## Содержание

## Введение

В современном мире регулирование широко используемых в профессиональном спорте трансферов (переходов) спортсменов — это тема, к которой спортивная общественность, СМИ и государство проявляют обоснованный интерес. На протяжении последнего столетия регламентными нормами неправительственных спортивных организаций сформирован особый механизм смены спортсменами своих работодателей. Ясно, что эти особенности наиболее ярко проявляются в самых популярных и массовых командных видах спорта — футболе, хоккее, баскетболе и т. д. Очевидно, что организациям необходимы удобные приложения для проведения сделок по покупке и продаже игроков.

Целью данной работы является реализация простого в использовании и многофункционального приложения для получения информации об игроках с возможностью их покупки и продажи.

Для достижения поставленной цели необходимо решить следующие задачи:

1) формализовать задание, выделив соответствующих пользователей и их функционал;
2) провести анализ СУБД и выбрать наиболее подходящую;
3) спроектировать базу данных;
4) спроектировать архитектуру приложения;
5) разработать приложение.

# 1 Аналитический раздел

В данном разделе будет поставлена задача, рассмотрены возможные пользователи системы, модели данных и СУБД.

## 1.1 Постановка задачи

Необходимо разработать программу, предоставляющую интерфейс для получения информации об игроках по заданным параметрами с возможностью их покупки и продажи.

## 1.2 Пользователи системы

### 1.2.1 Гость

Гость — это неавторизованный пользователь, обладающий только одной возможностью взаимодействия с системой - авторизация.

### 1.2.2 Аналитик

Аналитик - это авторизированный пользователь, закрепленный за определенной командой, обладающий базовыми возможностями взаимодействия с системой - просмотр информации об игроках, их статистике за прошедший сезон и командах. Также дополнительными возможностями аналитика являются просмотр желаемых игроков для своей команды, добавление игрока в список желаемых игроков команды и удаление желаемого игрока из этого списка.

### 1.2.3 Менеджер

Менеджер - это авторизованный пользователь, закрпеленный за определенной, обладающий базовыми возможностями взаимодействия с системой - просмотр информации об игроках, их статистике за прошедший сезон и командах. Также дополнительными возможностями менеджера являются просмотр желаемых игроков для своей команды, удаление желаемого игрока из списка желаемых игроков. Менеджер может запрашивать игрока у другой команды, подтверждать и отклонять доступные сделки.

### 1.2.4 Модератор

Модератор является авторизированным пользователем с повышенным уровнем полномочий — он обладает всеми базовыми правами пользователя, а также может проводить и удалять сделки, изменять баланс команды и команду игрока.

## 1.3 Анализ моделей баз данных

Модель базы данных - это тип модели данных, которая определяет логическую структуру базы данных и в корне определяет, каким образом данные могут храниться, организовываться и обрабатываться.

### 1.3.1 Иерархическая база данных

Иерархическая модель базы данных подразумевает, что элементы организованы в структуры, связанные между собой иерархическими или древо-

видными связями. Родительский элемент может иметь несколько дочерних элементов. Но у дочернего элемента может быть только один предок.

Иерархические базы данных графически могут быть представлены как перевернутое дерево, состоящее из объектов различных уровней. Верхний уровень (корень дерева) занимает один объект, второй - объекты второго уровня и так далее.

Такая модель подразумевает возможность существования одинаковых (преимущественно дочерних) элементов. Данные здесь хранятся в серии записей с прикреплёнными к ним полями значений. Модель собирает вместе все экземпляры определённой записи в виде «типов записей» — они эквивалентны таблицам в реляционной модели, а отдельные записи — столбцам таблицы.

### 1.3.2 Сетевая модель базы данных

Сетевая модель базы данных подразумевает, что у родительского элемента может быть несколько потомков, а у дочернего элемента — несколько предков. Записи в такой модели связаны списками с указателями.

Сетевая модель состоит из множества записей, которые могут быть владельцами или членами групповых отношений. Связь между записью-владельцем и записью-членом также имеет вид 1:N.

Сетевая модель позволяет иметь несколько предков и потомков, формирующих решётчатую структуру.

На рисунке 1 представлен пример сетевой модели базы данных.

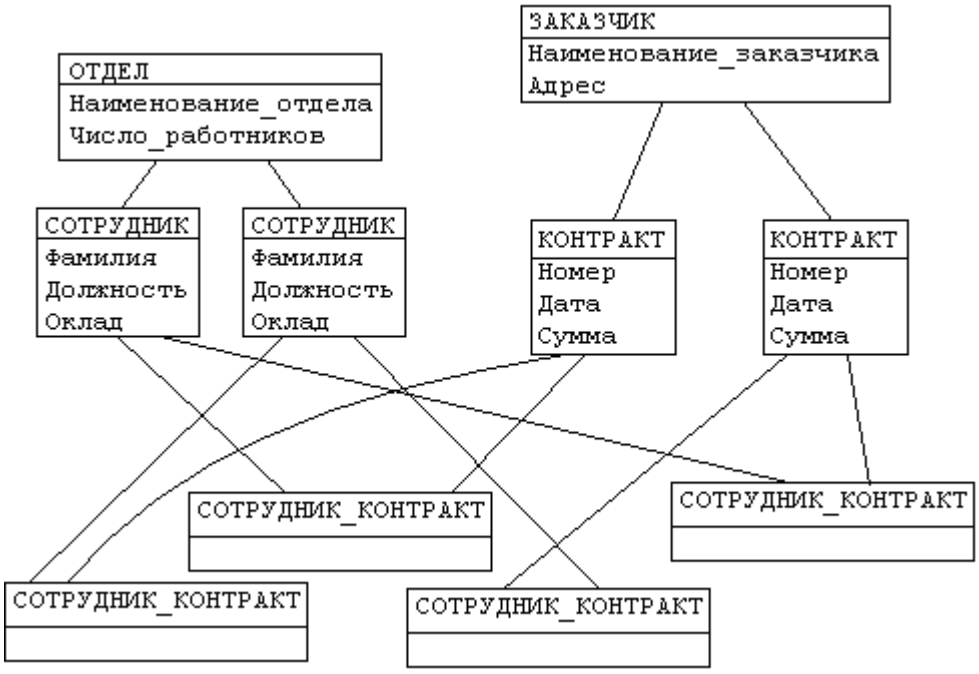


**Рисунок 1** – Иерархическая модель базы данных

### 1.3.3 Реляционная модель базы данных

В реляционной модели, в отличие от иерархической или сетевой, не существует физических отношений. Вся информация хранится в виде таблиц (отношений), состоящих из рядов и столбцов. А данные двух таблиц связаны общими столбцами, а не физическими ссылками или указателями. Для манипуляций с рядами данных существуют специальные операторы.

Реляционные таблицы обладают следующими свойствами:

1) все значения атомарны;

2) каждый ряд уникален;

3) порядок столбцов не важен;

4) порядок рядов не важен;

5) у каждого столбца есть своё уникальное имя.

Термин «реляционный» означает, что теория основана на математическом понятии отношение (relation). В качестве неформального синонима термину «отношение» часто встречается слово таблица. Необходимо помнить, что «таблица» есть понятие нестрогое и неформальное и часто означает не «отношение» как абстрактное понятие, а визуальное представление отношения на бумаге или экране. Некорректное и нестрогое использование термина «таблица» вместо термина «отношение» нередко приводит к недопониманию.

### 1.3.4 Выбор модели данных

В качестве модели данных была выбрана реляционная модель, так как структура данных однозначно определена, структура данных не является быстроизменяющейся и данные подчиняются строгим правилам и ограничениям.

## 1.4 Вывод

В данном разделе была поставлена задача, рассмотрены возможные пользователи системы, проанализированы модели данных и СУБД.

# 2 Конструкторский раздел

В данном разделе будут рассотрены сценарии пользователей, спроектирована база данных, описана ролевая модель и спроектировано приложение.

## 2.1 Сценарии пользователей

Необходимо определить функционал каждого из пользователей.

### 2.1.1 Гость

Гость - это неавторизованный пользователь, у которого есть только одна возможность - авторизоваться. На рисунке 2 представлена Use-Case-диаграмма для гостя.

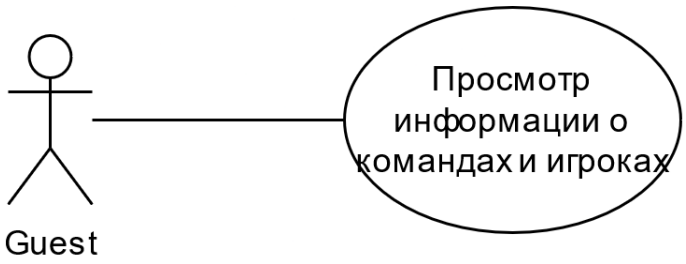


**Рисунок 2** – Сценарии для гостя.

### 2.1.2 Любой авторизированный пользователь

Функциональность, предоставляемая каждому авторизированного пользователя (аналитику, менеджеру, модератору):

1) просмотр всех игроков;

2) просмотр статистики игроков;

3) получение игроков выбранной команды;

4) поиск игрока по ID;

5) поиск игрока по имени;

6) просмотр всех команд;

7) поиск команды по ID;

8) поиск команды по имени;

9) получение статистики выбранного игрока.

### 2.1.3 Аналитик

Дополнительная функциональность, предоставляемая пользователю аналитик:

1) добавить желаемого игрока;

2) удалить желаемого игрока;

3) просмотр всех желаемых игроков.

На рисунке 3 представлена Use-Case-диаграмма для аналитика.

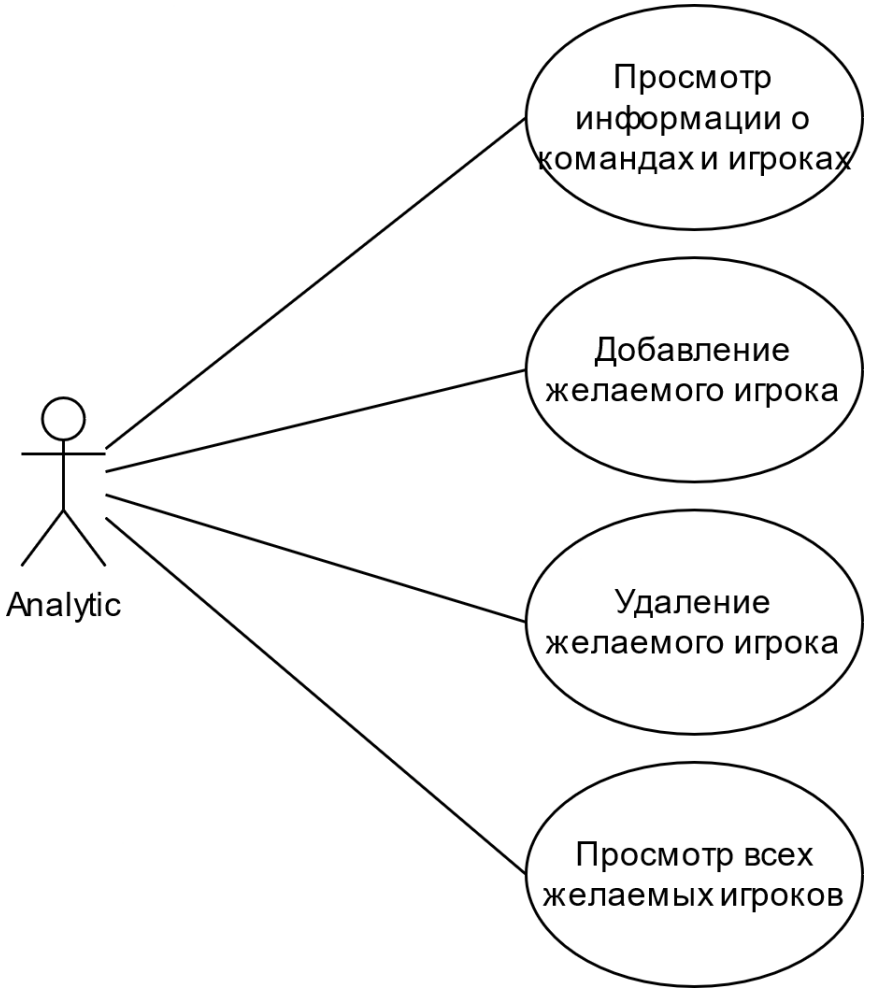


**Рисунок 3** – Сценарии для аналитика.

### 2.1.4 Менеджер

Дополнительная функциональность, предоставляемая пользователю менеджер:

1) просмотр всех желаемых игроков;

2) удалить желаемого игрока;

3) запросить желаемого игрока с указанием предлагаемой цены;

4) подтвердить сделку;

5) отклонить сделку;

6) получить входящие предложения;

7) получить исходящие предложения.

На рисунке 4 представлена Use-Case-диаграмма для менеджера.

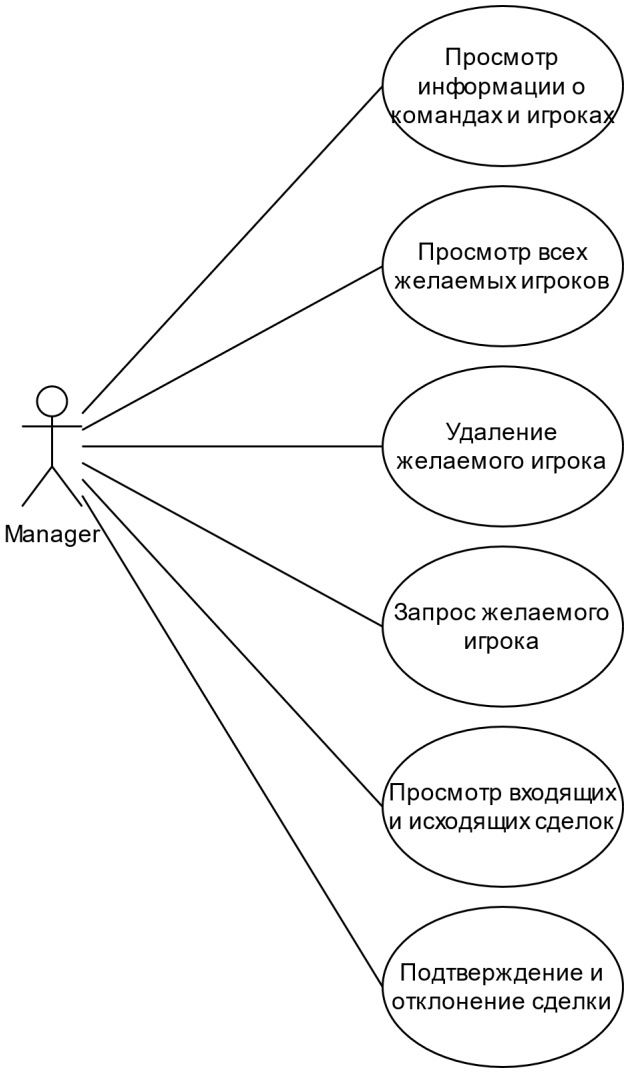


**Рисунок 4** – Сценарии для менеджера.

### 2.1.5 Модератор

Дополнительная функциональность, предоставляемая пользователю модератор:

1) провести сделку;

2) удалить сделку;

3) изменить команду игрока;

4) изменить баланс команды;

5) просмотр всех пользователей;

6) добавление нового пользователя;

7) удаление пользователя.

На рисунке 5 представлена Use-Case-диаграмма для модератора.

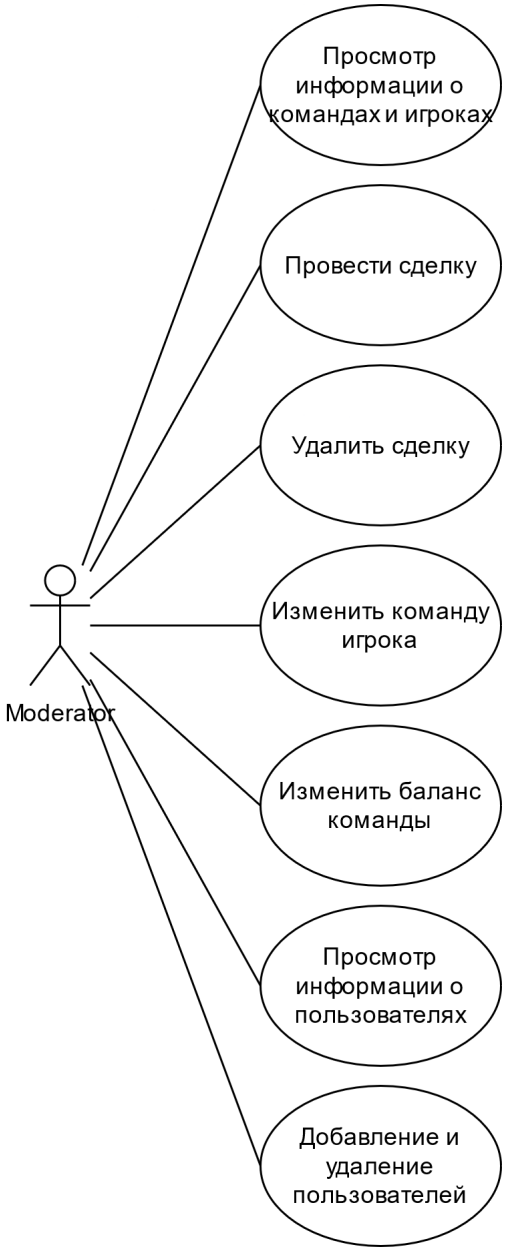


**Рисунок 5** – Сценарии для модератора.

## 2.2 Проектирование базы данных

### 2.2.1 Формализация сущностей системы

Необходимо выделить сущности предметной области и построить ER-диаграмму. На рисунке 6 представлена ER-диаграмма системы.

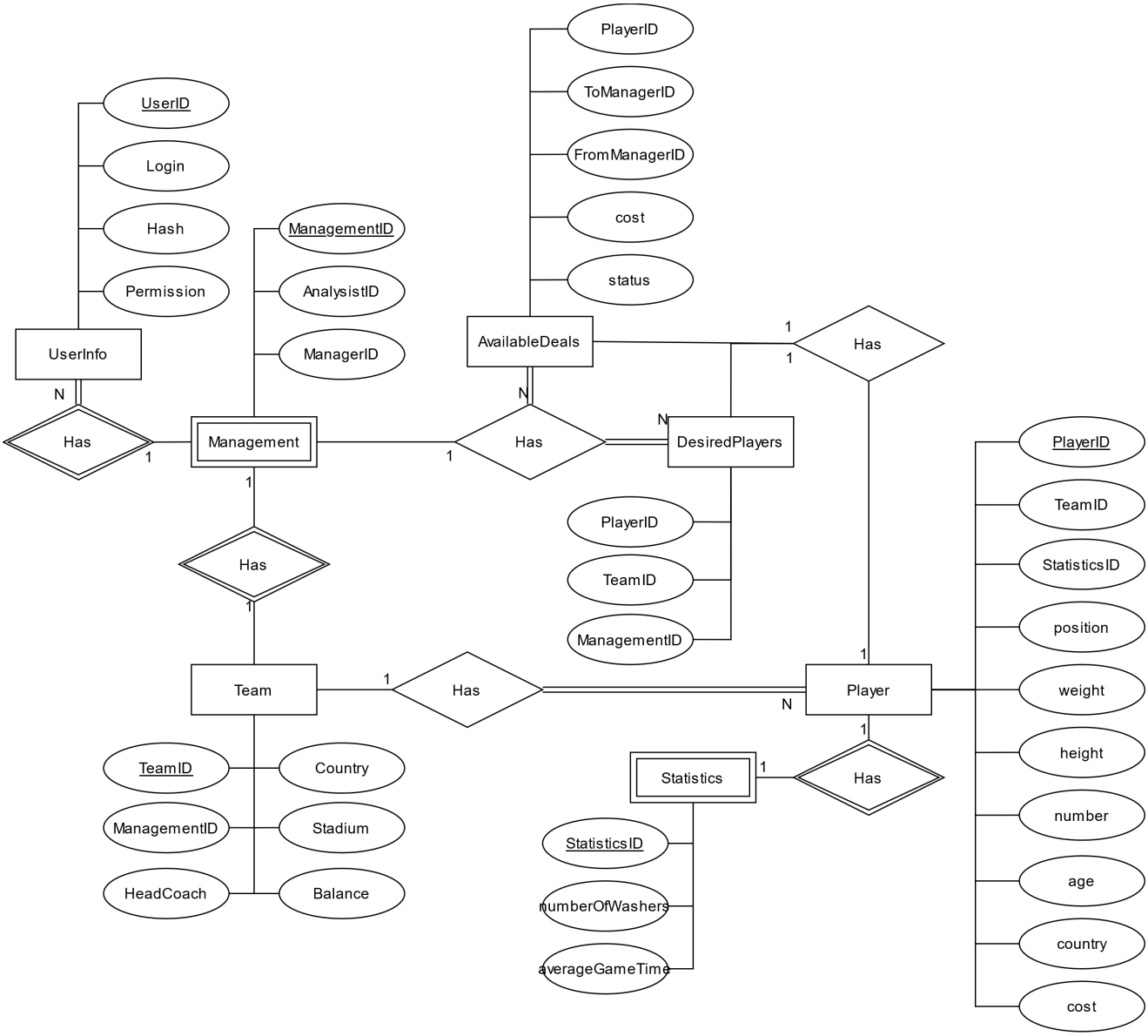


**Рисунок 6** – ER-диаграмма системы.

Выделенные сущности:

1) UserInfo - таблица, в которой хранится информация о пользователях;

2) Management - таблица, в которой хранятся ID аналитиков и менеджеров, которые закреплены за определенной командой;

3) Team - таблица, в которой хранится информация о командах;

4) Player - таблица, в которой хранится информация об игроках;

5) AvailableDeals - таблица, в которой хранится информация о доступных сделках определенного менеджмента;

6) DesiredPlayers - таблица, в которой хранится информация о желаемых игроках определенного менеджмента;

7) Statistics - таблица, в которой хранится информация о статистике определенного игрока.

### 2.2.2 Ролевая модель

На уровне базы данных выделена следующая ролевая модель:

1) guest - гость;

2) analytic - аналитик;

3) manager - менеджер;

4) moderator - модератор.

Использование ролевой модели на уровне базы данных гарантирует безопасность доступа к объектам базы данных.

**Гость**

Пользователь с ролью guest имеет следующие права: SELECT над таблицей UserInfo.

**Аналитик**

Пользователь с ролью analytic имеет следующие права:

1) SELECT над таблицей Player;

2) SELECT над таблицей Management;

3) SELECT над таблицей Team;

4) SELECT над таблицей Statistics;

5) SELECT, UPDATE, DELETE, INSERT над таблицей DesiredPlayers.

**Менеджер**

Пользователь с ролью manager имеет следующие права:

1) SELECT над таблицей Player;

2) SELECT над таблицей Management;

3) SELECT над таблицей Team;

4) SELECT над таблицей Statistics;

5) SELECT, DELETE над таблицей DesiredPlayers.

6) SELECT, DELETE, INSERT над таблицей AvailableDeals.

**Модератор**

Пользователь с ролью moderator обладает всеми правами.

### 2.2.3 Функции

Для того, чтобы пользователь мог одновременно просматривать информацию об игроке, его команде и его статистику, необходимо добавить функ-

цию, которая возвращает эту информацию. На рисунке 7 представлен алгоритм функции.


**Рисунок 7** – Алгоритм функции.

## 2.3 Проектирование приложения

Приложение для работы с базой данных представляет собой Desktop-приложение. Оно спроектировано по паттерну MVC. Выделены два компонента - доступа к данным и бизнес-логики. Компонент доступа к данным спроектирован по паттерну "Репозиторий". Для контроля ролей при авторизации пользователя создан класс Connection, в котором обрабатывается конфигурационный файл и возвращается строка подключения.

## 2.4 Вывод

В данном разделе были рассмотрены сценарии пользователей, спроектирована база данных, описана ролевая модель и спроектировано приложение.

# 3 Технологический раздел

В данном разделе будут выбраны средства реализации поставленной задачи, создана база данных и ролевая модель, разработаны компоненты и описан порядок работы.

## 3.1 Анализ СУБД

СУБД — комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД. Самыми популярными СУБД являются MySQL, Microsoft SQL Server, PostgreSQL и Oracle.

### 3.1.1 MySQL

MySQL — реляционная СУБД с открытым исходным кодом, главными плюсами которой являются ее скорость и гибкость, которая обеспечена поддержкой большого количества различных типов таблиц.

Кроме того, это надежная бесплатная система с простым интерфейсом и возможностью синхронизации с другими базами данных. В совокупности эти факторы позволяют использовать MySQL как крупным корпорациям, так и небольшим компаниям.

Преимущества:

1) простота в использовании;

2) обширный функционал;

3) безопасность;

4) масштабируемость;

5) скорость.

Недостатки: недостаточная надежность;

### 3.1.2 Microsoft SQL Server

Система позволяет синхронизироваться с другими программными продуктами компании Microsoft, а также обеспечивает надежную защиту данных и простой интерфейс, однако отличается высокой стоимостью лицензии и повышенным потреблением ресурсов.

Преимущества:

1) СУБД масштабируется;

2) простота в использовании;

3) возможность интеграции с другими продуктами Microsoft.

Недостатки:

1) высокая стоимость продукта для юридических лиц;

2) возможны проблемы в работе служб интеграции импорта файлов;

3) высокая ресурсоемкость SQL Server.

### 3.1.3 PostgreSQL

PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многочисленные возможности.

Преимущества:

1) бесплатное ПО с открытым исходным кодом;

2) большое количество дополнений;

3) расширения;

Недостатки:

1) производительность;

2) популярность;

### 3.1.4 Oracle

Oracle – это объектно-реляционная система управления базами данных.

Преимущества:

1) поддержка огромных баз данных;

2) быстрая обработка транзакций;

3) большой и постоянно развивающийся функционал.

Недостатки:

1) высокая стоимость;

2) значительные вычислительные ресурсы.

| | Oracle | MySQL | Microsoft SQL Server | PostgreSQL |
|---|---|---|---|---|
| Простота в использовании | + | + | + | + |
| Бесплатная | - | + | + | + |
| Безопасность данных | + | - | + | + |
| Поддержка стандарта SQL | + | + | + | + |
| Поддержка хранимых процедур и триггеров | + | + | + | + |
| Кроссплатформенность | + | + | - | + |

**Рисунок 8** – Анализ СУБД.

## 3.2 Средства реализации поставленной задачи

В качестве языка программирования был выбран язык C#[1], так как:

1) этот язык имеет удобные пакеты для работы с PostgreSQL;

2) данный язык программирования объектно-ориентирован, что позволит использовать наследование, интерфейсы, абстракции и т.д.

В качестве среды разработки была выбрана "Microsoft Visual Studio 2019"[2], поскольку:

1) она имеет множество удобств для написания и отладки кода;

2) является бесплатной для студентов;

3) обеспечивает работу с Windows Forms[3].

Для данной задачи PostgreSQL[4] выигрывает по многим параметрам и поэтому было решено использовать именно эту СУБД.

Для работы с базой данных был выбран Entity Framework[5], так как EF Core поддерживает запросы LINQ, отслеживание изменений, обновления и миграции схемы и работает с многими базами данных, включая PostgreSQL.

### 3.3 Создание базы данных

Необходимо построить диаграмму БД по выделенным сущностям. На рисунке 9 представлена диаграмма БД.

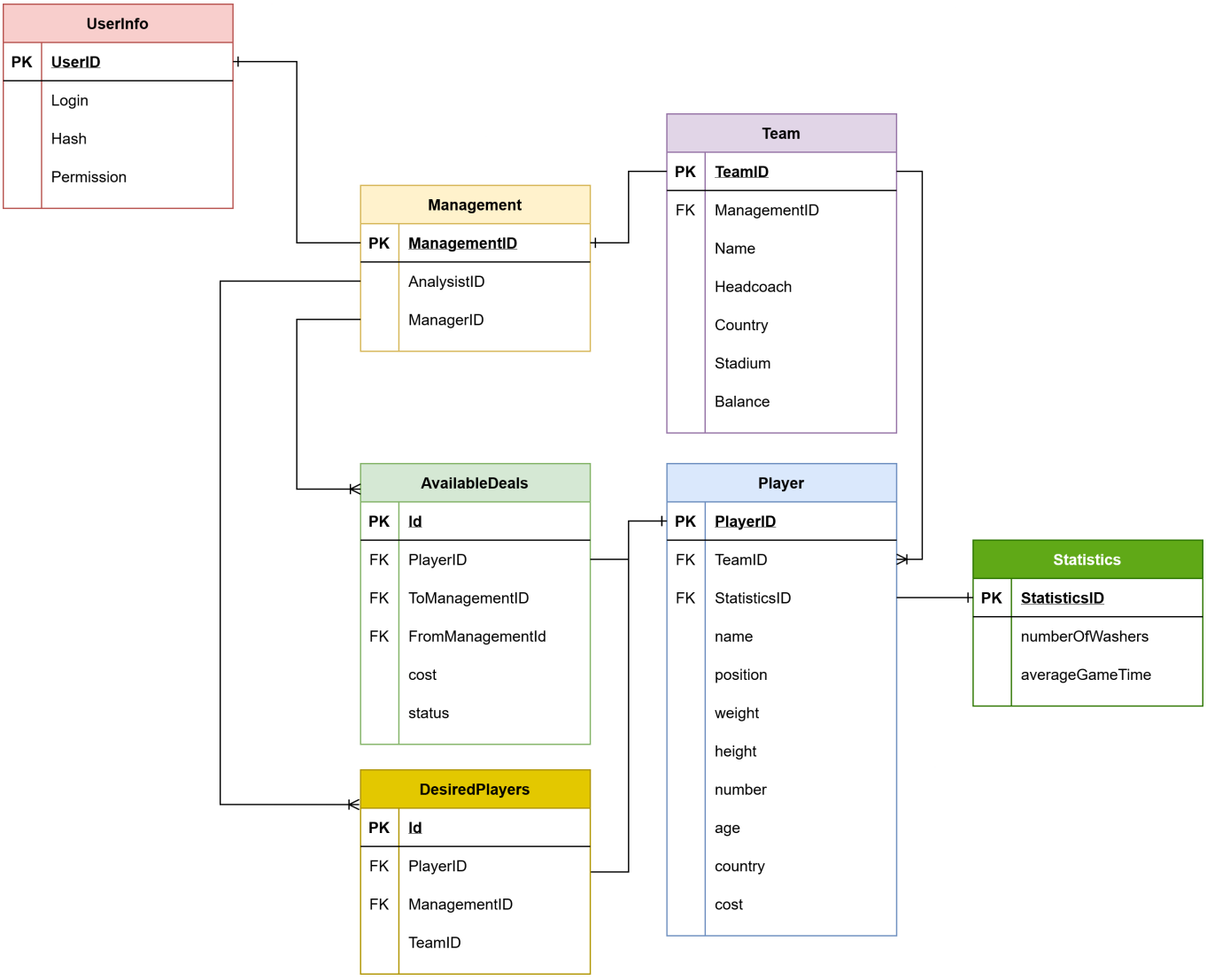


**Рисунок 9** – Диаграмма БД.

## 3.4 Создание таблиц

Далее представлены листинги создания таблиц с указанием типов данных каждого столбца.

### Таблица UserInfo.

**Листинг 1** – Создание таблицы UserInfo.

```
0 create table if not exists UserInfo (
1     id int not null primary key,
```

```
2     login varchar(30) not null,
3     hash varchar(30) not null,
4     permission int not null
5 );
```

**Таблица Management.**

**Листинг 2** – Создание таблицы Management.

```
0 create table if not exists Management (
1     ManagementID int not null primary key,
2     AnalysistID int references UserInfo(id),
3     ManagerID int references UserInfo(id)
4 );
```

**Таблица Team.**

**Листинг 3** – Создание таблицы Management.

```
0 create table if not exists Team (
1     TeamID int not null primary key,
2     ManagementID int references Management(ManagementID),
3     Name varchar(30) not null,
4     HeadCoach varchar(30) not null,
5     Country varchar(30) not null,
6     Stadium varchar(60) not null,
7     Balance int not null
8 );
```

**Таблица Player**.

**Листинг 4** – Создание таблицы Player.

```
0 create table if not exists Player (
1     PlayerID int not null primary key,
2     TeamID int references Team(TeamID),
3     Statistics int references Statistics(StatisticsID),
4     name varchar(30) not null,
```

```
    position varchar(10) not null,
    weight int not null,
    height int not null,
    number int not null,
    age int not null,
    country varchar(50) not null,
    cost int not null
);
```

**Таблица AvailableDeals.**

**Листинг 5** – Создание таблицы AvailableDeals.

```
create table if not exists AvailableDeals (
    Id int primary key not null,
    PlayerID int references Player(PlayerID),
    ToManagementID int references Management(ManagementID),
    FromManagementID int references Management(ManagementID),
    cost int not null,
    status int not null
);
```

**Таблица DesiredPlayers.**

**Листинг 6** – Создание таблицы AvailableDeals.

```
create table if not exists DesiredPlayers (
    Id int primary key not null,
    PlayerID int references Player(PlayerID),
    TeamID int references Team(TeamID),
    ManagementID int references Management(ManagementID)
);
```

**Таблица Statistics.**

**Листинг 7** – Создание таблицы AvailableDeals.

```
create table if not exists Statistics (
```

```
    StatisticsID int not null primary key,
    numberOfWashers int not null,
    averageGameTime int not null
);
```

## 3.5 Создание пользователей системы

Для того, чтобы обеспечить безопасность доступа к данных необходимо создать пользователей с соответствующими правами.

### Гость

**Листинг 8** – Создание пользователя guest.

```
create role guest with login password '1234';
grant select on table userinfo to guest;
```

### Аналитик

**Листинг 9** – Создание пользователя analytic.

```
create role analytic with login password '1234';
grant select on table player, team, management, statistics to analytic;
grant all privileges on table desiredplayers to analytic;
```

### Менеджер

**Листинг 10** – Создание пользователя manager.

```
create role manager with login password '1234';
grant select on table player, team, management, statistics to manager;
grant select, delete on table desiredplayers to manager;
grant select, delete, insert on table availabledeals to manager;
```

### Moderator

**Листинг 11** – Создание пользователя moderator.

```
0 create role moderator with login password '1234';
1 grant all privileges on table player, team, management, statistics,
     availabledeals, desiredplayer, userinfo to moderator;
```

## 3.6 Функции

В листинге 12 представлена реализация функции GetPlayers.

**Листинг 12** – Реализация функции GetPlayers.

```
0  drop function if exists GetPlayers;
1  create function GetPlayers()
2  returns table
3  (
4      PlayerID int,
5      Player varchar(40),
6      Team varchar(40),
7      Washers int,
8      gametime int
9  )
10 language sql
11 as $$
12     select PlayerTeam.playerid as PlayerID, PlayerTeam.player as Player,
13         PlayerTeam.Team as Team, statistics.numberOfWashers as Washers,
14         statistics.averageGameTime as gametime
15     from (
16         select PlayerID as playerid, player.Name as player,
17             player.Statistics as PlayerStat, team.Name as Team
18         from player join team on player.TeamID = team.TeamID
19     ) as PlayerTeam
20     join statistics on PlayerTeam.PlayerStat = statistics.StatisticsID;
21 $$;
```

Данная функция нужна для того, чтобы получить сразу информацию из

нескольких таблиц по игроку. Сначала соединяются таблицы Player и Team по ключу TeamID. Из нее выбираются поля PlayerID, player.Name, player.Statistics, team.Name. В полученной таблице они хранятся под названиями playerid, player, PlayerStat, Team. Ее название - PlayerTeam. Далее полученная таблица соединяется с Statistics по StatisticsID и PlayerStat и из нее выбираются PlayerID, Player, Team, numberOfWashers (Washers), averageGameTime (gametime).

Типы выходных параметров:

1) PlayerID - int, так как хранит идентификатор игрока;
2) Player - varchar(40), так как хранит имя игрока;
3) Team - varchar(40), так как хранит название команды;
4) Washers - int, так как хранит количество забитых шайб;
5) gametime - int, так как хранит среднее время игры.

### 3.7 Добавление функции в контекст базы данных

Так как в программе используется ORM EntityFramework Core, необходимо создать класс, содержащий поля выходных параметров функции.

**Листинг 13** – Класс PlayerTeamStat.

```
0 [Keyless]
1 public class PlayersTeamStat
2 {
3     public int playerid { get; set; }
4     public string player { get; set; }
```

```
5       public string team { get; set; }
6       public int washers { get; set; }
7       public int gametime { get; set; }
8   }
```

Keyless обозначает, что данная сущность без ключа.

Далее необходимо создать в классе контекста метод, который проецируется на хранимую функцию и через который можно вызывать данную функцию.

**Листинг 14** – Метод getplayers.

```
0 public IQueryable<PlayersTeamStat> getplayers() => FromExpression(() =>
      getplayers());
```

Здесь добавлен метод getplayers(), который соответствует хранимой функции в БД. Он не принимает никакие параметры и возвращает объект IQueryable<Play
Этот метод с помощью встроенного в классе DbContext метода FromExpression вызывает getplayers().

Далее в переопрепределенном методе OnModelCreating() класса контекста необходимо зарегестрировать метод getplayers с помощью вызова HasDbFunction():

**Листинг 15** – Регистрация метода getplayers.

```
0 modelBuilder.HasDbFunction(() => getplayers());
```

## 3.8 Разработка компонентов

Так как приложение спроектировано по паттерну MVC, необходимо реализовать компоненты: доступа к данным и бизнес-логики.

### 3.8.1 Компонент доступа к данным

На рисунке 10 представлена UML-диаграмма компонента доступа к данным.

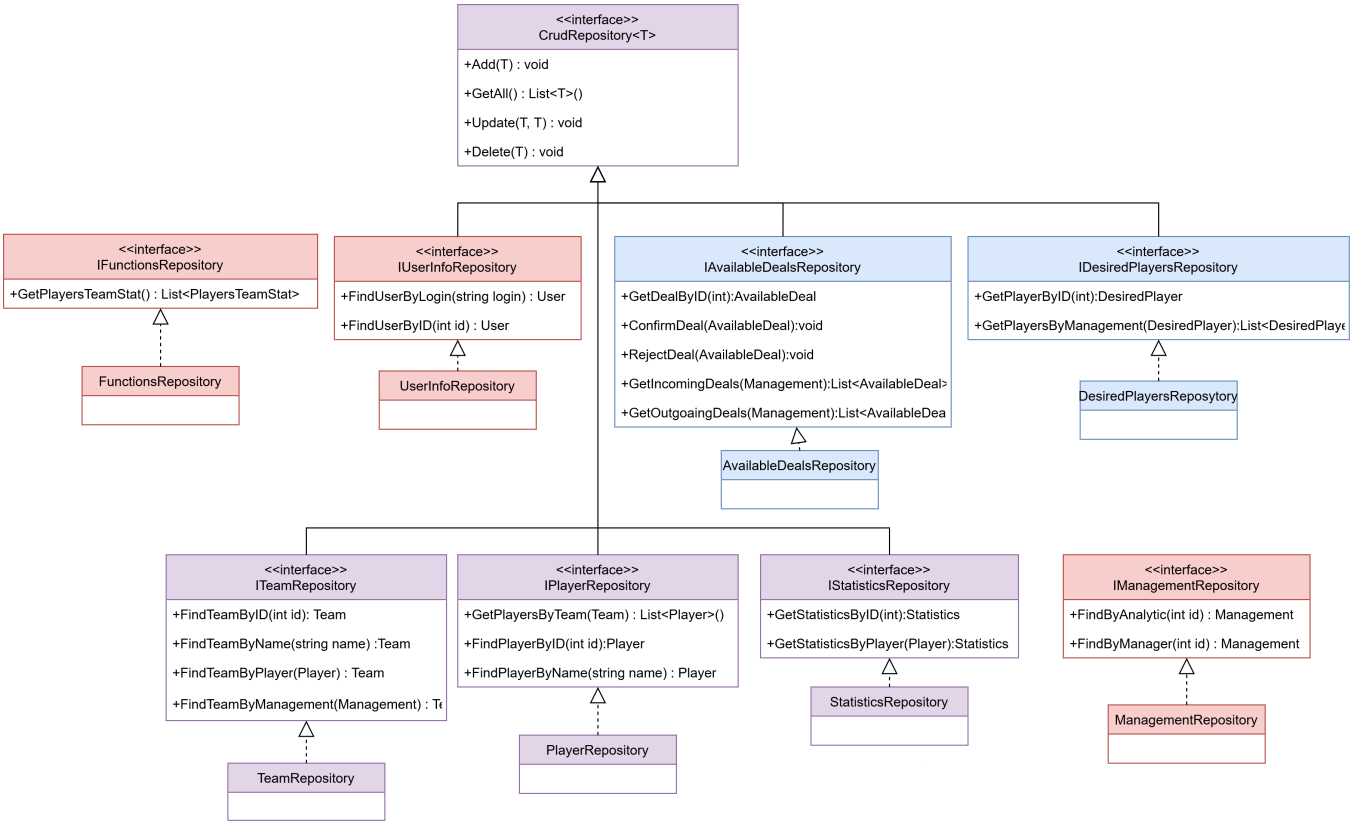


**Рисунок 10** – Компонент доступа к данным.

### 3.8.2 Компонент бизнес-логики

На рисунке 11 представлена UML-диаграмма компонента бизнес-логики.


**Рисунок 11** – Компонент бизнес-логики.

## 3.9 Интерфейс приложения

На рисунках ниже показан интерфейс авторизации и интерфейсы пользователей (аналитик, менеджер, модератор).

На рисунке 12 показана форма авторизации. Для входа в систему пользователю необходимо ввести логин и пароль.

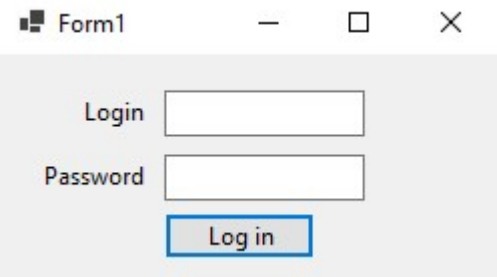


**Рисунок 12** – Окно авторизации.

На рисунке 13 показан основной интерфейс пользователя. Всего 4 группы: команды доступные любому пользователю и команды аналитика, менеджера и модератора. Также выведена информация о текущем пользователе (логин, пароль, права доступа и идентификатор).

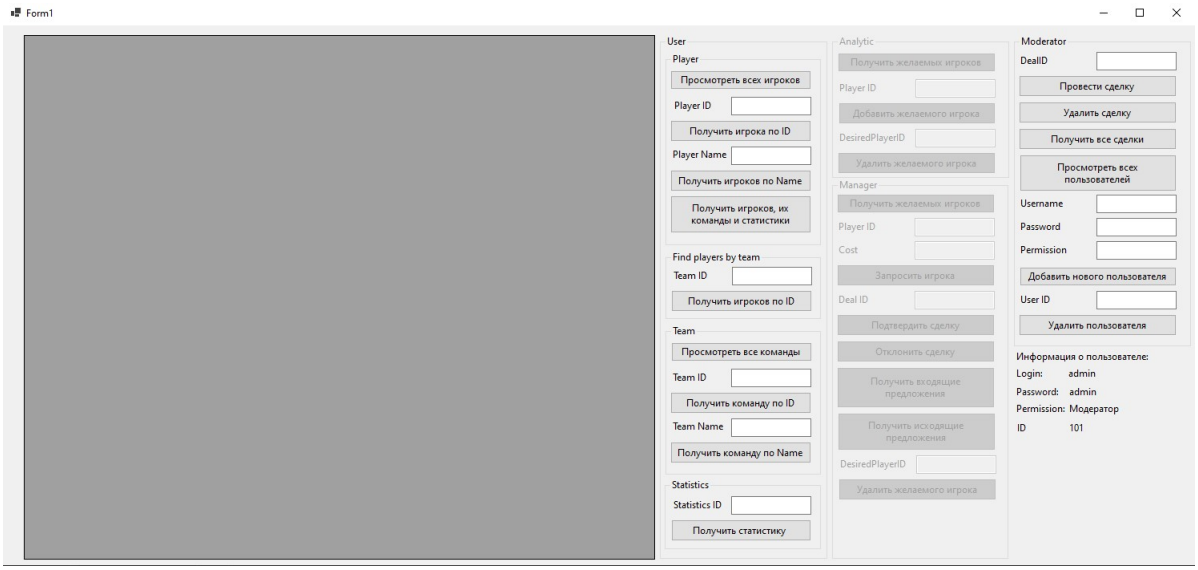


**Рисунок 13** – Основное окно.

На рисунке 14 показаны команды, доступные аналитику. Любой аналитик может просмотреть желаемых игроков, а также удалить и добавить нового. Функции менеджера и модератора для него недоступны.

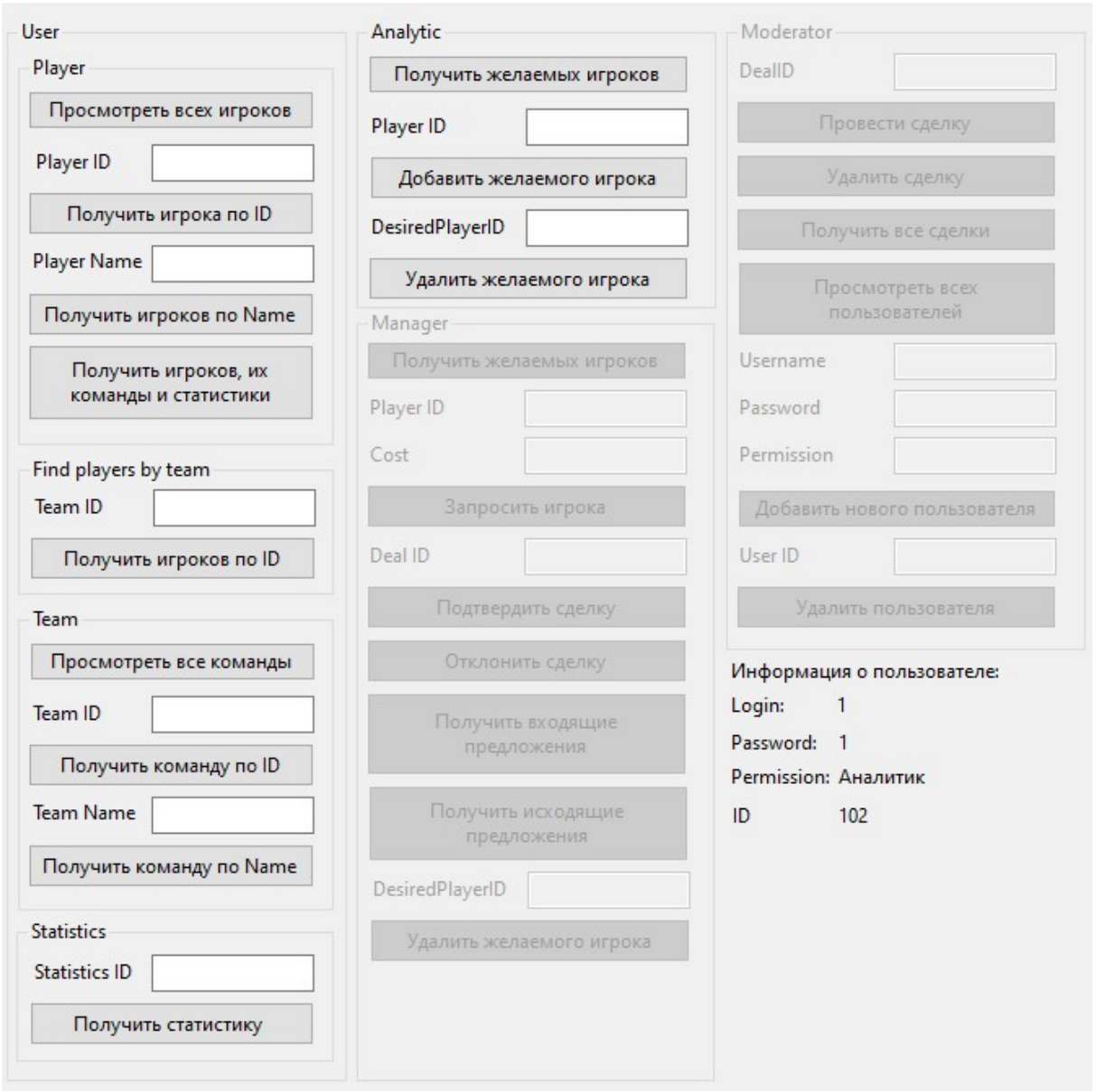


**Рисунок 14** – Операции, доступные аналитику.

На рисунке 15 показаны команды, доступные менеджеру. Любой менеджер может просмотреть желаемых игроков, а также удалить ненужного, также может запрашивать игрока на покупку, проводить и отклонять сделки и получить входящие и исходящие предложения. Функции аналитика и модератора для него недоступны.

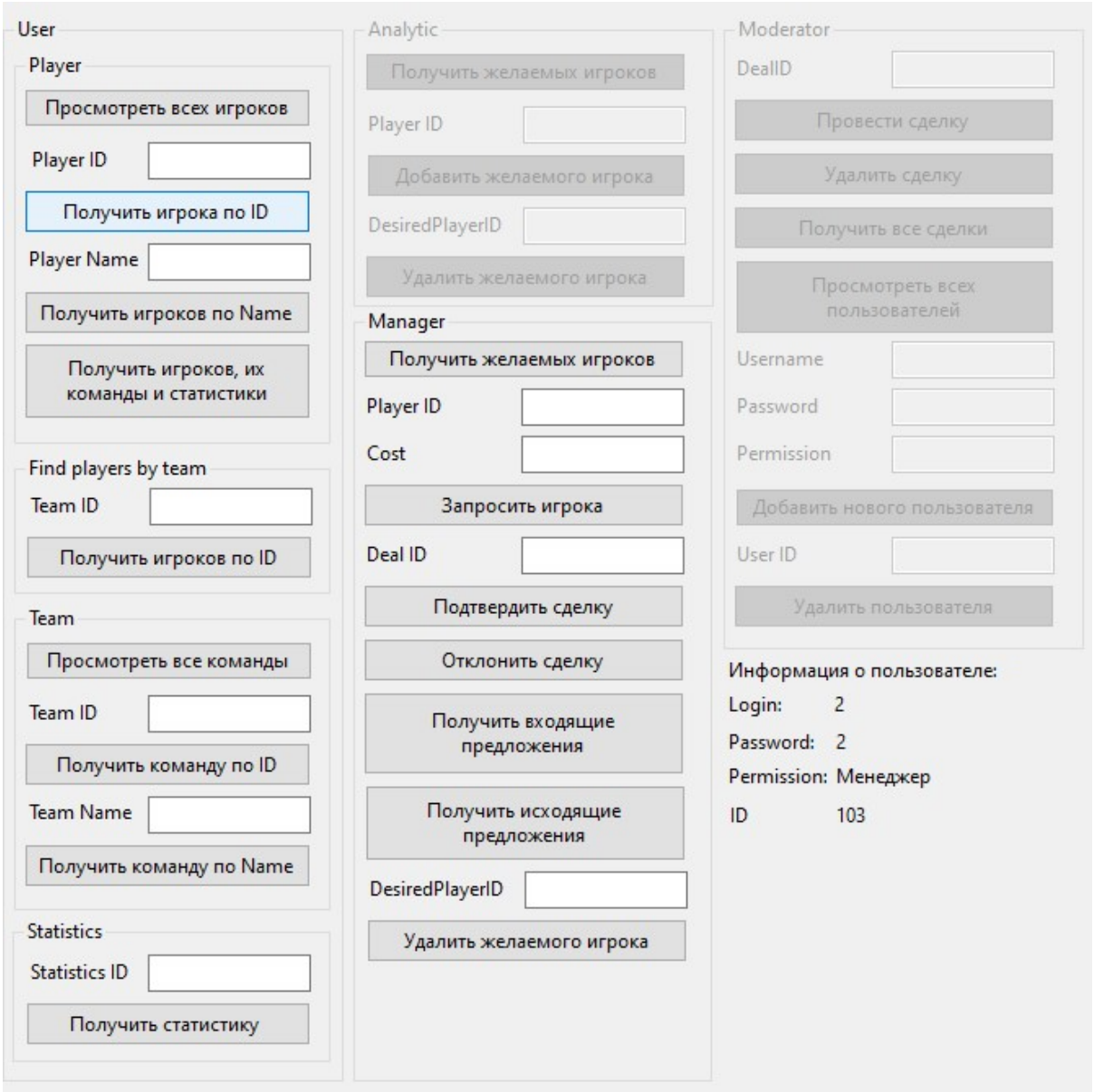


**Рисунок 15** – Операции, доступные менеджеру.

На рисунке 16 показаны команды, доступные модератору. Любой модератор может получить, провести и удалить сделки, добавить, удалить и просмотреть пользователей. Функции аналитика и менеджера для него недоступны.

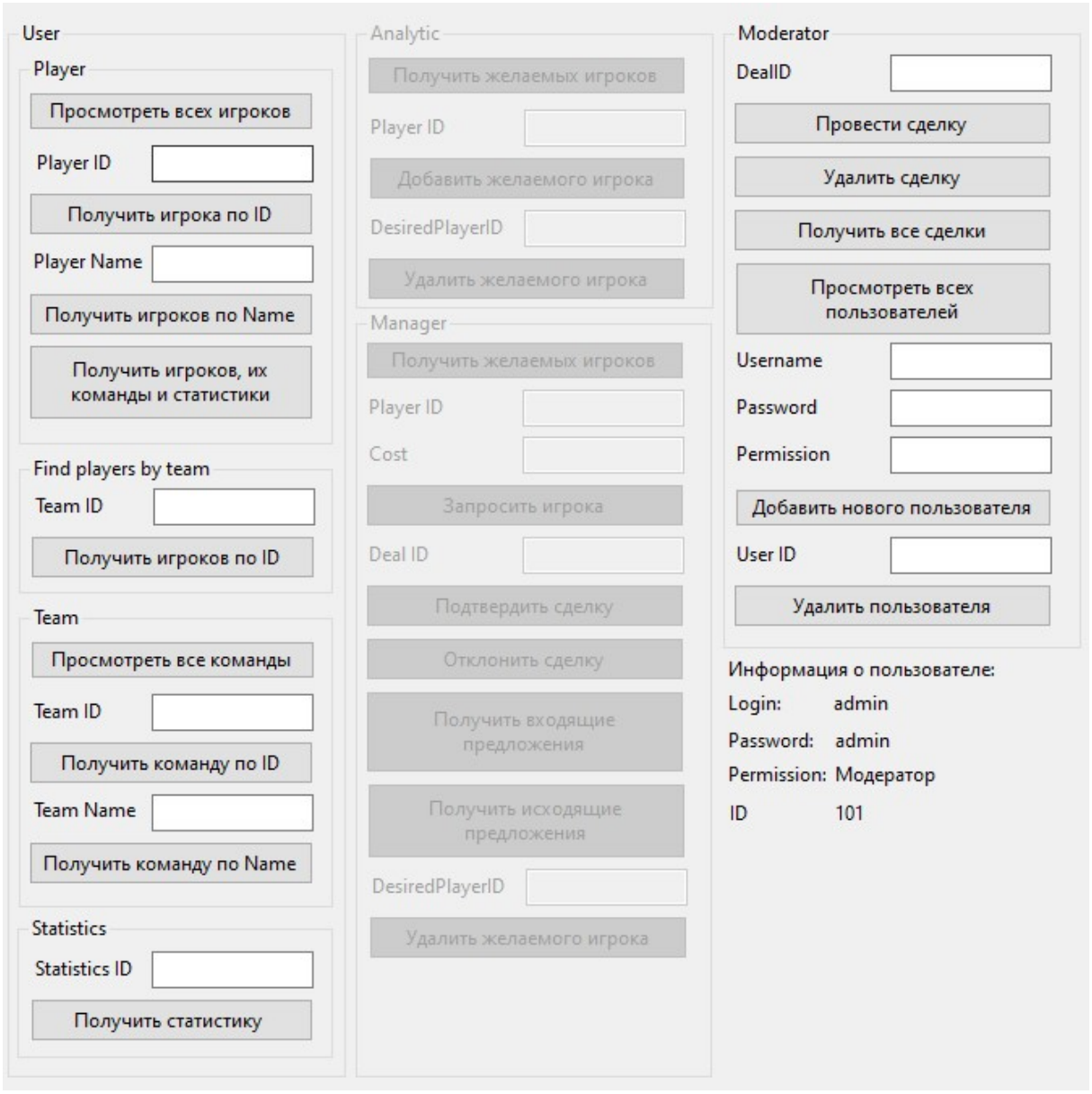


**Рисунок 16** – Операции, доступные модератору.

## 3.10 Вывод

В данном разделе были выбраны средства реализации поставленной задачи, создана база данных и описана ролевая модель на уровне БД, разработаны компоненты и описан порядок работы.

## Заключение

Цель курсовой работы достигнута.

В ходе выполнения курсовой работы было формализовано задание, выделены соответсвующие пользователи и их функционал, проведен анализ СУБД и выбрана наиболее подходящая для данной задачи, спроектирована база данных, архитектура приложение.

В результате, с использованием языка программирования C# и СУБД PostgreSQL было создано многофункциональное приложение для получения информации об игроках с возможностью их покупки и продажи. Получен опыт разработки базы данных и приложения по паттерну MVC.

В дальнейшей перспективе приложение и база данных могут был масштабированы. Может быть добавлен следующий функционал:

1) возможность добавления модератором игроков и команд;
2) возможность добавления нескольких лиг для команд и перехода игроков по лигам;
3) могут быть добавлены другие пользователи, например игрок, менеджер игрока и т.д.

## Список литературы


[1] Документация по C# [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 20.05.2021).

[2] Документация по семейству продуктов Visual Studio [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/visualstudio/?view=vs-2019 (дата обращения: 20.05.2021).

[3] Windows Forms [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-5.0 (дата обращения: 20.05.2021).

[4] PostgreSQL : Документация [Электронный ресурс] URL: https://postgrespro.ru/docs/postgresql (дата обращения: 20.05.2021).

[5] Документация по Entity Framework [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/ef/ (дата обращения: 20.05.2021).