МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТ:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Футбольный менеджер Goalscore»

Исполнитель

студент (ка) 2 курса группы 5 Кришталь Максим Олегович

(Ф.И.О.)

Руководитель работы преп.-стажер Север А.С.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В

(подпись)

Минск 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированные технологии программирования и стандарты проектирования"

|  |  |
| --- | --- |
| Специальность: 1-40 01 01Программное обеспечение информационных технологий  Студент: Кришталь Максим Олегович | Группа:\_\_5\_\_ |
| **Тема: Программное средство «Футбольный менеджер Goalscore»** | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2023 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора:
  + Поддерживать работу c базой данных;
  + Добавление завершенных матчей;
  + Добавление состава команды;
  + Составление расписания матчей.
* Функции клиента:
  + Выполнять регистрацию и авторизацию;
  + Редактирование профиля;
  + Поиск матчей;
  + Просмотр матчей;
  + Добавление в избранное;
  + Составление уникальной команды.

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. Использовать архитектурные и шаблоны проектирования.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Разработка функциональной модели и модели данных ПС (выполняемые функции)
* Руководство пользователя
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

***Календарный план***

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2023 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2023 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2023 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 02.04.2023 |  |
| 5 | Кодирование программного средства | 23.04.2023 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2023 |  |
| 7 | Оформление пояснительной записки | 07.05.2023 |  |
| 8 | Защита проекта | 20.05.2023 |  |

**5. Дата выдачи задания** 12.02.2023

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Север А.С.*

(подпись)

Задание принял к исполнению 12.02.2023\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Кришталь М.О.*

(дата и подпись студента)

Содержание

[ВВЕДЕНИЕ 6](#_Toc135340667)

[1 Аналитический обзор литературы и формирование требований 7](#_Toc135340668)

[1.1 Анализ прототипов 7](#_Toc135340669)

[1.1.1 Flashscore 7](#_Toc135340670)

[1.1.2 Belarus football 7](#_Toc135340671)

[1.2 Требования к проекту 8](#_Toc135340672)

[2 Анализ требований к программному средству и разработка функциональных требований 10](#_Toc135340673)

[2.1 Описание средств разработки 10](#_Toc135340674)

[2.2 Технологии и инструменты разработки 10](#_Toc135340675)

[2.3 Спецификация функциональных требований к программному средству 11](#_Toc135340676)

[2.4 Спецификация функциональных требований 12](#_Toc135340677)

[3 Проектирование программного средства 14](#_Toc135340678)

[3.1 Общая структура 14](#_Toc135340679)

[3.2 Разработка общей схемы приложения 22](#_Toc135340680)

[3.3 Взаимоотношение между классами 25](#_Toc135340681)

[3.4 Модель базы данных 25](#_Toc135340682)

[3.5 Проектирование архитектуры приложения 34](#_Toc135340683)

[3.6 Проектирование последовательностей проекта 34](#_Toc135340684)

[4 Реализация программного средства 36](#_Toc135340685)

[4.1 Основные классы программного средства 36](#_Toc135340686)

[4.2 Выполнение входа 36](#_Toc135340687)

[4.3 Регистрация 36](#_Toc135340688)

[4.4 Просмотр матчей 37](#_Toc135340689)

[4.5 Добавление игроков в избранное 37](#_Toc135340690)

[4.6 Составление уникальной пользовательской команды 38](#_Toc135340691)

[4.7 Функционирование панели администратора 38](#_Toc135340692)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 39](#_Toc135340693)

[5.1 Тестирование авторизации и регистрации 39](#_Toc135340694)

[5.2 Тестирование составления уникальной команды 40](#_Toc135340695)

[6 Руководство по использованию 41](#_Toc135340696)

[ЗАКЛЮЧЕНИЕ 45](#_Toc135340697)

[Список использованных источников 46](#_Toc135340698)

[ПРИЛОЖЕНИЕ А 47](#_Toc135340699)

[ПРИЛОЖЕНИЕ Б 48](#_Toc135340700)

[ПРИЛОЖЕНИЕ В 49](#_Toc135340701)

[ПРИЛОЖЕНИЕ Г 50](#_Toc135340702)

# ВВЕДЕНИЕ

Футбол — командный вид спорта, в котором целью является забить мяч в ворота соперника ногами или другими частями тела (кроме рук) большее количество раз, чем команда соперника.

В настоящее время самый популярный и массовый вид спорта в мире. В связи с этим требуются приложения для просмотра и анализа футбольной статистки.

Удобнее всего это делать через специальные приложения, которые показывают футбольную статистику в удобном в виде с возможностью поиска. С помощью разрабатываемого программного средства можно удобно просматривать футбольную статистику, а также составить уникальную команду для анализа возможной сыгранности игроков.

Программное средство «Футбольный менеджер Goalscore» – это программное средство, где каждый желающий может посмотреть интересующую его футбольную статистику, а также составить свою уникальную команду.

Приложение имеет 2 типа пользователей: администратор и пользователь. Они имеют разные полномочия. Пользователи могут просматривать матчи, искать нужные команды и игроков. Администраторы имеют полные права. Они могут удалять данными о матчах, соревнованиях и так далее.

Также будет реализована авторизация всех пользователей. Это значит, что перед тем, как просматривать футбольную статистику, нужно будет зарегистрироваться. Хранение всех пользователей будет осуществлено в базе данных, и при авторизации или регистрации будет непосредственное обращение к ней. Также вся информация о товаре будет хранится в базе данных.

Для того что быть в курсе настоящих матчей будет осуществлена работа с football-data.org API.

В ходе использования этой программы, пользователи смогут получать всю самую актуальную футбольную статистику, а также добавлять в избранное игроков, для составления уникальной команды.

# Аналитический обзор литературы и формирование требований

## Анализ прототипов

Были проанализированы цели и задачи, поставленные в данном курсовом проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

### Flashscore

Сервис live результатов на Flashscore предлагает футбольные онлайн результаты, итоговые результаты матчей Премьер-Лига Англии, Лига Чемпионов и 1000+ других футбольных соревнований по всему миру. Flashscore предлагает live результаты, итоговые результаты с данными о забитых мячах, желтых и красных карточках, сравнение кэфов и статистику очных встреч. Live Centre сайта Flashscore (для главных футбольных турниров) предлагает детальную статистику (владение мячом, ударов всего/в створ, штрафные, угловые, фолы), составы и текстовые онлайн комментарии. Также на Flashscore есть таблицы дома/в гостях/всего, форма команд и тоталы.

Интерфейс интернет-ресурса представлен на рисунке 1.1.

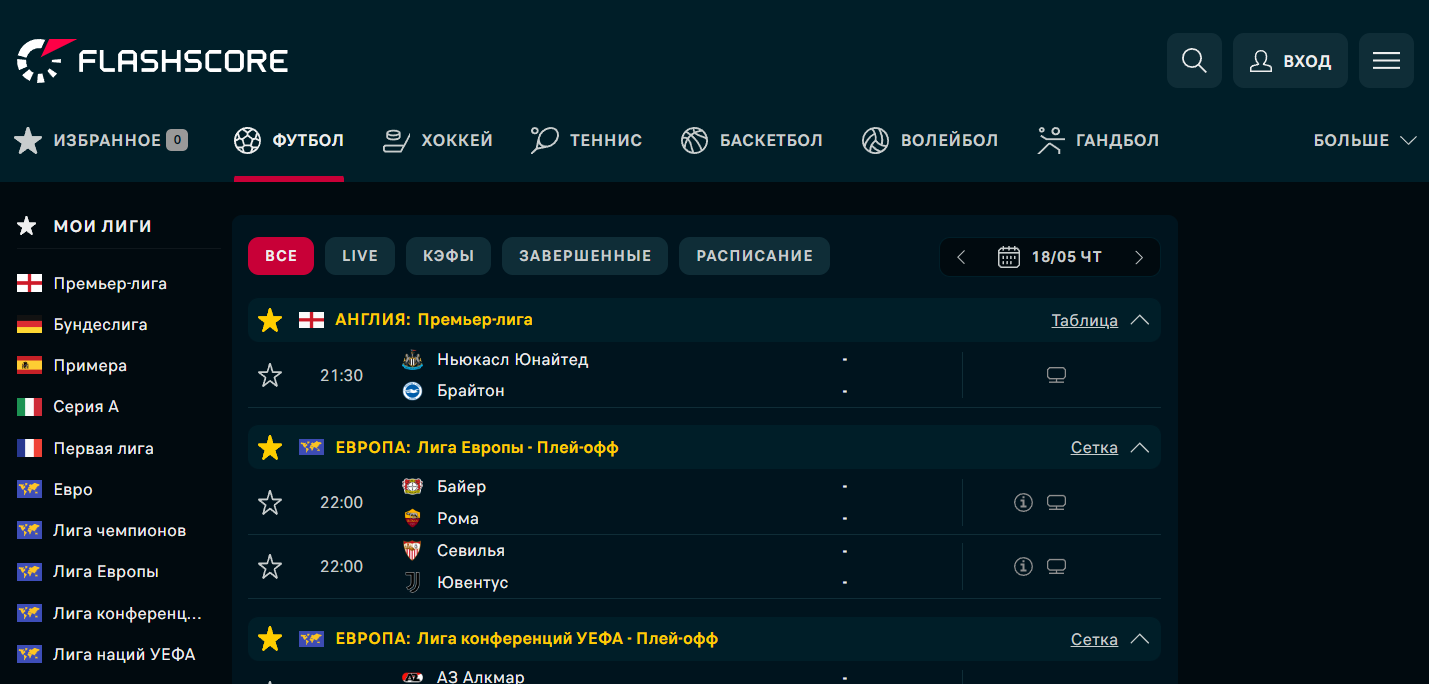


Рисунок 1.1 – Интернет-ресурс «Flashscore»

Интернет-ресурс «Flashscore» представляет удобную и свежую футбольную статистику. Также есть функции добавления в избранное.

### Belarus football

Мобильное приложение «Belarus football» позволяет быть в курсе результатов матчей, составов команд, забитых мячей и другой информации для всех соревнований, под эгидой Ассоциации «Белорусская федерация футбола». Календарь, результаты и статистика всех соревнований, проводимых под эгидой АБФФ.

Можете использовать приложение, чтобы следить за выступлением своих любимых команд, игроков и получать уведомления каждый раз, когда в системе СОМЕТ регистрируется событие матча, связанное с вашими фаворитами, например забивается мяч, игроку выносится предупреждение или звучит финальный свисток арбитра, оповещающий о завершении. Интерфейс мобильного приложения представлен на рисунке 1.2.

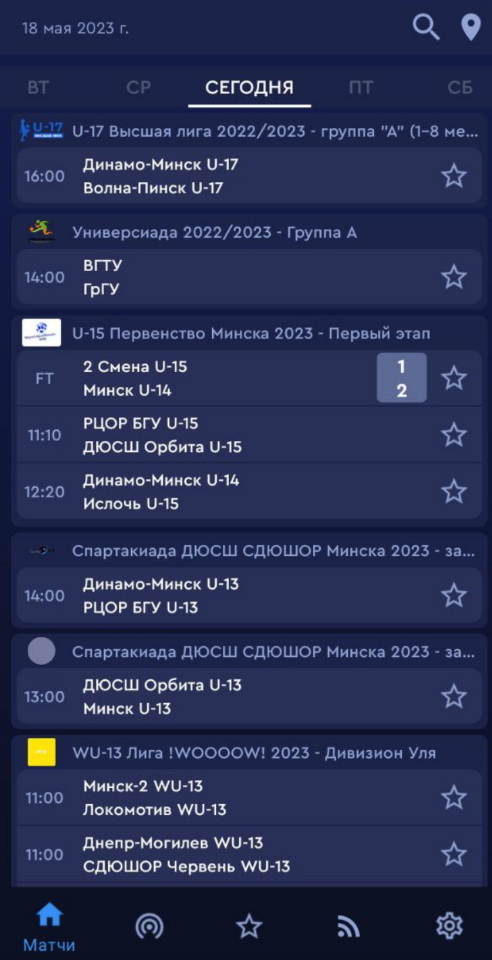


Рисунок 1.2 – Мобильное приложение «Belarus football»

Особенность приложения «Belarus football» в том, что оно мобильное, а также то что тут есть статистика белорусских футбольных соревнований, которые отсутствуют в предыдущем аналоге «Flashscore».

## Требования к проекту

Обзор вышеперечисленных известных аналогов позволяет проанализировать все преимущества и недостатки альтернативных возможностей и позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте. Программное средство должно обеспечивать возможность выполнения перечисленных ниже функций:

* управление администратором базой данных;
* возможность пользователю зарегистрироваться или войти в существующую учетную запись;
* возможность авторизированным пользователям просмотра информации о матчах, игроках;
* возможность просмотра личной страницы пользователя;
* возможность составления уникальной команды;
* возможность просмотра избранных игроков.

Таким образом, можно сказать, что требования к разрабатываемому программному средству основаны на анализе прототипов и аналогов, и представляют собой список функций, необходимых для управления базой данных и обеспечения комфортного пользовательского опыта.

# Анализ требований к программному средству и разработка функциональных требований

Для начала этапа разработки необходимо сформулировать функциональные требования к программному средству.

## Описание средств разработки

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET 6.0;
* язык программирования C#;
* технология WPF;
* расширяемый язык разметки XAML;
* технология Entity Framework Core;
* технология football-data.org API;
* MS SQL Server.

Данный набор средств разработки и технологий был выбран с учетом их широкой функциональности, эффективности и поддержки, чтобы обеспечить разработку качественного и масштабируемого программного средства. Использование интегрированной среды разработки Microsoft Visual Studio 2022, программной платформы .NET 6.0, языка программирования C# и других технологий позволило нам эффективно реализовать требования и достичь желаемого функционального результата.

## Технологии и инструменты разработки

Microsoft Visual Studio 2022 - это программный продукт, который объединяет инструменты для написания, тестирования и сборки кода, а также для публикации разработанных приложений. Он позволяет создавать различные типы приложений, включая консольные и десктопные, с использованием технологий, таких как WinForms или WPF.

Платформа .NET Framework - это технология, разработанная компанией Microsoft в 2002 году, которая обеспечивает возможность создания и выполнения веб-служб и приложений Windows. Она предоставляет среду программирования для локального сохранения и выполнения объектного кода, а также его выполнения в Интернете или удаленно. Основой платформы является общеязыковая среда Common Language Runtime, которая может выполнять код на разных языках программирования.

Для разработки в платформе .NET применяется язык программирования C#, который является основным в данной технологии. Это объектно-ориентированный язык, который характеризуется строгой статической типизацией, поддержкой перегрузки операторов, указателей на функции-члены классов, атрибутов, событий и свойств, а также исключений. Кроме того, C# используется в качестве основного языка разработки приложений на технологии WPF.

Microsoft WPF - это технология, используемая для создания клиентских приложений Windows с интерактивным пользовательским интерфейсом и разделением дизайна и бизнес-логики. Она является альтернативой WinForms и включена в состав .NET Framework. WPF использует язык разметки XAML для описания элементов пользовательского интерфейса и обеспечивает более высокую гибкость и масштабируемость в сравнении с WinForms.

WPF позволяет разрабатывать визуальный интерфейс, используя язык разметки XAML (eXtensible Application Markup Language). Он предоставляет различные элементы управления, возможности привязки данных, макеты, 2D и 3D графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. XAML - это декларативный язык описания интерфейса на основе XML, который позволяет отделить дизайн интерфейса от его реализации и бизнес-логики.

Entity Framework Core является расширяемой технологией, предоставляющей высокий уровень абстракции для взаимодействия с базами данных в приложении WPF. В отличие от традиционных средств ADO.NET, которые требуют создания подключений и объектов для взаимодействия с базами данных, Entity Framework Core позволяет абстрагироваться от деталей реализации базы данных и работать с данными на уровне объектов. Entity Framework Core также является кроссплатформенной и поддерживает .NET Core.

Существуют три возможных способа использования Entity Framework Core в приложении WPF: Database First, Model First и Code First.

При использовании Database First, Entity Framework Core создает классы, отражающие модель базы данных.

При использовании Model First, сначала создается модель базы данных, а затем на ее основе создается реальная база данных на сервере.

При использовании Code First, разработчик создает классы модели данных, которые будут храниться в базе данных, а затем Entity Framework Core генерирует базу данных и ее таблицы на основе этих классов.

Football-data.org API предоставляет футбольные данные и статистику (результаты в реальном времени, расписание матчей, таблицы, составы команд, замены и т.д.) в машиночитаемом виде.

MS SQL Server использует реляционную модель баз данных, где данные хранятся в таблицах, состоящих из строк и столбцов. Каждая строка содержит отдельный объект, а атрибуты объектов хранятся в столбцах. Для взаимодействия с базой данных используется язык SQL (Structured Query Language). Клиент отправляет запрос на языке SQL, который база данных интерпретирует и выполняет, а затем возвращает результат выполнения запроса клиенту. Основной используемый язык запросов в MS SQL Server - Transact-SQL, который основан на SQL с дополнительными расширениями.

## Спецификация функциональных требований к программному средству

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

* регистрация;
* авторизация;
* изменение данных своего профиля;
* просмотр доступных матчей;
* поиск матчей;
* добавление игроков в избранное;
* составление уникальной команды.

Для администратора:

* изменение данных своего профиля;
* добавление завершенных матчей;
* добавление будущих матчей;
* добавление состава команды;
* просмотр данных о игроках;
* просмотр данных о матчах;
* просмотр данных о судьях;
* просмотр данных о заявках на матч;
* просмотр данных о турнирах;
* просмотр данных о сезонах;
* просмотр данных о местах соревнований;
* просмотр данных о тренерах.

Реализация этих требований позволит создать удобное и функциональное программное средство, которое будет полезным для пользователей и обеспечит эффективное управление объявлениями.

По итогу работы над продуктом, его необходимо протестировать и отладить, проанализировать его возможности и, по необходимости, внести дополнения или улучшения проекта, с возможностью дальнейшего расширения проекта.

## Спецификация функциональных требований

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно база данных описано в следующем разделе.

В программном средстве необходимо реализовать регистрацию и авторизацию пользователей для доступа ко всем возможностям приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Для регистрации входными данными являются имя, и пароль. Введенные данные, успешно прошедшие валидацию, заносятся в базу данных.

Пользователь может просматривать матчи, добавлять игроков и избранное, составлять уникальную команду, просматривать состав, расстановку, информацию о выбранном матче, производить поиск матчей. Для каждого пользователя создается его профиль, в котором он может просмотреть информацию о своем профиле. Все сведения пользователе должны автоматически обновляться в базе данных.

Администратор обладает возможностями добавления завершенных и будущих матчей, добавления состава команды, просмотра информации о заявках на матч, матчах, турнирах, местах турниров, сезонов, судьях, тренерах, игроках.

Описание функциональности программного средства представлено на UML-схеме, изображённой на рисунке 2.1.

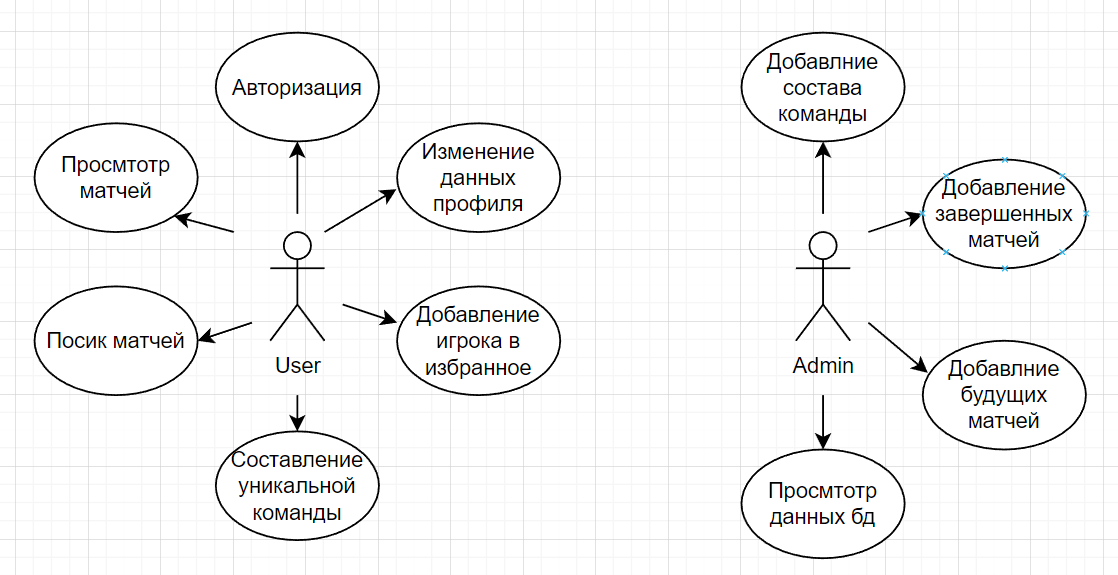


Рисунок 2.1 – UML-диаграмма возможностей

Таким образом, программа должна обеспечивать удобный интерфейс для пользователя и администратора, а также функционал для работы с базой данных, football-data.org API и другими необходимыми технологиями.

# Проектирование программного средства

## Общая структура

Программное средство «Футбольный менеджер Goalscore» имеет следующею структуру, представленную на рисунке 3.1.

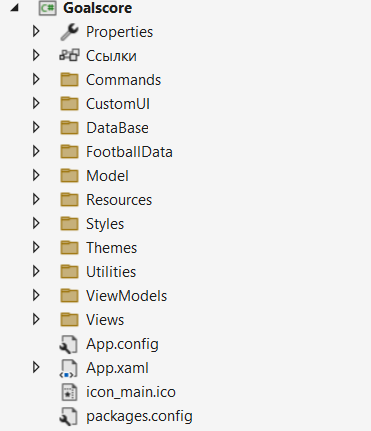


Рисунок 3.1 – Структура проекта

Описание структуры основных папок и файлов проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| Папка Commands | Содержит команды в которых содержится логика приложения |
| Папка CustomUI | Содержит пользовательские элементы управления |
| Папка FootballData | Содержит классы и интерфейсы для работы с football-data.org API |
| Папка DataBase | Содержит классы и интерфейсы для работы с данными БД. Они являются связующим элементом между данными в БД и теми данными, с которыми работает приложение |
| Папка Resources | Файлы ресурсов |

Продолжение таблицы 3.1

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| Папка Model | Здесь описаны модели, с которыми идёт работа в приложении:  Завершенный матч  Будущий матч  Пользователь  Уникальная команда пользователя  Избранный игрок пользователя  Игрок в уникальной команде пользователя |
| Папка View | Содержит все представления, которые позволяют пользователю работать с приложением. Они описывают графическую составляющую приложения. |
| Папка ViewModel | Содержит логику, которая позволяет получить данные при помощи View, обработать их, использую при этом Model, после чего передать в базу данных. |
| Папка Styles | Содержит стили |
| Папка Themes | Содержит темы |
| Папка Utilities | Содержит вспомогательные классы |
| App.config | Файл конфигурации приложения |
| App.xaml | Файл, который определяет ресурсы, которые являются общие для приложения |
| packages.config | Файл для для управления зависимостями и установки сторонних библиотек |
| icon\_main.ico | Логотип приложения для отображения в панели задач |

Таким образом, сформированная таблица помогает понять общую структуру проекта проектируемого программного средства.

Более подробная структура содержимого папок «Commands» и «CustomUI» программного средства показана на рисунке 3.2.

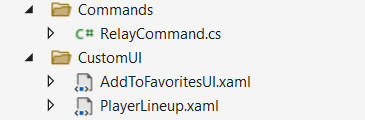


Рисунок 3.2 — Подробная структура папок «Commands» и «CustomUI»

Описание файлов папок представлено в таблице 3.2.

Таблица 3.2 – Описание файлов папок «Commands» и «CustomUI»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| RelayCommand.cs | Файл который представлят какие будут команды, наследуеться от интерфейса ICommand. |
| AddToFavoritesUI.xaml | Кнопка добавления игрока в избранное |
| PlayerLineup.xaml | Представлние игрока в расстановке |

Рассмотрим папку «Resources». Их подробная структура показана на рисунке 3.3.

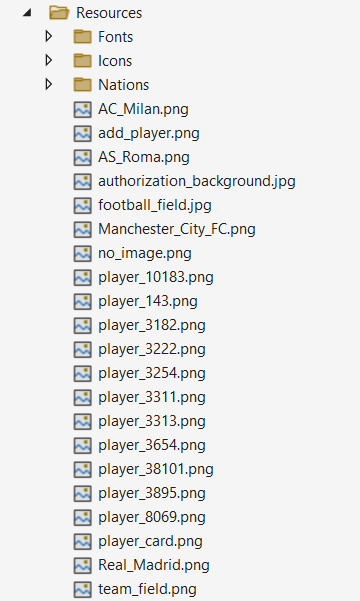


Рисунок 3.3 — Подробная структура папок «Resources»

Более подробное описание содержимого папки «Resources» представлено в таблице 3.3.

Таблица 3.3 – Описание файлов папки «Resources»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Папка Fonts | Содержит файлы шрифтов. |
| Папка Icons | Содержит изображения иконок. |
| Папка Nations | Содержит изображения флагов |
| Файлы .png | Изображения. |

Рассмотрим папку «ViewModels». Её подробная структура показана на рисунке 3.4.

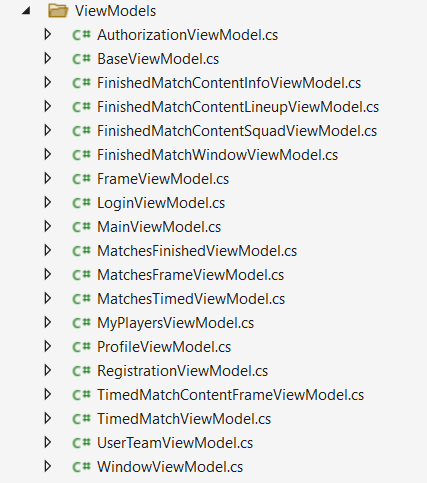


Рисунок 3.4 — Подробная структура папки «ViewModels»

Описание файлов папок представлено в таблице 3.4.

Таблица 3.4 – Описание файлов папки «ViewModels»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| AuthorizationViewModel.cs | Класс ViewModel для окна авторизации. |
| BaseViewModel.cs | Абстрактный класс ViewModel который описывает базовое представление. |

Продолжение таблицы 3.4

|  |  |
| --- | --- |
| Имя файла | Содержимое |
| FinishedMatchContentInfoViewModel.cs | Класс ViewModel для информации о завершенном матче. |
| FinishedMatchContentLineupViewModel.cs | Класс ViewModel для расстановки о завершенном матче. |
| FinishedMatchContentSquadViewModel.cs | Класс ViewModel для состава о завершенном матче. |
| FinishedMatchWindowViewModel.cs | Класс ViewModel для окна завершенного матча. |
| FrameViewModel.cs | Абстрактный класс от которого наследуются все Page для связи с окном в котором Page, а также для свзи ViewModel с самим View. |
| LoginViewModel.cs | Класс ViewModel для логина пользователя. |
| MainViewModel.cs | Класс ViewModel для главного окна приложения. |
| MatchesFinishedViewModel.cs | Класс ViewModel для списка завершенных матчей. |
| MatchesFrameViewModel.cs | Класс ViewModel для выбора завершенных/будуших матчей. |
| MatchesTimedViewModel.cs | Класс ViewModel для для списка будущих. |
| MyPlayersViewModel.cs | Класс ViewModel для страницы избранных игроков пользователя. |
| ProfileViewModel.cs | Класс ViewModel для профиля пользователя. |
| RegistrationViewModel.cs | Класс ViewModel для регистрации пользователя. |
| TimedMatchContentFrameViewModel.cs | Класс ViewModel для информации будущего матча. |
| TimedMatchViewModel.cs | Класс ViewModel для окна будущего матча. |
| UserTeamViewModel.cs | Класс ViewModel для страницы уникальной пользовательской команды. |
| WindowViewModel.cs | Абстрактный класс от которого наследуются все Window для связи ViewModel с View. |

Рассмотрим папку «Views». Её подробная структура показана на рисунке 3.5.

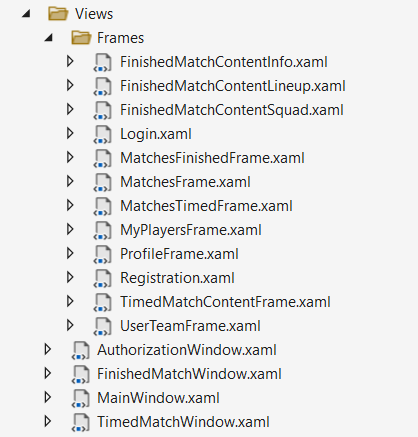


Рисунок 3.5 — Подробная структура папки «Views»

Описание файлов папок представлено в таблице 3.5.

Таблица 3.5 – Описание файлов папки «Views»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| AuthorizationWindow.xaml | Окно авторизации. |
| FinishedMatchWindow.xaml | Окно завершенного матча. |
| MainWindow.xaml | Главное окно приложения. |
| TimedMatchWindow.xaml | Окно будущего матча. |
| FinishedMatchContentInfo.xaml в папке Frames | Страница информации в окне завершенного матча. |
| FinishedMatchContentLineup.xaml в папке Frames | Страница расстановки в окне завершенного матча. |
| FinishedMatchContentSquad.xaml в папке Frames | Страница состава в окне завершенного матча. |
| Login.xaml в папке Frames | Страница логина пользователя. |
| MatchesFinishedFrame.xaml в папке Frames | Страница со списком завершенных матчей. |
| MatchesFrame.xaml в папке Frames | Страница матчей. |
| MatchesTimedFrame.xaml в папке Frames | Страница со списком будущих матчей. |

Продолжение таблицы 3.5

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| MyPlayersFrame.xaml в папке Frames | Страница избранных игроков пользователя. |
| ProfileFrame.xaml в папке Frames | Страница профиля пользователя. |
| Registration.xaml в папке Frames | Страница регистрации пользователя. |
| TimedMatchContentFrame.xaml в папке Frames | Страница информации будущего матча. |
| UserTeamFrame.xaml в папке Frames | Страница уникальной пользовательской команды. |

Рассмотрим папки «Styles», «Themes» и «Utilities». Их подробная структура показана на рисунке 3.6.

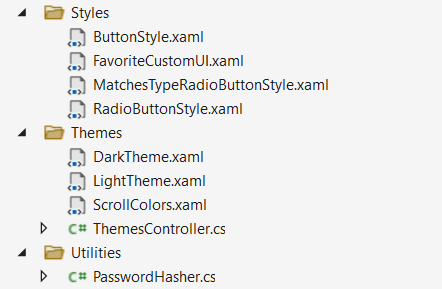


Рисунок 3.6 — Подробная структура папок «Styles», «Themes» и «Utilities»

Описание файлов папок представлено в таблице 3.6.

Таблица 3.6 – Описание файлов папок «Styles», «Themes» и «Utilities»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| ButtonStyle.cs | Стили для кнопки. |
| FavoriteCustomUI.cs | Стили для кнопок добавления в избранное. |
| MatchesTypeRadioButtonStyle | Стили для кнопок переключения между завершенными/будущими матчами. |
| RadioButtonStyle | Стили для кнопок в главном окне. |

Продолжение таблицы 3.6

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| DarkTheme.xaml | Словарь ресурсов для темной темы. |
| LightTheme.xaml | Словарь ресурсов для светлой темы. |
| ScrollColors.xaml | Словарь ресурсов для полоски прокрутки. |
| ThemesController.cs | Класс осуществляющий работу с переключением тем. |
| PasswordHasher.cs | Класс для работы с хэшированием паролей. |

Рассмотрим папки «DataBase», «FootballData» и «Model». Их подробная структура показана на рисунке 3.7.

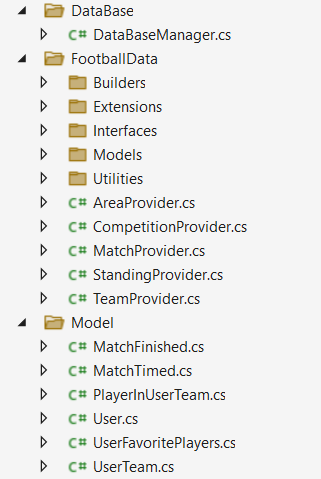


Рисунок 3.7 — Подробная структура папок «DataBase», «FootballData» и «Model»

Описание файлов папок представлено в таблице 3.7.

Таблица 3.7 – Описание файлов папок «DataBase», «FootballData» и «Model»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| DataBaseManager.cs | Класс для работы с БД. |
| Папка Builders | Папка которая содержит файлы для построения работы с football-data.org API. |

Продолжение таблицы 3.7

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| Папка Extensions | Содержит файлы для запросов данных из football-data.org API |
| Папка Interfaces | Содержит интерфейсы классов для работы с football-data.org API |
| Папка Models | Содержит основные модели |
| Папка Utilities | Содержит зависимости для работы с football-data.org API |
| AreaProvider.cs | Класс-провайдер данных локаций. |
| CompetitionProvider.cs | Класс-провайдер данных сорвенований. |
| MatchProvider.cs | Класс-провайдер данных матчей. |
| StandingProvider.cs | Класс-провайдер данных сезонов. |
| TeamProvider.cs | Класс-провайдер данных команд. |
| MatchFinished.cs | Модель завершенного матча |
| MatchTimed.cs | Модель будущего матча |
| PlayerInUserTeam.cs | Модель игрока в уникальной пользовательской команде. |
| User.cs | Модель пользователя. |
| UserFavoritePlayers.cs | Модель избранного игрока пользователя. |
| UserTeam.cs | Модель уникальной команды пользователя. |

В целом, описание структуры проекта позволяет лучше понимать, как устроено программное средство и какие компоненты в нем присутствуют.

## Разработка общей схемы приложения

Рассмотрим общую схему программного средства «Площадка объявлений». На рисунке 3.8 представлена общая схема работы приложения.

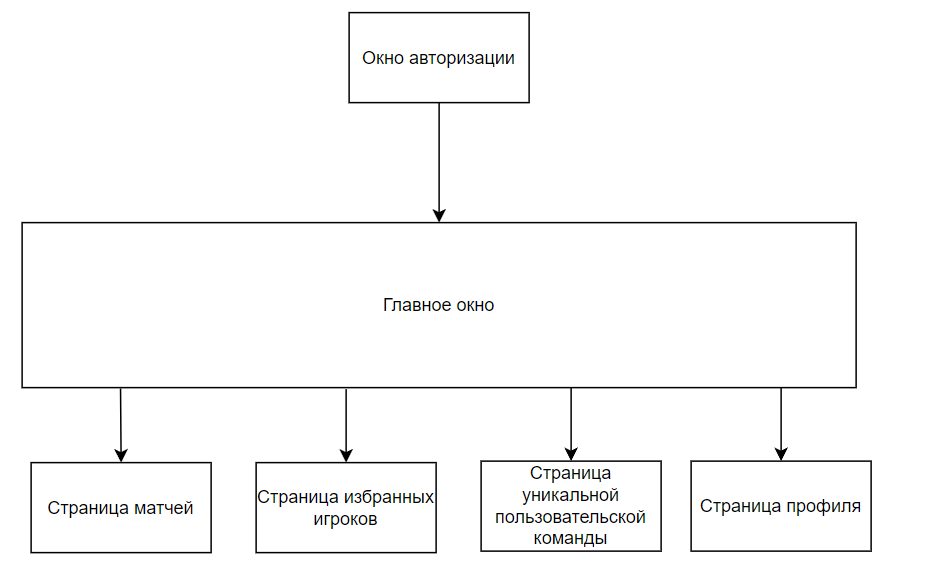


Рисунок 3.8 — Общая схема приложения

Из нее видно, что при запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы.

Если у нас нет учетной записи в системе, мы можем создать ее, нажав на кнопку "Регистрация" в окне авторизации. При регистрации нам нужно заполнить все поля, после чего вернуться в окно авторизации и ввести данные, указанные при регистрации.

После ввода логина и пароля в соответствующих полях, проверяется правильность ввода данных.

В случае неверного ввода данных, выведется соответствующее сообщение. В ином случае окно авторизации закроется и откроется главное окно приложения.

В главном окне работа идёт в страничном и оконных режимах.

Рассмотрим каждую страницу в отдельности.

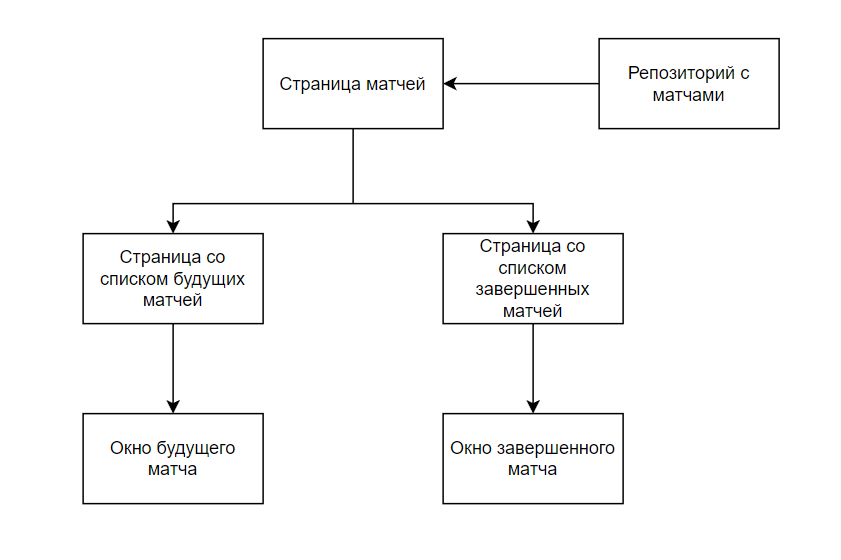


Рисунок 3.9 — Страница «Матчи»

Эта страница содержит в себе страницу со списков будущих/завершенных матчей.

На страницах будущих и завершенных матчей, можно осуществить поиск по названию команд. При двойном левом клике на выбранный матч, откроется его страница.

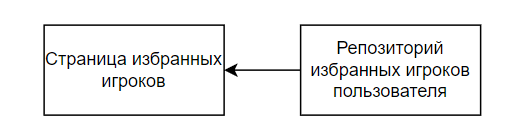


Рисунок 3.10 — Страница «Мои игроки»

На рисунке 3.10 показана схема работы страницы «Мои игроки».

На этой странице есть возможность просмотра, добавления и удаления избранных игроков пользователя.

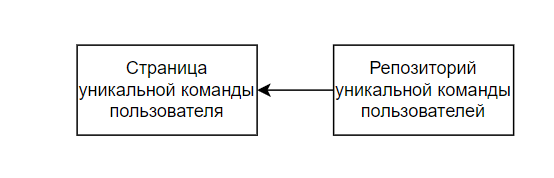


Рисунок 3.11 — Страница «Моя команда»

На рисунке 3.11 показана схема страницы «Моя команда».

На этой странице можно составлять уникальную пользовательскую команду.

Для добавления игрока нужно 1 нажатием левой кнопкой мыши выбрать из списка избранных игроков на нужного игрока. После этого на нужную позицию 1 раз нажать левой кнопкой мыши, для установки выбранного игрока на эту позицию.

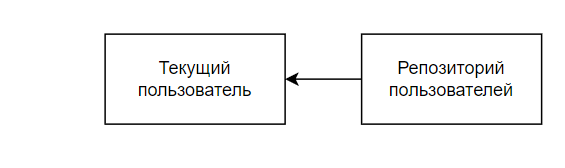


Рисунок 3.12 — Страница «Профиль»

Под картинкой профиля находится кнопка «Загрузить», нажатие на которую приводит к открытию окна с выбором файла картинки профиля, которую вы хотите установить в качестве основой.

## Взаимоотношение между классами

Для визуализации взаимосвязей между классами используется диаграмма UML – графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями).

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других. Диаграмма классов представлена в [приложении А](#_ПРИЛОЖЕНИЕ_А).

## Модель базы данных

Для реализации поставленной задачи была создана база данных Goalscore. Для ее создания использовалась система управления реляционными базами данных MS SQL Server. База данных состоит из таблиц, представленных на рисунке 3.13.

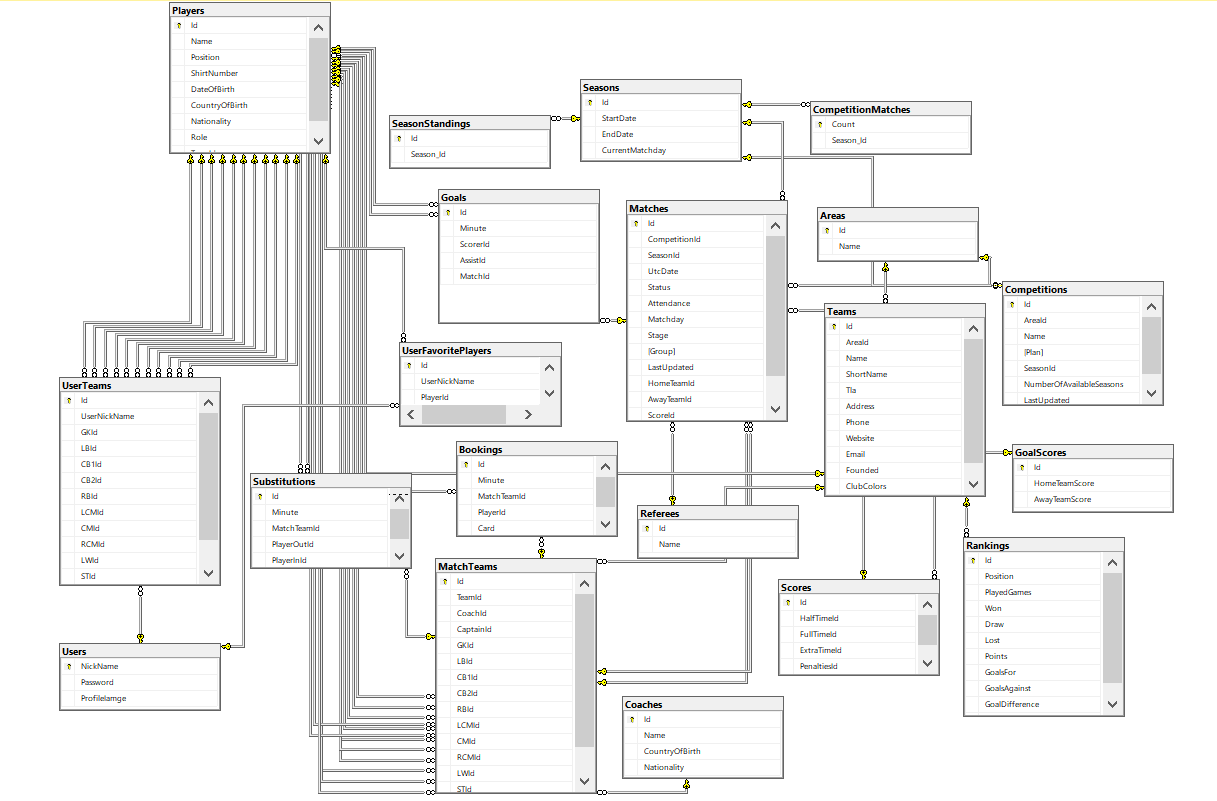


Рисунок 3.13 — База данных Goalscore

На рисунке 3.14 проиллюстрирована структура таблицы «Areas», которая содержит информацию странах. В данной таблице поле ID, представленное целочисленным значением, является первичным ключом. Поле Name хранит название страны.

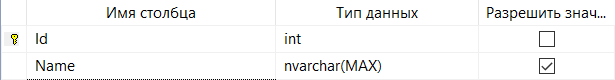


Рисунок 3.14 — Структура таблицы «Areas»

На рисунке 3.15 изображена структура таблицы «Bookings», содержащая информацию о предупреждениях. Первичный ключ Id хранит уникальный идентификатор, Minute – минута предупреждения, MatchTeamId – id заявки команды матча, PlayerId – id игрока, Card – карточка.

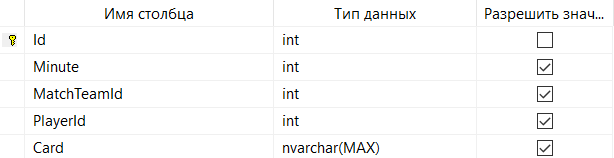


Рисунок 3.15 — Структура таблицы «Bookings»

На рисунке 3.16 изображена структура таблицы «Coaches», содержащая информацию о тренерах. Первичный ключ Id хранит уникальный идентификатор, Name – имя тренера, CountryOfBirth – страна рождения, Nationality – национальность.

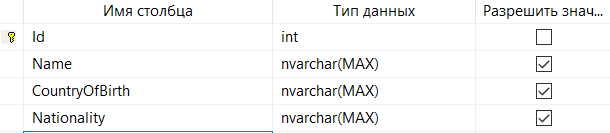


Рисунок 3.16 — Структура таблицы «Coaches»

На рисунке 3.17 изображена структура таблицы «CompetitionMatches», содержащая информацию о матчах соревнований. Первичный ключ Count хранит уникальный идентификатор, Season\_Id – Id сезона.

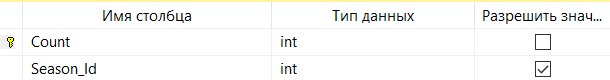


Рисунок 3.17 — Структура таблицы «CompetitionMatches»

На рисунке 3.18 изображена структура таблицы «Competitions», содержащая информацию о соревнований. Первичный ключ Id хранит уникальный идентификатор, AreaId – Id страны, Name – название турнира, [Plan] – план, SeasonId – Id сезона, NumberOfAvailableSeasons – количество допустимых сезонов, LastUpdated – время последнго обновления.

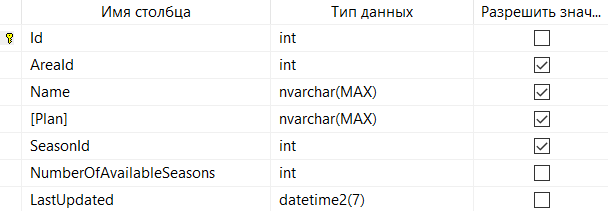


Рисунок 3.18 — Структура таблицы «Competitions»

На рисунке 3.19 изображена структура таблицы «Goals», содержащая информацию о голах. Первичный ключ Id хранит уникальный идентификатор, Minute – минута, ScorerId – Id забившего игрока, AssistId– Id игрока-ассистента, MatchId – Id матча.

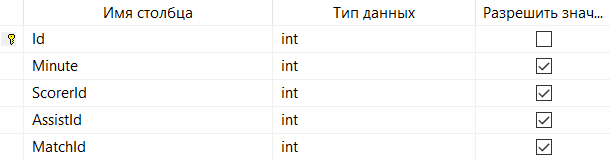


Рисунок 3.19 — Структура таблицы «Goals»

На рисунке 3.20 изображена структура таблицы «GoalScores», содержащая информацию о счёте. Первичный ключ Id хранит уникальный идентификатор, HomeTeamScore – счёт домашней команды, AwayTeamScore – счёт гостевой команды.

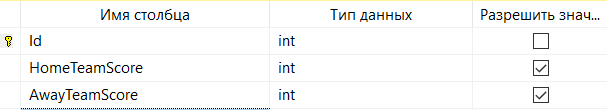


Рисунок 3.20 — Структура таблицы «GoalScores»

На рисунке 3.21 изображена структура таблицы «Matches», содержащая информацию о матчах. Первичный ключ Id хранит уникальный идентификатор, CompetitionId – Id соревнования, SeasonId – Id сезона, UtcDate – время начала матча, Status – статус матча, Attendance – посещяемость матча, Matchday – номер тура, Stage – стадия, [Group] – группа, LastUpdated – время последнего обновления, HomeTeamId – Id заявки домашней команды, AwayTeamId – Id заявки гостевой команды, ScoreId – Id счёта матча, RefereeId – Id судьи.



Рисунок 3.21 — Структура таблицы «Matches»

На рисунке 3.22 изображена структура таблицы «MatchTeams», содержащая информацию о заявке команды на матч. Первичный ключ Id хранит уникальный идентификатор, TeamId – Id команды, CoachId – Id тренера, GKId – Id игрока вратаря, LBId – Id игрока левого защитника, CB1Id – Id игрока центрального защитника 1, CB2Id – Id игрока центрального защитника 2, RBId – Id игрока правого защитника, LCMId – Id игрока левого полузащитника, CMId – Id игрока центрального полузащитника, RCMId – Id игрока правого полузащитника, LWId – Id игрока левого атакующего, STId – Id игрока форварда, RWId – Id игрока правого атакующего.

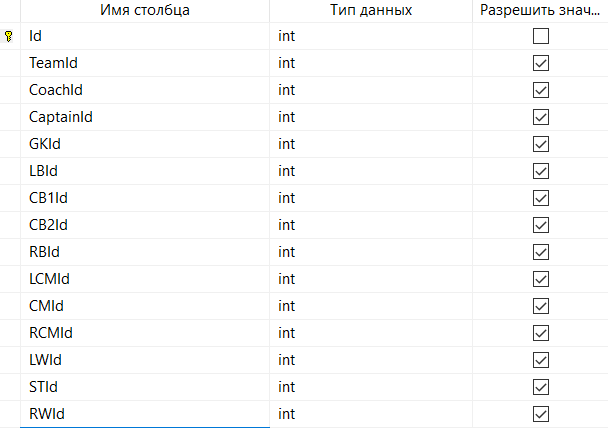


Рисунок 3.22 — Структура таблицы «MatchTeams»

На рисунке 3.23 изображена структура таблицы «Players», содержащая информацию о игроках. Первичный ключ Id хранит уникальный идентификатор, Name – имя игрока, Position – позицию игрока, ShirtNumber – номер игрока в команде, DateOfBirth – дата рождения игрока, CountryOfBirth – страна рождения игрока, Nationality – национальность игрока, Role – роль игрока, TeamId – Id команды игрока.

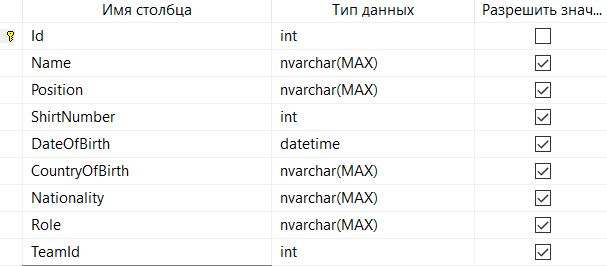


Рисунок 3.23— Структура таблицы «Players»

На рисунке 3.24 изображена структура таблицы «Rankings», содержащая информацию о рейтинге. Первичный ключ Id хранит уникальный идентификатор, Position – позиция, PlayedGames – количество игр, Won – количество побед, Draw – количество ничьих, Lost – количество поражений, Points – очки, GoalsFor – количество забитых голов, GoalsAgainst – количество пропущенных голов, GoalDifference – разница забитых и пропущенных голов, Team\_Id – Id команды.

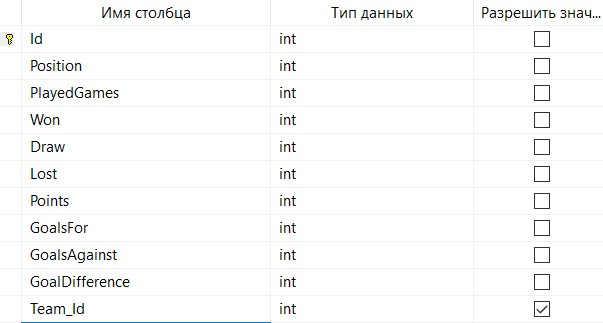


Рисунок 3.24 — Структура таблицы «Rankings»

На рисунке 3.25 изображена структура таблицы «Referees», содержащая информацию о судьях. Первичный ключ Id хранит уникальный идентификатор, Name – имя судьи.

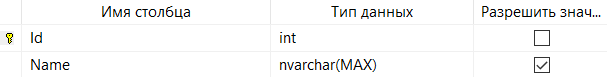


Рисунок 3.25 — Структура таблицы «Referees»

На рисунке 3.26 изображена структура таблицы «Scores», содержащая информацию о счётах матчей. Первичный ключ Id хранит уникальный идентификатор, HalfTimeId – Id счёта первого тайма, FullTimeId – Id счёта двух таймов, ExtraTimeId – Id счёта дополнтильеного тайма, PenaltiesId – Id счёта пенальти.

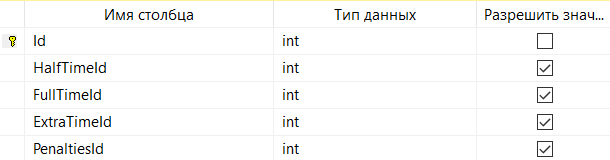


Рисунок 3.26 — Структура таблицы «Scores»

На рисунке 3.27 изображена структура таблицы «Seasons», содержащая информацию о счёте. Первичный ключ Id хранит уникальный идентификатор, StartDate – дата начала сезона, EndDate – дата конца сезона, CurrentMatchday – текущий тур.



Рисунок 3.27 — Структура таблицы «Seasons»

На рисунке 3.28 изображена структура таблицы «SeasonStandings», содержащая информацию о сезонных изменениях. Первичный ключ Id хранит уникальный идентификатор, Season\_Id – Id сезона.

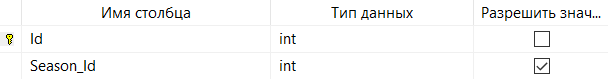


Рисунок 3.28 — Структура таблицы «SeasonStandings»

На рисунке 3.29 изображена структура таблицы «Standings», содержащая информацию о транирное состояние. Первичный ключ Id хранит уникальный идентификатор, Stage – стадия, Type – тип состояния .

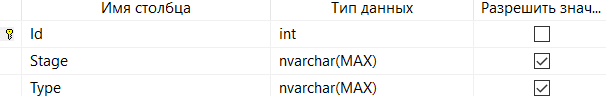


Рисунок 3.29 — Структура таблицы «Standings»

На рисунке 3.30 изображена структура таблицы «Substitutions», содержащая информацию о заменах. Первичный ключ Id хранит уникальный идентификатор, Minute – минута на которой сделана замена, MatchTeamId – Id заявки команды на матч, PlayerOutId – Id игрока которого заменили, PlayerInId – Id игрока который вышел на замену.

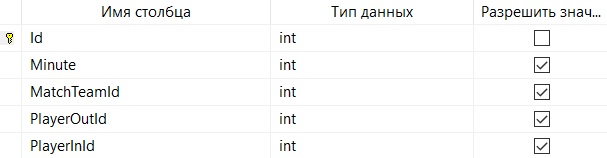


Рисунок 3.30 — Структура таблицы «Substitutions»

На рисунке 3.31 изображена структура таблицы «Teams», содержащая информацию о командах. Первичный ключ Id хранит уникальный идентификатор, AreaId – Id команды, Name – название команды, ShortName – короткое название команды, Tla – трехбуквенная аббревиатура команды, Address – адрес команды, Phone – телефон команды, Website – веб-сайт команды, Email – email команды, Founded – год основания команды, ClubColors – основные цвета команды, LastUpdated – последнее обновление информации о команде.

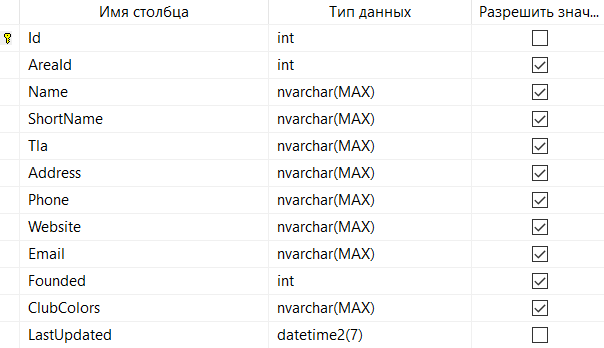


Рисунок 3.31 — Структура таблицы «Teams»

На рисунке 3.32 изображена структура таблицы «UserFavoritePlayers», содержащая информацию об избранных игроках пользователей. Первичный ключ Id хранит уникальный идентификатор, UserNickName – имя пользователя, PlayerId – Id игрока.

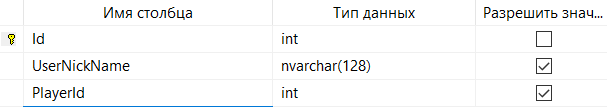


Рисунок 3.32 — Структура таблицы «UserFavoritePlayers»

На рисунке 3.33 изображена структура таблицы «Users», содержащая информацию о пользователях. Первичный ключ NickName хранит уникальное имя пользователя, Password – парль пользователя, ProfileImage – изображение профиля пользователя.

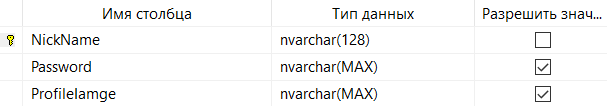


Рисунок 3.33 — Структура таблицы «Users»

На рисунке 3.34 изображена структура таблицы «UserTeams», содержащая информацию об уникальных командах пользователей. Первичный ключ Id хранит уникальный идентификатор, UserNickName – имя пользователя, GKId – Id игрока вратаря, LBId – Id игрока левого защитника, CB1Id – Id игрока центрального защитника 1, CB2Id – Id игрока центрального защитника 2, RBId – Id игрока правого защитника, LCMId – Id игрока левого полузащитника, CMId – Id игрока центрального полузащитника, RCMId – Id игрока правого полузащитника, LWId – Id игрока левого атакующего, STId – Id игрока форварда, RWId – Id игрока правого атакующего.

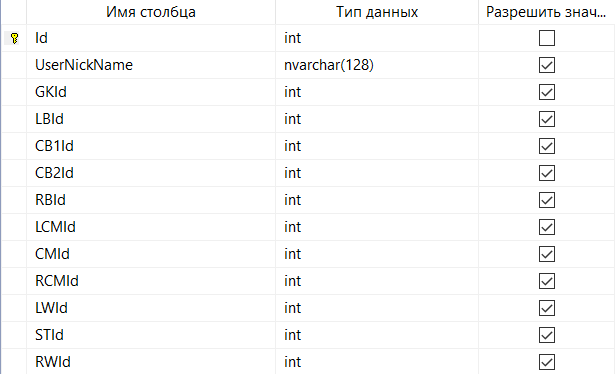


Рисунок 3.34 — Структура таблицы «UserTeams»

Создание базы данных позволит эффективно хранить и управлять информацией о пользователях, объявлениях и регионах, необходимой для функционирования интернет-магазина.

## Проектирование архитектуры приложения

Для общего представления функционального назначения системы используется диаграмма использования, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. На диаграмме использования применяются два типа основных сущностей: варианты использования и группы пользователей.

Разные группы пользователей в диаграмме называются актёрами, и обозначают любые сущности, использующие систему. Любая функция системы называется вариантом использования. Каждый вариант использования обозначает набор действий, который может быть использован актёром для взаимодействия с системой, и определяет набор действий, выполняемых этой системой.

Диаграмма использования представлена в [приложении Б](#_ПРИЛОЖЕНИЕ_Б).

## Проектирование последовательностей проекта

В целях визуализации взаимодействия объектов системы между собой во времени в едином сценарии использования используется ещё одна UML-диаграмма – диаграмма последовательностей. Данная диаграмма иллюстрирует, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования.

Для отображения течения времени используется линия жизни объекта, которая изображается с помощью штриховой линии, которая проводится вертикально вниз. С помощью линии жизни показывается период, в течение которого объект существует в системе. Сами объекты изображаются в виде прямоугольников, а сообщения, которыми они обмениваются – в виде линий со стрелками.

Диаграмма последовательностей представлена в [приложении В](#_ПРИЛОЖЕНИЕ_В).

Таким образом, благодаря проектированию программного средства удалось создать эффективное и удобное приложение, соответствующее всем требованиям и спецификациям, описанным в предыдущих разделах.

# Реализация программного средства

## Основные классы программного средства

Для выполнения технических задач программного средства «Футбольный менеджер Goalscore» должны быть реализованы следующие функции и соответствующие им классы и методы:

* выполнение входа;
* регистрация;
* просмотр матчей;
* составление уникальной команды;
* добавление игроков в избранное;
* функционирование панели администратора.

Далее подробно рассмотрены каждые из необходимых для выполнения технических задач функции, а также созданные для их выполнения классы и методы и их функционал и реализация.

## Выполнение входа

Графическая часть формы авторизации описана в Login.xaml. После нажатия кнопки «Login» сработает команда LoginCommand, преобразующий введённый пользователем пароль в строковое значение для последующей проверки на соответствие с сохранённым в базе данных паролем. Данная метод описан в классе LoginViewModel – логика модели представления для окна входа.

При успешной валидации введённых пользователем входных данных выполняется проверка на существование введённого логина в базе данных, и при нахождении значения выполняется проверка на соответствие введённого пароля и хранящегося в базе данных пароля. При успешном прохождении всех проверок пользователь перенаправляется на главную страницу, а противном случае – получает сообщение о соответствующей ошибке.

Таким образом, в классе AuthWindowViewModel определены свойства, методы и события, которые обеспечивают взаимодействие с базой данных и обработку действий пользователя при авторизации.

## Регистрация

Для совершения пользователем регистрации нового аккаунта реализованна страница Registration.xaml, в котором описана графическая составляющая окна «Registration». При переходе к окну регистрации с окна входа создается новый экземпляр страницы Registration.xaml.

При нажатии пользователя на кнопку «Registration» вызывается команда RegistrationCommand. Данная команда находится в классе RegistrationViewModel – логика модели представления для окна регистрации.

В началевведённые пользователем данные проходят проверку: поле с паролем и его подтверждением проверяются с помощью метода Equals().

После успешного прохождения вышеописанных проверок происходит попытка добавить пользователя в базу данных и перенаправление пользователя на главную страницу. Если логин, зарегистрирован в базе данных, то пользователь получает соответствующее уведомление об ошибке.

Таким образом, класс RegistrationViewModel содержит логику и данные, связанные с окном регистрации, и предоставляет методы и команды для выполнения регистрации и уведомлений об изменении значений свойств.

## Просмотр матчей

После успешной авторизации пользователь перенаправляется на основную страницу программного средства, графическая часть которой представлена классом MainWindow.

В классе предоставляющем логику для окна MainWindow содержится логика для перемещения между страницами матчей, избранных игроков, уникальной команды, профиля пользователя.

В классе отображения матчей MatchesFrame содержаться кнопки для переключения между завершенными и будущими матчами. В классе предоставляющем логику для страницы MatchesFrameViewModel осуществлена логика для переключения между страницами завершенных матчей, представленных классом MatchesFinishedFrame и классом его логики MatchesFinishedFrameViewModel, и будущих матчей, представленных страницей MatchesTimedFrame и классом его логики MatchesTimedFrameViewModel.

## Добавление игроков в избранное

Для добавления игрока в избранное необходимо перейти в страницу завершенных матчей. Двойным нажатием левой кнопкой мыши нажать по выбранному матчу. После чего откроется окно, представленное классом FinishedMatchWindow и классом его логики FinishedMatchWindowViewModel. Класс FinishedMatchWindowViewModel осуществляет логику переключения между страницами состава, расстановки, информации, их страницы представлены классами соответственно FinishedMatchContentSquad, FinishedMatchContentLineup, FinishedMatchContentInfo и соответственно их классами логики FinishedMatchContentSquadViewModel, FinishedMatchContentLineupViewModel, FinishedMatchContentInfoViewModel.

Необходимо перейти на страницу просмотра состава и нажать левой кнопкой мыши на кнопку в виде звезды, которая находиться рядом с именем игрока, для добавления игрока в избранное.

Реализация составляющих для функции добавления игрока в избранное представлена в приложении Г.

## Составление уникальной пользовательской команды

Для составления уникальной пользовательской команды необходимо в главном окне приложения перейти на страницу “My Team”, класс реализующий представление этой страницы и его класс логики соответственно UserTeamFrame и UserTeamViewModel. Далее в открытой странице в левом боковом списке избранных игроков выбрать игрока, и одним нажатием левой кнопкой мыши нажать на требуемую позицию для установки игрока на эту позицию.

Реализация составляющих для функции составления уникальной пользовательской команды представлена в приложении Г.

## Функционирование панели администратора

Для работы функций администратора осуществлены страницы представленные классами FinishedMatchFrame, MatchTeamFrame, MatchTeamsDataGrid, TimedMatchFrame которые отображаются на главной странице MainWindow. Их классы логики соответственно FinishedMatchViewModel, MatchTeamViewModel, MatchTeamsDataGridViewModel, TimedMatchViewModel, MainViewModel.

Листинги реализации данных классов представлены в приложении Г.

Таким образом, в разделе подробно описаны основные классы и методы, используемые для реализации функциональности программного средства «Футбольный менеджер Goalscore».

# Тестирование, проверка работоспособности и анализ полученных результатов

## Тестирование авторизации и регистрации

При тестировании данного приложения, были применены сценарии, которые могли бы привести к ошибке. В этой главе мы рассмотрим некоторые такие сценарии и посмотрим на их обработку.

В момент авторизации, возможна такая ситуация, в которой пользователь ничего ввел не существующий логин, или неправильный пароль. Обработка данного сценария приведена на рисунке 5.1.

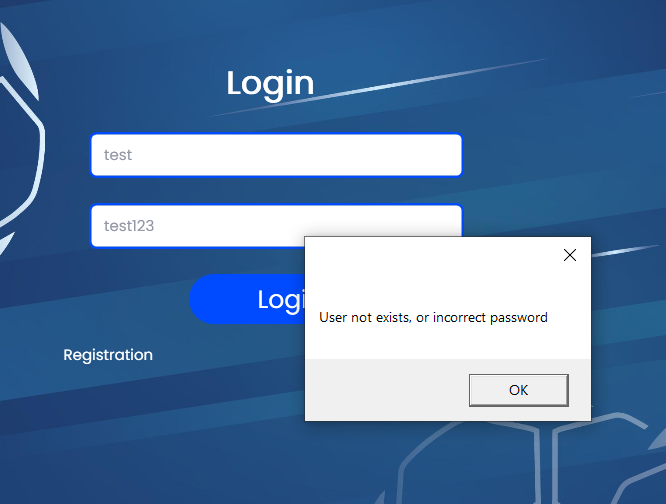


Рисунок 5.1 — Обработка неверных данных при авторизации

Как видно с рисунка, появилось окно с сообщением «User not exists, or incorrect password».

Также возможен такие случаи:

Пользователь, когда осуществляет ввод паролей, указывает несовпадающие пароли (рисунок 5.2).

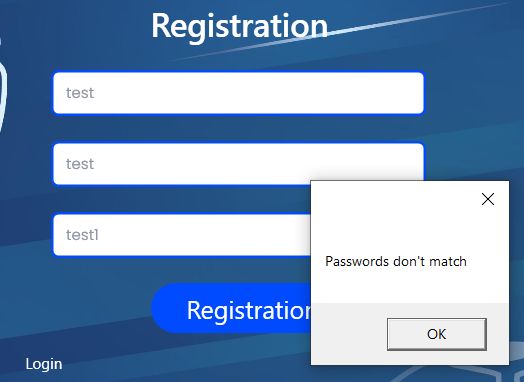


Рисунок 5.2 — Обработка не совпадающих паролей

## Тестирование составления уникальной команды

При заполнении формы объявления возможно возникновение исключительных ситуаций. При выставлении игрока нельзя повторно выставить одно и того же игрока в команду, а так же нельзя выставить игрока не выбрав при этом никакого игрока. Примеры обработки исключительных ситуаций представлены на рисунках 5.3 – 5.4.

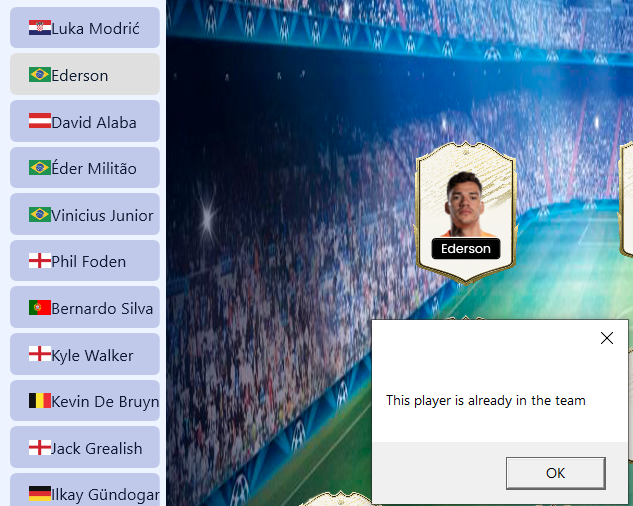


Рисунок 5.2 — Обработка установки повторного игрока

Как видно с рисунка, вывело сообщение «This player is already in the team» при попытке выставить повторного игрока.

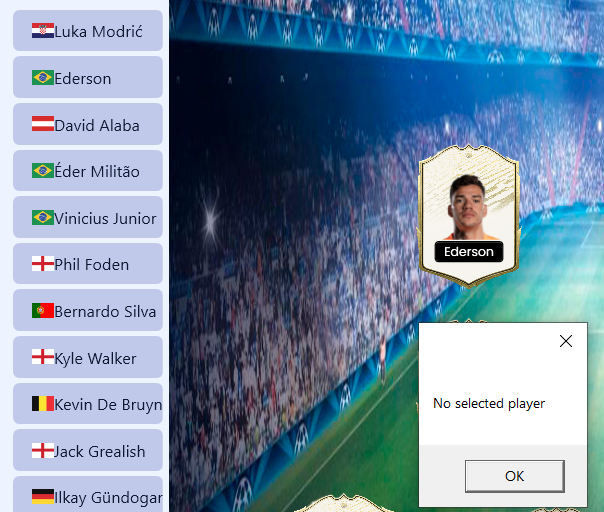


Рисунок 5.4 — Обработка установки игрока без выбора игрока

Как видно с рисунка, вывело сообщение «No selected player» при попытке выставить игрока, предварительно не выбрав нужного игрока.

Таким образом, в данном разделе были рассмотрены обработки всех исключений, которые могут быть вызваны пользователем при использовании программного средства.

# Руководство по использованию

При запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы (рисунок 6.1).

В случае, если вас нету своей учётной записи, её можно создать, нажав на кнопку «Registration» в окне авторизации. Вы будете направлены в окно регистрации, где вам нужно будет заполнить все поля. После успешной регистрации, вам нужно будет вернутся в окно авторизации и ввести ваши данные, указанные при регистрации.

Нажав на соответствующие кнопки в окне авторизации, у вас появится окно, где нужно будет ввести данные для входа в учётную запись, через которую производится вход.

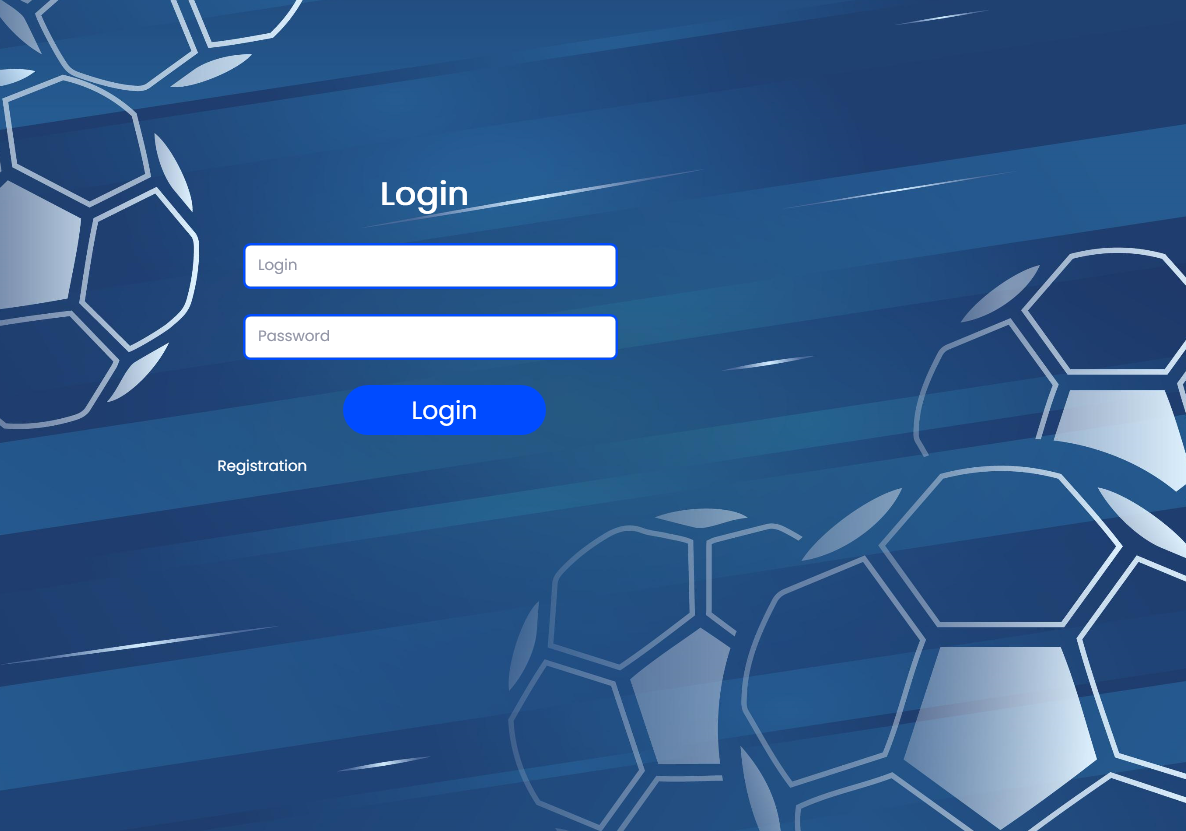


Рисунок 6.1 – Окно авторизации

После успешного входа, мы попадаем в главное окно, которое изображено на рисунке 6.2.

В этом окне мы можем посмотреть список завершенных и будущих матчей. Эти критерии находятся сверху страницы в виде кнопок «Timed» и «Finished». Поиск производится по в поле «Search» по названию команд.

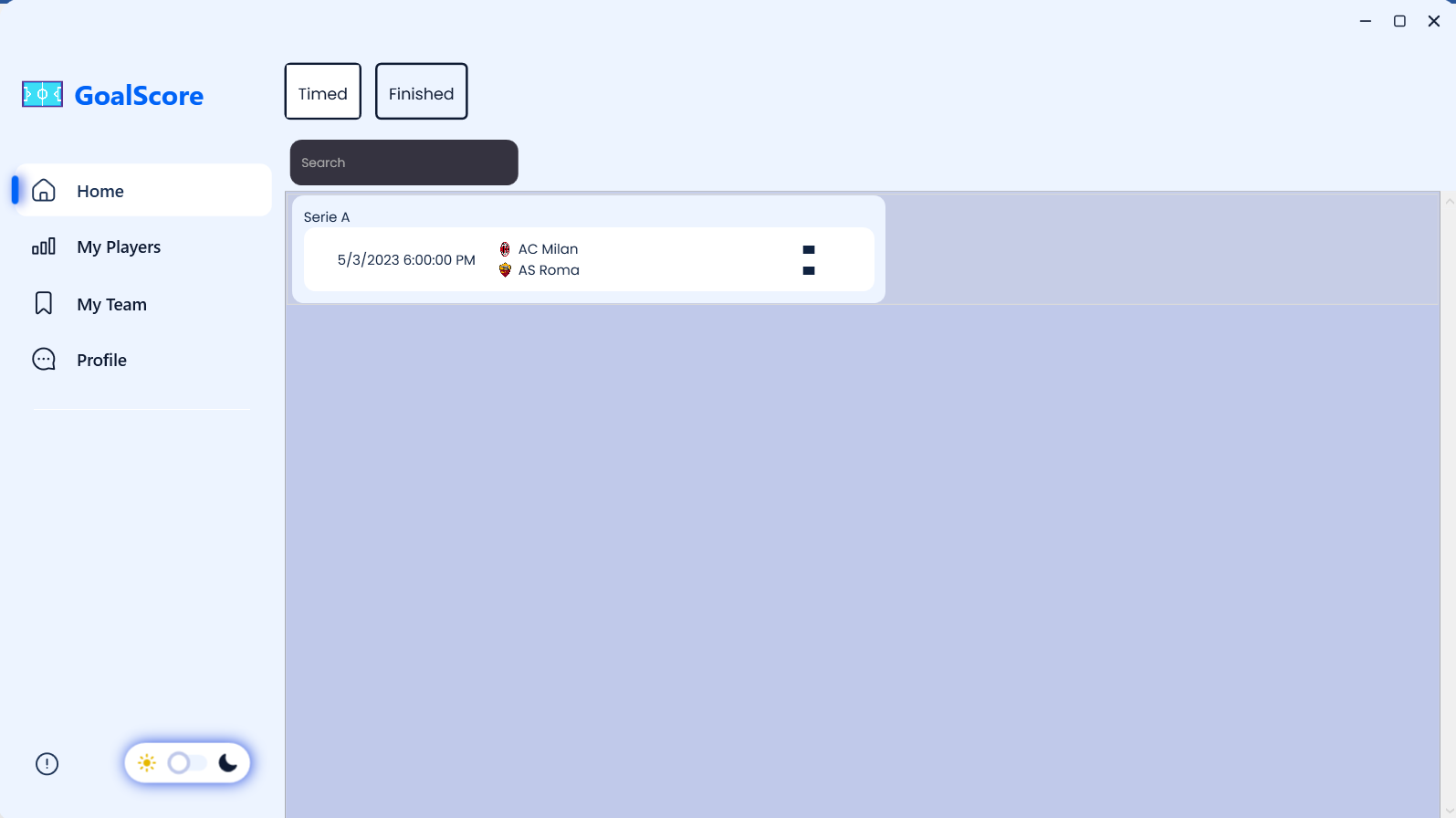


Рисунок 6.2 – Главное окно

Для просмотра определенного матча нужно двойным щелчком левой кнопкой мыши нажать по выбранному матчу. После чего откроется окно с выбранным матчем. У будущего матча не страниц, оно только содержит информацию о матче. У завершенного матча есть 3 страницы: состав, расстановка, информация. Данные страницы представлены на русинках 6.3 – 6.6.



Рисунок 6.3 – Окно будущего матча

Окно с информацией будущего матча.



Рисунок 6.4 – Страница состава окна завершенного матча

Страница со списком стартовых составов домашнем и гостевой команды, а также со списком запасных игроков домашней и гостевой команды.



Рисунок 6.5 – Страница расстановки окна завершенного матча

Страница с расстановкой матча, слева домашняя команда, справа гостевая.



Рисунок 6.6 – Страница информации окна завершенного матча

Страница с избранными игроками показывает список избранных игроков и их данные. Данная страница представлена на рисунке 6.7.



Рисунок 6.7 – Страница с избранными игроками пользователя

У игрока показаны фото, имя, позиция, дата рождения, национальность, фото команды, название команды.

Страница с уникальной пользовательской командой позволяет составить свою команду, для добавления игрока на требуемую позицию нужно выбрать игрока в левом боковом списке, после чего нажать левой кнопкой мыши на нужную позицию. Данная страница представлена на рисунке 6.8

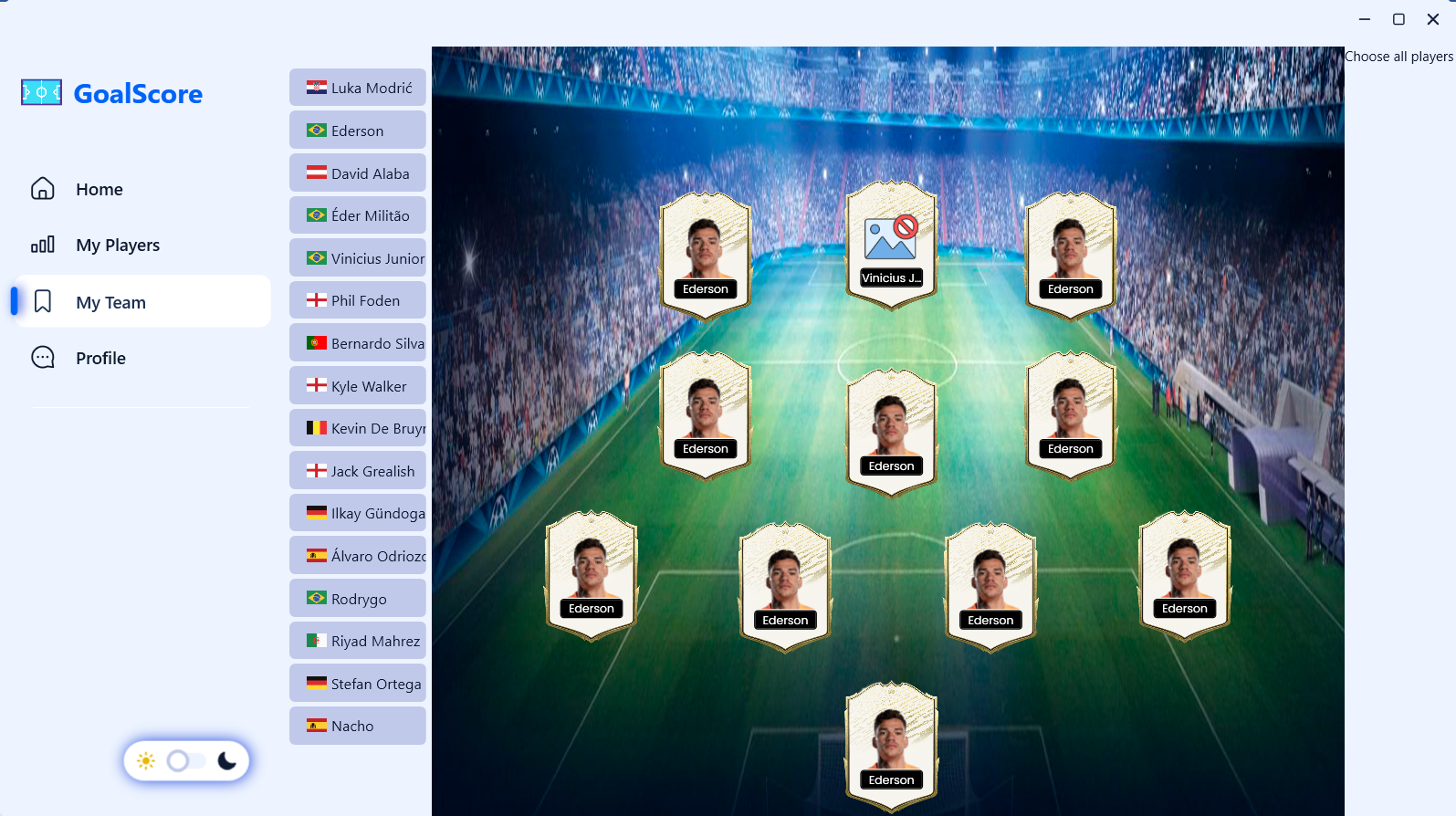


Рисунок 6.8 – Страница уникальной команды пользователя

В целом, данный предоставляет всю необходимую информацию, чтобы пользователи могли успешно использовать приложение, добавлять свои объявления, осуществлять поиск и управлять своими учетными записями, а администраторы имели полный контроль над системой и пользователями.

# ЗАКЛЮЧЕНИЕ

В данном курсовом проекте было разработано программное средство «Площадка объявлений» при помощи языка программирования C#, API-интерфейса Windows Presentation Foundation, технологии Entity Framework и базы данных Microsoft SQL Server. При выполнении курсового проекта использовались принципы и приемы ООП.

При разработке программного средства были выполнены все пункты из указанного списка предполагаемого основного функционала приложения, а именно:

Для пользователя:

* регистрация;
* авторизация;
* изменение данных своего профиля;
* просмотр доступных матчей;
* поиск матчей;
* добавление игроков в избранное;
* составление уникальной команды.

Для администратора:

* изменение данных своего профиля;
* добавление завершенных матчей;
* добавление будущих матчей;
* добавление состава команды;
* просмотр данных о игроках;
* просмотр данных о матчах;
* просмотр данных о судьях;
* просмотр данных о заявках на матч;
* просмотр данных о турнирах;
* просмотр данных о сезонах;
* просмотр данных о местах соревнований;
* просмотр данных о тренерах.

Тестирование программы показало, что она работает корректно и выполняет все свои функции.

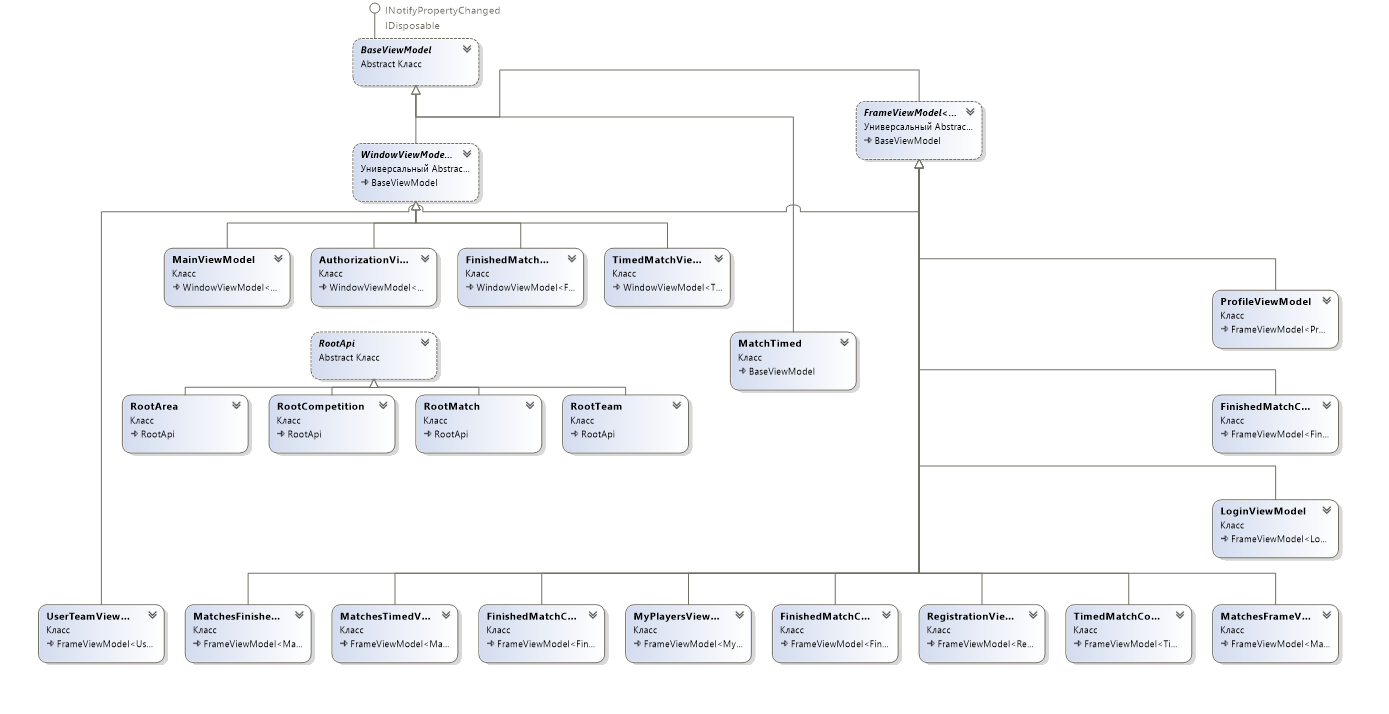
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

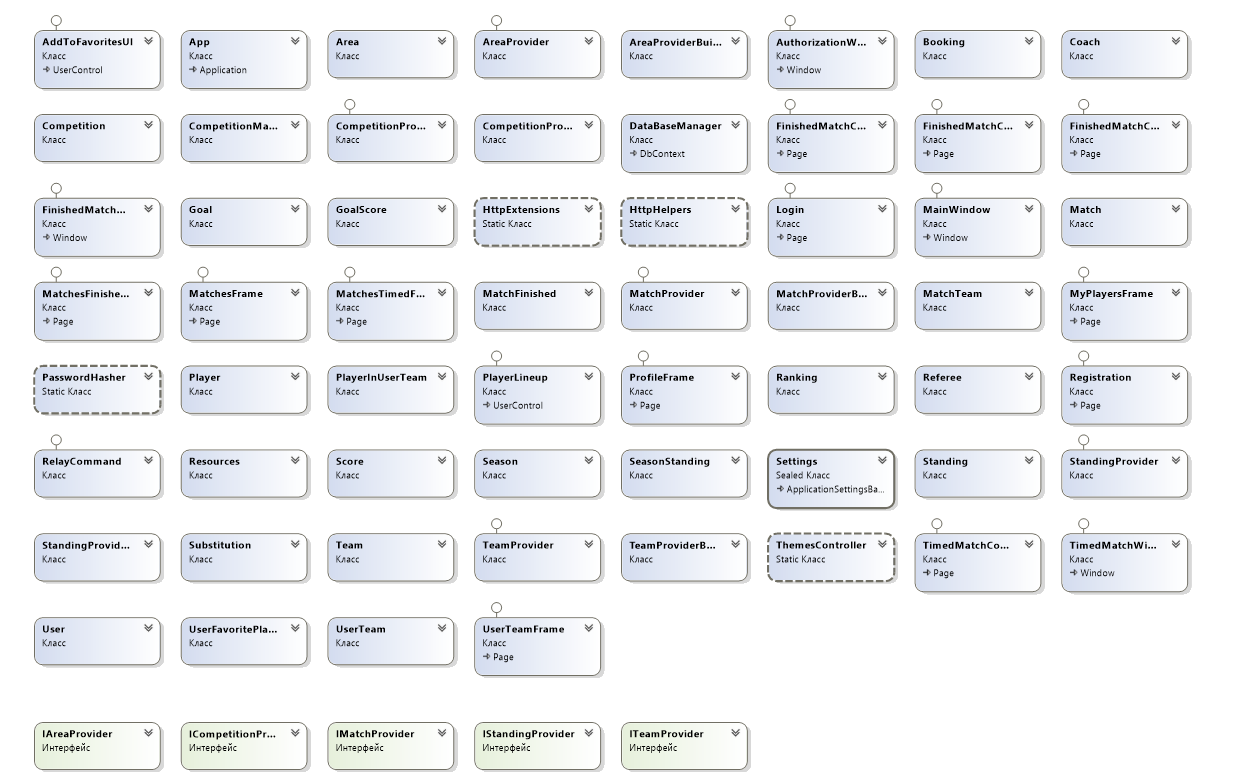
# Список использованных источников

1. Flashscore [Электронный ресурс] – https://www.flashscore.com/ – Дата доступа 12.03.2023
2. Microsoft Visual Studio [Электронный ресурс] – https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 23.04.2023
3. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 23.04.2023
4. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
5. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 25.04.2023
6. Руководство по XAML // [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/xaml/index.htm– Дата доступа: 25.04.2023
7. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity-framework/6/level1/ – Дата доступа 26.04.2023
8. Руководство по football-data.org API // [Электронный ресурс]. – Режим доступа: https://football-data.org/documentation/quickstart – Дата доступа: 28.04.2023
9. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

# ПРИЛОЖЕНИЕ А

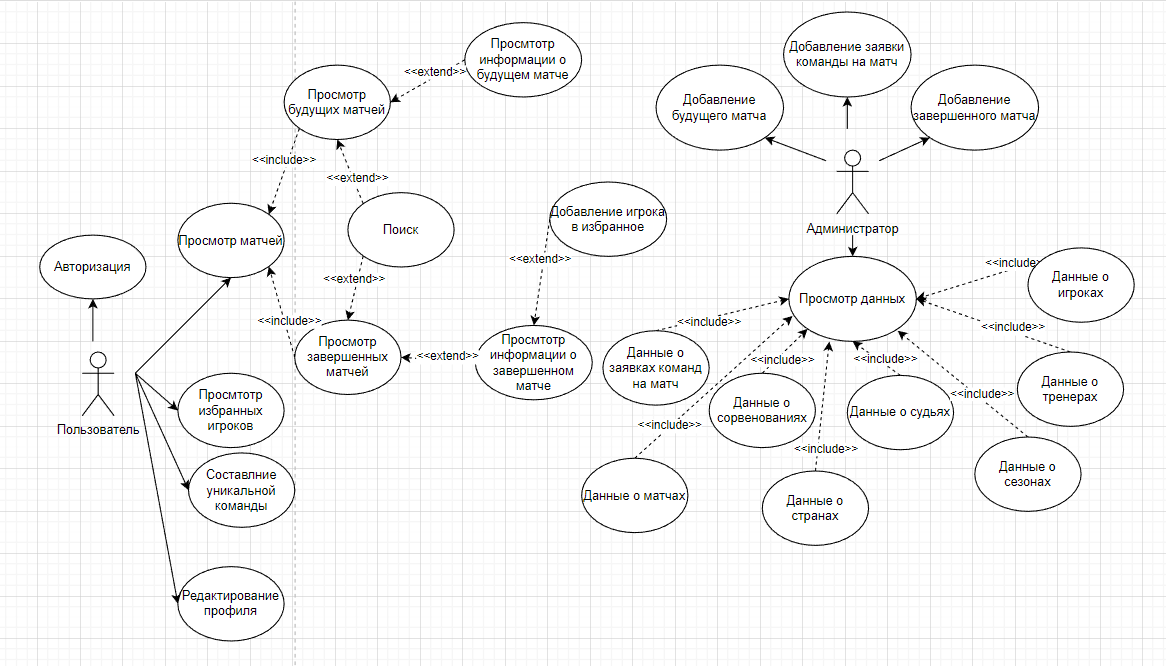
Диаграмма классов





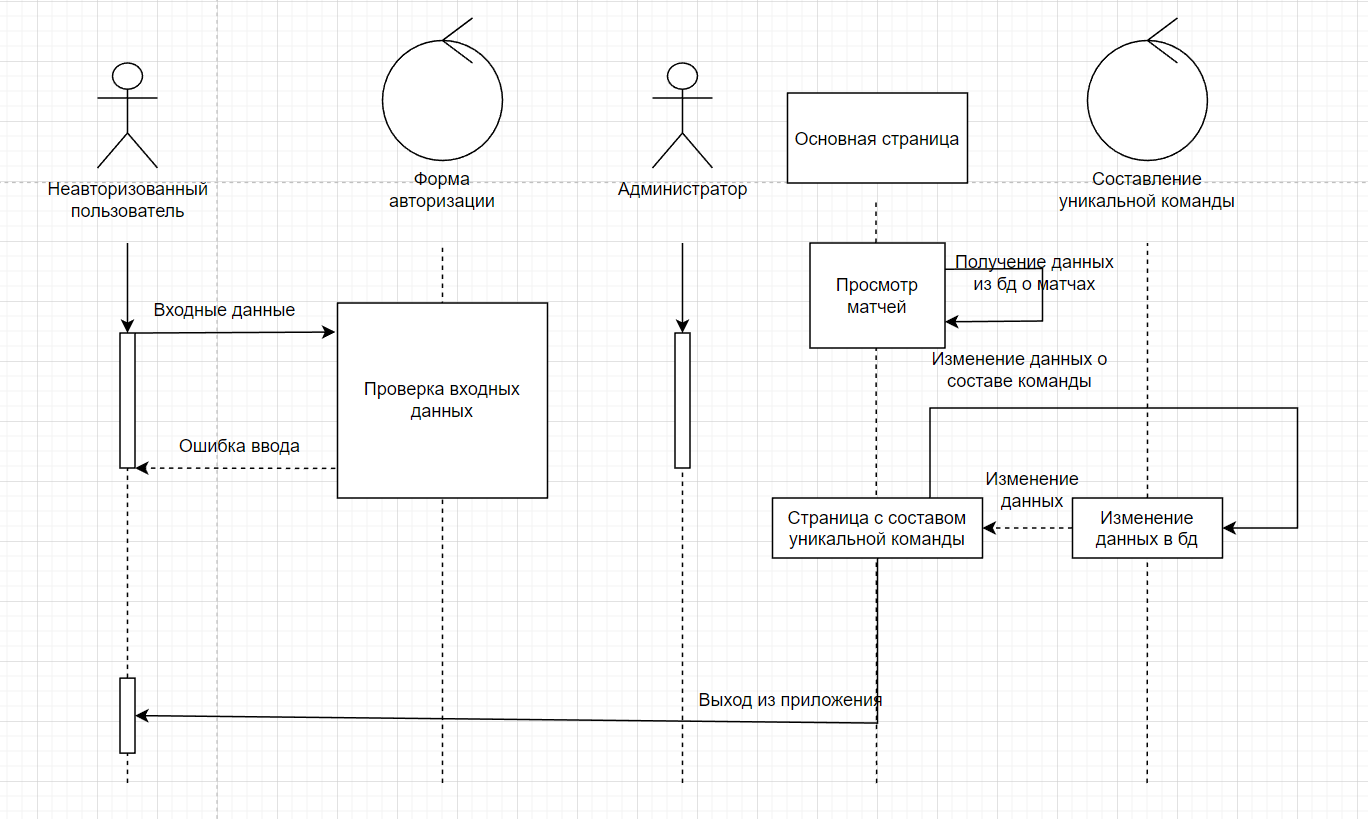
# ПРИЛОЖЕНИЕ Б

Диаграмма использований



# ПРИЛОЖЕНИЕ В

Диаграмма последовательностей



# ПРИЛОЖЕНИЕ Г

Листинг – AuthorizationViewModel

|  |
| --- |
| **using** **Goalscore.Views**;  **using** **Goalscore.Views.Frames**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **namespace** **Goalscore.ViewModels**  {  **public** **class** **AuthorizationViewModel** : WindowViewModel<AuthorizationWindow>  {  **public** **AuthorizationViewModel**(AuthorizationWindow owner) : **base**(owner)  {  SetFrame(**0**);  }  **public** **void** **SetFrame**(**int** id)  {  **switch** (id)  {  **case** **0**:  Owner.AuthorizationFrame.Content = **new** Login(Owner);  **break**;  **case** **1**:  Owner.AuthorizationFrame.Content = **new** Registration(Owner);  **break**;  }  }  }  } |

Листинг – BaseViewModel

|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Linq**;  **using** **System.Runtime.CompilerServices**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **namespace** **Goalscore.ViewModels**  {  **public** **abstract** **class** **BaseViewModel** : INotifyPropertyChanged, IDisposable  {  **public** **event** PropertyChangedEventHandler PropertyChanged;  **public** **void** **OnPropertyChanged**([CallerMemberName] **string** prop = "")  {  **if** (PropertyChanged != **null**)  PropertyChanged(**this**, **new** PropertyChangedEventArgs(prop));  }  **public** **virtual** **void** **Dispose**() { }  }} |

Листинг – FinishedMatchContentInfoViewModel

|  |
| --- |
| **using** **Goalscore.Model**;  **using** **Goalscore.Views**;  **using** **Goalscore.Views.Frames**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **namespace** **Goalscore.ViewModels**  {  **public** **class** **FinishedMatchContentInfoViewModel** : FrameViewModel<FinishedMatchContentInfo, FinishedMatchWindow>  {  **public** **FinishedMatchContentInfoViewModel**(FinishedMatchContentInfo owner, FinishedMatchWindow window, MatchFinished matchFinished) : **base**(owner, window)  {  Owner.RefereeTextBlock.Text = DataBase.DataBaseManager.Instance.refereeSet.ToList().Where(x => x.Id == matchFinished.RefereeId).Single().Name;  Owner.StadiumTextBlock.Text = DataBase.DataBaseManager.Instance.teamSet.ToList().Where(x => x.Id == matchFinished.HomeTeam.Id).Single().Address;  Owner.AttandenceTextBlock.Text = matchFinished.Attendance.ToString();  }  }  } |

Листинг – FinishedMatchContentLineupViewModel

|  |
| --- |
| **using** **FootballDataApi.Models**;  **using** **Goalscore.Model**;  **using** **Goalscore.Views**;  **using** **Goalscore.Views.Frames**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **namespace** **Goalscore.ViewModels**  {  **public** **class** **FinishedMatchContentLineupViewModel** : FrameViewModel<FinishedMatchContentLineup, FinishedMatchWindow>  {  **public** **FinishedMatchContentLineupViewModel**(FinishedMatchContentLineup owner, FinishedMatchWindow window, MatchFinished matchFinished) : **base**(owner, window)  {  MatchTeam homeMatchTeam = DataBase.DataBaseManager.Instance.matchTeamSet.ToList().Where(mt => mt.Id == matchFinished.HomeTeamId).Single();  MatchTeam awayMatchTeam = DataBase.DataBaseManager.Instance.matchTeamSet.ToList().Where(mt => mt.Id == matchFinished.AwayTeamId).Single();  Player HomeGK = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.GKId).Single();  Player HomeLB = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.LBId).Single();  Player HomeCB1 = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.CB1Id).Single();  Player HomeCB2 = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.CB2Id).Single();  Player HomeRB = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.RBId).Single();  Player HomeLCM = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.LCMId).Single();  Player HomeCM = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.CMId).Single();  Player HomeRCM = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.RCMId).Single();  Player HomeLW = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.LWId).Single();  Player HomeST = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.STId).Single();  Player HomeRW = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == homeMatchTeam.RWId).Single();  Player AwayGK = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.GKId).Single();  Player AwayLB = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.LBId).Single();  Player AwayCB1 = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.CB1Id).Single();  Player AwayCB2 = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.CB2Id).Single();  Player AwayRB = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.RBId).Single();  Player AwayLCM = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.LCMId).Single();  Player AwayCM = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.CMId).Single();  Player AwayRCM = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.RCMId).Single();  Player AwayLW = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.LWId).Single();  Player AwayST = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.STId).Single();  Player AwayRW = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == awayMatchTeam.RWId).Single();  Owner.HomeGK.Text = HomeGK.Name;  Owner.HomeGK.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeGK.Id);  Owner.HomeLB.Text = HomeLB.Name;  Owner.HomeLB.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeLB.Id);  Owner.HomeCB1.Text = HomeCB1.Name;  Owner.HomeCB1.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeCB1.Id);  Owner.HomeCB2.Text = HomeCB2.Name;  Owner.HomeCB2.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeCB2.Id);  Owner.HomeRB.Text = HomeRB.Name;  Owner.HomeRB.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeRB.Id);  Owner.HomeLCM.Text = HomeLCM.Name;  Owner.HomeLCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeLCM.Id);  Owner.HomeCM.Text = HomeCM.Name;  Owner.HomeCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeCM.Id);  Owner.HomeRCM.Text = HomeRCM.Name;  Owner.HomeRCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeRCM.Id);  Owner.HomeLW.Text = HomeLW.Name;  Owner.HomeLW.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeLW.Id);  Owner.HomeST.Text = HomeST.Name;  Owner.HomeST.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeST.Id);  Owner.HomeRW.Text = HomeRW.Name;  Owner.HomeRW.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(HomeRW.Id);  Owner.AwayGK.Text = AwayGK.Name;  Owner.AwayGK.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayGK.Id);  Owner.AwayLB.Text = AwayLB.Name;  Owner.AwayLB.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayLB.Id);  Owner.AwayCB1.Text = AwayCB1.Name;  Owner.AwayCB1.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayCB1.Id);  Owner.AwayCB2.Text = AwayCB2.Name;  Owner.AwayCB2.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayCB2.Id);  Owner.AwayRB.Text = AwayRB.Name;  Owner.AwayRB.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayRB.Id);  Owner.AwayLCM.Text = AwayLCM.Name;  Owner.AwayLCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayLCM.Id);  Owner.AwayCM.Text = AwayCM.Name;  Owner.AwayCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayCM.Id);  Owner.AwayRCM.Text = AwayRCM.Name;  Owner.AwayRCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayRCM.Id);  Owner.AwayLW.Text = AwayLW.Name;  Owner.AwayLW.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayLW.Id);  Owner.AwayST.Text = AwayST.Name;  Owner.AwayST.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayST.Id);  Owner.AwayRW.Text = AwayRW.Name;  Owner.AwayRW.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(AwayRW.Id);  }  }  } |

Листинг – FinishedMatchContentSquadViewModel

|  |
| --- |
| **using** **FootballDataApi.Models**;  **using** **Goalscore.Commands**;  **using** **Goalscore.CustomUI**;  **using** **Goalscore.DataBase**;  **using** **Goalscore.Model**;  **using** **Goalscore.Views**;  **using** **Goalscore.Views.Frames**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Linq**;  **using** **System.Security.Policy**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows**;  **using** **System.Windows.Input**;  **namespace** **Goalscore.ViewModels**  {  **public** **class** **FinishedMatchContentSquadViewModel** : FrameViewModel<FinishedMatchContentSquad, FinishedMatchWindow>  {  **private** List<Player> homeTeamStartPlayers;  **public** List<Player> HomeTeamStartPlayers  {  **get** => homeTeamStartPlayers;  }  **private** List<Player> awayTeamStartPlayers;  **public** List<Player> AwayTeamStartPlayers  {  **get** => awayTeamStartPlayers;  }  **private** List<Player> homeTeamBenchPlayers;  **public** List<Player> HomeTeamBenchPlayers  {  **get** => homeTeamBenchPlayers;  }  **private** List<Player> awayTeamBenchPlayers;  **public** List<Player> AwayTeamBenchPlayers  {  **get** => awayTeamBenchPlayers;  }  **private** List<Player> homeTeamPlayers;  **private** List<Player> awayTeamPlayers;  **public** **FinishedMatchContentSquadViewModel**(FinishedMatchContentSquad owner, FinishedMatchWindow window, MatchFinished matchFinished) : **base**(owner, window)  {  homeTeamPlayers = **new** List<Player>();  awayTeamPlayers = **new** List<Player>();  homeTeamStartPlayers = **new** List<Player>();  awayTeamStartPlayers = **new** List<Player>();  homeTeamBenchPlayers = **new** List<Player>();  awayTeamBenchPlayers = **new** List<Player>();  MatchTeam HomeMatchTeam = DataBase.DataBaseManager.Instance.matchTeamSet.ToList().Where(mt => mt.Id == matchFinished.HomeTeamId).Single();  MatchTeam AwayMatchTeam = DataBase.DataBaseManager.Instance.matchTeamSet.ToList().Where(mt => mt.Id == matchFinished.AwayTeamId).Single();  List<Player> players = DataBaseManager.Instance.playerSet.ToList();  **foreach** (Player p **in** players)  {  **if** (p.TeamId != **null**)  {  **if** (p.TeamId == matchFinished.HomeTeam.Id)  {  homeTeamPlayers.Add(p);  }  **if** (p.TeamId == matchFinished.AwayTeam.Id)  {  awayTeamPlayers.Add(p);  }  }  }  **foreach** (Player p **in** homeTeamPlayers)  **if** (p.Id == HomeMatchTeam.GKId ||  p.Id == HomeMatchTeam.LBId ||  p.Id == HomeMatchTeam.CB1Id ||  p.Id == HomeMatchTeam.CB2Id ||  p.Id == HomeMatchTeam.RBId ||  p.Id == HomeMatchTeam.LCMId ||  p.Id == HomeMatchTeam.CMId ||  p.Id == HomeMatchTeam.RCMId ||  p.Id == HomeMatchTeam.LWId ||  p.Id == HomeMatchTeam.STId ||  p.Id == HomeMatchTeam.RWId)  homeTeamStartPlayers.Add(p);  **foreach** (Player p **in** awayTeamPlayers)  **if** (p.Id == AwayMatchTeam.GKId ||  p.Id == AwayMatchTeam.LBId ||  p.Id == AwayMatchTeam.CB1Id ||  p.Id == AwayMatchTeam.CB2Id ||  p.Id == AwayMatchTeam.RBId ||  p.Id == AwayMatchTeam.LCMId ||  p.Id == AwayMatchTeam.CMId ||  p.Id == AwayMatchTeam.RCMId ||  p.Id == AwayMatchTeam.LWId ||  p.Id == AwayMatchTeam.STId ||  p.Id == AwayMatchTeam.RWId)  awayTeamStartPlayers.Add(p);  IEnumerable<Player> hometp = homeTeamPlayers.Where(p => !homeTeamStartPlayers.Select(spl => spl.Id).ToList().Contains(p.Id));  IEnumerable<Player> awaytp = awayTeamPlayers.Where(p => !awayTeamStartPlayers.Select(spl => spl.Id).ToList().Contains(p.Id));  **foreach**(Player p **in** hometp)  homeTeamBenchPlayers.Add(p);  **foreach** (Player p **in** awaytp)  awayTeamBenchPlayers.Add(p);  }  #region SelectedItemListViews  **private** Player homeStartSelectedPlayer;  **public** Player HomeStartSelectedPlayer  {  **get** { **return** homeStartSelectedPlayer; }  **set**  {  homeStartSelectedPlayer = **value**;  OnPropertyChanged("HomeStartSelectedPlayer");  }  }  **private** Player awayStartSelectedPlayer;  **public** Player AwayStartSelectedPlayer  {  **get** { **return** awayStartSelectedPlayer; }  **set**  {  awayStartSelectedPlayer = **value**;  OnPropertyChanged("AwayStartSelectedPlayer");  }  }  **private** Player homeBenchSelectedPlayer;  **public** Player HomeBenchSelectedPlayer  {  **get** { **return** homeBenchSelectedPlayer; }  **set**  {  homeBenchSelectedPlayer = **value**;  OnPropertyChanged("HomeBenchSelectedPlayer");  }  }  **private** Player awayBenchSelectedPlayer;  **public** Player AwayBenchSelectedPlayer  {  **get** { **return** awayBenchSelectedPlayer; }  **set**  {  awayBenchSelectedPlayer = **value**;  OnPropertyChanged("AwayBenchSelectedPlayer");  }  }  #endregion  }} |

Листинг – FinishedMatchWindowViewModel

|  |
| --- |
| **using** **Goalscore.Commands**;  **using** **Goalscore.Model**;  **using** **Goalscore.Views**;  **using** **Goalscore.Views.Frames**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows**;  **using** **System.Windows.Media.Imaging**;  **namespace** **Goalscore.ViewModels**  {  **public** **class** **FinishedMatchWindowViewModel** : WindowViewModel<FinishedMatchWindow>  {  **private** FinishedMatchContentInfo fmcINFO;  **private** FinishedMatchContentLineup fmcLineup;  **private** FinishedMatchContentSquad fmcSquad;  **public** **FinishedMatchWindowViewModel**(FinishedMatchWindow owner, MatchFinished match) : **base**(owner)  {  fmcINFO = **new** FinishedMatchContentInfo(owner, match);  fmcLineup = **new** FinishedMatchContentLineup(owner, match);  fmcSquad = **new** FinishedMatchContentSquad(owner, match);  owner.TimeMatchTextBlock.Text = match.HomeTeamScore.ToString() + " - " + match.AwayTeamScore.ToString();  **var** homeImageSource = **new** Uri(DataBase.DataBaseManager.GetTeamImageSource(match.HomeTeam.Id), UriKind.RelativeOrAbsolute);  Owner.HomeTeamImage.Source = **new** BitmapImage(homeImageSource);  **var** awayImageSource = **new** Uri(DataBase.DataBaseManager.GetTeamImageSource(match.AwayTeam.Id), UriKind.RelativeOrAbsolute);  Owner.AwayTeamImage.Source = **new** BitmapImage(awayImageSource);  DateTime d = (DateTime)match.UtcDate;  Owner.ScoreMatchTextBlock.Text = d.ToShortDateString();  }  **private** RelayCommand compositionCommand;  **public** RelayCommand CompositionCommand  {  **get**  {  **return** compositionCommand ??  (compositionCommand = **new** RelayCommand(obj =>  {  **try**  {  Owner.ContentFrame.Content = fmcSquad;  }  **catch** (Exception e)  {  MessageBox.Show(e.Message);  }  }));  }  }  **private** RelayCommand placementCommand;  **public** RelayCommand PlacementCommand  {  **get**  {  **return** placementCommand ??  (placementCommand = **new** RelayCommand(obj =>  {  **try**  {  Owner.ContentFrame.Content = fmcLineup;  }  **catch** (Exception e)  {  MessageBox.Show(e.Message);  }  }));  }  }  **private** RelayCommand informationCommand;  **public** RelayCommand InformationCommand  {  **get**  {  **return** informationCommand ??  (informationCommand = **new** RelayCommand(obj =>  {  **try**  {  Owner.ContentFrame.Content = fmcINFO;  }  **catch** (Exception e)  {  MessageBox.Show(e.Message);  }  }));  }  }  **private** RelayCommand closeWindowCommand;  **public** RelayCommand CloseWindowCommand  {  **get**  {  **return** closeWindowCommand ??  (closeWindowCommand = **new** RelayCommand(obj =>  {  Owner.Close();  }));  }  }  }  } |

Листинг - FrameViewModel

|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Controls**;  **using** **System.Windows**;  **namespace** **Goalscore.ViewModels**  {  **public** **abstract** **class** **FrameViewModel**<F, W> : BaseViewModel  **where** F : Page  **where** W : Window  {  **public** F Owner { **get**; **private** **set**; }  **public** W OwnerWindow { **get**; **private** **set**; }  **public** **FrameViewModel**(F owner, W window)  {  Owner = owner;  OwnerWindow = window;  }  }  } |

Листинг – LoginViewModel

|  |
| --- |
| **using** **Goalscore.Commands**;  **using** **Goalscore.Model**;  **using** **Goalscore.Utilities**;  **using** **Goalscore.Views**;  **using** **Goalscore.Views.Frames**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows**;  **namespace** **Goalscore.ViewModels**  {  **public** **class** **LoginViewModel** : FrameViewModel<Login, AuthorizationWindow>  {  **public** **LoginViewModel**(Login owner, AuthorizationWindow window) : **base**(owner, window)  {  }  **private** RelayCommand toRegistrationCommand;  **public** RelayCommand ToRegistrationCommand  {  **get**  {  **return** toRegistrationCommand ??  (toRegistrationCommand = **new** RelayCommand(obj =>  {  OwnerWindow.AuthorizationFrame.Content = **new** Registration(OwnerWindow);  }));  }  }  **private** RelayCommand loginCommand;  **public** RelayCommand LoginCommand  {  **get**  {  **return** loginCommand ??  (loginCommand = **new** RelayCommand(obj =>  {  **try**  {  **if** (DataBase.DataBaseManager.Instance.userSet.Find(Owner.LoginTextBox.Text) != **null**)  {  User user = DataBase.DataBaseManager.Instance.userSet.ToList().Where(u => u.NickName.Equals(Owner.LoginTextBox.Text)).Single();  **if** (!PasswordHasher.Compare(user.Password, Owner.PasswordTextBox.Text))  **throw** **new** **Exception**("User not exists, or incorrect password");  User.current = user;  MainWindow window = **new** MainWindow();  window.Show();  OwnerWindow.Close();  }  **else**  {  **throw** **new** **Exception**("User not exists, or incorrect password");  }  }  **catch** (Exception ex)  {  MessageBox.Show(ex.Message);  }  }));  }  }  }  } |

Листинг – MainViewModel

|  |
| --- |
| **using** **FootballDataApi**;  **using** **FootballDataApi.Models**;  **using** **Goalscore.Commands**;  **using** **Goalscore.Views**;  **using** **Goalscore.Views.Frames**;  **using** **Newtonsoft.Json**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Collections.ObjectModel**;  **using** **System.ComponentModel**;  **using** **System.Data.Entity**;  **using** **System.Data.Entity.Migrations**;  **using** **System.Linq**;  **using** **System.Net.Http**;  **using** **System.Runtime.CompilerServices**;  **using** **System.Security.Policy**;  **using** **System.Text**;  **using** **System.Threading**;  **using** **System.Threading.Tasks**;  **using** **System.Windows**;  **using** **WPFUIKitProfessional.Themes**;  **namespace** **Goalscore.ViewModels**  {  **public** **class** **MainViewModel** : WindowViewModel<MainWindow>  {  **private** **readonly** MatchesFrame matchesFrame;  **private** **readonly** ProfileFrame profileFrame;  **public** **MainViewModel**(MainWindow owner) : **base**(owner)  {  matchesFrame = **new** MatchesFrame(Owner);  profileFrame = **new** ProfileFrame(Owner);  Owner.MainFrame.Content = matchesFrame;  }  **private** RelayCommand themesChangeCommand;  **public** RelayCommand ThemesChangeCommand  {  **get**  {  **return** themesChangeCommand ??  (themesChangeCommand = **new** RelayCommand(obj =>  {  **if** (Owner.Themes.IsChecked == **true**)  ThemesController.SetTheme(ThemesController.ThemeTypes.Dark);  **else**  ThemesController.SetTheme(ThemesController.ThemeTypes.Light);  }));  }  }  #region SetPagesCommands  **private** RelayCommand homePageCommand;  **public** RelayCommand HomePageCommand  {  **get**  {  **return** homePageCommand ??  (homePageCommand = **new** RelayCommand(obj =>  {  Owner.MainFrame.Content = matchesFrame;  }));  }  }  **private** RelayCommand myPlayersPageCommand;  **public** RelayCommand MyPlayersPageCommand  {  **get**  {  **return** myPlayersPageCommand ??  (myPlayersPageCommand = **new** RelayCommand(obj =>  {  Owner.MainFrame.Content = **new** MyPlayersFrame(Owner);  }));  }  }    **private** RelayCommand userTeamPageCommand;  **public** RelayCommand UserTeamPageCommand  {  **get**  {  **return** userTeamPageCommand ??  (userTeamPageCommand = **new** RelayCommand(obj =>  {  Owner.MainFrame.Content = **new** UserTeamFrame(Owner);  }));  }  }  **private** RelayCommand profilePageCommand;  **public** RelayCommand ProfilePageCommand  {  **get**  {  **return** profilePageCommand ??  (profilePageCommand = **new** RelayCommand(obj =>  {  Owner.MainFrame.Content = profileFrame;  }));  }  }  #endregion  #region TopPanelButton  **private** RelayCommand closeWindowCommand;  **public** RelayCommand CloseWindowCommand  {  **get**  {  **return** closeWindowCommand ??  (closeWindowCommand = **new** RelayCommand(obj =>  {  Owner.Close();  }));  }  }  **private** RelayCommand restoreWindowCommand;  **public** RelayCommand RestoreWindowCommand  {  **get**  {  **return** restoreWindowCommand ??  (restoreWindowCommand = **new** RelayCommand(obj =>  {  **if** (Owner.WindowState == WindowState.Normal)  {  Owner.WindowState = WindowState.Maximized;  **return**;  }  **if** (Owner.WindowState == WindowState.Maximized)  {  Owner.WindowState = WindowState.Normal;  **return**;  }  }));  }  }  #endregion  }  } |

Листинг – MatchesFinishedViewModel

|  |
| --- |
| **using** **Goalscore.Commands**;  **using** **Goalscore.Model**;  **using** **Goalscore.Views**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Collections.ObjectModel**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows**;  **namespace** **Goalscore.ViewModels**  {  **public** **class** **MatchesFinishedViewModel** : FrameViewModel<MatchesFinishedFrame, MainWindow>  {  **private** ObservableCollection<MatchFinished> matches;  **public** ObservableCollection<MatchFinished> Matches  {  **get** => matches;  **set**  {  matches = **value**;  OnPropertyChanged("Matches");  }  }  **private** ObservableCollection<MatchFinished> originalMathces;  **public** **MatchesFinishedViewModel**(MatchesFinishedFrame owner, MainWindow window) : **base**(owner, window)  {  matches = **new** ObservableCollection<MatchFinished>();  originalMathces = **new** ObservableCollection<MatchFinished>();  **var** timedTeams = **from** m **in** DataBase.DataBaseManager.Instance.matchSet  **join** c **in** DataBase.DataBaseManager.Instance.competitionSet on m.CompetitionId **equals** c.Id  **join** a **in** DataBase.DataBaseManager.Instance.areaSet on c.AreaId **equals** a.Id  **where** m.Status == "FINISHED"  **select** **new**  {  MatchId = m.Id,  CompetitionName = c.Name,  UtcDate = m.UtcDate,  Status = m.Status,  HomeTeamId = m.HomeTeamId,  AwayTeamId = m.AwayTeamId,  ScoreId = m.ScoreId,  RefereeId = m.RefereeId,  LastUpdated = m.LastUpdated,  AreaName = a.Name,  Attendance = m.Attendance  };  **foreach** (**var** i **in** timedTeams)  {  originalMathces.Add(**new** MatchFinished(i.MatchId, i.CompetitionName, i.UtcDate, i.Status, i.HomeTeamId, i.AwayTeamId, i.ScoreId, i.RefereeId, i.LastUpdated, i.AreaName, i.Attendance));  matches.Add(**new** MatchFinished(i.MatchId, i.CompetitionName, i.UtcDate, i.Status, i.HomeTeamId, i.AwayTeamId, i.ScoreId, i.RefereeId, i.LastUpdated, i.AreaName, i.Attendance));  }  }  **private** RelayCommand homeTeamCommand;  **public** RelayCommand HomeTeamCommand  {  **get**  {  **return** homeTeamCommand ??  (homeTeamCommand = **new** RelayCommand(obj =>  {  MessageBox.Show("Hello");  }));  }  }  **private** RelayCommand showMatchCommand;  **public** RelayCommand ShowMatchCommand  {  **get**  {  **return** showMatchCommand ??  (showMatchCommand = **new** RelayCommand(obj =>  {  FinishedMatchWindow timedMatchWindow = **new** FinishedMatchWindow(OwnerWindow, Matches[SelectedIndex]);  timedMatchWindow.ShowDialog();  }));  }  }  **private** **int** \_selectedIndex;  **public** **int** SelectedIndex  {  **get** => \_selectedIndex;  **set**  {  **if** (\_selectedIndex == **value**)  **return**;  \_selectedIndex = **value**;  OnPropertyChanged("SelectedIndex");  }  }  **private** RelayCommand searchCommand;  **public** RelayCommand SearchCommand  {  **get**  {  **return** searchCommand ??  (searchCommand = **new** RelayCommand(obj =>  {  **if** (**string**.IsNullOrEmpty(Owner.SearchBox.Text))  {  Matches = originalMathces;  }  **else**  {  List<MatchFinished> lstPr = **new** List<MatchFinished>();  **foreach** (MatchFinished pr **in** originalMathces)  lstPr.Add(pr);  **var** listItog = **from** p **in** lstPr  **where** p.HomeTeamName.Contains(Owner.SearchBox.Text) || p.AwayTeamName.Contains(Owner.SearchBox.Text)  **select** p;  ObservableCollection<MatchFinished> SearchedMathces = **new** ObservableCollection<MatchFinished>();  **foreach** (MatchFinished p **in** listItog)  SearchedMathces.Add(p);  Matches = SearchedMathces;  }  }));  }  }  }  } |

Листинг – MatchesFrameViewModel

|  |
| --- |
| **using** **Goalscore.Commands**;  **using** **Goalscore.Views**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows**;  **namespace** **Goalscore.ViewModels**  {  **public** **enum** MatchesType  {  TIMED,  FINISHED  }  **public** **class** **MatchesFrameViewModel** : FrameViewModel<MatchesFrame, MainWindow>  {  **private** MatchesType m\_type = MatchesType.TIMED;  **public** MatchesType MatchesType { **get** { **return** m\_type; }  **set**  {  **if**(m\_type == **value**)  **return**;  m\_type = **value**;  OnPropertyChanged(nameof(MatchesType));  OnPropertyChanged("IsTimedMatches");  OnPropertyChanged("IsFinishedMatches");  }  }  **public** **bool** IsTimedMatches  {  **get** { **return** MatchesType == MatchesType.TIMED; }  **set** { MatchesType = **value** ? MatchesType.TIMED : MatchesType; }  }  **public** **bool** IsFinishedMatches  {  **get** { **return** MatchesType == MatchesType.FINISHED; }  **set** { MatchesType = **value** ? MatchesType.FINISHED : MatchesType; }  }  MatchesFinishedFrame mff;  MatchesTimedFrame mtf;  **public** **MatchesFrameViewModel**(MatchesFrame owner, MainWindow window) : **base**(owner, window)  {  mff = **new** MatchesFinishedFrame(Owner);  mtf = **new** MatchesTimedFrame(Owner);    Owner.MatchFrame.Content = mtf;  }  **private** RelayCommand setFinishedMatchesCommand;  **private** RelayCommand setTimedMatchesCommand;  **public** RelayCommand SetFinishedMatchesCommand  {  **get**  {  **return** setFinishedMatchesCommand ??  (setFinishedMatchesCommand = **new** RelayCommand(obj =>  {  Owner.MatchFrame.Content = mff;  }));  }  }  **public** RelayCommand SetTimedMatchesCommand  {  **get**  {  **return** setTimedMatchesCommand ??  (setTimedMatchesCommand = **new** RelayCommand(obj =>  {  Owner.MatchFrame.Content = mtf;  }));  }  }  }  } |

Листинг - MatchesTimedViewModel

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using FootballDataApi.Models;  using Goalscore.Commands;  using Goalscore.Model;  using Goalscore.Views;  namespace Goalscore.ViewModels  {  public class MatchesTimedViewModel : FrameViewModel<MatchesTimedFrame, MainWindow>  {  private ObservableCollection<MatchTimed> matches;  public ObservableCollection<MatchTimed> Matches  {  get => matches;  set  {  matches = value;  OnPropertyChanged("Matches");  }  }  private ObservableCollection<MatchTimed> originalMathces;  MainWindow mainWindow;  public MatchesTimedViewModel(MatchesTimedFrame owner, MainWindow window) : base(owner, window)  {  mainWindow = window;  matches = new ObservableCollection<MatchTimed>();  originalMathces = new ObservableCollection<MatchTimed>();  var timedTeams = from m in DataBase.DataBaseManager.Instance.matchSet  join c in DataBase.DataBaseManager.Instance.competitionSet on m.CompetitionId equals c.Id  join a in DataBase.DataBaseManager.Instance.areaSet on c.AreaId equals a.Id  where m.Status == "TIMED"  select new  {  MatchId = m.Id,  CompetitionName = c.Name,  UtcDate = m.UtcDate,  Status = m.Status,  HomeTeamId = m.HomeTeamId,  AwayTeamId = m.AwayTeamId,  ScoreId = m.ScoreId,  RefereeId = m.RefereeId,  LastUpdated = m.LastUpdated,  AreaName = a.Name  };  var mdf = DataBase.DataBaseManager.Instance.matchSet.Join(DataBase.DataBaseManager.Instance.competitionSet,  p => p.CompetitionId, // Match  c => c.Id, // Competition  (p, c) => new  {  MatchId = p.Id,  CompetitionName = c.Name,  UtcDate = p.UtcDate,  Status = p.Status,  HomeTeamId = p.HomeTeamId,  AwayTeamId = p.AwayTeamId,  ScoreId = p.ScoreId,  RefereeId = p.RefereeId,  LastUpdated = p.LastUpdated  });  foreach (var i in timedTeams)  {  if (i.HomeTeamId == null || i.AwayTeamId == null || i.RefereeId == null)  { continue; }  else  {  originalMathces.Add(new MatchTimed(i.MatchId, i.CompetitionName, i.UtcDate, i.Status, i.HomeTeamId, i.AwayTeamId, i.ScoreId, i.RefereeId, i.LastUpdated, i.AreaName));  matches.Add(new MatchTimed(i.MatchId, i.CompetitionName, i.UtcDate, i.Status, i.HomeTeamId, i.AwayTeamId, i.ScoreId, i.RefereeId, i.LastUpdated, i.AreaName));  }  }  }  private RelayCommand homeTeamCommand;  public RelayCommand HomeTeamCommand  {  get  {  return homeTeamCommand ??  (homeTeamCommand = new RelayCommand(obj =>  {  MessageBox.Show("Hello");  }));  }  }  private RelayCommand showMatchCommand;  public RelayCommand ShowMatchCommand  {  get  {  return showMatchCommand ??  (showMatchCommand = new RelayCommand(obj =>  {  TimedMatchWindow timedMatchWindow = new TimedMatchWindow(mainWindow, Matches[SelectedIndex]);  timedMatchWindow.ShowDialog();  }));  }  }  private int \_selectedIndex;  public int SelectedIndex  {  get => \_selectedIndex;  set  {  if (\_selectedIndex == value)  return;  \_selectedIndex = value;  OnPropertyChanged("SelectedIndex");  }  }  private RelayCommand searchCommand;  public RelayCommand SearchCommand  {  get  {  return searchCommand ??  (searchCommand = new RelayCommand(obj =>  {  if (string.IsNullOrEmpty(Owner.SearchBox.Text))  {  Matches = originalMathces;  }  else  {  List<MatchTimed> lstPr = new List<MatchTimed>();  foreach (MatchTimed pr in originalMathces)  lstPr.Add(pr);  var listItog = from p in lstPr  where p.HomeTeamName.Contains(Owner.SearchBox.Text) || p.AwayTeamName.Contains(Owner.SearchBox.Text)  select p;  ObservableCollection<MatchTimed> SearchedMathces = new ObservableCollection<MatchTimed>();  foreach (MatchTimed p in listItog)  SearchedMathces.Add(p);  Matches = SearchedMathces;  }  }));  }  }  }  } |

Листинг – MyPlayersViewModel

|  |
| --- |
| using FootballDataApi.Models;  using Goalscore.Model;  using Goalscore.Views.Frames;  using System;  using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Goalscore.ViewModels  {  public class MyPlayersViewModel : FrameViewModel<MyPlayersFrame, MainWindow>  {  protected internal ObservableCollection<Player> Players = new ObservableCollection<Player>();  public MyPlayersViewModel(MyPlayersFrame owner, MainWindow window) : base(owner, window)  {  IEnumerable<UserFavoritePlayers> userFav = DataBase.DataBaseManager.Instance.userFavoritePlayersSet.ToList().Where(u => u.UserNickName.Equals(User.current.NickName));  foreach(UserFavoritePlayers usF in userFav)  {  Players.Add(DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == usF.PlayerId).Single());  }  Owner.PlayersDataGrid.ItemsSource = Players;  }  }  } |

Листинг – RegistrationViewModel

|  |
| --- |
| using Goalscore.Commands;  using Goalscore.Model;  using Goalscore.Utilities;  using Goalscore.Views;  using Goalscore.Views.Frames;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  namespace Goalscore.ViewModels  {  internal class RegistrationViewModel : FrameViewModel<Registration, AuthorizationWindow>  {  public RegistrationViewModel(Registration owner, AuthorizationWindow window) : base(owner, window)  {  }  private RelayCommand toLoginCommand;  public RelayCommand ToLoginCommand  {  get  {  return toLoginCommand ??  (toLoginCommand = new RelayCommand(obj =>  {  OwnerWindow.AuthorizationFrame.Content = new Login(OwnerWindow);  }));  }  }  private RelayCommand registrationCommand;  public RelayCommand RegistrationCommand  {  get  {  return registrationCommand ??  (registrationCommand = new RelayCommand(obj =>  {  try  {  User user = new User();  user.NickName = Owner.LoginTextBox.Text;  if(!Owner.PasswordTextBox.Text.Equals(Owner.PasswordAgainTextBox.Text))  throw new Exception("Passwords don't match");  user.Password = PasswordHasher.GetHash(Owner.PasswordTextBox.Text);  if (DataBase.DataBaseManager.Instance.userSet.Find(user.NickName) == null)  {  DataBase.DataBaseManager.Instance.userSet.Add(user);  DataBase.DataBaseManager.Instance.SaveChanges();  User.current = user;  MainWindow window = new MainWindow();  window.Show();  OwnerWindow.Close();  }  else  {  throw new Exception("User exists");  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }    }));  }  }  }  } |

Листинг – TimedMatchContentFrameViewModel

|  |
| --- |
| using Goalscore.Views;  using Goalscore.Views.Frames;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Goalscore.ViewModels  {  public class TimedMatchContentFrameViewModel : FrameViewModel<TimedMatchContentFrame, TimedMatchWindow>  {  public TimedMatchContentFrameViewModel(TimedMatchContentFrame owner, TimedMatchWindow window, MatchTimed matchTimed) : base(owner, window)  {  Owner.RefereeTextBlock.Text = DataBase.DataBaseManager.Instance.refereeSet.ToList().Where(x => x.Id == matchTimed.RefereeId).Single().Name;  Owner.StadiumTextBlock.Text = DataBase.DataBaseManager.Instance.teamSet.ToList().Where(x => x.Id == matchTimed.HomeTeam.Id).Single().Address;  }  }  } |

Листинг – TimedMatchViewModel

|  |
| --- |
| using Goalscore.Views;  using Goalscore.Views.Frames;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Media.Imaging;  namespace Goalscore.ViewModels  {  public class TimedMatchViewModel : WindowViewModel<TimedMatchWindow>  {  public MatchTimed match;  public TimedMatchViewModel(TimedMatchWindow owner, MatchTimed match) : base(owner)  {  this.match = match;  var f = this.match.HomeTeamName;  Owner.HomeTeamImage.Source = DataBase.DataBaseManager.GetTeamImage(this.match.HomeTeam.Id);  Owner.AwayTeamImage.Source = DataBase.DataBaseManager.GetTeamImage(this.match.AwayTeam.Id);  Owner.TimeMatchTextBlock.Text = this.match.UtcDate.ToString();  Owner.ContentFrame.Content = new TimedMatchContentFrame(Owner, this.match);  }  }  } |

Листинг – UserTeamViewModel

|  |
| --- |
| **using** **FootballDataApi.Models**;  **using** **Goalscore.Commands**;  **using** **Goalscore.Model**;  **using** **Goalscore.Views.Frames**;  **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.Collections.ObjectModel**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows**;  **namespace** **Goalscore.ViewModels**  {  **public** **class** **UserTeamViewModel** : FrameViewModel<UserTeamFrame, MainWindow>  {  **protected** **internal** ObservableCollection<Player> Players = **new** ObservableCollection<Player>();  **private** UserTeam userTeam;  **public** **UserTeamViewModel**(UserTeamFrame owner, MainWindow window) : **base**(owner, window)  {  IEnumerable<UserFavoritePlayers> userFav = DataBase.DataBaseManager.Instance.userFavoritePlayersSet.ToList().Where(u => u.UserNickName.Equals(User.current.NickName));  **foreach** (UserFavoritePlayers usF **in** userFav)  {  Players.Add(DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == usF.PlayerId).Single());  }  Owner.PlayersDataGrid.ItemsSource = Players;  **if** (DataBase.DataBaseManager.Instance.userTeamSet.ToList().Any(t => t.UserNickName.Equals(User.current.NickName)))  {  userTeam = DataBase.DataBaseManager.Instance.userTeamSet.ToList().Where(t => t.UserNickName.Equals(User.current.NickName)).Single();    //GK  **if**(userTeam.GKId != **null**)  {  Owner.GK.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.GKId);  Owner.GK.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.GKId).Single().Name;  }  **else**  {  Owner.GK.ImageSource = "../../Resources/add\_player.png";  Owner.GK.Text = "No player";  }  //LB  **if** (userTeam.LBId != **null**)  {  Owner.LB.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.LBId);  Owner.LB.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.LBId).Single().Name;  }  **else**  {  Owner.LB.ImageSource = "../../Resources/add\_player.png";  Owner.LB.Text = "No player";  }  //CB1  **if** (userTeam.CB1Id != **null**)  {  Owner.CB1.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.CB1Id);  Owner.CB1.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.CB1Id).Single().Name;  }  **else**  {  Owner.CB1.ImageSource = "../../Resources/add\_player.png";  Owner.CB1.Text = "No player";  }  //CB2  **if** (userTeam.CB2Id != **null**)  {  Owner.CB2.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.CB2Id);  Owner.CB2.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.CB2Id).Single().Name;  }  **else**  {  Owner.CB2.ImageSource = "../../Resources/add\_player.png";  Owner.CB2.Text = "No player";  }  //RB  **if** (userTeam.RBId != **null**)  {  Owner.RB.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.RBId);  Owner.RB.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.RBId).Single().Name;  }  **else**  {  Owner.RB.ImageSource = "../../Resources/add\_player.png";  Owner.RB.Text = "No player";  }  //LCM  **if** (userTeam.LCMId != **null**)  {  Owner.LCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.LCMId);  Owner.LCM.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.LCMId).Single().Name;  }  **else**  {  Owner.LCM.ImageSource = "../../Resources/add\_player.png";  Owner.LCM.Text = "No player";  }  //CM  **if** (userTeam.CMId != **null**)  {  Owner.CM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.CMId);  Owner.CM.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.CMId).Single().Name;  }  **else**  {  Owner.CM.ImageSource = "../../Resources/add\_player.png";  Owner.CM.Text = "No player";  }  //RCM  **if** (userTeam.RCMId != **null**)  {  Owner.RCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.RCMId);  Owner.RCM.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.RCMId).Single().Name;  }  **else**  {  Owner.RCM.ImageSource = "../../Resources/add\_player.png";  Owner.RCM.Text = "No player";  }  //LW  **if** (userTeam.LWId != **null**)  {  Owner.LW.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.LWId);  Owner.LW.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.LWId).Single().Name;  }  **else**  {  Owner.LW.ImageSource = "../../Resources/add\_player.png";  Owner.LW.Text = "No player";  }  //ST  **if** (userTeam.STId != **null**)  {  Owner.ST.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.STId);  Owner.ST.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.STId).Single().Name;  }  **else**  {  Owner.ST.ImageSource = "../../Resources/add\_player.png";  Owner.ST.Text = "No player";  }  //RW  **if** (userTeam.RWId != **null**)  {  Owner.RW.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.RWId);  Owner.RW.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.RWId).Single().Name;  }  **else**  {  Owner.RW.ImageSource = "../../Resources/add\_player.png";  Owner.RW.Text = "No player";  }  }  **else**  {  userTeam = **new** UserTeam();  userTeam.UserNickName = User.current.NickName;  DataBase.DataBaseManager.Instance.userTeamSet.Add(userTeam);  DataBase.DataBaseManager.Instance.SaveChanges();  }  }  **private** RelayCommand setGK;  **public** RelayCommand SetGK  {  **get**  {  **return** setGK ??  (setGK = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.GKId = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.GK.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.GKId);  Owner.GK.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.GKId).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setLB;  **public** RelayCommand SetLB  {  **get**  {  **return** setLB ??  (setLB = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.LBId = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.LB.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.LBId);  Owner.LB.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.LBId).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setCB1;  **public** RelayCommand SetCB1  {  **get**  {  **return** setCB1 ??  (setCB1 = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.CB1Id = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.CB1.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.CB1Id);  Owner.CB1.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.CB1Id).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setCB2;  **public** RelayCommand SetCB2  {  **get**  {  **return** setCB2 ??  (setCB2 = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.CB2Id = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.CB2.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.CB2Id);  Owner.CB2.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.CB2Id).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setRB;  **public** RelayCommand SetRB  {  **get**  {  **return** setRB ??  (setRB = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.RBId = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.RB.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.RBId);  Owner.RB.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.RBId).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setLCM;  **public** RelayCommand SetLCM  {  **get**  {  **return** setLCM ??  (setLCM = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.LCMId = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.LCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.LCMId);  Owner.LCM.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.LCMId).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setCM;  **public** RelayCommand SetCM  {  **get**  {  **return** setCM ??  (setCM = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.CMId = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.CM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.CMId);  Owner.CM.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.CMId).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setRCM;  **public** RelayCommand SetRCM  {  **get**  {  **return** setRCM ??  (setRCM = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.RCMId = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.RCM.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.RCMId);  Owner.RCM.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.RCMId).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setLW;  **public** RelayCommand SetLW  {  **get**  {  **return** setLW ??  (setLW = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.LWId = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.LW.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.LWId);  Owner.LW.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.LWId).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setST;  **public** RelayCommand SetST  {  **get**  {  **return** setST ??  (setST = **new** RelayCommand(obj =>  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.STId = player.Id;  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.ST.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.STId);  Owner.ST.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.STId).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }));  }  }  **private** RelayCommand setRW;  **public** RelayCommand SetRW  {  **get**  {  **return** setRW ??  (setRW = **new** RelayCommand(obj =>  {  **try**  {  **if** (Owner.PlayersDataGrid.SelectedItem != **null**)  {  Player player = (Player)Owner.PlayersDataGrid.SelectedItem;  userTeam.RWId = player.Id;  **if** (userTeam.RBId == player.Id)  **throw** **new** **Exception**("This player is already in the team");  DataBase.DataBaseManager.Instance.SaveChanges();  Owner.RW.ImageSource = DataBase.DataBaseManager.GetPlayerImageSource(userTeam.RWId);  Owner.RW.Text = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.RWId).Single().Name;  }  **else**  {  MessageBox.Show("No selected player");  }  }  **catch**(Exception e)  {  MessageBox.Show(e.Message);  }  }));  }  }  **private** **void** **CalculateTeamPower**()  {  **if** (userTeam.GKId == **null** ||  userTeam.LBId == **null** ||  userTeam.CB1Id == **null** ||  userTeam.CB2Id == **null** ||  userTeam.RBId == **null** ||  userTeam.LCMId == **null** ||  userTeam.CMId == **null** ||  userTeam.RCMId == **null** ||  userTeam.LWId == **null** ||  userTeam.STId == **null** ||  userTeam.RWId == **null**)  **return**;  Player GK = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.GKId).Single();  Player LB = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.LBId).Single();  Player CB1 = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.CB1Id).Single();  Player CB2 = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.CB2Id).Single();  Player RB = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.RBId).Single();  Player LCM = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.LCMId).Single();  Player CM = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.CMId).Single();  Player RCM = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.RCMId).Single();  Player LW = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.LWId).Single();  Player ST = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.STId).Single();  Player RW = DataBase.DataBaseManager.Instance.playerSet.ToList().Where(p => p.Id == userTeam.RWId).Single();  **short** NationalPower = **0**;  **short** PlayersInOneTeamPower = **0**;  }  }  } |

Листинг – WindowViewModel

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Media.Animation;  using System.Windows;  namespace Goalscore.ViewModels  {  public abstract class WindowViewModel<T> : BaseViewModel  where T : Window  {  public T Owner { get; private set; }  public WindowViewModel(T owner)  {  Owner = owner;  }  public void AnimatedClose()  {  DoubleAnimation anim = new DoubleAnimation();  anim.From = Owner.ActualHeight; anim.To = 0;  anim.Duration = TimeSpan.FromSeconds(0.5);  anim.EasingFunction = new QuadraticEase() { EasingMode = EasingMode.EaseOut };  anim.Completed += (object sender, EventArgs e) => { Owner.Close(); };  Owner.BeginAnimation(FrameworkElement.HeightProperty, anim);  }  }  } |