МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное программирование»

Тема: Программное средство «Ведение турнирных таблиц»

Исполнитель

студент 2 курса 4 группы Пицуха Ярослав Анатольевич

(Ф.И.О.)

Руководитель работы Панченко О.Л.

(подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2019

**Содержание**

|  |  |
| --- | --- |
| Введение | 6 |
| 1. Аналитический обзор литературы | 7 |
| * 1. Challonge! | 7 |
| * 1. Спортивные таблицы 2.0 | 7 |
| * 1. Tourney Master 3 | 8 |
| * 1. Аналогичность и различие програмных средств | 9 |
| 1. Анализ требований к программному средству и разработка функциональных требований | 9 |
| * 1. Страница для создания турниров | 10 |
| * 1. Главная страница со списком всех созданных турнирах | 10 |
| 1. Использованные технологии и их роль в проекте | 11 |
| * 1. Entity Framework | 11 |
| * 1. Windows Presentation Foundation | 11 |
| * 1. Microsoft SQL Server | 12 |
| * 1. Microsoft Visual Studio | 12 |
| 1. Проектирование и разработка | 13 |
| * 1. Разработка базы данных | 13 |
| * 1. Диаграммы UML, взаимосвязь всех компонентов, диаграмма использования, блок-схема действий пользователя в приложении | 15 |
| 1. Создание (реализация) программного средства | 18 |
| * 1. Архитектура проектируемого программного средства | 18 |
| * + 1. Model | 19 |
| * + 1. View | 19 |
| * + 1. ViewModel | 20 |
| * 1. Структура проекта и описание его папок | 20 |
| 1. Тестирование программы | 23 |
| 1. Методика использования программного средства | 25 |
| * 1. Окно создания турнира | 25 |
| * 1. Окно для работы с готовыми турнирами | 26 |
| Заключение | 28 |
| Список использованной литературы | 29 |
| Приложение A. Логическая схема базы данных | 30 |
| Приложение B. Диаграмма классов UML | 31 |

**Введение**

**В современном мире всё чаще и чаще многие прикладные занятия оцифровываются и упрощаются с помощью программного обеспечения. Такие же процессы проходят и в соревновательной сфере. Раньше организаторам турниров и соревнований приходилось вести массивный учёт на листах бумаги, чертить всё от руки, высчитывать и заносить разнотипную информацию. Сейчас же уже такой нужды нет, организатор может иметь у себя на компьютере программное обеспечение в котором сможет выбирать удобный для себя формат проведения соревнования, вести учёт и подводить итоге. Всё это ускоряет и автоматизирует процесс администрирования.**

Цель курсового проекта направлена на создание программного средства, с помощью которого можно создавать, вести и анализировать различные турниры.

Задачами курсового проекта являются:

* разработка понятного и легко воспринимаемого интерфейса для пользователей всех возрастов;
* разработка программного средства для ведения турниров;
* разработка функции, связанной с созданием турниров, генерирования сетки, занесением результатов, подведением итогов.

# **Аналитический обзор литературы**

**При анализе прототипов были обнаружены несколько похожих по свойствам программ. Результат анализа приведен ниже.**

## **Challonge!**

Веб-сайт позволяющий онлайн работать с многофункциональными базами турниров

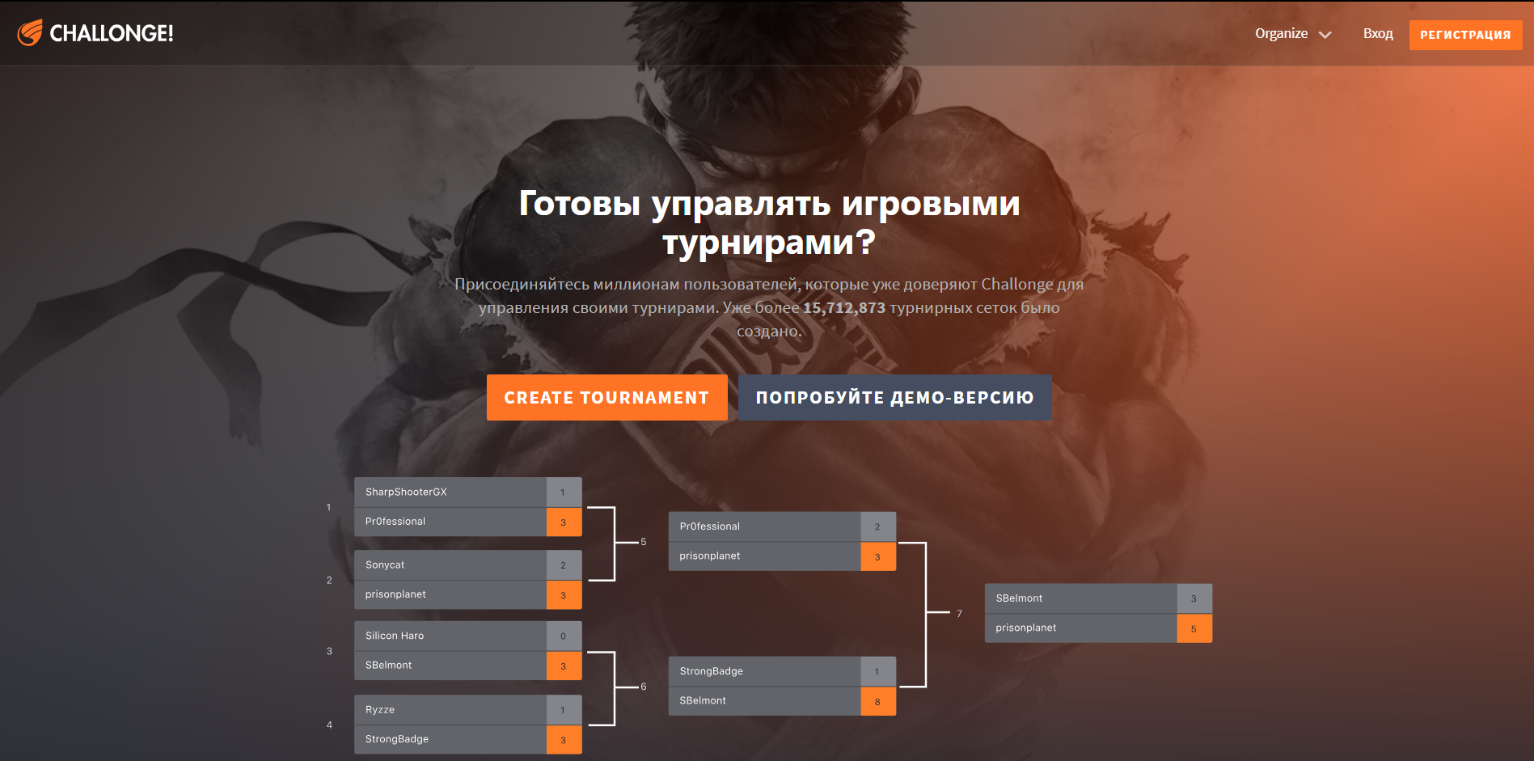


Рисунок 1.1 – Часть окна программы Challonge!

## **Спортивные таблицы 2.0**

Ещё более похожий проект, десктоп приложение. Отличие от моего проекта: формат создаваемых турниров, здесь он только круговой. Часть данной программы представлена на рисунке 1.2.

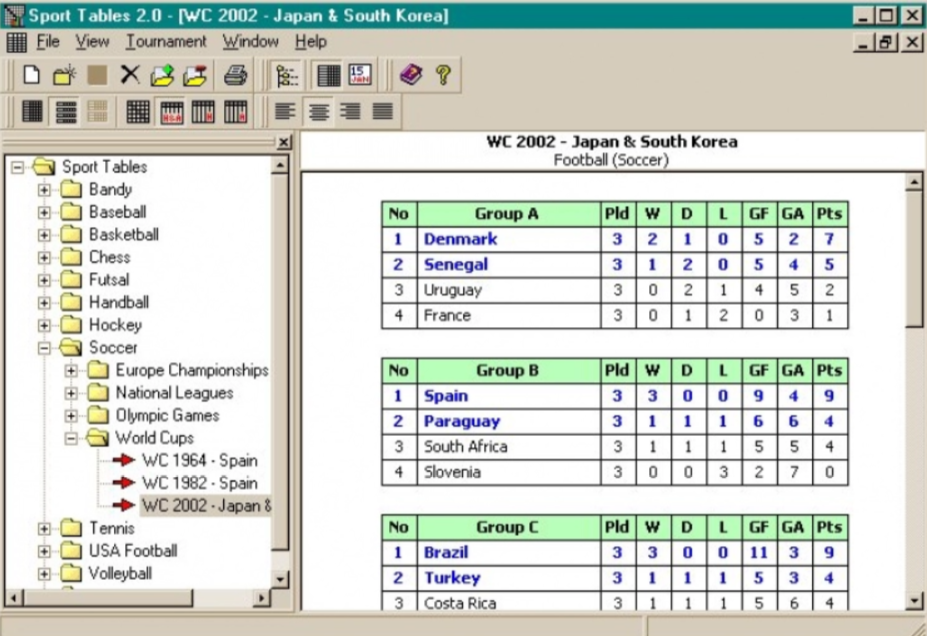


Рисунок 1.2 – Часть программы Sport Tables 2.0

## **Tourney Master 3**

Более обширная и наглядная программа чем та, что была в прошлом примере и чем мой проект. Часть программы представлено на рисунке 1.3

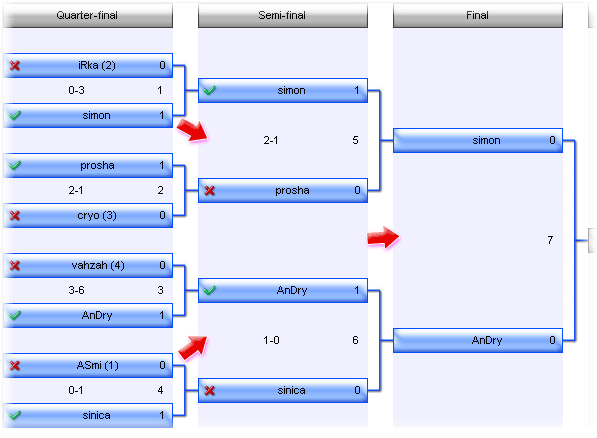


Рисунок 1.3 – Часть Интерфейса Tourney Master

## **Аналогичность и различие программных средств**

Аналогичность програмных средств заключается в том, что все они предназначены для создания и администрирования турниров

Главным отличием является то, что они используют разную платформу, разное наглядное представление таблиц и разные форматы представления этих таблиц.

# **Анализ требований к программному средству и разработка функциональных требований.**

При разработке курсового проекта были определены тебования к программному средству. Програмное средство должно иметь страницу для создания новых турниров, главную страницу со списком всех уже созданных турнирах и страницы для работы с каждым турниром по отдельности.

## **Страница для создания турниров**

Страница для создания турниров позволяет выбирать необходимые характеристики для будущего турнира и заносить необходимые входные данные. На странице необходимо:

* указать название
* выбрать формат проведения турнира
* выбрать необходимое количество участников
* занести данные о всех участниках соревнования

## **Главная страница со списком все созданных турнирах**

Главная страница выглядит как колонка со списком, всех имеющихся турниров, на главном окне работы с турнирами. При нажатии на нужный турнир, в оставшейся части окна откроется страница для работы с выбранным турниром. В любой момент можно перейти к работе с другим турниром, а затем вернутся обратно, состояния на которых была закончена работа сохранятся.

## **Страницы для работы с каждым турниром по отдельности**

Эта страница предназначена для работы с выбранным турниром, здесь отображается краткая информация о турнире, турнирная сетка и кнопки для работы с турниром.

# **Использованные технологии и их роль в проекте**

При разработке приложения были использованы следующие технологии:

* Entity Framework;
* Microsoft SQL Server;
* WPF (Windows Presentation Foundation);
* интегрированная среда разработки Microsoft Visual Studio 2017.

## **Entity Framework**

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами. Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами [5]

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

Windows Presentation Foundation (произносится — «даб-пи-эф») — система для построения клиентских приложений [Windows](https://ru.wikipedia.org/wiki/Windows) с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) (начиная с версии [3.0](https://ru.wikipedia.org/wiki/.NET_Framework_3.0)), использующая язык XAML. **WPF)** представляет собой обширный API-интерфейс для создания настольных графических программ имеющих насыщенный дизайн и интерактивность. В отличие от устаревшей технологии Windows Forms, WPF включает новую модель построения пользовательских приложений (в основе WPF лежит мощная инфраструктура, основанная на DirectX).

Это означает возможность применения развитых графических эффектов, н  платя за это производительностью, как это было в Windows Forms. Фактически даже становятся доступными такие расширенные средства, как поддержка видеофайлов и трехмерное содержимое. Используя эти средства (при наличии хорошего инструмента графического дизайна), можно создавать бросающиеся в глаза пользовательские интерфейсы и визуальные эффекты, которые были просто невозможны в Windows Forms [6].

## **Microsoft SQL Server**

Microsoft SQL Server — система управления реляционными базами данных, разработанная корпорацией Microsoft. Использует язык Transact-SQL, который является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. MS SQL Server – это платформа для решения критически важных задач в масштабе предприятия, обладающая высокой доступностью, повышенной производительностью и безопасностью. Решение представляет собой хорошо масштабируемый, полностью реляционный, быстродействующий сервер, способный обрабатывать большие объемы данных для клиент-серверных приложений. Рекордная производительность MS SQL Server обеспечивается новыми технологиями работы с памятью, что помогает предприятиям ускорить свой бизнес и реализовать новые сценарии работы. Кроме того, SQL Server позволяет использовать новые гибридные облачные решения и пользоваться новыми преимуществами облачных вычислений. Расширенные функции безопасности, в сочетании со встроенными, удобными для использования инструментами и управляемым доступом к данным, позволяют организации выполнить требования строгих политик соответствия нормам [7].

## **Microsoft Visual Studio 2017**

Microsoft Visual Studio — линейка продуктов компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), включающих [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), так и приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в том числе с поддержкой технологии [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), а также [веб-сайты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82), [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), [веб-службы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) как в [родном](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), так и в [управляемом](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) кодах для всех платформ, поддерживаемых [Windows](https://ru.wikipedia.org/wiki/Windows), [Windows Mobile](https://ru.wikipedia.org/wiki/Windows_Mobile), [Windows CE](https://ru.wikipedia.org/wiki/Windows_CE), [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), [Xbox](https://ru.wikipedia.org/wiki/Xbox), [Windows Phone](https://ru.wikipedia.org/wiki/Windows_Phone) [.NET Compact Framework](https://ru.wikipedia.org/wiki/.NET_Compact_Framework) и [Silverlight](https://ru.wikipedia.org/wiki/Silverlight).

Visual Studio включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) [8].

# **Проектирование и разработка**

## **Разработка базы данных**

При создании курсового проекта обязательным условием было наличие и работа базы данных. Структура всех таблиц базы данных была прописана в исходном коде программы по принципу CodeFirst

Таким образом была разработана база данных tournamentdb, которая включает в себя три таблицы Tournament, Team, Game. Структура данной базы данных представлена на рисунке 4.1-1,2,3.

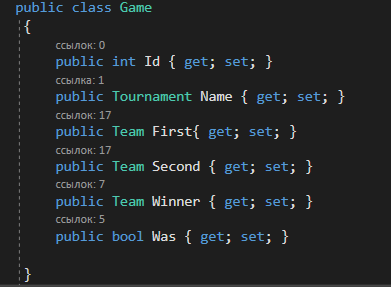
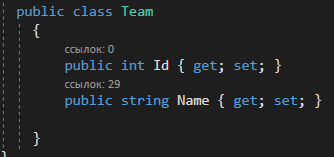
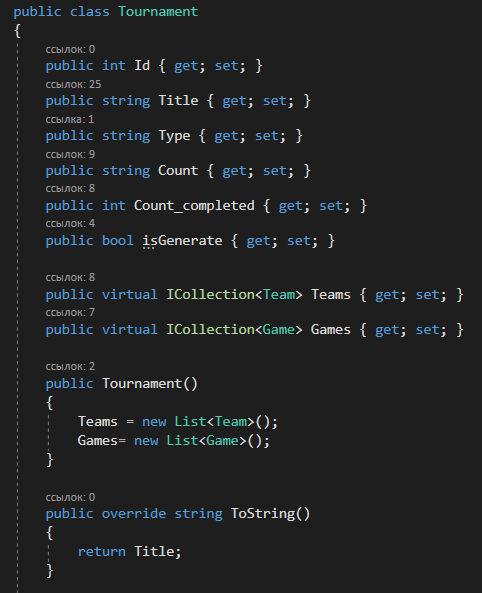


Рисунок 4.1‑1,2,3 – Структура данной базы данных

Таблица Tournament предназначена для сохранения, изменения и хранения информации о турнирах. Данная таблица содержит 6 столбцов:

* ID – ключевое поле таблицы, содержащее уникальные значения для каждого турнира;
* Title – название турнира;
* Type – формат турнира;
* Count – количество команд участвующих в турнире;
* Count\_completed – количество сыгранных матчей;
* IsGenerate – показывает сгенерирована ли у турнира сетка.

Таблица Team предназначена для сохранения, изменения и хранения информации о командах турниров. Данная таблица содержит 3 столбца:

* ID – ключевое поле таблицы, содержащее уникальные значения для каждой команды;
* Name – название участника;
* Tournament\_id – ссылается на турнир-владелец

Таблица Game предназначена для сохранения, изменения и хранения информации об играх в турнирах. Данная таблица содержит 6 столбцов:

* ID – ключевое поле таблицы, содержащее уникальные значения для каждой игры;
* Was – показывает прошла ли уже игра;
* First\_id – ссылается на первую команду матча
* Second\_id – ссылается на вторую команду матча
* Winner\_id – ссылается на команду победительницу
* Name\_id – ссылается на турнир-владелец

## **Диаграммы UML, взаимосвязь всех компонентов, диаграмма использования, блок-схема действий пользователя в приложении**

Диаграмма UML – это графическое представление набора элементов, изображаемое в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграмма для данного проекта представлена в приложении 2.

Блок-схема алгоритма, которая для данной курсовой работы представлена на рисунке 4.2-1 — графическое изображение алгоритма в виде связанных между собой с помощью стрелок (линий перехода), каждый из которых соответствует одному шагу алгоритма. Внутри блока дается описание соответствующего действия.

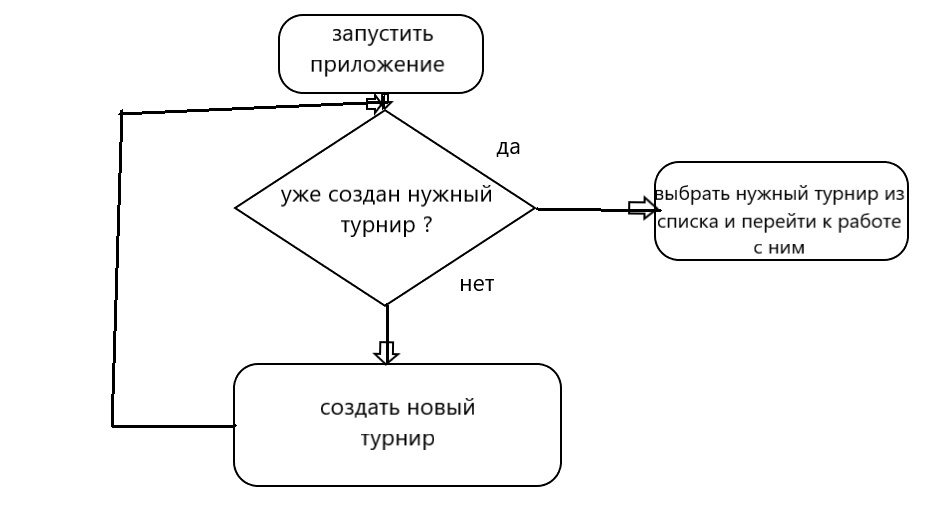


Рисунок 4.2‑1 – Блок-схема игры

Диаграмма вариантов использования, которая для данной курсовой работы представлена на рисунке 4.2-2 — диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

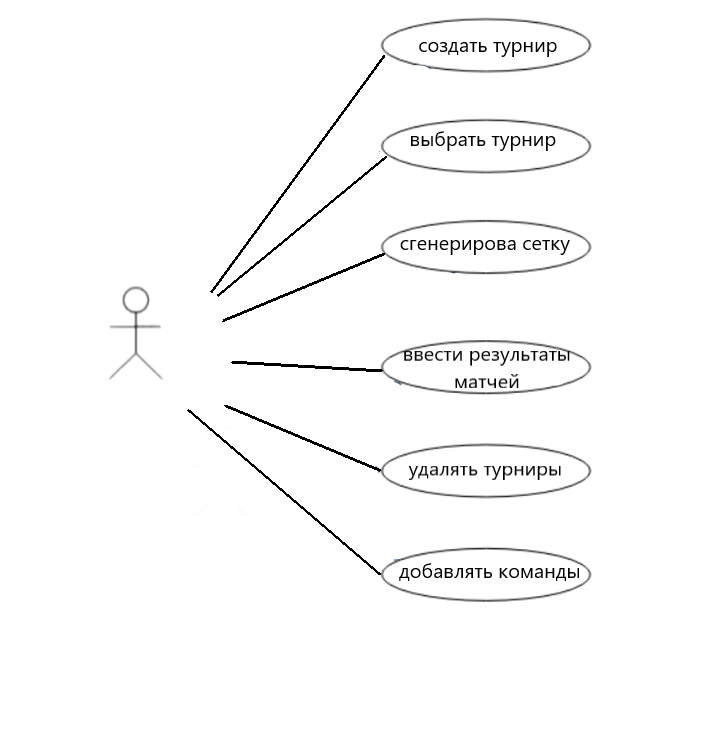


Рисунок 4.2‑2 – Диаграмма вариантов использования по

# **Создание (реализация) программного средства**

## **Архитектура проектируемого программного средства**

При создании программного средства был использован паттерн MVVM (Model-View-ViewModel). Модель данного паттерна представлен на рисунке 5.1. Паттерн MVVM (Model-View-ViewModel) позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения.

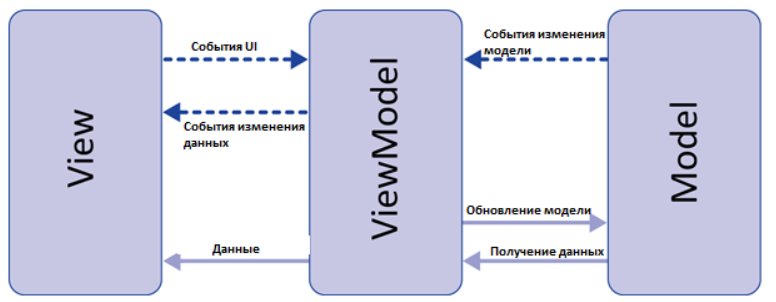


Рисунок 5.1 – Модель паттерна MVVM

Данный паттерн был представлен Джоном Госсманом (John Gossman) в 2005 году как модификация шаблона Presentation Model и был первоначально нацелен на разработку приложений в WPF. И хотя сейчас данный паттерн вышел за пределы WPF и применяется в самых различных технологиях, в том числе при разработке под Android, iOS, тем не менее WPF является довольно показательной технологией, которая раскрывает возможности данного паттерна. MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления.

Признаки View-модели:

* Двухсторонняя коммуникация с представлением;
* View-модель — это абстракция представления. Обычно означает, что свойства представления совпадают со свойствами View-модели / модели
* View-модель не имеет ссылки на интерфейс представления (IView). Изменение состояния View-модели автоматически изменяет представление и наоборот, поскольку используется механизм связывания данных (Bindings)
* Один экземпляр View-модели связан с одним отображением.

Преимущества использования шаблона MVVM заключаются в следующем:

* Если существует существующая реализация модели, которая инкапсулирует существующую бизнес-логику, изменить ее может быть сложно или рискованно. В этом сценарии модель представления выступает в качестве адаптера для классов модели и позволяет избежать каких-либо серьезных изменений в коде модели.
* Разработчики могут создавать модульные тесты для модели представления и модели, не используя представление. Модульные тесты для модели представления могут выполнять те же функции, что и представление.
* Пользовательский интерфейс приложения можно изменить, не касаясь кода, при условии, что представление полностью реализовано в XAML. Следовательно, новая версия представления должна работать с существующей моделью представления.
* Дизайнеры и разработчики могут работать независимо и одновременно над своими компонентами в процессе разработки. Дизайнеры могут сосредоточиться на представлении, а разработчики могут работать над моделью представления и компонентами модели.

Ключ к эффективному использованию MVVM заключается в понимании того, как разделить код приложения на правильные классы, и в понимании того, как классы взаимодействуют [10].

### **Model**

Классы модели - это невизуальные классы, которые инкапсулируют данные приложения. Таким образом, модель можно рассматривать как представление модели предметной области в приложении, которые, как правило, включает в себя модель данных вместе с бизнесом и логикой проверки. Примеры типовых объектов включают в себя передачу данных объектов (DTO), Plain Old CLR объектов (Pocos), и сгенерированный объект, и прокси-объекты.

Классы моделей обычно используются вместе со службами или репозиториями, которые инкапсулируют доступ к данным и кэширование.

### **View**

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление - это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

### **ViewModel**

Модель представления реализует свойства и команды, с которыми представление может связывать данные, и уведомляет представление о любых изменениях состояния посредством событий уведомления об изменении. Свойства и команды, предоставляемые моделью представления, определяют функциональные возможности, предлагаемые пользовательским интерфейсом, но представление определяет, как эти функциональные возможности должны отображаться.

Каждая модель представления предоставляет данные из модели в форме, которую представление может легко использовать. Для этого модель представления иногда выполняет преобразование данных. Размещение этого преобразования данных в модели представления является хорошей идеей, поскольку оно предоставляет свойства, с которыми представление может быть связано. Например, модель представления может объединять значения двух свойств, чтобы облегчить отображение представлением [11].

## **Структура проекта и описание его папок**

Проект курсовой работы содержит несколько папок, в которых хранится вся необходимая информация. Данные папки в раскрытом виде представлена на рисунке 5.2-2.

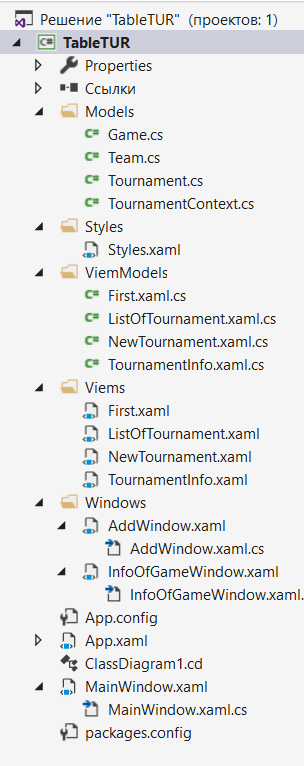


Рисунок 5.2‑2 – Папки проекта в раскрытом виде

Стоит отметить, что все файлы аналогичны по структуре строения. При создание курсового проекта был использован grid.

Grid *–* это наиболее мощный и часто используемый контейнер, напоминающий обычную таблицу. Он содержит столбцы и строки, количество которых задает разработчик.

Папка Model содержит всю архитектуру для таблиц базы данных. Данная папка в раскрытом виде представлена на рисунке 5.2-2.

Папка ViewModel содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели. Данная папка в раскрытом виде представлена на рисунке 5.2-2.

Папка Viem содержит все файлы содержащие xaml разметку. Данная папка в раскрытом виде представлена на рисунке 5.2-2.

Папка Styles содержит код дизайна приложения.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

# **Тестирование программы**

После построения основных моментов в программе курсового проекта началась его реализация, наполнение задачами, составление и добавление турниров, отслеживание наполнения базы данных.

При нарушении правил заполнения информации в турнирах, высвечивается предупреждение об ошибке либо действие блокируется.

Проектируемое программное средство было создано таким образом, что возникновение ошибок возможно только при нарушении заполнения граф текстовых боксов и чек боксов. Существую такие ошибки как не ввод данных при создании турнира, несовпадение заданного и реального количества команд в турнире, неверный ввод имени участника, не выбор победителя в окне результатов матча.

В окне добавления участника также существует извещение об ошибке или некорректности. Например, если пользователь захочет создать билет с именем, которое имеет пробелы, приложение уведомит его о наличии такой ошибки, придётся использовать в место пробела знак подчёркивания. Это продемонстрировано на рисунке 6-1

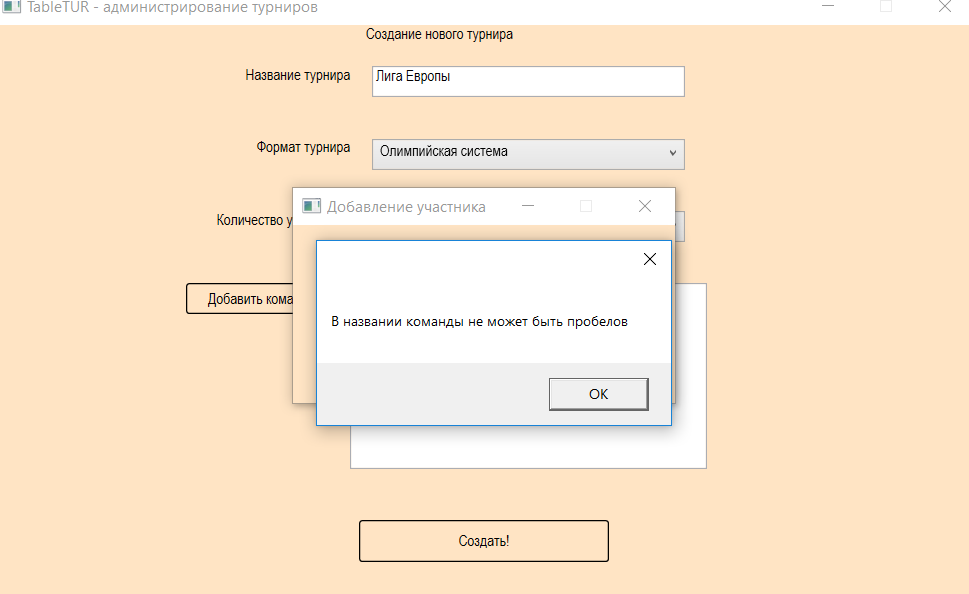


Рисунок 6-1 – Появление ошибки об некорректности данных

Так же оповещение об некорректных данных возникает если не указать победителя в окне матча. Это оповещение об ошибке или о некорректности введенных данных показано на рисунке 6-2.

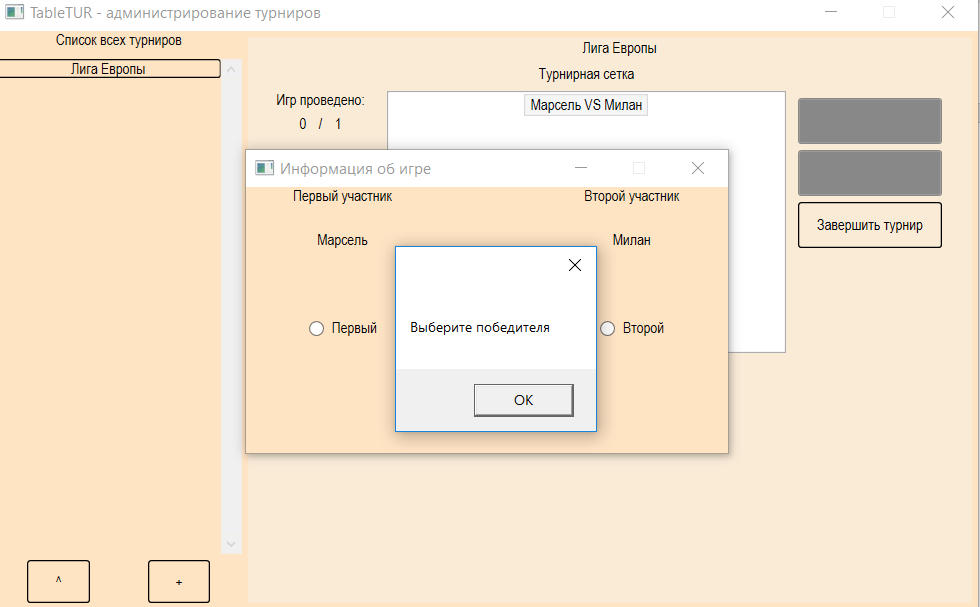


Рисунок 6-2 – Появление ошибки об некорректности данных

# **Методика использования программного средства**

После запуска программы пользователь наблюдает окно главного меню, которое представлено на рисунке 7-1.

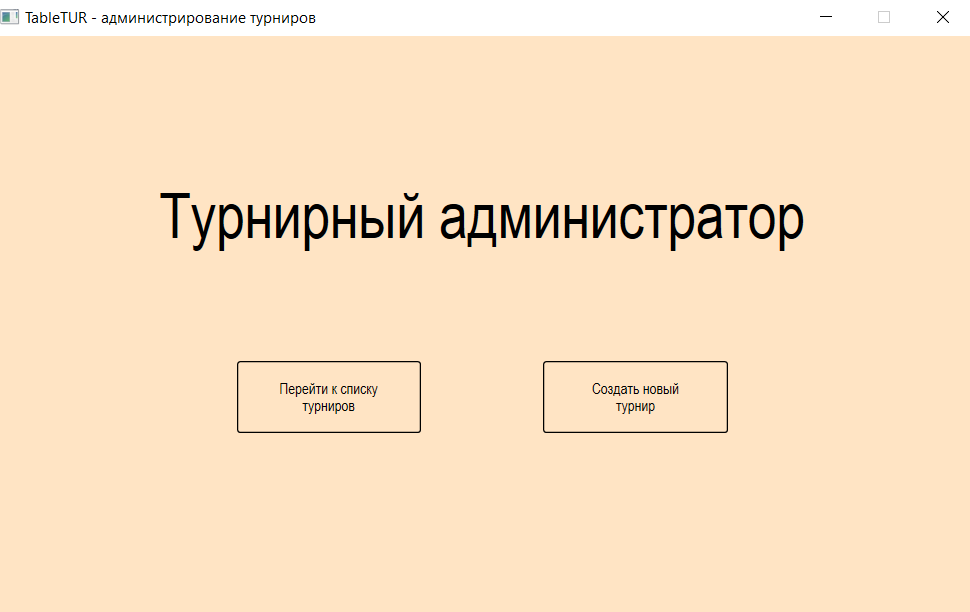


Рисунок 7‑1 – Главное окно программного средства

Далее можно перейти к созданию нового турнира, либо же перейти по кнопке и выбрать какой-нибудь из ранее созданных турниров.

## **Окно создания турнира**

После нажатия кнопки “Создать новый турнир” появится новое окно, в котором будут представлены необходимые для создания турнира поля, представленных на рисунке 7.1.

Каждое поле отвечает за конкретное свойство нового турнира.

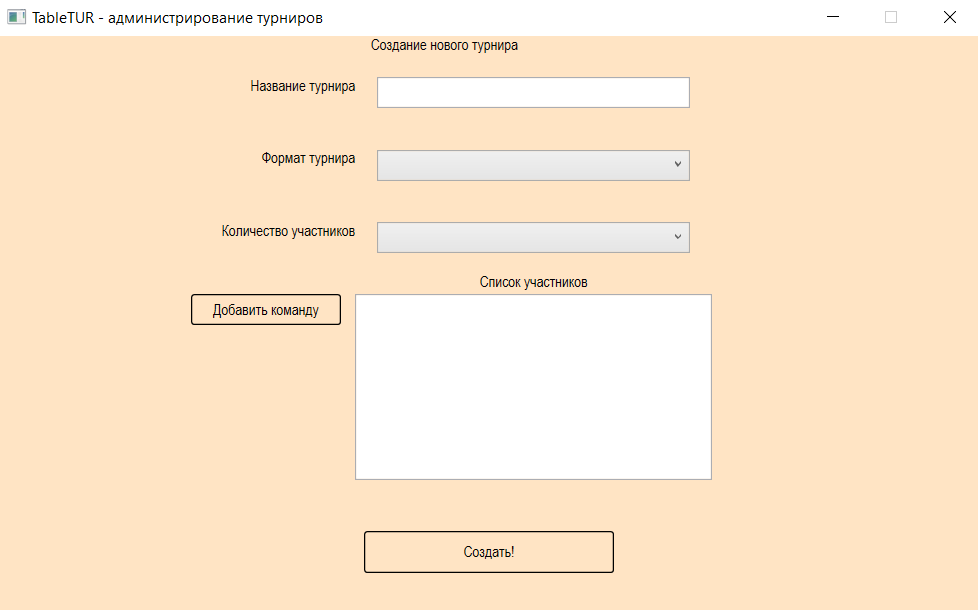


Рисунок 7.1 – Окно создания турнира

## **Окно для работы с готовыми турнирами**

Интерфейс окна работы с турнирами в какой-то момент времени выглядит так, как представлен на рисунке 7.2

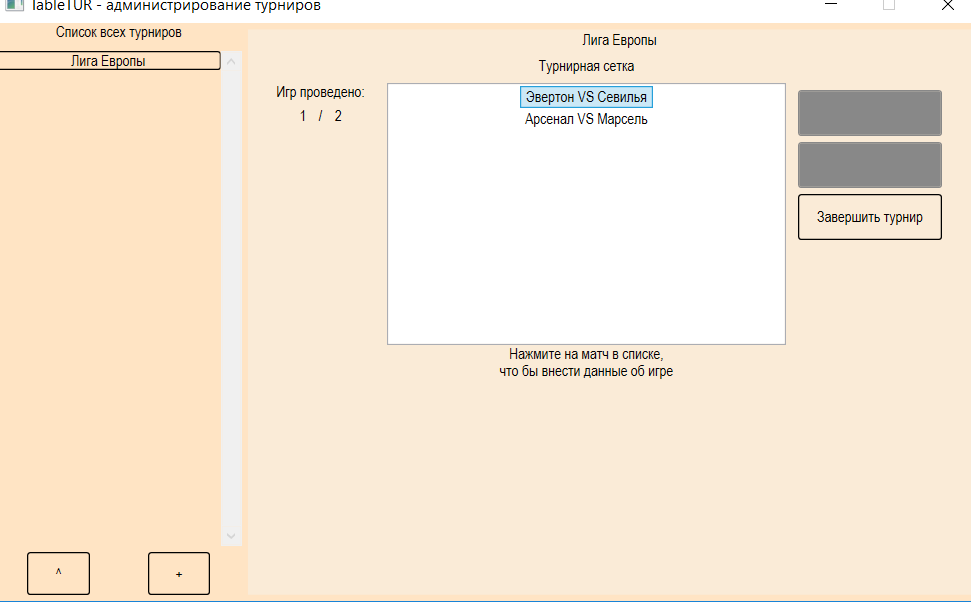


Рисунок 7.2 – Интерфейс окна со списком турниров

**Заключение**

В результате выполнения курсового проекта была разработано программное обеспечение, которое позволяет пользователю работать со своими турнирами, добавлять, удалять новые турниры, работать с базой данных, вносить данные, генерировать турнирную сетку. Таким образом, Программное средство удовлетворяет поставленным ранее задачам.

В процессе выполнения курсовой работы были закреплены знания в программировании с использованием языка программирования C# и технологии WPF, в использование Entity Framework, проектирования баз данных, был получен опыт прохождения всех этапов создания приложения.

**Список литературы**

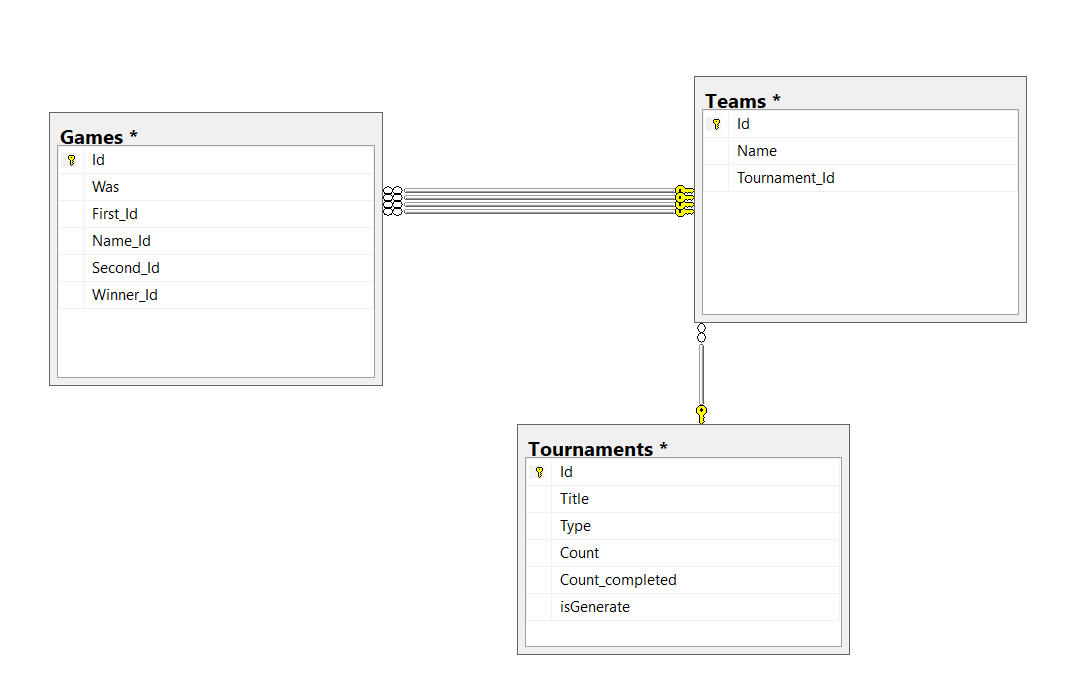
1. Аналоги и прототипы:  [виртуальная среда обучения](https://en.wikipedia.org/wiki/Virtual_learning_environment) WebCT [Электронный ресурс] – <https://en.wikipedia.org/wiki/WebCT> Дата доступа : 01.05.2019
2. Аналоги и прототипы: Система дистанционного обучения Learning Space 5.0 [Электронный ресурс] – <http://dl.nw.ru/software/LearningSpace5/index.shtml> Дата доступа :02.05.2019
3. Аналоги и прототипы: СДО «Прометей» [Электронный ресурс] – <https://www.prometeus.ru/actual/01_products/lms/opisanie.html> Дата доступа: 01.05.2019
4. Аналоги и прототипы: игра-головоломка [Code Hunt](https://www.codehunt.com/) по обучению программированию [Электронный ресурс] – <https://habr.com/ru/post/223173/> Дата доступа : 01.05.2019
5. Работа с Entity Framework [Электронный ресурс] – <https://professorweb.ru/my/entity-framework/6/level1/> Дата доступа 03.05.2019
6. Использование WPF [Электронный ресурс]– https://metanit.com/sharp/wpf/22.1.php Дата доступа: 09.05.2019

# Microsoft SQL Server [Электронный ресурс] – [https://ru.wikipedia.org/wiki/Microsoft\_SQL\_Server Дата доступа 11.03.2019](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server%20Дата%20доступа%2011.03.2019)

1. Microsoft Visual\_Studio [Электронный ресурс] – [https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio Дата доступа 02.05.2019](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio%20Дата%20доступа%2002.05.2019)
2. Связь многие-ко-многим [Электронный ресурс] – <https://habr.com/ru/post/193380/> Дата доступа : 28.04.2019
3. Пацей Н.В. Курс лекций по языку программирования C# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
4. Теория по MVVM [Электронный ресурс] – <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>. Дата доступа: 29.04.2019

# Пацей Н.В. Презентации к лекциям.

**Приложение A. Логическая схема базы данных**

****