Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра комп`ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Автоматизація процесів збору та обробки футбольної статистики»

Виконав: студент 2 курсу, групи 1АКІТР-24м спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка

(шифр і назва спеціальності)

Самарський Олександр Олександрович

(ПІБ студента)

Керівник: д.т.н., професор кафедри КСУ Марія ЮХИМЧУК

(науковий ступінь, вчене звання / посада, ПІБ керівника)

«12» 12 2025 р.

Допущено до захисту
Завідувач кафедри КСУ
д.т.н., проф.В'ячеслав КОВТУН
(науковий ступінь, вчене звання)

«12» 12 2025 р.

Опонент: доцент кафедри АІІТ Володимир ГАРМАШ
(науковий ступінь, вчене звання / посада, ПІБ опонента)

«12» 12 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра комп'ютерних систем управління

Рівень вищої освіти II-ий (магістерський)

Галузь знань – 14 – Електроніка, автоматизація та електронні комунікації

Спеціальність –174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Освітньо-професійна програма – - Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри КСУ

д.т.н., проф. В'ячеслав КОВТУН

«26» 09 2025р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Самарському Олександру Олександровичу

(ПІБ автора повністю)

1. Тема роботи: Автоматизація процесів збору та обробки футбольної статистики.
   Керівник роботи: д.т.н., проф каф. КСУ ЮХИМЧУК М.С.
   Затвердженні наказом ВНТУ від « 24 » вересня 2025 року № 313.
2. Строк подання роботи студентом: до « 12 » грудня 2025 року.
3. Вихідні дані до роботи: Шевчук В. П., Костенко С. І. Основи програмної інженерії: навчальний посібник. — Київ : Ліра-К, 2020. — 320 с., Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. — New York : McGraw-Hill, 2019. — 1376 p
4. Зміст текстової частини: Вступ; Дослідження предметної області та існуючих методів; Математичне та алгоритмічне забезпечення системи прийняття торгових рішень; Розробка програмного забезпечення та тестування системи; Економічний розділ; Висновки; Список використаних джерел.
5. Перелік ілюстративного (або графічного) матеріалу: Основні етапи розробки веб-ресурсу; Архітектура автоматизованої системи; Структура взаємодії елементів управління бек-енду; Архітектура програмної системи; Алгоритм отримання даних від API; Алгоритм збереження та обробки

інформації, діаграма класів для взаємодії з базою даних, UML – діаграма класів для управління API, UML – діаграма класів моделей предметної області,

6. Консультанти розділів роботи

Розділ змістової частини роботи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Ольга РАТУШНЯК, к.т.н..доцент каф. ЕПтаВМ		

7. Дата видачі завдання: «25» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	25.09–05.10.2025	
2	Розробка математичного та алгоритмічного забезпечення	06.10 – 25.10.2025	
3	Розробка програмного забезпечення	26.10 – 10.11.2025	
4	Тестування розробленого програмного забезпечення	05.11 – 20.11.2025	
5	Підготовка економічної частини	до 01.12.2025	
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	20.11 – 03.12.2025	
7	Попередній захист роботи	до 03.12.2025	
8	Захист роботи	до 19.12.2025	

Студент Олександр САМАРСЬКИЙ

Керівник роботи Марія ЮХИМЧУК

АНОТАЦІЯ

УДК 621.374.415

Самарський О.О. Автоматизація процесів збору та обробки футбольної статистики. Магістерська кваліфікаційна робота зі спеціальності 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітня програма - Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2025. 93с.

На укр. мові. Бібліогр.: 49 назв; рис.: 43; табл. 12. .

Спорт і технології завжди розвивалися у тісному взаємозв'язку, сприяючи один одному як у підвищенні якості гри, так і у впровадженні інноваційних рішень. З кожним роком зростає обсяг даних, що збираються під час спортивних подій, а також можливості для створення цінної спортивної аналітики.

У магістерській роботі розглянуто процес розробки автоматизованої системи управління футбольною статистикою, яка забезпечує збір, обробку та структурування даних відповідно до визначених моделей, а також наочне відображення отриманої статистики.

Проведено теоретичний аналіз принципів спортивної аналітики, який став основою для визначення способів зберігання та візуалізації спортивних даних із застосуванням сучасних технологічних підходів.

Розроблена автоматизована система реалізована у вигляді веб-додатка, створеного з використанням сучасних Web-технологій. У роботі здійснено порівняння можливих технологічних рішень для розробки веб-додатків і створено дизайн, що відображає суть дослідження.

Також проведено тестування системи, оптимізацію її параметрів та аналіз ефективності функціонування розробленого програмного продукту.

Ключові слова: футбольна статистика, автоматизована система управління, веб-ресурс, спортивна аналітика, база даних, веб-розробка.

ABSTRACT

UDC 621.374.415

Samarskyi O.O. Automation of Processes for Collecting and Processing Football Statistics. Master's qualification thesis in specialty 174 – Automation, Computer-Integrated Technologies and Robotics, educational program Intelligent Computer Systems Vinnytsia: VNTU, 2025. 93 p.

In Ukrainian. Bibliography: 49 sources; figures: 43; tables: 12.

Sport and technology have always developed in close interconnection, mutually contributing both to the improvement of game quality and to the implementation of innovative solutions. Every year, the volume of data collected during sporting events increases, as do the opportunities for creating valuable sports analytics.

The master's thesis considers the process of developing an automated football statistics management system that provides data collection, processing, and structuring in accordance with defined models, as well as visual representation of the obtained statistics.

A theoretical analysis of the principles of sports analytics was carried out, forming the basis for determining methods of storage and visualization of sports data using modern technological approaches.

The developed automated system is implemented as a web application created using modern web technologies. The work includes a comparison of possible technological solutions for web application development and the creation of a design that reflects the essence of the research.

System testing, parameter optimization, and analysis of the operational efficiency of the developed software product were also conducted.

Keywords: football statistics, automated management system, web resource, sports analytics, database, web development.

ЗМІСТ

ВСТУП

Актуальність. Використання веб-технологій помітно зростає у сферах людського життя, зокрема це стосується сфери спортивних подій [1]. Кожну хвилину відбувається певна спортивна подія, зокрема футбол [2]. В сучасному світі у спортивного фаната не вистачає часу відслідковувати всі події, а пошук статистичних результатів витрачає багато часу. Важливим для сприйняття і швидкого пошуку футбольної статистики є веб-ресурси з зрозуміли інтерфейсом там коротким описом важливих подій матчу. Існує багато видів подання інформації, такі як: перегляд спортивних новин, відео-огляд у мережі інтернет, новини у спортивних журналах та інші. Кожен з них має свої переваги та недоліки, але їх загальний недолік в порівнянні з створюваним веб-ресурсом це надання статистичних результатів.

Таким чином питання розробки нового веб-ресурсу для перегляду футбольної статистики є актуальною на сьогодні. Рішенням цієї проблеми може отримання статистичних результатів, їх сортування та доступне представлення у вигляді веб-ресурсу.

При розробці автоматизованої системи управління футбольною статистикою найчастіше використовують REST API [3], оскільки це сервіс, який надає точні статистичні дані та можливість їх обробки для представлення на веб-сайті у зручній формі для користувача.

Зв'язок роботи з науковими програмами, планами, темами.

Дослідження, результати якого викладено у магістерській дисертації, виконувалося відповідно до плану наукових робіт кафедри автоматики та інформаційно-вимірювальної техніки Вінницького національного технічного університету.

Метою роботи є підвищення ефективності процесів збору, фільтрації та відображення спортивних подій шляхом створення автоматизованої системи управління футбольною статистикою.

Для досягнення поставленої мети необхідно було розв’язати такі завдання:

-здійснити класифікацію існуючих методів подання футбольної статистики та визначити їхні сильні й слабкі сторони;

-на основі аналізу відомих підходів запропонувати власний метод прийняття рішень і довести його ефективність;

-розробити архітектуру та дизайн веб-ресурсу;

-реалізувати запропоновану архітектуру у вигляді веб-додатка з можливістю доступу через Інтернет;

-провести тестування та оцінку роботи розробленої системи.

Об’єктом дослідження виступає процес збору, фільтрації та візуалізації футбольної статистики,

а предметом дослідження — безпосередньо футбольна статистика.

Методологічною основою роботи є використання методів спостереження, технічного аналізу, а також методів фільтрації та візуалізації даних, що здійснюються на основі аналізу показників, обраних користувачем.

Наукова новизна отриманих результатів полягає у розробленні нового підходу до збору, фільтрації та подання спортивних подій, який, на відміну від існуючих, забезпечує гнучке представлення різних видів інформації у зручній формі та підвищує ефективність пошуку статистичних даних.

Практичне значення наукових результатів полягає в тому, що на основі проведених теоретичних досліджень:

- розроблено програмні засоби (веб-ресурс) для перегляду статистичних даних;
- розроблено дизайн, який візуально покращує сприйняття інформації користувачем;

Особистий внесок здобувача в роботі, виконаній у співавторстві:

- програмна реалізація веб-ресурсу для відображення актуальної статистичної інформації.

РОЗДІЛ 1
ОГЛЯД СТАНУ ПРОБЛЕМИ УПРАВЛІННЯ ФУТБОЛЬНОЮ СТАТИСТИКОЮ

В розділі розкрито поняття веб-ресурсу, проведено аналіз предметної області, обґрунтовано вибір ресурсу для отримання статистики. Наведена класифікація існуючих методів відображення футбольної статистики, зазначено сфери використання описаних методів та обґрунтовано їх переваги та недоліки для подальшої розробки автоматизованої системи управління футбольною статистикою.

1.1 Сутність об’єкту дослідження

Футбол - найпопулярніша у світі гра з м’ячом за кількістю учасників та глядачів. Простий за своїми основними правилами та необхідним обладнанням, даним спортом можна займатися майже де завгодно, від офіційних футбольних майданчиків, вулиць, шкільних майданчиків, парків чи пляжів. Керівний орган футболу, Fédération Internationale de Football Association (FIFA), підрахував, що на початку XXI століття було приблизно 250 мільйонів футболістів і понад 1,3 мільярда людей, "зацікавлених" у футболі [4].

В 2010 році загальна телевізійна аудиторія понад 26 мільярдів спостерігала за головним футбольним турніром – чемпіонатом світу.

Футбол має довгу історію. В поточній формі з'явився в Англії в середині 19-го століття. Але альтернативні версії гри існували набагато раніше. Перші відомі приклади гри в м’яч відбувався в старих культурах неоліту понад 3,000 роки тому, на той момент часу м'яч був зроблений з каменю.

Матч розділений на два тайми по 45 хвилин. Після перших 45 хвилин гравці отримують 15-хвилинний період відпочинку, який називається перервою. За вирішенням судді (рефері), додається додатковий час до тайму в залежності від кількості затримок, травм гравців, кількості порушень.

Кожна команда складається з 11 гравців, одного воротаря та десяти гравців. Рекомендовані розміри поля: 105 метрів в довжину і 68 метрів в ширину.

Кожна половина поля повинна бути дзеркальним відображенням іншої за розмірами [4].

Для перемоги потрібно забити більше голів, ніж у суперника. Якщо кількість забитих м'ячів рівна в обох команд, і основний час гри закінчився, то гра закінчиться нічиєю, крім кубкових ігор, де гра може перейти до додаткового часу і навіть до пенальті, щоб визначити переможця. Гравці повинні використовувати ноги, щоб бити м'яч, і забороняється користуватися руками, крім воротарів, які можуть використовувати будь-яку частину свого тіла в межах 16,5 метрів від воріт в довжину та 40,2 метра в ширину.

Футбольна статистика – це елемент сучасного футболу, основне завдання якого вдосконалення тренувального процесу та контролю якості індивідуальної та командної гри [5].

Види футбольної статистики:

– статистика команди в турнірі;
– статистика матчу;
– індивідуальна статистика гравця;

татистичні показники команд у чемпіонаті зазвичай подаються у форматі турнірної таблиці відповідного змагання (табл. 1.1). У ній зазначається кількість проведених матчів (М), яка наприкінці туру має бути однаковою для всіх команд, окрім випадків форс-мажорних обставин. Також у таблиці відображається кількість перемог (В), нічиїх (Н) та поразок (П).

$$О = В * 3 + Н * 1 + П * 0, \quad (1.1)$$

де

О – загальна кількість очок,

В – кількість виграних матчів,

Н – кількість матчів, зіграних унічию,

П – кількість програних матчів.

Таблиця 1.1 – Турнірна таблиця УПЛ 20/21 [6]

№	Команда	І	В	Н	П	М	О
1	Ворскла	3	3	0	0	9-2	**9**
2	Динамо	3	2	1	0	7-2	**7**
3	Колос	3	2	0	1	7-4	**6**
4	Олімпік	3	2	0	1	7-5	**6**
5	Маріуполь	3	2	0	1	4-5	**6**
6	Шахтар	3	1	2	0	6-4	**5**
7	Десна	3	1	2	0	4-2	**5**
8	Олександрія	3	1	0	2	4-4	**3**
9	Минай	2	1	0	1	1-3	**3**
10	Інгулець	3	0	2	1	2-4	**2**
11	Зоря	3	0	1	2	3-6	**1**
12	Дніпро-1	3	0	1	2	3-6	**1**
13	Рух	3	0	1	2	4-8	**1**
14	Львів	2	0	0	2	1-7	**0**

В залежності від позиції в таблиці, команди отримують грошові винагороди, також мають змогу грати в євро-кубках. В Європі є два головних міжнародних клубних турніра це Ліга Чемпіонів та Ліга Європи.

В залежності від рейтингу УЄФА встановлюється кількість команд, які можуть брати участь у турнірах. Україна знаходить на 12 місці, участь в євро кубках беруть перших 4 команди. Зазвичай за перше місце в Українській прем'єр -лізі боряться дві команди: “Шахтар” та “Динамо”.

Статистика матчу несе в собі безліч статистичної інформації, а саме:

– загальна інформація про команди;

– події матчу (жовті та червоні карточки, голи, заміни, травми та інше)

Загальна інформація про команди та стартові склади команд можна представити таким чином (рисунок 1.2).

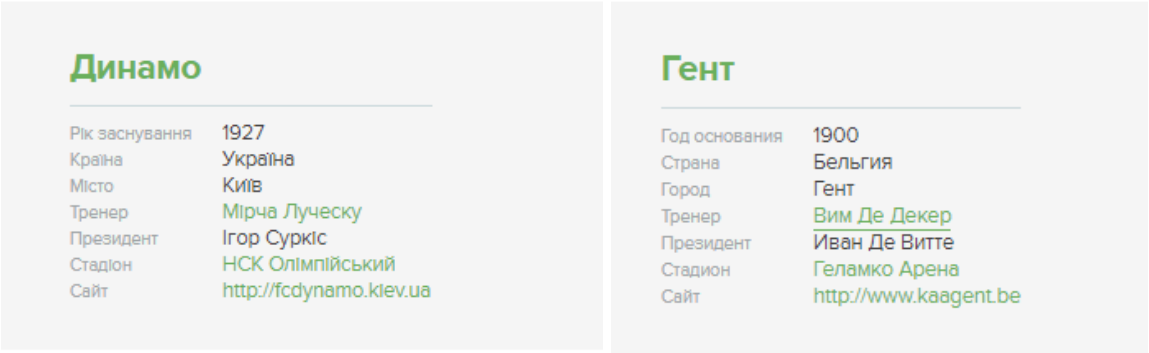


Рисунок 1.1 – Загальна інформація про команди

Стартові склади команд:

Динамо: Георгій Бущан (В)Віталій Миколенко (З) Томаш Кендзьора (З) Ілля Забарний (З) Сергій Сидорчук (П) Микола Шапаренко (П) Карлос Де Пена (П) Олександр Караваєв (П) Віталій Буяльський (П) Жерсон Родрігес (Н) Владислав Супряга (Н).

Гент: Даві Роф (В) Мішель Нгаду-Нгаджі (З) Алессіо Кастро-Монтес (З) Нуріу Фортуна (З) Ігор Пластун (З) Еліша Овусу (П) Вадіс Оджіджа-Офо (П) Роман Безус (П) Ніклас Дорш (П) Роман Яремчук (Н) Лоран Депуатр (Н).

Рисунок 1.1 відображає загальну інформацію про команду, а саме: Рік заснування, країну, місто, тренера, президента, місто та офіційний сайт команди.

Наступні рядки повідомляють про склад обох команд, причому порядку: воротар, захисники, півзахисники, нападники.

Інформація про матчі в цій формі зручна але складна для обробки, і коли справа доходить обробки, вона працює не добре. Тому доцільно бачити інформацію у статистичній формі. Також зберігання даних в такому вигляді спростить обробку: однією функцією запиту до бази даних можна отримати стартовий склад команд.

Події матчу зручно представити в виді інфо-графіки або в табличній формі (рисунок 1.2).

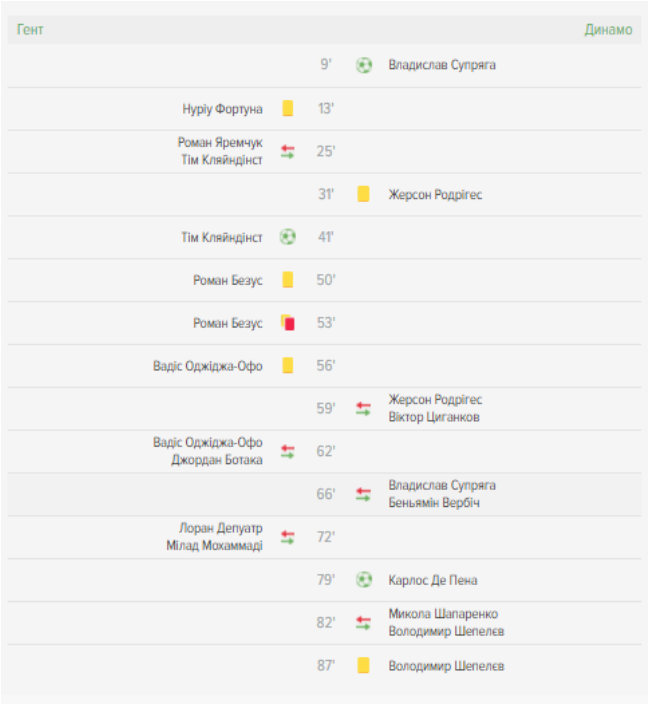


Рисунок 1.2 – Події матчу

Рисунок 1.2 відображає важливі події матчу, а саме: жовті/червоні картки, заміни, голи, травми.

Індивідуальна статистка гравця залежить від позиції на якій він грає. Для воротаря головний показник це – кількість відбитих м'ячів. Для захисника головним завданням є відбір м'яча у гравців суперника, та тактична побудова лінії захисту для перекриття дороги до воріт.

Півзахисник розташовується між захисниками та нападниками, головне завдання це підтримка захисників та нападників. Півзахисники поділяються на: опорних півзахисників та плеймеркерів. Головне завдання опорних півзахисників це перехоплювати м'яч у суперника, тому їх зазвичай оцінюють по кількості перехоплених м'ячів. Плеймекери задають тон гри, створюють гольові моменти для нападників.

Нападник розташовується найближче до воріт суперника, головне завдання це забити гол, або віддати гольовий пас.

На рисунку 1.3 наведена індивідуальна статистика найвідомішого українського гравця – Андрія Шевченко [7].

Клуб	Сезон	Чемпіонат		Кубок		Єврокубки		Інші		Всього	
		Ігри	Голи	Ігри	Голи	Ігри	Голи	Ігри	Голи	Ігри	Голи
Динамо-2 (перша ліга)	1992-93	6	0	–	–	–	–	–	–	6	0
	1993-94	31	12	1	0	–	–	–	–	32	12
	1994-95	13	4	4	5	–	–	–	–	17	9
	1996-97	1	0	–	–	–	–	–	–	1	0
Динамо	1994-95	17	1	4	1	2	1	–	–	23	3
	1995-96	31	16	5	1	2	2	–	–	38	19
	1996-97	20	6	–	–	–	–	–	–	20	6
	1997-98	23	19	8	8	10	6	–	–	41	33
	1998-99	26	18	4	5	14	10	–	–	44	33
Мілан	1999-00	32	24	4	4	6	1	1	0	43	29
	2000-01	34	24	3	1	14	9	–	–	51	34
	2001-02	29	14	3	0	6	3	–	–	38	17
	2002-03	24	5	4	1	11	4	–	–	39	10
	2003-04	32	24	1	0	10	5	2	0	45	29
	2004-05	29	17	–	–	10	6	1	3	40	26
	2005-06	28	19	–	–	12	9	–	–	40	28
Челсі	2006-07	30	4	6	3	10	3	5	4	51	14
	2007-08	17	5	1	0	5	1	2	2	25	8
Мілан	2008-09	18	0	1	1	7	1	–	–	26	2
Челсі	2009-10	1	0	–	–	–	–	–	–	1	0
Динамо	2009-10	21	7	2	0	6	1	–	–	29	8
	2010-11	18	10	2	1	12	5	–	–	32	16
	2011-12	16	6	1	0	5	0	–	–	22	6
Динамо-2		–	–	5	5	–	–	–	–	5	5
Динамо		172	83	26	16	51	25	–	–	249	124
Мілан		226	127	16	7	76	38	4	3	322	175
Челсі		48	9	7	3	15	4	7	6	77	22
Всього (на вищому рівні)		446	219	54	31	142	67	11	9	653	326

Рисунок 1.3 – Індивідуальна статистика Андрія Шевченко за кар'єру футболіста.

В першому стовбці вказується інформація про те в яких клубах грав даний футболіст, в другому відповідні сезони, важливою статистичною інформацією є кількість проведених матчів та забитих голів.

Таким чином, футбольна статистика забезпечує вдосконалення ігрового та тренувального процесу, а також його моніторинг розвитку на різному рівні.

Футбол має ґрунтуватися на Правилах, які гарантують «чесність» гри, адже саме чесність є ключовим елементом краси «гарної гри» та невід’ємною рисою її «духу». Найкращі матчі — це ті, де арбітр втручається мінімально, бо гравці

демонструють взаємну повагу, дотримуються Правил і шанобливо ставляться до офіційних осіб поєдинку [4].

Користуючись офіційним документом “IFAB Правила гри 20/21”, можна виділити 17 пунктів, які регламентують основні правила гри в футбол.

На основі визначених правил і характерних особливостей футболу було виділено ключові реальні об’єкти (сутності) предметної області, проведено їх абстрагування шляхом визначення набору властивостей, поведінкових характеристик та взаємозв’язків між ними.

Основними сутностями досліджуваної системи є:

турнір;

команда;

футболіст;

матч;

ігрова подія.

Кожна із сутностей має свої особливості та правила, які надалі використовуються при створенні бази даних.

Сутність «Турнір» — це чемпіонат, у межах якого проводиться певна кількість матчів. Турніри можуть бути національними, міжнародними або кубковими. Зазвичай вони відбуваються один раз на рік і утворюють сезон. Кількість команд-учасниць визначається регламентом змагань. Структура турніру може бути організована за двома основними принципами: груповим (коли кожна команда грає з усіма) або олімпійським (система вибування).

Сутність «Команда» містить основні відомості про футбольний клуб, який об’єднує групу гравців. Атрибути цієї сутності включають:

назву клубу, місто, президента, список гравців, тренера та офіційний вебсайт.

Команда може брати участь у кількох турнірах одночасно, якщо це не суперечить їхнім правилам. Наприклад, лише українські клуби можуть виступати в “Українській Прем’єр-лізі”.

Сутність «Матч» описує конкретну гру між двома командами в межах певного турніру (наприклад, Англійська Прем’єр-ліга [8], Ла Ліга [9], Ліга чемпіонів [10], Бундесліга [11]) або товариську зустріч. Гра проводиться на визначеному стадіоні, дата та час фіксуються заздалегідь, але можуть змінюватися за непередбачених обставин.

Кожен матч характеризується низкою статистичних показників, серед яких:

рахунок, кількість забитих м’ячів, картки, фоли, удари по воротах і у ствір, кутові, штрафні, відсоток володіння м’ячем та кількість сейвів.

Сутність «Футболіст» містить загальну інформацію про гравця: прізвище, ім’я, дату народження та ігрову позицію. Позиція може набувати лише чотирьох значень — воротар, захисник, півзахисник або нападник. Проте під час гри розташування гравця може змінюватися залежно від тактичної схеми команди, наприклад популярної 4-4-2 (чотири захисники, чотири півзахисники, два нападники).

Сутність «Ігрова подія» описує дії футболістів під час матчу, які впливають на його перебіг. Вона поєднує сутності «Футболіст» і «Матч». До таких подій належать заміни, удари по воротах, фоли, вилучення, голи тощо. Деякі події можуть повторюватися кілька разів (наприклад, удари чи порушення), а інші — відбуваються лише один раз (наприклад, вилучення з поля).

Після визначення основних сутностей предметної області було сформовано концептуальну модель, яка відображає взаємозв’язки між ними та слугує основою для проєктування бази даних.

Після того, як були вибрані основні сутності предметної області, на їх основі створюється реляційну схему бази даних [12]. Кожна сутність створює власну таблицю в базі даних, а властивості об’єктів, відповідно є атрибутами.

На рисунку 1.4 зображена узагальнена діаграми бази даних, яка демонструє загальну структуру предметної області.

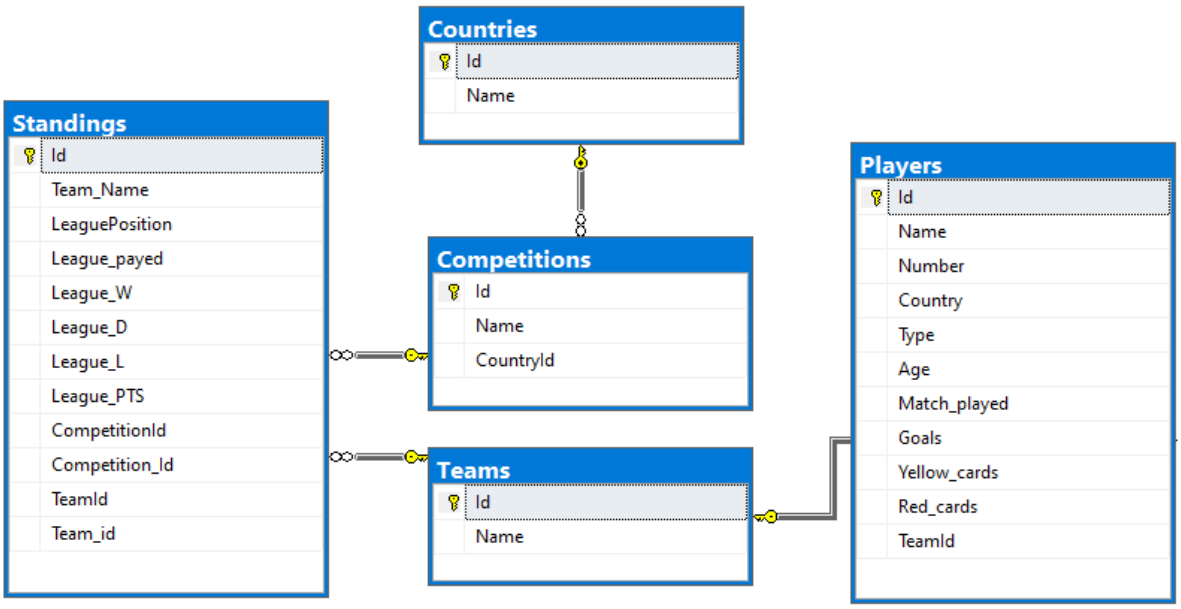


Рисунок 1.4 – Узагальнена діаграма бази даних

Кожен атрибут особливий та має певні обмеження: тип даних, первинний або зовнішній ключ. Назви атрибутів мають бути на англійській мові для полегшення читання коду.

Отже, в даному підрозділі було розглянуто особливості предметної області, виділено основні сутності та наведено узагальнену діаграму бази даних.

1.2 Аналіз етапів проектування веб-ресурсу

Веб-розробка стосується створення та обслуговування веб-сайтів. Вона включає такі аспекти, як веб-дизайн, веб-публікації, веб-програмування та управління базами даних [13].

Хоча терміни "веб-розробник" і "веб-дизайнер" часто використовуються синонімічно, вони не означають одне і те ж. Технічно веб-дизайнер розробляє лише інтерфейси веб-сайтів, використовуючи HTML [15] і CSS [16]. Веб-розробник може брати участь у розробці веб-сайту, але також може писати веб-сценарії такими мовами, як PHP [17] та JavaScript [18]. Крім того, веб-розробник може допомогти підтримувати та оновлювати базу даних, що використовується динамічним веб-сайтом [14].

Веб-розробка включає багато видів веб-вмісту. Деякі приклади включають ручне кодування веб-сторінок у текстовому редакторі, створення веб-сайту в такій програмі, як Notepad, та оновлення блогу через веб-сайт, що веде блог. В

останні роки системи управління вмістом, такі як WordPress, Drupal та Joomla, також стали популярними засобами веб-розробки. Ці інструменти полегшують кожному користувачу створювати та редагувати власний веб-сайт за допомогою веб-інтерфейсу [19].

Хоча існує багато методів створення веб-сайтів, часто існує компроміс між простотою та налаштуванням. Тому більшість великих підприємств не використовують системи управління вмістом, а натомість мають спеціальну команду веб-розробників, яка розробляє та підтримує веб-сайти компанії. Невеликі організації та приватні особи частіше обирають таке рішення, як WordPress, яке забезпечує базовий шаблон веб-сайту та спрощені інструменти редагування.

Але розробка веб-сайтів не така проста і не відбувається миттю. Повний процес веб-розробки забирає багато часу і є результатом багатогодинної праці бізнес-аналітиків, дизайнерів, розробників та тестувальників.

На рисунку 1.5 наведено основні етапи розробки веб-ресурсу.

Рисунок 1.5 – Основні етапи розробки веб-ресурсу

Найважливішим етапом у процесі веброзробки є розуміння потреб і очікувань клієнта, адже від цього залежить успіх усіх подальших кроків. Якщо початковий етап буде виконано неправильно, кінцевий продукт, швидше за все, не відповідатиме поставленим вимогам [20].

Перший етап розробки веб-додатка передбачає збір вимог. Визначення призначення ресурсу, його мети, цільової аудиторії та функціональних особливостей є ключовими завданнями цього етапу [21]. Основні вимоги включають:

Чітке формулювання мети — необхідно визначити, яку функцію виконуватиме вебресурс: рекламуватиме послугу, здійснюватиме продаж товарів чи надаватиме інформацію про діяльність організації.

Визначення кінцевих цілей — вони мають бути конкретними, вимірюваними та узгодженими з очікуваннями клієнта.

Орієнтація на цільову аудиторію — важливо розуміти, для кого створюється вебпродукт, щоб правильно вибудувати логіку інтерфейсу та користувацький досвід.

Другим етапом є дослідження та створення документа вимог. Після збору інформації всі вимоги формалізуються в документі, який слугує базовим орієнтиром для команди розробників. Такий документ повинен бути чітким, структурованим і зрозумілим. Він визначає функціональні й нефункціональні вимоги системи, а також дозволяє уникнути зайвих витрат часу та коштів у подальшому. Будь-які зміни до нього вносяться лише після додаткового погодження.

Третій етап — планування розробки. Коли вимоги узгоджені, формується план реалізації проєкту. На цьому етапі визначаються основні завдання, розподіляються ресурси, складається кошторис і графік робіт.

До етапу планування входить:

створення макета (wireframe) вебресурсу;

визначення архітектури сайту;

вибір технологічного стеку;

оцінка необхідних ресурсів.

Карта сайту (site map) є структурною схемою сторінок вебресурсу, що дозволяє команді чітко розуміти ієрархію контенту та взаємозв'язки між елементами. Вона забезпечує узгодженість дій дизайнерів і розробників.

Каркас (wireframe) — це візуалізація майбутнього інтерфейсу на структурному рівні. Він дозволяє виявити потенційні проблеми на ранніх етапах і визначити розташування основних елементів, навігації та компонентів. Наявність каркасу є обов'язковою умовою якісного проєктування, оскільки саме на ньому базується подальший дизайн.

Ретельне планування дозволяє ефективно використовувати ресурси, уникати перевитрат часу та забезпечити злагоджену роботу команди.

Завершальним етапом цієї фази є дизайн інтерфейсу. Хороший дизайн — це не лише естетика, а й зручність взаємодії користувача з вебдодатком. Після створення архітектури сайту дизайнери розробляють візуальні елементи: макет сторінок, кнопки, меню, навігацію, зображення, відео та інші графічні компоненти, які формують загальний вигляд і стиль вебресурсу.

Як вже обговорювалося раніше, дизайнерам потрібно пам'ятати про цільову аудиторію програми та розробляти веб-сайти відповідно до смаку та уподобань цільової аудиторії.

Наступний крок це розробка. На цьому етапі розробники починають розробляти функції та втілювати задумане в життя. Розробка програмного забезпечення складається з двох частин:

Фронт-енд розробка: її називають «розробкою на стороні клієнта». Це те, що бачать і з чим взаємодіють користувачі в браузері. Взаємодія з користувачем має першорядне значення [22].

Фронт-енд розробники веб-сайтів фактично не впливають на функціональність веб-сайту, але вони відповідають за втілення дизайну в життя. Перетворюють статичні елементи у повноцінно функціонуючі інтерактивні веб-сайти.

Бек-енд розробка: Частина веб-сайту, яку користувачі не бачать. Бекенд взаємодіє з інтерфейсом і надсилає інформацію від клієнта до сервера, щоб користувачі могли взаємодіяти з функціями веб-сайту [22].

Розробники програмного забезпечення бекенда відповідають за бізнес-логіку та сховище даних, створюють тести для того, щоб перевірити чи належним чином працює система. Створення та інтеграція бази даних, розробка та інтеграція API, перевірка безпеки тощо - все це частина розробки бек-енду.

Наступним етапом є розробка. На цьому рівні фахівці починають реалізовувати функціональні можливості та втілювати задумане в практичну форму. Процес створення програмного забезпечення поділяється на дві основні складові:

Фронт-енд розробка — або «розробка на стороні клієнта». Вона охоплює все, що бачить користувач у браузері, і з чим він безпосередньо взаємодіє. Саме якість цієї взаємодії визначає зручність користування [22].

Фронт-енд розробники безпосередньо не впливають на логіку роботи вебсайту, однак саме вони відповідають за реалізацію дизайну, перетворюючи статичні макети на повноцінні інтерактивні сторінки. Їхня робота робить сайт «живим» і зручним для користувачів.

Бек-енд розробка охоплює ту частину вебсайту, яка прихована від користувачів. Вона забезпечує взаємодію між інтерфейсом і сервером, обмін даними та виконання функцій вебресурсу [22].

Фахівці з бек-енду відповідають за бізнес-логіку, управління базами даних, створення та тестування системи. Вони займаються розробкою й інтеграцією API, забезпеченням безпеки, а також підтримують стабільну роботу всіх внутрішніх процесів.

Наступним етапом є тестування та розгортання. Будь-яке програмне забезпечення має пройти ретельну перевірку перед впровадженням у виробництво. Наявність чіткого процесу тестування є ключовою умовою якості.

Перед публікацією вебресурсу проводиться повне тестування для виявлення помилок, непрацюючих посилань і перевірки готовності програми до запуску.

До основних видів тестування належать: бета-тестування, тестування функціональності та продуктивності, а також методи «білого» і «чорного ящика». Ці підходи допомагають виявити недоліки, що можуть вплинути на стабільність та якість продукту [23].

Після завершення всіх перевірок код розміщується на основному сервері, де відбувається його компіляція та впровадження для кінцевих користувачів. Обов’язково також проводяться післярелізні перевірки.

Заключним етапом є технічне обслуговування. Процес веброзробки не завершується розгортанням, адже подальша підтримка програмного забезпечення відіграє важливу роль у його життєвому циклі [24].

Метою технічного обслуговування є оновлення, вдосконалення та модифікація програм для виправлення помилок, підвищення продуктивності та додавання нових функцій.

Будь-які зміни — від дрібних виправлень до масштабних оновлень — спрямовані на покращення ефективності та стабільності роботи системи.

Таким чином, процес веброзробки складається з кількох етапів, кожен з яких має вирішальне значення для успішного завершення проєкту.

Будь-які оновлення, модифікації, виправлення помилок, виправлення та розробка додаткових функцій з метою підвищення продуктивності існуючого програмного забезпечення включаються в технічне обслуговування програмного забезпечення.

Процес веб-розробки розділений на етапи, і кожен крок має вирішальне значення для успіху проекту.

1.3 Порівняльна харектеристика існуючих веб-ресурсів

На сайтах присвячених футбольним змаганням зазвичай на однійсторінці можливо подивитися результати у двох видах: загальнуінформацію поточного

туру (рисунок 1.2) та детальну інформацію однієїгри (рисунок 1.3) [25]. Кожен вид подання інформації корисний та має свої недоліки та переваги.

Проте для поставленої цілі, а саме: надання конкретної статистики, необхідно розробити веб-ресурс з фільтрами та чітким розподілом турнірних таблиць, подій в матчах, індивідульних досягнень гравців.

Потрібно надати можливість пошуку будь-якої статистики, для будь-якого чемпіонату у Світі, розробити систему, яка є обширною за кількістю чемпіонатів, та має змогу надавати статистику матчу в режимі онлайн.

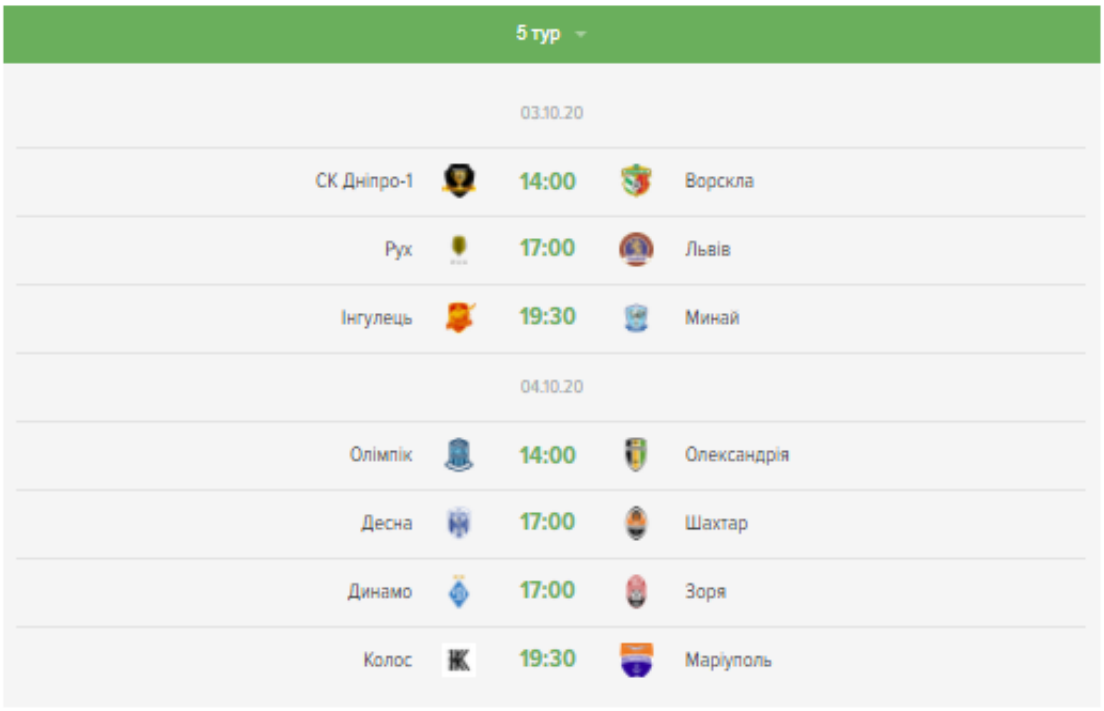


Рисунок 1.6 – Інформація туру

Більшість букмекерських компаній, мають власну базу даних, яка містить відомості про завершені матчі. Вона може бути поверхневою, тобто містити дані тільки про гру, та поглибленою – містити статистику про кожний елемент гри.

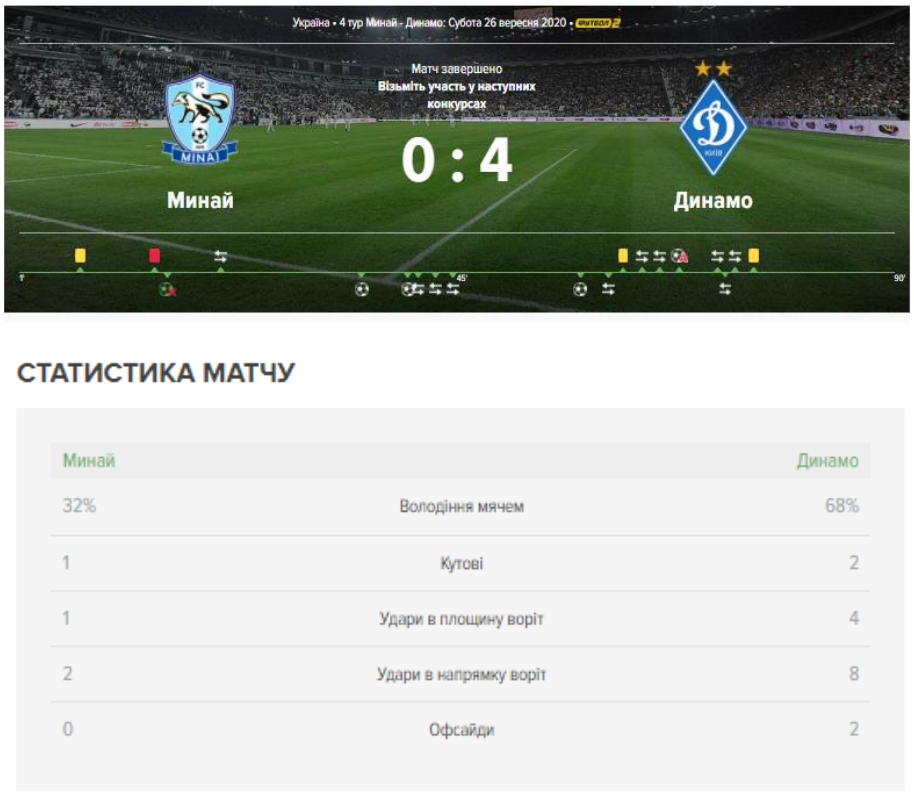


Рисунок 1.7 – Інформація матчу

Збір поглибленої статистики клопітливий процес, який потребую великої кількості ресурсів, як обчислювальних, так і людських. Деякі ресурси розміщують дані у вільний доступ, або створюють API для звернення до їхніх серверів. Серед найбільш відомих можна виділити наступні. В таблиці 1.2 наведені ресурси, які надають футбольну статистику в якості бази даних [22].

Таблиця 1.2 – Ресурси статистичної інформації [26, 27, 28, 29]

№	Назва ресрсу	Формат надання даних	Необхідність платної підписки	Об’єм статистичної інформації
1	football.db	TXT	Ні	Тільки результат гри
2	football-data.co.uk	TXT	Так	Відсутня інформація про події в матчі в режимі реального часу
3	api-football	API	Частково	Надання повної інформації
4	football-data.org	API	Так	Надання повної інформації

Найкращім варіантом для створення автоматизованої системи управління інформаційними ресурсами , які мають найбільше кількісних характеристик гри. Отже, отримання статистичних даних обрані дані з третього сервісу. Дані мають наступну структуру:

а) дата гри;
б) команда-господар (Н);
в) команда-гість (А);
г) кількість м’ячів забитих Н;
д) кількість м’ячів забитих А;
е) результат гри;
ж) удари по воротах Н;
з) удари по воротах А;
и) удари в стійку воріт Н;
к) удари в стійку воріт А;
л) кутові Н;
м) кутові А;
н) фоли Н;
о) фоли А;
п) жовті картки Н;
р) жовті картки А;
с) червоні картки Н;
т) червоні картки А.

Такої кількості параметрів повинно бути достатньо для того, щоб користувач мав можливість переглянути цікаву для нього інформацію.

Для того, щоб мати можливість скласти конкуренцію на даному ринку, необхідно докласти достатню кількість зусиль при розробці сайту: користувач повинен захотіти використовувати саме цей сервіс. Для досягнення такого ефекту необхідно, щоб кінцевий продукт не поступався, а краще перевершував існуючі альтернативи.

1.4 Підходи до вирішення проблеми

Автоматизована системи управління футбольною статистикою має виконувати два головних завдання:

1) отримання, обробка та сортування статистичної інформації;
2) відображення інформації для користувачів системи;

Виконання першого завдання має відбуватись автоматично та не вимагати втручання людини. Оскільки дана система має основною метою надання інформації користувачам, доцільно розробляти веб-сайт.

Для побудови веб-сайту потрібно розглянути можливі варіанти побудови системи. Для деяких випадків послуги дизайнера веб-сайтів та розробника програмного забезпечення, не матиме сенсу, оскільки рентабельність інвестицій зведе нанівець витрати. Для інших ситуацій створення власного веб-сайту буде більш економічно ефективною, і врешті-решт, це буде приносити дохід від реклами та маркетингу.

Виділимо два основних способи побудови веб-сайту.

WordPress або подібні CMS. CMS - це система управління вмістом, і незалежно від того, який тип обрано (WordPress широко визнаний найкращим, але існує багато інших), це надає простий спосіб побудови веб-сайту, не знаючи технічної інформації для створення веб-ресурсу. В даного методу є переваги та недоліки.

Adobe Dreamweaver це візуальна система дизайну, яка пропонує простий спосіб створення складних веб-сайтів, задопомогою HTML та CSS. Надає більш творчий контроль над дизайном та побудовою сайту, не використовуючи дорогих програм веб-дизайну та використання складних мов програмування.

До переваг відноситься:

– простота в управлінні;
– вибір різних варіантів вже побудованого дизайну;
– швидкість створення та впровадження у роботу.

Недоліки:

– не повний контроль;

– відсутність вибору особливого дизайну та внесення змін у “back-end”.

Враховуючи переваги та недоліки даний спосіб краще використовувати малому та середньому бізнесу, та системам які мають просту структуру.

Другий та найбільш поширений спосіб розробки веб-сайті це використання HTML / CSS та мов програмування, таких як PHP, Java, C#, Jquery, Python тощо.

Переваги:

– повний контроль над системою;

– повна свобода дій;

– можливість використання різних сучасних технологій;

Недоліки:

– відносно довга розробка та впровадження продукту;

– висока вартість розробки.

Даний метод являється класичним та забезпечує неперевершену функціональність та універсальність.

Оскільки, для розробки автоматизованої системи управління футбольною статистикою, необхідно взаємодіяти з API (REST) для отримання даних, використовувати базу даних для зберігання оброблених даних, розробляти алгоритми для відображення унікальної статистики необхідно використовувати другий метод.

1.5 Постановка задач розробки

Після аналізу питання і методів його вирішення було визначено, що автоматизована система управління футбольною статистикою – веб ресурс, який відображує відсортовану інформацію та унікальні статистичні показники команд та гравців. Для досягнення цієї цілі виділено наступні завдання, які необхідно виконати для розробки автоматизованої системи:

1) на основі аналізу переваг відомих підходів запропонувати власний підхід для обчислення унікальних даних;

2) розробити унікальний алгоритм (метод) на основі запропонованого підходу;

3) визначити найбільш ефективну мову та середовище розробки веб-сайту;

4) розробити структуру сайту, яка буде зрозуміла для користувача;

5) розробити дизайн сайту, який буде мати привабливий інтерфейс;

6) розробити базу даних для збереження інформації про користувача;

7) забезпечити можливість автентифікації та авторизації на сайті;

8) провести тестування програмного продукту.

РОЗДІЛ 2
РОЗРОБКА АРХІТЕКТУРИ ТА АЛГОРИТМІВ РОБОТИ ВЕБ-РЕСУРСУ

В розділі наводиться розробка архітектури та алгоритмів автоматизованої системи. На основі розробленої архітектури формується структура веб-ресурсу та дизайн головних сторінок. Також наведено розробку модуля авторизації та реєстрації.

2.1 Розробка архітектури автоматизованої системи

При проектуванні автоматизованої системи потрібно розробити її архітектуру та розподілити функції між компонентами даної системи. Оскільки в даній магістерській роботі було визначено, що автоматизована система буде реалізована у вигляді веб-ресурсу, архітектура має бути розроблена таким чином щоб враховувались особливості проектування веб-ресурсу.

На рисунку 2.1 наведено архітектуру автоматизованої системи управління футбольної статистикою.

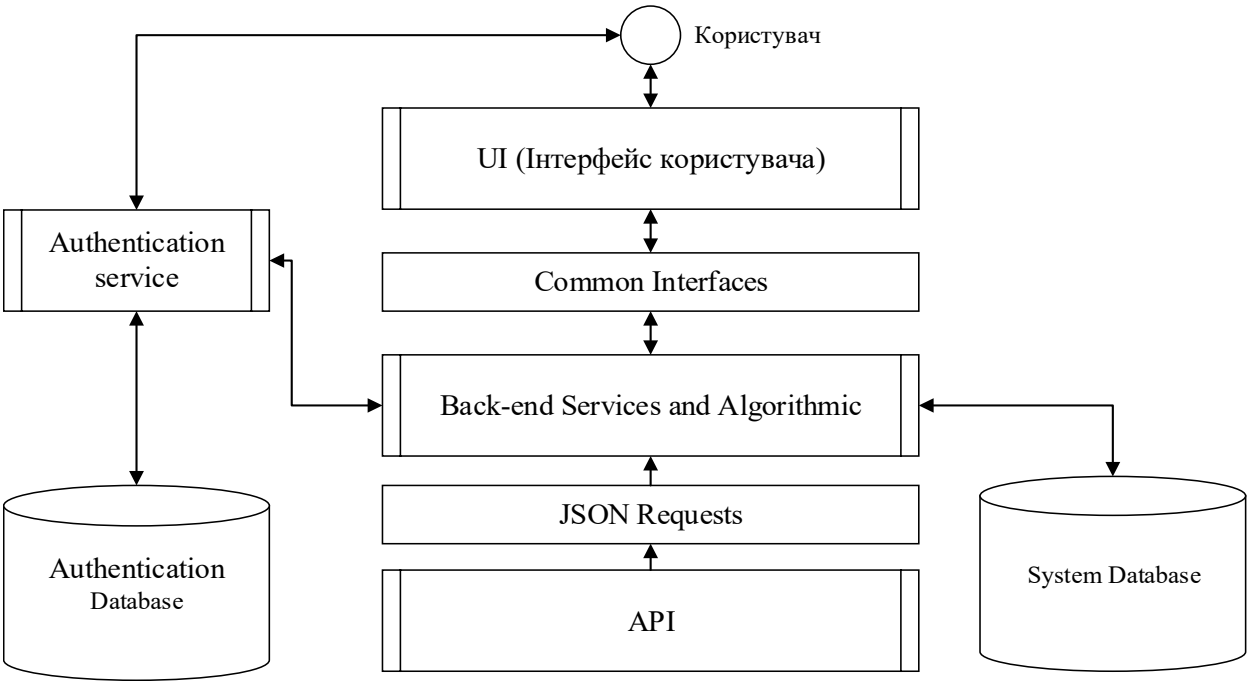


Рисунок 2.1 – Архітектура автоматизованої системи

Архітектура системи складається з чотирьох основних програмних компонентів:

- API;
- Back-end Services and Algorithmic;
- UI (Інтерфейс користувача);
- Authentication service.

Дана модель архітектури дозволяє чітко розподілити функції та ролі, які повинні виконувати кожний компонент системи.

Football API – це компонент системи, який дозволяє отримати великі об’єми статистичних даних та за допомою JSON передати дані для подальшої функціональної обробки.

Back-end Services and Algorithmic – основний функціональний модуль даної системи, який містить в собі логіку обробки отриманої статистичної інформації, сервіси для взаємодії з базою даних, формує загальну структуру веб-ресурсу та відповідає за виконання представленого для користувача функціоналу.

UI (Інтерфейс користувача) – відображає інформацію для користувача та відповідає за дизайн веб-ресурсу.

Authentication service – відповідає за процес перевірки особистості людини, а саме введення імені користувача та пароля під час входу на веб-сайт. Введення правильної інформації для входу дозволяє користуватись можливостями веб-сайту.

Розбиття системи на менші ієрархічні компоненти вирішує проблему розробки та допомагає визначити.

Детально розглянемо основні компоненти архітектури системи і засоби їх реалізації.

На рисунку 2.2 наведена структура основного компоненту системи – Бекенду.

Бек-енд містить усі модулі, що реалізують логіку системи за допомогою програмних рішень. Забезпечує проміжне програмне забезпечення для взаємодії з іншими компонентами системи:

- отримання та обробка даних від API;
- збереження та редагування інформації в базі даних;
- обробка статистичних запитів від користувача;
- виконання алгоритмів для відображення унікальної статистичної інформації;
- управління інтерфейсом користувача.

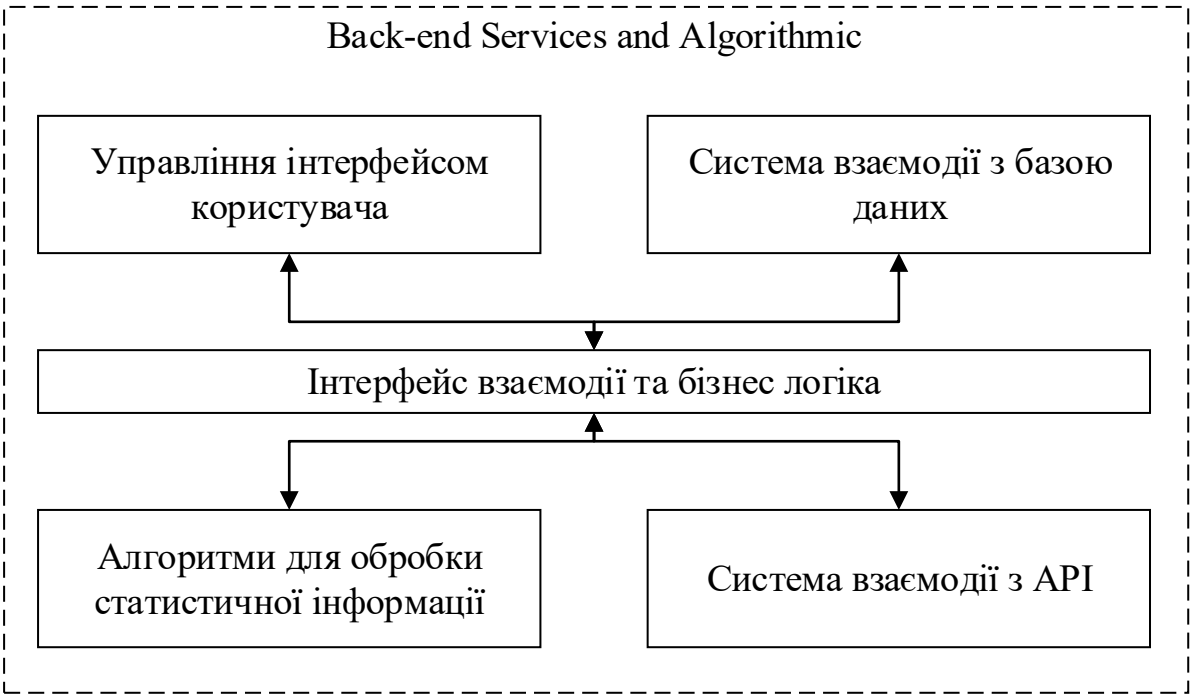


Рисунок 2.2 – Структура взаємодії елементів управління бек-енду.

Реалізація внутрішніх модулів базується на принципах проектування MVC та форматі JSON для обміну даними між системою та API. Алгоритми для обробки статистичної інформації відповідають за проведення аналізу статистики та обрахування відповідних коефіцієнтів.

2.2 Розробка структури веб-ресурсу

Структура сайту – описує те, з яких частин він складається і як ці частини розташовуються на сайті.

Від структури сайту залежить, чи зможе користувач швидко знайти цікавий йому матеріал, наскільки йому буде комфортно на сайті. Структуру сайту можна умовно розділити на зовнішню і внутрішню [30].

Сайти створюються для різних цілей і від цього багато в чому залежить яка у нього буде внутрішня структура. Тому для того щоб правильно створити внутрішню структуру сайту потрібно визначити його цілі, мету, аудиторію та забезпечити зручність користування сайтом.

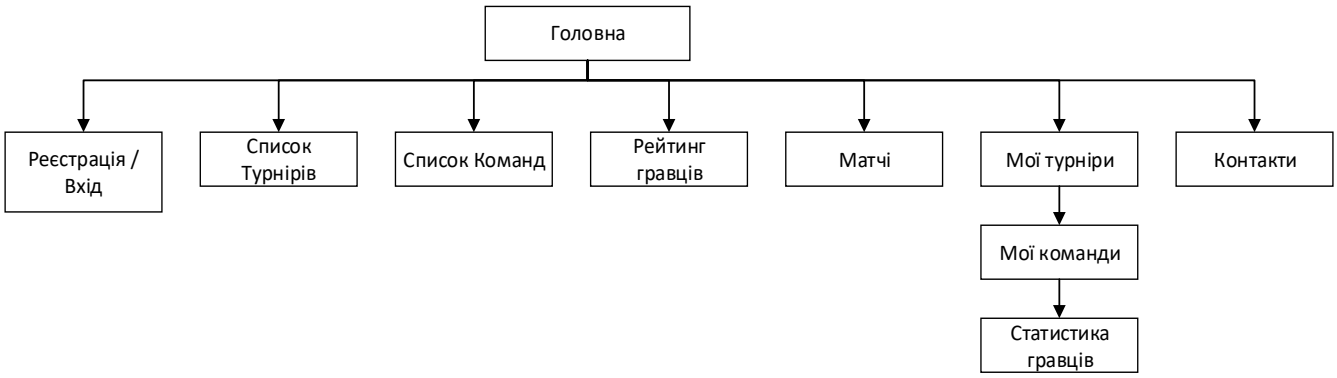


Рисунок 2.3 – Структура веб-ресурсу

Веб-сторінки виглядають досить різними одна від одної, але всі вони, як правило, мають схожі стандартні компоненти. До них відносяться:

- заголовок;
- панель навігації;
- робоча область або основний зміст;
- бічна панель;
- нижній колонтитул.

Головна сторінка є візиткою сайту, коли користувач відвідує дану сторінку він оцінює її вигляд та функціональність. Важливим елементами є меню та робоча область. Залежно від вибору елемента меню буде змінюватись робоча область.

Також при вході на сайт, коли будь-який елемент меню не вибраний відображаються спортивні новини. Більш детально ознайомитись з новинами можна перейшовши по силці, яка буде вказана при натисненні на новину.

Головна сторінка сайту в даному випадку називається «Головна» схематично вигляд якої показаний на рисунку 2.4.

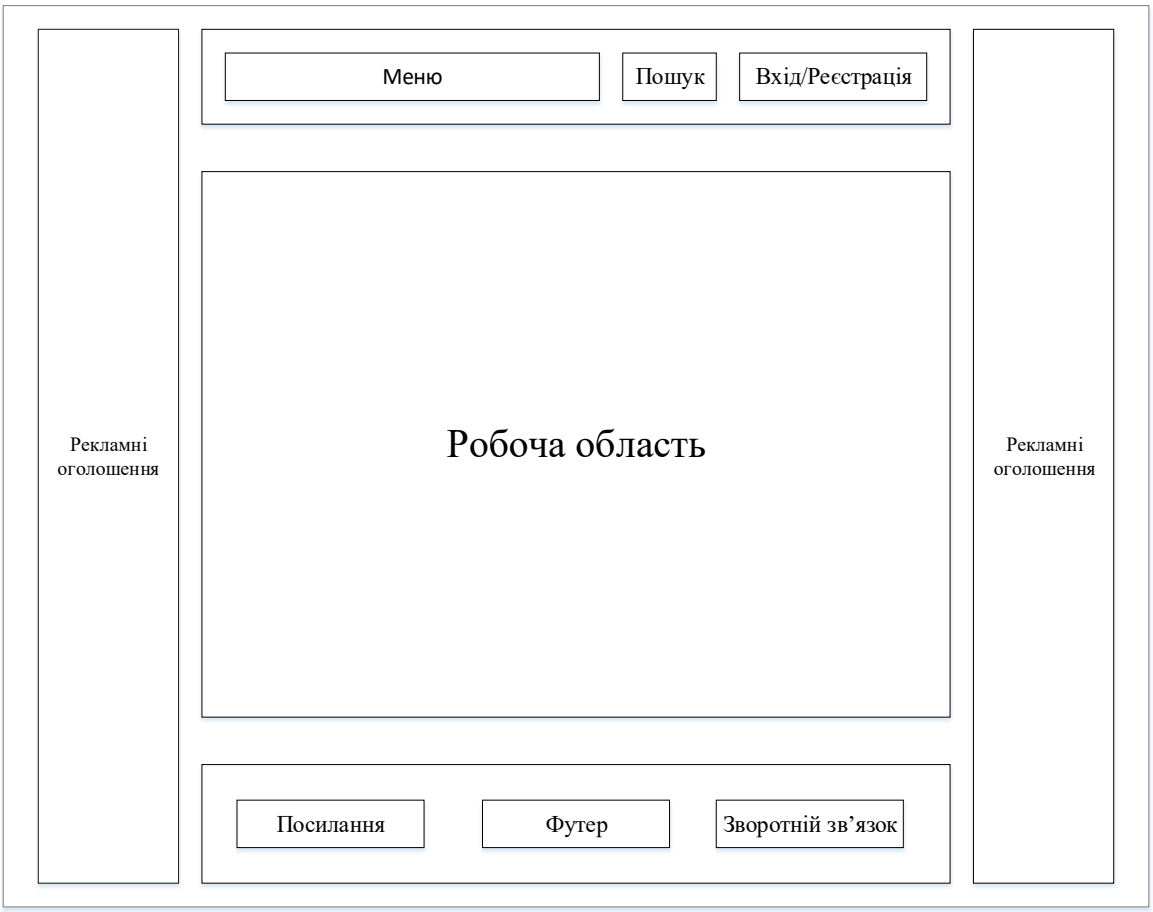


Рисунок 2.4 – Схематичне зображення головної сторінки

Зазвичай у верхній частині вебсайту розміщується головна смуга (header) — велика панель із заголовком, логотипом і, за потреби, слоганом. Цей елемент, як правило, залишається незмінним на всіх сторінках ресурсу.

Панель навігації (navigation bar) забезпечує перехід між основними розділами сайту. Вона також зберігає однакове розташування та вигляд на кожній сторінці, що сприяє зручності користувача та узгодженості інтерфейсу.

Деякі вебдизайнери розглядають панель навігації як частину заголовка, проте це не є обов'язковим правилом — у різних проєктах вона може існувати як окремий компонент.

Центральна частина сторінки, або контентна область, містить унікальний вміст, який відрізняється на кожній сторінці. Саме тут розміщуються відео, статті, новини, зображення та інші інформаційні блоки, що формують зміст вебресурсу.Велика робоча область в центрі, яка містить унікальний вмісту веб-сторінки, наприклад, відео або заголовки новин, і т. д. Це одна частина веб-сайту, яка, безумовно, буде відрізнятися від сторінки до сторінки.

Бічна панель відповідає за периферійну інформацію, посилання, цитати, оголошення тощо.

Нижній колонтитул (footer) зазвичай містить текст невеликого розміру — повідомлення про авторські права, контактні дані або додаткову інформацію. Як правило, ці відомості не є критично важливими для основного функціонування веб-сайту. Водночас, колонтитул може використовуватись і для SEO-оптимізації, надаючи посилання для швидкого переходу до популярних розділів ресурсу.

На рисунку 2.5 схематично подано вигляд сторінки після вибору одного з пунктів меню — наприклад, розділу «Турніри».

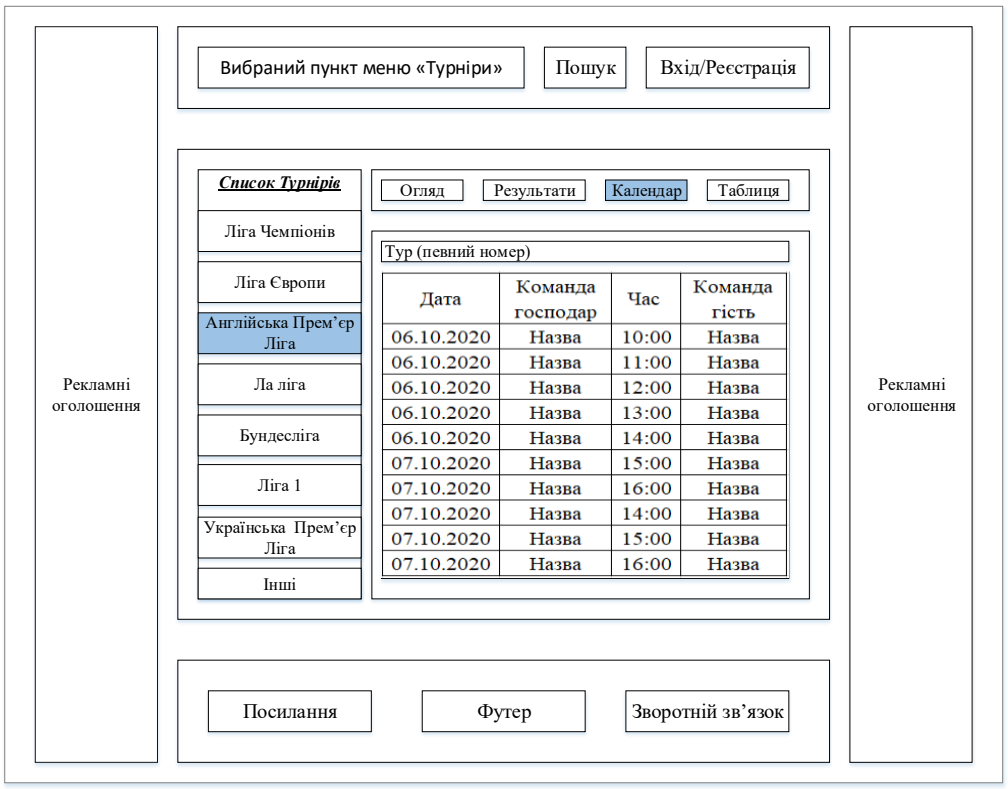


Рисунок 2.5 – Схематичне зображення сторінки при виборі пункту меню

Після того як користувач переходить до розділу «Турніри», центральна робоча область сторінки оновлюється. У наведеному прикладі відображається список доступних турнірів, серед яких на початку подано найпопулярніші. Припустимо, користувач обрав «Англійську Прем’єр-лігу», а в межах цього розділу — вкладку «Календар».

У календарі представлено перелік матчів із зазначенням дати, часу проведення, команди-господаря та команди-гостя.

Кількість матчів у турнірі залежить від кількості команд-учасників. Наприклад, в Англійській Прем’єр-лізі змагаються 20 команд, тому кожен тур включає 10 матчів. В Українській Прем’єр-лізі бере участь 14 команд, відповідно, у кожному турі проходить 7 ігор.

Завдяки вкладці «Календар» користувач може переглянути, з якими суперниками його улюблена команда зіграє найближчим часом.

Інформаційне наповнення сайту має ключове значення, оскільки саме воно формує цінність ресурсу для користувача. Якщо дизайн і структура сторінок приваблюють увагу відвідувача, то якісний і змістовний контент забезпечує його зацікавленість і довіру.

2.3 Розробка алгоритмів автоматизованої системи

Автоматизована система управління футбольною статистикою призначена для отримання, обробки, збереження та подання інформації про турніри, команди, матчі, події в матчах, гравців тощо.

Алгоритми системи можна умовно поділити на три основні типи:

алгоритми отримання статистичних даних через API;

алгоритми обробки та збереження отриманих даних;

алгоритми представлення інформації користувачеві.

Для доступу до необхідних даних потрібно створити алгоритм взаємодії з API. Під час формування запитів слід визначити, яку саме інформацію потрібно отримати, і проаналізувати структуру відповіді. Основна перевага API полягає в тому, що він надає доступ до великої бази даних із детальною документацією.

Обмін даними з API здійснюється за допомогою HTTP-запитів. Найпоширенішим методом є GET, що перекладається як «отримати». Саме цей метод найчастіше використовується для звернення до сервера. Найпростіший приклад GET-запиту — введення URL-адреси безпосередньо в адресний рядок браузера:

https://apiv2.apifootball.com/?action=get_countries&APIkey=xxxxxxxxxxxxx

Запит складається з двох частин:

1) рядок запиту (Request Line) – https://apiv2.apifootball.com/
2) заголовки (Message Headers) – action=get_countries
3) API ключ – xxxxxxxxxxx

Звертаємо увагу, що GET запит не має тіла повідомлення. Але, це не означає, що з його допомогою не можна передати серверу будь-яку інформацію. Це можна робити за допомогою спеціальних GET параметрів.

Щоб додати GET параметри до запиту, потрібно в кінці URL-адреси поставити знак «?» і після нього починати ставити їх за таким правилом:

Ім'я_параметра1=значення_пареметра1&ім'я_параметра2=значення_параметра2

API-ключ є персональним ідентифікатором адміністратора системи. Його наявність гарантує безпечний доступ до даних. Сервіс **APIfootball** не дозволяє надсилати запити користувачам, які не мають зареєстрованого облікового запису.

На рисунку 2.7 наведено список підписок, які пропонує даний сервіс.

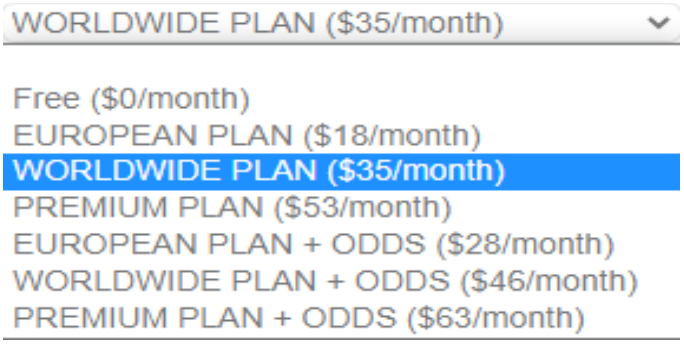


Рисунок 2.6 – Список підписок

Для того, щоб отримати ключ, необхідно зареєструватись, та вибрати одну із платних підписок.

Розглянемо детально опис сутностей FootballAPI. Сутності FootballAPI зручно буде відобразити в таблиці 2.1

Таблиця 2.1 – Сутності FootballAPI

Сутність	Параметри	Опис	Приклад запиту До кожного запиту потрібно додати(&APIkey=xxxxxxxxxxxxxx)
Countries	**get_countries**	Повертає список підтримуваних країн	https://apiv2.apifootball.com/?action=get_countries
Competitions	**get_leagues** country_id	Повертає список підтримуваних змагань	https://apiv2.apifootball.com/?action=get_leagues&country_id=41
Teams	**get_teams** team_id league_id	Повертає список доступних команд	https://apiv2.apifootball.com/?action=get_teams&league_id=148&
Players	**get_players** player_id player_name	Повертає доступних гравців	https://apiv2.apifootball.com/?action=get_players&player_name=ronaldocristiano&
Standings	get_standings league_id	Повертає турнірну таблицю ліг	https://apiv2.apifootball.com/?action=get_standings&league_id=148
Events (Results Fixtures)	get_events timezone from / to	Повертає події та результати	https://apiv2.apifootball.com/?action=get_events&from=2019-04-01&to=2019-04-03&league_id=148
Lineups	get_lineups match_id	Повертає склади однієї події	https://apiv2.apifootball.com/?action=get_lineups&match_id=24562
Statistics	**get_statistics** match_id	Повертає статистику однієї події	https://apiv2.apifootball.com/?action=get_statistics&match_id=24562
H2H	**get_H2H** timezone firstTeam secondTeam	Повертає останні ігри між командами, та останні ігри кожної команди	https://apiv2.apifootball.com/?action=get_H2H&firstTeam=Chelsea&secondTeam=Arsenal

Якщо виконати запит інформація буде отримана у форматі JSON. На рисунку 2.7 наведено JSON відповідь після запиту: https://apiv2.apifootball.com/?action=get_players&player_name=ronaldocristiano&APIkey=xxxx;


```
[
  {
    "player_key": 3183500916,
    "player_name": "Ronaldo Cristiano",
    "player_number": "7",
    "player_country": "Portugal",
    "player_type": "Forwards",
    "player_age": "34",
    "player_match_played": "31",
    "player_goals": "21",
    "player_yellow_cards": "3",
    "player_red_cards": "0",
    "team_name": "Juventus",
    "team_key": "4187"
  }
]
```


Рисунок 2.7 – JSON відповідь

Інформація подається у форматі (“key”: value). Однак для звичайного користувача вебресурсу такий вигляд є складним для сприйняття та неінтуїтивним. Тому необхідно обробити отримані дані й представити їх у зрозумілій і зручній для користувача формі.

Після аналізу можливостей і структур, що надає API, було створено алгоритм отримання статистичних даних (рисунок 2.7).

Розглянемо основні етапи роботи цього алгоритму:

1. Встановлення зв’язку з API. Це початковий етап алгоритму. Для встановлення з’єднання необхідно надіслати запит на сервер, у якому містяться дані, потрібні для авторизації або реєстрації.
2. Формування запиту. Якщо зв’язок успішно встановлено, формується запит залежно від обраної сутності (див. таблицю 1.2). До запиту додаються необхідні параметри. У випадку помилки підключення

відбувається перехід до блоку обробки помилок, який визначає тип і причину збою.

3. Відправлення запиту. Якщо запит сформовано коректно, його надсилають на сервер. Якщо ж під час формування виникла помилка — процес створення запиту повторюється з початку.

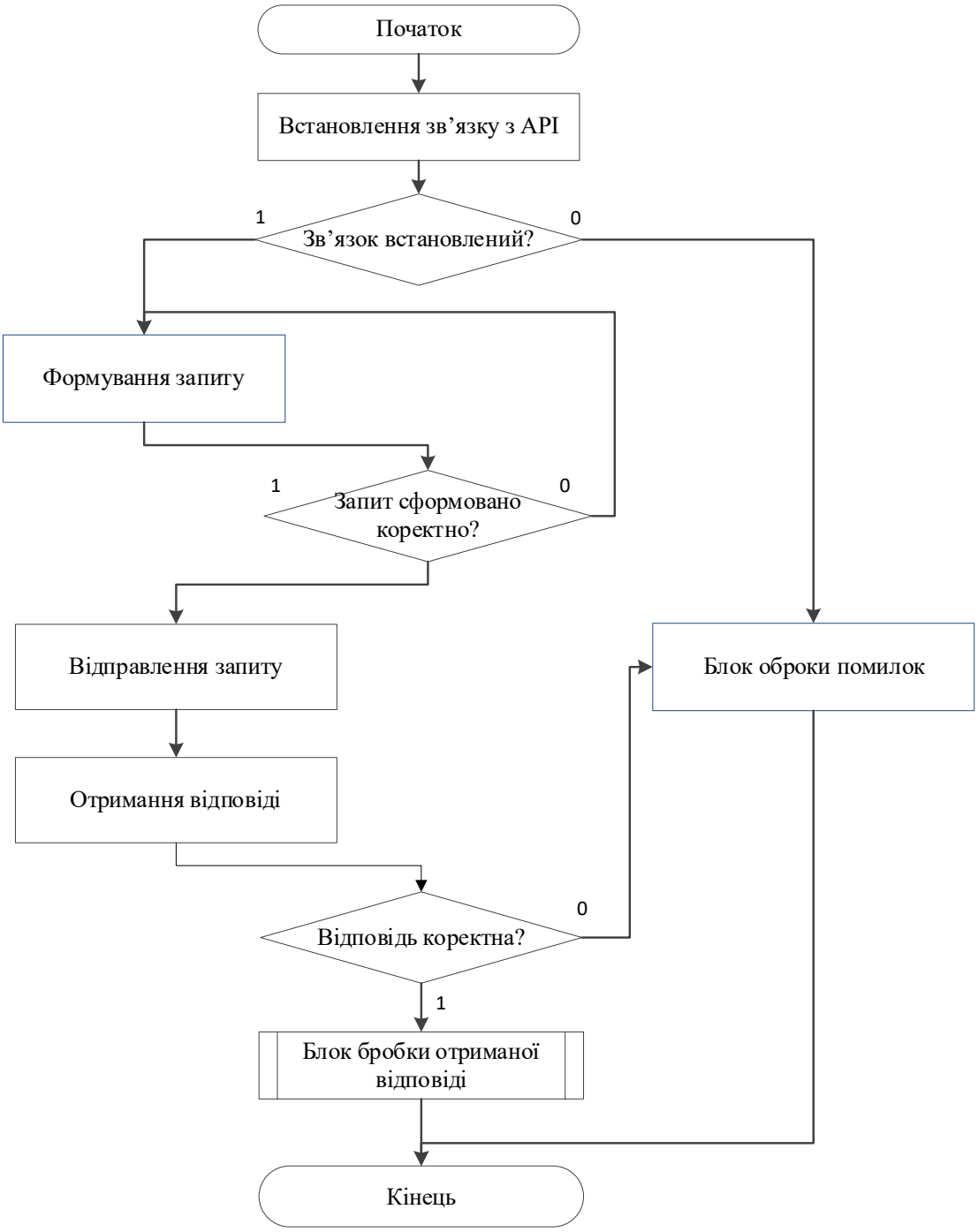


Рисунок 2.8 - Алгоритм отримання даних від API

4) Після відправлення запиту, API передає дані в форматі JSON, і частина алгоритму, яка відповідає за отримання відповіді, перевіряє її і якщо вона відповідає формату JSON то передає її в блок обробки отриманої відповіді, в іншому випадку, операції виконує блок обробки помилок.

Розглянемо більш детально алгоритм обробки та збереження даних. На рисунку 2.8 наведений універсальний алгоритм обробки інформації отриманої від API. Для кожної сутності він має певні зміни у зв'язку з тим, що отримується різна по структурі інформація.

Блок обробки отриманої відповіді (БООВ) змінює формат даних, та в залежності від переданої інформації, формує об'єкт або список об'єктів з певними параметрами, з якими в подальшому можна взаємодіяти. Сформовані об'єкти зручні для визначення різною цікавої статистичної інформації.

В залежності від вибору інформації, яку бажає побачити користувач, алгоритм виконує функції збереження даних в базу даних системи, або відображає інформацію на сайті.

Приклад: Користувач бажає переглянути турнірну таблицю АПЛ, та натискає на відповідну кнопку на сайті, система розпізнає дану подію, аналізує яка інформація для цього необхідна та робить запит до API, після отримання інформації, алгоритм перетворює JSON Result в об'єкт.

Турнірна таблиця АПЛ, як було наведено на початку розділу являється сутністю API, тому її можна відобразити на сайті без збереження в базу даних, та без використання алгоритмів обрахування унікальної статистики.

Наступний приклад: Користувач хоче дізнатись команду в турнірі, яка забила найбільшу кількість голів. Для обрахування цієї інформації необхідно зробити декілька запитів до API, обробити та зберегти дані в базу даних. Далі використовуючи необхідний алгоритм визначити команду, та показати дані користувачу.

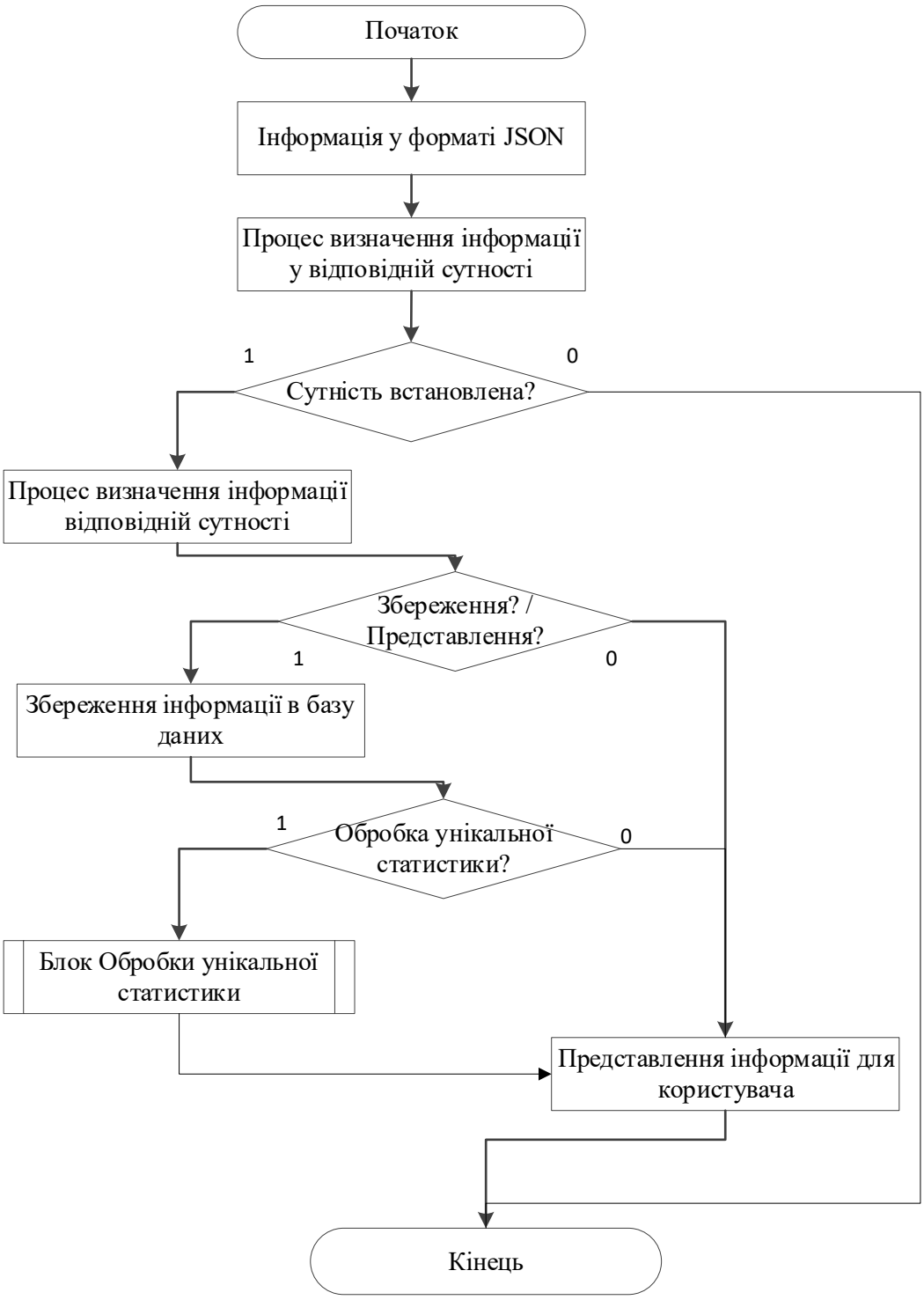


Рисунок 2.9 – Алгоритм збереження та обробки інформації

Розглянемо алгоритми, які розширюють базову функціональність автоматизованої системи. Окрім, представлення звичайної статистики, для підвищення конкуренто спроможності системи розроблено алгоритми, які роблять дану систему особливою.

Перший алгоритм це визначення команди сезону в склад, якої входять 11 найкращих гравців. Для кожної позиції визначено статистичні критерії по яким, гравець з найвищою оцінку займає місце в команді.

Спочатку потрібно визначити схему по якій буде побудована команда. Враховуючи сучасну тенденцію по побудові позиційної гри, та команд, які грають в домінуючий футбол, вибрана схема 4-3-3. Тобто 4 – захисника, 3 – півзахисника та 3 нападаючих.

Для нападаючих головним показником є кількість забитих голів, також важливим є кількість гольових передач. Деяким нападникам достатньо одного точного удару для одного голу, а деяким більше десяти, тому також буде враховано співвідношення ударів і голів. Позначимо коефіцієнт успішності нападаючого, як КН.

$$КН = Г * 0.6 + П * 0.4 - (\frac{Г}{У} * 0.1) \quad (2.1)$$

де КН коефіцієнт нападаючого;

Г – кількість забитих голів за сезон;

П – кількість гольових передач.

Як було сказано в попередньому розділі, півзахисник може виконувати роль

“плеймейкера” (ПП), або опорного-півзахисника (ОП). В даній схемі в якій 3 півзахисника, буде два плеймейкера та один опорний-півзахисник.

Основне завдання ПП це ведення гри, створення гольових моментів для нападаючих. Також враховуються кількість ключових передач за гру. Ключова передача це пас партнеру в гольове положення (вихід один на один, положення перед порожніми воротами і так далі).

Позначемо коефіцієнт успішності плеймейкера, як КПП.

$$КПП = Г * 0.2 + П * 0.5 + \frac{КП}{І} * 0{,}3 \quad (2.2)$$

де КПП – коефіцієнт півзахисника плеймейкера;

Г – кількість забитих голів за сезон;

П – кількість гольових передач;

КП – кількість ключових передач;

Основне завдання ОП це відстежувати та контролювати переміщення гравців суперника, відбирати м’яч у противника в центральній зоні, та віддавати точні передачі партнерам в команді.

Позначимо коефіцієнт успішності опорного-півзахисника, як КОП.

$$КОП = \frac{В}{І} - \frac{Ф}{І} + ТП * 0{,}0015 \quad (2.3)$$

де КОП – коефіцієнт опорного-півзахисника;

В – кількість відборів за сезон;

Ф – кількість порушень;

ТП – відсоток точних передач;

І – кількість ігор за сезон.

Основне завдання захисників – не допустити щоб, команда суперник забила гол. Захисники поділяються на центральних та флангових. Зазвичай у схемі 4:3:3 два центральних захисника і два флангових захисника. Для центрального захисника характерні такі вміння, як гра головою, стрибки і підкати. Крайнього захисника можна охарактеризувати як універсального солдата, він виконує функції оборони і атаки.

Позначимо коефіцієнт успішності центрального захисника, як КЦЗ.

$$КЦЗ = \frac{І}{П} + Є * 0{,}01 \quad (2.4)$$

де КОП – коефіцієнт опорного-півзахисника;

П – кількість пропущених голів команди;

Є – Відсоток успішних єдиноборств;

І – кількість ігор за сезон.

Позначимо коефіцієнт успішності для флангового захисника, як КФЗ.

$$КФЗ = \frac{І}{П} + П * 0{,}2 \quad (2.5)$$

де КФЗ – коефіцієнт флангового захисника;

П – кількість пропущених голів команди;

П – кількість гольових передач;

І – кількість ігор за сезон.

Головне завдання воротаря – не дати супернику забити гол у свої ворота. Важливою характеристикою для воротаря є співвідношення кількості ударів в ствір воріт, та кількість пропущених голів. Відбитий удар голкіпером називають сейвом. Але команда гравців суперників за одну гру може не завдати ні одного удару в ствір воріт. Тому вводитися поняття, як матч зіграний на нуль.

Позначимо коефіцієнт успішності для голкіпера, як КВ.

$$КВ = \frac{І}{П} + \frac{Н}{І} + \frac{П}{С} \tag{2.6}$$

де КВ – коефіцієнт успішності голкіпера;

П – кількість пропущених голів;

Н – кількість зіграних матчів на нуль ;

С – кількість сейвів;

І – кількість ігор за сезон.

Визначивши коефіцієнти для 11 гравців сезону, сформуємо алгоритм для визначення унікальної статистики.

На рисунку 2.10 зображено алгоритм для формування команди сезону.

2.4 Розробка модуля авторизації та реєстрації

Авторизація — це механізм безпеки, який визначає рівні доступу або права користувача (клієнта) до системних ресурсів, таких як файли, служби, програми, дані та функції додатків. Вона є процесом надання або обмеження доступу до мережевих ресурсів і забезпечує можливість користувачеві працювати лише з тими об’єктами, до яких йому дозволено доступ.

Більшість систем веб-безпеки ґрунтуються на двоетапному процесі. Першим етапом є автентифікація, під час якої відбувається ідентифікація користувача. Другий етап — авторизація, що визначає, до яких саме ресурсів користувач має право доступу після успішної автентифікації.

Сучасні операційні системи значною мірою покладаються на ефективні механізми авторизації, які спрощують розгортання та адміністрування програм. Основними факторами, що впливають на процес авторизації, є тип користувача, його облікові дані, а також ролі й дозволені дії, які перевіряються перед наданням доступу.

Контроль доступу у комп’ютерних системах і мережах базується на політиці безпеки та складається з двох етапів:

1. Етап визначення політики — коли встановлюється, до яких ресурсів може бути надано доступ.
2. Етап застосування політики — коли відбувається перевірка запитів і приймається рішення про дозвіл або заборону доступу.

Таким чином, авторизація належить до фази визначення політики доступу, що передує її практичному застосуванню, коли система ухвалює рішення про дозвіл або відмову в доступі на основі попередньо визначених правил.

Крім того, контроль доступу використовує автентифікацію для підтвердження особи користувача. Якщо користувач намагається отримати доступ до певного ресурсу, система перевіряє, чи має він відповідні повноваження. Механізми авторизації зазвичай реалізуються на сервері безпеки, який може контролювати доступ як до окремих файлів, так і до програмних модулів.

Реєстрація – внесення даних користувача в базу даних, а саме логін та пароль, з метою можливості отримати доступ до функціональності сайту.

Переважна більшість веб-сайтів інтернету вимагає реєстрації учасників веб-сайту. В даному випадку зареєстровані користувачі мають можливість отримати доступ до формування унікальної статистики.

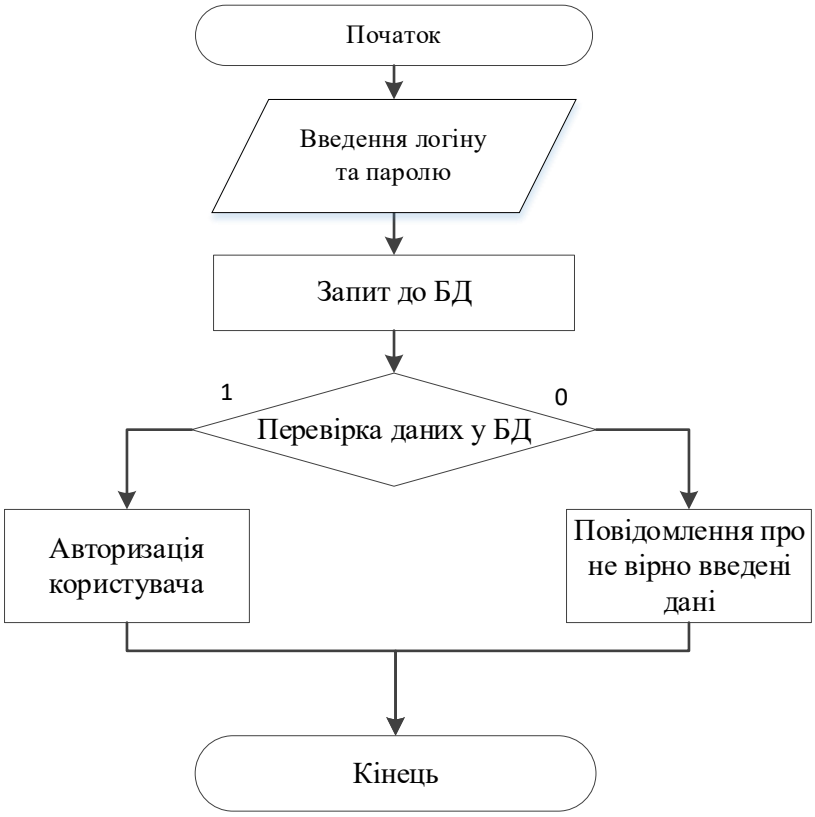


Рисунок 2.11 – Алгоритм авторизації

2.5 Вибір кольорової гами

Перш ніж перейти до процесу вибору кольорової гами сайту, важливо зрозуміти, чому саме кольорова схема веб-сайту так важлива. Коли користувач заходить на сайт, першу увагу він звертає на дизайн сайту. Необхідно, щоб вибрані кольори відповідали контенту та його аудиторії.

Кольори впливають на те як відвідувач сприймає інформацію, настільки ж, як макет сайту, якщо все зроблено добре, вони можуть мати позитивний вплив на оцінку сайту.

Колірний круг Іоханнеса Іттена [31] найбільш вдалі поєднання кольорів, їх ще називають колірними схемами, підбирають для того, щоб зробити дизайн гармонійним. Є чотири основних способи поєднувати кольори, їх вибирають в залежності від мети і кількості потрібних відтінків.

1) монохроматична – це поєднання трьох відтінків одного базового кольору. Наприклад, рожевий, світло-рожевий і ще більш світло-рожевий. Тобто один колір різного ступеня насиченості, без домішки інших відтінків. Це консервативний і простий варіант.

Безпечне поєднання, за допомогою якого можна створити гармонійний дизайн. Проте при використанні даної схеми, користувачі сайту можуть прийняти її як занадто нудною, також важливі елементи не будуть виділеними. Вирішенням даної проблеми може бути додання контрасту за допомогою чорних і білих елементів.

2) аналогова – це комбінація від двох до шести кольорів, які знаходяться один за одним на колірному колі. Така комбінація не може складатися з одного кольору в різних відтінках, це обов'язково кілька різних, але близьких кольорів.

Подібні комбінації виглядають особливо і яскравіше монохроматичних, чудовий вибір для оформлення сайтів. Щоб зберегти гармонію, слід вибрати один домінуючий, а решту два використовувати в меншій кількості, для акцентів.

3) комплементарна комбінація – це поєднання з двох максимально контрастних кольорів, які підкреслюють один одного. Комплементарні пари утворюються з двох строго протилежних кольорів.

Такі поєднання виглядають яскраво, вони привертають увагу в першу чергу, тому їх краще використовувати, щоб виділити щось важливе. Наведену комбінацію краще використовувати рівномірно, через високу контрастність вони можуть втомити користувача.

4) тріадне поєднання кольорів – це три рівновіддалених один від одного кольори на колірному колі, які створюють контрастні поєднання, але не такі яскраві, як компліментарні. Це ті кольори, які можна з'єднати рівностороннім трикутником.

Дана схема буде доречною для сміливого дизайну і залучення уваги, але вони агресивні. У цієї схеми найбільша кількість можливих комбінацій, тому такі поєднання є чудовим варіантом для фірмової кольорової палітри.

Використовуючи коло Іттена та обрану колірну схему, отримуємо додаткові
кольори для сайту (рисунок 2.11).

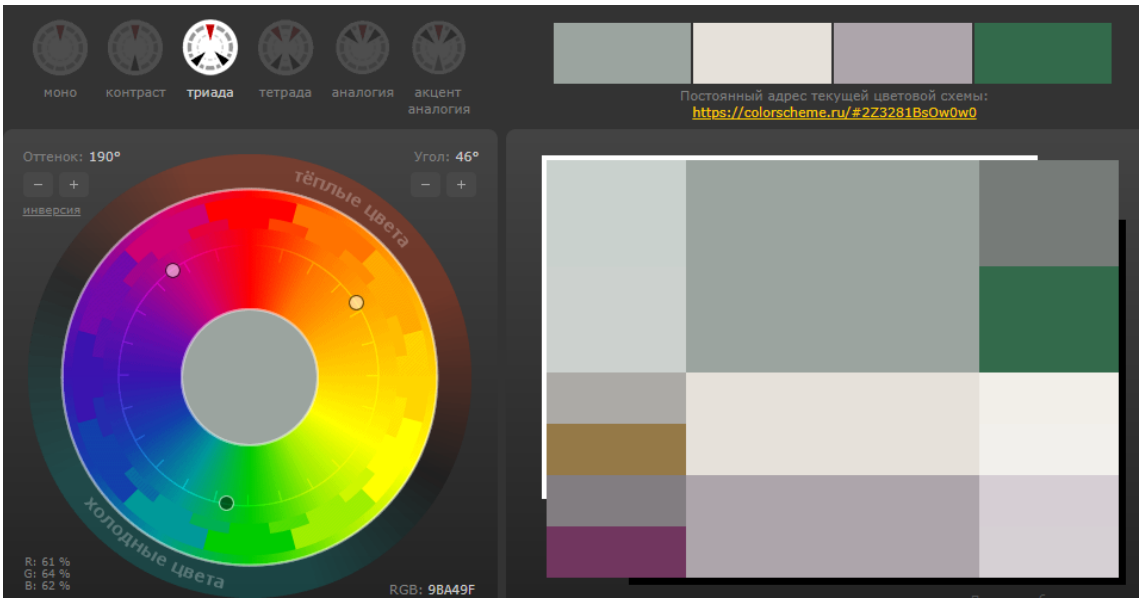


Рисунок 2.12 – Вибір кольорової гами

Для створення кольорової схеми сайту необхідно обрати три кольори. Основним кольором обрано сірий колір. На його фоні інші кольори виглядають більш глибокими і насиченими. Щоб відмітити структурні елементи сайту необхідно застосувати тріадне поєднання кольорів.

РОЗДІЛ 3
РОЗРОБКА ВЕБ-РЕСУРСУ

В розділі проведено аналіз та порівняння сучасних веб-технологій для розробки веб-ресурсів. Обґрунтовано вибір платформа та технологій для розробки та тестування веб-додатку. Подано огляд розробленої системи, зазначено функції, які система виконує.

3.1 Аналіз сучасних веб-технологій для розробки веб-ресурсів

HTML є абревіатурою від мови розмітки HyperText. Мова розмітки - це комп'ютерна мова, яка використовується для визначення структури та відображення простого тексту на веб-сторінці. Мова розмітки використовує теги та елементи навколо простого тексту, щоб зробити його зрозумілим на комп'ютері [32].

Отже, HTML використовується для створення веб-сторінок, і він логічно структурує вміст веб-сторінок. HTML робить веб-сторінки читабельними, і це дозволяє браузеру правильно відображати веб-сторінки.

HTML визначає структуру і призначення веб-сторінки, а також дозволяє прикріплювати носії, вставляти таблиці, посилання, списки, форми та всі інші речі, які веб-сторінка має за допомогою тегів HTML, які також називаються елементами.

Остання версія HTML – це 5-та версія, яка називається HTML5, і вона є основною та фундаментальною мовою розмітки технологій в Інтернеті, яка використовується для структурування та відображення вмісту для всесвітньої мережі. HTML5 замінив багато старих функцій HTML для просування стандартизованого набору практик, що використовуються веб-розробниками.

Розглянемо переваги HTML5.

1) Краща доступність: HTML5 покращує доступність веб-сайтів. Він сприяє використанню семантичної розмітки шляхом введення нових семантичних елементів, таких як елемент <header>, які можна використовувати для цілої сторінки та конкретного вмісту, доступного на сторінці. Деякі інші нові

семантичні елементи включають <nav>, <aside>, <footer>, <section>, <article>, <main>, <address> і <time>. Ці елементи можуть допомогти пошуковим системам та користувачам легко знайти потрібну веб-сторінкую

2) Чистіший код: HTML5 відокремлює дизайн від вмісту без жодних клопотів. Це дозволяє писати код чисто за допомогою змістовної семантики .

3) Зручний та адаптивний дизайн для мобільних пристроїв: більшість людей у наш час використовують мобільні телефони та планшети для перегляду веб-сторінок замість персональних комп'ютерів чи ноутбуків, оскільки мобільні телефони прості у використанні. HTML5 дозволяє використовувати адаптивний дизайн та розробляти веб-додатки, сумісні з широким спектром мобільних пристроїв.

4) Сумісний з багатьма браузерами: Поряд із тим, що HTML5 зручний для мобільних пристроїв, HTML5 дозволяє веб-сайтам бути сумісними з багатьма браузерами, такими як Chrome, Firefox, Safari та Opera, і легко перевірити, які функції HTML5 підтримує кожен браузер.

5) Підтримує медіа. До HTML5 було тяжко додавати аудіо та відео на веб-сторінки, оскільки для цього потрібні сторонні плагіни або використання Flash. Цей недолік попередніх версій обмежив обсяг веб-додатків. Але тепер за допомогою HTML5 легко додавати аудіо та відеоелементи у веб-додаток.

6) Вміст, орієнтований на користувача: HTML5 дозволяє знати місцезнаходження користувача за допомогою API геолокації. API вимагає згоди користувача, перш ніж дізнатися про своє місцезнаходження. Ця функція дозволяє створювати вміст для користувачів, орієнтований на їх потреби. Це також може бути корисним для вибору мови залежно від місцезнаходження користувача.

7. Покращене зберігання: Майже всі веб-сайти використовують файли cookie для відстеження даних користувачів. Але файли cookie з'їдають місце для зберігання, що також впливає на час відгуку веб-програми. HTML5 пропонує два

типи сховища, тобто сесійне та локальне сховище замість файлів cookie. Це дозволяє тимчасово зберігати дані.

8. Офлайн-перегляд: HTML5 дозволяє розробнику вказати файли, які браузер повинен кешувати.

C# — це сучасна об’єктно-орієнтована мова програмування загального призначення, створена корпорацією Microsoft і стандартизована Європейською асоціацією виробників комп’ютерів (ECMA) та Міжнародною організацією зі стандартизації (ISO) [33].

Розробником C# став Андерс Хейлсберг разом зі своєю командою під час створення платформи .NET Framework.

C# побудовано для Common Language Infrastructure (CLI) — середовища виконання, що об’єднує виконуваний код і бібліотеки, забезпечуючи сумісність різних мов програмування на різних апаратних платформах.

Попри те, що C# є об’єктно-орієнтованою мовою, вона також підтримує компонентно-орієнтоване програмування, що дозволяє створювати самостійні, багаторазово використовувані модулі функціональності. Такі компоненти мають властивості, методи та події, а також атрибути, які описують їх поведінку. Мова C# пропонує зручні засоби для роботи з цими концепціями, що робить її природним вибором для розробки складних програмних систем [34].

До основних можливостей C# належать:

Автоматичне керування пам’яттю завдяки механізму збирання сміття, який звільняє ресурси, зайняті непотрібними об’єктами.

Обробка винятків, що забезпечує гнучкий і структурований підхід до виявлення та усунення помилок.

Типобезпечний дизайн, який запобігає помилкам, пов’язаним із неініціалізованими змінними, виходом за межі масивів або некоректними перетвореннями типів.

C# має уніфіковану систему типів: усі типи, включаючи базові (наприклад, int і double), успадковуються від єдиного кореневого типу object. Це дозволяє працювати з усіма типами даних узгоджено, зберігати та передавати їх

універсальним способом. Крім того, мова підтримує як типи значень, так і типи посилань, забезпечуючи гнучкість у розподілі пам’яті й ефективності виконання.

Основні властивості мови C#:

широкий спектр застосування — від простих консольних і графічних додатків до тривимірних ігор і систем математичного моделювання в реальному часі;

підтримка сучасних парадигм програмування;

безпечність, надійність і висока продуктивність.Виділимо основні властивості мови C#

- широкий спектр застосувань, від простих додатків до графічних інтерфейсів до яскравих 3d-ігор та математичного моделювання в реальному часі;
- ефективний, швидкий і потужний;
- дуже портативний, найкращий вибір для кросплатформеної розробки;
- об'єктно-орієнтовані класи мови програмування, абстракція даних та інкапсуляція, успадкування та поліморфізм;
- багаті бібліотеки функцій;
- підтримує обробку виключень і перевантаження функцій [35].

3.1.1 Технологія ASP.NET

ASP.NET - це набір інструментів веб-розробки, запропонованих корпорацією Майкрософт. Такі програми, як Visual Studio .NET та Visual Web Developer, дозволяють веб-розробникам створювати динамічні веб-сайти за допомогою візуального інтерфейсу.

Звичайно, програмісти можуть писати власний код та сценарії та включати його також на веб-сайти ASP.NET. Хоча ASP.NET часто розглядається як наступник технології програмування ASP від Microsoft, ASP.NET також підтримує Visual Basic.NET, JScript .NET та мови з відкритим кодом, такі як Python та Perl.

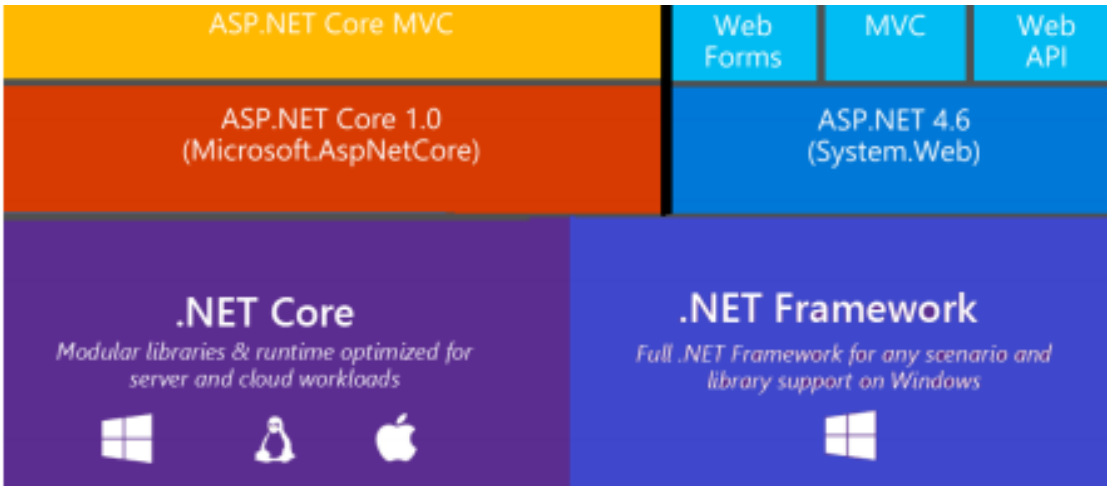


Рисунок 3.1 – Технології ASP.NET

ASP.NET побудований на платформі .NET, яка забезпечує інтерфейс прикладних програм (API) для програмістів. Засоби розробки .NET можна використовувати для створення додатків як для операційної системи Windows, так і для Інтернету. Такі програми, як Visual Studio .NET, надають розробникам візуальний інтерфейс для створення своїх додатків, що робить .NET розумним вибором для розробки веб-інтерфейсів.

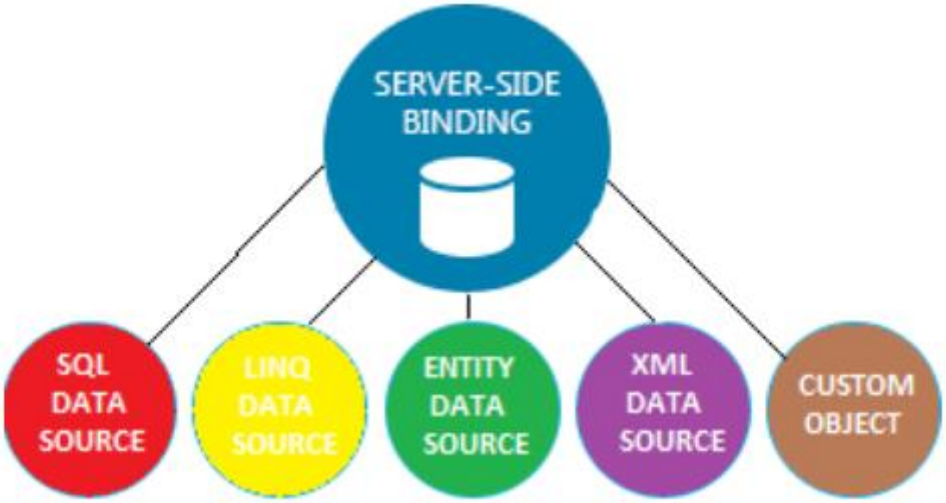


Рисунок 3.2 – Можливості технології ASP.NET

Щоб веб-сайт ASP.NET функціонував належним чином, його потрібно опублікувати на веб-сервері, що підтримує програми ASP.NET. Веб-сервер Інтернет-служб Інтернету (IIS) Microsoft на сьогоднішній день є найпоширенішою платформою для веб-сайтів ASP.NET. Хоча для систем на базі Linux доступні деякі варіанти з відкритим кодом, ці альтернативи часто забезпечують менше, ніж повну підтримку програм ASP.NET.

3.1.2 Технологія ASP.NET MVC

Архітектурний шаблон Model-View-Controller (MVC) поділяє застосунок на три ключові компоненти: модель, представлення та контролер. Такий підхід дає змогу розділити логіку програми, інтерфейс користувача та обробку взаємодії між ними, що робить розробку більш гнучкою та масштабованою.

Фреймворк ASP.NET MVC є альтернативою традиційній моделі веб-форм ASP.NET і дозволяє створювати вебзастосунки, побудовані на основі принципів MVC. Це легкий презентаційний фреймворк, який, як і Web Forms, тісно інтегрований із базовими можливостями платформи ASP.NET. Реалізація MVC у .NET визначена в просторі імен System.Web.Mvc і входить до складу бібліотеки System.Web [37].

Шаблон MVC є добре відомим стандартом проєктування, який широко використовується розробниками. Деякі типи вебдодатків особливо виграють від застосування цієї архітектури, отримуючи більшу керованість кодом і можливість гнучкого тестування. Водночас інші застосунки продовжують використовувати традиційний підхід ASP.NET на основі вебформ і зворотних викликів, або ж комбінують обидва методи, поєднуючи переваги кожного.

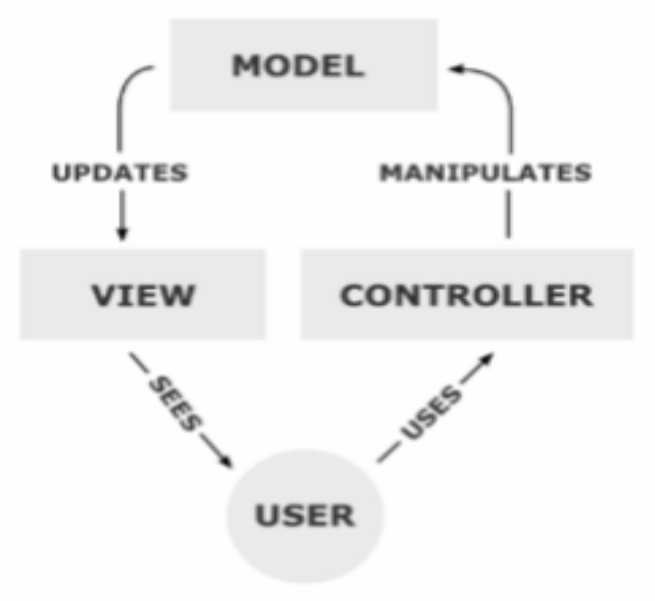


Рисунок 3.3 – Структура MVC

Моделі. Об’єкти моделі — це елементи програми, які реалізують бізнес-логіку та оперують даними домену. Зазвичай вони відповідають за отримання, обробку й збереження інформації у базі даних.

У невеликих застосунках модель може існувати лише концептуально, без окремої фізичної реалізації. Наприклад, якщо програма лише зчитує дані та передає їх для відображення, набір даних може виконувати роль моделі, навіть без створення окремого шару чи класів.

Представлення. Компоненти представлення відповідають за візуалізацію даних і формування користувацького інтерфейсу (UI). Як правило, інтерфейс побудований на основі інформації, отриманої з моделі, і слугує для взаємодії користувача з додатком.

Контролери. Контролери є посередниками між моделлю та представленням — вони отримують дані від користувача, обробляють їх і визначають, яку відповідь або вигляд слід відобразити.

Шаблон MVC дозволяє чітко розділити різні аспекти роботи програми:

вхідну логіку — контролер,

бізнес-логіку — модель,

логіку інтерфейсу користувача — представлення.

Таке розділення спрощує розробку, обслуговування та розширення вебдодатків, адже розробник може зосередитися лише на одному з рівнів програми, не зачіпаючи інші.

Крім того, слабка зв’язність між трьома компонентами MVC сприяє паралельній роботі в команді. Один програміст може розробляти інтерфейс, інший — контролери, а третій — логіку моделі, що значно підвищує ефективність усього процесу розробки.

3.2 Обґрунтування вибраних платформ та технологій

1) REST архітектура більш за все підходить для поставленої задачі, тому що надає статистичні дані для подальшої обробки та представленя її користувачу [38].

2) C# – це мова програмування, що була вибрана для написання даної автоматизованої системи для управління фубольною статистикою. Вона дозволяє за мінімальний час реалізувати необхідний функціонал, зручний для побудови веб-додатків [33].

3) ASP.NET – це .NET фреймворк, призначений для створення веб сервісів, який надає безліч можливостей для створення унікальних систем.

4) Microsoft SQL Server Management Studio – це багатофункціональна СУБД, призначена для управління та конфігурування компонентів Microsoft SQL Server [39].

5) Entity Framework– це технологія для доступу до баз даних, зручна у Використанні [40].

6) AutoMapper – це утиліта, яка дозволяє спроектувати одну модель на іншу для скорочення об’єму кода та спрощення системи [41].

7) Postman – це дуже зручна платформа для перевірки та корегування запитів до REST API [42].

8) Identity Server 4 – зручна система аутентифікації та авторизації. Дана система дозволяє користувачам створювати облікові записи, управляти ними та використовувати для входу на сайт облікові записи інших провайдерів [43].

3.3 Розробка веб-додатку

У магістерській дипломній роботі виконується реалізація веб-додатку у середовищі Microsoft Visual Studio [44] з використанням мови розробки – C#.

Оскільки для розробки був вибраний архітектурний шаблон Model-View-Controller. В структурі проекту є 3 основних папки: Controllers, Models, Views.

Класи в інших папках виконують роль обслуговуючих класів.

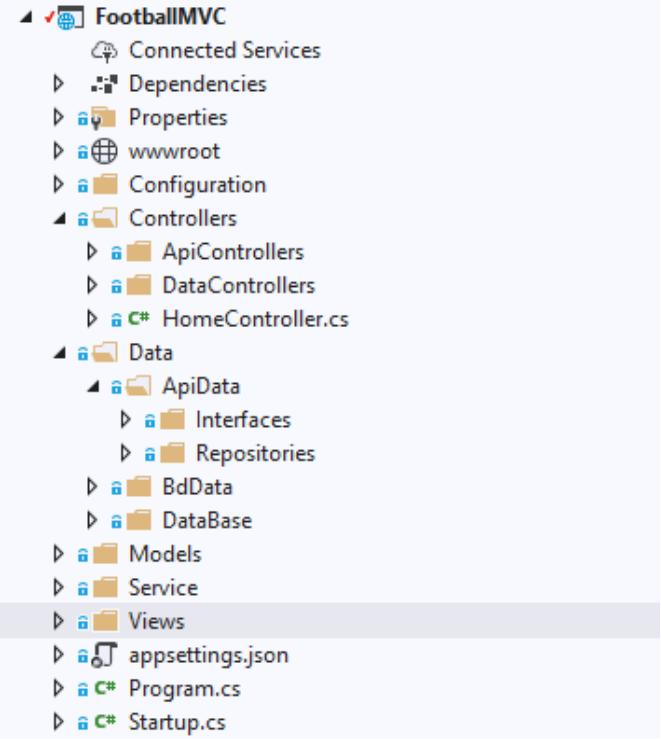


Рисунок 3.4 – Структура проекту

Клас Startup.cs являється вхідною точкою в додаток ASP.NET. Цей клас створює конфігурацію додатка, налаштовує сервіси, які веб-додаток буде використовувати, встановлює компоненти для обробки запитів.

Даний клас має два головних метода:

- void ConfigureServices()
- void Configure()

Метод ConfigureServices () реєструє сервіси, які використовуються додатком. Як параметр він приймає об'єкт IServiceCollection, який і представляє колекцію сервісів в додатку.

В даному методі налаштовується зв’язок з базою даних та за допомогою Dependency injection (DI) [45] встановлюються впровадження залежностей об'єктів які пов'язані між собою через абстракції, наприклад, через інтерфейси, що робить всю систему більш гнучкою, більш адаптованою і обширнішою.

Метод Configure встановлює, як додаток буде обробляти запити. Для установки компонентів, які обробляють запит, використовуються методи об'єкта IApplicationBuilder. Об'єкт IApplicationBuilder є обов'язковим параметром для методу Configure.

В даному методі реєструються сервіс авторизації, сервіс використання статистичних файлів, налаштовується маршутриразації додатку.

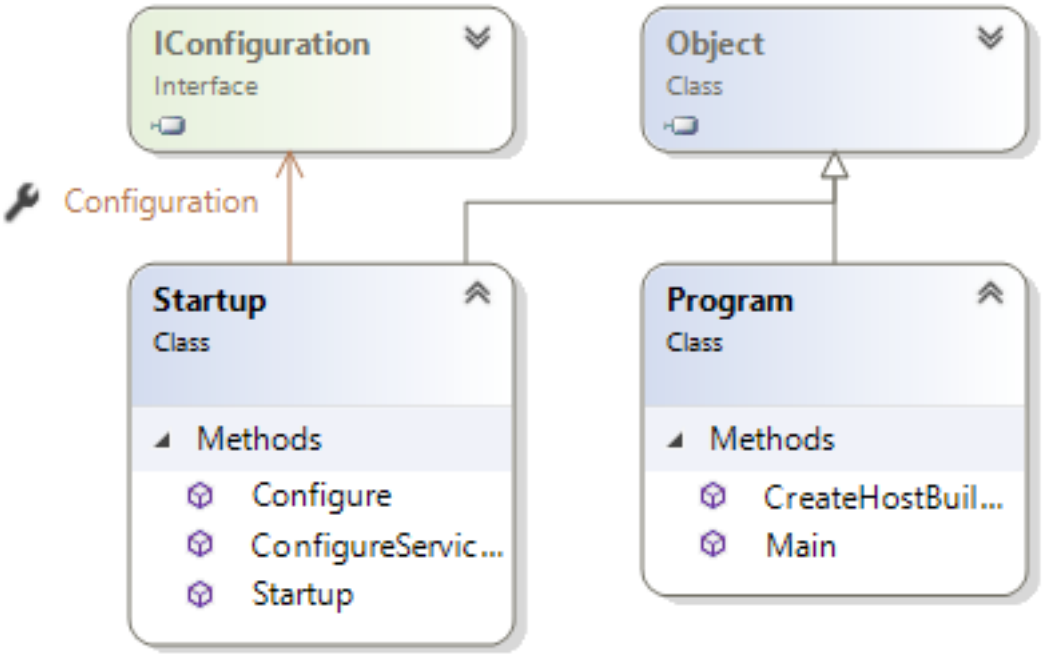


Рисунок 3.5 – Діаграма класів вхідної точки

Щоб взаємодіяти з базою даних через Entity Framework необхідний контекст даних - клас, успадкований від класу Microsoft EntityFramework DbContext. Тому було створено клас AppDBContext.cs. Даний клас має основну властивість DbSet являє собою колекцію об'єктів, яка зіставляється з певною таблицею в базі даних.

Наприклад, таблиці Players в базі даних відповідає властивість: public DbSet<Player> Players { get; set; }. Звертаючись до властивості Players можна отримувати, додавати та редагувати дані в базі даних. Клас AppDBContext.cs унаслідується від інтерфейсу DbContext.

На рисунку 3.5 зображена діаграма класів для взаємодії з базою даних.

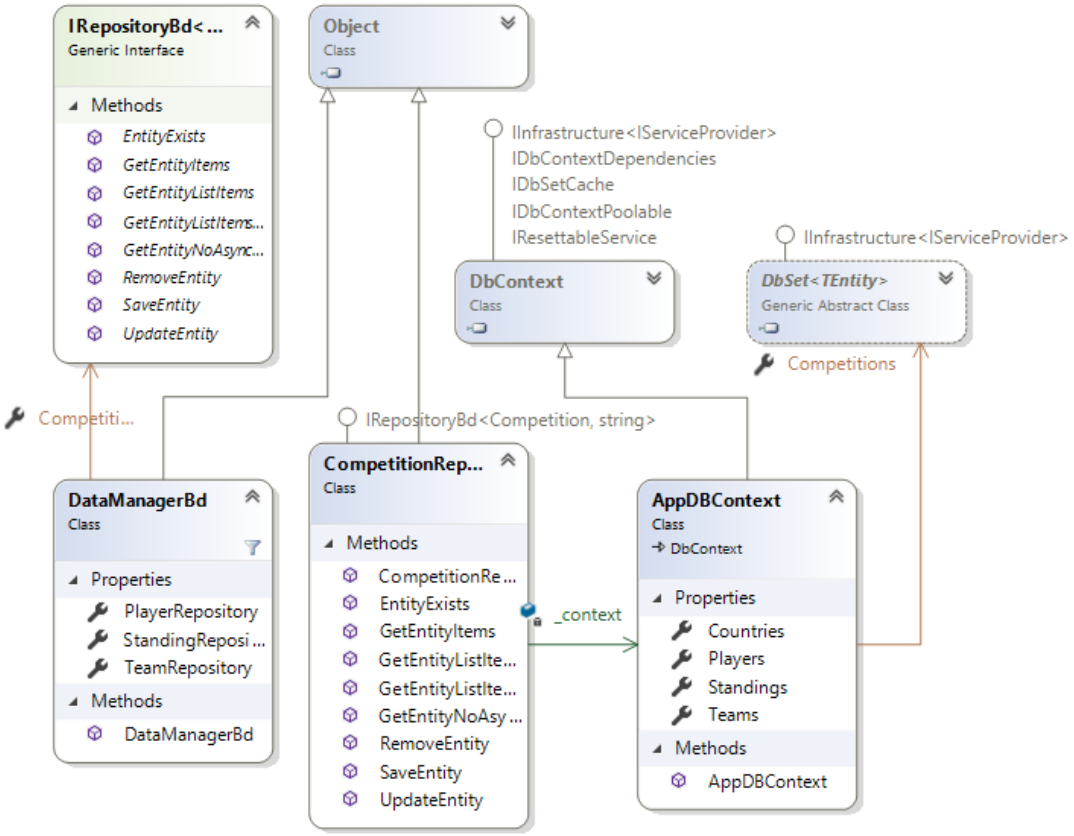


Рисунок 3.6 – Діаграма класів для взаємодії з базою даних

В Інтерфейсі IRepositoryBd описані методи завдяки яким, виконуються різноманітні запити до бази даних. Для прикладу на рисунку 3.5 показано клас CompetitionRepositortDb який реалізує дані методи. Клас DataManagerBd використовується для впровадження залежностей.

Web API представляє собою веб-службу, до якої можуть звертатися інші додатки, також розроблений для роботи в стилі REST (Representation State Transfer або "передача стану вистави"). REST-архітектура передбачає застосування таких методів або типів запитів HTTP для взаємодії з сервером:

– GET;
– POST;
– PUT;
– DELETE.

В даному випадку найчастіше буде використовуватись типи запиту HTTP GET.

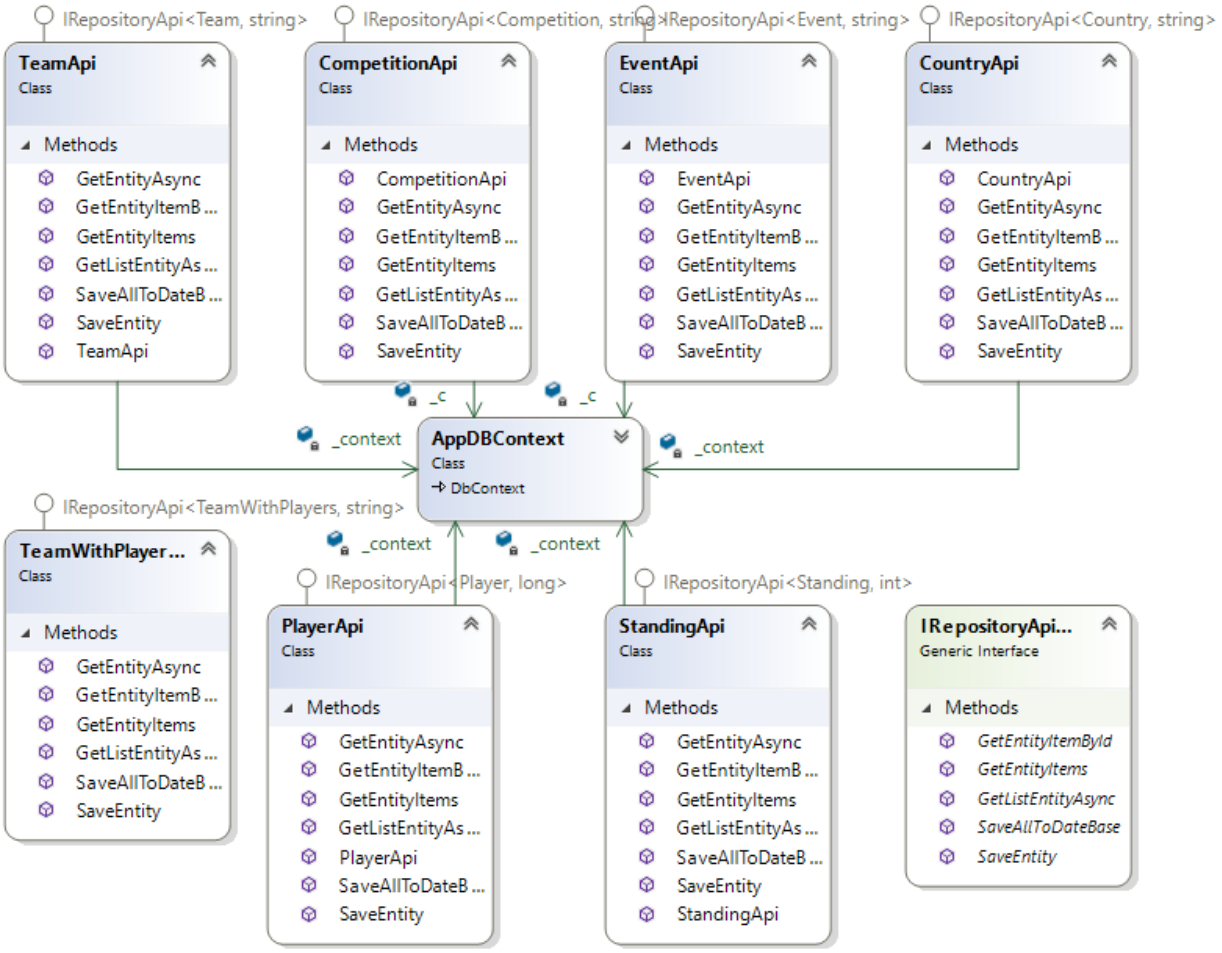


Рисунок 3.7 – Діаграма класів для взаємодії з WepApi

Інтерфейс IRepositoryAPI описує метод GetListEntityAsync. Даний метод реалізують класи, які унаслідують вище наведений інтефейс.

Опис методу GetListEntityAsync наведений нище.

```
public async Task<List<Event>> GetListEntityAsync(string path, HttpClient client)
      {
          string json;
          List<Event> events = new List<Event>();
```

```
            HttpResponseMessage response = await client.GetAsync(path);
            if (response.IsSuccessStatusCode)
        {
                json = await response.Content.ReadAsStringAsync();
                events = JsonSerializer.Deserialize<List<Event>>(json);
              }
            return events;
         }
```

Для встановлення зв’язку відправляється запит по переданому в метод URL, якщо сервер відправляє статус “200”, то веб-додаток отримує відповідь в форматі JSON. Далі виконується перетворення JSON в список об’єктів Event.

Приклад використання використання методу GetListEntityAsync наведений нище.

```
        private static HttpClient client = new HttpClient();
        string defolt = URL +"get_events&from=2020-11-27&to=2020-11-29" + APIkey + "&league_id=149";
        await dataManager.EventApi.GetListEntityAsync(defolt, client);
```

Інтерфейс IRepositoryAPI також реалізує багато інших методів, які виконують різний функціонал.

Оскільки веб-додаток реалізує архітектурний патерн MVC, потрібно розробити моделі, які реалізують логіку предметної області.

Як правило, всі використовувані сутності в додатку інтерпретуються в окремі моделі, які і описують структуру кожної сутності. Залежно від завдань і предметної області ми можемо виділити різну кількість моделей в додатку.

У веб-додатку моделі можна розділити за ступенем застосування на кілька груп:

- моделі, які використовуються для передачі даних в представлення;
- моделі, об'єкти яких зберігаються в базах даних;
- допоміжні моделі для проміжних обчислень.

На рисунку 3.8 наведена діаграма класів моделей предметної області.

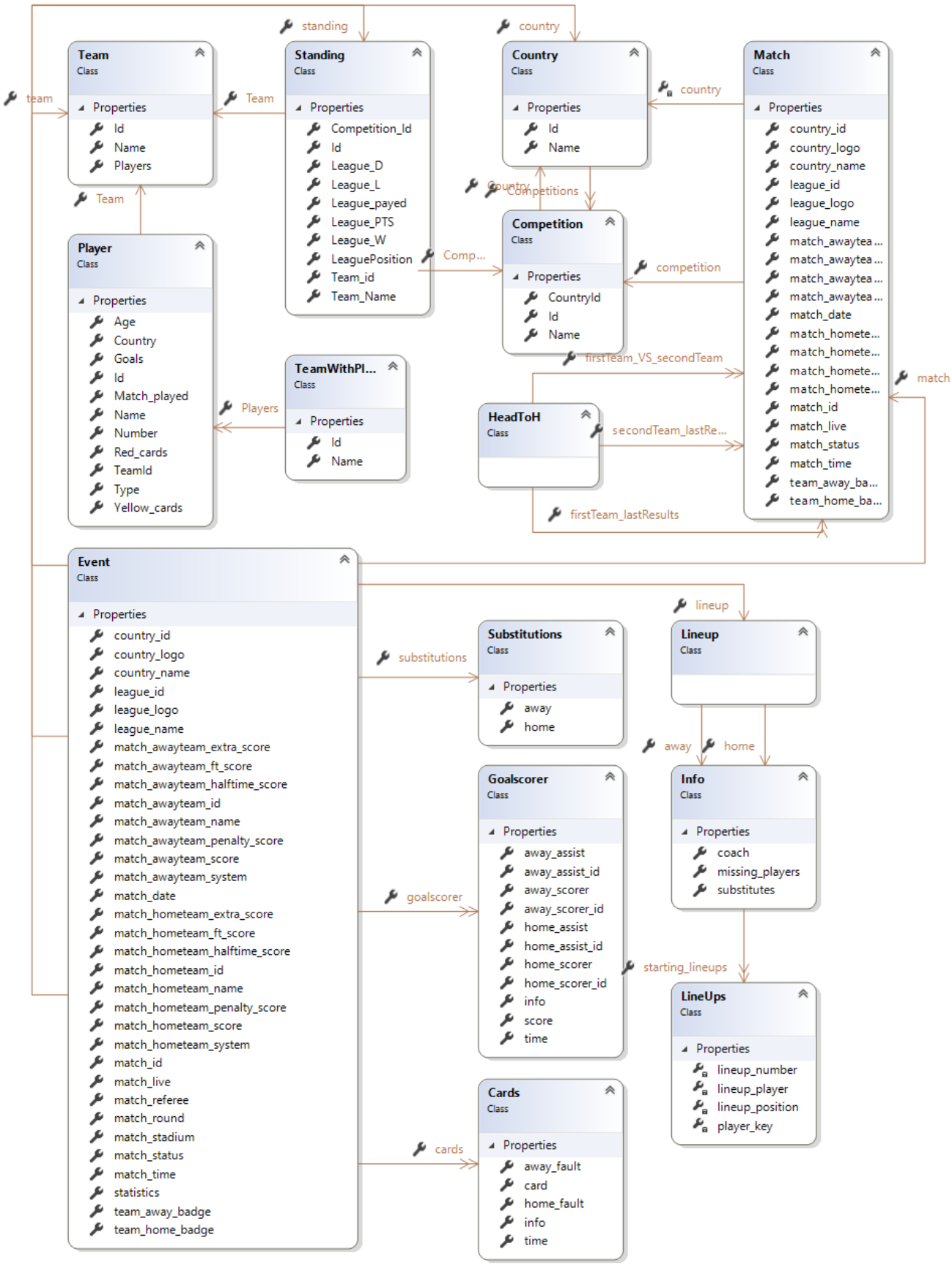


Рисунок 3.8 – Діаграма класів моделей предметної області

Центральною ланкою в архітектурі ASP.NET MVC є контролер. При отриманні запиту система маршрутизації обирає для обробки запиту потрібний контролер і передає йому дані запиту. Контролер обробляє ці дані і посилає назад результат обробки.

Контролер успадковується від абстрактного базового класу Microsoft.AspNetCore.Mvc.Controller.

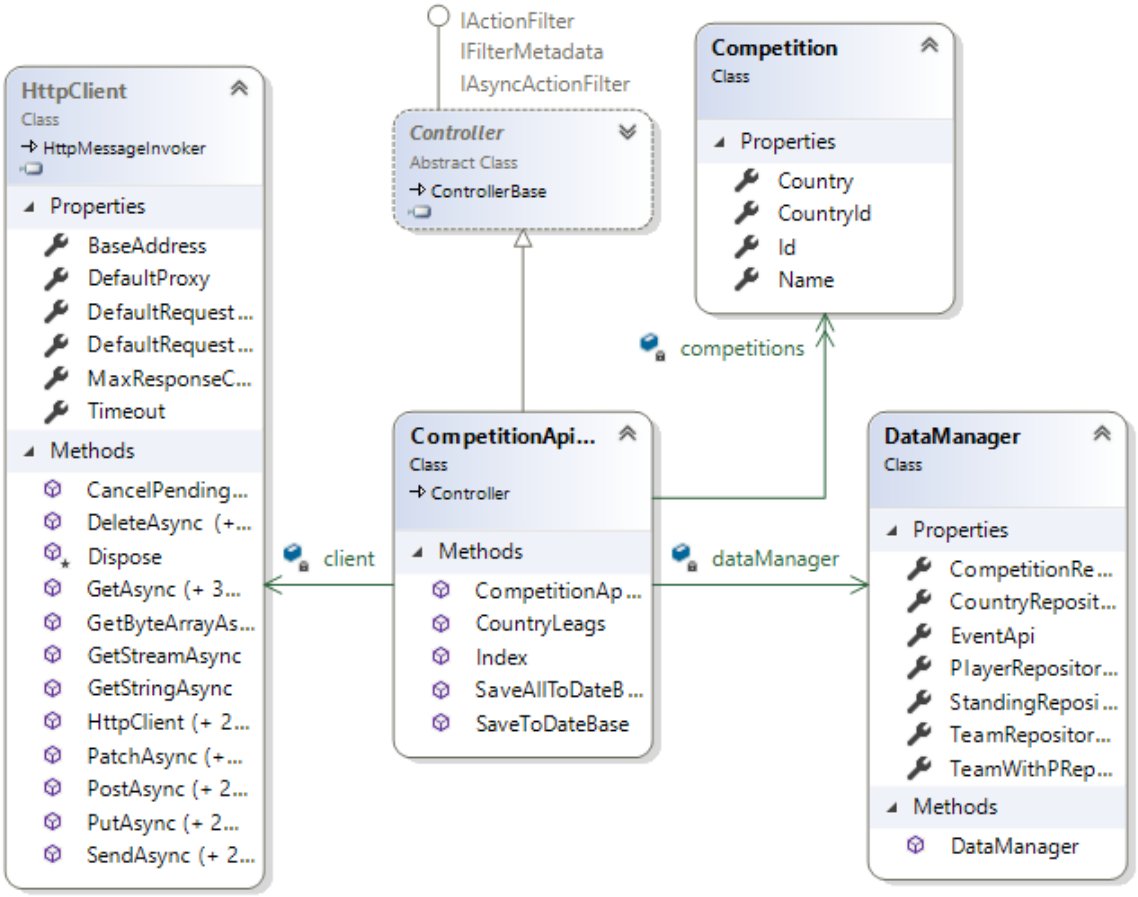


Рисунок 3.9 – Діаграма класів для взаємодії з контролером

При зверненні до веб-додатку, як правило, користувач очікує отримати деяку відповідь, наприклад, у вигляді веб-сторінки, яка наповнена даними. На стороні сервера метод контролера, отримуючи параметри і дані запиту, обробляє їх і формує відповідь у вигляді результату дії. Результат дії - це той об'єкт, який повертається методом після обробки запиту.

В більшості випадків контролер формує об'єкти типу IActionResult, які безпосередньо призначені для генерації результату дії. Інтерфейс IActionResult знаходиться в просторі імен Microsoft.AspNet.Mvc.

Приклад використання інтерфейсу IActionResult наведений нище.

```
public async Task<IActionResult> SaveAllToDateBase(string id) {
      List<Competition> coun = new List<Competition>();
      ViewData["Answer"] = "Збережено";
      if (dataManager.CompetitionRepositoryApi.GetEntityItems(){
          coun = dataManager.CompetitionRepositoryApi.SaveAllToDateBase(competitions);
          if(coun.Count == 0)
          {
            ViewData["Answer"] = "Всі турніри даної країни вже були збережені в базі даних";
            return View(competitions[0]);}
           foreach (Competition c in coun) {
                  await dataManager.CompetitionRepositoryApi.SaveEntity(c); }
                  return View(competitions[0]);
           }
           foreach (Competition c in competitions){
                   await dataManager.CompetitionRepositoryApi.SaveEntity(c);}
                  return View(competitions[0]);
    }
```

Даний метод виконує функцію збереження турнірів в базу даних і також за допомогою виклику методу View відображає дані в представлені.

Представлення в ASP.NET MVC може містити не тільки стандартний код html, а й також вставки коду на мові C #.

Представлення зберігаються в папці Views.

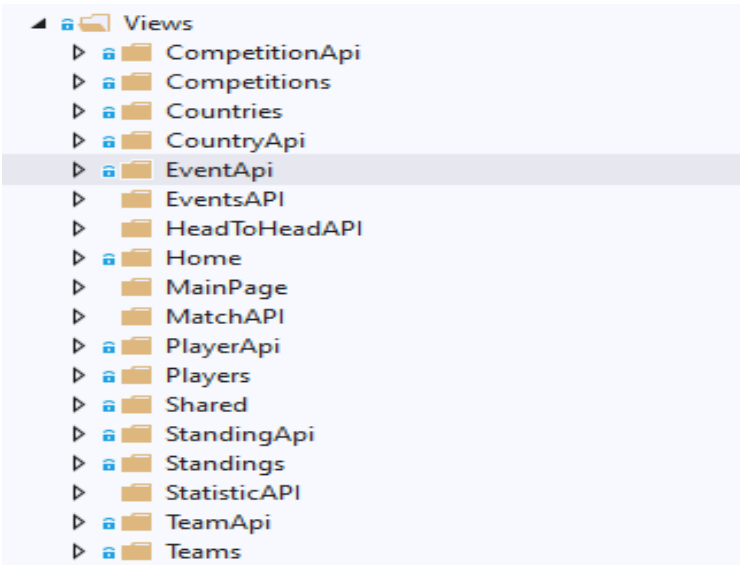


Рисунок 3.10 – Структура папки Views

При виклику методу View контролер не виконує рендеринг представлення і не генерує розмітку html. Контролер тільки готує дані і вибирає, яке представлення потрібно повернути в якості об'єкта ViewResult.

Макет сайту знаходитись в папці Shared, в ній знаходяться такі файли:

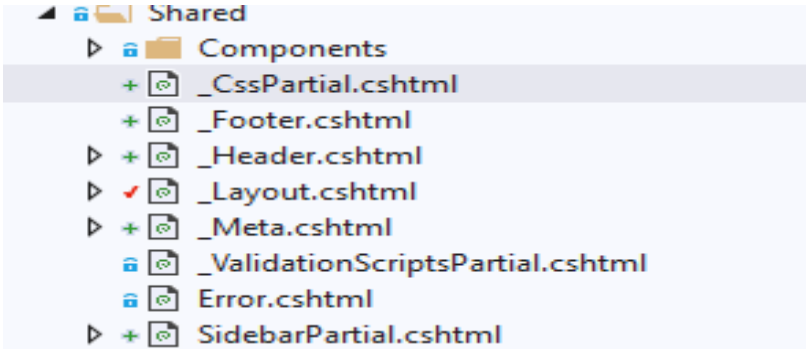


Рисунок 3.11 – Вміст папки Shared

Макет сайту розподілений по файлам, з розширенням cshtml. Головний файл це – Layout.cshtml, при відкриті веб-сторінки на сайті, саме цей файл генерує макет сайту.

Загальний вигляд веб-додатку зображений на рисунку 3.12.

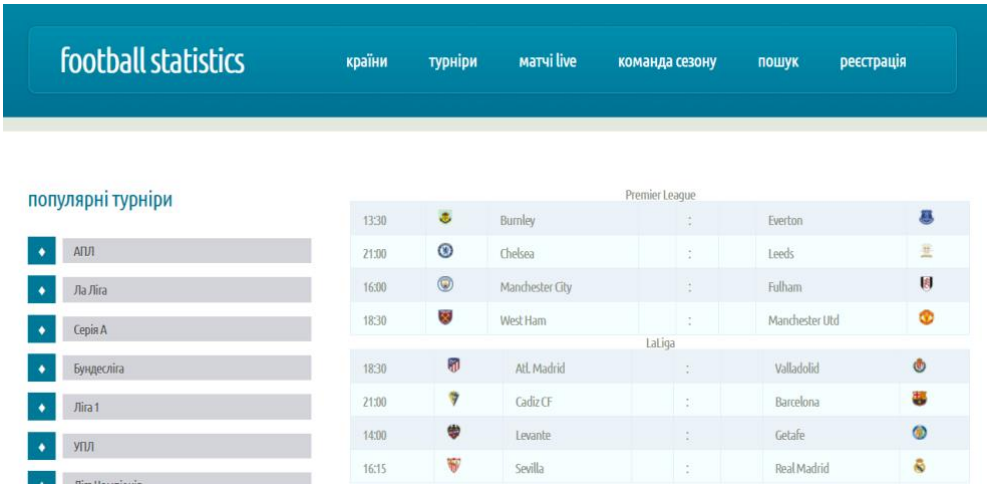


Рисунок 3.12 – Загальний вигляд веб-додатку

Умовна головна сторінка розділена на головне меню, список популярних турнірів для бистрого доступу, та робочу область. Лістинг програми наведено в додатку Г.

РОЗДІЛ 4
ТЕСТУВАННЯ ВЕБ-РЕСУРСУ

В розділі наведено опис можливостей та проведено тестування веб-ресурсу у таких напрямках: тестування програмної частини та інтерфейсу користувача параметрів. Розглянуто інструкцію користувача.

4.1 Огляд підходів тестування веб-ресурсів

ВЕБ-ТЕСТУВАННЯ – це перевірка веб-сайту на наявність потенційних помилок до їх публікації та доступності для широкої громадськості. Веб-тестування перевіряє функціональність, зручність використання, безпеку, сумісність, ефективність веб-програми [46].

На цьому етапі перевіряються такі питання, як безпека веб-додатків, функціонування сайту, його доступ для звичайних користувачів та його здатність обробляти трафік.

В загальному виділяють наступні підходи тестування:

– тестування функціональності веб-сайту;
– тестування зручності використання;
– тестування інтерфейсів;
– тестування бази даних;
– тестування на сумісність і різними веб-браузерами;
– тестування продуктивності;
– тестування безпеки.

Тестування функціональності веб-сайту - це процес, що включає кілька параметрів тестування, таких як користувальницький інтерфейс, тестування безпеки, тестування клієнта та сервера та основні функціональні можливості веб-сайту. Функціональне тестування є дуже зручним і дозволяє користувачам проводити як ручне, так і автоматизоване тестування. Він проводиться для перевірки функціональних можливостей кожної функції на веб-сайті.

– Засоби веб-тестування включають:
– перевірку внутрішніх та зовнішніх посилань;

- перевірку заповнення форм;
- Наприклад - якщо користувач не заповнює обов’язкове поле у формі, відображається повідомлення про помилку.
- перевірку файлів cookie;
- перевірка HTML і CSS, щоб переконатися, що пошукові системи можуть легко сканувати ваш сайт;
- перевірка бізнес-логіки робочого процесу.
- Також необхідно тестувати негативні сценарії, наприклад користувач виконує несподіваний крок, і у веб-додатку відображається відповідне повідомлення про помилку.

Тестування зручності використання є важливою частиною будь-якого веб-проекту. Дане тестування можуть проводити, як і розробники, так і цільова аудиторія веб-програми. Меню, кнопки або посилання на різні сторінки веб-сайту повинні бути легко видимими та узгодженими з іншими сторінками. Вміст повинен бути розбірливим, без орфографічних або граматичних помилок.

- При тестуванні інтерфейсів, виділяють три області, які потрібно перевірити – це бізнес логіка, веб-сервер і сервер баз даних
- Тестові запити правильно надсилаються до бази даних, та на стороні клієнта відображаються правильно. Виявлені помилки повинні відображатися лише адміністратору, а не кінцевому користувачеві.
- База даних є однією з найважливіших складових веб-додатку, і для її ретельного тестування необхідно зробити стресову ситуацію, тобто велике навантаження на базу даних.
- Тести сумісності гарантує, що веб-ресурс відображається правильно на різних пристроях.
- Необхідно перевірити, чи правильно відображається веб-ресурс в різних браузерах та чи працює нормально JavaScript, AJAX і автентифікація. Також потрібно перевірити сумісність мобільного пристрою.

- Візуалізація веб-елементів, таких як кнопки, текстові поля тощо, змінюється із зміною операційної системи. Тому виконується перевірка на різних операційних систем, таких як Windows, Linux, Mac та в браузерах, таких як Firefox, Internet Explorer, Safari тощо.
- Тестування продуктивності одне із найважливіших, та в майбутньому забезпечить роботу сайту під будь-яким навантаженням.
- Для визначення продуктивності необхідно виконати наступні тести:
- перевірити час відгуку веб-програми з різною швидкістю з’єднання;
- перевірка поведінки при нормальних та пікових навантаженнях;
- стресовий тест, для визначення піку навантаження при якому, веб-додаток буде працювати коректно.
- Тестування безпеки є важливим для веб-сайту оскільки, ресурс зберігає конфіденційну інформацію про клієнтів, наприклад, логін та пароль.
- Необхідно виконати наступні перевірки:
- несанкціонований доступ до захищених сторінок;
- завантаження файлів без відповідного дозволу;
- завершення сеансів користувачів, після тривалої бездіяльності;
- при використанні сертифікатів SSL веб-сайт повинен перенаправляти на зашифровані сторінки SSL.

4.2 Інструкція користувача

Під час запуску веб-ресурсу завантажується головна сторінка сайту (рисунок 4.1).

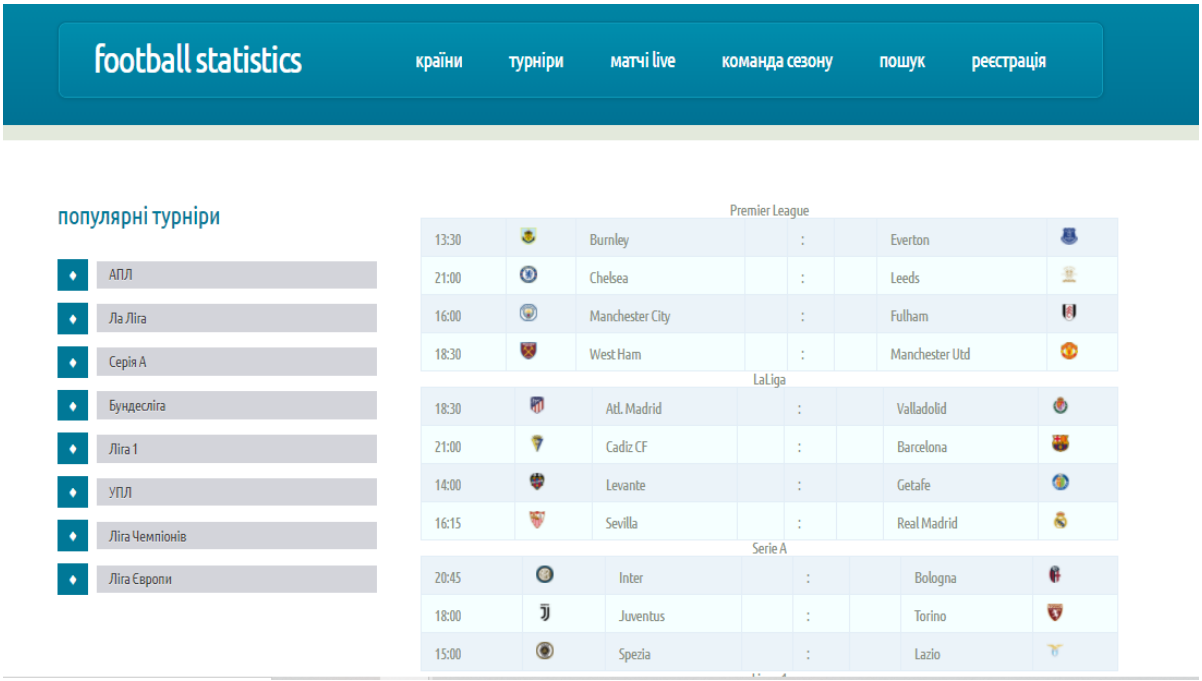


Рисунок 4.1 – Головна сторінка сайту

Головна сторінка демонструє основні функціональні можливості веб-ресурсу.

Зверху знаходиться «Меню». Меню містить наступні пункти: «Країни», «Турніри», «Матчі live», «Команда сезону», «Пошук» та «Реєстрація».

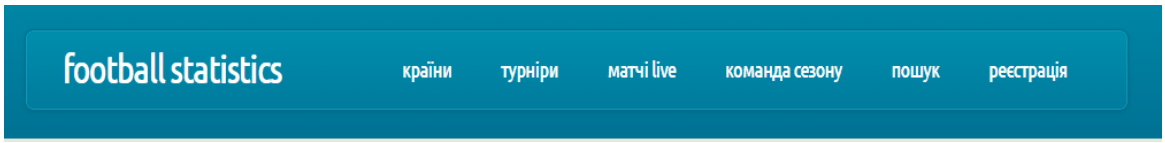


Рисунок 4.2 – Вигляд меню

Зліва знаходиться панель бистрого доступу, вона містити секцію «Популярні турніри». В списку турнірів знаходяться Топ 5 ліг Європи, Євро-кубки, також додано Українську Прем’єр Лігу.

Рисунок 4.3 – Вигляд панелі бистрого доступу

Робоча область займає місце по середині, на головній сторінці, на ній відображаються матчі з популярних турнірів, які будуть зіграні сьогодні.

Рисунок 4.5 – Вигляд таблиці зі списком матчів на сьогодні

За допомогою календаря, користувачу надається можливість вибрати дату, та по вказаній даті будуть відображені матчі, причому дата може бути як і минула так і майбутня, наприклад користувач вибирає дату 28.11.2020.

Якщо матч відбувся, також показується рахунок.

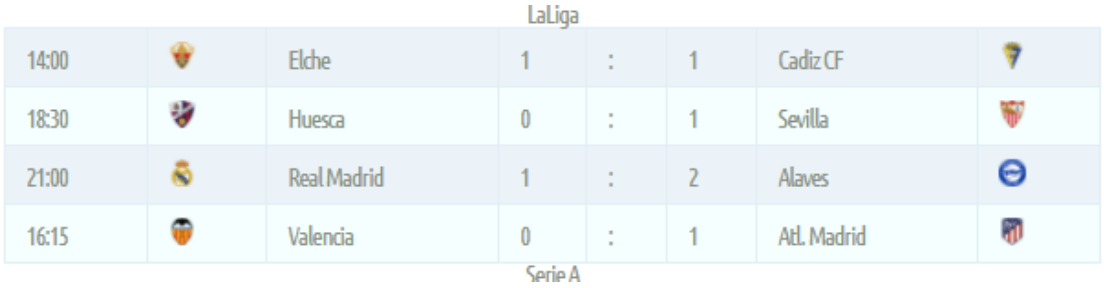


Рисунок 4.6 – Вигляд таблиці зі списком матчів за вказаною датою

Для перегляду детальної статистики матчу необхідно на нього натиснути.

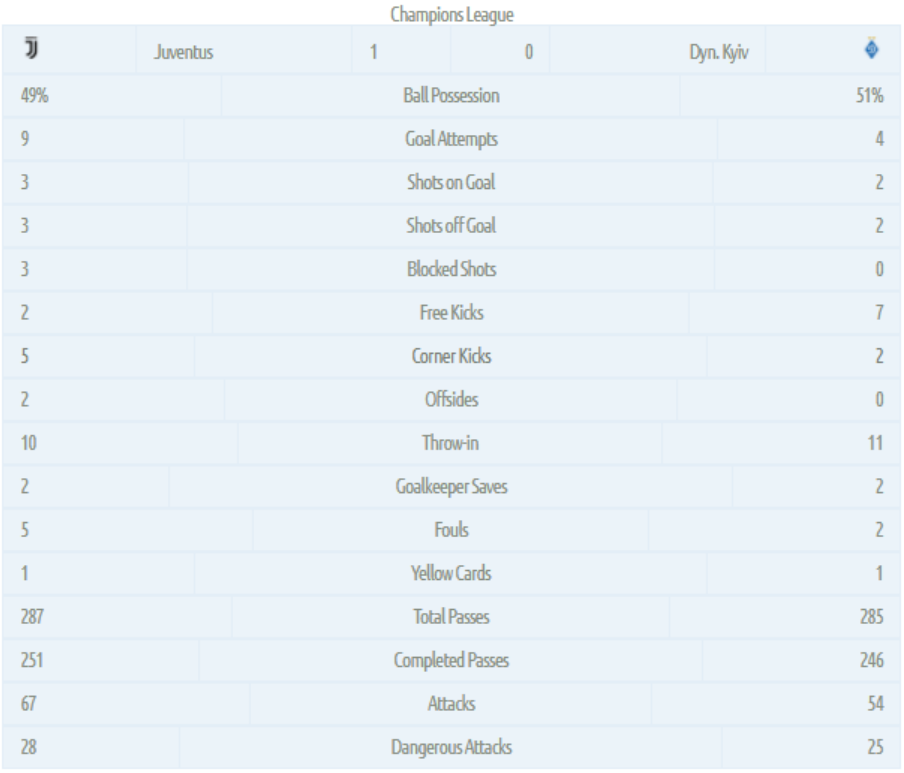


Рисунок 4.7 – Статистика матчу

Далі відкривається сторінка, яка має вміст наведений на рисунку 4.7, вона показує основну статистику матчу.

Також, автоматизована система надає можливість переглядати статистичу інформацію про будь-який турнір. Для цього потрібно натиснути на пункт меню сторінки “Країни”.

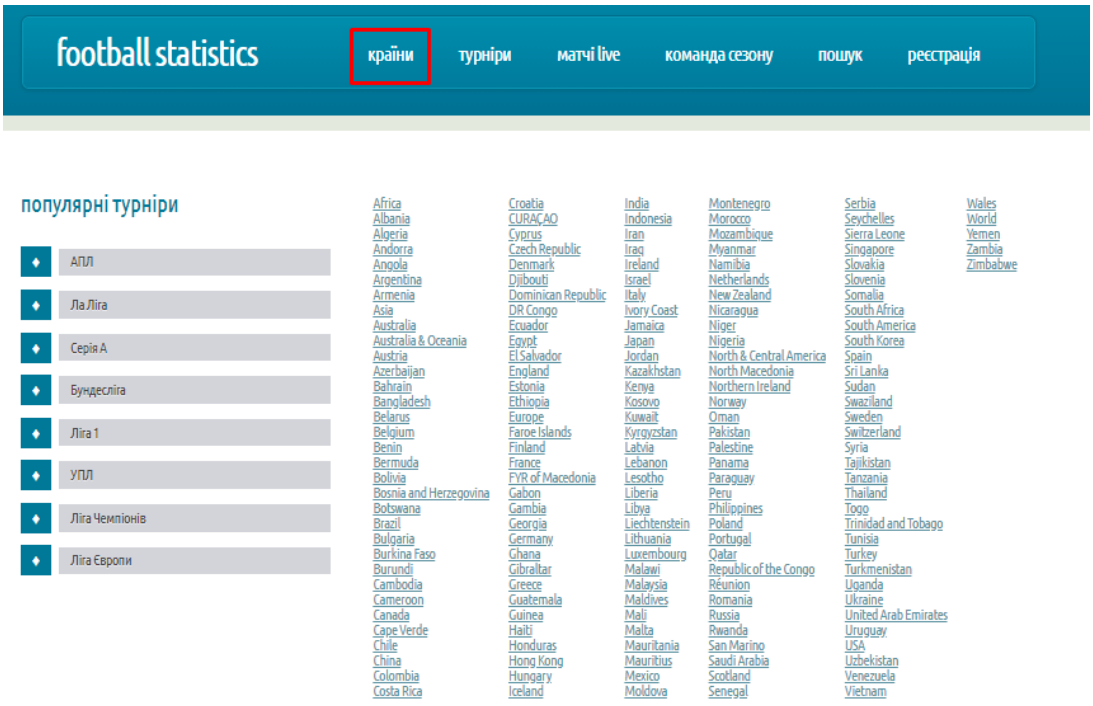


Рисунок 4.8 – Зображення сторінки зі списком країн

При натиснені на країну, відображається інформації про офіційні футбольні турніри в даній країні. Для прикладу переглянемо список турнірів для країни Бельгія.

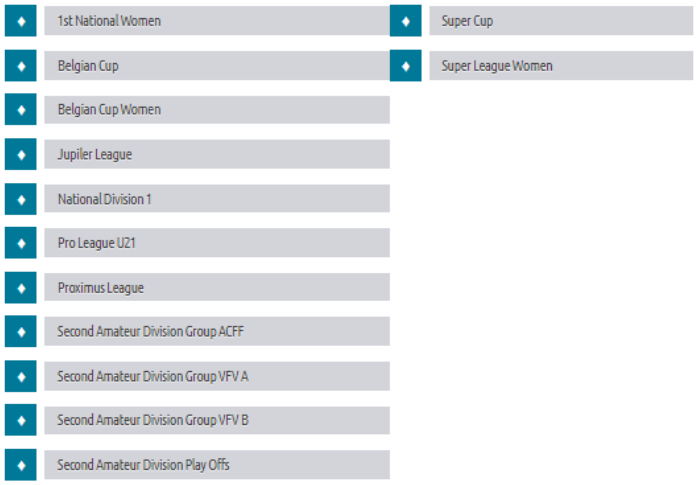


Рисунок 4.9 – Зображення сторінки із скисок турнірів Бельгії

Для того щоб переглянути інформацію про турнір необхідно на нього натиснути.

Позиція в таблиці	Назва	Кількість зіграних матчів	Кількість виграних матчів	Кількість нічиїх	Кількість програних матчів	Кількість очків
1	Beerschot VA	14	9	1	4	28
2	Genk	14	8	4	2	28
3	Club Brugge KV	14	8	3	3	27
4	Antwerp	14	7	4	3	25
5	St. Liege	14	6	6	2	24
6	Charleroi	14	7	2	5	23
7	Anderlecht	14	5	7	2	22
8	Kortrijk	14	5	4	5	19
9	Leuven	13	5	4	4	19
10	Cercle Brugge KSV	14	6	0	8	18
11	Gent	14	5	1	8	16
12	Oostende	14	4	4	6	16
13	Eupen	13	3	7	3	16
14	Waregem	14	4	2	8	14
15	KV Mechelen	13	3	3	7	12
16	Waasland-Beveren	14	3	3	8	12
17	St. Truiden	14	2	5	7	11

Рисунок – 4.10 – Вигляд сторінки із відображенням статистичної інформації про турнір

При натиснені на турнір відкривається сторінка, яка має наступну навігацію “Таблиця”, “Календар”, “Команди”, “Результати”.

Та за замовчуваням відкривається сторінка, яка демонтрує турнірну таблицю чемпіонату Бельгії.

У вкладці “Календар” демонструєця інформація про матчі наступних турів, коли вони відбудуться, які команди будуть грати між собою.

Вкладка команди, демонструє список команд, які приймають участь в турнірі.

Вкладка результати, демострує останні результати матчів, які відбулись в минулому турі.

4.3 Тестування методом «чорного ящика»

Тестування методом “чорного ящика” - це метод тестування програмного забезпечення, при якому функціональність програмних додатків перевіряється без знання внутрішньої структури коду, деталей реалізації та іншого. Тому тестувальнику необхідно знати тільки вимоги та специфікації продукту [47].

Основні види тестування чорних ящиків наведені нижче [48]:

Функціональне тестування – це тип тестування на чорну скриньку, який пов’язаний з функціональними вимогами системи.

Нефункціональне тестування – даний тип стосується нефункціональних вимог і призначений спеціально для оцінки системи до різних критеріїв, які не охоплюються шляхом функціонального тестування.

Регресійне тестування – проводиться після виправлення коду, оновлення або будь-якого іншого обслуговування системи, щоб перевірити чи нові зміни не вплинули на будь-яку існуючу функціональність.

Переваги:

– чітко відокремлює позицію кінцевого користувача від точки зору розробника;
– велика кількість середньо кваліфікованих тестувальників може протестувати додаток, не знаючи реалізації, мови програмування та операційних систем.

Недоліки:

– обмежене охоплення, оскільки насправді виконується лише вибрана кількість тестових сценаріїв;
– тестові кейси важко розробити.

Таблиця 4.1 Тест кейси тестування

Дія	Наявний результат	Очікуваний результат
Відкриття сторінки «Головна»	Головне вікно відкрито; назва вікна – FootballStatistic; Логотип сайту відображається зверху зліва; На формі поле реєстрація.	Головне вікно відкрито; назва вікна – FootballStatistic; Логотип сайту відображається зверху зліва; На формі поле реєстрація.
Ввести будь яке слова в Пароль	Символи заміняються на *****	Символи заміняються на *****
Ввод не вірного логіну	«Логін не вірний»	«Логін не вірний»
Ввод не Вірного паролю	«Пароль не вірний»	«Пароль не вірний»
Відкриття сторінки «Країни »	Вікно Країни відкрито; назва вікна – Країни; Відображається список Країн;	Вікно Країни відкрито; назва вікна – Країни; Відображається список Країн;
Відкриття сторінки «Турніри »	Вікно Турніри відкрито; назва вікна – Турніри; Відображається список Турнірів;	Вікно Турніри відкрито; назва вікна – Турніри; Відображається список Турнірів;
Відкриття сторінки «Турнірна таблиця »	Вікно Турнірна таблиця відкрито; назва вікна – Турнірна таблиця; Відображається список команд, кількість набраних очків та їх позиція;	Вікно Турнірна таблиця відкрито; назва вікна – Турнірна таблиця; Відображається список команд, кількість набраних очків та їх позиція;
Відкриття сторінки «Команди »	Вікно Команди відкрито; назва вікна – Команди; Відображається список команд, які приймають участь в турнірі та інформація про них;	Вікно Команди відкрито; назва вікна – Команди; Відображається список команд, які приймають участь в турнірі та інформація про них;

Після проведеного тестування, було створено тест кейси тестування, які підтверджують що система працює коректно та відповідає поставленим вимогам.

4.3 Тестування на валідність та кросбраузерність

Виконаємо перевірку на кроссбраузерність в браузерах: Google Chrome, Mozilla Firefox, Microsoft Edge.

Тестування головної сторінки в браузері Microsoft Edge наведено на рисунку 4.11

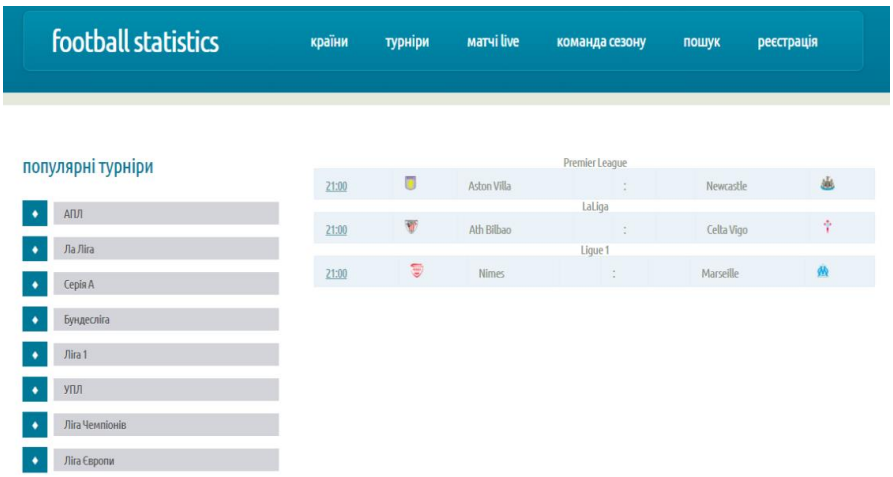


Рисунок 4.11 – Перевірка сайту в браузері Microsoft Edge

Тестування головної сторінки в браузері Mozilla Firefox наведено на рисунку 4.12

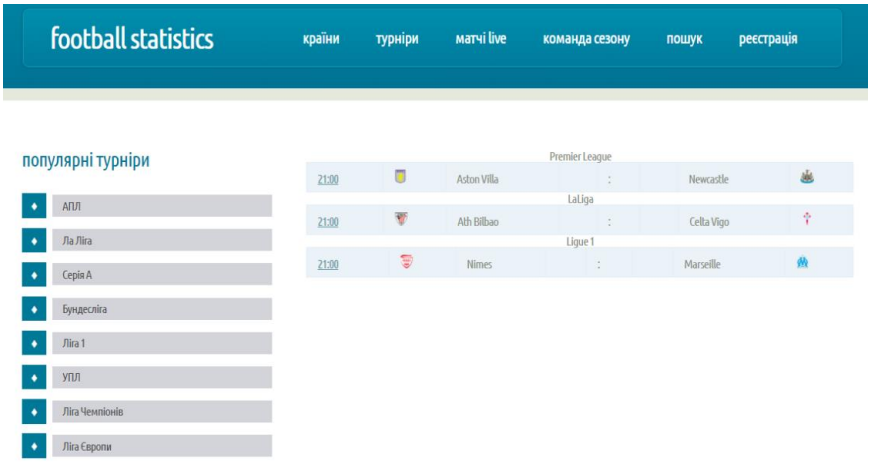


Рисунок 4.12 – Перевірка сайту в браузері Mozilla Firefox

Тестування головної сторінки в браузері Google Chrome наведено на рисунку 4.13

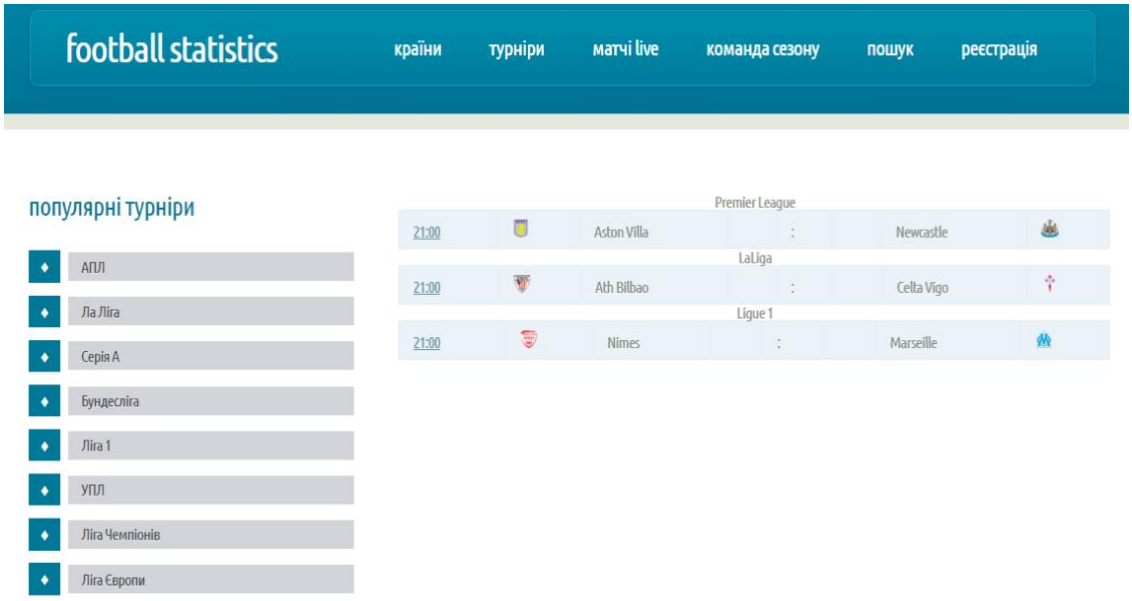


Рисунок 4.13 – Перевірка сайту в браузері Microsoft Edge

Під час тестування користувацького інтерфейсу було виявлено та виправлено помилки, завдяки проведеному тестуванню веб-ресурс працює коректно, та чудово відображається в різних браузерах. Веб-ресурс не підвисає та працює швидко.

РОЗДІЛ 5
ЕКОНОМІЧНИЙ РОЗДІЛ

5.1 Оцінювання комерційного потенціалу розробки

Спортивна статистика завжди була і є актуальною для спортивних фанатів та аналітиків, розвиток мережі інтернет надав змогу дізнатись будь-яку інформацію за лічені секунди, але якість отриманої інформації потрібно фільтрувати. Так само із футбольною статистикою існує безліч веб-сервісів, які надають поверхневу інформацію, тому важливо надати користувачам сервіс, який буде покращувати те що є, та надавати структуровану та цікаву статистичну інформацію. Важливим для управління футбольною статистикою є вміння надавати статистичну інформацію під час матчу, а не після його закінчення.

Складність процесу управління футбольною статистикою пов’язана з необхідністю збору та обробки великих обсягів даних. На сьогодні стан розвитку методів обробки та відтворення тісно пов'язаний з розвитком інформаційних технологій. Сьогодні вони є невід’ємними складовими процесів управління статистикою.

Тому була досліджена та розроблена ефективна автоматизована система управління футбольною статистикою, яка реалізована у вигляді веб-ресурсу.

Для проведення технологічного аудиту залучено трьох незалежних експертів. У рамках цієї роботи експертами виступають викладачі кафедри КСУ ВНТУ, зокрема:

Юхимчук М. С. (д.т.н., професор кафедри КСУ ВНТУ);

Ковтун В. П. (д.т.н., професор кафедри КСУ ВНТУ);

Ковалюк О.О (к.т.н., доцент кафедри КСУ ВНТУ).

Для оцінювання використано критерії, наведені у таблиці 5.1.

Таблиця 5.1 - Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 5.1

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та звільнення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні.	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування.
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки зведено до таблиці 5.2.

Таблиця 5.2 - Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	4	4
2. Ринкові переваги (наявність аналогів)	3	4	3
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	4	4	3
5. Ринкові переваги (експлуатаційні витрати)	4	4	4
6. Ринкові перспективи (розмір ринку)	4	5	4
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	4	4
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	3
12. Практична здійсненність (розробка документів)	3	4	3
Сума балів	46	46	42
Середньоарифметична сума балів $СБ_с$	44,7		

На основі даних, наведених у таблиці 5.2, можна здійснити аналіз комерційного потенціалу розробки. Далі порівняємо ці результати з рівнями комерційного потенціалу, представленими в таблиці 5.3.

Таблиця 5.3 - Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Результати експертного оцінювання показали, що середньоарифметична сума балів становить 44,7 балів. Це підтверджує високий науково-технічний рівень та потенційну комерційну успішність проведених досліджень, як вказано у таблиці 5.3.

5.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

Під час планування, обліку та калькулювання витрат, пов'язаних із проведенням науково-дослідної роботи на тему «Система прийняття торгових рішень на фінансових ринках на основі нечіткого кластерного аналізу часових рядів», витрати групуються за відповідними категоріями.

До категорії «Витрати на оплату праці» включаються витрати, пов'язані з виплатою основної та додаткової заробітної плати працівникам, які займають керівні посади та науковим, інженерно-технічним працівникам, безпосередньо залученим до виконання цієї роботи.

Для визначення фонду основної заробітної плати ($З_о$) використовується аналіз трудомісткості. Оскільки роботи мають дослідницький характер і виконуються частину робочого дня, розрахунок є більш точним при використанні годинних тарифних ставок. Витрати на основну заробітну плату дослідників $З_о$ розраховуємо за формулою:

$$З_о = \sum_{i=1}^{k} \frac{М_{пі} \times t_і}{T_р} \quad (5.1)$$

де k - кількість виконавців, залучених до процесу досліджень;

Мпі - місячний посадовий оклад конкретного дослідника, грн;

$t_і$ - число днів роботи конкретного дослідника, дні;

Тр - середнє число робочих днів в місяці, Тр = 21 дня.

$$З_о = \frac{18000}{21} \times 6 + \frac{25000}{21} \times 52 = 5142{,}9 + 61904{,}8 = 67047{,}7 \text{ грн}$$

Таблиця 5.4 - Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	18000	857,1	6	5142,9
Інженер-розробник	25000	1190,5	52	61904,8
			Всього	67047,7

Додаткову заробітну плату розраховуємо як 10-12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \times \frac{Н_{дод}}{100\%} \quad (5.2)$$

де Ндод - норма нарахування додаткової заробітної плати. Ндод приймемо як 12%.

$$З_{дод} = 67047{,}7 \times \frac{12}{100\%} = 8045{,}7 \text{ грн}$$

До статті «Відрахування на соціальні заходи» включаються внески на загальнообов'язкове державне соціальне страхування та витрати на соціальний захист населення, зокрема єдиний соціальний внесок (ЄСВ).

Нарахування на заробітну плату дослідників та працівників становить 22% від суми їх основної та додаткової заробітної плати і розраховується за наступною формулою:

$$З_н = (З_о + З_р + З_{дод}) \times \frac{Н_{зп}}{100\%} \quad (5.3)$$

де Нзп - норма нарахування на заробітну плату.

$$З_н = (67047{,}7 + 8045{,}7) \times \frac{22}{100\%} = 16520{,}3 \text{ грн}$$

До статті «Сировина та матеріали» відносяться витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, придбані у сторонніх підприємств, установ і організацій та використані для проведення досліджень за прямим призначенням згідно з нормами їх витрачання. Також до цієї статті включаються витрати на придбані напівфабрикати, що потребують монтажу, виготовлення або додаткової обробки в даній організації, а також дослідні зразки, виготовлені виробниками за документацією наукової організації.

Вартість матеріалів (М) розраховується окремо для кожного виду матеріалів за наступною формулою:

$$М = \sum_{j=1}^{n} (Н_j \times Ц_j \times К_j) - \sum_{j=1}^{n} (В_j \times Ц_{в_j}) \quad (5.4)$$

де Нj - норма витрат матеріалу j-го найменування, кг;

n - кількість видів матеріалів;

Цj - вартість матеріалу j-го найменування, грн/кг;

Кj - коефіцієнт транспортних витрат, (Кj = 1,1...1,15);

Вj - маса відходів j-го найменування, кг;

Цвj - вартість відходів j-го найменування, грн/кг.

Таблиця 5.5 - Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Вартість витраченого матеріалу, грн
Папір для принтера	200	3	600
Нотатки (стікери)	120	2	240
Канцелярський набір (ручка, олівець, лінійка)	100	3	300
Файли	70	2	140
USB-флеш-накопичувачі	150	2	300
		Всього	1580

До статті «Спеціальне обладнання для наукових (експериментальних) робіт» входять витрати на виготовлення та придбання спеціалізованого обладнання, яке може бути необхідним для проведення досліджень, а також витрати на його проектування, транспортування, монтаж і встановлення. У рамках цієї роботи витрати на спеціальне обладнання також не заплановані.

До статті «Програмне забезпечення для наукових (експериментальних) робіт» відносяться витрати на розробку та придбання програмного забезпечення, зокрема програм, алгоритмів і баз даних, необхідних для виконання досліджень, а також витрати на їх проектування, створення та інсталяцію. Балансова вартість програмного забезпечення розраховується за формулою:

$$В_{прг} = \sum_{i=1}^{k} (Ц_{іпрг} \times С_{прг.і} \times К_і) \quad (5.5)$$

де Ціпрг - ціна придбання одиниці програмного засобу даного виду, грн;

Спрг.і - кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

Кі - коефіцієнт, що враховує інсталяцію, налагодження програмного засобу

тощо, (Кі = 1,10...1,12);

k - кількість найменувань програмних засобів.

$$В_{прг} = 7500 \times 1 \times 1{,}1 + 14000 \times 1 \times 1{,}1 + 5000 \times 1 \times 1{,}1 = 30250 \text{ грн}$$

Таблиця 5.6 - Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Python 3.11 + бібліотеки (pandas, scikit-learn)	1	7500	8250
IDE PyCharm Professional	1	14000	15400
Jupyter Notebook + MySQL	1	5000	5500
		Всього	29150

До статті «Амортизація обладнання, програмних засобів та приміщень» включаються амортизаційні відрахування за кожним видом обладнання, устаткування, інших приладів і пристроїв, а також програмного забезпечення, які використовуються для проведення науково-дослідної роботи, за їх наявності в дослідницькій організації або на підприємстві.

У спрощеному вигляді амортизаційні відрахування за кожним видом обладнання, приміщень та програмного забезпечення можуть бути розраховані за допомогою прямолінійного методу амортизації за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} \times \frac{t_{вик}}{12} \quad (5.6)$$

де Цб - балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

tвик - термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

Тв - строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$А_{обл} = \frac{26000}{2} \times \frac{3}{12} + \frac{6000}{2} \times \frac{3}{12} = 4000 \text{ грн}$$

Таблиця 5.7 - Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук LENOVO	26000	2	3	3250
Монітор Asus ROG	6000	2	3	750
			Всього	4000

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на придбання палива у сторонніх підприємств, установ та організацій, яке використовується з технологічною метою для проведення досліджень. Ця стаття формується у разі проведення енергоємних наукових досліджень за методом прямого віднесення витрат і може становити значну частку у собівартості досліджень. Витрати на силову електроенергію (Ве) розраховуються за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \times t_i \times Ц_e \times K_{впі}}{\eta_i} \quad (5.7)$$

де Wyi - встановлена потужність обладнання на визначеному етапі розробки, кВт;
ti - тривалість роботи обладнання на етапі дослідження, год;
Це - вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Це = 12,50 грн;
Квпі - коефіцієнт, що враховує використання потужності, Квпі < 1;
ηі - коефіцієнт корисної дії обладнання, ηі < 1.

$$B_e = \frac{0{,}5 \times 480 \times 12{,}5 \times 0{,}95}{0{,}97} + \frac{0{,}03 \times 480 \times 12{,}5 \times 0{,}95}{0{,}97} = 3127{,}3 \text{ грн}$$

Таблиця 5.8 - Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук LENOVO	0,5	480	2950,3
Монітор Asus ROG	0,03	480	177,0
		Всього	3127,3

Стаття «Службові відрядження» охоплює витрати, пов'язані з відрядженнями штатних працівників, працівників за цивільно-правовими договорами, аспірантів, що зайняті науково-дослідницькою діяльністю, які пов'язані з тестуванням смарт-контрактів та взаємодією з DeFi платформами, а також витрати на участь у наукові конференції та виставки з безпеки смарт-контрактів, що мають прямий зв'язок з виконанням конкретних досліджень.

Витрати за цією статтею розраховуються у розмірі 20-25% від суми основної заробітної плати дослідників та робітників за допомогою формули:

$$В_{сп} = (З_о + З_р) \times \frac{Н_{сп}}{100\%} \quad (5.8)$$

де Нсп - норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо Нсп = 25%.

$$В_{сп} = 67047{,}7 \times \frac{25}{100\%} = 16761{,}9 \text{ грн}$$

Стаття «Інші витрати» включає витрати, які не були охарактеризовані у попередніх статтях витрат і можуть бути прямо віднесені до собівартості досліджень за безпосередніми показниками. Витрати за цією статтею обчислюються у розмірі 50-100% від суми основної заробітної плати дослідників та робітників за допомогою такої формули:

$$І_{ів} = (З_о + З_р) \times \frac{Н_{ів}}{100\%} \quad (5.9)$$

де Нів - норма нарахування за статтею «Інші витрати», приймемо Нів = 60%.

$$І_{ів} = 67047{,}7 \times \frac{60}{100\%} = 40228{,}6 \text{ грн}$$

Сталими (загальновиробничими) витратами охоплюються різноманітні витрати, пов'язані з управлінням організацією, зусиллями в інноваціях та раціоналізації, а також з набором та підготовкою персоналу, банківськими послугами, освоєнням виробництва, а також науково-технічною інформацією та рекламою.

Витрати за цією статтею розраховуються у розмірі 100-150% від суми основної заробітної плати дослідників та працівників з використанням такої формули:

$$В_{нзв} = (З_о + З_р) \times \frac{Н_{нзв}}{100\%} \quad (5.10)$$

де Ннзв - норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо Ннзв = 120%.

$$В_{нзв} = 67047{,}7 \times \frac{120}{100\%} = 80457{,}2 \text{ грн}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_{дод} + З_н + М + В_{прг} + А_{обл} + В_е + В_{сп} + I_{ів} + В_{нзв}$$

$$В_{заг} = 67047{,}7 + 8045{,}7 + 16520{,}3 + 1580 + 29150 + 4000 + 3127{,}3 + 16761{,}9 + 40228{,}6 + 80457{,}2 = 266918{,}7 \text{ грн}$$

Вартість завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів обчислюється відповідно до наступної формули:

$$ЗВ = \frac{В_{заг}}{\eta} \quad (5.11)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η = 0,8.

$$ЗВ = \frac{266918{,}7}{0{,}8} = 333648{,}4 \text{ грн}$$

Отже, прогноз загальних витрат ЗВ на виконання та впровадження результатів виконаної роботи складає 333648,4 грн.

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У ринкових умовах позитивний результат від можливого впровадження науково-технічної розробки для потенційного інвестора полягає у збільшенні чистого прибутку. Дослідження з розробки системи прийняття торгових рішень передбачають комерціалізацію протягом чотирьох років.

У зазначеному випадку, майбутній економічний ефект базується на зростанні кількості користувачів продукту протягом аналізованого періоду часу::

- у перший рік - 380 користувачів;
- у другий рік - 620 користувачів;
- у третій рік - 850 користувачів;
- у четвертий рік - 1100 користувачів.

N - кількість потенційних користувачів торгових систем у році до впровадження результатів розробки, приймемо 8500 користувачів;

Цб - вартість ліцензії програмного продукту у році до впровадження результатів розробки, приймемо 3500,00 грн;

±ΔЦо - зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 1200,00 грн.

Для кожного з випадків потенційне збільшення чистого прибутку у потенційного інвестора ΔПі в роки очікуваного позитивного результату розраховується за формулою::

$$\Delta П_i = (\pm \Delta Ц_o \times N + Ц_o \times N_i) \times \lambda \times \rho \times (1 - \frac{\vartheta}{100}) \quad (5.12)$$

де λ - коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт λ = 0,8333;

ρ - коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ = 40%;

ϑ - ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$.

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (1200 \times 8500 + 3500 \times 380) \times 0{,}8333 \times 0{,}45 \times (1 - \frac{0.18}{100}) = 4{,}380{,}920 \text{ грн}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (1200 \times 8500 + 3500 \times (380 + 620)) \times 0{,}8333 \times 0{,}45 \times (1 - \frac{0.18}{100}) = 5{,}920{,}400 \text{ грн}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (1200 \times 8500 + 3500 \times (380 + 620 + 850)) \times 0{,}8333 \times 0{,}45 \times (1 - \frac{0.18}{100}) = 11{,}893{,}750 \text{ грн}$$

Збільшення чистого прибутку 4-го року:

$$\Delta П_4 = (1200 \times 8500 + 3500 \times (380 + 620 + 850 + 1100)) \times 0{,}8333 \times 0{,}45 \times (1 - \frac{0.18}{100}) = 11{,}893{,}750 \text{ грн}$$

Приведена вартість потоків прибутку розраховується за формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \quad (5.13)$$

де ΔПі - збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т - період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ - ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}2$;

t - період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{4380920}{(1+0{,}2)^1} + \frac{5920400}{(1+0{,}2)^2} + \frac{7158680}{(1+0{,}2)^3} + \frac{8118620}{(1+0{,}2)^4} = 15{,}815{,}907 \text{ грн}$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Ключовими факторами, що визначають обґрунтованість інвестування певним інвестором у наукову розробку, є абсолютна та відносна ефективність інвестицій, а також термін їх повернення. Першим кроком на цьому шляху є розрахунок сучасної вартості інвестицій (PV), вкладених у наукову розробку.

Для цього можна використати формулу:

$$PV = k_{\text{інв}} \times \text{ЗВ} \quad (5.14)$$

де kінв - коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо kінв = 3;

ЗВ - загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 333648,4 грн.

$$PV = 3{,}5 \times 333648{,}4 \ = 1167769{,}3 \text{ грн}$$

Таким чином, чистий приведений дохід (NPV) або абсолютний економічний ефект (Еабс) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки буде таким:

$$\text{Е}_{\text{абс}} = \text{ПП} - PV \quad (5.15)$$

де ПП - приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 15 815 907 грн;

PV - теперішня вартість початкових інвестицій, 1,167,769,3 грн.

$$\text{Е}_{\text{абс}} = 15815907 - 1167769{,}3 \ = 14{,}648{,}137{,}47 \text{грн}$$

Внутрішня економічна дохідність (Ев) інвестицій, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, обчислюється за допомогою такої формули:

$$\text{Е}_{\text{в}} = \sqrt[\text{Т}_{\text{ж}}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1 \quad (5.16)$$

де Еабс - абсолютний економічний ефект вкладених інвестицій, 14,648,137,47 грн;

PV - теперішня вартість початкових інвестицій, 1 167 769,3 грн;

Тж - життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt{1 + \frac{14648137,7}{1167769,3}} - 1 = 0,9184 = 91,84\%$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій (τмін) визначається згідно такою формулою:

$$\tau_{мін} = d + f \quad (5.17)$$

де d - середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d = 0,18;

f - показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,30.

$$\tau_{мін} = 0,18 + 0,30 = 0,48$$

Оскільки Ев = 154,2% > τmin = 48%, то внутрішня економічна дохідність інвестицій перевищує мінімальну внутрішню дохідність. Таким чином, інвестування у науково-дослідну роботу є економічно обґрунтованим і доцільним.

Далі обчислюємо період окупності інвестицій (Ток або DPP, Discounted Payback Period), які потенційний інвестор може вкласти у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в} \quad (5.18)$$

$$T_{ок} = \frac{1}{0,9184} = 1,0889 \text{ років} = 13,1 \text{ місяців}$$

З огляду на те, що період окупності інвестицій становить менше восьми місяців, можна дійти висновку, що фінансування цієї розробки є виправданим.

5.5 Висновки до розділу

У ході виконання економічного аналізу було встановлено, що розроблена **система автоматизації обробки футбольної статистики** має високий комерційний потенціал. За результатами експертного оцінювання середньоарифметичний показник становить 44,7 балів, що відповідає категорії «високий рівень» і свідчить про конкурентоспроможність технології на ринку та реальну можливість її комерціалізації.

Прогнозовані витрати на розробку та впровадження становлять 333 648,4 грн, а теперішня вартість очікуваних інвестицій - 1 167 769,3 грн. При цьому приведена вартість майбутніх прибутків оцінюється у 15 815 907 грн, що забезпечує значний позитивний чистий приведений дохід у розмірі 14 648 137,7 грн. Такий результат однозначно демонструє економічну доцільність розробки.

Внутрішня економічна дохідність проєкту становить 91,84%, що перевищує мінімально допустиме значення 48%, розраховане з урахуванням депозитної ставки та ризиковості інвестицій. Крім того, період окупності становить близько 13 місяців, що також підтверджує привабливість інвестування у впровадження розробленої системи.

Отримані результати узгоджуються між собою й показують, що розробка має не лише технічну і наукову цінність, а й реальні перспективи практичного застосування. Вона здатна забезпечити істотний економічний ефект, адекватний період окупності й конкурентні переваги на ринку рішень у сфері спортивної аналітики та технологій обробки статистичних даних.

Таким чином, проведення науково-дослідної роботи є повністю обґрунтованим, економічно доцільним і перспективним з точки зору подальшої комерціалізації та впровадження у практичну діяльність аналітичних центрів, спортивних організацій та компаній, що працюють з великими масивами футбольних даних.

ВИСНОВКИ

Проаналізовано предметну область. Досліджено актуальність розробки веб-сайту, та проведено аналіз аналогів. Розглянуто усі типи, структури і різновиди веб-сайтів, для точнішого розуміння методів вирішення завдання.

Поставлено задачі для розробки веб-сайту та його функціонування, основними є: розробити зручну структуру, налаштувати зв’язок між UI та API, забезпечити можливість автентифікації та авторизації на сайті.

В результаті, виявлено доцільність розробки автоматизованої системи управління футбольною статистикою.

На основі запропонованої постановки задачі розроблено архітектуру системи, структуру веб-ресурсу, моделі, та алгоритми роботи з API та базою даних. Розроблена архітектура та алгоритм лягли в основу автоматизованої системи управління футбольною статистикою. Система була розроблена в середовищі Microsoft Visual Studio 2019, з використанням мови програмування C#, дизайн був оформлений за допомогою HTML 5 та CSS.

Тестування розробленої системи проходило в різних браузерах, була створена інструкція користувача. Тестування методом «чорного ящика» підтвердило ефективність системи. Було проведено тестування на валідність та кросбраузерність, що показала відмінну роботу системи, не залежно від вибору браузера.

В економічній частині показано, що автоматизована система управління статистичною інформацією є економічно вигідною і привабливою для інвесторів, тому має хороші перспективи для розповсюдження.

ПЕРЕЛІК ПОСИЛАНЬ

1. Mackenzie R., Cushion C. Performance Analysis in Football: A Critical Review and Implications for Future Research // *Journal of Sports Sciences*. 2013. Vol. 31, No. 6. P. 639–676.

2. Pappalardo L., Cintia P. PlayerRank: Data-Driven Performance Evaluation in Football // *PLoS ONE*. 2017. Vol. 12(8). P. 1–18.

3. Masse M. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces 1st Edition, Kindle Edition – 2012 - Vol. 59, no. 2. – P. 150–181.

4. Правила гри 2020/21 Міжнародна Рада футбольних асоціацій Munstergasse 9, 8001 Zurich, Switzerland [Електронний ресурс] – Режим доступу: www.theifab.com

5. Футбольна статистика [Електронний ресурс] – Режим доступу: https://sport.ua/football/results/ukraine/1/fixture

6. Турнірна таблиця УПЛ [Електронний ресурс] – Режим доступу: https://instatsport.com/football/statistical_reports

7. Індивідуальна статистика Андрія Шевченко [Електронний ресурс] – Режим доступу: https://www.transfermarkt.ru/andriy-shevchenko

8. Англійська прем'єр ліга – [Електронний ресурс] – Режим доступу: https://www.premierleague.com/about

9. Ла ліга – [Електронний ресурс] – Режим доступу: https://aficionesunidas.laliga.com/quienes-somos

10. Ліга Чемпіонів – [Електронний ресурс] – Режим доступу: https://ru.uefa.com/uefachampionsleague/

11. Бундесліга – [Електронний ресурс] – Режим доступу: https://www.bundesliga.com/en/bundesliga/awards/all/2020-2021

12. Романюк О.Н. «Організація баз даних і знань» / О.Н.Романюк, Т.О.Савчук. Навчальний посібник. – Вінниця: УНІВЕРСУМ-Вінниця, 2003. – 6 с.

13. Романюк О.Н., Катєльніков Д.І. Веб-дизайн і комп'ютерна графіка. / О.Н. Романюк, Д.І. Катєльніков. – Вінниця: ВНТУ, 2007. – 144 с.

14. Структура та різновиди сайтів» – Урок інформатики [Електронний ресурс] – Режим доступу: http://urokinformatiki.in.ua/urok-26-struktura-veb-sajtiv-riznovidi-veb-sajtiv/

15. HTML – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/Html/

16. CSS – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/Html/

17. PHP – [Електронний ресурс] – Режим доступу: http://php.net

18. Decroos T., Van Haaren J., Davis J., Raedt L. VAEP: An Objective Framework for Evaluating Actions in Football // *KDD Conference Proceedings.* 2019. P. 1–11.

19. Wordpress – [Електронний ресурс] – Режим доступу: https://wordpress.com/create/

20. Основні етапи веб-розробки – [Електронний ресурс] – Режим доступу: https://web-systems.solutions/blog/veb-rozrobka-etapy-i-standarty/

21. Проценко О.Б. Web-програмування та web-дизайн. Технологія XML: Навчальний посібник. – Суми: Видавництво СумДУ, 2009. – 166 с.

22. Stankov, S. Ontology as a Foundation for Knowledge Evaluation in Intelligent E-learning Systems / S. Stankov, B. Žitko, A. Grubišić // International Workshop on Applications of Semantic Web Technologies for E-Learning (SW-EL'05) in conjunction with 12th International Conference on Artificial Intelligence in Education (AI-ED 2005), Amsterdam, Netherlands. – 2005. – P. 81-84.

23. Goldman M., Rao J. M. Live Sports Analytics: Predicting Football Plays with Machine Learning // *Harvard Data Science Review.* 2020. P. 1–25.

24. Технічна підтримка сайтів – [Електронний ресурс] – Режим доступу: https://webkitchen.kiev.ua/ua/services/tekh-podderzhka

25. Інформація туру УПЛ – [Електронний ресурс] – Режим доступу: https://football24.ua/ru/ukraina_tables_tag50819/

26. football.db – [Електронний ресурс] – Режим доступу: https://www.footballdb.com/standings/index.html

27. football-data.co.uk – [Електронний ресурс] – Режим доступу: https://freebets.football-data.co.uk/

28. api-football – [Електронний ресурс] – Режим доступу: https://www.api-football.com/pricing

29. football-data.org – [Електронний ресурс] – Режим доступу: https://www.football-data.org/documentation/quickstart

30. Розробка структури сайту – [Електронний ресурс] – Режим доступу: https://webmaestro.com.ua/ua/struktura-sayta/

31. Дьяков О. В. Методи аналізу і прогнозування результатів футбольних матчів // *Науковий часопис НУФВСУ*. 2018. № 4. С. 33–39.

32. Огляд HTML 5 – [Електронний ресурс] – Режим доступу: https://www.pcreckoner.com/10-benefits-of-html5-html-overview/

33. Мова програмування C# – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction)

34. Jeffrey Richter, CLR via C#. Programming Microsoft.NET Framework 4.5 platform in the language C#: Peter. - 2016. - 896 p.

35. Andrew Troelsen, Philip Dzhepiks C# 6.0 programming language and platform .NET 4.6: Williams. - 2016 - 1440.

36. Технологія ASP.NET – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/overview

37. Технологія ASP.NET MVC – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0

38. REST архітектура – [Електронний ресурс] – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

39. Microsoft SQL Server Management Studio – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/sql/ssms/download-sql-servermanagement-studio-ssms?view=sql-server-ver15

40. Entity Framework – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/ef/core/get-started/overview/

41. AutoMapper – [Електронний ресурс] – Режим доступу: https://automapper.org/

42. Postman – [Електронний ресурс] – Режим доступу: https://www.postman.com/api-documentation-tool/

43. Identity Server 4 – [Електронний ресурс] – Режим доступу: https://identityserver4.readthedocs.io/en/latest/

44. Microsoft Visual Studio – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019

44. Microsoft Visual Studio – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019

45. Dependency injection (DI) – [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/aspnet5/6.1.php

46. Гончаренко О. М. Застосування інформаційних технологій у спортивній аналітиці // *Вісник Київського національного університету ім. Тараса Шевченка. Серія «Інформатика».* 2021. № 22. С. 18–25.

47. Тестування за допомогою методу чорного ящика – [Електронний ресурс] – Режим доступу: http://ru.qatestlab.com/services/no-documentation/black-box-testing/

48. Види тестування ПО – [Електронний ресурс] – Режим доступу: https://qalearning.com.ua/theory/lectures/material/testing-types-functional/

49. Методичні вказівки до виконання студентами-магістрантами наукового напрямку економічної частини магістерських кваліфікаційних робіт /Уклад. В. О. Козловський – Вінниця: ВНТУ, 2012. – 22с.

ДОДАТКИ

ДОДАТОК А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Автоматизація процесів збору та обробки футбольної статистики

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра КСУ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 14,85 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☒ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Ковтун В.В., завідувач кафедри КСУ _____
(прізвище, ініціали, посада) (підпис)

Ковалюк О.О., доцент кафедри КСУ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Дубовой В.М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Юхимчук М.С., професор кафедри КСУ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Самарський О.О.
(підпис) (прізвище, ініціали)

Додаток Б

Технічне завдання

ЗАТВЕРДЖУЮ

Завідувач кафедри КСУ

д.т.н.,проф. В'ячеслав КОВТУН

«17» жовтня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

««Автоматизація процесів збору та обробки футбольної статистики»»

08-31.МКР.006.02.000 ТЗ

Керівник роботи:

д.т.н., професор кафедри КСУ

Марія ЮХИМЧУК

" 16 " 10 2025р.

Виконавець: Олександр САМАРСЬКИЙ

" 16 " 10 2025р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

Назва – Розробка автоматизованої системи управління футбольною статистикою.

Галузь застосування – Комп'ютеризовані системи автоматизації управлінської діяльності.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ №313 від 24.09.2025 р.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення ефективності отримання, фільтрації та відображення спортивних подій за рахунок розроблення автоматизованої системи управління футбольною статистикою.

4. Джерела розробки

1. Шевчук В. П., Костенко С. І. Основи програмної інженерії: навчальний посібник. — Київ : Ліра-К, 2020. — 320 с.
2. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. — New York : McGraw-Hill, 2019. — 1376 p.
3. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. — Boston : Addison-Wesley, 1994. — 395 p.
4. Ляшенко В. І., Кравець Р. В. Штучний інтелект: методи і системи: навч. посіб. — Київ : Кондор, 2018. — 280 с.
5. Papandreou G., Alexiou A. Player tracking technologies in football: a review // Int. J. Sports Science & Coaching. — 2017. — № 12(3). — С. 345–358.

5. Показники призначення

5.1. Перелік головних функцій:

- управління футбольною статистикою.;
- управління документами та звітністю;
- управління списком контрагентів компанії;
- управління підрозділами компанії;

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 7\8\10;

- Версія ОС Android вище API 14: Android 4.0 (IceCreamSandwich).

5.2.2. Умови експлуатації системи:

- робота на стандартних ПЕОМ в приміщеннях зі стандартними умовами;
- можливість цілодобового функціонування системи;
- текст програмного забезпечення системи є цілком закритим.

6. Економічні показники

До економічних показників входять:

- витрати на розробку – до 250 тис. грн.
- узагальнений коефіцієнт якості розробки – більше 2-х
- термін окупності – до 3х років

7. Стадії розробки:

7.1 Пояснювальна записка:

- Аналіз методів управління футбольною статистикою. Постановка задач дослідження	04.09.2025р.
- Удосконалення технології управління футбольною статистикою	22.09.2025р.
- Практична реалізація та аналіз отриманих результатів	3.11.2025р.
- Підготовка економічної частини	12.11.2025р.
- Апробація результатів дослідження	20.11.2020р.
- Публікації	
- Оформлення пояснювальної записки, графічного матеріалу і презентації	08.12.2025р.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 10.12.2025.

2. Попередній захист магістерської кваліфікаційної роботи провести до 02.12.2025.

3. Захист магістерської кваліфікаційної роботи провести до 19.12.2025 р.

ДОДАТОК В (довідковий)

Лістинг програми

```
    public class Startup
    {
        public IConfiguration Configuration { get; }
        public Startup(IConfiguration configuration) => Configuration = configuration;
        // This method gets called by the runtime. Use this method to add services to the
container.
        public void ConfigureServices(IServiceCollection services)
        {
            Configuration.Bind("Project", new Config());
            services.AddDbContext<AppDBContext>(x =>
x.UseSqlServer(Config.ConnectionString));

            services.AddTransient<IRequest, CountryApi>();

            services.AddTransient<IRepositoryApi<Country, string>, CountryApi>();
            services.AddTransient<IRepositoryApi<Competition, string>, CompetitionApi>();
            services.AddTransient<IRepositoryApi<Team, string>, TeamApi>();
            services.AddTransient<IRepositoryApi<Standing, int>, StandingApi>();
            services.AddTransient<IRepositoryApi<Player, long>, PlayerApi>();
            services.AddTransient<IRepositoryApi<TeamWithPlayers, string>,
TeamWithPlayersApi>();
            services.AddTransient<IRepositoryApi<Event, string>, EventApi>();
            services.AddTransient<DataManager>();

            services.AddTransient<IRepositoryBd<Country, string>, CountryRepositoryBd>();
            services.AddTransient<IRepositoryBd<Competition, string>,
CompetitionRepositoryBd>();
            services.AddTransient<IRepositoryBd<Team, string>, TeamRepositoryBd>();
            services.AddTransient<IRepositoryBd<Standing, int>, StandingRepositoryBd>();
            services.AddTransient<IRepositoryBd<Player, long>, PlayerRepositoryBd>();
            services.AddTransient<IRepositoryBd<TeamWithPlayers, string>,
TeamWithPlayersRepositoryBd>();
            services.AddTransient<IRepositoryBd<Event, string>, EventRepositoryBd>();
            services.AddTransient<DataManagerBd>();
            services.AddControllersWithViews();

        }

        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                // The default HSTS value is 30 days. You may want to change this for
production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthorization();
```

```
            using (var scope = app.ApplicationServices.CreateScope())
            {
                AppDBContext context =
scope.ServiceProvider.GetRequiredService<AppDBContext>();

                SimpleData.Initialize(context);
            }


            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }

public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }

namespace FootballMVC.Models.Entities
{
    public class Team
    {
        [JsonPropertyName("team_key")]
        [Key]
        public string Id { get; set; }

        [JsonPropertyName("team_name")]
        [Display(Name = "Ім'я")]
        public string Name { get; set; }

        [JsonPropertyName("players")]
        public List<Player> Players { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace FootballMVC.Models.Entities
{
    public class Standing
    {

      public int Id { get; set; }
```

```
        [JsonPropertyName("team_name")]
        [Display(Name = "Назва")]
        public string Team_Name { get; set; }

        [JsonPropertyName("overall_league_position")]
        [Display(Name = "Позиція в таблиці")]
        public string LeaguePosition { get; set; }

        [JsonPropertyName("overall_league_payed")]
        [Display(Name = "Кількість зіграних матчів")]
        public string League_payed { get; set; }

        [JsonPropertyName("overall_league_W")]
        [Display(Name = "Кількість виграних матчів")]
        public string League_W { get; set; }

        [JsonPropertyName("overall_league_D")]
        [Display(Name = "Кількість нічіїх")]
        public string League_D { get; set; }

        [JsonPropertyName("overall_league_L")]
        [Display(Name = "Кількість програних матчів")]
        public string League_L { get; set; }

        [JsonPropertyName("overall_league_PTS")]
        [Display(Name = "Кількість очків")]
        public string League_PTS { get; set; }

          public Competition Competition { get; set; }

          [JsonPropertyName("league_id")]
          public string Competition_Id { get; set; }

          public Team Team { get; set; }

          [JsonPropertyName("team_id")]
          public string Team_id { get; set; }
    }
}
public class Country
    {
        [JsonPropertyName("country_id")]

        public string Id { get; set; }

        [Display(Name = "Назва країни")]
        [JsonPropertyName("country_name")]
        public string Name { get; set; }

        public List<Competition> Competitions { get; set; }

    }

        [JsonPropertyName("league_id")]

        public string Id { get; set; }

        [JsonPropertyName("league_name")]
        [Display(Name = "Назва")]
        public string Name { get; set; }

        [JsonPropertyName("country_id")]
        public string CountryId { get; set; }
        public string leag_logo { get; set; }
        public string country_flag { get; set; }
```

```
        public string country_name { get; set; }
        public Country Country { get; set; }

}

public class TeamWithPlayers
    {
        [JsonPropertyName("team_key")]
        [Key]
        public string Id { get; set; }

        [JsonPropertyName("team_name")]
        [Display(Name = "Ім'я")]
        public string Name { get; set; }

        [JsonPropertyName("players")]
        public List<Player> Players { get; set; }
    }

public class HeadToH
    {
       public List<Match> firstTeam_VS_secondTeam { get; set; }
       public List<Match> firstTeam_lastResults { get; set; }
       public List<Match> secondTeam_lastResults { get; set; }
    }
    public class Match
    {
        public string match_id { get; set; }
        public string country_id { get; set; }
        public string country_name { get; set; }
        public string league_id { get; set; }
        public string league_name { get; set; }
        public string match_date { get; set; }
        public string match_status { get; set; }
        public string match_time { get; set; }
        public string match_hometeam_id { get; set; }
        public string match_hometeam_name { get; set; }
        public string match_hometeam_score { get; set; }
        public string match_awayteam_id { get; set; }
        public string match_awayteam_name { get; set; }
        public string match_awayteam_score { get; set; }
        public string match_hometeam_halftime_score { get; set; }
        public string match_awayteam_halftime_score { get; set; }
        public string match_live { get; set; }
        public string team_home_badge { get; set; }
        public string team_away_badge { get; set; }
        public string league_logo { get; set; }
        public string country_logo { get; set; }

        public Competition competition { get; set; }

        Country country { get; set; }
    }

public class Event
    {
        public string id { get; set; }

        public string country_name { get; set; }

        public string league_name { get; set; }

        public string match_date { get; set; }

        public string match_status { get; set; }
```

```
public string match_time { get; set; }

public string match_hometeam_id { get; set; }

public string match_hometeam_name { get; set; }

public string match_hometeam_score { get; set; }

public string match_awayteam_name { get; set; }

public string match_awayteam_id { get; set; }

public string match_awayteam_score { get; set; }

public string match_hometeam_halftime_score { get; set; }

public string match_awayteam_halftime_score { get; set; }

public string match_hometeam_extra_score { get; set; }

public string match_awayteam_extra_score { get; set; }

public string match_hometeam_penalty_score { get; set; }

public string match_awayteam_penalty_score { get; set; }

public string match_hometeam_ft_score { get; set; }

public string match_awayteam_ft_score { get; set; }

public string match_hometeam_system { get; set; }

public string match_awayteam_system { get; set; }

public string match_live { get; set; }

public string match_round { get; set; }

public string match_stadium { get; set; }

public string match_referee { get; set; }

public string team_home_badge { get; set; }

public string team_away_badge { get; set; }

public string league_logo { get; set; }

public string country_logo { get; set; }

public List<Goalscorer> goalscorer { get; set; }
public List<Cards> cards { get; set; }
public Substitutions substitutions { get; set; }
public Lineup lineup { get; set; }
public List<Statistics> statistics { get; set; }


//create
public MatchBd Match { get; set; }
public string match_id { get; set; }

public CountryBd Country { get; set; }
public string country_id { get; set; }
```

```
        public CompetitionBd Competition { get; set; }
        public string league_id { get; set; }
        public TeamBd Team { get; set; }

    }
   public class Substitutions
    {
        public List<Home> home { get; set; }

        public List<Away> away { get; set; }
    }

    public class Home
    {
        public string time { get; set; }
        public string substitution { get; set; }
    }
    public class Away
    {
        public string time { get; set; }
        public string substitution { get; set; }
    }
public class Goalscorer
    {
        public string time { get; set; }
        public string home_scorer { get; set; }
        public string home_scorer_id { get; set; }
        public string home_assist { get; set; }
        public string home_assist_id { get; set; }
        public string score { get; set; }
        public string away_scorer { get; set; }
        public string away_scorer_id { get; set; }
        public string away_assist { get; set; }
        public string away_assist_id { get; set; }
        public string info { get; set; }
}
public class Cards
    {
        public string time { get; set; }
        public string home_fault { get; set; }
        public string card { get; set; }
        public string away_fault { get; set; }
        public string info { get; set; }
}
public class Statistics
    {
        public string type { get; set; }
        public string home { get; set; }
        public string away { get; set; }
    }
  public class Lineup
    {
        public Info home { get; set; }

        public Info away { get; set; }
    }

    public class Info
    {
        public List<LineUps> starting_lineups { get; set; }
        public List<LineUps> substitutes { get; set; }
        public List<LineUps> coach { get; set; }
        public List<LineUps> missing_players { get; set; }

    }
```

```
    public class LineUps
    {
        string lineup_player { get; set; }
        string lineup_number { get; set; }
        string lineup_position { get; set; }
        string player_key { get; set; }
    }
public class ErrorMessage
    {
        [JsonPropertyName("error")]
        public int error { get; set; }
        [JsonPropertyName("message")]
        public string  message { get; set; }
    }

public class DataManager
    {
        public IRepositoryApi<Country, string> CountryRepositoryApi { get; set; }
        public IRepositoryApi<Competition, string> CompetitionRepositoryApi { get; set; }
        public IRepositoryApi<Standing, int> StandingRepositoryApi { get; set; }
        public IRepositoryApi<Player, long> PlayerRepositoryApi { get; set; }
        public IRepositoryApi<Team, string> TeamRepositoryApi { get; set; }
        public IRepositoryApi<TeamWithPlayers, string> TeamWithPRepositoryApi { get; set; }
        public IRepositoryApi<Event, string> EventApi { get; set; }
        public IRequest Country { get; set; }

        public DataManager(IRepositoryApi<Country, string> countryRepositoryApi,
IRepositoryApi<Competition, string> competitionRepositoryApi,
            IRepositoryApi<Standing, int> standingRepositoryApi, IRepositoryApi<Player,
long>  playerRepositoryApi,
             IRepositoryApi<Team, string> teamRepositoryApi, IRepositoryApi<TeamWithPlayers,
string> teamWithPRepositoryApi,
           IRepositoryApi<Event, string> eventApi, IRequest country)
        {
            CountryRepositoryApi = countryRepositoryApi;
            CompetitionRepositoryApi = competitionRepositoryApi;
            StandingRepositoryApi = standingRepositoryApi;
            PlayerRepositoryApi = playerRepositoryApi;
            TeamRepositoryApi = teamRepositoryApi;
            TeamWithPRepositoryApi = teamWithPRepositoryApi;
            EventApi = eventApi;
            Country = country;
        }

    }
namespace FootballMVC.Data.DataBase
{
    public class AppDBContext : DbContext
    {
        public DbSet<Player> Players { get; set; }
        public DbSet<Country> Countries { get; set; }
        public DbSet<Competition> Competitions { get; set; }
        public DbSet<Team> Teams { get; set; }
        public DbSet<Standing> Standings { get; set; }
        public DbSet<Event> Events { get; set; }
        public DbSet<HeadToH> HeadToHs { get; set; }
        public DbSet<Substitutions> Substitutions { get; set; }
        public DbSet<Goalscorer> Goalscorers { get; set; }
        public DbSet<Cards> Cards { get; set; }
        public DbSet<Lineup> Lineups { get; set; }
        public DbSet<Statistics> Statistics { get; set; }

        public AppDBContext(DbContextOptions<AppDBContext> options) : base(options)
        {
            Database.EnsureCreated();
```

```
        }
    }
using FootballMVC.Data.BdData.Interfaces;
using FootballMVC.Data.DataBase;
using FootballMVC.Models.Entities;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace FootballMVC.Data.BdData.Repositories
{
    public class TeamRepositoryBd : IRepositoryBd<Team, string>
    {
        AppDBContext _context;
        public TeamRepositoryBd(AppDBContext context)
        {
            _context = context;
        }
        public bool EntityExists(string id)
        {
            return _context.Teams.Any(e => e.Id == id);
        }

        public async Task<Team> GetEntityItems(string id)
        {
            return await _context.Teams
                  .FirstOrDefaultAsync(m => m.Id == id);

        }

        public async Task<List<Team>> GetEntityListItems()
        {
            return await _context.Teams.ToListAsync();

        }

        public Task<List<Team>> GetEntityListItemsByKey()
        {
            throw new NotImplementedException();
        }

        public IQueryable<Team> GetEntityNoAsyncListItems()
        {
            return _context.Teams;
        }

        public async Task RemoveEntity(Team entity)
        {
            _context.Teams.Remove(entity);
            await _context.SaveChangesAsync();

        }

        public async Task SaveEntity(Team entity)
        {
            _context.Teams.Add(entity);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateEntity(Team entity)
        {
            _context.Teams.Update(entity);
            await _context.SaveChangesAsync();
```

```
        }
    }
}
using FootballMVC.Data.BdData.Interfaces;
using FootballMVC.Data.DataBase;
using FootballMVC.Models.Entities;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace FootballMVC.Data.BdData.Repositories
{
    public class StandingRepositoryBd : IRepositoryBd<Standing, int>
    {
        AppDBContext _context;
        public StandingRepositoryBd(AppDBContext context)
        {
            _context = context;
        }

        public bool EntityExists(int id)
        {
            return _context.Players.Any(e => e.Id == id);
        }

        public async Task<Standing> GetEntityItems(int id)
        {
            return await _context.Standings
                   .FirstOrDefaultAsync(m => m.Id == id);
        }

        public async Task<List<Standing>> GetEntityListItems()
        {
            return await _context.Standings.ToListAsync();
        }

        public Task<List<Standing>> GetEntityListItemsByKey()
        {
            throw new NotImplementedException();
        }

        public IQueryable<Standing> GetEntityNoAsyncListItems()
        {
            return _context.Standings;
        }

        public async Task RemoveEntity(Standing entity)
        {
            _context.Standings.Remove(entity);
            await _context.SaveChangesAsync();
        }

        public async Task SaveEntity(Standing entity)
        {
            _context.Standings.Add(entity);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateEntity(Standing entity)
        {
            _context.Standings.Update(entity);
            await _context.SaveChangesAsync();
        }
```

```
    }
}
using FootballMVC.Data.BdData.Interfaces;
using FootballMVC.Data.DataBase;
using FootballMVC.Models.Entities;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace FootballMVC.Data.BdData.Repositories
{
    public class PlayerRepositoryBd : IRepositoryBd<Player, long>
    {
        AppDBContext _context;
        public PlayerRepositoryBd(AppDBContext context)
        {
            _context = context;
        }
        public bool EntityExists(long id)
        {
            return _context.Players.Any(e => e.Id == id);
        }

        public async Task<Player> GetEntityItems(long id)
        {
            return await _context.Players.Include(p => p.Team)
                  .FirstOrDefaultAsync(m => m.Id == id);
        }

        public async Task<List<Player>> GetEntityListItems()
        {
            return await _context.Players.ToListAsync();
        }

        public async Task<List<Player>> GetEntityListItemsByKey()
        {
            return await _context.Players.Include(c => c.Team).ToListAsync();
        }

        public IQueryable<Player> GetEntityNoAsyncListItems()
        {
            return _context.Players;
        }
        public async Task RemoveEntity(Player entity)
        {
            _context.Players.Remove(entity);
            await _context.SaveChangesAsync();
        }

        public async Task SaveEntity(Player entity)
        {
            _context.Players.Add(entity);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateEntity(Player entity)
        {
            _context.Players.Update(entity);
            await _context.SaveChangesAsync();
        }
    }}
using FootballMVC.Data.BdData.Interfaces;
using FootballMVC.Data.DataBase;
```

```
using FootballMVC.Models.Entities;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace FootballMVC.Data.BdData.Repositories
{
    public class CompetitionRepositoryBd : IRepositoryBd<Competition, string>
    {
        AppDBContext _context;
        public CompetitionRepositoryBd(AppDBContext context)
        {
            _context = context;
        }
        public bool EntityExists(string id)
        {
            return _context.Competitions.Any(e => e.Id == id);
        }

        public async Task<Competition> GetEntityItems(string id)
        {
            return await _context.Competitions.Include(c => c.Country)
                 .FirstOrDefaultAsync(m => m.Id == id);
        }

        public async Task<List<Competition>> GetEntityListItemsByKey()
        {
            return await _context.Competitions.Include(c => c.Country).ToListAsync();
        }
        public async Task<List<Competition>> GetEntityListItems()
        {
            return await _context.Competitions.ToListAsync();
        }

        public async Task RemoveEntity(Competition entity)
        {
            _context.Competitions.Remove(entity);
            await _context.SaveChangesAsync();
        }

        public async Task SaveEntity(Competition entity)
        {
            _context.Competitions.Add(entity);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateEntity(Competition entity)
        {
            _context.Competitions.Update(entity);
            await _context.SaveChangesAsync();
        }

        public IQueryable<Competition> GetEntityNoAsyncListItems()
        {
            return _context.Competitions;
        }
    }
}
  public interface IRepositoryBd<T,K>
    {
        bool EntityExists(K id);
        Task<List<T>> GetEntityListItems();
        Task<List<T>> GetEntityListItemsByKey();
```

```
        Task<T> GetEntityItems(K id);
        Task SaveEntity(T entity);
        Task RemoveEntity(T entity);
        Task UpdateEntity(T entity);
        IQueryable<T> GetEntityNoAsyncListItems();
    }
using FootballMVC.Data.DataBase;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;

namespace FootballMVC.Data.ApiData.Interfaces
{

        public interface IRepositoryApi<T,K>
        {
            Task<T> GetEntityAsync(string path, HttpClient client);
            Task<List<T>> GetListEntityAsync(string path, HttpClient client);
            List<T> SaveAllToDateBase(List<T> elements);
            IQueryable<T> GetEntityItems();
            T GetEntityItemById(K id);
            Task SaveEntity(T entity);


    }
} using FootballMVC.Data.DataBase;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using FootballMVC.Configuration;
using FootballMVC.Models.Entities;
using FootballMVC.Models.Entities.Events;
using FootballMVC.Models.Entities.TopLeagsWiewModels;

namespace FootballMVC.Controllers.Main
{
    public class MainController : Controller
    {

        private static HttpClient client = new HttpClient();
        static List<Country> countries = new List<Country>();
        static List<Competition> competitions = new List<Competition>();
        static List<Standing> standings = new List<Standing>();

        private readonly DataManager dataManager;
        public MainController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        public async Task<IActionResult> Index()
        {


            return View();
        }
        public async Task<IActionResult> GetAllCountries ()
        {
            string defolt = dataManager.Country.CreateRequest();
```

```
            countries = await dataManager.CountryRepositoryApi.GetListEntityAsync(defolt,
client);
            return View(countries);
        }
        public async Task<IActionResult> GetListCompetitionsByCountry(string id)
        {
            if (id == null)
            {
                return NotFound();
            }
            string defolt = APIConfig.URI + "get_leagues" + APIConfig.APIkey +
"&country_id=" + id;

            competitions = await
dataManager.CompetitionRepositoryApi.GetListEntityAsync(defolt, client);
            return View(competitions);
        }
        public async Task<IActionResult>CompetitionsInfo(string id)
        {
            if (id == null)
            {
                return NotFound();
            }
            var competition = competitions.Where(x => x.Id == id).FirstOrDefault();

            string defolt = APIConfig.URI + "get_events&from=2020-11-0&to=2020-11-29" +
APIConfig.APIkey + "&league_id=" + competition.Id;
            List<Event> events =  await dataManager.EventApi.GetListEntityAsync(defolt,
client);
            var ev  = events.FirstOrDefault();
            competition.leag_logo = ev.league_logo;
            competition.country_flag = ev.country_logo;
            competition.country_name = ev.country_name;

            return View(competition);
        }
        public async Task<IActionResult> ViewStanding(string id)
        {

            if (id == null)
            {
                return NotFound();
            }
            string defolt = APIConfig.URI +  "get_standings" + APIConfig.APIkey +
"&league_id=" + id;

            standings = await dataManager.StandingRepositoryApi.GetListEntityAsync(defolt,
client);
            if (standings == null)
            {
                return NotFound();
            }
            return View (standings);
        }
        public async Task<IActionResult> LastResults()
        {

             DateTime dateTime =  DateTime.Now;

            string defolt = APIConfig.URI + "get_events" + ParseDateTimeToString(dateTime) +
APIConfig.APIkey + "&league_id=";

            var apldefolt = defolt  + TopLeagsInfo.APLId;
            var LCHdefolt = defolt + TopLeagsInfo.LigaChempId;
            var Ledefolt = defolt + TopLeagsInfo.LigaEuropeId;
```

```
            var Lalidefolt = defolt + TopLeagsInfo.LaLigaId;
            var seriadefolt = defolt + TopLeagsInfo.SeriaAId;
            var Bunddefolt = defolt + TopLeagsInfo.BundesId;
            var Liga1defolt = defolt + TopLeagsInfo.Liga1Id;
            var Upldefolt = defolt + TopLeagsInfo.UPLId;
            TopLeagViewModel result = new TopLeagViewModel();
            result.TopleagsEvent = new List<List<Event>>();
            result.TopleagsEvent.Add(await
dataManager.EventApi.GetListEntityAsync(apldefolt, client));
            result.TopleagsEvent.Add(await
dataManager.EventApi.GetListEntityAsync(LCHdefolt, client));
            result.TopleagsEvent.Add(await dataManager.EventApi.GetListEntityAsync(Ledefolt,
client));
            result.TopleagsEvent.Add(await
dataManager.EventApi.GetListEntityAsync(Lalidefolt, client));
            result.TopleagsEvent.Add(await
dataManager.EventApi.GetListEntityAsync(Bunddefolt, client));
            result.TopleagsEvent.Add(await
dataManager.EventApi.GetListEntityAsync(Liga1defolt, client));
            result.TopleagsEvent.Add(await
dataManager.EventApi.GetListEntityAsync(seriadefolt, client));
            result.TopleagsEvent.Add(await
dataManager.EventApi.GetListEntityAsync(Upldefolt, client));



            return View(result);
        }

        public async Task<IActionResult> DetailsMatch(string id)
        {

            if (id == null)
            {
                return NotFound();
            }
            DateTime dateTime = DateTime.Now;
            string match_id = GetMatchId(id);
            string date = GetDate(id);
            string defolt = APIConfig.URI + "get_events" + date + APIConfig.APIkey +
"&match_id=" + match_id;

            List<Event> events = await dataManager.EventApi.GetListEntityAsync(defolt,
client);

            if (standings == null)
            {
                return NotFound();
            }
            return View(events);
        }

        private string ParseDateTimeToString(DateTime dateTime)
        {
            string result = "&from=";
            if (dateTime != null)
            {
                string date = dateTime.Year.ToString() + "-" + dateTime.Month.ToString() +
"-" + dateTime.Day.ToString();

                result += date + "&to=" + date;
                return result;
            }
            return result;
        }
```

```
        private string ParseDateTimeToString(DateTime fromDateTime, DateTime toDateTime)
        {
            string from = "&from=";
            string to = "&to=";
            string result = "";
            if (fromDateTime != null && toDateTime != null )
            {
                string fromDate = fromDateTime.Year.ToString() + "-" +
fromDateTime.Month.ToString() + "-" + fromDateTime.Day.ToString();
                string toDate = toDateTime.Year.ToString() + "-" +
toDateTime.Month.ToString() + "-" + toDateTime.Day.ToString();

                result += from + fromDate + to + toDate;
                return result;
            }
            return result;
        }
        private string GetMatchId(string id)
        {
            string[] subs = id.Split('&');

            return subs[0];
        }
        private string GetDate(string id)
        {
            string[] subs = id.Split('&');
            string result = "&from=" + subs[1] + "&to=" + subs[1];

            return result;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using FootballMVC.Data.ApiData;
using FootballMVC.Data.ApiData.Repositories;
using FootballMVC.Data.DataBase;
using FootballMVC.Models.Entities;
using Microsoft.AspNetCore.Mvc;

namespace FootballMVC.Controllers.ApiControllers
{
    public class StandingApiController : Controller
    {
        private static HttpClient client = new HttpClient();

        static List<Standing> standings = new List<Standing>();

        private readonly DataManager dataManager;
        public StandingApiController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        public async Task<IActionResult> Index()
        {

            string defolt =
"https://apiv2.apifootball.com?action=get_standings&league_id=148&APIkey=345ebb1d5cb902a9de9
280564d2c2467de4d55af83f5ee1b7b25c4591ebb078e";
            List<Standing>  standings = await
dataManager.StandingRepositoryApi.GetListEntityAsync(defolt, client);
```

```
            return View(standings);
        }
        public async Task<IActionResult> ViewStandingByLeagID(string id)
        {

            if (id == null)
            {
                return NotFound();
            }
            string defolt =
"https://apiv2.apifootball.com?action=get_standings&APIkey=345ebb1d5cb902a9de9280564d2c2467d
e4d55af83f5ee1b7b25c4591ebb078e&league_id=";
            defolt += id;

            standings = await dataManager.StandingRepositoryApi.GetListEntityAsync(defolt,
client);
            if (standings == null)
            {
                return NotFound();
            }
            return View(standings);
        }
        public async Task<IActionResult> SaveAllToDateBase(string id)
        {
            ViewData["Answer"] = "Збережено";

            if (dataManager.StandingRepositoryApi.GetEntityItems().
                Where(p => p.Competition_Id == id).Count() != 0)
            {
                List<Standing> stan =
dataManager.StandingRepositoryApi.SaveAllToDateBase(standings);
                if (stan.Count() == 0)
                {
                    ViewData["Answer"] = "Турнірна таблиця вже була збережена в базі даних";
                    return View(standings[0]);
                }
                foreach (Standing s in stan)
                {
                    await dataManager.StandingRepositoryApi.SaveEntity(s);
                }
                return View(standings[0]);
            }
            foreach (Standing s in standings)
            {
                await dataManager.StandingRepositoryApi.SaveEntity(s);
            }
            return View(standings[0]);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using FootballMVC.Data.ApiData;
using FootballMVC.Data.ApiData.Repositories;
using FootballMVC.Data.ApiData.Repositories.PlayersInTeam;
using FootballMVC.Data.DataBase;
using FootballMVC.Models.Entities;
using FootballMVC.Models.Entities.PlayersInTeam;
using Microsoft.AspNetCore.Mvc;

namespace FootballMVC.Controllers.ApiControllers
{
```

```
    public class PlayerApiController : Controller
    {
        private static HttpClient client = new HttpClient();

        static List<Player> players = new List<Player>();
        static string TeamId;

        private readonly DataManager dataManager;
        public PlayerApiController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        public async Task<IActionResult> Index()
        {
            string defolt =
"https://apiv2.apifootball.com?action=get_players&player_id=3183500916&APIkey=345ebb1d5cb902
a9de9280564d2c2467de4d55af83f5ee1b7b25c4591ebb078e";
            return View(await dataManager.PlayerRepositoryApi.GetEntityAsync(defolt,
client));
        }

        public async Task<IActionResult> ViewPlayersByTeamId(string id)
        {
            TeamId = id;
            ViewBag.Id = TeamId;
            if (id == null)
            {
                return NotFound();
            }
            string defolt =
"https://apiv2.apifootball.com?action=get_teams&APIkey=345ebb1d5cb902a9de9280564d2c2467de4d5
5af83f5ee1b7b25c4591ebb078e&team_id=";
            defolt += id;
            TeamWithPlayers team = await
dataManager.TeamWithPRepositoryApi.GetEntityAsync(defolt, client);
            foreach (Player p in team.Players)
            {
                p.TeamId = team.Id;
                players.Add(p);
            }
            return View(team.Players.ToList());
        }
        public async Task<IActionResult> SaveAllToDateBase()
        {
            ViewBag.Id = TeamId;
            ViewData["Answer"] = "Збережено";
            List<Player> play = dataManager.PlayerRepositoryApi.SaveAllToDateBase(players);
            if (play.Count() == 0)
            {
                ViewData["Answer"] = "Гравці вже були збережені в базі даних";
                return View();
            }
            foreach (Player p in play)
            {
                await dataManager.PlayerRepositoryApi.SaveEntity(p);
            }
            return View();
        }




    }
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace FootballMVC.Configuration
{
    public static class StatisticsType
    {
        public const string Ball_Possession = "Ball Possession";
        public const string Goal_Attempts = "Goal Attempts";
        public const string Shots_on_Goal = "Shots on Goal";
        public const string Shots_off_Goal = "Shots off Goal";
        public const string Blocked_Shots = "Blocked Shots";
        public const string Corner_Kicks = "Corner Kicks";
        public const string Offsides = "Offsides";
        public const string Goalkeeper_Saves = "Goalkeeper Saves";
        public const string Fouls = "Fouls";
        public const string Yellow_Cards = "Yellow Cards";
        public const string Attacks = "Attacks";
        public const string Dangerous_Attacks = "Dangerous Attacks";

    }
    public static class APIConfig
    {
        public const string URI = "https://apiv2.apifootball.com/?action=";
        public const string APIkey =
"&APIkey=557a4c8dd12d182ad230f3683aecd88058a44d6487da02cb5516a1f57114b1c4";
        public const string GetCountries = "get_countries";
    }
    public static class TopLeagsInfo
    {
        public const string APLId = "148";
        public const string Liga1Id = "176";
        public const string UPLId = "523";
        public const string LigaChempId = "589";
        public const string LigaEuropeId = "590";
        public const string LaLigaId = "468";
        public const string SeriaAId = "262";
        public const string BundesId = "262";

        public static List<string> GetTopLigsId()
        {
            List<string> ids = new List<string>()
            {
                "148", "176", "523", "589", "590", "468",  "262",
            };
            return ids;
        }
    }

}
@model IEnumerable<FootballMVC.Models.Entities.Standing>

<link rel="stylesheet" href="~/css/onlymy.css" />

<table class="table_blur">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.LeaguePosition)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Team_Name)
            </th>
            <th>
```

```
                @Html.DisplayNameFor(model => model.League_payed)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.League_W)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.League_D)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.League_L)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.League_PTS)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.LeaguePosition)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Team_Name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.League_payed)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.League_W)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.League_D)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.League_L)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.League_PTS)
                </td>
            </tr>
        }
    </tbody>
</table>
@model FootballMVC.Models.Entities.TopLeagsWiewModels.TopLeagViewModel

<link rel="stylesheet" href="~/css/onlymy.css" />
@{
    var matchInfo = new FootballMVC.Models.ViewComponents.MatchSimple();
}

@foreach (var items in Model.TopleagsEvent)
{
    bool HaveMatch = items.Count == 1 && items.Any(x => x.league_id == "0");
    if (!HaveMatch)
    {

        <table class="table_blur" width="800px">
            <caption style="font-size: 18px;"> @items.FirstOrDefault().league_name
</caption>
            <tbody>
                @foreach (var item in items)
                {
```

```
                    <tr>
                        <td>
                            <a asp-controller="Main" asp-action="DetailsMatch" asp-route-
id="@matchInfo.matchAndTime(item.match_id, item.match_date)">  @Html.DisplayFor(modelItem =>
item.match_time)</a>
                        </td>
                        <td>
                            <img src=@Html.DisplayFor(modelItem => item.team_home_badge)
width="20" height="20">
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.match_hometeam_name)
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.match_hometeam_score)
                        </td>
                        <td>
                            :
                        </td>
                        @*away*@
                        <td>
                            @Html.DisplayFor(modelItem => item.match_awayteam_score)
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.match_awayteam_name)
                        </td>
                        <td>
                            <img src=@Html.DisplayFor(modelItem => item.team_away_badge)
width="20" height="20">
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    }
}

@model IEnumerable<FootballMVC.Models.Entities.Competition>

<link rel="stylesheet" href="~/css/onlymy.css" />


<ul class="category-list-competitions clearfix">

    @foreach (var item in Model)
    {
        <li class="category-item"> <a asp-controller="Main" asp-action="CompetitionsInfo"
asp-route-id="@item.Id"> @Html.DisplayFor(modelItem => item.Name)</a></li>
    }
</ul>
@model IEnumerable<FootballMVC.Models.Entities.Country>

    <link rel="stylesheet" href="~/css/onlymy.css" />

    <ul class="category-list-countries clearfix">

            @foreach (var item in Model)
            {
                <li class="category-item"> <a asp-controller="Main" asp-
action="GetListCompetitionsByCountry" asp-route-id="@item.Id"> @Html.DisplayFor(modelItem =>
item.Name)</a></li>
            }
    </ul>
@model  IEnumerable<FootballMVC.Models.Entities.Events.Event>
```

```
@{
    var statistics = Model.FirstOrDefault().statistics;
    var lineups = Model.FirstOrDefault().lineup;
}
<div>
    <div>

        @foreach (var item in Model)
        {
        <table class="table_blur" width="800px">
            <caption style="font-size: 18px;"> @item.league_name </caption>
            <tbody>
                <tr>
                    <td>
                        <img src=@Html.DisplayFor(modelItem => item.team_home_badge)
width="20" height="20">
                    </td>
                    <td style="text-align:left">
                        @Html.DisplayFor(modelItem => item.match_hometeam_name)
                    </td>
                    <td style="text-align:left">
                        @Html.DisplayFor(modelItem => item.match_hometeam_score)
                    </td>
                    <td style="text-align: right">
                        @Html.DisplayFor(modelItem => item.match_awayteam_score)
                    </td>
                    <td style="text-align: right">
                        @Html.DisplayFor(modelItem => item.match_awayteam_name)

                    </td>
                    <td style="text-align: right">
                        <img src=@Html.DisplayFor(modelItem => item.team_away_badge)
width="20" height="20">
                    </td>
                </tr>
            </tbody>
        </table>
        }
    </div>
    <div>
        @foreach (var stat in statistics)
        {
        <table class="table_blur" width="800px">
            <caption style="font-size: 18px;"> </caption>
            <tbody>
                <tr>
                    <td style="text-align: left">
                        @Html.DisplayFor(modelItem => stat.home)
                    </td>
                    <td style="text-align: center">
                        @Html.DisplayFor(modelItem => stat.type)
                    </td>
                    <td style="text-align: right">
                        @Html.DisplayFor(modelItem => stat.away)
                    </td>
                </tr>
            </tbody>
        </table>
        }
    </div>
</div>
```

ДОДАТОК Г (обов’язковий)

Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

АЛГОРИТМИ АДАПТИВНОГО ПЛАНУВАННЯ ДЛЯ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ ПРОЦЕСІВ. ЧАСТИНА 1. АНАЛІЗ ВИМОГ І РОЗРОБКА ІНФОРМАЦІЙНОЇ СТРУКТУРИ.

Перелік ілюстративних матеріалів:

Слайд 1 - Титульний слайд

Слайд 2 - Актуальність дослідження

Слайд 3 – Мета досліджень

Слайд 4 – Предмет, об’єкт та сутності системи

Слайд 5 - Джерело даних та структура API

Слайд 6 - Архітектура системи

Слайд 7 - Розробка структури веб-ресурсу

Слайд 8 - Алгоритми та обробка статистики

Слайд 9 - Методи дослідження та наукова новизна

Слайд 10 - Тестування системи

Слайд 11 - Діаграма класів для взаємодії з базою даних

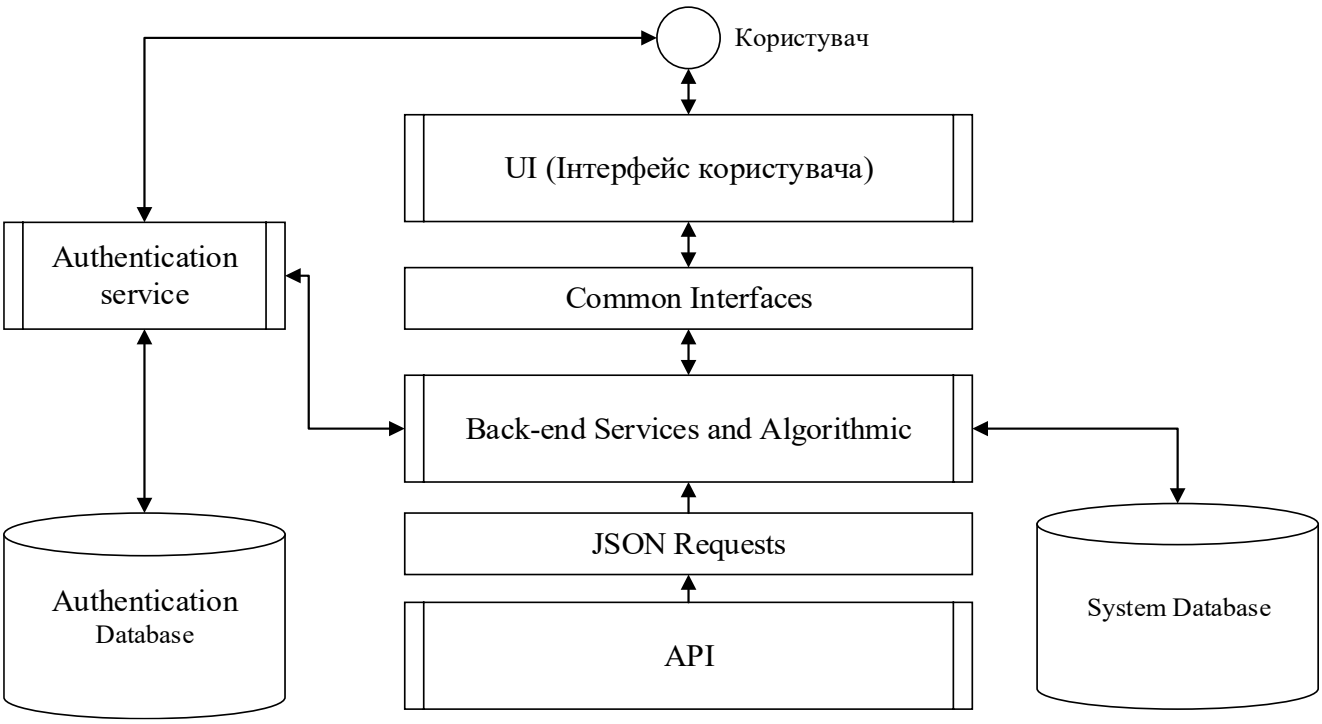
Слайд 12 - Діаграма класів для управління API

Слайд 13 - Діаграма класів моделей предметної області

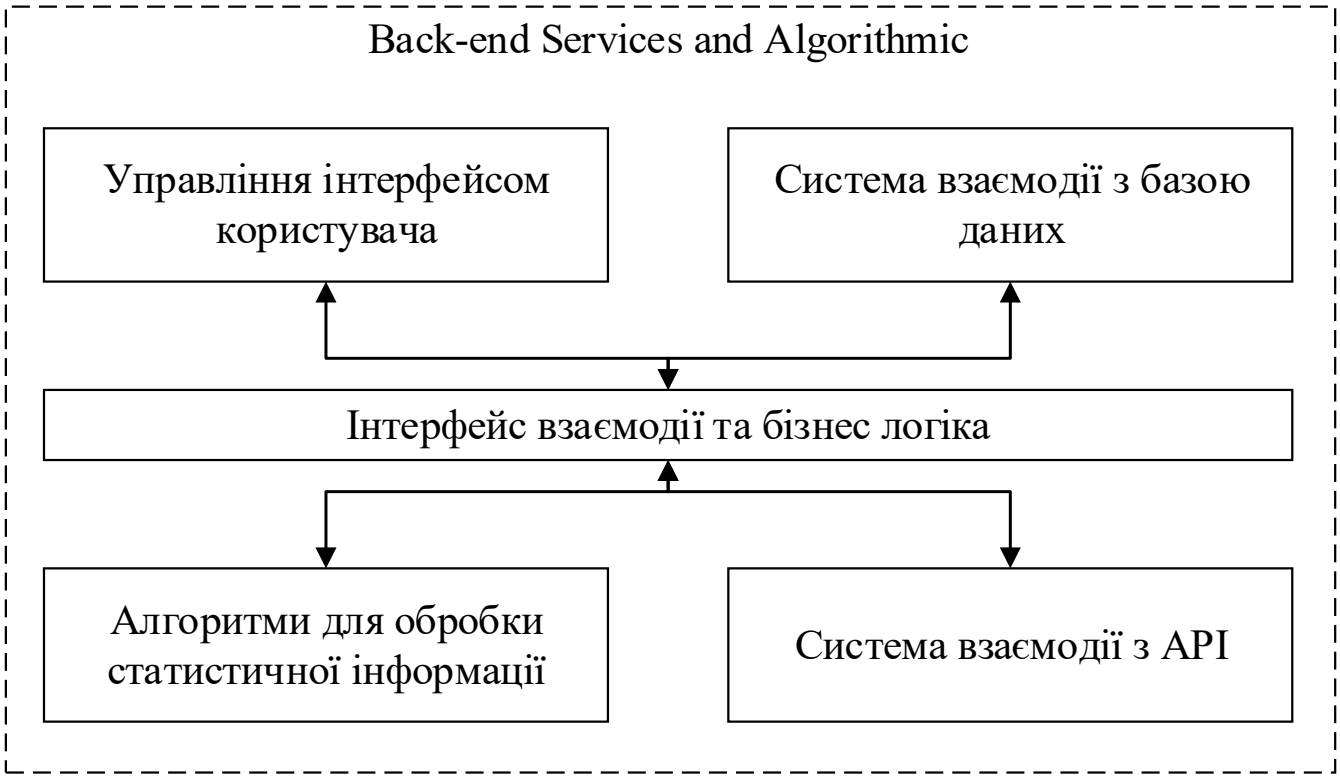
Слайд 14 – Висновки

Основні етапи розробки веб-ресурсу

Архітектура автоматизованої системи

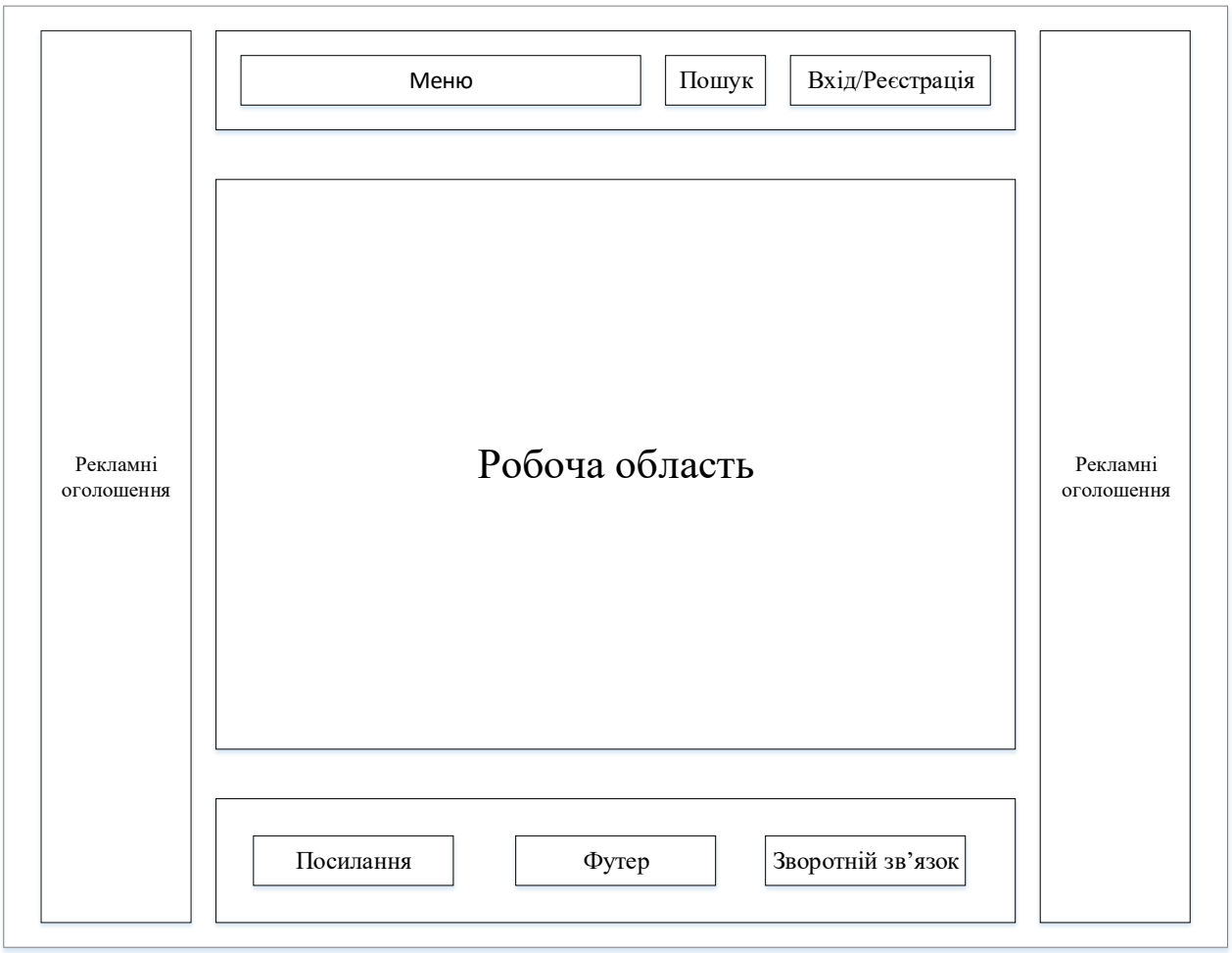


Структура взаємодії елементів управління бек-енду.

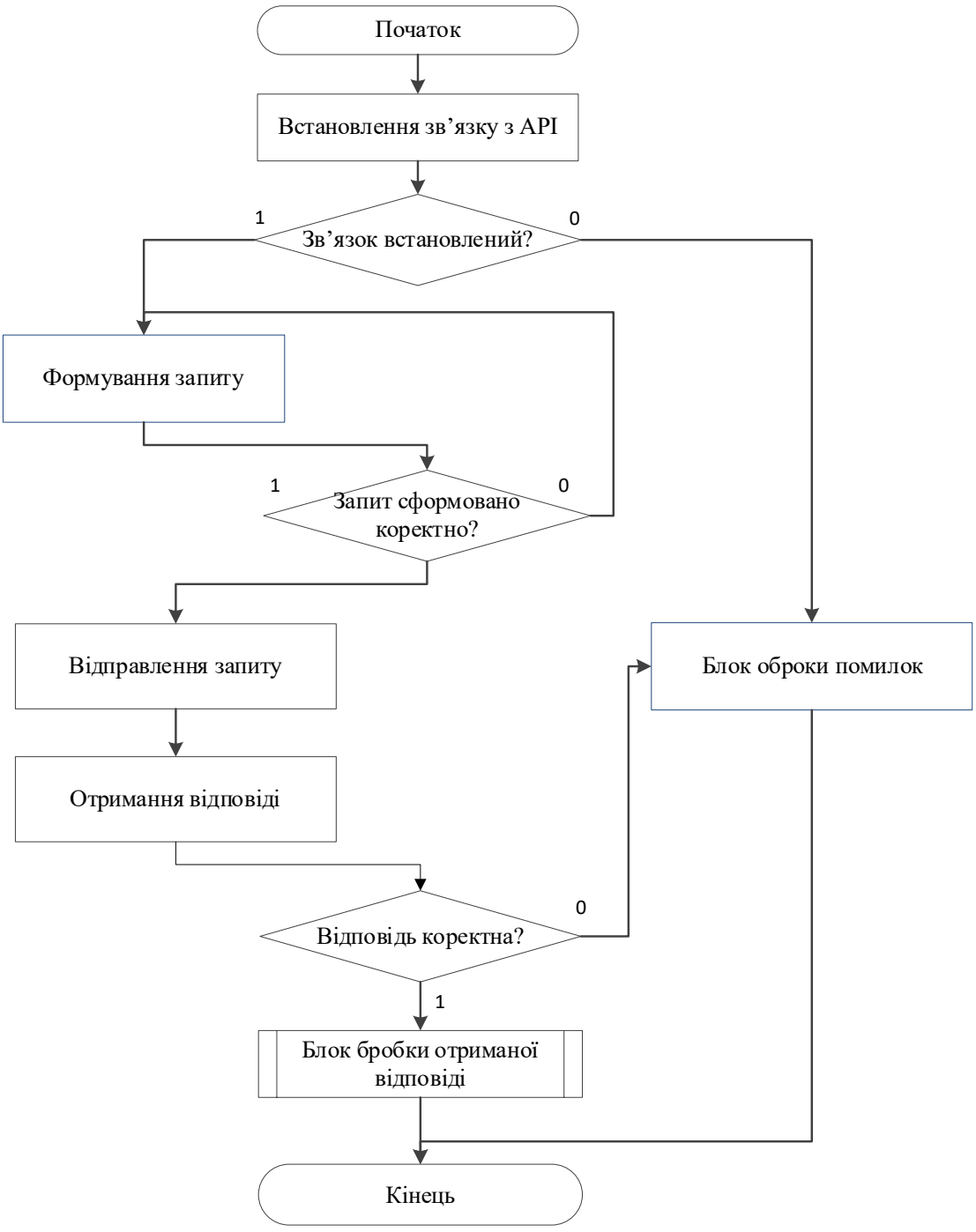


Структура взаємодії елементів управління бек-енду.

Схема зображення головної сторінки

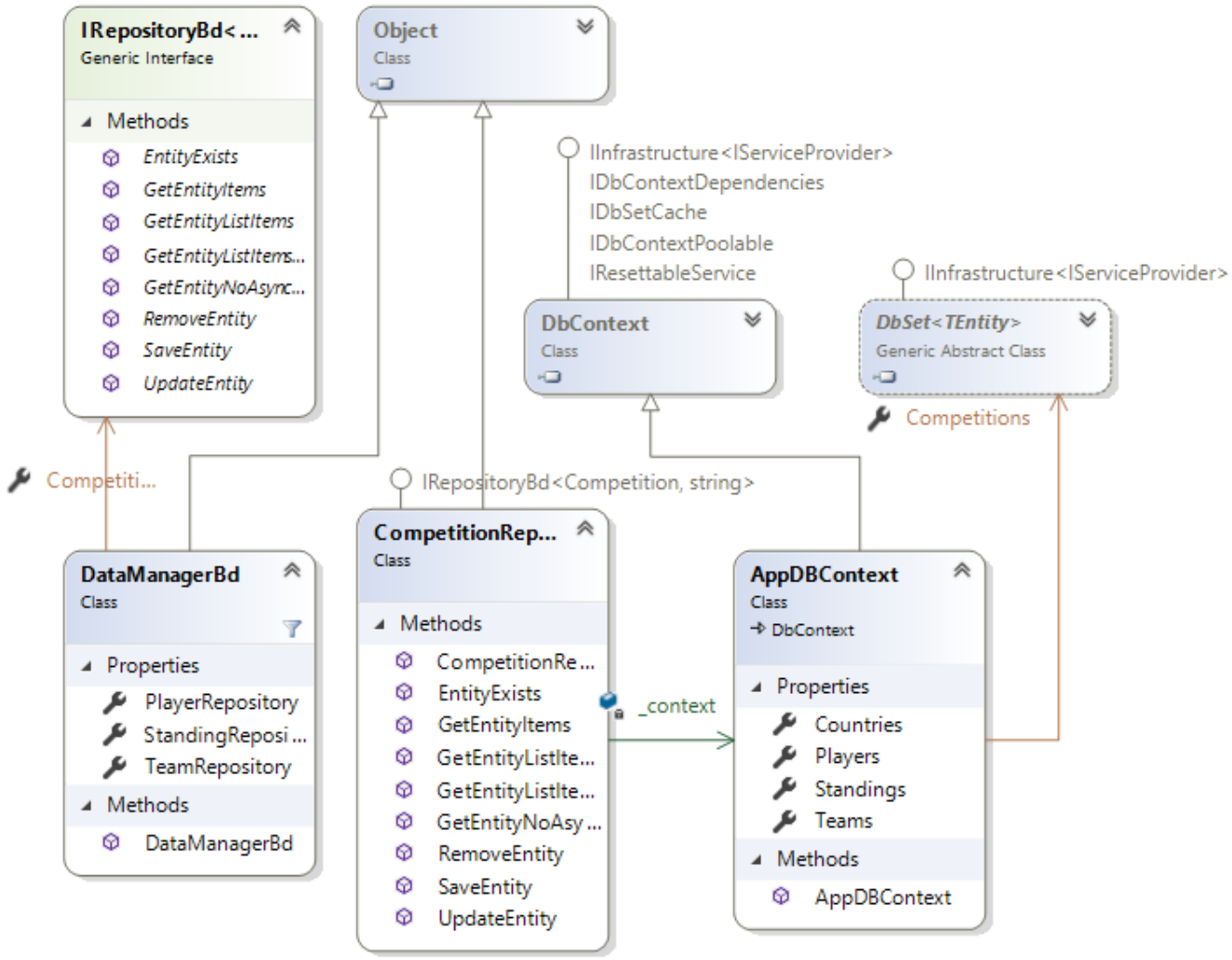


Алгоритм отримання даних від API

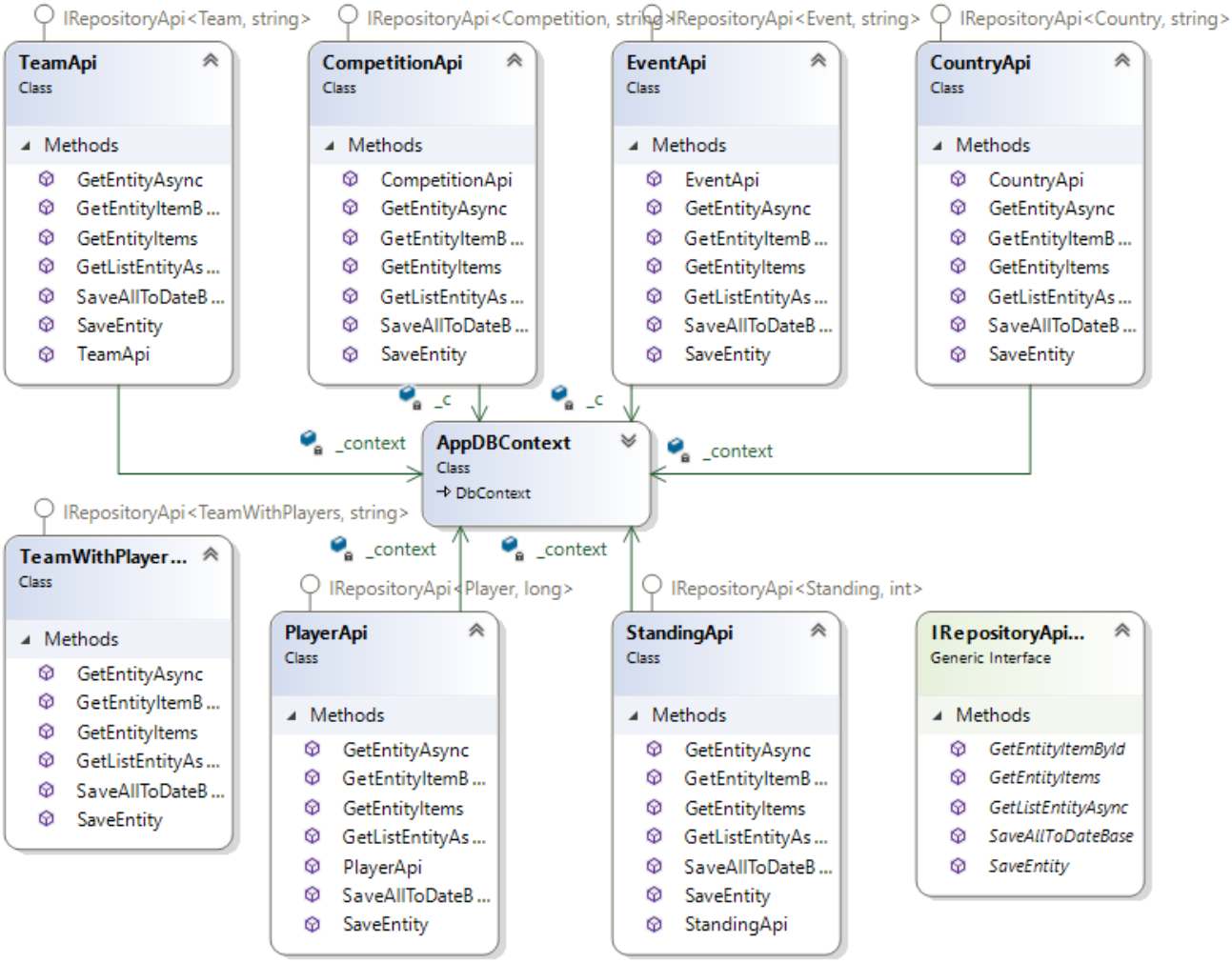


Алгоритм збереження та обробки інформації

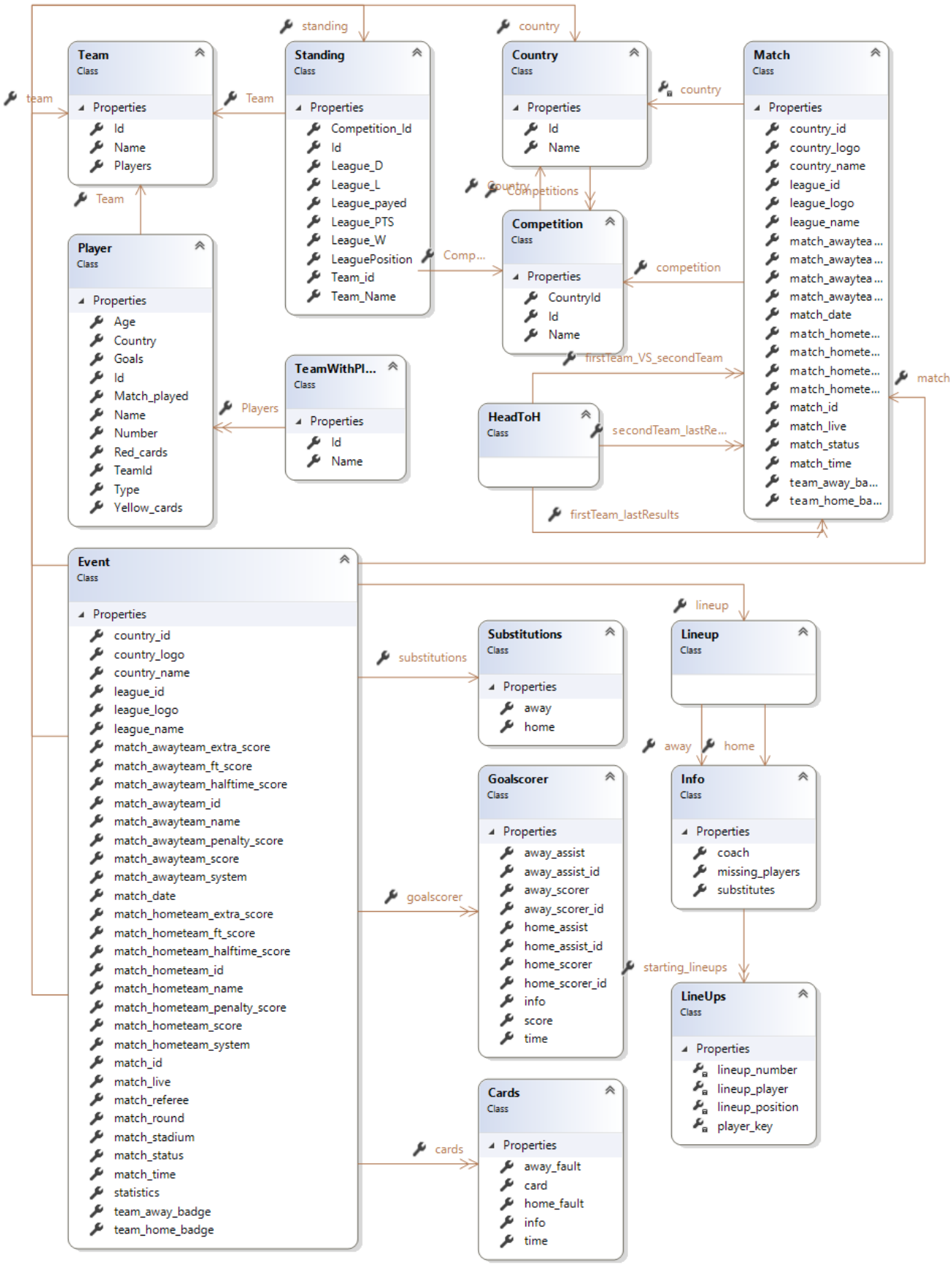
Діаграма класів для взаємодії з базою даних



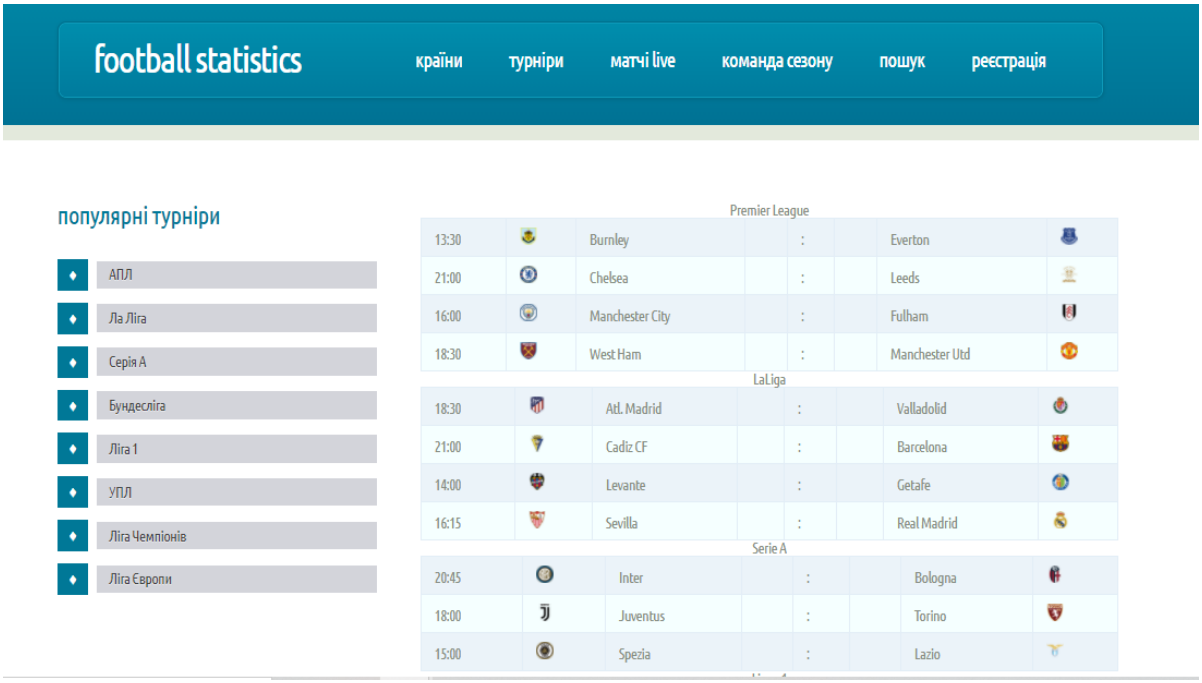
UML – діаграма класів для управління API

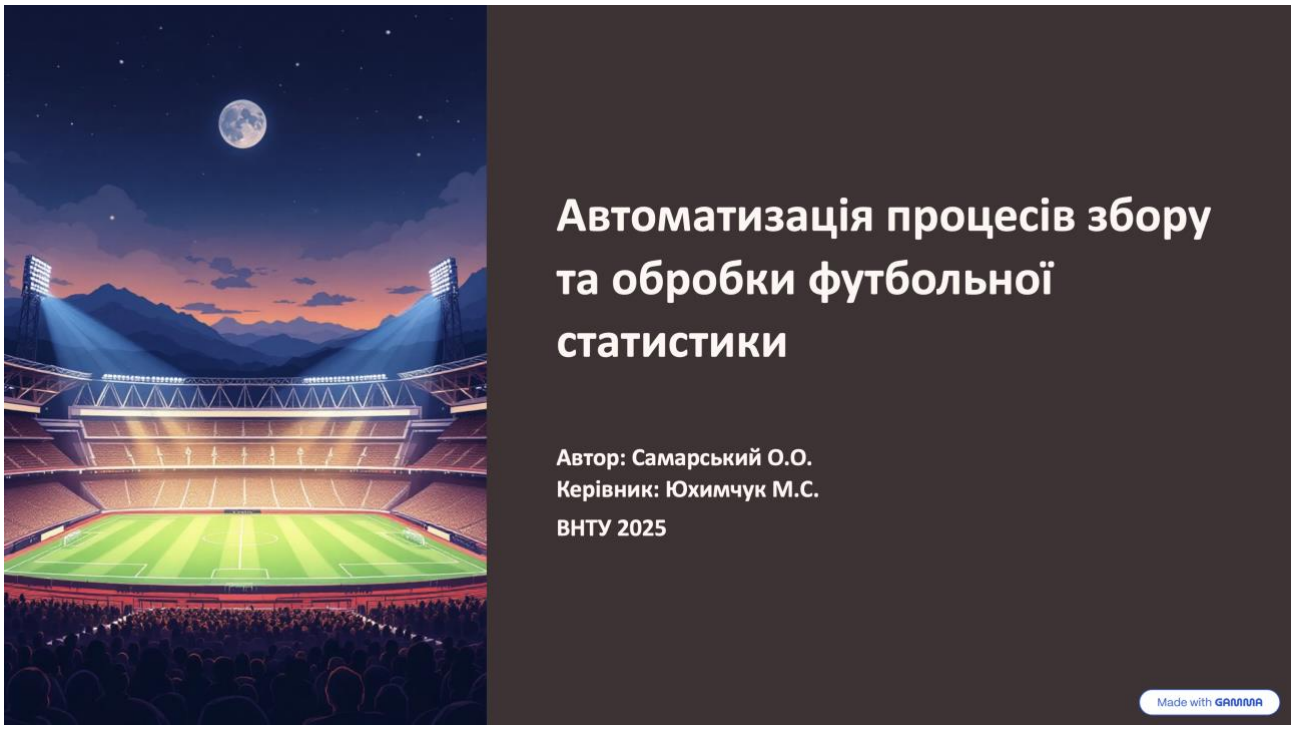


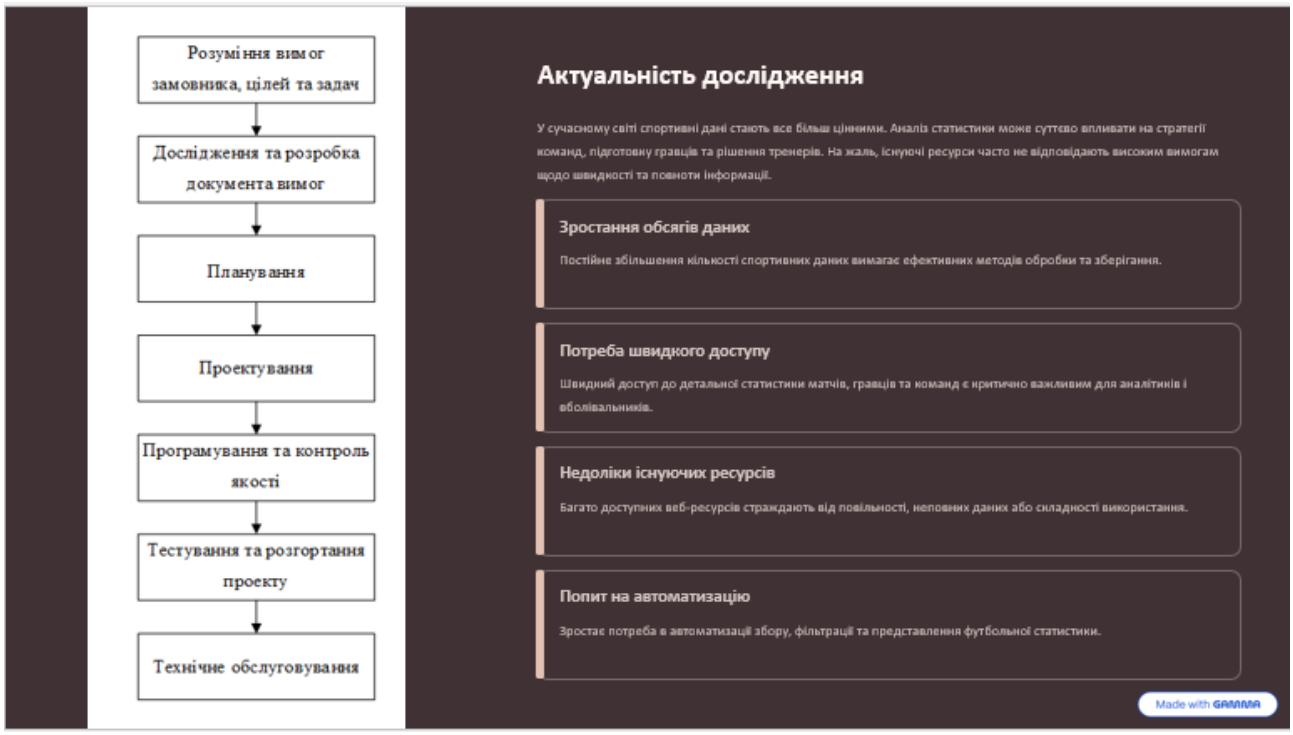
UML – діаграма класів моделей предметної області



Вигляд екрану «Головна сторінка»




Автоматизація процесів збору та обробки футбольної статистики
Автор: Самарський О.О.
Керівник: Юхимчук М.С.
ВНТУ 2025
Made with GAMMA


Розуміння вимог замовника, цілей та задач
Дослідження та розробка документа вимог
Планування
Проектування
Програмування та контроль якості
Тестування та розгортання проекту
Технічне обслуговування
Актуальність дослідження
У сучасному світі спортивні дані стають все більш цінними. Аналіз статистики може суттєво впливати на стратегії команд, підготовку гравців та рішення тренерів. На жаль, існуючі ресурси часто не відповідають високим вимогам щодо швидкості та повноти інформації.
Зростання обсягів даних
Постійне збільшення кількості спортивних даних вимагає ефективних методів обробки та зберігання.
Потреба швидкого доступу
Швидкий доступ до детальної статистики матчів, гравців та команд є критично важливим для аналітиків і вболівальників.
Недоліки існуючих ресурсів
Багато доступних веб-ресурсів страждають від повільності, неповних даних або складності використання.
Попит на автоматизацію
Зростає потреба в автоматизації збору, фільтрації та представлення футбольної статистики.
Made with GAMMA

Мета та завдання роботи

Мета: створення веб-ресурсу для автоматичного збору, обробки та відображення футбольних футбольних даних.

01

Аналіз предметної області

Глибоке вивчення вимог до футбольної статистики та існуючих рішень.

02

Створення архітектури системи

Розробка надійної та масштабованої структури для веб-ресурсу.

03

Проєктування бази даних

Створення ефективної структури зберігання даних для швидкого доступу та обробки.

04

Розробка веб-додатку

Реалізація функціоналу фронтенду та бекенду з урахуванням сучасних стандартів.

05

Тестування та оцінка роботи

Перевірка працездатності, коректності та продуктивності розробленої системи.

Предмет, об'єкт та сутності системи

Об'єкт дослідження:

Процес збору, обробки та аналізу футбольної статистики.

Предмет дослідження:

Статистичні показники матчів, команд і гравців, включаючи детальні параметри гри.

Ключові сутності:

Турнір

Ліги, кубки, чемпіонати.

Команда

Історія, склад, результати.

Футболіст

Біографія, статистика.

Матч

Результати, події, статистика.

Подія

Гол, фол, заміна тощо.

Джерело даних та структура API

Використано API-Football

Надійне та комплексне джерело футбольних даних, що дозволяє отримувати актуальну інформацію.

- 380+ матчів за сезон
- 14+ параметрів на матч
- Дані оновлюються в реальному часі
- Формат обміну: JSON

Детальні дані включають:

- Склади команд та заміни
- Удари по воротах, фоли та картки
- Ключові події в реальному часі (голи, пенальті)
- Індивідуальна статистика гравців (паси, відбори)

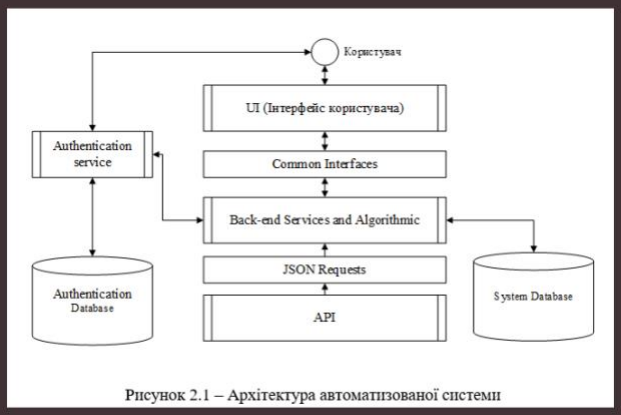
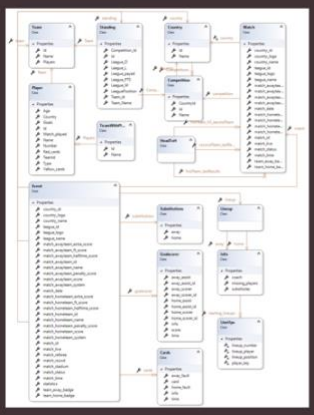


Рисунок 2.1 – Архітектура автоматизованої системи



Made with GAMMA

Архітектура системи

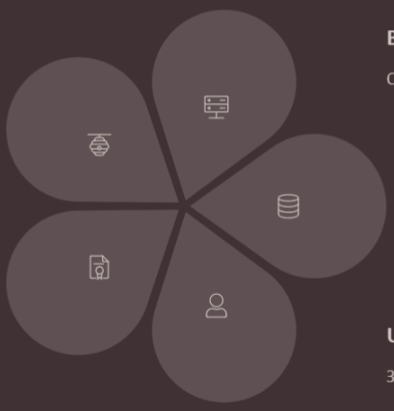
Система побудована на **модульній архітектурі**, що забезпечує гнучкість, масштабованість та легкість у підтримці. Вона розподіляє ролі між різними компонентами та чітко відокремлює бізнес-логіку від представлення даних.

Football API

Отримання даних.

Back-end

Обробка даних, алгоритми, бізнес-логіка.



Database

Збереження структурованих даних.

Authentication

Реєстрація та вхід користувачів.

UI

Зручний інтерфейс користувача.

Такий підхід дозволяє легко додавати нові функції, змінювати окремі модулі без впливу на всю систему та ефективно масштабувати ресурси.

Made with GAMMA

Розробка структури веб-ресурсу

Структура веб-ресурсу розроблена для максимальної зручності користувачів та швидкого доступу до необхідної інформації. Кожен розділ інтуїтивно зрозумілий та функціональний.

1. **Головна сторінка**
   З актуальними новинами та основними подіями.
2. **Розділ Турніри**
   Інформація про ліги, кубки, турнірні таблиці.
3. **Матчі, Команди, Гравці**
   Детальна статистика та профілі.
4. **Рейтинг гравців**
   Рейтинг за різними показниками.
5. **Мої турніри / Команди**
   Персоналізований контент для зареєстрованих користувачів.

Інтерфейс є адаптивним, забезпечуючи коректне відображення на будь-яких пристроях – від мобільних телефонів до великих моніторів.

Алгоритми та обробка статистики

Серцем системи є набір ефективних алгоритмів, що перетворюють сирі дані з API на структуровану, аналітично цінну інформацію. Це дозволяє надавати користувачам глибокі інсайти щодо футбольних подій.

Автоматизована обробка даних

Безперервний імпорт та обробка інформації з API-Football.

Сортування по типах статистики

Категоризація даних за різними параметрами (голи, паси, фоли).

Побудова турнірних таблиць

Автоматичне оновлення положень команд у лігах та групах.

Аналіз індивідуальних показників

Детальна оцінка ефективності кожного гравця.

Генерація статистичних вибірок

Створення унікальних звітів та графіків на запит користувача.

Тестування системи

Для забезпечення високої якості та надійності системи було проведено всебічне тестування. Застосований метод метод «чорного ящика» дозволив перевірити функціональність з точки зору кінцевого користувача, не занурюючись у внутрішню структуру коду.

Перевірка функціональності інтерфейсу

Забезпечення інтуїтивності та працездатності всіх елементів UI.

Тестування API-запитів

Перевірка коректності обміну даними між системою та зовнішнім API.

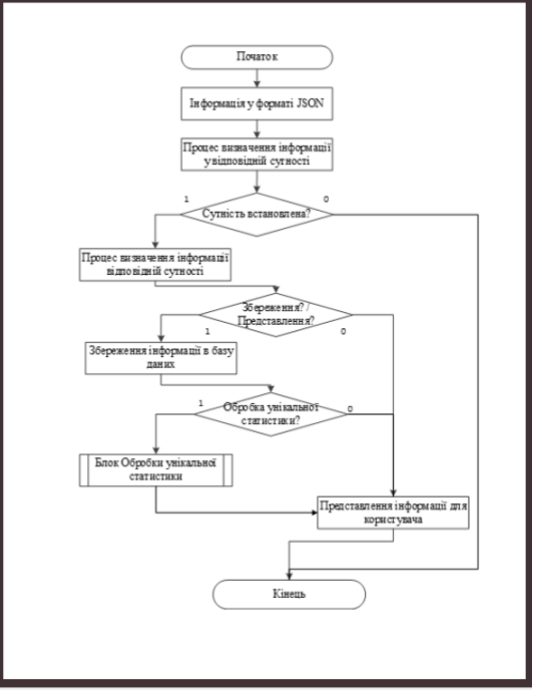
Перевірка кросбраузерності

Гарантія стабільної роботи на різних веб-браузерах та пристроях.

Підтвердження всіх ключових функцій

Успішне проходження тестів основними функціональними можливостями системи.

Результати тестування підтвердили високу працездатність та відповідність системи заявленим вимогам.



Висновки

Проведена робота дозволила створити повноцінну автоматизовану систему управління футбольною статистикою, яка відповідає сучасним вимогам до швидкості, точності та зручності використання.

1. **Створено повноцінний веб-ресурс**
   З реалізацією всіх запланованих функцій та модулів.
2. **Автоматизовано збір та фільтрацію даних**
   Забезпечено ефективну роботу з API-Football.
3. **Реалізовано зручну структуру та дизайн**
   З акцентом на користувацький досвід та адаптивність.
4. **Проведено тестування**
   Підтверджено високу якість та надійність системи.
5. **Готовність до масштабування**
   Система спроектована для подальшого розширення на інші країни, турніри та впровадження розширеної аналітики.