

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»  
(СПбГУТ)

АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ  
ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ  
(АКТ (ф) СПбГУТ)

# КУРСОВОЙ ПРОЕКТ

НА ТЕМУ

РАЗРАБОТКА ПОДСИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ

СОРЕВНОВАНИЯМИ ПО ТХЭКВОНДО

Л109. 25КП01. 012 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки

программного обеспечения

Студент	ИСПП-21	08.12.2025	В.А. Колосов
	(Группа)	(Подпись)	(И.О. Фамилия)
Преподаватель		09.12.2025	Ю.С. Маломан
		(Подпись)	(И.О. Фамилия)

Архангельск 2025

## СОДЕРЖАНИЕ

Перечень сокращений и обозначений .....	3
Введение .....	4
1 Анализ и разработка требований .....	6
1.1 Назначение и область применения .....	6
1.2 Постановка задачи .....	6
1.3 Выбор состава программных и технических средств .....	?
2 Проектирование программного обеспечения .....	?
2.1 Проектирование интерфейса пользователя .....	?
2.2 Разработка архитектуры программного обеспечения .....	?
2.3 Проектирование базы данных .....	?
3 Разработка и интеграция модулей программного обеспечения .....	?
3.1 Разработка программных модулей .....	?
3.2 Реализация интерфейса пользователя .....	?
3.3 Разграничение прав доступа пользователей .....	?
3.4 Экспорт и импорт данных .....	?
4 Тестирование и отладка программного обеспечения .....	?
4.1 Структурное тестирование .....	?
4.2 Функциональное тестирование .....	?
5 Инструкция по эксплуатации программного обеспечения .....	?
5.1 Установка программного обеспечения .....	?
5.2 Инструкция по работе .....	?
Заключение .....	?
Список использованных источников .....	30

## **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

В настоящем курсовом проекте применяют следующие сокращения и обозначения:

АРОО –

БД – база данных

ГТФ – глобальная федерация тхэквондо

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базами данных

API – интерфейс программирования приложения

DTO – объект передачи данных

ERD – диаграмма «сущность-связь»

HTTP – протокол передачи гипертекста

IDE – интегрированная среда разработки

SQL – язык структурированных запросов

WPF – платформа для построения графических интерфейсов Windows

## **ВВЕДЕНИЕ**

Актуальность разрабатываемого проекта заключается в автоматизации процессов формирования турнирных сеток и учета участников турниров в спортивных организациях.

В современных условиях для эффективного управления соревнованиями важны оперативность и точность данных. Спортивные федерации и клубы сталкиваются с проблемами, связанными с учетом данных об участниках, формированием турнирных сеток и подсчетом результатов.

Разработка подсистемы для управления соревнованиями по тхэквондо позволит значительно упростить процесс организации турниров, улучшить взаимодействие с участниками и повысить общую эффективность проведения турниров.

Целью курсового проекта является разработка подсистемы, обеспечивающей возможность комплексного управления турнирами по тхэквондо, включая учет участников, формирование турнирных сеток и отслеживания результатов поединков в реальном времени.

Для достижения поставленной цели требуется решить следующие задачи:

- провести сбор и анализ требований целевой аудитории (спортивные федерации, тренеры, организаторы турниров);
- проанализировать информационные источники по предметной области;
- изучить существующие решения в области автоматизации проведения турниров;
- спроектировать архитектуру подсистемы;
- спроектировать диаграмму использования подсистемы;
- выбрать состав программных и технических средств для реализации проекта;
- спроектировать БД;

- создать БД в выбранной СУБД;
- разработать API для взаимодействия клиентских приложений с БД;
- реализовать разграничение прав доступа пользователей (организаторы, судьи);
- обеспечить защиту данных;
- разработать пользовательский интерфейс;
- реализовать функциональность регистрации участников в турнире;
- реализовать функциональность формирования турнирных сеток;
- реализовать функциональность проведения поединков;
- выполнить структурное тестирование ПО;
- выполнить функциональное тестирование ПО;
- разработать программную документацию;
- разработать эксплуатационную документацию.

В результате выполнения поставленных задач будет создана подсистема для управления соревнованиями, которая значительно упростит процесс формирования турнирных сеток и повысит скорость обработки данных участников турниров.

# **1 Анализ и разработка требований**

## **1.1 Назначение и область применения**

Основным назначением подсистемы является автоматическое формирование турнирных сеток и учет участников турнира, что позволит снизить временные затраты организаторов и судей на турнирах и минимизировать человеческие ошибки.

ПО предназначено для использования организаторами турниров по тхэквондо, судьями и участниками турниров в АРОО «Федерация тхэквондо ГТФ», а также в других спортивных федерациях и объединениях, осуществляющих проведение соревнований по тхэквондо по правилам ГТФ.

## **1.2 Постановка задачи**

Необходимо разработать оконное приложение, в котором будут реализованы следующие функциональные возможности:

- автоматическое формирование турнирных сеток;
- просмотр и редактирование данных о результатах поединков;
- просмотр и редактирование данных о турнирах;
- просмотр и редактирование данных о категориях;
- формирование отчетов по результатам соревнований в формате \*.docx;
- экспорт и импорт информации об участниках в формате \*.xlsx.

Интерфейс должен быть интуитивно понятен для пользователя.

В подсистеме должна быть обязательная авторизация пользователей.

Гость – неавторизованный пользователь, имеющий доступ к просмотру списка турниров, а также возможность зарегистрироваться в системе и выполнить вход.

Судья может просматривать список турниров, состав их участников и турнирные сетки и вносить результаты боёв в режиме реального времени.

Организатор имеет полный доступ ко всем функциям подсистемы, а также может формировать турнирные сетки, редактировать данные о категориях, участниках и турнирах.

На рисунке 1 изображена диаграмма вариантов использования подсистемы.

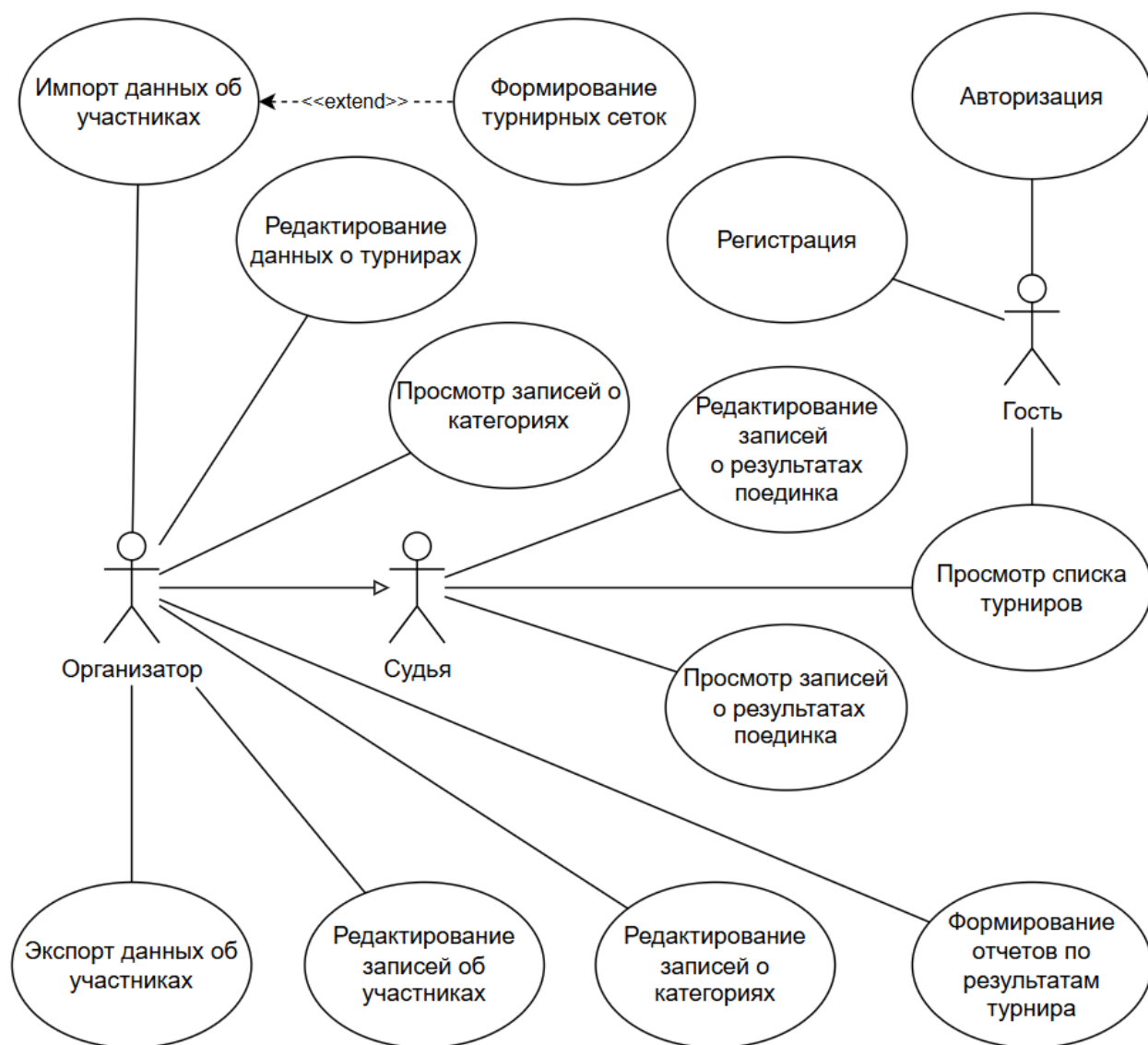


Рисунок 1 – Диаграмма вариантов использования

Работа с системой осуществляется по следующему алгоритму:

- пользователь входит в систему;

- организатор создает запись о новом турнире, заполняя всю информацию о нем;
- организатор создает записи о категориях в созданном турнире;
- система автоматически проверяет все данные и добавляет запись о турнире в общий список;
- организатор турнира импортирует данные об участниках;
- на основе данных об участниках автоматически формируются турнирные сетки;
- в ходе проведения турнира и отдельных поединков судьи и организаторы вносят информацию о результатах поединков;
- после завершения всех поединков в каждой категории система автоматически определяет победителей и призеров;
- организатор формирует и экспортирует отчеты по итогам турнира.

### **1.3 Выбор состава программных и технических средств**

Работа с оконным приложением будет осуществляться на ПК и ноутбуках с ОС Windows (Windows 10 и выше).

В качестве СУБД выбрана MySQL 8.0, так как эта СУБД обладает высокой производительностью, масштабируемостью и простотой в использовании, что позволяет эффективно обрабатывать данные о турнирах в реальном времени.

Клиентская и серверная части приложения будут разработаны на C#, так как с помощью этого языка можно эффективно создавать современные приложения с использованием технологии WPF для клиентской части и Web-API ASP.NET Core для серверной части.

Для разработки оконного приложения будет использоваться IDE Visual Studio 2022, так как эта среда предлагает удобные инструменты для работы с C#, включая инструменты для работы с Git и средства отладки.



Для функционирования системы на стороне сервера необходимы следующие программные и технические средства:

- ОС Ubuntu Server версии 24 и выше;
- сервер БД MySQL версии не ниже 8.0;
- процессор частотой 2 ГГц;
- свободная оперативная память 4 ГБ;
- свободное место на диске не менее 1 ГБ.

Для функционирования системы на стороне клиента необходимы следующие программные и технические средства:

- ОС Windows 10 и выше;
- .NET 8.0;
- процессор частотой 2 ГГц;
- свободная оперативная память 4 ГБ;
- свободное место на диске не менее 500 МБ;
- постоянное подключение к локальной сети или сети интернет.

Указанный набор программных средств обеспечит эффективное функционирование оконного приложения.

## 2 Проектирование программного обеспечения

### 2.1 Проектирование интерфейса пользователя

В рамках разработки оконного приложения создан интерфейс пользователя в виде набора wireframe при помощи инструмента draw.io. Эти визуальные представления позволяют представить структуру приложения, его основные элементы и функциональность.

Wireframe страниц «Авторизация», «Турниры», «Создание турнира», «Редактирование турнира» представлен на рисунке 2.

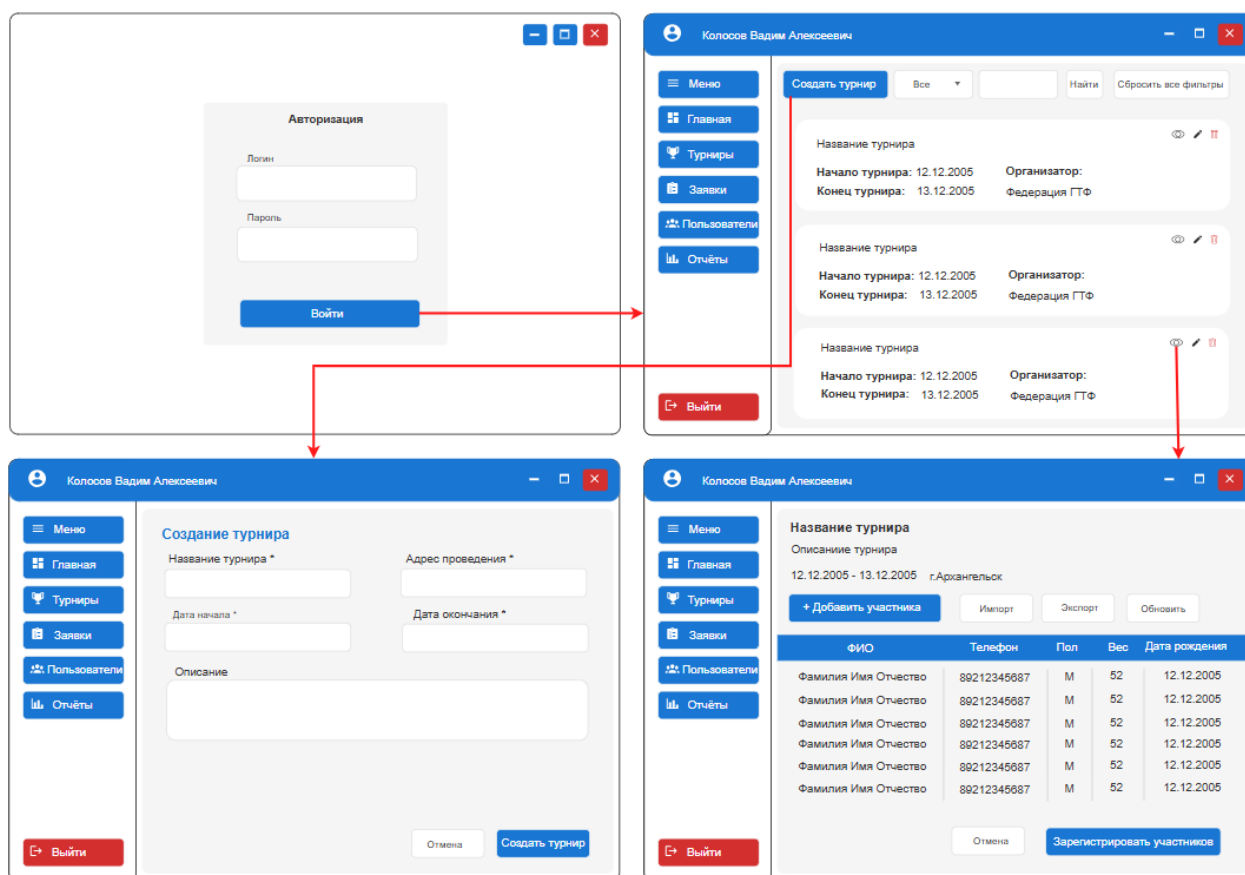


Рисунок 2 – Wireframe основных окон подсистемы

Для оконного приложения были выбраны следующие цвета:

- основной цвет: #FF1976D2;

- цвет фона: #FFF5F5F5;
- основной цвет текста: #FF333333;
- вторичный цвет текста: #FF666666.

## 2.2 Разработка архитектуры программного обеспечения

Архитектура построена на основе клиент-серверной модели и включает в себя несколько ключевых компонентов: оконное приложение, БД, API, позволяющий клиенту взаимодействовать с сервером. Диаграмма развертывания изображена на рисунке 3.

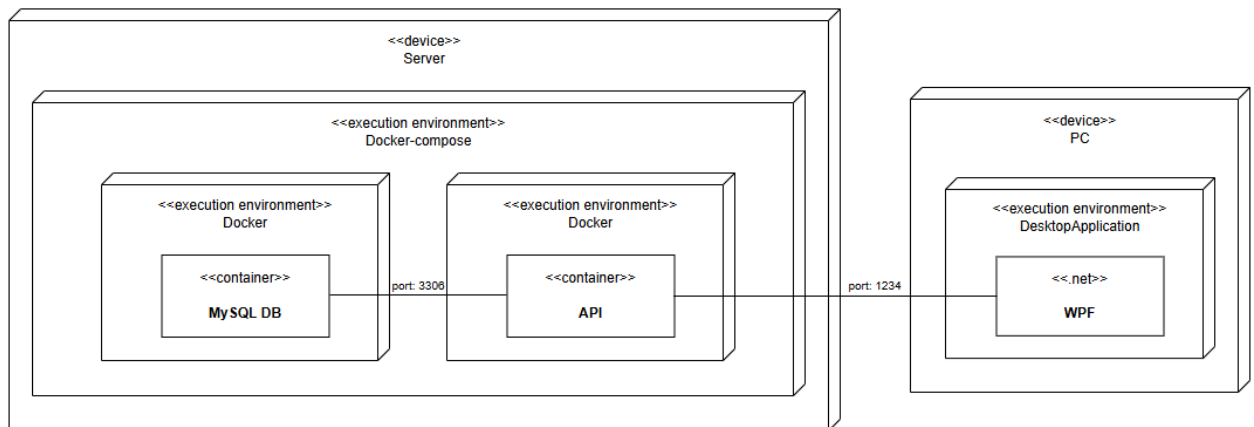


Рисунок 3 – Диаграмма развертывания

## 2.3 Проектирование базы данных

В рамках проектирования подсистемы требуется разработать БД для хранения данных о турнирах, их участниках и турнирах, турнирных сетках и пользователях.

На рисунке 4 в виде ERD показана физическая модель предметной области, созданная при помощи MySQL Workbench.

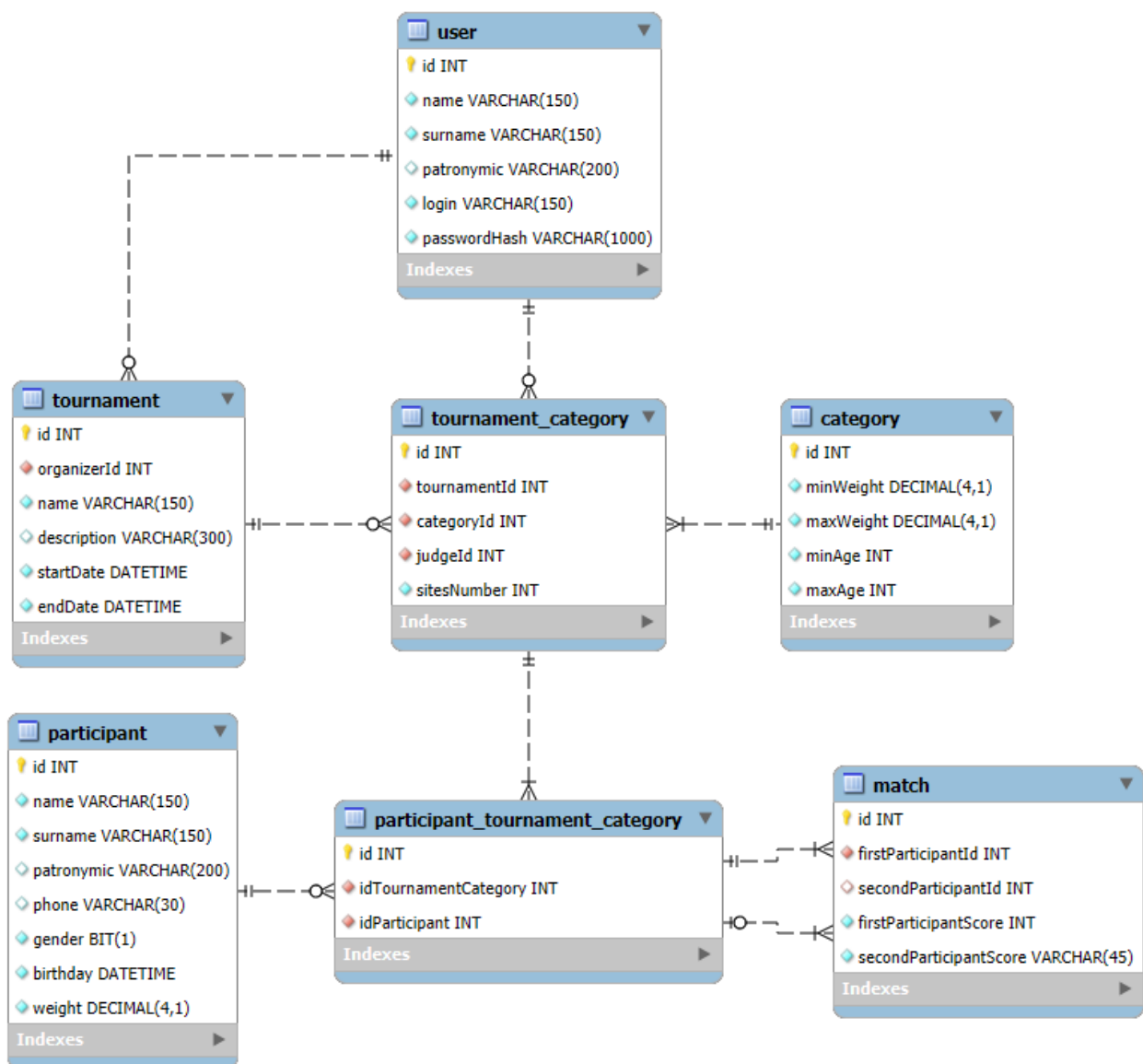


Рисунок 4 – Физическая модель БД

## 3 Разработка и интеграция модулей программного обеспечения

### 3.1 Разработка программных модулей

В ходе проектирования были разработаны Web-API и оконное приложение WPF на C# в Visual Studio 2022.

Взаимодействие приложения с сервером осуществляется при помощи HTTP-запросов к API, ответы возвращаются в виде HTTP-статуса и ссылки на созданный объект. Для реализации HTTP-запросов использован сетевой клиент `http`. Код метода для создания записи о турнире в БД отправкой POST-запроса на сервер представлен листингом 1.

Листинг 1 – Код метода для отправки POST-запроса на сервер

```
[HttpPost]
public async Task<IActionResult> PostTournament(TournamentDto
tournamentDto)
{
    //Создание нового объекта Tournament на основе данных из DTO
    var tournament = new Tournament
    {
        Id = tournamentDto.Id,
        Name = tournamentDto.Name,
        Description = tournamentDto.Description,
        StartDate = tournamentDto.StartDate,
        EndDate = tournamentDto.EndDate,
        Address = tournamentDto.Address,
        OrganizerId = tournamentDto.OrganizerId
    };

    //Добавление турнира в контекст базы данных
    context.Tournaments.Add(tournament);
    //Сохранение изменений в базе данных
    await context.SaveChangesAsync();
    //Возврат ответа 201 Created с ссылкой на созданный ресурс
    return CreatedAtAction("GetTournament", new { id =
tournamentDto.Id }, tournament);
}
```

Для получения списка турниров в приложении разработана функция, представленная листингом 2.

## Листинг 2 – Код функции получения списка турниров

```
[RelayCommand]
private async Task LoadTournaments()
{
    //Установка флага загрузки в true для отображения индикатора
    загрузки
    IsLoading = true;

    try
    {
        // Асинхронное получение списка турниров из сервиса
        var tournaments = await
        _tournamentService.GetAllAsync();
        // Проверка, что полученные данные не null
        if (tournaments is not null)
        {
            _allTournaments.Clear();
            // Добавление каждого турнира в коллекцию с
            проверкой на null
            foreach (var tournament in tournaments)
            {
                if (tournament is not null)
                    _allTournaments.Add(tournament);
            }
            // Применение фильтров к загруженным данным
            ApplyFilters();
        }
        catch (Exception ex)
        {
            // Обработка ошибок с выводом сообщения пользователю
            MessageBox.Show($"Ошибка загрузки турниров:
            {ex.Message}", "Ошибка");
        }
        finally
        {
            // Сброс флага загрузки независимо от результата
            выполнения
            IsLoading = false;
        }
    }
}
```

Изменение информации в БД осуществляется посредством Web-API приложения, код изменения информации о турнире по идентификатору представлен листингом 3

Листинг 3 – Код функции изменения информации о турнире по идентификатору

```
[HttpPut("{id}")]
public async Task<IActionResult> PutTournament(int id,
TournamentDto tournamentDto)
{
    //Проверка соответствия идентификатора в URL и в теле
запроса
    if (id != tournamentDto.Id)
        return BadRequest();

    try
    {
        //Создание объекта Tournament на основе данных из DTO
        var tournament = new Tournament
        {
            Id = tournamentDto.Id,
            Name = tournamentDto.Name,
            Description = tournamentDto.Description,
            StartDate = tournamentDto.StartDate,
            EndDate = tournamentDto.EndDate,
            Address = tournamentDto.Address,
            OrganizerId = tournamentDto.OrganizerId
        };

        //Установка состояния сущности как измененной для
обновления в БД
        context.Entry(tournament).State = EntityState.Modified;
        //Сохранение изменений в базе данных
        await context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        // Обработка ошибки параллельного доступа к данным
        if (!TournamentExists(id))
            return NotFound(); // Турнир не найден
        return BadRequest();
    }

    // Возврат статуса 204 No Content при успешном обновлении
    return NoContent();
}
```

## 3.2 Реализация интерфейса пользователя

Интерфейс приложения реализован с использованием постраничной навигации в статичном главном окне. В приложении разработаны различные элементы управления, стили и карточки для упрощения работы. Навигация в приложении осуществляется с помощью бокового меню, которое обеспечивает переход между страницами.

Для отображения информации о турнире разработана карточка, которая представлена на рисунке 5.

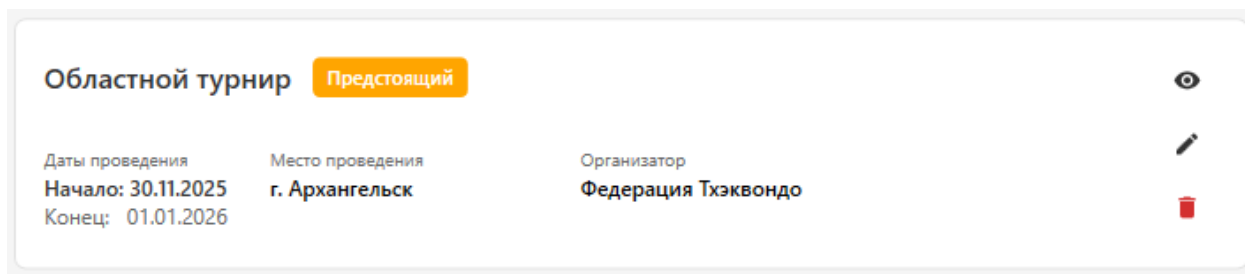


Рисунок 5 – Карточка турнира

Фрагмент кода карточки турнира, образующий кнопки управления, представлен листингом 4.

Листинг 4 – Фрагмент кода карточки турнира

```
<StackPanel Grid.Column="1"
            HorizontalAlignment="Right"
            VerticalAlignment="Top">
    <!--Код для просмотра деталей турнира -->
    <Button Command="{Binding
DataContext.ViewTournamentDetailsCommand,
RelativeSource={RelativeSource AncestorType=ItemsControl}}"
            CommandParameter="{Binding}"
            Style="{StaticResource IconButtonStyle}">
        <!-- Иконка "глаз" для визуального обозначения просмотра -->
        <materialDesign:PackIcon Kind="Eye"/>
    </Button>
```



```

        <!--Кнопка для редактирования турнира -->
        <Button Command="{Binding DataContext.EditTournamentCommand,
RelativeSource={RelativeSource AncestorType=ItemsControl}}"
            CommandParameter="{Binding}"
            Style="{StaticResource IconButtonStyle}">
            <!-- Иконка "карандаш" для визуального обозначения
редактирования -->
            <materialDesign:PackIcon Kind="Pencil"/>
        </Button>

        <!-- Кнопка для удаления турнира -->
        <Button Command="{Binding
DataContext.DeleteTournamentCommand,
RelativeSource={RelativeSource AncestorType=ItemsControl}}"
            CommandParameter="{Binding}"
            Style="{StaticResource IconButtonStyle}"
            Foreground="{StaticResource ErrorBrush}">
            <!-- Иконка "корзина" для визуального обозначения
удаления -->
            <materialDesign:PackIcon Kind="Delete"/>
        </Button>
    </StackPanel>

```

### 3.3 Разграничение прав доступа пользователей

Разграничение прав доступа реализовано разделением организаторов и судей по разным таблицам. Права организатора у пользователя появляются при True значении в поле IsOrganizer.

Пример кода разграничения прав пользователей приложения представлен листингом 5.

#### Листинг 5 – Код разграничения прав пользователей приложения

```

private void InitializeNavigation()
{
    //Инициализация коллекции элементов меню
    MenuItems = new ObservableCollection<MenuItem>();

    //Проверка роли пользователя (организатор или обычный
пользователь)
    if (CurrentUser.IsOrganizer)
    {
        //Добавление пунктов меню для организатора
        MenuItems.Add(new MenuItem("Заявки", new RelayCommand(()
=> Navigate("Applications")), "ClipboardList"));
    }
}

```

```

        MenuItems.Add(new MenuItem("Пользователи", new
RelayCommand(() => Navigate("Users")), "AccountGroup"));
        MenuItems.Add(new MenuItem("Отчёты", new RelayCommand(()
=> Navigate("Reports")), "ChartBar"));
    }
    else
    {
        //Добавление пунктов меню для обычного пользователя
        MenuItems.Add(new MenuItem("Мои площадки", new
RelayCommand(() => Navigate("MyMatches")), "Stadium"));
        MenuItems.Add(new MenuItem("Судьи", new RelayCommand(()
=> Navigate("Judges")), "Gavel"));
    }
}

```

### 3.4 Экспорт и импорт данных

Для автоматизации процесса получения данных об участниках из турнира в приложении реализован экспорт файла в формате xlsx. Код метода экспорта данных об участниках представлен листингом 6.

Листинг 6 – Код метода экспорта данных об участниках.

```

[RelayCommand]
private async Task ExportParticipants()
{
    //Проверка прав доступа: только организаторы могут
экспортировать участников
    if (!IsOrganizer)
    {
        MessageBox.Show("Доступ запрещен. Только организаторы
могут экспортировать участников.", "Ошибка доступа");
        return;
    }

    try
    {
        //Проверка наличия участников для экспорта
        if (!Participants.Any())
        {
            MessageBox.Show("Нет участников для экспорта",
"Экспорт");
            return;
        }

        //Настройка диалога сохранения файла

```

```

        var saveFileDialog = new SaveFileDialog
        {
            Filter = "Excel files (*.xlsx)|*.xlsx|All files (*.*)|*.*",
            Title = "Сохранить файл с участниками",
            FileName = $"Участники_турнира_{DateTime.Now:yyyy-MM-dd}.xlsx"
        };

        //Если пользователь выбрал место сохранения и подтвердил
        if (saveFileDialog.ShowDialog() == true)
        {
            //Включение индикатора загрузки
            IsLoading = true;

            //Асинхронный экспорт данных в Excel файл
            await Task.Run(() =>
            ExportToExcel(saveFileDialog.FileName));

            //Уведомление пользователя об успешном экспорте
            MessageBox.Show($"Участники успешно экспортированы в файл: {saveFileDialog.FileName}", "Экспорт завершен");
        }
    }
    catch (Exception ex)
    {
        //Обработка ошибок при экспорте
        MessageBox.Show($"Ошибка при экспорте участников: {ex.Message}", "Ошибка экспорта");
    }
    finally
    {
        //Выключение индикатора загрузки в любом случае
        IsLoading = false;
    }
}

```

Для автоматизации добавления данных об участниках в турнир реализован импорт из файла в формате xlsx. Фрагмент кода метода импорта данных об участниках представлен листингом 7.

Листинг 7 – Фрагмент кода метода импорта данных об участниках.

```

//Настройка диалога выбора файла для импорта
var openFileDialog = new OpenFileDialog
{
    Filter = "Excel files (*.xlsx;*.xls)|*.xlsx;*.xls|All files (*.*)|*.*",

```

```

        Title = "Выберите файл для импорта участников"
    };

    //Если пользователь выбрал файл и подтвердил выбор
    if (openFileDialog.ShowDialog() == true)
    {
        // Включение индикатора загрузки
        IsLoading = true;

        //Парсинг Excel-файла и получение списка участников
        var importedParticipants = await
        ParseExcelFile(openFileDialog.FileName);

        //Если участники были успешно импортированы
        if (importedParticipants.Any())
        {
            //Добавление каждого участника в основную коллекцию
            foreach (var participant in importedParticipants)
                Participants.Add(participant);

            //Установка флага наличия несохраненных изменений
            HasUnsavedChanges = true;
            MessageBox.Show($"Успешно импортировано
            {importedParticipants.Count} участников", "Импорт завершен");
        }
        else
        {
            MessageBox.Show("Не удалось импортировать участников из
            файла", "Ошибка импорта");
        }
    }
}

```