Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора  по учебной работе  З.З. Курмашева  « » 2023 г. |

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ

ДЛЯ ФОРМИРОВАНИЯ ЗАЯВКИ НА УЧАСТИЕ

В СОРЕВНОВАНИЯХ ПО ДЗЮДО

Пояснительная записка к курсовому проекту

МДК 011.01 Технология разработки и защиты баз данных

|  |  |
| --- | --- |
|  | Руководитель проекта  Р.Ф.Каримова  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  | Студент гр. 20П-1  Ш.Н.Кулиев  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |

2023

Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора  по учебной работе  З.З. Курмашева  « » 2023 г. |

ЗАДАНИЕ

на курсовой проект студенту дневного отделения, группы 20П-1, специальности 09.02.07 Информационные системы и программирование.

Фамилия, имя, отчество: Кулиев Шамиль Назарович

Тема курсового проекта: «Проектирование и разработка базы данных для формирования заявки на участие в соревнованиях по дзюдо».

Текст задания:

при выполнении курсового проекта должны быть решены следующие задачи:

1. спроектировать базу данных;
2. разработать структуру программы;
3. реализовать функции организации соревнований по дзюдо, регистрация участников, подачи заявки на участие, формирование таблиц участников.

В результате выполнения курсового проекта должны быть представлены:

1. пояснительная записка, состоящая из следующих разделов:

Введение

1 Постановка задачи

2 Экспериментальный раздел

Заключение

Приложения

Список сокращений

Список источников

1. электронный носитель, содержащий разработанный программный продукт;
2. презентация курсового проекта в электронном виде.

Список рекомендуемых источников:

1. Голицына, О. Л. Основы проектирования баз данных: учеб. пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. — 2-е изд., перераб. и доп. — Москва : ФОРУМ : ИНФРА-М, 2019. — 416 с.: ил. — (Cреднее профессиональное образование). – ISBN 978-5-91134-655-3. - Текст: электронный. - URL: https://znanium.com/catalog/product/1018906 (дата обращения: 22.03.2023). –Режим доступа: по подписке.
2. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1: учебник для среднего профессионального образования / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. И доп. — Москва: Издательство Юрайт, 2023. — 310 с. — (Профессиональное образование). — ISBN 978-5-534-11626-7. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/518510 (дата обращения: 14.03.2023)
3. Гордеев, С. И. Организация баз данных в 2 ч. Часть 2: учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — Москва: Издательство Юрайт, 2023. — 513 с. — (Высшее образование). — ISBN 978-5-534-04470-6. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/515097 (дата обращения: 14.03.2023).
4. Карпова, И. П. Базы данных: учебное пособие / И. П. Карпова. – Санкт-Петербург: Питер, 2021. - 240 с. - (Серия «Учебное пособие»). - ISBN 978-5-4461- 9681-4. - Текст: электронный. - URL: https://znanium.com/catalog/product/1857026 (дата обращения: 22.03.2023). – Режим доступа: по подписке.
5. Новиков, Б. А. Основы технологий баз данных : учебное пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева ; под ред. Е. В. Рогова. — 2-е изд. — Москва : ДМК Пресс, 2020. - 582 с. - ISBN 978-5-97060-841-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/1210665 (дата обращения: 22.03.2023). – Режим 62 доступа: по подписке.
6. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: учебное пособие / Г.Н. Федорова. — Москва: КУРС : ИНФРА-М, 2023. — 336 с. — (Среднее профессиональное образование). - ISBN 978-5-906818-41-6. - Текст: электронный. - URL: https://znanium.com/catalog/product/1896457 (дата обращения: 22.03.2023). – Режим доступа: по подписке.
7. Документация по MySQL. URL: https://dev.mysql.com/doc/ (дата обращения: 14.03.2023).
8. Metanit: Сайт о программировании. URL: https://metanit.com/ (дата обращения: 14.03.2023).

Задание к выполнению получил «16» января 2023 г.

Студент Кулиев Шамиль Назарович

Срок окончания «31» мая 2023 г.

Руководитель курсового проекта Р.Ф. Каримова

Задание рассмотрено на заседании цикловой комиссии информатики

протокол № 4 от «9» января 2023 г.

Председатель цикловой комиссии информатики О.В.Фатхулова

Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

ЗАКЛЮЧЕНИЕ

на курсовой проект

Студент Кулиев Шамиль Назарович

Группа 20П-1

Специальность 09.02.07 Информационные системы и программирование

Тема Проектирование и разработка базы данных для формирования заявки на участие в соревнованиях по дзюдо

Объем курсового проекта:

количество листов пояснительной записки

количество листов графической части

Заключение о степени соответствия заданию на курсовое проектирование

Характеристика качеств, проявленных студентом при работе над проектом: самостоятельность, дисциплинированность, умение планировать работу и пользоваться литературным материалом и т.д.

Положительные стороны курсового проекта

Недостатки курсового проекта

Характеристика общетехнической и специальной подготовки студента

Заключение и предлагаемая оценка за курсовой проект

Руководитель курсового проекта Каримова Резида Флюновна

«\_\_» 2023 г.

Подпись

АННОТАЦИЯ

Пояснительная записка к курсовому проекту содержит постановку и программу решения задачи «Проектирование и разработка базы данных для формирования заявки на участие в соревнованиях по дзюдо».

Программа JudoArena.exe написана на языке C# с использованием паттерна MVVM в среде программирования Visual Studio 2022 с использованием системы управления базой данных MySQL Workbench 8.0 предназначена для работы в операционной системе MS 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 Описание входной информации | 9 |
| 1.3 Описание выходной информации | 9 |
| 1.4 Концептуальное моделирование | 9 |
| 1.5 Логическое моделирование | 10 |
| 1.6 Описание структуры базы данных | 12 |
| 1.7 Контрольный пример | 16 |
| 1.8 Общие требования к программному продукту | 16 |
| 2 Экспериментальный раздел | 18 |
| 2.1 Описание программы | 18 |
| 2.2 Протокол тестирования программного продукта | 25 |
| 2.3 Руководство пользователя | 35 |
| Заключение | 40 |
| Приложения | 41 |
| Список сокращений | 68 |
| Список использованных источников | 69 |

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 участника.

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 участника в каждой весовой и возрастной категории. Весовые категории: участники 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 участники 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 участников, порядок проведения встреч. Ключевые этапы это – организация соревнования, регистрация участников, 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 участников с документацией, имеет большую актуальность. Такое приложение может предоставлять электронную форму для заполнения данных об участниках 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 курсового проекта заключается в разработке приложения, которое будет способствовать ускорению и упрощению взаимодействия судей и участников с документацией на соревнованиях по дзюдо.

Задачами курсового проекта являются:

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

1 Постановка задачи

* 1. Описание предметной области

Требуется разработать информационную систему для регистрации тренеров и участников, подачи заявки на участие в соревнованиях по дзюдо и формирование таблиц участников.

Данная информационная система предполагает наличие трех групп пользователей: участник, тренер и судья.

Участник имеет возможность:

* зарегистрироваться и войти;
* просматривать списки запланированные и проведённые соревнования;
* просматривать подробную информацию о соревновании, если он в нем участвует.

В базе данных должны храниться следующая информация об участнике:

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

Участник должен соответствовать своей возрастной и весовой категории.

Тренер имеет возможность:

* зарегистрироваться и войти;
* просматривать списки запланированные и проведённые соревнования;
* просматривать подробную информацию о всех соревновании;
* подавать заявку на участие.

В базе данных должны храниться следующая информация о тренере:

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

Судья имеет возможность:

* войти;
* просматривать подробную информацию о всех соревновании;
* запланировать новое соревнование;
* сформировать таблицы участников.

В базе данных должны храниться следующая информация о судье:

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

Авторизация.

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

Список соревнований.

После успешной авторизации или регистрации пользователь переходит к странице списка соревнований. Где представлена следующая информация:

* название соревнования;
* дата проведения;
* адрес;
* организатор.

На странице представлен функционал:

* выход к окну авторизации;
* поиск по названию соревнования;
* фильтрация по дате: все, запланированные и завершенные соревнования;
* сортировка по дате;
* у тренера и судьи возможность просмотра подробной информации всех соревнований, у участника только тех соревнований, в которых он участвует;
* судья может запланировать новое соревнование.

Подробно о соревновании.

Вкладка «О соревновании» представлена информация:

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

Вкладка «Участники» представлена информация:

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

Судья может сформировать отчет по соревнованию, где представлены все участники, сгруппированные по категориям.

Планирование нового соревнования.

Поля ввода:

* название соревнования;
* дата проведения;
* адрес.

Добавление весовых и возвратных категорий.

Подача заявки.

Список всех спортсменов данной категории. Возможность поиска по ФИО. Формирование и подача заявки.

1.2 Описание входной информации

Входной информацией для выполнения задачи являть данные тренеров и участников, проток соревнования, по которому заполняться запланированные соревнования.

1.3 Описание выходной информации

Выходной информацией является отчет по соревнованию.

Описание выходных документов:

* заголовок: название и номер соревнования, дата, адрес;
* тело документа: наименование категории, список и количество участников;
* окончание: количество категорий, количество всего участников, организатор, дата формирование документа.

Шаблоны выходных документов представлены в приложении А.

1.4. Концептуальное моделирование

Концептуальная модель базы данных — это некая наглядная диаграмма, нарисованная в принятых обозначениях и подробно показывающая связь между объектами и их характеристиками. Создается концептуальная модель для дальнейшего проектирования базы данных и перевод ее, например, в реляционную базу данных. На концептуальной модели в визуально удобном виде прописываются связи между объектами данных и их характеристиками.

В концептуальной модели есть принятые обозначения элементов. Сущность или объект обозначать прямоугольником, отношения обозначать ромбом, атрибуты объектов, обозначаются овалом.

Концептуальная модель базы данных представлена в рисунке 1.4.1

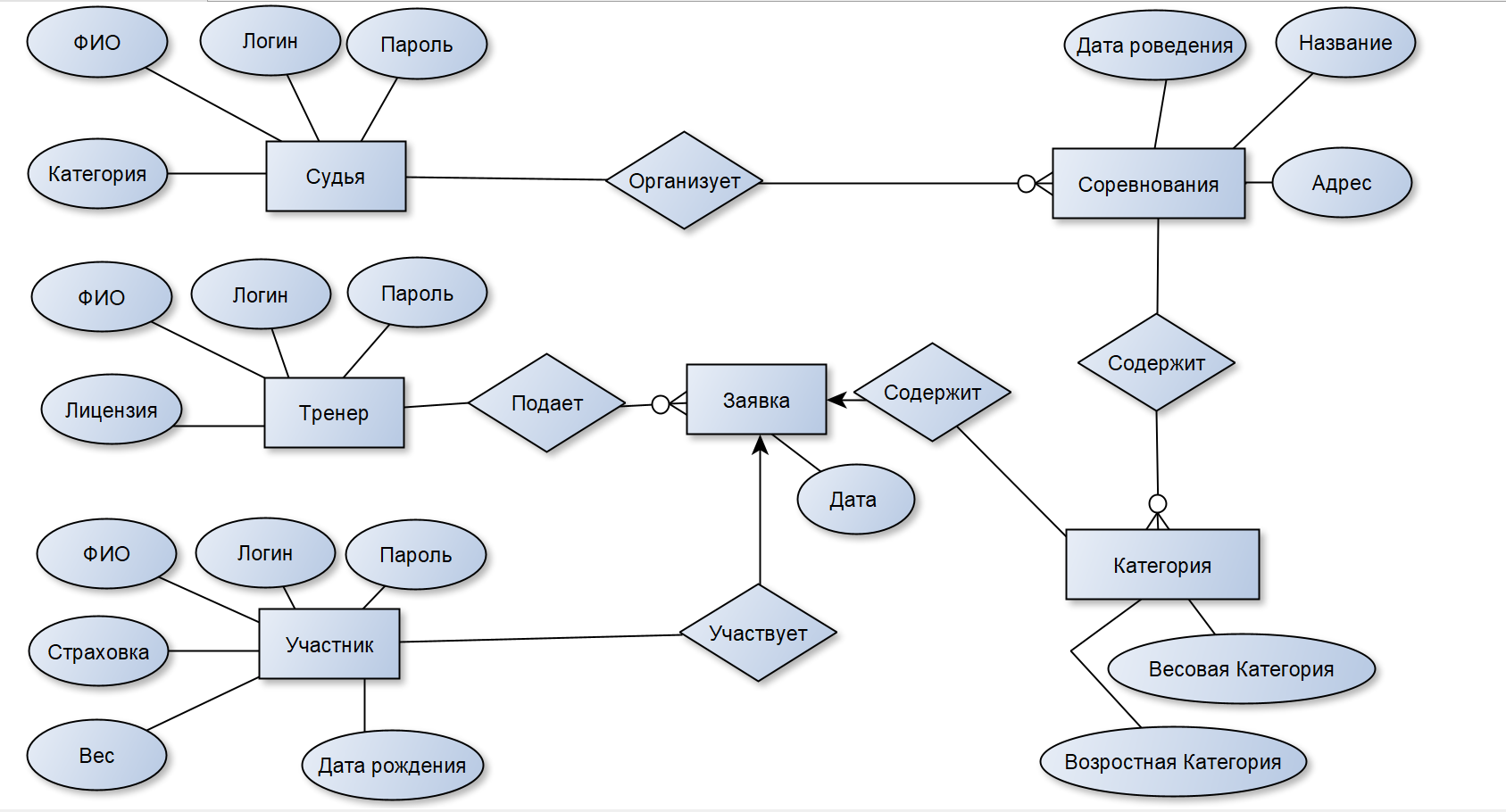


Рисунок 1.4.1 – Концептуальная модель БД

1.5. Логическое моделирование

При логическом моделировании происходит окончательное определение структуры данных, определяются ограничения, накладываемые на эти данные, целью которых является обеспечить целостность данных. Наиболее распространенной моделью данных является реляционная модель. В этой модели данных каждая сущность представляется в виде таблицы.

Логическое моделирование заключается в переходе от концептуальной модели к взаимосвязанным таблицам. Этот переход состоит из следующих шагов:

1. Преобразование сущностей:

- каждая простая сущность становится таблицей.

- каждый атрибут становится столбцом таблицы.

-добавляться уникальные индификаторы.

2. Преобразование связи:

- сущности, связанные обязательной связью один к одному можно объединить в одну таблицу.

- связи типа один к одному возможные и связи типа один ко многим реализуются путем переноса ключевых атрибутов таблиц, соответствующих сущностей, стоящих со стороны один в таблице соответствующих сущностей, стоящих со стороны многие.

- связи типа многие ко многим реализуются при помощи промежуточных таблиц, содержащих ключевые атрибуты связываемых таблиц в качестве внешних ключей.

Схема 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 данных представлена на рисунке 1.5.1.

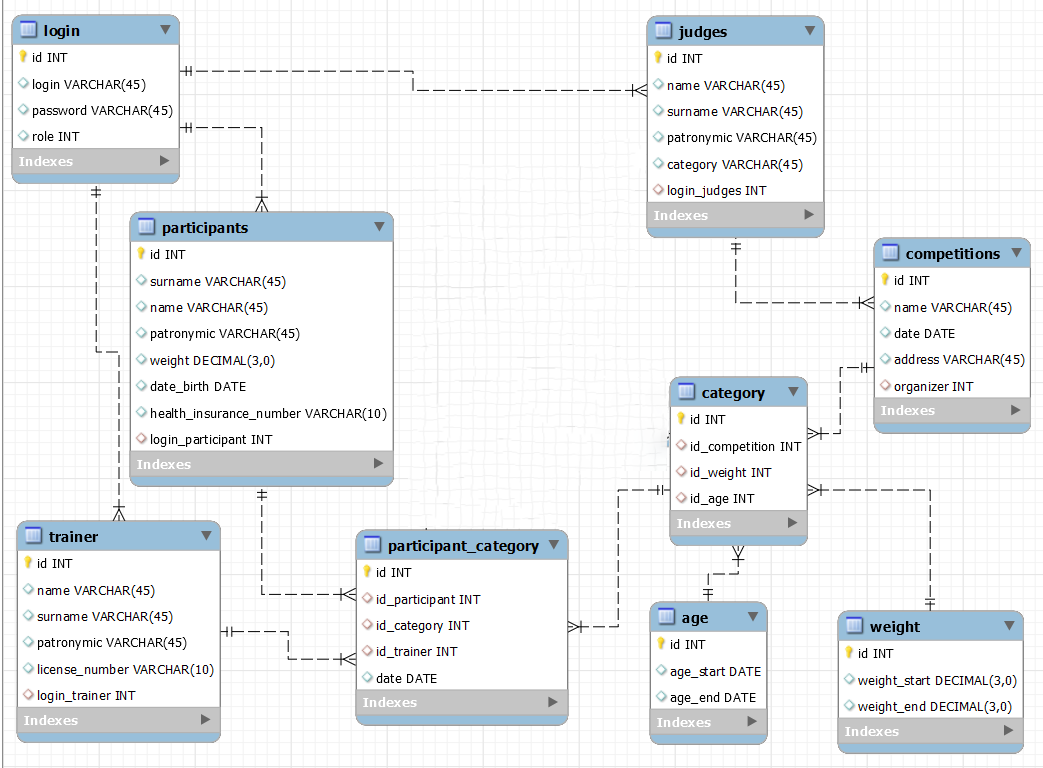


Рисунок 1.5.1 – Схема данных БД

1.6. Описание структуры базы данных

Описание структуры базы данных представлено в таблицах 1.6.1 – 1. 6.9

Таблица 1.6.1 - logit (Логины и пароли пользователей)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа  (РК-первичный, FK- внешний) |
| id | ID логина | INT |  | PK |
| login | Логин | VARCHAR | 45 |  |
| password | Пароль | VARCHAR | 45 |  |
| role | Роль пользователя | INT |  |  |

Таблица 1.6.2 - judges (Судьи)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа  (РК-первичный, FK- внешний) |
| id | ID Судьи | INT |  | PK |
| name | Имя судьи | VARCHAR | 45 |  |
| surname | Фамилия судьи | VARCHAR | 45 |  |
| patronymic | Отчество судьи | VARCHAR | 45 |  |
| category | Категория | VARCHAR | 45 |  |
| login\_judges | ID логина | INT |  | FK |

Таблица 1.6.3 - participants (Участник)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа  (РК-первичный, FK- внешний) |
| id | ID участника | INT |  | PK |
| surname | Фамилия участника | VARCHAR | 45 |  |
| name | Имя участника | VARCHAR | 45 |  |
| patronymic | Отчество участника | VARCHAR | 45 |  |
| weight | Вес | DECIMAL | 3,1 |  |
| date\_birth | Дата рождения | DATE |  |  |
| health\_insurance\_number | Номер страховки | VARCHAR | 10 |  |
| login\_participant | ID Логина | INT |  | FK |

Таблица 1.6.4 - trainer (Тренер)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа  (РК-первичный, FK- внешний) |
| id | ID тренера | INT |  | PK |
| surname | Фамилия тренера | VARCHAR | 45 |  |
| name | Имя тренера | VARCHAR | 45 |  |
| patronymic | Отчество тренера | VARCHAR | 45 |  |
| health\_insurance\_number | Номер лицензии | VARCHAR | 10 |  |
| login\_participant | ID Логина | INT |  | FK |

Таблица 1.6.5 - competitions (Соревнования)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа  (РК-первичный, FK- внешний) |
| id | ID соревнования | INT |  | PK |
| name | Название соревнования | VARCHAR | 45 |  |
| date | Дата проведения | DATE |  |  |
| address | Адрес | VARCHAR | 45 |  |
| organizer | ID организатора | INT |  | FK |

Таблица 1.6.6 – category (Категории соревнований)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа  (РК-первичный, FK- внешний) |
| id | ID категории | INT |  | PK |
| id\_competition | ID соревнования | INT |  | FK |
| id\_weight | ID веса | INT |  | FK |
| id\_age | ID возраста | INT |  | FK |

Таблица 1.6.7 – weight (Вес)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа  (РК-первичный, FK- внешний) |
| id | ID веса | INT |  | PK |
| weight\_start | Начала весовой категории | DECIMAL | 3,0 |  |
| weight\_end | Конец весовой категории | DECIMAL | 3,0 |  |

Таблица 1.6.8 – weight (Вес)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа  (РК-первичный, FK- внешний) |
| 1 | 2 | 3 | 4 | 5 |
| id | ID возраста | INT |  | PK |

Продолжение Таблица 1.6.8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| age\_start | Начала возрастной категории | DECIMAL | 3,0 |  |
| age\_end | Конец возрастной категории | DECIMAL | 3,0 |  |

Таблица 1.6.9 – participant\_category (Заявка)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа  (РК-первичный, FK- внешний) |
| id | ID заявки | INT |  | PK |
| id\_participant | ID участника | INT |  | FK |
| id\_category | ID категории | INT |  | FK |
| id\_trainer | ID тренера | INT |  | FK |
| date | Дата заявки | DATE |  |  |

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 Б.

Выходные данные для контрольных примеров показаны в приложении В.

1.8 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 Workbench.

2. Экспериментальный раздел

2.1 Описание программы

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

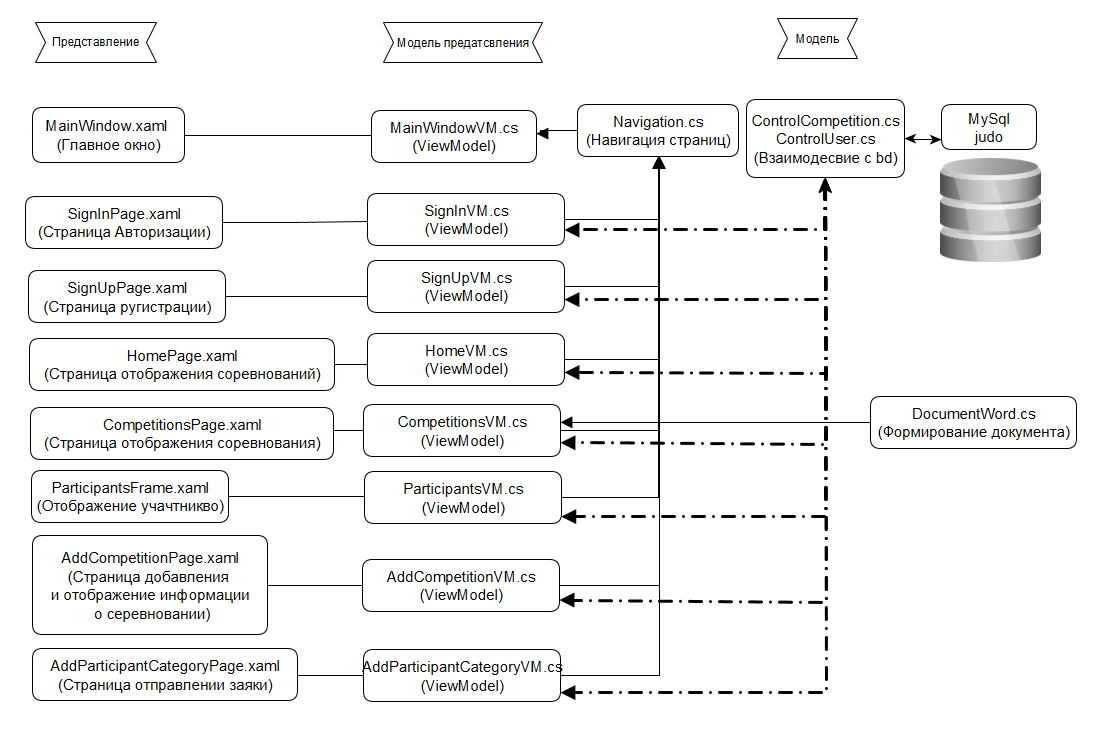


Рисунок 2.1.1 – Схема взаимодействия модулей

Таблица 2.1.1. - Описание модулей

|  |  |
| --- | --- |
| Элементы класса | Назначение |
| 1 | 2 |
| MainWindowVM.cs |  |
| public object CurrentView | Свойство отображения страницы |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| public MainWindowVM() | Конструктор инициализация элементов |
| SignInVM.cs |  |
| public DelegateCommand SignIn | Команда авторизации |
| public DelegateCommand GoSignUp | Команда перехода на регистрацию |
| public SignInVM() | Конструктор инициализация элементов |
| SignUpVM |  |
| public SignUpVM() | Конструктор инициализация элементов |
| private (bool, string) ValidateInitials(string PathName, int Length) | Функция валидации ФИО |
| private bool IsNullTrainer() | Метод проверки на пустоту тренера |
| private bool IsCheckedPassword(string Password, int Length) | Метод валидации логина и пароля |
| public DelegateCommand TrainerSignUp | Команда регистрации тренера |
| private bool IsNullParticipant() | Метод проверки на пустоту участника |
| public DelegateCommand ParticipantSignUp | Команда регистрации участника |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| public DelegateCommand GoSignIn | Команда переход на страницу авторизации |
| HomeVM.cs |  |
| public DelegateCommand GoSignIn | Команда переход на страницу авторизации |
| public DelegateCommand<object> ChangePage | Команда перехода на страницу соревнования |
| public DelegateCommand Sorting | Команда сортировки соревнований |
| public DelegateCommand AddCompetition | Команда перехода на страницу добавления соревнования |
| public HomeVM() | Конструктор инициализация элементов |
| CompetitionsVM.cs |  |
| public DelegateCommand Abuot | Команда отображения информации о соревновании |
| public DelegateCommand Participants | Команда отображения участников соревнования |
| public DelegateCommand Document | Команда формирования документа |
| public DelegateCommand GoHome | Команда перехода на страницу списка соревнований |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| public CompetitionsVM() | Конструктор инициализация элементов |
| ParticipantsVM.cs |  |
| public DelegateCommand<object> CategorySelection | Команда переключения отображения весовой категории |
| public DelegateCommand AddParticipantCategory | Команда перехода на формирования заявки |
| public ParticipantsVM() | Конструктор инициализация элементов |
| AddСompetitionVM.cs |  |
| public DelegateCommand GoHome | Команда перехода на страницу списка соревнований |
| private bool IsChecedNull() | Метод проверки на пустоту полей |
| public DelegateCommand AddC | Команда добавления соревнования |
| public DelegateCommand AddWeight | Команда добавления весовой категории |
| public DelegateCommand<object> DeleteWeight | Команда удаления весовой категории |
| public DelegateCommand AddAge | Команда добавления возрастной категории |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| public DelegateCommand<object> DeleteAge | Команда удаления возрастной категории |
| public AddСompetitionVM() | Конструктор инициализация элементов |
| AddParticipantCategoryVM.cs |  |
| public DelegateCommand Sorting | Команда сортировки участников |
| public DelegateCommand<object> Add | Команда добавления участников |
| public DelegateCommand<object> Delete | Команда удаления участников |
| public DelegateCommand Get | Команда отправки заявки |
| public DelegateCommand GoHome | Команда перехода на страницу списка соревнований |
| public AddParticipantCategoryVM() | Конструктор инициализация элементов |
| ControlCompetition.cs |  |
| public List<Competition> GetCompetitionList() | Метод получения соревнований из бд |
| public Competition AddCompetition(string name, DateOnly date, string address, int organizer) | Метод добавления соревнований в бд |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| public Weight WeightSearch(decimal start, decimal end) | Метод получения веса из бд |
| public void AddWeight(decimal start, decimal end) | Метод добавления веса в бд |
| public Age AgeSearch(DateOnly? start, DateOnly? end) | Метод получения возраста из бд |
| public void AddAge(DateOnly? start, DateOnly? end) | Метод добавления возраста в бд |
| public void AddCategory(int IdCompetition, List<Weight> weights, List<Age> ages) | Метод добавления категории в бд |
| public List<Category> GetCategory(int IdCompetition) | Метод получения категории из бд |
| public List<ParticipantCategory> GetParticipantCategory(int IdCategory) | Метод получения заявок из бд |
| public ParticipantCategory ParticipantCategorySearch(int Id) | Метод получения заявки из бд |
| public void SaveContext() | Метод сохранения изменений |
| public List<Participant> GetParticipant | Метод получения участников из бд |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| public void AddParticipantCategorie(int IdTrainer, int IdCategory, ObservableCollection<Participant> Participants) | Метод добавления заявок в бд |
| public List<ParticipantCategory> GetParticipantCategory(int IdParticipant, bool flag) | Метод получения заявок в бд |
| ControlUser.cs |  |
| public Participant AddParticipant(string Surname, string Name, string Patronymic, decimal Weight,  DateOnly DateBirth, string HealthInsuranceNumber, string login, string Password) | Метод добавления Участника |
| public Trainer AddTrainer(string Surname, string Name, string Patronymic, string License, string Login, string Password) | Метод добавления Тренера |
| public Login IsCheckedLogin(string login) | Метод правки существования логина |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| public async Task<Trainer> GetTrainer(int IDLogin) | Метод получения тренера |
| public async Task<Participant> GetParticipant(int IDLogin) | Метод получения участника |
| public async Task<Judge> GetJudges(int IDLogin) | Метод получения судьи |
| public async Task<Login> GetUserByLogin(string login) | Метод получения логина |
| DocumentWord.cs |  |
| public void GenerateWord(Competition competition) | Метод генерации документа |

Код программы представлен в приложении Г.

2.2 Протокол тестирования программного продукта

В протоколе тестирования отражаются:

− тестирование на корректных данных;

− тестирование на некорректных данных;

− тестировании продукта на данных контрольного примера.

Тестирование на авторизацию при корректных данных, ожидается переход на страницу списка соревнования (рисунок 2.2.1 и рисунок 2.2.2).

Тестирование регистрации тренера при корректных данных, ожидается сообщение «Успешно зарегистрировались» (рисунок 2.2.3).



Рисунок 2.2.1 – Ввод корректных данных в авторизации.



Рисунок 2.2.2 – Переход на страницу отображения списка соревнований.

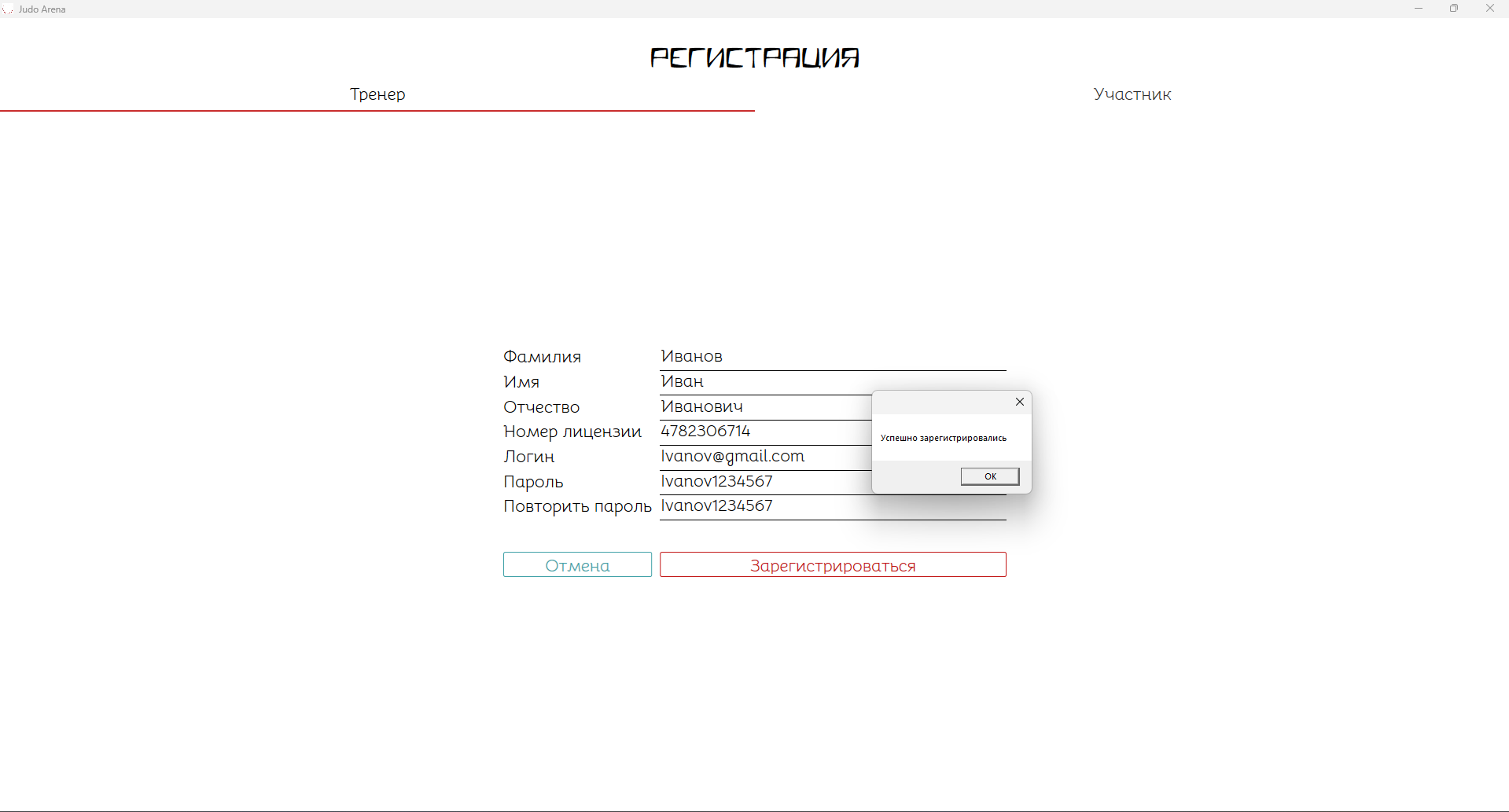


Рисунок 2.2.3 – Успешная регистрация тренера.

Тестирование регистрации участника при корректных данных, ожидается сообщение «Успешно зарегистрировались» (рисунок 2.2.4).

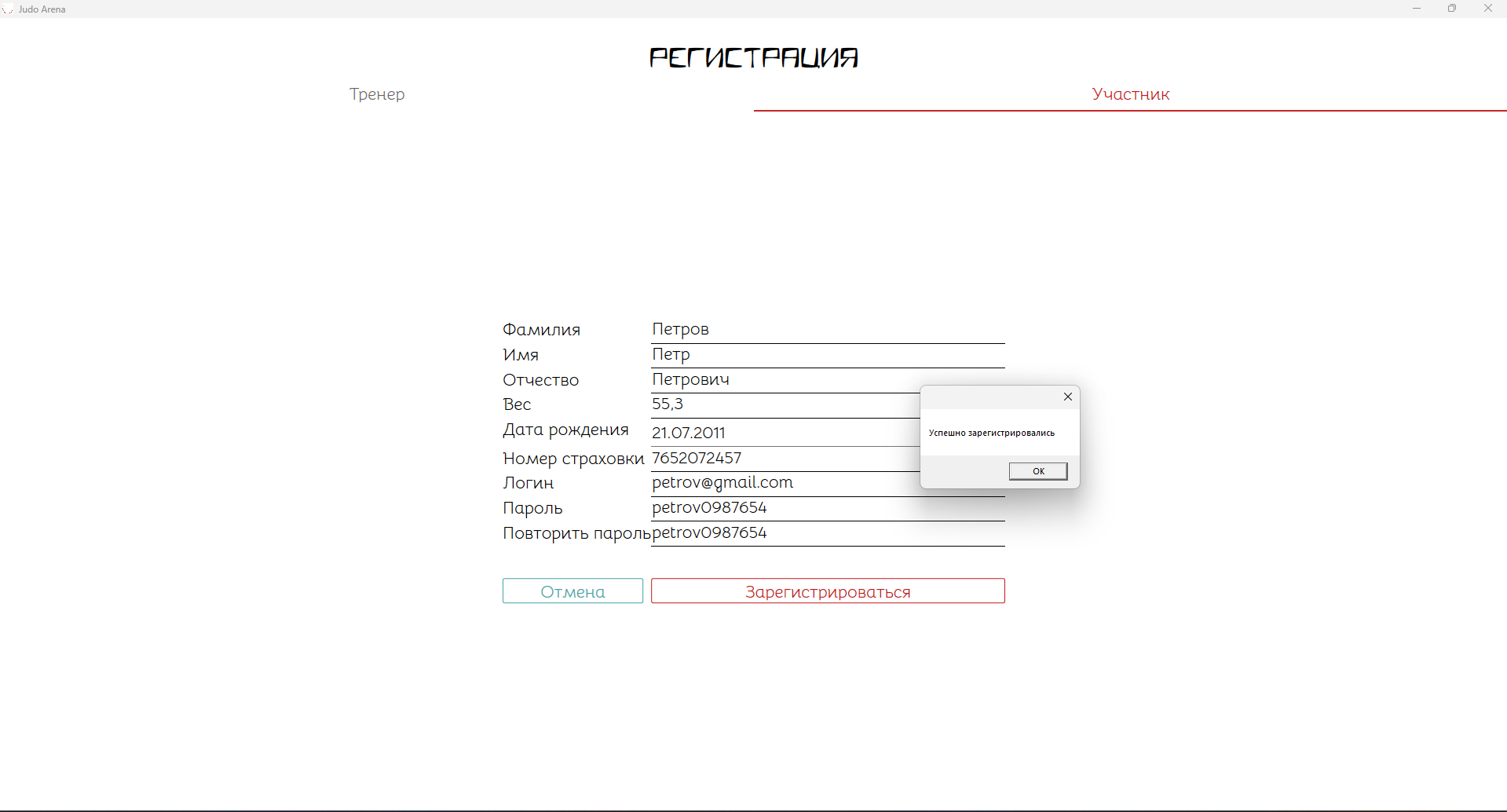


Рисунок 2.2.4 – Успешная регистрация участника.

Тестирование добавление соревнования, ожидается сообщение «Добавлено» (рисунок 2.2.5).

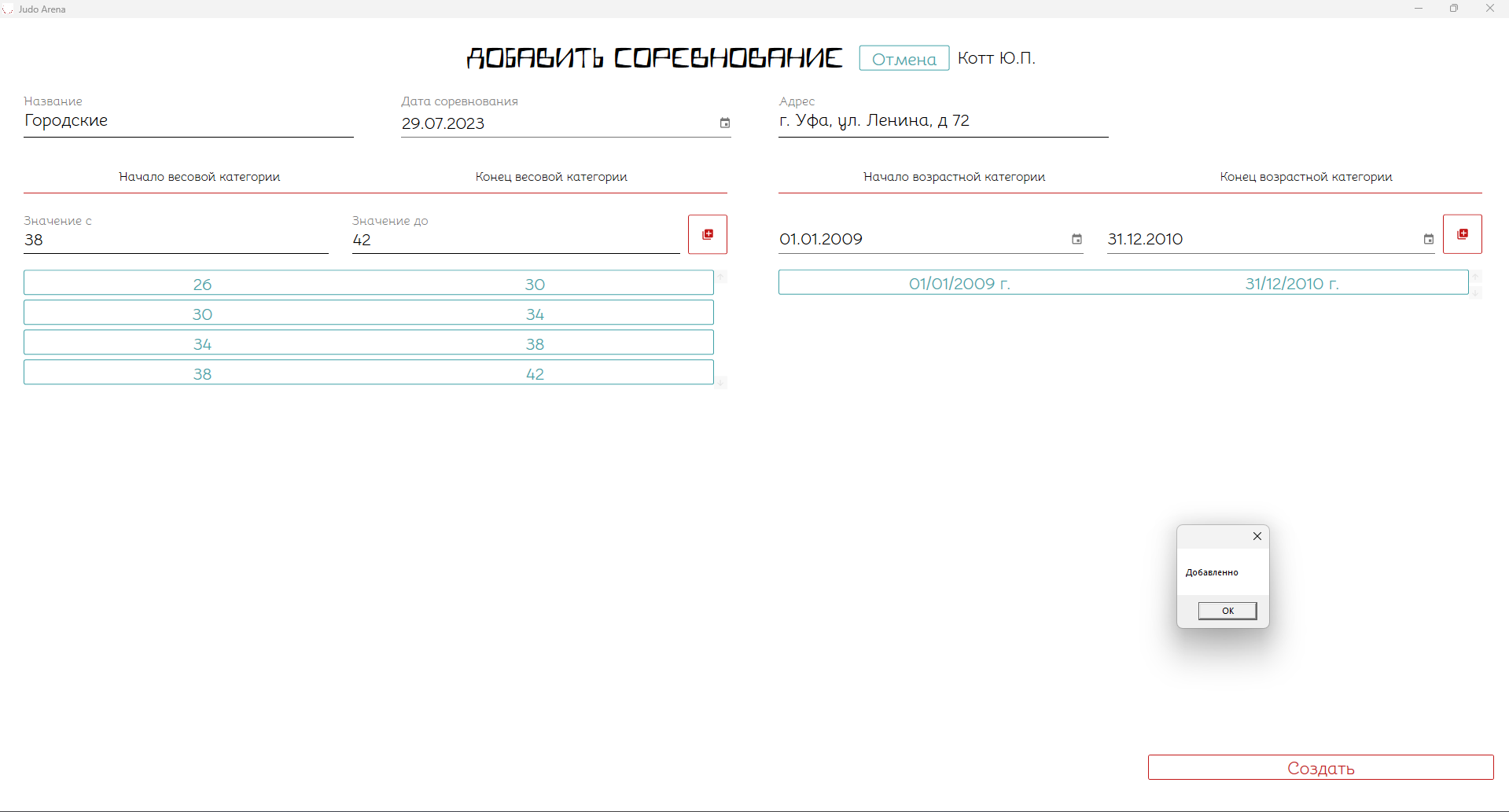


Рисунок 2.2.6 – Успешное добавление соревнования.

Тестирование добавление заявки, ожидается сообщение «Заявка отправлена» (рисунок 2.2.7).

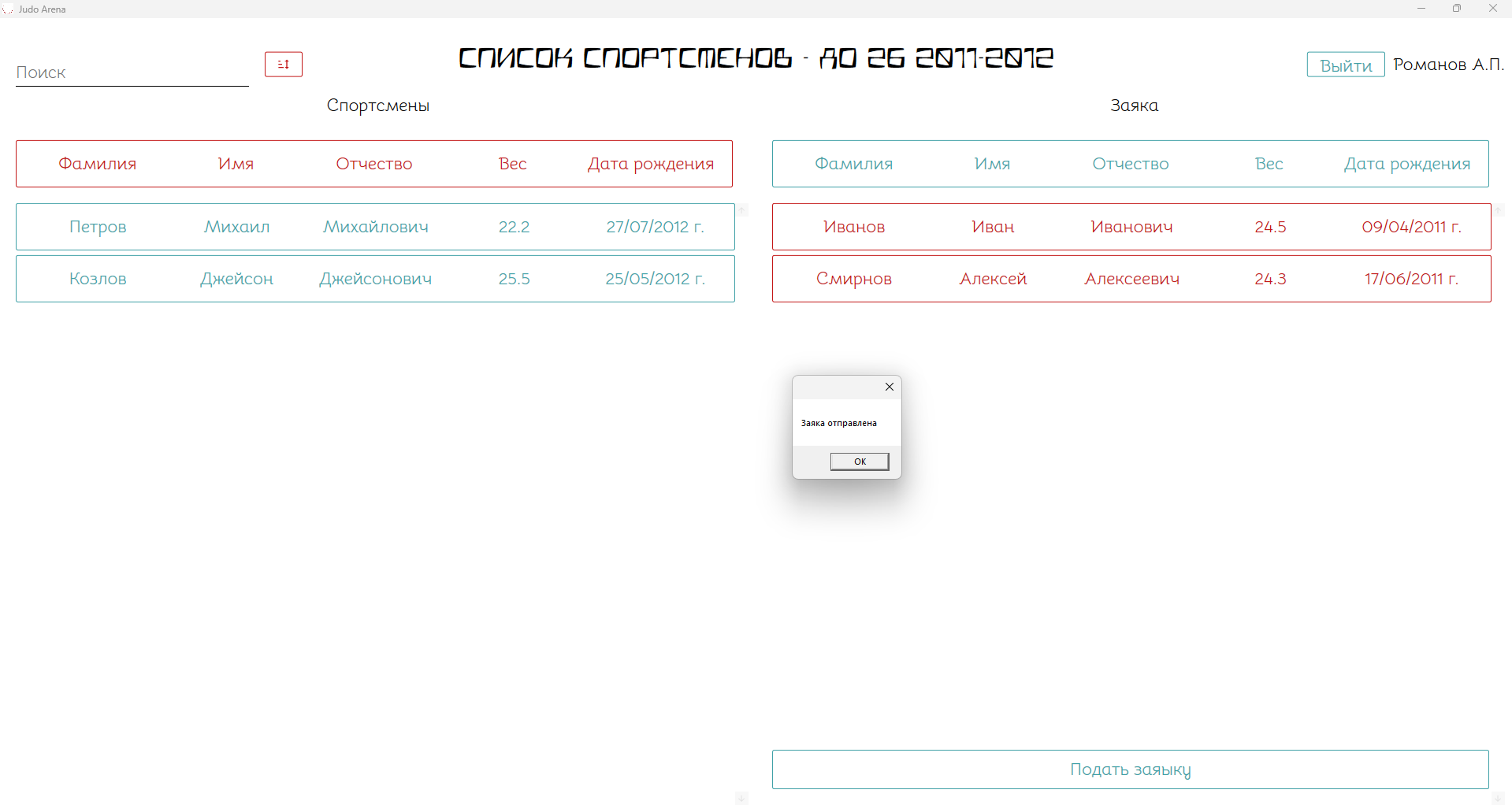


Рисунок 2.2.7 – Успешное добавление заявки.

Тестирование авторизации при незаполненных полях ожидаемое сообщение «Что-то пусто», кнопка входа не активна. (рисунки 2.2.8).



Рисунок 2.2.8 – Авторизация данные не введены

Тестирование авторизация при несуществующем логине ожидаемое сообщение «Пользователь не найден» (рисунки 2.2.9).



Рисунок 2.2.9 – Авторизация при несуществующем логине.

Тестирование авторизация при неверном пароле ожидаемое сообщение «Не тот пароль» (рисунки 2.2.10).



Рисунок 2.2.10 – Aавторизация при неверном пароле.

Тестирование регистрации при не введённом значении ожидаемое сообщение «Что-то пусто», остальные поля протестированы аналогично (рисунки 2.2.11).

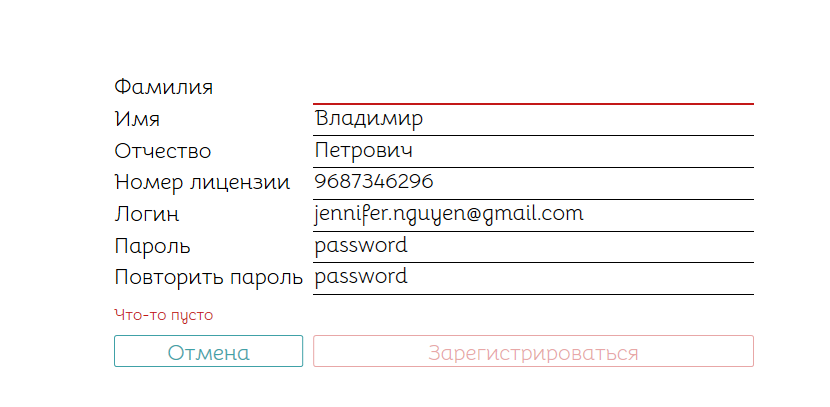


Рисунок 2.2.11 – Регистрации при не введённом значении.

Тестирование регистрации введение уже существующего логина ожидаемое сообщение «Логин уже есть» (рисунки 2.2.12).

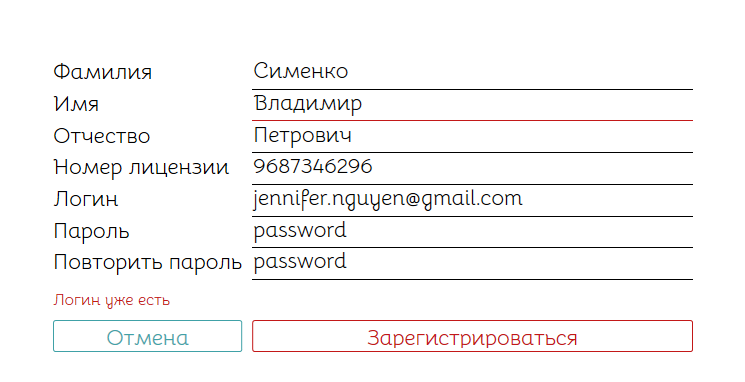


Рисунок 2.2.12 – Регистрации введение уже существующего логина.

Тестирование добавление в соревнование категории, которая уже есть ожидаемое сообщение «Уже есть» (рисунки 2.2.13).

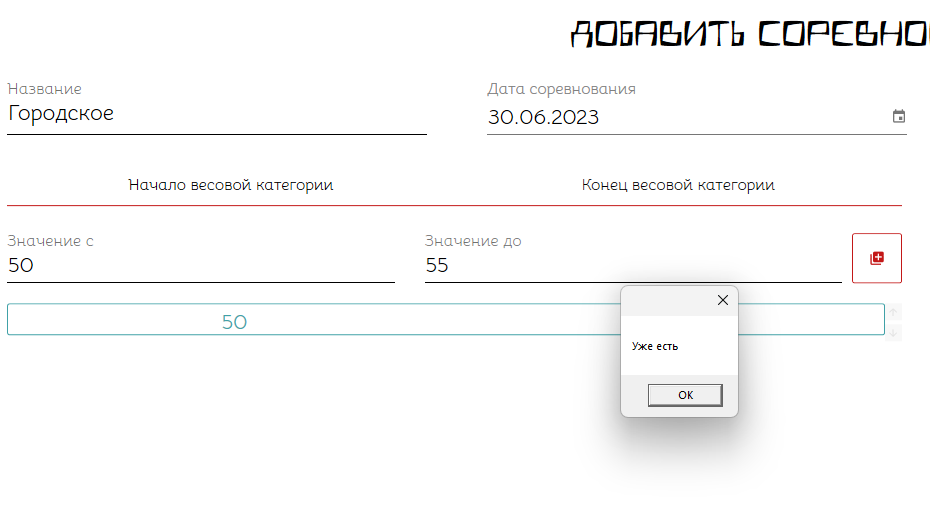


Рисунок 2.2.13 – Добавление в соревнование категории, которая уже есть.

Тестирование добавление соревнования с пустым полем ожидаемое сообщение «Что-то пусто», аналогичным образом проверены все поля (рисунки 2.2.14).

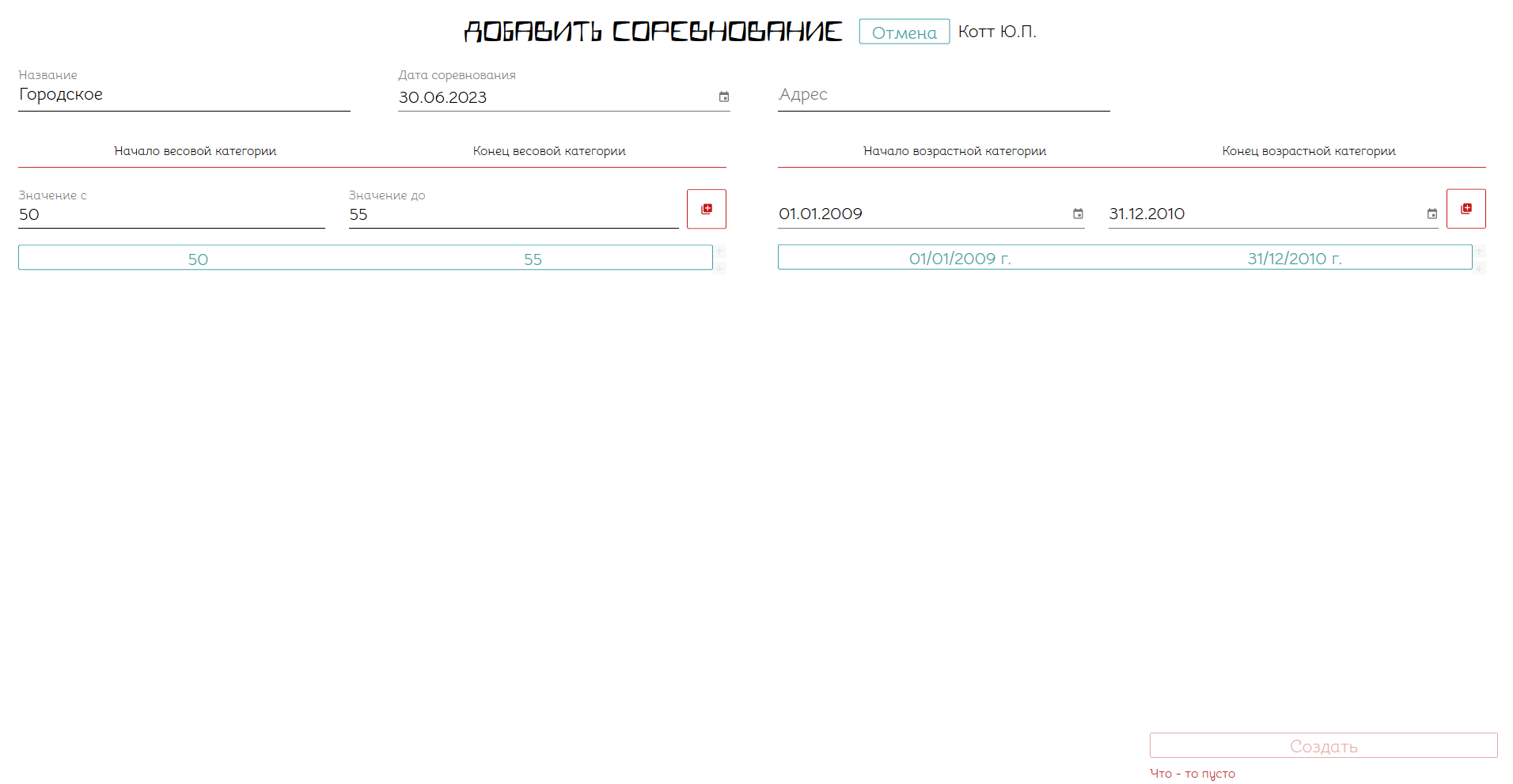


Рисунок 2.2.14 – Добавление соревнования с пустым полем.

Тестирование добавление заявки с пустым полем ожидаемое сообщение «Пусто» (рисунки 2.2.15).

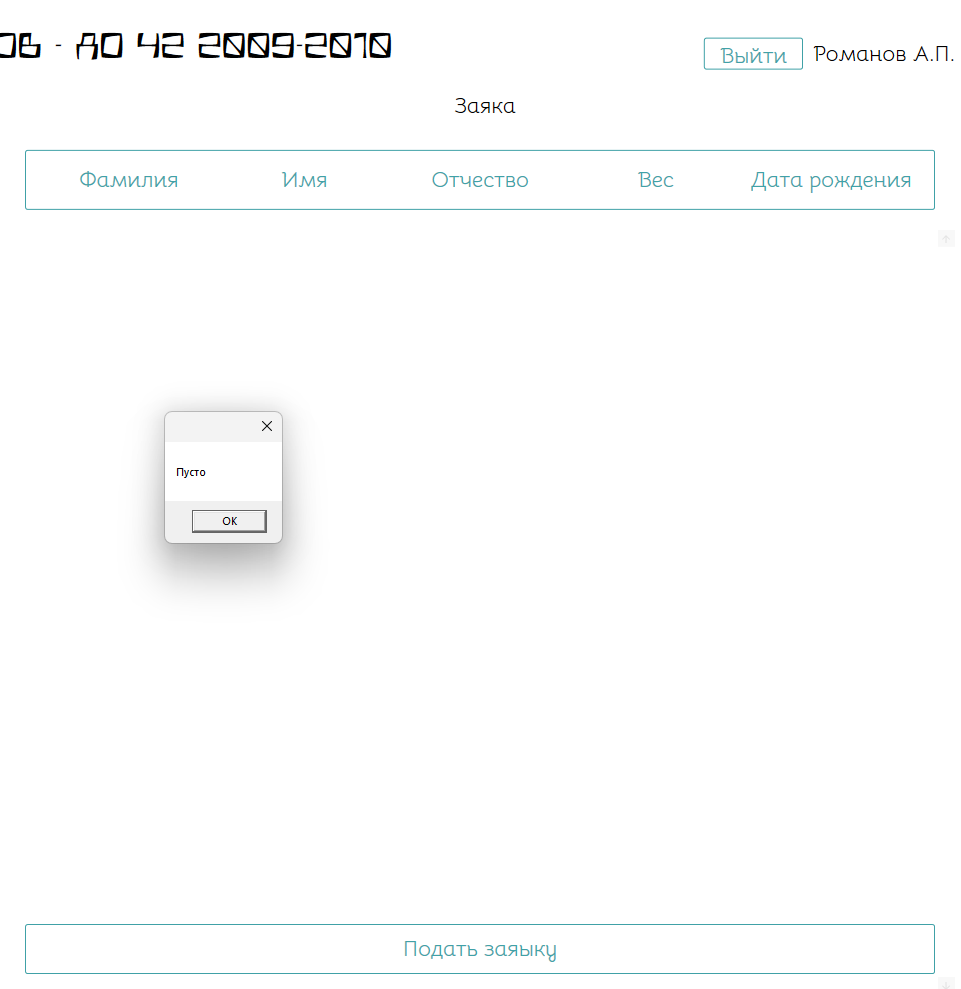


Рисунок 2.2.15 – Добавление заявки с пустым полем.



Рисунок 2.2.16 – авторизация на контрольном примере

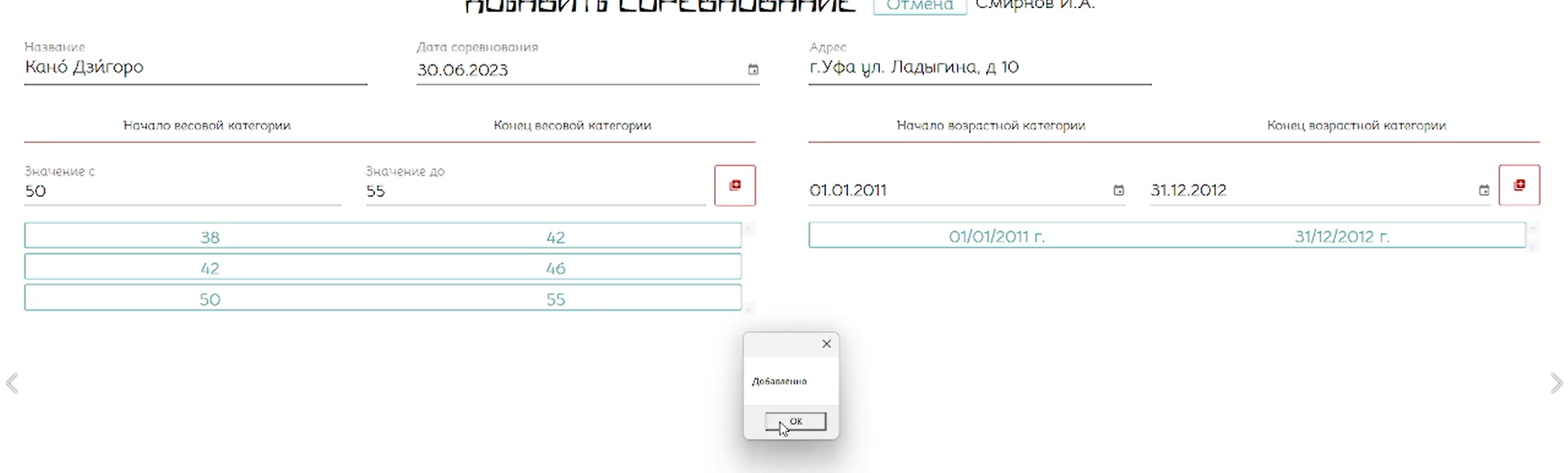


Рисунок 2.2.17– Добавление соревнования на контрольном примере

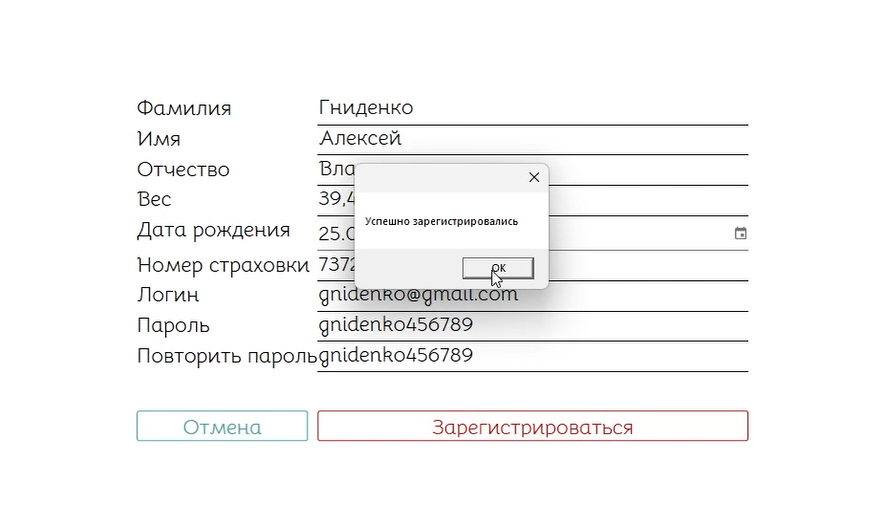


Рисунок 2.2.18– Регистрация участника на контрольном примере

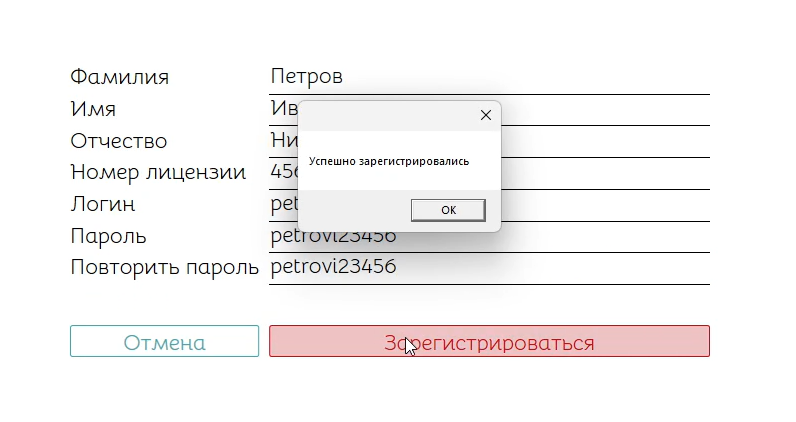


Рисунок 2.2.19– Регистрация тренера на контрольном примере

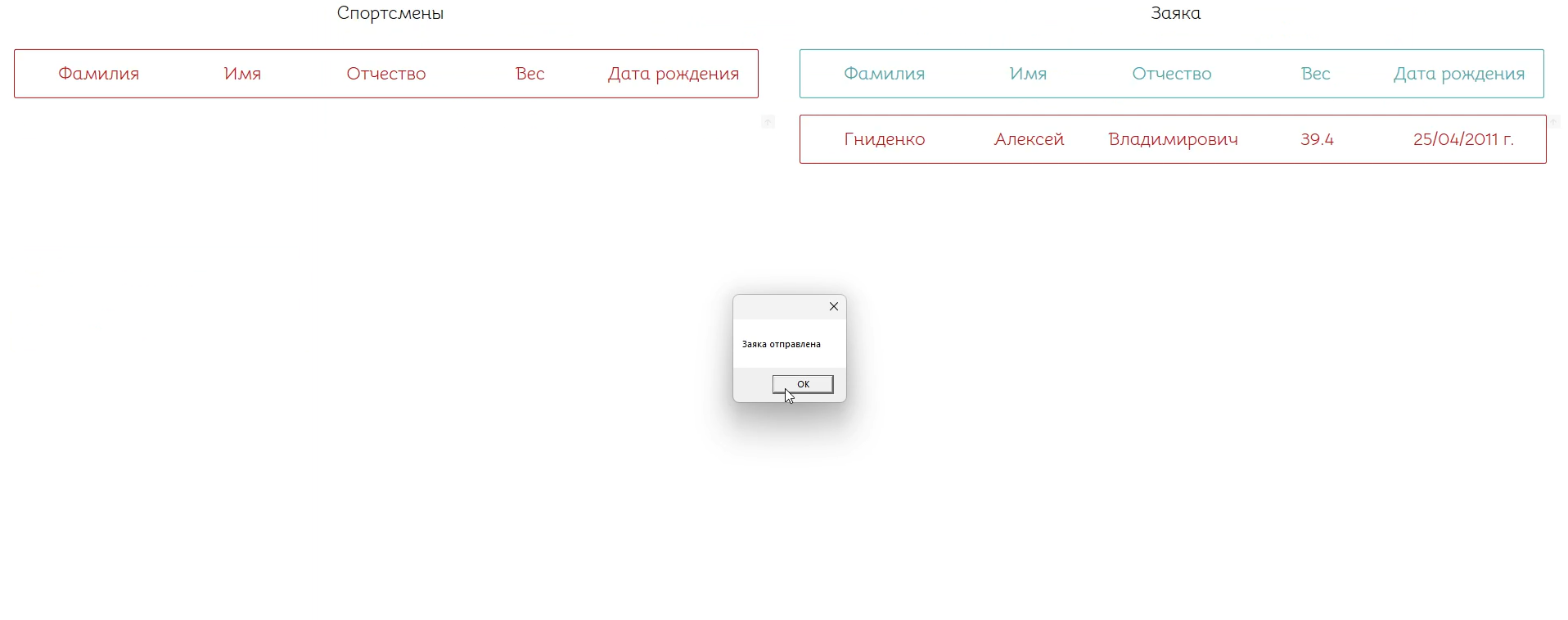


Рисунок 2.2.20 – Подача заявки на контрольном примере

2.3 Руководство пользователя

Назначение системы

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

Цель системы состоит в упрощении и ускорении процесса взаимодействия судей и участников 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 участниках 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.

Подготовка системы к работе

Для запуска программы необходимо запустить приложение JudoArena.exe из каталога, в котором установлен данный программный продукт. После этого открывается окно авторизации для входа в программу (рисунок 2.3.1).

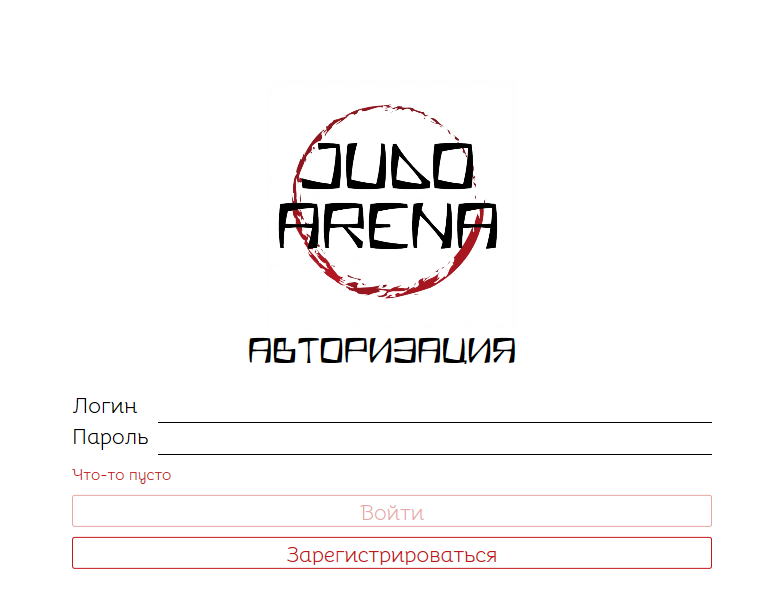


Рисунок 2.3.1– Страница Авторизации

Есть возможно прейти на окно регистрации (рисунок 2.3.2 и русинок 2.3.3) тренера и участника. 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. Судья может добавить согревание (рисунок 2.3.5) Судья и тренер может просматривать все соревнования. Участник только те в которых участвует.



Рисунок 2.3.2– Окно регистрации тренера

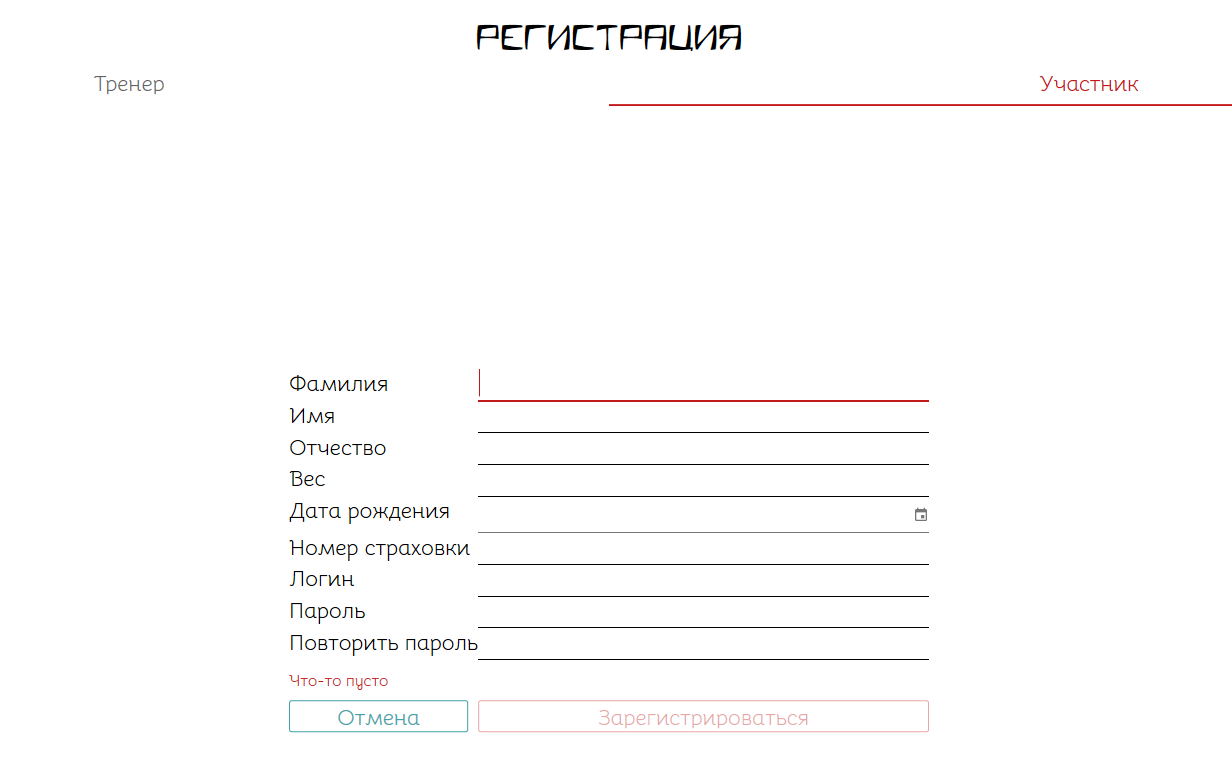


Рисунок 2.3.3– Окно регистрации участника

Для регистрации нужно ввести каретные данные всех полей.

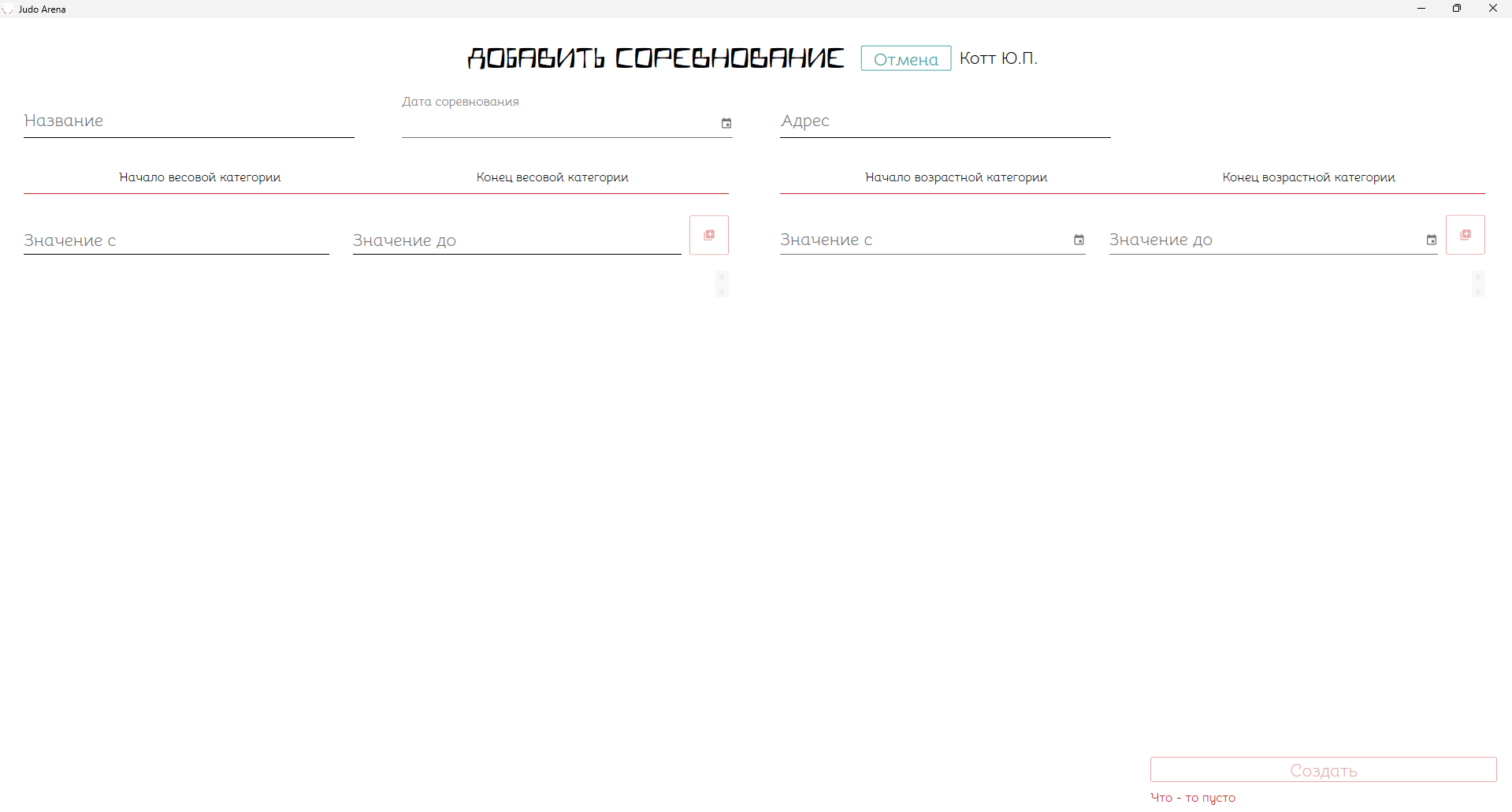


Рисунок 2.3.5– Окно добавления соревнования

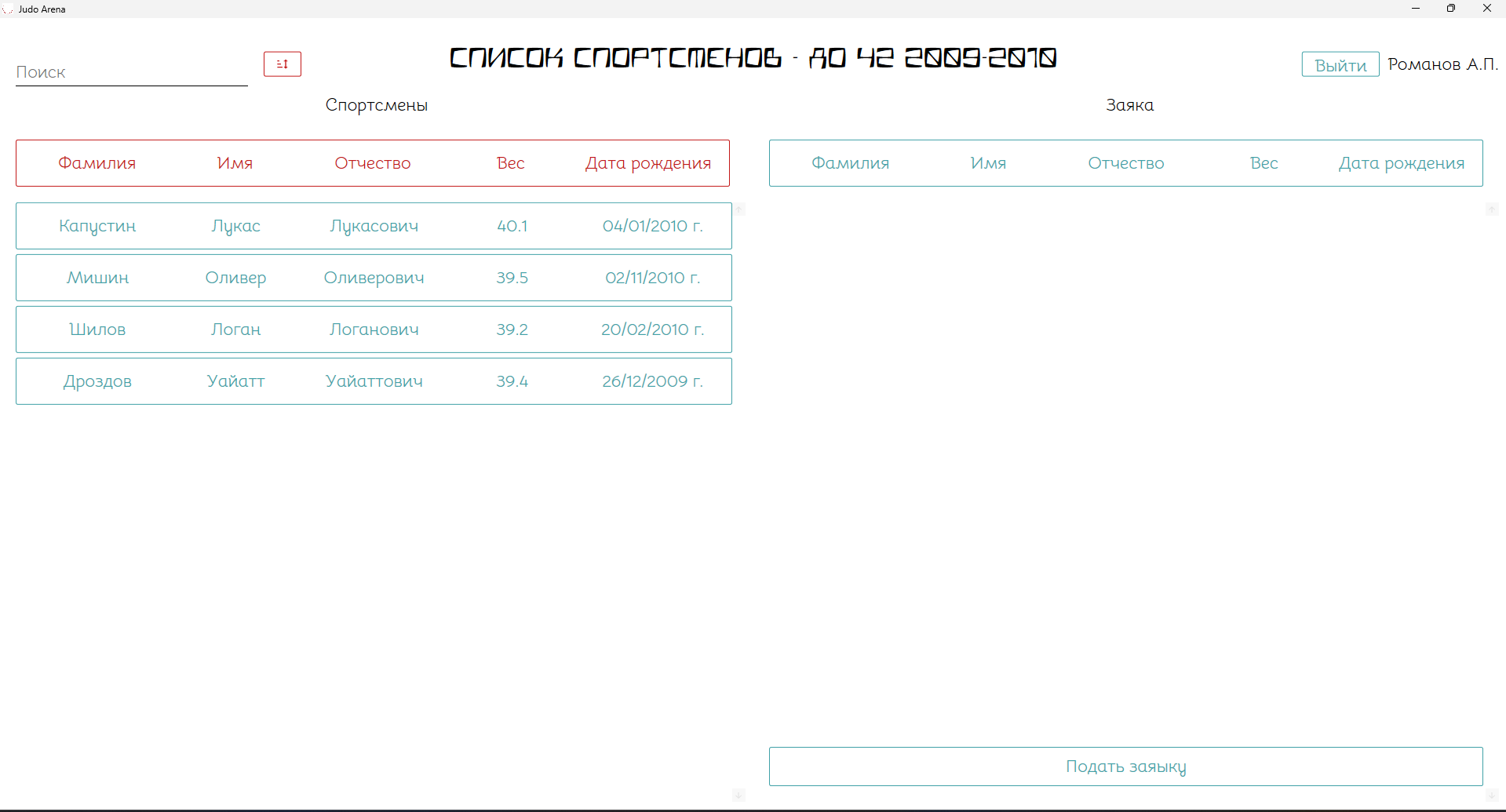


Рисунок 2.3.6– Окно добавления участника

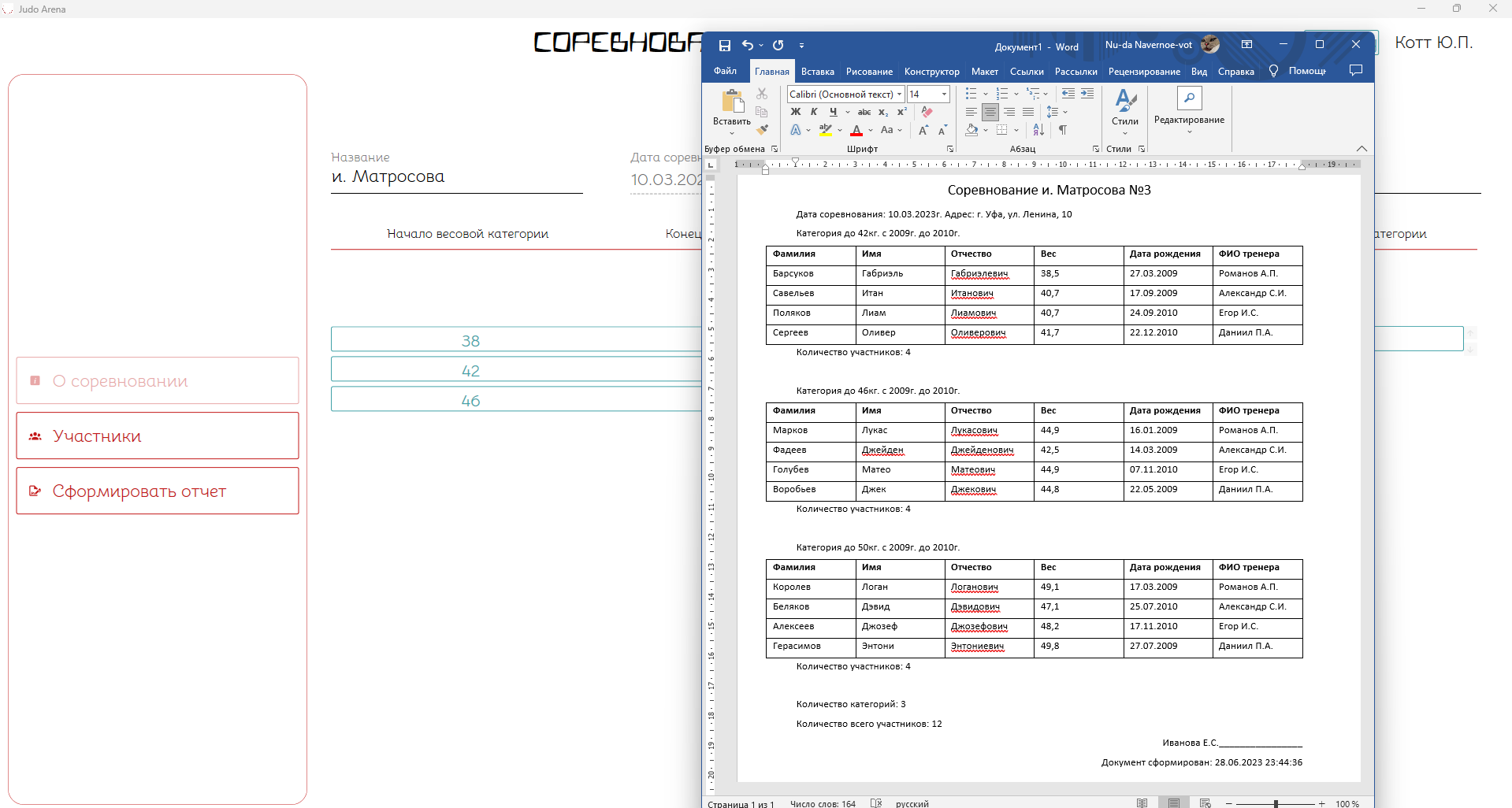


Рисунок 2.3.7– Формирование отчета

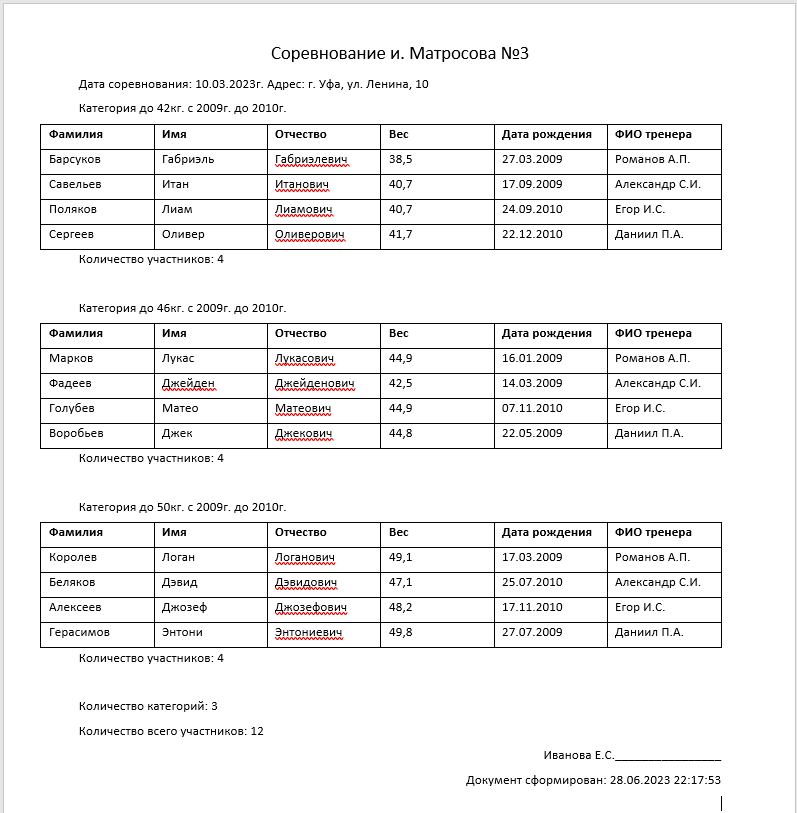
ЗАКЛЮЧЕНИЕ

В ходе разработки курсового проекта было создано приложение JudoArena.exe на языке C# с использованием паттерна MVVM в среде программирования Visual Studio 2022. Приложение предназначено 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 участников 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.exe предлагает электронную форму для заполнения данных об участниках 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, участниках, результатах и других важных аспектах проведения соревнований по дзюдо.

Шаблоны выходных документов



Входные данные контрольного примера

Таблица Б.1 – Участник соревнования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ФИО | Вес | Дата рождения | Номер страховки | Логин | Пароль |
| Гниденко Алексей Владимирович | 39,4 | 25.04.2011 | 7372940541 | gnidenko@gmail.com | gnidenko456789 |

Таблица Б.2 – Участник соревнования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ФИО | Вес | Дата рождения | Номер страховки | Логин | Пароль |
| Петров Иван Николаевич | 4569249701 | 25.04.2011 | 7372940541 | petrovi@gmail.com | Petrovi452751 |

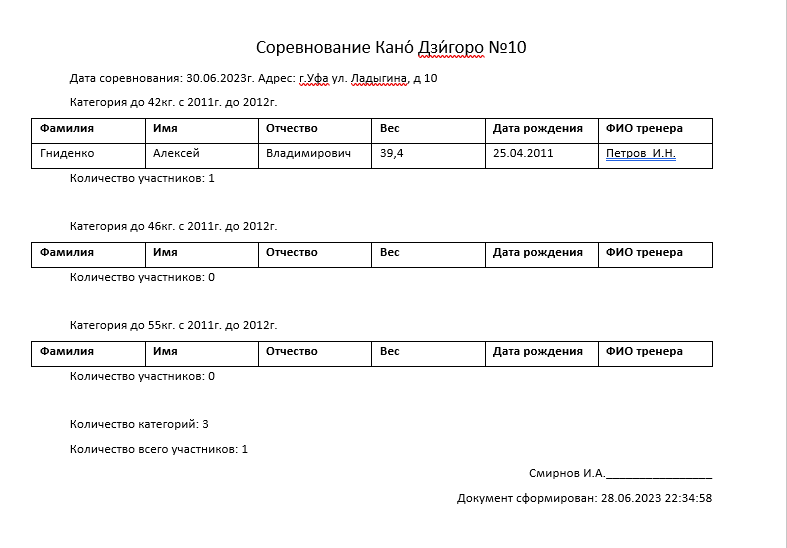
Таблица Б.3 – Данные для авторизации судьи

|  |  |
| --- | --- |
| Логин | Пароль |
| john.doe@gmail.com | 9K3dJf7g2H |

Таблица Б.4 – Данные для добавления соревнования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Дата | Адрес | Весовые категории | Возрастные категории |
| Кано́ Дзи́горо | 30.06.20023 | г.Уфа ул. Ладыгина, д 10 | 38 - 42  42 - 46  46 - 50 | 01.01.2011  31.12.2012 |

Выходные данные контрольного примера



class MainWindowVM

public static readonly Navigation Navigation = new Navigation();

public object CurrentView => Navigation.CurrentView;

public MainWindowVM()

{

Navigation.PropertyChanged += (s, e) => RaisePropertiesChanged(e.PropertyName);

}

class SignInVM

public DelegateCommand SignIn => new(() =>

{

BoollLoading = true;

Error = string.Empty;

try

{

var login = controlUser.GetUserByLogin(Login).Result;

if (login.Password.Equals(Password))

{

UserInterface User = null;

switch (login.Role)

{

case 0:

User = controlUser.GetParticipant(login.Id).Result;

\_navigation.Data = new object[] { User, 0 };

break;

case 1:

User = controlUser.GetTrainer(login.Id).Result;

\_navigation.Data = new object[] { User, 1 };

break;

case 2:

User = controlUser.GetJudges(login.Id).Result;

\_navigation.Data = new object[] { User, 2 };

break;

}

\_navigation.CurrentView = new HomeVM();

}

else { Error = "Не тот пароль"; }

}

catch (AggregateException ex)

{

Error = "Пользователь не найден";

}

BoollLoading = false;

}, bool () =>

{

if(string.IsNullOrWhiteSpace(Login) | string.IsNullOrWhiteSpace(Password))

{

Error = "Что-то пусто";

return false;

}

else

{

if(Error == "Что-то пусто")

Error = string.Empty;

return true;

}

});

public DelegateCommand GoSignUp => new(() =>

{

\_navigation.Data = new string[] { Login, Password };

\_navigation.CurrentView = new SignUpVM();

});

public SignInVM()

{

\_navigation = MainWindowVM.Navigation;

}

class SignUpVM

public DelegateCommand GoSignIn => new(() =>

{

\_navigation.CurrentView = new SignInVM();

});

private (bool, string) ValidateInitials(string PathName, int Length)

{

Regex regex = new Regex(@"\d");

if (PathName.Length < Length)

return (false, $"Слишком коротко - {PathName}");

if (regex.IsMatch(PathName))

return (false, $"Есть числа - {PathName}");

return (true, "");

}

private bool IsNullTrainer()

{

if (string.IsNullOrWhiteSpace(TrainerSurname))

return false;

if (string.IsNullOrWhiteSpace(TrainerName))

return false;

if (string.IsNullOrWhiteSpace(TrainerPatronymic))

return false;

if (string.IsNullOrWhiteSpace(TrainerLicense))

return false;

if (string.IsNullOrWhiteSpace(TrainerLogin))

return false;

if (string.IsNullOrWhiteSpace(TrainerPassword))

return false;

if (string.IsNullOrWhiteSpace(TrainerRepeat))

return false;

return true;

}

public DelegateCommand TrainerSignUp => new(() =>

{

TrainerError = "";

if (!ValidateInitialsFullT())

return;

if (new Regex(@"\D").IsMatch(TrainerLicense))

{

TrainerError = $"Не число - {TrainerLicense}";

return;

}

if (TrainerLicense.Length != 10)

{

TrainerError = $"Лицензия состоит из 10 цифр";

return;

}

if (TrainerLogin.Length < 7)

{

TrainerError = $"Логин состоит из 7 символов";

return;

}

if (controlUser.IsCheckedLogin(TrainerLogin) != null)

{

TrainerError = $"Логин уже есть";

return;

}

if (!IsCheckedPassword(TrainerPassword, 10))

{

TrainerError = $"Пароль должен состоять из 10 символов,\nиз букв и цифр";

return;

}

if (!TrainerPassword.Equals(TrainerRepeat))

{

TrainerError = $"Пароли не совпадают";

return;

}

TrainerError = "";

try

{

\_navigation.Data = null;

UserInterface User = controlUser.AddTrainer(TrainerSurname, TrainerName, TrainerPatronymic,

TrainerLicense, TrainerLogin, TrainerPassword);

\_navigation.Data = new object[] { User, 1 };

MessageBox.Show("Успешно зарегистрировались");

\_navigation.CurrentView = new HomeVM();

TrainerError = "";

}

catch (Exception ex)

{

TrainerError = "Ошибка";

}

}

, bool () =>

{

if (!IsNullTrainer())

{

TrainerError = "Что-то пусто";

return false;

}

else

{

if (TrainerError == "Что-то пусто")

TrainerError = string.Empty;

return true;

}

}

);

public SignUpVM()

{

\_navigation = MainWindowVM.Navigation;

TrainerLogin = ((string[])\_navigation.Data)[0];

TrainerPassword = ((string[])\_navigation.Data)[1];

ParticipantLogin = ((string[])\_navigation.Data)[0];

ParticipantPassword = ((string[])\_navigation.Data)[1];

DateStart = new DateTime(1900, 1, 1);

ParticipantDateBirth = null;

DateEnd = DateTime.Today.AddYears(-3);

}

public DelegateCommand ParticipantSignUp => new(() =>

{

ParticipantError = "";

if (!ValidateInitialsFullP())

return;

if (!new Regex(@"^-?\d+(\.\d+)?(,\d+)?$").IsMatch(ParticipantWeight))

{

ParticipantError = $"Не число - {ParticipantWeight}";

return;

}

decimal result;

if (!decimal.TryParse(ParticipantWeight.Replace(".", ","), out result))

{

ParticipantError = $"Ошибка";

return;

}

if (result < 5)

{

ParticipantError = $"Малый вес - {result}";

return;

}

if (new Regex(@"\D").IsMatch(ParticipantHealth))

{

ParticipantError = $"Не число - {ParticipantHealth}";

return;

}

if (ParticipantHealth.Length != 10)

{

ParticipantError = $"Страховка состоит из 10 цифр";

return;

}

if (ParticipantLogin.Length < 7)

{

ParticipantError = $"Логин состоит из 7 символов";

return;

}

if (controlUser.IsCheckedLogin(ParticipantLogin) != null)

{

ParticipantError = $"Логин уже есть";

return;

}

if (!IsCheckedPassword(ParticipantPassword, 10))

{

ParticipantError = $"Пароль должен состоять из 10 символов, из букв и цифр";

return;

}

if (!ParticipantPassword.Equals(ParticipantRepeat))

{

ParticipantError = $"Пароли не совпадают";

return;

}

ParticipantError = "";

try

{

\_navigation.Data = null;

DateOnly data = DateOnly.FromDateTime(ParticipantDateBirth.Value);

UserInterface User = controlUser.AddParticipant(ParticipantSurname, ParticipantName, ParticipantPatronymic,

result, data, ParticipantHealth, ParticipantLogin, ParticipantPassword);

\_navigation.Data = new object[] { User, 0 };

MessageBox.Show("Успешно зарегистрировались");

\_navigation.CurrentView = new HomeVM();

}

catch (Exception ex)

{

ParticipantError = "Ошибка";

}

}

, bool () =>

{

if (!IsNullParticipant())

{

ParticipantError = "Что-то пусто";

return false;

}

else

{

if (ParticipantError == "Что-то пусто")

ParticipantError = string.Empty;

return true;

}

}

);

class HomeVM

public DelegateCommand GoSignIn => new(() =>

{

\_navigation.Data = null;

\_navigation.CurrentView = new SignInVM();

});

public DelegateCommand GoProfile => new(() =>

{

\_navigation.Data = null;

\_navigation.CurrentView = new SignInVM();

});

public DelegateCommand<object> ChangePage => new DelegateCommand<object>(obj =>

{

\_navigation.Competition = (Competition)obj;

if (dostyp)

{

List<ParticipantCategory> p = \_competition.GetParticipantCategory(User.Id, dostyp);

foreach(ParticipantCategory da in p)

{

if (da.IdCategoryNavigation.IdCompetitionNavigation.Id == \_navigation.Competition.Id)

dostyp = false; break;

}

if (dostyp) { MessageBox.Show("Нет доступа"); return; };

}

\_navigation.CurrentView = new CompetitionsVM();

});

public DelegateCommand Sorting => new(() =>

{

ObjectList = IsSorting ? ObjectList.OrderBy(p => p.Date).ToList() : ObjectList.OrderByDescending(p => p.Date).ToList();

IsSorting = !IsSorting;

RaisePropertiesChanged("ObjectList");

});

public HomeVM()

{

\_navigation = MainWindowVM.Navigation;

Visibility = Visibility.Hidden;

switch (((int)((object[])\_navigation.Data)[1]))

{

case 0:

FullName = ((Participant)((object[])\_navigation.Data)[0]).Surname;

dostyp = true;

break;

case 1:

FullName = ((Trainer)((object[])\_navigation.Data)[0]).Surname;

break;

case 2:

FullName = ((Judge)((object[])\_navigation.Data)[0]).Surname;

Visibility = Visibility.Visible;

break;

}

User = ((UserInterface)((object[])\_navigation.Data)[0]);

\_objectList = \_competition.GetCompetitionList();

ObjectList = \_objectList;

}

class CompetitionsVM

public DelegateCommand Abuot => new(() =>

{

CurrentFrame = new AddСompetitionVM();

}, () =>

{

return !(CurrentFrame is AddСompetitionVM);

});

public DelegateCommand Participants => new(() =>

{

CurrentFrame = new ParticipantsVM();

});

public Visibility Visibility { get; set; }

public DelegateCommand Document => new(() =>

{

DocumentWord doc = new DocumentWord();

doc.GenerateWord(\_navigation.Competition);

});

public CompetitionsVM()

{

Visibility = Visibility.Hidden;

\_navigation = MainWindowVM.Navigation;

\_competition = \_navigation.Competition;

NameCompetition = \_competition.Name;

User = ((UserInterface)((object[])\_navigation.Data)[0]);

\_navigation.NameWin = "О соревновании";

\_currentFrame = new AddСompetitionVM();

switch (((int)((object[])\_navigation.Data)[1]))

{

case 0:

break;

case 1:

break;

case 2:

Visibility = Visibility.Visible;

break;

}

}

public DelegateCommand GoHome => new(() =>

{

\_navigation.CurrentView = new HomeVM();

});

class ParticipantsVM

public DelegateCommand<object> CategorySelection => new DelegateCommand<object>(obj =>

{

\_category = (Category)obj;

ParticipantCategoryList = \_competition.GetParticipantCategory(\_category.Id);

});

public ParticipantsVM()

{

\_navigation = MainWindowVM.Navigation;

Visibility = Visibility.Hidden;

switch (((int)((object[])\_navigation.Data)[1]))

{

case 0:

break;

case 1:

Visibility = Visibility.Visible;

break;

case 2:

break;

}

\_categoryList = \_competition.GetCategory(\_navigation.Competition.Id);

CategoryList = \_categoryList;

\_category = CategoryList[0];

ParticipantCategoryList = \_competition.GetParticipantCategory(\_category.Id);

}

public DelegateCommand AddParticipantCategory => new(() =>

{

\_navigation.Category = \_category;

List<Participant> participant = new();

foreach(var item in ParticipantCategoryList)

{

participant.Add(item.IdParticipantNavigation);

}

\_navigation.Participants = (\_competition.GetParticipant(\_navigation.Category)).Except(participant).ToList(); ;

\_navigation.CurrentViewOld = \_navigation.CurrentView;

\_navigation.CurrentView = new AddParticipantCategoryVM();

});

private bool IsChecedNull()

{

if(string.IsNullOrWhiteSpace(\_nameСompetition) || string.IsNullOrWhiteSpace(\_address) )

{

return false;

}

if(\_nameСompetition == string.Empty)

{

return false;

}

if (\_address == string.Empty)

{

return false;

}

if (!Date.HasValue)

{

return false;

}

if (ObjectWeight.Count < 1)

{

return false;

}

if (ObjectAge.Count < 1)

{

return false;

}

return true;

}

public DelegateCommand AddC => new(() =>

{

Competition competition = \_competition.AddCompetition(NameСompetition, DateOnly.FromDateTime(Date.Value), Address, User.Id);

\_competition.AddCategory(competition.Id, ObjectWeight, ListAge);

MessageBox.Show("Добавленно");

\_navigation.CurrentView = new HomeVM();

}

, bool () =>

{

return Error == string.Empty;

}

);

public DelegateCommand AddWeight => new(() =>

{

Error = $"";

list = ObjectWeight;

decimal WeightStart1;

if (!decimal.TryParse(WeightStartSTR.Replace(".", ","), out WeightStart1))

{

Error = $"Ошибка";

return;

}

decimal WeightEnd1;

if (!decimal.TryParse(WeightEndSTR.Replace(".", ","), out WeightEnd1))

{

Error = $"Ошибка";

return;

}

if (WeightStart1 >= WeightEnd1)

{

Error = "Начало больше завершения";

MessageBox.Show("Начало больше завершения");

return;

}

Weight weight = \_competition.WeightSearch(WeightStart1, WeightEnd1);

if (!ObjectWeight.Contains(weight))

if (weight != null)

{

list.Add(weight);

}

else

{

\_competition.AddWeight(WeightStart1, WeightEnd1);

weight = \_competition.WeightSearch(WeightStart1, WeightEnd1);

list.Add(weight);

}

else MessageBox.Show("Уже есть");

ObjectWeight = new List<Weight>();

ObjectWeight = list;

RaisePropertiesChanged("ObjectWeight");

}

, bool () =>

{

return !WeightStartSTR.Equals("") && !WeightEndSTR.Equals("");

});

public DelegateCommand<object> DeleteWeight => new DelegateCommand<object>(obj =>

{

if (\_navigation.Competition == null)

{

MessageBoxResult result = MessageBox.Show(

$"Вы уверены, что хотите удалить объект {WeightStartSTR}/{WeightEndSTR}?",

"Подтверждение удаления", MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes)

{

Weight weight = (Weight)obj;

list = ObjectWeight;

list.Remove(weight);

ObjectWeight = new List<Weight>();

ObjectWeight = list;

RaisePropertiesChanged("ObjectWeight");

}

}

});

public DelegateCommand AddAge => new(() =>

{

Error = $"";

ListAge = ObjectAge;

if (DateBRStart >= DateBREnd)

{

Error = "Начало больше завершения";

MessageBox.Show("Начало больше завершения");

return;

}

Age age = \_competition.AgeSearch(new DateOnly(DateBRStart.Value.Year, DateBRStart.Value.Month, DateBRStart.Value.Day),

new DateOnly(DateBREnd.Value.Year, DateBREnd.Value.Month, DateBREnd.Value.Day));

if (!ObjectAge.Contains(age))

if (age != null)

{

ListAge.Add(age);

}

else

{

\_competition.AddAge(new DateOnly(DateBRStart.Value.Year, DateBRStart.Value.Month, DateBRStart.Value.Day),

new DateOnly(DateBREnd.Value.Year, DateBREnd.Value.Month, DateBREnd.Value.Day));

age = \_competition.AgeSearch(new DateOnly(DateBRStart.Value.Year, DateBRStart.Value.Month, DateBRStart.Value.Day),

new DateOnly(DateBREnd.Value.Year, DateBREnd.Value.Month, DateBREnd.Value.Day));

ListAge.Add(age);

}

else MessageBox.Show("Уже есть");

ObjectAge = new List<Age>();

ObjectAge = ListAge;

RaisePropertiesChanged("ObjectAge");

}

, bool () =>

{

return DateBRStart != null && DateBREnd != null;

}

);

public DelegateCommand<object> DeleteAge => new DelegateCommand<object>(obj =>

{

if (\_navigation.Competition == null)

{

MessageBoxResult result = MessageBox.Show(

$"Вы уверены, что хотите удалить объект {DateBRStart.Value.Date}/{DateBREnd.Value.Date}?",

"Подтверждение удаления", MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes)

{

Age age = (Age)obj;

ListAge = ObjectAge;

ListAge.Remove(age);

ObjectAge = new List<Age>();

ObjectAge = ListAge;

RaisePropertiesChanged("ObjectAge");

}

}

});

public AddСompetitionVM()

{

\_navigation = MainWindowVM.Navigation;

IsReadOnly = !(\_navigation.Competition == null);

IsEnabled = !IsReadOnly;

if (\_navigation.Competition == null)

{

NameWin = \_navigation.NameWin;

User = ((UserInterface)((object[])\_navigation.Data)[0]);

DateStart = DateTime.Now;

Date = null;

DateEnd = DateTime.Today.AddYears(1);

DateBr = DateTime.Today.AddYears(-150);

\_ojectWeight = new List<Weight>();

\_ojectAge = new List<Age>();

DateBRStart = null;

DateBREnd = null;

VisibilityPage = Visibility.Visible;

VisibilityOrganizer = Visibility.Hidden;

}

else

{

NameWin = \_navigation.NameWin;

Competition competition = \_navigation.Competition;

List<Category> categories = \_competition.GetCategory(competition.Id);

\_ojectWeight = new List<Weight>();

\_ojectAge = new List<Age>();

foreach (Category category in categories)

{

Weight weight = category.IdWeightNavigation;

if (!\_ojectWeight.Contains(weight))

\_ojectWeight.Add(weight);

Age age = category.IdAgeNavigation;

if (!\_ojectAge.Contains(age))

\_ojectAge.Add(age);

}

Organizer = competition.OrganizerNavigation.Surname;

NameWin = "";

User = null;

NameСompetition = competition.Name;

Date = new DateTime(competition.Date.Value.Year, competition.Date.Value.Month, competition.Date.Value.Day);

Address = competition.Address;

VisibilityPage = Visibility.Hidden;

VisibilityOrganizer = Visibility.Visible;

}

}

public DelegateCommand Sorting => new(() =>

{

ObjectList = IsSorting ? new ObservableCollection<Participant>(ObjectList.OrderBy(p => p.Surname).ToList()) : new ObservableCollection<Participant>(ObjectList.OrderByDescending(p => p.Surname).ToList());

IsSorting = !IsSorting;

RaisePropertiesChanged("ObjectList");

});

public DelegateCommand<object> Add => new DelegateCommand<object>(obj =>

{

Participant participant = (Participant)obj;

AddList.Add(participant);

RaisePropertiesChanged("AddList");

list = ObjectList;

ObjectList = new ObservableCollection<Participant>();

list.Remove(participant);

ObjectList = list;

RaisePropertiesChanged("ObjectList");

});

public DelegateCommand<object> Delete => new DelegateCommand<object>(obj =>

{

Participant participant = (Participant)obj;

ObjectList.Add(participant);

RaisePropertiesChanged("AddList");

AddList.Remove(participant);

RaisePropertiesChanged("ObjectList");

});

public DelegateCommand Geft => new(() =>

{

if(AddList.Count > 0)

{

\_competition.AddParticipantCategorie(User.Id, \_navigation.Category.Id, AddList);

MessageBox.Show("Заяка отправлена");

\_navigation.CurrentView = \_navigation.CurrentViewOld;

}

else

{

MessageBox.Show("Пусто");

}

});

public AddParticipantCategoryVM()

{

\_navigation = MainWindowVM.Navigation;

User = ((UserInterface)((object[])\_navigation.Data)[0]);

\_objectList = new ObservableCollection<Participant>(\_navigation.Participants);

\_addList = new ObservableCollection<Participant>();

ObjectList = \_objectList;

CategoriyT = string.Format("до {0} {1}-{2}", \_navigation.Category.IdWeightNavigation.WeightEnd, \_navigation.Category.IdAgeNavigation.AgeStart.Value.Year, \_navigation.Category.IdAgeNavigation.AgeEnd.Value.Year);

}

class ControlCompetition

{

private readonly JudoContext \_context = new();

public List<Competition> GetCompetitionList()

{

return \_context.Competitions.Include(c => c.OrganizerNavigation).ToList();

}

public Competition AddCompetition(string name, DateOnly date, string address, int organizer)

{

Competition competition = new Competition

{

Id = \_context.Competitions.Max(c => c.Id) + 1,

Name = name,

Date = date,

Address = address,

Organizer = organizer

};

\_context.Competitions.Add(competition);

\_context.SaveChanges();

return competition;

}

public Weight WeightSearch(decimal start, decimal end)

{

return \_context.Weights.SingleOrDefault(p => p.WeightStart == start && p.WeightEnd == end);

}

public void AddWeight(decimal start, decimal end)

{

Weight weight = new Weight

{

Id = \_context.Weights.Max(c => c.Id) + 1,

WeightStart = start,

WeightEnd = end

};

\_context.Weights.Add(weight);

\_context.SaveChanges();

}

public Age AgeSearch(DateOnly? start, DateOnly? end)

{

return \_context.Ages.SingleOrDefault(p => p.AgeStart == start && p.AgeEnd == end);

}

public void AddAge(DateOnly? start, DateOnly? end)

{

Age age = new Age

{

Id = \_context.Ages.Max(c => c.Id) + 1,

AgeStart = start,

AgeEnd = end

};

\_context.Ages.Add(age);

\_context.SaveChanges();

}

public void AddCategory(int IdCompetition, List<Weight> weights, List<Age> ages)

{

foreach (Weight weight in weights)

{

foreach(Age age in ages)

{

\_context.Categories.Add(new Category

{

Id = \_context.Categories.Max(c => c.Id) + 1,

IdCompetition = IdCompetition,

IdWeight = weight.Id,

IdAge = age.Id

}

);

\_context.SaveChanges();

}

}

}

public List<Category> GetCategory(int IdCompetition)

{

return \_context.Categories.Where(p => p.IdCompetition == IdCompetition).Include(c => c.IdAgeNavigation).Include(c => c.IdWeightNavigation).ToList();

}

public List<ParticipantCategory> GetParticipantCategory(int IdCategory)

{

return \_context.ParticipantCategories.Where(p => p.IdCategory == IdCategory).Include(c => c.IdParticipantNavigation).Include(c => c.IdTrainerNavigation).ToList();

}

public ParticipantCategory ParticipantCategorySearch(int Id)

{

return \_context.ParticipantCategories.SingleOrDefault(p => p.Id == Id );

}

public void SaveContext()

{

\_context.SaveChanges();

}

public List<Participant> GetParticipant(Category Category)

{

return \_context.Participants.Where(p => (p.Weight >= Category.IdWeightNavigation.WeightStart && p.Weight < Category.IdWeightNavigation.WeightEnd) &&

(p.DateBirth.Value.Year >= Category.IdAgeNavigation.AgeStart.Value.Year && p.DateBirth.Value.Year <= Category.IdAgeNavigation.AgeEnd.Value.Year)).ToList();

}

public void AddParticipantCategorie(int IdTrainer, int IdCategory, ObservableCollection<Participant> Participants)

{

foreach (var participant in Participants)

{

DateTime currentDate = DateTime.Now.Date;

\_context.ParticipantCategories.Add(new ParticipantCategory

{

Id = \_context.ParticipantCategories.Max(c => c.Id) + 1,

IdParticipant = participant.Id,

IdCategory = IdCategory,

IdTrainer = IdTrainer,

Date = new DateOnly(currentDate.Year, currentDate.Month, currentDate.Day)

});

\_context.SaveChanges();

}

}

public List<ParticipantCategory> GetParticipantCategory(int IdParticipant, bool flag)

{

return \_context.ParticipantCategories.Where(p => p.IdParticipant == IdParticipant).Include(c => c.IdCategoryNavigation.IdCompetitionNavigation).ToList();

}

}

class ControlUser

{

private readonly JudoContext \_context = new ();

public async Task<Login> GetUserByLogin(string login)

{

return \_context.Logins.SingleAsync(p => p.Login1 == login).Result;

}

public async Task<Judge> GetJudges(int IDLogin)

{

return \_context.Judges.SingleAsync(p => p.LoginJudges == IDLogin).Result;

}

public async Task<Participant> GetParticipant(int IDLogin)

{

return \_context.Participants.SingleAsync(p => p.LoginParticipant == IDLogin).Result;

}

public async Task<Trainer> GetTrainer(int IDLogin)

{

return \_context.Trainers.SingleAsync(p => p.LoginTrainer == IDLogin).Result;

}

public Login IsCheckedLogin(string login)

{

return \_context.Logins.SingleOrDefault(p => p.Login1 == login) ;

}

public Trainer AddTrainer(string Surname, string Name, string Patronymic, string License, string Login, string Password)

{

\_context.Logins.Add(new Login

{

Login1 = Login,

Password = Password,

Role = 1

});

\_context.SaveChanges();

Trainer trainer = new Trainer

{

Surname = Surname,

Name = Name,

Patronymic = Patronymic,

LicenseNumber = License,

LoginTrainer = IsCheckedLogin(Login).Id

};

\_context.Trainers.Add(trainer);

\_context.SaveChanges();

return trainer;

}

public Participant AddParticipant(string Surname, string Name, string Patronymic, decimal Weight,

DateOnly DateBirth, string HealthInsuranceNumber, string login, string Password)

{

\_context.Logins.Add(new Login

{

Login1 = login,

Password = Password,

Role = 0

});

\_context.SaveChanges();

Participant participant = new Participant

{

Surname = Surname,

Name = Name,

Patronymic = Patronymic,

Weight = Weight,

DateBirth = DateBirth,

HealthInsuranceNumber = HealthInsuranceNumber,

LoginParticipant = IsCheckedLogin(login).Id

};

\_context.Participants.Add(participant);

\_context.SaveChanges();

return participant;

}

}

class DocumentWord

{

public DocumentWord() { }

public void GenerateWord(Competition competition)

{

List<Category> categories = new List<Category>();

ControlCompetition controlCompetition = new();

categories = controlCompetition.GetCategory(competition.Id);

List<ParticipantCategory> participantsCategory = new List<ParticipantCategory>();

Word.Application oWordApp = new Word.Application();

oWordApp.Visible = true;

Word.Document oDoc = oWordApp.Documents.Add();

const float cmToPoints = 2.835f; // Константа для преобразования сантиметров в пункты

// Удаление отступов на странице (в сантиметрах)

oDoc.PageSetup.LeftMargin = 10 \* cmToPoints;

oDoc.PageSetup.RightMargin = 20 \* cmToPoints;

oDoc.PageSetup.TopMargin = 10 \* cmToPoints;

oDoc.PageSetup.BottomMargin = 20 \* cmToPoints;

// Добавление заголовка

Word.Paragraph titlePara = oDoc.Paragraphs.Add();

titlePara.Range.Text = $"Соревнование {competition.Name} №{competition.Id}";

titlePara.Range.Font.Size = 14;

titlePara.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

titlePara.Range.InsertParagraphAfter();

// Добавление даты соревнования

Word.Paragraph infoCompetition = oDoc.Paragraphs.Add();

infoCompetition.Range.Text = $"Дата соревнования: {competition.Date.Value.ToShortDateString()}г. " +

$"Адрес: {competition.Address}";

infoCompetition.Range.Font.Size = 10;

infoCompetition.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

infoCompetition.Range.InsertParagraphAfter();

int col = 0;

foreach (Category category in categories)

{

Word.Paragraph categoryName = oDoc.Paragraphs.Add();

categoryName.Range.Text = $"Категория " +

$"до {category.IdWeightNavigation.WeightEnd}кг. " +

$"c {category.IdAgeNavigation.AgeStart.Value.Year}г. " +

$"до {category.IdAgeNavigation.AgeEnd.Value.Year}г.";

categoryName.Range.Font.Size = 10;

categoryName.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

categoryName.Range.InsertParagraphAfter();

Word.Paragraph para = oDoc.Paragraphs.Add();

participantsCategory = (controlCompetition.GetParticipantCategory(category.Id));

Word.Table table = oDoc.Tables.Add(para.Range, participantsCategory.Count + 1, 6); // Создание таблицы

// Заголовки столбцов

table.Cell(1, 1).Range.Text = "Фамилия";

table.Cell(1, 2).Range.Text = "Имя";

table.Cell(1, 3).Range.Text = "Отчество";

table.Cell(1, 4).Range.Text = "Вес";

table.Cell(1, 5).Range.Text = "Дата рождения";

table.Cell(1, 6).Range.Text = "ФИО тренера";

// Заполнение данных

for (int i = 0; i < participantsCategory.Count; i++)

{

ParticipantCategory pc = participantsCategory[i];

// Фамилия

table.Cell(i + 2, 1).Range.Text = pc.IdParticipantNavigation.Surname;

// Имя

table.Cell(i + 2, 2).Range.Text = pc.IdParticipantNavigation.Name;

// Отчество

table.Cell(i + 2, 3).Range.Text = pc.IdParticipantNavigation.Patronymic;

// Вес

table.Cell(i + 2, 4).Range.Text = pc.IdParticipantNavigation.Weight.ToString();

// Дата рождения

table.Cell(i + 2, 5).Range.Text = pc.IdParticipantNavigation.DateBirth.Value.ToShortDateString();

// ФИО тренера

table.Cell(i + 2, 6).Range.Text = $"{pc.IdTrainerNavigation.Surname} " +

$"{pc.IdTrainerNavigation.Name[0]}." +

$"{pc.IdTrainerNavigation.Patronymic[0]}.";

col++;

}

// Форматирование таблицы

table.Borders.Enable = 1; // Включение границ таблицы

table.Rows[1].Range.Font.Bold = 1; // Жирный шрифт для заголовков

// Количесво участников

Word.Paragraph catpar = oDoc.Paragraphs.Add();

catpar.Range.Text = $"Количество участников: {participantsCategory.Count}";

catpar.Range.Font.Size = 10;

catpar.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

catpar.Range.InsertParagraphAfter();

para.Range.InsertParagraphAfter(); // Добавление пустого абзаца после таблицы

}

// Количесво категорий

Word.Paragraph categoryPara = oDoc.Paragraphs.Add();

categoryPara.Range.Text = $"Количество категорий: {categories.Count}";

categoryPara.Range.Font.Size = 10;

categoryPara.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

categoryPara.Range.InsertParagraphAfter();

// Количесво участников

Word.Paragraph colPara = oDoc.Paragraphs.Add();

colPara.Range.Text = $"Количество всего участников: {col}";

colPara.Range.Font.Size = 10;

colPara.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

colPara.Range.InsertParagraphAfter();

// Добавление даты

Word.Paragraph OrganizerPara = oDoc.Paragraphs.Add();

OrganizerPara.Range.Text = $"{competition.OrganizerNavigation.Surname} " +

$"{competition.OrganizerNavigation.Name[0]}." +

$"{competition.OrganizerNavigation.Patronymic[0]}.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

OrganizerPara.Range.Font.Size = 10;

OrganizerPara.Alignment = Word.WdParagraphAlignment.wdAlignParagraphRight;

OrganizerPara.Range.InsertParagraphAfter();

// Добавление даты

Word.Paragraph datePara = oDoc.Paragraphs.Add();

datePara.Range.Text = $"Документ сформирован: {DateTime.Now.ToString()}";

datePara.Range.Font.Size = 10;

datePara.Alignment = Word.WdParagraphAlignment.wdAlignParagraphRight;

datePara.Range.InsertParagraphAfter();

}

}

Список сокращений

* БД – база данных
* ФИО - Фамилия Имя Отчество

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Потопахин В. Искусство алгоритмизации. [Текст] / В. Потопахин. – Москва: ДМК Пресс, 2018. – 320 с.

2 Семакин И. Основы алгоритмизации и программирования. Учебник [Текст] / И. Семакин, А. Шестаков. – 2-е изд., Москва: Академия, 2018. – 304 с.

3 C# 10 и .NET 6. Современная кросс-платформенная разработка[Текст] / Прайс Марк, 2023 – 848 с.

4 C#. Практическое руководство[Текст] / Евдокимов П. В., 2022 – 416с.

5 Справочник C#. Кратко, быстро, под рукой[Текст] / Дубовик Е. В., Евдокимов П. В., 2023 – 336 с.

6 Программирование на C, C# и Java [Электронный ресурс] / MySQL и C# работаем с базой данных. – Режим доступа https://vscode.ru/proglessons/mysql-c-sharp.html, свободный

7 Metanit [Электронный ресурс] / Руководство по языку C#. – Режим доступа https://metanit.com/sharp/tutorial/, свободный

8 Metanit [Электронный ресурс] / Руководство по WPF. – Режим доступа https://metanit.com/sharp/wpf/, свободный

9 Metanit [Электронный ресурс] / Паттерн MVVM – Режим доступа https://metanit.com/sharp/wpf/22.1.php, свободный

10 Metanit[Электронный ресурс] / Entity Framework Core и C# | MySQL – Режим доступа https://metanit.com/sharp/efcore/7.2.php, свободный

11 Habr [Электронный ресурс] / MVVM: полное понимание (+WPF) Часть 1 – Режим доступа https://habr.com/ru/articles/338518/, свободный

12 Microsoft [Электронный ресурс] / Руководство по языку C#. – Режим доступа https://docs.microsoft.com/ru-ru/dotnet/csharp/, свободный

13 Professor Web [Электронный ресурс] / Основы WPF. – Режим доступа https://professorweb.ru/my/WPF/base\_WPF/level1/info\_WPF.php, свободный

14 Sql-tutorial [Электронный ресурс] / SQL Задачи и решения. – Режим доступа http://www.sql-tutorial.ru/ru/content.html, свободный

15 W3schools [Электронный ресурс] / SQL Tutorial. – Режим доступа https://www.w3schools.com/sql/default.asp, свободный

16 Webmasterwiki [Электронный ресурс] / Базы данных: Основные понятия. – Режим доступа https://www.webmasterwiki.ru/MySQL, свободный

17 WorldSkills Russia [Электронный ресурс] / Программные решения для бизнеса. – Режим доступа https://nationalteam.worldskills.ru/skills/programmnye-resheniya-dlyabiznesa/, свободны

18 Youtube[Электронный ресурс]/WPF(Павел Шмачилин)– https://youtube.com/playlist?list=PL-p05fYs48r-HOAA4\_5qZfS0rTIka3fgc

19 Skillbox [Электронный ресурс] / MVVM: проектирование приложений для Windows – Режим доступа https://skillbox.ru/media/code/mvvm\_proektirovanie\_prilozheniy\_dlya\_windows/ свободный

20 Vscode.ru [Электронный ресурс] / MySQL и C# — работаем с базой данных. Запросы SELECT – Режим доступа https://vscode.ru/prog-lessons/mysql-c-sharp.html свободный