот вес уже существует"  
  
 else:  
 category\_form = CategoryForm(request.POST)  
 if category\_form.is\_valid():  
 id\_weight = category\_form.cleaned\_data['id\_weight']  
 id\_age = category\_form.cleaned\_data['id\_age']  
  
 try:  
 Category.objects.get(id\_competition=competition, id\_weight=id\_weight, id\_age=id\_age)  
 return render(request, 'AppArena/create\_category.html', {  
 'category\_form': category\_form,  
 'age\_form': age\_form,  
 'weight\_form': weight\_form,  
 'error\_message': 'Эта категория уже добавлена',  
 'message': '',  
 'comp': competition,  
 'title': f"Добавить категорию на {competition.name\_competition}",  
 })  
 except ObjectDoesNotExist:  
 category = category\_form.save(commit=False)  
 category.id\_competition = competition  
 category.save()  
 return render(request, 'AppArena/create\_category.html', {  
 'category\_form': category\_form,  
 'age\_form': age\_form,  
 'weight\_form': weight\_form,  
 'error\_message': '',  
 'message': 'Успешно добавлена категория',  
 'comp': competition,  
 'title': f"Добавить категорию на {competition.name\_competition}",  
 })  
  
 return render(request, 'AppArena/create\_category.html', {  
 'category\_form': category\_form,  
 'age\_form': age\_form,  
 'weight\_form': weight\_form,  
 'error\_message': error\_message,  
 'message': '',  
 'comp': competition,  
 'title': f"Добавить категорию на {competition.name\_competition}",  
 })  
  
  
@login\_required  
@group\_required('Тренер')  
def suitable\_participants(request, id\_category, id\_participant=None):  
 category = Category.objects.get(id=id\_category)  
 logged\_in\_trainer = request.user  
  
 participant\_ids = TrainerParticipant.objects.filter(user\_trainer=logged\_in\_trainer).values\_list('user\_participant',  
 flat=True)  
 query = request.GET.get('q')  
 if query:  
 participants = User.objects.filter(  
 last\_name\_\_icontains=query,  
 id\_\_in=participant\_ids,  
 expansionuser\_\_date\_birth\_participants\_\_range=(category.id\_age.age\_start, category.id\_age.age\_end),  
 expansionuser\_\_weight\_participants\_\_range=(category.id\_weight.weight\_start, category.id\_weight.weight\_end)  
 ).exclude(  
 id\_\_in=Application.objects.filter(  
 Q(participant\_\_id\_\_in=participant\_ids) & Q(category=category)  
 ).values\_list('participant\_\_id', flat=True)  
 )  
 else:  
 participants = User.objects.filter(  
 id\_\_in=participant\_ids,  
 expansionuser\_\_date\_birth\_participants\_\_range=(category.id\_age.age\_start, category.id\_age.age\_end),  
 expansionuser\_\_weight\_participants\_\_range=(category.id\_weight.weight\_start, category.id\_weight.weight\_end)  
 ).exclude(  
 id\_\_in=Application.objects.filter(  
 Q(participant\_\_id\_\_in=participant\_ids) & Q(category=category)  
 ).values\_list('participant\_\_id', flat=True)  
 )  
  
 if id\_participant is not None:  
 participant = get\_object\_or\_404(User, id=id\_participant)  
 application, created = Application.objects.get\_or\_create(trainer=logged\_in\_trainer, participant=participant, category=category)  
 if created:  
 application\_form = ApplicationForm()  
 else:  
 error\_message = "Ошибка"  
  
 context = {  
 'participants': participants,  
 'category': category,  
 'comp\_slug': category.id\_competition.slug,  
 'title': f"Падача заявки на {category.id\_competition.name\_competition}",  
 'title2': f"Подходящие спортсмени категории. Вес {category.id\_weight}кг. Возраст{category.id\_age}",  
 }  
 return render(request, 'AppArena/suitable\_participants.html', context)  
  
  
@login\_required  
def meets(request, comp\_slug, id\_category):  
 category = get\_object\_or\_404(Category, id=id\_category)  
 competitor\_referees = CompetitorReferee.objects.filter(competition=category.id\_competition)  
  
 referees = [competitor\_referee.referee for competitor\_referee in competitor\_referees]  
 applications = Application.objects.filter(category\_id=category, status\_application=True)  
  
 meets = list(Meet.objects.filter(id\_category\_id=category).order\_by('sequence\_number'))  
  
 applications\_list = list(applications)  
 participant\_count = next\_power\_of\_two(len(applications\_list))  
  
 rounds = {}  
 total\_rounds = power\_of\_two\_exponent(participant\_count)  
 current\_start\_index = 0  
 match\_number = 0  
 offset = 0  
  
 for round\_index in range(total\_rounds -1, 0, -1):  
 meets = list(Meet.objects.filter(id\_category\_id=category).order\_by('sequence\_number'))  
 matches\_in\_round = 2 \*\* round\_index  
 current\_round\_meets = []  
  
 for j in range(current\_start\_index, current\_start\_index + matches\_in\_round):  
 if j < len(meets):  
 current\_round\_meets.append(meets[j])  
 match\_number += 1  
  
 match\_number -= offset  
 rounds[total\_rounds - round\_index - 1] = current\_round\_meets  
 offset = 0  
 previous\_round\_index = total\_rounds - round\_index - 1  
 for j in range(0, current\_start\_index + matches\_in\_round, 2):  
 if j < len(rounds.get(previous\_round\_index, [])):  
 meet1 = rounds[previous\_round\_index][j]  
 if j + 1 < len(rounds[previous\_round\_index]):  
 meet2 = rounds[previous\_round\_index][j + 1]  
 else:  
 break  
  
 if meet1.result is None or meet2.result is None:  
 match\_number += 1  
 continue  
  
 winner1 = meet1.id\_white if meet1.result else meet1.id\_blue  
 winner2 = meet2.id\_white if meet2.result else meet2.id\_blue  
 match\_number += 1  
 offset += 1  
  
 Meet.objects.get\_or\_create(  
 id\_white=winner1,  
 id\_blue=winner2,  
 sequence\_number=match\_number,  
 id\_category=category  
 )  
  
 current\_start\_index += matches\_in\_round  
  
 sum = 0  
 for i in range(total\_rounds):  
 sum += 2\*\*i  
 if sum == len(meets):  
 rounds[len(rounds)] = [meets[len(meets)-1]]  
  
 context = {  
 'title': "Список встреч",  
 'applications': applications,  
 'meets': meets,  
 'category': category,  
 'comp\_slug': comp\_slug,  
 'rounds': rounds,  
 'competitor\_referee': referees,  
 'title2': f"Встречи категории. Вес {category.id\_weight}кг.<br>Возраст {category.id\_age}",  
 }  
  
 return render(request, 'AppArena/meets\_view.html', context)  
  
  
def next\_power\_of\_two(x):  
 return 1 if x == 0 else 2\*\*(x - 1).bit\_length()  
  
  
def power\_of\_two\_exponent(n):  
 if n < 1 or (n & (n - 1)) != 0:  
 raise ValueError("The input must be a power of 2.")  
 return int(math.log2(n))  
  
  
@login\_required  
@group\_required('Судья')  
def draw\_meet(request, comp\_slug, id\_category):  
 try:  
 category = Category.objects.get(pk=id\_category)  
 except Category.DoesNotExist:  
 return render(request, 'error.html', {'message': 'Category not found'})  
  
 Meet.objects.filter(id\_category=category, id\_judge\_\_isnull=True).delete()  
  
 participants = User.objects.filter(  
 groups\_\_name='Спортсмен',  
 participant\_applications\_\_category=category,  
 participant\_applications\_\_status\_application=True  
 ).distinct()  
 participant\_list = list(participants)  
  
 target\_length = next\_power\_of\_two(len(participant\_list))  
  
 while len(participant\_list) < target\_length:  
 participant\_list.append(None)  
  
 random.shuffle(participant\_list)  
  
 valid\_pairs = []  
 i = 0  
 while i < len(participant\_list):  
 white = participant\_list[i]  
 blue = participant\_list[i + 1] if participant\_list[i + 1] else None  
 if white is None and blue is None:  
 random.shuffle(participant\_list)  
 i = 0  
 valid\_pairs = []  
 continue  
  
 valid\_pairs.append((white, blue))  
 i += 2  
  
 sequence\_number = 1  
 for white, blue in valid\_pairs:  
 result = None  
 if white is None:  
 result = 0  
 elif blue is None:  
 result = 1  
 meet, created = Meet.objects.get\_or\_create(  
 id\_white=white,  
 id\_blue=blue,  
 sequence\_number=sequence\_number,  
 id\_category=category,  
 result=result  
 )  
 if not created:  
 error\_message = "Этот вес уже существует"  
  
 sequence\_number += 1  
  
 return redirect('meets', comp\_slug, id\_category)  
  
  
@login\_required  
@group\_required('Секретарь')  
def edit\_meet(request, meet\_id):  
  
 meet = get\_object\_or\_404(Meet, id=meet\_id)  
 category = get\_object\_or\_404(Category, id=meet.id\_category.pk)  
 applications = Application.objects.filter(category=meet.id\_category)  
 if request.method == 'POST':  
 form = MeetForm(request.POST, instance=meet)  
 if form.is\_valid():  
  
 form.save()  
  
 return redirect('meets', comp\_slug=meet.id\_category.id\_competition.slug, id\_category=meet.id\_category.pk)  
 else:  
 form = MeetForm(instance=meet)  
 context = {  
 'applications': applications,  
 'comp\_slug': category.id\_competition.slug,  
 'category\_id': category.pk,  
 'form': form,  
 }  
 return render(request, 'AppArena/edit\_meet.html', context)  
  
  
@login\_required  
def meet\_detail(request, meet\_id):  
 meet = get\_object\_or\_404(Meet, id=meet\_id)  
 return render(request, 'AppArena/meet\_detail.html', {'meet': meet})  
  
  
def generate\_doc\_view(request, comp\_slug):  
 try:  
 competition = get\_object\_or\_404(Competition, slug=comp\_slug)  
 response = generate\_docx\_for\_competition(competition, competition.organizer)  
  
 today = date.today()  
 comp = get\_object\_or\_404(Competition, slug=comp\_slug)  
 categories = Category.objects.filter(id\_competition=comp)  
 referees = CompetitorReferee.objects.filter(competition=comp).select\_related('referee')  
 expansion\_users = ExpansionUser.objects.filter(user\_\_in=[ref.referee for ref in referees])  
 context = {  
 'title': f"{comp.name\_competition}",  
 'comp': comp,  
 'categories': categories,  
 'role': request.user.groups.all(),  
 'today': today,  
 'referees': expansion\_users,  
 }  
 return render(request, 'AppArena/competition\_view.html', context)  
 except Competition.DoesNotExist:  
 return HttpResponse("Соревнование не найдено", status=404)  
  
  
def generate\_docx\_for\_competition(competition, organizer):  
 try:  
 doc\_date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')  
 competition\_name = competition.name\_competition.replace('"', '')  
 date\_event = str(competition.date\_event)  
 address = competition.address  
 description = competition.description  
  
 doc = Document()  
 doc.styles['Normal'].font.name = 'Times New Roman'  
 doc.styles['Normal'].font.size = Pt(12)  
 doc.styles['Normal'].element.rPr.rFonts.set(qn('w:eastAsia'), 'Times New Roman')  
  
 # Title  
 title = doc.add\_heading(level=1)  
 run = title.add\_run(f'Протокол {competition\_name}')  
 run.font.size = Pt(14)  
 run.font.color.rgb = RGBColor(0, 0, 0)  
 title.alignment = WD\_ALIGN\_PARAGRAPH.CENTER  
  
 doc.add\_paragraph(f"Дата проведения: {date\_event}")  
 doc.add\_paragraph(f"Адрес: {address}")  
 doc.add\_paragraph(f"Описание: {description}")  
  
 # Referees  
 referees = User.objects.filter(competitorreferee\_\_competition\_id=competition)  
 subheading = doc.add\_heading('Судьи', level=2)  
 subheading\_run = subheading.runs[0]  
 subheading\_run.font.color.rgb = RGBColor(0, 0, 0)  
 for referee in referees:  
 full\_name = f"{referee.last\_name} {referee.first\_name} {referee.expansionuser.patronymic}"  
 doc.add\_paragraph(full\_name, style='List Bullet')  
  
 # Categories and Participants  
 categories = Category.objects.filter(id\_competition=competition)  
 subheading = doc.add\_heading('Категории и Спортсмени', level=2)  
 subheading\_run = subheading.runs[0]  
 subheading\_run.font.color.rgb = RGBColor(0, 0, 0)  
  
 total\_applications = 0  
  
 for category in categories:  
 category\_id\_weight = category.id\_weight  
 category\_id\_age = category.id\_age  
  
 participants = [  
 (  
 f"{app.participant.last\_name} {app.participant.first\_name} {app.participant.expansionuser.patronymic}",  
 f"{app.participant.expansionuser.weight\_participants} кг",  
 app.participant.expansionuser.date\_birth\_participants  
 )  
 for app in Application.objects.filter(category=category)  
 ]  
  
 total\_applications += len(participants)  
  
 subheading = doc.add\_heading(f"Вес: {category\_id\_weight.weight\_start:0.0f} - {category\_id\_weight.weight\_end:0.0f}\nВозраст: {category\_id\_age}", level=2)  
 subheading\_run = subheading.runs[0]  
 subheading\_run.font.color.rgb = RGBColor(0, 0, 0)  
  
 table = doc.add\_table(rows=1, cols=4)  
 table.style = 'Table Grid'  
 table.alignment = WD\_TABLE\_ALIGNMENT.CENTER  
 hdr\_cells = table.rows[0].cells  
 hdr\_cells[0].text = 'Фамилия Имя Отчество'  
 hdr\_cells[1].text = 'Вес'  
 hdr\_cells[2].text = 'Дата Рождения'  
 hdr\_cells[3].text = 'Возраст'  
  
 for name, weight, birth\_date in participants:  
 row\_cells = table.add\_row().cells  
 row\_cells[0].text = name  
 row\_cells[1].text = weight  
 row\_cells[2].text = birth\_date.strftime('%d.%m.%Y')  
 age = (competition.date\_event.year - birth\_date.year) - ((competition.date\_event.month, competition.date\_event.day) < (birth\_date.month, birth\_date.day))  
 row\_cells[3].text = str(age)  
  
 doc.add\_paragraph(f"Количество заявок: {len(participants)}")  
  
 subheading = doc.add\_heading('Общее количество заявок', level=2)  
 subheading\_run = subheading.runs[0]  
 subheading\_run.font.color.rgb = RGBColor(0, 0, 0)  
 doc.add\_paragraph(f"Всего заявок: {total\_applications}")  
  
 doc.add\_paragraph(f"Дата формирования документа: {doc\_date}")  
 doc.add\_paragraph(f"Главный судья: \_\_\_\_\_\_\_\_\_\_\_\_ {organizer.user.last\_name} {organizer.user.first\_name} {organizer.patronymic}")  
  
 file\_name = f'{competition\_name}\_протокол\_{datetime.now().strftime("%d.%m.%Y")}.docx'  
 doc.save(file\_name)  
 print(f"DOCX saved as {file\_name}")  
 except Exception as e:  
 print(f"Error: {e}")  
 return redirect('competition', competition.slug)  
  
  
def generate\_meets\_docx\_for\_competition(request, id\_category):  
 try:  
 category = get\_object\_or\_404(Category, id=id\_category)  
 competition = category.id\_competition  
 competition\_name = competition.name\_competition.replace('"', '')  
  
 doc = Document()  
 doc.styles['Normal'].font.name = 'Times New Roman'  
 doc.styles['Normal'].font.size = Pt(12)  
 doc.styles['Normal'].element.rPr.rFonts.set(qn('w:eastAsia'), 'Times New Roman')  
  
 # Title  
 title = doc.add\_heading(level=1)  
 run = title.add\_run(f" Жеребьевка {competition\_name}")  
 run.font.size = Pt(14)  
 run.font.color.rgb = RGBColor(0, 0, 0)  
 title.alignment = WD\_ALIGN\_PARAGRAPH.CENTER  
  
 # Category  
 subheading = doc.add\_heading(f"Категория: вес {category.id\_weight} кг, возраст {category.id\_age}", level=2)  
 subheading\_run = subheading.runs[0]  
 subheading\_run.font.color.rgb = RGBColor(0, 0, 0)  
  
 # Meets  
 subheading = doc.add\_heading('Встречи', level=2)  
 subheading\_run = subheading.runs[0]  
 subheading\_run.font.color.rgb = RGBColor(0, 0, 0)  
  
 applications = Application.objects.filter(category\_id=category, status\_application=True)  
 applications\_list = list(applications)  
 participant\_count = next\_power\_of\_two(len(applications\_list))  
  
 total\_rounds = power\_of\_two\_exponent(participant\_count)  
 current\_start\_index = 0  
 match\_number = 0  
  
 current\_round\_meets = []  
 for round\_index in range(total\_rounds - 1, 0, -1):  
 meets = list(Meet.objects.filter(id\_category\_id=category).order\_by('sequence\_number'))  
 matches\_in\_round = 2 \*\* round\_index  
 for j in range(current\_start\_index, current\_start\_index + matches\_in\_round):  
 if j < len(meets):  
 current\_round\_meets.append(meets[j])  
 match\_number += 1  
 break  
  
 table = doc.add\_table(rows=1, cols=3)  
 table.style = 'Table Grid'  
 table.alignment = WD\_TABLE\_ALIGNMENT.CENTER  
  
 table.columns[0].width = Inches(0.5)  
 table.columns[1].width = Inches(2.5)  
 table.columns[2].width = Inches(2.5)  
  
 hdr\_cells = table.rows[0].cells  
 hdr\_cells[0].text = '№'  
 hdr\_cells[1].text = 'Спортсмен в белый'  
 hdr\_cells[2].text = 'Спортсмен в синий'  
  
 hdr\_cells[0].width = Inches(0.5)  
 hdr\_cells[1].width = Inches(2.5)  
 hdr\_cells[2].width = Inches(2.5)  
  
 for meet in current\_round\_meets:  
 row\_cells = table.add\_row().cells  
 row\_cells[0].text = str(meet.sequence\_number)  
 row\_cells[1].text = (f"{meet.id\_white.last\_name} {meet.id\_white.first\_name} "  
 f"{meet.id\_white.expansionuser.patronymic}") if meet.id\_white else 'Нет соперника'  
 row\_cells[2].text = (f"{meet.id\_blue.last\_name} {meet.id\_blue.first\_name} "  
 f"{meet.id\_blue.expansionuser.patronymic}") if meet.id\_blue else 'Нет соперника'  
  
 row\_cells[0].width = Inches(0.5)  
 row\_cells[1].width = Inches(2.5)  
 row\_cells[2].width = Inches(2.5)  
  
 doc\_date = datetime.now().strftime('%d.%m.%Y')  
 doc.add\_paragraph(f"Дата формирования документа: {doc\_date}")  
  
 doc.add\_paragraph(f"Главный судья: \_\_\_\_\_\_\_\_\_\_\_\_ {request.user.last\_name} {request.user.first\_name} {request.user.expansionuser.patronymic} ")  
  
 file\_name = f'{competition\_name}\_жеребьевка.docx'  
 doc.save(file\_name)  
 print(f"DOCX saved as {file\_name}")  
 return redirect('meets', comp\_slug=meet.id\_category.id\_competition.slug, id\_category=meet.id\_category.pk)  
 except Exception as e:  
 print(f"Error: {e}")  
 return redirect('meets', comp\_slug=meet.id\_category.id\_competition.slug, id\_category=meet.id\_category.pk)  
  
  
def generate\_meets2\_docx\_for\_competition(request, id\_category):  
 try:  
 category = get\_object\_or\_404(Category, id=id\_category)  
 competition = category.id\_competition  
 competition\_name = competition.name\_competition.replace('"', '')  
  
 doc = Document()  
 doc.styles['Normal'].font.name = 'Times New Roman'  
 doc.styles['Normal'].font.size = Pt(12)  
 doc.styles['Normal'].element.rPr.rFonts.set(qn('w:eastAsia'), 'Times New Roman')  
  
 # Title  
 title = doc.add\_heading(level=1)  
 run = title.add\_run(f" Итоги {competition\_name}")  
 run.font.size = Pt(14)  
 run.font.color.rgb = RGBColor(0, 0, 0)  
 title.alignment = WD\_ALIGN\_PARAGRAPH.CENTER  
  
 # Category  
 subheading = doc.add\_heading(f"Категория: вес {category.id\_weight} кг, возраст {category.id\_age}", level=2)  
 subheading\_run = subheading.runs[0]  
 subheading\_run.font.color.rgb = RGBColor(0, 0, 0)  
  
  
  
 applications = Application.objects.filter(category\_id=category, status\_application=True)  
 meets = list(Meet.objects.filter(id\_category\_id=category).order\_by('sequence\_number'))  
  
 applications\_list = list(applications)  
 participant\_count = next\_power\_of\_two(len(applications\_list))  
  
 rounds = {}  
 total\_rounds = power\_of\_two\_exponent(participant\_count)  
 current\_start\_index = 0  
 match\_number = 0  
 offset = 0  
  
 for round\_index in range(total\_rounds - 1, 0, -1):  
 meets = list(Meet.objects.filter(id\_category\_id=category).order\_by('sequence\_number'))  
 matches\_in\_round = 2 \*\* round\_index  
 current\_round\_meets = []  
  
 for j in range(current\_start\_index, current\_start\_index + matches\_in\_round):  
 if j < len(meets):  
 current\_round\_meets.append(meets[j])  
 match\_number += 1  
  
 match\_number -= offset  
 rounds[total\_rounds - round\_index - 1] = current\_round\_meets  
  
 offset = 0  
 previous\_round\_index = total\_rounds - round\_index - 1  
 for j in range(0, current\_start\_index + matches\_in\_round, 2):  
 if j < len(rounds.get(previous\_round\_index, [])):  
 meet1 = rounds[previous\_round\_index][j]  
 if j + 1 < len(rounds[previous\_round\_index]):  
 meet2 = rounds[previous\_round\_index][j + 1]  
 else:  
 break  
  
 if meet1.result is None or meet2.result is None:  
 match\_number += 1  
 continue  
  
 match\_number += 1  
 offset += 1  
 current\_start\_index += matches\_in\_round  
  
 sum\_matches = sum(2 \*\* i for i in range(total\_rounds))  
 print(sum\_matches == len(meets), sum\_matches, len(meets))  
 if sum\_matches == len(meets) and meets[len(meets) - 1].result is not None:  
 rounds[len(rounds)] = [meets[len(meets) - 1]]  
  
 print(meets[len(meets) - 3:len(meets)])  
  
 subheading = doc.add\_heading('Занимаемые места', level=2)  
 subheading\_run = subheading.runs[0]  
 subheading\_run.font.color.rgb = RGBColor(0, 0, 0)  
  
 table = doc.add\_table(rows=1, cols=2)  
 table.style = 'Table Grid'  
 table.alignment = WD\_TABLE\_ALIGNMENT.CENTER  
  
 table.columns[0].width = Inches(0.5)  
 table.columns[1].width = Inches(2.5)  
  
 hdr\_cells = table.rows[0].cells  
 hdr\_cells[0].text = 'Место №'  
 hdr\_cells[1].text = 'Фио участнника'  
  
 hdr\_cells[0].width = Inches(1.5)  
 hdr\_cells[1].width = Inches(2.5)  
  
 row\_cells = table.add\_row().cells  
 row\_cells[0].text = '1'  
  
 row\_cells[1].text = (f"{meets[len(meets) - 1].id\_white.last\_name} "  
 f"{meets[len(meets) - 1].id\_white.first\_name} "  
 f"{meets[len(meets) - 1].id\_white.expansionuser.patronymic}") if meets[  
 len(meets) - 1].result else (f"{meets[len(meets) - 1].id\_blue.last\_name} "  
 f"{meets[len(meets) - 1].id\_blue.first\_name} "  
 f"{meets[len(meets) - 1].id\_blue.expansionuser.patronymic}")  
  
 row\_cells = table.add\_row().cells  
 row\_cells[0].text = '2'  
  
 row\_cells[1].text = (f"{meets[len(meets) - 1].id\_white.last\_name} "  
 f"{meets[len(meets) - 1].id\_white.first\_name} "  
 f"{meets[len(meets) - 1].id\_white.expansionuser.patronymic}") if not meets[  
 len(meets) - 1].result else (f"{meets[len(meets) - 1].id\_blue.last\_name} "  
 f"{meets[len(meets) - 1].id\_blue.first\_name} "  
 f"{meets[len(meets) - 1].id\_blue.expansionuser.patronymic}")  
  
 row\_cells = table.add\_row().cells  
 row\_cells[0].text = '3'  
  
 row\_cells[1].text = (f"{meets[len(meets) - 2].id\_white.last\_name} "  
 f"{meets[len(meets) - 2].id\_white.first\_name} "  
 f"{meets[len(meets) - 2].id\_white.expansionuser.patronymic}") if not meets[  
 len(meets) - 2].result else (f"{meets[len(meets) - 2].id\_blue.last\_name} "  
 f"{meets[len(meets) - 2].id\_blue.first\_name} "  
 f"{meets[len(meets) - 2].id\_blue.expansionuser.patronymic}")  
  
 row\_cells = table.add\_row().cells  
 row\_cells[0].text = '3'  
  
 row\_cells[1].text = (f"{meets[len(meets) - 3].id\_white.last\_name} "  
 f"{meets[len(meets) - 3].id\_white.first\_name} "  
 f"{meets[len(meets) - 3].id\_white.expansionuser.patronymic}") if not meets[  
 len(meets) - 3].result else (f"{meets[len(meets) - 3].id\_blue.last\_name} "  
 f"{meets[len(meets) - 3].id\_blue.first\_name} "  
 f"{meets[len(meets) - 3].id\_blue.expansionuser.patronymic}")  
  
 subheading = doc.add\_heading('Встречи', level=2)  
 subheading\_run = subheading.runs[0]  
 subheading\_run.font.color.rgb = RGBColor(0, 0, 0)  
  
 table = doc.add\_table(rows=1, cols=6)  
 table.style = 'Table Grid'  
 table.alignment = WD\_TABLE\_ALIGNMENT.CENTER  
  
 table.columns[0].width = Inches(0.5)  
 table.columns[1].width = Inches(2.5)  
 table.columns[2].width = Inches(2.5)  
 table.columns[3].width = Inches(2.5)  
 table.columns[4].width = Inches(2.5)  
 table.columns[5].width = Inches(2.5)  
  
 hdr\_cells = table.rows[0].cells  
 hdr\_cells[0].text = '№'  
 hdr\_cells[1].text = 'Спортсмен в белый'  
 hdr\_cells[2].text = 'Спортсмен в синий'  
 hdr\_cells[3].text = 'Оценки и время'  
 hdr\_cells[4].text = 'Судья'  
 hdr\_cells[5].text = 'Резулитат'  
  
 hdr\_cells[0].width = Inches(0.5)  
 hdr\_cells[1].width = Inches(2.5)  
 hdr\_cells[2].width = Inches(2.5)  
 hdr\_cells[3].width = Inches(0.5)  
 hdr\_cells[4].width = Inches(2.5)  
 hdr\_cells[5].width = Inches(2.5)  
  
 for meet in meets:  
 row\_cells = table.add\_row().cells  
 row\_cells[0].text = str(meet.sequence\_number)  
 row\_cells[1].text = (f"{meet.id\_white.last\_name} {meet.id\_white.first\_name} "  
 f"{meet.id\_white.expansionuser.patronymic}") if meet.id\_white else 'Нет соперника'  
 row\_cells[2].text = (f"{meet.id\_blue.last\_name} {meet.id\_blue.first\_name} "  
 f"{meet.id\_blue.expansionuser.patronymic}") if meet.id\_blue else 'Нет соперника'  
 row\_cells[3].text = f'{meet.duration}мин.\n{meet.assessments}' if meet.duration is not None else "-"  
 row\_cells[4].text = (f"{meet.id\_judge.user.last\_name} {meet.id\_judge.user.first\_name} "  
 f"{meet.id\_judge.patronymic}") if meet.id\_judge is not None else "-"  
 row\_cells[5].text = 'Победил белый спортсмен' if meet.result else 'Победил синий спортсмен'  
  
 row\_cells[0].width = Inches(0.5)  
 row\_cells[1].width = Inches(2.5)  
 row\_cells[2].width = Inches(2.5)  
 row\_cells[3].width = Inches(0.5)  
 row\_cells[4].width = Inches(2.5)  
 row\_cells[5].width = Inches(2.5)  
  
 doc\_date = datetime.now().strftime('%d.%m.%Y')  
 doc.add\_paragraph(f"Дата формирования документа: {doc\_date}")  
  
 doc.add\_paragraph(f"Главный судья: \_\_\_\_\_\_\_\_\_\_\_\_ {request.user.last\_name} {request.user.first\_name} {request.user.expansionuser.patronymic}")  
  
 file\_name = f'{competition\_name}\_итоги.docx'  
 doc.save(file\_name)  
 print(f"DOCX saved as {file\_name}")  
 return redirect('meets', comp\_slug=competition.slug, id\_category=category.pk)  
 except Exception as e:  
 print(f"Error: {e}")  
 return redirect('meets', comp\_slug=competition.slug, id\_category=category.pk)

from django.urls import path  
  
from .views import \*  
  
  
urlpatterns = [  
 path('signup/', register, name='sign\_un'),  
 path('', user\_login, name='login'),  
 path('sign\_out/', user\_logout, name='sign\_out'),  
 path('competitions/', show\_competitions, name='show\_competitions'),  
 path('profile/<int:id\_user>', edit\_profile, name='profile'),  
 path('add\_competition/', add\_competition, name='add\_competition'),  
 path('suitable\_participants/<int:id\_category>/', suitable\_participants, name='suitable\_participants'),  
 path('suitable\_participants/<int:id\_category>/participant/<int:id\_participant>', suitable\_participants, name='suitable\_participants\_with'),  
 path('competition/<slug:comp\_slug>', show\_competition, name='competition'),  
 path('competition/<slug:comp\_slug>/choose-judges/', choose\_judges, name='choose\_judges'),  
 path('competition/<slug:comp\_slug>/category/<int:id\_category>', category, name='category'),  
 path('competition/<slug:comp\_slug>/category/<int:id\_category>/meets/', meets, name='meets'),  
 path('competition/<slug:comp\_slug>/create\_category/', create\_category, name='create\_category'),  
 path('competition/<slug:comp\_slug>/category/<int:id\_category>/application/<int:id\_application>', reject\_application, name='reject\_application'),  
 path('competition/<slug:comp\_slug>/category/<int:id\_category>/meets/draw/', draw\_meet, name='draw\_meet'),  
 path('meet/<int:meet\_id>/', meet\_detail, name='meet\_detail'),  
 path('meet/<int:meet\_id>/edit/', edit\_meet, name='edit\_meet'),  
 path('generate\_doc\_cop/<slug:comp\_slug>/', generate\_doc\_view, name='generate\_pdf'),  
 path('generate\_doc\_meet1/<int:id\_category>/', generate\_meets\_docx\_for\_competition, name='meets\_docx'),  
 path('generate\_doc\_meet2/<int:id\_category>/', generate\_meets2\_docx\_for\_competition, name='meets\_docx2'),  
]

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 ГОСТ Р 7.0.100-2018. Библиографическая запись. Библиографическое описание. Общие требования и правила составления : национальный стандарт Российской Федерации : дата введения 2019-07-01 - https://internet-law.ru/gosts/gost/70535/ (дата обращения: 12.01.2022). – Текст : электронный.

2 ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения– Введ. 1992-01-01. – М.: Изд-во стандартов, 1992. – 24 с. – Текст: электронный.

3 ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными: межгосударственный стандарт: дата введения 2001-07-01 - https://docs.cntd.ru/document/1200017662?ysclid=lrq6wn65z5753511468 – Текст: электронный.

4 ГОСТ Р 51583-2014. Защита информации. Порядок создания автоматизированных систем в защищенном исполнении. Общие положения. [Текст]. – Введ. 2014- 09 - 01. – М.: Стандартинформ, 2007.

5 ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. – Введ. 1990 – 01 - 01. – М.: Стандартинформ, 2007. – Текст: электронный.

6 Русско-английский англо-русский словарь 45 тысяч слов. — Киров: ЛадКом, 2019 — 541 c. – Текст: непосредственный

7 Мартин Р. Чистый код: создание, анализ и рефакторинг. - Питер, 2018. - 464 с. – ISBN 978-5-44-610960-9. – Текст: непосредственный.

8 Роббинс, Дж. Веб-дизайн для начинающих. HTML, CSS, JavaScript и веб-графика / Роббинс Дженнифер. СПб. : БХВ-Петербург, 2021. – 956 с. – ISBN - 978-5-9775-4050-6. – Текст: непосредственный

9 Васильев, А. Программирование на Python в примерах и задачах / А. Васильев., – Москва: Эксмо, 2021. – 616с. – (Российский компьютерный бестселлер). – ISBN 978-5-04-103199-2. – Текст: непосредственный.

10 Шварц Б. MySQL по максимуму. - 3-е изд. - Питер, 2018. - 864 с. – ISBN 978-5-44-610696-7. – Текст: непосредственный.

11 Ботрос, С. MySQL по максимуму / C. Ботрос, Д. Тинли., – 4-е изд., перераб. и доп. – Питер, 2023. – 432 с. – ISBN 978-5-4461-2261-5. – Текст: электронный.

12 Изменения в правилах / [Электронный ресурс] // ФЕДЕРАЦИЯ ДЗЮДО РОССИИ : [сайт]. — URL: https://judo.ru/video/9001 (дата обращения: 01.04.2024).

13 Django 3 для python (уроки) / [Электронный ресурс] // youtube : [сайт]. — URL: https://www.youtube.com/watch?v=FyTL1bnUx5I&list=PLA0M1Bcd0w8xO\_39zZll2u1lz\_Q-Mwn1F&index=1 (дата обращения: 21.03.2024).

14 Документация Django. – 2024.: Текст: электронный. – URL: https://docs.djangoproject.com/en/5.0/ (дата обращения: 26.03.2024).

15 Документация Python. – 2024. Текст: электронный. – URL: https://docs.python.org/3/index.html (дата обращения: 26.03.2024).

16 Документация HTML. – 2024. Текст : электронный. – URL: https://developer.mozilla.org/ru/docs/Web/HTML (дата обращения: 26.03.2024).

17 Django - Use different MySQL users for manage.py vs serving app / [Электронный ресурс] // stackoverflow : [сайт]. — URL: stackoverflow (дата обращения: 26.03.2024).

18 Customizing authentication in Django / [Электронный ресурс] // djangoproject : [сайт]. — URL: https://docs.djangoproject.com/en/5.0/topics/auth/customizing/ (дата обращения: 26.03.2024).

19 Ways to Extend the Existing User Model / [Электронный ресурс] // simpleisbetterthancomplex : [сайт]. — URL: https://simpleisbetterthancomplex.com/tutorial/2016/07/22/how-to-extend-django-user-model.html (дата обращения: 10.04.2024).

20 Элементы для ввода // metanit URL: https://metanit.com/web/html5/3.9.php (дата обращения: 15.042

1. Первичный ключ [↑](#footnote-ref-1)
2. Вторичный ключ [↑](#footnote-ref-2)