МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Інститут післядипломної освіти

Кваліфікаційна робота бакалавра

на тему: **«Розробка веб-додатку для статистики футбольних матчів на Django»**

Виконав студент групи **КН-5**
спеціальності 122 Комп'ютерні науки
Іовчев Олексій Васильович

Керівник **ст. викладач**
Штефан Наталія Зінов'ївна

Консультант **док. техн. наук, професор Казакова Н.Ф.**

Рецензент **к.т.н., доцент**
Сергієнко Андрій Володимирович

Одеса 2023

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

БД – база даних

СКБД – система керування базами даних

API – Application programming interface

ORM – Object-relational mapping

MVC – Model-view-controller

Django (Джанго) – високорівневий відкритий Python-фреймворк (програмний каркас) для розробки вебсистем

Flask – мікрофреймворк для вебдодатків, створений з використанням Python

MySQL – вільна система керування реляційними базами даних

PostgreSQL – об'єктно-реляційна система керування базами даних

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією

SQL – Structured query language

SQLite – полегшена реляційна система керування базами даних

RESTful API – це інтерфейс, що використовуються двома комп'ютерними системами для безпечного обміну інформацією через Інтернет

ВСТУП

Сучасний світ характеризується широким поширенням футбольних ігор і підвищеним інтересом до футбольної статистики серед уболівальників цього виду спорту. Наразі існує безліч веб-сайтів і програм, які надають статистику футбольних матчів, але багато з них не забезпечують легкого та швидкого доступу до останньої інформації. Таким чином, проект, спрямований на створення веб-сайту з актуальною статистикою футбольних матчів, має велике значення та потенціал для задоволення потреб користувачів, яким потрібна надійна та швидка інформація про футбольні матчі.

Робота надає користувачам зручний доступ до актуальної інформації про останні футбольні матчі. Завдяки використанню API, сайт отримуватиме свіжі дані про матчі, включаючи результати, статистику гравців, час проведення та інші важливі деталі.

Також, сайт дозволяє користувачам проводити аналітичні дослідження на основі зібраної статистики. Вони можуть порівнювати різні команди, гравців та ліги, аналізувати їхню продуктивність та аналізувати результати минулих матчів. Це може бути корисним для футбольних уболівальників, аналітиків та тренерів, допомагаючи їм приймати поінформовані рішення.

Крім того, проект може сприяти створенню спільноти футбольних ентузіастів, які можуть обговорювати матчі, ділитися своїми прогнозами та статистичними висновками. Це може зробити сайт популярним та допомогти залучити аудиторію.

Загалом, робота надає корисний ресурс для футбольних уболівальників та аналітиків, сприяє обміну інформацією та може сприяти розвитку та вдосконаленню навичок розробників.

Сайт надаватиме інформацію про матчі та статистику, яка може бути корисною для людей, які займаються ставками на футбол або беруть участь у фентезі-лігах. Вони зможуть використовувати статистику та дані про минулі

матчі для прийняття обґрунтованих рішень при складанні своїх команд або прогнозуванні результатів матчів.

Мета: метою роботи є розробка та реалізація веб-сайту з актуальною статистикою футбольних матчів за допомогою API на основі Django framework.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Опис предметної області

Футбол – це популярний вид спорту, який має величезну кількість фанатів по всьому світу. Футбольні матчі проводяться регулярно, і фанати бажають бути в курсі останніх результатів, статистики та інших деталей матчів. Однак пошук актуальної інформації може бути часо- та ресурсомістким завданням. Тому створення веб-додатку, який забезпечує зручний доступ до статистики останніх футбольних матчів, має великий практичний інтерес.

Проект передбачає розробку та впровадження веб-сайту, який надає статистику та інформацію про останні футбольні матчі. Основною метою цього проекту є отримання останніх даних про футбольні матчі через API (інтерфейс прикладного програмування). API надає структурований спосіб доступу та отримання даних із зовнішніх джерел, таких як спортивні бази даних або служби живих результатів. API, які використовуються в цьому проекті, працюють с базою данних сервісу футбольних подій та надає таку інформацію, як результати матчів, статистика гравців, деталі команди та інші відповідні дані.

Функціональність сайту включатиме відображення останніх футбольних результатів, турнірну таблицю ліг, статистику гравців, історичні дані тощо. Дані, отримані з API, будуть оброблені та представлені у візуально привабливому та зручному для користувача вигляді [1].

1.2 Огляд аналогів

Для проведення аналізу були вибрані декілька схожих проектів, які надають статистику футбольних матчів. Нижче наведено скріншоти та короткий опис кожного з них.

Аналог перший «Flashscore» (рис. 1) – це онлайн-сервіс футбольних результатів на Flashscore.ua пропонує результати матчів понад 1000 футбольних ліг, кубків та турнірів (таких як УПЛ, Ліга чемпіонів, Прем'єр-ліга, Ла Ліга, Серія А), надаючи також турнірні таблиці, дані про авторів голів, жовті і червоні картки; відеоогляди, сигнали про забиті голи та іншу інформацію про сьогоднішні футбольні матчі live. Отримувати звукові оповіщення, стежити за обраними поєдинками, дізнаватись поточні та підсумкові результати футбольних зустрічей live.

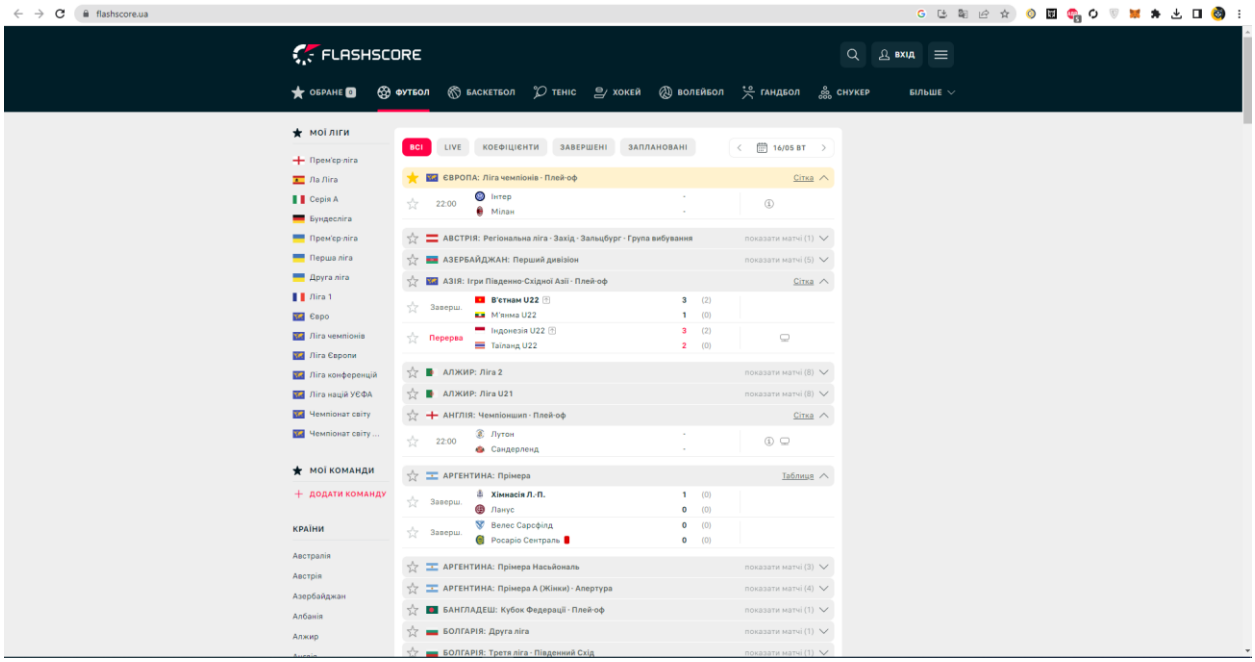


Рисунок 1 – Скріншот сайту «Flashscore.ua»

Наступний аналог – це українськи ресурс «Sport.ua» (рис. 2) загальноспортивний новинний портал в Україні. Один з найвідвідуваніших новинних спортивних сайтів в Україні. На ньому зібрано новини з багатьох видів спорту.

У жовтні 2010 року аудиторія сайту склала 915 тисяч читачів. Зростання аудиторії сайту не зупинявся і вже в жовтні 2019 року вона склала вже понад 11 мільйонів читачів [2].

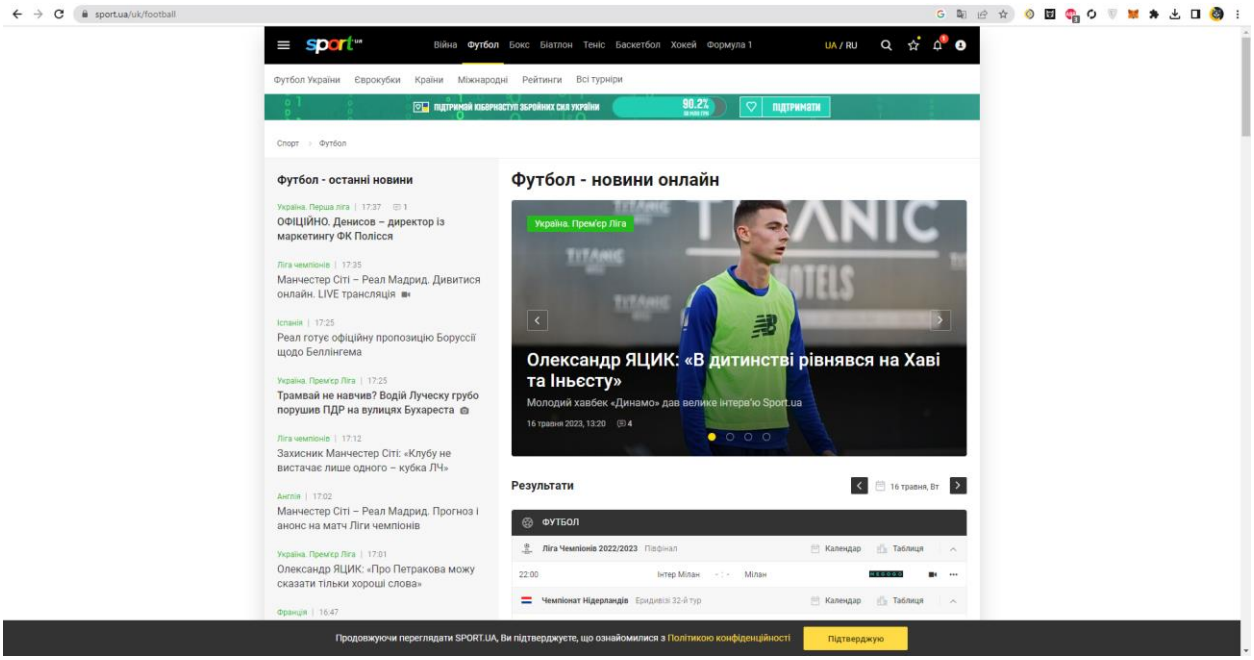


Рисунок 2 – Скріншот сайту «Sport.ua»

Охоплення сайту за даними InMind серед українських користувачів інтернету у лютому 2011 року становило 4,61 %. У березні 2011 року цей показник дорівнював 6,34 %.

Згідно з результатами соціологічного опитування Київського міжнародного інституту соціології та проєкту InPoll проведеного серед футбольних уболівальників у квітні 2011 року Sport.ua відвідало 24 % українських користувачів.

За даними досліджень платформ контент-маркетингу PRNEWS.IO і SimilarWeb у 2019 році сайт входить до 10 найвідвідуваніших українцями в Німеччині сайтів і є одним з двох спортивної тематики [3].

Третій аналог – це сервіс «OFStats.com» (рис. 3) надає статистику футбольних матчів, покриваючи понад 50 ліг та кубків. попередній матч: порівняльна статистика гравців, інтенсивність календаря матчів команд, статистичні факти, передматчева статистика на основі останніх матчів.

Сервіс OFStats.com (рис. 3) надає статистику футбольних матчів, покриваючи понад 50 ліг та кубків. попередній матч: порівняльна статистика

гравців, інтенсивність календаря матчів команд, статистичні факти, передматчева статистика на основі останніх матчів.

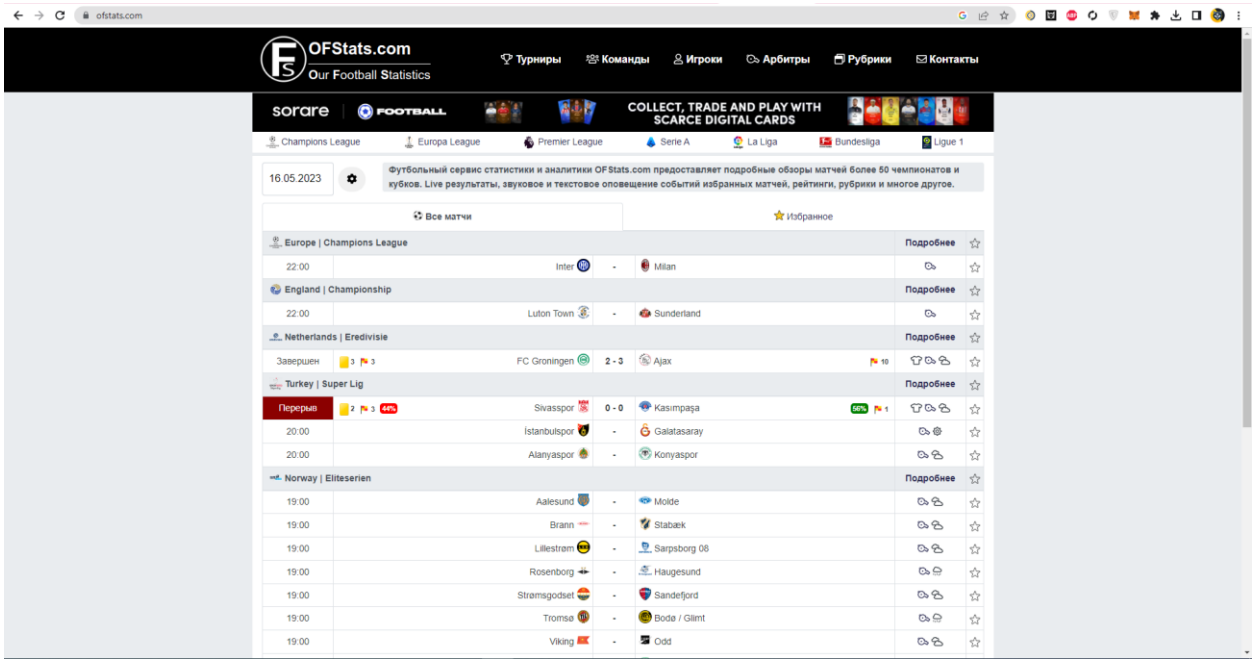


Рисунок 3 – Скріншот сайту «OFstats.com»

LIVE результати матчів, аналітика потенційних кутових ударів та жовтих карток по ходу матчу, оновлення статистики та відображення подій матчу в режимі реального часу. Текстові Live трансляція матчу. Відео матеріали та коефіцієнти зустрічі. Експрес-аналіз арбітра у кожному матчі. Порівняльна статистика команд у матчі за основними показниками: голи, кутові, жовті картки у часових періодах та загальні тотали показані на графіках та діаграмах. Статистика турнірів – таблиця, календар матчів, результати. Рейтинги гравців з турніру: голи, асисти, дисципліна.

Статистика команд: голи, дисципліна, кутові, володіння м'ячем, удари за гру та інші параметри. Загальна інформація про команду, тренер, стадіон, склад. Розклад матчів команди, поточна ігрова форма, результати сезону. Визначення за статистичними показниками як найкращих, так і найгірших команд сезону. Відстеження серій матчів команд: перемоги, нічиї чи поразки. Статистика гравців – найкращі та найгірші гравці тижня, футболісти місяця.

Статистика гравців протягом сезону у розрізі їхніх позицій на полі: рейтинг, голи, асисти, удари по воротах, дисципліна, дриблінг, відбори та інші статистичні показники.

Статистика арбітрів – картки у матчах минулого тижня, тенденції у роботі рефері за останній місяць, відстеження найбрутальніших матчів за минулий місяць, загальний рейтинг арбітрів. Футбольні рубрики: свято на вулиці андердогу та інші. Пошук гравців, команди чи арбітра [4].

1.3 Вибір засобів розробки

На цьому етапі для дипломного проекту необхідно обрати фреймворкі та баз даних для створення веб-сайту. Мною були розглянуті такі СКБД, як MySQL, PostgreSQL та SQLite. Також, мною було проаналізовано фреймворкі Django та Flask.

1.3.1 Системи управління баз даних MySQL і PostgreSQL

MySql і PostgreSQL є популярними та надійними системами керування реляційними базами даних. Вони добре підходять для великих проектів, що вимагають високої масштабованості, складних зв’язків даних і розширених функцій бази даних. Обидві бази даних пропонують чудову продуктивність, надійність і підтримку одночасного доступу користувачів.

Обидві системи надають надійне зберігання та керування структурованими даними та широко застосовуються в різних додатках та веб-сайтах. MySQL та PostgreSQL бази даних ґрунтуються на реляційній моделі даних, яка представляє дані у вигляді таблиць, пов'язаних один з одним ключами. Це забезпечує структуроване зберігання інформації та можливість виконання складних запитів та операцій.

Обидві СКБД підтримують SQL (Structured Query Language) для створення та управління базами даних. SQL дозволяє виконувати операції

пошуку, вставки, оновлення та видалення даних, а також проводити аналітичні запити та агрегування даних. Обидві системи дозволяють одночасний доступ кільком користувачам до бази даних і забезпечують контроль доступу за допомогою користувачів, ролей і прав доступу.

Як MySQL, і PostgreSQL надають механізми забезпечення цілісності даних через транзакції. Це дозволяє групувати операції в рамках одного логічного блоку, гарантуючи, що всі операції виконуються успішно, або жодна з них не застосовується. PostgreSQL має велику гнучкість і розширюваність, дозволяючи розробникам створювати власні типи даних, функції та розширення. MySQL також пропонує деякі можливості розширення, але меншою мірою в порівнянні з PostgreSQL.

Однак, SQLite має перевагу через його простоту в налаштуванні та управлінні, а також здатність відповідати вимогам проекту без додаткової конфігурації сервера.

1.3.2 Система керування баз даних SQLite

SQLite – це легка безсерверна система управління реляційною базою даних.його легко налаштувати та налаштувати, оскільки він не потребує встановлення окремого сервера бази даних.

Вона не вимагає окремого сервера або складного налаштування, оскільки база даних зберігається в єдиному файлі. SQLite підходить для різних програм та платформ, включаючи мобільні пристрої та вбудовані системи. Він забезпечує підтримку транзакцій, ACID-властивості і має хорошу продуктивність при обробці запитів.

SQLite є популярним вибором для проектів, які потребують простоти та ефективності роботи з даними. Бази даних SQLite зберігаються в одному файлі, що спрощує розгортання та розповсюдження. Для невеликих проектів або проектів із низьким або помірним трафіком SQLite пропонує хорошу продуктивність і може ефективно впоратися з потребами проекту в базі даних.

SQLite підтримує стандарт SQL та надає потужну мову запитів для керування даними. Розробники можуть використовувати SQL-запити для створення, зміни, вилучення та видалення даних у базі даних SQLite. Він також підтримує індекси, тригери та уявлення, що забезпечує гнучкість та розширюваність бази даних. Данна СКБД має гарну продуктивність і здатний обробляти великі обсяги даних. Він має ефективні алгоритми індексації, що забезпечує швидкий доступом до даних.

Крім того, SQLite пропонує оптимізацію запитів, щоб забезпечити ефективне виконання складних запитів на пошук, фільтрацію та агрегацію даних. Інтеграція з Django безперебійна, оскільки Django забезпечує власну підтримку SQLite, полегшуючи розробку та тестування.

1.3.3 Django Framework

Django – це високорівневий веб-фреймворк, написаний мовою Python, який дозволяє розробляти потужні веб-програми швидко і ефективно. Він дотримується архітектурного шаблону Model-View-Controller (MVC), полегшуючи організацію коду та розділення завдань. Django надає вбудовані функції для автентифікації користувачів, міграції баз даних і URL-маршрутизації, що прискорює розробку.

Одна з ключових особливостей Django – це ORM (Object-Relational Mapping), який полегшує взаємодію з базою даних, представляючи дані у вигляді об'єктів та автоматично створюючи відповідну структуру бази даних на основі певних моделей. Важливою частиною Django є вбудований адміністративний інтерфейс, який дозволяє управляти даними моделей та створювати панель керування адміністратора без додаткової роботи. Це спрощує адміністрування та дозволяє швидко налаштувати та налаштувати веб-додаток.

Django також пропонує широкий набір інструментів для обробки URL-маршрутів, шаблонів, форм, автентифікації користувачів та авторизації, а

також для обробки запитів та створення RESTful API. Фреймворк також підтримує кешування, міжнародну локалізацію, захист від CSRF-атак та багато іншого. Однією з переваг Django є його активна спільнота розробників, які постійно працюють над покращенням фреймворку та наданням документації, посібників та розширень для допомоги у розробці. Django також підтримує шаблонизацію, що полегшує створення динамічних сторінок із використанням шаблонів Jinja.

В цілому, Django є потужним інструментом для розробки веб-додатків, забезпечуючи зручність використання, гнучкість і безліч функцій, що робить його популярним вибором для розробників, які прагнуть створити високоякісні та масштабовані веб-додатки.

1.3.4 Flask Framework

Flask – це легкий та гнучкий веб-фреймворк, написаний мовою програмування Python, який дозволяє розробляти веб-програми швидко та ефективно. Він надає мінімальний набір інструментів та функціональності, що робить його простим у освоєнні та розумінні.

Однією з основних особливостей Flask є мінімалістичний підхід. Він не нав'язує розробникам певні архітектурні рішення та надає свободу вибору в організації коду та структури проекту. Flask сприяє створенню маленьких, модульних і додатків, що легко підтримуються.

Фреймворк надає простий у використанні маршрутизатор URL, який дозволяє пов'язувати URL-адреси з функціями обробників. Це дозволяє легко визначити, яка дія повинна виконуватися при зверненні до певної URL-адреси.

Додатково, Flask надає архітектуру, що розширюється, що дозволяє розробникам інтегрувати різні розширення і сторонні бібліотеки в свої проекти. Це сприяє створенню функціональних та настроюваних веб-додатків з урахуванням специфічних потреб.

Flask також має вбудований сервер розробки, що дозволяє запускати програму без необхідності налаштування окремого сервера. Це спрощує розробку та налагодження програм у локальному середовищі.

За результатами аналізу програмних засобів було прийнято рішення зупинитися на: Django та SQLite для розробки дипломного проекту завдяки їхнім сукупним перевагам. Комплексні функції Django, простота розробки та багата екосистема зробили його чудовим вибором для створення надійного веб-сайту.

З іншого боку, SQLite надав легке та ефективне рішення для баз даних, яке бездоганно інтегрувалося з Django, задовольняючи вимоги проекту без необхідності додаткового налаштування сервера. Цей вибір дозволив швидко розробити, спростити розгортання та ефективне керування даними проекту.

1.4 Постановка завдання

Дипломний проект вимагатиме знань і навичок у веб-розробці, зокрема у використанні фреймворку Django, роботі з API, а також обробці та представленні даних у змістовний спосіб. Це також передбачає розробку інтуїтивно зрозумілого інтерфейсу користувача та забезпечення адаптивності веб-сайту на різних пристроях.

Загалом, проект спрямований на створення функціонального та інформативного веб-сайту, який надає користувачам актуальну статистику та інформацію про останні футбольні матчі, покращуючи їхній загальний досвід перегляду футболу.

Проект буде використовувати клієнт-серверну архітектуру. Клієнтська частина буде представлена у вигляді веб-інтерфейсу, доступного з різних пристроїв.

Серверна частина буде реалізована на базі фреймворка Django. Вона буде відповідати за обробку запитів, взаємодію з базою даних та зовнішніми API для отримання інформації про футбольні матчі. Архітектура проекту буде

розподіленою, що дозволить забезпечити швидку та ефективну роботу системи.

Для реалізації проекту необхідно:

1. Налаштування засобів розробки: для роботи з Django потрібно встановити Python, оскільки Django є фреймворком, написаним мовою програмування Python. Після інсталяції Python необхідно встановити Django.
2. Визначити структуру проекту: створити та визначити структуру проекту Django, включаючи налаштування бази даних, маршрутизацію URL, моделі даних, шаблони та уявлення.
3. Отримати доступ до футбольного API: знайти відповідний API, який надає статистику про останні футбольні матчі. Зареєструватися на API, отримати ключ доступу та налаштувати запити для отримання необхідних даних.
4. Написати уявлення та шаблони: створити уявлення, які будуть обробляти запити від користувачів та взаємодіяти з API для отримання статистики матчів. Потім створити відповідні шаблони, які відображатимуть ці дані користувачеві.
5. Конфігурація взаємодії: налаштувати відповідну роботу HTML шаблонів уявлення з функціями видачі результату запиту користувачу.
6. Поліпшити інтерфейс користувача: оформити сайт зі статистикою матчів, щоб забезпечити зручність використання та привабливий зовнішній вигляд. Це включає роботу з CSS, HTML і JavaScript для додавання динамічних елементів.
7. Тестування та налагодження: перевірити функціональність сайту, протестувати його на різних пристроях та браузерах, а також виправити можливі помилки та проблеми.
8. Розгортання: розгорнути програму на локальному сервері.

1.4.1 Функціональні вимоги проекту

До функціональних вимог слід віднести наступні:

- перегляд списку останніх футбольних матчів;
- пошук матчів за певними критеріями (команда, турнір, гравці тощо);
- відображення статистики/біографії гравців та останніх матчів гравця;
- генерація графіків та діаграм для візуалізації статистики;
- інформація о турнірах та останні матчі за турніром;
- швидкий доступ з меню до гравців тощо;
- адміністративний доступ для управління матчами, командами та користувачами.

1.4.2 Вимоги до інтерфейсу

Слід визначити наступні вимоги:

- зручний та інтуїтивно зрозумілий інтерфейс для користувачів;
- чистий та привабливий дизайн, забезпечення гармонійного спілкування з користувачем;
- навігаційне меню для легкого доступу до різних функцій та сторінок;
- адаптивний дизайн, щоб сайт коректно відображався на різних пристроях (комп'ютерах, планшетах, мобільних телефонах).

До бази даних:

- зберігання інформації про футбольні матчі, команди, гравців, результати та статистику;
- ефективна структура бази даних для швидкого доступу та оптимального зберігання даних;
- забезпечення цілісності та безпеки даних;
- система управління базами даних – ORM Django.

До безпеки:

- застосування механізмів шифрування та хешування для захисту пароля адміністратора;
- запобігання вразливостям, таким як SQL-ін'єкція та інші ін’єкціям;
- обмеження доступу до адміністративних функцій та забезпечення рівня доступу для різних користувачів.

До розгортання та інтеграції:

- можливість розгортання сайту на різних хостинг-платформах або серверах;
- інтеграція з відповідними футбольними API для отримання актуальних даних про матчі та статистику;
- забезпечення підтримки мов програмування та залежностей, необхідних для роботи проекту;
- документація щодо процесу розгортання та інтеграції, включаючи інструкції та вимоги до середовища.

2 РОЗДІЛ ПРОЕКТУВАННЯ

2.1 Архітектура веб-додатку

MVT – це шаблон проектування програмного забезпечення. Це сукупність трьох важливих компонентів, представлення моделі та шаблону. Модель допомагає працювати з базою даних. Це рівень доступу до даних, який обробляє дані.

Шаблон – це рівень презентації, який повністю обробляє частину інтерфейсу користувача. Представлення використовується для виконання бізнес-логіки та взаємодії з моделлю для перенесення даних і візуалізації шаблону. Хоча Django слідує шаблону MVC, але зберігає свої власні угоди. Отже, керування здійснюється самим фреймворком. Немає окремого контролера, і повна програма базується на представленні моделі та шаблоні. Ось чому його називають додатком MVT.

Перегляним наступний графік (рис. 4), який показує потік керування на основі MVT:

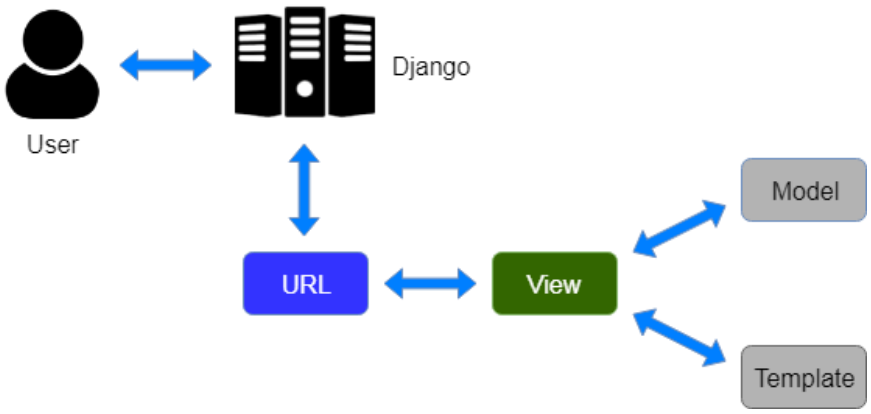


Рисунок 4 – Графік MVT

Користувач запитує ресурс у Django, Django працює як контролер і перевіряє доступний ресурс за URL-адресою. Якщо URL-адреса відображається, називається представлення, яке взаємодіє з моделлю та

шаблоном, воно відображає шаблон. Django відповідає користувачеві та надсилає шаблон у відповідь [6].

2.2 UML use-case діаграма

Unified Modeling Language (UML) – уніфікована мова моделювання. Простіше кажучи, UML – це сучасний підхід до моделювання та документування програмного забезпечення. Фактично, це одна з найпопулярніших технік моделювання бізнес-процесів.Він заснований на схематичному зображенні програмних компонентів. Як говорить старе прислів'я: «картинка варта тисячі слів». Використовуючи візуальні представлення, ми можемо краще зрозуміти можливі вади або помилки в програмному забезпеченні чи бізнес-процесах [5].

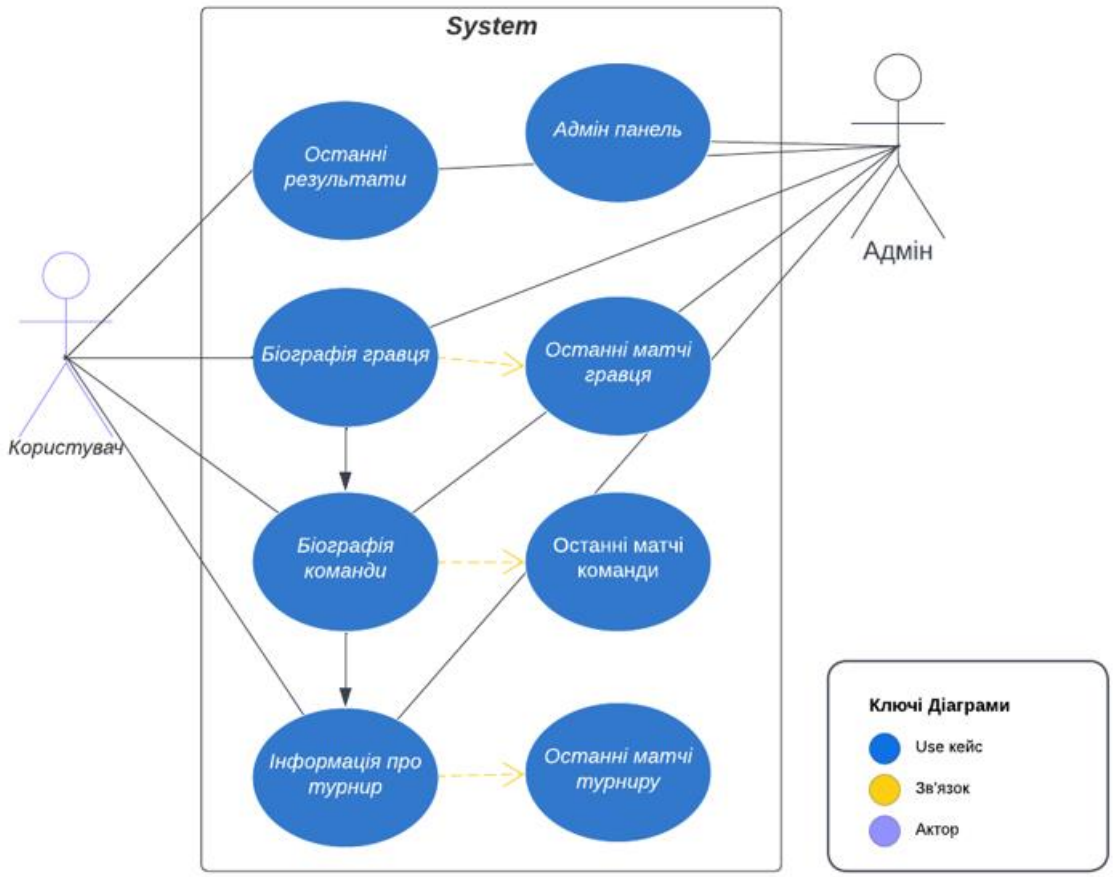


Рисунок 5 – UML діаграма

На рис. 5 зображена UML діаграма, яка відображає можливості взаємодії користувача із сайтом, а також адміністратора. Адміністратор, на відміну від користувача має доступ до адмін-панелі, яка дає йому право на редагування, видалення та внесення змін інформації на сайті.

2.3 Проектування БД

У моєму проекті розроблено чотири моделі (таблиці бази даних) – Scores, Players, Players_matches, Champions_League_Scores. Кожна з них має власну логіку та функцію взаємодії з усім проектом. Розглянемо їх більш деьально.

Таблиця «Scores» – це основна таблиця, яка зберігає інформацію про всіх останніх і актуальних матчів, що закінчилися, також їх рахунок і дату. Розглянемо поля таблиці:

Таблиця 1 – «Scores»

Ім’я поля	Тип даних	Призначення
id	IntegerField	айді кожної нової запису до таблиці
first_team	CharField	назва першої команди
first_team_icon	URLField	іконка першої команди
second_team	CharField	назва другої команди
second_team_icon	URLField	іконка другої команди
match_date	DateTiemField	дата та час матчу
score_first_team	IntegerField	кількість забитих м’ячів першої команди.
score_second_team	IntegerField	кількість забитих м’ячів другої команди.

Наступна таблиця «Players», ця таблиця містить інформацію про кожного гравця доданим до неї. Вона має біографію, фотографію, додаткове фото, дату народження, ім'я, національність, актуальну команду конкретного гравця.

Таблиця 2 – «Players»

Ім’я поля	Тип даних	Призначення
id	IntegerField	айді кожної нової запису до таблиці
name	CharField	ім’я та прізвище гра
bio	TextField	біографія гравця
date_of_birth	DateTimeField	дата народження гравця
nationality	CharField	національність гравця
current_team	CharField	актуальна команда
photo	ImageField	фото гравця
second_photo	ImageField	додаткове фото гравця

Таблиця «Player_matches» – ця таблиця, також як і Scores має ті самі поля, але її відрізняє те що вона має ForeignKey, що дає їй зв'язок між матчем і конкретним гравцем з таблиці Players. Завдяки цьому, ми можемо відображати матчі конкретного гравця.

Таблиця 3 – «Player_matches»

Ім’я поля	Тип даних	Призначення
player	ForeignKey	зв’язок с таблицею Players

Продовження таблиці 3

id	IntegerField	айді кожної нової запису до таблиці
first_team	CharField	назва першої команди
first_team_icon	URLField	іконка першої команди
second_team	CharField	назва другої команди
second_team_icon	URLField	іконка другої команди
match_date	DateTiemField	дата та час матчу
score_first_team	IntegerField	кількість забитих м’ячів першої команди.
score_second_team	IntegerField	кількість забитих м’ячів другої команди.

ForeignKey у Django дозволяє збігатися з основним ключем з окремої таблиці. Поле зовнішнього ключа дозволить створювати зв’язки «багато-до-одного» залежно від типу згенерованого зв’язку для гнучкого з’єднання кількох таблиць з однією таблицею на іншому кінці.

Остання таблиця «Champions_League_Scores», дана таблиця відповідає за результати матчів у чемпіонаті "Ліга Чемпіонів", вона містить також дані про команди, дату та рахунок матчів.

Таблиця 4 – «Champions_League_Scores»

Ім’я поля	Тип даних	Призначення
id	IntegerField	айді кожної нової запису до таблиці
first_team	CharField	назва першої команди
first_team_icon	URLField	іконка першої команди
second_team	CharField	назва другої команди
second_team_icon	URLField	іконка другої команди

Продовження таблиці 4

match_date	DateTiemField	дата та час матчу
score_first_team	IntegerField	кількість забитих м’ячів першої команди.
score_second_team	IntegerField	кількість забитих м’ячів другої команди.

2.4 Опис моделей даних

Model (модель) — це єдине остаточне джерело інформації про ваші дані. Моделі даних в проекті будуть відображати основні сутності, такі як команди, гравці, футбольні матчі, результати та статистику. Наприклад, модель "Команда" буде містити поля, які описують назву команди, логотип, країну та інші деталі. Модель "Матч" буде включати поля, які описують команди, результат, дату, голи, картки та іншу інформацію. Гравці будуть пов'язані з командами, і в моделі "Гравець" будуть зберігатись дані про їхню статистику.

База даних буде проектуватись у відповідності до принципів реляційної моделі. Таблиці бази даних відповідатимуть моделям даних і будуть містити відповідні поля для зберігання інформації про команди, гравців, матчі, результати та статистику. Зв'язки між таблицями будуть встановлені за допомогою ключів, що дозволить забезпечити цілісність даних та зручну навігацію між сутностями. Реляційна база даних забезпечить ефективне зберігання та операції з даними, а також можливість виконання складних запитів для отримання статистики та аналізу даних.

Модель містить основні поля та поведінку даних, які ви зберігаєте. Як правило, кожна модель відображається в одній таблиці бази даних.

Основи:

- кожна модель є класом Python, який є підкласами django.db.models.Model;

- кожен атрибут моделі представляє поле бази даних;
- може мати значення за замовчуванням;
- з усім цим Django надає вам автоматично створений API доступу до бази даних.

2.5 Вибір технологій та інструментів

У роботі обрано фреймворк Django для реалізації серверної частини проекту, оскільки він забезпечує широкий функціонал та високий рівень безпеки.

Django за замовчуванням пропонує готову функціональність для ряду завдань, наприклад, аутентифікації системи, генерації карт сайту тощо, завдяки чому нам не можна ізобрети велосипед і достатньо взяти вже готові компоненти. У Django велика увага приділяється безпеці, завдяки чому фреймворк допомагає розробникам уникнути багатьох поширених проблем у системі безпеки, наприклад, sql-ін'єкцій [7].

Мова програмування Python буде використовуватись для розробки логіки бізнес-процесів та взаємодії з базою даних.

HTML, CSS, JavaScript для розробки клієнтського інтерфейсу будуть використовуватись стандартні веб-технології.

Jinja2 – це швидкий, виразний, розширюваний механізм створення шаблонів. Спеціальні заповнювачі в шаблоні дозволяють писати код, схожий на синтаксис Python.

Шаблону передаються дані для візуалізації остаточного документа. Це включає: спадкування та включення шаблонів, визначати та імпортувати макроси в шаблонах, ізольоване середовище може безпечно відтворювати ненадійні шаблони, асинхронна підтримка для створення шаблонів, які автоматично обробляють синхронізацію та асинхронні функції, винятки вказують на правильний рядок у шаблонах, щоб полегшити налагодження, розширювані фільтри, тести, функції та навіть синтаксис.

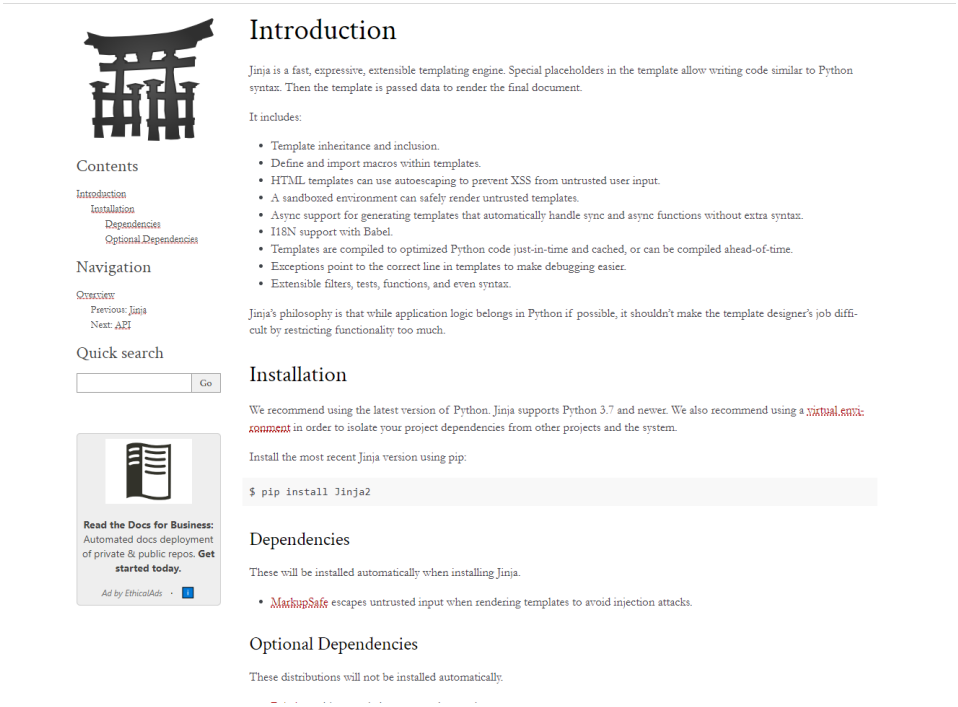


Рисунок 6 – Скріншот з сайту с документацією «Jinja»

База даних для зберігання даних про футбольні матчі та гравців може використовуватись реляційна база даних SQLite (стандартна в Django Framework).

Football-data.org (API) – надає футбольні дані та статистику (рахунки в реальному часі, розклад матчів, таблиці, склади команд, склади/заміни тощо) у машинозчитуваному вигляді.

API – це програмний посередник, який дозволяє двом програмам спілкуватися одна з одною (рис. 7). API – це доступний спосіб витягувати та обмінюватися даними всередині та між організаціями.

Часто реалізується окремою програмною бібліотекою чи сервісом операційної системи. Зазвичай входить до опису будь-якого інтернет-протоколу, програмного каркасу або стандарту викликів функцій операційної системи. Використовується програмістами під час написання різноманітних додатків. Простіше кажучи, це набір компонентів, за допомогою яких комп'ютерна програма (бот або сайт) може використовувати іншу програму.

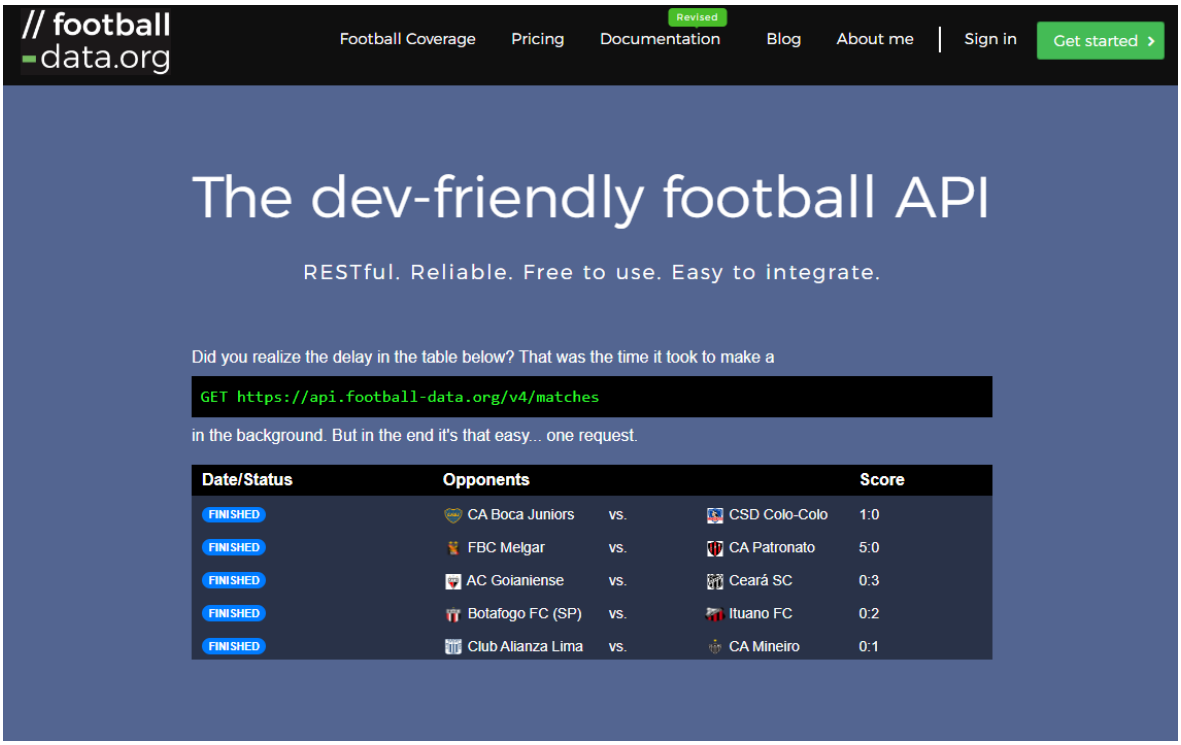


Рисунок 7 – Скріншот API сервісу «football-data»

Bootstrap – це безкоштовний фреймворк CSS із відкритим вихідним кодом, спрямований на адаптивну інтерфейсну веб-розробку, орієнтовану на мобільні пристрої (рис. 8). Він містить шаблони дизайну на основі HTML, CSS і JavaScript для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу.

Основною метою додавання його до веб-проекту є застосування вибору кольору, розміру, шрифту та макета Bootstrap до цього проекту. Таким чином, основним фактором є те, чи відповідальні розробники вважають цей вибір своїм смаком. Після додавання до проекту Bootstrap надає базові визначення стилю для всіх елементів HTML. Результатом є уніфікований вигляд прози, таблиць і елементів форм у веб-переглядачах. Крім того, розробники можуть скористатися перевагами класів CSS, визначених у Bootstrap, щоб додатково налаштувати зовнішній вигляд свого вмісту. Наприклад, Bootstrap підготував таблиці світлого та темного кольорів, заголовки сторінок, більш помітні цитати та текст із виділенням.

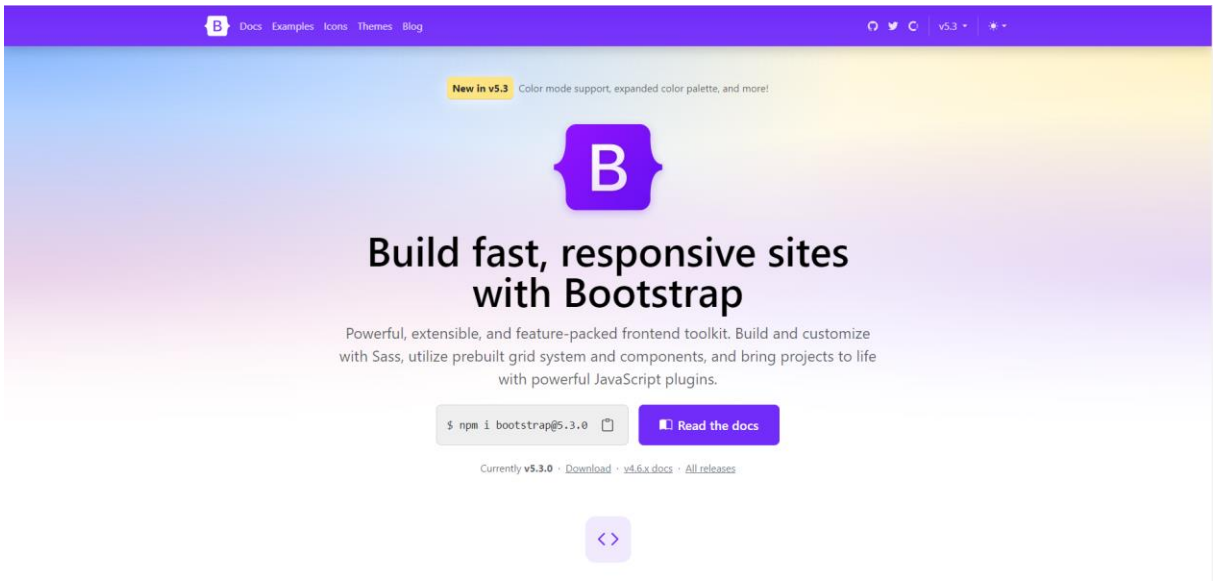


Рисунок 8 – Скріншот сайту фреймворка «Bootstrap»

3 РОЗДІЛ РОЗРОБКИ

3.1 Налаштування в Django

Розглянемо покроково послідовність створення проекту у Django Framework:

1. З командного рядка переходжу до каталогу, де хочу зберегти свій код, а потім виконаю таку команду:

   `$ django-admin startproject djangoUI` – це створить каталог football у моєму поточному каталозі.

   Структура, що створив startproject:

   project/
   - manage.py
   - djangoUI/
     - __init__.py
     - settings.py
     - urls.py
     - asgi.py
     - wsgi.py

2. Потім, нам слід створити програму Django, для цього нам потрібно скористатися командою:

   `$ python manage.py startapp football` – це створить каталог «football», який виглядає так:

   football/
   - __init__.py
   - admin.py
   - apps.py
   - migrations/
     - __init__.py
   - models.py
   - tests.py
   - views.py

3. Також, створення системи контролю версій: Використання системи контролю версій, такої як Git, для відстеження змін коду (рис. 9).

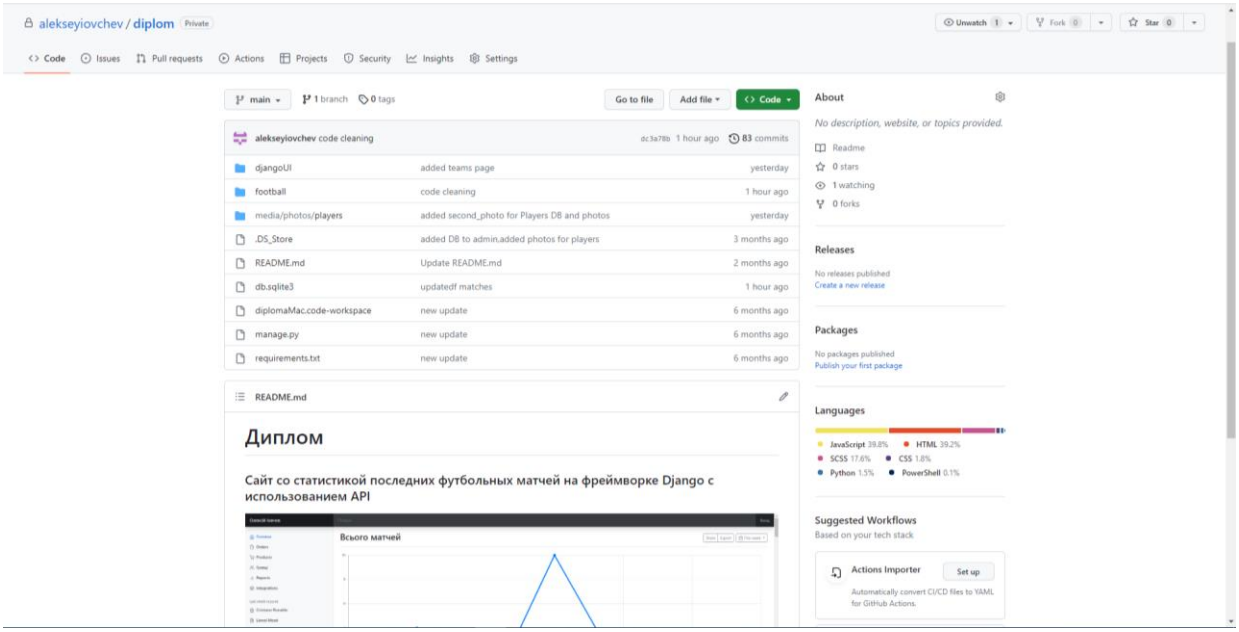


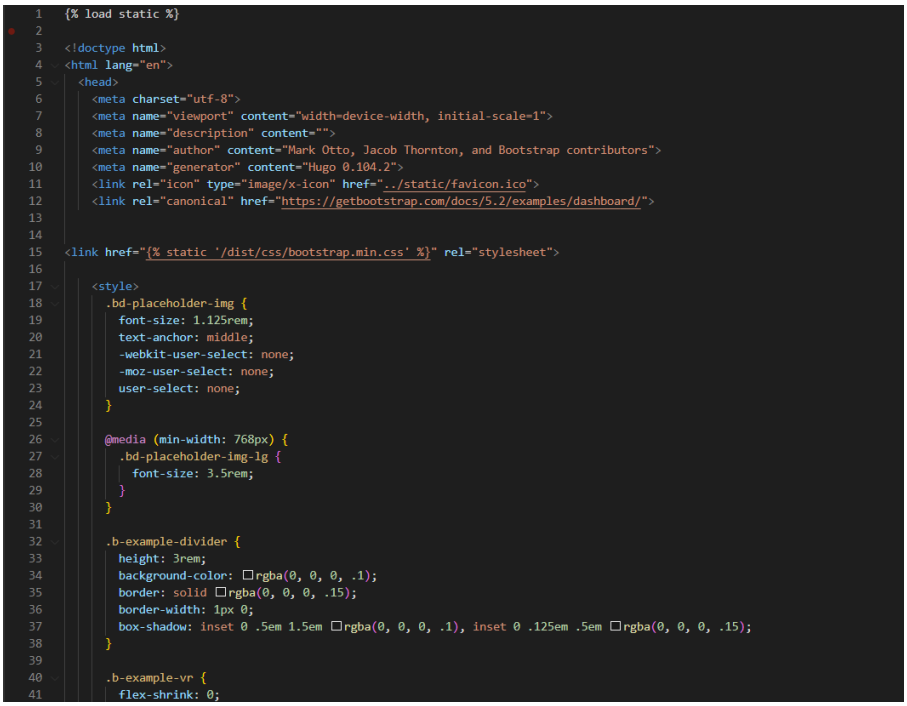
Рисунок 9 – Репозиторій проекту на GitHub.

Ці підготовчі етапи є важливими для забезпечення успішного та ефективного розвитку проекту. Вони допоможуть визначити основні напрямки роботи, встановити необхідні інструменти та підготувати середовище для подальшої розробки системи статистики останніх футбольних матчів на фреймворку Django з використанням API.

3.2 Опис інтерфейсу

Користувацький інтерфейс: буде реалізований у вигляді веб-сторінок, що дозволить користувачам зручно переглядати статистику матчів, шукати необхідну інформацію та взаємодіяти з сайтом.

Адміністративний інтерфейс: буде наданий адміністратору системи для керування матчами, командами та користувачами. Цей інтерфейс дозволить додавати, редагувати та видаляти записи з бази даних.



```
{% load static %}

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
    <meta name="generator" content="Hugo 0.104.2">
    <link rel="icon" type="image/x-icon" href="../static/favicon.ico">
    <link rel="canonical" href="https://getbootstrap.com/docs/5.2/examples/dashboard/">


<link href="{% static '/dist/css/bootstrap.min.css' %}" rel="stylesheet">

    <style>
      .bd-placeholder-img {
        font-size: 1.125rem;
        text-anchor: middle;
        -webkit-user-select: none;
        -moz-user-select: none;
        user-select: none;
      }

      @media (min-width: 768px) {
        .bd-placeholder-img-lg {
          font-size: 3.5rem;
        }
      }

      .b-example-divider {
        height: 3rem;
        background-color: rgba(0, 0, 0, .1);
        border: solid rgba(0, 0, 0, .15);
        border-width: 1px 0;
        box-shadow: inset 0 .5em 1.5em rgba(0, 0, 0, .1), inset 0 .125em .5em rgba(0, 0, 0, .15);
      }

      .b-example-vr {
        flex-shrink: 0;
```

Рисунок 10 – Скрін коду у «base.html»

«base.html» – базовий шаблон який використовується на всіх сторінках та має header(шапка) та меню, та розмітку для подальших відображень інших блоків контенту (має CSS, JS, HTML) (рис .10).

Меню: у верхній частині сторінки розташоване меню, яке містить різні розділи або вкладки для навігації по сайту команди. Це можуть бути розділи, такі як "Головна", "Команди", "Ліга Чемпіонів - 2023", "Гравці" та бистрий пошук з гравцями, такими як “Lionel Messi”, “Cristiano Ronaldo”, “ Neymar Jr”. Це дає можливість користувачам швидко знайти необхідну інформацію про команду.

Шапка: вгорі сторінки розташована шапка, на якій представлені ім’я та призвище виконавця проекту, строка пошуку, та кнопа “Вихід”. Завдяки цій шапці, ми можемо її легко інтегрувати на інші сторінки і користувач може зручно користуватися нею, на всіх сторінках сайту.

Стилі: використовуються для задання зовнішнього вигляду веб-сторінок. Вони використовуються для забезпечення дизайну, форматування та

стилізації веб-елементів, що дає можливість контролювати вигляд і розміщення елементів на сторінці.

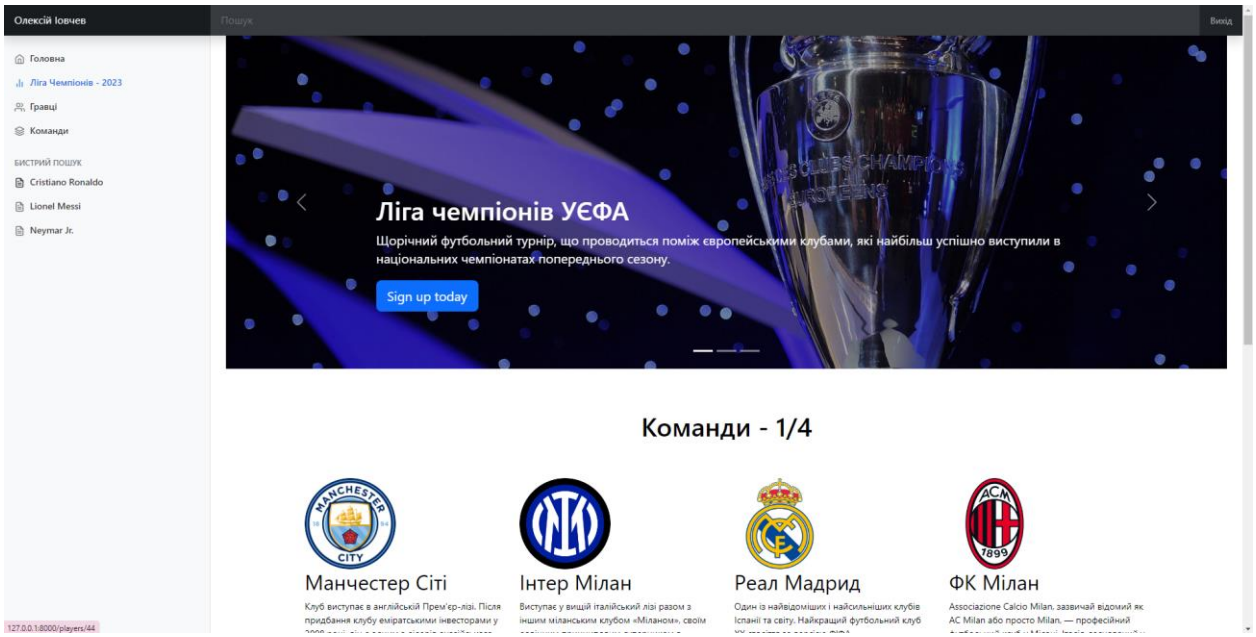


Рисунок 11 – Сторінка «champions_league.html»

«champions_league.html» – відображення сторінки з інформацією о турнірі (рис. 11).

На цій сторінці видно меню, пошук, карусель з фото та інформацією про турнір, команди (також їх коротку біографію) і маємо таблицю на якій видно останні результати матчів турніру.

Отже, сторінка з інформацією про турнір Ліги Чемпіонів 2023 надає користувачам можливість отримати загальний огляд про турнір, ознайомитися з командами та їхніми біографіями, відстежувати результати матчів та зручно навігуватися на сторінці. Вона створює цінний ресурс для шанувальників футболу, журналістів та будь-яких користувачів, які цікавляться цим турніром і бажають бути в курсі його подій. Ця сторінка також може бути корисною для фанатів конкретних команд, які беруть участь у турнірі. Вони зможуть відстежувати результати своїх улюблених команд і отримувати оновлення про їх виступи.

Загалом, ця сторінка забезпечує широкий спектр інформації про турнір і команди, що допомагає задовольнити цікавість та підвищує залучення до футбольного світу.

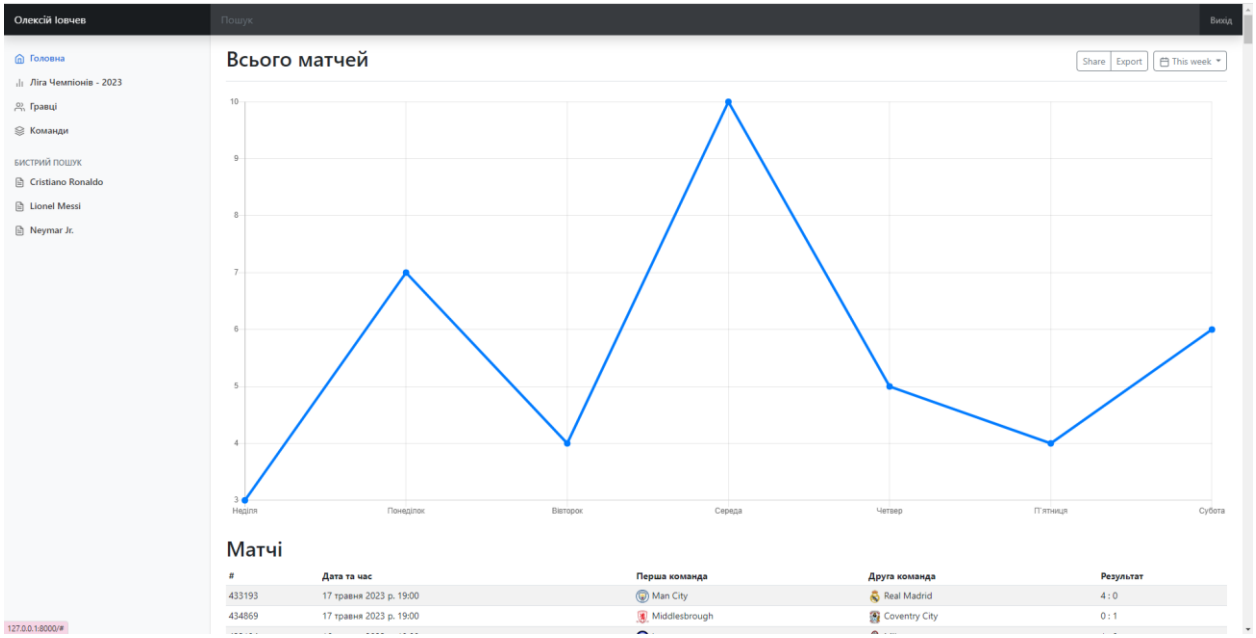


Рисунок 12 – Сторінка «index.html»

«index.html» – головна сторінка з останніми результатами (рис. 12).

Це головна сторінка сайту, на ній у нас також є меню та шапка, контент сторінки складається з графіка кількості останніх матчів за останні 7 днів, а також результати всіх останніх матчів у світі. Враховуючи, що це головна сторінка сайту, переходячи просто за посиланням сайту - ми потраплятимемо на цю сторінку і вже завдяки меню зможемо перейти на інші сторінки сайту. Всі сторінки, як головна сторінки, використовують шаблон base.html, який був розглянутий і описаний вище

У таблиці матчів ми можемо побачити: номер матчу, дату та час, назви команд та їх іконки, а також рахунок матчу. Сортування таблиці походить від останніх матчів згори до старіших внизу.

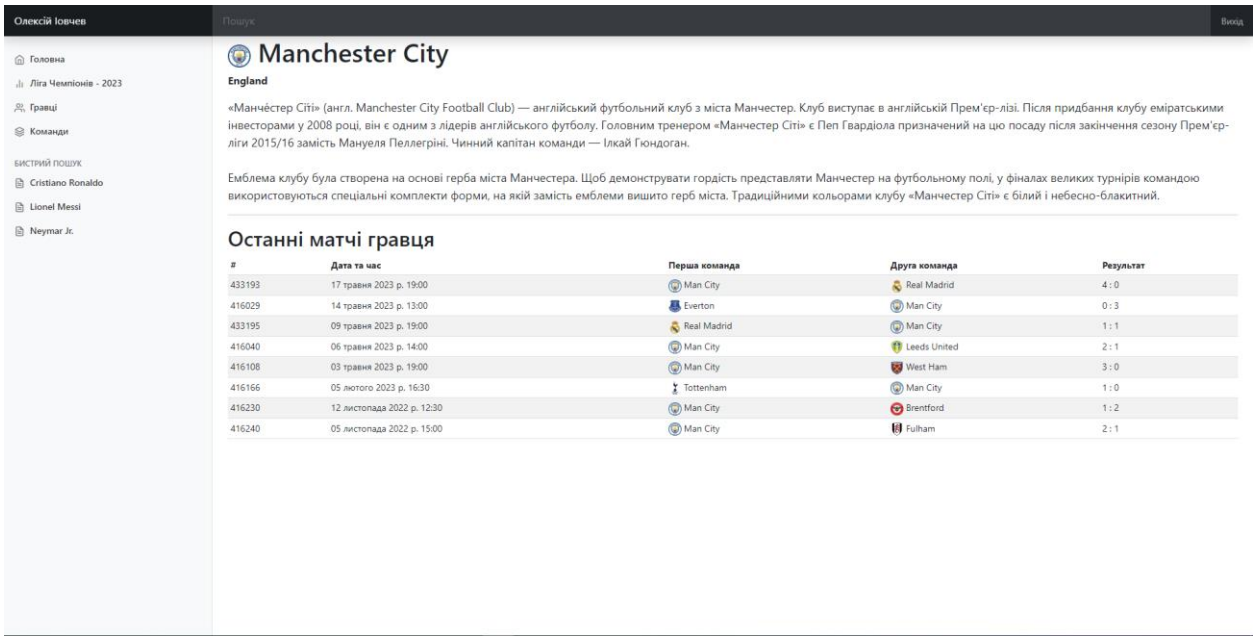


Рисунок 13 – Сторінка «manchester_city.html»

«manchetser_city.html» – відображення сторінки з командою Manchester city (біографія команди, останні результати, логотип команди) (рис. 13).

Це сторінки певної команди - Manchester City, на ній можна ознайомитись з біографією команди, також звідки команда, з її логотипом та найголовніше з останніми результатами команди, з матчами у вигляді таблиці та їх результатами.

Ці елементи сторінки команди Manchester City надають користувачам можливість отримати інформацію про команду, її історію, останні результати та встановити зв'язок з її визначною символікою.

Для шанувальників футболу, журналістів та будь-якого, хто цікавиться цією командою, сторінка є цінним ресурсом для отримання актуальної інформації та взаємодії з командою Manchester City.

«players_post.html» – сторінка з конкретним гравцем (біографія, ім’я, актуальна команда, останні результати) (рис. 14).

Ця сторінка – це сторінка конкретного гравця, по суті це шаблон сторінки гравця і залежно від обраного гравця, вона буде показувати вам інформацію про конкретного гравця, якого ви обрали. На сторінці можна

побачити: ім'я та прізвище, актуальну команду, біографію та таблицю з результатами останніх матчів конкретного гравця. Також, у нас є меню і шапка, як і на інших сторінках сайту.

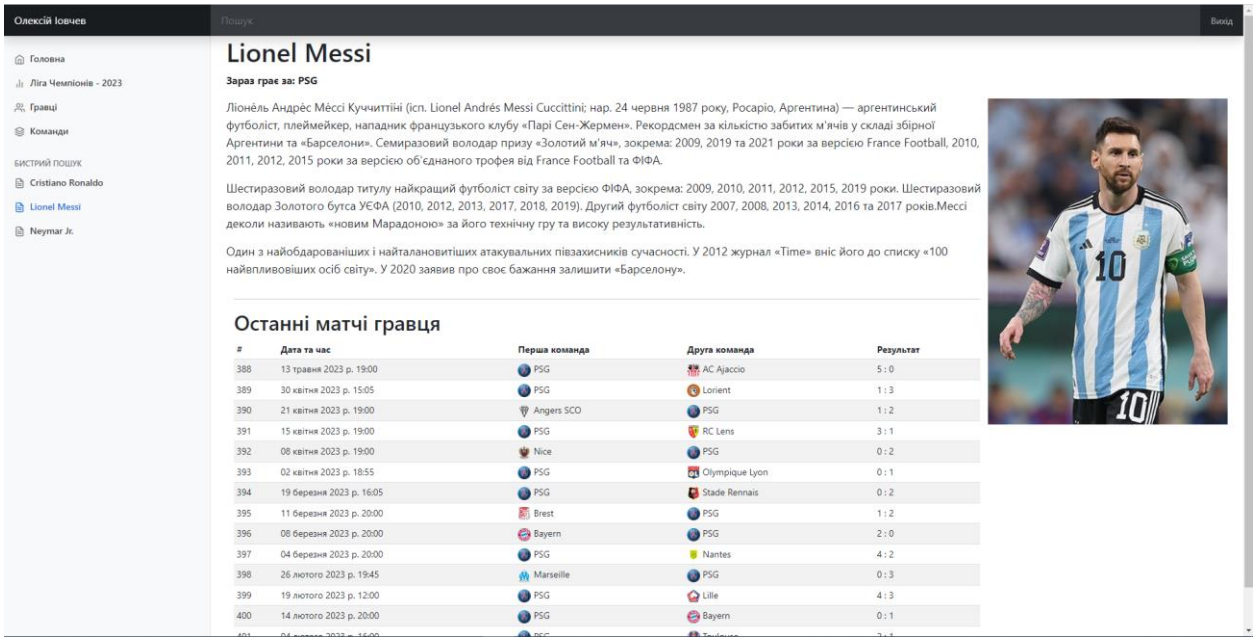


Рисунок 14 – Сторінка «players_post.html»

Сторінка з конкретним гравцем надає більш детальну інформацію про нього, його кар'єру та останні результати. Це дозволяє шанувальникам футболу отримати знання про гравця та слідкувати за його виступами на полі.

Таким чином, сторінка з конкретним гравцем є цінним джерелом інформації для шанувальників футболу, журналістів та будь-якого, хто хоче дізнатися більше про визначного гравця та його внесок у світ футболу.

«players.html» – сторінка з списком гравців та короткою біографію (рис. 15). Ця сторінка зі списком гравців, де ви можете побачити заголовок, невеликий опис сторінки, меню, шапку, кнопки, що переводять вас на інші сторінки і найголовніше картки гравців з їхніми фотографіями, невелику біографію, дату народження та кнопок для більш повного ознайомлення (перехід на сторінку players_post.html, показана інформація буде відображена залежно від вибраного гравця).

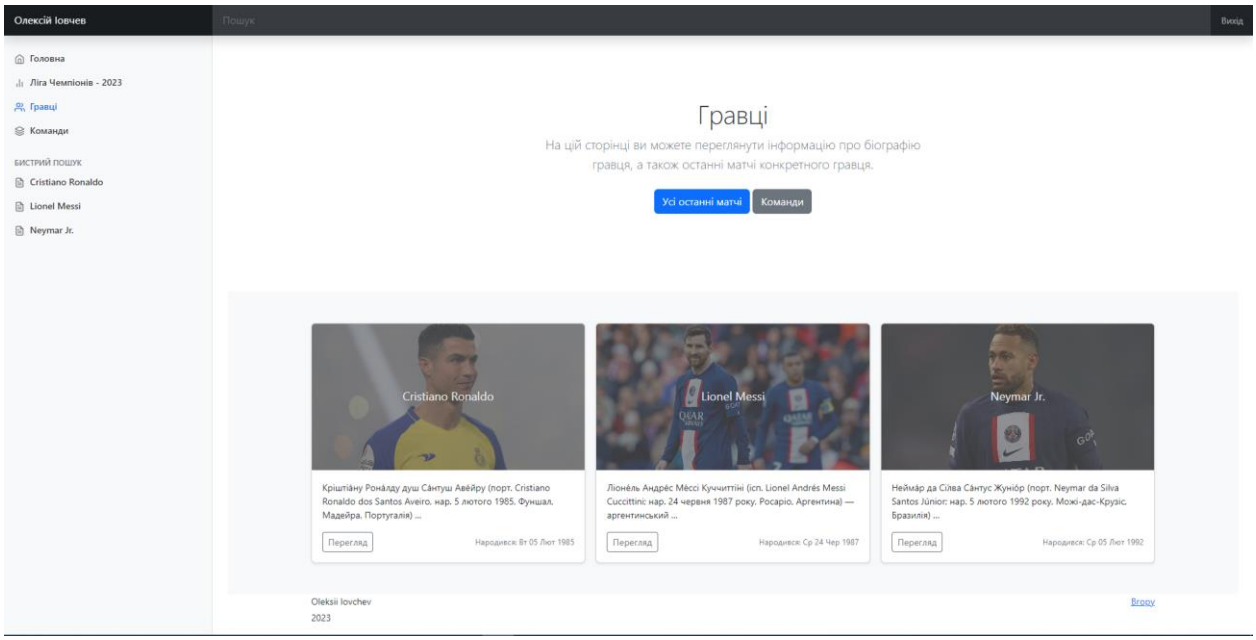


Рисунок 15 – Сторінка «players.html»

На цій сторінці також використовується додаткове фото з бази даних на картках кожного з гравців. Додавати нових гравців на цю сторінку можна як на пряму через базу даних, так і через адмін-панель.

Таким чином, сторінка зі списком гравців та короткою біографією створює зручний огляд складу команди та надає шанувальникам футболу можливість ознайомитися з гравцями, їхніми фотографіями та основними відомостями про них. Вона сприяє залученню користувачів та створює зручність у навігації на сторінці, щоб користувачі могли отримати більш повну інформацію про гравців та взаємодіяти з командою більш ефективно.

«real_madrid.html» – відображення сторінки з командою Real Madrid (біографія команди, останні результати, логотип команди) (рис. 16).

Ця сторінка ідентична сторінки з Manchester City, на ній вся така ж іноформація у вигляді меню, шапки, біографії, країни команди та таблиці результатів, але вже про Real Madrid.

Описані елементи сторінки команди Real Madrid надають користувачам можливість дізнатися більше про команду, її історію, досягнення та останні результати.

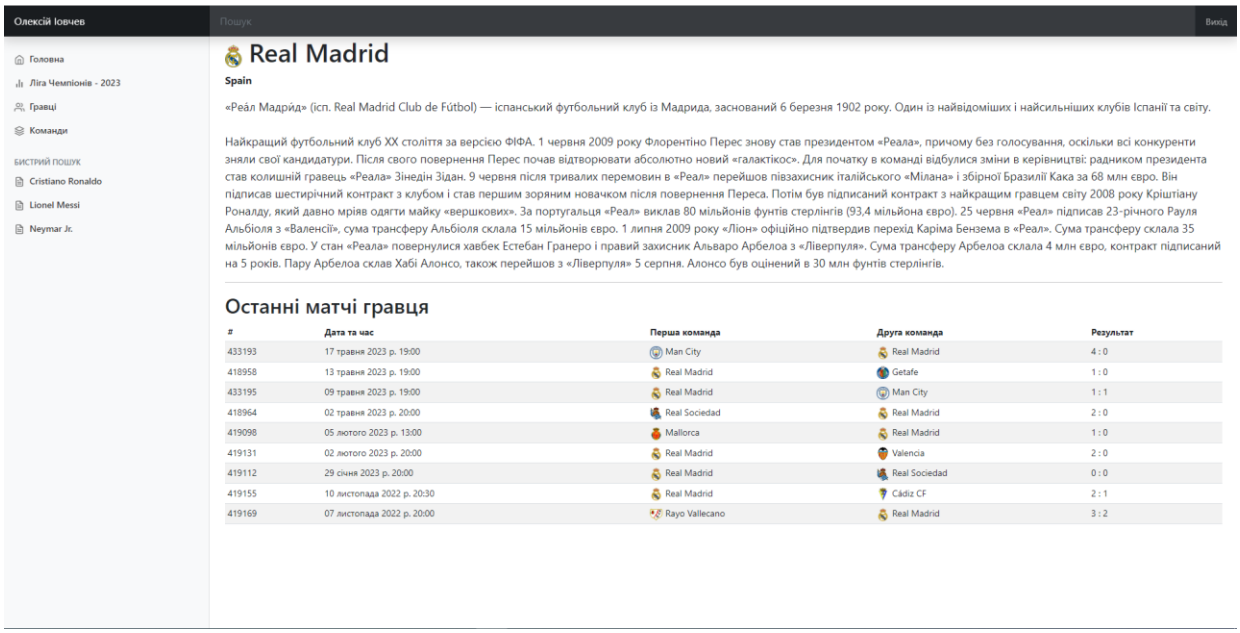


Рисунок 16 – Сторінка «real_madrid.html»

Остання сторінка teams.html схожа на сторінку players.html, тому що тут також перелік, але вже не гравців, а команд. На сторінці є: меню, шапка, заголовок першого рівня, заголовок другого рівня (невелика інформація про сторінку), кнопки переходу на інші сторінки і картки команд з їх логотипом, назвою, невеликою біографією і кнопкою переходу далі для більшого ознайомлення з командою.

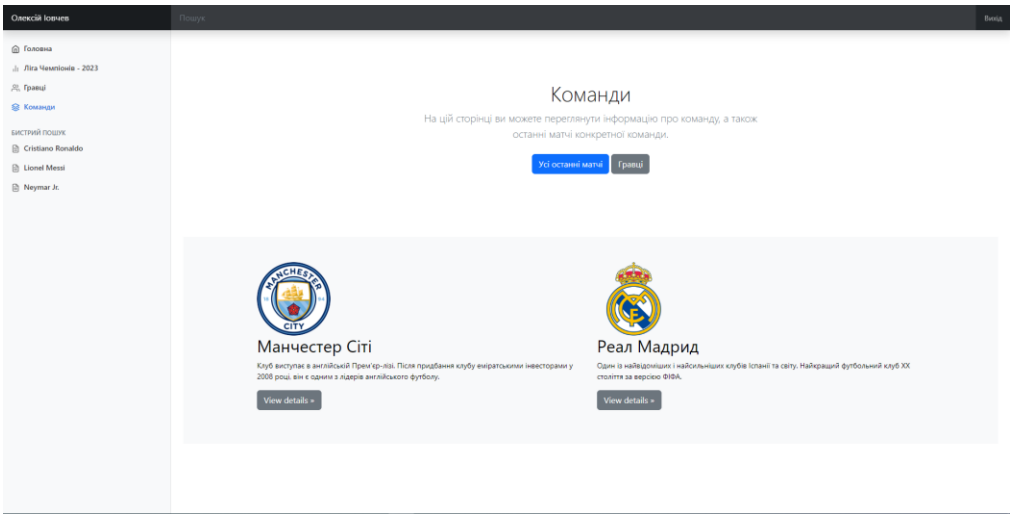


Рисунок 17 – Сторінка «teams.html»

«teams.html» – сторінка з списком команд та інформацією о командах.

Загалом, сторінка зі списком команд та їх інформацією створює зручну та інформативну платформу для користувачів, які цікавляться футболом та бажають ознайомитися з різними командами більш детально. Вона надає компактний та впорядкований перегляд команд, дозволяючи користувачам швидко знайти необхідну інформацію та глибше досліджувати команди, які їх зацікавили.

3.3 Реалізація моделей даних

Розберемо таблиці розглянуті у пункті 3.2, створення таблиць відбувається у файлі models.py, використовуючи бібліотеки Django та клас, який успадковується від model.Model, який і був імпортований. прописуючи у вигляді змінних додаються нові поля в нову таблицю, які ми розглянули вище.

Також, у нас у всіх класах є функція "__str__" – вона використовується для зручного відображення її в адмін панелі, а також клас "Meta" – він також використовується для перетворення назв в адмін-панелі та сортування таблиці на сторінках.

Таблиця «Scores»:

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

# Create your models here.
class Scores(models.Model):
    id = models.IntegerField(primary_key=True)
    first_team = models.CharField(max_length=30)
    first_team_icon = models.URLField(max_length =
200,default='null')
    second_team = models.CharField(max_length=30)
    second_team_icon = models.URLField(max_length =
200,default='null')
    match_date = models.DateTimeField()
    score_first_team = models.IntegerField(default=0)
    score_second_team = models.IntegerField(default=0)

    def __str__(self):
```

```
            return "{} -
{}".format(self.first_team,self.second_team)

        class Meta:
            verbose_name = 'Матч'
            verbose_name_plural = 'Все матчи'
            ordering = ['-match_date']
```

Це таблиця всіх останніх матчів, вона є основною таблицею сайту.

Таблиця «Players»:

```
    class Players(models.Model):
        id = models.IntegerField(primary_key=True)
        name = models.CharField(max_length=30)
        bio = models.TextField(null=True)
        date_of_birth = models.DateTimeField()
        nationality = models.CharField(max_length=30)
        current_team = models.CharField(max_length=30,null=True)
        photo =
models.ImageField(upload_to='photos/players/',default='url')

second_photo=models.ImageField(upload_to='photos/players/',defau
lt='url')

        def get_absolute_url(self):
            return reverse('players_post',
kwargs={'players_id':self.id})

        def __str__(self):
            return self.name

        class Meta:
            verbose_name = 'Игрок'
            verbose_name_plural = 'Игроки'
            ordering = ['name']
```

Таблиця з інформацією про гравців. Таблиця «Player_matches»:

```
    class Player_matches(models.Model):
        player = models.ForeignKey(Players,
on_delete=models.CASCADE)
        match_id = models.IntegerField()
        first_team = models.CharField(max_length=30)
        first_team_icon = models.URLField(max_length =
200,default='null')
        second_team = models.CharField(max_length=30)
```

```
        second_team_icon = models.URLField(max_length =
200,default='null')
        match_date = models.DateTimeField()
        score_first_team = models.IntegerField(default=0)
        score_second_team = models.IntegerField(default=0)

        def __str__(self):
            return "{} -
{}".format(self.first_team,self.second_team)

        class Meta:
            verbose_name = 'Матч игрока'
            verbose_name_plural = 'Матчи игроков'
            ordering = ['player']
```

Таблиця матчів гравців, ця таблиця використовується для відображення останніх матчів конкретних гравців.

Таблиця «Chempions_League_Scores»:

```
    class Champions_League_Scores(models.Model):
        id = models.IntegerField(primary_key=True)
        first_team = models.CharField(max_length=30)
        first_team_icon = models.URLField(max_length =
200,default='null')
        second_team = models.CharField(max_length=30)
        second_team_icon = models.URLField(max_length =
200,default='null')
        match_date = models.DateTimeField()
        score_first_team = models.IntegerField(default=0)
        score_second_team = models.IntegerField(default=0)

        def __str__(self):
            return "{} -
{}".format(self.first_team,self.second_team)

        class Meta:
            verbose_name = 'Матч ЛЧ'
            verbose_name_plural = 'Все матчи ЛЧ'
            ordering = ['-match_date']
```

У цій таблиці зберігаються всі останні матчі турніру “Ліга Чемпіонів 2023”.

3.4 Розробка функціональності сайту

Конфігурація Django проекту (application):

«urls.py» – Щоб створити URL-адреси для програми, ви створюєте модуль Python, який неофіційно називається URLconf (конфігурація URL-адреси). Цей модуль є чистим кодом Python і є відображенням між виразами URL-шляху функціями Python (вашими представленнями).

Це відображення може бути як коротким, так і довгим, скільки потрібно. Він може посилатися на інші відображення. І оскільки це чистий код Python, його можна створювати динамічно.

```
from django.contrib import admin
    from django.urls import path
    from football.views import *
    from django.conf import settings
    from django.conf.urls.static import static

    urlpatterns = [
        path('admin/', admin.site.urls),
        path('', index, name = 'home'),
        path('players/', players, name = 'players'),
        path('teams/', teams, name = 'teams'),
        path('teams/manchester_city', manchester_city, name =
'manchester_city'),
        path('teams/real_madrid', real_madrid, name =
'real_madrid'),
        path('players/<int:players_id>/', show_player,
name='players_post'),
        path('champions_league_2023/', champions_league_2023,
name = 'champions_league_2023'),
    ]

    if settings.DEBUG:
        urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

«settings.py» – Файл налаштувань Django містить усі налаштування вашої інсталяції Django. У цьому документі пояснюється, як працюють налаштування та які налаштування доступні.

Фреймворк Django реалізує архітектурний патерн Model-View-Template або скорочено MVT, який фактично є модифікацією розпростертого у веб-програмуванні шаблону MVC (Model-View-Controller).

Основні елементи паттерна:

- URL-диспетчер: при отриманні запиту на підставі адреси запрошення URL визначає, який ресурс повинен обробляти даний запит.
- View (перегляд): отримує запит, обробляє його і відправляє у відповідь користувачеві інший відповідь. Якщо для обробки запиту необхідно звернення до моделі та бази даних, то View взаємодіє з ними. Для створення відповіді можна застосувати Шаблон або шаблони. В архітектурі MVC цьому компонентя відповідають контролери (но не представлені).
- Model (модель): описує дані, використовувані в приложении. Окремі класи, як правило, відповідають таблицям в базі даних.
- Template (шаблон): представляє логіку представлення у вигляді згенерованої розмітки html. В MVC цей компонент відповідає View, то є представлення.

Розглянемо представлення MVT в проекті:

Views («views.py» – Функція view) – це функція Python, яка приймає веб-запит і повертає веб-відповідь. Цією відповіддю може бути HTML-вміст веб-сторінки, перенаправлення, помилка 404, документ XML або зображення тощо. Саме подання містить будь-яку довільну логіку, необхідну для повернення відповіді.

```
from django.shortcuts import render,
get_object_or_404,HttpResponse

    from django.db.models import Q

    from football.models import Scores, Players,
Player_matches,Champions_League_Scores
```

```
    # Create your views here.

    def index(request):
        return render(request, 'football/html/index.html', {
'data' : Scores.objects.all().order_by('-match_date') })
    def players(request):
        return render(request, 'football/html/players.html',{
'data' : Players.objects.all()})

    def teams(request):
        return render(request, 'football/html/teams.html',)

    def manchester_city(request):
        return render(request,
'football/html/manchester_city.html',{ 'matches' :
Scores.objects.filter(Q(first_team='Man City') |
Q(second_team='Man City'))})

    def real_madrid(request):
        return render(request,
'football/html/real_madrid.html',{ 'matches' :
Scores.objects.filter(Q(first_team='Real Madrid') |
Q(second_team='Real Madrid'))})

    def champions_league_2023(request):
        return render(request,
'football/html/champions_league.html',{ 'data' :
Champions_League_Scores.objects.all().order_by('-match_date')})

    def show_player(request,players_id):
        post = get_object_or_404(Players, pk=players_id)
        matches =
Player_matches.objects.filter(player=players_id).order_by('-
match_date')
        context = {
            'post':post,
            'matches':matches,
            'active':str(post.pk)
            }

        return
render(request,'football/html/players_post.html',context=context
)
```

Функції та їх опис:

- index: головна сторінка, відображення шаблону index.html та виклик таблиці Scores;
- players: сторінка зі списком гравців, відображення шаблону players.html та виклик таблиці Players;

- teams: сторінка зі списком команд, відображення шаблону teams.html;
- manchester_city: сторінка з інформацією команди Manchester City, відображення шаблону manchester_city.html, виклик таблиці Scores, але тільки з фільтром матчів в яких грала команда Manchester City;
- real_madrid: сторінка з інформацією команди Real Madrid, відображення шаблону real_madrid.html, виклик таблиці Scores, але тільки з фільтром матчів в яких грала команда Real Madrid;
- champions_league: сторінка з інофрмацією про турнір, відображення шаблону champions_leage.html та виклик таблиці Champions_League_Scores та сортування за останніми матчами;
- show_players: функція-шаблон, яка відображає інформацію про конкретно обраного гравця, в цій функції є логіка, яка орієнтуючись на айді обраного гравця і підставляє про нього інформацію у шаблон, відображення шаблону players_post.html, виклик таблиці в залежності від обраного(айді) гравця.

Усі функції в даному файлі відповідають за відображення html сторінки та видачу її клієнту, а перенаправлення відбувається за рахунок файлу, який ми розібрали вище urls.py там і використовується ці функції.Template (html шаблони) – Django визначає стандартний API для завантаження та відтворення шаблонів незалежно від серверної частини. З

авантаження складається з пошуку шаблону для заданого ідентифікатора та його попередньої обробки, зазвичай компіляції до представлення в пам’яті. Рендеринг означає інтерполяцію шаблону контекстними даними та повернення отриманого рядка.

Рекомендується створювати папку templates для зберігання там шаблонів html, які ми потім використовуємо у view.py:

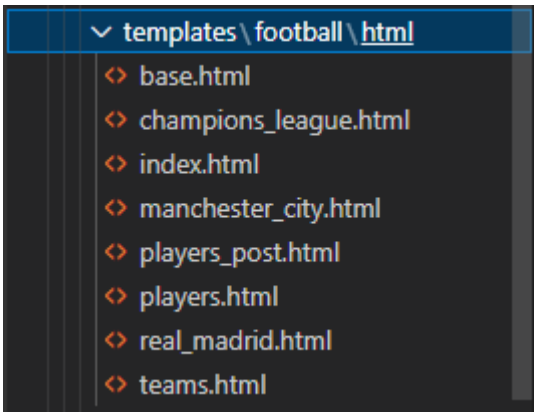


Рисунок 18 – Структура папки «templates»

Також, в html коді ми застосовуємо шаблонізатор Jinja, для роботи з базою даних та змінними в «views.py». Приклад: index.html

```
    {% for match in data %}
                <tr>
                  <td>{{match.id}}</td>
                  <td>{{match.match_date}}</td>
                  <td><img src="{{ match.first_team_icon }}"
width="20" height="20"/> {{match.first_team}}</td>
                  <td><img src="{{ match.second_team_icon }}"
width="20" height="20"/> {{match.second_team}}</td>
                  <td>{{match.score_first_team}} :
{{match.score_second_team}}</td>
                </tr>
        {% endfor %}
```

{% for match in data %} – є циклом для використання кожної записі з таблиці Scores, яку ми налаштовуємо в «views.py», як змінну “data” та відображуємо всі результати у таблиці, які у нас і зберігаються у базі даних. Як це виглядає у «views.py»:

```
    def index(request):
        return render(request, 'football/html/index.html', {
'data' : Scores.objects.all().order_by('-match_date') })
```

Саме `{ 'data' : Scores.objects.all().order_by('-match_date') })` передає інформацію з таблиці до змінної data, яка потім проходячи через цикл відображає кожен матч у таблиці.

3.5 Інтеграція з API для отримання статистики

Для роботи з API використовується сервіс «https://www.football-data.org/», після реєстрації на сервісі ми отримуємо API ключ, який нам дає доступ для надсилання GET запитів та отримання інформації у JSON форматі.

Технічно функції із запитами розділені на 4 різні файли (скрипти) і інтергровані в один головний скрипт:

1. «scores_main.py»
2. «players_main.py»
3. «champions_league.py»
4. «scripts_main.py»

Розберемо кожен з них:

1. «scores_main.py» – має функцію run_scores(), яка робить запит через API до сервісу результатів матчів, після чого він обробляє JSON відповідь і завантажує в нашу вже створену таблицю Scores (всі останні матчі).

```
    def run_scores(result):
        for match in result:
            print(Scores.objects.filter(id=match['id']))
            if not Scores.objects.filter(id=match['id']):
                Scores.objects.create(
                    id=match['id'],
                    first_team=match['homeTeam']['shortName'],
                    first_team_icon =
match['homeTeam']['crest'],
                    second_team=match['awayTeam']['shortName'],
                    second_team_icon =
match['awayTeam']['crest'],
                    match_date=match['utcDate'],

score_first_team=int(match['score']['fullTime']['home']),
```

```
score_second_team=int(match['score']['fullTime']['away']))
```

2. «players_main.py» – має дві функції, перша create_player() - створює нового гравця в таблиці Players, а друга player_matches() - додає все нові матчі гравця в таблицю Player_matches.

```
    def create_player(result):
        if not Players.objects.filter(id=result['id']):

Players.objects.create(id=result['id'],name=result['name'],date_
of_birth=result['dateOfBirth'],nationality=result['nationality']
)

    def player_matches(result,id):
        for data in result:

     if not Player_matches.objects.filter(match_id=data['id']):
                Player_matches.objects.create(
                    player= Players(id=id),
                    match_id=data['id'],
                    first_team=data['homeTeam']['shortName'],
                    first_team_icon = data['homeTeam']['crest'],
                    second_team=data['awayTeam']['shortName'],
                    second_team_icon =
data['awayTeam']['crest'],
                    match_date=data['utcDate'],

score_first_team=int(data['score']['fullTime']['home']),

score_second_team=int(data['score']['fullTime']['away']))
```

3. «champions_league.py» - має функцію, яка додає всі останні матчі турніру (Ліга Чемпіонів) до таблиці Champions_League_Scores.

```
    def run_Champions_League_Scores(result):

        for match in result:

            if not
Champions_League_Scores.objects.filter(id=match['id']):
                Champions_League_Scores.objects.create(
                    id=match['id'],
                    first_team=match['homeTeam']['shortName'],
                    first_team_icon =
match['homeTeam']['crest'],
```

```
                    second_team=match['awayTeam']['shortName'],
                    second_team_icon =
match['awayTeam']['crest'],
                    match_date=match['utcDate'],

score_first_team=int(match['score']['fullTime']['home']),

score_second_team=int(match['score']['fullTime']['away']))
```

4. «scripts_main.py» – найголовніший скрипт, саме в цьому скрипті відбувається імпорт усіх попередніх файлів (скриптів), виклик всіх функцій і всі запити через API, що використовуються в функціях.

У даному скрипті ми маємо url, api key та інші змінні для роботи з API та функціями, розглянемо цей скрипт:

```
import scores_main
import players_main
import champions_league_script
import requests
from datetime import *

from football.models import Scores,Players

host = 'api.football-data.org/v4'
api = {'X-Auth-Token':'b5611168bafe4dd2a3fcc7b6b7e19e9a'}

# Scores
dateFrom = (datetime.now() -
timedelta(days=10)).strftime("%Y-%m-%d")

dateTo = datetime.now().strftime("%Y-%m-%d")
games_url =
f'http://{host}/matches/?dateFrom={dateFrom}&dateTo={dateTo}'

chempions_league_url =
f'https://{host}/matches?competitions=2001&status=FINISHED&dateF
rom={dateFrom}&dateTo={dateTo}'

def get_info(url,data=None):
    request = requests.get(url,headers=api)
    result = request.json()[data]
    return result

if __name__ == "__main__":
    ## For whole scores
    try:
```

```
    scores_main.run_scores(get_info(games_url,data='matches'))

champions_league_script.run_Champions_League_Scores(get_info(che
mpions_league_url,data='matches'))

        except KeyError:
            pass

        ### For Players matches
        ids = list(Players.objects.all().values_list('id',
flat=True))

        for id in ids:
            player_url =
f'https://{host}/persons/{id}/matches?status=FINISHED'

            try:

players_main.create_player(get_info(player_url,data='person'))

players_main.player_matches(get_info(player_url,data='matches'),
id)

            except KeyError:
                pass
```

Змінні «host» та «api» – відповідають за посилання до сервісу та за API ключ підключений до сервісу та для доступу до БД сервісу.

Функція get_info() – функція, при виклику, яка робить запит до сервісу через API.

Потім викликаються всі функції, які були в скриптах вище, можна помітити, що ми заздалегідь зробили імпорт всіх файлів для використання цих функцій. У них ми надаємо запит через функцію get_info() і потрібні нам атрибути: просто посилання, або посилання з айді гравця (для гравців), або посилання з айді турніру (для турнірів).

Після запуску даного скрипту безпосередньо через python3, у нас автоматично оновлюватимуться всі таблиці, додавши нові актуальні матчі за останній час, таким чином одним запуском або одною кнопкою ми оновили всі таблиці новими результатами матчів та об’єднали всі скріпти до купи в один головний скрипт.

3.6 Розробка адміністративного інтерфейсу

Адміністративний інтерфейс є важливою частиною проекту, оскільки він надає можливість адміністратору здійснювати управління системою, додавати, редагувати та видаляти дані, а також здійснювати контроль над користувачами та їхніми правами. Нижче наведено кроки, які будуть виконувались під час розробки адміністративного інтерфейсу:

1. Визначення функціональності: Перш за все, необхідно визначити, які операції адміністратор повинен мати змогу виконувати у системі. Це можуть бути такі дії, як додавання/редагування/видалення команд, гравців, матчів, перегляд статистики та інше.

2. Розробка інтерфейсу: Django надає вбудовану адміністративну панель, яка є потужним інструментом для управління даними та налаштування системи.

3. Налаштування автентифікації та авторизації: у Django є поняття "superuser" - це особливий користувач, який має повні права доступу до адміністративної панелі та може виконувати всі дії, пов'язані з управлінням системою. Налаштування та створення superuser в Django зазвичай виконується на початковому етапі розробки проекту [8].

4. Реалізація логіки: Розробники реалізовують логіку, яка дозволяє виконувати необхідні операції в адміністративному інтерфейсі. Це включає обробку даних, валідацію введених значень, зберігання змін до бази даних та відображення повідомлень про стан операцій.

5. Тестування та налагодження: Після реалізації адміністративного інтерфейсу проводяться тестування, щоб переконатись у його коректній роботі та відповідності вимогам.

Розробка адміністративного інтерфейсу є важливим етапом проекту, оскільки він забезпечує зручність та ефективність управління системою для

адміністраторів. Правильна розробка та налагодження цього інтерфейсу дозволить забезпечити зручну та безпроблемну роботу з системою.

ВИСНОВОК

У вступній частині були обґрунтовані контекст і актуальність роботи, зазначено необхідність створення такого сайту, оскільки футбол є популярним видом спорту, а доступ до актуальної статистики матчів є важливим для багатьох фанатів та професіоналів.

У процесі розробки були визначені завдання, які включали розробку бази даних для зберігання інформації про матчі, розробку веб-інтерфейсу для користувачів та адміністраторів, розгортання проекту на веб-сервері та проведення тестування.

В аналізі вимог були визначені функціональні та нефункціональні вимоги до системи, а також вимоги до інтерфейсу, бази даних, безпеки, розгортання та інтеграції. Ці вимоги допомогли визначити основні функціональності та властивості системи, а також забезпечити її стабільну та безпечну роботу.

У проектуванні було визначено архітектуру веб-додатку, описано моделі даних для зберігання інформації про матчі, вибрано технології та інструменти для розробки, а також було розроблено структуру бази даних для виконання необхідних операцій.

Під час розробки були виконані підготовчі етапи, такі як налаштування середовища розробки, створення проекту Django та встановлення необхідних залежностей. Далі було реалізовано функціонал, пов'язаний з отриманням даних зі зовнішнього API, обробкою та збереженням їх у базі даних.

У розділі про впровадження та розгортання були описані кроки, пов'язані з підготовкою середовища для розгортання, конфігурацією сервера та бази даних, розгортанням проекту на сервері та проведенням тестування в робочому середовищі.

Загальною метою було розробити функціональний та ефективний веб-додаток, який надає користувачам зручний доступ до статистики останніх футбольних матчів. Використання фреймворку Django та API дозволяє

отримати актуальну інформацію та забезпечити зручність використання для користувачів. Результатом роботи є розроблений і розгорнутий сайт зі статистикою футбольних матчів, який задовольняє вимоги та потреби користувачів.

СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

1. Why is football so popular? The real reason football is the most popular sport in the world. URL: https://sportsbrief.com/football/41203-why-football-popular-real-reason-football-popular-sport-world/ (дата звернення 27.04.2023)

2. Футбольні результати онлайн – live результати, рахунки футбольних матчів, турнірні таблиці. URL: https://www.flashscore.ua/(дата звернення 27.04.2023)

3. Why Are Football Matches Getting Popular with the Time? URL: https://ventsmagazine.com/2022/04/29/why-are-football-matches-getting-popular-with-the-time/ (дата звернення 03.05.2023)

4. Футбольний сервис статистики и аналітики. URL: https://ofstats.com/ (дата звернення 03.05.2023)

5. UML для бізнес-моделювання: для чого потрібні діаграми процесів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення 09.05.2023)

6. Л.С. Глоба, Т. М. Кот. Розробка інформаційних ресурсів та систем. Київ: НТУУ «КПІ», 2014. -318 с.

7. Django MVT. URL: https://www.javatpoint.com/django-mvt (дата звернення 13.05.2023)

8. Django documentation. URL: https://docs.djangoproject.com/ (дата звернення 18.05.2023)

9. Django admin. URL: https://tutorial.djangogirls.org/en/django_admin/ (дата звернення 25.05.2023)