**МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ И МОЛОДЁЖНОЙ**

**ПОЛИТИКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский арктический горно-политехнический техникум»**

**КУРСОВОЙ ПРОЕКТ**

по МДК 05.01 Проектирование и дизайн информационных систем

**ТЕМА: РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ**

**«Run 2Gether»**

Автор: студент гр.ИСП-22 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ ДЬЯКОВ Л.А./

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Д.П./

(подпись) (Ф.И.О.)

Воркута

2025

**Оглавление**

[**1.1 Платформа .NET 6**](#_Toc195265391)

[**1.2 Язык программирования C# 6**](#_Toc195265392)

[**1.3 Windows Presentation Foundation (WPF) 7**](#_Toc195265393)

[**1.4 СУБД SQL Server 7**](#_Toc195265394)

[**1.5 Microsoft SQL Server Management Studio (SSMS) 8**](#_Toc195265395)

[**1.6 Entity Framework (EF) 8**](#_Toc195265396)

[**2.1 Разработка диаграммы ERD 10**](#_Toc195265397)

[**2.2 Разработка базы данных 12**](#_Toc195265398)

[**3.1 Разработка прототипа информационной системы 19**](#_Toc195265399)

[**3.2 Программирование информационной системы 21**](#_Toc195265400)

[**3.2.1 Разработка модуля «Авторизация» 21**](#_Toc195265401)

[**3.2.2 Разработка модуля «Окно для Организатора» 23**](#_Toc195265402)

[**3.2.3 Разработка модуля «Окно для Тренера» 26**](#_Toc195265403)

[**3.2.4 Разработка модуля «Окно для Участника» 27**](#_Toc195265404)

[**3.2.5 Разработка модуля «Отчет» 27**](#_Toc195265405)

**ВВЕДЕНИЕ**

Современный мир спорта и здорового образа жизни активно развивается, и бег остается одним из самых популярных и доступных видов физической активности. Организация соревнований по бегу требует тщательного планирования, учета участников, обработки документов и управления данными. В настоящее время многие процессы, связанные с проведением таких мероприятий, выполняются вручную, что приводит к значительным временным затратам, возможным ошибкам и неэффективному использованию ресурсов. Автоматизация этих процессов позволит оптимизировать работу организаторов, тренеров и участников, обеспечив удобство, скорость и надежность.

Целью данного курсового проекта является разработка автоматизированной информационной системы «Run 2Gether», предназначенной для планирования и организации соревнований по бегу. Система призвана упростить процессы регистрации участников, учета документов, управления соревнованиями и генерации отчетов, а также обеспечить разграничение прав доступа для различных категорий пользователей: организаторов, тренеров и участников.

Актуальность проекта обусловлена необходимостью внедрения современных информационных технологий в сферу организации спортивных мероприятий. Использование системы «Run 2Gether» позволит сократить время на выполнение рутинных операций, минимизировать ошибки при обработке данных и повысить качество обслуживания участников соревнований.

В рамках проекта были поставлены следующие задачи:

1. Анализ требований к функционалу системы.

2. Выбор инструментария для разработки (платформа .NET, язык C#, СУБД SQL Server).

3. Проектирование базы данных и интерфейса системы.

4. Реализация модулей, включая авторизацию, управление соревнованиями и участниками, а также генерацию отчетов.

5. Тестирование и отладка системы.

Курсовая работа состоит из трех глав. В первой главе обосновывается выбор инструментов разработки, во второй описывается проектирование базы данных, а в третьей — этапы создания информационной системы. Результатом работы станет готовое программное обеспечение, отвечающее требованиям заказчика и способное значительно упростить процесс организации беговых соревнований.

Разрабатываемая система «Run 2Gether» представляет собой практическое решение, которое может быть внедрено в реальных условиях, обеспечивая организаторам и участникам удобство, прозрачность и эффективность взаимодействия.

**ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ**

В данной главе описывается обоснование выбора используемого инструментария для разработки информационной системы «Run 2Gether». Выбор основан на требованиях к производительности, масштабируемости, удобству разработки и доступности ресурсов.

# **1.1 Платформа .NET**

В качестве основной платформы разработки выбрана .NET. Это кросс-платформенная, высокопроизводительная среда выполнения, предлагающая широкий набор библиотек и фреймворков для создания различных типов приложений. Выбор .NET обусловлен следующими преимуществами:

Высокая производительность: .NET обеспечивает высокую скорость выполнения кода, что критично для обработки больших объемов данных, работы в режиме реального времени.

Кросс-платформенность: .NET позволяет создавать приложения, работающие на различных операционных системах (Windows, Linux, macOS), что обеспечивает гибкость и расширяемость системы в будущем .

Большое сообщество разработчиков: Наличие огромного сообщества обеспечивает доступ к обширной документации, библиотекам с открытым исходным кодом и поддержке.

Широкий спектр инструментов и фреймворков: .NET предлагает множество инструментов и фреймворков для упрощения разработки, включая ASP.NET для веб-приложений, .NET MAUI для кросс-платформенных мобильных приложений и многое другое.

Поддержка Microsoft: .NET активно поддерживается Microsoft, что гарантирует долгосрочную стабильность и развитие платформы.

# **1.2 Язык программирования C#**

Для разработки выбрали язык программирования C#. Это современный, объектно-ориентированный язык, разработанный Microsoft для платформы .NET. Основные причины выбора:

Простота и читаемость: C# обладает ясным и понятным синтаксисом, что упрощает разработку, отладку и поддержку кода.

Объектно-ориентированное программирование: Поддержка принципов ООП (инкапсуляция, наследование, полиморфизм) позволяет создавать модульный, масштабируемый и легко поддерживаемый код.

Богатый набор библиотек: C# предоставляет доступ к обширной библиотеке классов .NET Framework, что значительно ускоряет процесс разработки.

Интеграция с .NET: C# тесно интегрирован с платформой .NET, что обеспечивает бесшовную работу с другими компонентами и фреймворками.

Сильная типизация: Сильная типизация помогает предотвращать ошибки на этапе компиляции и повышает надежность кода.

# **1.3 Windows Presentation Foundation (WPF)**

Для создания пользовательского интерфейса выбрана технология WPF (Windows Presentation Foundation). WPF позволяет создавать современные, интерактивные и визуально привлекательные графические интерфейсы. Преимущества WPF:

XAML: Использование XAML (Extensible Application Markup Language) для описания пользовательского интерфейса обеспечивает разделение логики приложения и дизайна, упрощая разработку и поддержку.

Data Binding: Механизм привязки данных позволяет легко связать элементы интерфейса с данными, упрощая обновление интерфейса при изменении данных.

Поддержка различных типов элементов управления: WPF предоставляет широкий набор элементов управления для создания разнообразных и сложных интерфейсов.

Поддержка 2D и 3D графики: WPF позволяет создавать приложения с богатой графикой и анимацией.

Интеграция с .NET: WPF тесно интегрируется с платформой .NET, обеспечивая удобство разработки и доступа к другим компонентам.

# **1.4 СУБД SQL Server**

В качестве системы управления базами данных (СУБД) выбран Microsoft SQL Server. Этот выбор обусловлен:

Надежность и производительность: SQL Server обеспечивает высокую надежность и производительность для обработки больших объемов данных.

Масштабируемость: SQL Server легко масштабируется для удовлетворения растущих потребностей приложения.

Безопасность: SQL Server предоставляет мощные механизмы безопасности для защиты данных.

Интеграция с .NET: SQL Server легко интегрируется с платформой .NET, что упрощает работу с данными из приложения.

Широкие возможности: SQL Server предлагает богатый набор функций и инструментов для управления данными.

# **1.5 Microsoft SQL Server Management Studio (SSMS)**

SSMS — это бесплатное графическое приложение от Microsoft, предназначенное для администрирования и управления базами данных SQL Server. Его использование в проекте обосновано следующими преимуществами:

Управление базами данных: SSMS позволяет создавать, изменять и удалять базы данных, таблицы, представления, хранимые процедуры и другие объекты базы данных. Это обеспечивает полный контроль над структурой и данными.

Запросы и анализ данных: Встроенный редактор запросов (T-SQL) позволяет создавать и выполнять SQL-запросы для извлечения, обновления и удаления данных. Результаты запросов отображаются в удобном формате, что упрощает анализ данных.

Администрирование сервера: SSMS предоставляет инструменты для мониторинга производительности сервера, управления пользователями и ролями, резервного копирования и восстановления баз данных, а также для выполнения других задач администрирования.

Графический интерфейс: Интуитивно понятный графический интерфейс упрощает работу с базами данных, даже для пользователей с ограниченным опытом работы с SQL.

Интеграция с другими инструментами: SSMS хорошо интегрируется с другими инструментами Microsoft, такими как Visual Studio, что облегчает разработку и развертывание приложений.

# 

# **1.6 Entity Framework (EF)**

Entity Framework— это объектно-реляционное отображение (ORM) от Microsoft, которое упрощает взаимодействие приложений .NET с базами данных SQL Server (и другими СУБД). Его применение в проекте оправдано следующими факторами:

Абстракция от базы данных: EF позволяет работать с данными на уровне объектов .NET, без необходимости писать SQL-запросы вручную. Это упрощает разработку и повышает производительность разработчиков.

Удобство работы с данными: EF предоставляет удобный API для выполнения операций CRUD (создание, чтение, обновление, удаление) над данными.

Поддержка различных СУБД: Хотя изначально EF был ориентирован на SQL Server, он поддерживает и другие СУБД, позволяя легко переносить приложение на другую платформу.

Code First подход: EF Core позволяет создавать схему базы данных на основе модели данных, определенной в коде приложения. Это упрощает разработку и миграцию базы данных.

Интеграция с Visual Studio: EF тесно интегрируется с Visual Studio, что упрощает разработку и отладку.

**ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ**

# **2.1 Разработка диаграммы ERD**

ER-диаграммы: ключевые аспекты и применение

ER-диаграмма(Entity-Relationship Diagram) — это графическая модель структуры базы данных, которая отображает сущности, их атрибуты и взаимосвязи.

**Основные компоненты:**

- Сущности— объекты или понятия, имеющие значение в системе (например, "Пользователь", "Заказ").

- Атрибуты— характеристики сущностей (например, "Имя пользователя", "Дата заказа").

- Связи— взаимодействия между сущностями (например, "Пользователь совершает Заказ").

**Типы связей:**

1. Один-к-одному (1:1) — одна запись в сущности А соответствует одной записи в сущности В.

2. Один-ко-многим (1:N) — одной записи в сущности А могут соответствовать несколько записей в сущности В.

3. Многие-ко-многим (M:N) — множественные связи между сущностями (реализуется через промежуточную таблицу).

Нотация "Crows Foot"

Нотация "Воронья лапка" — популярный способ визуализации связей. Символы на концах линий указывают на тип связи:

- Одинарная черта — "один".

- "Воронья лапка" — "многие".

- Круг — "ноль" (необязательная связь).

**Уровни детализации:**

1. Концептуальный — общее представление предметной области без технических деталей.

2. Логический — детализированные сущности, атрибуты и связи.

3. Физический — соответствует конкретной СУБД (типы данных, индексы и т. д.).

Роль в проектировании БД

**ER-диаграммы:**

- Упрощают понимание структуры данных.

- Служат основой для создания реляционных таблиц.

- Помогают выявить избыточность и оптимизировать хранение информации.

Использование ER-моделирования ускоряет разработку и обеспечивает целостность базы данных.

**Связи между сущностями:**

- User связан с Role через поле Role\_id (один пользователь имеет одну роль).

- Competition связан с User через поле User\_id (соревнование организовано пользователем).

- Participant связан с User через поле User\_id (участник может быть зарегистрирован пользователем).

- ParticipantLog связан с Participant и Competition через поля Participant\_id и Competition\_id (лог содержит информацию об участнике и соревновании).

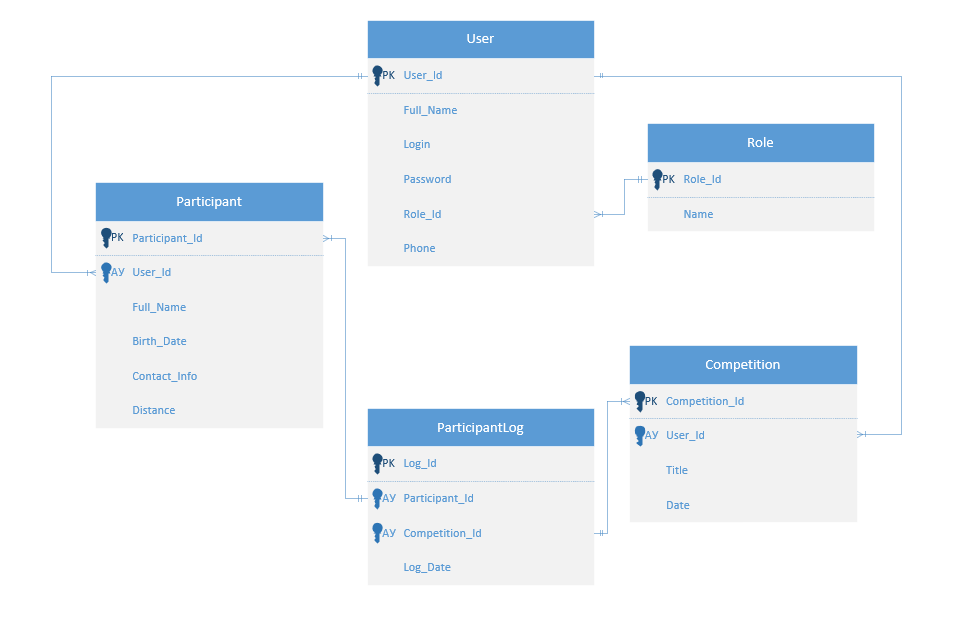
**Возможные улучшения:**

- Использовать стандартные обозначения для первичных (PK) и внешних (FK) ключей.

- Указать типы данных для атрибутов.

- Добавить обязательные (NOT NULL) и опциональные поля.

- Уточнить назначение поля Distance в сущности Participant.

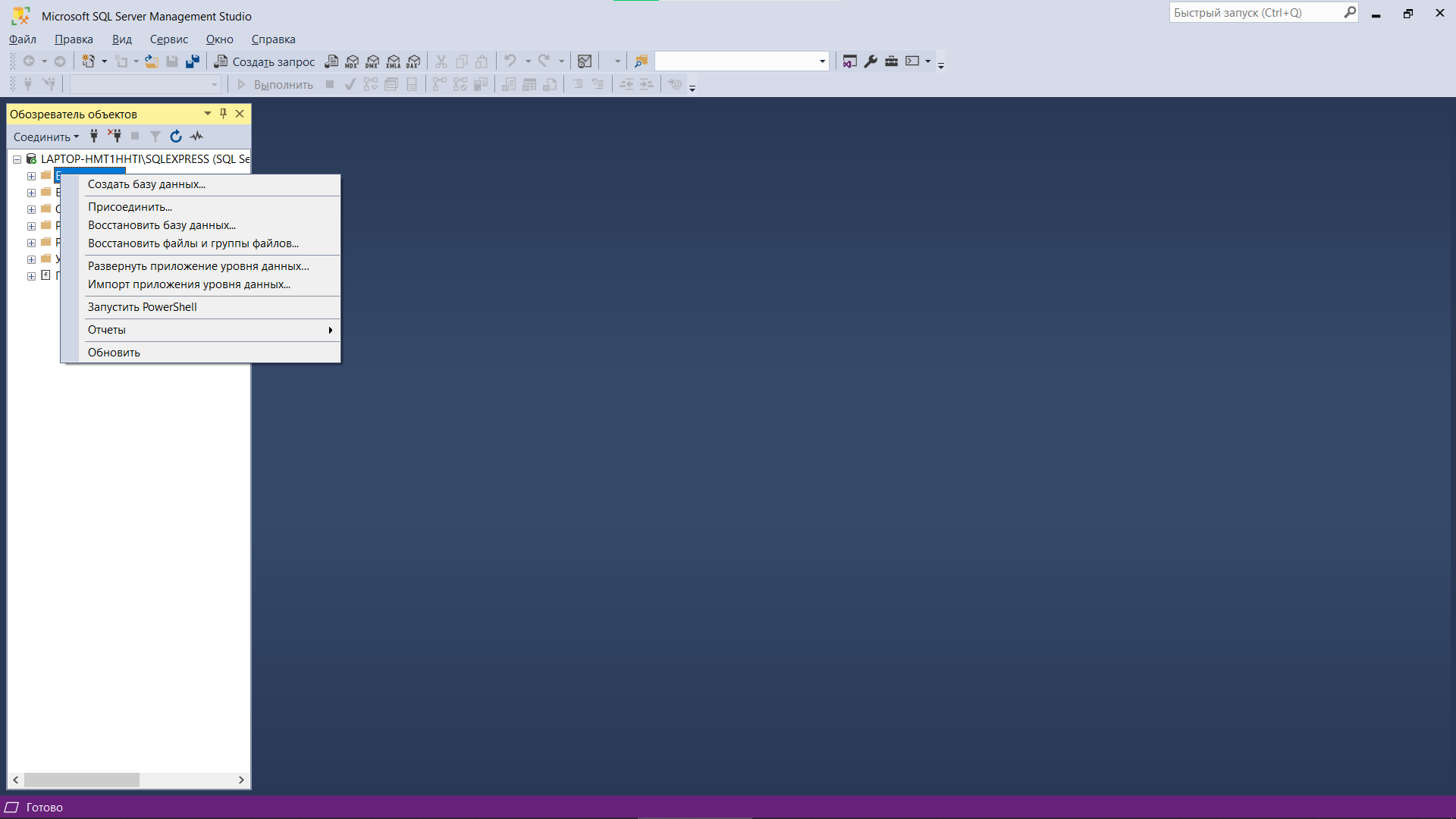
****

*Рис. 2.1.1 ER-диаграмма «Run 2Gether»*

# **2.2 Разработка базы данных**

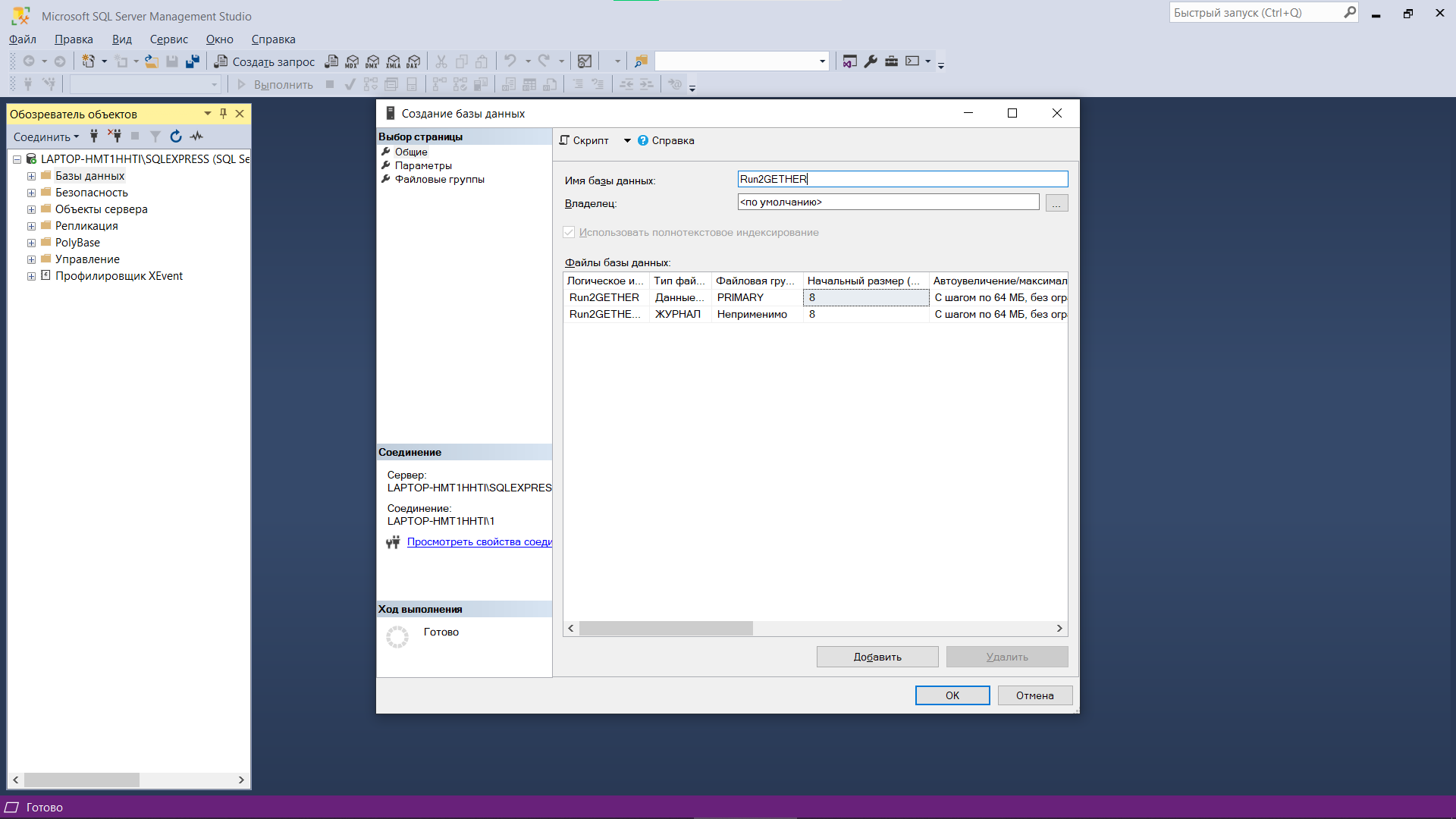
SQL Server Management Studio (SSMS) давно зарекомендовал себя как основной инструмент для взаимодействия с SQL Server. Эта интегрированная среда объединяет в себе все необходимые функции для администрирования, разработки и анализа производительности баз данных.

SSMS предоставляет разработчикам и администраторам удобный графический интерфейс для выполнения повседневных задач. С его помощью можно не только писать и выполнять SQL-запросы, но и полностью управлять инфраструктурой сервера. Особенно ценными являются встроенные средства для анализа планов выполнения запросов, позволяющие находить узкие места в работе базы данных.



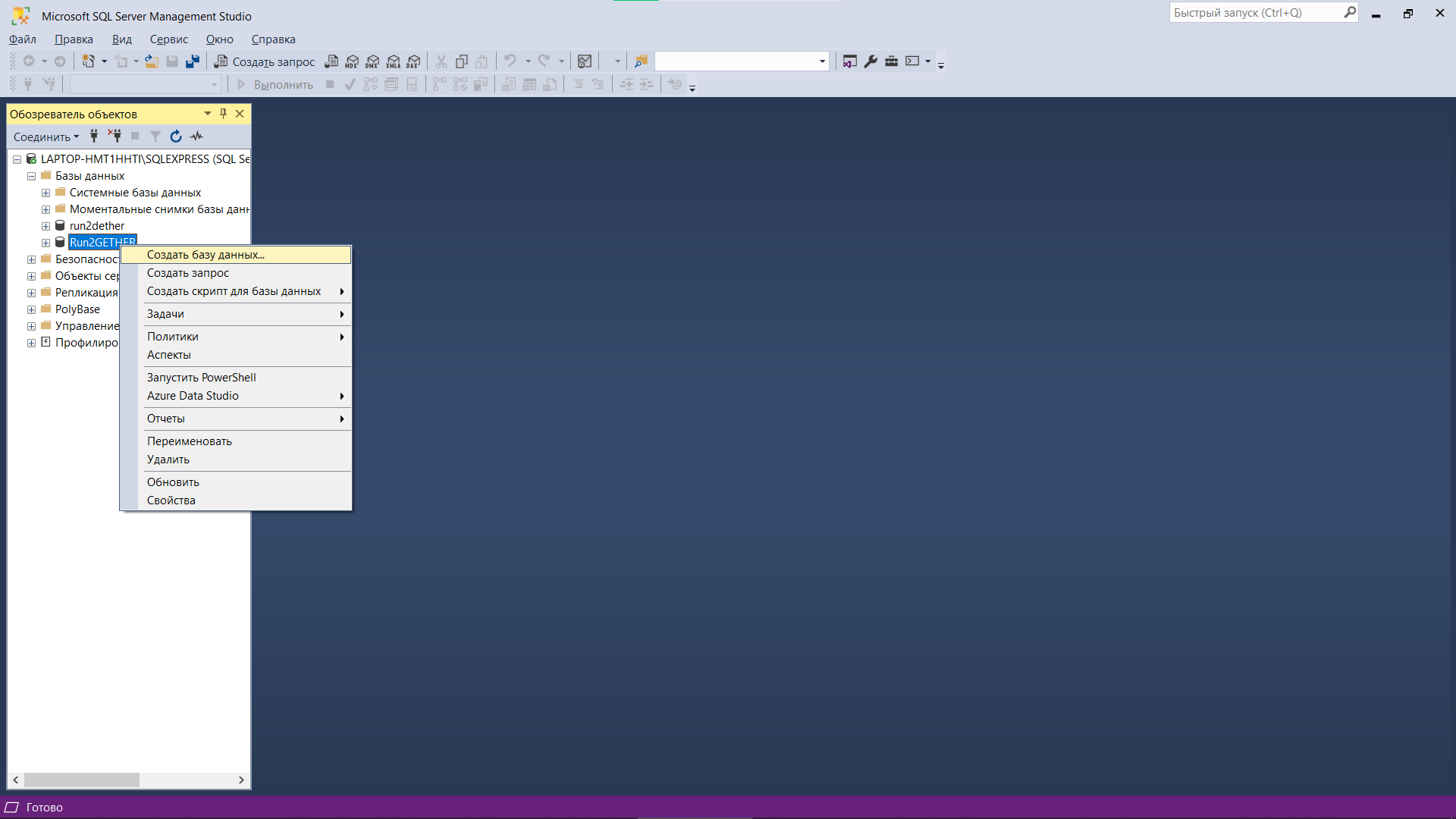
*Рис. 2.2.1 Создание базы данных*

На изображении показан интерфейс Microsoft SQL Server Management Studio (SSMS) с открытым меню управления базами данных. Интерфейс позволяет создавать базы данных, таблицы, а также управлять резервным копированием и восстановлением.



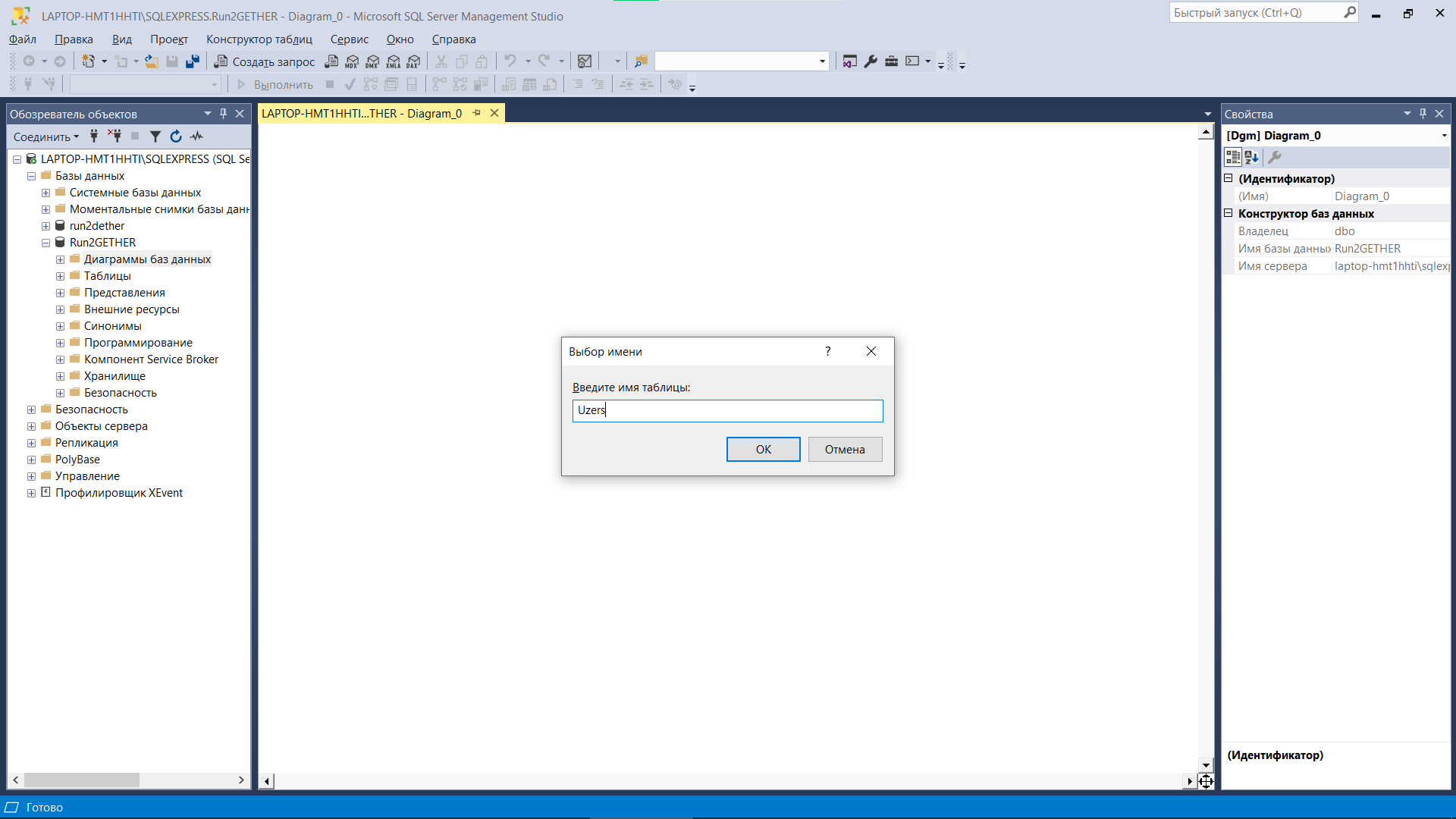
*Рис. 2.2.2 Создание базы данных «Run 2Gether»*

На изображении представлено окно создания новой базы данных в Microsoft SQL Server Management Studio. Основные параметры базы данных включают имя «Run 2Gether» и владельца по умолчанию.



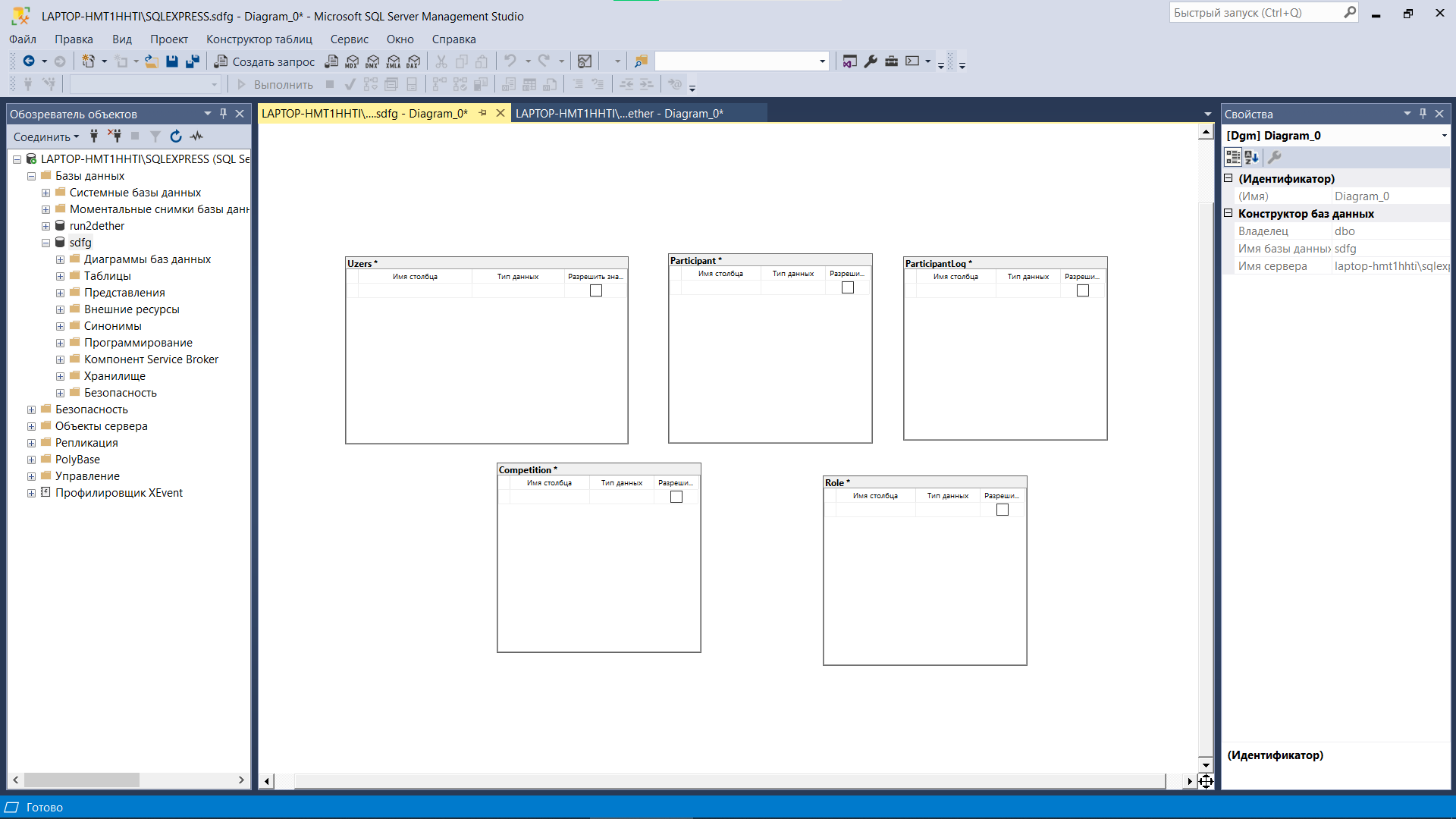
*Рис. 2.2.3 Создание диаграммы базы данных*

На изображении показан интерфейс создания диаграммы базы данных в Microsoft SQL Server Management Studio (SSMS). В левой части экрана видно меню Object Explorer с развернутым разделом выбранной базы данных «Run 2Gether». Этот инструмент позволяет визуально проектировать структуру базы данных, отображая таблицы, их поля, связи и ключи в графическом виде.



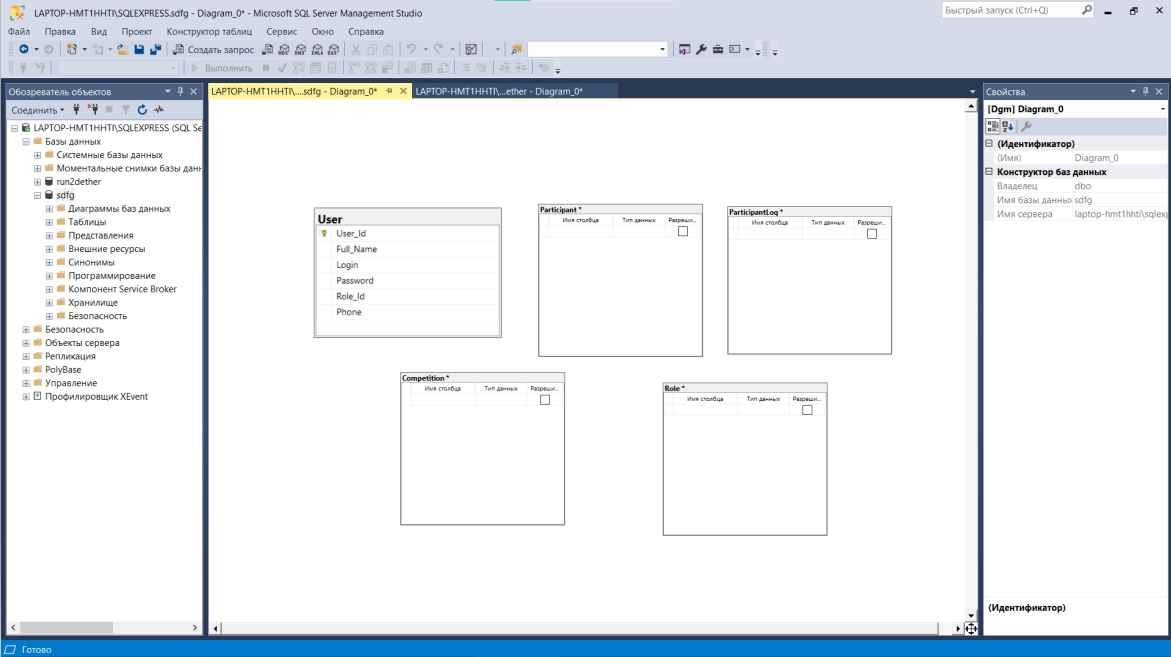
*Рис. 2.2.4 Создание таблицы «Users»*

На изображении представлено диалоговое окно Microsoft SQL Server Management Studio для создания новой таблицы в режиме проектирования диаграммы базы данных.



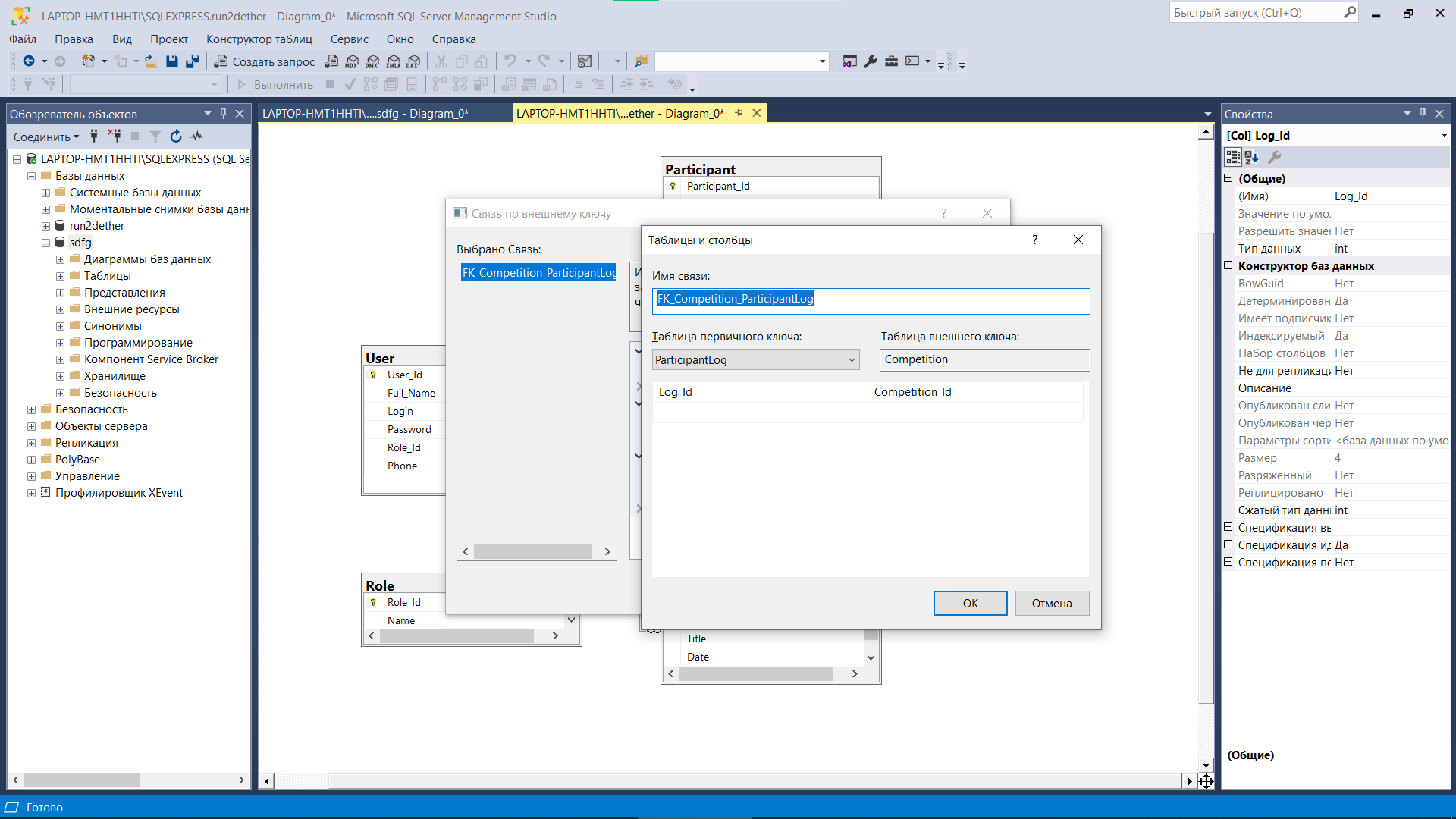
*Рис. 2.2.5 Все таблицы базы данных*

На изображении показан интерфейс проектирования базы данных в Microsoft SQL Server Management Studio, где представлена диаграмма с таблицами: User, Role,Participant, ParticipantLog и Competition.



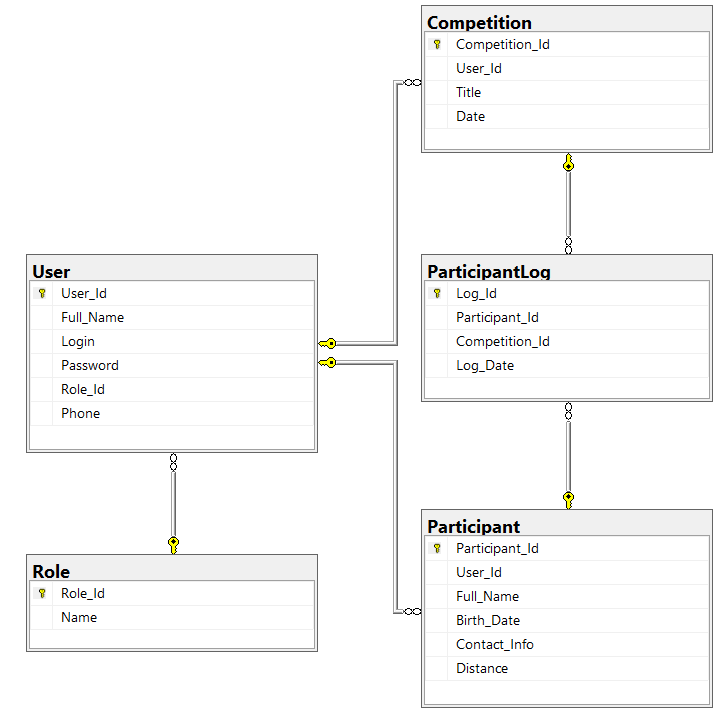
*Рис. 2.2.6 Заполненная таблица «User»*

На изображении представлена таблица базы данных, заполненная в соответствии со структурой, определенной в ER-диаграмме.



*Рис. 2.2.7 Создание связи между таблицами*

На изображении отображен интерфейс создания связи между таблицами базы данных.

****

*Рис. 2.2.8 Готовая диаграмма базы данных*

Данная база данных предназначена для системы управления соревнованиями (Competition) и участниками (Participant). Она состоит из 5 взаимосвязанных таблиц:

1. Competition (Соревнования)

- Competition\_Id уникальный идентификатор соревнования

- User\_Id связь с организатором (из таблицы User)

- Title название соревнования

- Date дата проведения

2. User (Пользователи)

- User\_Id уникальный идентификатор пользователя

- Full\_Name полное имя

- Login логин для входа

- Password пароль

- Role\_Id связь с ролью (из таблицы Role)

- Phone контактный телефон

3. ParticipantLog (Логи участников)

- Log\_Id уникальный идентификатор записи

- Participant\_Id связь с участником

- Competition\_Id связь с соревнованием

- Log\_Date дата/время записи

4. Role (Роли)

- Role\_Id уникальный идентификатор роли

- Name название роли (например, "Администратор", "Организатор", "Участник")

5. Participant (Участники)

- Participant\_Id уникальный идентификатор участника

- User\_Id связь с пользователем (если участник зарегистрирован в системе)

- Full\_Name полное имя участника

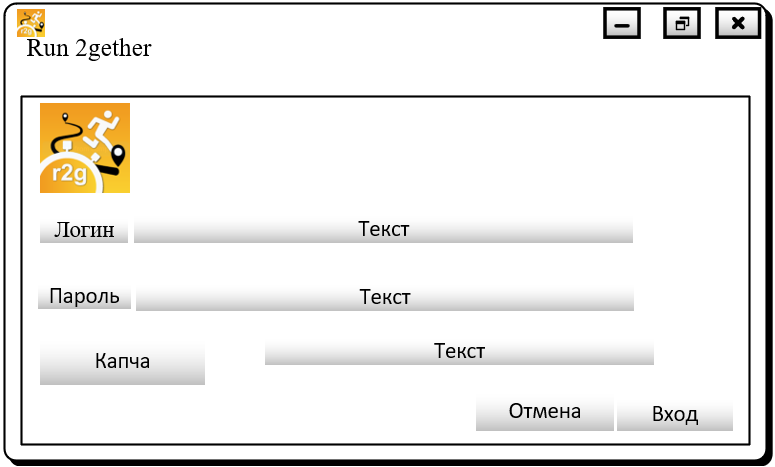
- Birth\_Date дата рождения

- Contact\_info контактная информация

- Distance дистанция/категория участия

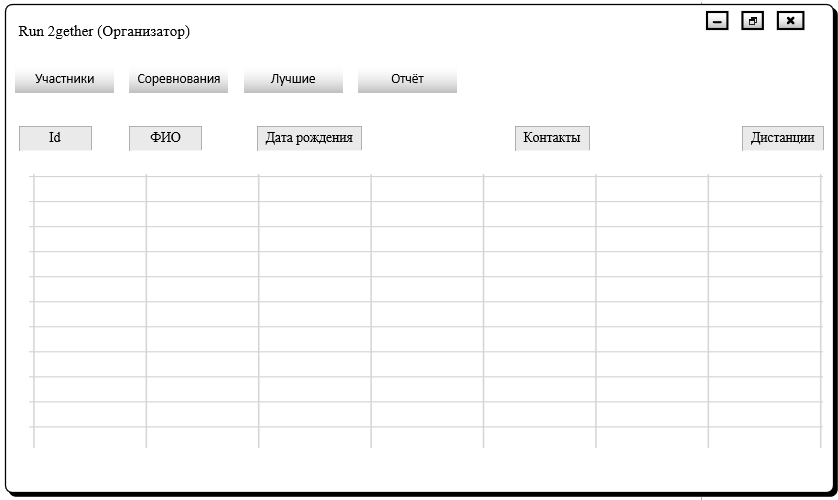
**ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ**

# **3.1 Разработка прототипа информационной системы**

****

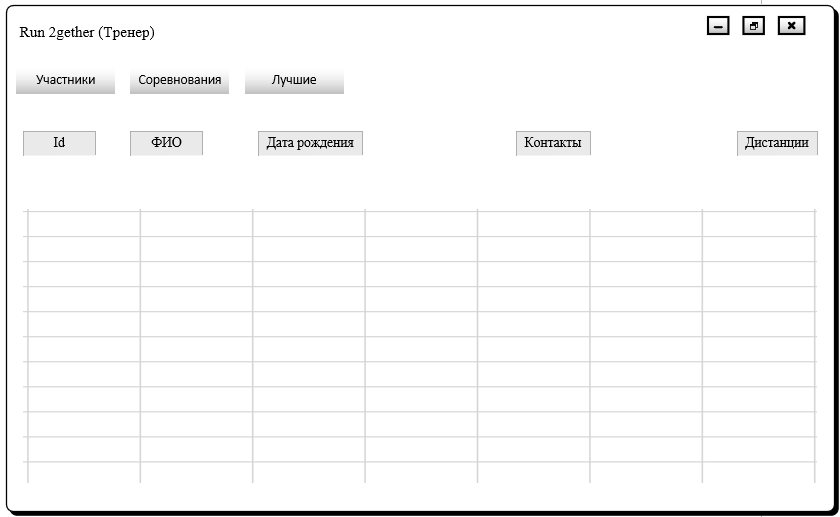
*Рис. 3.1.1 «Прототип регистрации»*

На рисунке предоставлено окно авторизации, с полями для Логина и Пароля, а также кнопки Отмена и Вход.

****

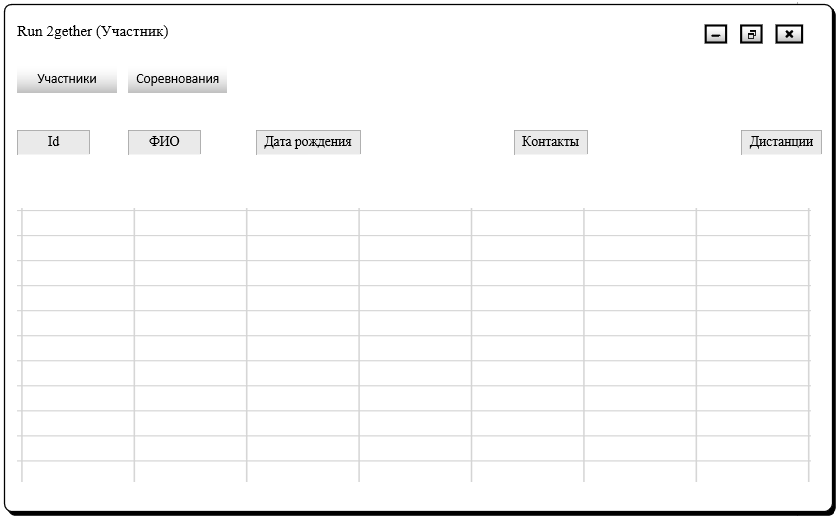
*Рис. 3.1.2 «Прототип окна Организатора»*

На рисунке показанно окно для Организатора на котором есть все предоставлные функции особенно отчёт так он доступен только Организатору.

****

*Рис. 3.1.3 «Прототип окна Тренера»*

На рисунке показанно окно для Тренера на котором есть все предоставлные функции кроме отчёта.

****

*Рис. 3.1.4 «Прототип окна Участника»*

На рисунке показанно окно для Участника на котором есть такие функции как просмотр участников и соревнований.

# **3.2 Программирование информационной системы**

# **3.2.1 Разработка модуля «Авторизация»**

*Рис. 3.2.1.1 «Окно Авторизация»*

public partial class MainWindow : Window

{

private readonly User \_currentUser;

private readonly run2detherEntities \_dbContext;

public MainWindow(User user)

{

InitializeComponent();

\_currentUser = user;

\_dbContext = new run2detherEntities();

this.Title = $"Run2gether - {user.Full\_Name} ({user.Role.Name})";

statusBar.Text = $"Добро пожаловать, {\_currentUser.Full\_Name}! Ваша роль: {\_currentUser.Role.Name}";

ApplyRolePermissions();

LoadInitialData();

}

private void ApplyRolePermissions()

{

string roleName = \_currentUser.Role.Name;

btnParticipants.IsEnabled = false;

btnCompetitions.IsEnabled = false;

btnLogs.IsEnabled = false;

PartItem.IsEnabled = false;

LogItem.IsEnabled = false;

btnReports.IsEnabled = false;

switch (roleName)

{

case "Организатор":

btnParticipants.IsEnabled = true;

btnCompetitions.IsEnabled = true;

btnLogs.IsEnabled = true;

btnReports.IsEnabled = true;

PartItem.IsEnabled = true;

LogItem.IsEnabled = true;

break;

case "Тренер":

btnParticipants.IsEnabled = true;

btnCompetitions.IsEnabled = true;

btnLogs.IsEnabled = true;

LogItem.IsEnabled = true;

PartItem.IsEnabled = true;

break;

case "Участник":

btnCompetitions.IsEnabled = true;

btnParticipants.IsEnabled = true;

PartItem.IsEnabled = true;

break;

default:

MessageBox.Show("Неизвестная роль пользователя.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

this.Close();

return;

}

}

private void LoadInitialData()

{

LoadParticipantsData();

LoadCompetitionsData();

LoadLogsData();

}

private void btnParticipants\_Click(object sender, RoutedEventArgs e)

{

tabControl.SelectedIndex = 0;

LoadParticipantsData();

}

private void btnCompetitions\_Click(object sender, RoutedEventArgs e)

{

tabControl.SelectedIndex = 1;

LoadCompetitionsData();

}

private void btnLogs\_Click(object sender, RoutedEventArgs e)

{

tabControl.SelectedIndex = 2;

LoadLogsData();

}

private void LoadParticipantsData()

{

participantsGrid.ItemsSource = \_dbContext.Participant

.Include("User")

.ToList();

}

private void LoadCompetitionsData()

{

competitionsGrid.ItemsSource = \_dbContext.Competition

.Include("User")

.ToList();

}

private void LoadLogsData()

{

logsGrid.ItemsSource = \_dbContext.ParticipantLog

.Include("Participant")

.Include("Competition")

.ToList();

}

private void btnReports\_Click(object sender, RoutedEventArgs e)

{

var saveFileDialog = new Microsoft.Win32.SaveFileDialog

{

FileName = "Отчет",

DefaultExt = ".docx",

Filter = "Word documents (.docx)|\*.docx"

};

if (saveFileDialog.ShowDialog() == true)

{

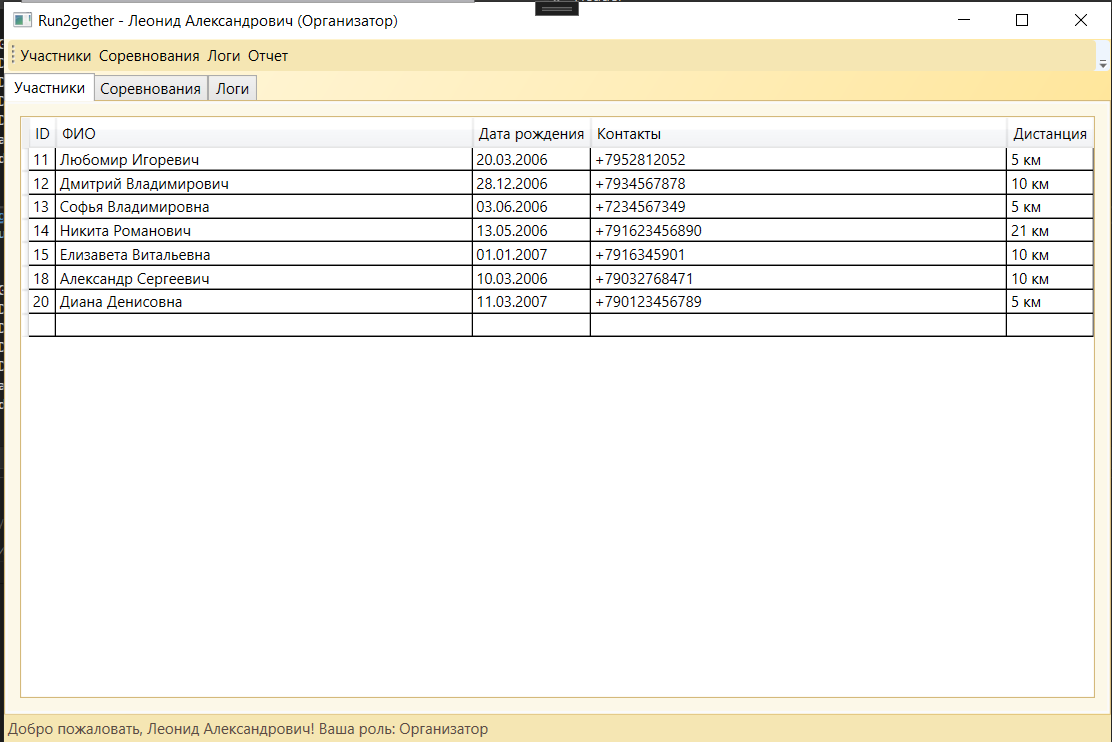
ExportWord.ExportToWord(saveFileDialog.FileName);

}

}

}

# **3.2.2 Разработка модуля «Окно для Организатора»**

****

*Рис. 3.2.2.1 «Окно для Организатора»*

public partial class MainWindow : Window

{

private readonly User \_currentUser;

private readonly run2detherEntities \_dbContext;

public MainWindow(User user)

{

InitializeComponent();

\_currentUser = user;

\_dbContext = new run2detherEntities();

this.Title = $"Run2gether - {user.Full\_Name} ({user.Role.Name})";

statusBar.Text = $"Добро пожаловать, {\_currentUser.Full\_Name}! Ваша роль: {\_currentUser.Role.Name}";

ApplyRolePermissions();

LoadInitialData();

}

private void ApplyRolePermissions()

{

string roleName = \_currentUser.Role.Name;

btnParticipants.IsEnabled = false;

btnCompetitions.IsEnabled = false;

btnLogs.IsEnabled = false;

PartItem.IsEnabled = false;

LogItem.IsEnabled = false;

btnReports.IsEnabled = false;

switch (roleName)

{

case "Организатор":

btnParticipants.IsEnabled = true;

btnCompetitions.IsEnabled = true;

btnLogs.IsEnabled = true;

btnReports.IsEnabled = true;

PartItem.IsEnabled = true;

LogItem.IsEnabled = true;

break;

case "Тренер":

btnParticipants.IsEnabled = true;

btnCompetitions.IsEnabled = true;

btnLogs.IsEnabled = true;

LogItem.IsEnabled = true;

PartItem.IsEnabled = true;

break;

case "Участник":

btnCompetitions.IsEnabled = true;

btnParticipants.IsEnabled = true;

PartItem.IsEnabled = true;

break;

default:

MessageBox.Show("Неизвестная роль пользователя.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

this.Close();

return;

}

}

private void LoadInitialData()

{

LoadParticipantsData();

LoadCompetitionsData();

LoadLogsData();

}

private void btnParticipants\_Click(object sender, RoutedEventArgs e)

{

tabControl.SelectedIndex = 0;

LoadParticipantsData();

}

private void btnCompetitions\_Click(object sender, RoutedEventArgs e)

{

tabControl.SelectedIndex = 1;

LoadCompetitionsData();

}

private void btnLogs\_Click(object sender, RoutedEventArgs e)

{

tabControl.SelectedIndex = 2;

LoadLogsData();

}

private void LoadParticipantsData()

{

participantsGrid.ItemsSource = \_dbContext.Participant

.Include("User")

.ToList();

}

private void LoadCompetitionsData()

{

competitionsGrid.ItemsSource = \_dbContext.Competition

.Include("User")

.ToList();

}

private void LoadLogsData()

{

logsGrid.ItemsSource = \_dbContext.ParticipantLog

.Include("Participant")

.Include("Competition")

.ToList();

}

private void btnReports\_Click(object sender, RoutedEventArgs e)

{

var saveFileDialog = new Microsoft.Win32.SaveFileDialog

{

FileName = "Отчет",

DefaultExt = ".docx",

Filter = "Word documents (.docx)|\*.docx"

};

if (saveFileDialog.ShowDialog() == true)

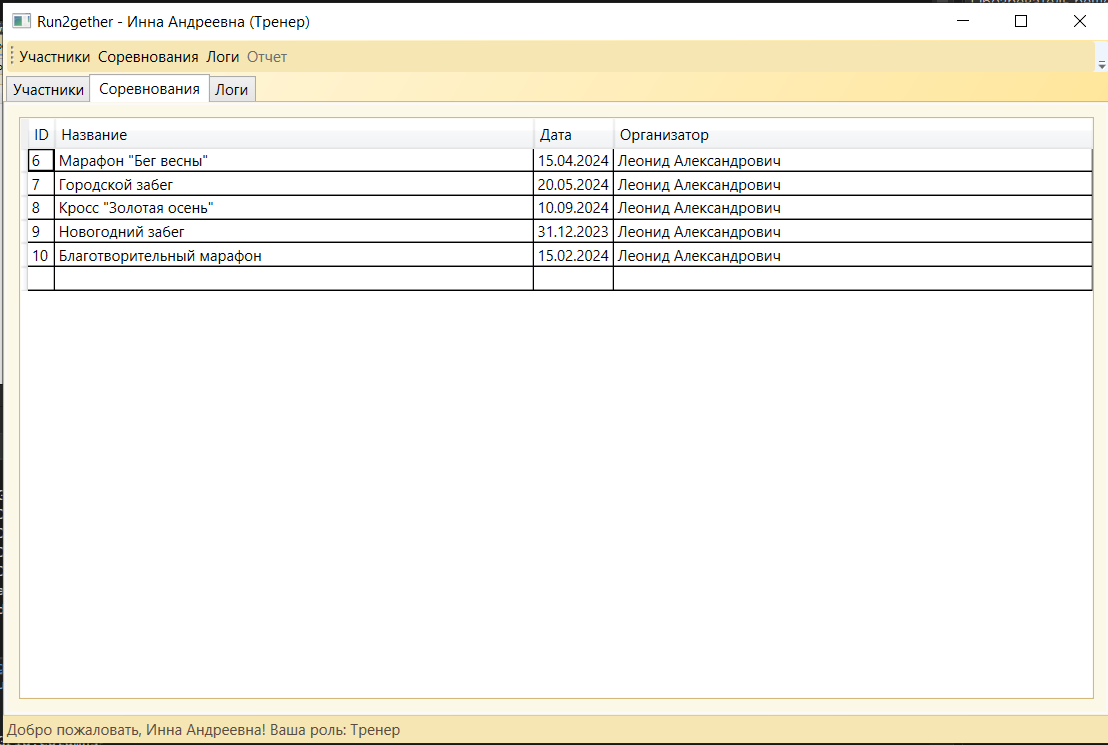
{

ExportWord.ExportToWord(saveFileDialog.FileName);

}

}

# **3.2.3 Разработка модуля «Окно для Тренера»**

****

*Рис. 3.2.3.1 «Окно Тренера»*

public partial class MainWindow : Window

{

private readonly User \_currentUser;

private readonly run2detherEntities \_dbContext;

public MainWindow(User user)

{

InitializeComponent();

\_currentUser = user;

\_dbContext = new run2detherEntities();

this.Title = $"Run2gether - {user.Full\_Name} ({user.Role.Name})";

statusBar.Text = $"Добро пожаловать, {\_currentUser.Full\_Name}! Ваша роль: {\_currentUser.Role.Name}";

ApplyRolePermissions();

LoadInitialData();

}

case "Тренер":

btnParticipants.IsEnabled = true;

btnCompetitions.IsEnabled = true;

btnLogs.IsEnabled = true;

LogItem.IsEnabled = true;

PartItem.IsEnabled = true;

break;

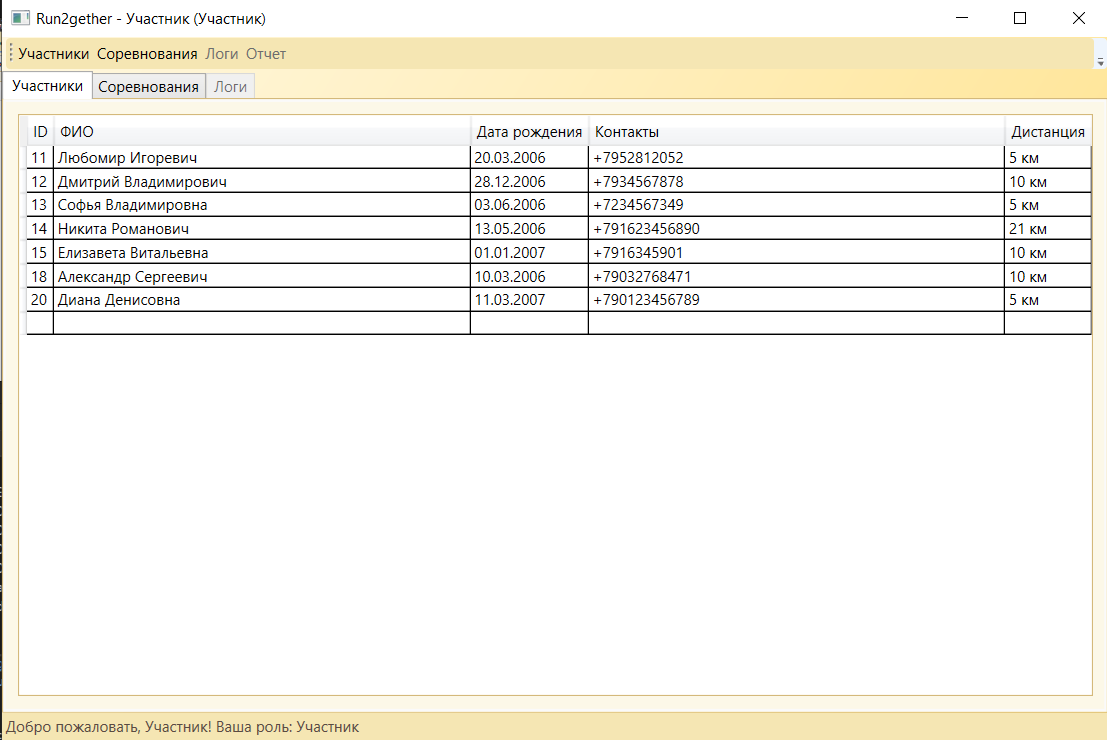
default:

MessageBox.Show("Неизвестная роль пользователя.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

this.Close();

return;

# **3.2.4 Разработка модуля «Окно для Участника»**

****

*Рис. 3.2.4.1 «Окно Участника»*

public partial class MainWindow : Window

{

private readonly User \_currentUser;

private readonly run2detherEntities \_dbContext;

public MainWindow(User user)

{

InitializeComponent();

\_currentUser = user;

\_dbContext = new run2detherEntities();

this.Title = $"Run2gether - {user.Full\_Name} ({user.Role.Name})";

statusBar.Text = $"Добро пожаловать, {\_currentUser.Full\_Name}! Ваша роль: {\_currentUser.Role.Name}";

ApplyRolePermissions();

LoadInitialData();

}

case "Участник":

btnCompetitions.IsEnabled = true;

btnParticipants.IsEnabled = true;

PartItem.IsEnabled = true;

break;

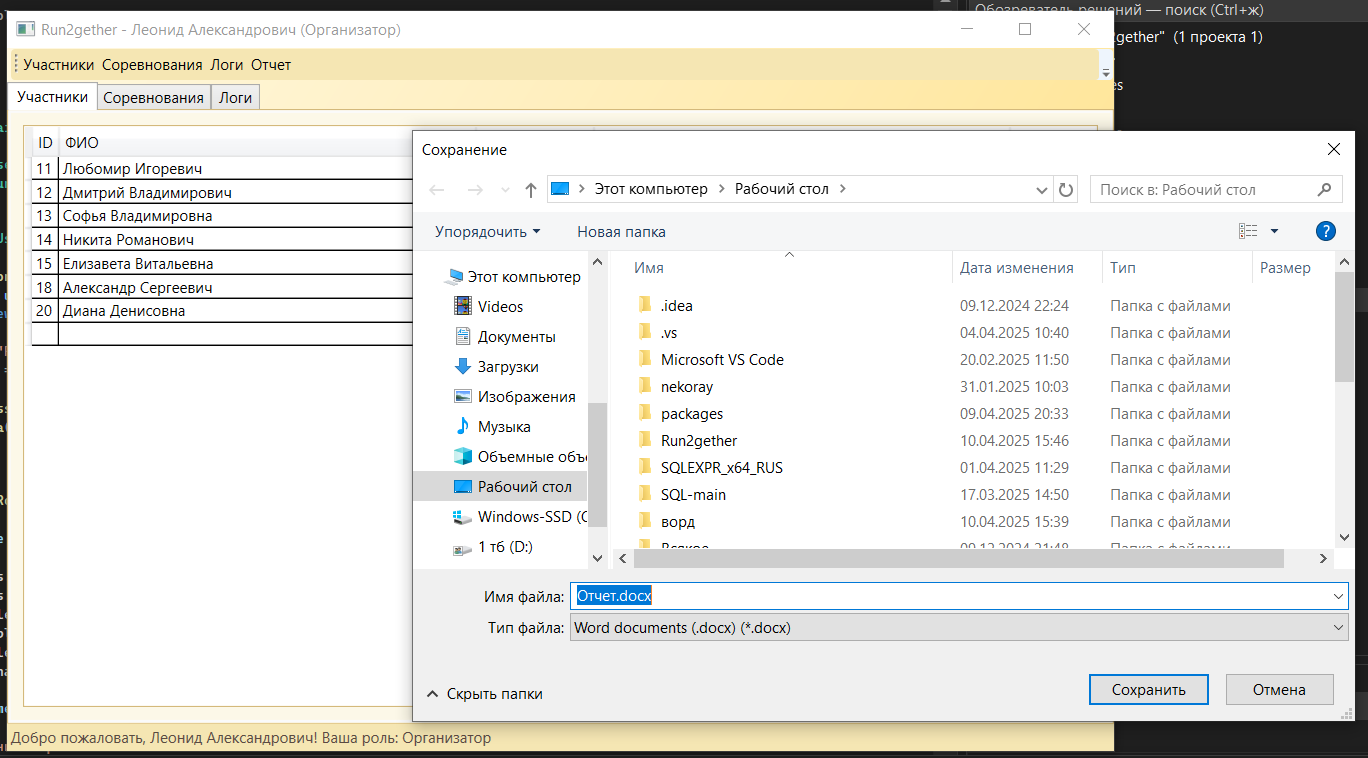
default:

MessageBox.Show("Неизвестная роль пользователя.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

this.Close();

return;

# **3.2.5 Разработка модуля «Отчет»**

****

*Рис. 3.2.5.1 «Отчёт»*

namespace Run2gether

{

class ExportWord

{

public static void ExportToWord(string filePath)

{

try

{

using (var context = new run2detherEntities())

{

var data = new

{

Participants = context.Participant.ToList(),

Competitions = context.Competition.Include("User").ToList(),

Users = context.User.Include("Role").ToList(),

ParticipantLogs = context.ParticipantLog.Include("Participant").Include("Competition").ToList(),

Roles = context.Role.ToList()

};

if (File.Exists(filePath)) File.Delete(filePath);

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

wordApp.Visible = false;

Word.Document doc = wordApp.Documents.Add();

SetDocumentStyles(doc);

AddTitle(doc, "Отчет системы Run2gether");

ExportParticipantsToWord(doc, data.Participants);

AddPageBreak(doc);

ExportCompetitionsToWord(doc, data.Competitions);

AddPageBreak(doc);

ExportUsersToWord(doc, data.Users);

AddPageBreak(doc);

ExportParticipantLogsToWord(doc, data.ParticipantLogs);

AddPageBreak(doc);

ExportRolesToWord(doc, data.Roles);

doc.SaveAs2(filePath);

doc.Close();

wordApp.Quit();

ReleaseWordObjects(doc, wordApp);

OpenExportedFile(filePath);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при экспорте в Word: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private static void OpenExportedFile(string filePath)

{

try

{

Process.Start(new ProcessStartInfo(filePath) { UseShellExecute = true });

}

catch (Exception ex)

{

MessageBox.Show($"Не удалось открыть файл: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private static void SetDocumentStyles(Word.Document doc)

{

doc.Content.Font.Name = "Times New Roman";

doc.Content.Font.Size = 14;

doc.Content.ParagraphFormat.LineSpacing = 18f;

doc.Content.ParagraphFormat.SpaceBefore = 0;

doc.Content.ParagraphFormat.SpaceAfter = 0;

doc.Content.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

}

private static void AddTitle(Word.Document doc, string text)

{

Word.Paragraph title = doc.Paragraphs.Add();

title.Range.Text = text;

title.Range.Font.Bold = 1;

title.Range.Font.Size = 16;

title.Format.SpaceBefore = 0;

title.Format.SpaceAfter = 0;

title.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

title.Range.InsertParagraphAfter();

}

private static void AddPageBreak(Word.Document doc)

{

Word.Paragraph lastParagraph = doc.Paragraphs.Add();

lastParagraph.Range.InsertBreak(Word.WdBreakType.wdPageBreak);

}

private static void ExportParticipantsToWord(Word.Document doc, List<Participant> participants)

{

AddTableTitle(doc, "Участники");

Word.Table table = CreateWordTable(doc, new string[] { "ID", "ФИО", "Дата рождения", "Контактная информация", "Дистанция", "Пользователь" });

foreach (var item in participants)

{

AddRowToWordTable(table, new string[] {

item.Participant\_Id.ToString(),

item.Full\_Name,

item.Birth\_Date.ToShortDateString(),

item.Contact\_Info,

item.Distance,

item.User?.Full\_Name ?? "Неизвестно"

});

}

FinalizeWordTable(table);

}

private static void ExportCompetitionsToWord(Word.Document doc, List<Competition> competitions)

{

AddTableTitle(doc, "Соревнования");

Word.Table table = CreateWordTable(doc, new string[] { "ID", "Название", "Дата", "Организатор" });

foreach (var item in competitions)

{

AddRowToWordTable(table, new string[] {

item.Competition\_Id.ToString(),

item.Title,

item.Date.ToShortDateString(),

item.User?.Full\_Name ?? "Неизвестно"

});

}

FinalizeWordTable(table);

}

private static void ExportUsersToWord(Word.Document doc, List<User> users)

{

AddTableTitle(doc, "Пользователи");

Word.Table table = CreateWordTable(doc, new string[] { "ID", "ФИО", "Логин", "Роль", "Телефон" });

foreach (var item in users)

{

AddRowToWordTable(table, new string[] {

item.User\_Id.ToString(),

item.Full\_Name,

item.Login,

item.Role?.Name ?? "Неизвестно",

item.Phone

});

}

FinalizeWordTable(table);

}

private static void ExportParticipantLogsToWord(Word.Document doc, List<ParticipantLog> logs)

{

AddTableTitle(doc, "Логи участников");

Word.Table table = CreateWordTable(doc, new string[] { "ID", "Участник", "Соревнование", "Дата" });

foreach (var item in logs)

{

AddRowToWordTable(table, new string[] {

item.Log\_Id.ToString(),

item.Participant?.Full\_Name ?? "Неизвестно",

item.Competition?.Title ?? "Неизвестно",

item.Log\_Date.ToShortDateString()

});

}

FinalizeWordTable(table);

}

private static void ExportRolesToWord(Word.Document doc, List<Role> roles)

{

AddTableTitle(doc, "Роли");

Word.Table table = CreateWordTable(doc, new string[] { "ID", "Название" });

foreach (var item in roles)

{

AddRowToWordTable(table, new string[] {

item.Role\_Id.ToString(),

item.Name

});

}

FinalizeWordTable(table);

}

private static void AddTableTitle(Word.Document doc, string title)

{

Word.Paragraph tableTitle = doc.Paragraphs.Add();

tableTitle.Range.Text = title;

tableTitle.Range.Font.Bold = 1;

tableTitle.Range.Font.Size = 14;

tableTitle.Format.SpaceBefore = 0;

tableTitle.Format.SpaceAfter = 0;

tableTitle.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

tableTitle.Range.InsertParagraphAfter();

}

private static Word.Table CreateWordTable(Word.Document doc, string[] headers)

{

Word.Table table = doc.Tables.Add(doc.Range(doc.Content.End - 1), 1, headers.Length);

for (int i = 0; i < headers.Length; i++)

{

table.Cell(1, i + 1).Range.Text = headers[i];

table.Cell(1, i + 1).Range.Font.Bold = 1;

table.Cell(1, i + 1).Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

}

return table;

}

private static void AddRowToWordTable(Word.Table table, string[] values)

{

table.Rows.Add();

int rowIndex = table.Rows.Count;

for (int i = 0; i < values.Length; i++)

{

table.Cell(rowIndex, i + 1).Range.Text = values[i] ?? "";

table.Cell(rowIndex, i + 1).Range.Font.Bold = 0;

table.Cell(rowIndex, i + 1).Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

}

}

private static void FinalizeWordTable(Word.Table table)

{

table.Columns.AutoFit();

table.Borders.Enable = 1;

foreach (Word.Row row in table.Rows)

{

foreach (Word.Cell cell in row.Cells)

{

cell.VerticalAlignment = Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

}

}

}

private static void ReleaseWordObjects(params object[] objects)

{

foreach (var obj in objects)

{

try

{

System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);

}

catch { }

finally

{

GC.Collect();

}

}

}

}

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта была разработана информационная система «Run 2Gether», направленная на автоматизацию процессов планирования и организации соревнований по бегу. Как отмечено во введении, бег остается одним из самых популярных и доступных видов физической активности, однако организация таких мероприятий часто сопряжена с рутинными операциями, временными затратами и возможными ошибками. Разработанная система призвана решить эти проблемы, обеспечив удобство, скорость и надежность взаимодействия между организаторами, тренерами и участниками.

Основные результаты работы соответствуют поставленным во введении целям и задачам:

**1.** Автоматизация процессов: Система упростила регистрацию участников, учет документов и управление соревнованиями, что значительно сократило временные затраты и минимизировало ошибки.

**2.** Разграничение прав доступа: Была реализована система авторизации, обеспечивающая различные уровни доступа для организаторов, тренеров и участников, что повысило безопасность и удобство использования.

**3.** Использование современных технологий: Выбор платформы .NET, языка C#, СУБД SQL Server и других инструментов, как обосновано в первой главе, позволил создать производительное и масштабируемое решение.

**4.** Практическая значимость: Система «Run 2Gether» готова к внедрению в реальных условиях, что подтверждает её актуальность и востребованность в сфере организации спортивных мероприятий.

Таким образом, проект успешно достиг своей главной цели — автоматизации процессов организации беговых соревнований. Разработанная система не только отвечает современным требованиям, но и открывает возможности для дальнейшего развития, такого как расширение функционала, интеграция с мобильными приложениями или социальными сетями. Результаты работы демонстрируют, что применение информационных технологий в спортивной сфере способно значительно повысить эффективность и качество организации мероприятий, делая их более доступными и удобными для всех участников.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. run2gather

<https://www.run2gather.com/?lang=en/>

(Дата обращения 27.03.2025).

1. sports

<https://www.sports.ru/health/calendar/running/>

(Дата обращения 30.03.2025).

1. rus-visio-blog-archive.github.io

<https://rus-visio-blog-archive.github.io/2014/04/27/1-2.htm/>

(Дата обращения 30.03.2025).

1. secure.qgiv

<https://secure.qgiv.com/for/5hztlp/event/2024run2getherregistrationcrew/>

Дата обращения 02.04.2025).