|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

**«*Сравнительный анализ данных об игроках и результатах матчей НХЛ*»**

Студент ИУ7-61Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  М.В.Ефимова

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  А.С. Григорьев

(Подпись, дата) (И.О.Фамилия)

*Москва, 2022 г.*

## РЕФЕРАТ

Курсовая работа представляет собой реализацию базы данных и приложения, содержащего информацию о статистике клубов Национальной Хоккейной Лиги и о статистике игроков, а также позволяющего пользователю отслеживать и сравнивать статистику клубов и хоккеистов. Готовое приложение позволяет создавать список избранных игроков и клубов и отслеживать его, а также отслеживать текущие шансы каждой команды на выход в плей-офф.

Приложение реализовано на языке программирования Python в среде разработки Visual Studio Code. Взаимодействие с приложением производится через консоль. В качестве СУБД был выбран PostgreSQL.

Ключевые слова: НХЛ, команды, хоккеисты, список избранного, Python, PostgreSQL.

Расчетно-пояснительная записка к курсовой работе содержит 56 страниц, 38 иллюстраций, 2 таблицы.

**СОДЕРЖАНИЕ**

[РЕФЕРАТ 2](#_Toc106894814)

[ВВЕДЕНИЕ 5](#_Toc106894815)

[1 Аналитический раздел 6](#_Toc106894816)

[1.1 Обзор существующих аналогов 6](#_Toc106894817)

[1.1.1 NHL 6](#_Toc106894818)

[1.1.2 Flashscore 7](#_Toc106894819)

[1.1.3 EliteProspects 8](#_Toc106894820)

[1.2 Формализация задачи 9](#_Toc106894821)

[1.3 Формализация данных 10](#_Toc106894822)

[1.4 Формализация ролей 11](#_Toc106894823)

[1.5 Описание существующих СУБД 13](#_Toc106894824)

[1.5.1 Классификация СУБД по модели данных 14](#_Toc106894825)

[1.6 Выводы из аналитического раздела 17](#_Toc106894826)

[2 Конструкторский раздел 18](#_Toc106894827)

[2.1 Проектирование базы данных 18](#_Toc106894828)

[2.1.1 Таблицы 18](#_Toc106894829)

[2.1.2 Целостность данных 21](#_Toc106894830)

[2.1.3 Триггеры 23](#_Toc106894831)

[2.2 Требования к программе 25](#_Toc106894832)

[2.3 Выводы из конструкторского раздела 26](#_Toc106894833)

[3 Технологический раздел 27](#_Toc106894834)

[3.1 Средства реализации 27](#_Toc106894835)

[3.2 Детали реализации 28](#_Toc106894836)

[3.2.1 Создание таблиц 28](#_Toc106894837)

[3.2.2 Используемые функции и триггеры 30](#_Toc106894838)

[3.3 Взаимодействие с приложением 38](#_Toc106894839)

[3.4 Выводы из технологического раздела 43](#_Toc106894840)

[4 Экспериментальный раздел 44](#_Toc106894841)

[4.1 Технические характеристики 44](#_Toc106894842)

[4.2 Анализ зависимости 44](#_Toc106894843)

[4.2 Выводы из экспериментального раздела 53](#_Toc106894844)

[ЗАКЛЮЧЕНИЕ 54](#_Toc106894845)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 55](#_Toc106894846)

[ПРИЛОЖЕНИЕ А 56](#_Toc106894847)

## ВВЕДЕНИЕ

В настоящее время в Национальной Хоккейной Лиге более 600 действующих хоккеистов и 32 хоккейных клуба, которые ежедневно набирают очки в течение проходящих матчей, а также другую статистику. Эту статистику необходимо учитывать, а также было бы удобно не только следить за успехами любимых хоккеистов или клубов, но и иметь возможность сравнивать ее между собой.

Цель данной работы – создание базы данных для хранения данных о результативности игроков и о статистике команд НХЛ по результатам прошедших матчей, разработка приложения для регистрации и обновления статистики команд и игроков и построения статистических отчетов по введенным данным для выявления шансов каждой команды на выход в плей-офф в текущий момент времени.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1) формализовать задание, определить необходимый функционал;

2) провести анализ СУБД;

3) описать структуру базы данных, включая объекты, из которых она состоит;

4) спроектировать приложение для доступа к БД;

5) создать и заполнить БД;

6) разработать программное обеспечение, которое позволит пользователю получать, искать и сравнивать статистику игроков и команд Национальной Хоккейной Лиги;

7) исследовать зависимость шанса на выход в плей-офф команд от их статистики в регулярном чемпионате и количестве сыгранных матчей.

# 1 Аналитический раздел

В данном разделе будет проведен анализ существующих аналогов, схожих по функциональности с создаваемым приложением, проанализирована поставленная задача и рассмотрены различные способы ее реализации. Также будет проведена формализация данных и ролей, представлена диаграмма сущностей и диаграмма вариантов использования и проведен сравнительный анализ СУБД по модели данных.

## 1.1 Обзор существующих аналогов

В настоящий момент существует несколько сервисов для просмотра статистики хоккеистов и команд НХЛ, но у каждого сервиса есть недостатки. Большинство из сервисов предоставляют возможность только просмотра статистики, не давая возможности ее сравнить, но предоставляя возможность добавлять в избранное хоккеистов и команды, или только команды. Подробное описание нескольких существующих аналогов представлено в следующих разделах.

### 1.1.1 NHL

Данный сайт предоставляет удобный интерфейс для просмотра статистики хоккеистов как по всей лиге, так и отдельно по командам, а также статистики самих команд [1]. Он включает в себя возможность добавления команды в избранное, поиска игроков, перехода по клику на страницу игрока, содержащую только его статистику.

Однако на данном сайте отсутствует возможность добавлять игроков в избранное, а также сравнивать статистику игроков и команд между собой. Отсутствует и возможность просмотра шансов на выход команд в плей-офф.

Интерфейс сайта представлен на рисунке 1.1.

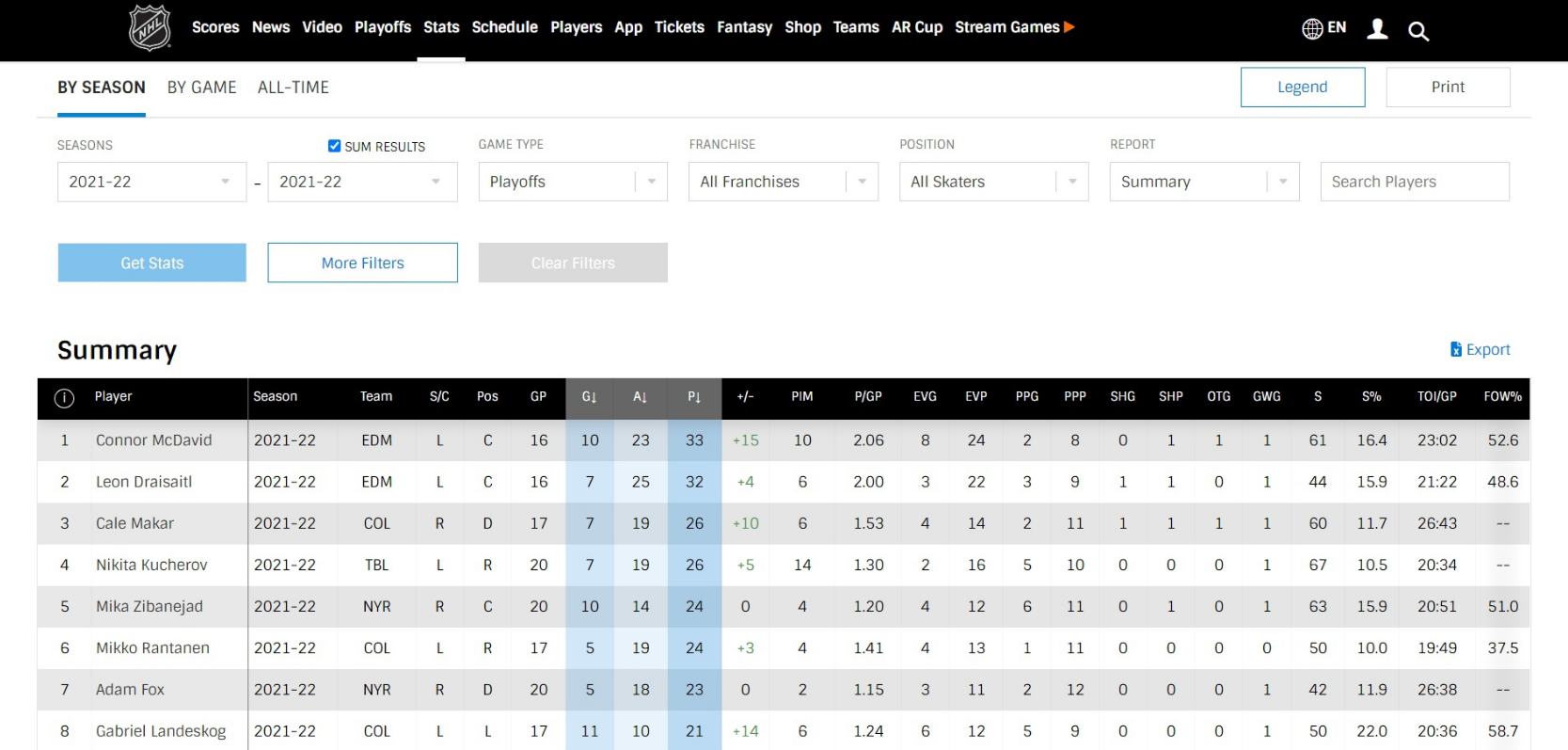


Рисунок 1.1 – Интерфейс сайта NHL

### 1.1.2 Flashscore

Данный сайт предоставляет возможность просмотра статистики команд, но статистику хоккеистов можно посмотреть только по одному [2]. В отличие от предыдущего сервиса, на данном сайте имеется возможность добавлять в избранное не только команды, но и отдельных игроков. Также присутствует возможность поиска команд и игроков.

Основным недостатком является отсутствующая возможность сравнить команды и игроков между собой, посмотреть статистику сразу всех игроков, а также увидеть шансы команд на выход в плей-офф.

Интерфейс сайта представлен на рисунке 1.2.

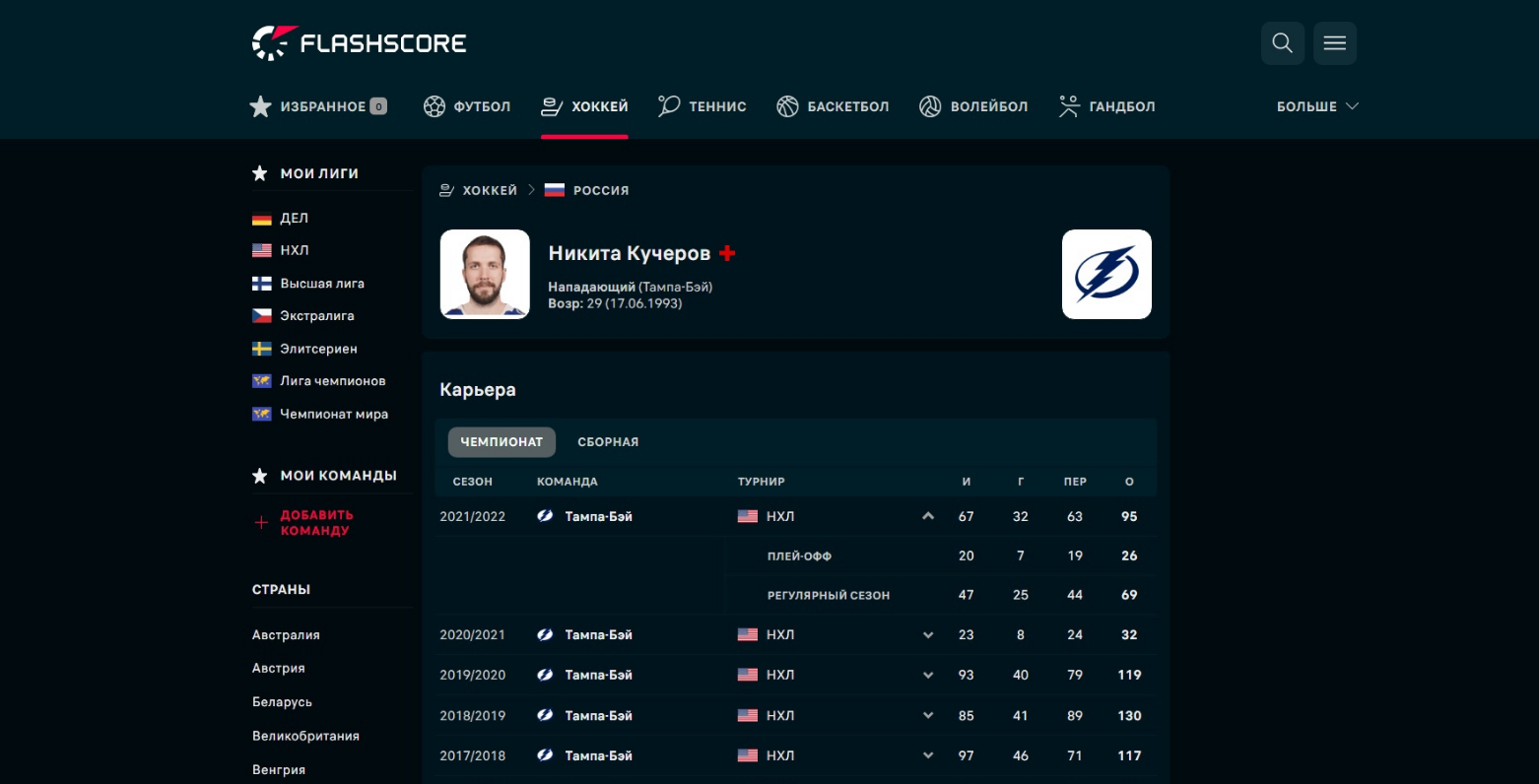


Рисунок 1.2 – Интерфейс сайта Flashscore

### 1.1.3 EliteProspects

Данный сайт имеет удобный интерфейс для просмотра информации и статистики об игроках и командах, предоставляет возможность поиска игроков и команд, просмотра как всех игроков лиги, так и отдельно по командам, также имеется возможность посмотреть информацию о персонале команд, прослушать произношение имен хоккеистов [3].

В отличие от предыдущих сервисов, на данном сайте возможность добавить команду или игрока в избранное отсутствует. Также нет возможности сравнить статистику игроков и команд, узнать шансы команд на выход в плей-офф.

Интерфейс сайта представлен на рисунке 1.3.

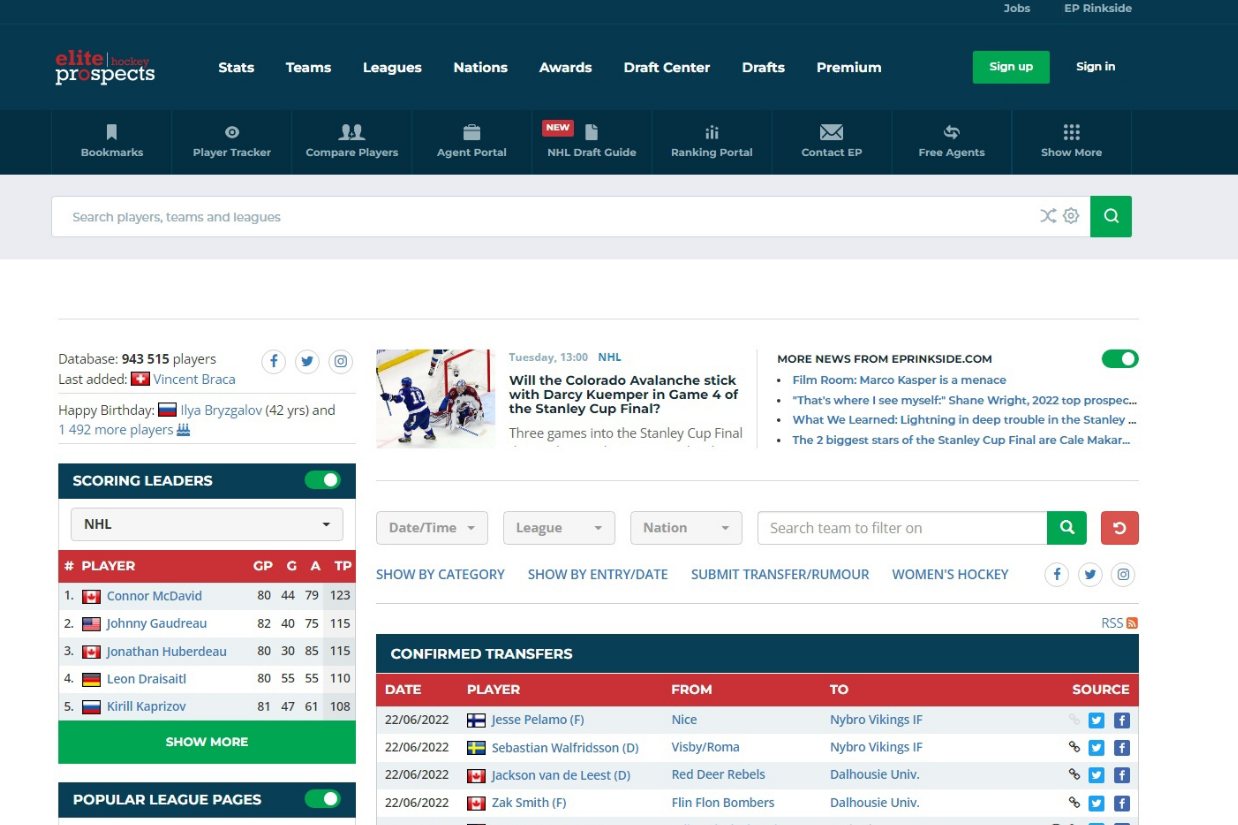


Рисунок 1.3 – Интерфейс сайта EliteProspects

## 1.2 Формализация задачи

В ходе выполнения курсовой работы необходимо разработать программу для отображения статистики игроков и команд НХЛ, хранящую информацию о личной статистике каждого игрока (голы, передачи, очки, плюс/минус и т.д.), статистике каждой команды (сыгранные матчи, очки, разница забитых и пропущенных шайб и т.д.). Также пользователь должен иметь возможность формировать персональный список любимых хоккеистов и любимых команд для дальнейшего быстрого просмотра информации о них, то есть добавлять игроков и команды в закладки. Также у пользователя должна иметься возможность просмотреть шансы команд на выход в плей-офф в текущий момент времени.

## 1.3 Формализация данных

База данных должна хранить информацию о:

* командах;
* хоккеистах;
* пользователях и их избранном списке хоккеистов и команд.

Сведения о каждой категории данных предоставлены в таблице 1.1.

Таблица 1.1 – категории и сведения о данных

|  |  |
| --- | --- |
| **Категория** | **Сведения** |
| Команда | Id, название, страна, количество сыгранных матчей, количество побед в основное время и по буллитам/в овертайме, количество поражений, очки, разница шайб |
| Хоккеисты | Id, имя, фамилия, дата рождения, место рождения, представляемая страна, рост, вес, хват, игровой номер, команда, количество очков, количество голов, количество передач, количество сыгранных матчей, плюс/минус |
| Пользователь | Id, имя, адрес электронной почты, страна, год рождения, логин, пароль, права доступа |
| Личный список игроков | Id, id пользователя, id хоккеиста |
| Личный список команд | Id, id пользователя, id команды |

На рисунке 1.4 приведена ER-диаграмма схемы сущностей.

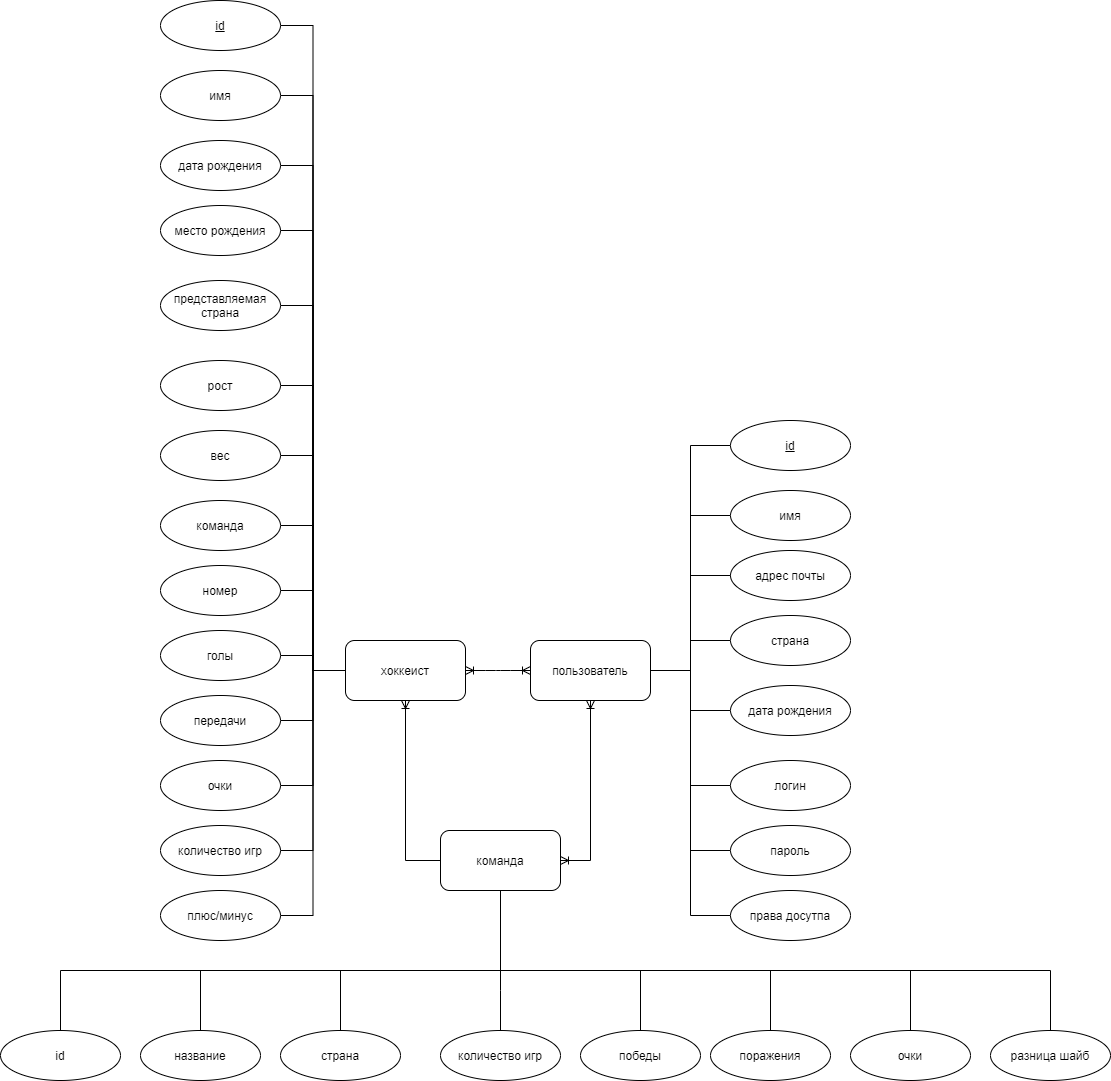


Рисунок 1.4 – ER-диаграмма сущностей нотации Чена

## 1.4 Формализация ролей

В данном курсовом проекте предусмотрено наличие двух типов пользователей: авторизованных и неавторизованных, так как для составления личного списка хоккеистов и команд необходима авторизация. Неавторизованные пользователи могут войти или зарегистрироваться. Авторизованные пользователи могут составлять личный список хоккеистов и команд и изменять его, а также пользоваться всеми возможностями приложения.

Для управления статистикой команд и хоккеистов, а также правами доступа пользователей предусмотрена роль администратора.

Функционал каждой роли представлен в таблице 1.2.

Таблица 1.2 – типы пользователей и их функционал

|  |  |
| --- | --- |
| **Тип пользователя** | **Функционал** |
| Неавторизованный пользователь | Регистрация, авторизация |
| Авторизованный пользователь | Просмотр статистики игроков и команд, поиск игроков и команд, сравнение игроков и команд, составление и редактирования личного списка игроков и команд, просмотр шансов команд на выход в плей-офф. |
| Модератор | Просмотр статистики игроков и команд, поиск игроков и команд, сравнение игроков и команд, составление и редактирования личного списка игроков и команд, просмотр шансов команд на выход в плей-офф.  Добавление игроков и команд, удаление игроков и команд. |
| Администратор | Просмотр статистики игроков и команд, поиск игроков и команд, сравнение игроков и команд, составление и редактирования личного списка игроков и команд, просмотр шансов команд на выход в плей-офф.  Добавление игроков и команд, удаление игроков и команд.  Изменение прав доступа пользователей, удаление пользователей. |

На рисунке 1.5 приведена Use-Case-диаграмма сущностей.

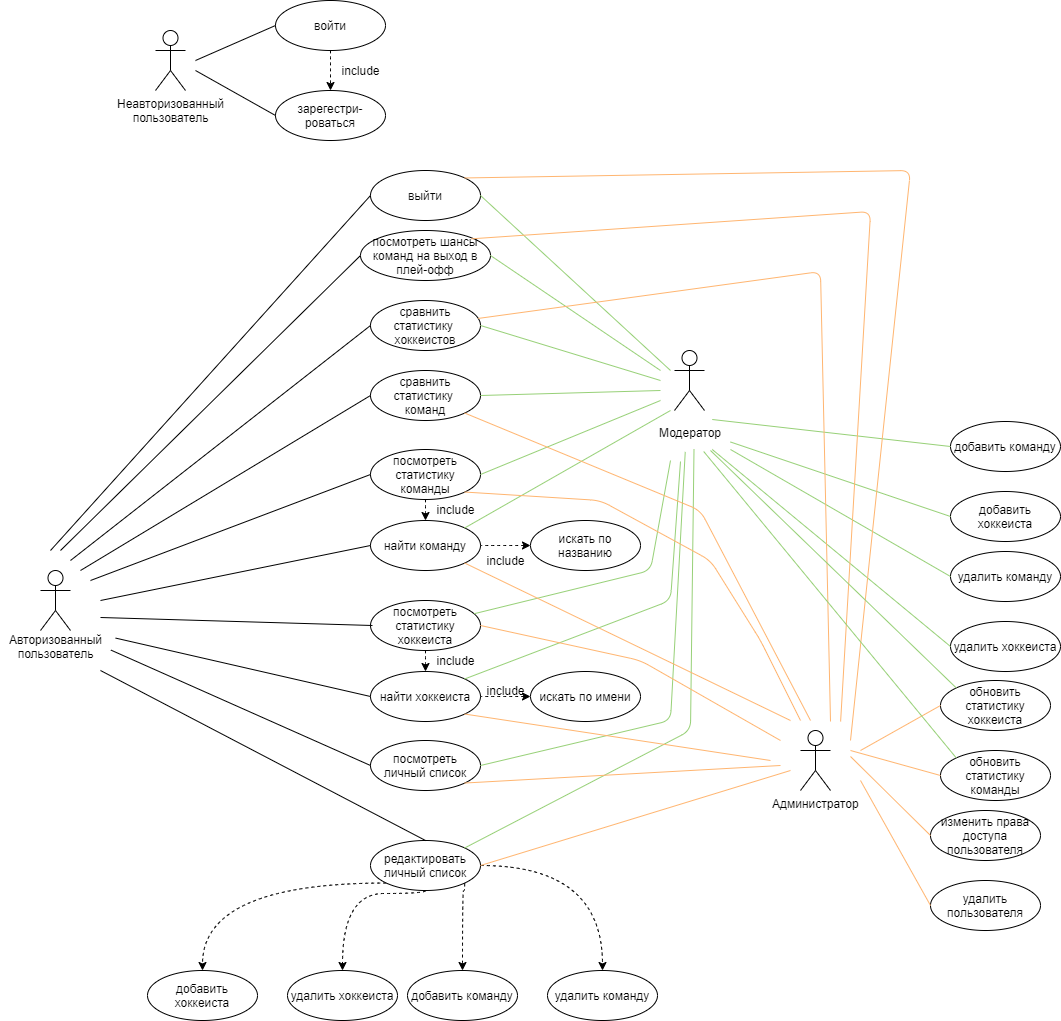


Рисунок 1.5 – Use-Case-диаграмма сущностей

## 1.5 Описание существующих СУБД

СУБД (система управления базами данных) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [4].

Основными функциями СУБД являются:

* управление данными во внешней памяти;
* управление данными в оперативной памяти с использованием дискового кэша;
* журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
* поддержка языков БД.

### 1.5.1 Классификация СУБД по модели данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных [5].

Существует 4 основных типа моделей организации данных:

* иерархическая;
* сетевая;
* реляционная;
* функциональная.

В иерархической модели данных используется представление базы данных в виде древовидной структуры, состоящей из объектов разных уровней. Среди объектов существуют связи, любой объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка к потомку, при этом вероятна ситуация, когда объект-предок имеет несколько потомков, тогда как у объекта-потомка обязательно должен быть только один предок.

Структура иерархической модели данных представлена на рисунке 1.6.

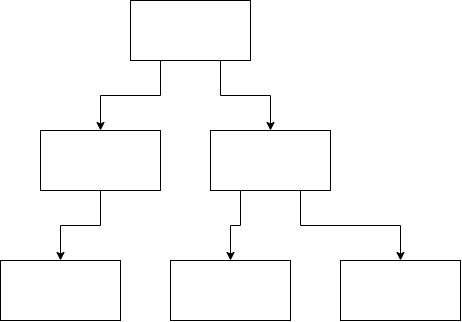


Рисунок 1.6 – структура иерархической модели данных

В сетевой модели данных, в отличии от иерархической, у потомка может быть любое число предков. Сетевая база данных состоит из набора экземпляров определенного типа записи и набора экземпляров определенного типа связей между этими записями [6].

Главным недостатком сетевой модели данных являются жесткость и высокая сложность схемы базы данных, построенной на основе этой модели. Так как логика процедуры выбора данных зависит от физической организации этих данных, то эта модель не является полностью независимой от приложения. Если придется изменить структуру данных, то нужно будет изменять и приложение.

Структура сетевой модели данных представлена на рисунке 1.7.

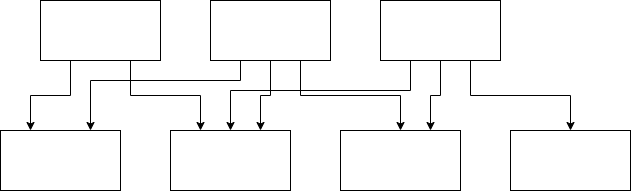


Рисунок 1.7 – структура сетевой модели данных

Реляционная модель данных является совокупностью данных и состоит из набора двумерных таблиц [7]. При табличной организации отсутствует иерархия элементов. Таблицы состоят из строк – записей и столбцов – полей. На пересечении строк и столбцов находятся конкретные значения. Для каждого поля определяется множество его значений. За счет возможности просмотра строк и столбцов в любом порядке достигается гибкость выбора подмножества элементов.

Реляционная модель является удобной и наиболее широко используемой формой представления данных.

Структура реляционной модели данных представлена на рисунке 1.8.



Рисунок 1.8 – структура реляционной модели данных

Наиболее популярными реляционными СУБД являются Oracle, Microsoft SQL Server и PostgreSQL.

Функциональные базы данных используются для решения аналитических задач, таких как финансовое моделирование и управление производительностью. Вместо того чтобы представлять объект записью с определенным содержанием или же кортежем в дереве, функциональная модель сообщает, какие функции (или операции) определены на этом объекте. Представление объекта — это дело реализации, и оно определяется на более низком уровне абстракции.

## 1.6 Выводы из аналитического раздела

В данном разделе был проведен обзор существующих аналогов, разобраны предоставляемы ими возможности и их недостатки, были формализованы задача, данные и роли. Также были представлены диаграммы компонентов и варианты использования. Были рассмотрены разные типы СУБД. В качестве используемой в данной работе была выбрана реляционная СУБД, так как она наиболее удобна в использовании при работе с таблицами и не имеет существенных недостатков, влияющих на сложность разработки программного обеспечения в данной работе.

# 2 Конструкторский раздел

В данном разделе будет спроектированы база данных и приложение, а именно, будут описаны таблицы базы данных и их компоненты, выделены первичные ключи, рассмотрены требования к программе и к каждому типу пользователей. Представлена диаграмма компонентов, а также проанализирована целостность данных. Также будут описаны функционалы используемых в программе триггеров и представлены их схемы.

## 2.1 Проектирование базы данных

### 2.1.1 Таблицы

База данных должна хранить рассмотренные в таблице 1.1 данные. В соответствии с этой таблицей можно выделить следующие таблицы:

* таблица команд Teams;
* таблица хоккеистов Players;
* таблица пользователей User;
* таблица хоккеистов, добавленных в избранное пользователями FavPlayers;
* таблица команд, добавленных в избранное пользователями FavTeams.

Таблица Teams должна хранить информацию о командах:

* id – уникальный идентификатор команды, PRIMARY KEY, INT;
* name – название команды, VARCHAR(50);
* country – страна команды, VARCHAR(50);
* games – количество сыгранных матчей, INT;
* wins – количество побед, INT;
* loses – количество поражений, INT;
* points – количество очков, INT;
* gdifference – разница забитых и пропущенных шайб, INT.

Таблица Players должна хранить информацию о хоккеистах:

* id – уникальный идентификатор игрока, PRIMARY KEY, INT;
* name – полное имя игрока, VARCHAR(50);
* number – игровой номер, INT;
* team\_id – идентификатор команды, за которую играет хоккеист, внешний ключ (таблица Teams, поле id), FOREIGN KEY, INT;
* birth\_year – год рождения, INT;
* birth\_place – место рождения, VARCHAR(50);
* country – представляемая страна, VARCHAR(50);
* shoots – хват, VARCHAR(10);
* height – рост, INT;
* weight – вес, INT;
* points – количество очков, INT;
* goals – количество забитых голов, INT;
* assists – количество сделанных передач, INT;
* games – количество сыгранных матчей, INT;
* plusminus – полезность игрока (плюс/минус), INT.

Таблица Users должна хранить информацию о пользователях:

* id – уникальный идентификатор пользователя, PRIMARY KEY, INT;
* name – имя пользователя, VARCHAR(50);
* country – страна, VARCHAR(50);
* mail – почта пользователя, VARCHAR(50);
* birth\_year – год рождения, INT;
* login – логин, VARCHAR(50) UNIQUE;
* password – пароль, VARCHAR(50);
* type – права доступа пользователя (0 – обычный пользователь, 1 – администратор, 2 – модератор), INT.

Таблица FavPlayers должна хранить информацию о хоккеистах, добавленных пользователями в личный список:

* id – уникальный идентификатор записи в личном списке, PRIMARY KEY, INT;
* usr\_id – идентификатор пользователя, которому принадлежит запись в личном списке, внешний ключ (таблица Users, поле id), FOREIGN KEY, INT;
* player\_id – идентификатор хоккеиста, которого добавили в личный список, внешний ключ (таблица Players, поле id), FOREIGN KEY, INT;

Таблица FavTeams должна хранить информацию о командах, добавленных пользователями в личный список:

* id – уникальный идентификатор записи в личном списке, PRIMARY KEY, INT;
* usr\_id – идентификатор пользователя, которому принадлежит запись в личном списке, внешний ключ (таблица Users, поле id), FOREIGN KEY, INT;
* team\_id – идентификатор команды, которую добавили в личный список, внешний ключ (таблица Teams, поле id), FOREIGN KEY, INT;

Диаграмма разрабатываемой базы данных представлена на рисунке 2.1.

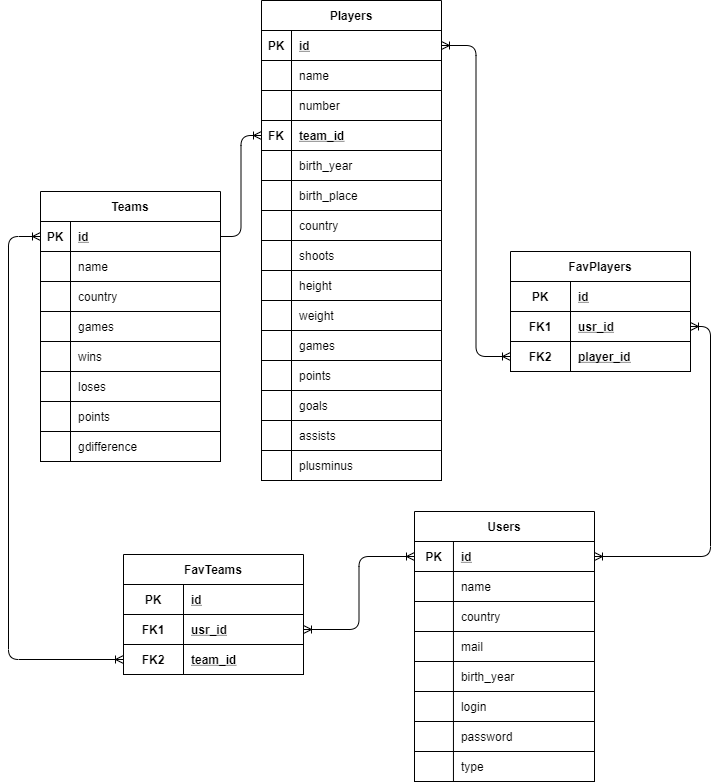


Рисунок 2.1 – ER-диаграмма

### 2.1.2 Целостность данных

Для обеспечения целостности таблиц необходимо, чтобы все строки в таблице имели уникальный идентификатор (первичный ключ).

Как было показано в предыдущем разделе, каждая таблица имеет первичный ключ:

1. у таблицы Teams – поле id;

2. у таблицы Players – поле id;

3. у таблицы Users – поле id;

4. у таблицы FavPlayers – поле id;

5. у таблицы FavTeams – поле id.

Для обеспечения ссылочной целостности необходимы внешние ключи, которые обеспечивают согласованное состояние между таблицей, на которую ссылаются, и таблицей, которая ссылается на другую.

Между таблицами Teams и Players существует связь "один ко многим", для ее формализации создается один внешний ключ: поле team\_id таблицы Players ссылается на первичный ключ таблицы Teams (поле id).

Между таблицами Users и FavPlayers существует связь "многие ко многим", для ее формализации создаются два внешних ключа: поля usr\_id и players\_id таблицы FavPlayers ссылаются на первичные ключи таблиц Users и Players соответственно (поля id).

Между таблицами Users и FavTeams существует связь "многие ко многим", для ее формализации создаются два внешних ключа: поля usr\_id и teams\_id таблицы FavTeams ссылаются на первичные ключи таблиц Users и Teams соответственно (поля id).

В связи с тем, что при удалении строк в основных таблицах Teams, Players, Users могут иметься внешние ключи, которые ссылаются на данные строки, что означает, что строки не смогут быть удалены, пока не уничтожится связь, были созданы триггеры на удаление данных для каждой таблицы.

Для обеспечения целостности полей необходимо указать набор значений данных, которые являются допустимыми для поля, и определить, возможно ли использование значения NULL.

В таблице Teams для полей name, games, wins, loses, points, gdifference недопустимо значение NULL, поле country может принимать значение NULL.

В таблице Players для полей name, team\_id, birth\_year, height, weight, points, goals, assists, games, number, plusminus недопустимо значение NULL, поля birth\_place, country, shoots могут принимать значение NULL.

В таблице Users для полей name, birth\_year, mail, login, password, type недопустимо значение NULL, поле country может принимать значение NULL. Поле login также должно быть уникальным (в связи с тем, что авторизация пользователя осуществляется по его логину). Поле type может принимать только значения 0, 1, 2.

В таблицах FavPlayers и FavTeams для всех полей недопустимо значение NULL.

### 2.1.3 Триггеры

Триггер – это особая разновидность хранимой процедуры, которая автоматически выполняется при возникновении события на сервере базы данных, поэтому с их помощью можно успешно реализовать определенные действия.

Для данной работы было создано три триггера на удаление, каждый из которых удаляет записи, которые имеют связь с записями основных таблиц, так как для того, чтобы избежать ошибки, необходимо следить за корректным удалением данных. Это и осуществляют триггеры.

Триггер для таблицы Teams обеспечивает удаление из таблицы FavTeams всех записей, в которых участвует удаляемая команда.

Триггер для таблицы Players обеспечивает удаление из таблицы FavPlayers всех записей, в которых участвует удаляемый хоккеист.

Триггер для таблицы Users обеспечивает удаление всего списка избранного пользователя, то есть удаление из таблиц FavPlayers и FavTeams всех записей, которые добавил туда пользователь.

Схемы триггеров для основных таблиц приведены на рисунках 2.2-2.4.

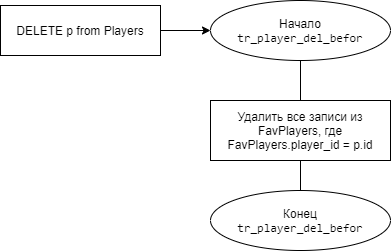


Рисунок 2.2 – схема триггера на удаление записей, связанных с хоккеистом, из другой таблицы

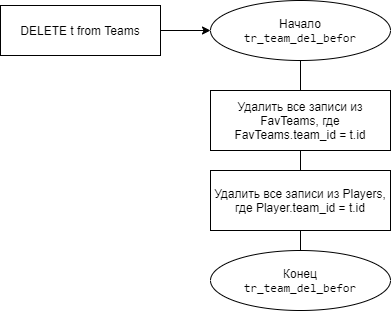


Рисунок 2.3 – схема триггера на удаление записей, связанных с командой, из других таблиц

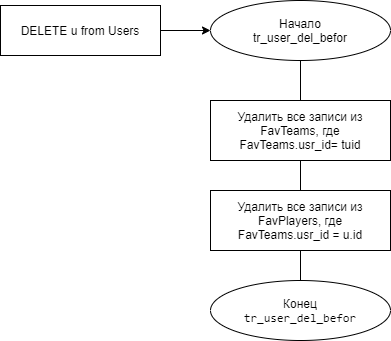


Рисунок 2.4 – схема триггера на удаление записей, связанных с пользователем, из других таблиц

## 2.2 Требования к программе

Для успешной реализации поставленной задачи в соответствии с представленной раннее Use-Case диаграммой на рисунке 1.5 программа должна предоставлять следующие возможности для обычного пользователя:

* авторизация;
* регистрация;
* просмотр статистики всех игроков;
* просмотр статистики всех клубов;
* просмотр статистики игроков определенной команды;
* просмотр статистики определенной команды;
* добавление игрока в избранное;
* добавление команды в избранное;
* удаление игрока из избранного;
* удаление команды из избранного;
* просмотр списка избранного;
* поиск игрока по имени;
* поиск команды по названию;
* сравнение статистики игроков;
* сравнение статистики команд;
* просмотр текущих шансов команды на выход в плей-офф;
* выход из аккаунта.

Для модератора добавляются возможности:

* добавление нового игрока;
* добавление новой команды;
* удаление игрока;
* удаление команды;
* изменение статистики игроков;
* изменение статистики команд.

Для администратора добавляются возможности:

* изменение прав доступа пользователей;
* удаление пользователей.

## 2.3 Выводы из конструкторского раздела

В данном разделе была спроектирована база данных и приложение для доступа к ней. Были описаны таблицы базы данных и их компоненты. Представлена диаграмма компонентов и схемы триггеров. Описаны функционалы используемых триггеров. Также были формализованы требования к программе и описаны все функции, которые она должна выполнять, формализованы требования к каждому типу пользователей.

# 3 Технологический раздел

В данном разделе будет обоснован выбор СУБД, языка программирования и среды разработки. Также будут описана архитектура приложения, представлены детали реализации: создание описанных раннее таблиц, список используемых в программном продукте функций, листинги ключевых функций программного продукта, способы взаимодействия с программным продуктом и примеры работы программного продукта.

## 3.1 Средства реализации

В данной работе в качестве СУБД был выбран PostgreSQL[8], так как:

* распространяется свободно;
* поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных;
* предоставляет обширный функционал, включая создание пользовательских функций, процедур, триггеров и возможности индексирования.

В данной работе в качестве языка программирования был выбран Python[9], так как:

* данный язык программирования позволяет обеспечить работу с PostgreSQL, а также имеет простой интерфейс подключения к базе данных;
* данный язык программирования обеспечивает быструю разработку программного продукта, он универсален, а написанный продукт можно легко расширять;
* данный язык программирования обладает полной документацией.

В данной работе в качестве среды разработки была выбрана «Visual Studio Code»[10] по следующим причинам:

* является бесплатной средой разработки;
* имеет большое количество различных расширений, облегчающих процесс разработки, также расширения просты в установке;
* имеет понятный и удобный в использовании интерфейс.

## 3.2 Детали реализации

### 3.2.1 Создание таблиц

На листинге 3.1 представлено создание таблицы Teams. Для всех полей за исключением поля country недопустимо значение NULL. Поле id является уникальным идентификатором и имеет ограничение PRIMARY KEY.

Листинг 3.1 – создание таблицы Teams

|  |
| --- |
| CREATE TABLE teams(  id SERIAL PRIMARY KEY NOT NULL,  name VARCHAR(50) NOT NULL,  country VARCHAR(50),  games INTEGER NOT NULL,  wins INTEGER NOT NULL,  loses INTEGER NOT NULL,  points INTEGER NOT NULL,  gdifference INTEGER NOT NULL  ); |

На листинге 3.2 представлено создание таблицы Players. Для полей name, team\_id, birth\_year, height, weight, points, goals, assists, games, number, plusminus недопустимо значение NULL, поля birth\_place, country, shoots могут принимать значение NULL. Поле id является уникальным идентификатором и имеет ограничение PRIMARY KEY. Поле team\_id таблицы Players является внешним ключом и ссылается на первичный ключ таблицы Teams (поле id).

Листинг 3.2 – создание таблицы Players

|  |
| --- |
| CREATE TABLE players(  id SERIAL PRIMARY KEY NOT NULL,  name VARCHAR(50) NOT NULL,  team\_id INTEGER NOT NULL,  birth\_year INTEGER NOT NULL,  country VARCHAR(50),  birth\_place VARCHAR(50),  height INTEGER NOT NULL,  weight INTEGER NOT NULL,  shoots VARCHAR(10),  points INTEGER NOT NULL,  goals INTEGER NOT NULL,  assists INTEGER NOT NULL,  number INTEGER NOT NULL,  games INTEGER NOT NULL,  plusminus INTEGER NOT NULL,  FOREIGN KEY (team\_id) REFERENCES teams(id)  ); |

На листинге 3.3 представлено создание таблицы Users. Для полей name, birth\_year, mail, login, password, type недопустимо значение NULL, поле country может принимать значение NULL. Поле login также должно быть уникальным (в связи с тем, что авторизация пользователя осуществляется по его логину). Поле type может принимать только значения 0, 1, 2. Поле id является уникальным идентификатором и имеет ограничение PRIMARY KEY.

Листинг 3.3 – создание таблицы Users

|  |
| --- |
| CREATE TABLE users(  id SERIAL PRIMARY KEY NOT NULL,  name VARCHAR(50) NOT NULL,  country VARCHAR(50),  mail VARCHAR(80) NOT NULL,  birth\_year INTEGER NOT NULL,  login VARCHAR(50) UNIQUE NOT NULL,  password VARCHAR(50) NOT NULL,  type INTEGER NOT NULL  ); |

На листинге 3.4 представлено создание таблиц FavPlayers и FavTeams. В таблицах FavPlayers и FavTeams для всех полей недопустимо значение NULL. Поля id являюся уникальными идентификатороми для каждой из таблиц и имеют ограничение PRIMARY KEY.

Листинг 3.4 – создание таблиц FavPlayers и FavTeams

|  |
| --- |
| CREATE TABLE favplayers(  id SERIAL PRIMARY KEY NOT NULL,  usr\_id INTEGER NOT NULL,  player\_id INTEGER NOT NULL,  FOREIGN KEY (usr\_id) REFERENCES users(id),  FOREIGN KEY (player\_id) REFERENCES players(id)  );  CREATE TABLE favteams(  id SERIAL PRIMARY KEY NOT NULL,  usr\_id INTEGER NOT NULL,  team\_id INTEGER NOT NULL,  FOREIGN KEY (usr\_id) REFERENCES users(id),  FOREIGN KEY (team\_id) REFERENCES teams(id)  ); |

### 3.2.2 Используемые функции и триггеры

Для соответствия заявленным выше требованиям к функционалу приложения были написаны следующие функции:

* existing\_log – проверка на существование логина;
* existing\_mail – проверка на существование адреса электронной почты в системе;
* registration\_us – регистрация нового пользователя;
* log\_in – авторизация пользователя;
* print\_all\_players – печать статистики всех игроков лиги;
* print\_all\_teams – печать статистики всех клубов лиги;
* print\_player\_from\_team – печать статистики игроков определенной команды;
* print\_team – печать статистики определенной команды;
* add\_player\_to\_list – добавить игрока в список избранных;
* add\_team\_to\_list – добавить команду в избранное;
* del\_player\_from\_list – удалить игрока из избранного;
* del\_team\_from\_list – удалить команду из избранного;
* print\_user\_list – распечатать список избранного;
* find\_player\_by\_name – поиск игрока по имени;
* find\_team\_by\_name – поиск команды по названию;
* comparison\_players – сравнение статистики игроков;
* comparison\_teams – сравнение статистики команд;
* analyzing\_teams – анализ шансов команд на выход в плей-офф;
* add\_player\_to\_db – добавить игрока в базу данных;
* add\_team\_to\_db – добавить команду в базу данных;
* delete\_player\_from\_db – удалить игрока из базы данных;
* delete\_team\_from\_db – удалить команду из базы данных;
* update\_player\_in\_db – обновить статистику игроков в базе данных;
* update\_team\_in\_db – обновить статистику команд в базе данных;
* delete\_user\_from\_db – удалить пользователя из базы данных;
* change\_user\_type – изменить права доступа пользователя.

Далее на листингах 3.5-3.17 представлены коды ключевых функций данного программного продукта.

Листинг 3.5 – регистрация нового пользователя

|  |
| --- |
| def registration\_us(name\_us, country\_us, mail\_us, birth\_us, log\_us, pass\_us, type\_us):  insert\_query = "('{name}', '{country}', '{mail}', '{birth\_year}', '{login}', '{password}', '{type}')".format(  name=name\_us,  country=country\_us,  mail=mail\_us,  birth\_year=birth\_us,  login=log\_us,  password=pass\_us,  type=type\_us)    cursor.execute("INSERT INTO users (name, country, mail, birth\_year, login, password, type) VALUES " + insert\_query)  connection.commit() |

Листинг 3.6 – добавление игрока в список избранного

|  |
| --- |
| def add\_player\_to\_list(player\_name, team\_name, log\_us):  request = "select id from players where name = %s and team\_id = (select id from teams where name = %s)"  cursor.execute(request, (player\_name, team\_name,))  res = cursor.fetchone()    request = "select id from users where login = %s"  cursor.execute(request, (log\_us,))  res1 = cursor.fetchone()    insert\_query = "('{usr\_id}', '{player\_id}')".format(usr\_id = res1[0], player\_id = res[0])  cursor.execute("INSERT INTO favplayers (usr\_id, player\_id) VALUES " + insert\_query)  connection.commit() |

Листинг 3.7 – добавление команды в список избранного

|  |
| --- |
| def add\_team\_to\_list(team\_name, log\_us):  request = "select id from teams where name = %s"  cursor.execute(request, (team\_name,))  res = cursor.fetchone()  request = "select id from users where login = %s"  cursor.execute(request, (log\_us,))  res1 = cursor.fetchone()    insert\_query = "('{usr\_id}', '{team\_id}')".format(usr\_id = res1[0], team\_id = res[0])  cursor.execute("INSERT INTO favteams (usr\_id, team\_id) VALUES " + insert\_query)  connection.commit() |

Листинг 3.8 – удаление игрока из списка избранного

|  |
| --- |
| def del\_player\_from\_list(player\_name, team\_name, log\_us):  request = "select id from players where name = %s and team\_id = (select id from teams where name = %s)"  cursor.execute(request, (player\_name, team\_name,))  res = cursor.fetchone()    request = "select id from users where login = %s"  cursor.execute(request, (log\_us,))  res1 = cursor.fetchone()    request = "delete from favplayers where usr\_id = %s and player\_id = %s"  cursor.execute(request, (res1[0], res[0],))  connection.commit() |

Листинг 3.9 – удаление команды из списка избранного

|  |
| --- |
| def del\_team\_from\_list(team\_name, log\_us):  request = "select id from teams where name = %s"  cursor.execute(request, (team\_name,))  res = cursor.fetchone()  request = "select id from users where login = %s"  cursor.execute(request, (log\_us,))  res1 = cursor.fetchone()    request = "delete from favteams where usr\_id = %s and team\_id = %s"  cursor.execute(request, (res1[0], res[0],))  connection.commit() |

Листинг 3.10 – просмотр списка избранного

|  |
| --- |
| def print\_user\_list(log\_us):  request = "select player\_id from favplayers where usr\_id = (select id from users where login = %s)"  cursor.execute(request, (log\_us,))  res = cursor.fetchall()  table\_pl = PrettyTable()  table\_pl.field\_names = ["Имя", "Игровой номер", "Год рождения", "Место рождения", "Страна", "Рост", "Вес", "Хват", "Очки", "Голы", "Передачи", "Количество игр", "+/-"]    for row in res:  request = "select \* from players where id = %s"  cursor.execute(request, (row[0],))  res1 = cursor.fetchone()  table\_pl.add\_row([res1[1], res1[3], res1[5], res1[4], res1[6], res1[7], res1[8], res1[9], res1[10])    print("Избранные игроки: ")  print(table\_pl)  request = "select team\_id from favteams where usr\_id = (select id from users where login = %s)"  cursor.execute(request, (log\_us,))  res2 = cursor.fetchall()  table\_tm = PrettyTable()  table\_tm.field\_names = ["Название", "Страна", "Количество игр", "Победы", "Поражения", "Очки", "Разница шайб"]    for row in res2:  request = "select \* from teams where id = %s"  cursor.execute(request, (row[0],))  res3 = cursor.fetchone()  table\_tm.add\_row([res3[1], res3[2], res3[3], res3[4], res3[5], res3[6], res3[7]])  print("Избранные команды: ")  print(table\_tm) |

Листинг 3.11 – сравнение статистики игроков

|  |
| --- |
| def comparison\_players(n):  table\_pl = PrettyTable()  table\_pl.field\_names = ["Имя", "Игровой номер", "Год рождения", "Место рождения", "Страна", "Рост", "Вес", "Хват", "Очки", "Голы", "Передачи", "Количество игр", "+/-"]    for i in range(n):  player\_name = input("Введите полное имя игрока: ")  team\_name = input("Введите полное название команды: ")  request = "select \* from players where name = %s and team\_id = (select id from teams where name = %s)"  cursor.execute(request, (player\_name, team\_name,))  res = cursor.fetchone()  if res != []:  table\_pl.add\_row([res[1], res[3], res[5], res[4], res[6], res[7], res[8], res[9], res[10]])  else:  print("\nТаких игроков нет")  print(table\_pl) |

Листинг 3.12 – сравнение статистики команд

|  |
| --- |
| def comparison\_teams(n):  table\_tm = PrettyTable()  table\_tm.field\_names = ["Название", "Страна", "Количество игр", "Победы", "Поражения", "Очки", "Разница шайб"]  for i in range(n):  team\_name = input("Введите полное название команды: ")  request = "select \* from teams where name = %s"  cursor.execute(request, (team\_name,))  res = cursor.fetchone()  if res != []:  table\_tm.add\_row([res[1], res[2], res[3], res[4], res[5], res[6], res[7]])  else:  print("\nКоманд с таким названием нет")  print(table\_tm) |

Листинг 3.13 – анализ текущих шансов команд на выход в плей-офф

|  |
| --- |
| def analyzing\_teams():  request = "select \* from teams"  cursor.execute(request)  res = cursor.fetchall()  flag = True    if res != []:  table\_tm = PrettyTable()  table\_tm.field\_names = ["Название", "Шанс выйти в плей-офф (%)"]  max\_games = 0  all\_games = 86  n = 0  res.sort(key = lambda x: x[6], reverse=True)  for row in res:  n = n + 1  if row[3] != all\_games:  flag = False  if row[3] >= max\_games:  max\_games = row[3]  if n == 16:  points\_half\_table = row[6]  wins\_half\_table = row[4]  if n == 17:  points\_half\_table2 = row[6]  games\_half\_table = row[3]  max\_points = 2 \* max\_games  schance = 0.0  n = 0  for row in res:  n = n + 1  schance = row[6] / (max\_points / 100)  if n > 16:  temp\_points = (all\_games - row[3]) \* 2  if (temp\_points + row[6]) < points\_half\_table:  schance = 0.0  if (temp\_points + row[6]) == points\_half\_table:  if row[4] < wins\_half\_table:  schance = 0.0  if n <= 16:  temp\_points = (all\_games - games\_half\_table) \* 2  if (temp\_points + points\_half\_table2) < row[6]:  schance = 100.0    if n <= 16 and flag:  schance = 100.0  if n > 16 and flag:  schance = 0.0  schance = float('{:.2f}'.format(schance))  table\_tm.add\_row([row[1], schance])  else:  print("\nКоманд нет")  print(table\_tm) |

Листинг 3.14 – добавление игрока

|  |
| --- |
| def add\_player\_to\_db():  player\_name = input("Введите полное имя игрока: ")  team = int(input("Введите id команды: "))  birth\_year1 = int(input("Введите год рождения: "))  birth\_place1 = input("Введите место рождения: ")  country1 = input("Введите представляемую страну: ")  shoot = input("Введите хват(R/L): ")  height1 = int(input("Введите рост: "))  weight1 = int(input("Введите вес: "))  games1 = int(input("Введите количество сыгранных игр: "))  points1 = int(input("Введите количество очков: "))  goals1 = int(input("Введите количество голов: "))  assists1 = int(input("Введите количество передач: "))  number1 = int(input("Введите игровой номер: "))  plusminus1 = int(input("Введите плюс/минус: "))  insert\_query += "('{name}', '{team\_id}', '{birth\_year}', '{country}', '{birth\_place}', {height}, '{weight}', '{shoots}', '{points}', '{goals}', '{assists}', '{number}', '{games}', '{plusminus}')".format(  name=player\_name,  team\_id=team,  birth\_year=birth\_year1,  country=country1,  birth\_place=birth\_place1,  height=height1,  weight=weight1,  shoots=shoot,  points=points1,  goals=goals1,  assists=assists1,  number=number1,  games=games1,  plusminus=plusminus1)  cursor.execute("INSERT INTO players (name, team\_id, birth\_year, country, birth\_place, height, weight, shoots, points, goals, assists, number, games, plusminus) VALUES " + insert\_query)  connection.commit() |

Листинг 3.15 – удаление игрока

|  |
| --- |
| def delete\_player\_from\_db():  player\_id = int(input("Введите id игрока: "))  request = "delete from players where id = %s"  cursor.execute(request, (player\_id,))  connection.commit() |

Листинг 3.16 – удаление команды

|  |
| --- |
| def delete\_team\_from\_db():  team\_id = int(input("Введите id команды: "))  request = "delete from teams where id = %s"  cursor.execute(request, (team\_id,))  connection.commit() |

Листинг 3.17 – удаление пользователя

|  |
| --- |
| def delete\_user\_from\_db():  user\_id = int(input("Введите id пользователя: "))  request = "delete from users where id = %s"  cursor.execute(request, (user\_id,))  connection.commit() |

Для функций удаления игроков, команд и пользователей были написаны триггеры, представленные на листингах 3.18-3.20.

Листинг 3.17 – триггер на удаление записей, связанных с игроком, из другой таблицы

|  |
| --- |
| CREATE FUNCTION trigger\_players\_before\_del () RETURNS trigger AS '  BEGIN  if (select count(\*) from favplayers where favplayers.player\_id = OLD.id)>0  then delete from favplayers where favplayers.player\_id = OLD.id;  end if;  return OLD;  END;  ' LANGUAGE plpgsql;  CREATE TRIGGER tr\_player\_del\_befor  BEFORE DELETE ON players FOR EACH ROW  EXECUTE PROCEDURE trigger\_players\_before\_del(); |

Листинг 3.18 – триггер на удаление записей, связанных с командой, из других таблиц

|  |
| --- |
| CREATE FUNCTION trigger\_teams\_before\_del () RETURNS trigger AS '  BEGIN  if (select count(\*) from favteams where favteams.team\_id = OLD.id)>0  then delete from favteams where favteams.team\_id = OLD.id;  end if;  if (select count(\*) from players where players.team\_id = OLD.id)>0  then delete from players where players.team\_id = OLD.id;  end if;  return NULL;  END;  ' LANGUAGE plpgsql;  CREATE TRIGGER tr\_team\_del\_befor  BEFORE DELETE ON teams FOR EACH ROW  EXECUTE PROCEDURE trigger\_teams\_before\_del(); |

Листинг 3.18 – триггер на удаление записей, связанных с пользователем, из других таблиц

|  |
| --- |
| CREATE FUNCTION trigger\_users\_before\_del () RETURNS trigger AS '  BEGIN  if (select count(\*) from favplayers where favplayers.user\_id = OLD.id)>0  then delete from favteams where favplayers.user\_id = OLD.id;  end if;  if (select count(\*) from favteams where favteams.user\_id = OLD.id)>0  then delete from favteams where favteams.user\_id = OLD.id;  end if;  return NULL;  END;  ' LANGUAGE plpgsql;  CREATE TRIGGER tr\_user\_del\_befor  BEFORE DELETE ON users FOR EACH ROW  EXECUTE PROCEDURE trigger\_users\_before\_del(); |

## 3.3 Взаимодействие с приложением

В качестве программного продукта было создано консольное приложение.

Работа программы начинается c авторизации (рисунок 3.1). Пользователь должен ввести свои логин и пароль, чтобы выполнить вход. Если пользователь еще не имеет аккаунт, он может зарегистрироваться, выбрав пункт «Зарегистрироваться».

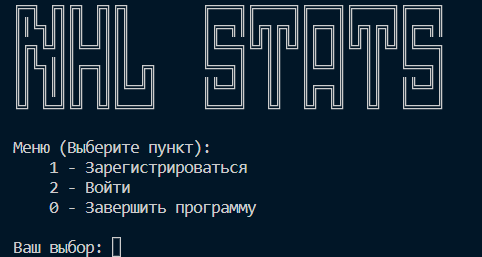


Рисунок 3.1 – авторизация пользователя

При выборе регистрации необходимо ввести все запрашиваемые данные, после чего пользователь будет зарегистрирован и произойдет вход в систему, что представлено на рисунке 3.2.

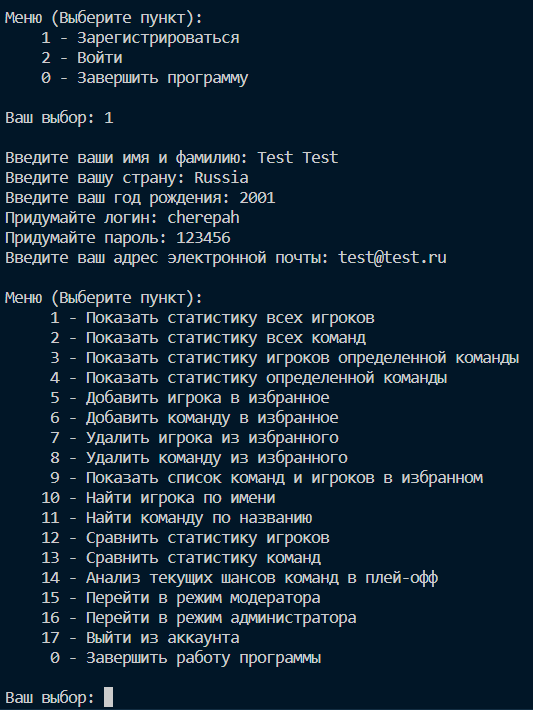


Рисунок 3.2 – успешная регистрация в системе

При выборе пункта «Войти» необходимо ввести корректные логин и пароль, после чего будет произведен вход в систему, что представлено на рисунке 3.3.

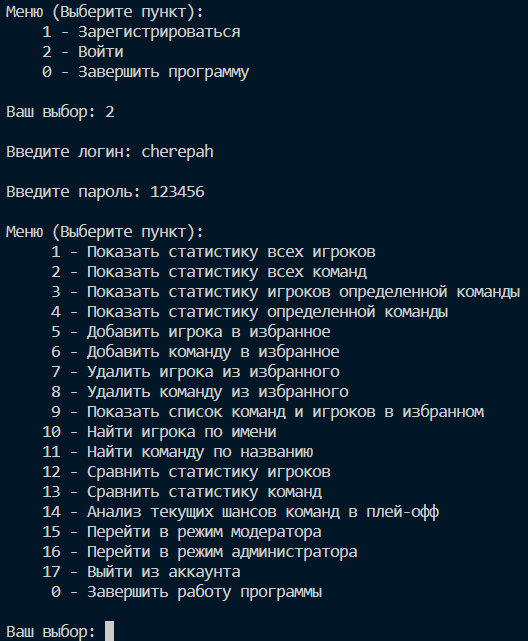


Рисунок 3.3 – успешный авторизация в системе

При успешном входе в систему появляется меню выбора действий, состоящее из 17 пунктов действия и пункта завершающего работу программы.

Ниже, на рисунках 3.4-3.14 представлены примеры работы нескольких пунктов данного меню.

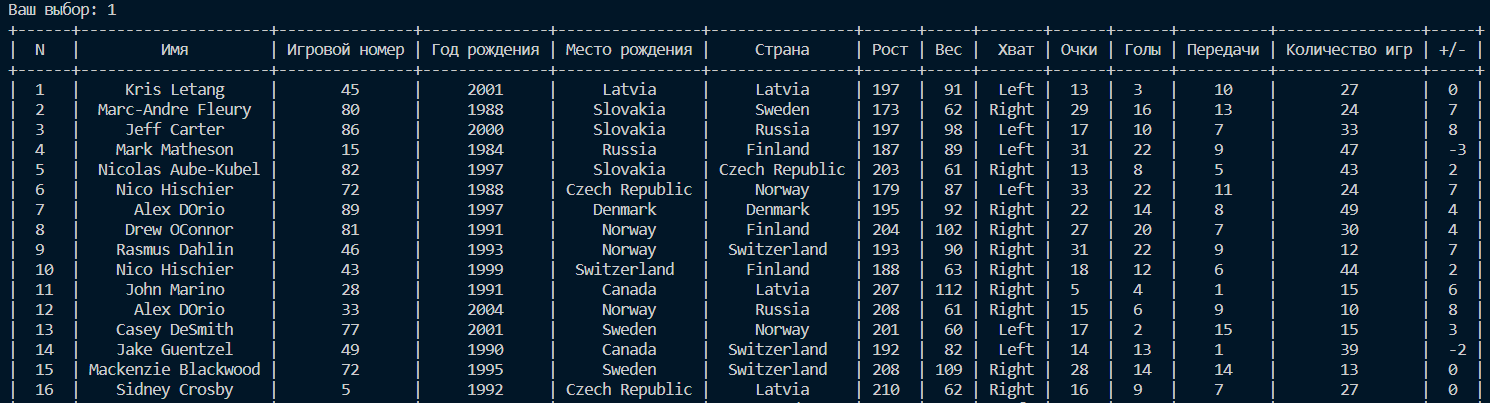


Рисунок 3.4 – фрагмент таблицы со статистикой всех игроков

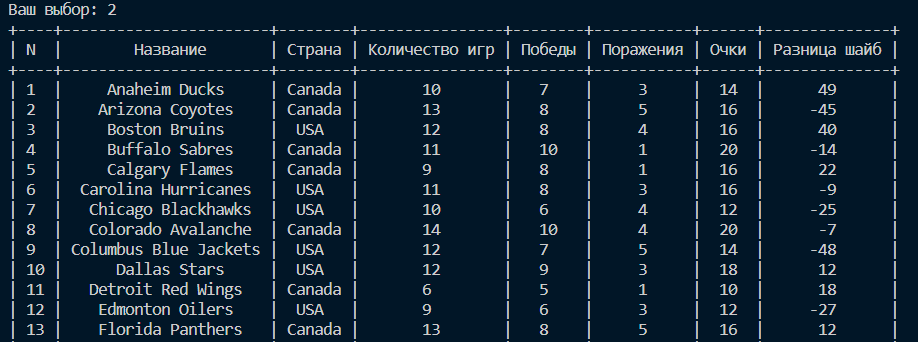


Рисунок 3.5 – фрагмент таблицы со статистикой всех команд

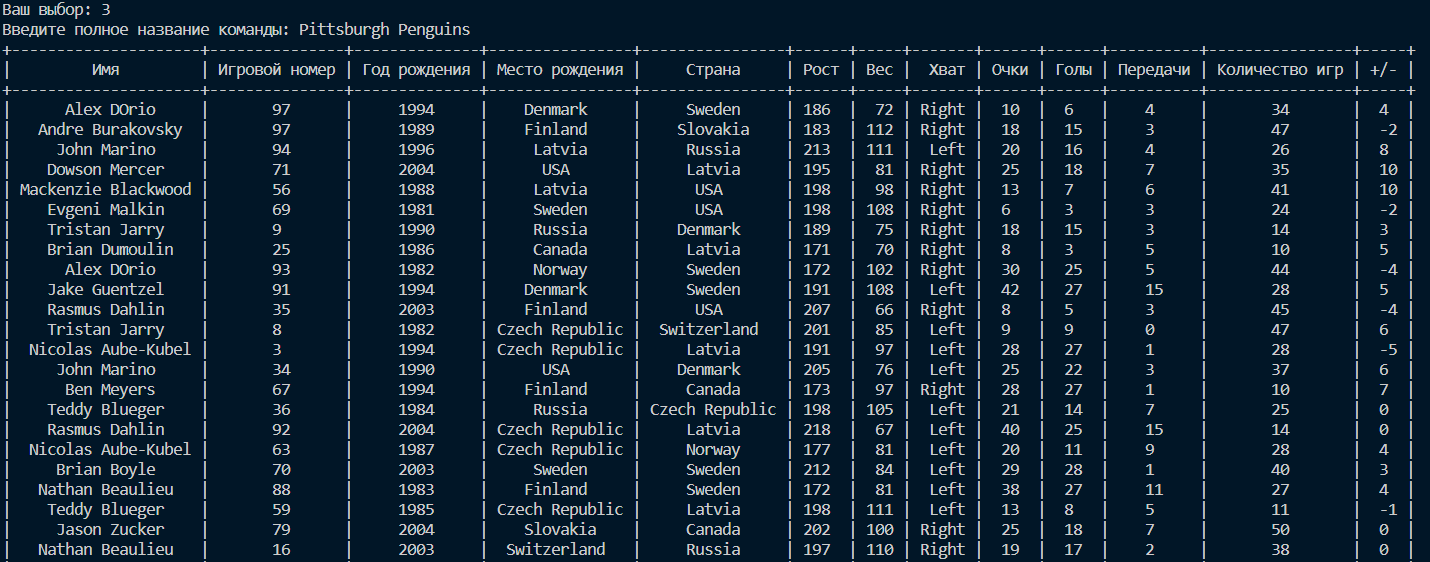


Рисунок 3.6 – фрагмент таблицы со статистикой всех игроков определенной команды

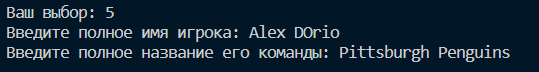


Рисунок 3.7 – добавление игрока в список избранного



Рисунок 3.8 – добавление команды в список избранного



Рисунок 3.9 – просмотр списка избранного

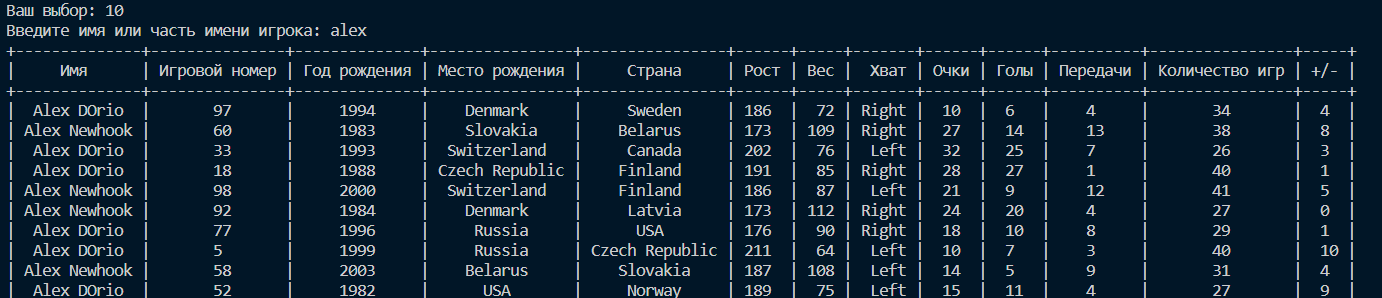


Рисунок 3.10 – фрагмент таблицы результата поиска игрока по имени

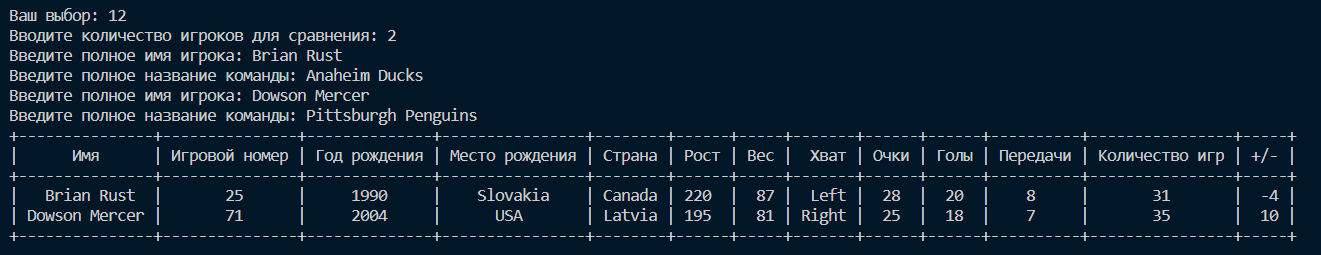


Рисунок 3.11 – сравнение игроков

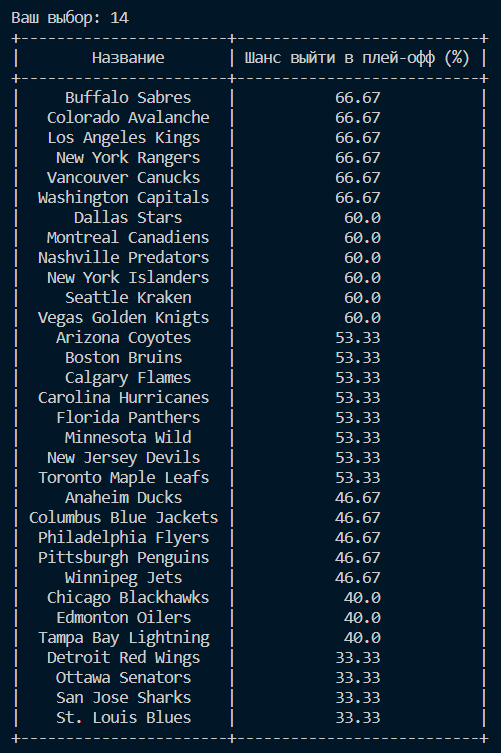


Рисунок 3.12 – пример результата анализа шансов команд на выход в плей-офф

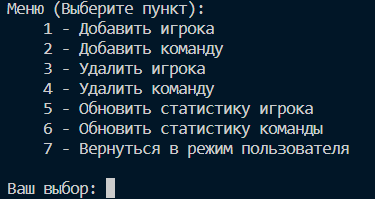


Рисунок 3.13 – возможности модератора

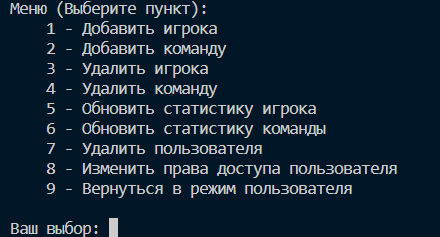


Рисунок 3.14 – возможности администратора

## 3.4 Выводы из технологического раздела

В данном разделе были выбраны средства разработки, такие как СУБД – PostgreSQL, язык программирования – Python, среда разработки – Visual Studio Code, а также обоснован выбор данных средств разработки. Были рассмотрены функции, которые должна выполнять программа и триггеры, используемые в программе. Также были предоставлены сведения об интерфейсе программы, процессе взаимодействия с приложением и примеры работы программы.

# 4 Экспериментальный раздел

## 4.1 Технические характеристики

В данном разделе будет проведен анализ зависимости шансов на выход в плей-офф команд Национальной Хоккейной Лиги от количества проведенных ими матчей (то есть от того, насколько процентов регулярный сезон уже завершен).

Ниже приведены технические характеристики устройства, на котором было проведено тестирование ПО:

* операционная система: Windows 10 64-bit;
* оперативная память: 16 GB;
* процессор: 2.6 GHz 6‑ядерный процессор Intel Core i7.

## 4.2 Анализ зависимости

Исходные данные представлены на рисунке 4.1. Для представленных данных регулярный сезон завершен на 15%, то есть команды сыграли в среднем 12 матчей из 86 возможных.

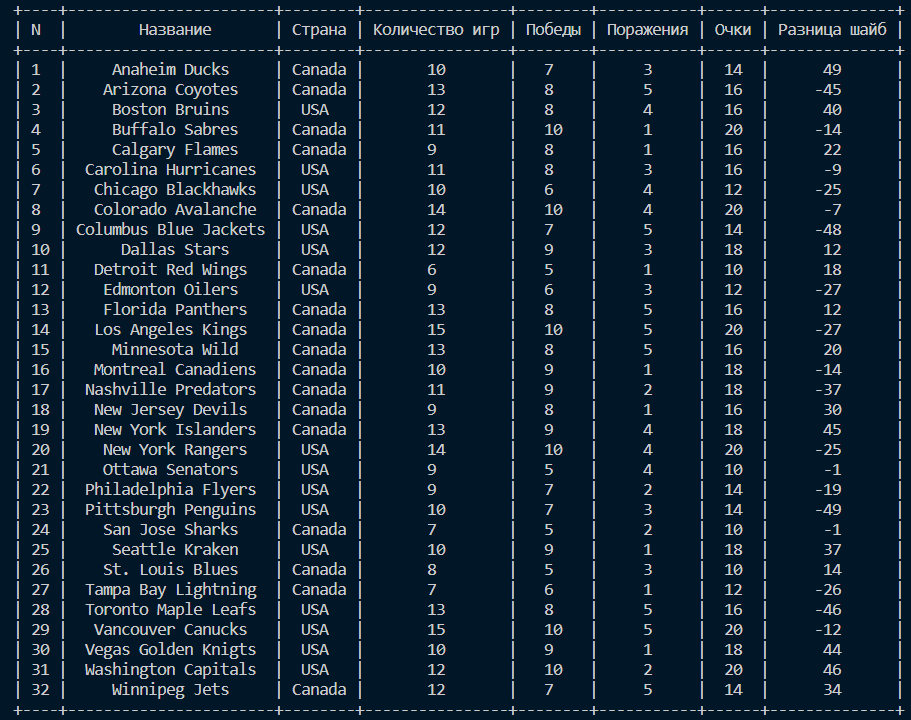


Рисунок 4.1 – исходные данные статистики команд

На рисунке 4.2 представлены текущие шансы всех команд на выход в плей-офф. При маленьком количестве матчей ненулевой шанс на выход в плей-офф есть у каждой команды, из-за чего невозможно определить какие команды действительно выйдут в плей-офф. Также шанс напрямую зависит от процента выигранных матчей.

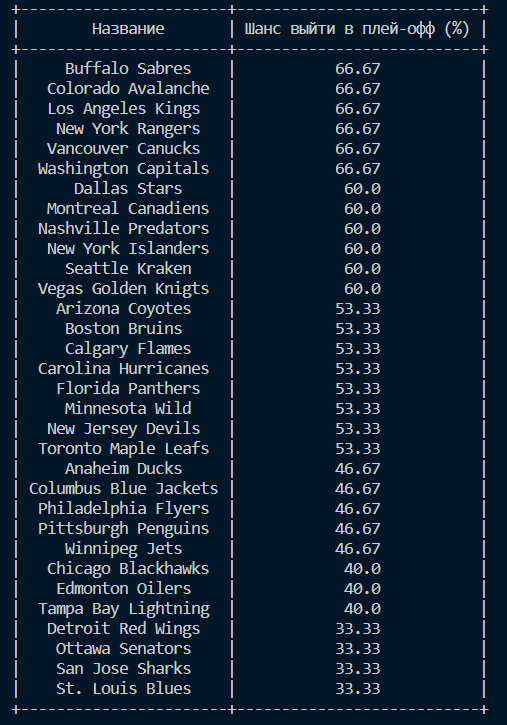


Рисунок 4.2 – шанс выйти в плей-офф у команд при 15% завершенности регулярного чемпионата

Следующие данные представлены для регулярного сезона, завершившегося на 25%, то есть команды сыграли в среднем 21 матч. Статистика команд представлена на рисунке 4.3.

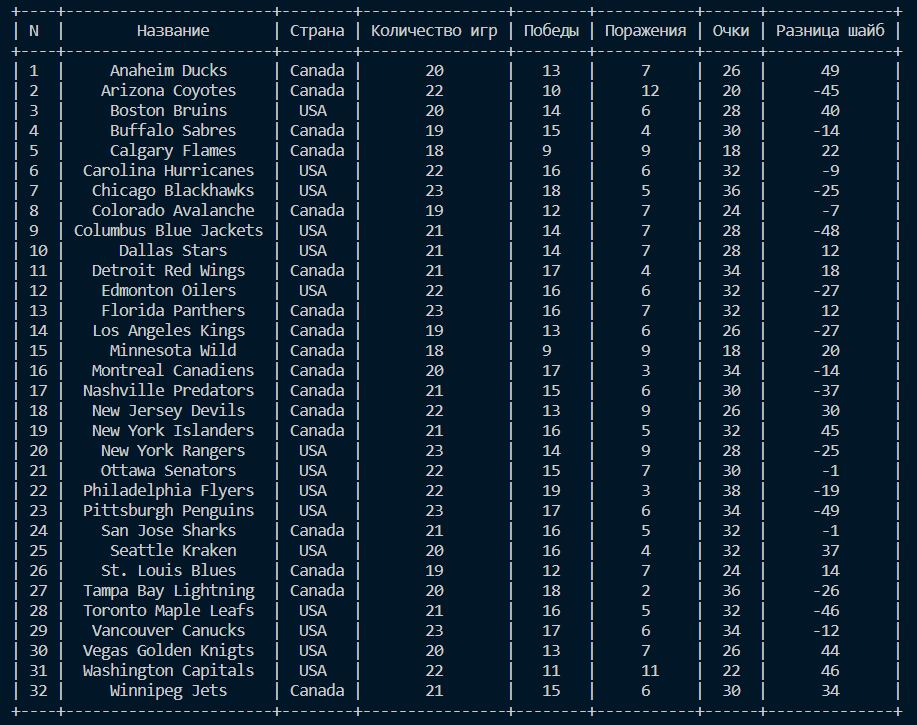


Рисунок 4.3 – данные о статистике команд для 25% завершенности регулярного сезона

На рисунке 4.4 представлены текущие шансы всех команд на выход в плей-офф при 25% завершенности сезона. Ненулевой шанс на выход в плей-офф все еще есть у каждой команды, из-за чего все также невозможно определить какие команды действительно выйдут в плей-офф, но уже сейчас вероятность выйти в плей-офф у команд с наибольшими шансами повышается.

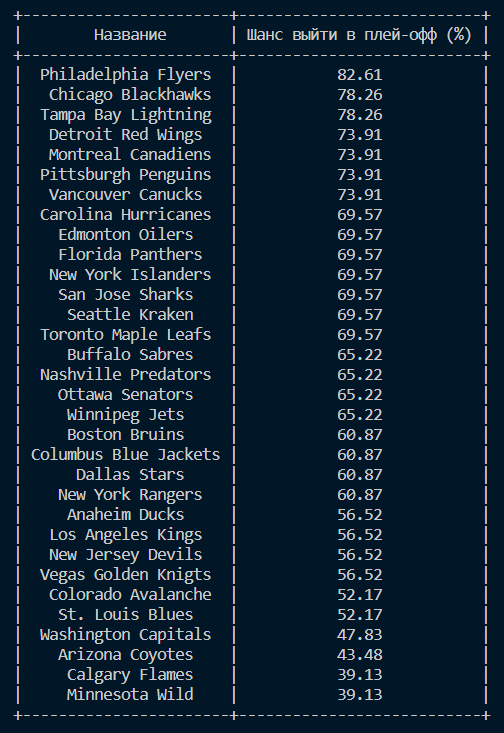


Рисунок 4.4 – шанс выйти в плей-офф у команд при 25% завершенности регулярного чемпионата

Следующие данные представлены для регулярного сезона, завершившегося на 50%, то есть команды сыграли в среднем 43 матча. Статистика команд представлена на рисунке 4.5.

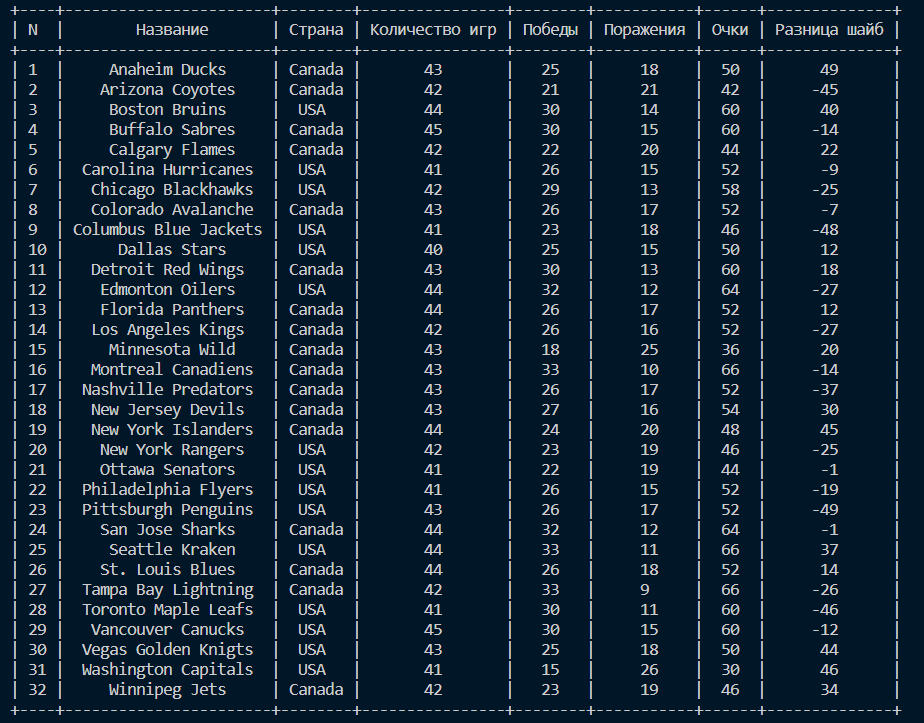


Рисунок 4.5 – данные о статистике команд для 50% завершенности регулярного сезона

На рисунке 4.6 представлены текущие шансы всех команд на выход в плей-офф при 50% завершенности сезона. Ненулевой шанс на выход в плей-офф все еще есть у каждой команды, из-за чего все также невозможно определить какие команды действительно выйдут в плей-офф, но уже сейчас вероятность выйти в плей-офф у команд с наибольшими шансами значительно повышается, так как количество оставшихся до конца сезона матчей уменьшается.

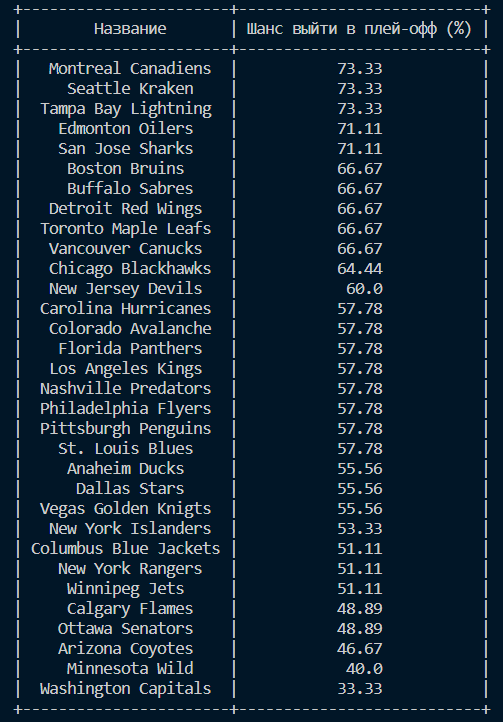


Рисунок 4.6 – шанс выйти в плей-офф у команд при 50% завершенности регулярного чемпионата

Следующие данные представлены для регулярного сезона, завершившегося на 75%, то есть команды сыграли в среднем 65 матчей. Статистика команд представлена на рисунке 4.7.

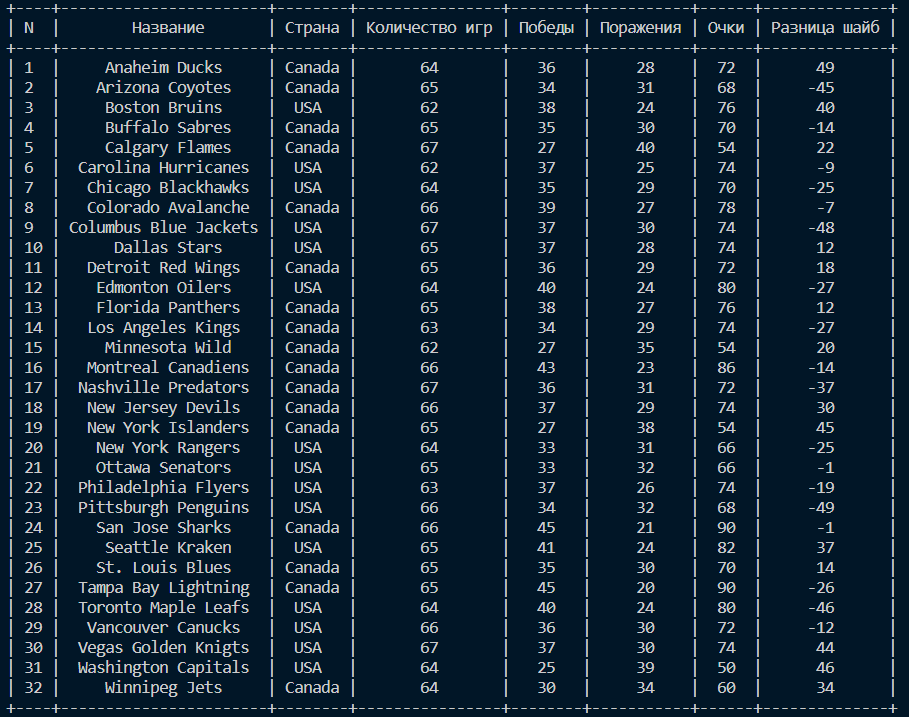


Рисунок 4.7 – данные о статистике команд для 75% завершенности регулярного сезона

На рисунке 4.8 представлены текущие шансы всех команд на выход в плей-офф при 75% завершенности сезона. Ненулевой шанс на выход в плей-офф все еще есть у каждой команды, из-за чего все также невозможно определить какие команды действительно выйдут в плей-офф, но уже сейчас вероятность выйти в плей-офф у команд с наименьшими шансами значительно уменьшается, так как количество оставшихся до конца сезона матчей слишком мало.

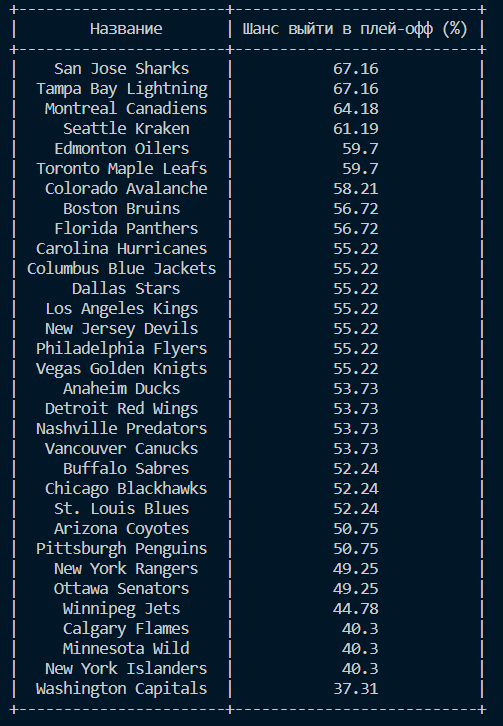


Рисунок 4.8 – шанс выйти в плей-офф у команд при 75% завершенности регулярного чемпионата

Следующие данные представлены для регулярного сезона, завершившегося на 90%, то есть команды сыграли в среднем 78 матчей. Статистика команд представлена на рисунке 4.9.

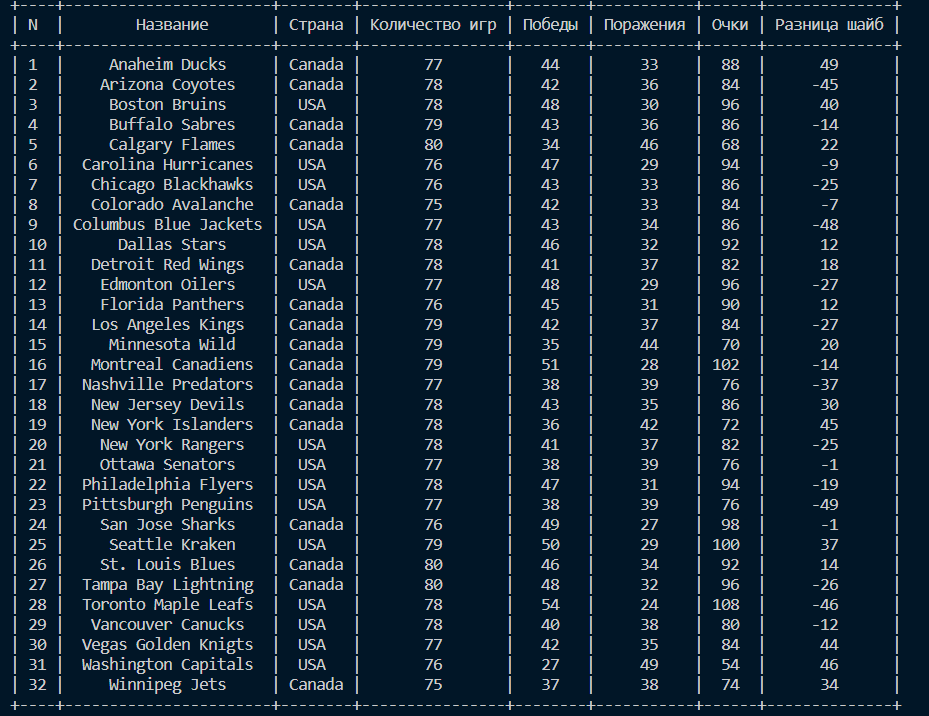


Рисунок 4.9 – данные о статистике команд для 90% завершенности регулярного сезона

На рисунке 4.10 представлены текущие шансы всех команд на выход в плей-офф при 90% завершенности сезона. Нулевой шанс на выход в плей-офф оказывается у команд, количество очков которых за оставшиеся 10% матчей не позволяет им пройти в 16 первых команд. Стопроцентный шанс выхода в плей-офф оказывается у команд, которые за оставшуюся часть сезона уже не смогут спуститься ниже 16-ой строчки по очкам.

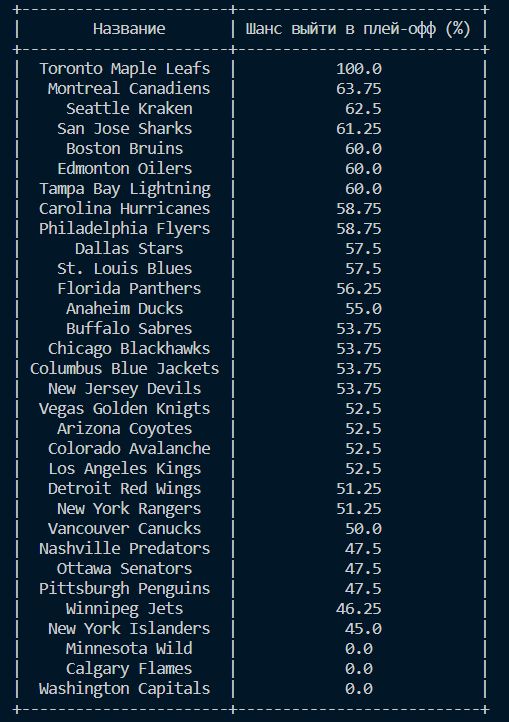


Рисунок 4.10 – шанс выйти в плей-офф у команд при 90% завершенности регулярного чемпионата

Следующие данные представлены для регулярного сезона, завершившегося полностью, то есть все команды сыграли все 86 матчей. Статистика команд представлена на рисунке 4.11.

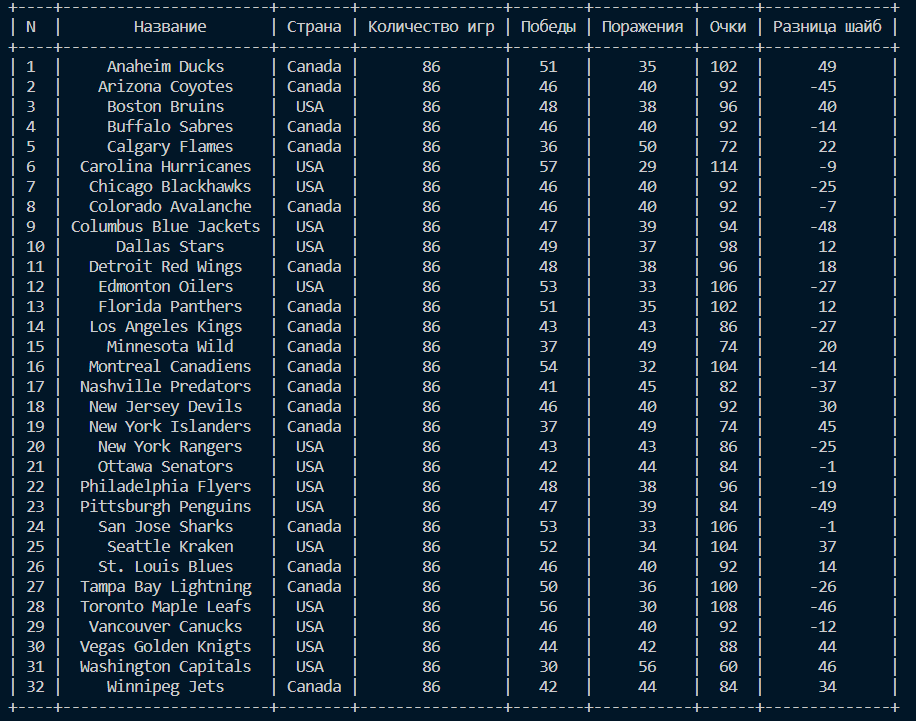


Рисунок 4.11 – данные о статистике команд для 100% завершенности регулярного сезона

На рисунке 4.12 представлены шансы всех команд на выход в плей-офф при полностью завершенном сезоне. Нулевой шанс на выход в плей-офф оказался у команд, количество очков которых не позволило им пройти в 16 первых команд. Стопроцентный шанс выхода в плей-офф у команд, которые по очкам находятся первых 16-ти строчках. Таким образом, данная таблица показывает команды, которые вышли в плей-офф.

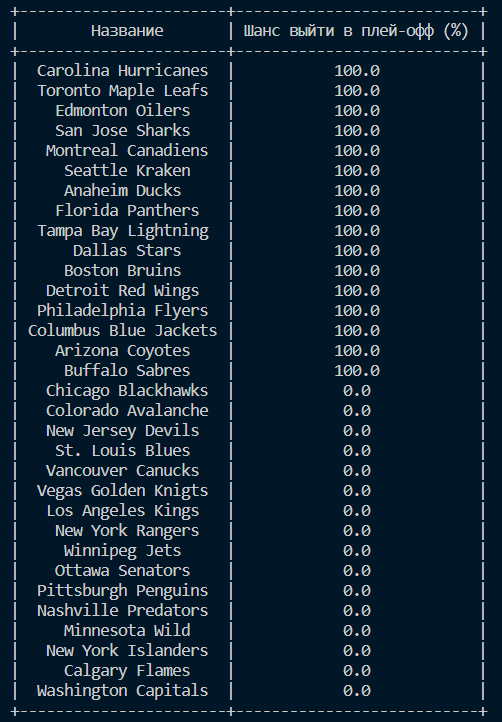


Рисунок 4.12 – шанс выйти в плей-офф у команд при 100% завершенности регулярного чемпионата

## 4.2 Выводы из экспериментального раздела

Проанализировав результаты, можно сделать вывод, что процент завершенности регулярного сезона сильно влияет на шанс выйти в плей-офф менее успешных команд и слабо влияет на шанс выйти в плей-офф у более успешных. Также с увеличением количества сыгранных матчей шансы у ряда команд уже становятся стопроцентами или же нулевыми еще до окончания сезона и становится возможным понять, кто действительно выйдет в плей-офф, а кто нет.

# ЗАКЛЮЧЕНИЕ

Во время выполнения курсового проекта были рассмотрены существующие виды СУБД, описана структура базы данных и приложения. Были разработаны база данных и программное обеспечение, предоставляющее к ней интерфейс.

Программа реализована таким образом, что пользователь может получать информацию о статистике команд и хоккеистов Национальной Хоккейной Лиги, сравнивать эту статистику, осуществлять поиск хоккеистов и команд, создавать список избранных хоккеистов и команд, просматривать шансы команд на выход в плей-офф в текущий момент; модератор может добавлять и удалять хоккеистов и команды, а также обновлять их статистику; администратор может изменять права пользователей и удалять их.

В ходе выполнения поставленной задачи были изучены возможности языка Python, получен опыт работы с PostgreSQL и получены знания в области баз данных.

Цель работы достигнута, выполнены следующие задачи:

1) формализовано задание, определен необходимый функционал;

2) проведен анализ СУБД;

3) описана структуру базы данных, включая объекты, из которых она состоит;

4) спроектировано приложение для доступа к БД;

5) создана и заполнена БД;

6) разработано программное обеспечение, которое позволит пользователю получать, искать и сравнивать статистику игроков и команд Национальной Хоккейной Лиги;

7) исследована зависимость шанса на выход в плей-офф команд от их статистики в регулярном чемпионате и количестве сыгранных матчей.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. NHL. [Электронный ресурс]. Режим доступа: <https://www.nhl.com/> (дата обращения 21.06.2022)
2. Flashscore. [Электронный ресурс]. Режим доступа: <https://www.flashscore.ru.com/> (дата обращения 21.06.2022)
3. EliteProspects. [Электронный ресурс]. Режим доступа: <https://www.eliteprospects.com/> (дата обращения 21.06.2022)
4. ISO/IEC TR 10032:2003 Information technology — Reference model of data management
5. Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: «Вильямс», 2006.
6. Лазицкас Е.А., Загумённикова И.Н., Гилевский П.Г. Базы данных и системы управления базами данных. – 2-е изд. – Минск : Республиканский институт профессионального образования (РИПО), 2018. — 268 c.
7. Бородина А. И. Реляционная модель данных [Электронный ресурс]. Режим доступа: <http://www.bseu.by/it/tohod/lekcii2_3.htm> (дата обращения 21.06.2022)
8. PostgreSQL: Документация. [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/> (дата обращения: 22.06.2022)
9. Python: Документация. [Электронный ресурс]. Режим доступа: <https://docs.python.org/3/> (дата обращения: 22.06.2022)
10. Visual Studio Code: Документация. [Электронный ресурс]. Режим доступа: <https://code.visualstudio.com/docs> (дата обращения 22.06.2022)

## ПРИЛОЖЕНИЕ А