Міністерство освіти і науки України

Національний університет «Одеська політехніка»

Інститут комп’ютерних систем

Кафедра інформаційних систем

Іванов Віталій Андрійович,

Канарський Анатолій Сергійович

студенти групи АІ-211

ДИСЦИПЛІНА

Об’єктно-орієнтоване програмування

КУРСОВА РОБОТА

Розробка веб-додатку для слідкування за англійською прем'єр-лігою

Спеціальність:

122 Комп’ютерні науки

Освітня програма:

Комп’ютерні науки

Керівник:

Годовиченко Микола Анатолійович,

кандидат технічних наук, доцент

Одеса – 2023

ЗМІСТ

[1 Огляд систем-аналогів та технологій їх розробки](#_Toc137765214) 6

[1.1 Особливості використання веб-технологій для слідкування за Англійською прем'єр-лігою (EPL)](#_Toc137765215) 6

[1.2 Огляд додатків для ведення заміток](#_Toc137765216) 7

[1.2.1 Веб-додаток Premier League](#_Toc137765217) 8

[1.2.2 Веб-додаток ESPN 10](#_Toc137765218)

[1.3 Формування вимог до основних функцій додатку 11](#_Toc137765219)

[1.4 Огляд інформаційних технологій для розробки веб-додатку 12](#_Toc137765220)

[1.4.1 Фреймворк Spring Boot 12](#_Toc137765221)

[1.4.2 Football Data API 14](#_Toc137765222)

[1.4.3 Фреймворк Spring Security 15](#_Toc137765223)

[1.4.4 Thymeleaf 16](#_Toc137765224)

[1.4.5 JPA 17](#_Toc137765225)

[1.4.6 Lombok 19](#_Toc137765226)

[1.4.7 Hibernate 20](#_Toc137765227)

[1.4.8 Java 21](#_Toc137765228)

[1.4.9 Bootstrap 5 22](#_Toc137765229)

[1.4.10 Html 23](#_Toc137765230)

[1.4.11 MySQL 23](#_Toc137765231)

[1.5 Висновки до першого розділу 24](#_Toc137765232)

[2 проєктування додатку 2](#_Toc137765233)5

[2.1 Мета та задачі додатку 2](#_Toc137765234)5

[2.2 Визначення функціональних вимог до додатку 27](#_Toc137765235)

[2.3 Формування користувацьких історій додатку 28](#_Toc137765236)

[2.4 Визначення нефункціональних вимог до додатку 30](#_Toc137765237)

[2.5 Ідентифікація архетипу веб-додатку 31](#_Toc137765238)

[2.6 Проектування користувацького інтерфейсу веб-додатку 31](#_Toc137765239)

[2.7 Висновки до другого розділу 3](#_Toc137765240)2

[3 ПРОГРАМНА РЕАЛІЗАЦІЯ додатку для ведення заміток 3](#_Toc137765241)3

[3.1 Структура серверного програмного проєкту 34](#_Toc137765242)

[3.2 Діаграма класів веб-ресурсу 36](#_Toc137765243)

[3.3 Керування вихідним кодом веб додатку 36](#_Toc137765244)

[3.4 Функціональне тестування розробленого веб-ресурсу 39](#_Toc137765245)

[3.5 Інструкція користувача веб-ресурсу 42](#_Toc137765246)

[3.6 Вихідний код веб-ресурсу 47](#_Toc137765247)

[3.7 Висновки до третього розділу 48](#_Toc137765248)

[ВИСНОВКИ 4](#_Toc137765249)7

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ](#_Toc137765250) 49

АНОТАЦІЯ

Ця курсова робота присвячена розробці веб-додатку для слідкування за Англійською прем'єр-лігою (EPL). Метою цього проекту є створення зручного та функціонального інструменту, який дозволить користувачам бути в курсі останніх новин, розкладу матчів, результатів, статистики та інших статей, пов'язаних з EPL. В ході розробки веб-додатку будуть використовуватися сучасні веб-технології, зокрема Spring Boot для розробки серверної логіки, Thymeleaf для шаблонізації та генерації веб-сторінок, JPA для взаємодії з базою даних та Hibernate для ORM (Object-Relational Mapping). Також використовуються відкриті API футбольних даних для отримання актуальних даних про EPL.

ABSTRACT

This coursework is devoted to the development of a web application for following the English Premier League (EPL). The goal of this project is to create a user-friendly and functional tool that will allow users to stay up-to-date with the latest news, fixtures, results, statistics and other articles related to the EPL. During the development of the web application, modern web technologies will be used, including Spring Boot for the development of server logic, Thymeleaf for templating and generation of web pages, JPA for interaction with the database, and Hibernate for ORM (Object-Relational Mapping). Open football data APIs are also used to get up-to-date EPL data.

ВСТУП

У сучасному світі футбол є одним з найпопулярніших видів спорту, який залучає мільйони людей по всьому світу. Англійська прем'єр-ліга (EPL) є однією з найвизначніших та найпопулярніших футбольних ліг, яка збирає найсильніші команди та найталановитіших гравців.

Зацікавлення у футболі та бажання слідкувати за розвитком Англійської прем'єр-ліги призводить до необхідності доступу до актуальної інформації про матчі, розклад, результати, статистику та інші дані, які дозволяють бути в курсі подій і відчувати спільноту з іншими фанатами.

У даній курсовій роботі ми пропонуємо розробити веб-додаток для слідкування за Англійською прем'єр-лігою, який надасть користувачам зручний і змістовний доступ до всієї необхідної інформації про лігу. Веб-додаток буде включати розклад матчів, результати, статистику, новини, прогнози та інші функції, що дозволяють користувачам бути на зв'язку зі своїми улюбленими командами та гравцями.

Мета даної курсової роботи полягає у розробці веб-додатку, який буде забезпечувати зручний та повний функціонал для слідкування за Англійською прем'єр-лігою. Процес розробки буде включати використання сучасних веб-технологій, таких як Spring Boot, HTML, CSS, а також використання відкритих API для отримання актуальних даних про матчі та інші статистичні дані.

Результатом цієї роботи буде функціональний веб-додаток, який дозволить користувачам з легкістю отримувати актуальну інформацію про Англійську прем'єр-лігу, відслідковувати матчі своїх улюблених команд, а також отримувати новини, статистику та прогнози. Розроблений веб-додаток сприятиме покращенню взаємодії фанатів, сприяючи їхньому більш глибокому та насиченому досвіду під час слідкування за Англійською прем'єр-лігою.

# Огляд систем-аналогів та технологій їх розробки

## Особливості використання веб-технологій для слідкування за Англійською прем'єр-лігою (EPL)

Використання веб-технологій для слідкування за Англійською прем'єр-лігою (EPL) може мати деякі особливості, які забезпечують зручність та доступність інформації для користувачів. Ось особливості використання веб-технологій для цієї цілі:

* Живі оновлення: Веб-технології дозволяють отримувати живі оновлення стосовно матчів, результатів, розкладу та іншої інформації про Англійську прем'єр-лігу. Це може включати сповіщення про голи, карти, підсумкові дані тощо.
* Розклад та календар: Веб-технології дозволяють створювати і відображати розклад матчів Англійської прем'єр-ліги, а також календарні події, що пов'язані з матчами. Користувачі можуть легко переглядати дати та часи матчів, розташування стадіонів та інші деталі.
* Результати та статистика: Веб-технології дозволяють публікувати результати матчів та надавати детальну статистику про гру. Це може включати інформацію про голи, асисти, карти, відсоток володіння м'ячем, кількість ударів та багато іншого. Користувачі можуть переглядати цю інформацію для кожного матчу окремо або загальну статистику команд чи гравців.
* Новини та коментарі: Веб-технології дозволяють публікувати новини, огляди матчів, коментарі від експертів та фанатів. Користувачі можуть отримувати свіжі новини про Англійську прем'єр-лігу, оновлення стосовно трансферів гравців, звіти про тренування команд і багато іншого.
* Прогнози та аналітика: Веб-технології дозволяють публікувати прогнози матчів та аналітичні огляди. Користувачі можуть переглядати прогнози експертів та аналізувати статистику команд для прийняття власних прогнозів.
* Онлайн-трансляції: Веб-технології дозволяють проводити онлайн-трансляції матчів Англійської прем'єр-ліги. Користувачі можуть переглядати матчі в реальному часі через веб-сайти або застосунки, отримуючи доступ до відео трансляцій, коментарів та інших додаткових функцій.
* Соціальна взаємодія: Веб-технології дозволяють користувачам обмінюватися думками, коментарями, ставити лайки, ділитися інформацією про матчі та команди через соціальні медіа. Це створює спільноту фанатів, яка може обговорювати та спілкуватися на веб-платформах.

Ці особливості використання веб-технологій допомагають користувачам слідкувати за Англійською прем'єр-лігою, отримувати актуальну інформацію, статистику, новини та насолоджуватися матчами в режимі реального часу.

## Огляд додатків для ведення заміток

Для розробки веб-додатку для перегляду розкладу англійської прем'єр-ліги необхідно з'ясувати, які функції повинні бути доступні в ньому. Для цього потрібно дослідити та проаналізувати інші додатки, які також дозволять переглядати розклад англійської прем'єр-ліги. Це дозволить встановити особливості роботи цих додатків та з'ясувати позитивні та негативні моменти їх використання. Також цей аналіз допоможе краще зрозуміти потреби користувачів в цій області. Після проведення пошуку в Інтернеті, були знайдені найбільш популярні та відомі додатки, які будуть використовуватися як аналоги в даній роботі:

* веб-додаток Premier League [3];
* веб-додаток ESPN [4];

Далі потрібно провести аналіз цих додатків з метою встановлення особливостей їх роботи та з'ясування позитивних та негативних моментів їх використання. Це допоможе краще зрозуміти, які функції повинні бути доступні веб-додатку для створення заміток, та відповісти на потреби користувачів в цій області.

### Веб-додаток Premier League

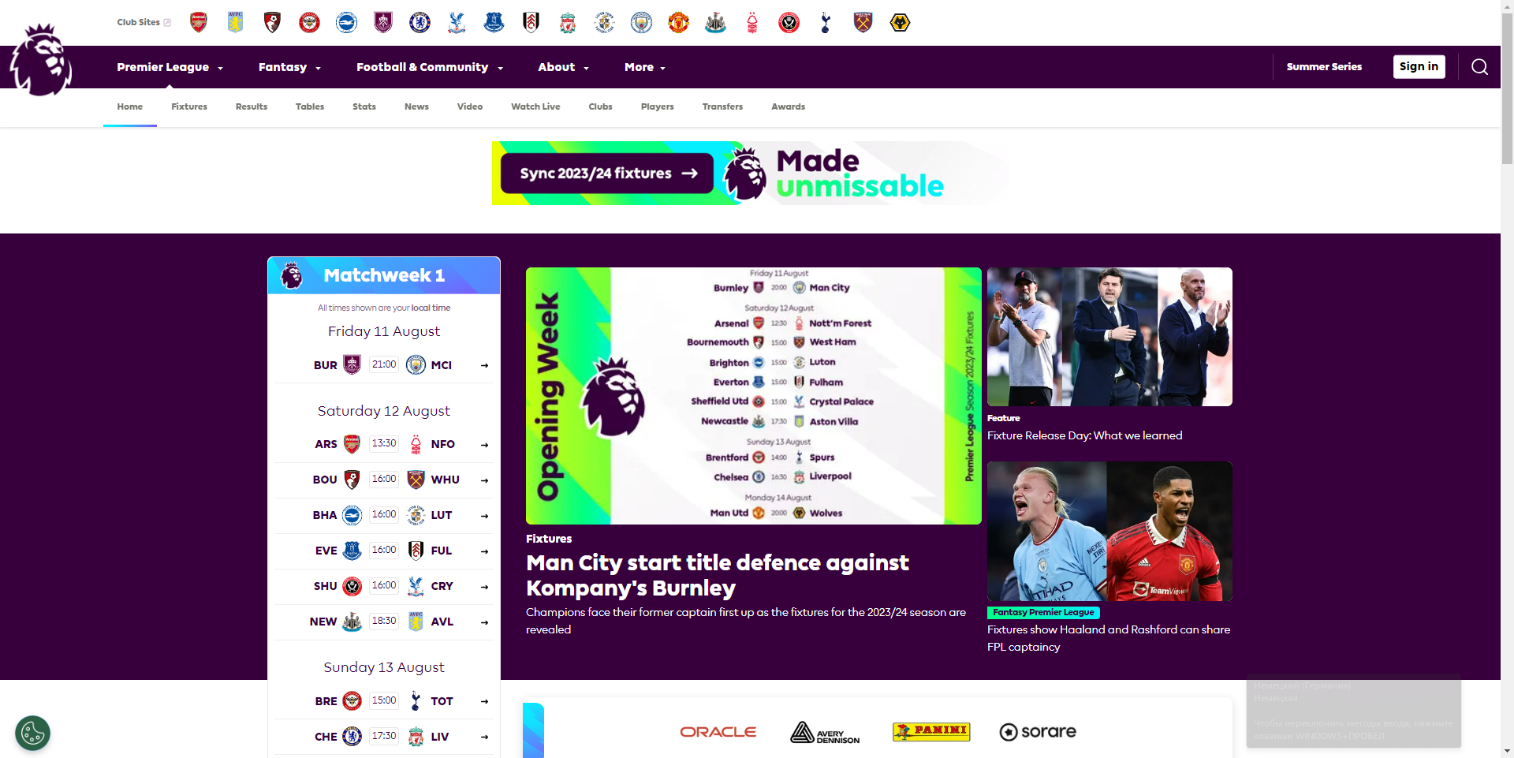
Веб-додаток Premier League надає широкий функціонал для слідкування за EPL та отримання актуальної інформації. (рис. 1.2) [3].

Рисунок 1.2 – Знімки екрану Веб-додатку Premier League

Основний функціонал додатку:

* можливість створювати списки справ;
* можливість отримати нагадування через електрону пошту, мобільний пристрій чи популярні месенджери;
* можливість ділитися своїми списками з іншими людьми;
* синхронізація справ на всіх пристроях;
* можливість додавати пріоритети, терміни закінчення, кількість повторів справи, мітки тощо;
* пошук справ та нотаток за текстом, збереження справ та нотаток в розумних списках;
* можливість бачити справи, які фізично розташовані найближче та у найбільш ефективний спосіб.
* Новини: Веб-сайт пропонує розділ новин, де ви зможете знайти останні оновлення, інтерв'ю, коментарі та інші матеріали, пов'язані з EPL.
* Розклад та результати матчів: На веб-сайті можна знайти розклад матчів EPL, включаючи дату, час та місце проведення. Після завершення матчів ви можете переглянути результати та статистику кожного поєдинку.
* Турнірна таблиця: Ви можете побачити кількість набраних очок, різницю м'ячів, перемоги, нічиї та поразки для кожної команди. Таблиця оновлюється після кожного матчу, що дозволяє вам відстежувати поточну ситуацію в лізі.
* Статистика: Веб-сайт надає великий обсяг статистичних даних про EPL. Ви можете ознайомитися зі статистикою гравців, команд, голів, асистів, жовтих та червоних карток тощо.
* Відеоматеріали: На веб-сайті можна переглянути відеоогляди матчів, голи, найкращі моменти, інтерв'ю та інші відеоматеріали, пов'язані з EPL.

Зареєстровані юзери мають додатковий функціонал:

* Фентезі футбольна Прем'єр-ліга
* Ексклюзивні фан-послуги
* Індивідуальний контент сайту
* Інформація та сповіщення про улюблений клуб

### Веб-додаток ESPN

ESPN є відомим спортивним веб-сайтом, який також надає широкий функціонал для слідкування за різними спортивними подіями, включаючи Англійську прем'єр-лігу (EPL). Розглянемо можливості додатку:

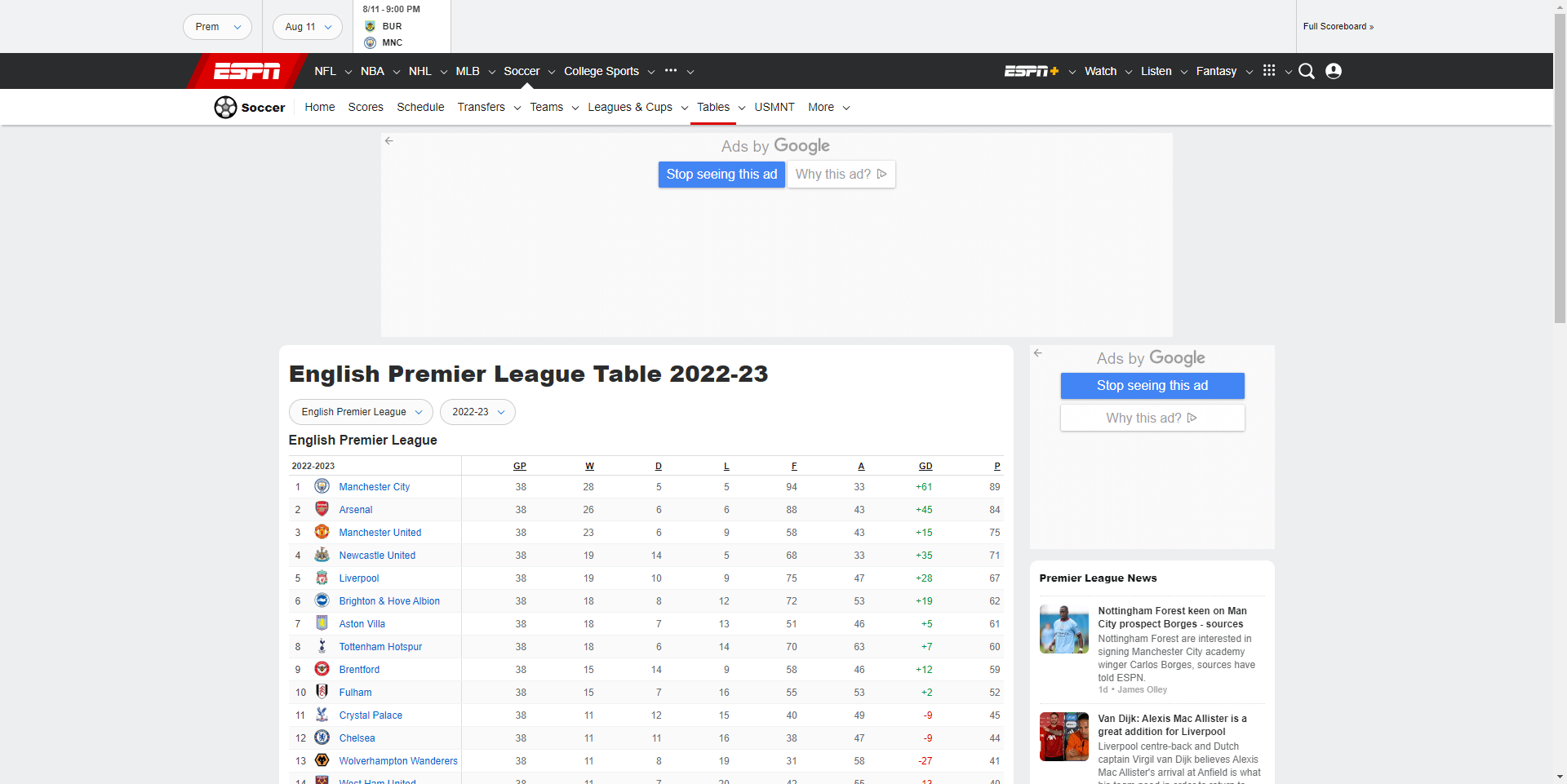
* Новини та статті: ESPN надає актуальні новини, статті, аналізи та коментарі про EPL.
* Розклад та результати матчів: Ви зможете знайти розклад матчів EPL на веб-сайті ESPN. Після завершення матчів ви можете переглянути результати, статистику, огляди та інші деталі про кожен поєдинок.
* Відеоматеріали: ESPN пропонує велику кількість відеоматеріалів, пов'язаних з EPL. Ви зможете переглянути відеоогляди матчів, голи, найкращі моменти, інтерв'ю з гравцями та тренерами, а також ексклюзивний контент.
* Статистика та дані: ESPN надає детальну статистику про гравців, команди, голи, асисти, жовті та червоні картки та інші показники, що дозволяє зробити глибокий аналіз про події в EPL.
* Аналітичні матеріали: ESPN пропонує аналітичні матеріали, де експерти висловлюють свої думки, роблять прогнози та аналізують стратегію команд та гравців в EPL.

Рисунок 1.3 – Знімки екрану веб-додатку ESPN

## Формування вимог до основних функцій додатку

Аналіз веб-додатків дозволив виділити їх переваги та недоліки та сформувати вимоги до власного веб-додатку.

Основний функціонал веб-додатку для перегляду розкладу англійської прем'єр-ліги

* Відображення розкладу: Додаток повинен забезпечувати можливість відображення актуального розкладу матчів EPL. Розклад повинен відображати дату, час, команди та іншу важливу інформацію;
* Фільтрація та сортування: Додаток повинен надавати можливість фільтрації та сортування розкладу матчів за різними параметрами, наприклад, за датою, командами;
* Оновлення розкладу: Додаток повинен мати механізм оновлення розкладу матчів автоматично, щоб користувачі завжди отримували актуальну інформацію про найближчі матчі та зміни в розкладі;
* Деталі матчу: Користувачі повинні мати можливість отримувати детальну інформацію про кожен матч, включаючи статистику матчу, прямі посилання на трансляції та іншу важливу інформацію;
* Можливість робити ставки: Користувачі повинні мати можливість зробити прогноз балами, щоб мати змогу виграти бали;
* Безпека даних: з метою забезпечення конфіденційності та безпеки, веб-додаток може пропонувати захист аккаунту за допомогою паролю. Це гарантує, що тільки авторизовані користувачі можуть отримати доступ до приватних даних.

## Огляд інформаційних технологій для розробки веб-додатку

### Фреймворк Spring Boot

Spring Boot - це корисний проект, метою якого є спрощення створення програм на основі Spring. Він дозволяє найпростішим способом створити web-додаток, вимагаючи від розробників мінімум зусиль щодо його налаштування та написання коду.

Розглянемо переваги фреймворку Spring Boot:

* Автоконфігурація: Spring Boot автоматично конфігурує додаток, виходячи з його класифікації та залежностей. Він надає розумні значення за замовчуванням для багатьох налаштувань, що полегшує старт проекту без необхідності вручну налаштовувати багато компонентів.
* Вбудований контейнер сервлетів: Spring Boot містить вбудований контейнер сервлетів (зазвичай Tomcat або Jetty), що дозволяє запускати додаток як самодостатній веб-сервер без необхідності установки окремого сервера.
* Залежності та управління версіями: Spring Boot надає систему управління залежностями Maven або Gradle, що дозволяє зручно управляти версіями бібліотек та інших компонентів.
* Велика екосистема: Spring Boot має велику активну спільноту розробників та розширень. Існує багато сторонніх бібліотек, які підтримуються та інтегруються з Spring Boot, що дозволяє розширити його функціональність та швидко розробляти різноманітні додатки.
* Вбудовані інструменти для моніторингу та управління: Spring Boot надає зручні інструменти для моніторингу, налаштування та управління додатком, такі як Actuator, що дозволяють переглядати метрики, стан додатку, налаштовувати його поведінку та інше.

Розглянемо недоліки фреймворку Spring Boot:

* Надмірна комплектація: Через автоконфігурацію та вбудовані залежності, додаток Spring Boot може мати великий обсяг і завантажувати багато бібліотек, навіть якщо вам не потрібні всі функції. Це може призвести до збільшення розміру додатку і тривалого часу завантаження.
* Складність налаштування: Хоча автоконфігурація спрощує початкове налаштування додатку, вона також може бути складною, коли потрібно змінити чи перевизначити деякі значення за замовчуванням. Неправильні налаштування можуть призвести до непередбачуваної поведінки додатку.
* Залежність від Spring-екосистеми: Spring Boot побудований на основі фреймворку Spring, тому для ефективного використання його функціоналу потрібно мати розуміння Spring-екосистеми. Це може становити виклик для розробників, які не мають досвіду з Spring або бажають використовувати інші технології.
* Менша контроль над конфігурацією: За замовчуванням, Spring Boot пропонує розумні значення конфігурації, що полегшує розробку. Проте, це також може обмежувати гнучкість, оскільки ви залежите від автоматичного вибору значень. Для складних або нестандартних випадків конфігурації може бути потрібне додаткове втручання.
* Обмежена сумісність з деякими технологіями: Хоча Spring Boot підтримує багато технологій і бібліотек, він може мати обмеження щодо сумісності з деякими менш популярними або екзотичними технологіями. В деяких випадках може знадобитися додаткова конфігурація або адаптація для використання з певними технологіями.
* Вимога до ресурсів: Spring Boot вимагає певних ресурсів. Для менш потужних систем або в обмежених середовищах це може бути проблемою.

Не дивлячись на ці недоліки, Spring Boot залишається популярним фреймворком з великою спільнотою розробників і широким функціоналом, що допомагає розробникам ефективно створювати веб-додатки на мові Java.

### Football Data API

Football Data API - це веб-сервіс, який надає доступ до розширеного набору даних про футбольні матчі, команди, гравців, змагання та багато іншого. Цей API дозволяє розробникам отримувати актуальну та детальну інформацію про футбольні події з різних ліг та змагань по всьому світу.

Основні можливості та функціонал Football Data API включають:

* Розклад матчів: API дозволяє отримувати розклад матчів для різних ліг та змагань. Інформація включає дату, час, команди, місце проведення та інші деталі.
* Результати матчів: API надає результати футбольних матчів, включаючи вказання про переможців, рахунки та статистику матчу.
* Турнірні таблиці: Ви можете отримувати інформацію про турнірні таблиці для різних ліг та змагань, включаючи позиції команд, набрані очки, різницю забитих та пропущених м'ячів тощо.
* Інформація про команди та гравців: API дозволяє отримувати детальну інформацію про футбольні команди, включаючи склад, статистику гравців, тренерів тощо.
* Статистика та аналітика: API надає доступ до різноманітної статистики та аналітики футбольних матчів, таких як кількість забитих м'ячів, паси, володіння м'ячем та інші показники.
* Обмеження запитів: Існують обмеження щодо кількості запитів, які можна виконати до API за певний часовий період.

Щоб використовувати Football Data API, вам зазвичай потрібно зареєструватись на їх веб-сайті, отримати API-ключ та користуватись його документацією для виконання запитів та отримання необхідної інформації.

### Фреймворк Spring Security

Spring Security – це фреймворк безпеки, який надає механізми аутентифікації, авторизації та захисту веб-додатків на основі Spring. Він забезпечує різні функції безпеки, що допомагають захистити додаток від атак, забезпечують контроль доступу та забезпечують конфіденційність даних.

Основні можливості та функціонал Spring Security включають:

* Аутентифікація: Spring Security надає механізми для аутентифікації користувачів. Ви можете використовувати різні способи аутентифікації, такі як форма входу, базова аутентифікація, OAuth і багато інших. Ви також можете налаштовувати власні провайдери аутентифікації.
* Авторизація: Spring Security дозволяє налаштовувати правила авторизації для контролерах, методів або ресурсів вашого додатку. Ви можете використовувати анотації для визначення рівнів доступу, ролей користувачів або власних правил авторизації.
* Захист від атак: Spring Security включає в себе заходи захисту від різних видів атак, таких як CSRF (Cross-Site Request Forgery), XSS (Cross-Site Scripting), SQL Injection та інші. Він автоматично застосовує потрібні фільтри та механізми для запобігання таким атакам.
* Інтеграція з іншими фреймворками: Spring Security може інтегруватись з іншими фреймворками Spring, такими як Spring MVC, Spring Boot і Spring Data, що спрощує розробку безпечних веб-додатків.
* Журналювання та моніторинг: Spring Security дозволяє ведення журналу безпеки для фіксації подій та виявлення підозрілих дій. Ви також можете моніторити активність користувачів та встановлювати правила для заблокування підозрілих дій.
* Розширення та налаштування: Spring Security має гнучку архітектуру, яку можна легко розширювати та налаштовувати залежно від ваших потреб. Ви можете створювати власні фільтри, провайдери безпеки та інші компоненти для реалізації специфічних сценаріїв безпеки.

Spring Security є інструментом для забезпечення безпеки веб-додатків на основі Spring. Він дозволяє розробникам ефективно впроваджувати механізми аутентифікації, авторизації та захисту, що є критичними для багатьох додатків.

### Thymeleaf

Thymeleaf є популярним шаблонним двигуном для розробки веб-додатків на основі Java і фреймворку Spring. Він використовується для створення динамічних веб-сторінок, які можуть генеруватись на стороні сервера та відображатись на клієнтському браузері.

Основні особливості та можливості Thymeleaf включають:

* Інтеграція зі Spring: Thymeleaf підтримує безпроблемну інтеграцію з фреймворком Spring, що дозволяє легко використовувати його разом з іншими компонентами Spring, такими як Spring MVC та Spring Boot.
* Читабельний синтаксис: Thymeleaf використовує природний та інтуїтивно зрозумілий синтаксис, який нагадує стандартний HTML. Це робить його легким у використанні та розумінні, навіть для тих, хто не має досвіду з шаблонними двигунами.
* Динамічні дані: Thymeleaf дозволяє вбудовувати та відображати динамічні дані в шаблонах, такі як змінні, колекції, об'єкти тощо. Ви можете використовувати вирази Thymeleaf для обробки даних та умовного відображення елементів в залежності від значень.
* Міжнародна підтримка: Thymeleaf надає зручні інструменти для локалізації та інтернаціоналізації веб-додатків. Ви можете легко використовувати мовні ключі, форматування дат, чисел та інші інструменти для підтримки міжнародних потреб.
* Широкий набір функцій: Thymeleaf має багато вбудованих функцій і можливостей, таких як ітерація, умови, шаблонні фрагменти, включення і заміна шаблонів. Це дозволяє розробникам створювати складні та повторно використовувані шаблони.
* Підтримка статичних ресурсів: Thymeleaf дозволяє легко включати статичні ресурси, такі як CSS, JavaScript, зображення тощо, в шаблони сторінок.

Thymeleaf є потужним і гнучким інструментом для створення динамічних веб-сторінок у веб-додатках на основі Spring. Він дозволяє ефективно розробляти шаблони з динамічними даними та легко інтегруватися з іншими компонентами Spring.

### JPA

JPA (Java Persistence API) - це стандартний інтерфейс для роботи з реляційною базою даних в Java-програмах. Він входить до Java EE (Enterprise Edition) і базується на концепції ORM (Object-Relational Mapping), що дозволяє розробникам працювати з об'єктами в коді, не залежно від конкретної бази даних.

Основні особливості та можливості JPA включають:

* Об'єктно-орієнтоване відображення даних: JPA дозволяє відображати об'єкти Java на таблиці бази даних і виконувати операції CRUD (створення, читання, оновлення, видалення) над цими об'єктами. Це дозволяє розробникам працювати з даними, використовуючи об'єктно-орієнтований підхід замість SQL-запитів.
* Управління транзакціями: JPA надає можливості для управління транзакціями в базі даних. Ви можете оголосити транзакційні методи, які гарантують атомарність, консистентність, ізольованість та стійкість (ACID) даних.
* Запити та фільтри: JPA надає механізми для виконання складних запитів до бази даних з використанням мови запитів, відомої як JPQL (Java Persistence Query Language). Ви можете виконувати вибірки, об'єднання, сортування та інші операції над даними.
* Кешування: JPA підтримує механізми кешування для покращення продуктивності запитів до бази даних. Ви можете налаштовувати кешування на рівні об'єктів або запитів для зменшення навантаження на базу даних.
* Наслідування та асоціації: JPA дозволяє моделювати наслідування між класами та встановлювати асоціації (зв'язки) між об'єктами. Ви можете використовувати анотації для визначення взаємозв'язків та мапування до таблиць бази даних.
* Підтримка транзакційного контексту: JPA забезпечує транзакційний контекст для керування життєвим циклом об'єктів в базі даних. Він автоматично відстежує зміни, зберігає їх у базі даних та забезпечує консистентність даних під час збереження чи видалення об'єктів.

JPA спрощує взаємодію з базою даних у Java-програмах і дозволяє розробникам працювати на вищому рівні абстракції з використанням об'єктно-орієнтованого підходу. Він підтримує багато функцій, які полегшують розробку, підтримку транзакцій та оптимізацію роботи з базою даних.

### Lombok

Lombok є бібліотекою для Java, яка дозволяє зменшити кількість повторюваного та нудного коду, який часто потрібно писати в Java-класах. Вона працює шляхом автоматичної генерації методів, таких як геттери, сеттери, конструктори та інші, на основі анотацій, які додаються до класу або його полів.

Основні особливості та можливості Lombok включають:

* Анотації для автоматичної генерації коду: Lombok надає набір анотацій, таких як @Getter, @Setter, @NoArgsConstructor, @AllArgsConstructor, @ToString та інші, які можна застосовувати до класів або їх полів. Після компіляції ці анотації автоматично генерують відповідний код.
* Упрощення створення об'єктів: Lombok дозволяє легко створювати об'єкти з використанням анотації @Builder. Вона генерує паттерн "Builder", який забезпечує зручну ініціалізацію об'єктів з великою кількістю полів.
* Зменшення повторення коду: Завдяки Lombok можна зменшити кількість повторюваного коду, такого як геттери і сеттери, що полегшує розробку та підтримку коду.
* Підтримка інтеграції з іншими інструментами: Lombok підтримує інтеграцію з популярними інструментами розробки, такими як IDE (Integrated Development Environment), засоби аналізу коду та інші.
* Розширення можливостей: Lombok також надає можливість створення власних анотацій та кастомізацію генерації коду за допомогою аннотаційного процесора.

Lombok допомагає зробити код Java більш читабельним, скорочує кількість написаного коду та полегшує розробку. Однак, варто враховувати, що використання Lombok може зробити код менш явним, особливо для інших розробників, які не знайомі з цією бібліотекою.

### Hibernate

Hibernate є одним з найпопулярніших фреймворків для роботи з базами даних в Java. Він є реалізацією специфікації JPA (Java Persistence API) і надає розробникам зручні інструменти для роботи з об'єктно-реляційним відображенням (ORM).

Основні особливості та можливості Hibernate включають:

* Об'єктно-реляційне відображення: Hibernate дозволяє відображати об'єкти Java на таблиці бази даних і забезпечує мапування між ними. Він автоматично створює SQL-запити для створення, читання, оновлення та видалення даних з бази даних, що дозволяє розробникам працювати з об'єктами, не залежно від конкретних деталей роботи з базою даних.
* Управління транзакціями: Hibernate надає механізми для управління транзакціями в базі даних. Ви можете оголосити транзакційні методи або використовувати анотації для позначення границь транзакцій. Це забезпечує консистентність даних і забезпечує атомарність операцій.
* Запити та критерії: Hibernate надає потужні механізми для створення SQL-запитів до бази даних. Ви можете використовувати мову HQL (Hibernate Query Language) або критерії для створення складних запитів. Це дозволяє легко вибирати, сортувати, фільтрувати та агрегувати дані з бази даних.
* Кешування: Hibernate підтримує механізми кешування, які дозволяють зберігати певні дані в оперативній пам'яті для покращення продуктивності запитів. Це може значно зменшити навантаження на базу даних та покращити швидкодію додатка.
* Наслідування та асоціації: Hibernate підтримує різні види відносин між об'єктами, включаючи один-до-одного, один-до-багатьох та багато-до-багатьох. Він також підтримує наслідування між класами та дозволяє використовувати поліморфізм при роботі з об'єктами.
* Підтримка різних баз даних: Hibernate працює з різними базами даних, включаючи MySQL, PostgreSQL, Oracle, Microsoft SQL Server та багато інших. Він надає можливість налаштовувати підключення до бази даних та оптимізувати роботу з конкретною СУБД.

Hibernate є потужним інструментом для роботи з базами даних у Java-додатках. Він дозволяє розробникам працювати з об'єктами, абстрагуючи від деталей роботи з базою даних і полегшуючи розробку, тестування та підтримку коду.

### Java

Java - це потужна об'єктно-орієнтована мова програмування, яка використовується для розробки різноманітних додатків, включаючи веб-додатки. Вона була створена з метою бути простою у використанні, переносною та масштабованою.

Основні переваги використання Java для веб-розробки включають:

* Платформонезалежність: Код, написаний на Java, може працювати на різних операційних системах, що дає можливість створювати веб-додатки, які можуть бути запущені на різних пристроях.
* Об'єктно-орієнтована природа: Java підтримує об'єктно-орієнтований підхід, що дозволяє створювати модульний і легкозмінний код, полегшує розробку та підтримку веб-додатків.
* Багатий набір бібліотек: Велика кількість багатофункціональних бібліотек та фреймворків доступна для Java, що спрощує розробку веб-додатків та надає різноманітні можливості, такі як робота з базами даних, мережевими операціями, шифруванням тощо.
* Висока продуктивність: Java відома своєю високою продуктивністю завдяки оптимізованому виконанню коду та вбудованій підтримці багатопотоковості.
* Безпека: Java має вбудовані механізми безпеки, які допомагають уникнути вразливостей та забезпечити надійність веб-додатків.
* Широке співробітництво: Java має велике співтовариство розробників та підтримується великими компаніями, що забезпечує наявність багато ресурсів, документації та підтримки для веб-розробників.

Щоб використати Java для розробки веб-додатків, можна скористатися різними фреймворками та інструментами, такими як Spring, JavaServer Faces (JSF), Play Framework тощо. Вони допомагають спростити розробку та підтримку веб-додатків, надаючи шаблони, бібліотеки та інші корисні функції.

### Bootstrap 5

Bootstrap 5 є одним з найпопулярніших фреймворків для розробки веб-додатків та веб-сайтів. Він надає набір інструментів, шаблонів і компонентів, що дозволяють швидко та ефективно створювати стильний та респонсивний дизайн.

Основні особливості та переваги використання Bootstrap 5:

* Респонсивний дизайн: Bootstrap 5 дозволяє легко створювати респонсивний дизайн, що адаптується до різних розмірів екранів та пристроїв. Це дозволяє забезпечити коректне відображення вашого веб-додатку на різних пристроях, включаючи комп'ютери, планшети та смартфони.
* Готові компоненти: Bootstrap 5 постачається з великою кількістю готових компонентів, таких як кнопки, форми, навігація, каруселі, модальні вікна та багато інших. Це дозволяє швидко додавати функціональність до вашого веб-додатку без необхідності писати код з нуля.
* Гнучкість та налаштування: Bootstrap 5 надає багато можливостей для налаштування та стилізації компонентів. Ви можете змінювати кольори, шрифти, розміри та інші властивості, щоб відповідати вимогам вашого дизайну.
* Підтримка сучасних браузерів: Bootstrap 5 підтримує останні версії популярних браузерів, забезпечуючи сумісність та коректне відображення вашого веб-додатку для широкого кола користувачів.
* Легкість використання: Завдяки чистому та зрозумілому коду, Bootstrap 5 є дуже легким у використанні. Ви можете швидко оволодіти його основами і почати створювати стильний дизайн для вашого веб-додатку.

Використання Bootstrap 5 допоможе вам ефективно створити сучасний та привабливий веб-дизайн для вашого додатку, зекономивши час та зусилля при розробці.

### Html

HTML (HyperText Markup Language) - це стандартна мова розмітки для створення веб-сторінок. Вона використовується для структурування та візуалізації контенту на веб-сторінках. HTML використовується разом з CSS (Cascading Style Sheets) та JavaScript для створення інтерактивних та естетично привабливих веб-додатків.

### MySQL

MySQL є однією з найпопулярніших відкритих реляційних систем керування базами даних (СКБД). Він використовує мову SQL (Structured Query Language) для зберігання та управління структурованою інформацією, такою як дані, таблиці, зв'язки та індекси.

Особливості та переваги використання MySQL:

* Надійність: MySQL відомий своєю стабільністю та надійністю. Він може обробляти великий обсяг даних та забезпечувати швидкий доступ до них, що робить його відмінним вибором для веб-додатків з великим навантаженням.
* Простота використання: MySQL має простий та зрозумілий синтаксис мови SQL, що полегшує розробку та виконання запитів до бази даних. Він також надає графічний інтерфейс для адміністрування та управління базою даних.
* Підтримка транзакцій: MySQL підтримує механізми транзакцій, що дозволяють виконувати групу запитів як одну атомарну операцію. Це забезпечує цілісність даних та уникнення втрати інформації в разі помилки або збою системи.
* Масштабованість: MySQL може працювати з великими обсягами даних та підтримує розподілені системи баз даних, що дозволяє горизонтально масштабувати інфраструктуру для забезпечення високої продуктивності та доступності.
* Сумісність: MySQL підтримує стандарти SQL, що робить його сумісним з багатьма іншими СКБД та інструментами. Ви можете легко імпортувати та експортувати дані між різними базами даних або використовувати MySQL разом з іншими технологіями та мовами програмування.

MySQL є потужним та надійним інструментом для зберігання та управління даними веб-додатків. Він часто використовується разом з іншими технологіями, такими як Java, PHP, Python та інші, для розробки різноманітних веб-проектів.

## Висновки до першого розділу

В першому розділі курсової роботи був проведений огляд веб-додатків для перегляду розкладу англійської прем'єр-ліги та технологій їх розробки.

Було розглянуто питання впливу веб-технологій на задачу перегляду розкладу англійської прем'єр-ліги. Було визначено, що веб-технологій відіграють значну роль у поліпшенні досвіду та заохочує пильніше слідкувати за EPL. Вони надають користувачам широкі можливості доступу до інформації.

Далі був проведений огляд додатків-аналогів, який дозволив визначити основні переваги та недоліки існуючих веб-додатків для ведення заміток, встановити їх основні функції та можливості. На базі проведеного огляду були визначені основні вимоги до власного веб-ресурсу, а також наданий детальний опис цих вимог.

Був проведений огляд інформаційних технологій для розробки веб-додатку для перегляду розкладу англійської прем'єр-ліги. У якості мови програмування було обрано Java. У якості фреймворку для розробки веб-додатку було вирішено обрати фреймворк Spring Boot. Залучення цього фреймворку дозволяє зосередитись на бізнес-логіці додатку, а не на конфігурації та інфраструктурних аспектах, що суттєво полегшує процес його створення. Для робити з базою даних використовується Hibernate та JPA. У якості бази даних обрана MySQL. Для полегшення роботи з кодом використовується Lombok. Для створення динамічних веб-сторінок використовується Thymeleaf разом з HTML та Bootstrap. Для захисту додатку від атак використовується Spring Security

# Проєктування додатку для ведення заміток

## Мета та задачі додатку

Мета веб-сайту Epl Prediction полягає в наданні користувачам зручного та організованого інструменту для перегляду розкладу матчів Англійської Прем'єр-ліги та здійснення прогнозів на результати матчів. Основною метою веб-сайту є полегшення процесу прогнозування перемог, поразок або нічиїх, дозволяючи користувачам зберігати свої прогнози та отримувати актуальну інформацію про матчі та результати.

Завдання веб-сайту Epl Prediction можуть включати:

- Відображення розкладу матчів: веб-сайт повинен надавати користувачам актуальну інформацію про дати, час та команди, які змагаються у матчах Англійської Прем'єр-ліги.

- Прогнозування результатів: користувачам потрібно мати можливість вибирати один з трьох варіантів (перемога, поразка, нічия) для кожного матчу та зберігати свої прогнози.

- Розподіл балів: після завершення матчу, веб-сайт повинен автоматично розподілити бали між користувачами, пропорційно до точності їх прогнозів. Програні бали будуть розподілені між користувачами, які зробили правильний прогноз, згідно з встановленою системою очкування.

- Актуальна таблиця: веб-сайт може надавати користувачам актуальну таблицю Англійської Прем'єр-ліги, щоб вони могли відстежувати місце команд у чемпіонаті та порівнювати їх результати.

Основна мета веб-сайту Epl Prediction полягає в полегшенні процесу прогнозування результатів матчів Англійської Прем'єр-ліги та наданні користувачам зручного середовища для зберігання та організації їх прогнозів. Веб-сайт допомагає футбольним уболівальникам отримувати задоволення від прогнозування та відстеження результатів матчів.

## Визначення функціональних вимог до додатку

Визначення функціональних вимог є важливим етапом у процесі створення додатку для ведення заміток. Функціональні вимоги визначають, які конкретні функції та можливості повинен мати додаток. Вони встановлюють чітку спрямованість розробки, допомагають уникнути неоднозначностей та непорозумінь.

Крім того, визначення функціональних вимог дозволяє зосередитися на потребах та вимогах користувачів. Вони допомагають врахувати, які функції та можливості будуть найбільш корисними для користувачів додатку.

Також, функціональні вимоги слугують основою для комунікації між розробниками, дизайнерами та іншими учасниками проекту. Вони допомагають зрозуміти, що саме потрібно реалізувати та які очікувані результати.

Встановлення функціональних вимог дозволяє визначити обсяг роботи та потребу в ресурсах для реалізації додатку. Це допомагає планувати час, бюджет та ресурси проекту ефективно.

Єдиним актором додатку є актор «користувач»:

* користувач – користувач додатку. Має доступ до всіх функцій додатку та несе всю відповідальність за його роботу.

З метою визначення користувацьких історій та нефункціональних вимог до веб-ресурсу, було розроблено діаграму сценаріїв використання додатку (рис. 2.1). Діаграма сценаріїв UML (Unified Modeling Language) - це графічний інструмент для опису функціональної взаємодії між користувачами та системою. Вона складається з акторів, сценаріїв та взаємодії між ними.

Ця діаграма містить основних акторів системи та описує сценарії їх взаємодії з системою. Вона допомагає проаналізувати залежності між акторами та можливими варіантами використання системи. Також ця діаграма швидко демонструє основні функції системи для розробників.

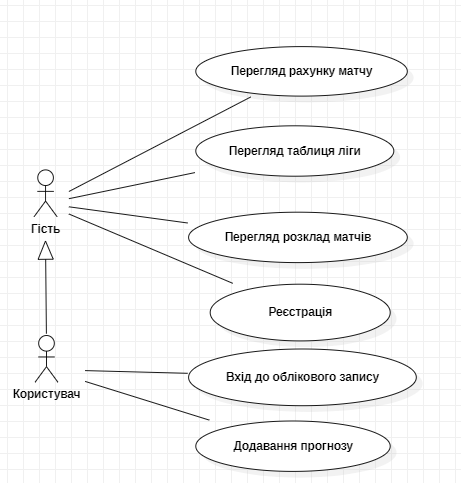


Рисунок 2.1 – Діаграма варіантів використання

## Формування користувацьких історій додатку

Проектування діаграми прецедентів дозволяє визначити такі користувацькі історії до додатку для ведення заміток.

Epl Prediction - веб-сайт, який надає користувачам можливість прогнозування результатів матчів Англійської Прем'єр-ліги та отримання актуальної інформації про матчі та команди. Веб-сайт має наступні функціональності:

US1 Як користувач, я хочу переглядати розклад матчів, щоб бути в курсі найближчих поєдинків.

Сценарій користувацької історії:

- Користувач відкриває веб-сайт Epl Prediction.

- Він переглядає розклад матчів Англійської Прем'єр-ліги, де вказані дати, час та команди, які змагатимуться.

US2 Як користувач, я хочу робити прогнози на матчі, щоб визначити переможців.

Сценарій користувацької історії:

- Користувач переглядає список найближчих матчів.

- Він обирає один з матчів і вносить свій прогноз на результат (перемога, поразка, нічия).

- Після збереження прогнозу, користувач може переглянути свої прогнози та їх статус (правильний/неправильний) в своєму профілі.

US3 Як користувач, я хочу бачити актуальні результати матчів, щоб відстежувати успішність своїх прогнозів.

Сценарій користувацької історії:

- Після завершення матчу, користувач переглядає веб-сайт Epl Prediction.

- Він переглядає актуальні результати матчів, де вказано команди, рахунок та інші важливі деталі.

- Користувач може перевірити свої прогнози та порівняти їх з фактичними результатами.

US4 Як користувач, я хочу мати особистий профіль, де зберігатимуться мої прогнози та статистика.

Сценарій користувацької історії:

- Користувач створює особистий профіль на веб-сайті Epl Prediction.

- Він вводить свої особисті дані, такі як ім'я, електронну пошту та пароль.

- Після створення профілю, користувач може переглядати свою кількість балів.

Цей веб-сайт Epl Prediction надає футбольним уболівальникам зручну платформу для прогнозування результатів матчів Англійської Прем'єр-ліги та відстеження їх прогнозів і актуальних результатів.

## Визначення нефункціональних вимог до додатку

Після визначення функціональних вимог, необхідно сформувати нефункціональні вимоги, які висуваються до додатку для ведення заміток.

NFR1 Підтримка версій операційних систем - веб-сайт Epl Prediction повинен працювати на різних версіях веб-браузерів, таких як Google Chrome, Mozilla Firefox, Safari тощо. Він має підтримувати сучасні версії браузерів та забезпечувати сумісність з популярними операційними системами, такими як Windows, macOS, Linux тощо.

NFR2 Відповідна архітектура процесора - веб-сайт Epl Prediction має бути оптимізований для різних архітектур процесорів, таких як ARM або x86. Це дозволить користувачам з різними типами пристроїв, включаючи комп'ютери, ноутбуки, планшети та телефони, використовувати веб-сайт без проблем.

NFR3 Обсяг оперативної пам'яті (RAM) - веб-сайт Epl Prediction повинен працювати ефективно з різними розмірами оперативної пам'яті, починаючи від 1 ГБ і більше. Це дозволить користувачам з різними пристроями, які мають різну потужність, використовувати веб-сайт без значного впливу на продуктивність їх пристроїв.

NFR4 Обсяг внутрішньої пам'яті - веб-сайт Epl Prediction має вимагати обмежений обсяг внутрішньої пам'яті для збереження даних про користувачів, матчі та прогнози. Це допоможе забезпечити ефективне використання простору зберігання та швидкий доступ до інформації.

NFR5 Роздільна здатність екрану - веб-сайт Epl Prediction має бути добре адаптований для різних роздільних здатностей екранів пристроїв. Він повинен забезпечувати належний візуальний досвід користувачів незалежно від розміру екрану, включаючи великі монітори, ноутбуки, планшети та мобільні телефони.

NFR6 Інтернет-підключення - веб-сайт Epl Prediction може вимагати активного Інтернет-підключення для отримання актуальних даних про матчі, оновлень результатів, новин та статистики. Користувачі повинні мати доступ до надійного Інтернет-з'єднання для повноцінного використання функціоналу веб-сайту.

## Ідентифікація архетипу веб-додатку

Інформаційна система відноситься до архетипу Web Application (WA) - класичний додаток, який запускається у веб-браузері на різних пристроях, включаючи комп'ютери, ноутбуки, планшети та мобільні телефони.

## Проектування користувацького інтерфейсу веб-додатку

На основі вимог до веб-ресурсу, які включають функціональні та нефункціональні вимоги, можна приступити до проектування користувацького інтерфейсу веб-додатку. Це вимагає розробки макетів ключових вікон веб-сайту, визначення сценаріїв взаємодії користувачів з цими вікнами, складання стислого опису кожного макету та визначення їх ролі в досягненні визначеної мети в цій кваліфікаційній роботі.

На рисунку 2.4 зображений макет головного вікна веб-ресурсу для продуктової крамниці. Головне вікно веб-ресурсу продуктової крамниці містить різні елементи. Елемент шапки включає посилання на розклад матчів, таблицю ліги, профіль користувача та реєстрацію.

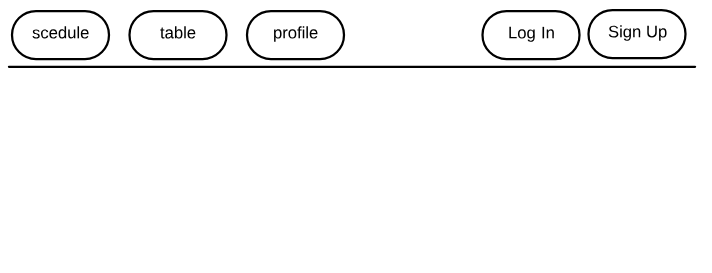


Рисунок 2.4 – Макет сторінки головного вікна веб-ресурсу

На рисунку 2.5 зображений макет сторінки з розкладом матчів, він має таку ж саму шапку, що на головній сторінці та інформацію про матчі у вигляді таблиці.

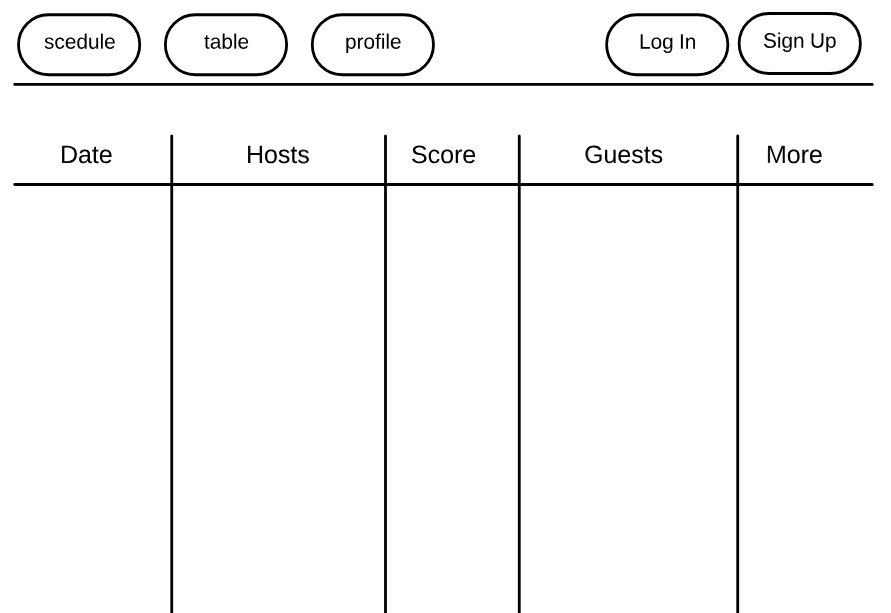


Рисунок 2.5 – Макет сторінки з розкладом матчів

На рисунку 2.6 зображений макет вікна окремого матчу. Макет містить інформацію про матч. А також у макеті є можливість зробити прогноз.

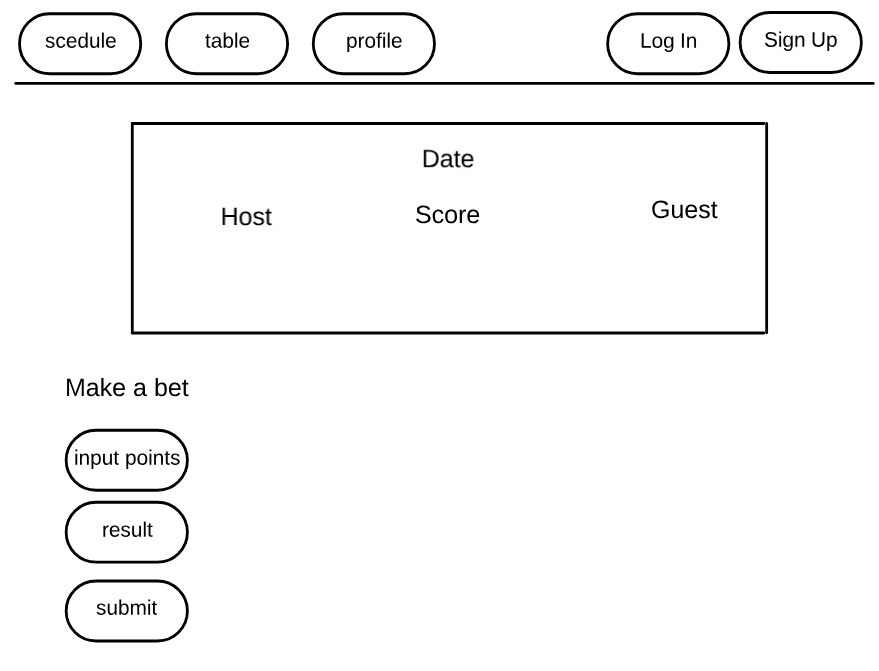


Рисунок 2.6 – Макет вікна матчу

## Висновки до другого розділу

У другому розділі цієї курсової роботи було проведено проектування веб-ресурсу продуктової крамниці на базі фреймворку Spring Boot.

У процесі проектування була визначена мета сайту. На основі цих даних були встановлені основні вимоги до веб-ресурсу, включаючи функціональні та нефункціональні. Також була розроблена діаграма сценаріїв використання веб-ресурсу. Крім того, була розроблена діаграма інформаційних потоків та визначені вхідні та вихідні інформаційні потоки системи.

З метою проектування інтерфейсу користувача були розроблені макети ключових вікон веб-ресурсу, а також діаграма станів системи. Крім того, була розроблена діаграма розгортання веб-ресурсу. Для побудови веб-ресурсу була визначена схема даних системи, а також ключові технології, які були використані в процесі проєктування та реалізації веб-ресурсу.

# ПРОГРАМНА РЕАЛІЗАЦІЯ додатку для ведення заміток

## Структура серверного програмного проєкту

Пакет програмного проекту Spring для веб-ресурсу продуктової крамниці може містити різні компоненти. Визначено ці компоненти та визначимо, які класи реалізують ці компоненти.

Першим компонентом є контролери. Вони відповідають за обробку запитів, які приходять від користувачів та повернення відповіді у формі HTML сторінки. В рамках серверного проєкту були створені наступні контролери:

* контролер для перегляду клубів (ClubController*.java*) – відповідає за відображення сторінки для перегляду рейтингу команд та сторінок клубів;
* контролер для матчів (*MatchController.java*) – відповідає за відображення матчів, розкладу матчів;
* контролер для обробки ставок (*PredictionController.java*) – відповідає за обробку ставок;
* контролер для аутентифікації та авторизації користувачів (UserServicer.java) – відповідає за відображення сторінок входу, реєстрації та інші функції, що пов'язані з аутентифікацією та авторизацією користувачів.

Компонент сервісів відповідає за логіку бізнесу та операції, які пов'язані з даними. Сервіси можуть включати методи, які повертають дані з бази даних, змінюють їх та зберігають зміни. В рамках серверного проєкту були створені наступні сервіси:

* *ClubService.java* – сервіс, що забезпечує роботу з клубами, такі як створення, оновлення, видалення та отримання даних про клуби;
* *CustomUserDetailsService.java* – сервіс, що забезпечує автентифікацію та авторизацію користувачів на веб-ресурсі;
* *MatchService.java* – сервіс, що забезпечує роботу з матчами, такі як додавання, оновлення, видалення та отримання даних про матчі;
* *PredictionService.java* – сервіс, що забезпечує роботу зі ставкою, такі як додавання, винагороду та отримання даних про ставку;
* *UserService.java* – сервіс, що забезпечує обробку юзерів, тобто створення, видалення, оновлення та отримання даних юзерів;

Також, в проєкті містяться такі компоненти:

Репозиторії– відповідають за збереження та отримання даних з бази даних. Репозиторії можуть містити методи, які зберігають, оновлюють та видаляють дані з бази даних;

* ApiRepository – репозиторій, що забезпечує обробку апі, тобто отримання даних про клуби, матчів.
* ClubRepository – репозиторій, що забезпечує обробку даних з бази даних про клуби.
* MatchRepository – репозиторій, що забезпечує обробку даних з бази даних про матчі.
* PredictionRepository – репозиторій, що забезпечує обробку даних з бази даних про ставки.
* UserRepository – репозиторій, що забезпечує обробку даних з бази даних про користувачів.
* FootballDataApiRepositoryImpl – репозиторій, що забезпечує обробку апі, тобто отримання даних про клуби, матчів.

Моделі – представляють сутності, які використовуються в системі. Моделі можуть містити поля, які відображають дані з бази даних, та методи, які допомагають взаємодіяти з цими даними;

* Club – модель, що представляє сутність клубу
* Match – модель, що представляє сутність матчу
* Prediction – модель, що представляє сутність ставки
* User – модель, що представляє сутність користувача

Конфігураційні файли – відповідають за конфігурування та налаштування Spring Framework. Конфігураційні файли можуть містити налаштування зв'язку з базою даних, налаштування безпеки та інші параметри, які необхідні для правильної роботи системи.

* CustomUserDetailsService – налаштовує Spring Security, а саме реєстрацію, вхід в систему і т.д.

## Діаграма класів веб-ресурсу

Діаграма класів є одним з видів UML-діаграм і представляє зовнішній вигляд класів, інтерфейсів та їх зв'язків у системі або додатку. Діаграма класів допомагає візуалізувати структуру системи та відношення між її компонентами.

Діаграма класів веб-ресурсу продуктової крамниці на базі фреймворку Angular наведена на рисунку 3.1.

## Керування вихідним кодом веб додатку

Використання системи контролю версій є дуже важливим для будь-якого програмного проекту, в тому числі і для веб-ресурсу продуктової крамниці. Використання системи контролю версій дозволяє зберігати історію всіх змін, які були внесені до веб-ресурсу.

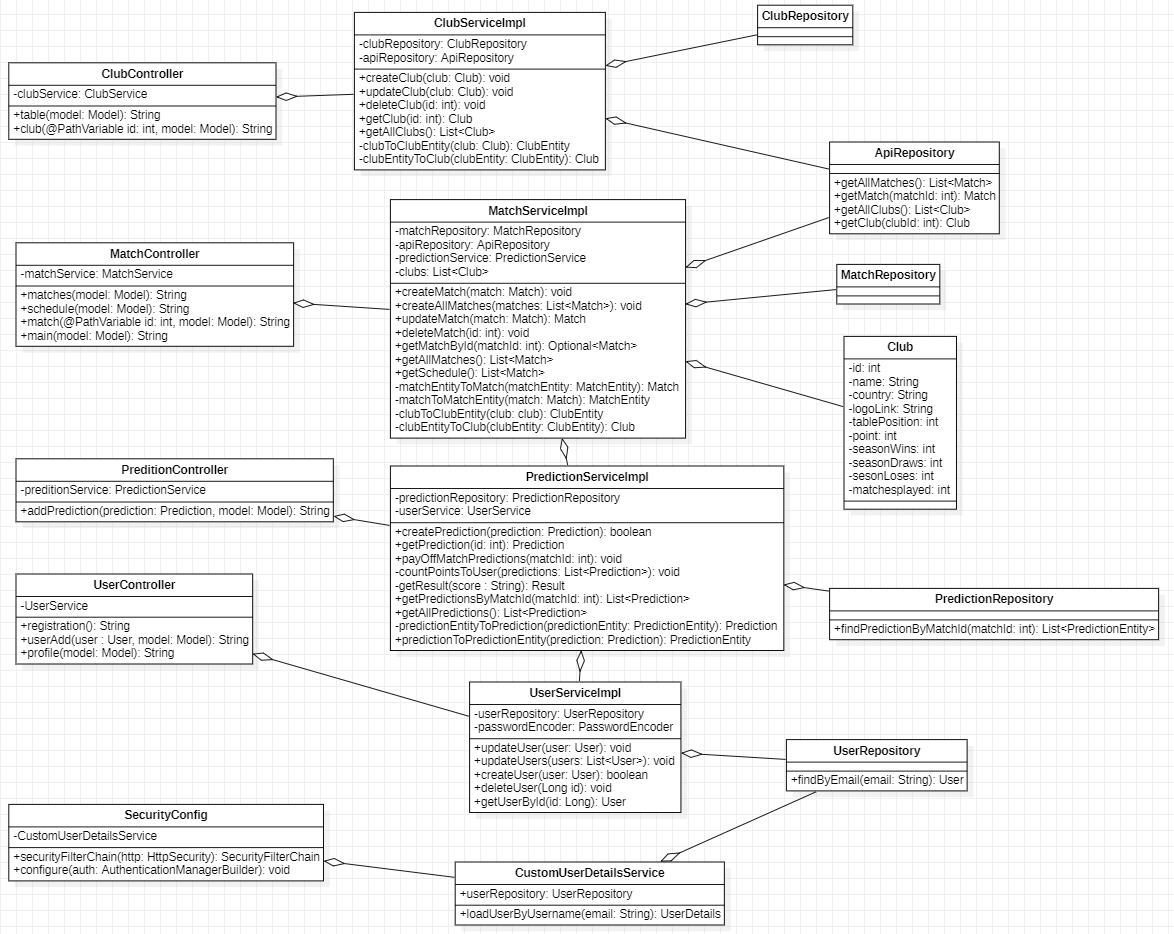
Це дозволяє в разі потреби повернутися до попередніх версій веб-ресурсу та відновити попередній стан, який був на момент попередньої версії.

Також, система контролю версій дозволяє контролювати версії веб-ресурсу, що значно полегшує роботу з веб-ресурсом та дозволяє підтримувати його в актуальному стані.

Основні метрики репозиторію веб-ресурсу наведені в таблиці 3.1.

Таблиця 3.1 – Метрики керування програмним кодом додатку

|  |  |  |  |
| --- | --- | --- | --- |
| Веб-ресурс | Кількість комітів | Кількість pull-реквестів | Кількість гілок у репозиторії |
| Клієнтська частина | 10 | 0 | 1 |
| Серверна частина | 13 | 0 | 1 |

Рисунок 3.1 – Діаграма класів веб-ресурсу

## Функціональне тестування розробленого веб-ресурсу

Функціональне тестування веб-ресурсу продуктової крамниці є дуже важливим процесом, оскільки воно дозволяє перевірити правильність функціонування різних функцій та можливостей цього ресурсу перед його введенням в експлуатацію.

Функціональне тестування дозволяє перевірити, що всі функції та можливості веб-ресурсу продуктової крамниці працюють правильно, як очікувалося, що забезпечує коректну роботу веб-сайту та надійність для його використання.

Також функціональне тестування дозволяє виявити та виправити помилки веб-ресурсу, що можуть негативно впливати на досвід користувачів, підвищуючи їх задоволеність та вірогідність повернення на сайт в майбутньому.

Для проведення функціонального тестування необхідно розробити протокол тестування.

Протокол тестування – це документ, який містить опис кроків, процедур та результатів тестування програмного продукту, системи чи окремої їх частини. Протокол тестування має на меті систематично задокументувати процес тестування та отримані результати, щоб забезпечити максимальну об'єктивність та повторюваність результатів.

Для функціонального тестування веб-ресурсу продуктової крамниці був розроблений наступний протокол тестування.

TC1 Тест-кейс для перегляду розкладу матчів:

* Відкрити веб-додаток для перегляду розкладу Англійської прем'єр-ліги.
* Перевірити наявність списку майбутніх матчів.
* Впевнитись, що кожен матч відображається з датою, часом, командами та локацією.
* Переконатись, що матчі відсортовані за датою від найближчого до найвіддаленішого.

TC2 Тест-кейс для перегляду деталей конкретного матчу:

* Відкрити веб-додаток для перегляду розкладу Англійської прем'єр-ліги.
* Обрати певний матч зі списку.
* Перевірити, що відображаються деталі про обраний матч, такі як команди, дата, час, локація, стадіон тощо.
* Переконатись, що інші матчі відсутні у списку.

TC3 для перегляду турнірної таблиці:

* Відкрити веб-додаток для перегляду розкладу Англійської прем'єр-ліги.
* Обрати опцію перегляду турнірної таблиці.
* Переконатись, що відображаються команди з відповідними показниками, такими як кількість очок, перемоги, нічиї, поразки тощо.
* Перевірити, що команди відсортовані за кількістю очок у зворотному порядку.

TC4 для відображення статистики команд:

* Відкрити веб-додаток для перегляду розкладу Англійської прем'єр-ліги.
* Обрати певну команду зі списку команд.
* Переконатись, що відображаються статистичні дані про обрану команду, такі як кількість голів, жовтих та червоних карток, відсоток володіння м'ячем тощо.

TC5 Тест-кейс для сортування матчів за датою:

* Відкрити веб-додаток для перегляду розкладу Англійської прем'єр-ліги.
* Вибрати опцію сортування матчів за датою.
* Перевірити, що матчі відображаються від найближчого до найвіддаленішого за датою.
* Переконатись, що інші матчі відсутні у списку.

TC6 Тест-кейс для додавання ставки на матч:

* Відкрити веб-додаток для перегляду розкладу Англійської прем'єр-ліги.
* Обрати певний матч зі списку.
* Перейти до сторінки матчу та знайти опцію "Додати ставку".
* Ввести деталі ставки, такі як тип ставки (наприклад, перемога команди А), суму ставки та коефіцієнт.
* Натиснути кнопку "Зберегти ставку".
* Переконатись, що ставка успішно додана і з'явилась у списку ставок користувача.
* Переконатись, що відображаються правильні дані про ставку, такі як тип ставки, сума, коефіцієнт та обраний матч.
* Переконатись, що ставка має статус "Активна" або "В очікуванні".

TC7 Тест-кейс для перегляду попередніх результатів матчів:

* Відкрити веб-додаток для перегляду розкладу Англійської прем'єр-ліги.
* Обрати певний матч зі списку.
* Перейти до сторінки матчу та переглянути попередні результати змагань команд.
* Переконатись, що відображаються дані про попередні матчі, такі як результати, голи, карти тощо.

TC8 Тест-кейс для реєстрації користувача:

* Відкрити веб-додаток для перегляду розкладу Англійської прем'єр-ліги.
* Обрати певний матч зі списку.
* Перейти до сторінки матчу та переглянути попередні результати змагань команд.
* Переконатись, що відображаються дані про попередні матчі, такі як результати, голи, карти тощо.

Результати функціонального тестування веб-ресурсу продуктової крамниці наведені в таблиці 3.2.

Як можна побачити з результатів тестування, всі функціональні тести пройдені успішно, що свідчить про те, що фактична поведінка веб-ресурсу співпадає з очікуваною поведінкою, визначеною у функціональних вимогах.

Таблиця 3.2 – Протокол функціонального тестування веб-ресурсу

|  |  |  |  |
| --- | --- | --- | --- |
| Номер тест-кейсу | Очікуваний результат | Фактичний результат | Результат тестування |
| TC1 | Успішний перегляд розкладу матчів | Розклад матчів успішно з’являться на екрані | Успішно |
| TC2 | Успішний перегляд деталей конкретного матчу | Інформація про конкретний матч успішно з’являться на екрані | Успішно |
| TC3 | Успішний перегляд турнірної таблиці | Турнірна таблиця успішно з’являться на екрані | Успішно |
| TC4 | Успішне відображення статистики команд | Статистика команд успішно з’являться на екрані | Успішно |
| TC5 | Успішне сортування матчів за датою | Сортування матчів за датою відбувається успішно | Успішно |
| TC6 | Успішне додавання ставки на матч | Ставка успішно додається до бази даних | Успішно |
| TC7 | Успішний перегляд попередніх результатів матчів | Попередні результати матчів успішно виводяться на екран | Успішно |
| TC8 | Успішне реєстрація користувача | Новий користувач був успішно доданий до бази даних | Успішно |

## Інструкція користувача веб-ресурсу

Для забезпечення успішного користувацького досвіду використання розробленого веб-ресурсу необхідно скласти інструкцію користувача, яка включає в себе знімки екранів та пояснювальний текст до кожного знімка. Це допоможе користувачам легко зорієнтуватись у функціоналі веб-ресурсу та ефективно використовувати його.

На рисунку 3.2 зображено вікно створення нового акаунту. Дизайн сторінки виконаний в світлих тонах з використанням контрастних елементів, що робить її зручною для використання.

На сторінці присутня форма реєстрації, яка містить поля для введення основної інформації, такої як ім'я, адреса електронної пошти та пароль. Також є кнопка "Register", яка відправляє заповнену форму на сервер для обробки. Кнопка входу знаходиться під формою реєстрації та дозволяє користувачам, які вже зареєструвалися, увійти на сайт.

Ця сторінка є важливою для користувачів, оскільки дозволяє їм зареєструватися на сайті та отримати доступ до послуг. Дизайн сторінки є зручним та легким для використання, що сприяє покращенню користувацького досвіду.

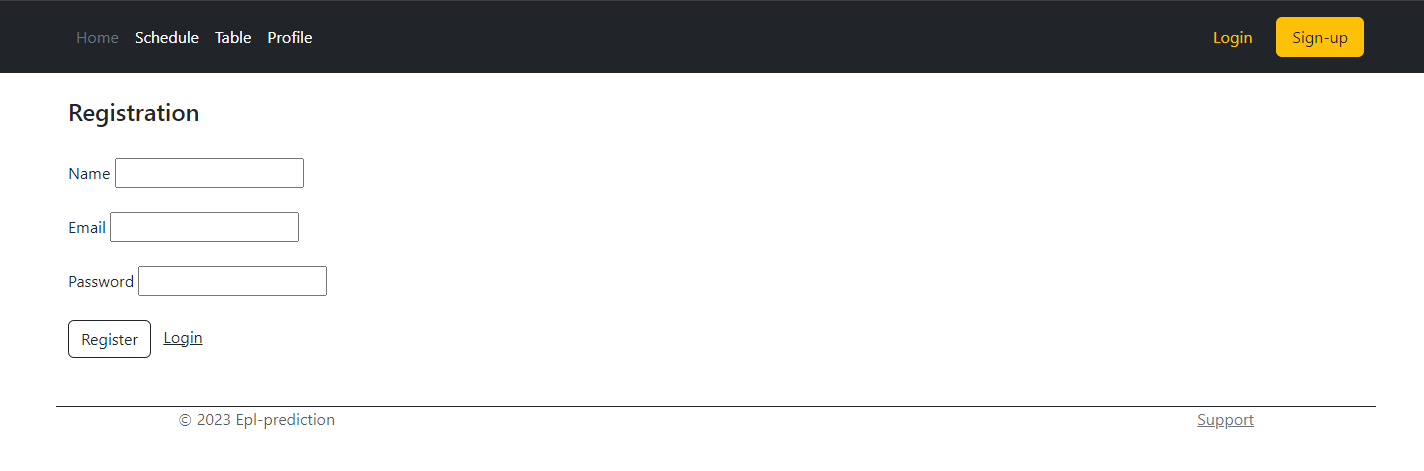


Рисунок 3.2 – Екранна форма створення нового акаунту

На рисунку 3.3 зображене вікно логіну веб-ресурсу. Вона містить форму для введення інформації користувача та кнопку "Login", щоб здійснити вхід на сайт.

На сторінці присутня форма входу, яка містить поля для введення ім'я користувача та пароля. Крім того, є кнопка "Login", яка дозволяє користувачам здійснити вхід на сайт.

Посилання на сторінку реєстрації дозволяє користувачам зареєструватися на сайті та отримати доступ до послуг.



Рисунок 3.3 – Екранна форма логіну

На рисунку 3.4 зображена екранна форма головного вікна веб-ресурсу. Вона містить інформацію про продукти та надає користувачам можливість знайти потрібну інформацію та здійснити необхідні дії.

Шапка сайту містить навігаційне меню та кнопки "Login" та "Sign-up", щоб здійснити вхід на сайт.

Секція "Schedule" містить більш детальну інформацію про розклад. Футер сайту містить контактну інформацію.

Ця сторінка є важливою для користувачів, оскільки вона надає інформацію про розклад та дозволяє здійснити необхідні дії, такі як вхід на сайт, навігація сайтом і т.д..

Дизайн сторінки є зручним та легким для використання, що сприяє покращенню користувацького досвіду.

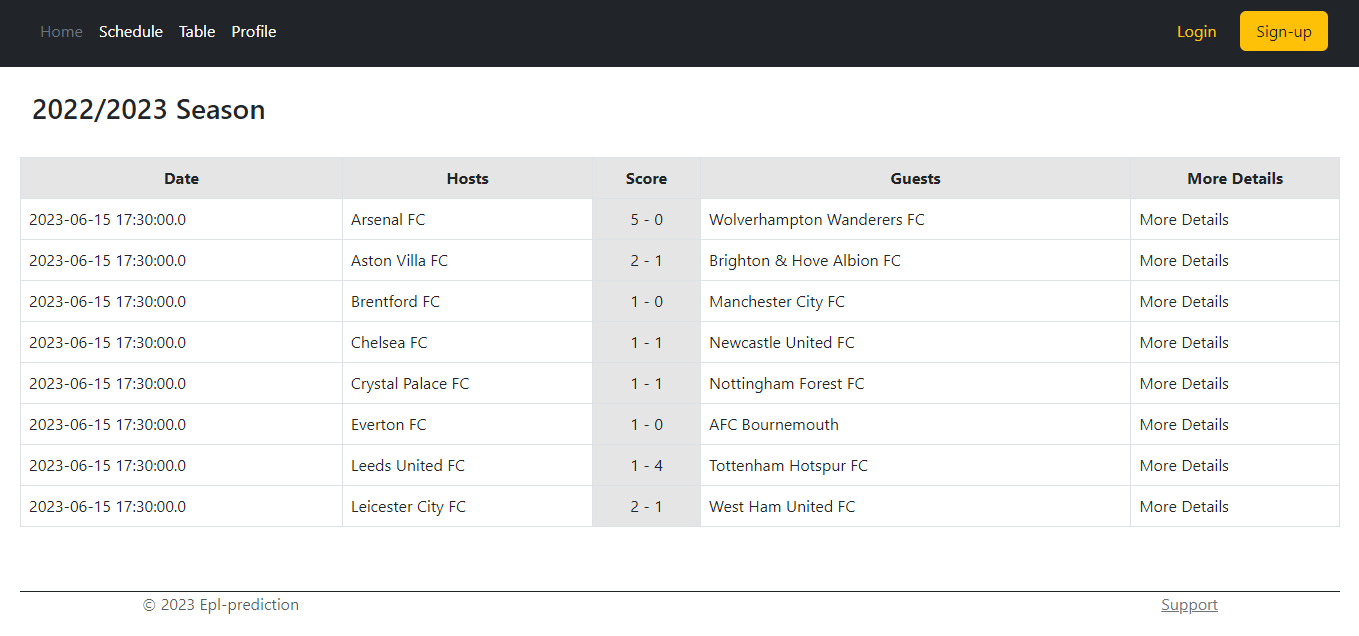


Рисунок 3.4 – Екранна форма головного вікна веб-ресурсу

На рисунку 3.5 зображена екранна форма турнірної таблиці. На сторінці надається оновлена турнірна таблиця, де можно побачити позицію кожної команди, кількість набраних очок, виграні, нічиї та програні матчі. Дизайн сторінки привабливий та легкий для сприйняття. На сторінці присутні наступні елементи:

* шапка сайту, яка містить навігаційне меню;
* турнірна таблиця, яка містить позицію кожної команди, кількість набраних очок, виграні, нічиї та програні матчі.

Футер сайту містить навігаційне меню, контактну інформацію. Дизайн сторінки є зручним та легким для використання, що сприяє покращенню користувацького досвіду.

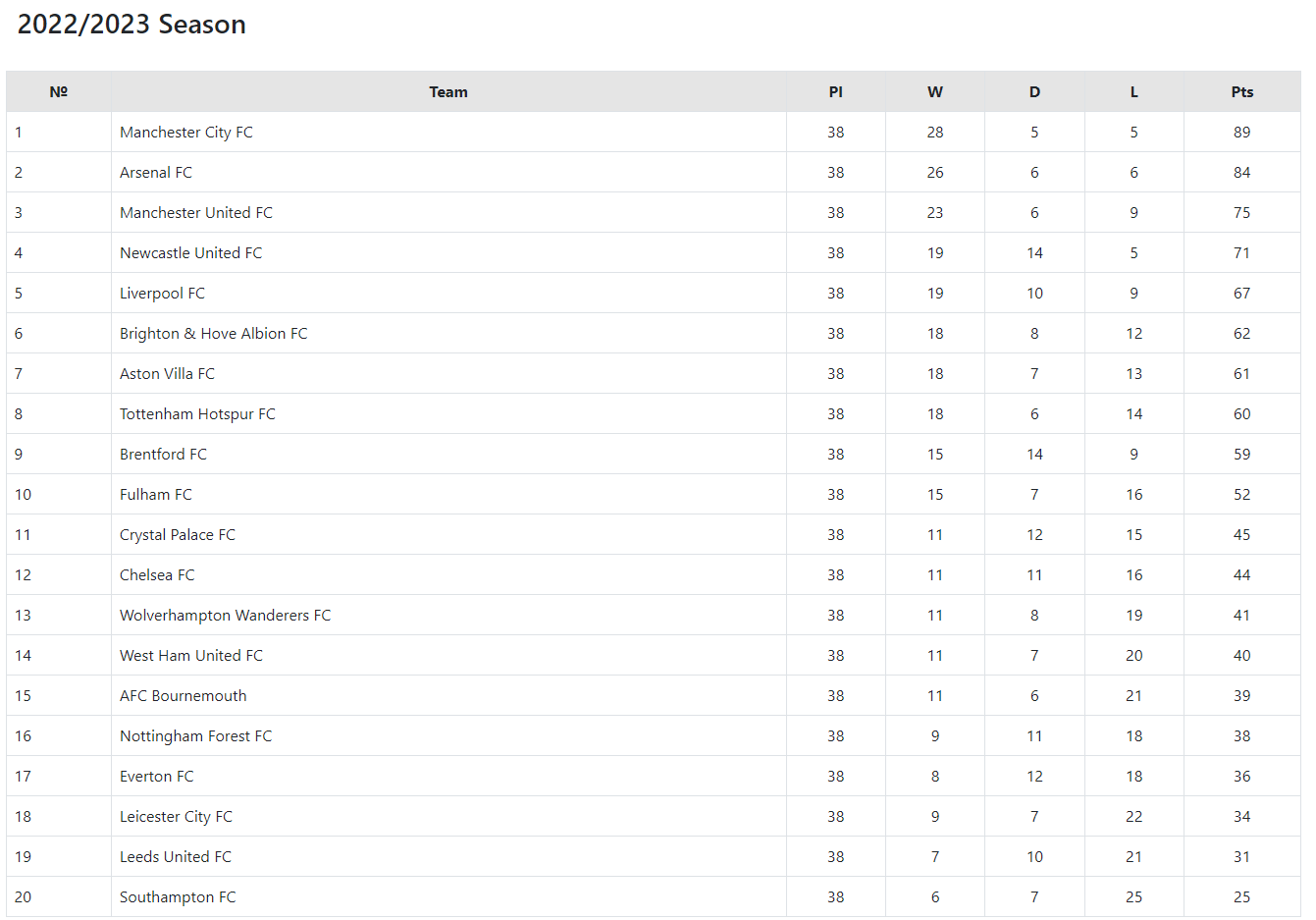


Рисунок 3.5 – Екранна форма турнірної таблиці

На рисунку 3.6 зображена екранна форма вікна клубу. На сторінці розміщена основні відомості про клуб, такі як його логотип, назва та статистика

На сторінці присутні картка клубу, яка містить логотип, назву, та статистику клубу.

Футер сайту містить навігаційне меню, контактну інформацію.

Структура сторінки досить логічна та зрозуміла, що дозволяє користувачам легко зорієнтуватися та знайти необхідну інформацію. Дизайн сторінки є досить мінімалістичним, що сприяє увазі на основну інформацію та забезпечує більш зручний користувацький досвід.

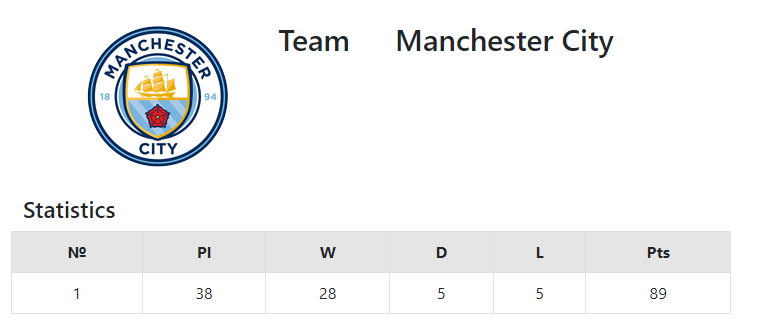
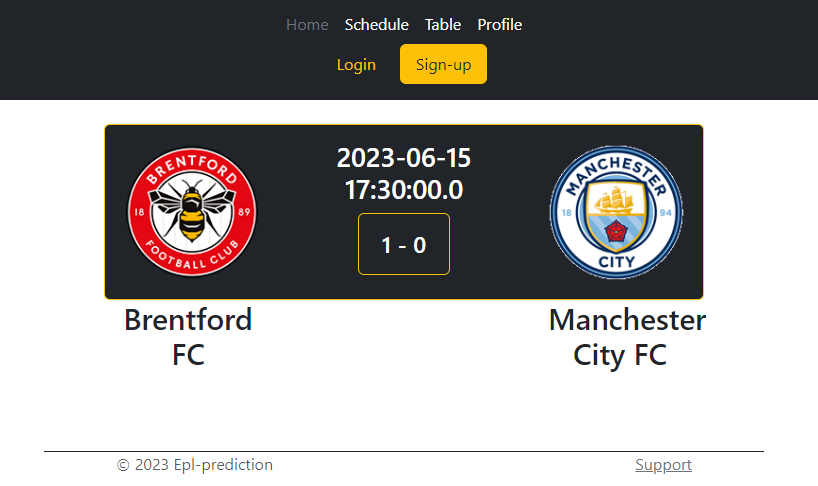


Рисунок 3.6 – Екранна форма вікна клубу

На рисунку 3.7 зображена екранна форма вікна матчу. На цій сторінці користувач може переглянути детальну інформацію про матч та зробити ставку (рисунок 3.8).

На сторінці присутні важливі елементи для успішного оформлення ставки, такі як кнопка "Bet", яка відправляє інформацію про ставку на сервер.

Сторінка має адаптивний дизайн, що дозволяє користувачам зручно переглядати її на різних пристроях та розмірах екранів.



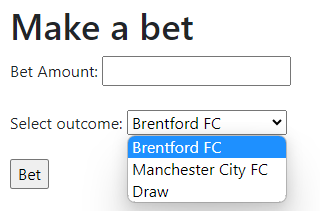
Рисунок 3.7 – Екранна форма вікна матчу

Рисунок 3.8 – Екранна форма оформлення ставки

## Вихідний код веб-ресурсу

Вихідний код програмних класів розробленого веб-ресурсу наведений в Додатку А даної роботи.

## Висновки до третього розділу

В даному розділі курсової роботи було проведено програмну реалізацію веб-ресурсу продуктової крамниці на базі фреймворку Spring Boot. Для того, щоб провести програмну реалізацію, був використаний проєкт веб-ресурсу, який був розроблений в другому розділі даної роботи.

Перш за все, була розглянута структура серверного програмного проєкту. Були визначені основні програмні компоненти, з яких складається програмний проєкт веб-ресурсу, а також наданий перелік основних класів, які відносяться до цих компонентів.

Далі була розроблена концептуальна діаграма класів, яка дозволяє визначити основні класи веб-ресурсу та визначити взаємовідносини між цими класами.

Окремо було розглянуто питання використання системи контроля версій для упорядкування розробки веб-ресурсу. Були визначені основні репозиторії проєкту та надані основні показники метрик цих репозиторіїв.

Далі було проведено функціональне тестування розробленого веб-ресурсу. Був розроблений протокол тестування у вигляді множини тест-кейсів та проведено тестування. Було визначено, що всі тест-кейси були пройдені, а отже фактична поведінка веб-ресурсу співпадає з очікуваною поведінкою, визначеною у функціональних вимогах.

Також була розроблена інструкція користувача у вигляді множини знімків екрану та пояснювального тексту, який визначає функціонал додатку у даному вікна та можливі дії користувача. Крім того, наданий вихідний код розробленого веб-ресурсу.

# ВИСНОВКИ

У даній курсовій роботі було розроблено веб-додаток Epl Prediction на основі фреймворку Spring Boot. .

Цей додаток дозволяє користувачам переглядати розклад матчів Англійської прем'єр-ліги та робити прогнози на результати матчів, такі як перемога, поразка або нічия. Розклад та оновлення результатів матчів будуть отримуватися в реальному часі за допомогою football-data-api. Також на додатку буде доступна актуальна турнірна таблиця.

У першому розділі даної курсової роботи був проведений огляд інформаційних технологій для розробки веб-додатку для перегляду розкладу англійської прем'єр-ліги. У якості мови програмування було обрано Java. У якості фреймворку для розробки веб-додатку було вирішено обрати фреймворк Spring Boot. Залучення цього фреймворку дозволяє зосередитись на бізнес-логіці додатку, а не на конфігурації та інфраструктурних аспектах, що суттєво полегшує процес його створення. Для робити з базою даних використовується Hibernate та JPA. У якості бази даних обрана MySQL. Для полегшення роботи з кодом використовується Lombok. Для створення динамічних веб-сторінок використовується Thymeleaf разом з HTML та Bootstrap. Для захисту додатку від атак використовується Spring Security

У другому розділі роботи було проведено проектування веб-ресурсу. Була визначена мета сайту, його потенційна аудиторія та основні можливості. Визначені основні вимоги до веб-ресурсу, включаючи функціональні та нефункціональні. Також була розроблена діаграма сценаріїв, діаграма інформаційних потоків, діаграма станів. Крім того, була розроблена діаграма розгортання веб-ресурсу. Також була визначена схема даних системи та наданий опис ключовим технологіям.

У третьому розділі проекту було здійснено програмну реалізацію веб-ресурсу. Була розглянута структура серверного програмного проєкту, визначені основні компоненти програмного забезпечення та розроблена діаграма класів. Також було розглянуто питання використання системи контролю версій для організації розробки веб-ресурсу. Було проведено функціональне тестування, розроблено інструкцію користувача у вигляді набору знімків екрану та пояснювального тексту, а також надано вихідний код розробленого веб-ресурсу.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The importance of information technology in retail. URL: https://www.retailcouncil.org/community/technology/the-importance-of-information-technology-in-retail/ (дата звернення 01.03.2023).
2. The Importance of Information Technology in Retail. URL: https://www.aeologic.com/blog/the-importance-of-information-technology-in-retail/ (дата звернення 01.03.2023).
3. Веб-додаток Premeir League URL: https://www.premierleague.com/ (дата звернення 01.03.2023).
4. Веб-додаток ESPN URL: https://www.espn.co.uk/ (дата звернення 01.03.2023).
5. Spring Boot. URL: https://spring.io/guides/gs/rest-service/ (дата звернення 01.03.2023).
6. Building a RESTful Web Service URL: https://angular.io/guide/component-overview (дата звернення 01.03.2023).
7. Bootstrap docx. URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення 01.03.2023).
8. Spring Security URL: https://spring.io/projects/spring-security (дата звернення 01.03.2023).
9. Spring Security Architecture URL: https://spring.io/guides/topicals/spring-security-architecture/ (дата звернення 01.03.2023).
10. MySQL Home. URL: https://www.mysql.com/ (дата звернення 01.03.2023).
11. Football Data Api Documentation

<https://docs.football-data-api.com/d/>

ДОДАТОК А

КОД ПРОГРАМИ

Клас SecurityConfig:

package com.OOP.eplpredictions.configurations;  
  
import com.OOP.eplpredictions.services.CustomUserDetailsService;  
import lombok.AllArgsConstructor;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configurers.AbstractHttpConfigurer;  
import org.springframework.security.config.annotation.web.configurers.LogoutConfigurer;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;  
  
@Configuration  
@EnableWebSecurity  
@AllArgsConstructor  
public class SecurityConfig {  
  
 private final CustomUserDetailsService userDetailService;  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 http  
 .authorizeHttpRequests((requests) -> requests  
 .requestMatchers("/profile")  
 .hasAnyAuthority("ROLE\_ADMIN", "ROLE\_USER")  
 .requestMatchers("/", "/error","/table", "/club/\*\*","/match/\*\*","/registration", "/schedule", "/css/style.css", "https://icons-for-free.com/download-icon-basketball-131983719702443362\_512.png")  
 .permitAll()  
 .anyRequest().authenticated()  
 )  
// .headers(headers -> headers.frameOptions().disable())  
 .formLogin((form) -> form  
 .loginPage("/login")  
 .permitAll()  
 )  
 .csrf(AbstractHttpConfigurer::disable)  
 .logout(LogoutConfigurer::permitAll);  
  
 return http.build();  
 }  
  
  
 public void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.userDetailsService(userDetailService)  
 .passwordEncoder(passwordEncoder());  
 }  
  
 @Bean  
 PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder(8);  
 }  
}

Клас ClubController:

package com.OOP.eplpredictions.controllers;  
  
import com.OOP.eplpredictions.entities.Club;  
import com.OOP.eplpredictions.services.ClubService;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
  
@Controller  
public class ClubController {  
 ClubService clubService;  
  
  
// @GetMapping(path = "/table")  
// public ResponseEntity<List<Club>> table(){  
// return new ResponseEntity<>(clubService.getAllClubs(), HttpStatus.OK);  
// }  
  
 public ClubController(ClubService clubService) {  
 this.clubService = clubService;  
 }  
  
 @GetMapping(path = "/table")  
 public String table(Model model){  
 Iterable<Club> clubs = clubService.getAllClubs();  
 model.addAttribute("clubs", clubs);  
 return "table";  
 }  
  
 @GetMapping(path = "/club/{id}")  
 public String club(@PathVariable int id, Model model){  
 model.addAttribute("club", clubService.getClub(id));  
 return "club";  
 }  
}

Клас MatchController:

package com.OOP.eplpredictions.controllers;  
  
  
import com.OOP.eplpredictions.entities.Match;  
import com.OOP.eplpredictions.entities.User;  
import com.OOP.eplpredictions.services.MatchService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
  
import java.util.List;  
  
@Controller  
public class MatchController {  
 private final MatchService matchService;  
  
  
 @Autowired  
 public MatchController(final MatchService matchService) {  
 this.matchService = matchService;  
 }  
  
// @GetMapping(path = "/matches")  
// public ResponseEntity<List<Match>> listMatches() {  
// return new ResponseEntity<>(matchService.getAllMatches(), HttpStatus.OK);  
// }  
 @GetMapping("/matches")  
 public String matches(Model model) {  
 Iterable<Match> matches  
// = matchService.getSchedule();  
 = matchService.getAllMatches();  
  
 model.addAttribute("matches", matches);  
 return "schedule";  
 }  
  
 @GetMapping("/schedule")  
 public String schedule(Model model) {  
 Iterable<Match> matches  
 = matchService.getSchedule();  
  
 model.addAttribute("matches", matches);  
 return "schedule";  
 }  
  
 @GetMapping("/match/{id}")  
 public String match(@PathVariable int id, Model model) {  
 matchService.getMatchById(id).ifPresent(o -> model.addAttribute("match", o));  
 return "match";  
 }  
  
 @GetMapping("/")  
 public String main(Model model) {  
 Iterable<Match> matches  
 = matchService.getSchedule();  
  
 model.addAttribute("matches", matches);  
 return "schedule";  
 }  
  
}

Клас PredictionController:

package com.OOP.eplpredictions.controllers;  
  
import com.OOP.eplpredictions.entities.Prediction;  
import com.OOP.eplpredictions.entities.User;  
import com.OOP.eplpredictions.services.PredictionService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.PostMapping;  
  
@Controller  
@RequiredArgsConstructor  
public class PredictionController {  
  
 public final PredictionService predictionService;  
  
 @PostMapping("/bet")  
 public String addPrediction(Prediction prediction, Model model) {  
 if (predictionService.createPrediction(prediction)) {  
 model.addAttribute("Error");  
 return "/";  
 }  
 return "/schedule";  
 }  
  
}

Клас UserController:

package com.OOP.eplpredictions.controllers;  
  
import com.OOP.eplpredictions.entities.User;  
import com.OOP.eplpredictions.services.UserService;  
import com.OOP.eplpredictions.services.impl.UserServiceImpl;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
  
@Controller  
@RequiredArgsConstructor  
public class UserController {  
 public final UserService userService;  
  
 @GetMapping("/login")  
 public String login() {  
 return "login";  
 }  
  
 @GetMapping("/registration")  
 public String registration() {  
 return "registration";  
 }  
  
 @PostMapping("/registration")  
 public String userAdd(User user, Model model) {  
 if (userService.createUser(user)) {  
 model.addAttribute("Error", "Email " + user.getEmail() + "has already been used");  
 return "registration";  
 }  
 return "redirect:/login";  
 }  
  
 @GetMapping("/profile")  
 public String profile(Model model) {  
 model.addAttribute("user", model);  
 return "profile";  
 }  
}

Клас Club:

package com.OOP.eplpredictions.entities;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Id;  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@Builder  
public class Club {  
 private int id;  
 private String name;  
 private String country;  
// private int founded;  
 private String logoLink;  
 private int tablePosition;  
 private int points;  
 private int seasonWins;  
 private int seasonDraws;  
 private int seasonLoses;  
 private int matchesPlayed;  
}

Клас ClubEntity:

package com.OOP.eplpredictions.entities;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import lombok.experimental.SuperBuilder;  
  
@Entity  
@NoArgsConstructor  
@AllArgsConstructor  
@Data  
@SuperBuilder  
@Table(name = "clubs")  
public class ClubEntity {  
 @Id  
 @Column(name = "club\_id")  
 private int id;  
 @Column(name = "name")  
 private String name;  
 @Column(name = "country")  
 private String country;  
// @Column(name = "founded")  
// private int founded;  
  
 @Column(name = "logo\_link")  
 private String logoLink;  
}

Клас Match:

package com.OOP.eplpredictions.entities;  
  
import lombok.\*;  
  
import java.util.Date;  
  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@Builder  
public class Match {  
 private int id;  
 private Club homeTeam;  
 private Club awayTeam;  
 private Date time;  
  
 //dynamic variables  
 private String status;  
 private String score;  
}

Клас MatchEntity:

package com.OOP.eplpredictions.entities;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import lombok.experimental.SuperBuilder;  
  
import java.util.Date;  
  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@SuperBuilder  
@Entity  
@Table(name = "matches")  
public class MatchEntity {  
 @Id  
 @Column(name = "match\_id")  
 private int id;  
// @Column(name = "home\_name")  
// private String homeName;  
// @Column(name = "away\_name")  
// private String awayName;  
  
 @JoinColumn(name = "home\_team", referencedColumnName = "club\_id")  
 @ManyToOne(cascade = CascadeType.*ALL*, optional = false)  
 private ClubEntity homeTeam;  
  
 @JoinColumn(name = "away\_team", referencedColumnName = "club\_id")  
 @ManyToOne(cascade = CascadeType.*ALL*, optional = false)  
 private ClubEntity awayTeam;  
  
 @Column(name = "time")  
 @Temporal(TemporalType.*TIMESTAMP*)  
 private Date time;  
 @Column(name = "status")  
 private String status;  
  
}

Клас Prediction:

package com.OOP.eplpredictions.entities;  
  
import com.OOP.eplpredictions.entities.enums.Result;  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@Builder  
public class Prediction {  
 private int id;  
 private MatchEntity match;  
 private User user;  
 private int points;  
 private Result result;  
}

Клас PredictionEntity:

package com.OOP.eplpredictions.entities;  
  
import com.OOP.eplpredictions.entities.enums.Result;  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import lombok.experimental.SuperBuilder;  
  
@Entity  
@Table(name = "predictions", uniqueConstraints = {  
 @UniqueConstraint(columnNames = {"'match'", "'user'"})  
})  
//@Table(name = "predictions")  
@NoArgsConstructor  
@AllArgsConstructor  
@SuperBuilder  
@Data  
public class PredictionEntity {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "prediction\_id")  
 private int id;  
 @JoinColumn(name = "'match'", referencedColumnName = "match\_id")  
 @ManyToOne(cascade = CascadeType.*ALL*, optional = false)  
 private MatchEntity match;  
 @JoinColumn(name = "'user'", referencedColumnName = "id")  
 @ManyToOne(cascade = CascadeType.*ALL*, optional = false)  
 private User user;  
 @Column(name = "points")  
 private int points;  
 @Column(name = "result")  
 private Result result;  
}

Клас User:

package com.OOP.eplpredictions.entities;  
  
import com.OOP.eplpredictions.entities.enums.Role;  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;  
  
import java.time.LocalDateTime;  
import java.util.Collection;  
import java.util.HashSet;  
import java.util.Set;  
  
@Entity  
@Table(name = "users")  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
public class User implements UserDetails {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "id")  
 private Long id;  
 @Column(name = "email", unique = true)  
 private String email;  
 @Column(name = "name")  
 private String name;  
 @Column(name = "points")  
 private Integer points;  
 @Column(name = "active")  
 private boolean active;  
 @Column(name = "password")  
 private String password;  
 @ElementCollection(targetClass = Role.class, fetch = FetchType.*EAGER*)  
 @CollectionTable(name = "user\_role", joinColumns = @JoinColumn(name = "user\_id"))  
 @Enumerated(EnumType.*STRING*)  
 private Set<Role> roles = new HashSet<>();  
 private LocalDateTime dateOfCreation;  
  
 @PrePersist  
 private void init() {  
 dateOfCreation = LocalDateTime.*now*();  
 }  
  
 //security  
  
 @Override  
 public Collection<? extends GrantedAuthority> getAuthorities() {  
 return roles;  
 }  
  
 @Override  
 public String getUsername() {  
 return email;  
 }  
  
 @Override  
 public boolean isAccountNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isAccountNonLocked() {  
 return true;  
 }  
  
 @Override  
 public boolean isCredentialsNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isEnabled() {  
 return active;  
 }  
}

Клас FootballDataApiRepositoryImpl:

package com.OOP.eplpredictions.repositories.impl;  
  
import com.OOP.eplpredictions.entities.Club;  
import com.OOP.eplpredictions.entities.Match;  
import com.OOP.eplpredictions.utils.ApiUtil;  
import com.OOP.eplpredictions.repositories.ApiRepository;  
import com.fasterxml.jackson.databind.JsonNode;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import org.springframework.stereotype.Component;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
@Component  
public class FootballDataApiRepositoryImpl implements ApiRepository {  
 private final String key = "test85g57";  
 private final int seasonId = 7704; // EPL season 2022/2023  
  
  
 @Override  
 public List<Match> getAllMatches() {  
 String link = String  
 .*format*("https://api.football-data-api.com/league-matches?key=%s&season\_id=%d", key, seasonId);  
  
 return mapResponseToMatchList(ApiUtil.*apiToString*(link));  
 }  
  
 @Override  
 public Match getMatch(int matchId) {  
 String link = String  
 .*format*("https://api.football-data-api.com/match?key=%s&match\_id=%d", key, matchId);  
  
 return mapResponseToMatch(ApiUtil.*apiToString*(link));  
 }  
  
 @Override  
 public List<Club> getAllClubs() {  
 String link = String  
 .*format*("https://api.football-data-api.com/league-teams?key=%s&season\_id=%d&include=stats", key, seasonId);  
  
 return mapResponseToClubList(ApiUtil.*apiToString*(link));  
 }  
  
 @Override  
 public Club getClub(int clubId) {  
 String link = String  
 .*format*("https://api.football-data-api.com/team?key=%s&team\_id=%d", key, clubId);  
  
 return mapResponseToClub(ApiUtil.*apiToString*(link));  
 }  
  
 private List<Match> mapResponseToMatchList(String response) {  
 ObjectMapper objectMapper = new ObjectMapper();  
 List<Match> matches = new ArrayList<>();  
 List<Club> clubs = getAllClubs();  
 try {  
 JsonNode root = objectMapper.readTree(response);  
  
 JsonNode fixtures = root.get("data");  
  
 int id;  
 Club homeTeam;  
 Club awayTeam;  
 Date date;  
 String dateStr;  
 String status;  
 String score;  
  
 for (JsonNode n : fixtures) {  
 id = n.get("id").asInt();  
 homeTeam = clubs.stream().filter(club -> club.getId()==n.get("homeID").asInt())  
 .findFirst().orElse(new Club());  
 awayTeam = clubs.stream().filter(club -> club.getId()==n.get("awayID").asInt())  
 .findFirst().orElse(new Club());  
  
 int dateUnix = n.get("date\_unix").asInt();  
 date = new Date((long) dateUnix \* 1000);  
  
 int homeGoals = n.get("homeGoalCount").asInt();  
 int awayGoals = n.get("awayGoalCount").asInt();  
 score = String.*format*("%d - %d", homeGoals, awayGoals);  
  
 status = n.get("status").asText();  
 matches.add(new Match(id, homeTeam, awayTeam, date, status, score));  
 }  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
  
 return matches;  
 }  
  
 private Match mapResponseToMatch(String response) {  
 ObjectMapper objectMapper = new ObjectMapper();  
 Match match = new Match();  
 try {  
 JsonNode root = objectMapper.readTree(response);  
  
 JsonNode fixture = root.get("data");  
  
 int id;  
 Club homeTeam;  
 Club awayTeam;  
 Date date;  
 String dateStr;  
 String status;  
 String score;  
  
 id = fixture.get("id").asInt();  
 homeTeam = getClub(fixture.get("homeID").asInt());  
 awayTeam = getClub(fixture.get("awayID").asInt());  
  
 int dateUnix = fixture.get("date\_unix").asInt();  
 date = new Date((long) dateUnix \* 1000);  
  
 int homeGoals = fixture.get("homeGoalCount").asInt();  
 int awayGoals = fixture.get("awayGoalCount").asInt();  
 score = String.*format*("%d - %d", homeGoals, awayGoals);  
  
 status = fixture.get("status").asText();  
  
 match = Match.*builder*()  
 .id(id)  
 .homeTeam(homeTeam)  
 .awayTeam(awayTeam)  
 .time(date)  
 .status(status)  
 .score(score)  
 .build();  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
  
 return match;  
 }  
  
 private List<Club> mapResponseToClubList(String response) {  
 ObjectMapper objectMapper = new ObjectMapper();  
 List<Club> clubs = new ArrayList<>();  
 try {  
 JsonNode root = objectMapper.readTree(response);  
  
 JsonNode data = root.get("data");  
  
 int id;  
 String name;  
 String country;  
 int founded;  
 int tablePosition;  
 int points;  
 int seasonWins;  
 int seasonDraws;  
 int seasonLoses;  
 int matchesPlayed;  
 String logoLink;  
  
 for (JsonNode n : data) {  
 id = n.get("id").asInt();  
 name = n.get("name").asText();  
 country = n.get("country").asText();  
// founded = n.get("founded").asInt();  
 tablePosition = n.get("table\_position").asInt();  
 logoLink = n.get("image").asText();  
  
 JsonNode stats = n.get("stats");  
  
 seasonWins = stats.get("seasonWinsNum\_overall").asInt();  
 seasonDraws = stats.get("seasonDrawsNum\_overall").asInt();  
 seasonLoses = stats.get("seasonLossesNum\_overall").asInt();  
 points = seasonDraws + seasonWins \* 3;  
 matchesPlayed = stats.get("seasonMatchesPlayed\_overall").asInt();  
  
  
 clubs.add(new Club(id, name, country, logoLink, tablePosition, points, seasonWins, seasonDraws, seasonLoses, matchesPlayed));  
// matches.add(new Match(id, homeName, awayName, date, status, score));  
 }  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
  
 return clubs;  
 }  
  
 private Club mapResponseToClub(String response) {  
 ObjectMapper objectMapper = new ObjectMapper();  
 Club club = new Club();  
 try {  
 JsonNode root = objectMapper.readTree(response);  
  
 JsonNode data = root.get("data");  
 JsonNode clubInEpl = data.get(3);  
 for (JsonNode n : data) {  
 if (n.get("competition\_id").asInt() == seasonId){  
 clubInEpl = n;  
 }  
 }  
  
  
 int id;  
 String name;  
 String country;  
 int founded;  
 String logoLink;  
 int tablePosition;  
 int points;  
 int seasonWins;  
 int seasonDraws;  
 int seasonLoses;  
 int matchesPlayed;  
  
 id = clubInEpl.get("id").asInt();  
 name = clubInEpl.get("name").asText();  
 country = clubInEpl.get("country").asText();  
 tablePosition = clubInEpl.get("table\_position").asInt();  
 logoLink = clubInEpl.get("image").asText();  
  
 JsonNode stats = clubInEpl.get("stats");  
  
 seasonWins = stats.get("seasonWinsNum\_overall").asInt();  
 seasonDraws = stats.get("seasonDrawsNum\_overall").asInt();  
 seasonLoses = stats.get("seasonLossesNum\_overall").asInt();  
 points = seasonDraws + seasonWins \* 3;  
 matchesPlayed = stats.get("seasonMatchesPlayed\_overall").asInt();  
  
  
 club = new Club(id, name, country, logoLink, tablePosition, points, seasonWins, seasonDraws, seasonLoses, matchesPlayed);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
  
 return club;  
 }

Інтерфейс ApiRepository:

package com.OOP.eplpredictions.repositories;  
  
import com.OOP.eplpredictions.entities.Club;  
import com.OOP.eplpredictions.entities.Match;  
  
import java.util.List;  
  
public interface ApiRepository {  
 public List<Match> getAllMatches();  
  
 public Match getMatch(int matchId);  
  
 public List<Club> getAllClubs();  
  
 public Club getClub(int clubId);  
}

Інтерфейс ClubRepository:

package com.OOP.eplpredictions.repositories;  
  
import com.OOP.eplpredictions.entities.Club;  
import com.OOP.eplpredictions.entities.ClubEntity;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface ClubRepository extends JpaRepository<ClubEntity,Integer> {  
}

Інтерфейс MatchRepository:

package com.OOP.eplpredictions.repositories;  
  
import com.OOP.eplpredictions.entities.MatchEntity;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import java.util.List;  
  
@Repository  
public interface MatchRepository extends JpaRepository<MatchEntity, Integer> {  
}

Інтерфейс PredictionRepository:

package com.OOP.eplpredictions.repositories;  
  
import com.OOP.eplpredictions.entities.Prediction;  
import com.OOP.eplpredictions.entities.PredictionEntity;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.stereotype.Repository;  
  
import java.util.List;  
  
@Repository  
public interface PredictionRepository extends JpaRepository<PredictionEntity, Integer> {  
 @Query("SELECT p FROM PredictionEntity p WHERE p.match.id = :matchId")  
 List<PredictionEntity> findPredictionByMatchId(int matchId);  
}

Інтерфейс UserRepository:

package com.OOP.eplpredictions.repositories;  
  
import com.OOP.eplpredictions.entities.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
  
public interface UserRepository extends JpaRepository<User, Long> {  
 User findByEmail(String email);  
}

Інтерфейс ClubService:

package com.OOP.eplpredictions.services;  
  
import com.OOP.eplpredictions.entities.Club;  
  
import java.util.List;  
  
public interface ClubService {  
 void createClub(Club club);  
  
 void updateClub(Club club);  
  
 void deleteClub(int id);  
  
 Club getClub(int id);  
  
 List<Club> getAllClubs();  
}

Інтерфейс MatchService:

package com.OOP.eplpredictions.services;  
  
import com.OOP.eplpredictions.entities.Match;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface MatchService {  
 void createMatch(Match match);  
 void createAllMatches(List<Match> matches);  
 Match updateMatch(Match match);  
  
 void deleteMatch(int id);  
  
 Optional<Match> getMatchById(int matchId);  
  
 List<Match> getAllMatches();  
 List<Match> getSchedule();  
}

Інтерфейс PredictionService:

package com.OOP.eplpredictions.services;  
  
import com.OOP.eplpredictions.entities.Prediction;  
  
import java.util.List;  
  
public interface PredictionService {  
 public boolean createPrediction(Prediction prediction);  
  
 public Prediction getPrediction(int id);  
  
 public void payOffMatchPredictions(int matchId);  
  
 public List<Prediction> getPredictionsByMatchId(int matchId);  
  
 public List<Prediction> getAllPredictions();  
}

Інтерфейс PredictionService:

package com.OOP.eplpredictions.services;  
  
import com.OOP.eplpredictions.entities.User;  
import com.OOP.eplpredictions.entities.enums.Role;  
  
import java.util.List;  
  
public interface UserService {  
 public void updateUser(User user);  
 public void updateUsers(List<User> users);  
 public boolean createUser(User user);  
  
 public void deleteUser(Long id);  
  
 public User getUserById(Long id);  
}

Клас ClubServiceImpl:

package com.OOP.eplpredictions.services.impl;  
  
import com.OOP.eplpredictions.entities.Club;  
import com.OOP.eplpredictions.entities.ClubEntity;  
import com.OOP.eplpredictions.repositories.ApiRepository;  
import com.OOP.eplpredictions.repositories.ClubRepository;  
import com.OOP.eplpredictions.services.ClubService;  
import org.springframework.stereotype.Service;  
  
import java.util.Comparator;  
import java.util.List;  
  
@Service  
public class ClubServiceImpl implements ClubService {  
 private final ClubRepository clubRepository;  
 private final ApiRepository apiRepository;  
  
 public ClubServiceImpl(ClubRepository clubRepository, ApiRepository apiRepository) {  
 this.clubRepository = clubRepository;  
 this.apiRepository = apiRepository;  
 }  
  
 @Override  
 public void createClub(Club club) {  
 clubRepository.save(clubToClubEntity(club));  
 }  
  
 @Override  
 public void updateClub(Club club) {  
 clubRepository.save(clubToClubEntity(club));  
 }  
  
 @Override  
 public void deleteClub(int id) {  
 clubRepository.deleteById(id);  
 }  
  
 @Override  
 public Club getClub(int id) {  
 return clubEntityToClub(clubRepository.findById(id).orElse(new ClubEntity()));  
 }  
  
 @Override  
 public List<Club> getAllClubs() {  
 List<Club> clubs = apiRepository.getAllClubs();  
 clubs.sort(Comparator.*comparing*(Club::getTablePosition));//sorting by table position  
 return clubs;  
 }  
  
 private ClubEntity clubToClubEntity (Club club) {  
 return ClubEntity.*builder*()  
 .id(club.getId())  
 .country(club.getCountry())  
 .name(club.getName())  
 .logoLink(club.getLogoLink())  
 .build();  
 }  
  
 private Club clubEntityToClub (ClubEntity clubEntity) {  
 return apiRepository.getClub(clubEntity.getId());  
// return clubs.stream().filter(club -> club.getId() == clubEntity.getId()).findFirst().orElse(new Club());  
 }  
}

Клас MatchServiceImpl:

package com.OOP.eplpredictions.services.impl;  
  
import com.OOP.eplpredictions.entities.Club;  
import com.OOP.eplpredictions.entities.ClubEntity;  
import com.OOP.eplpredictions.entities.Match;  
import com.OOP.eplpredictions.entities.MatchEntity;  
import com.OOP.eplpredictions.repositories.ApiRepository;  
import com.OOP.eplpredictions.repositories.MatchRepository;  
import com.OOP.eplpredictions.services.MatchService;  
import com.OOP.eplpredictions.services.PredictionService;  
import com.OOP.eplpredictions.utils.DateUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.Date;  
import java.util.List;  
import java.util.Objects;  
import java.util.Optional;  
  
@Service  
public class MatchServiceImpl implements MatchService {  
 private final MatchRepository matchRepository;  
 private final ApiRepository apiRepository;  
 private final PredictionService predictionService;  
 private final List<Club> clubs;  
  
  
 @Autowired  
 public MatchServiceImpl(MatchRepository matchRepository, ApiRepository apiRepository, PredictionService predictionService) {  
 this.matchRepository = matchRepository;  
 this.apiRepository = apiRepository;  
 this.predictionService = predictionService;  
 this.clubs = apiRepository.getAllClubs();  
 }  
  
 @Override  
 public void createMatch(Match match) {  
 matchRepository.save(matchToMatchEntity(match));  
 }  
  
 @Override  
 public void createAllMatches(List<Match> matches) {  
 List<MatchEntity> matchEntityList = matches.stream().map(this::matchToMatchEntity).toList();  
 matchRepository.saveAll(matchEntityList);  
 }  
  
 @Override  
 public Match updateMatch(Match match) {  
 return matchEntityToMatch(matchRepository.save(matchToMatchEntity(match)));  
 }  
  
 @Override  
 public void deleteMatch(int id) {  
 matchRepository.deleteById(id);  
 }  
  
 @Override  
 public Optional<Match> getMatchById(int matchId) {  
// MatchEntity match = matchRepository.findById(matchId);  
 Optional<MatchEntity> match = matchRepository.findById(matchId);  
 return match.map(this::matchEntityToMatch);  
 }  
  
 @Override  
 public List<Match> getAllMatches() {  
 List<MatchEntity> matchEntities = matchRepository.findAll();  
 return matchEntities.stream().map(this::matchEntityToMatch).toList();  
 }  
  
 @Override  
 public List<Match> getSchedule() {  
 return getAllMatches().stream()  
 .filter(match -> DateUtil.*isScheduleDate*(match.getTime())).toList();  
// .filter(match -> !DateUtil.isFirstDate7DaysLater(match.getTime(), new Date())  
// && new Date().before(match.getTime())).toList();  
 }  
  
  
 private Match matchEntityToMatch(MatchEntity matchEntity) {  
 if (!Objects.*equals*(matchEntity.getStatus(), "incomplete")) {  
 return Match.*builder*()  
 .id(matchEntity.getId())  
 .homeTeam(clubEntityToClub(matchEntity.getHomeTeam()))  
 .awayTeam(clubEntityToClub(matchEntity.getAwayTeam()))  
 .time(matchEntity.getTime())  
 .score(matchEntity.getStatus())  
 .status("complete")  
 .build();  
 }  
  
 if(new Date().before(matchEntity.getTime())){// checks if date is later than current time  
 return Match.*builder*()  
 .id(matchEntity.getId())  
// .homeName(matchEntity.getHomeName())  
// .awayName(matchEntity.getAwayName())  
 .homeTeam(clubEntityToClub(matchEntity.getHomeTeam()))  
 .awayTeam(clubEntityToClub(matchEntity.getAwayTeam()))  
 .time(matchEntity.getTime())  
 .score("- : -")  
 .status("incomplete")  
 .build();  
 }  
  
 Match match = apiRepository.getMatch(matchEntity.getId());  
  
 if (Objects.*equals*(match.getStatus(), "complete")) {  
 matchRepository.save(matchToMatchEntity(match));  
 predictionService.payOffMatchPredictions(match.getId());  
 }  
  
 return match;  
 }  
  
 private MatchEntity matchToMatchEntity(Match match) {  
 MatchEntity matchEntity = MatchEntity.*builder*()  
 .id(match.getId())  
// .homeName(match.getHomeName())  
// .awayName(match.getAwayName())  
 .homeTeam(clubToClubEntity(match.getHomeTeam()))  
 .awayTeam(clubToClubEntity(match.getAwayTeam()))  
 .time(match.getTime())  
 .status(match.getStatus())  
 .build();  
  
 if (matchEntity.getStatus().equals("complete"))  
 matchEntity.setStatus(match.getScore());  
  
 return matchEntity;  
 }  
  
 private ClubEntity clubToClubEntity (Club club) {  
 return ClubEntity.*builder*()  
 .id(club.getId())  
 .country(club.getCountry())  
 .name(club.getName())  
 .logoLink(club.getLogoLink())  
 .build();  
 }  
  
 private Club clubEntityToClub (ClubEntity clubEntity) {  
// return apiRepository.getClub(clubEntity.getId());  
 return clubs.stream().filter(club -> club.getId() == clubEntity.getId()).findFirst().orElse(new Club());  
 }  
}

Клас PredictionServiceImpl:

package com.OOP.eplpredictions.services.impl;  
  
import com.OOP.eplpredictions.entities.Prediction;  
import com.OOP.eplpredictions.entities.PredictionEntity;  
import com.OOP.eplpredictions.entities.User;  
import com.OOP.eplpredictions.entities.enums.Result;  
import com.OOP.eplpredictions.repositories.PredictionRepository;  
import com.OOP.eplpredictions.services.PredictionService;  
import com.OOP.eplpredictions.services.UserService;  
import org.springframework.stereotype.Service;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
@Service  
public class PredictionServiceImpl implements PredictionService {  
 private final PredictionRepository predictionRepository;  
 private final UserService userService;  
  
 @Override  
 public boolean createPrediction(Prediction prediction) {  
 if (!Objects.*equals*(prediction.getMatch().getStatus(), "incomplete")  
 || prediction.getUser().getPoints() < prediction.getPoints()) {  
 //u cant make a match prediction that is finished  
 return true;  
 } else {  
 User user = prediction.getUser();  
 user.setPoints(user.getPoints() - prediction.getPoints());  
 userService.updateUser(user);  
 predictionRepository.save(predictionToPredictionEntity(prediction));  
 return false;  
 }  
 }  
  
 @Override  
 public Prediction getPrediction(int id) {  
 return predictionEntityToPrediction(predictionRepository.findById(id)  
 .orElse(new PredictionEntity()));  
 }  
  
 @Override  
 public void payOffMatchPredictions(int matchId) {  
 List<Prediction> allPredictions = predictionRepository.findPredictionByMatchId(matchId)  
 .stream().map(this::predictionEntityToPrediction).toList();  
 if(!allPredictions.isEmpty()){  
 countPointsToUser(allPredictions);  
 }  
 }  
 private void countPointsToUser(List<Prediction> predictions) {  
 int allPoints = predictions.stream().mapToInt(Prediction::getPoints).sum();  
 Result matchResult = getResult(predictions.get(0).getMatch().getStatus());  
 List<Prediction> wonPredictions = predictions  
 .stream().filter(prediction -> prediction.getResult() == matchResult).toList();  
  
 if(!wonPredictions.isEmpty()) {  
 int winPoints = wonPredictions.stream().mapToInt(Prediction::getPoints).sum();  
 double coefficient = (allPoints - winPoints) / (double)winPoints;  
 List<User> wonUsers = new ArrayList<>();  
 for (Prediction p :  
 wonPredictions) {  
 User u = p.getUser();  
 u.setPoints((int) (u.getPoints() + p.getPoints() \* coefficient));  
 wonUsers.add(u);  
 }  
 userService.updateUsers(wonUsers);  
 }  
 }  
 private Result getResult(String score) {  
 Pattern pattern = Pattern.*compile*("(\\d+) - (\\d+)");  
 Matcher matcher = pattern.matcher(score);  
  
 if (matcher.matches()) {  
 int homeScore = Integer.*parseInt*(matcher.group(1));  
 int awayScore = Integer.*parseInt*(matcher.group(2));  
  
 if (homeScore > awayScore) {  
 return Result.*HOME\_WIN*;  
 } else if (homeScore < awayScore) {  
 return Result.*AWAY\_WIN*;  
 }  
 }  
  
 return Result.*DRAW*;  
 }  
  
  
 @Override  
 public List<Prediction> getPredictionsByMatchId(int matchId) {  
 return predictionRepository.findPredictionByMatchId(matchId)  
 .stream().map(this::predictionEntityToPrediction).toList();  
// return null;  
 }  
  
 @Override  
 public List<Prediction> getAllPredictions() {  
 return predictionRepository.findAll().stream()  
 .map(this::predictionEntityToPrediction).toList();  
 }  
  
 public PredictionServiceImpl(PredictionRepository predictionRepository, UserService userService) {  
 this.predictionRepository = predictionRepository;  
 this.userService = userService;  
 }  
  
 private Prediction predictionEntityToPrediction(PredictionEntity predictionEntity) {  
 return Prediction.*builder*()  
 .id(predictionEntity.getId())  
 .user(predictionEntity.getUser())  
 .match(predictionEntity.getMatch())  
 .points(predictionEntity.getPoints())  
 .result(predictionEntity.getResult())  
 .build();  
 }  
  
 private PredictionEntity predictionToPredictionEntity(Prediction prediction) {  
 return PredictionEntity.*builder*()  
 .id(prediction.getId())  
 .user(prediction.getUser())  
 .match(prediction.getMatch())  
 .points(prediction.getPoints())  
 .result(prediction.getResult())  
 .build();  
 }  
}

Клас UserServiceImpl:

package com.OOP.eplpredictions.services.impl;  
  
import com.OOP.eplpredictions.entities.User;  
import com.OOP.eplpredictions.entities.enums.Role;  
import com.OOP.eplpredictions.repositories.UserRepository;  
import com.OOP.eplpredictions.services.UserService;  
import lombok.RequiredArgsConstructor;  
import lombok.extern.slf4j.Slf4j;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
@Slf4j  
@RequiredArgsConstructor  
public class UserServiceImpl implements UserService {  
 private final UserRepository userRepository;  
 private final PasswordEncoder passwordEncoder;  
  
 @Override  
 public void updateUser(User user) {  
 userRepository.save(user);  
 }  
  
 @Override  
 public void updateUsers(List<User> users) {  
 userRepository.saveAll(users);  
 }  
  
 @Override  
 public boolean createUser(User user) {  
 if (userRepository.findByEmail(user.getEmail()) != null)  
 return false;  
  
 user.setActive(true);  
 user.setPoints(1000);  
 user.setPassword(passwordEncoder.encode(user.getPassword()));  
 user.getRoles().add(Role.*ROLE\_USER*);  
 userRepository.save(user);  
 *log*.info("Save user" + user.getEmail());  
  
 return true;  
 }  
  
 @Override  
 public void deleteUser(Long id) {  
 userRepository.deleteById(id);  
 }  
  
 @Override  
 public User getUserById(Long id) {  
 return userRepository.findById(id).orElse(null);  
 }  
  
}

Клас CustomUserDetailsService:

package com.OOP.eplpredictions.services;  
  
import com.OOP.eplpredictions.repositories.UserRepository;  
import lombok.RequiredArgsConstructor;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.stereotype.Service;  
  
@Service  
@RequiredArgsConstructor  
public class CustomUserDetailsService implements UserDetailsService {  
 private final UserRepository userRepository;  
  
 @Override  
 public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {  
 return userRepository.findByEmail(email);  
 }  
}

Абстрактний клас ApiUtil:

package com.OOP.eplpredictions.utils;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.net.HttpURLConnection;  
import java.net.URL;  
  
public abstract class ApiUtil {  
  
 public static String apiToString(String link) {  
 HttpURLConnection connection = null;  
 StringBuilder response = new StringBuilder();  
  
 try {  
 URL url = new URL(link);  
 connection = (HttpURLConnection) url.openConnection();  
 connection.setRequestMethod("GET");  
 connection.connect();  
  
 int responseCode = connection.getResponseCode();  
  
 if (responseCode == HttpURLConnection.*HTTP\_OK*) {  
 BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));  
 String inputLine;  
  
 while ((inputLine = in.readLine()) != null) {  
 response.append(inputLine);  
 }  
  
 connection.disconnect();  
 in.close();  
 } else {  
 System.*out*.println("Error: " + responseCode);  
 }  
 } catch (IOException e) {  
 return "";  
 }  
  
 return response.toString();  
 }  
}

Абстрактний клас DateUtil:

package com.OOP.eplpredictions.utils;  
  
import java.time.temporal.ChronoUnit;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.concurrent.TimeUnit;  
  
public class DateUtil {  
 public static boolean isFirstDate7DaysLater(Date firstDate, Date secondDate) {  
 long sevenDaysInMillis = 7 \* 24 \* 60 \* 60 \* 1000; // 7 days in milliseconds  
 long differenceInMillis = firstDate.getTime() - secondDate.getTime();  
  
 return differenceInMillis >= sevenDaysInMillis;  
 }  
 public static boolean isScheduleDate(Date date){  
  
 // Get today's date  
 Calendar today = Calendar.*getInstance*();  
 today.setTime(new Date());  
  
 // Get the date passed in as a parameter  
 Calendar cal = Calendar.*getInstance*();  
 cal.setTime(date);  
  
 // Subtract 2 days from today's date  
 Calendar twoDaysAgo = (Calendar) today.clone();  
 twoDaysAgo.add(Calendar.*DAY\_OF\_MONTH*, -2);  
  
 // Add 7 days to today's date  
 Calendar sevenDaysLater = (Calendar) today.clone();  
 sevenDaysLater.add(Calendar.*DAY\_OF\_MONTH*, 7);  
  
 // Check if the date falls within the valid range  
 if (cal.before(twoDaysAgo) || cal.after(sevenDaysLater)) {  
 return false;  
 }  
  
 return true;  
 }  
  
}

Клас EplPredictionsApplication:

package com.OOP.eplpredictions;  
  
import com.OOP.eplpredictions.services.MatchService;  
import com.OOP.eplpredictions.services.PredictionService;  
import com.OOP.eplpredictions.services.UserService;  
import jakarta.annotation.PostConstruct;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class EplPredictionsApplication {  
  
 UserService userService;  
 MatchService matchService;  
 PredictionService predictionService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(EplPredictionsApplication.class, args);  
 }  
  
// @PostConstruct  
// public void test() {  
// List<Match> matches = new FootballDataApiRepositoryImpl().getAllMatches();  
// matchService.createAllMatches(matches);  
// }  
  
  
 public EplPredictionsApplication(UserService userService, MatchService matchService, PredictionService predictionService) {  
 this.userService = userService;  
 this.matchService = matchService;  
 this.predictionService = predictionService;  
 }  
  
}

Темплейт footer:

<div th:fragment="footer" class="border-top border-dark row">  
 <div class="col-md-10 offset-md-1 d-flex justify-content-between">  
 <p class="text-muted text-start">© 2023 Epl-prediction</p>  
 <p class="text-end">  
 <a href="#" class="link-secondary">Support</a>  
 </p>  
 </div>  
</div>

Темплейт header:

<div class="container" th:fragment="header">  
 <div class="d-flex flex-wrap align-items-center justify-content-center justify-content-lg-start">  
 <ul class="nav col-12 col-lg-auto me-lg-auto mb-2 justify-content-center mb-md-0">  
 <li><a href="/" class="nav-link px-2 text-secondary">Home</a></li>  
 <li><a href="/schedule" class="nav-link px-2 text-white">Schedule</a></li>  
 <li><a href="/table" class="nav-link px-2 text-white">Table</a></li>  
 <li><a href="/profile" class="nav-link px-2 text-white">Profile</a></li>  
 </ul>  
 <ul class="nav nav-pills text-end">  
 <li class="nav-item me-2">  
 <a class="nav-link link-warning" href="/login">Login</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link bg-warning link-dark" aria-current="page" href="/registration">Sign-up</a>  
 </li>  
 <li class="nav-item">  
 <img th:src="@{../static/images/avatar.png}">  
 </li>  
 </ul>  
 </div>  
</div>

Темплейт club:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <!-- <link rel="stylesheet" type="text/css" href="\static\css\style.css">-->  
 <link rel="stylesheet" th:href="@{/css/style.css}">  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Epl-prediction</title>  
</head>  
<body>  
<header th:insert="~{blocks/header :: header}" class="p-3 bg-dark text-white"></header>  
<div class="container"><br>  
 <div class="row">  
 <div class="d-flex justify-content-between col-md-10 offset-md-1">  
 <img th:src="${club.logoLink}">  
 <h2>Team</h2>  
 <h2 th:text="${club.getName}"></h2><br>  
  
 </div>  
 </div><br>  
 <h4>Statistics</h4>  
 <div class="row">  
 <table class="table table-bordered table-hover table-condensed col-md-6">  
 <thead class="success">  
 <th class="text-center table-active">№</th>  
 <th class="text-center table-active">PI</th>  
 <th class="text-center table-active">W</th>  
 <th class="text-center table-active">D</th>  
 <th class="text-center table-active">L</th>  
 <th class="text-center table-active">Pts</th>  
 </thead>  
 <tbody>  
 <tr>  
 <td class="px-2 text-center" th:text="${club.tablePosition}"></td>  
 <td class="px-2 text-center" th:text="${club.matchesPlayed}"></td>  
 <td class="px-2 text-center" th:text="${club.seasonWins}"></td>  
 <td class="px-2 text-center" th:text="${club.seasonDraws}"></td>  
 <td class="px-2 text-center" th:text="${club.seasonLoses}"></td>  
 <td class="px-2 text-center" th:text="${club.points}"></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
  
</div>  
<br><br>  
<footer th:insert="~{blocks/footer :: footer}" class="container"></footer>  
</body>  
</html>

Темплейт login:

<!DOCTYPE html>  
<!--suppress InjectedReferences -->  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <link rel="stylesheet" th:href="@{/css/style.css}"/>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Epl-prediction</title>  
</head>  
<body>  
<header th:insert="~{blocks/header :: header}" class="p-3 bg-dark text-white"></header>  
<div class="container"><br>  
 <h4>Login</h4><br>  
 <div class="row">  
 <form action="/login" method="post" name="login" class="col-md-6">  
 <label for="username">Email</label>  
 <input type="email" name="username"><br><br>  
 <label for="password">Password</label>  
 <input type="password" name="password"><br><br>  
 <input type="hidden" name="\_csrf" value="${\_csrf.token}">  
 <input type="submit" value="Login" class="btn btn-outline-dark me-2">  
 <a href="/registration" class="link-dark">Register</a>  
 </form>  
 </div>  
</div>  
<br><br>  
<footer th:insert="~{blocks/footer :: footer}" class="container"></footer>  
</body>  
</html>

Темплейт match:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <!-- <link rel="stylesheet" type="text/css" href="\static\css\style.css">-->  
 <link rel="stylesheet" th:href="@{/css/style.css}">  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Epl-prediction</title>  
</head>  
<body>  
<header th:insert="~{blocks/header :: header}" class="p-3 bg-dark text-white"></header>  
<div class="container"><br>  
 <div class="row">  
 <div class="d-flex justify-content-between col-md-10 offset-md-1 bg-dark text-white border border-warning rounded p-3">  
 <img th:src="${match.homeTeam.logoLink}" class="matchLogo col-md-3">  
 <div class="col-md-4 justify-content-center flex-column d-flex text-center">  
 <h3 th:text="${match.time}"></h3>  
 <h4 th:text="${match.score}" class="border border-warning p-3 mx-5 rounded"></h4>  
 </div>  
 <img th:src="${match.awayTeam.logoLink}" class="matchLogo col-md-3">  
 </div>  
 </div>  
 <div class="row">  
 <div class="d-flex justify-content-between col-md-10 offset-md-1">  
 <h2 th:text="${match.homeTeam.name}"class="col-md-3 text-center"></h2>  
 <h2 th:text="${match.awayTeam.name}"class="col-md-3 offset-md-4 text-center"></h2>  
 </div>  
 </div><br>  
 <div>  
 <h1>Make a bet</h1>  
 <form action="/bet" method="post">  
 <label for="amount">Bet Amount:</label>  
 <input type="number" id="amount" name="amount" required><br><br>  
  
 <label for="team">Select outcome:</label>  
 <select id="team" name="team">  
 <option value="HOME\_WIN" th:text="${match.homeTeam.name}" ></option>  
 <option value="AWAY\_WIN" th:text="${match.awayTeam.name}"></option>  
 <option value="DRAW">Draw</option>  
 </select><br><br>  
  
 <input type="submit" value="Bet">  
 </form>  
 </div>  
  
</div>  
<br><br>  
<footer th:insert="~{blocks/footer :: footer}" class="container"></footer>  
</body>  
</html>

Темплейт profile:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <!-- <link rel="stylesheet" type="text/css" href="\static\css\style.css">-->  
 <link rel="stylesheet" th:href="@{/css/style.css}">  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Epl-prediction</title>  
</head>  
<body>  
<header th:insert="~{blocks/header :: header}" class="p-3 bg-dark text-white"></header>  
<div class="container"><br>  
 <div class="row"><br>  
 <form action="/logout" method="post" class="col-md-6">  
 <input type="hidden" name="\_csrf" value="${\_csrf.token}">  
 <input type="submit" value="Exit" class="btn btn-outline-dark me-2">  
 </form>  
 </div>  
 <div class="row" th:text="${user.points}">Points:</div>  
</div>  
<br><br>  
<footer th:insert="~{blocks/footer :: footer}" class="container"></footer>  
</body>  
</html>

Темплейт registration:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <!-- <link rel="stylesheet" type="text/css" href="\static\css\style.css">-->  
 <link rel="stylesheet" th:href="@{/css/style.css}">  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Epl-prediction</title>  
</head>  
<body>  
<header th:insert="~{blocks/header :: header}" class="p-3 bg-dark text-white"></header>  
<div class="container"><br>  
 <h4>Registration</h4><br>  
 <div class="row">  
 <form action="/registration" method="post" name="registration" class="col-md-6">  
 <label for="name">Name</label>  
 <input type="text" name="name"><br><br>  
 <label for="email">Email</label>  
 <input type="email" name="email"><br><br>  
 <label for="password">Password</label>  
 <input type="password" name="password"><br><br>  
 <input type="hidden" name="\_csrf" value="${\_csrf.token}">  
 <input type="submit" value="Register" class="btn btn-outline-dark me-2">  
 <a href="/login" class="link-dark">Login</a>  
 </form>  
 </div>  
</div>  
<br><br>  
<footer th:insert="~{blocks/footer :: footer}" class="container"></footer>  
</body>  
</html>

Темплейт schedule:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <!-- <link rel="stylesheet" type="text/css" href="\static\css\style.css">-->  
 <link rel="stylesheet" th:href="@{/css/style.css}"/>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Epl-prediction</title>  
</head>  
<body>  
<header th:insert="~{blocks/header :: header}" class="p-3 bg-dark text-white"></header>  
<div class="container"><br>  
 <h3>2022/2023 Season</h3><br>  
 <div class="row">  
 <table class="table table-bordered table-hover table-condensed col-md-6">  
 <thead class="success">  
 <th class="text-center table-active">Date</th>  
 <th class="text-center table-active">Hosts</th>  
 <th class="text-center table-active">Score</th>  
 <th class="text-center table-active">Guests</th>  
 <th class="text-center table-active">More Details</th>  
 </thead>  
 <tbody>  
 <div th:each="e : ${matches}">  
 <tr>  
 <td class="px-6" th:text="${e.getTime}"></td>  
 <td class="px-6"><a th:href="@{'/club/'+${e.homeTeam.id}}" th:text="${e.homeTeam.name}" class="link-dark link-underline-opacity-0"></a></td>  
<!-- <td class="px-6 text-center table-active" th:text="${e.score}"></td>-->  
 <td class="px-6 text-center table-active"><a th:href="@{'/match/'+${e.id}}" th:text="${e.score}" class="link-dark link-underline-opacity-0"></a></td>  
 <td class="px-6"><a th:href="@{'/club/'+${e.awayTeam.id}}" th:text="${e.awayTeam.name}" class="link-dark link-underline-opacity-0"></a></td>  
 <td class="px-6"><a th:href="@{'/match/'+${e.id}}" class="link-dark link-underline-opacity-0"> More Details </a> </td>  
 </tr>  
 </div>  
 </tbody>  
 </table>  
 </div>  
</div>  
<br><br>  
<footer th:insert="~{blocks/footer :: footer}" class="container"></footer>  
</body>  
</html>

Темплейт table:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <!-- <link rel="stylesheet" type="text/css" href="\static\css\style.css">-->  
 <link rel="stylesheet" th:href="@{/css/style.css}"/>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Epl-prediction</title>  
</head>  
<body>  
<header th:insert="~{blocks/header :: header}" class="p-3 bg-dark text-white"></header>  
<div class="container"><br>  
 <h3>2022/2023 Season</h3><br>  
 <div class="row">  
 <table class="table table-bordered table-hover table-condensed col-md-6">  
 <thead class="success">  
 <th class="text-center table-active">№</th>  
 <th class="text-center table-active">Team</th>  
 <th class="text-center table-active">PI</th>  
 <th class="text-center table-active">W</th>  
 <th class="text-center table-active">D</th>  
 <th class="text-center table-active">L</th>  
 <th class="text-center table-active">Pts</th>  
 </thead>  
 <tbody>  
 <tr th:each="e : ${clubs}">  
 <td class="px-2" th:text="${e.tablePosition}"></td>  
 <td class="px-2"><a th:href="@{'/club/'+${e.id}}" th:text="${e.getName}" class="link-dark link-underline-opacity-0"></a></td>  
 <td class="px-2 text-center" th:text="${e.matchesPlayed}"> </td>  
 <td class="px-2 text-center" th:text="${e.seasonWins}"></td>  
 <td class="px-2 text-center" th:text="${e.seasonDraws}"></td>  
 <td class="px-2 text-center" th:text="${e.seasonLoses}"></td>  
 <td class="px-2 text-center" th:text="${e.points}"></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
</div>  
<br><br>  
<footer th:insert="~{blocks/footer :: footer}" class="container"></footer>  
</body>  
</html>