

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,  
Факультет комп'ютерних наук і технологій  
(повне найменування інституту, факультету)

Кафедра програмних засобів  
(повне найменування кафедри)

## Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ  
ПРОГНОЗУВАННЯ ІНДИВІДУАЛЬНОЇ СТАТИСТИКИ ФУТБОЛІСТІВ  
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR  
PREDICTING A FOOTBALL PLAYER STATISTICS

Виконав: студент(ка) 2 курсу, групи КНТ-120м  
Спеціальності 121 Інженерія програмного  
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Мазалов Д.О.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) \_\_\_\_\_  
Слайди презентації \_\_\_\_\_

## 6. Консультанти розділів проєкту (роботи)

| Розділ              | Прізвище, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|-------------------------------------------|----------------|---------------------------|
|                     |                                           | завдання видав | прийняв виконане завдання |
| 1-5 Основна частина | Скрупський С.Ю., доцент                   |                |                           |
| Нормоконтролер      | Камінська Ж.К., асистент                  |                |                           |
|                     |                                           |                |                           |
|                     |                                           |                |                           |

7. Дата видачі завдання “ 7 ” вересня 2021 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)             | Строк виконання етапів проєкту ( роботи ) | Примітка             |
|-------|------------------------------------------------------|-------------------------------------------|----------------------|
| 1     | Аналіз та дослідження предметної області             | 1-2 тиждень                               |                      |
| 2     | Розробка та затвердження технічного завдання         | 3 тиждень                                 | Технічне завдання    |
| 3     | Дослідження методів розв’язання проблеми             | 4 тиждень                                 |                      |
| 4     | Проектування програмного забезпечення                | 5 тиждень                                 |                      |
| 5     | Розроблення модулів програми                         | 6-8 тиждень                               |                      |
| 6     | Розроблення програмного коду                         | 9-10 тиждень                              | Текст програми       |
| 7     | Тестування та відлагодження програмного забезпечення | 11 тиждень                                | Програма             |
| 8     | Розробка розділів пояснювальної записки              | 12-14 тиждень                             | Пояснювальна записка |
| 9     | Розробка графічного матеріалу                        | 15 тиждень                                | Графічний матеріал   |
| 10    | Захист магістерської роботи                          | 16 тиждень                                |                      |
|       |                                                      |                                           |                      |
|       |                                                      |                                           |                      |
|       |                                                      |                                           |                      |

Студент(ка)

\_\_\_\_\_ Мазалов Д.О.  
( підпис ) (прізвище та ініціали)

Керівник проєкту (роботи)

\_\_\_\_\_ Скрупський С.Ю.  
( підпис ) (прізвище та ініціали)

## РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 93 с., 25 рис., 3 табл., 3 дод., 19 джерел.

ФУТБОЛІСТ, ІНДИВІДУАЛЬНА СТАТИСТИКА,  
ПРОГНОЗУВАННЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ.

Об'єкт роботи – процес визначення футбольної статистики.

Предмет роботи – методи прогнозування індивідуальної статистики футболістів.

Мета роботи – розроблення програмного забезпечення накопичення футбольної статистики з можливістю прогнозування для підвищення рівня обґрунтованості прийняття рішень щодо оптимального вибору складу футбольної команди.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Python, бібліотека Tensorflow, система MySQL.

Результати. Науковою новизною роботи є запропонований метод прогнозування індивідуальної статистики футболістів, який характеризується використанням моделі на основі нейронної мережі з повнозв'язними шарами та вентильними рекурентними вузлами, що дозволяє виконувати прогнозування кількості голів, результативних передач та відсотку успішних передач, виконаних футболістом у грі.

Висновки. Розроблений вебзастосунок дозволяє вносити дані про індивідуальні та командні результати ігор, переглядати статистику за гравцями і за командою за окремою грою або сезоном, отримувати рейтинг гравців за різними показниками індивідуальної статистики, прогнозувати результати виступу гравців у наступних іграх.

Галузь використання – платформи накопичення індивідуальної статистики футболістів.

## ABSTRACT

Explanatory note to the qualifying work of the master: 93 pages, 25 figures, 3 tables, 3 appendixes, 19 sources.

FOOTBALL PLAYER, INDIVIDUAL STATISTICS, FORECASTING, MACHINE LEARNING, NEURAL NETWORKS.

The object of work is the process of determining football statistics.

Subject of the work is methods of forecasting individual statistics of football players.

The purpose of the work is to develop software for the accumulation of football statistics with the ability to predict individual football statistics to increase the level of validity of decisions on the optimal choice of football team.

Materials, methods and technical means: object-oriented programming, Python programming language, Tensorflow library, MySQL system.

Results. The scientific novelty of the work is the proposed method of forecasting individual player statistics, which is characterized by the use of a neural network model with fully connected layers and gated recurrent units, which enables predicting the number of goals, assists and percentage of successful assists in the game.

Conclusions. The developed web application enables users to enter data on individual and team results of games, view statistics by players and team statistics by game or season, get a rating of players on various indicators of individual statistics, predict the performance of players in the future games.

Field of use is platforms for the accumulation of individual statistics of football players.

## ЗМІСТ

|                                                                           |    |
|---------------------------------------------------------------------------|----|
|                                                                           | С. |
| Перелік скорочень та умовних познач.....                                  | 8  |
| Вступ.....                                                                | 9  |
| 1 Аналіз проблеми та постановка завдань дослідження.....                  | 10 |
| 1.1 Використання інформаційних технологій в футболі .....                 | 10 |
| 1.2 Огляд програмних аналогів .....                                       | 11 |
| 1.3 Визначення особливостей програми, що розробляється .....              | 14 |
| 1.4 Визначення сутностей предметної області .....                         | 15 |
| 1.5 Висновки за розділом 1 та постановка завдань дослідження.....         | 16 |
| 2 Матеріали і методи .....                                                | 18 |
| 2.1 Методи прогнозування в футбольній сфері .....                         | 18 |
| 2.2 Модель рекурентної нейронної мережі gated recurrent unit.....         | 19 |
| 2.3 Метод прогнозування індивідуальної статистики футболістів.....        | 21 |
| 2.4 Висновки за розділом 2 .....                                          | 24 |
| 3 Проєктування програмного забезпечення.....                              | 25 |
| 3.1 Вибір мови програмування .....                                        | 25 |
| 3.2 Моделювання роботи користувачів з програмою .....                     | 26 |
| 3.3 Проєктування бази даних.....                                          | 28 |
| 3.4 Висновки за розділом 3 .....                                          | 42 |
| 4 Основні рішення щодо реалізації компонентів системи.....                | 43 |
| 4.1 Структура програмної системи .....                                    | 43 |
| 4.2 Опис модулів реалізованої програмної системи .....                    | 44 |
| 4.3 Висновки за розділом 4 .....                                          | 47 |
| 5 Експлуатація, тестування та експериментальне дослідження програми ..... | 48 |
| 5.1 Призначення програми.....                                             | 48 |
| 5.2 Умови виконання програми.....                                         | 48 |
| 5.3 Виконання програми .....                                              | 48 |
| 5.4 Експериментальне дослідження.....                                     | 52 |
| 5.5 Висновки за розділом 5 .....                                          | 53 |

|                                    |    |
|------------------------------------|----|
| Висновки.....                      | 54 |
| Перелік джерел посилання.....      | 56 |
| Додаток А Технічне завдання.....   | 59 |
| Додаток Б Текст програми .....     | 63 |
| Додаток В Слайди презентації ..... | 86 |

## **ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК**

GRU – gated recurrent unit;

ПЗ – програмне забезпечення.

## ВСТУП

Футбол є одним з видів спорту, що протягом довгого часу достатньо повільно розвивався відносно використання інформаційних технологій. Проте протягом останнього періоду цей процес почав активно розвиватися.

У майбутньому футбол може виглядати зовсім інакше, оскільки нові технології стають все більш важливою частиною спорту: основна гра не зміниться, але допомога від технологій може вплинути як на гравців, так і покращити досвід шанувальників, при цьому футбол пройшов довгий шлях з моменту створення перших універсальних правил майже 160 років тому, і оскільки суспільство все більше переплітається з технологіями, у футболу не залишиться іншого вибору, окрім як прийняти технологію [1]-[2].

На даний момент інформаційні технології стали просякати в різні футбольні процеси, що стосуються безпосередньо самої гри, тренувань, слідування за виступами гравців. Майже кожен шанувальник футболу використовує вебсайти для пошуку інформації про футбольні новини, футбольну статистику. Багато шанувальників роблять ставки на футбол, тому потребують додаткової інформації про гру для прийняття рішень. Така інформація може отримуватися як за історією ігор, так і за результатами прогнозування на майбутнє.

У роботі потрібно досягти її мету, що полягає в розробленні програмного забезпечення накопичення футбольної статистики з можливістю прогнозування для підвищення рівня обґрунтованості прийняття рішень щодо оптимального вибору складу футбольної команди.

# **1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ**

## **1.1 Використання інформаційних технологій в футболі**

До завдань у футболі, в яких зараз активно використовуються інформаційні технології, можна віднести:

- тренування: технології використовуються, щоб покращити гравців, піднявши їх на більш високий рівень, ніж будь-коли раніше, при цьому застосовуються монітори серцевого ритму, GPS-трекери, передові системи моніторингу та камери, програми для відстеження офіційних ігор і тренувань, записаних за допомогою дронів, такі технології як розумні трикотажні вироби adidas використовуються для відстеження дистанції, швидкості та пульсу, можуть допомогти оптимізувати графік тренувань, розробити ігрові стратегії і навіть допомогти виявити закономірності або зміни в діяльності гравців, які можуть свідчити про майбутню травму;

- передбачення певних травм до того, як вони відбудуться: зараз фахівці в Німеччині розробляють роботизований пристрій для важкої атлетики, щоб допомогти спортсменам покращити продуктивність і скоротити час відновлення після травми, він м'який до суглобів, зберігає силу м'язів, а також допомагає запобігти травмам, тренувальні вправи можна навіть адаптувати для окремих гравців, зберігати в хмарі та отримати доступ до них на будь-якій машині;

- прогнозування травм: штучний інтелект дозволяє отримувати інформацію про те, як підтримувати гравця в найкращому стані або передбачати, коли може статися травма, існуючі системи надають зелені, жовті та червоні індикатори щоденного рівня ризику гравця, що дає тренерам уявлення про те, чи потрібно знижувати інтенсивність тренувань;

- моніторинг здоров'я футболіста: штучний інтелект може автоматизувати діагностику серцевих аритмій, проводити дистанційну діагностику та раннє виявлення;

- внутрішньоігрові технології: технологія визначення перетину лінії

воріт зараз є добре закріпленим елементом сучасного футболу, з камерами, які надзвичайно точно здатні визначити, чи перетнув м'яч лінію воріт і чи слід вважати це голом, чи ні [1]-[4].

## 1.2 Огляд програмних аналогів

Далеко не всі вебсайти футбольної статистики пропонують надання інформації про індивідуальну статистику виступу футболістів у іграх, а тим більше деталізованої. Одним з найвідоміших вебсайтів футбольної статистики є whoscored (рис. 1.1) [5]. Він надає статистичні дані по окремим матчам і сезонам загалом за футболістами та за командами.

The screenshot displays the Whoscored website interface. At the top, there are navigation tabs: Live Scores, Statistics, Previews, News, Top Facts, and Comparison. Below these, a 'Detailed Tournaments' section lists various leagues like Premier League, Serie A, LaLiga, etc. The main content area shows the profile of Ilkay Gündogan, including his photo, name, shirt number (8), height (180cm), and current team (Manchester City). Below the profile, there is a 'Current Participations' table with tabs for Summary, Defensive, Offensive, Passing, and Detailed. The table lists various tournaments and matches with columns for Apps, Mins, Goals, Assists, Yel, Red, SpG, PS%, AerialsWon, MotM, and Rating.

| Tournament                    | Apps      | Mins        | Goals     | Assists  | Yel      | Red      | SpG        | PS%         | AerialsWon | MotM     | Rating      |
|-------------------------------|-----------|-------------|-----------|----------|----------|----------|------------|-------------|------------|----------|-------------|
| FIFA World Cup Germany        | 7(1)      | 586         | 5         | -        | -        | -        | 2.6        | 89.3        | 0.3        | 2        | 7.81        |
| Premier League                | 11        | 897         | 3         | 1        | 1        | -        | 2.5        | 90.1        | 0.3        | 1        | 7.43        |
| League Cup                    | 1         | 121         | -         | -        | -        | -        | 3          | 94.4        | -          | -        | 7.12        |
| UEFA Nations League A Germany | 4         | 344         | 1         | -        | -        | -        | 0.5        | 93.3        | 0.3        | -        | 6.82        |
| Champions League              | 3(2)      | 334         | -         | 2        | -        | -        | 1.4        | 88.8        | 0.6        | -        | 6.73        |
| European Championship Germany | 3         | 221         | -         | -        | 1        | -        | 0.3        | 93.5        | -          | -        | 6.33        |
| Community Shield              | 1         | N/A         | -         | N/A      | -        | -        | N/A        | N/A         | N/A        | N/A      | -           |
| Club Friendlies               | 1(1)      | N/A         | 2         | N/A      | -        | -        | N/A        | N/A         | N/A        | N/A      | -           |
| Int. Friendly Germany         | 1         | N/A         | -         | N/A      | -        | -        | N/A        | N/A         | N/A        | N/A      | -           |
| Int. Friendly Germany         | 1         | N/A         | 1         | N/A      | 1        | -        | N/A        | N/A         | N/A        | N/A      | -           |
| <b>Total / Average</b>        | <b>37</b> | <b>2503</b> | <b>12</b> | <b>3</b> | <b>3</b> | <b>0</b> | <b>1.9</b> | <b>90.6</b> | <b>0.3</b> | <b>3</b> | <b>7.23</b> |

Mins: Minutes played  
 Yel: Yellow card  
 PS%: Pass success percentage  
 Goals: Total goals  
 Red: Red card  
 AerialsWon: Aerial duels won per game  
 Assists: Total assists  
 SpG: Shots per game  
 MotM: Man of the match

Рисунок 1.1 – Вебсайт whoscored

Програма дозволяє слідкувати за ходом ігор між командами в межах різних турнірів. При цьому відображаються не тільки поточні результати, але і деталізована статистика за командою і за кожним гравцем. Саме ці інструменти зробити даний вебсайт на стільки популярним у футбольному світі.

Проте сучасні інформаційні технології в світі футболу обмежуються далеко не тільки статистикою. Існують окремі вебсайти, які надають механізми прогнозування результатів футбольних матчів (рис. 1.2). Таким застосунком є NVTips [6].

#### NVtips.com - Free football statistics and predictions with artificial intelligence

Choose a day:

2021-12-10-FRIDAY

TODAY

2021-12-12-SUNDAY

Choose a country:

All

#### Tips for: 2021-12-4

\*Tip: Click on the row for more details

Search:

|  | Country | League  | Home | Away  | Tip | Avg Odds | Tip Handicap | Tip CS | Tip HT/FT | Tip Over/Under Goals | Tip Both To Score | Result FT |
|--|---------|---------|------|-------|-----|----------|--------------|--------|-----------|----------------------|-------------------|-----------|
|  | Italy   | Serie A | Roma | Inter | X   | 3.67     | Roma 0.5     |        | X / X     | Over 2.5             | No                | 0-3       |

Date: 2021-12-04

Time: 17:00

Tip Number of goals scored: 2.87

Home win probability: Roma 37 %

Draw probability: 40 %

Away win probability: Inter 23 %

Home Odds: Roma 3.77

Draw Odds: 3.67

Away Odds: Inter 1.98

Home Attack Strength: Roma 1.35

Away Defense Strength: Inter 0.83

Away Attack Strength: Inter 1.36

Рисунок 1.2 – Вебсайт NVtips.com

Вебсайт NVTips дозволяє для кожної гри отримувати інформацію про

прогнозовану ймовірність перемоги кожної команди, нічиєї, ймовірність пропущених м'ячів зі сторони кожної команди, ймовірність забитих голів, ймовірність відповідних результатів після першої половини гри та після завершення гри загалом. Ймовірність більшої кількості порушень правил, створених нагод, жовтих, червоних карток відповідною стороною гри. Тобто все це командні показники.

Вебсайт bettingStats (рис. 1.3) також дозволяє виконати прогнозування, але дещо в іншому ключі [7].

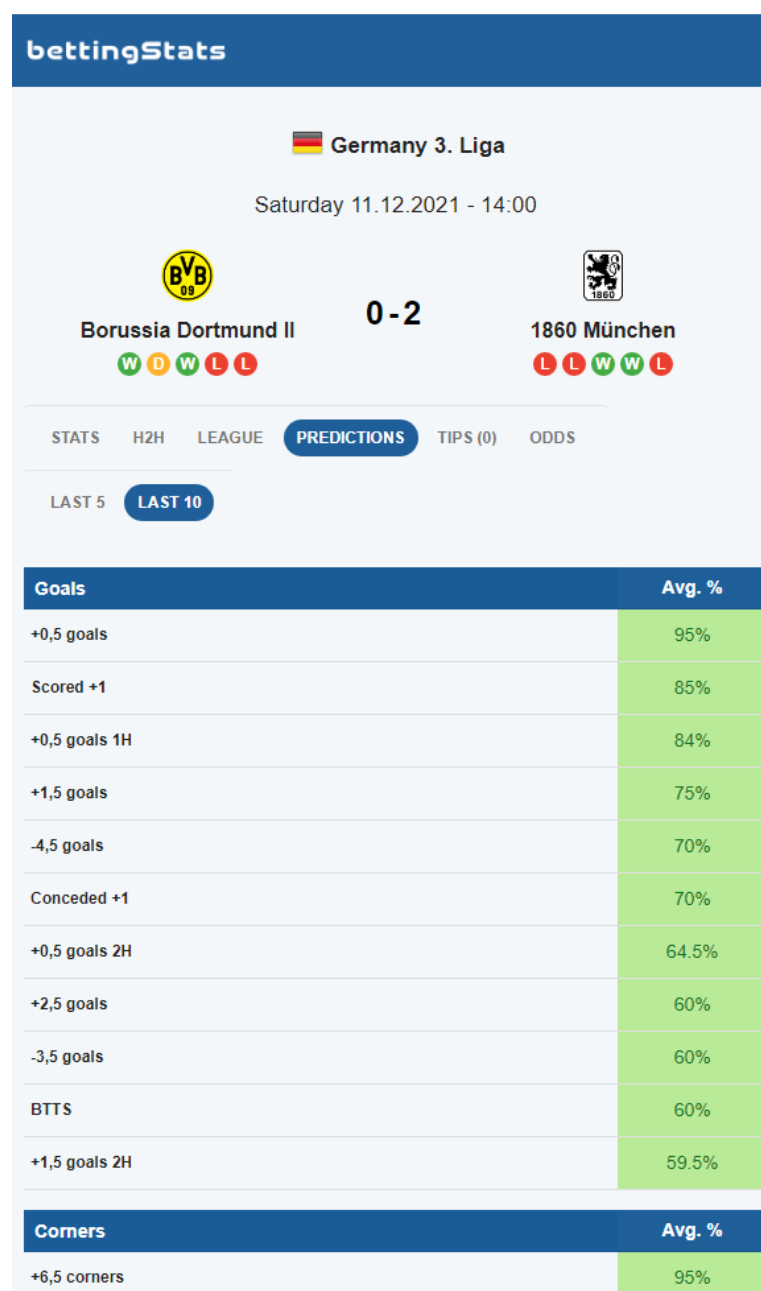


Рисунок 1.3 – Вебсайт bettingStats

Прогнозування в програмі також стосується саме командних показників. До наявних командних показників належать різниця забитих і пропущених м'ячів, наявність забитих або пропущених м'ячів, різниця в кількості кутових, отриманих карток за порушення правил. Варто звернути увагу, що всі прогнозовані показники це стандартні сценарії виконання ставок на ігри.

Також слід відзначити, що засоби прогнозування більше направлені на підтримку визначення стандартних ставок щодо результатів змагань. Проте навіть ті же ставки робляться також і на індивідуальні виступу футболістів також, але такі інструменти не представлені в сучасних програмних засобах.

### **1.3 Визначення особливостей програми, що розробляється**

Результати проведеного аналізу вказують на те, що зазвичай інструменти, пов'язані з прогнозуванням результатів, і інструменти, пов'язані зі статистикою виступу футболістів, представлені в різних вебзастосунках. Не зважаючи на логічну пов'язаність цих даних, до того ж актуальність під час прийняття рішень використання не тільки самих прогнозованих значень, але і історії виступів, для цього на даний момент доводиться використовувати різні рішення.

Окрім того інструменти прогнозування в основному направлені на галузь ставок, а тому включають засоби прогнозування результатів матчів і пов'язаних показників. Індивідуальна статистика, яка безпосередньо впливає і на виступ всієї команди (деякі показники, наприклад, кількість обведень, виконаних футболістом, можуть не впливати на стан команди загалом і не виходити з цього стану, але інші, як то захисні дії, впливають на результат виступу команди), на даний момент не прогнозується, а тому ця задача потребує вирішення.

Порівняння проаналізованих програмних засобів з розробкою, яка виконується в роботі, представлено в табл. 1.1.

Таблиця 1.1 – Аналіз виконуваної розробки та її програмних аналогів

| Характеристика                                               | NVtips | Whoscored | Дипломна розробка |
|--------------------------------------------------------------|--------|-----------|-------------------|
| Прогнозування командних результатів                          | +      | –         | –                 |
| Індивідуальна статистика виступу футболістів                 | –      | +         | +                 |
| Прогнозування індивідуальних результатів виступу футболістів | –      | –         | +                 |

#### 1.4 Визначення сутностей предметної області

Проведений аналіз предметної області, визначені функціональні особливості дозволили визначити основні сутності, з якими відбувається робота.

Сутність *Footballer* є однією з базових, визначає футболіста, який бере участь в турнірах

Сутність *Championship* визначає турнір (чемпіонат), в межах відбуваються ігри між футбольними клубами.

Сутність *ChampionshipSeason* визначає конкретний сезон проведення турніру.

Сутність *Club* визначає футбольний клуб, який бере участь в турнірі, представлений відповідною командою.

Сутність *City* визначає місто, яке представляє футбольний клуб.

Сутність *Country* визначає країну, що представляє національність футболіста або судді.

Сутність *Game* визначає конкретну гру команд в межах певного турніра, в якій беруть участь футболісти.

Сутність `FootballerGame` визначає участь футболіста у грі та статистику цієї участі.

Сутність `FootballerSeason` визначає статистику участі футболіста в іграх певного турніру протягом всього сезону.

Сутність `ClubGame` визначає виступ футбольного клубу в конкретній грі.

Сутність `ClubChampionshipSeason` визначає статистику участі футбольного клубу в сезоні чемпіонату.

Сутність `GameSubstitution` визначає виконану заміну під час гри.

Сутність `Goal` визначає забитий гол під час гри команд.

Сутність `Position` визначає наявні позиції, на яких можуть розташовуватися футболісти на полі.

Сутність `Referee` визначає суддю гри чемпіонату.

Сутність `Stage` визначає доступний структурну одиницю будь-якого чемпіонату, фактично етап чемпіонату.

### **1.5 Висновки за розділом 1 та постановка завдань дослідження**

Проведено аналіз завдань у футбольній сфері, в яких можуть використовуватися інформаційні технології. Виконано огляд програмних аналогів, зокрема `whoscored`, `NVtips.com`, `bettingStats`. Програма, що розробляється, відрізняється від них тим, що об'єднує в собі засоби накопичення індивідуальної футбольної статистики та прогнозування, при чому прогнозування індивідуальної статистики, а не командної. Визначено основні сутності предметної області.

Завданнями дипломної роботи є:

- проаналізувати предметну область на основі існуючих програмних засобів;

- створити метод прогнозування індивідуальної статистики футболістів;

- виконати проєктування ПЗ;
- реалізувати ПЗ;
- виконати експериментальне дослідження проблеми прогнозування індивідуальної статистики футболістів.

## 2 МАТЕРІАЛИ І МЕТОДИ

### 2.1 Методи прогнозування в футбольній сфері

За результатами проведеного аналізу інтелектуальних методів прогнозування, які застосовуються в футбольній сфері, було виділено наступні існуючі рішення.

У роботі «Прогнозування результатів футбольних матчів з використанням байєсівських мереж для англійської прем'єр-ліги» розглянуто три типи підходи для прогнозування результатів футбольних матчів, які включають статистичні підходи, машинне навчання та байєсівські підходи, в результаті чого було запропоновано байєсівську мережу для прогнозування результатів футбольних матчів у розрізі: перемога господарів, перемога виїзної команди або нічия [8].

У роботі «Прогноз на футбольний матч використовуючи атрибути гравців» прогнозування ігор відбувалось на основі характеристик, виділених для гравців з відеогри FIFA, використовуючи модель на основі нейронної мережі довгострокової пам'яті [9].

У роботі «Прогнозування рейтингів матчів футболістів за допомогою машинного навчання» досліджено вплив характеристик футболіста на його рейтинг. Рейтинг визначено в категорійній системі. В залежності від певних виконаних дій встановлюється відповідне значення рейтингу:

- 6 балів, якщо футболіст виконав хоча би одну дію, що включає голи, результативні передачі, ключові передачі, створені можливості;
- 2 бали, якщо футболіст отримав жовту і червону картку;
- 3 бали, якщо футболіст отримав тільки червону картку;
- 4 бали, якщо футболіст отримав жовту картку;
- 5 балів, якщо футболіст не був попереджений і не мав значного впливу [10].

Прогнозування рейтингу виконувалось за допомогою методу опорних векторів, методу регресії XGBoost.

Слід відзначити, що незважаючи на активне використання інтелектуальних методів у футбольній сфері, в більшій мірі вони зводяться до прогнозування результатів футбольних матчів.

## **2.2 Модель рекурентної нейронної мережі gated recurrent unit**

Рекурентні нейронні мережі було створено, оскільки нейронні мережі прямого поширення сигналу характеризуються наступними недоліками:

- неможливо обробляти послідовні дані;
- враховує лише поточний вхід;
- неможливо запам'ятати попередні введення [11].

Архітектурою нейронної мережі, яка дозволила вирішити ці проблеми, є рекурентні нейронні мережі.

Рекурентні нейронні мережі можуть обробляти послідовні дані, приймаючи поточні вхідні дані та раніше отримані вхідні дані, вони можуть запам'ятовувати попередні введення завдяки своїй внутрішній внутрішній пам'яті [11] - [12].

Однак зі стандартними рекурентними нейронними мережами виникли наступні проблеми:

- проблема зникнення градієнта: рекурентні нейронні мережі дозволяють моделювати залежні від часу та послідовні проблеми з даними, такі як прогнозування певних показників, однак, рекурентні нейронні мережі важко тренувати через проблему градієнта: градієнти несуть інформацію, яка використовується в рекурентних нейронних мережах, і коли градієнт стає занадто малим, оновлення параметрів стають незначними, це ускладнює вивчення довгих послідовностей даних;
- проблема вибухового градієнта: під час навчання нейронної мережі, якщо нахил має тенденцію зростати експоненційно, а не розпадатися, це називається вибухаючим градієнтом, ця проблема виникає, коли

накопичуються великі градієнти помилок, що призводить до дуже великих оновлень ваг моделі нейронної мережі під час процесу навчання [11].

Одними з найвідоміших варіацій рекурентної нейронної мережі є мережа довгострокової пам'яті та мережа gated recurrent unit (GRU), які направлені на вирішення проблеми зникаючого градієнта в рекурентних нейронних мережах [13]. GRU є простішим варіантом організації блоків, з яких складається мережа.

На відміну від мережі довгострокової пам'яті, GRU складається лише з трьох вентилів і не підтримує внутрішній стан клітини, інформація, яка зберігається у стані внутрішньої комірки в повторюваній одиниці довгострокової пам'яті, включається в прихований стан повторюваного блоку, ця сукупна інформація передається до наступного рекурентного блоку із закритим доступом [13].

Вентилі GRU включають наступні:

- вентиль оновлення визначає, скільки знань минулого потрібно передати в майбутнє, це потужний механізм, оскільки модель може вирішити скопіювати всю інформацію з минулого та усунути ризик зникнення градієнта;
- вентиль скидання визначає, скільки минулих знань потрібно забути, він вирішує, важливий попередній стан клітини чи ні: вентиль скидання вступає в дію, він зберігає відповідну інформацію з минулого кроку часу в новий вміст пам'яті, потім він множить вхідний вектор і прихований стан на їх ваги, далі він обчислює поелементне множення між елементом скидання і раніше прихованим множителем стану, після підсумовування вищезазначених кроків застосовується нелінійна функція активації і генерується наступна послідовність [13]-[15].

Мережа довгострокової пам'яті відрізняється від GRU наступними показниками:

- GRU має два вентиля, а довгострокова пам'ять має три вентиля;
- GRU не має внутрішньої пам'яті, не має вихідного вентиля, який присутній у довгостроковій пам'яті;

– у довгостроковій мережі вхідний і цільовий вентиля з'єднані вентилям оновлення, а в GRU вентиль скидання застосовується безпосередньо до попереднього прихованого стану, в довгостроковій мережі відповідальність за вентиль скидання беруть на себе два вентиля [14].

### **2.3 Метод прогнозування індивідуальної статистики футболістів**

Прогнозування індивідуальної статистики футболістів потребує визначення множини вихідних ознак для такого прогнозування і множини вхідних ознак. Кожен гравець може характеризуватися цілим набором характеристик, розширений набір яких буде розглянуто в розділі 3.

В якості вихідних ознак було обрано наступні:

- кількість забитих голів футболістом у матчі;
- кількість результативних передач, виконаних футболістом у матчі;
- відсоток успішних передач футболіста у грі.

Таким чином, задача прогнозування індивідуальної статистики футболістів розглядається в постановці, яка включає перелічений набір вихідних ознак.

Для прогнозування можна використати один з 2 підходів: прогнозувати за набором вхідних ознак одночасно значення всіх показників або створити 3 окремі моделі і за допомогою кожної з них прогнозувати значення на виході окремо. Оскільки модель може бути розширена і до більшої кількості показників, адже індивідуальна статистика гравця включає велику кількість окремих показників, то в подальшому набір може бути і розширений. Через це, а також враховуючи, що збільшення виходів призводить до підвищення складності моделі, то для прогнозування 3 показників буде створено 3 окремі моделі.

Задача прогнозування полягає у визначенні значень вихідних ознак, які описують виступ гравця у наступній грі, яка ще не відбулась, за історичними даними. До таких даних зокрема належать значення відповідних характеристик у попередніх іграх. Значення показника за  $N$  попередніх ігор

повинно дозволити виконати прогнозування значення показника у  $N + 1$  грі. Додатково можуть бути використані інші характеристики гри футболіста, які характеризують його попередній стан.

Оскільки серед даних, які потрібно враховувати, є послідовність значень, то основною моделлю прогнозування було обрано модель GRU.

На першому етапі роботи методу потрібно отримати вибірку даних про індивідуальні показники футболістів у іграх, що включають:

- значення кількості забитих голів футболістом у іграх;
- значення кількості результативних передач, виконаних футболістом у іграх;
- значення відсотку успішних передач футболіста в іграх;
- значення рейтингу футболіста за участь у іграх;
- значення командного рейтингу суперника для кожної гри.

Командний рейтинг обчислюється як середній рейтинг всіх футболістів команди, що брали участь у грі. Таким чином визначається вплив суперника на гру футболіста.

На другому етапі має бути підготовлено вибірку даних, визначаючи кожен екземпляр даних як сукупність:

- кількість голів (результативних передач, відсотку успішних передач), забитих футболістом у 5 послідовних іграх;
- середній рейтинг футболіста за 5 останніх ігор;
- середній рейтинг команди суперника за 5 останніх ігор;
- значення кількості голів (результативних передач, відсотку успішних передач), забитих футболістом у шостій грі.

Набори по 6 значень, п'ять з яких є вхідними, а одне – вихідним, формуються порціями з вхідного набору даних. Кожна порція формується зсувом на 1 позицію правіше.

Якщо в якійсь грі футболіст не приймав участь, то усереднення відбувається за меншою кількістю днів (не 5, а відповідно 4, 3).

На третьому етапі моделі прогнозування (рис. 2.1) реалізується передоброблення вибірки, коли відсоток успішних передач усереднюється для пропущеної гри за попередньою та наступною іграми, а всі дані загалом нормалізуються.



Рисунок 2.1 – Схема роботи методу прогнозування індивідуальної статистики футболістів

На четвертому етапі створюється модель прогнозування.

Модель прогнозування складається з двох шарів GRU, куди подаються вхідні дані, що визначають послідовність значень голів (результативних передач, відсотку успішних передач), та двох повнозв'язних шарів, на перший

з яких подаються результати з останнього шару GRU і вхідного шару, куди направляються значення середнього рейтингу.

На п'ятому етапі виконується навчання моделі на основі вибірки, підготовленої на третьому етапі.

## **2.4 Висновки за розділом 2**

Проаналізовано методи прогнозування, які використовуються в футбольній сфері. Вони охоплюють методи прогнозування результатів ігор, прогнозування рейтингу футболіста за виступ у грі. Запропоновано метод прогнозування індивідуальної статистики футболістів, який характеризується використанням моделі на основі нейронної мережі з повнозв'язними шарами та вентильними рекурентними вузлами, що дозволяє виконувати прогнозування кількості голів, результативних передач та відсотку успішних передач, виконаних футболістом у грі.

### 3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

#### 3.1 Вибір мови програмування

Мови програмування Java і Python дозволяють виконувати розробку вебзастосунків (табл. 3.1) [16]-[17].

Таблиця 3.1 – Вибір мови програмування

| Характеристика                                  | Java                          | Python                    |
|-------------------------------------------------|-------------------------------|---------------------------|
| Доступні види типізації                         | Статична, сильна, явна        | Динамічна, сильна, неявна |
| Тип мови                                        | Компільована і інтерпретована | Інтерпретована            |
| Легкість створення програм                      | Ні                            | Так                       |
| Багатоплатформність                             | Так                           | Так                       |
| Створення моделей на основі нейронних мереж GRU | Інтерфейс Tensorflow для Java | Основна мова Tensorflow   |

Критеріями для порівняння мов програмування було виділено:

- легкість створення програм: Python надає можливість створювати код швидше, код часто коротший, що важливо для розробки в короткий строк;
- доступні види типізації: динамічна типізація мови Python є однією з причин прискорення швидкості розробки;
- багатоплатформність: обидві мови дозволяють створити кросплатформні рішення, за допомогою яких потім можна розширювати платформу;
- доступність засобів для створення моделей на основі нейронних мереж GRU: обидві мови надають доступ до бібліотеки Tensorflow, хоча саме мова Python є дещо кращою у цьому відношенні, адже початково бібліотека була створена для застосування в програмах, написаних цією мовою,

відповідно повна і найбільш актуальна підтримка виконується саме для Python.

### **3.2 Моделювання роботи користувачів з програмою**

Програма повинна забезпечувати наступні функції для роботи з індивідуальною статистикою футболістів:

- внесення і перегляд статистики участі футболіста у окремій грі чемпіонату;
- внесення результатів гри чемпіонату;
- визначення чемпіонатів і їх сезонів;
- внесення команд до участі в чемпіонаті;
- визначення розкладу ігор;
- внесення команд;
- визначення турнірного положення команд в сезоні чемпіонату;
- перегляд переліку ігор сезону чемпіонату;
- перегляд переліку ігор команди в сезоні чемпіонату;
- обчислення і перегляд статистики команди у грі чемпіонату;
- обчислення і перегляд статистики команди за сезон чемпіонату;
- обчислення і перегляд статистики футболіста за сезон чемпіонату;
- визначення найкращих футболістів у грі сезону чемпіонату;
- визначення найкращих футболістів у команді за сезон;
- порівняння виступів двох футболістів у грі;
- порівняння статистики двох футболістів за сезон;
- пошук потрібного футболіста;
- пошук потрібної команди;
- визначення найкращих команд у сезоні чемпіонату;
- прогнозування кількості забитих м'ячів футболістом у грі сезону;
- прогнозування кількості відданих результативних передач футболістом у грі сезону;

- прогнозування відсотку успішно виконаних передач футболістом у грі сезону;
- авторизація облікового запису.

На рис. 3.1 виконано моделювання прецедентів для роботи користувача з програмою, що не передбачає авторизації.



Рисунок 3.1 – Діаграма прецедентів користувача

На рис. 3.2 виконано моделювання прецедентів для роботи адміністратора, що передбачає авторизацію. Виконати реєстрацію адміністратор не може, внесення облікових записів відбувається тільки через адміністративну панель застосунку.

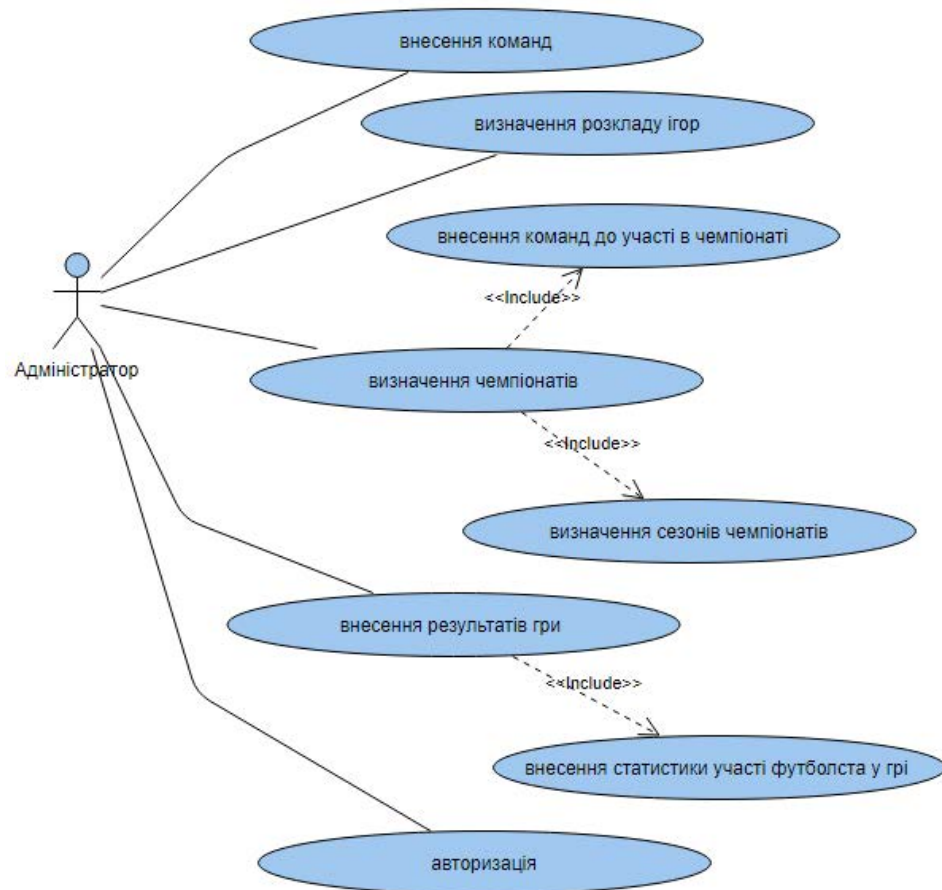


Рисунок 3.2 – Діаграма прецедентів адміністратора

### 3.3 Проєктування бази даних

Сутності предметної області з розділу 1 було деталізовано і визначено зв'язки між цими сутностями. У результаті створено реляційну базу даних на основі системи керування MySQL.

Структура таблиці Footballer включає:

- поле id, що визначає номер футболіста;
- поле name, що визначає повне ім'я футболіста;
- поле height, що визначає зріст футболіста;
- поле weight, що визначає вагу футболіста;
- поле goals, що визначає кількість голів, яку за весь час забив футболіст;

– поле `games`, що визначає кількість ігор, яку за весь час провів футболіст;

– поле `assists`, що визначає кількість результативних передач, яку за весь час віддав футболіст;

– поле `birthday`, що визначає день народження футболіста;

– поле `photo`, що визначає фотографію футболіста;

– поле `position`, що визначає позицію, на якій зазвичай діє футболіст, і є зовнішнім ключем;

– поле `club`, що визначає футбольний клуб, за який на поточний момент грає футболіст, і є зовнішнім ключем;

– поле `country`, що визначає країну, яку представляє футболіст (його національність).

Структура таблиці `Club` включає:

– поле `id`, що визначає номер клубу;

– поле `title`, що визначає назву футбольного клубу;

– поле `city`, що визначає місто, в якому знаходиться футбольний клуб, і є зовнішнім ключем;

– поле `photo`, що визначає шлях до фотографії з емблемою клубу.

Структура таблиці `City` включає:

– поле `id`, що визначає номер міста;

– поле `title`, що визначає назву міста;

– поле `country`, що визначає країну, в якій розташовано місто, і є зовнішнім ключем.

Структура таблиці `Country` включає:

– поле `id`, що визначає номер країни;

– поле `title`, що визначає назву країни.

Структура таблиці `Game` включає:

– поле `id`, що визначає номер гри;

– поле `club1`, що визначає команду, яка є господарем у цій грі, і є зовнішнім ключем;

- поле club2, що визначає команду, яка є гостем у цій грі, і є зовнішнім ключем;
- поле goals1, що визначає кількість голів, забиту в грі командою господарів;
- поле goals2, що визначає кількість голів, забиту в грі командою гостей;
- поле start, що визначає дату і час початку гри;
- поле mom, що визначає найкращого гравця матчу і є зовнішнім ключем з таблиці Footballer;
- поле referee, що визначає суддю гри і є зовнішнім ключем;
- поле championshipseason, що визначає сезон чемпіонату, в якому відбувається гра, і є зовнішнім ключем;
- поле tour, що визначає номер туру чемпіонату, в межах якого відбувається гра.

Структура таблиці FootballerGame включає:

- поле id, що визначає номер статистики участі гравця в грі;
- поле footballer, що визначає футболіста, який приймає участь у грі, і є зовнішнім ключем;
- поле game, що визначає гру, у якій приймає участь футболіст, і є зовнішнім ключем;
- поле position, що визначає позицію, на якій бере участь у грі футболіст, і є зовнішнім ключем;
- поле minutes, що визначає кількість хвилин, яку провів на полі футболіст;
- поле goals, що визначає кількість голів, яку забив у цій грі футболіст;
- поле start, що визначає момент появи футболіста на полі у грі вперше (кількість хвилин);
- поле end, що визначає момент, коли футболіст залишив поле у грі (кількість хвилин) – остання хвилина або хвилина заміни;

- поле assists, що визначає кількість результативних передач футболіста у грі (які призвели до забиття голів);
- поле passes, що визначає кількість передач, виконану футболістом у грі;
- поле yellows, що визначає кількість жовтих карток, отриманих футболістом у грі;
- поле red, що визначає, чи отримав у цій грі футболіст червону картку;
- поле rating, що визначає оцінку футболіста за гру;
- поле successpasses, що визначає кількість передач, які дійшли до адресата, виконану цим футболістом;
- поле aerialswon, що визначає кількість верхових дуелей, які виграв футболіст у грі;
- поле shots, що визначає кількість ударів по воротах, виконану футболістом у грі;
- поле fouls, що визначає кількість порушень правил, виконану футболістом у грі;
- поле dribbles, що визначає кількість обведень, виконану футболістом у грі;
- поле tackles, що визначає кількість перехоплень, виконану футболістом у грі за рахунок пресингу та силової боротьби;
- поле interceptions, що визначає кількість перехоплень загалом, виконану футболістом у грі;
- поле offsides, що визначає кількість положень поза грою, в яку потрапив футболіст протягом гри;
- поле clear, що визначає кількість виносів м'яча з власного штрафного майданчика, виконану футболістом у грі;
- поле blocks, що визначає кількість блокувань, виконану футболістом у грі;
- поле owngoals, що визначає кількість голів, забитих футболістом у власні ворота у грі;

- поле `keypasses`, що визначає кількість ключових передач (які призвели до удару по воротах), виконану футболістом у грі;
- поле `disposses`, що визначає кількість відборів, виконану футболістом у грі без силового контакту;
- поле `crosses`, що визначає кількість кросів, виконану футболістом у грі;
- поле `longballs`, що визначає кількість довгих передач, виконану футболістом у грі;
- поле `outofbox`, що визначає кількість ударів, нанесених з-за меж штрафного майданчика, нанесену футболістом у грі;
- поле `sixyardbox`, що визначає кількість ударів, нанесених з середини воротарського майданчика футболістом у грі;
- поле `penalshots`, що визначає кількість ударів, нанесених з середини одинадцятиметрової зони футболістом у грі;
- поле `penalties`, що визначає кількість пенальті, виконаних футболістом у грі;
- поле `penaltygoals`, що визначає кількість голів, проведених з пенальті футболістом у грі;
- поле `postshots`, що визначає кількість ударів у площину воріт, виконану футболістом у грі;
- поле `errors`, що визначає кількість помилок, які призвели до голу суперника, виконану футболістом у грі;
- поле `saves`, що визначає кількість порятунків, виконану футболістом у грі: для воротаря є основним показником його роботи, але може виконуватися і іншими гравцями;
- поле `goalspredicted`, що визначає прогнозовану кількість голів, забитих футболістом у грі;
- поле `assistpredicted`, що визначає, прогнозовану кількість голевих передач футболіста у грі;

– поле `successpredicted`, що визначає прогнозований відсоток успішних передач футболіста у грі.

Структура таблиці `FootballerSeason` включає:

- поле `id`, що визначає номер статистики участі гравця в сезоні чемпіонату;
- поле `footballer`, що визначає футболіста, для якого сформована статистика, і є зовнішнім ключем;
- поле `clubchampionshipseason`, що визначає участь футбольного клубу в конкретному сезоні чемпіонату і є зовнішнім ключем;
- поле `number`, що визначає номер, під яким брав участь футболіст в іграх сезону;
- поле `minutes`, що визначає кількість хвилин, яку провів на полі футболіст загалом;
- поле `goals`, що визначає кількість голів, яку забив футболіст за сезон;
- поле `assists`, що визначає кількість результативних передач футболіста за сезон;
- поле `passes`, що визначає кількість передач, виконану футболістом за сезон;
- поле `yellows`, що визначає кількість жовтих карток, отриманих футболістом за сезон;
- поле `reds`, що визначає кількість червоних карток, отриманих футболістом за сезон;
- поле `rating`, що визначає середню оцінку футболіста за сезон;
- поле `successpasses`, що визначає кількість передач, які дійшли до адресата, виконану цим футболістом у середньому за сезон;
- поле `aerialswon`, що визначає кількість верхових дуелей, які виграв футболіст у середньому за сезон;
- поле `shots`, що визначає кількість ударів по воротах, виконану футболістом у середньому за сезон;

- поле fouls, що визначає кількість порушень правил, виконану футболістом у середньому за сезон;
- поле dribbles, що визначає кількість обведень, виконану футболістом у середньому за сезон;
- поле tackles, що визначає кількість перехоплень, виконану футболістом у середньому за сезон за рахунок пресингу та силової боротьби;
- поле interceptions, що визначає кількість перехоплень загалом, виконану футболістом у середньому за сезон;
- поле offsides, що визначає кількість положень поза грою, в яку потрапив футболіст у середньому за сезон;
- поле clear, що визначає кількість виносів м'яча з власного штрафного майданчика, виконану футболістом у середньому за сезон;
- поле blocks, що визначає кількість блокувань, виконану футболістом у середньому за сезон;
- поле owngoals, що визначає кількість голів, забитих футболістом у власні ворота у середньому за сезон;
- поле keypasses, що визначає кількість ключових передач (які призвели до удару по воротах), виконану футболістом у середньому за сезон;
- поле disposses, що визначає кількість відборів, виконану футболістом у середньому за сезон без силового контакту;
- поле crosses, що визначає кількість кросів, виконану футболістом у середньому за сезон;
- поле longballs, що визначає кількість довгих передач, виконану футболістом у середньому за сезон;
- поле outofbox, що визначає кількість ударів, нанесених з-за меж штрафного майданчика, нанесену футболістом у середньому за сезон;
- поле sixyardbox, що визначає кількість ударів, нанесених з середини воротарського майданчика футболістом у середньому за сезон;
- поле penalsshots, що визначає кількість ударів, нанесених з середини одинадцятиметрової зони футболістом у середньому за сезон;

- поле `penalties`, що визначає кількість пенальті, виконаних футболістом у середньому за сезон;
- поле `penaltygoals`, що визначає кількість голів, проведених з пенальті футболістом у середньому за сезон;
- поле `games`, що визначає кількість ігор, проведenu футболістом за сезон;
- поле `postshots`, що визначає кількість ударів у площину воріт, виконану футболістом сумарно за сезон;
- поле `errors`, що визначає кількість помилок, які призвели до голу суперника, виконану футболістом сумарно за сезон;
- поле `saves`, що визначає кількість порятунків, виконану футболістом сумарно за сезон.

Структура таблиці `ClubChampionshipSeason` включає:

- поле `id`, що визначає номер участі футбольного клубу в сезоні чемпіонату;
- поле `club`, що визначає футбольний клуб, для якого сформована статистика, і є зовнішнім ключем;
- поле `clubchampionshipseason`, що визначає участь футбольного клубу в конкретному сезоні чемпіонату і є зовнішнім ключем;
- поле `place`, що визначає місце клубу в підсумковій таблиці;
- поле `minutes`, що визначає кількість хвилин, яку провів на полі футболіст загалом;
- поле `goals`, що визначає кількість голів, яку забила команда за сезон;
- поле `goalsagainst`, що визначає кількість голів, пропущену командою за сезон;
- поле `score`, що визначає кількість очок, набрану командою за сезон;
- поле `games`, що визначає кількість ігор, проведenu командою за сезон;
- поле `wins`, що визначає кількість перемог команди за сезон;
- поле `losses`, що визначає кількість поразок команди за сезон;
- поле `draws`, що визначає кількість нічий команди за сезон;

- поле passes, що визначає кількість передач, виконану командою за сезон;
- поле yellows, що визначає кількість жовтих карток, отриманих гравцями команди за сезон;
- поле reds, що визначає кількість червоних карток, отриманих гравцями команди за сезон;
- поле rating, що визначає середню оцінку виставлену футболісту команди за ігри сезону;
- поле successpasses, що визначає відсоток передач, які дійшли до адресата, серед всіх виконаних передач за сезон;
- поле aerialswon, що визначає кількість верхових дуелей, які виграв у середньому футболіст команди за гру сезону;
- поле shots, що визначає кількість ударів по воротах, виконану у середньому футболістом команди за гру сезону;
- поле fouls, що визначає кількість порушень правил, виконану у середньому футболістом команди за гру сезону;
- поле dribbles, що визначає кількість обведень, виконану у середньому футболістом команди за гру сезону;
- поле tackles, що визначає кількість перехоплень, виконану у середньому футболістом команди за гру сезону за рахунок пресингу та силової боротьби;
- поле interceptions, що визначає кількість перехоплень загалом, виконану у середньому футболістом команди за гру сезону;
- поле offsides, що визначає кількість положень поза грою, в яку потрапив у середньому футболіст команди за гру сезону;
- поле clear, що визначає кількість виносів м'яча з власного штрафного майданчика, виконану у середньому футболістом команди за гру сезону;
- поле blocks, що визначає кількість блокувань, виконану у середньому футболістом команди за гру сезону;

- поле `owngoals`, що визначає кількість голів, забитих у власні ворота у середньому футболістом команди за гру сезону;
- поле `keypasses`, що визначає кількість ключових передач (які призвели до удару по воротах), виконану у середньому футболістом команди за гру сезону;
- поле `disposses`, що визначає кількість відборів, виконану у середньому футболістом команди за гру сезону без силового контакту;
- поле `crosses`, що визначає кількість кросів, виконану у середньому футболістом команди за гру сезону;
- поле `longballs`, що визначає кількість довгих передач, виконану у середньому футболістом команди за гру сезону;
- поле `outofbox`, що визначає кількість ударів, нанесених з-за меж штрафного майданчика, нанесену у середньому футболістом команди за гру сезону;
- поле `sixyardbox`, що визначає кількість ударів, нанесених з середини воротарського майданчика у середньому футболістом команди за гру сезону;
- поле `penalshots`, що визначає кількість ударів, нанесених з середини одинадцятиметрової зони у середньому футболістом команди за гру сезону;
- поле `penalties`, що визначає кількість пенальті, виконаних у середньому футболістом команди за гру сезону;
- поле `penaltygoals`, що визначає кількість голів, проведених з пенальті сумарно всіма футболістами команди за гру сезону.

Структура таблиці `ClubGame` включає:

- поле `id`, що визначає номер статистики виступу футбольного клубу в грі;
- поле `club`, що визначає футбольний клуб, для якого сформована статистика за гру, і є зовнішнім ключем;
- поле `game`, що визначає гру, за якою сформована статистика, і є зовнішнім ключем;
- поле `goals`, що визначає кількість голів, яку забила у цій грі;

- поле goalsagainst, що визначає кількість голів, пропущену командою у цій грі;
- поле passes, що визначає кількість передач, виконану гравцем команди у середньому у цій грі;
- поле yellows, що визначає кількість жовтих карток, отриманих гравцями команди у цій грі;
- поле reds, що визначає кількість червоних карток, отриманих гравцями команди у цій грі;
- поле rating, що визначає середню оцінку виставлену футболісту команди у цій грі;
- поле successpasses, що визначає відсоток передач, які дійшли до адресата, серед всіх виконаних гравцями команди передач у цій грі;
- поле aerialswon, що визначає кількість верхових дуелей, які виграв у середньому футболіст команди у цій грі;
- поле shots, що визначає кількість ударів по воротах, виконану у середньому футболістом команди у цій грі;
- поле fouls, що визначає кількість порушень правил, виконану у середньому футболістом команди у цій грі;
- поле dribbles, що визначає кількість обведень, виконану у середньому футболістом команди у цій грі;
- поле tackles, що визначає кількість перехоплень, виконану у середньому футболістом команди у цій грі за рахунок пресингу та силової боротьби;
- поле interceptions, що визначає кількість перехоплень, виконану у середньому футболістом команди у цій грі;
- поле offsides, що визначає кількість положень поза грою, в яку потрапив у середньому футболіст команди у цій грі;
- поле clear, що визначає кількість виносів м'яча з власного штрафного майданчика, виконану у середньому футболістом команди у цій грі;

- поле `blocks`, що визначає кількість блокувань, виконану у середньому футболістом команди у цій грі;
- поле `owngoals`, що визначає кількість голів, забитих у власні ворота у середньому футболістом команди у цій грі;
- поле `keypasses`, що визначає кількість ключових передач (які призвели до удару по воротах), виконану у середньому футболістом команди у цій грі;
- поле `disposses`, що визначає кількість відборів, виконану у середньому футболістом команди у цій грі без силового контакту;
- поле `crosses`, що визначає кількість кросів, виконану у середньому футболістом команди у цій грі;
- поле `longballs`, що визначає кількість довгих передач, виконану у середньому футболістом команди у цій грі;
- поле `outofbox`, що визначає кількість ударів, нанесених з-за меж штрафного майданчика, нанесену у середньому футболістом команди у цій грі;
- поле `sixyardbox`, що визначає кількість ударів, нанесених з середини воротарського майданчика у середньому футболістом команди у цій грі;
- поле `penalshots`, що визначає кількість ударів, нанесених з середини одинадцятиметрової зони у середньому футболістом команди у цій грі;
- поле `penalties`, що визначає кількість пенальті, виконаних футболістами команди у цій грі;
- поле `penaltygoals`, що визначає кількість голів, проведених з пенальті сумарно всіма футболістами команди у цій грі;
- поле `possession`, що визначає відсоток володіння м'ячем командою у цій грі.

Структура таблиці `GameSubstitution` включає:

- поле `id`, що визначає номер виконаної заміни футболістів у грі;
- поле `footballer1`, що визначає футболіста, якого було замінено, і є зовнішнім ключем;

- поле `footballer2`, що визначає футболіста, який вийшов на заміну, і є зовнішнім ключем;

- поле `minutes`, що визначає хвилину, на якій відбулась заміна.

Структура таблиці `Goal` включає:

- поле `id`, що визначає номер забитого голу;

- поле `footballer`, що визначає футболіста, який забиив гол, і є зовнішнім ключем;

- поле `footballer2`, що визначає футболіста, який віддав результативну передачу, і є зовнішнім ключем;

- поле `minutes`, що визначає хвилину, на якій забито гол.

Структура таблиці `ChampionshipSeason` включає:

- поле `id`, що визначає номер сезону чемпіонату;

- поле `start`, що визначає рік початку чемпіонату;

- поле `end`, що визначає рік завершення чемпіонату;

- поле `participants`, що визначає кількість учасників у цьому сезоні чемпіонату;

- поле `championship`, що визначає чемпіонат, до якого належить цей сезон, і є зовнішнім ключем.

Структура таблиці `Referee` (рис. 3.3) включає:

- поле `id`, що визначає номер судді;

- поле `name`, що визначає ім'я судді;

- поле `country`, що визначає країну, яку представляє суддя, і є зовнішнім ключем.

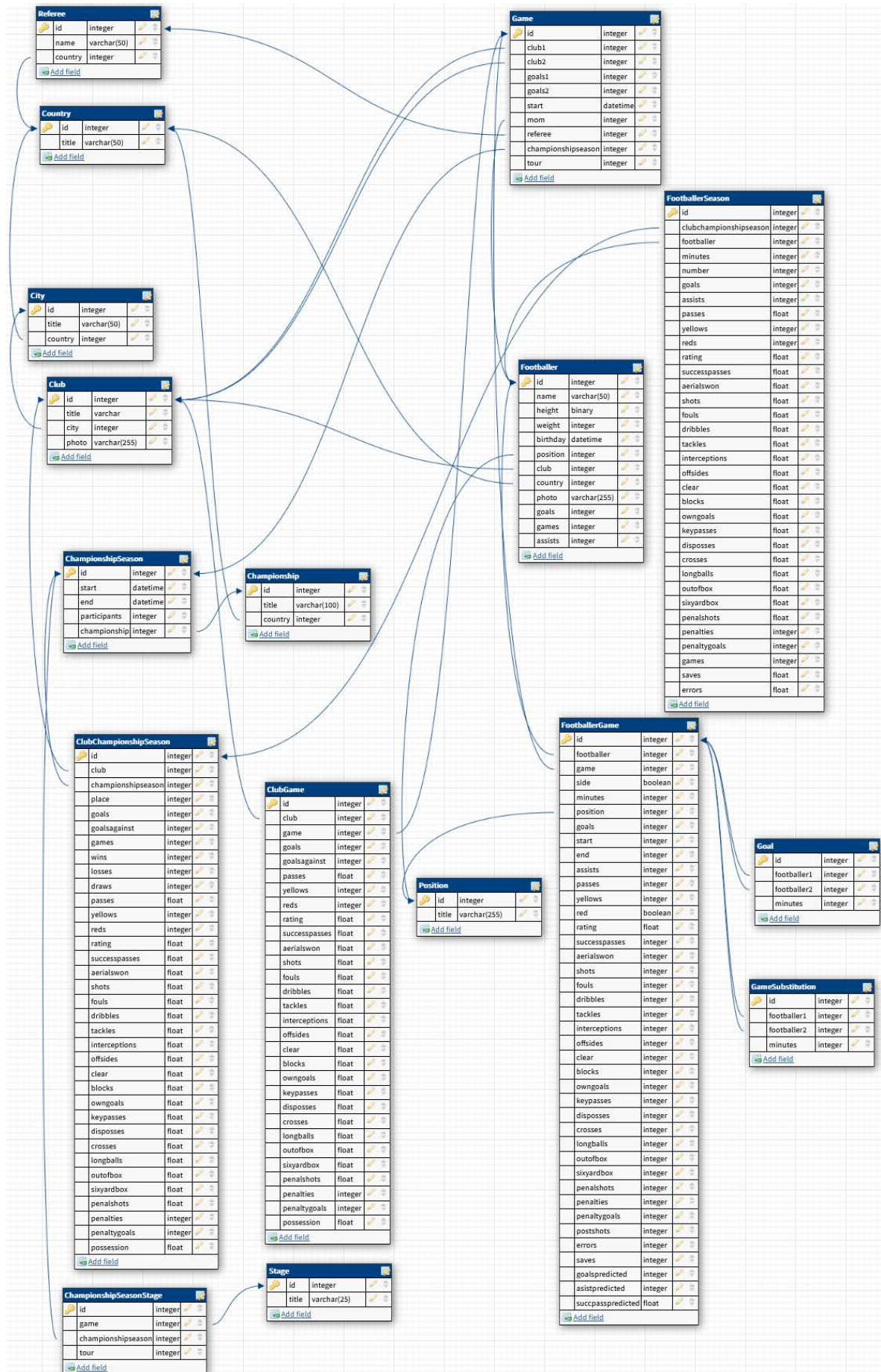


Рисунок 3.3 – Схема бази даних

Структура таблиці Position включає:

- поле id, що визначає номер позиції футболіста на полі;
- поле title, що визначає назву позиції, на якій грає футболіст.

Структура таблиці Stage включає:

- поле id, що визначає номер етапу чемпіонату;
- поле title, що визначає назву етапу чемпіонату (це може бути груповий турнір, чвертьфінал тощо).

Таблиця ChampionshipSeasonStage визначає відповідність етапу чемпіонату. Структура таблиці ChampionshipSeasonStage включає:

- поле id, що визначає номер запису про відповідність етапу чемпіонату;
- поле championshipseason, що визначає сезон чемпіонату, до якого належить етап;
- поле stage, що визначає етап, на якому відбувається ця гра.

Структура таблиці Championship включає:

- поле id, що визначає номер чемпіоната;
- поле title, що визначає назву чемпіонату;
- поле country, що визначає країну, в якій відбувається чемпіонат, і є зовнішнім ключем.

### **3.4 Висновки за розділом 3**

Мова програмування Python обрана для розробки ПЗ за результатами порівняння з мовою Java. Реалізовано моделювання роботи користувачів з програмою. Визначено структуру бази даних з описом всіх таблиць, зв'язків між ними та побудовано схему структури.

## 4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

### 4.1 Структура програмної системи

Програмна система була розроблена таким чином, щоб вмістити і сам вебзастосунок, і реалізацію методу прогнозування (рис. 4.1). Метод прогнозування було виділено в окремий компонент у складі вебзастосунку. Його не було відділено на рівні вище, оскільки він потребує в своїй роботі роботи з базою даних, яка знаходиться на рівні вебзастосунку і створюється його моделями.

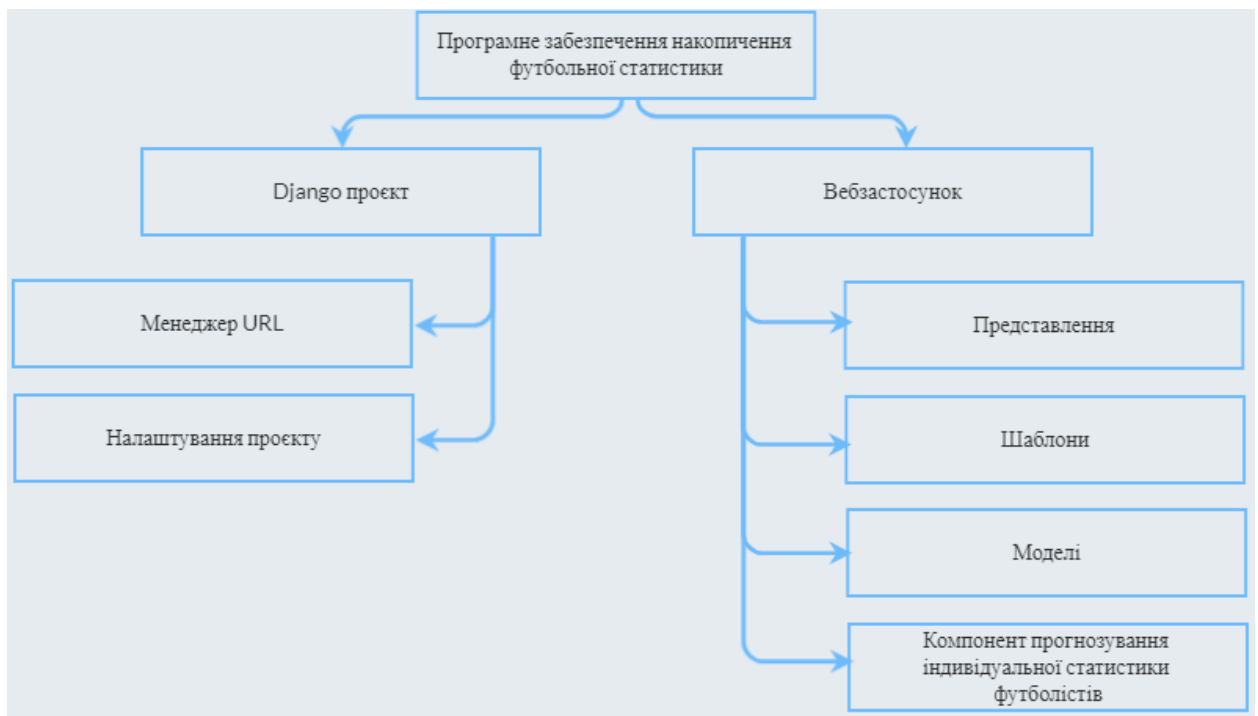


Рисунок 4.1 – Схема структури програмної системи

Розробку виконано на основі фреймворку Django, що вплинуло на структуру програмного забезпечення. Однією з основних складових проєкту є Django-проєкт, який зокрема містить менеджер URL, що визначає зв'язок між представленнями вебзастосунку та URL-адресами, вказаними користувачем.

## 4.2 Опис модулів реалізованої програмної системи

Розробку виконано з застосуванням вебфреймворку Django, що дозволило зменшити кількість створеного коду. Це було досягнуто зокрема через наявність об'єктно-реляційного відображення Django, що призводило до автоматичного зв'язування коду програми з базою даних. Відповідно не потрібно було прописувати велику кількість стандартних запитів, а структура таблиць була побудована на основі коду, прописаного в моделях застосунку.

Під кожену таблицю зі структури бази даних, описаної в розділі 3, було створено відповідний клас моделі. У цілому структура цих класів повторює структуру таблиць. Окрім того до деяких класів було додано засоби обчислення статистики, які вимагаються функціональною вимогою стосовно автоматичного обчислення. За цим принципом користувач має один раз вводити дані про статистику виступу футболіста у грі, а при цьому статистика футболіста за сезон, статистика команди у грі та статистика команди в сезоні має обчислюватися автоматично. Ці обчислення мають виконуватися під час збереження об'єктів. Для цього було перевизначено методи `save(self, *args, **kwargs)` класу `FootballerSeason` зі статистикою футболіста за сезон, яка обчислюється шляхом усереднення показників або знаходження суми значень за відповідним показником (як наприклад, кількість забитих голів), класу `ClubChampionshipSeason` зі статистикою виступу футбольного клубу в сезоні чемпіонату, класу `ClubGame` зі статистикою футбольного клубу за гру чемпіонату.

Окрім того в класі `ClubChampionshipSeason` було реалізовано переобчислення турнірної таблиці. Тобто виконується обчислення кількості перемог, поразок і нічиїх, а також кількості очок, набраних командою. За кількістю очок відбувається сортування команд і визначаються позиції в турнірній таблиці кожної команди.

Клас `FootballerView` є одним з класів, реалізованих на основі стандартного класу деталізації `DetailView`, і передбачає відображення даних

футболіста, включаючи його загальні параметри, як то зріст, вага, дата народження, національність тощо, так і заповнюючи перелік, що складається з результатів виступу футболіста за проведені сезони.

Клас `FootballerNewView` реалізований на основі стандартного класу створення нового об'єкту `CreateView`, який забезпечує підтримку заповнення форми створення нового футболіста даними. Реалізовано підтримку перевірки правильності заповнення форми на основі методу `form_valid(self, form)`, в якому зокрема кількість голів, результативних передач та ігор ініціалізується нулями. Окрім того визначається базовим класом клас `LoginRequiredMixin`, який вимагає проведення авторизації перед використанням цієї форми. Атрибутом `login_url`, що визначає представлення авторизації користувача, задано `loginview`. Атрибутом `success_url`, що визначає відтворювану в разі успішного виконання дії сторінку, визначено представлення перегляду даних футболіста `footballerview`.

Також на основі класу `CreateView` реалізовано зокрема класи `ClubNewView` для створення нового футбольного клубу, `ChampionshipNewView` для створення нового турніру (чемпіонату), `ChampionshipSeasonNewView` для створення нового сезону чемпіонату, `RefereeNewView` для створення нового судді, клас `FootballerGameNewView` для визначення участі футболіста у грі (об'єкт створюється під час заповнення стартового складу команди) клас `FootballerSeasonNewView` для створення нового сезону (об'єкт створюється під час заповнення складу команди на сезон), в якому бере участь футболіст, при цьому таких об'єктів за сезон може створюватися декілька, якщо протягом сезону футболіст змінює свій футбольний клуб. Клас `ClubChampionshipSeasonNewView` визначає створення нового об'єкта при визначенні складу футбольних клубів, які беруть участь у проведення нового сезону турніру. Клас `ClubGameNewView` забезпечує внесення даних про участь клубу в конкретній грі сезону чемпіонату. Внесення самих ігор сезону підтримується класом `GameNewView`. Клас

GameSubstitutionNewView забезпечує внесення заміни у складі команд під час гри. Клас GoalNewView забезпечує внесення голів під час гри.

Клас FootballerUpdateView реалізований на основі стандартного класу редагування даних об'єкту UpdateView, який забезпечує підтримку заповнення форми редагування даних про футболіста. Редагованими полями є тільки ім'я, дата народження, вага, зріст, поточна позиція на полі, поточний клуб та національність. Усі інші поля, які визначають загальну статистику виступів, включаючи сумарну кількість забитих голів, результативних передач та відсотку успішних передач, не можуть бути змінені в цій формі. Для використання представлення також потрібно спочатку авторизуватися. Атрибутом login\_url, що визначає представлення авторизації користувача, задано loginview. Атрибутом success\_url, що визначає відтворення в разі успішного виконання дії сторінку, визначено представлення перегляду даних футболіста footballerview.

Аналогічно влаштовані класи ClubUpdateView, ChampionshipUpdateView, ChampionshipSeasonUpdateView, RefereeUpdateView, FootballerGameUpdateView, FootballerSeasonUpdateView, ClubChampionshipSeasonUpdateView, ClubGameUpdateView, GameUpdateView, GameSubstitutionUpdateView, GoalUpdateView.

Клас FootballerFaceToFaceView реалізує наповнення шаблону footbftf.html даними футболістів для проведення їх порівняння. Шаблони створено на основі вільного шаблону [18].

Клас FootballerSeasonView реалізує наповнення шаблону fseason.html даними статистики виступу футболіста за сезон. Відповідні параметри передаються з форми через GET-запит. Параметр footballer GET-запиту визначає футболіста, параметр season визначає номер сезону, який цікавить користувача. Окрім того підтримується можливість визначення сезону за допомогою року початку, що визначається з параметра startseason, року закінчення, що визначається з параметра endseason, та номера чемпіонату champ, в якому відбувається змагання.

Клас `ClubSeasonBestFootballers` реалізує наповнення шаблону `clubbestfootb.html` даними про футболістів клубу з номером `club`, переданим через параметр GET-запиту, під час сезону з номером `season`, визначаючи найкращих за показником `measure` (таким показником може бути будь-яка характеристика виступу футболіста за сезон).

Клас `SeasonBestFootballers` реалізує наповнення шаблону `seasonbestfootb.html` даними про футболістів за сезон з номером `season`, визначаючи найкращих за показником `measure`.

Клас `FootballerSeasonFaceToFaceView` реалізує наповнення шаблону `footseasftf.html` даними про статистику виступу футболістів з переліку для порівняння за сезон з номером `season`.

Клас `FootballerGameFaceToFaceView` реалізує наповнення шаблону `footgameftf.html` даними про статистику виступу футболістів з переліку для порівняння у конкретній грі `game`.

Клас `SearchFootballerView` забезпечує пошук футболіста з іменем `name`, заповнюючи відповідними футболістами шаблон `searchfootballer.html`. Подібно влаштований клас `SearchClubView` для пошуку футбольних клубів.

### **4.3 Висновки за розділом 4**

Представлено реалізовану структуру програмної системи, надано опис основних зі створених класів, які надають можливості організації даних, підтримки виконання дій користувача, відображення даних користувачу.

## **5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ**

### **5.1 Призначення програми**

Програма призначена для отримування інформації про індивідуальну статистику виступу футболістів у окремих іграх, сезоні чемпіонату загалом, створення командної статистики на її основі, аналізу цієї статистики та прогнозування індивідуальних результатів виступу футболістів в окремих іграх.

### **5.2 Умови виконання програми**

Умови виконання програми визначаються наступними характеристиками:

- процесор: чотири ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять: 10 Гб;
- жорсткий диск: 300 Мб доступного простору.

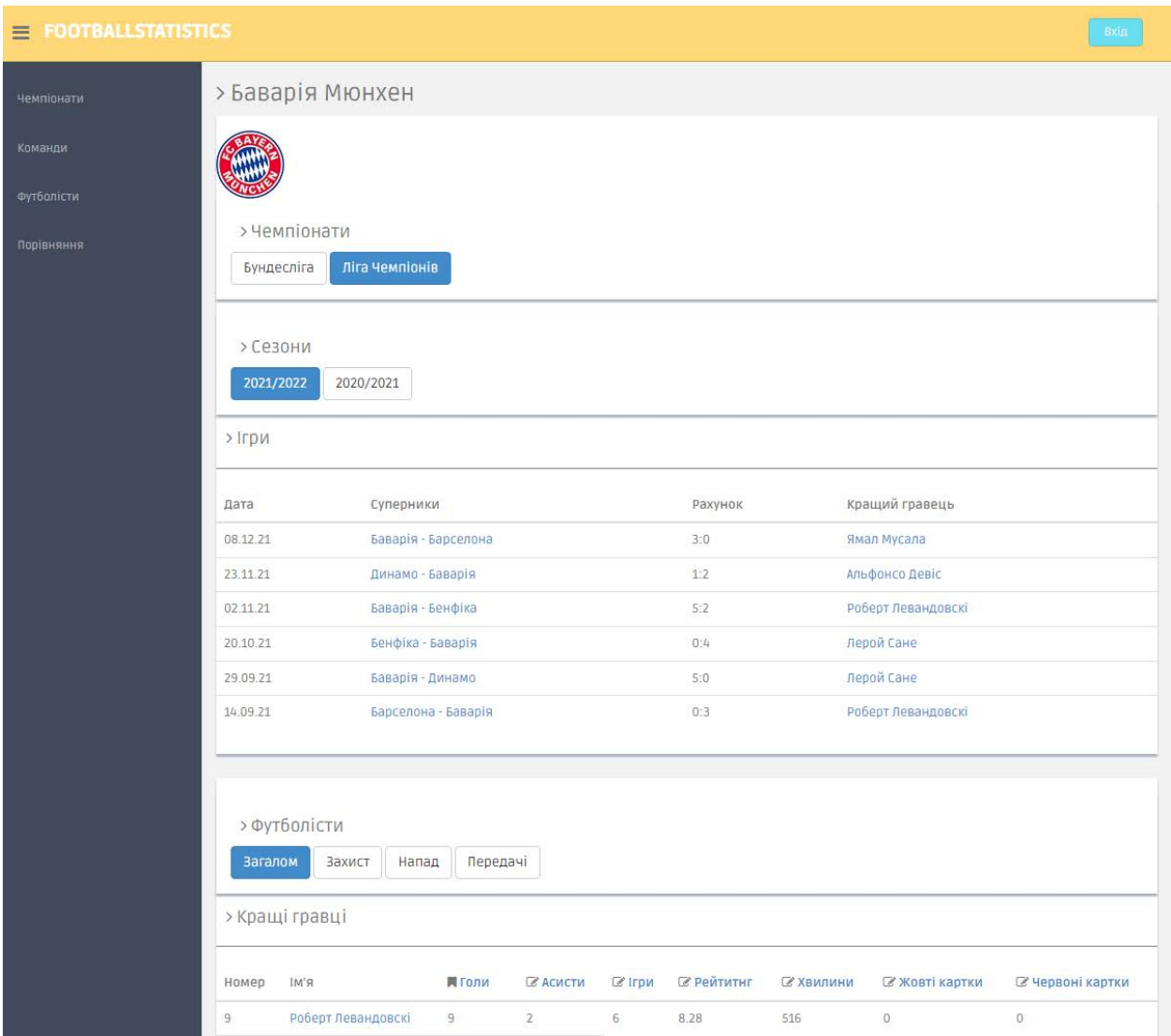
Програма потребує встановлення системи керування базою даних MySQL на сервер.

### **5.3 Виконання програми**

Основою роботи з програмою є індивідуальна статистика футболістів, в чистому вигляді і об'єднана різним чином.

На рис. 5.1 показано, як користувачу відображається сторінка статистики за командою. Вона відкривається при натисканні на будь-якій зі сторінок на назву команди. На цій сторінці доступний вибір чемпіонату та сезону. Для цього потрібно натиснути на кнопку з відповідною назвою. За обраним сезоном відображаються результати ігор, що дозволяє перейти на сторінку відповідної гри. Нижче розташовано таблицю з найкращими

гравцями, де за обраним критерієм демонструються 5 найкращих гравців. Для вибору критерія потрібно натиснути на назву стовпчика.



**FOOTBALLSTATISTICS** Вхід

Чемпіонати  
Команди  
Футболісти  
Порівняння

> Баварія Мюнхен



> Чемпіонати  
Бундесліга **Ліга Чемпіонів**

> Сезони  
**2021/2022** 2020/2021

> Ігри

| Дата     | Суперники           | Рахунок | Кращий гравець     |
|----------|---------------------|---------|--------------------|
| 08.12.21 | Баварія - Барселона | 3:0     | Ямал Мусала        |
| 23.11.21 | Динамо - Баварія    | 1:2     | Альфонсо Девіс     |
| 02.11.21 | Баварія - Бенфіка   | 5:2     | Роберт Левандовскі |
| 20.10.21 | Бенфіка - Баварія   | 0:4     | Лерой Сане         |
| 29.09.21 | Баварія - Динамо    | 5:0     | Лерой Сане         |
| 14.09.21 | Барселона - Баварія | 0:3     | Роберт Левандовскі |

> Футболісти  
**Загалом** Захист Напад Передачі

> Кращі гравці

| Номер | Ім'я               | Голи | Асисті | Ігри | Рейтинг | Хвилини | Жовті картки | Червоні картки |
|-------|--------------------|------|--------|------|---------|---------|--------------|----------------|
| 9     | Роберт Левандовскі | 9    | 2      | 6    | 8.28    | 516     | 0            | 0              |

Рисунок 5.1 – Сторінка статистики за командою

Ще нижче розташована таблиця з інформацією по всім гравцям. Усі індивідуальні показники згруповані в групи, що представляють основні загальні показники, захисні показники, показники гри в нападі, передачі.

Меню організовує роботу з програмою, надаючи доступ до сторінок клубів, чемпіонатів, футболістів за допомогою пунктів з відповідними назвами. З кожного з них можна дістатися до інформації про ігри, гравців, але меню команд та футболістів особливе тим, що звідти можна виконувати пошук відповідно команд або футболістів.

Приклад роботи адміністратора з програмою наведено на рис. 5.2. Звідси адміністратор може вносити всю необхідну інформацію, що стосується чемпіонатів, сезонів, окремих ігор, команд, футболістів.

FOOTBALLSTATISTICS

Вихід

Чемпіонати

Сезони

Ігри

Команди

Футболісти

Міста

> Баварія - Майнц

> Показники

Баварія

Майнц

Загальні

Захист

Напад

Передачі

Інше

Голи

Асисті

Рейтинг

Заміни

Жовті

Червоні

Футболіст

Роберт Левандовскі

Хвилина

90

Внести

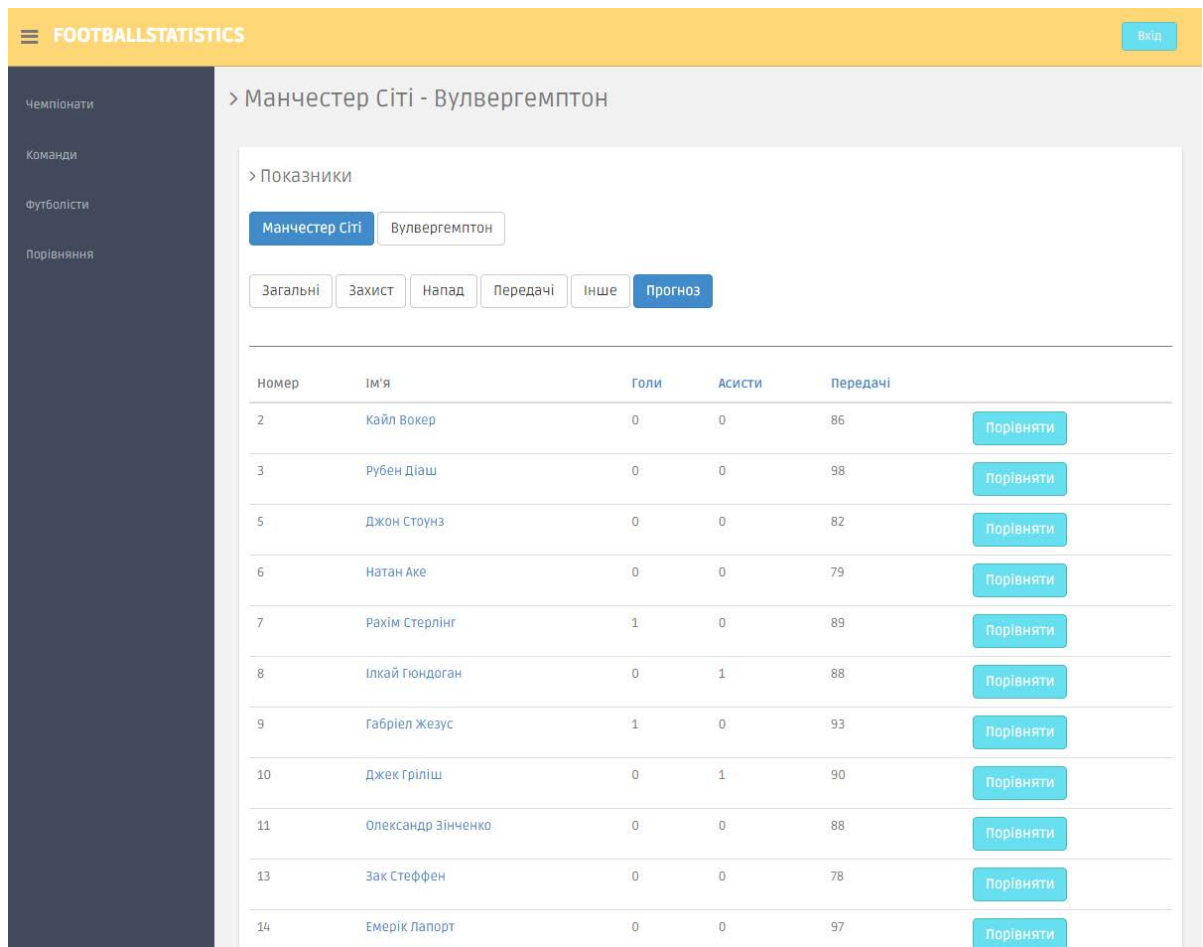
| Ігровий номер | Прізвище           | Голів | Хвилини |            |
|---------------|--------------------|-------|---------|------------|
| 1             | Мануель Нойєр      | 0     |         | Редагувати |
| 2             | Дайо Упамекано     | 0     |         | Редагувати |
| 3             | Омар Річардс       | 0     |         | Редагувати |
| 4             | Ніклас Шуле        | 0     |         | Редагувати |
| 5             | Бенджамін Павар    | 0     |         | Редагувати |
| 7             | Серж Гнабрі        | 0     |         | Редагувати |
| 9             | Роберт Левандовскі | 0     |         | Редагувати |
| 10            | Лерой Сане         | 0     |         | Редагувати |
| 11            | Кінгслі Коман      | 1     | 53      | Редагувати |
| 17            | Мікаель Кюїзанс    | 0     |         | Редагувати |
| 19            | Альфонсо Девіс     | 0     |         | Редагувати |
| 20            | Буна Сарр          | 0     |         | Редагувати |
| 21            | Лукас Ернандес     | 0     |         | Редагувати |
| 22            | Марк Рока          | 0     |         | Редагувати |
| 23            | Тангі Ньянзу       | 0     |         | Редагувати |
| 24            | Корентін Толісо    | 0     |         | Редагувати |
| 25            | Томас Мюлер        | 0     |         | Редагувати |
| 26            | Свен Ульрайх       | 0     |         | Редагувати |
| 40            | Малік Тільман      | 0     |         | Редагувати |
| 42            | Ямал Мусала        | 1     | 74      | Редагувати |

Рисунок 5.2 – Внесення показників футболістів під час матчу

Основою обчислення статистики команд за гру, команд за сезон, гравців за сезон є статистика гравців за гру. Накопичується ця статистика способом, представленим на рис. 5.2. Після вибору конкретної гри адміністратор повинен обрати одного з двох учасників гри, потім обрати групу показників, а нижче конкретний показник.

При відкритті відповідного розділу зверху виводиться поле для вибору конкретного гравця і підтвердження відповідного значення. Таким значенням у випадку голів є хвилина, на якій гол відбувся. Аналогічно для результативних передач з групи «Асисти». Для точності передач це конкретне значення. Для підтвердження потрібно натиснути на кнопку «Внести».

Доступ же до прогнозованих значень виконується зі сторінки гри з групи показників «Прогноз» (рис. 5.3).



FOOTBALLSTATISTICS

Вхід

Чемпіонати

Команди

Футболісти

Порівняння

> Манчестер Сіті - Вулвергемптон

> Показники

Манчестер Сіті Вулвергемптон

Загальні Захист Напад Передачі Інше Прогноз

| Номер | Ім'я               | Голи | Асисти | Передачі |           |
|-------|--------------------|------|--------|----------|-----------|
| 2     | Кайл Вокер         | 0    | 0      | 86       | Порівняти |
| 3     | Рубен Діаш         | 0    | 0      | 98       | Порівняти |
| 5     | Джон Стоунз        | 0    | 0      | 82       | Порівняти |
| 6     | Натан Аке          | 0    | 0      | 79       | Порівняти |
| 7     | Рахім Стерлінг     | 1    | 0      | 89       | Порівняти |
| 8     | Ілкай Гюндоган     | 0    | 1      | 88       | Порівняти |
| 9     | Габріел Жезус      | 1    | 0      | 93       | Порівняти |
| 10    | Джек Гріліш        | 0    | 1      | 90       | Порівняти |
| 11    | Олександр Зінченко | 0    | 0      | 88       | Порівняти |
| 13    | Зак Стеффен        | 0    | 0      | 78       | Порівняти |
| 14    | Емерік Лапорт      | 0    | 0      | 97       | Порівняти |

Рисунок 5.3 – Перегляд результатів прогнозування

Тоді відображається перелік всіх гравців і прогнозовані значення для кожного гравця кількості забитих голів, відданих результативних передач та відсотку успішних передач.

Кнопка «Порівняти» дозволяє додати гравця до порівняння в режимі його статистики за цією грою. Додавання його з персональної сторінки дозволяє виконати порівняння в режимі загальних показників. Відобразити результати порівняння можна за допомогою відповідного пункту меню.

Авторизація і вихід з облікового запису ініціюються кнопками «Вхід» або «Вихід» з меню вгорі сторінки.

#### 5.4 Експериментальне дослідження

Експериментальне дослідження було проведено для вирішення задачі прогнозування кількості забитих м'ячів футболістом у грі, кількості результативних передач, відсотка успішних передач. Для дослідження використовувалась вибірка [19].

Дослідження проводилось на основі використання багатошарового перцептрону, моделі на основі GRU та розширеного варіанту даної моделі з додаванням показників рейтингу на основі запропонованого методу.

Отримані результати для кожного показника були усереднені і в такому вигляді представлені в табл. 5.1.

Таблиця 5.1 – Результати експериментального дослідження

| Метод                                       | Похибка |
|---------------------------------------------|---------|
| Багатошаровий перцептрон                    | 0,319   |
| GRU                                         | 0,226   |
| Запропонований в роботі метод прогнозування | 0,183   |

Таким чином, додавання показників рейтингу призвело до покращення результатів на 19 %. Також створена модель в основі запропонованого методу

виявилась кращою за точністю ніж базова модель – багатошаровий перцептрон.

### **5.5 Висновки за розділом 5**

Описано процес роботи з програмою на рівні організації інтерфейсів користувача.

Проведено експериментальне дослідження задачі прогнозування індивідуальної статистики футболістів, що підтвердило підвищення точності запропонованим методом порівняно з багатошаровим перцептроном та звичайною нейронною мережею на основі GRU.

## ВИСНОВКИ

Проведено аналіз завдань у футбольній сфері, в яких можуть використовуватися інформаційні технології. Виконано огляд програмних аналогів, зокрема whoscored, NVtips.com, bettingStats. Виконана розробка відрізняється від них тим, що об'єднує в собі засоби накопичення індивідуальної футбольної статистики та прогнозування, при чому прогнозування індивідуальної статистики, а не командної. Визначено основні сутності предметної області.

Проаналізовано методи прогнозування, які використовуються в футбольній сфері. Вони охоплюють методи прогнозування результатів ігор, прогнозування рейтингу футболіста за виступ у грі.

Мова програмування Python обрана для розробки ПЗ за результатами порівняння з мовою Java. Реалізовано моделювання роботи користувачів з програмою. Визначено структуру бази даних з описом всіх таблиць, зв'язків між ними та побудовано схему структури.

Представлено реалізовану структуру програмної системи, надано опис основних зі створених класів, які надають можливості організації даних, підтримки виконання дій користувача, відображення даних користувачу.

Розроблений вебзастосунок дозволяє вносити дані про індивідуальні та командні результати ігор, переглядати статистику за гравцями і за командою за окремою грою або сезоном, отримувати рейтинг гравців за різними показниками індивідуальної статистики, прогнозувати результати виступу гравців у наступних іграх.

Проведено експериментальне дослідження задачі прогнозування індивідуальної статистики футболістів, що підтвердило підвищення точності запропонованим методом порівняно з багатоваріантним перцептроном та звичайною нейронною мережею на основі GRU.

Науковою новизною роботи є запропонований метод прогнозування індивідуальної статистики футболістів, який характеризується використанням

моделі на основі нейронної мережі з повнозв'язними шарами та вентильними рекурентними вузлами, що дозволяє виконувати прогнозування кількості голів, результативних передач та відсотку успішних передач, виконаних футболістом у грі.

Завдання роботи виконані повністю.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Spencer, J. How technology is shaping the future of football [Electronic resource] / J. Spencer. – Access mode : <https://www.90min.com/posts/how-technology-is-shaping-the-future-of-football>.
2. Jones, M. 10 Ways Technology Will Change Football in the Next 10 Years [Electronic resource] / M. Jones. – Access mode : <https://bleacherreport.com/articles/2366562-10-ways-technology-will-change-football-in-the-next-10-years>.
3. The future of football is all about high-tech innovation [Electronic resource]. – Access mode : <https://blog.richardvanhooijdonk.com/en/the-future-of-football-is-all-about-high-tech-innovation/>.
4. Technology in soccer [Electronic resource] : An overview of five new advancements in the sport. – Access mode : <https://khelnow.com/football/soccer-technology-five-new-advancements>.
5. Football Statistics. Football Live Scores. WhoScored.com [Electronic resource]. – Access mode : <https://1xbet.whoscored.com/>.
6. Free football statistics and predictions with artificial intelligence [Electronic resource]. – Access mode : <https://nvtips.com/en/>.
7. Football predictions – BettingStats.org [Electronic resource]. – Access mode : <https://bettingstats.org/football-predictions/>.
8. Predicting Football Matches Results using Bayesian Networks for English Premier League (EPL) [Electronic resource] / Nazim Razali, Aida Mustapha, Faiz Ahmad Yatim, Ruhaya Ab Aziz // IOP Conference Series : Materials Science and Engineering 2017. – Access mode : <https://iopscience.iop.org/article/10.1088/1757-899X/226/1/012099/pdf>.
9. Danisik, N. Football Match Prediction using Players Attributes [Electronic resource] / Norbert Danisik, Peter Lacko, Michal Farkas. – Access mode : <https://sci-hub.se/https://ieeexplore.ieee.org/document/8490613>.
10. Patnaik, S. S. Predicting Match Ratings of Football Players using

Machine Learning [Electronic resource] / S. S. Patnaik. – Access mode : <https://medium.com/analytics-vidhya/predicting-match-ratings-of-football-players-using-machine-learning-51bf7a5ab6ad>.

11. Recurrent Neural Network (RNN) Tutorial [Electronic resource] : Types, Examples, LSTM & More. – Access mode : <https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn>.

12. Illustrated Guide to LSTM's and GRU's [Electronic resource] : A step by step explanation. – Access mode : <https://www.thetechplatform.com/post/illustrated-guide-to-lstm-s-and-gru-s-a-step-by-step-explanation>.

13. Gated Recurrent Unit Networks [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/gated-recurrent-unit-networks/>.

14. Lendave, V. LSTM Vs GRU in Recurrent Neural Network [Electronic resource] : A Comparative Study / V. Lendave. – Access mode : <https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/>.

15. Tsang, S.-H. Review [Electronic resource] : Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling (GRU) / S.-H. Tsang. – Access mode : <https://sh-tsang.medium.com/review-empirical-evaluation-of-gated-recurrent-neural-networks-on-sequence-modeling-gru-2adb86559257>.

16. Python for Everybody [Text] : Exploring Data in Python 3 / Charles Russell Severance, Sue Blumenberg, Elliott Hauser, Aimee Andrion. – Southern Carolina : CreateSpace Independent Publishing Platform, 2016. – 247 p.

17. Web Development with Django [Text] : Learn to build modern web applications with a Python-based framework / B. Shaw, S. Badfwar, A. Bird et al. – Birmingham : Packt Publishing, 2021. – 826 p.

18. Dashgum Basic – Free Dashboard Template [Electronic resource]. – Access mode : <https://gridgum.com/themes/dashgum-basic-free-dashboard-template/>.

19. Soccer (Football) dataset from WhoScored [Electronic resource]. –  
Access mode : <https://www.kaggle.com/dejoski/soccer-football-dataset-from-whoscored>.

**ДОДАТОК А**  
**Технічне завдання**

## **Вступ**

Індивідуальна статистика виступу футболістів є основою для прийняття рішень футбольними тренерами, менеджерами. У цій сфері достатньо активно розвиваються інтелектуальні технології. Відносно виступу футболістів важливо прогнозувати їх індивідуальні результати, щоб в результаті приймати рішення, які будуть найкращими для команди.

### **A.1 Підстави для розробки**

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів прогнозування індивідуальної статистики футболістів», яку затверджено наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

### **A.2 Призначення розробки**

Розробка призначена для отримування інформації про індивідуальну статистику виступу футболістів у окремих іграх, сезоні чемпіонату загалом, створення командної статистики на її основі, аналізу цієї статистики та прогнозування індивідуальних результатів виступу футболістів в окремих іграх.

### **A.3 Основні вимоги до програми**

#### **A.3.1 Вимоги до функціональних характеристик**

Функціональні характеристики програмного забезпечення повинні включати:

- внесення і перегляд статистики участі футболіста у грі чемпіонату;

- внесення результатів гри чемпіонату;
- визначення чемпіонатів і їх сезонів;
- внесення команд до участі в чемпіонаті;
- визначення розкладу ігор;
- внесення команд;
- визначення турнірного положення команд в сезоні чемпіонату;
- перегляд ігор сезону чемпіонату;
- перегляд ігор команди в сезоні чемпіонату;
- обчислення і перегляд статистики команди у грі чемпіонату;
- обчислення і перегляд статистики команди за сезон чемпіонату;
- обчислення і перегляд статистики футболіста за сезон чемпіонату;
- визначення найкращих футболістів у грі сезону чемпіонату;
- визначення найкращих футболістів у команді за сезон;
- порівняння виступів двох футболістів у грі;
- порівняння статистики двох футболістів за сезон;
- пошук потрібного футболіста;
- пошук потрібної команди;
- визначення найкращих команд у сезоні чемпіонату;
- прогнозування кількості забитих м'ячів футболістом у грі сезону;
- прогнозування кількості відданих результативних передач футболістом у грі сезону;
- прогнозування відсотку успішно виконаних передач у грі сезону.

### **А.3.2 Вимоги до складу та параметрів технічних засобів**

Апаратне забезпечення роботи вебсервера повинно відповідати вимогам роботи з щонайменше 200 користувачами одночасно. Для підтримки цих вимог апаратне забезпечення повинно мінімально включати:

- процесор: чотири ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять: 10 Гб;

– жорсткий диск: 300 Мб доступного простору.

#### **A.4 Порядок контролю та приймання**

Календарний план є основою для контролю виконання роботи, у свою чергу приймання роботи здійснюється під час її захисту перед екзаменаційною комісією.

**ДОДАТОК Б**  
**Текст програми**

## Б.1 Текст файла models.py

```

from django.db import models
from django.urls import reverse


class FootballerGame (models.Model):
    footballer = models.ForeignKey (Footballer,
on_delete=models.CASCADE)
    side = models.BooleanField()
    game = models.ForeignKey (Game, on_delete=models.CASCADE)
    position = models.ForeignKey (Position,
on_delete=models.CASCADE)
    minutes = models.IntegerField()
    goals = models.IntegerField()
    start = models.IntegerField()
    end = models.IntegerField()
    assists = models.IntegerField()
    passes = models.IntegerField()
    yellows = models.IntegerField()
    red = models.BooleanField()
    rating = models.FloatField()
    successpasses = models.IntegerField()
    aerialswon = models.IntegerField()
    shots = models.IntegerField()
    fouls = models.IntegerField()
    dribbles = models.IntegerField()
    tackles = models.IntegerField()
    interceptions = models.IntegerField()
    offsides = models.IntegerField()
    clear = models.IntegerField()
    blocks = models.IntegerField()
    owngoals = models.IntegerField()
    keypasses = models.IntegerField()
    disposses = models.IntegerField()
    crosses = models.IntegerField()
    longballs = models.IntegerField()
    outofbox = models.IntegerField()
    sixyardbox = models.IntegerField()
    penalshots = models.IntegerField()

```

```

penalties = models.IntegerField()
penaltygoals = models.IntegerField()
postshots = models.IntegerField()
errors = models.IntegerField()
saves = models.IntegerField()
goalspredicted = models.IntegerField()
assistpredicted = models.IntegerField()
succpasspredicted = models.FloatField ()

def get_absolute_url(self):
    return reverse('footballgame', args=[self.id])

class FootballerSeason (models.Model):
    footballer = models.ForeignKey (Footballer,
on_delete=models.CASCADE)
    clubchampionshipseason = models.ForeignKey
(ClubChampionshipSeason, on_delete=models.CASCADE)
    number = models.IntegerField()
    minutes = models.IntegerField()
    goals = models.IntegerField()
    assists = models.IntegerField()
    passes = models.FloatField()
    yellows = models.IntegerField()
    reds = models.IntegerField ()
    rating = models.FloatField()
    successpasses = models.FloatField ()
    aeriawon = models.FloatField ()
    shots = models.FloatField ()
    fouls = models.FloatField ()
    dribbles = models.FloatField ()
    tackles = models.FloatField ()
    interceptions = models.FloatField ()
    offsides = models.FloatField ()
    clear = models.FloatField ()
    blocks = models.FloatField ()
    owngoals = models.FloatField ()
    keypasses = models.FloatField ()
    disposses = models.FloatField ()
    crosses = models.FloatField ()

```

```

longballs = models.FloatField ()
outofbox = models.FloatField ()
sixyardbox = models.FloatField ()
penalshots = models.FloatField ()
penalties = models.IntegerField ()
penaltygoals = models.IntegerField ()
games = models.IntegerField ()
postshots = models.FloatField ()
errors = models.IntegerField ()
saves = models.FloatField ()

def get_absolute_url(self):
    return reverse('footballerseason', args=[self.id])

def save(self, *args, **kwargs):
    fgames = Game.objects.filter(championshipseason =
self.clubchampionshipseason.championshipseason & (club1 =
self.clubchampionshipseason.club | club2 =
self.clubchampionshipseason.club))
    totminutes = 0
    totgoals = 0
    totassists = 0
    totpasses = 0
    totyellows = 0
    totreds = 0
    totrating = 0
    totsuccesspasses = 0
    totaerialswon = 0
    totshots = 0
    totfouls = 0
    totdribbles = 0
    tottackles = 0
    totinterceptions = 0
    totoffsides = 0
    totclear = 0
    totblocks = 0
    totowngoals = 0
    totkeypasses = 0
    totdisposses = 0
    totcrosses = 0

```

```

totlongballs = 0
totoutofbox = 0
totsixyardbox = 0
totpenalshots = 0
totpenalties = 0
totpenaltygoals = 0
totgames = 0
totpostshots = 0
toterrors = 0
totsaves = 0
for fgame in fgames:
    fg = FootballerGame.objects.get(game = fgame)
    totminutes = totminutes + fg.minutes
    totgoals = totgoals + fg.goals
    totassists = totassists + fg.assists
    totpasses = totpasses + fg.passes
    totyellows = totyellows + fg.yellows
    if fg.red:
        totreds = totreds + 1
    totrating = totrating + fg.rating
    totsucceedspasses = totsucceedspasses +
fg.succeedspasses
    totaerialswon = totaerialswon + fg.aerialswon
    totshots = totshots + fg.shots
    totfouls = totfouls + fg.fouls
    totdribbles = totdribbles + fg.dribbles
    tottackles = tottackles + fg.tackles
    totinterceptions = totinterceptions +
fg.interceptions
    totoffsides = totoffsides + fg.offsides
    totclear = totclear + fg.clear
    totblocks = totblocks + fg.blocks
    totowngoals = totowngoals + fg.owngoals
    totkeypasses = totkeypasses + fg.keypasses
    totdisposses = totdisposses + fg.disposses
    totcrosses = totcrosses + fg.crosses
    totlongballs = totlongballs + fg.longballs
    totoutofbox = totoutofbox + fg.outofbox
    totsixyardbox = totsixyardbox + fg.sixyardbox
    totpenalshots = totpenalshots + fg.penalshots
    totpenalties = totpenalties + fg.penalties

```

```

        totpenaltygoals = totpenaltygoals + fg.penaltygoals
        totgames = totgames + fg.1
        totpostshots = totpostshots + fg.postshots
        toterrors = toterrors + fg.errors
        totsaves = totsaves + fg.saves
    self.minutes = totminutes
    self.goals = totgoals
    self.assists = totassists
    self.passes = totpasses / totgames
    self.yellows = totyellows
    self.reds = totreds
    self.rating = totrating / totgames
    self.successpasses = totsucceedspasses / totgames
    self.aerialswon = totaerialswon / totgames
    self.shots = totshots / totgames
    self.fouls = totfouls / totgames
    self.dribbles = totdribbles / totgames
    self.tackles = tottackles / totgames
    self.interceptions = totinterceptions / totgames
    self.offsides = totoffsides / totgames
    self.clear = totclear / totgames
    self.blocks = totblocks / totgames
    self.owngoals = totowngoals / totgames
    self.keypasses = totkeypasses / totgames
    self.disposses = totdisposses / totgames
    self.crosses = totcrosses / totgames
    self.longballs = totlongballs / totgames
    self.outofbox = totoutofbox / totgames
    self.sixyardbox = totsixyardbox / totgames
    self.penalshots = totpenalshots / totgames
    self.penalties = totpenalties / totgames
    self.penaltygoals = totpenaltygoals / totgames
    self.games = totgames
    self.postshots = totpostshots / totgames
    self.errors = toterrors
    self.saves = totsaves / totgames
    super(FootballerSeason, self).save(*args, **kwargs)

```

```

class ClubChampionshipSeason (models.Model):
    club = models.ForeignKey (Club, on_delete=models.CASCADE)

```

```

        championshipseason = models.ForeignKey (ChampionshipSeason,
on_delete=models.CASCADE)
        place = models.IntegerField()
        goals = models.IntegerField()
        goalsagainst = models.IntegerField()
        games = models.IntegerField()
        wins = models.IntegerField()
        losses = models.IntegerField()
        draws = models.IntegerField()
        scores = models.IntegerField()
        passes = models.FloatField()
        yellows = models.IntegerField()
        reds = models.IntegerField ()
        rating = models.FloatField()
        successpasses = models.FloatField ()
        aerialswon = models.FloatField ()
        shots = models.FloatField ()
        fouls = models.FloatField ()
        dribbles = models.FloatField ()
        tackles = models.FloatField ()
        interceptions = models.FloatField ()
        offsides = models.FloatField ()
        clear = models.FloatField ()
        blocks = models.FloatField ()
        owngoals = models.FloatField ()
        keypasses = models.FloatField ()
        disposses = models.FloatField ()
        crosses = models.FloatField ()
        longballs = models.FloatField ()
        outofbox = models.FloatField ()
        sixtyyardbox = models.FloatField ()
        penalshots = models.FloatField ()
        penalties = models.IntegerField ()
        penaltygoals = models.IntegerField ()
        possession = models.FloatField ()

    def get_absolute_url(self):
        return reverse('clubchampionshipseason', args=[self.id])

    def save(self, *args, **kwargs):

```

```

    fgames = ClubGame.objects.filter(club = self.club &
game_championshipseason = self.championshipseason)
    totgoals = 0
    totgoalsagainst = 0
    totwins = 0
    totlosses = 0
    totdraws = 0
    totpasses = 0
    totyellows = 0
    totreds = 0
    totrating = 0
    totsucceppasses = 0
    totaerialswon = 0
    totshots = 0
    totfouls = 0
    totdribbles = 0
    tottackles = 0
    totinterceptions = 0
    totoffsides = 0
    totclear = 0
    totblocks = 0
    totowngoals = 0
    totkeypasses = 0
    totdisposses = 0
    totcrosses = 0
    totlongballs = 0
    totoutofbox = 0
    totsixyardbox = 0
    totpenalshots = 0
    totpenalties = 0
    totpenaltygoals = 0
    totgames = 0
    totpostshots = 0
    totpenalties = 0
    totpenaltygoals = 0
    totpossession = 0
    totscores = 0
    for fgame in fgames:
        fg = ClubGame.objects.get(game = fgame)
        totminutes = totminutes + fg.minutes
        totgoals = totgoals + fg.goals

```

```

totgoalsagainst = totgoalsagainst + fg.goalsagainst
if fg.goals > fg.goalsagainst:
    totwins = totwins + 1
    totalscores = totalscores + 3
elif fg.goals < fg.goalsagainst:
    totlosses = totlosses + 1
else:
    totdraws = totdraws + 1
    totalscores = totalscores + 1
totpasses = totpasses + fg.passes
totyellows = totyellows + fg.yellows
totreds = totreds + fg.reds
totrating = totrating + fg.rating
totsuccesspasses = totsuccesspasses +
fg.successpasses
totaerialswon = totaerialswon + fg.aerialswon
totshots = totshots + fg.shots
totfouls = totfouls + fg.fouls
totdribbles = totdribbles + fg.dribbles
tottackles = tottackles + fg.tackles
totinterceptions = totinterceptions +
fg.interceptions
totoffsides = totoffsides + fg.offsides
totclear = totclear + fg.clear
totblocks = totblocks + fg.blocks
totowngoals = totowngoals + fg.owngoals
totkeypasses = totkeypasses + fg.keypasses
totdisposses = totdisposses + fg.disposses
totcrosses = totcrosses + fg.crosses
totlongballs = totlongballs + fg.longballs
totoutofbox = totoutofbox + fg.outofbox
totsixyardbox = totsixyardbox + fg.sixyardbox
totpenalshots = totpenalshots + fg.penalshots
totpenalties = totpenalties + fg.penalties
totpenaltygoals = totpenaltygoals + fg.penaltygoals
totpossession = totpossession + fg.possession
totgames = totgames + 1
self.goals = totgoals
self.goalsagainst = totgoalsagainst
self.scores = totalscores
self.wins = totwins

```

```

self.losses = totlosses
self.draws = totdraws
self.passes = totpasses / totgames
self.yellows = totyellows
self.reds = totreds
self.rating = totrating / totgames
self.successpasses = totsuccespasses / totgames
self.aerialswon = totaerialswon / totgames
self.shots = totshots / totgames
self.fouls = totfouls / totgames
self.dribbles = totdribbles / totgames
self.tackles = tottackles / totgames
self.interceptions = totinterceptions / totgames
self.offsides = totoffsides / totgames
self.clear = totclear / totgames
self.blocks = totblocks / totgames
self.owngoals = totowngoals / totgames
self.keypasses = totkeypasses / totgames
self.disposses = totdisposses / totgames
self.crosses = totcrosses / totgames
self.longballs = totlongballs / totgames
self.outofbox = totoutofbox / totgames
self.sixyardbox = totsixyardbox / totgames
self.penalshots = totpenalshots / totgames
self.penalties = totpenalties / totgames
self.penaltygoals = totpenaltygoals / totgames
self.possession = totpossession / totgames
super(ClubChampionshipSeason, self).save(*args,
**kwargs)

clubs =
ClubChampionshipSeason.objects().filter(championshipseason =
self.championshipseason)
clubs.order_by('-scores')
q = 1
for club in clubs:
    club.place = q
    club.save()
    q = q + 1

class ClubGame (models.Model):
    club = models.ForeignKey (Club, on_delete=models.CASCADE)

```

```

game = models.ForeignKey (Game, on_delete=models.CASCADE)
goals = models.IntegerField()
goalsagainst = models.IntegerField()
passes = models.FloatField()
yellows = models.IntegerField()
reds = models.IntegerField ()
rating = models.FloatField()
successpasses = models.FloatField ()
aerialswon = models.FloatField ()
shots = models.FloatField ()
fouls = models.FloatField ()
dribbles = models.FloatField ()
tackles = models.FloatField ()
interceptions = models.FloatField ()
offsides = models.FloatField ()
clear = models.FloatField ()
blocks = models.FloatField ()
owngoals = models.FloatField ()
keypasses = models.FloatField ()
disposses = models.FloatField ()
crosses = models.FloatField ()
longballs = models.FloatField ()
outofbox = models.FloatField ()
sixyardbox = models.FloatField ()
penalshots = models.FloatField ()
penalties = models.IntegerField ()
penaltygoals = models.IntegerField ()
possession = models.FloatField ()

def get_absolute_url(self):
    return reverse('clubgame', args=[self.id])

def save(self, *args, **kwargs):
    footballers = []
    if game.club1 = self.club:
        footballers = FootballerGame.objects.filter(game =
self.game & side = False))
    else:
        footballers = FootballerGame.objects.filter(game =
self.game & side = True))

```

```

totpasses = 0
totyellows = 0
totreds = 0
totrating = 0
totsuccesspasses = 0
totaerialswon = 0
totshots = 0
totfouls = 0
totdribbles = 0
tottackles = 0
totinterceptions = 0
totoffsides = 0
totclear = 0
totblocks = 0
totowngoals = 0
totkeypasses = 0
totdisposses = 0
totcrosses = 0
totlongballs = 0
totoutofbox = 0
totsixyardbox = 0
totpenalshots = 0
totpenalties = 0
totpenaltygoals = 0
fcount = 0
for football in footballers:
    fg = FootballerGame.objects.get(game = self.game &
footballer = fg)
    totpasses = totpasses + fg.passes
    totyellows = totyellows + fg.yellows
    if fg.red:
        totreds = totreds + 1
    totrating = totrating + fg.rating
    totsuccesspasses = totsuccesspasses +
fg.successpasses
    totaerialswon = totaerialswon + fg.aerialswon
    totshots = totshots + fg.shots
    totfouls = totfouls + fg.fouls
    totdribbles = totdribbles + fg.dribbles
    tottackles = tottackles + fg.tackles

```

```

        totinterceptions = totinterceptions +
fg.interceptions
        totoffsides = totoffsides + fg.offsides
        totclear = totclear + fg.clear
        totblocks = totblocks + fg.blocks
        totowngoals = totowngoals + fg.owngoals
        totkeypasses = totkeypasses + fg.keypasses
        totdisposses = totdisposses + fg.disposses
        totcrosses = totcrosses + fg.crosses
        totlongballs = totlongballs + fg.longballs
        totoutofbox = totoutofbox + fg.outofbox
        totsixyardbox = totsixyardbox + fg.sixyardbox
        totpenalshots = totpenalshots + fg.penalshots
        totpenalties = totpenalties + fg.penalties
        totpenaltygoals = totpenaltygoals + fg.penaltygoals
        fcount = fcount + 1
    if self.game.club1 = self.club:
        self.goals = self.game.goals1
        self.goalsagainst = self.game.goals2
    else:
        self.goals = self.game.goals2
        self.goalsagainst = self.game.goals1
    self.passes = totpasses / fcount
    self.yellows = totyellows
    self.reds = totreds
    self.rating = totrating / fcount
    self.successpasses = totsuccespasses / fcount
    self.aerialswon = totaerialswon / fcount
    self.shots = totshots / fcount
    self.fouls = totfouls / fcount
    self.dribbles = totdribbles / fcount
    self.tackles = tottackles / fcount
    self.interceptions = totinterceptions / fcount
    self.offsides = totoffsides / fcount
    self.clear = totclear / fcount
    self.blocks = totblocks / fcount
    self.owngoals = totowngoals / fcount
    self.keypasses = totkeypasses / fcount
    self.disposses = totdisposses / fcount
    self.crosses = totcrosses / fcount
    self.longballs = totlongballs / fcount

```

```

        self.outofbox = totoutofbox / fcount
        self.sixyardbox = totsixyardbox / fcount
        self.penalshots = totpenalshots / fcount
        self.penalties = totpenalties / fcount
        self.penaltygoals = totpenaltygoals / fcount
        super(ClubGame, self).save(*args, **kwargs)

class Game (models.Model):
    club1 = models.ForeignKey (Club, on_delete=models.CASCADE)
    club2 = models.ForeignKey (Club, on_delete=models.CASCADE)
    goals1 = models.IntegerField()
    goals2 = models.IntegerField()
    start = models.DateTimeField()
    mom = models.ForeignKey (Footballer,
on_delete=models.CASCADE)
    referee = models.ForeignKey (Referee,
on_delete=models.CASCADE)
    championshipseason = models.ForeignKey (ChampionshipSeason,
on_delete=models.CASCADE)
    tour = models.IntegerField()

    def get_absolute_url(self):
        return reverse('game', args=[self.id])

class Footballer (models.Model):
    name = models.CharField(max_length=50)
    height = models.IntegerField()
    weight = models.IntegerField()
    goals = models.IntegerField()
    games = models.IntegerField()
    assists = models.IntegerField()
    birthday = models.DateField()
    photo = models.CharField(max_length=255)
    position = models.ForeignKey (Position,
on_delete=models.CASCADE)
    club = models.ForeignKey (Club, on_delete=models.CASCADE)
    country = models.ForeignKey (Country,
on_delete=models.CASCADE)

    def get_absolute_url(self):

```

```

        return reverse('footballer', args=[self.id])

    class GameSubstitution (models.Model):
        footballer1 = models.ForeignKey (FootballerGame,
on_delete=models.CASCADE)
        footballer2 = models.ForeignKey (FootballerGame,
on_delete=models.CASCADE)
        minutes = models.IntegerField()

    class Goal (models.Model):
        footballer = models.ForeignKey (FootballerGame,
on_delete=models.CASCADE)
        footballer2 = models.ForeignKey (FootballerGame,
on_delete=models.CASCADE)
        minutes = models.IntegerField()

    class Position (models.Model):
        title = models.CharField(max_length=255)

    class Championship (models.Model):
        title = models.CharField(max_length=100)
        country = models.ForeignKey (Country,
on_delete=models.CASCADE)

    class ChampionshipSeason (models.Model):
        start = models.DateField()
        end = models.DateField()
        participants = models.IntegerField()
        championship = models.ForeignKey (Championship,
on_delete=models.CASCADE)

    class Club (models.Model):
        title = models.CharField(max_length=50)
        city = models.ForeignKey (City, on_delete=models.CASCADE)
        photo = models.CharField(max_length=255)

```

```

class City (models.Model):
    title = models.CharField(max_length=50)
    country = models.ForeignKey (Country,
on_delete=models.CASCADE)

class Country (models.Model):
    title = models.CharField(max_length=50)

class Referee (models.Model):
    name = models.CharField(max_length=50)
    country = models.ForeignKey (Country,
on_delete=models.CASCADE)

class Stage (models.Model):
    title = models.CharField(max_length=25)

class ChampionshipSeasonStage (models.Model):
    championshipseason = models.ForeignKey (ChampionshipSeason,
on_delete=models.CASCADE)
    stage = models.ForeignKey (Stage, on_delete=models.CASCADE)
    tour = models.IntegerField()

```

## Б.2 Текст файла footballerview.py

```

from django.views.generic.detail import DetailView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView, UpdateView
import datetime
import footballstatistics.models

class FootballerView (DetailView):

    model = Footballer
    template_name = 'footballer.html'

    def get_context_data(self, **kwargs):

```

```

        fcd = super().get_context_data(**kwargs)
        footballer =
Footballer.objects.get(id=self.kwargs['pk'])
        fcd ['details'] = footballer
        fseasons =
FootballerSeason.objects.filter(footballer__id = self.kwargs['pk'])
        fseasons.order_by('-id')
        fcd ['seasons'] = fseasons
        return fcd

```

```

class FootballerNewView(LoginRequiredMixin, CreateView):

```

```

    template_name = 'addfootballer.html'
    model = Footballer

```

```

    fields = [
        'name',
        'height',
        'weight',
        'birthday',
        'position',
        'club',
        'country'
    ]

```

```

    login_url = 'loginview'
    success_url = 'footballerview'

```

```

    def form_valid(self, form):
        form.goals = 0
        form.assists = 0
        form.games = 0
        return super().form_valid(form)

```

```

class FootballerUpdateView (LoginRequiredMixin, UpdateView):

```

```

    template_name = 'fupdate.html'
    model = Footballer

```

```

fields = [
    'name',
    'height',
    'weight',
    'birthday',
    'position',
    'club',
    'country'
]

login_url = 'loginview'
success_url = 'footballerview'

```

```

class FootballerFaceToFaceView (ListView):

    model = Footballer
    template_name = 'footbftf.html'

    def get_queryset(self, *args, **kwargs):
        fcd = super(FootballerFaceToFaceView,
self).get_queryset(*args, **kwargs)
        facetoface = self.request.session.get('fllers', [])
        footballers = []
        for f in facetoface:
            footballer = Footballer.objects.get(id = f)
            footballers.append({'fb': footballer })
        fcd ['data'] = footballers
        return fcd

```

```

class FootballerSeasonView (DetailView):

    model = FootballerSeason
    template_name = 'fseason.html'

    def get_queryset(self, *args, **kwargs):
        fcd = super(FootballerSeasonView,
self).get_queryset(*args, **kwargs)
        fnum = self.request.GET.get('footballer')
        seasonnum = self.request.GET.get('season')

```

```

        footballer = []
        if seasonnum:
            footballer = Footballer.objects.get(footballer__id =
fnum, clubchampionshipseason__id = seasonnum)
        else:
            seasonnum1 = self.request.GET.get('startseason')
            seasonnum2 = self.request.GET.get('endseason')
            champnum = self.request.GET.get('champ')
            seasons =
ChampionshipSeason.objects.filter(championship_id = champnum)
            for s in seasons:
                if s.start.strftime("%Y") == seasonnum1 and
s.end.strftime("%Y") = seasonnum2:
                    season = s
                    footballer = Footballer.objects.get(footballer__id =
fnum, clubchampionshipseason = season)
                    fcd['data'] = footballer
            return fcd

```

```

class ClubFootballers (ListView):

    model = Footballer
    template_name = 'clubfootb.html'

    def get_queryset(self, *args, **kwargs):
        fcd = super(ClubFootballers, self).get_queryset(*args,
**kwargs)

        clubnum = self.request.GET.get('club')
        seasonnum = self.request.GET.get('season')
        season = []
        if seasonnum:
            season = ChampionshipSeason.objects.get(pk =
seasonnum)
        else:
            seasonnum1 = self.request.GET.get('startseason')
            seasonnum2 = self.request.GET.get('endseason')
            champnum = self.request.GET.get('champ')
            seasons =
ChampionshipSeason.objects.filter(championship_id = champnum)
            for s in seasons:

```

```

        if s.start.strftime("%Y") == seasonnum1 and
s.end.strftime("%Y") = seasonnum2:
            season = s
            clubchseason =
ClubChampionshipSeason.objects.get(club_id = clubnum,
championshipseason = season)
            footballers =
FootballerSeason.objects.filter(clubchampionshipseason = clubchseason)
            footballers = footballers.order_by('number')
            fcd['data'] = footballers
            return fcd

```

```

class ClubSeasonBestFootballers (ListView):

    model = FootballerSeason
    template_name = 'clubbestfootb.html'

    def get_queryset(self, *args, **kwargs):
        clubnum = self.request.GET.get('club')
        measure = self.request.GET.get('measure')
        seasonnum = self.request.GET.get('season')
        season = []
        if seasonnum:
            season = ChampionshipSeason.objects.get(pk =
seasonnum)
        else:
            seasonnum1 = self.request.GET.get('startseason')
            seasonnum2 = self.request.GET.get('endseason')
            champnum = self.request.GET.get('champ')
            seasons =
ChampionshipSeason.objects.filter(championship_id = champnum)
            for s in seasons:
                if s.start.strftime("%Y") == seasonnum1 and
s.end.strftime("%Y") = seasonnum2:
                    season = s
                    clubchseason =
ClubChampionshipSeason.objects.get(club_id = clubnum,
championshipseason = season)
                    footballers = fcd.filter(clubchampionshipseason =
clubchseason)

```

```

        footballers = footballers.order_by('measure')
        return footballers

class SeasonBestFootballers (ListView):

    model = FootballerSeason
    template_name = 'seasonbestfootb.html'

    def get_queryset(self, *args, **kwargs):
        seasonnum = self.request.GET.get('season')
        measure = self.request.GET.get('measure')
        season = []
        if seasonnum:
            season = ChampionshipSeason.objects.get(pk =
seasonnum)
        else:
            seasonnum1 = self.request.GET.get('startseason')
            seasonnum2 = self.request.GET.get('endseason')
            champnum = self.request.GET.get('champ')
            seasons =
ChampionshipSeason.objects.filter(championship_id = champnum)
            for s in seasons:
                if s.start.strftime("%Y") == seasonnum1 and
s.end.strftime("%Y") = seasonnum2:
                    season = s
            clubs =
ClubChampionshipSeason.objects.filter(championship = season)
            footballersfull = []
            for club in clubs:
                footballers =
FootballerSeason.objects.filter(clubchampionshipseason = club)
                footballersfull.add(footballers)
            footballersfull.order_by('measure')
            return footballers

class FootballerSeasonFaceToFaceView (ListView):

    model = FootballerSeason
    template_name = 'footseasftf.html'

```

```

    def get_queryset(self, *args, **kwargs):
        fcd = super(FootballerSeasonFaceToFaceView,
self).get_queryset(*args, **kwargs)
        seasonnum = self.request.GET.get('season')
        season = []
        if seasonnum:
            season = ChampionshipSeason.objects.get(pk =
seasonnum)
        else:
            seasonnum1 = self.request.GET.get('startseason')
            seasonnum2 = self.request.GET.get('endseason')
            champnum = self.request.GET.get('champ')
            seasons =
ChampionshipSeason.objects.filter(championship_id = champnum)
            season = seasons[0]
            for s in seasons:
                if s.start.strftime("%Y") == seasonnum1 and
s.end.strftime("%Y") = seasonnum2:
                    season = s
            facetoface = self.request.session.get('fllers', [])
            footballers = []
            for f in facetoface:
                footballer = FootballerSeason.objects.get(footballer
= f, clubchampionshipseason__championshipseason = season)
                footballers.append({'fb': footballer })
            fcd ['data'] = footballers
            return fcd

```

```

class FootballerGameFaceToFaceView (ListView):

    model = FootballerSeason
    template_name = 'footgameftf.html'

    def get_queryset(self, *args, **kwargs):
        fcd = super(FootballerSeasonFaceToFaceView,
self).get_queryset(*args, **kwargs)
        gamenum = self.request.GET.get('game')
        fgame = Game.objects.get(pk = gamenum)
        footballers = []
        for f in facetoface:

```

```

        footballer = FootballerGame.objects.get(footballer =
f, game = fgame)
        footballers.append({'fb': footballer })
        fcd ['data'] = footballers
        return fcd

```

```

class SearchFootballerView (ListView):

    model = Footballer
    template_name = 'searchfootballer.html'

    def get_queryset(self, *args, **kwargs):
        fcd = super(SearchFootballerView,
self).get_queryset(*args, **kwargs)
        fname = self.request.GET.get('name')
        fcd = fcd.filter(name__contains = fname)
        return fcd

```

**ДОДАТОК В**  
**Слайди презентації**

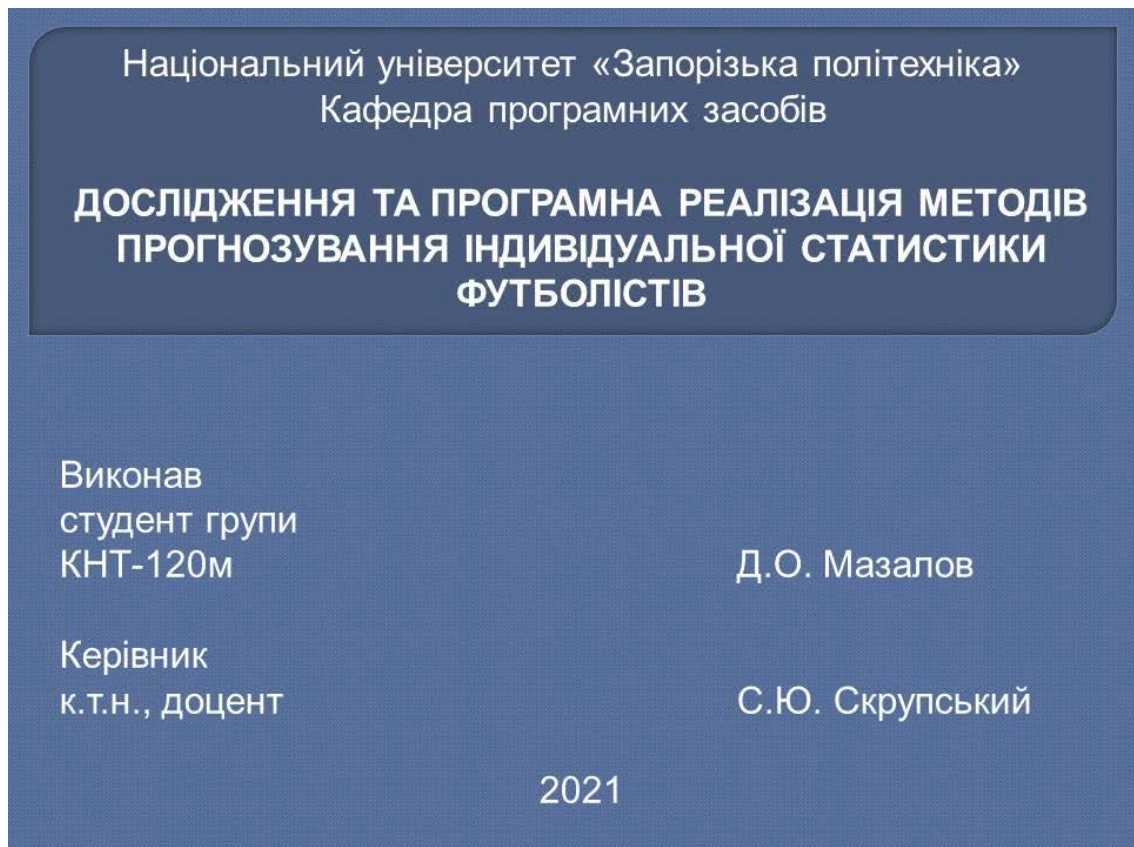


Рисунок В.1 – Слайд 1

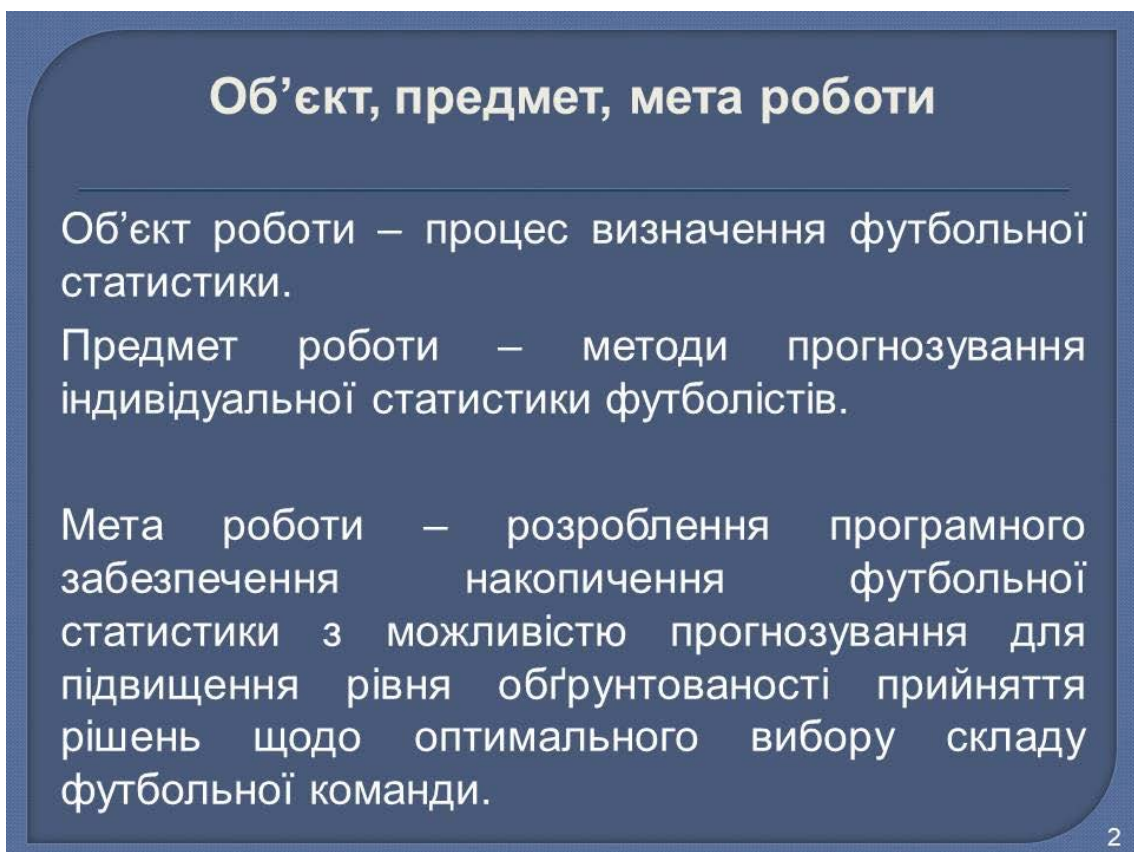


Рисунок В.2 – Слайд 2

## Завдання роботи

- ✓ проаналізувати предметну область на основі існуючих програмних засобів;
- ✓ створити метод прогнозування індивідуальної статистики футболістів;
- ✓ виконати проектування програмного забезпечення;
- ✓ реалізувати програмне забезпечення;
- ✓ виконати експериментальне дослідження проблеми прогнозування індивідуальної статистики футболістів.

3

Рисунок В.3 – Слайд 3

## Аналіз предметної області

| Характеристика                                               | NVtips | Whoscored | Дипломна розробка |
|--------------------------------------------------------------|--------|-----------|-------------------|
| Прогнозування командних результатів                          | +      | –         | –                 |
| Індивідуальна статистика виступу футболістів                 | –      | +         | +                 |
| Прогнозування індивідуальних результатів виступу футболістів | –      | –         | +                 |

4

Рисунок В.4 – Слайд 4

## Метод прогнозування індивідуальної статистики футболістів



5

Рисунок В.5 – Слайд 5

## Вибір мови програмування

| Характеристика                                  | Java                          | Python                    |
|-------------------------------------------------|-------------------------------|---------------------------|
| Доступні види типізації                         | Статична, сильна, явна        | Динамічна, сильна, неявна |
| Тип мови                                        | Компільована і інтерпретована | Інтерпретована            |
| Легкість створення програм                      | Ні                            | Так                       |
| Багатоплатформність                             | Так                           | Так                       |
| Створення моделей на основі нейронних мереж GRU | Інтерфейс Tensorflow для Java | Основна мова Tensorflow   |

6

Рисунок В.6 – Слайд 6

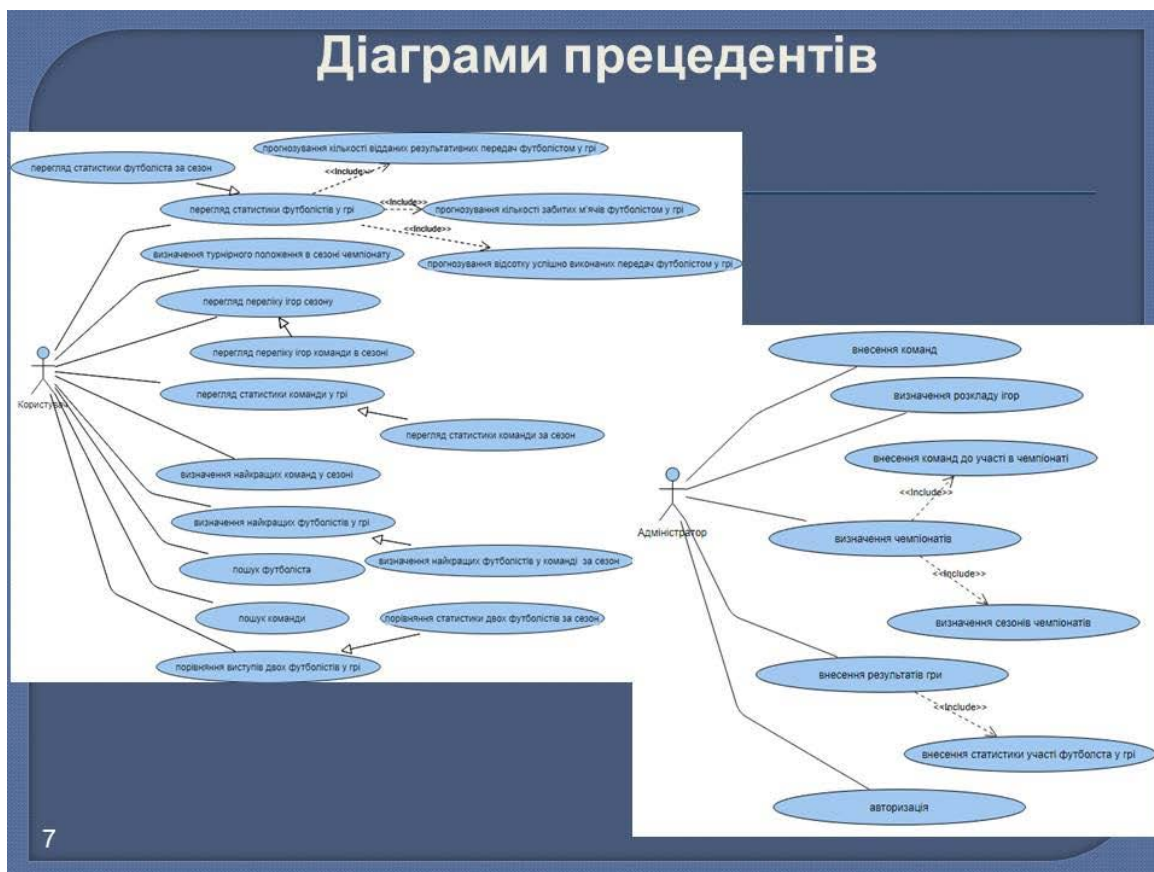


Рисунок В.7 – Слайд 7

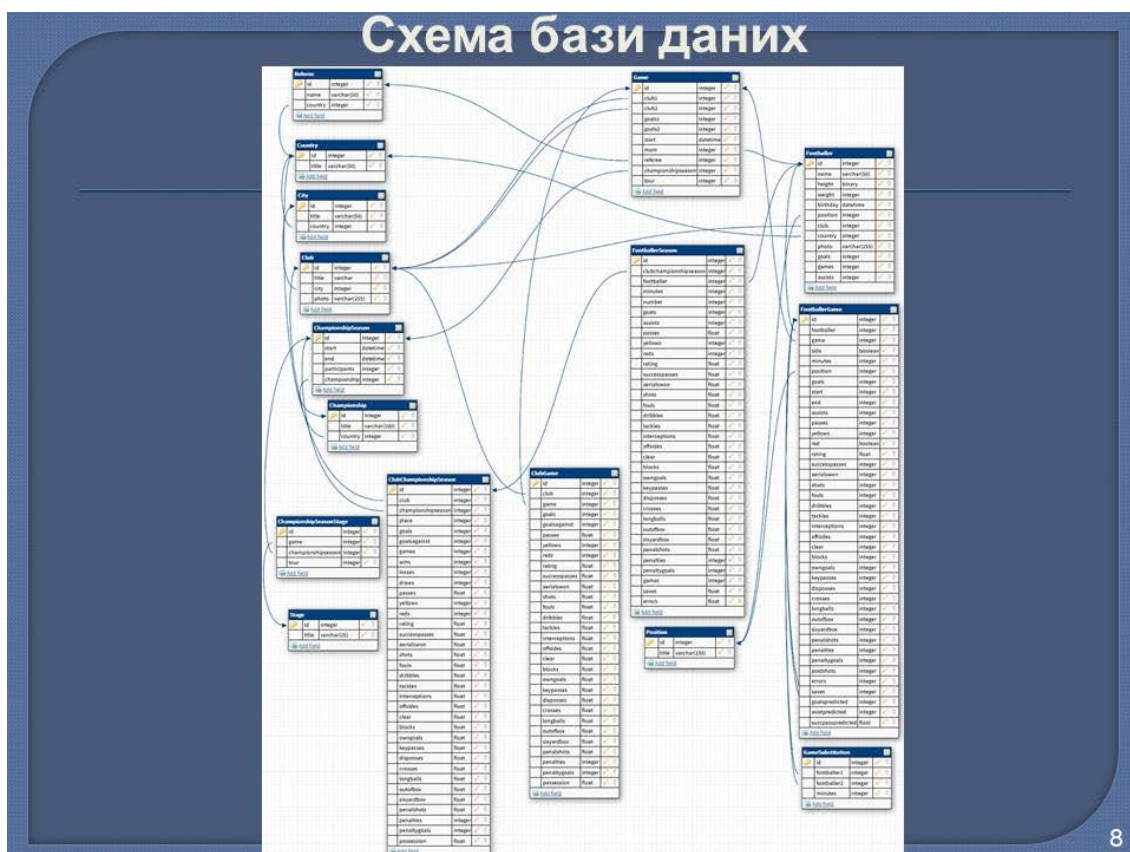


Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9

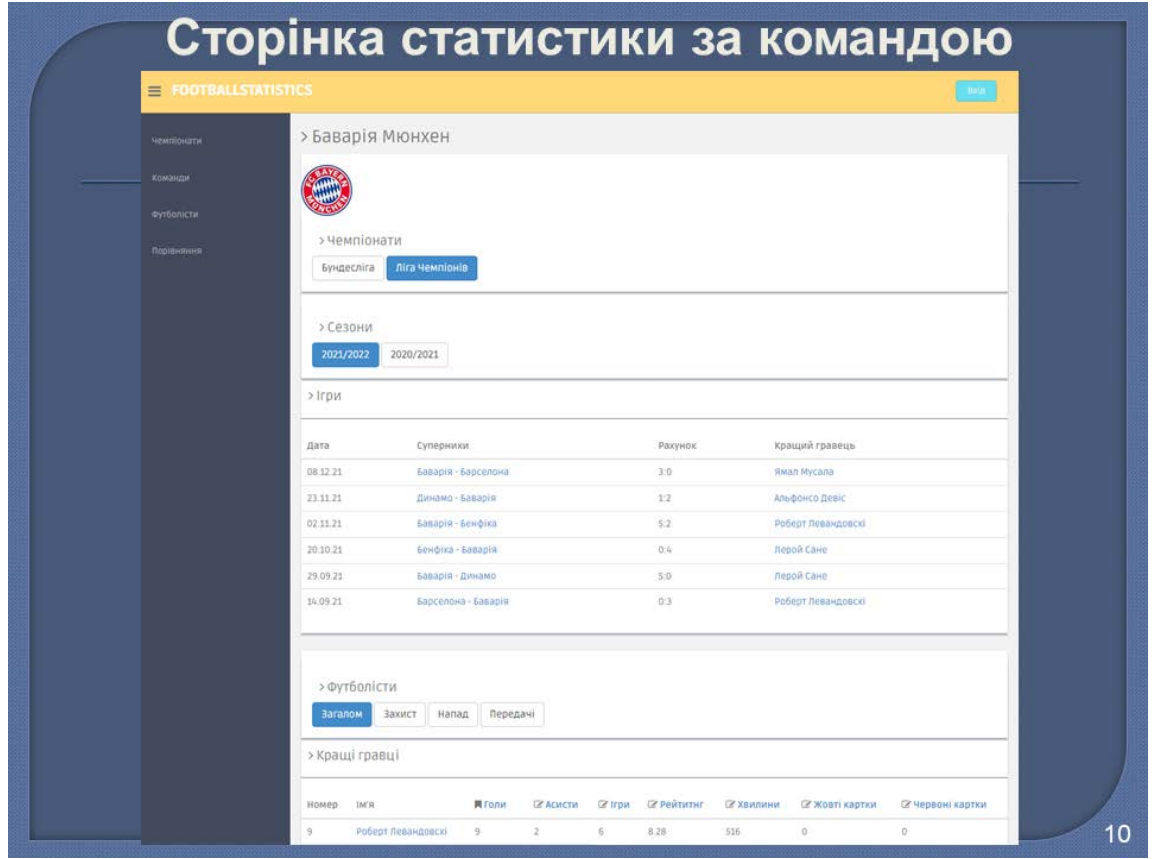


Рисунок В.10 – Слайд 10

FOOTBALLSTATISTICS

Баварія - Майнц

Показники

Баварія Майнц

Загальні Захист Напад Передачі Інше

Голі Асисті Райтинг Замени Жетти Червоні

Футболіст

Роберт Левандовський

Номери

90

Вибрати

| Кравий номер | Позиція              | Голи | Асистен | Порівняти |
|--------------|----------------------|------|---------|-----------|
| 1            | Мануель Нойер        | 0    |         | Порівняти |
| 2            | Давід Алава          | 0    |         | Порівняти |
| 3            | Серж Греббер         | 0    |         | Порівняти |
| 4            | Марк Келлер          | 0    |         | Порівняти |
| 5            | Доминик Равен        | 0    |         | Порівняти |
| 7            | Серж Греббер         | 0    |         | Порівняти |
| 8            | Роберт Левандовський | 0    |         | Порівняти |
| 10           | Ларс Бенд            | 0    |         | Порівняти |
| 11           | Кіндрік Ханс         | 1    | 10      | Порівняти |
| 17           | Міхал Козмін         | 0    |         | Порівняти |
| 19           | Андрей Девіс         | 0    |         | Порівняти |
| 20           | Віктор Клар          | 0    |         | Порівняти |
| 21           | Ларс Бенд            | 0    |         | Порівняти |
| 22           | Марк Келлер          | 0    |         | Порівняти |
| 23           | Томас Мюллер         | 0    |         | Порівняти |
| 24           | Віктор Клар          | 0    |         | Порівняти |
| 25           | Томас Мюллер         | 0    |         | Порівняти |
| 26           | Давід Алава          | 0    |         | Порівняти |
| 40           | Марк Келлер          | 0    |         | Порівняти |
| 42           | Віктор Клар          | 1    | 10      | Порівняти |

11

Рисунок В.11 – Слайд 11

FOOTBALLSTATISTICS

Манчестер Сіті - Вулвергемптон

Показники

Манчестер Сіті Вулвергемптон

Загальні Захист Напад Передачі Інше Прогноз

| Номер | Ім'я               | Голи | Асисті | Передачі | Порівняти |
|-------|--------------------|------|--------|----------|-----------|
| 2     | Кайл Вокер         | 0    | 0      | 86       | Порівняти |
| 3     | Рубен Діаш         | 0    | 0      | 98       | Порівняти |
| 5     | Джон Стоунс        | 0    | 0      | 82       | Порівняти |
| 6     | Натан Аке          | 0    | 0      | 79       | Порівняти |
| 7     | Рахім Стерлінг     | 1    | 0      | 89       | Порівняти |
| 8     | Ілкай Гюндоган     | 0    | 1      | 88       | Порівняти |
| 9     | Габріел Жезус      | 1    | 0      | 93       | Порівняти |
| 10    | Джек Гріліш        | 0    | 1      | 90       | Порівняти |
| 11    | Олександр Зінченко | 0    | 0      | 88       | Порівняти |
| 13    | Зак Стеффен        | 0    | 0      | 78       | Порівняти |
| 14    | Емерік Лапорт      | 0    | 0      | 97       | Порівняти |

12

Рисунок В.12 – Слайд 12

## Експериментальне дослідження

| Метод                                       | Похибка |
|---------------------------------------------|---------|
| Багатошаровий перцептрон                    | 0,319   |
| <b>GRU</b>                                  | 0,226   |
| Запропонований в роботі метод прогнозування | 0,183   |

13

Рисунок В.13 – Слайд 13

## Висновки

Розроблений вебзастосунок дозволяє вносити дані про індивідуальні та командні результати ігор, переглядати статистику за гравцями і за командою за окремою грою або сезоном, отримувати рейтинг гравців за різними показниками індивідуальної статистики, прогнозувати результати виступу гравців у наступних іграх.

Науковою новизною роботи є запропонований метод прогнозування індивідуальної статистики футболістів, який характеризується використанням моделі на основі нейронної мережі з повнозв'язними шарами та вентильними рекурентними вузлами, що дозволяє виконувати прогнозування кількості голів, результативних передач та відсотку успішних передач, виконаних футболістом у грі.

Завдання дипломної роботи виконані повністю.

14

Рисунок В.14 – Слайд 14