МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ**

ННІ/факультет Інформаційних технологій
Кафедра Інформатики і прикладного програмного забезпечення
Спеціальність Інженерія програмного забезпечення
Форма навчання Денна

# КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Черненко Денис Сергійович
*(прізвище, ім'я, по батькові здобувача)*

на тему «Розробка програмного забезпечення для футбольного клубу»
*(повна назва теми)*

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД
*(повна назва бази дослідження)*

науковий керівник к.т.н. і Медведєв Д.Г.
*(наук. ступінь, вчене звання)* *(підпис)* *(прізвище, ініціали)*

**Робота допущена до захисту в ЕК**
Протокол засідання кафедри
від 11.06.2025 р. № 12
Завідувач кафедри _____
(підпис)
д.т.н., професор Зеленський О.С.
*Наук. ступень, вчене звання* *Ініціали, прізвище*

Кривий Ріг – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ**

| | |
|---|---|
| ННІ/факультет | Інформаційних технологій |
| Кафедра | Інформатики і прикладного програмного забезпечення |
| Спеціальність | Інженерія програмного забезпечення |
| Форма навчання | Денна |

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
« 11 » червня 2025 року

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

1. Тема роботи «Розробка програмного забезпечення для футбольного клубу»

Керівник роботи к.т.н. Медведєв Д.Г.,
затверджені наказом закладу вищої освіти від «04» квітня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об'єкт, предмет та мета дослідження:

**Розділ 1. ДОСЛІДЖЕННЯ ЗАДАЧІ АВТОМАТИЗАЦІЇ ЗБОРУ ТА ВІДОБРАЖЕННЯ ФУТБОЛЬНОЇ СТАТИСТИКИ**

**Розділ 2. РОЗРОБКА АЛГОРИТМУ ВИРІШЕННЯ ЗАДАЧІ**

**Розділ 3. ПРОЕКТУВАННЯ БАЗИ ДАНИХ**

**Розділ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

*Об'єкт дослідження: футбольна статистика*

*Предмет дослідження: автоматизація збору, обробки та аналізу футбольної статистика*

*Мета кваліфікаційної роботи*: *розробка програмного забезпечення для аналізу футбольної статистики*

5. Дата видачі завдання «04» квітня 2025 р.

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів МДР | Строк виконання етапів роботи | Відмітка керівника про виконання етапів (дата, підпис) |
|---|---|---|---|
| 1 | Підготовка розділу 1 | 04.04.2025-13.04.2025 | |
| 2 | Підготовка розділу 2 | 14.04.2025-26.04.2025 | |
| 3. | Підготовка розділу 3 | 27.04.2025-15.05.2025 | |
| 4 | Підготовка розділу 4 | 16.05.2025-08.06.2025 | |
| 5 | Реєстрація завершеної кваліфікаційної роботи | 09.06.2025 | Реєстраційний №___ «09»червня 2025 р. |
| 6 | Отримання відгуку від наукового керівника | 10.06.2025 | |
| 7 | Подання кваліфікаційної роботи на перегляд завідувачу кафедри | 11.06.2025 | |
| 8 | Отримання зовнішньої рецензії | 12.06.2025 | |
| 9 | Попередній захист кваліфікаційної роботи на кафедрі | 13.06.2025 | |
| 10 | Підготовка до захисту в ЕК | 16.06.2025-21.06.2025 | |

Завдання підготував науковий керівник _____ (підпис) Медведєв Д.Г. (прізвище та ініціали)

Завдання одержав _____ (підпис) Черненко Д.С. (прізвище та ініціали)

**ЗГОДА здобувача вищої освіти**

Державного університету економіки і технологій про перевірку кваліфікаційної роботи на прояви академічного плагіату
та розміщення в Репозитарії Університету

Я, Черненко Денис Сергійович (ПІП), підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота «Розробка програмного забезпечення для футбольного клубу» виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету. З умовами такого розміщення ознайомлений(на).

Дата підпис ініціали, прізвище (власноруч)

# АНОТАЦІЯ

## на кваліфікаційну бакалаврську роботу

## «Розробка програмного забезпечення для футбольного клубу»

**Черненко Денис Сергійович**

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

У бакалаврській дипломній роботі розроблене програмне забезпечення для відображення та аналізу футбольної статистики. При розробці використані сучасна мова програмування C# та технології .NET Windows Forms, ADO.NET. Для відображення статистики розроблений вебсайт з використанням ASP.NET.

Ключові слова: ФУТБОЛ, АЛГОРИТМИ, СУБД, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, WINDOWS FORMS, ASP.NET, MS SQL SERVER, MS ACCESS.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

| | |
|---|---|
| БД(база даних) | Впорядкований набір логічно взаємопов’язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів. |
| СУБД | Система управління базами даних. |
| ПЗ | Програмне забезпечення. |
| ASP .NET | Active Server Pages для .NET – технологія розробки веб-сайтів на платформі .NET |

ЗМІСТ

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ЗАДАЧІ АВТОМАТИЗАЦІЇ ЗБОРУ ТА ВІДОБРАЖЕННЯ ФУТБОЛЬНОЇ СТАТИСТИКИ ..... 9
1.1. Характеристика задачі ..... 9
1.2. Аналіз існуючого програмного забезпечення ..... 12
1.3. Аналіз вимог до програмного забезпечення ..... 15
РОЗДІЛ 2 РОЗРОБКА АЛГОРИТМУ ВИРІШЕННЯ ЗАДАЧІ ..... 25
РОЗДІЛ 3 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ..... 28
3.1. Структура бази даних ..... 28
3.2. Розробка бази даних в середовищі СУБД MS Access ..... 36
3.3. Розробка бази даних SQL Server ..... 40
РОЗДІЛ 4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ..... 49
4.1. Вибір технологій розробки ..... 49
4.2. Розробка програмного забезпечення з використанням Windows Forms ... 50
4.3. Розробка модулю імпорту даних на Windows Forms ..... 56
4.4. Розробка web-сайту на ASP.NET ..... 57
4.5. Розробка форм в СУБД Access ..... 63
ВИСНОВКИ ..... 68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 69
ДОДАТКИ

# ВСТУП

Метою роботи є розробити програмне забезпечення для аналізу футбольної статистики футбольних клубів та гравців.

Виходячи з мети, задачами роботи є:

1. Розглянути підходи до розробки програмного забезпечення для аналізу футбольної статистики.
2. Зробити критичний огляд існуючих аналогічних рішень.
3. Сформулювати вимоги до розробки.
4. Розробити алгоритм реалізації задачі.
5. Спроектувати базу даних та структуру її таблиць.
6. Реалізувати БД в середовищі MS ACCESS.
7. Реалізувати БД в середовищі MS SQL SERVER.
8. Обґрунтувати вибір технологій розробки.
9. Розробити програмне забезпечення.
10. Розробити модуль імпорту даних із зовнішніх джерел.
11. Розробити web-сайт.

Об’єкт дослідження: футбольна статистика.

Предмет дослідження: автоматизація збору, обробки та аналізу футбольної статистика.

Програмне забезпечення: Microsoft Access, Microsoft SQL SERVER, .NET платформа, Windows Forms, ASP.NET.

# РОЗДІЛ 1 ДОСЛІДЖЕННЯ ЗАДАЧІ АВТОМАТИЗАЦІЇ ЗБОРУ ТА ВІДОБРАЖЕННЯ ФУТБОЛЬНОЇ СТАТИСТИКИ

## 1.1. Характеристика задачі

В наш час усі сфери людського життя описуються у вигляді статистики. Вона широко використовується у повсякденному житті. Завдяки статистичним даним можна завжди проаналізувати ситуацію, порівняти її з іншими, прийняти рішення, провести наукові дослідження. Статистичні дані мають широкий спектр використання. Від допомоги компаніям зрозуміти смаки споживачів до аналізу клінічних досліджень. Знання статистики надає можливість правильно аналізувати: перелік і параметри товарів та послуг, вплив інфляції на ціни й на доходи, цінові пропозиції тощо. Без статистичних даних неможливі управління державою, розвиток економіки і культури, розробка програм розвитку країни. Ці міркування розповсюджуються також і на спорт, зокрема футбол, як на одну з найпоширеніших розваг.

Футбол — це командний вид спорту, в який грають два колективи по одинадцять гравців кожний. Гра проводиться на прямокутному полі з трав'яним або штучним покриттям, на обох кінцях якого розташовані ворота. Мета гри — забити м'яч у ворота суперника, використовуючи для цього ноги, голову або інші частини тіла, крім рук і передпліч. Тільки воротарям дозволяється торкатися м'яча руками, але лише в межах свого штрафного майданчика. Кожен матч складається з двох таймів по 45 хвилин із перервою між ними на 15 хвилин. У разі рівного рахунку після основного часу можуть бути додані два додаткових тайми по 15 хвилин кожний або проведена серія післяматчевих пенальті, залежно від правил турніру.

Правила гри, розроблені Міжнародною радою футбольних асоціацій (IFAB). Вони регулюють аспекти гри, такі як розмір поля, розмір і вага м'яча, кількість гравців, які можуть бути замінені, та правила офсайду. Гра починається з початкового удару в центрі поля, а потім гравці намагаються контролювати

м'яч, передаючи його один одному пасами або ведучи його самостійно, намагаючись уникнути гравців команди суперника. Основна тактична мета — знайти можливість для удару по воротах суперника. Водночас команда, яка захищається, намагається перехопити м'яч і організувати власну атаку. Ключовими елементами футбольної стратегії є командна робота, швидкість, витривалість, технічні навички та тактичне мислення. Кожен гравець має свою специфічну роль на полі, яка визначає його функції та завдання в команді.

Функції гравців на полі

Воротар:

Воротар — це гравець, головне завдання якого полягає у захисті воріт від ударів суперника. Він є єдиним гравцем, якому дозволено використовувати руки в межах свого штрафного майданчика. Воротар має бути швидким, мати хорошу реакцію та вміння правильно вибирати позицію для відбиття ударів.

Захисники:

Захисники поділяються на центральних і крайніх (флангових).Центральні захисники розташовуються в центрі захисту і відповідають за зупинку атак суперника, перехоплення м'яча та блокування ударів. Вони часто є високими і фізично сильними гравцями, здатними ефективно грати головою. Крайні захисники грають ближче до бокових ліній поля. Їхні завдання включають як захист, так і підтримку атаки своєї команди, просуваючись вперед по флангах і виконуючи подачі в штрафний майданчик суперника.

Півзахисники:

Півзахисники є ключовими гравцями, які зв'язують захист і напад. Вони можуть бути розподілені на різні ролі:

1. Опорні півзахисники (або оборонні) розташовуються перед захисниками і допомагають у захисті, перехоплюють м'ячі та розпочинають атаки своєї команди.

2. Центральні півзахисники відповідають за контроль м'яча в центрі поля, розподіляють паси і часто беруть участь у побудові атак.

3. Атакувальні півзахисники грають ближче до нападників і

зосереджуються на створенні гольових моментів, віддаючи точні передачі або самостійно пробиваючи по воротах.

Нападники:

Нападники займаються безпосередньо атакою і забиттям голів. Вони також можуть поділятися на різні ролі:

1. Центральні нападники або форварди грають у центрі атаки і намагаються забивати голи, використовуючи свою швидкість, техніку і вміння грати головою.\

2. Флангові нападники (вінгери) грають на краях поля і використовують свою швидкість та дриблінг, щоб обігравати захисників суперника і виконувати подачі в штрафний майданчик.

Основним завданням даної курсової роботи є вирішення таких функціональних задач, як підрахунок різноманітних дій футболістів, подій у матчі, а також структурування всіх необхідних індивідуальних, командних та інших статистичних даних тощо.

Теорія та практика футбольної професійної статистики є динамічним і перспективним напрямом розвитку сучасного футболу, що сприяє вдосконаленню ігрового та тренувального процесу, а також моніторингу його прогресу на різних рівнях.

Основна задача футбольної статистики – допомогти тренерам приймати стратегічні рішення, оцінювати ефективність гравців і визначати сильні та слабкі сторони команди. Аналітика також дозволяє прогнозувати результативність у майбутніх матчах. Використання великих даних і машинного навчання підвищує рівень гри та сприяє розвитку футболу.

## 1.2. Аналіз існуючого програмного забезпечення

На сьогоднішній день існує велика кількість онлайн-платформ, що надають найрізноманітнішу футбольну статистику. Нижче продемонстровано декілька прикладів таких порталів:

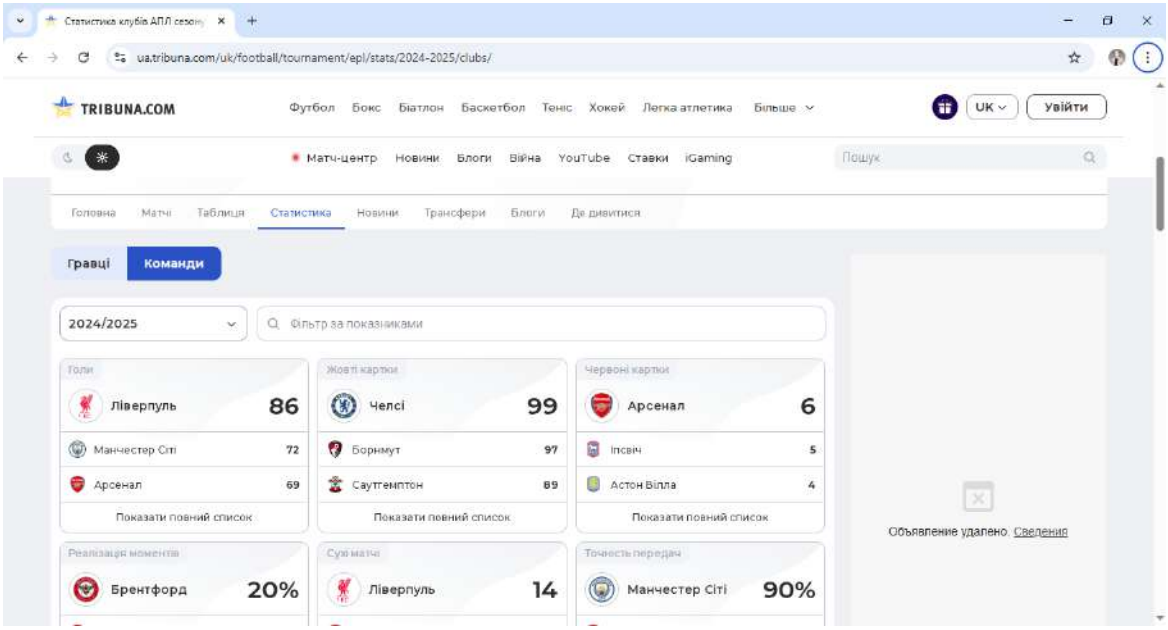



Рис. 1.1. Футбольна статистика на сайті «Трибуна»

На рис. 1.1 продемонстровано приклад статистичних показників команд однієї ліги у сезоні та їх сортування за рейтингом відповідно до параметрів, таких як: забиті м'ячі, дисциплінарні порушення, реалізація гольових моментів, відсоток точних передач, кількість матчів без пропущених м’ячів.

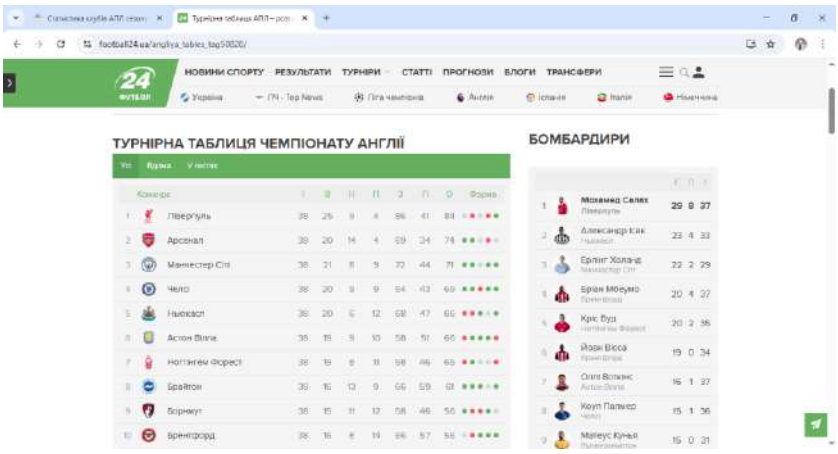



Рис. 1.2. Інтерфейс турнірної таблиці сайту «Football 24»

На рис 1.2 показано  інший приклад статистики з іншого джерела – турнірна таблиця. Вона відображає усі основні параметри за якими і визначається переможець. Найголовнішим індикатором є кіл-ть набраних очок, вони в свою чергу формуються з наступних трьох пунктів: кількість перемог, нічиїх та поразок. А закривають таблицю дані забитих, пропущених голів команди та їх різниця.

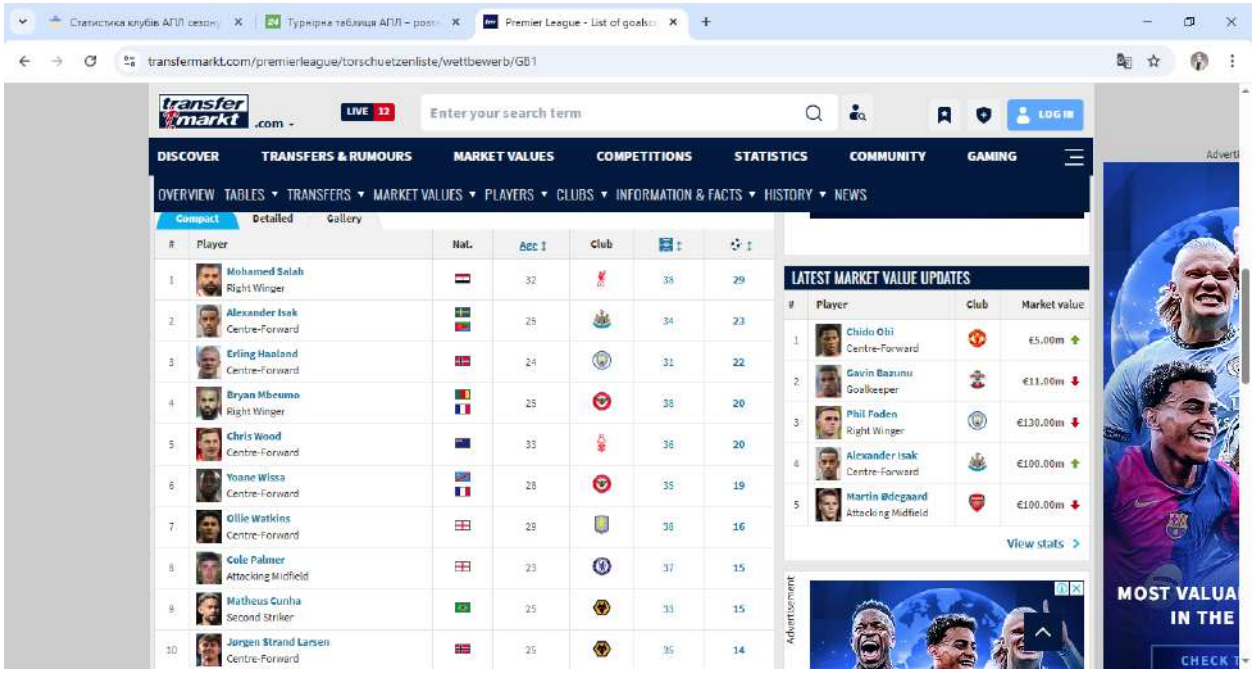



Рис. 1.3. Індивідуальна статистика футболістів на сайті «Transfer markt»

На рис 1.3 показано окрему статистику футболістів, а саме таблицю найкращих бомбардирів турніру. Тут надається уся необхідна інформація про кожного футболіста: національність, вік, клуб за який грає, кількість матчів та кількість забитих м’ячів. Також є опції сортування гравців за віком, позицією, і сезоном, та додаткова кнопка для докладнішої статистики про кожного футболіста (хвилини на полі, хвилини на забитий гол, середні голи за матч і т.д.). Біля імені кожного гравця розташовані фотокартки для більш зручного пошуку потрібного спортсмена та кращого візуального сприйняття інтерфейсу. Теж саме можна сказати і про емблеми клубів та прапори країн. Все це разом забезпечує

приємну картинку, інтуїтивно зрозумілий інтерфейс та багатий функціонал даного продукту.

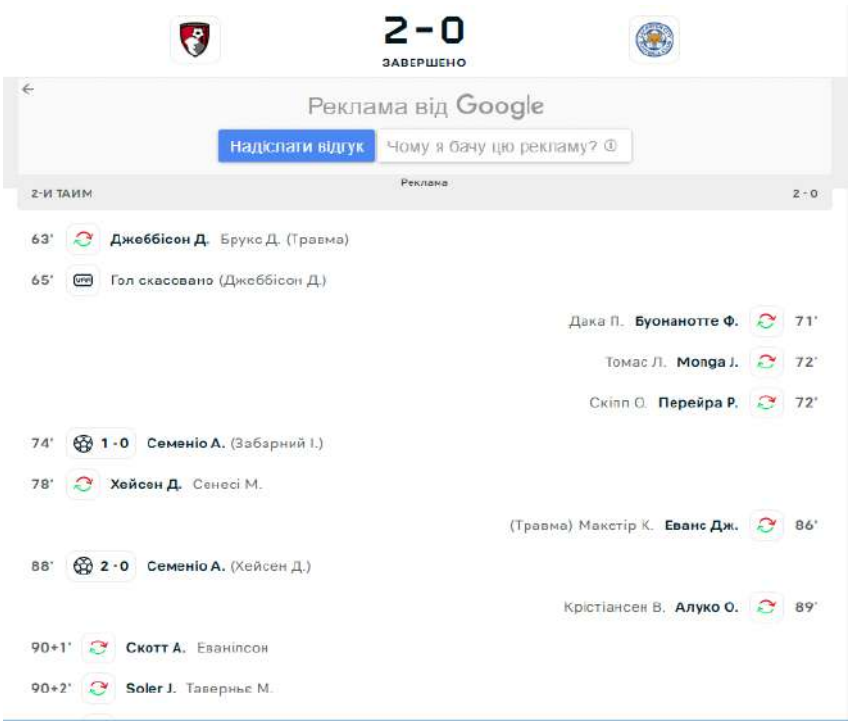



Рис. 1.4 Історія футбольного матчу на сайті «Flashscore»

На рис 1.4 ми бачимо інтерфейс сторінки сайту з грунтовним описом історії футбольного матчу. Уся інформація про події на полі кожного колективу подається симетрично, відповідно до розташування логотипів команд-учасниць, які виведені зверху разом з основною інформацією, а саме: рахунком та статусом матчу. Події у матчі, в свою чергу, мають таку структуру: точна хвилина => іконка події => виконавець(якщо є). Також є графічний розділ першого та другого таймів. Додатково, імена гравці та логотипи команд – є посиланнями. Натиснувши на них ми можемо отримати глибшу інформацію про конкретного гравця чи клуб Така структура є зручною для зчитування інформації з неї будь-яким користувачем.

## 1.3. Аналіз вимог до програмного забезпечення

Згідно з досліджуваною предметною областю було виділено ряд функціональних задач, на вирішення яких буде направлений результат цієї робо-ти. Їх перелік наведений у табл. 1.1.

**Індивідуальні показники гравців**

Індивідуальні показники гравців є фундаментом для оцінки їхньої ефективності та форми. Ці показники включають:

- голи: кількість забитих м'ячів за певний період (матч, сезон);
- асисти: кількість результативних передач, які призвели до голів;
- кількість ударів по воротах: загальна кількість ударів, виконаних гравцем, та їх точність;
- точність пасів: відсоток успішних передач від загальної кількості спроб;
- дриблінг: кількість успішних спроб обіграти суперника один на один;
- захисні дії: кількість відборів, перехоплень, блокувань ударів та інших захисних дій;
- кількість фолів та карток: інформація про дисциплінарні порушення, включаючи жовті та червоні картки тощо.

Таблиця 1.1

Комплекс функціональних задач

| № | Найменування задачі | Призначення задачі | Режим виконання | Періодичність вирішення |
|---|---|---|---|---|
| 1 | Турнірна таблиця | Облік турнірного положення команд у лізі | Запитний | Постійно |
| 2 | Рейтинг найкращих футболістів за системою гол + асист | Аналітичний облік рейтингу бомбардирів | Запитний | Постійно |
| 3 | Рейтинг футболістів за дисциплінарними показниками (жовті та червоні картки) | Аналітичний облік недисциплінованих гравців | Запитний | Постійно |

Нижче будуть показані приклади звітів, які ми маємо отримати на основі індивідуальних показників футболістів (див. табл. 1.2):

Таблиця 1.2

Приклад звіту індивідуальної гольової статистики гравців

| Місце | Гравець | Клуб | Кіл-ть матчів | Голи | Асисти |
|---|---|---|---|---|---|
| 1 | Ерлінг Голанд | Манчестер Сіті | 31 | 27 | 5 |
| 2 | Коул Палмер | Челсі | 34 | 22 | 11 |
| 3 | Александер Ісак | Ньюкасл | 30 | 21 | 2 |
| 4 | Оллі Воткінс | Астон Вілла | 37 | 19 | 13 |
| 5 | Філіп Фоден | Манчестер Сіті | 35 | 19 | 8 |

У табл. 1.2 ми можемо побачити топ-5 найкращих гравців за кількістю голів та гольових передач у сезоні. Також можемо порівняти за яку кількість матчів їм вдалося це зробити та спробувати виявити, яка команда має найрезультативніших гравців. Результати цього звіту є важливим показником атакуючих дій команд та особливо самих футболістів і їх безпосередній вплив на гру.

Таблиця 1.3

Приклад звіту про дисциплінарну статистику футболістів

| Місце | Гравець | Клуб | Кіл-ть ігор | Жовт. Картки |
|---|---|---|---|---|
| 1 | Маркос Сенесі | Борнмут | 31 | 13 |
| 2 | Жоау Палінья | Фулгем | 33 | 13 |
| 3 | Дуглас Луїс | Астон Вілла | 35 | 12 |
| 4 | Джеймс Тарковскі | Евертон | 38 | 11 |
| 5 | Нельсон Семеду | Вулверхемптон | 36 | 11 |

Таблиця 1.4

Топ-5 гравців з найбільшою кількістю червоних карток за сезон

| Місце | Гравець | Клуб | Кіл-ть ігор | Черв. Картки |
|---|---|---|---|---|
| 1 | Ів Біссума | Тоттенгем | 28 | 2 |
| 2 | Олівер Макберні | Шеффілд Юнайтед | 21 | 2 |
| 3 | Ріс Джеймс | Челсі | 10 | 2 |
| 4 | Джо Уоррелл | Ноттінгем Форест | 7 | 1 |
| 5 | Лайл Фостер | Бернлі | 24 | 1 |

У цих таблицях (див. табл. 1.3, 1.4) наведені приклади гравців з найбільшою кількістю попереджень та вилучень, тобто жовтих та червоних карток. Аналізуючи цей звіт можна помітити, що основна частина футболістів мають захисні позиції. Їм доводиться зупиняти атаки суперника, тому іноді це доводиться робити навіть ціною жовтих та червоних карток. Завдяки цим показникам можна побачити стиль гри у захисті кожного гравця, його функції на полі, та іноді настанови тренерів.

**Командні показники**

Командні показники дозволяють оцінити загальну продуктивність команди та її тактичні особливості. Основні командні показники включають:

- володіння м'ячем: відсоток часу, протягом якого команда контролює м'яч під час матчу;
- кількість атак: кількість атакуючих дій, що включають просування м'яча до зони суперника;
- кількість ударів по воротах: загальна кількість ударів, виконаних командою за матч;
- процент реалізації моментів: співвідношення забитих голів до загальної кількості створених гольових моментів;
- захисні показники: кількість відборів, перехоплень, блокувань та інших захисних дій, виконаних командою;
- точність передач: середній відсоток успішних передач всієї команди.

Розглянемо приклади звітів з показниками командної статистики:

Таблиця 1.5

Приклад рейтингу командної статистики за атакуючими показниками

| Місце | Команда | Турнір | Голи | Удари за гру | Володіння % |
|---|---|---|---|---|---|
| 1 | Баєр | Бундесліга | 89 | 18,2 | 62.1% |
| 2 | Манчестер Сіті | АПЛ | 96 | 18,20 | 65.4% |
| 3 | Баварія | Бундесліга | 94 | 18,8 | 61.3% |
| 4 | Реал Мадрид | ЛаЛіга | 87 | 15,7 | 59.3% |
| 5 | ПСЖ | Ліга 1 | 81 | 15,1 | 65.7% |

У табл. 1.5 продемонстровано рейтинг футбольних клубів з найкращими атакуючими статистичними показниками без прив’язки до ліги. Найхарактернішими індикаторами є: голи (Що суттєво відображають атакувальну спрямованість команд.), удари за гру та відсоток володіння м’ячем.

Таблиця 1.6

Приклад рейтингу командної статистики за захисними показниками

| Місце | Команда | Турнір | Допущ. уд.з.гр | Відбри з. гр. | Перехоплення з. гр. |
|---|---|---|---|---|---|
| 1 | Крістал Пелес | АПЛ | 12,1 | 20,7 | 8,5 |
| 2 | Брест | Ліга 1 | 10,4 | 20,0 | 9,4 |
| 3 | Евертон | АПЛ | 14,2 | 19,6 | 9,4 |
| 4 | Монако | Ліга 1 | 11,6 | 19,5 | 11,6 |
| 5 | Вулверхемптон | АПЛ | 15,2 | 19,4 | 7,5 |

У табл. 1.6 представлено рейтинг команд за найкращими захисними показниками. Основними складовими захисту тут виступають такі параметри: допущені удари з гри, відбори м’яча з гри, перехоплення.

**Додаткові метрики**

Крім основних індивідуальних та командних показників, існує ряд додаткових метрик, які допомагають більш детально аналізувати гру:

- дистанція пробігу: кількість кілометрів, яку пробігає гравець або команда за матч. Ця метрика дозволяє оцінити фізичну підготовку та

витривалість;

- швидкість: максимальна швидкість, яку досягав гравець під час матчу, та середня швидкість переміщення;

- кількість фолів: загальна кількість порушень правил, які були зафіксовані у матчі;

- кількість виграних повітряних дуелей: кількість успішних спроб виграти боротьбу за м'яч у повітрі;

- ефективність стандартних положень: відсоток успішних реалізацій стандартних положень (кутові, штрафні удари, вільні удари).

Таблиця 1.7

Приклад турнірної таблиці

| Місце | Команда | Кіл-ть матчів | Перемоги | Нічії | Поразки | Забиті гол. | Пропущені гол. | Різниця | Очки |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Манчестер Сіті | 38 | 28 | 7 | 3 | 96 | 34 | 62 | 91 |
| 2 | Арсенал | 38 | 28 | 5 | 5 | 91 | 29 | 62 | 89 |
| 3 | Ліверпуль | 38 | 24 | 10 | 4 | 86 | 41 | 45 | 82 |
| 4 | Астон Вілла | 38 | 20 | 8 | 10 | 76 | 61 | 15 | 68 |
| 5 | Тоттенгем | 38 | 20 | 6 | 12 | 74 | 61 | 13 | 66 |
| 6 | Челсі | 38 | 18 | 9 | 11 | 77 | 63 | 14 | 63 |
| 7 | Ньюкасл | 38 | 18 | 6 | 14 | 85 | 62 | 23 | 60 |
| 8 | Манчестер Юнайтед | 38 | 18 | 6 | 14 | 57 | 58 | -1 | 60 |
| 9 | Вест Гем | 38 | 14 | 10 | 14 | 60 | 74 | -14 | 52 |
| 10 | Крістал Пелес | 38 | 13 | 10 | 15 | 57 | 58 | -1 | 49 |
| 11 | Брайтон | 38 | 12 | 12 | 14 | 55 | 62 | -7 | 48 |
| 12 | Борнмут | 38 | 13 | 9 | 16 | 54 | 67 | -13 | 48 |
| 13 | Фулгем | 38 | 13 | 8 | 17 | 55 | 61 | -6 | 47 |
| 14 | Вулверхемптон | 38 | 13 | 7 | 18 | 50 | 65 | -15 | 46 |
| 15 | Евертон | 38 | 13 | 9 | 16 | 40 | 51 | -11 | 40 |
| 16 | Брентфорд | 38 | 10 | 9 | 19 | 56 | 65 | -9 | 39 |
| 17 | Ноттінгем Форест | 38 | 9 | 9 | 20 | 49 | 67 | -18 | 32 |
| 18 | Лутон | 38 | 6 | 8 | 24 | 52 | 85 | -33 | 26 |
| 19 | Бернлі | 38 | 5 | 9 | 24 | 41 | 78 | -37 | 24 |
| 20 | Шеффілд Юнайтед | 38 | 3 | 7 | 28 | 35 | 104 | -69 | 16 |

Як видно з табл. 1.7 — найпростішою формою футбольної бази даних та візуалізації її значень є турнірна таблиця.

Футбольна база даних — це необхідний елемент футбольної статистики та її аналітики, яка слугує для систематизації, обробки та візуалізації статистичного матеріалу. Статистичний та аналітичний матеріал окремих матчів заноситься до єдиної футбольної бази даних. Це дозволяє апелювати великим масивом даних, забезпечувати його фільтрацію та сортування на різному рівні, проводити моніторинг розвитку окремих гравців, команд, чемпіонатів розраховувати різного роду індекси та рейтинги тощо. Вона враховує статистику основних показників кожної з команд у ході турніру.

На основі аналізу системи оперативних первинних, довідкових, допоміжних та вихідних документів, що належать до комплексу , створюється їх перелік та надаються загальні їх властивості у вигляді таблиці (табл. 1.8).

Таблиця 1.8

Опис документообігу

| № | Назва документу | Джерело надходження або місце передачі | Частота находження, формування або відновлення |
|---|---|---|---|
| Оперативні первинні документи | | | |
| 1 | Протокол результату матчу | Керівництво футбольної ліги | В міру прийняття |
| 2 | Суддівський протокол | | В міру прийняття |
| 3 | Заявки футболістів на матч | Менеджмент футбольного клубу | В міру прийняття |
| Довідкові документи | | | |
| 4 | Особиста картка футболіста | Менеджмент футбольного клубу | Ведеться постійно |
| Вихідні документи | | | |
| 5 | Турнірна таблиця | Керівництво футбольної ліги | Щотижня |

У табл. 1.8 показано опис футбольного документообігу. Він складається з

первинних, довідкових, та вихідних документів. Кожен із них адмініструється відповідними офіційними структурами (Лігою, організаторами турніру, адміністрацією футбольного клубу), та мають свій облік і частоту відновлення.

Проаналізувавши всі необхідні документи, виділяються реквізити документів, які будуть зберігатися чи розраховуватися у базі даних. Їх перелік наведено у табл. 1.9.

Таблиця 1.9

Таблиця реквізитів документів

| № | Назва реквізиту | Тип реквізиту | Зберігання у БД |
|---|---|---|---|
| Футбольні клуби | | | |
| 3 | Назва клубу | Строковий | Так |
| 4 | Місто | Строковий | |
| 5 | Країна | Строковий | |
| 6 | Ліга | Числовий | |
| Футболісти | | | |
| 7 | Прізвище | Строковий | Так |
| 8 | Ім’я | Строковий | |
| 9 | Дата народження | Дата | |
| 10 | Позиція | Строковий | |
| 11 | Клуб | Строковий | |
| 12 | Ліга | Строковий | |
| Матчі | | | |
| 24 | Команда, що приймає | Строковий | Так |
| 25 | Команда гість | Строковий | |
| 26 | Рахунок матчу | Строковий | |
| 28 | Дата матчу | Дата | |
| Протокол замін матчів | | | |
| 30 | Матч | Строковий | Так |
| 31 | Клуб | Строковий | |
| 32 | Гравець замінений | Строковий | |
| 33 | Гравець на заміну | Строковий | |
| 36 | Час заміни(хв) | Час | |

У табл. 1.9 показано дані таких документів як: футболісти, футбольні клуби, матчі, протокол замін матчу. Вказано тип даних кожного з полів і статус

чи зберігаються реквізити у базі даних.

Футбольна статистика є ключовим елементом аналізу продуктивності як індивідуальних гравців, так і команд в цілому. Збір та обробка цих даних дозволяють тренерам, аналітикам та менеджерам приймати обґрунтовані рішення щодо стратегії гри, підбору гравців та їх тренування.

Для детального аналізу статистики необхідно зібрати вхідні дані, які охоплюють різноманітні аспекти.

**Ліги**

Усе починається з футбольних ліг та турнірів. Вони включають у себе:

- назву ліги;
- країну;
- кількість команд;
- кількість матчів;
- формат;
- систему очок тощо.

Усі ці відомості про футбольну лігу, є невід'ємною частиною широкого масиву даних, які використовуються для всебічного аналізу та ведення статистики. Ці відомості пов'язують лігу з іншими ключовими компонентами, такими як команди, матчі та гравці, кожен з яких має свої специфічні дані, необхідні для комплексного розуміння функціонування ліги.

**Команди**

Інформація про команди є основою статистики футбольної ліги. Кожна команда має свої особливості та характеристики, які включають:

– назву команди;

– розташування: географічне положення команди, місто, де вона базується;

– стадіон: інформація про домашній стадіон, його місткість, архітектурні особливості та модернізації;

– тренерський штаб: дані про головного тренера, його помічників, тренерів воротарів та інших спеціалістів.

Командні дані також включають тактичну інформацію, такі як типи формацій, які використовуються під час матчів, і стратегічні підходи до гри. Ці дані є критично важливими для аналізу продуктивності команди та розробки тренувальних програм.

**Матчі**

Статистика матчів є ключовим компонентом даних, зібраних про футбольну лігу. Вона включає:

- результати матчів: підсумкові рахунки, кількість перемог, поразок і нічиїх кожної команди протягом сезону;
- голи: кількість забитих і пропущених м'ячів у кожному матчі, хто і коли забив гол;
- ігрові ситуації: хвилини, коли були забиті голи, частота голів у перших і других таймах, кількість пенальті, автоголів та інших важливих моментів;
- статистика володіння м'ячем: відсоток часу, який кожна команда володіла м'ячем під час матчів;
- фізичні дані: відстань, яку пробігла кожна команда, інтенсивність гри, кількість спринтів;

**Гравці**

Індивідуальні дані про гравців є важливими для оцінки їхнього внеску в команду і загальної продуктивності. Ці дані включають:

1. Особисті дані: ім'я, дата народження, національність, зріст, вага.
2. Позиція на полі: основна позиція, на якій грає гравець (нападник, півзахисник, захисник, воротар), а також додаткові позиції.
3. Контрактні дані: Інформація про контракт, заробітну плату, термін контракту, умови трансферу.

Висновки до розділу 1:

Було проведено аналіз підходів до автоматизації збору та обробки футбольної статистики. Було визначено основні характеристики команд та

футболістів при здійснені аналізу футбольної інформації. Здійснено критичний огляд найбільш популярних ресурсів. В ході аналізу визначено перелік задач та модулів, які притаманні веб-сайтам з аналізу футбольної статистики. Було формалізовано вхідну інформацію для аналізу, а також визначено перелік і структуру звітів, які формують при аналізі футбольної статистики.

# РОЗДІЛ 2 РОЗРОБКА АЛГОРИТМУ ВИРІШЕННЯ ЗАДАЧІ

Аналітичні дані у сучасному футболі – те, без чого вже неможливо уявити гру номер один. Кожен уболівальник і фахівець зараз може знайти статистику на будь-який смак та гаманець – від звичайних показників до найпросунутіших метрик, на кшталт очікуваних голів (xG), просування м'яча (Packing) та трекінгових даних, які дозволяють отримувати інформацію про бігову роботу футболістів.

Якщо розділити статистичні дані на категорії, то можна виділити три основні: базова статистика по матчу, дані про події та трекінгові дані.

1. Статистика матчу - те, що на самій поверхні. Це склади, розміщення, заміни, картки, голи і т.д.

2. Дані про події (Events Data) – впорядкований набір даних, що описує послідовність дій гравців з м'ячем (удари, передачі, єдиноборства, обведення, перехоплення, підбирання тощо). Такі дані систематизують розбирачі (анотатори), які аналізують відеозаписи матчів та формують відповідні звіти. Таку роботу роблять аналітичні компанії Opta Sports, Wyscout, Instat, StatsBomb та інші.

3. Трекінгові дані (Tracking data) – точні просторові координати всіх гравців та м'яча у кожний момент часу матчу. Ці дані збираються спеціалізованими оптичними системами відстеження переміщень футболістів на основі камер, які встановлюються на стадіоні. Найвідомішими постачальниками таких даних є компанії ChyronHego, STATS Perform, BallJames та Second Spectrum.

Кожна статистична компанія має власний алгоритм підрахунку даних. Тому іноді той самий параметр у двох різних компаній може відрізнятися. Наприклад, WyScout вважає відбори та підбирання в один осередок, інші компанії розмежовують ці дії. Існують різні методики оцінки того чи іншого епізоду – наприклад, часто платформи дають різні цифри за обведеннями та загострюючими передачами, просто підхід до одного епізоду у двох компаній

може бути різним.

WyScout також використовує комбіновану систему підрахунку – спеціальна програма обраховує інформацію, а людина все контролює. Компанія має офіси в різних країнах, де працюють «розбирачі» – в Україні, Росії, Молдові, Болгарії.

У американського гіганта Hudl (до цієї компанії входить WyScout) є сервіс, що дозволяє тренерському штабу розробити власну методологію. Завдяки платформі Sportscode тренер може залити відео та розібрати його з погляду потрібних йому параметрів. Їх може бути всього п'ять, а не 100, як у звичайному звіті WyScout – аж до того, що там може бути інформація про те, скільки разів півзахисник повернув головою перед м'ячем. Тут ти сам знаєш матч, тим самим знижуючи відсоток похибки. В Україні Sportscode використовують шість клубів УПЛ.

Стрімкий прогрес аналітики і, зокрема даних про події, дав нам одну з найпопулярніших метрик на сьогоднішній день – очікувані голи (xG). Ця метрика дозволяє вимірювати якість моментів, створюваних гравцями, і безпосередньо оцінює ймовірність голу після кожного удару в діапазоні від 0 до 1.

Вся справа у різних моделях підрахунку очікуваних голів, які використовують компанії. Немає єдиної методики, кожен має свою. Показник небезпеки удару обчислюється, ґрунтуючись на різних даних: відстань до воріт, тип удару, позиція для удару, спосіб і місце передачі під удар. Навіть враховується швидкість атаки, дистанція, подолана футболістом з м'ячем, дії перед ударом (обведення, удар в один дотик), рахунок поєдинку, перешкоди з боку захисників, кількість захисників перед б'ючим і т.д. І у кожного ці змінні можуть бути різними та мати різну вагу.

Наприклад, Instat ділить удари по категоріях - ногою, головою, іншою частиною тіла. Також розглядає стандартні положення та окремо – пенальті. У удару з 11-метрової позначки коефіцієнт 0,75.

Важливо розуміти, що сумарний xG команди в одному конкретному матчі не обов'язково повинен відображати кількість голів, які мала забити команда. Вона показує нам, скільки небезпечних моментів було створено і скільки мало бути з них голів у матчі. Ця метрика значно ефективніша на дистанції.

Ще однією популярною у футбольних колах метрикою, особливо у Східній Європі, є Instat Index. По суті це оцінка гравця за матч, яка вираховується за складною схемою, розробленою компанією.

Для воротарів Index розраховується на основі приблизно 15 показників (гра на виходах, парірувані удари, паси тощо). Для польових гравців індекс розраховується на основі приблизно 30 показників (удари, єдиноборства, фоли, відбори тощо).

Instat Index не використовується в аналітиці, він частіше використовується у скаутингу. В одному окремо взятому матчі Index – це оцінка. Але якщо ми беремо цей показник на дистанції, то можемо побачити як футболіст виступає в динаміці. Можна побудувати діаграму, наочно побачити, як футболіст грає у домашніх чи гостьових матчах окремо.

## Висновки до розділу 2

Було проаналізовано підходи та алгоритми розрахунку статистичних показників для аналізу ефективності виступів гравців у матчах та аналізу їх фізичної форми.

# РОЗДІЛ 3 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

## 3.1. Структура бази даних

Першим етапом створення проекту бази даних для обраної предметної області є побудова моделі. Хоча реляційна модель даних надає достатньо можливостей для моделювання предметної області, цей спосіб значно ускладнює процес проектування бази даних, а також його розуміння кимось окрім спеціалістів з БД. Тому зазвичай для цього використовується інфологічна модель. Вона будується без орієнтації на конкретну СКБД, саме тому цей етап настільки важливий на початку проектування.

До інфологічної моделі висуваються наступні вимоги:

- однозначність;
- достатність інформації;
- гнучкість (для можливих майбутніх змін);
- зрозумілість;

У результаті аналізу предметної області повинні бути виділені сутності (об'єкти), інформація про яких повинна бути представлена в базі даних, а також властивості цих сутностей і зв'язку між ними.

Таким чином, на початковому етапі побудови інфологічної моделі предметної області вирішуються наступні завдання:

- в предметної області виділяються об'єкти-сутності, інформація про які повинна буде фіксуватися в базі даних;
- визначаються властивості, що дозволяють ідентифікувати окремі екземпляри виділених об'єктів-сутностей;
- виділяються властивості, що дозволяють об'єднувати окремі об'єкти (екземпляри) в абстрактні сутності, що є класами чи типами об'єктів.

В результаті аналізу предметної області було виділено інформаційні об’єкти, перелік яких надано у табл. 3.1.

Таблиця 3.1

Опис характеристики об’єктів

| Найменування об'єкта | Спосіб звертання до екземплярів | Структурна активність | Обмеження на право звертання | Кількість екземплярів | Примітка |
|---|---|---|---|---|---|
| Ліги | К (код ліги) | — | Менеджер ФК | 0 – 2^32-1 | — |
| Список футболістів | К (код футболіста) | — | Менеджер ФК | 0 – 2^32-1 | — |
| Список футбольних клубів | К(код клубу) | — | Менеджер ФК | 0 – 2^32-1 | — |
| Список позицій футболістів | К(код позиції) | — | Менеджер ФК | 0 – 2^32-1 | — |
| Список Матчів | К(код матчу) | — | Менеджер ФК | 0 – 2^32-1 | — |
| Історії матчів | К(код історії матчу) | — | Менеджер ФК | 0 – 2^32-1 | — |
| Список таймів | К(код тайму) | — | Менеджер ФК | 0 – 2^32-1 | — |
| Список типів подій у матчі | К(код типу події) | — | Менеджер ФК | 0 – 2^32-1 | — |

Як видно з табл. 3.1, складові елементи інформаційних об’єктів є реквізитами документів, аналіз яких був проведений раніше, що будуть зберігатися в комп’ютерній базі даних.

Окрім опису інформаційних об’єктів і їх властивостей необхідно виконати опис структурних зв’язків між об’єктами. Перелік зв’язків наведено у таблиці

Для реалізації проекту бази даних було обрано наступні СКБД: MS Access та MS SQL Server. В структурі таблиць також приведена можливість реалізації БД для СУБД MySQL.

Microsoft Access відрізняється простотою використання та інтеграцією з іншими програмами Microsoft Office. Його інтуїтивний інтерфейс дозволяє користувачам швидко створювати та керувати базами даних. Access ідеально підходить для малих і середніх підприємств завдяки своїм можливостям для швидкого розвитку додатків [9,18,20,23].

Microsoft SQL Server – це потужна реляційна система управління базами даних, що забезпечує високу продуктивність, масштабованість та безпеку. Вона

підходить для великих корпоративних додатків і підтримує різноманітні типи даних, транзакції та складні запити. SQL Server пропонує інтегровані інструменти для аналізу даних, звітності та бізнес-аналітики, що робить його універсальним рішенням для великих організацій [2,4,8,10,11,14].

Властивості стовпчиків отриманих таблиць описані в табл. 3.2

Таблиця 3.2

Опис властивостей таблиці «Ліги»

| № п/п | Назва стовпчика | Ім’я поля | Тип даних | | | Властивості (первинний ключ, зовнішній ключ та інші обмеження) |
|---|---|---|---|---|---|---|
| | | | Access | MySQL | SQL | |
| Таблиця «Ліги» | | | | | | |
| 1 | Код ліги | Код ліги(league_id) | Long Integer | Integer | int | PK, Not null |
| 2 | Назва ліги | Назва ліги (League name) | Text(255) | varchar(255) | varchar(255) | Not null |
| 3 | Країна(регіон) | Країна(регіон), Country(region) | Text(255) | varchar(255) | varchar(255) | Not null |

Як видно з табл. 3.2 в якій подається опис властивостей стовпчиків таблиці бази даних «Ліги» -- структура має 3 поля. Перший стовпчик цієї таблиці бази даних має назву «Код ліги», а саме поле – league_id. Тип даних буде -- int, оскільки воно виконує роль цілочисельного лічильника, а також є ключовим. Другий стовпчик таблиці має назву «Назва ліги», а саме поле -- «league_name».Тип даних буде рядок (varchar). Третій стовпчик таблиці має назву «Країна», а саме поле -- «country».Тип даних буде -- рядок (varchar).

Таблиця 3.3

Опис властивостей таблиці «Футбольні клуби»

| № п/п | Назва стовпчика | Ім’я поля | Тип даних | | | Властивості (первинний ключ, зовнішній ключ та інші обмеження) |
|---|---|---|---|---|---|---|
| | | | Access | MySQL | SQL | |
| 1 | Код клубу | Код клубу(club_id) | Long Integer | Integer | int | PK, Not null |
| 2 | Назва клубу | Назва клубу(Club name) | Text(255) | varchar(255) | varchar(255) | Not null |
| 3 | Місто | Місто(City) | Text(255) | varchar(255) | varchar(255) | Not null |
| 4 | Країна | Країна, (Country) | Text(255) | varchar(255) | varchar(255) | Not null |
| 5 | Ліга | Ліга, (League) | Long Integer | Integer | int | FK, Not null |

Як видно з табл. 3.3 в якій подається опис властивостей полів таблиці бази даних «Футбольні клуби» — структура має 5 полів. Перший стовпчик цієї таблиці бази даних має назву «Код клубу», а саме поле – club_id. Тип даних буде — int, оскільки воно виконує роль цілочисельного лічильника, а також є ключовим. Другий стовпчик таблиці має назву «Назва клубу», а саме поле — «club_name».Тип даних — текстовий. Третій стовпчик таблиці має назву «Місто», а саме поле — «city».Тип даних — текстовий. Четверте і п’яте поле мають назву «Країна» і «Ліга» відповідно, та мають текстовий тип даних і ціле число (int). Поле «Ліга» також є зовнішнім ключем. Ця таблиця є одною з найважливіших, бо забезпечує основу зв’язків між іншими структурами: «Футболісти», «Матчі» і т.д., попри те, що вона є лише таблицею для зберігання структурованої інформації про футбольні колективи, які є ключовими одиницями усієї статистики. А завдяки своїм зв’язкам, вона полегшує виконнання запитів щодо складів команд, стаитистики матчів і т.д.

Таблиця 3.4

Опис властивостей таблиці «Футболісти»

| № п/п | Назва стовпчика | Ім’я поля | Тип даних | | | Властивості (первинний ключ, зовнішній ключ та інші обмеження) |
|---|---|---|---|---|---|---|
| | | | Access | MySQL | SQL | |
| 1 | Код футболіста | Код футболіста (player_id) | Long Integer | Integer | int | PK, Not null |
| 2 | Ім’я | Ім’я ( Player name) | Text(255) | varchar(255) | varchar(255) | Not null |
| 3 | Прізвище | Прізвище (Last name) | Text(255) | varchar(255) | varchar(255) | Not null |
| 4 | Дата народження | Дата народження (Birth_date) | Date/Time | Date | datetime | Not null |
| 5 | Позиція | Позиція (Position) | Long Integer | Integer | int | FK, Not null |
| 6 | Клуб | Клуб (Club) | Long Integer | Integer | int | FK, Not null |

У табл. 3.4 продемонстровано опис властивостей таблиці «Футболісти». Вона складається з 6 полів. Перший стовпчик цієї таблиці бази даних має назву «Код футболіста», а саме поле – player_id. Тип даних буде — int, оскільки воно виконує роль цілочисельного лічильника, а також є ключовим. Другий стовпчик таблиці має назву «Ім’я», а саме поле — «player_name».Тип даних — текстовий. Третій стовпчик таблиці має назву «Прізвище», а саме поле — «last_name».Тип даних — текстовий. Четверте поле містить дату народження та має однойменну назву стовпця, а назва самого поля – «birth_date» з типом даних дата. У п’ятому полі записано позицію футболіста, код якої береться з іншої структури, а отже поле є зовнішнім ключем. Стовпець вихідної таблиці називається «Позиція», а поле – «position» з типом даних int. Шосте поле також є зовнішнім ключем, що

містить у собі код клубу гравця, тож стовпець таблиці має назву «Клуб», а саме поле – «club» з типом даних int.

Таблиця 3.5

Опис властивостей таблиці «Позиції футболістів»

| № п/п | Назва стовпчика | Ім’я поля | Тип даних | | | Властивості (первинний ключ, зовнішній ключ та інші обмеження) |
|---|---|---|---|---|---|---|
| | | | Access | MySQL | SQL | |
| 1 | Код позиції | Код позиції (position_id) | Long Integer | Integer | int | PK, Not null |
| 2 | Назва позиції | Назва позиції (Position name) | Text(255) | varchar(255) | varchar(255) | Not null |

Як видно з табл. 3.5, де ми бачимо опис властивостей таблиці «Позиції футболістів» – вона має незначну кількість інформації, а саме: поле «position_id» з типом даних int і властивістю первинного ключа, та поле «position name» з текстовим типом даних. Стовпці таблиці називаються «Код позиції» і «Назва позиції». Як зрозуміло з імен властивостей – ця таблиця зберігає у собі ідентифікатори та назви позицій футболістів на полі.

Таблиця 3.6

Опис властивостей таблиці «Тайми»

| № п/п | Назва стовпчика | Ім’я поля | Тип даних | | | Властивості (первинний ключ, зовнішній ключ та інші обмеження) |
|---|---|---|---|---|---|---|
| | | | Access | MySQL | SQL | |
| 1 | Код тайму | Код тайму (period_id) | Long Integer | Integer | int | PK, Not null |
| 2 | Назва тайму | Назва тайму (Period name) | Text(255) | varchar(255) | varchar(255) | Not null |

У табл. 3.6 показано опис властивостей таблиці тайми. Ця таблиця містить в собі інформацію про кожен з таймів у двох полях: «period_id» (певинний ключ, тип даних int, властивість лічильник) та «period_name» (текстовий тип даних).

Таблиця 3.7

Опис властивостей таблиці «Типи подій матчу»

| № п/п | Назва стовпчика | Ім’я поля | Тип даних | | | Властивості (первинний ключ, зовнішній ключ та інші обмеження) |
|---|---|---|---|---|---|---|
| | | | Access | MySQL | SQL | |
| 1 | Код події | Код події (event_id) | Long Integer | Integer | Int | PK, Not null |
| 2 | Подія | Подія (Event) | Text(255) | varchar(255) | varchar(255) | Not null |

У табл. 3.7 показано опис властивостей таблиці «Типи подій матчу», що є довідником з полями («event_id», «event»), які безпосередньо зберігають типи подій матчу: голи, гольові передачі, червоні та жовті картки, заміни тощо.

Таблиця 3.8

Опис властивостей таблиці «Матчі»

| № п/п | Назва стовпчика | Ім’я поля | Тип даних | | | Властивості |
|---|---|---|---|---|---|---|
| | | | Access | MySQL | SQL | |
| 1 | Код матчу | Код матчу (match_id) | Long Integer | Integer | int | PK, Not null |
| 2 | Команда 1 | Команда 1 (Team 1) | Long Integer | Integer | int | FK, Not null |
| 3 | Команда 2 | Команда 2 ( Team 2) | Long Integer | Integer | int | FK, Not null |
| 4 | Рахунок 1 | Рахунок 1 (Score 1) | Long Integer | Integer | int | Not null |
| 5 | Рахунок 2 | Рахунок 1 ( Score 2) | Long Integer | Integer | int | Not null |
| 6 | Дата матчу | (Match date) | Date/Time | Date | datetime | Not null |

З табл. 3.8 ми бачимо опис властивостей таблиці «Матчі». У ній використовується зв’язка з іншою таблицею -- «Футбольні клуби», з якої ми беремо ідентифікатори  та назви клубів, що грають матч та записуємо у відповідні поля. Крім команд ця таблиця містить результат, тобто кількість голів , що забила перша і друга команди, дані про які зберігаються у полях з назвою стовпчика «Рахунок 1» і «Рахунок 2». Останньою властивістю таблиці є дата матчу.

Таблиця 3.9

Опис властивостей таблиці «Історія матчів»

| № п/п | Назва стовпчика | Ім’я поля | Тип даних | | | Властивості (первинний ключ, зовнішній ключ та інші обмеження) |
|---|---|---|---|---|---|---|
| | | | Access | MySQL | SQL | |
| 1 | Матч | Матч (Match) | Long Integer | Integer | int | FK, Not null |
| 2 | Клуб | Клуб (Club) | Long Integer | Integer | int | FK, Not null |
| 3 | Гравець | Гравець (Player) | Long Integer | Integer | int | FK, Not null |
| 4 | Гравець на заміну | Гравець на заміну (Change) | Long Integer | Integer | int | FK, Not null |
| 5 | Подія | Подія (Event) | Long Integer | Integer | int | FK, Not null |
| 6 | Тайм | Тайм (Period) | Long Integer | Integer | int | FK, Not null |
| 7 | Час події | Час події (Event time) | Date/Time | Date | Datetime | Not null |

Як видно з табл. 3.9, де описуються властивості таблиці «Історія матчів», увесь набір даних є найбільшим за усі попередні структури. Такий висновок витікає з того, що завдання цієї таблиці – виводити глибоку, детальну статистику матчу. Тому вона має найбільшу кількість зв’язків і дані у майже кожному полі береться з інших таблиць.

## 3.2. Розробка бази даних в середовищі СУБД MS Access

Для вирішення вказаних функціональних задач у MS Access було застосовано запити, таблиці, форми та звіти.

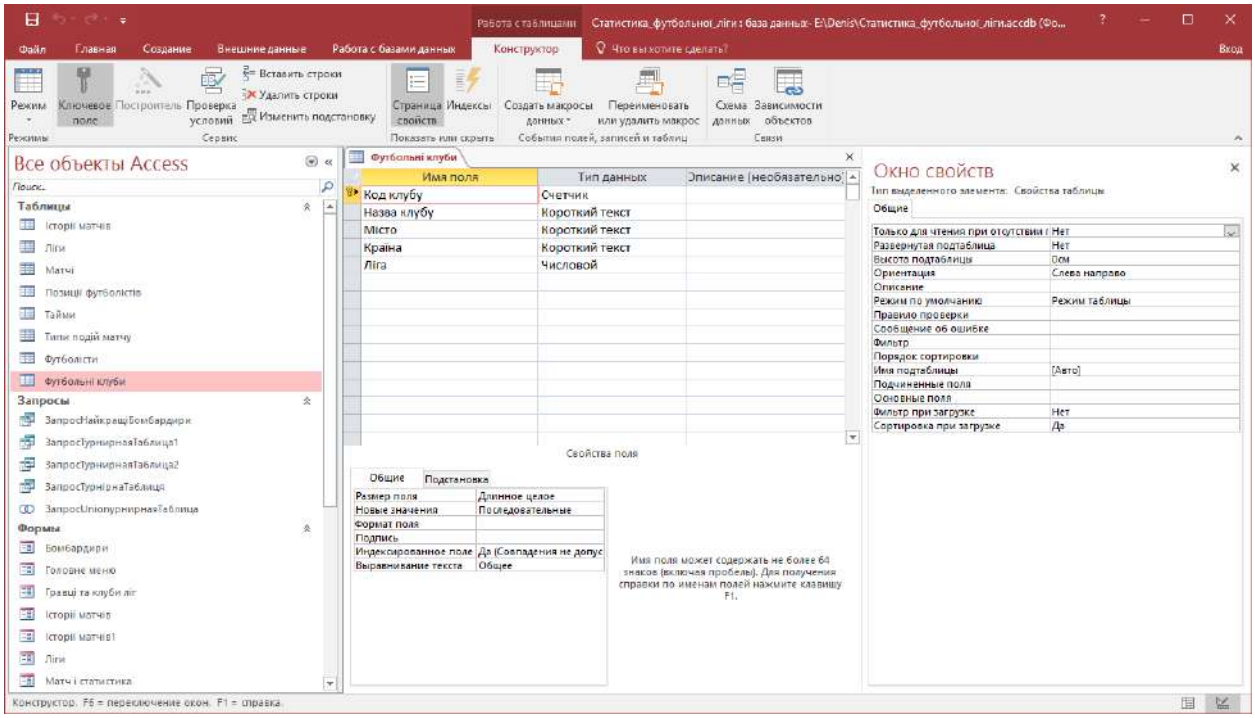



Рис. 3.1. Структура таблиці «Футбольні клуби»

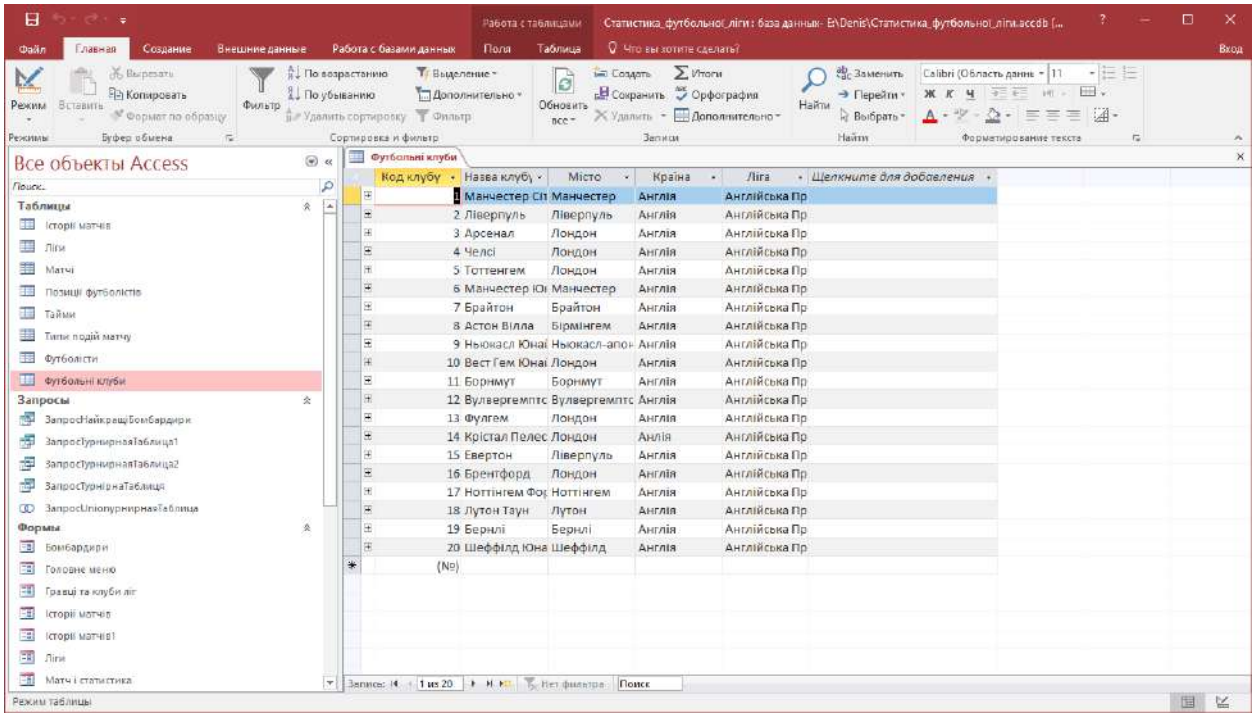



Рис. 3.2. Таблиця «Футбольні клуби»

На рис. 3.1 та 3.2 показано структуру таблиці “Футбольні клуби” та її наповнення. Першим та ключовим(внутрішнім) є поле «Код клубу», типом даних якого є «Лічильник». Наступні поля: «Назва команди», «Місто», «Країна» є текстовими . Поле «Ліга» має числовий тип даних та є зовнішнім ключем у даній таблиці.

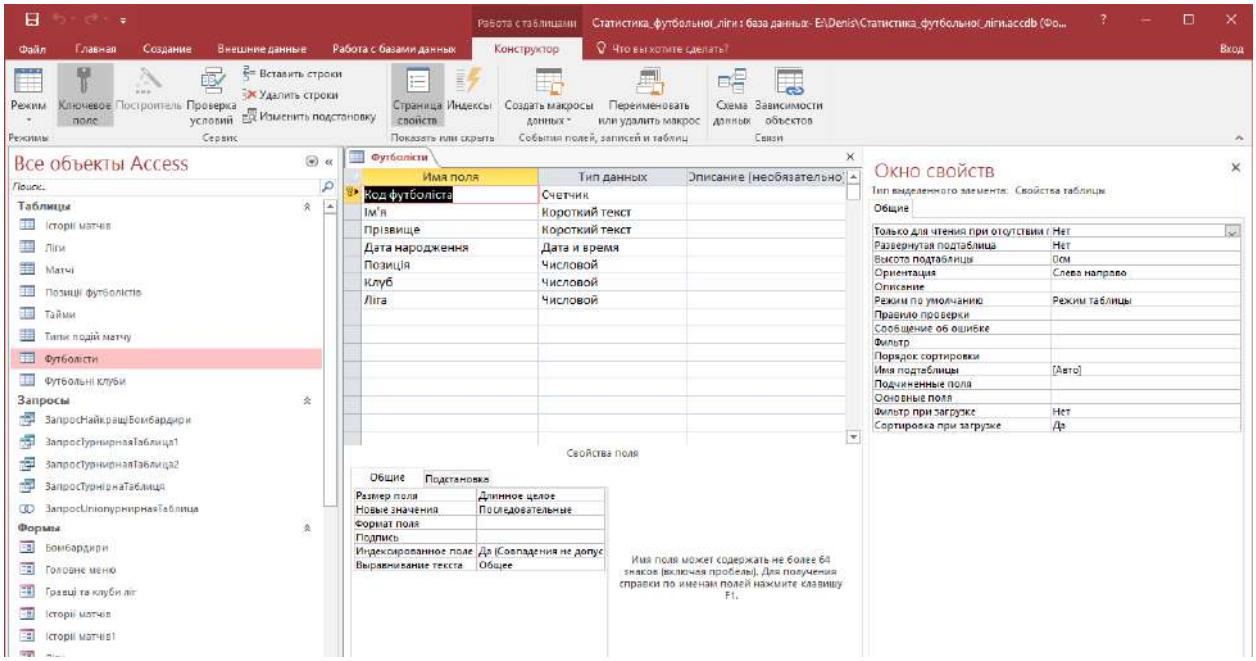



Рис. 3.3. Структура таблиці «Футболісти»

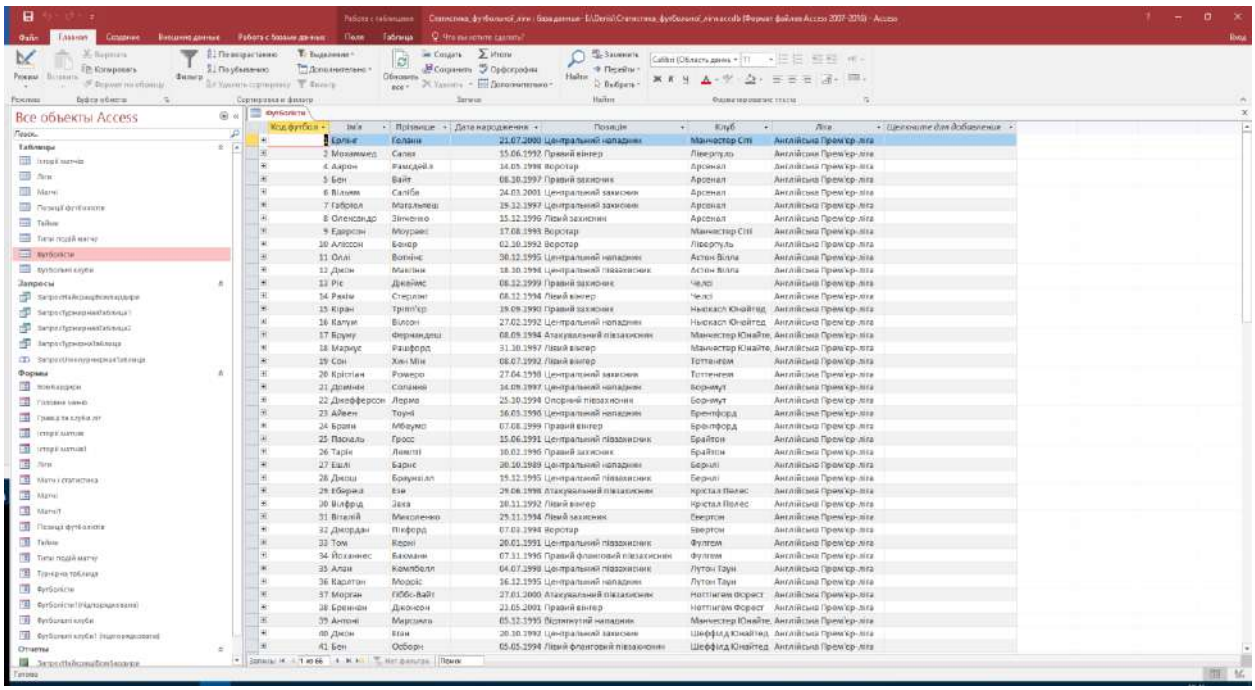



Рис.3.4. Таблиця «Футболісти»

На рис 3.3 та 3.4 зображено структуру та наповнення таблиці «Футболісти». У конструкторі таблиці ми бачимо, що внутрішнім ключем виступає поле «Код футболіста», яке відповідно має тип «Лічильник». Нижче розташовані інші поля, що характеризують футболіста у контексті даної курсової роботи, такі як: «Ім’я» та «Прізвище» (мають текстовий тип даних), «Дата народження»( тип поля «Дата і час»), «Позиція», «Клуб»(два поля, що є зовнішніми ключами).

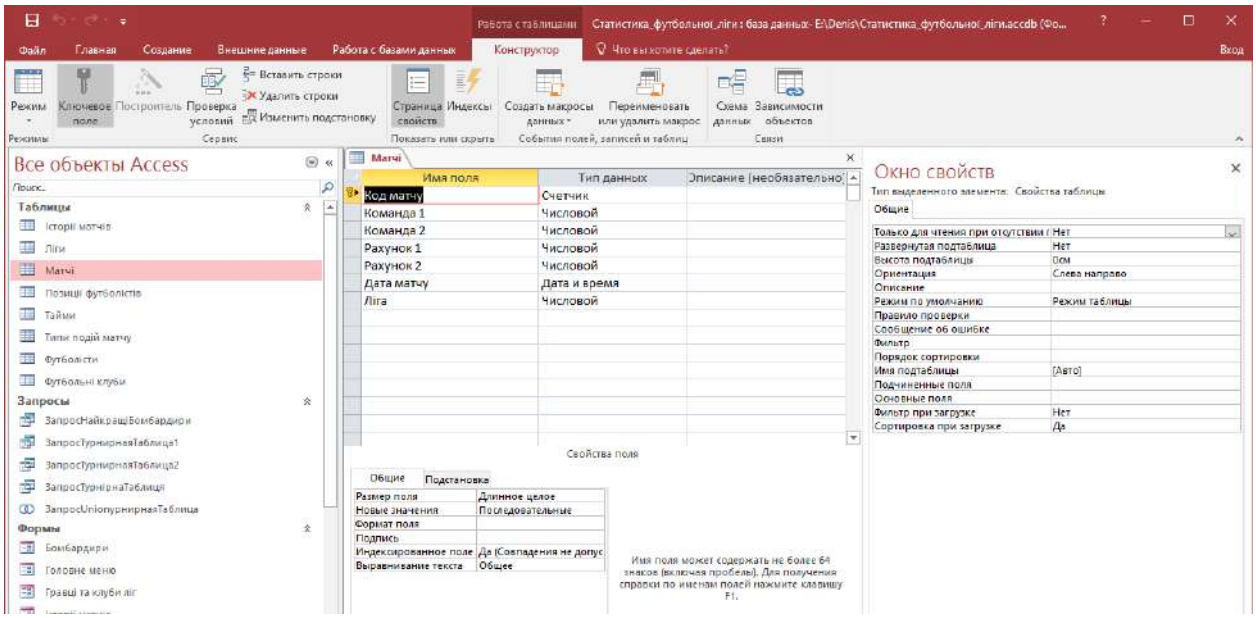



Рис. 3.5. Структура таблиці «Матчі»

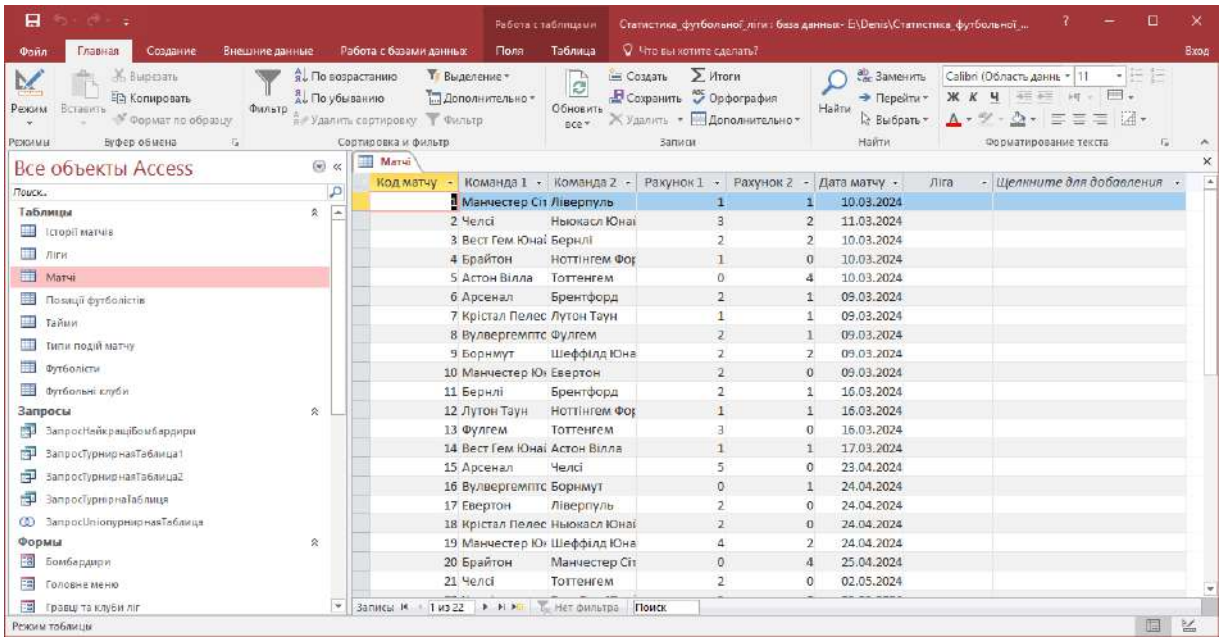



Рис. 3.6. Таблиця «Матчі»

На рис 3.5 та 3.6 ми бачимо структуру та наповнення таблиці «Матчі». Внутрішнім ключовим полем тут є «Код матчу». Поля: «Команда 1», «Команда 2», «Ліга» є зовнішніми ключами і дані у них беруться з інших таблиц. Також ми бачимо поля «Рахунок 1» та «Рахунок 2»(числовий тип даних), і «Дата матчу» (тип даних дата і час).

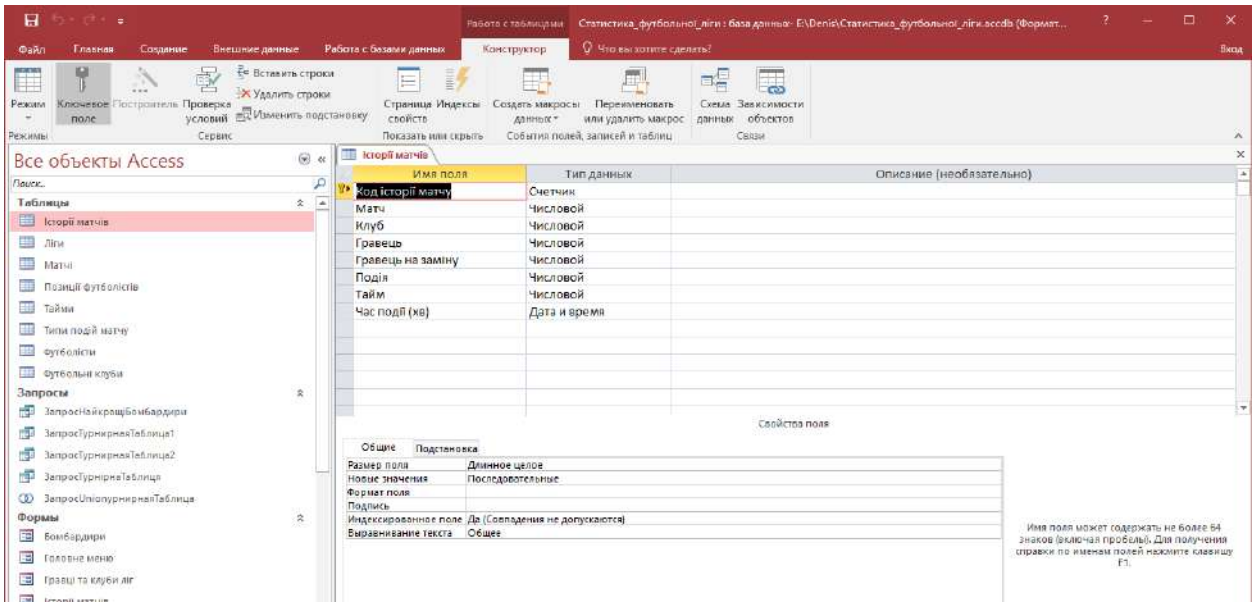



Рис. 3.7. Структура таблиці «Історії матчів»

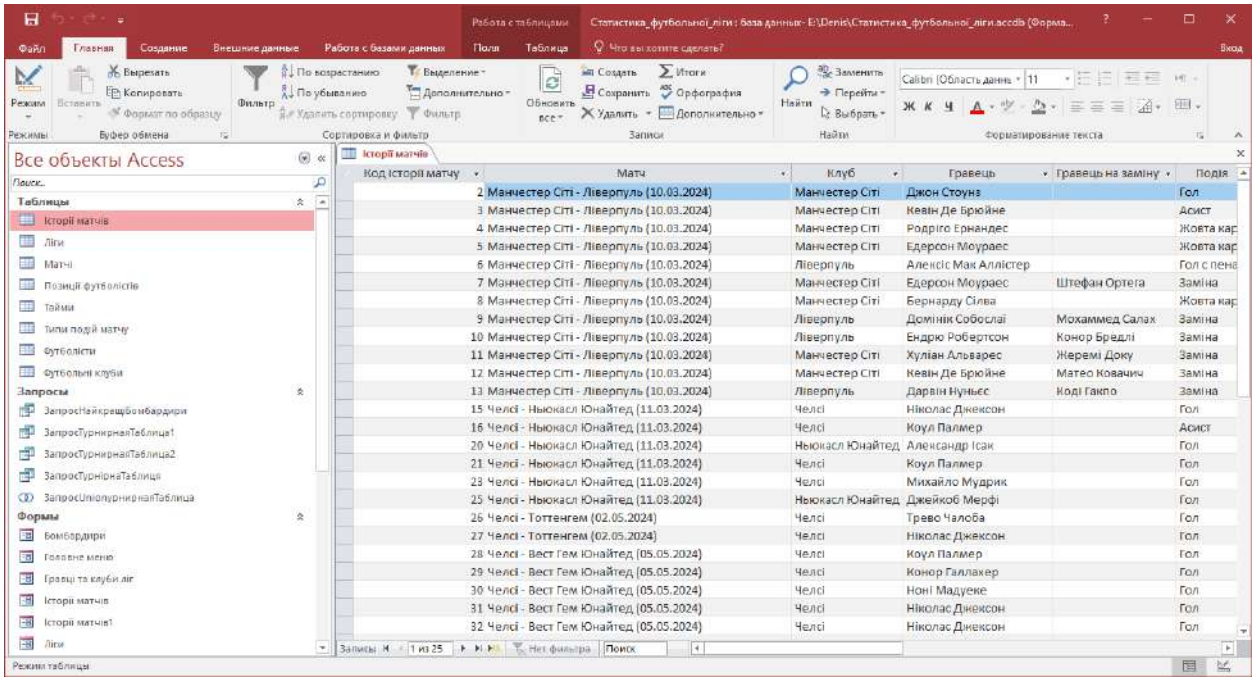



Рис 3.8. Таблиця «Історії матчів»

На рис 3.7 та 3.8 ми бачимо структуру та наповнення таблиці «Історії матчів». Внутрішнім ключовим полем тут є «Код історії матчу». Поля: «Матч», «Клуб», «Гравець», «Гравець на заміну», «Подія» і «Тайм» є зовнішніми ключами в яких дані беруться з інших таблиць. Також ми бачимо поле «Час події(хв)» (тип даних дата і час).

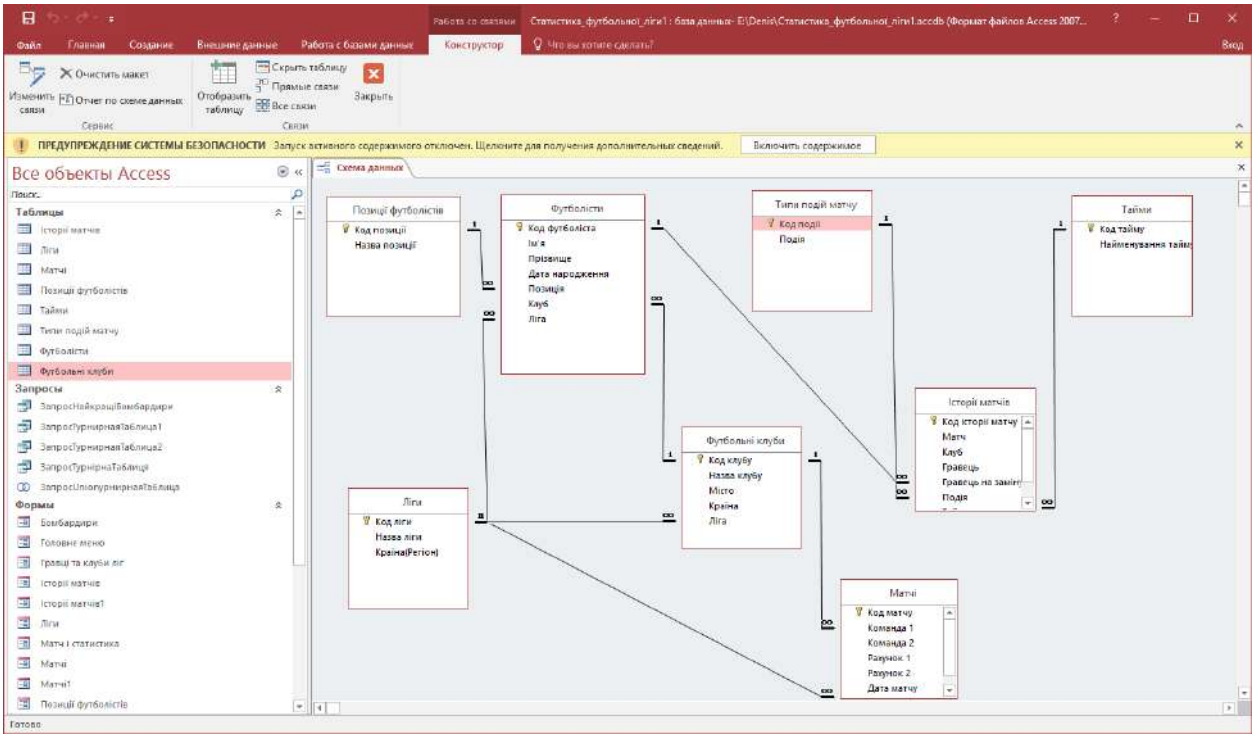



Рис. 3.9 «Схема даних»

На рис. 3.9 ми бачимо схему даних на якій зображено усі зв’язки таблиць між собою. Деякі таблиці є довідниками та мають лише по одному зв’язку, інші мають більше зв’зків. бо мають більше даних для оперування з інших таблиць.

## 3.3. Розробка бази даних SQL Server

Під час розробки проєкту було використано систему керування базами даних MS SQL Server. Нижче представлено структури усіх таблиць бази даних та запитів для їх створення на мові C#, що були створені під час розробки програмного забезпечення.

| | Имя | Тип данных | Допустимы значения NULL | По умолчанию |
|---|---|---|---|---|
| 🔑 | league_id | int | ☐ | |
| | league_name | nvarchar(100) | ☐ | |
| | external_id | int | ☑ | |
| | country_id | int | ☑ | |

Рис. 3.10. Структура таблиці «Ліги»

Як видно з рис. 3.10, на якому представлено опис властивостей таблиці «Ліги», її функція – зберігання основної інформації про футбольні ліги. Дані про них зберігаються у полях цієї структури, а саме такі дані як: унікальний ідентифікатор, назва ліги, зовнішній ідентифікатор, та зовнішній ключ країни. Для створення структури таблиці в середовищі SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[leagues] (
    [league_id]   INT            IDENTITY (1, 1) NOT NULL,
    [league_name] NVARCHAR (100) NOT NULL,
    [external_id] INT            NULL,
    [country_id]  INT            NULL,
    PRIMARY KEY CLUSTERED ([league_id] ASC)
);
```

| | Имя | Тип данных | Допустимы значения NULL | По умолчанию |
|---|---|---|---|---|
| 🔑 | club_id | int | ☐ | |
| | club_name | nvarchar(100) | ☐ | |
| | league_id | int | ☐ | |
| | country_id | int | ☐ | |
| | external_id | nchar(10) | ☑ | |

Рис. 3.11. Структура таблиці «Клуби»

Таблиця клуби (див. рис. 3.11), призначена для зберігання даних про футбольні клуби. Це завдання вона виконує за допомогою полів, що містять інформацію про : унікальний ідентифікатор клубу, назву клубу, зовнішній ідентифікатор, та країну походження. Для створення структури таблиці в середовищі SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[football_clubs] (
    [club_id]     INT            IDENTITY (1, 1) NOT NULL,
    [club_name]   NVARCHAR (100) NOT NULL,
```

```
    [league_id]   INT            NOT NULL,
    [country_id]  INT            NOT NULL,
    [external_id] NCHAR (10)     NULL,
    PRIMARY KEY CLUSTERED ([club_id] ASC)
);
```

| | Имя | Тип данных | Допустимы значения NULL | По умолчанию |
|---|---|---|---|---|
| 🔑 | country_id | int | ☐ | |
| | name | nvarchar(100) | ☑ | |
| | code | nvarchar(20) | ☑ | |
| | | | ☐ | |

Рис. 3.12. Структура таблиці «Країни»

Як видно з рис. 3.12, на ньому зображено структуру таблиці «Країни». Вона характеризується полями, що містять у собі такі дані як: унікальний ідентифікатор країни, назву країни, та дволітерний код країни за міжнародним стандартом «ISO 3166-1 alpha-2». Для створення структури таблиці в середовищі SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[countries] (
    [country_id] INT            IDENTITY (1, 1) NOT NULL,
    [name]       NVARCHAR (100) NULL,
    [code]       NVARCHAR (20)  NULL,
    PRIMARY KEY CLUSTERED ([country_id] ASC)
);
```

| | Имя | Тип данных | Допустимы значения NULL | По умолчанию |
|---|---|---|---|---|
| 🔑 | season_id | int | ☐ | |
| | season_year | int | ☐ | |
| | date_start | datetime | ☐ | |
| | date_end | datetime | ☐ | |
| | external_uid | varchar(50) | ☑ | |
| | league_id | int | ☐ | |

Рис. 3.13. Структура таблиці «Сезони»

Як видно з рис. 3.13, сезони хаорактеризуються дату початкую та кінця, а також роком початку. Для створення структури таблиці в середовищі SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[seasons] (
    [season_id]    INT          IDENTITY (1, 1) NOT NULL,
```

```
    [season_year]  INT          NOT NULL,
    [date_start]   DATETIME     NOT NULL,
    [date_end]     DATETIME     NOT NULL,
    [external_uid] VARCHAR (50) NULL,
    [league_id]    INT          NOT NULL,
    PRIMARY KEY CLUSTERED ([season_id] ASC)
);
```

| | Имя | Тип данных | Допустимы значения NULL | По умолчанию |
|---|---|---|---|---|
| 🔑 | match_id | int | ☐ | |
| | home_team_id | int | ☐ | |
| | away_team_id | int | ☐ | |
| | home_result | int | ☑ | |
| | away_result | int | ☑ | |
| | match_date | datetime | ☐ | |
| | home_ht_result | int | ☑ | |
| | away_ht_result | int | ☑ | |
| | home_ft_result | int | ☑ | |
| | away_ft_result | int | ☑ | |
| | season_id | int | ☐ | |
| | league_id | int | ☐ | |
| | status | int | ☑ | |
| | external_id | int | ☑ | |
| | home_formation | nvarchar(50) | ☑ | |
| | away_formation | nvarchar(50) | ☑ | |
| | home_coach_id | int | ☑ | |
| | away_coach_id | int | ☑ | |

Рис. 3.14. Структура таблиці «Матчі»

На рис. 3.14 зображено структуру таблиці «Матчі». Футбольний матч складається з достатньо великого масиву інформації, саме тому завданням цієї таблиці є зберігання усіх основних даних системи, оскільки матчі є однією з основних сутностей. Головне чим характеризуються матчі це: унікальний ідентифікатор; команди що грають між собою та їх ідентифікатори; дата матчу; забиті голи обома командами у таймах та фінальний результат; ідентифікатор сезону, коли відбувся матч; ідентифікатор ліги матчу; склад кожної з команд; тренери. Для створення структури таблиці в середовищі SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[matches] (
    [match_id]       INT          IDENTITY (1, 1) NOT NULL,
    [home_team_id]   INT          NOT NULL,
    [away_team_id]   INT          NOT NULL,
    [home_result]    INT          NULL,
    [away_result]    INT          NULL,
    [match_date]     DATETIME     NOT NULL,
    [home_ht_result] INT          NULL,
    [away_ht_result] INT          NULL,
```

```
    [home_ft_result] INT           NULL,
    [away_ft_result] INT           NULL,
    [season_id]      INT           NOT NULL,
    [league_id]      INT           NOT NULL,
    [status]         INT           NULL,
    [external_id]    INT           NULL,
    [home_formation] NVARCHAR (50) NULL,
    [away_formation] NVARCHAR (50) NULL,
    [home_coach_id]  INT           NULL,
    [away_coach_id]  INT           NULL,
    PRIMARY KEY CLUSTERED ([match_id] ASC)
);
```

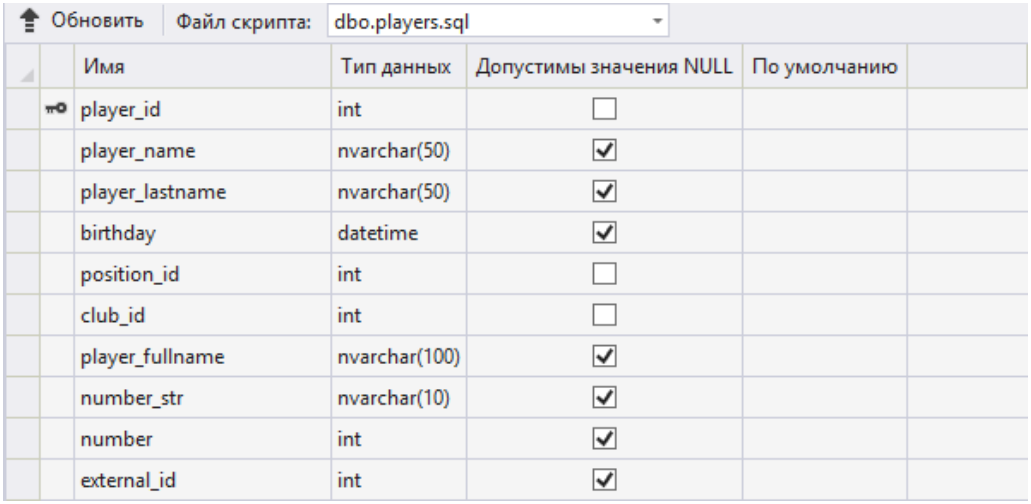



Рис. 3.15. Структура таблиці «Футболісти»

На рис. 3.15 зображено структуру таблиці «Футболісти». Як видно кожен гравець має багато властивостей, які його характеризують. В першу чергу це поле, що містить унікальний ідентифікатор гравця. Наступними показниками є: ім’я гравця, прізвище та дата його народження. Також кожен футболіст має, безпосередньо, свою позицію на полі, клуб, ігровий номер. Другорядним, але не менш важливим, елементом цієї таблиці виступає поле «external_id», що містить зовнішній ідентифікатор гравця. Для створення структури таблиці в середовищі SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[players] (
    [player_id]       INT           NOT NULL,
    [player_name]     NVARCHAR (50) NULL,
    [player_lastname] NVARCHAR (50) NULL,
    [birthday]        DATETIME      NULL,
    [position_id]     INT           NOT NULL,
    [club_id]         INT           NOT NULL,
    [player_fullname] NVARCHAR (100) NULL,
    [number_str]      NVARCHAR (10) NULL,
    [number]          INT           NULL,
    [external_id]     INT           NULL,
```

```
    PRIMARY KEY CLUSTERED ([player_id] ASC)
);
```

Обновить | Файл скрипта: dbo.matches_history.sql

| Имя | Тип данных | Допустимы значения NULL | По умолчанию |
|---|---|---|---|
| history_id | int | ☐ | |
| match_id | int | ☐ | |
| club_id | int | ☐ | |
| player_id | int | ☐ | |
| assist_player_id | int | ☑ | |
| event_id | int | ☐ | |
| period_id | int | ☐ | |
| elapsed | int | ☐ | |
| substitution_number | int | ☑ | |
| extarnal_id | int | ☑ | |

Рис 3.16. Структура таблиці «Історія матчів»

З рис. 3.16 видно структуру таблиці «Історія матчів», що має багато полів, які у більшості зв'язані з іншими структурами. Так, поля: «match_id», «club_id», «player_id», «assist_player_id», «event_id», «period_id» -- зберігають ідентифікатори відповідних даних, якими вже можна маніпулювати запитами з цієї таблиці. Звісно ця таблиця не обходиться без поля, що містить її первинний ідентифікатор. Для створення структури таблиці в середовищі SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[matches_history] (
    [history_id]          INT IDENTITY (1, 1) NOT NULL,
    [match_id]            INT NOT NULL,
    [club_id]             INT NOT NULL,
    [player_id]           INT NOT NULL,
    [assist_player_id]    INT NULL,
    [event_id]            INT NOT NULL,
    [period_id]           INT NOT NULL,
    [elapsed]             INT NOT NULL,
    [substitution_number] INT NULL,
    [external_id]         INT NULL,
    PRIMARY KEY CLUSTERED ([history_id] ASC)
);
```

| Имя | Тип данных | Допустимы значения NULL | По умолчанию |
|---|---|---|---|
| coach_d | int | ☐ | |
| external_id | int | ☑ | |
| coach_name | nvarchar(50) | ☑ | |

Рис. 3.17. Структура таблиці «Тренери»

Як видно з рис 3.17, таблиця «Тренери» має мало полів, адже її основна функція це зберігати дані про тренерів. Цю задачу виконують поля: «coach_id» та «coach_name». Для створення в SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[coaches] (
    [coach_d]     INT           IDENTITY (1, 1) NOT NULL,
    [external_id] INT           NULL,
    [coach_name]  NVARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([coach_d] ASC)
);
```

| Имя | Тип данных | Допустимы значения NULL | По умолчанию |
|---|---|---|---|
| match_player_id | int | ☐ | |
| match_id | int | ☐ | |
| club_id | int | ☐ | |
| player_id | int | ☐ | |
| is_start | bit | ☐ | ((0)) |
| grid | nvarchar(20) | ☑ | |
| position_id | int | ☐ | |
| capitan | bit | ☐ | ((0)) |
| substitute | bit | ☐ | ((0)) |
| offsides | int | ☑ | |
| shots_total | int | ☑ | |
| goals_total | int | ☑ | |
| goals_conceded | int | ☑ | |
| goals_assists | int | ☑ | |
| goals_saves | int | ☑ | |
| passes_total | int | ☑ | |
| passes_key | int | ☑ | |
| passes_accuracy | int | ☑ | |
| tackles_total | int | ☑ | |
| tackles_blocks | int | ☑ | |
| tackles_interceptions | int | ☑ | |
| duels_total | int | ☑ | |
| duels_won | int | ☑ | |
| dribbles_attempts | int | ☑ | |
| dribbles_success | int | ☑ | |
| dribbles_past | int | ☑ | |
| fouls_drawn | int | ☑ | |
| fouls_committed | int | ☑ | |
| cards_yellow | int | ☑ | |
| cards_red | int | ☑ | |
| penalty_won | int | ☑ | |
| penalty_commited | int | ☑ | |
| penalty_scored | int | ☑ | |
| penalty_saved | int | ☑ | |
| rating | real | ☐ | ((0)) |

Рис. 3.18. Структура таблиці «Статистика гравців матчу»

Як видно з рис. 3.18, однією із головних таблиць є таблиця, яка містить дані про гравців матчу та їх статистичні ігрові показники, а саме: ідентифікатор гравця, клубу, матчу та позиції, містить дані про кількість офсайдів гравця, ударів по воротах, ударів у площину воріт, гольових передач, сейви (якщо це воротар), загальну кількість пасів, кількість ключових передач та їх точність, кількість підкатів, блоків та перехоплень м'яча, кіл-ть дуелей та кіл-ть виграних з них, кіл-ть спроб дриблінгу та кіл-ть успішних з них, зафіксовані та зароблені фоли гравця, червоні та жовті картки, забиті, відбиті чи зароблені пенальті, рейтинг гравця за матч. Такий великий масив даних ми маємо у цій таблиці.

Для створення структури таблиці в середовищі SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[matches_players] (
    [match_player_id]       INT           IDENTITY (1, 1) NOT NULL,
    [match_id]              INT           NOT NULL,
    [club_id]               INT           NOT NULL,
    [player_id]             INT           NOT NULL,
    [is_start]              BIT           DEFAULT ((0)) NOT NULL,
    [grid]                  NVARCHAR (20) NULL,
    [position_id]           INT           NOT NULL,
    [capitan]               BIT           DEFAULT ((0)) NOT NULL,
    [substitute]            BIT           DEFAULT ((0)) NOT NULL,
    [offsides]              INT           NULL,
    [shots_total]           INT           NULL,
    [goals_total]           INT           NULL,
    [goals_conceded]        INT           NULL,
    [goals_assists]         INT           NULL,
    [goals_saves]           INT           NULL,
    [passes_total]          INT           NULL,
    [passes_key]            INT           NULL,
    [passes_accuracy]       INT           NULL,
    [tackles_total]         INT           NULL,
    [tackles_blocks]        INT           NULL,
    [tackles_interceptions] INT           NULL,
    [duels_total]           INT           NULL,
    [duels_won]             INT           NULL,
    [dribbles_attempts]     INT           NULL,
    [dribbles_success]      INT           NULL,
    [dribbles_past]         INT           NULL,
    [fouls_drawn]           INT           NULL,
    [fouls_committed]       INT           NULL,
    [cards_yellow]          INT           NULL,
    [cards_red]             INT           NULL,
    [penalty_won]           INT           NULL,
    [penalty_commited]      INT           NULL,
    [penalty_scored]        INT           NULL,
    [penalty_saved]         INT           NULL,
    [rating]                REAL          DEFAULT ((0)) NOT NULL,
    PRIMARY KEY CLUSTERED ([match_player_id] ASC)
);
```

| | Имя | Тип данных | Допустимы значения NULL | По умолчанию | |
|---|---|---|---|---|---|
| 🔑 | match_statistic_id | int | ☐ | | |
| | match_id | int | ☐ | | |
| | club_id | int | ☐ | | |
| | statistic_param_id | int | ☐ | | |
| | value | int | ☐ | | |

Рис. 3.19. Структура таблиці «Статистичні показники матчу»

Як видно з рис. 3.19, таблиця, що зображена, містить дані про статистичні показники матчу. Для цього у ній існує унікальний ідентифікатор статистики матчу, ідентифікатор самого матчу, клубу, та ідентифікатор конкретного статистичного параметру.

Для створення структури таблиці в середовищі SQL Server необхідно виконати такий запит:

```
CREATE TABLE [dbo].[match_statistics] (
    [match_statistic_id] INT IDENTITY (1, 1) NOT NULL,
    [match_id]           INT NOT NULL,
    [club_id]            INT NOT NULL,
    [statistic_param_id] INT NOT NULL,
    [value]              INT NOT NULL,
    PRIMARY KEY CLUSTERED ([match_statistic_id] ASC)
);
```

## Висновки до розділу 3

В ході виконання роботи було спроектовано базу даних та структуру її таблиць. Базу даних було реалізовано в середовищі сучасних систем управління базами даних СУБД SQL Server та Access. Спроектовані таблиці містять усю необхідну інформацію про футбольні матчі, його події та гравців і їх статистичні показники. Що дозволяє ефективно аналізувати перебіг матчів за допомогою sql-запитів.

# РОЗДІЛ 4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 4.1. Вибір технологій розробки

Windows Forms (WinForms) — це одна з найбільш популярних технологій для створення додатків під операційну систему Windows. Вона входить до складу платформи .NET і забезпечує зручний інтерфейс для побудови графічних користувацьких інтерфейсів. Можна виділити такі основні переваги Windows Forms [7]:

**1. Зручна та проста розробка**

Використовуючи Windows Forms можна створювати додатки завдяки візуальному редактору форм та елементів керування всередині. З нього можна перетягнути більшість елементів на форму та налаштовувати їх властивості. Такий функціонал є дуже зручним та інтуїтивно зрозумілим під час безпосередньої розробки програмного забезпечення.

**2. Інтеграція з Visual Studio**

Повна підтримка в середовищі розробки Visual Studio дає можливість зручно писати код, відлагоджувати додаток, використовувати дизайнер форм, а також інтегрувати бази даних і сторонні бібліотеки.

**3. Стабільність і підтримка**

Технологія перевірена часом і використовується у великій кількості корпоративних додатків. Microsoft продовжує підтримувати її в .NET Core та .NET 5/6/7.

C# — це сучасна об’єктно-орієнтована мова програмування, розроблена компанією Microsoft спеціально для роботи з платформою .NET. Вона поєднує в собі потужність C++ і простоту Java, і має низку переваг:

1. **Зручний синтаксис.** Синтаксис C# логічний, зрозумілий та інтуїтивний. Це знижує кількість помилок і спрощує навчання для новачків.
2. **Об'єктно-орієнтоване програмування (ООП).** Підтримка ООП дозволяє створювати масштабовані, модульні й повторно використовувані

програми, що особливо важливо для великих проектів.

3. **Безпека типів і управління пам’яттю.** C# — строго типізована мова, яка автоматично керує пам’яттю за допомогою механізму "збирання сміття" (Garbage Collector), що значно знижує ймовірність помилок, пов’язаних із пам’яттю.

4. **Інтеграція з .NET Framework / .NET Core / .NET 5+.** C# тісно інтегрована з усіма можливостями платформи .NET, включаючи роботу з базами даних, мережею, XML, JSON, криптографією, асинхронним програмуванням тощо.

5. **Велика спільнота і документація.** Завдяки великій спільноті розробників, C# має широку документацію, безліч прикладів і бібліотек з відкритим кодом.

6. **Мультиплатформеність.** Завдяки .NET Core / .NET 5+ C# підтримує створення додатків для Windows, Linux, macOS, а також мобільних платформ через MAUI або Xamarin.

Поєднання Windows Forms і мови C# є ефективним вибором для створення настільних програм під Windows. Windows Forms забезпечує простий спосіб побудови інтерфейсу, а C# дозволяє реалізувати логіку програми на надійній і сучасній мові [1, 5, 12, 15, 17, 21, 22]. Разом ці технології дозволяють швидко розробити стабільний і зручний у користуванні програмний продукт.

## 4.2. Розробка програмного забезпечення з використанням Windows Forms

При розробці програмного забезпечення було використано технологію Windows Forms. Нижче описано структуру основних форм, реалізовану навігацію між модулями, механізми обробки подій, та валідації введених даних:

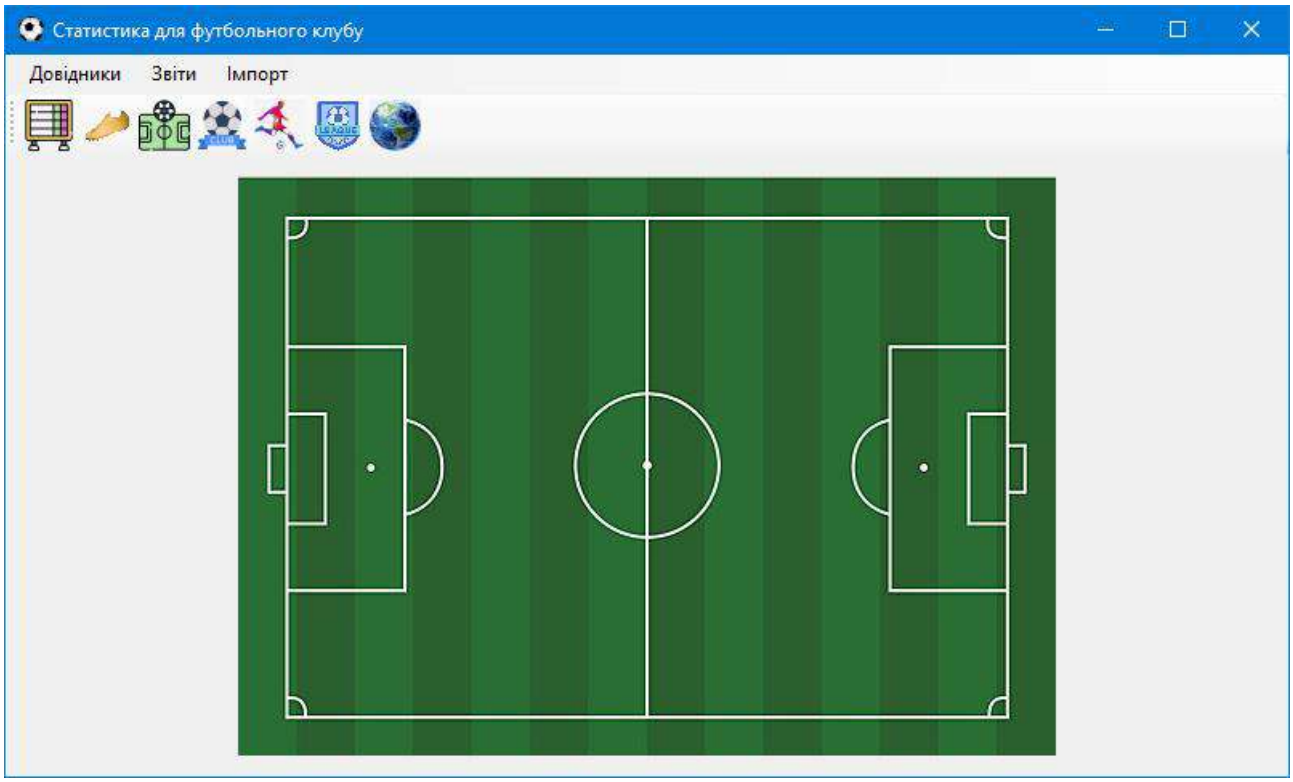



Рис. 4.1. Головна форма

На рис 4.1 зображено головну форму програми, звідки відбувається майже уся навігація. Тут розташовані 3 пункти меню: «Довідники», де зберігаються усі таблиці з даними, «Звіти», де зберігаються таблиці зі сформованими запитами для виведення статистики, «Імпорт» завдяки якому можна завантажити дані із зовнішнього джерела та наповнити таблиці даними. Нижче продубльовано основні пункти меню у вигляді іконок для кращої навігації та візуалізації.

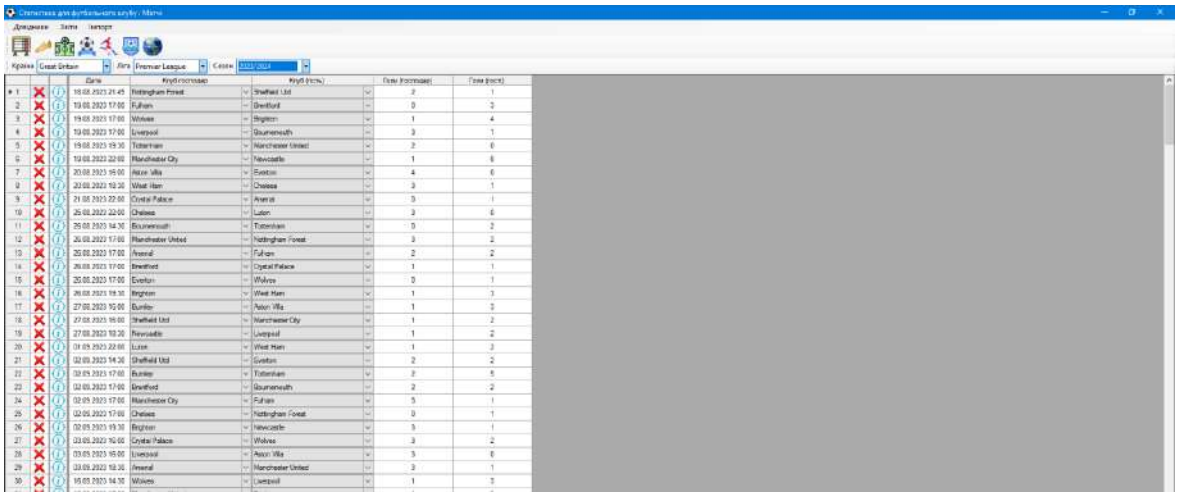


Рис. 4.2. Форма «Матчі»

На рис. 4.2 зображено форму «Матчі». Щоб побачити список матчів перш за все треба заповнити 3 поля з випадаючим списком: у першому полі треба вибрати країну, в наступному – лігу, а в останньому – сезон, матчі якого користувач хоче отримати. Після цих дій можна буде побачити усі матчі згідно із фільтром. Кожен матч на цій сторінці містить: дату, коли відбувся, команду-господаря, команду-гість, кінцевий результат матчу(голи однієї та другої команди). У лівій частині розташовані 2 кнопки: перша (червоний хрестик) дає нам змогу видаляти матч, а при натисненні на другу відбувається перехід до повної, детальної статистики матчу.

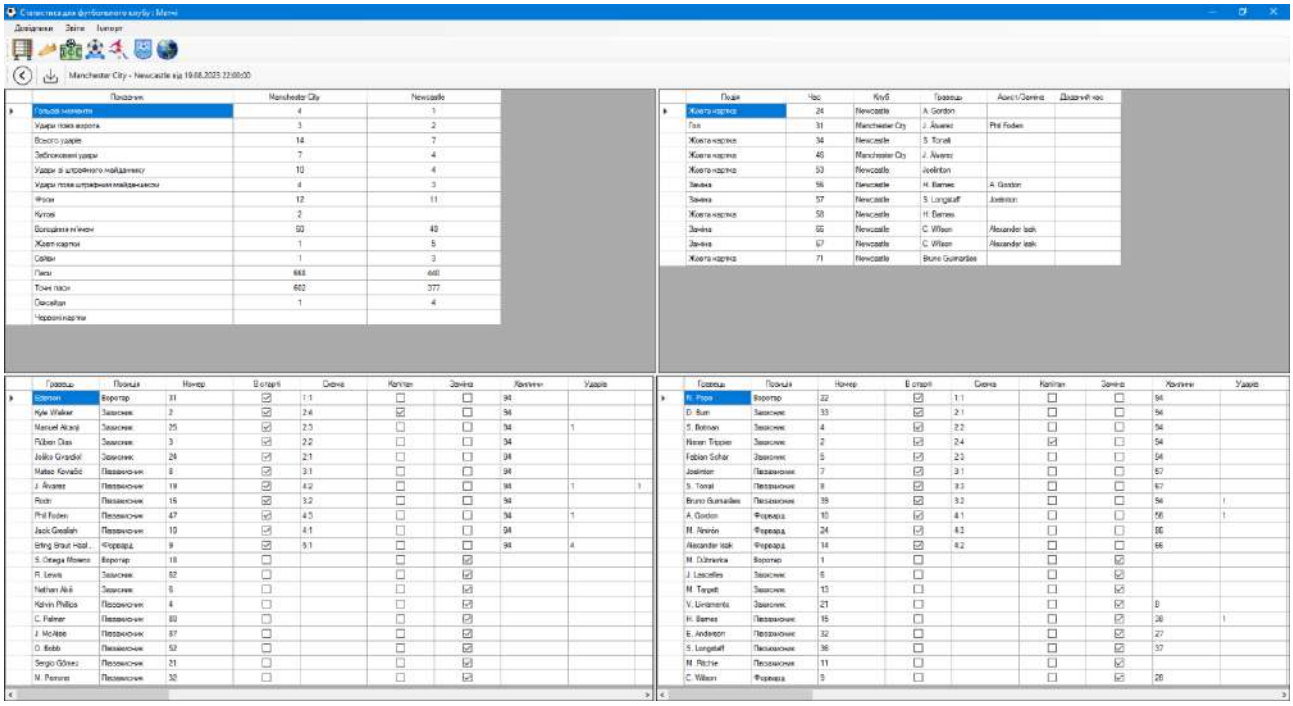


Рис. 4.3. Форма «Статистика матчу»

Як видно з рис. 4.3, статистика матчу має великий масив даних, саме тому вона поділена на 4 частини. Верхня ліва частина описує статистику обидвох команд за матч, крайня ліва колонка містить назви показників, центральна і права колонка – числові значення статистики першої і другої команди. до якої входять: гольові моменти, загальна кількість ударів, удари зі штрафного майданчику, володіння м’ячем, фоли, сейви, кіл-ть пасів тощо. У верхній правій частині виведені головні події матчу. У цій таблиці розписано, в який час, з гравцем

якого клубу сталася певна подія (гол, жовта картка,, заміна…). Нижня частина розподілена таким чином, що права частина описує одну команду, ліва — іншу. А конкретно ми маємо інформацію про усіх гравців заявлених на матч, стартовий склад та усю детальну персональну статистику кожного гравця у цьому матчі, від кількості проведених на полі хвилин та відповідних до позиції статистичних показників, до рейтингу гравця за матч.

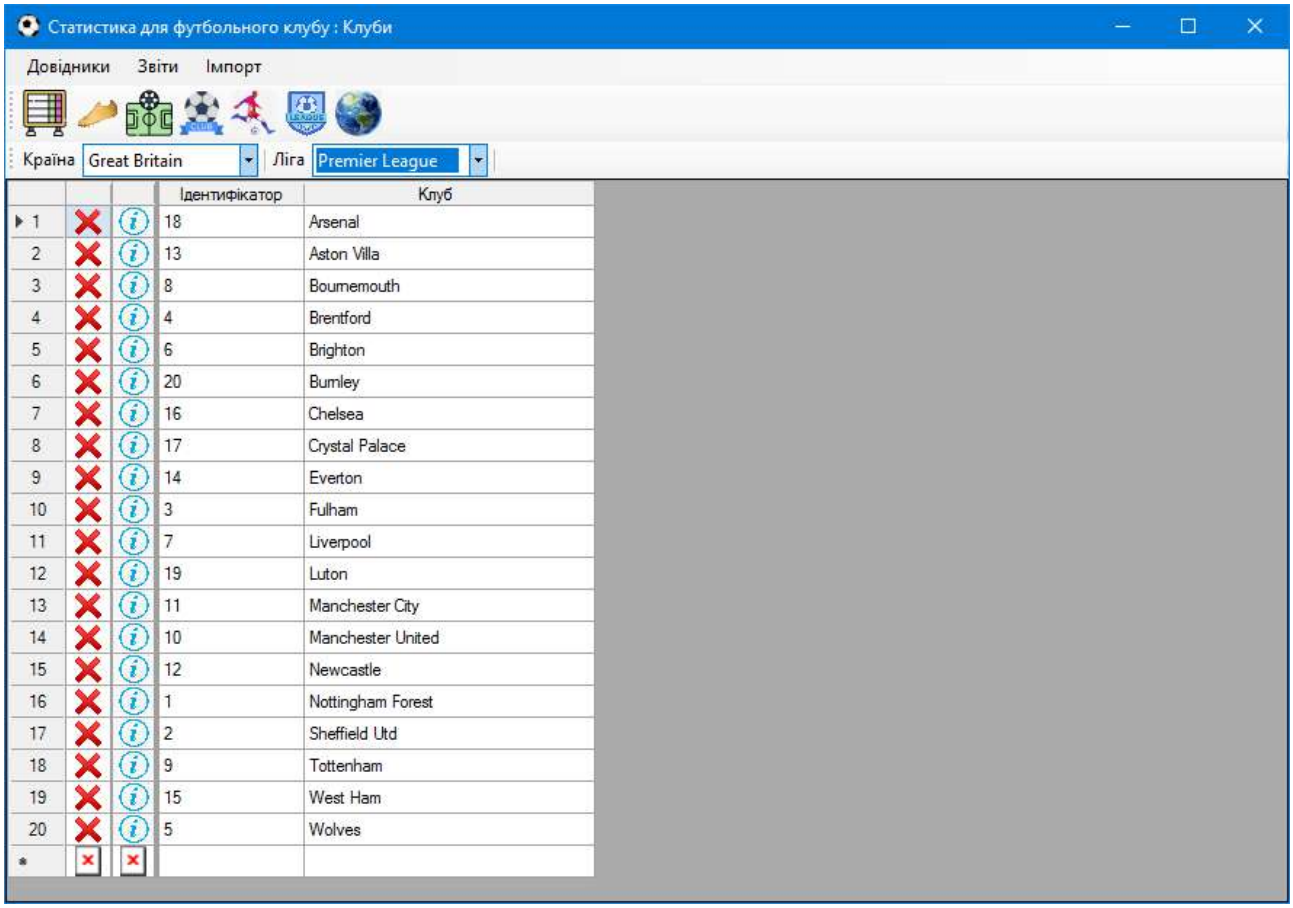



Рис. 4.4. Форма «Клуби»

На рис. 4.4 зображено форму у якій показано список клубів ліги. Вгорі є два поля з випадаючим списком в яких користувач вибирає країну та лігу, клуби якої він хоче вивести на екран. Після цих дій з’явиться таблиця з певною кількістю клубів, що залежить від вибраної ліги. Також є можливість видалення клубу(крайня ліва кнопка). Ця функція існує на випадок, якщо у лізі буде зменшено кількість команд. Поруч із кнопкою видалення елемент керування, при натисненні на який відкривається статистика матчу. В останньому рядку таблиці

можна вводити найменування нового запису, який буде одразу додано до бази даних.

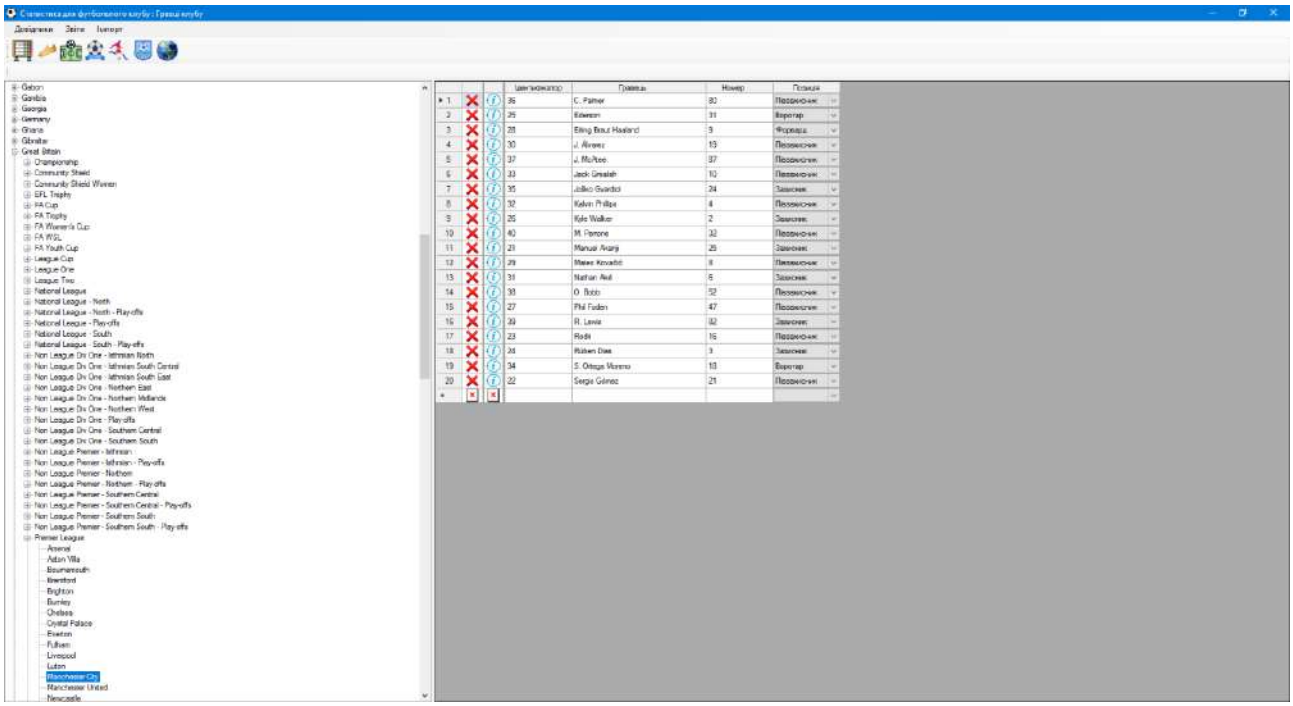


Рис. 4.5. Форма «Футболісти»

Як видно з рис. 4.5, візуальна складова форми побудована таким чином: у лівій частині екрану розташований список ліг, при натиснені на який, нижче з’являється випадаючий список клубів, а після кліку на потрібний – заповнюється таблиця футболістами вибраного клубу.

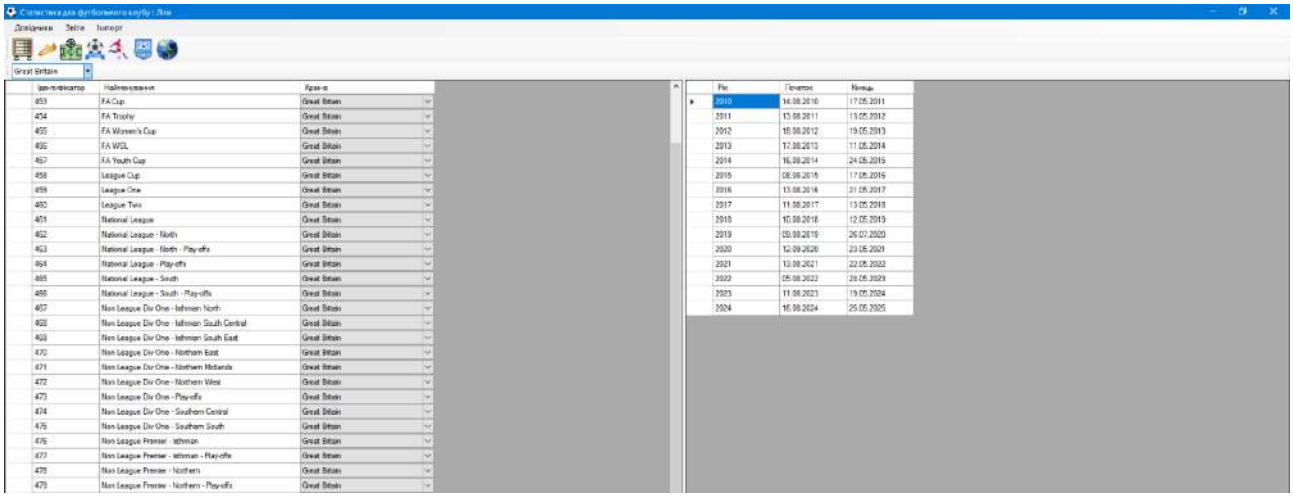
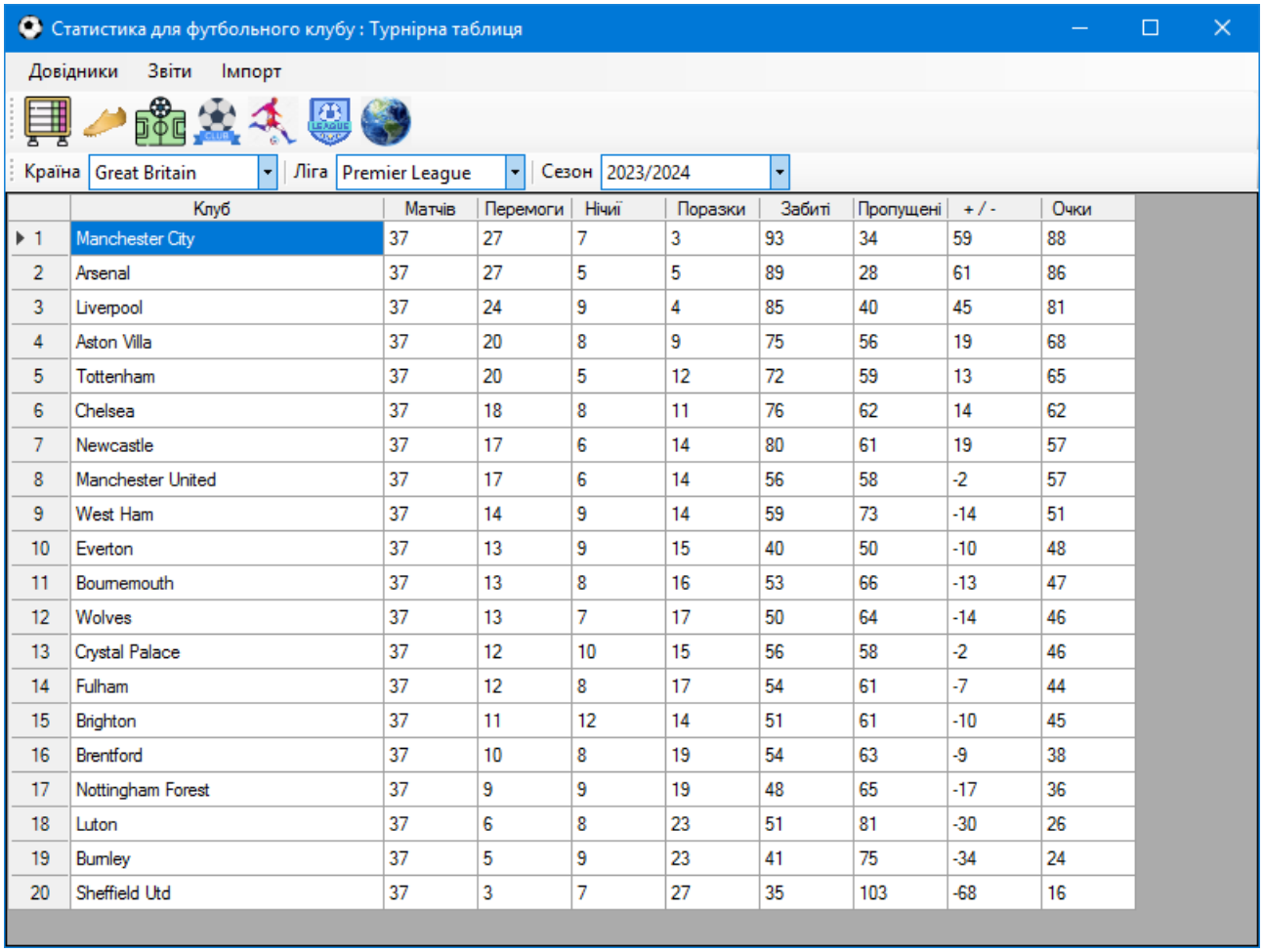


Рис. 4.6. Форма «Ліги»

На рис. 4.6 зображено форму для вибору ліг. У верхньому полі з випадаючим списком користувач має вибрати країну. Після вибору у лівій частині вікна оновиться таблиця з відсортованими лігами за вибраною країною. Вже після цього, натиснувши на потрібну лігу у правій частині вікна з’явиться список сезонів ліги, а користувач буде мати можливість перейти вже до матчів відповідного сезону.

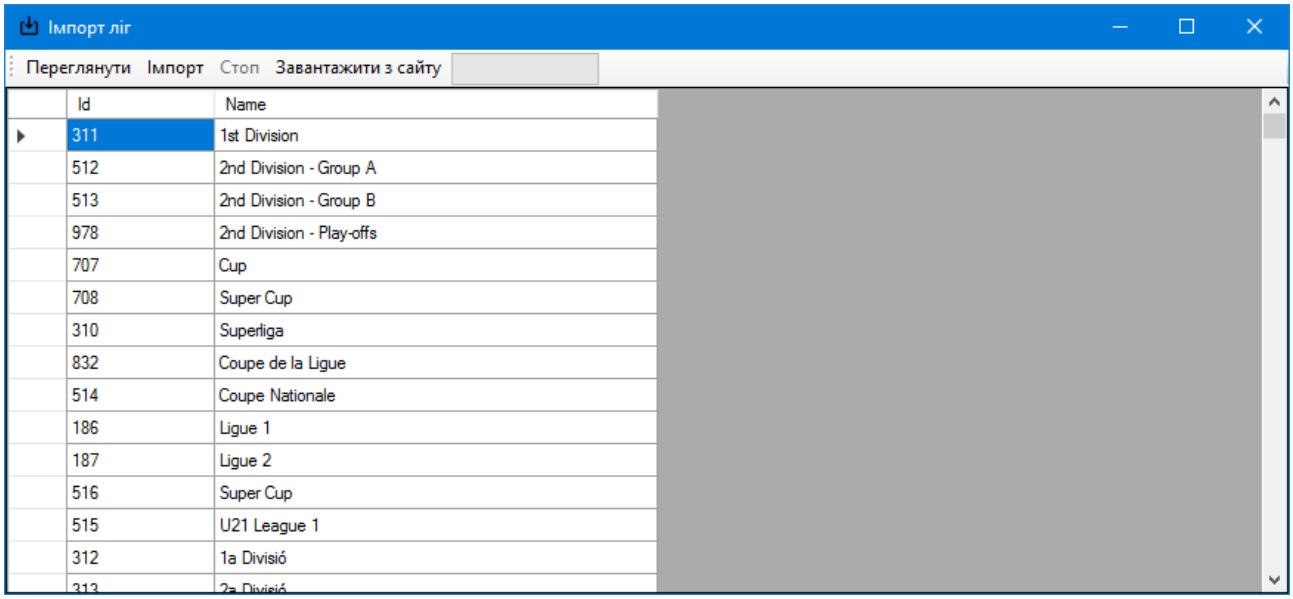



| | Клуб | Матчів | Перемоги | Нічиї | Поразки | Забиті | Пропущені | + / - | Очки |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Manchester City | 37 | 27 | 7 | 3 | 93 | 34 | 59 | 88 |
| 2 | Arsenal | 37 | 27 | 5 | 5 | 89 | 28 | 61 | 86 |
| 3 | Liverpool | 37 | 24 | 9 | 4 | 85 | 40 | 45 | 81 |
| 4 | Aston Villa | 37 | 20 | 8 | 9 | 75 | 56 | 19 | 68 |
| 5 | Tottenham | 37 | 20 | 5 | 12 | 72 | 59 | 13 | 65 |
| 6 | Chelsea | 37 | 18 | 8 | 11 | 76 | 62 | 14 | 62 |
| 7 | Newcastle | 37 | 17 | 6 | 14 | 80 | 61 | 19 | 57 |
| 8 | Manchester United | 37 | 17 | 6 | 14 | 56 | 58 | -2 | 57 |
| 9 | West Ham | 37 | 14 | 9 | 14 | 59 | 73 | -14 | 51 |
| 10 | Everton | 37 | 13 | 9 | 15 | 40 | 50 | -10 | 48 |
| 11 | Bournemouth | 37 | 13 | 8 | 16 | 53 | 66 | -13 | 47 |
| 12 | Wolves | 37 | 13 | 7 | 17 | 50 | 64 | -14 | 46 |
| 13 | Crystal Palace | 37 | 12 | 10 | 15 | 56 | 58 | -2 | 46 |
| 14 | Fulham | 37 | 12 | 8 | 17 | 54 | 61 | -7 | 44 |
| 15 | Brighton | 37 | 11 | 12 | 14 | 51 | 61 | -10 | 45 |
| 16 | Brentford | 37 | 10 | 8 | 19 | 54 | 63 | -9 | 38 |
| 17 | Nottingham Forest | 37 | 9 | 9 | 19 | 48 | 65 | -17 | 36 |
| 18 | Luton | 37 | 6 | 8 | 23 | 51 | 81 | -30 | 26 |
| 19 | Burnley | 37 | 5 | 9 | 23 | 41 | 75 | -34 | 24 |
| 20 | Sheffield Utd | 37 | 3 | 7 | 27 | 35 | 103 | -68 | 16 |

Рис. 4.7. Форма турнірної таблиці

Як видно з рис. 4.7, представлена турнірна таблиця ліги. Для її формування користувач має заповнити три поля з випадаючим списком, що знаходяться у верхній частині, а саме такі поля як: «Країна», «Ліга» та «Сезон». Після цих дій буде виведено відповідну таблицю, що містить ключові показники виступів команд: місце, кіл-ть очок, матчів, перемог, нічиїх, поразок; забиті та пропущені м’ячі та їх різниця.

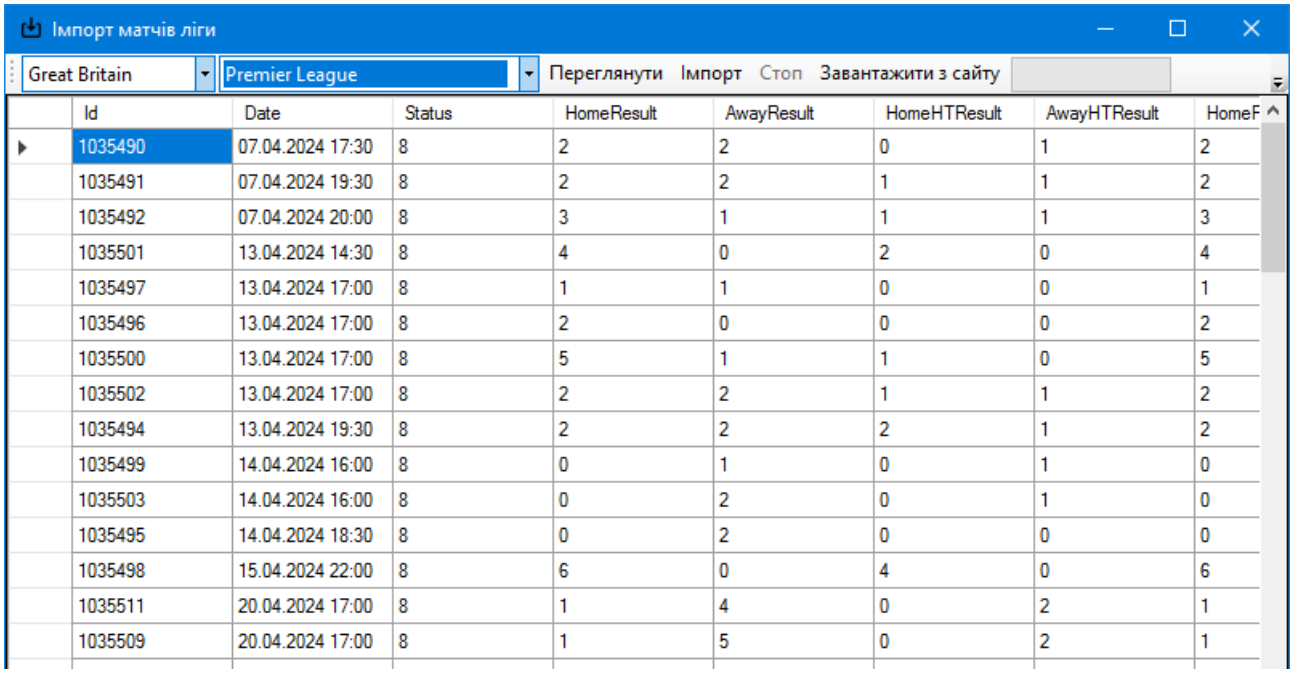
## 4.3. Розробка модулю імпорту даних на Windows Forms




| Id | Name |
|---|---|
| 311 | 1st Division |
| 512 | 2nd Division - Group A |
| 513 | 2nd Division - Group B |
| 978 | 2nd Division - Play-offs |
| 707 | Cup |
| 708 | Super Cup |
| 310 | Superliga |
| 832 | Coupe de la Ligue |
| 514 | Coupe Nationale |
| 186 | Ligue 1 |
| 187 | Ligue 2 |
| 516 | Super Cup |
| 515 | U21 League 1 |
| 312 | 1a Divisió |
| 313 | 2a Divisió |

Рис. 4.8. Форма імпорту ліг

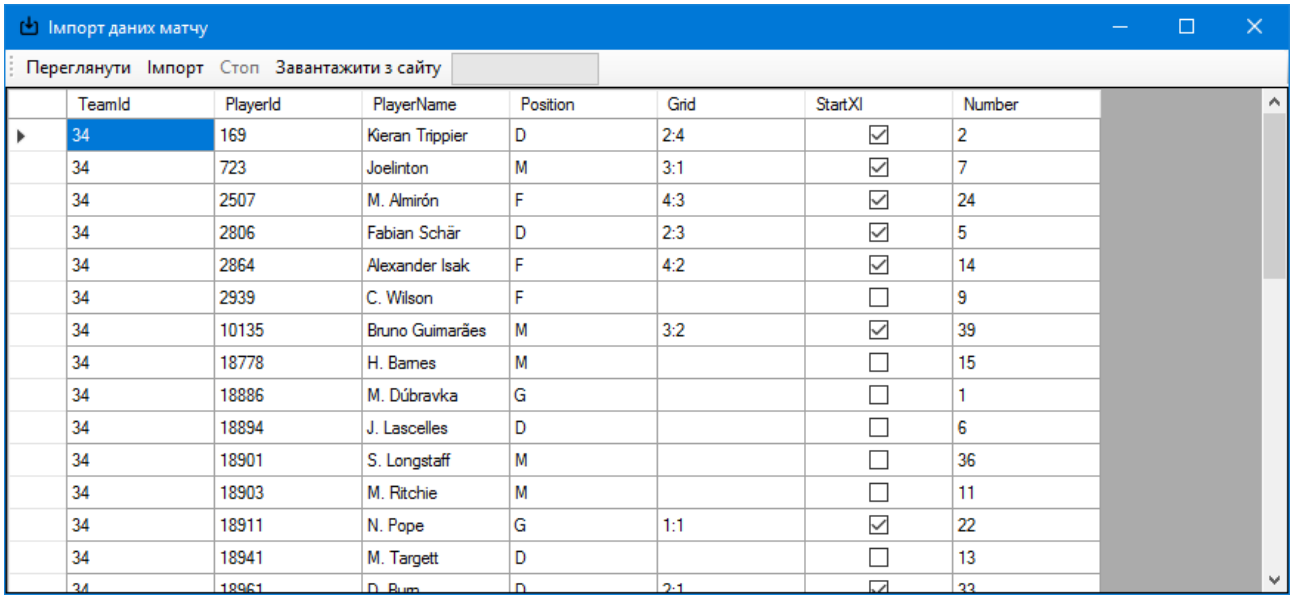
На рис 4.8 ми бачимо форму на якій зображено інтерфейс імпорту даних ліг. У верхньому меню є функція перегляду ліг, що імпортуються, для перевірки перед імпортом. Наступним елементом йде сам імпорт, при натиснені на нього у базу даних завантажуються ліги. Цей процес можна відслідкувати у індикаторі виконання у тому ж меню.




| Id | Date | Status | HomeResult | AwayResult | HomeHTResult | AwayHTResult | HomeF |
|---|---|---|---|---|---|---|---|
| 1035490 | 07.04.2024 17:30 | 8 | 2 | 2 | 0 | 1 | 2 |
| 1035491 | 07.04.2024 19:30 | 8 | 2 | 2 | 1 | 1 | 2 |
| 1035492 | 07.04.2024 20:00 | 8 | 3 | 1 | 1 | 1 | 3 |
| 1035501 | 13.04.2024 14:30 | 8 | 4 | 0 | 2 | 0 | 4 |
| 1035497 | 13.04.2024 17:00 | 8 | 1 | 1 | 0 | 0 | 1 |
| 1035496 | 13.04.2024 17:00 | 8 | 2 | 0 | 0 | 0 | 2 |
| 1035500 | 13.04.2024 17:00 | 8 | 5 | 1 | 1 | 0 | 5 |
| 1035502 | 13.04.2024 17:00 | 8 | 2 | 2 | 1 | 1 | 2 |
| 1035494 | 13.04.2024 19:30 | 8 | 2 | 2 | 2 | 1 | 2 |
| 1035499 | 14.04.2024 16:00 | 8 | 0 | 1 | 0 | 1 | 0 |
| 1035503 | 14.04.2024 16:00 | 8 | 0 | 2 | 0 | 1 | 0 |
| 1035495 | 14.04.2024 18:30 | 8 | 0 | 2 | 0 | 0 | 0 |
| 1035498 | 15.04.2024 22:00 | 8 | 6 | 0 | 4 | 0 | 6 |
| 1035511 | 20.04.2024 17:00 | 8 | 1 | 4 | 0 | 2 | 1 |
| 1035509 | 20.04.2024 17:00 | 8 | 1 | 5 | 0 | 2 | 1 |

Рис. 4.9. Форма імпорту статистики матчу

Як видно з рис 4.9, зображено форму імпорту статистики матчів. Її інтерфейс схожий з формою імпорту ліг (див. рис. 3.8), але є ключова відмінність. Для імпортування матчів у верхньому меню є два поля з випадаючим списком, які зроблені для вибору країни та подальшого вибору ліги матчу. Імпорт включає усю його детальну статистику.




| TeamId | PlayerId | PlayerName | Position | Grid | StartXI | Number |
|---|---|---|---|---|---|---|
| 34 | 169 | Kieran Trippier | D | 2:4 | ☑ | 2 |
| 34 | 723 | Joelinton | M | 3:1 | ☑ | 7 |
| 34 | 2507 | M. Almirón | F | 4:3 | ☑ | 24 |
| 34 | 2806 | Fabian Schär | D | 2:3 | ☑ | 5 |
| 34 | 2864 | Alexander Isak | F | 4:2 | ☑ | 14 |
| 34 | 2939 | C. Wilson | F | | ☐ | 9 |
| 34 | 10135 | Bruno Guimarães | M | 3:2 | ☑ | 39 |
| 34 | 18778 | H. Barnes | M | | ☐ | 15 |
| 34 | 18886 | M. Dúbravka | G | | ☐ | 1 |
| 34 | 18894 | J. Lascelles | D | | ☐ | 6 |
| 34 | 18901 | S. Longstaff | M | | ☐ | 36 |
| 34 | 18903 | M. Ritchie | M | | ☐ | 11 |
| 34 | 18911 | N. Pope | G | 1:1 | ☑ | 22 |
| 34 | 18941 | M. Targett | D | | ☐ | 13 |

Рис. 4.10. Форма імпорту даних матчу

На рис 4.10 показано форму, що імпортує дані конкретного матчу, а саме дані про футболістів, що заявлені на матч та стартові склад команд. Імпортуються: позиції гравців, схема за якою грає команда, індикатори, які позначають, що гравець у стартовому складі, номери гравців.

Для доступу до бази даних використано класи ADO.NET [6, 13, 16].

Код основний імпорту та статистики наведений у додатках А-В.

## 4.4. Розробка web-сайту на ASP.NET

У цьому пункті представлено основні сторінки веб-сайту, реалізованого за допомогою технологій ASP.NET Web Forms [3, 19]. Наведено опис ключових елементів інтерфейсу, їх призначення та функціональність. Особливу увагу приділено структурі сторінок, логіці переходів, адаптивності та зручності користування. Загалом, поданий матеріал демонструє зручність та цілісність

реалізованого веб-сайту.

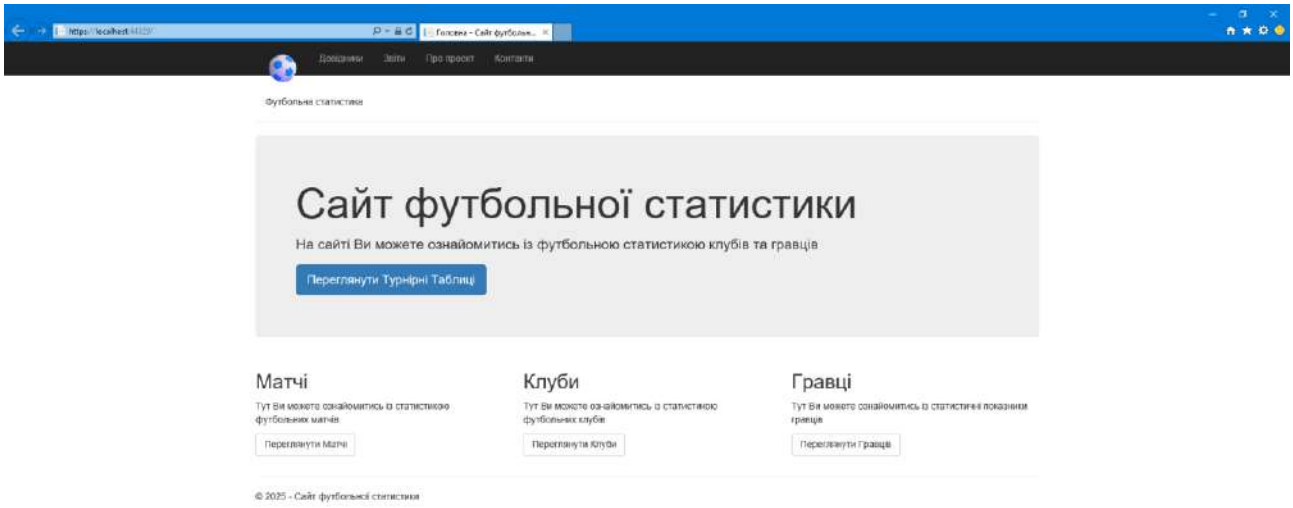



Рис. 4.11. Головна сторінка сайту

На рис 4.11 продемонстровано інтерфейс головної сторінки сайту. В центрі сторінки зустрічає великий напис про основні функції сайту, де одразу можна перейти до перегляду турнірних таблиць. Нижче зручно розташовані окремі елементи для перегляду статистики матчів, клубі та гравців. У шапці сайту розміщено логотип, та меню з інформацією про проект та контактами.

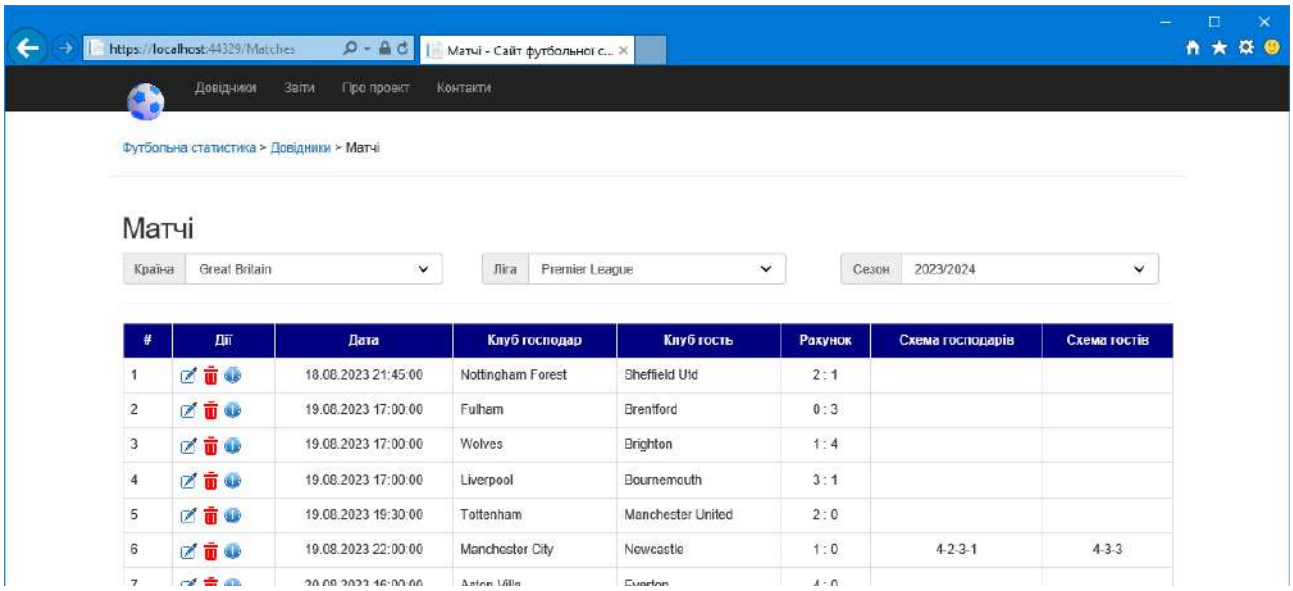



Рис. 4.12. Сторінка матчів ліги

Як видно з рис. 4.12, показано сторінку сайту з таблицею матчів ліги. Вона має інтуїтивно зрозумілий інтерфейс з приємною картинкою. Зверху зручно виведено елементи фільтрації за якими знаходяться матчі, ними є три поля з випадаючим списком: «Країни», «Ліга», «Сезон».

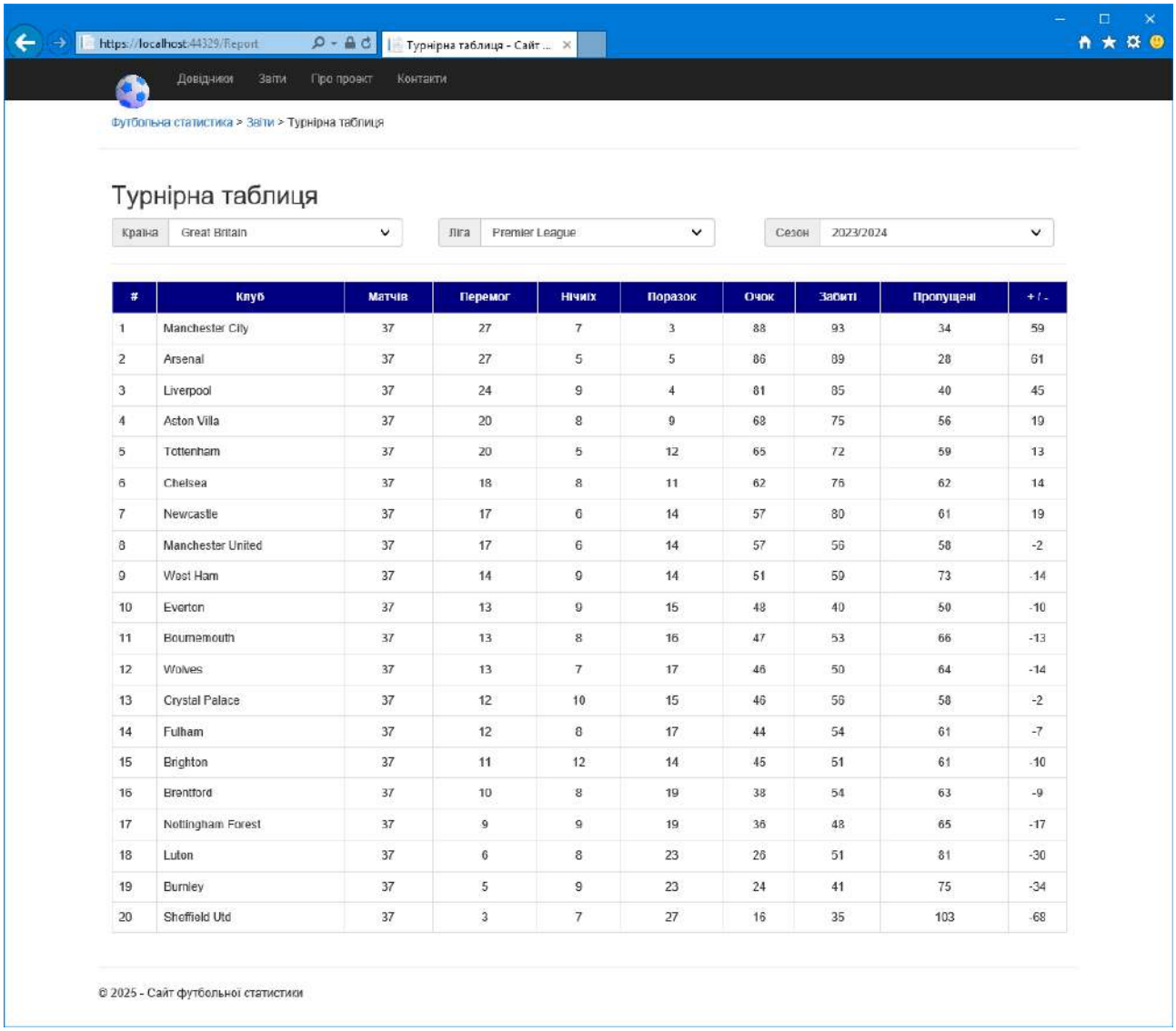



Рис. 4.13. Сторінка турнірної таблиці

На рис 4.13 показано інтерфейс сторінки турнірної таблиці. Вона є одною з основних адже кнопка з посиланням на неї зустрічає користувача одразу при вході на сайт. Тут реалізовано вже не раз використані способи фільтрації та виводу даних через поля з випадаючим списком. А результатом отримується велика турнірна таблиця з усіма командами ліги та їх показниками у сезоні, за якими і визначається переможець.

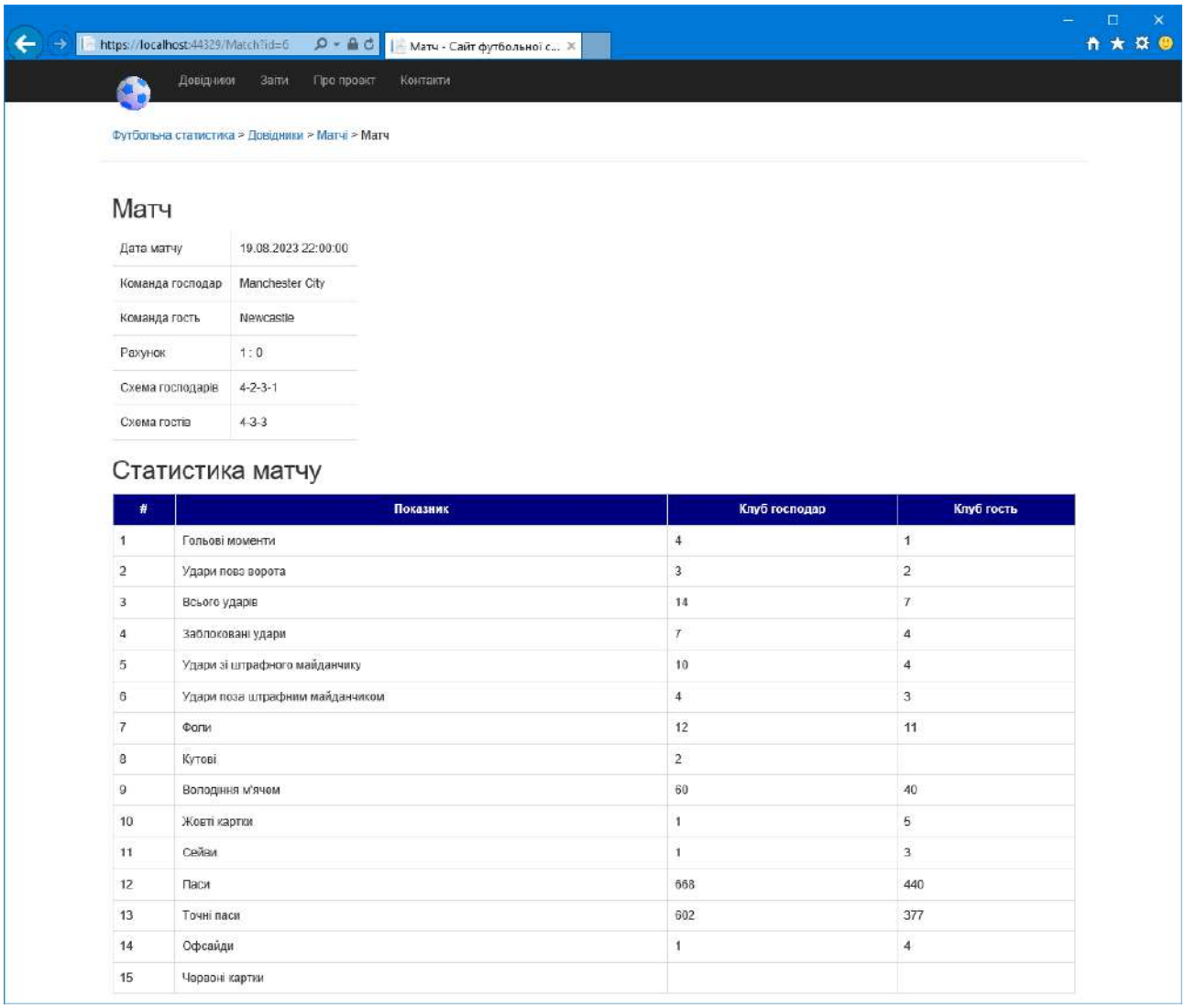



Рис. 4.14. Сторінка статистичних показників матчу

Як видно з рис. 4.14, на сторінці веб-сайту реалізовано інтерфейс детальних статистичних показників матчу. У верхній частині сторінки виведено загальні статистику матчу, таку як: дата, команди-учасники, рахунок, схеми команд. Під ними вже виведена велика таблиця статистичних даних матчу , що дозволяє робити глибокий аналіз гри команд. Уся інформація подана зручно, зрозуміло та приємно для ока.

Аналіз статистики матчу та гравців дозволяє тренерському штабу оцінювати ефективність роботи команди, приймати відповідні рішення та планувати наступні матчі.

Код сторінки статистичних показників матчу наведений у додатку Г.

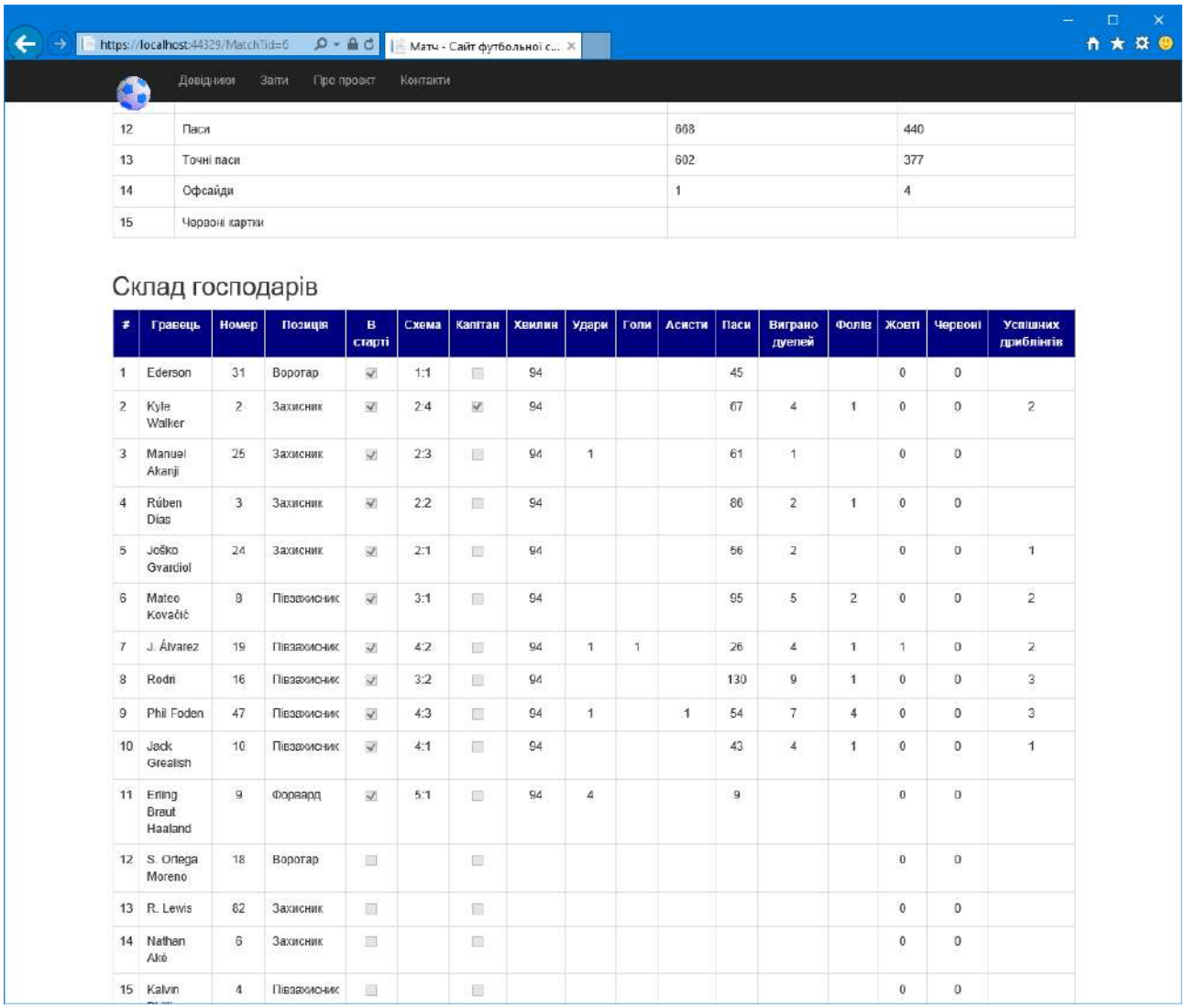



Рис. 4.15. Інтерфейс виведення складу та дій гравців на полі команди у матчі

На рис. 4.15 зображено інтерфейс виведення складу команди матчу та дій футболістів на полі. У таблиці виведена окрема колонка з усіма футболістами заявленими на матч, навпроти кожного з них показані його дії на футбольному полі: скільки хвилин провів, чи зіграв у стартовому складі, на кого був замінений, чи був замінений, кількість голів чи гольових передач, позиція на полі тощо. Кожен елемент статистики корелюється саме з позицією гравця на полі, адже дії та показники на полі у захисника, нападника чи воротаря кардинально відрізняються. Такий великий масив даних тому і зручно показувати у табличному вигляді, оскільки це дозволяє компактно й наглядно відобразити велику кількість показників одночасно.

Події матчу

| # | Подія | Хвилина | Команда | Гравець | Асист / Заміна | Екстра час |
|---|---|---|---|---|---|---|
| 1 | Жовта картка | 24 | Newcastle | A. Gordon | | |
| 2 | Гол | 31 | Manchester City | J. Álvarez | Phil Foden | |
| 3 | Жовта картка | 34 | Newcastle | S. Tonali | | |
| 4 | Жовта картка | 46 | Manchester City | J. Álvarez | | |
| 5 | Жовта картка | 53 | Newcastle | Joelinton | | |
| 6 | Заміна | 56 | Newcastle | H. Barnes | A. Gordon | |
| 7 | Заміна | 57 | Newcastle | S. Longstaff | Joelinton | |
| 8 | Жовта картка | 58 | Newcastle | H. Barnes | | |
| 9 | Заміна | 66 | Newcastle | C. Wilson | Alexander Isak | |
| 10 | Заміна | 67 | Newcastle | C. Wilson | Alexander Isak | |
| 11 | Жовта картка | 71 | Newcastle | Bruno Guimarães | | |

© 2025 - Сайт футбольної статистики

Рис. 4.16. Інтерфейс виведення подій матчу

На рис. 4.16 зображено інтерфейс подій матчу. Він виведений у окремій таблиці та слугує джерелом отримання даних про точний час та тип події, що відбувалися на полі. Така статистика не про окремого гравця чи команду, а про структурування та запису подій у хронологічному порядку.

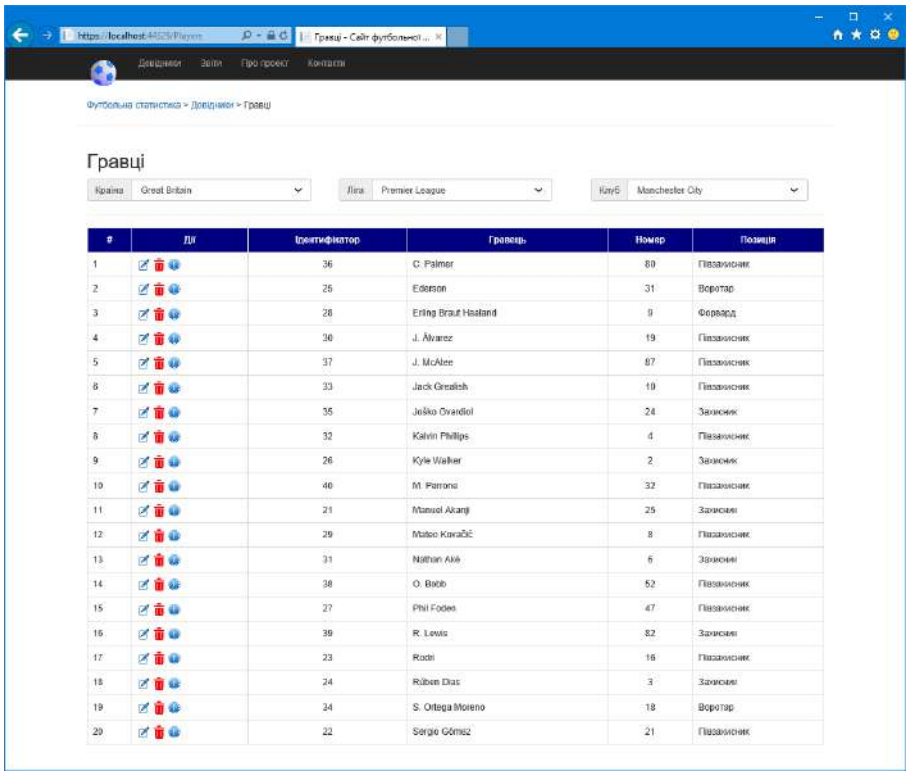



Рис. 4.17. Сторінка виведення складу гравців клубу

Як видно з рис. 4.17, показано інтерфейс сторінки веб-сайту зі складом футбольного клубу. Склад гравців зручно показаний у таблиці, яку можна фільтрувати виводячи склади різних команд.

## 4.5. Розробка форм в СУБД Access

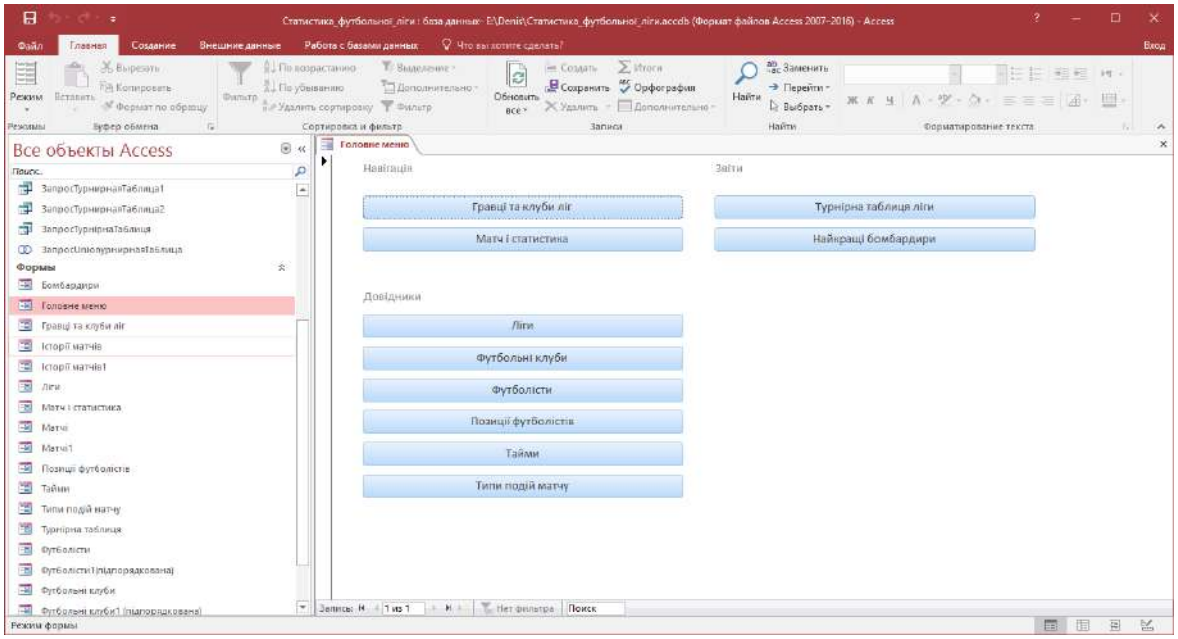



Рис. 4.18. Форма «Головне меню»

На рис. 4.18. зображено форму «Головне меню», що містить у собі усі необхідні інструменти для знаходження та переміщення між головними формами, таблицями, звітами.

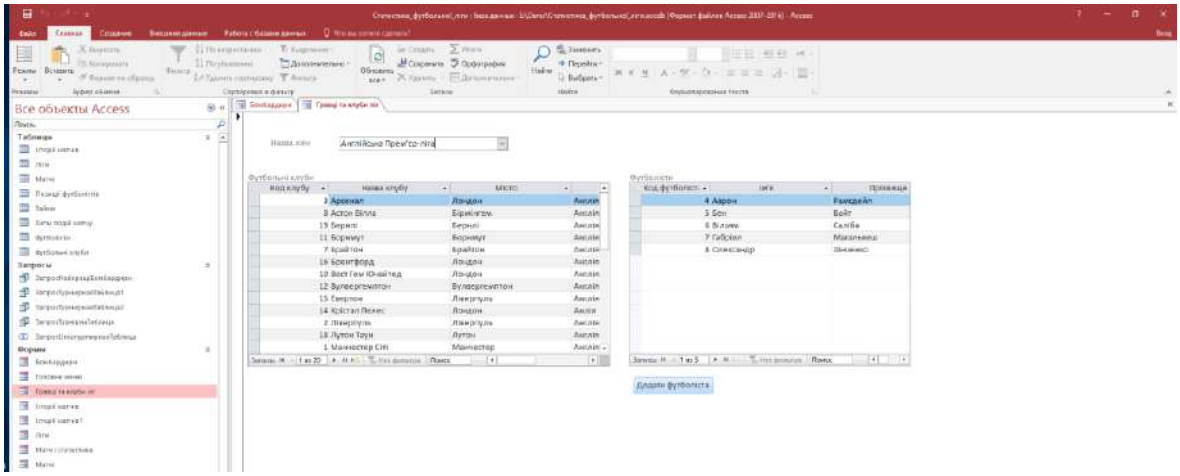



Рис. 4.19. Форма «Гравці та клуби ліг»

На рис. 4.19 показано форму «Гравці та клуби ліг». У верхній частині ми бачимо поле зі списком де треба вибрати потрібну лігу. Після вибору ліги у підпорядкованій таблиці нижче «Футбольні клуби» з’явиться список клубів, що мають відношення до вибраної ліги. Якщо зробити поле з будь-яким клубом активним, то праворуч з’явиться список футболістів цього клубу. Також в цій формі можна одразу додавати футболістів за допомогою кнопки під таблицею «Футболісти».

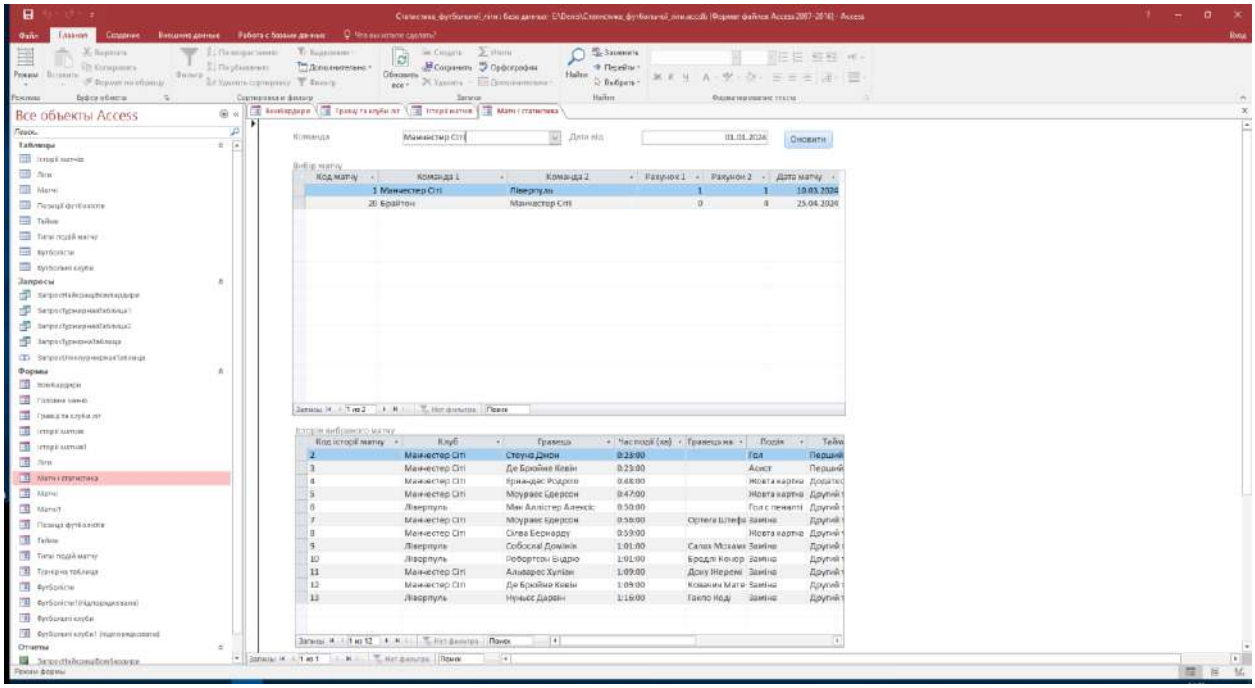


Рис. 4.20. Форма «Матч і статистика»

На рис. 4.20 виведено форму «Матч і статистика». У верхньому лівому кутку даної форми розташоване поле з випадаючим списком для вибору футбольного клубу. Правіше ми бачимо поле, в якому можемо вибрати початкову дату, тобто дату з якої потрібно вивести матчі вибраного клубу. Після вибору дати, щоб вибрані параметри вступили в дію потрібно натиснути кнопку «Оновити», розташовану справа від місця вибору дати. Основна інформація( а саме: команда-господар, команда-гість, рахунок та дата гри) виведуться у розташованій нижче таблиці матчі. А повна хронологія подій виведеться у

таблиці «Історія матчу», після того як поле конкретного матчу стане «активним».

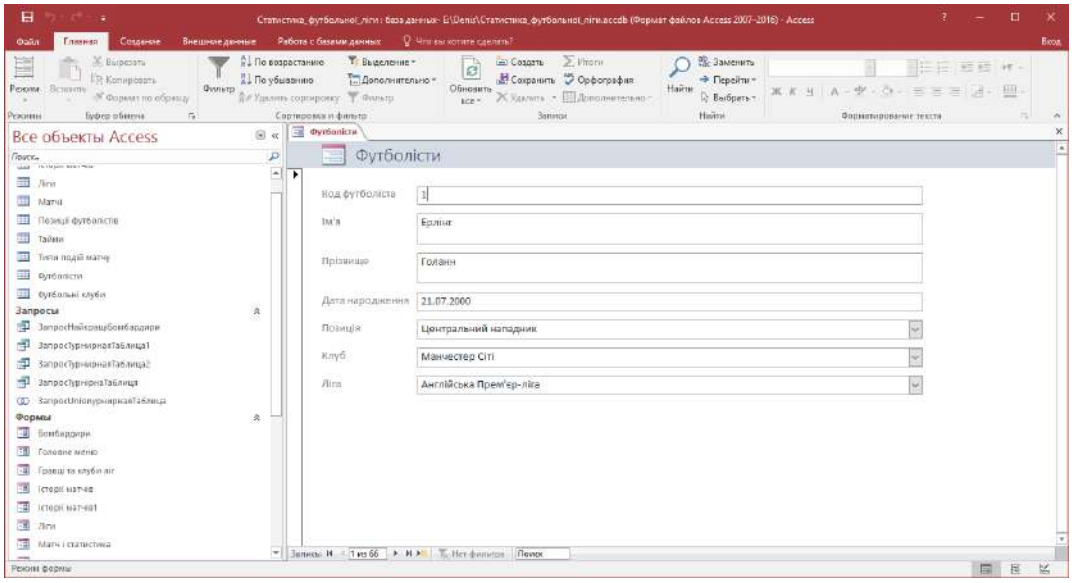



Рис. 4.21. Форма «Футболісти»

На рис. 4.21. зображено форму «Футболісти», що має усі необхідні поля, для вводу даних гравця. Насамперед це ім’я, прізвище та дата народження. Позицію та клуб гравця можна вибрати із поля з випадаючим списком.

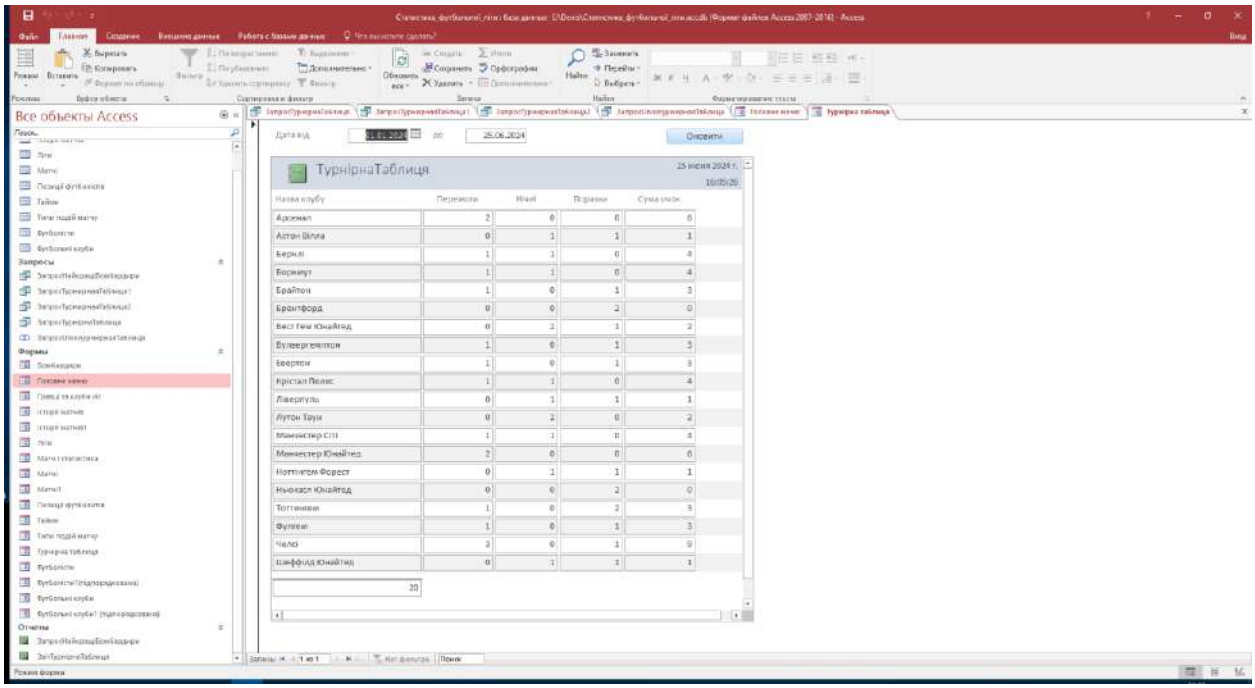



Рис. 4.22. Звіт «Турнірна таблиця»

На рис. 4.22 зображено звіт «Турнірна таблиця», такий результат ми

одержали за допомогою таких запитів:

1. SELECT Матчі.[Команда 1], Count(Матчі.[Код матчу]) AS [Count-Код матчу], Sum(IIf([Матчі]![Рахунок 1]>[Матчі]![Рахунок 2],1,0)) AS Перемоги, Sum(IIf([Матчі]![Рахунок 1]=[Матчі]![Рахунок 2],1,0)) AS Нічиї, Sum(IIf([Матчі]![Рахунок 1]<[Матчі]![Рахунок 2],1,0)) AS Поразки, Sum(IIf([Матчі]![Рахунок 1]>[Матчі]![Рахунок 2],1,0))*3+Sum(IIf([Матчі]![Рахунок 1]=[Матчі]![Рахунок 2],1,0)) AS [Сума очок]

FROM Матчі

WHERE [Матчі]![Дата матчу]>=[Формы]![Турнірна таблиця]![ПочатковаДата] AND [Матчі]![Дата матчу]<=[Формы]![Турнірна таблиця]![КінцеваДата]

GROUP BY Матчі.[Команда 1]

2. SELECT Матчі.[Команда 2], Count(Матчі.[Код матчу]) AS [Count-Код матчу], Sum(IIf([Матчі]![Рахунок 1]<[Матчі]![Рахунок 2],1,0)) AS Перемоги, Sum(IIf([Матчі]![Рахунок 1]=[Матчі]![Рахунок 2],1,0)) AS Нічиї, Sum(IIf([Матчі]![Рахунок 1]>[Матчі]![Рахунок 2],1,0)) AS Поразки, Sum(IIf([Матчі]![Рахунок 1]<[Матчі]![Рахунок 2],1,0))*3+Sum(IIf([Матчі]![Рахунок 1]=[Матчі]![Рахунок 2],1,0)) AS [Сума очок]

FROM Матчі

WHERE [Матчі]![Дата матчу]>=[Формы]![Турнірна таблиця]![ПочатковаДата] AND [Матчі]![Дата матчу]<=[Формы]![Турнірна таблиця]![КінцеваДата]

GROUP BY Матчі.[Команда 2]

3. SELECT ЗапросТурнирнаяТаблица1.[Команда 1], ЗапросТурнирнаяТаблица1.[Count-Код матчу], ЗапросТурнирнаяТаблица1.Перемоги, ЗапросТурнирнаяТаблица1.Нічиї, ЗапросТурнирнаяТаблица1.Поразки, ЗапросТурнирнаяТаблица1.[Сума очок]

FROM ЗапросТурнирнаяТаблица1

UNION ALL SELECT ЗапросТурнирнаяТаблица2.[Команда 2], ЗапросТурнирнаяТаблица2.[Count-Код матчу], ЗапросТурнирнаяТаблица2.Перемоги, ЗапросТурнирнаяТаблица2.Нічиї, ЗапросТурнирнаяТаблица2.Поразки, ЗапросТурнирнаяТаблица2.[Сума очок]

FROM ЗапросТурнирнаяТаблица2

4. SELECT [Футбольні клуби].[Назва клубу], Sum(ЗапросUnionурнирнаяТаблица.Перемоги) AS [Sum-Перемоги], Sum(ЗапросUnionурнирнаяТаблица.Нічиї) AS [Sum-Нічиї], Sum(ЗапросUnionурнирнаяТаблица.Поразки) AS [Sum-Поразки], Sum(ЗапросUnionурнирнаяТаблица.[Сума очок]) AS [Sum-Сума очок]

FROM ЗапросUnionурнирнаяТаблица INNER JOIN [Футбольні клуби] ON ЗапросUnionурнирнаяТаблица.[Команда 1] = [Футбольні клуби].[Код клубу]

GROUP BY [Футбольні клуби].[Назва клубу]

ORDER BY Sum(ЗапросUnionурнирнаяТаблица.[Сума очок]) DESC

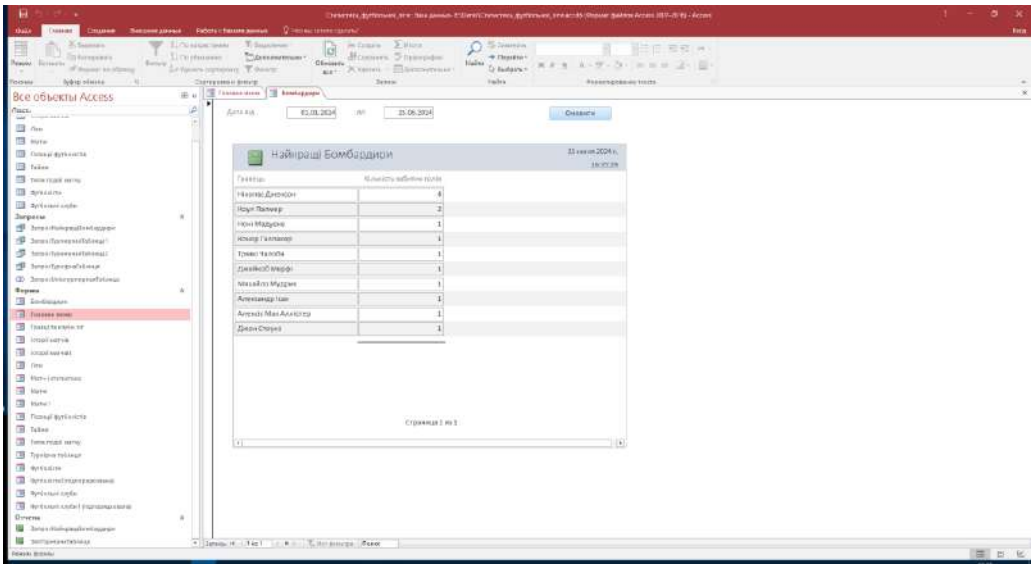



Рис. 4.23. Звіт «Найкращі бомбардири»

На рис. 4.23 зображено звіт «Найкращі бомбардири», такий результат ми одержали за допомогою такого запиту:

```
SELECT First([Футболісти]![Ім'я])+" "+First([Футболісти]![Прізвище]) AS Гравець, Count([Історії матчів].[Код історії матчу]) AS [Count-Код історії матчу]
FROM ([Історії матчів] INNER JOIN Футболісти ON [Історії матчів].Гравець = Футболісти.[Код футболіста]) INNER JOIN Матчі ON [Історії матчів].Матч = Матчі.[Код матчу]
WHERE ((([Історії матчів].Подія)=1 Or ([Історії матчів].Подія)=3) AND (([Матчі]![Дата матчу])>=[Формы]![Бомбардири]![ПочатковаДата] And ([Матчі]![Дата матчу])<=[Формы]![Бомбардири]![КінцеваДата]))
GROUP BY [Історії матчів].Гравець
ORDER BY Count([Історії матчів].[Код історії матчу]) DESC;
```

## Висновки до розділу 4

З використанням сучасної мови програмування C# та технологій .NET, Windows Forms, ASP.NET було розроблено програмне забезпечення та веб-сайт, які надають користувачу можливість отримувати аналітичні звіти щодо футбольних матчів. Також розроблено інструменти універсального імпорту даних із сторонніх систем, які надають аналітичні дані про футбольні матчі та гравців. Розроблене програмне забезпечення має гнучкий інтерфейс користувача при виведенні табличних і графічних звітів.

# ВИСНОВКИ

На сучасному етапі розвитку інформаційних технологій важливим питанням для розвитку та конкурентоспроможності клубів є аналіз стану гравців та прогнозування командної роботи. При цьому автоматизація збору, обробки та аналізу статистичної інформації дозволяє оперативно та швидко реагувати на проведення гри.

Було проведено аналіз підходів до автоматизації збору та обробки футбольної статистики. Було визначено основні характеристики команд та футболістів при здійснені аналізу футбольної інформації. Здійснено критичний огляд найбільш популярних ресурсів. В ході аналізу визначено перелік задач та модулів, які притаманні веб-сайтам з аналізу футбольної статистики. Було формалізовано вхідну інформацію для аналізу, а також визначено перелік і структуру звітів, які формують при аналізі футбольної статистики.

Було проаналізовано підходи та алгоритми розрахунку статистичних показників для аналізу ефективності виступів гравців у матчах та аналізу їх фізичної форми.

В ході виконання роботи було спроектовано базу даних та структуру її таблиць. Базу даних було реалізовано в середовищі сучасних систем управління базами даних СУБД SQL Server та Access. Спроектовані таблиці містять усю необхідну інформацію про футбольні матчі, його події та гравців і їх статистичні показники. Що дозволяє ефективно аналізувати перебіг матчів за допомогою sql-запитів.

З використанням сучасної мови програмування C# та технологій .NET, Windows Forms, ASP.NET було розроблено програмне забезпечення та веб-сайт, які надають користувачу можливість отримувати аналітичні звіти щодо футбольних матчів. Також розроблено інструменти універсального імпорту даних із сторонніх систем, які надають аналітичні дані про футбольні матчі та гравців. Розроблене програмне забезпечення має гнучкий інтерфейс користувача при виведенні табличних і графічних звітів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Andrew Troelsen, Phil Japikse (2022) Pro C# 10 with .NET 6, Apress Berkeley, CA, 1640 p,- https://doi.org/10.1007/978-1-4842-7869-7

2. Angelo Bobak, (2023) SQL Server Analytical Toolkit, Apress Berkeley, CA, 1055 p.- https://doi.org/10.1007/978-1-4842-8667-8

3. Beasley, R.E. (2020) Ajax Programming. In: Essential ASP. NET Web Forms Development. Springer, 565 p.- https://doi.org/10.1007/978-1-4842-5784-5

4. Bob Ward, (2022) SQL Server 2022 Revealed, Apress Berkeley, CA, 483 p.- https://doi.org/10.1007/978-1-4842-8894-8

5. Gerard Byrne (2022) Target C#, Apress Berkeley, CA, 1078 p.- https://doi.org/10.1007/978-1-4842-8619-7

6. Mahesh Chand (2002) A Programmer's Guide to ADO.NET in C#, A Programmer's Guide to ADO.NET in C#, Apress Berkeley, CA, 718 p.- https://doi.org/10.1007/978-1-4302-1133-4

7. Matthew MacDonald (2005) Pro .NET 2.0 Windows Forms and Custom Controls in C#, Apress Berkeley, CA, 1080 p.- https://doi.org/10.1007/978-1-4302-0110-6

8. Matthew McGiffen (2022) Pro Encryption in SQL Server 2022, Apress Berkeley, CA, 351 p.- https://doi.org/10.1007/978-1-4842-8664-7

9. Microsoft Access 2016: навчальний посібник в електронному вигляді / Укладачі В.О. Нелюбов, Ю.Ю. Білак. Ужгород: ДВНЗ «УжНУ», 2019. 73 с.

10. Peter, A. Carter (2025) Expert Scripting and Automation for SQL Server DBAs, Apress Berkeley, CA, 290 p.- https://doi.org/10.1007/979-8-8688-1151-7

11. Peter, A. Carter (2023) Pro SQL Server 2022 Administration, Apress Berkeley, CA, 987 p.- https://doi.org/10.1007/978-1-4842-8864-1

12. Radek Vystavěl (2021) C# Programming for Absolute Beginners, Apress Berkeley, CA, 369 p.- https://doi.org/10.1007/978-1-4842-7147-6

13. Talbot, D., & Chand, M. (2008). Applied ADO. NET: Building Data-driven Solutions. Apress, 928 p.- https://doi.org/10.1007/978-1-4302-0759-7

14. Thomas LaRock, Enrico van de Laar () 2023 Pro SQL Server 2022 Wait

Statistics, Apress Berkeley, CA, 402 p.- https://doi.org/10.1007/978-1-4842-8771-2

15. Vaskaran Sarcar (2023), Introducing Functional Programming Using C#, Apress Berkeley, CA, 301 p.- https://doi.org/10.1007/978-1-4842-9697-4

16. Бошемин, Боб. Основы ADO.NET.: Пер. с англ. – М.: Издательський дом «Вильямс», 2003. – 448с. : ил. – Парал. тит. англ.

17. Ватсон Б. C#4.0 на примерах. — СПб.: БХВ-Петербург, 2011. — 608 с:

18. Верес О.М., Рішняк І.В. Проектування баз даних у середовищі MS Access 2010.-Львів: Львівська політехніка, 2016.-232 с.

19. Мак-Дональд, Мэтью, Фримен, Адам, Шпушта, Марио. Microsoft ASP.NET 4 с примерами на C# 2010 для профессионалов, 4-е изд. : Пер. с англ. — М. : ООО "И.Д. Вильяме", 2011. — 1424 с.

20. Надія Баловсяк, Ія Григоришин, Людмила Кулібаба. Система управління базами даних Microsoft Access для самостійного вивчення.-К.: Дакор, КНТ, 2006.-156 с.

21. Нейгел, Кристиан, Ивьен, Билл, Глинн, Джей, Уотсон, Карли. C# 4.0 и платформа .NET 4 для профессионалов. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2011. — 1440 с.

22. Шилдт, Герберт. C# 4.0: полное руководство. : Пер. с англ. — М. : ООО "И.Д. Вильяме", 2011. — 1056 с.

23. Шпортько О. В. Розробка баз даних в СУБД Microsoft Access : практикум для студ. вищ. навч. закл. / О. В. Шпортько, Л. В. Шпортько, П. С. Янчук. - Рівне : МЕГУ ім. акад. С. Дем’янчука, 2023.- Вид. 2-ге доп. і перероб. - 184 с.

# ДОДАТКИ

# ДОДАТОК А

## Модуль імпорту даних матчів (MatchesImportForm.cs)

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using WindowsFormsTestApp2.Models.Import;
using System.Configuration;

namespace WindowsFormsTestApp2
{
    public partial class MatchesImportForm : Form
    {
        int? LeagueId = null;

        public class ComboBoxCountryItem
        {
            public string Title { get; set; }
            public int Id { get; set; }  // or string, depending on your
case

            public override string ToString()
            {
                return Title; // what will be displayed in ComboBox
            }
        }

        public class ComboBoxLeagueItem
        {
            public string Title { get; set; }
            public int Id { get; set; }  // or string, depending on your
```

```
case
                public int ExternalId { get; set; }

                public override string ToString()
                {
                    return Title; // what will be displayed in ComboBox
                }
            }

            MatchesImportModel matches = new MatchesImportModel();
            bool isUrl = false;
            bool isView = true;

            public MatchesImportForm()
            {
                InitializeComponent();
            }


            private void backgroundWorkerMatches_DoWork(object sender,
DoWorkEventArgs e)
            {
                try
                {
                    using (SqlConnection conn = new
SqlConnection(AppConfig.ConnectionStr))
                    {
                        conn.Open();
                        try
                        {
                            string response;
                            if (isUrl)
                            {
                                string url = AppConfig.ApiUrl +
"games/list?From=2023-08-15&LeagueId=" + LeagueId.ToString() + "&To=2024-05-30";
                                HttpWebRequest httpWebRequest =
(HttpWebRequest)WebRequest.Create(url);
                                httpWebRequest.Headers.Add("apiKey",
AppConfig.ApiKey);
                                httpWebRequest.Method = "GET";
                                HttpWebResponse httpWebResponse =
(HttpWebResponse)httpWebRequest.GetResponse();
```

```
                                using (StreamReader streamReader = new
StreamReader(httpWebResponse.GetResponseStream()))
                                {
                                    response = streamReader.ReadToEnd();
                                }
                            }
                            else
                            {
                                using (StreamReader streamReader = new
StreamReader("matches2.json"))
                                {
                                    response = streamReader.ReadToEnd();
                                }
                            }

                            matches =
JsonConvert.DeserializeObject<MatchesImportModel>(response); //?
                            if (matches.Status.ToLower() != "ok")
                            {
                                backgroundWorkerMatches.ReportProgress(-1,
"Error load data from api");
                                return;
                            }

                            if (!isView)
                            {
                                int i = 0;
                                backgroundWorkerMatches.ReportProgress(0,
matches.Data.Count);

                                //INSERT INTO [table] (field1, field2)
VALUES()
                                //INSERT INTO [table] (field1, season_id,
field2) VALUES(@b, SELECT TOP 1 season_id FROM dbo.seasons WHERE
external_uid=@season_uid ,@a)

                                SqlCommand seasonExist = new
SqlCommand("SELECT TOP 1 season_id FROM [dbo].[seasons] WHERE external_uid =
@external_uid", conn);
                                SqlCommand matchExist = new SqlCommand("SELECT
TOP 1 match_id FROM [dbo].[matches] WHERE external_id=@external_id", conn);
                                SqlCommand clubExist = new SqlCommand("SELECT
```

```
TOP 1 club_id FROM [dbo].[football_clubs] WHERE external_id=@external_id",
conn);
                                    SqlCommand clubAdd = new SqlCommand("INSERT
INTO [dbo].[football_clubs] (club_name, league_id, country_id, external_id)
VALUES(@club_name, @league_id, @country_id, @external_id)", conn);
                                    SqlCommand countryExist = new
SqlCommand("SELECT TOP 1 country_id FROM [dbo].[Countries] WHERE code=@code",
conn);
                                    SqlCommand leagueExist = new
SqlCommand("SELECT TOP 1 league_id FROM [dbo].[leagues] WHERE
external_id=@external_id", conn);
                                    SqlCommand matchAdd = new SqlCommand("INSERT
INTO [dbo].[matches] (external_id, home_team_id, away_team_id, home_result,
away_result, match_date, home_ht_result, away_ht_result, home_ft_result,
away_ft_result, season_id, league_id, status) VALUES(@external_id,
@home_team_id, @away_team_id, @home_result, @away_result, @match_date,
@home_ht_result, @away_ht_result, @home_ft_result, @away_ft_result, @season_id,
@league_id, @status)", conn);



                                    foreach (var match in matches.Data)
                                    {
                                        if
(backgroundWorkerMatches.CancellationPending == true)
                                        {
                                            e.Cancel = true;
                                            break;
                                        }

                                        matchExist.Parameters.Clear();

matchExist.Parameters.AddWithValue("@external_id", match.Id);
                                        object match_id =
matchExist.ExecuteScalar();
                                        if (match_id != null)
                                        {
                                            i++;

backgroundWorkerMatches.ReportProgress(i);
                                            continue;
                                        }
```

```
                                    seasonExist.Parameters.Clear();

seasonExist.Parameters.AddWithValue("@external_uid", match.Season.Uid);
                                    object season_id =
seasonExist.ExecuteScalar();
                                    if (season_id == null)
                                    {
                                        i++;

backgroundWorkerMatches.ReportProgress(i);
                                        continue;
                                    }
                                    countryExist.Parameters.Clear();

countryExist.Parameters.AddWithValue("@code", match.Season.League.Country.Code);
                                    object country_id =
countryExist.ExecuteScalar();
                                    if (country_id == null)
                                    {
                                        i++;

backgroundWorkerMatches.ReportProgress(i);
                                        continue;
                                    }
                                    leagueExist.Parameters.Clear();

leagueExist.Parameters.AddWithValue("@external_id", match.Season.League.Id);
                                    object league_id =
leagueExist.ExecuteScalar();
                                    if (league_id == null)
                                    {
                                        i++;

backgroundWorkerMatches.ReportProgress(i);
                                        continue;
                                    }

                                    clubExist.Parameters.Clear();

clubExist.Parameters.AddWithValue("@external_id", match.HomeTeam.Id);
                                    object home_team_id =
clubExist.ExecuteScalar();
```

```
                                    if (home_team_id == null)
                                    {
                                        clubAdd.Parameters.Clear();

clubAdd.Parameters.AddWithValue("@club_name", match.HomeTeam.Name);

clubAdd.Parameters.AddWithValue("@league_id", (int)league_id);

clubAdd.Parameters.AddWithValue("@country_id", (int)country_id);

clubAdd.Parameters.AddWithValue("@external_id", match.HomeTeam.Id);
                                        clubAdd.ExecuteNonQuery();
                                        home_team_id =
clubExist.ExecuteScalar();
                                    }

                                    clubExist.Parameters.Clear();

clubExist.Parameters.AddWithValue("@external_id", match.AwayTeam.Id);
                                    object away_team_id =
clubExist.ExecuteScalar();
                                    if (away_team_id == null)
                                    {
                                        clubAdd.Parameters.Clear();

clubAdd.Parameters.AddWithValue("@club_name", match.AwayTeam.Name);

clubAdd.Parameters.AddWithValue("@league_id", (int)league_id);

clubAdd.Parameters.AddWithValue("@country_id", (int)country_id);

clubAdd.Parameters.AddWithValue("@external_id", match.AwayTeam.Id);
                                        clubAdd.ExecuteNonQuery();
                                        away_team_id =
clubExist.ExecuteScalar();
                                    }

                                    matchAdd.Parameters.Clear();

matchAdd.Parameters.AddWithValue("@external_id",match.Id);

matchAdd.Parameters.AddWithValue("@home_team_id",(int)home_team_id);
```

```
matchAdd.Parameters.AddWithValue("@away_team_id",(int)away_team_id);

matchAdd.Parameters.AddWithValue("@home_result",match.HomeResult);

matchAdd.Parameters.AddWithValue("@away_result",match.AwayResult);

matchAdd.Parameters.AddWithValue("@match_date",match.Date);

matchAdd.Parameters.AddWithValue("@home_ht_result",match.HomeHTResult);

matchAdd.Parameters.AddWithValue("@away_ht_result",match.AwayHTResult);

matchAdd.Parameters.AddWithValue("@home_ft_result",match.HomeFTResult);

matchAdd.Parameters.AddWithValue("@away_ft_result",match.AwayFTResult);

matchAdd.Parameters.AddWithValue("@season_id", (int)season_id);

matchAdd.Parameters.AddWithValue("@league_id", (int)league_id);

matchAdd.Parameters.AddWithValue("@status",match.Status);
                                        matchAdd.ExecuteNonQuery();
                                        i++;
                                        backgroundWorkerMatches.ReportProgress(i);
                                    }
                                }
                            }
                            catch (SqlException error)
                            {
                                backgroundWorkerMatches.ReportProgress(-1,
error.Message);
                                return;
                            }
                            catch (Exception error)
                            {
                                backgroundWorkerMatches.ReportProgress(-1,
error.Message);
                                return;
                            }
                        }
                    }
```

```
                catch (SqlException error)
                {
                    backgroundWorkerMatches.ReportProgress(-1, error.Message);
                }
                catch (InvalidOperationException error)
                {
                    backgroundWorkerMatches.ReportProgress(-1, error.Message);
                }
                catch (Exception error)
                {
                    backgroundWorkerMatches.ReportProgress(-1, error.Message);
                }
            }

            private void toolStripButtonImportMatches_Click(object sender,
EventArgs e)
            {
                if (LeagueId == null) {
                    MessageBox.Show("Обери лігу");
                    return;
                }

                isView = false;
                isUrl = toolStripButtonFromUrl.Checked;
                toolStripProgressBar1.Minimum = 0;
                toolStripProgressBar1.Maximum = 0;
                toolStripProgressBar1.Value = 0;
                toolStripButtonStop.Enabled = true;
                toolStripButtonImportMatches.Enabled = false;
                toolStripButtonShowMatches.Enabled = false;
                toolStripLabelError.Text = "";

                backgroundWorkerMatches.RunWorkerAsync();
            }

            private void MatchesImportForm_Load(object sender, EventArgs e)
            {
                try
                {
                    using (SqlConnection conn = new
SqlConnection(AppConfig.ConnectionStr))
                    {
```

```
                        conn.Open();
                        SqlCommand countries = new SqlCommand("SELECT
country_id, name FROM [dbo].[countries] ORDER BY name", conn);
                        var reader = countries.ExecuteReader();
                        toolStripComboBoxCountries.Items.Clear();
                        while (reader.Read())
                        {
                            var item = new ComboBoxCountryItem();
                            item.Title = reader["name"].ToString();
                            item.Id = (int)reader["country_id"];
                            toolStripComboBoxCountries.Items.Add(item);
                            //toolStripComboBoxCountries.Items.Add(new
ComboBoxCountryItem() { Title = reader["name"].ToString(), Id =
(int)reader["country_id"]});
                        }
                    }
                }
                catch (SqlException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (InvalidOperationException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (Exception error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
            }

            private void toolStripComboBoxLeagues_SelectedIndexChanged(object
sender, EventArgs e)
            {
                LeagueId =
((ComboBoxLeagueItem)toolStripComboBoxLeagues.SelectedItem).ExternalId;
            }

            private void
toolStripComboBoxCountries_SelectedIndexChanged(object sender, EventArgs e)
            {
                //очищаем вібор лиги
```

```
                LeagueId = null;
                toolStripComboBoxLeagues.Items.Clear();

                //берем country_id выбранной страны
                int countryId =
((ComboBoxCountryItem)toolStripComboBoxCountries.SelectedItem).Id;
                try
                {
                    using (SqlConnection conn = new
SqlConnection(AppConfig.ConnectionStr))
                    {
                        conn.Open();
                        SqlCommand leagues = new SqlCommand("SELECT league_id,
external_id, league_name FROM [dbo].[leagues] WHERE external_id IS NOT NULL AND
[dbo].[leagues].[country_id] = @country_id ORDER BY league_name", conn);
                        leagues.Parameters.Clear();
                        leagues.Parameters.AddWithValue("@country_id",
countryId);
                        var reader = leagues.ExecuteReader();
                        while (reader.Read())
                        {
                            toolStripComboBoxLeagues.Items.Add(new
ComboBoxLeagueItem() { Title = reader["league_name"].ToString(), Id =
(int)reader["league_id"], ExternalId = (int)reader["external_id"] });
                        }
                    }
                }
                catch (SqlException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (InvalidOperationException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (Exception error)
                {
                    toolStripLabelError.Text = error.ToString();
                }

            }
```

```
            private void backgroundWorkerMatches_ProgressChanged(object
sender, ProgressChangedEventArgs e)
            {
                if (e.ProgressPercentage < 0)
                {
                    toolStripLabelError.Text = ((string)e.UserState);
                }
                else if (e.ProgressPercentage == 0)
                {
                    toolStripProgressBar1.Maximum = (int)e.UserState;
                    toolStripProgressBar1.Value = e.ProgressPercentage;
                }
                else
                {
                    toolStripProgressBar1.Value = e.ProgressPercentage;
                }

            }

            private void toolStripButtonShowMatches_Click(object sender,
EventArgs e)
            {
                isView = true;
                isUrl = toolStripButtonFromUrl.Checked;
                toolStripProgressBar1.Minimum = 0;
                toolStripProgressBar1.Maximum = 0;
                toolStripProgressBar1.Value = 0;
                toolStripButtonStop.Enabled = true;
                toolStripButtonImportMatches.Enabled = false;
                toolStripButtonShowMatches.Enabled = false;
                toolStripLabelError.Text = "";

                backgroundWorkerMatches.RunWorkerAsync();

            }

            private void toolStripButtonStop_Click(object sender, EventArgs e)
            {
                if (backgroundWorkerMatches.IsBusy)
                {
                    toolStripButtonStop.Enabled = false;
                    backgroundWorkerMatches.CancelAsync();
```

```
                }
            }

            private void backgroundWorkerMatches_RunWorkerCompleted(object
sender, RunWorkerCompletedEventArgs e)
            {
                toolStripButtonStop.Enabled = false;
                toolStripButtonImportMatches.Enabled = true;
                toolStripButtonShowMatches.Enabled = true;
                dataGridViewMatches.DataSource = matches.Data;
            }
        }
    }
```

# ДОДАТОК Б

## Модуль імпорту даних матчу (MatchImportForm.cs)

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data.SqlClient;
using System.IO;
using System.Net;
using System.Windows.Forms;
using WindowsFormsTestApp2.Models.Import;

namespace WindowsFormsTestApp2
{
    public partial class MatchImportForm : Form
    {
        MatchInfoImportModel MatchInfo = new MatchInfoImportModel();
        public int MatchId { get; set; }
        public int ExternalId { get; set; }
        public Dictionary<string, int> StatisticsParams = new
Dictionary<string, int>();

        bool isUrl = false;
        bool isView = true;

        public MatchImportForm()
        {
            InitializeComponent();
        }

        private void backgroundWorker1_DoWork(object sender,
DoWorkEventArgs e)
        {
            try
            {
                using (SqlConnection conn = new
SqlConnection(AppConfig.ConnectionStr))
                {
                    conn.Open();
                    try
```

```
                        {
                            string response;
                            if (isUrl)
                            {
                                string url = AppConfig.ApiUrl + "games/" +
ExternalId.ToString();
                                HttpWebRequest httpWebRequest =
(HttpWebRequest)WebRequest.Create(url);
                                httpWebRequest.Headers.Add("apiKey",
AppConfig.ApiKey);
                                httpWebRequest.Method = "GET";
                                HttpWebResponse httpWebResponse =
(HttpWebResponse)httpWebRequest.GetResponse();
                                using (StreamReader streamReader = new
StreamReader(httpWebResponse.GetResponseStream()))
                                {
                                    response = streamReader.ReadToEnd();
                                }
                            }
                            else
                            {
                                using (StreamReader streamReader = new
StreamReader("match_info.json"))
                                {
                                    response = streamReader.ReadToEnd();
                                }
                            }

                            MatchInfo =
JsonConvert.DeserializeObject<MatchInfoImportModel>(response);
                            if (MatchInfo.Status.ToLower() != "ok") {
                                backgroundWorker1.ReportProgress(-1, "Error
load data from api");
                                return;
                            }
                            //import leagues
                            if (!isView)
                            {

                                int i = 0;
                                backgroundWorker1.ReportProgress(0,
MatchInfo.Data.LineupPlayers.Count);
```

```
                                SqlCommand statExist = new SqlCommand("SELECT
TOP 1 match_statistic_id FROM [dbo].[match_statistics] WHERE match_id=@match_id
AND club_id=@club_id AND statistic_param_id=@statistic_param_id", conn);
                                SqlCommand statAdd = new SqlCommand("INSERT
INTO [dbo].[match_statistics] (match_id, club_id, statistic_param_id, value)
VALUES(@match_id, @club_id, @statistic_param_id, @value)", conn);

                                SqlCommand coachExist = new SqlCommand("SELECT
TOP 1 coach_id FROM [dbo].[coaches] WHERE external_id=@external_id", conn);
                                SqlCommand coachAdd = new SqlCommand("INSERT
INTO [dbo].[coaches] (coach_name, external_id) VALUES(@coach_name,
@external_id)", conn);

                                SqlCommand positionExist = new
SqlCommand("SELECT TOP 1 position_id FROM [dbo].[players_position] WHERE
position_code=@position_code", conn);
                                SqlCommand positionAdd = new
SqlCommand("INSERT INTO [dbo].[players_position] (position_name, position_code)
VALUES(@position_name, @position_code)", conn);
                                SqlCommand clubExist = new SqlCommand("SELECT
TOP 1 club_id FROM [dbo].[football_clubs] WHERE external_id=@external_id",
conn);

                                SqlCommand playerExist = new
SqlCommand("SELECT TOP 1 player_id FROM [dbo].[players] WHERE
external_id=@external_id", conn);
                                SqlCommand playerAdd = new SqlCommand("INSERT
INTO [dbo].[players] (club_id,position_id,player_fullname, number_str, number,
external_id) VALUES(@club_id,@position_id,@player_fullname, @number_str,
@number, @external_id)", conn);

                                SqlCommand playerStatsExist = new
SqlCommand("SELECT TOP 1 match_player_id FROM [dbo].[matches_players] WHERE
match_id=@match_id AND club_id=@club_id AND player_id=@player_id", conn);
                                SqlCommand playerStatAdd = new
SqlCommand("INSERT INTO [dbo].[matches_players] (match_id, club_id, player_id,
position_id, is_start, grid, number_str, number, is_stat) VALUES(@match_id,
@club_id, @player_id, @position_id, @is_start, @grid, @number_str, @number,
@is_stat)", conn);
                                SqlCommand playerStatAdd1 = new
SqlCommand("INSERT INTO [dbo].[matches_players] (match_id, club_id, player_id,
```

```
position_id, is_start, grid, number_str, number,
is_stat,capitan,substitute,offsides,shots_total,goals_total,goals_conceded,goals
_assists,goals_saves,passes_total,passes_key,passes_accuracy,tackles_total,tackl
es_blocks,tackles_interceptions,duels_total,duels_won,dribbles_attempts,dribbles
_success,dribbles_past,fouls_drawn,fouls_committed,cards_yellow,cards_red,penalt
y_won,penalty_commited,penalty_scored,penalty_saved,penalty_missed,rating,minute
s) VALUES(@match_id, @club_id, @player_id, @position_id, @is_start, @grid,
@number_str, @number,
@is_stat,@capitan,@substitute,@offsides,@shots_total,@goals_total,@goals_concede
d,@goals_assists,@goals_saves,@passes_total,@passes_key,@passes_accuracy,@tackle
s_total,@tackles_blocks,@tackles_interceptions,@duels_total,@duels_won,@dribbles
_attempts,@dribbles_success,@dribbles_past,@fouls_drawn,@fouls_committed,@cards_
yellow,@cards_red,@penalty_won,@penalty_commited,@penalty_scored,@penalty_saved,
@penalty_missed,@rating,@minutes)", conn);

                                    clubExist.Parameters.Clear();

clubExist.Parameters.AddWithValue("@external_id",
MatchInfo.Data.Game.HomeTeam.Id);
                                    object home_team_id =
clubExist.ExecuteScalar();
                                    clubExist.Parameters.Clear();

clubExist.Parameters.AddWithValue("@external_id",
MatchInfo.Data.Game.AwayTeam.Id);
                                    object away_team_id =
clubExist.ExecuteScalar();

                                    foreach (var prop in
MatchInfo.Data.Statistics.GetType().GetProperties())
                                    {
                                        int club_id;
                                        if (prop.Name.Contains("Home"))
                                        {
                                            club_id = (int)home_team_id;
                                        } else
                                        {
                                            club_id = (int)away_team_id;
                                        }

                                        string name = prop.Name.Replace("Home",
"").Replace("Away", "");
```

```
                                    int stat_id = StatisticsParams[name];

                                    object v =
prop.GetValue(MatchInfo.Data.Statistics);
                                    statExist.Parameters.Clear();

statExist.Parameters.AddWithValue("@match_id", MatchId);

statExist.Parameters.AddWithValue("@club_id", club_id);

statExist.Parameters.AddWithValue("@statistic_param_id", stat_id);

                                    if (statExist.ExecuteScalar() == null)
                                    {
                                        statAdd.Parameters.Clear();

statAdd.Parameters.AddWithValue("@match_id", MatchId);

statAdd.Parameters.AddWithValue("@club_id", club_id);

statAdd.Parameters.AddWithValue("@statistic_param_id", stat_id);
                                        if (v == null)
                                        {

statAdd.Parameters.AddWithValue("@value", DBNull.Value);
                                        }
                                        else
                                        {

statAdd.Parameters.AddWithValue("@value", v);
                                        }
                                        statAdd.ExecuteNonQuery();
                                    }
                                }
                                //
                                if (MatchInfo.Data.Lineups.HomeCoach != null)
                                {
                                    coachExist.Parameters.Clear();

coachExist.Parameters.AddWithValue("@external_id",
MatchInfo.Data.Lineups.HomeCoach.Id);
```

```
                                    object coach_id =
coachExist.ExecuteScalar();
                                    if (coach_id == null)
                                    {
                                        coachAdd.Parameters.Clear();

coachAdd.Parameters.AddWithValue("@coach_name",
MatchInfo.Data.Lineups.HomeCoach.Name);

coachAdd.Parameters.AddWithValue("@external_id",
MatchInfo.Data.Lineups.HomeCoach.Id);
                                        coachAdd.ExecuteNonQuery();
                                        coach_id = coachExist.ExecuteScalar();
                                    }
                                    SqlCommand coachUpdate = new
SqlCommand("UPDATE [dbo].[matches] SET home_coach_id=@home_coach_id WHERE
match_id=@match_id", conn);
                                    coachUpdate.Parameters.Clear();

coachUpdate.Parameters.AddWithValue("@home_coach_id", coach_id);

coachUpdate.Parameters.AddWithValue("@match_id", MatchId);
                                    coachUpdate.ExecuteNonQuery();
                                }
                                if (MatchInfo.Data.Lineups.AwayCoach != null)
                                {
                                    coachExist.Parameters.Clear();

coachExist.Parameters.AddWithValue("@external_id",
MatchInfo.Data.Lineups.AwayCoach.Id);
                                    object coach_id =
coachExist.ExecuteScalar();
                                    if (coach_id == null)
                                    {
                                        coachAdd.Parameters.Clear();

coachAdd.Parameters.AddWithValue("@coach_name",
MatchInfo.Data.Lineups.AwayCoach.Name);

coachAdd.Parameters.AddWithValue("@external_id",
MatchInfo.Data.Lineups.AwayCoach.Id);
                                        coachAdd.ExecuteNonQuery();
```

```
                                        coach_id = coachExist.ExecuteScalar();
                                    }
                                    SqlCommand coachUpdate = new
SqlCommand("UPDATE [dbo].[matches] SET away_coach_id=@away_coach_id WHERE
match_id=@match_id", conn);
                                    coachUpdate.Parameters.Clear();

coachUpdate.Parameters.AddWithValue("@away_coach_id", coach_id);

coachUpdate.Parameters.AddWithValue("@match_id", MatchId);
                                    coachUpdate.ExecuteNonQuery();
                                }
                                SqlCommand formationUpdate = new
SqlCommand("UPDATE [dbo].[matches] SET
home_formation=@home_formation,away_formation=@away_formation  WHERE
match_id=@match_id", conn);
                                formationUpdate.Parameters.Clear();

formationUpdate.Parameters.AddWithValue("@home_formation",
MatchInfo.Data.Lineups.HomeFormation ?? "");

formationUpdate.Parameters.AddWithValue("@away_formation",
MatchInfo.Data.Lineups.AwayFormation ?? "");

formationUpdate.Parameters.AddWithValue("@match_id", MatchId);
                                formationUpdate.ExecuteNonQuery();

                                //

                                SqlCommand eventExist = new SqlCommand("SELECT
TOP 1 history_id FROM [dbo].[matches_history] WHERE external_id=@external_id",
conn);
                                SqlCommand eventAdd = new SqlCommand("INSERT
INTO [dbo].[matches_history]
(match_id,club_id,player_id,assist_player_id,event_id,period_id,elapsed,extra,su
bstitution_number,external_id)
VALUES(@match_id,@club_id,@player_id,@assist_player_id,@event_id,@period_id,@ela
psed,@extra,@substitution_number,@external_id)", conn);

                                SqlCommand eventTypeExist = new
SqlCommand("SELECT TOP 1 event_id FROM [dbo].[event_types] WHERE
external_id=@external_id", conn);
```

```
                                SqlCommand eventTypeAdd = new
SqlCommand("INSERT INTO [dbo].[event_types] (event_name, external_id)
VALUES(@event_name, @external_id)", conn);

                                foreach (var ev in MatchInfo.Data.Events)
                                {
                                    playerExist.Parameters.Clear();

playerExist.Parameters.AddWithValue("@external_id", ev.Player.Id);
                                    object event_player_id =
playerExist.ExecuteScalar();

                                    clubExist.Parameters.Clear();

clubExist.Parameters.AddWithValue("external_id", ev.TeamId);
                                    object event_club_id =
clubExist.ExecuteScalar();

                                    eventExist.Parameters.Clear();

eventExist.Parameters.AddWithValue("@external_id", ev.Id);
                                    if (eventExist.ExecuteScalar() == null)
                                    {
                                        eventAdd.Parameters.Clear();

eventAdd.Parameters.AddWithValue("@match_id", MatchId);

eventAdd.Parameters.AddWithValue("@club_id", (int)event_club_id);

eventAdd.Parameters.AddWithValue("@player_id", (int)event_player_id);
                                        if (ev.AssistPlayer != null)
                                        {
                                            playerExist.Parameters.Clear();

playerExist.Parameters.AddWithValue("@external_id", ev.AssistPlayer.Id);
                                            event_player_id =
playerExist.ExecuteScalar();

eventAdd.Parameters.AddWithValue("@assist_player_id", (int)event_player_id);
                                        }
                                        else
                                        {
```

```
eventAdd.Parameters.AddWithValue("@assist_player_id", DBNull.Value);
                                        }
                                        eventTypeExist.Parameters.Clear();

eventTypeExist.Parameters.AddWithValue("@external_id", ev.Type);
                                        object event_id =
eventTypeExist.ExecuteScalar();
                                        if (event_id == null)
                                        {
                                            eventTypeAdd.Parameters.Clear();

eventTypeAdd.Parameters.AddWithValue("@event_name", ev.Name);

eventTypeAdd.Parameters.AddWithValue("@external_id", ev.Type);
                                            eventTypeAdd.ExecuteNonQuery();
                                            event_id =
eventTypeExist.ExecuteScalar();
                                        }

eventAdd.Parameters.AddWithValue("@event_id", (int)event_id);
                                        if (ev.Extra != null)
                                        {
                                            if (ev.Extra > 15)
                                            {

eventAdd.Parameters.AddWithValue("@period_id", 6);
                                            }
                                            else
                                            {

eventAdd.Parameters.AddWithValue("@period_id", 5);
                                            }
                                        }
                                        else if (ev.Elapsed > 45)
                                        {

eventAdd.Parameters.AddWithValue("@period_id", 2);
                                        }
                                        else
                                        {
```

```
eventAdd.Parameters.AddWithValue("@period_id", 1);
                                        }


eventAdd.Parameters.AddWithValue("@elapsed", ev.Elapsed);
                                        if (ev.Extra != null)
                                        {

eventAdd.Parameters.AddWithValue("@extra", ev.Extra);
                                        }
                                        else
                                        {

eventAdd.Parameters.AddWithValue("@extra", DBNull.Value);
                                        }
                                        if (ev.Type == 3)
                                        {

eventAdd.Parameters.AddWithValue("@substitution_number",
int.Parse(ev.Name.Split(' ')[1]));
                                        }
                                        else
                                        {

eventAdd.Parameters.AddWithValue("@substitution_number", DBNull.Value);
                                        }


eventAdd.Parameters.AddWithValue("@external_id", ev.Id);
                                        eventAdd.ExecuteNonQuery();
                                    }
                                }

                                foreach (var player in
MatchInfo.Data.LineupPlayers)
                                {
                                    if (backgroundWorker1.CancellationPending
== true)
                                    {
                                        e.Cancel = true;
                                        break;
                                    }
```

```
                                    //add position
                                    positionExist.Parameters.Clear();

positionExist.Parameters.AddWithValue("@position_code", player.Position);
                                    object position_id =
positionExist.ExecuteScalar();
                                    if (position_id == null)
                                    {
                                        positionAdd.Parameters.Clear();

positionAdd.Parameters.AddWithValue("@position_name", player.Position);

positionAdd.Parameters.AddWithValue("@position_code", player.Position);
                                        positionAdd.ExecuteNonQuery();
                                        position_id =
positionExist.ExecuteScalar();
                                    }

                                    clubExist.Parameters.Clear();

clubExist.Parameters.AddWithValue("external_id", player.TeamId);
                                    object club_id =
clubExist.ExecuteScalar();
                                    if (club_id == null)
                                    {
                                        backgroundWorker1.ReportProgress(-1,
"Error imported club: " + player.TeamId);
                                        return;
                                    }
                                    //

                                    //add to players
                                    playerExist.Parameters.Clear();

playerExist.Parameters.AddWithValue("@external_id", player.PlayerId);
                                    object player_id =
playerExist.ExecuteScalar();
                                    if (player_id == null)
                                    {
                                        playerAdd.Parameters.Clear();
```

```
playerAdd.Parameters.AddWithValue("@club_id", club_id);

playerAdd.Parameters.AddWithValue("@position_id", position_id);

playerAdd.Parameters.AddWithValue("@player_fullname", player.PlayerName);

playerAdd.Parameters.AddWithValue("@number_str", player.Number.ToString());

playerAdd.Parameters.AddWithValue("@number", player.Number);

playerAdd.Parameters.AddWithValue("@external_id", player.PlayerId);
                                        playerAdd.ExecuteNonQuery();
                                        player_id =
playerExist.ExecuteScalar();
                                    }
                                    //end to players


                                    playerStatsExist.Parameters.Clear();

playerStatsExist.Parameters.AddWithValue("@match_id", MatchId);

playerStatsExist.Parameters.AddWithValue("@club_id", club_id);

playerStatsExist.Parameters.AddWithValue("@player_id", player_id);

                                    if (playerStatsExist.ExecuteScalar() !=
null) {
                                        i++;
                                        backgroundWorker1.ReportProgress(i);
                                        continue;
                                    }

                                    var stats =
MatchInfo.Data.PlayerStats.Find(p => p.PlayerId == player.PlayerId);
                                    if (stats == null)
                                    {
                                        playerStatAdd.Parameters.Clear();

playerStatAdd.Parameters.AddWithValue("@match_id", MatchId);

playerStatAdd.Parameters.AddWithValue("@club_id", club_id);
```

```
playerStatAdd.Parameters.AddWithValue("@player_id", player_id);

playerStatAdd.Parameters.AddWithValue("@position_id", position_id);

playerStatAdd.Parameters.AddWithValue("@is_start", player.StartXI?1:0);

playerStatAdd.Parameters.AddWithValue("@grid", player.Grid);

playerStatAdd.Parameters.AddWithValue("@number_str", player.Number.ToString());

playerStatAdd.Parameters.AddWithValue("@number", player.Number);

playerStatAdd.Parameters.AddWithValue("@is_stat", 0);
                                        playerStatAdd.ExecuteNonQuery();
                                        i++;
                                        backgroundWorker1.ReportProgress(i);
                                        continue;
                                    }

                                    playerStatAdd1.Parameters.Clear();

playerStatAdd1.Parameters.AddWithValue("@match_id", MatchId);

playerStatAdd1.Parameters.AddWithValue("@club_id", club_id);

playerStatAdd1.Parameters.AddWithValue("@player_id", player_id);

playerStatAdd1.Parameters.AddWithValue("@position_id", position_id);

playerStatAdd1.Parameters.AddWithValue("@is_start", player.StartXI ? 1 : 0);

playerStatAdd1.Parameters.AddWithValue("@grid", player.Grid);

playerStatAdd1.Parameters.AddWithValue("@number_str", player.Number.ToString());

playerStatAdd1.Parameters.AddWithValue("@number", player.Number);

playerStatAdd1.Parameters.AddWithValue("@is_stat", 1);

playerStatAdd1.Parameters.AddWithValue("@capitan",stats.Capitan ? 1 : 0);
```

```
playerStatAdd1.Parameters.AddWithValue("@substitute",stats.Substitute ? 1 : 0);


playerStatAdd1.Parameters.AddWithValue("@offsides", stats.Offsides);

playerStatAdd1.Parameters.AddWithValue("@shots_total", stats.ShotsTotal);

playerStatAdd1.Parameters.AddWithValue("@goals_total", stats.GoalsTotal);

playerStatAdd1.Parameters.AddWithValue("@goals_conceded", stats.GoalsConceded);

playerStatAdd1.Parameters.AddWithValue("@goals_assists",stats.GoalsAssists);

playerStatAdd1.Parameters.AddWithValue("@goals_saves",stats.GoalsSaves);

playerStatAdd1.Parameters.AddWithValue("@passes_total",stats.PassesTotal);

playerStatAdd1.Parameters.AddWithValue("@passes_key",stats.PassesKey);

playerStatAdd1.Parameters.AddWithValue("@passes_accuracy",stats.PassesAccuracy);

playerStatAdd1.Parameters.AddWithValue("@tackles_total",stats.TacklesTotal);

playerStatAdd1.Parameters.AddWithValue("@tackles_blocks",stats.TacklesBlocks);

playerStatAdd1.Parameters.AddWithValue("@tackles_interceptions",stats.TacklesInt
erceptions);

playerStatAdd1.Parameters.AddWithValue("@duels_total",stats.DuelsTotal);

playerStatAdd1.Parameters.AddWithValue("@duels_won",stats.DuelsWon);

playerStatAdd1.Parameters.AddWithValue("@dribbles_attempts",stats.DribblesAttemp
ts);

playerStatAdd1.Parameters.AddWithValue("@dribbles_success",stats.DribblesSuccess
);

playerStatAdd1.Parameters.AddWithValue("@dribbles_past",stats.DribblesPast);

playerStatAdd1.Parameters.AddWithValue("@fouls_drawn",stats.FoulsDrawn);
```

```
playerStatAdd1.Parameters.AddWithValue("@fouls_committed",stats.FoulsCommitted);

playerStatAdd1.Parameters.AddWithValue("@cards_yellow",stats.CardsYellow);

playerStatAdd1.Parameters.AddWithValue("@cards_red",stats.CardsRed);

playerStatAdd1.Parameters.AddWithValue("@penalty_won",stats.PenaltyWon);

playerStatAdd1.Parameters.AddWithValue("@penalty_commited",stats.PenaltyCommited
);

playerStatAdd1.Parameters.AddWithValue("@penalty_scored",stats.PenaltyScored);

playerStatAdd1.Parameters.AddWithValue("@penalty_saved",stats.PenaltySaved);

playerStatAdd1.Parameters.AddWithValue("@penalty_missed", stats.PenaltyMissed);

playerStatAdd1.Parameters.AddWithValue("@rating",stats.Rating);

playerStatAdd1.Parameters.AddWithValue("@minutes",stats.Minutes);
                                        foreach (SqlParameter param in
playerStatAdd1.Parameters)
                                        {
                                            if (param.Value == null) param.Value =
DBNull.Value;
                                        }
                                        playerStatAdd1.ExecuteNonQuery();
                                        //end to matches_players


                                        i++;
                                        backgroundWorker1.ReportProgress(i);
                                    }
                                }

                            }
                            catch (SqlException error)
                            {
                                backgroundWorker1.ReportProgress(-1,
error.Message);
                                return;
                            }
```

```
                        catch (Exception error)
                        {
                            backgroundWorker1.ReportProgress(-1,
error.Message);
                            return;
                        }
                    }
                }
                catch (SqlException error)
                {
                    backgroundWorker1.ReportProgress(-1, error.Message);
                }
                catch (InvalidOperationException error)
                {
                    backgroundWorker1.ReportProgress(-1, error.Message);
                }
                catch (Exception error)
                {
                    backgroundWorker1.ReportProgress(-1, error.Message);
                }
                //

            }

            private void backgroundWorker1_ProgressChanged(object sender,
ProgressChangedEventArgs e)
            {
                if (e.ProgressPercentage < 0) {
                    toolStripLabelError.Text = ((string)e.UserState);
                }
                else if (e.ProgressPercentage == 0)
                {
                    toolStripProgressBar1.Maximum = (int)e.UserState;
                    toolStripProgressBar1.Value = e.ProgressPercentage;
                }
                else {
                    toolStripProgressBar1.Value = e.ProgressPercentage;
                }
            }

            private void backgroundWorker1_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
```

```
            {
                toolStripButtonStop.Enabled = false;
                toolStripButtonImport.Enabled = true;
                toolStripButtonShow.Enabled = true;
                if (MatchInfo.Data != null && MatchInfo.Data.LineupPlayers !=
null) {
                    dataGridViewLeagues.DataSource =
MatchInfo.Data.LineupPlayers;
                }
                else
                {
                    dataGridViewLeagues.DataSource = new
List<MatchLineupPlayerImportModel>();
                }
            }


            private void toolStripButtonImport_Click(object sender, EventArgs
e)
            {
                isView = false;
                isUrl = toolStripButtonFromUrl.Checked;
                toolStripProgressBar1.Minimum = 0;
                toolStripProgressBar1.Maximum = 0;
                toolStripProgressBar1.Value = 0;
                toolStripButtonStop.Enabled = true;
                toolStripButtonImport.Enabled = false;
                toolStripButtonShow.Enabled = false;
                toolStripLabelError.Text = "";

                backgroundWorker1.RunWorkerAsync();

            }

            private void toolStripButtonStop_Click(object sender, EventArgs e)
            {
                if (backgroundWorker1.IsBusy)
                {
                    toolStripButtonStop.Enabled = false;
                    backgroundWorker1.CancelAsync();
                }
            }
```

```
        private void toolStripButtonShow_Click(object sender, EventArgs e)
        {
            isView = true;
            isUrl = toolStripButtonFromUrl.Checked;
            toolStripProgressBar1.Minimum = 0;
            toolStripProgressBar1.Maximum = 0;
            toolStripProgressBar1.Value = 0;
            toolStripButtonStop.Enabled = true;
            toolStripButtonImport.Enabled = false;
            toolStripButtonShow.Enabled = false;
            toolStripLabelError.Text = "";

            backgroundWorker1.RunWorkerAsync();
        }

        private void MatchImportForm_Load(object sender, EventArgs e)
        {
            try
            {
                using (SqlConnection conn = new
SqlConnection(AppConfig.ConnectionStr))
                {
                    conn.Open();
                    SqlCommand leagues = new SqlCommand("SELECT * FROM
[dbo].[statistics_params]", conn);
                    var reader = leagues.ExecuteReader();
                    while (reader.Read())
                    {

StatisticsParams[reader["statistic_param_code"].ToString()] =
(int)reader["statistic_param_id"];
                    }
                }
            }
            catch (SqlException error)
            {
                toolStripLabelError.Text = error.ToString();
            }
            catch (InvalidOperationException error)
            {
                toolStripLabelError.Text = error.ToString();
```

```
            }
            catch (Exception error)
            {
                toolStripLabelError.Text = error.ToString();
            }

        }
    }
}
```

# ДОДАТОК В

## Форма статистики матчу

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace WindowsFormsTestApp2
{
    public partial class MatchesForm : Form
    {

        public class ComboBoxCountryItem
        {
            public string Title { get; set; }
            public int Id { get; set; }  // or string, depending on your
case

            public override string ToString()
            {
                return Title; // what will be displayed in ComboBox
            }
        }

        public class ComboBoxLeagueItem
        {
            public string Title { get; set; }
            public int Id { get; set; }  // or string, depending on your
case

            public override string ToString()
            {
                return Title; // what will be displayed in ComboBox
            }
```

```
        }

        public MatchesForm()
        {
            InitializeComponent();
        }

        private void
toolStripComboBoxCountries_SelectedIndexChanged(object sender, EventArgs e)
        {
            toolStripComboBoxLeagues.Items.Clear();
            toolStripComboBoxSeasons.Items.Clear();

            //берем country_id выбранной страны
            int countryId =
((ComboBoxCountryItem)toolStripComboBoxCountries.SelectedItem).Id;
            try
            {
                using (SqlConnection conn = new
SqlConnection(AppConfig.ConnectionStr))
                {
                    conn.Open();
                    SqlCommand leagues = new SqlCommand("SELECT league_id,
league_name FROM [dbo].[leagues] WHERE external_id IS NOT NULL AND
[dbo].[leagues].[country_id] = @country_id ORDER BY league_name", conn);
                    leagues.Parameters.Clear();
                    leagues.Parameters.AddWithValue("@country_id",
countryId);
                    var reader = leagues.ExecuteReader();
                    while (reader.Read())
                    {
                        toolStripComboBoxLeagues.Items.Add(new
ComboBoxLeagueItem() { Title = reader["league_name"].ToString(), Id =
(int)reader["league_id"]});
                    }
                }
            }
            catch (SqlException error)
            {
                toolStripLabelError.Text = error.ToString();
            }
            catch (InvalidOperationException error)
```

```
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (Exception error)
                {
                    toolStripLabelError.Text = error.ToString();
                }

            }

            private void TableReportForm_Load(object sender, EventArgs e)
            {
                try
                {
                    using (SqlConnection conn = new
SqlConnection(AppConfig.ConnectionStr))
                    {
                        conn.Open();
                        SqlCommand countries = new SqlCommand("SELECT
country_id, name FROM [dbo].[countries] ORDER BY name", conn);
                        using (var reader = countries.ExecuteReader())
                        {
                            toolStripComboBoxCountries.Items.Clear();
                            while (reader.Read())
                            {
                                var item = new ComboBoxCountryItem();
                                item.Title = reader["name"].ToString();
                                item.Id = (int)reader["country_id"];
                                toolStripComboBoxCountries.Items.Add(item);
                            }
                        }
                    }
                }
                catch (SqlException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (InvalidOperationException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (Exception error)
```

```
                {
                    toolStripLabelError.Text = error.ToString();
                }

            }

            private void toolStripComboBoxLeagues_SelectedIndexChanged(object
sender, EventArgs e)
            {
                toolStripComboBoxSeasons.Items.Clear();
                dataGridView1.AllowUserToAddRows = false;
                dataGridView1.DataSource = null;
                if (toolStripComboBoxLeagues.SelectedIndex < 0)
                {
                    return;
                }

                int leagueId =
((ComboBoxLeagueItem)toolStripComboBoxLeagues.SelectedItem).Id;
                try
                {
                    using (SqlConnection conn = new
SqlConnection(AppConfig.ConnectionStr))
                    {
                        conn.Open();
                        SqlCommand seasons = new SqlCommand("SELECT season_id,
season_year FROM [dbo].[seasons] WHERE league_id=@league_id ORDER BY
season_year", conn);
                        seasons.Parameters.Clear();
                        seasons.Parameters.AddWithValue("@league_id",
leagueId);
                        using (var reader = seasons.ExecuteReader())
                        {
                            while (reader.Read())
                            {
                                toolStripComboBoxSeasons.Items.Add(new
ComboBoxLeagueItem() { Title = reader["season_year"].ToString() + "/" +
((int)reader["season_year"] + 1).ToString(), Id = (int)reader["season_id"] });
                            }
                        }
                        SqlCommand clubs = new SqlCommand("SELECT club_id,
club_name FROM [dbo].[football_clubs] WHERE league_id=" + leagueId.ToString() +
```

```
" ORDER BY club_name", conn);
                        using (var reader = clubs.ExecuteReader())
                        {
                            HomeClub.DisplayMember = "Title";
                            HomeClub.ValueMember = "Id";
                            AwayClub.DisplayMember = "Title";
                            AwayClub.ValueMember = "Id";

                            HomeClub.Items.Clear();
                            AwayClub.Items.Clear();
                            while (reader.Read())
                            {
                                var item = new ComboBoxCountryItem();
                                item.Title = reader["club_name"].ToString();
                                item.Id = (int)reader["club_id"];
                                HomeClub.Items.Add(item);
                                AwayClub.Items.Add(item);
                            }
                        }

                    }
                }
                catch (SqlException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (InvalidOperationException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (Exception error)
                {
                    toolStripLabelError.Text = error.ToString();
                }

            }

            private void toolStripComboBoxSeasons_SelectedIndexChanged(object
sender, EventArgs e)
            {
                if (toolStripComboBoxSeasons.SelectedIndex < 0)
                {
```

```
                    dataGridView1.AllowUserToAddRows = false;
                    dataGridView1.DataSource = null;
                    return;
                }
                int seasonId =
((ComboBoxLeagueItem)toolStripComboBoxSeasons.SelectedItem).Id;
                try
                {
                    using (SqlConnection conn = new
SqlConnection(AppConfig.ConnectionStr))
                    {
                        conn.Open();
                        string query = "SELECT * FROM [dbo].[matches] WHERE
season_id=" + seasonId.ToString() + " ORDER BY match_date";
                        SqlDataAdapter adapter = new SqlDataAdapter(query,
conn);
                        DataTable table = new DataTable();
                        adapter.Fill(table); // Заповнює DataTable
                        dataGridView1.AutoGenerateColumns = false;
                        dataGridView1.DataSource = table; // Прив'язка до
DataGridView
                        dataGridView1.AllowUserToAddRows = true;
                    }
                }
                catch (SqlException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (InvalidOperationException error)
                {
                    toolStripLabelError.Text = error.ToString();
                }
                catch (Exception error)
                {
                    toolStripLabelError.Text = error.ToString();
                }

            }

            private void dataGridView1_RowPostPaint(object sender,
DataGridViewRowPostPaintEventArgs e)
            {
```

```
                if (dataGridView1.AllowUserToAddRows &&
dataGridView1.Rows[e.RowIndex].IsNewRow)
                {
                    return;
                }
                // Отримуємо номер рядка(починаючи з 1)
                string rowNumber = (e.RowIndex + 1).ToString();

                // Визначаємо, де малювати номер (у заголовку рядка)
                var centerFormat = new StringFormat()
                {
                    Alignment = StringAlignment.Center,
                    LineAlignment = StringAlignment.Center
                };

                Rectangle headerBounds = new Rectangle(
                    e.RowBounds.Left,
                    e.RowBounds.Top,
                    dataGridView1.RowHeadersWidth,
                    e.RowBounds.Height);

                // Малюємо текст
                e.Graphics.DrawString(rowNumber,
                                      this.Font,
                                      SystemBrushes.ControlText,
                                      headerBounds,
                                      centerFormat);
            }

            private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
            {
                if (e.ColumnIndex == InfoAction.DisplayIndex &&
!dataGridView1.Rows[e.RowIndex].IsNewRow)
                {
                    var form = new MatchForm();
                    form.MatchId =
(int)dataGridView1.Rows[e.RowIndex].Cells[MatchId.DisplayIndex].Value;
                    form.FormBorderStyle = FormBorderStyle.None;
                    form.TopLevel = false;
                    this.Parent.Controls[0].Hide();
                    this.Parent.Controls.Add(form);
```

```
                form.Dock = DockStyle.Fill;
                form.Show();
            }
        }
    }
}
```

# ДОДАТОК Г

## Сторінка вебсайту для виведення статистики матчу

```
      <%@ Page Title="Матчі" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Matches.aspx.cs"
Inherits="FootballWebApplication.Matches" %>
      <asp:Content ID="Content1" ContentPlaceHolderID="MainContent"
runat="server">
          <div class="container">
              <div class="row">
                  <div class="col-md-12">
                      <h2>Матчі</h2>
                      <div class="row">
                          <div class="col-md-4">
                              <div class="input-group">
                                  <span class="input-group-addon">Країна</span>
                                  <asp:DropDownList ID="DropDownList1"
runat="server" CssClass="form-control" DataSourceID="SqlDataSource1"
DataTextField="name" DataValueField="country_id"
AutoPostBack="True"></asp:DropDownList>
                              </div>
                          </div>
                          <div class="col-md-4">
                              <div class="input-group">
                                  <span class="input-group-addon">Ліга</span>
                                  <asp:DropDownList ID="DropDownList2"
runat="server" CssClass="form-control" DataSourceID="SqlDataSource2"
DataTextField="league_name" DataValueField="league_id"
AutoPostBack="True"></asp:DropDownList>
                              </div>
                          </div>
                          <div class="col-md-4">
                              <div class="input-group">
                                  <span class="input-group-addon">Сезон</span>
                                  <asp:DropDownList ID="DropDownList3"
runat="server" CssClass="form-control" DataSourceID="SqlDataSource3"
DataTextField="year" DataValueField="season_id"
AutoPostBack="True"></asp:DropDownList>
                              </div>
                          </div>
                          <asp:SqlDataSource ID="SqlDataSource1" runat="server"
```

```
ConnectionString="<%$ ConnectionStrings:Football_leagueConnectionString %>"
SelectCommand="SELECT [country_id], [name] FROM [countries] ORDER BY
name"></asp:SqlDataSource>
                        <asp:SqlDataSource ID="SqlDataSource2" runat="server"
ConnectionString="<%$ ConnectionStrings:Football_leagueConnectionString %>"
SelectCommand="SELECT [league_id], [league_name] FROM [leagues] WHERE country_id
= @CountryID  ORDER BY league_name">
                            <SelectParameters>
                                <asp:ControlParameter Name="CountryID"
ControlID="DropDownList1" PropertyName="SelectedValue" Type="Int32" />
                            </SelectParameters>
                        </asp:SqlDataSource>
                        <asp:SqlDataSource ID="SqlDataSource3" runat="server"
ConnectionString="<%$ ConnectionStrings:Football_leagueConnectionString %>"
SelectCommand="SELECT [season_id], CONCAT(season_year, '/', season_year+1) as
year FROM [seasons] WHERE league_id = @LeagueID  ORDER BY season_year">
                            <SelectParameters>
                                <asp:ControlParameter Name="LeagueID"
ControlID="DropDownList2" PropertyName="SelectedValue" Type="Int32" />
                            </SelectParameters>
                        </asp:SqlDataSource>
                    </div>
                    <hr>
                    <div class="row">
                        <div class="col-md-12">
                            <asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource4" AutoGenerateColumns="False" CssClass="table"
OnRowCommand="GridView1_RowCommand">
                                <Columns>

                                    <asp:TemplateField HeaderText="#">
                                        <ItemTemplate>
                                            <%# Container.DataItemIndex + 1 %>
                                        </ItemTemplate>
                                    </asp:TemplateField>
                                    <asp:TemplateField HeaderText="Дії">
                                        <ItemTemplate>
                                            <asp:LinkButton ID="btnEdit"
runat="server"
                                                            Text=""
                                                            CommandName="edit"
```

```
CommandArgument='<%# Eval("match_id") %>'>
                                                <asp:Image ID="Image1"
runat="server" ImageUrl="~/images/edit.png" Width="20" />
                                            </asp:LinkButton>

                                            <asp:LinkButton ID="btnRemove"
runat="server"
                                                            Text=""

CommandName="remove"

CommandArgument='<%# Eval("match_id") %>'>
                                                <asp:Image ID="Image2"
runat="server" ImageUrl="~/images/remove.png" Width="20" />
                                            </asp:LinkButton>
                                            <asp:LinkButton ID="btnInfo"
runat="server"
                                                            Text=""
                                                            CommandName="info"

CommandArgument='<%# Eval("match_id") %>'>
                                                <asp:Image ID="Image3"
runat="server" ImageUrl="~/images/info.png" Width="20" />
                                            </asp:LinkButton>

                                            </ItemTemplate>
                                        </asp:TemplateField>
                                    <asp:BoundField DataField="match_date"
HeaderText="Дата">
                                        <ItemStyle CssClass="text-center" />
                                    </asp:BoundField>
                                    <asp:BoundField DataField="home_name"
HeaderText="Клуб господар" />
                                    <asp:BoundField DataField="away_name"
HeaderText="Клуб гость" />
                                    <asp:TemplateField HeaderText="Рахунок">
                                        <ItemTemplate>
                                            <%# Eval("home_result") %> : <%#
Eval("away_result") %>
                                        </ItemTemplate>
                                        <ItemStyle CssClass="text-center" />
                                    </asp:TemplateField>
```

```
                                    <asp:BoundField DataField="home_formation"
HeaderText="Схема господарів">
                                        <ItemStyle CssClass="text-center" />
                                    </asp:BoundField>
                                    <asp:BoundField DataField="away_formation"
HeaderText="Схема гостів">
                                        <ItemStyle CssClass="text-center" />
                                    </asp:BoundField>
                                </Columns>
                            </asp:GridView>
                        </div>
                    </div>
                    <asp:SqlDataSource ID="SqlDataSource4" runat="server"
ConnectionString="<%$ ConnectionStrings:Football_leagueConnectionString %>"
SelectCommand="SELECT match_id,match_date, home.club_name as home_name,
away.club_name as away_name, home_result, away_result, home_formation,
away_formation  FROM [dbo].[matches] INNER JOIN [dbo].[football_clubs] as home
ON home.club_id=[dbo].[matches].home_team_id INNER JOIN [dbo].[football_clubs]
as away ON away.club_id=[dbo].[matches].away_team_id WHERE season_id=@SeasonID
ORDER BY match_date">
                        <SelectParameters>
                            <asp:ControlParameter Name="SeasonID"
ControlID="DropDownList3" PropertyName="SelectedValue" Type="Int32" />
                        </SelectParameters>
                    </asp:SqlDataSource>

                </div>
            </div>
        </div>
    </asp:Content>
```