



Uniwersytet Rzeszowski
Kolegium Nauk Przyrodniczych
Instytut Informatyki

Praca projektowa Bazy danych

Liga piłkarska

Prowadzący:

dr inż. Piotr Grochowalski

Autor:

Tomasz Knapik, Sebastian Kuzyk

nr albumu: 131452, 131461

Kierunek: Informatyka, grupa lab 1

Rzeszów 2025

Spis treści

1.	Opis założeń projektu.....	3
2.	Specyfikacja wymagań.....	5
3.	Opis struktury projektu	9
4.	Harmonogram realizacji projektu.....	31
5.	Prezentacja warstwy użytkowej projektu.....	32
6.	Podsumowanie	39
7.	Literatura.....	40

1. Opis założeń projektu

Cel projektu

Celem projektu jest opracowanie kompleksowej aplikacji webowej do zarządzania rozgrywkami piłkarskimi, która umożliwi efektywne śledzenie statystyk, wyników meczów oraz danych dotyczących drużyn, zawodników i trenerów. System zapewni użytkownikom dostęp do kluczowych informacji w czasie rzeczywistym, eliminując konieczność ręcznego monitorowania danych i zwiększając efektywność analizy sportowej.

Opis problemu

Głównym problemem, który zostanie rozwiązany, jest brak centralnego i zautomatyzowanego systemu do zarządzania ligą piłkarską. Aktualnie wiele organizacji piłkarskich korzysta z arkuszy kalkulacyjnych lub rozproszonych systemów, co prowadzi do problemów związanych z błędami w danych, trudnościami w aktualizacji informacji oraz brakiem dostępu do kluczowych statystyk w czasie rzeczywistym.

Problem jest istotny, ponieważ nieefektywne zarządzanie danymi sportowymi wpływa negatywnie na analizę rozgrywek, co może prowadzić do błędnych decyzji taktycznych, organizacyjnych i finansowych. Dowody na istnienie problemu obejmują trudności klubów i federacji piłkarskich w gromadzeniu i aktualizowaniu danych, a także niską dostępność dokładnych statystyk meczowych dla kibiców oraz analityków sportowych.

Proponowane rozwiązanie

Aby rozwiązać ten problem, konieczne jest opracowanie zintegrowanego systemu do zarządzania rozgrywkami piłkarskimi, który umożliwi:

- Automatyczne przetwarzanie i aktualizację danych meczowych w bazie danych,
- Dynamiczne generowanie statystyk zawodników i drużyn,
- Dostęp do wyników w czasie rzeczywistym,
- Zarządzanie danymi drużyn, zawodników i trenerów w jednym miejscu,
- Eliminację błędów wynikających z ręcznego przetwarzania danych.

System będzie wykorzystywał nowoczesne technologie, w tym Spring Boot, REST API oraz PostgreSQL, zapewniając wysoką wydajność, skalowalność oraz bezpieczeństwo przechowywanych danych.

Kroki realizacji projektu:

- Analiza wymagań i projektowanie systemu.
 - Określenie kluczowych informacji aplikacji.
 - Zaprojektowanie struktury bazy danych oraz API.
 - Opracowanie modeli danych dla zawodników, drużyn, trenerów i meczów.
- Opracowanie prototypu aplikacji.
 - Stworzenie podstawowej wersji użytkownika.
 - Wdrożenie pierwszych funkcjonalności backendu o bazy danych.
 - Testy działania podstawowych operacji CRUD.
- Implementacja kluczowych funkcji systemu.
 - Zarządzanie danymi (dodawanie, usuwanie, edycja).
 - Rejestrowanie wyników meczów oraz generowanie statystyk.
 - Obsługa rankingów drużyn.
 - Obsługa rankingów najlepszych strzelców, asystentów i bramkarzy.

Efektem końcowym projektu będzie w pełni funkcjonalna aplikacja "Liga Piłkarska", która umożliwi kompleksowe zarządzanie rozgrywkami piłkarskimi, znacząco zwiększając efektywność analizy sportowej oraz dostępność statystyk dla użytkowników. Dzięki zastosowaniu nowoczesnych technologii system zapewni szybki dostęp do aktualnych wyników, rankingów oraz kluczowych informacji o zawodnikach i drużynach, co wpłynie na lepszą organizację rozgrywek i zwiększy satysfakcję użytkowników.

2. Specyfikacja wymagań

Niniejsza specyfikacja zawiera szczegółowe wymagania funkcjonalne i нефункционалне, które są fundamentalne dla projektu Liga Piłkarska.

Wymagania funkcjonalne

Poniżej znajduje się lista kluczowych funkcji, które powinny być spełnione przez system. Opisują one wszystkie dostępne operacje dla użytkowników aplikacji.

- Zarządzanie drużynami:
 - System umożliwia administratorowi dodanie nowej drużyny, podając jej nazwę, miasto oraz przypisany stadion.
 - Administrator może edytować dane drużyny, zmieniając jej nazwę, miasto lub stadion.
 - Administrator może usunąć drużynę.
 - System automatycznie aktualizuje statystyki drużyn (punkty, mecze rozegrane, zwycięstwa, remisy, porażki, bilans bramek) na podstawie wyników meczów.
 - Użytkownik może przeglądać listę drużyn oraz ich szczegółowe informacje.
- Zarządzanie zawodnikami:
 - System umożliwia administratorowi dodanie nowego zawodnika, podając jego imię, nazwisko, drużynę i pozycję.
 - Administrator może edytować dane zawodnika, zmieniając jego drużynę lub pozycję oraz usunąć go.
 - Użytkownik może przeglądać listę zawodników oraz ich szczegółowe dane.
- Zarządzanie statystykami zawodników:
 - System przechowuje dane statystyczne zawodników: liczba bramek, asyst, żółtych i czerwonych kartek, czyste konta.
 - Administrator może ręcznie edytować statystyki zawodników.
 - Użytkownik może przeglądać rankingi zawodników według liczby bramek, asyst, kartek oraz czystych kont.
 - Użytkownik może filtrować statystyki zawodników według drużyny.

- Zarządzanie trenerami:
 - Administrator może dodać nowego trenera, podając jego imię, nazwisko oraz przypisaną drużynę.
 - Administrator może edytować dane trenera, zmieniając przypisaną drużynę, imię lub nazwisko.
 - Administrator może usunąć trenera.
- Zarządzanie sędziami:
 - Administrator może dodać nowego sędziego, podając jego imię i nazwisko.
 - Administrator może edytować dane sędziego.
 - Administrator może usunąć sędziego.
 - Użytkownik może przeglądać listę sędziów danych meczy.
- Zarządzanie stadionami:
 - Administrator może dodać nowy stadion, podając jego nazwę.
 - Administrator może edytować dane stadionu.
 - Administrator może usunąć stadion.
 - Użytkownik może przeglądać listę stadionów.
- Zarządzanie meczami:
 - Administrator może dodać nowy mecz, podając drużyny, datę meczu, wynik, sędziego oraz stadion.
 - Administrator może edytować dane meczu przed jego rozpoczęciem.
 - Administrator może zaktualizować wynik meczu po jego zakończeniu.
 - Administrator może usunąć mecz.
 - System automatycznie aktualizuje statystyki drużyn i zawodników po wprowadzeniu wyniku meczu.
 - Użytkownik może przeglądać historię meczów.
 - Użytkownik może filtrować mecze według daty.
- Interfejs użytkownika
 - System zapewnia dynamiczne generowanie tabel wyników i statystyk w interaktywnej formie.
 - System umożliwia przeglądanie szczegółowych raportów dotyczących meczów i zawodników.
 - System umożliwia intuicyjne wyszukiwanie i filtrowanie danych.

Wymagania niefunkcjonalne

Poniżej znajdują się założenia dotyczące jakości i wydajności systemu. Obejmują one aspekty, takie jak bezpieczeństwo, wydajność, dostępność oraz inne właściwości, które wpływają na doświadczenia użytkowników.

- **Użyteczność:**
 - Aplikacja powinna być intuicyjna i łatwa w obsłudze, zarówno dla administratorów systemu, jak i użytkowników końcowych.
 - Interfejs użytkownika musi zapewniać czytelną i estetyczną prezentację danych, takich jak informacje o drużynach, zawodnikach, meczach i statystykach.
 - Wszystkie kluczowe funkcje powinny być łatwo dostępne i uporządkowane w sposób logiczny, umożliwiając szybkie przeglądanie i analizowanie wyników.
 - System powinien obsługiwać dynamiczne sortowanie i filtrowanie wyników w tabelach statystycznych.
- **Wydajność:**
 - Aplikacja powinna działać płynnie i bez zbędnych opóźnień, zapewniając użytkownikom szybki dostęp do kluczowych informacji.
 - Czas odpowiedzi na zapytania użytkowników, takie jak pobieranie listy drużyn, zawodników czy meczów, nie może przekraczać 2 sekund.
 - System powinien być w stanie obsłużyć równoczesne zapytania wielu użytkowników, zapewniając stabilność działania.
 - Optymalizacja zapytań do bazy danych powinna minimalizować czas wykonywania operacji na dużych zbiorach danych.
- **Dostępność:**
 - System powinien być dostępny dla użytkowników w dowolnym miejscu i czasie, umożliwiając śledzenie statystyk meczowych oraz wyników na różnych urządzeniach.
 - Aplikacja powinna działać poprawnie na różnych systemach operacyjnych oraz przeglądarkach internetowych, takich jak Google Chrome, Mozilla Firefox, Microsoft Edge i Safari.
 - W przypadku awarii systemu, administratorzy powinni mieć możliwość szybkiego przywrócenia działania aplikacji na podstawie zapisanych kopii zapasowych.
- **Skalowalność:**
 - System musi umożliwiać łatwe dodawanie nowych funkcji, takich jak rozszerzone statystyki meczowe, prognozy wyników czy integracja z zewnętrznymi bazami danych sportowych.
 - Architektura aplikacji powinna pozwalać na obsługę rosnącej liczby użytkowników, bez wpływu na wydajność i stabilność działania.
 - Możliwość rozbudowy systemu o dodatkowe moduły analizy statystycznej, raportowania oraz automatycznego generowania podsumowań sezonowych.

- Środowisko:
 - System wykorzystuje nowoczesne technologie backendowe, takie jak Spring Boot i PostgreSQL, zapewniające stabilność i wydajność działania.
 - API aplikacji jest zgodne ze standardem REST, a dane przesyłane są w formacie JSON.
 - Kod źródłowy aplikacji jest zarządzany za pomocą systemu kontroli wersji Git, umożliwiające efektywne śledzenie zmian i wdrażanie poprawek.
 - System powinien być regularnie aktualizowany, zapewniając kompatybilność z najnowszymi wersjami frameworków i bibliotek.
- Utrzymanie:
 - Aplikacja powinna być łatwa w utrzymaniu, dzięki zastosowaniu czytelnej struktury kodu oraz zgodności z dobrymi praktykami programistycznymi.
 - Dokumentacja techniczna powinna zawierać szczegółowy opis API, struktury bazy danych oraz kluczowych funkcji systemu.
 - Aktualizacje i poprawki powinny być wdrażane bez przerywania działania aplikacji.
 - System musi umożliwiać monitorowanie błędów i zdarzeń, co pozwala na szybkie diagnozowanie problemów i ich eliminację.
 - Podsumowanie
 - Aplikacja Liga Piłkarska została zaprojektowana z myślą o zapewnieniu wydajnego, intuicyjnego i skalowalnego systemu do zarządzania danymi piłkarskimi.
- Podsumowanie:

Aplikacja Liga Piłkarska została zaprojektowana z myślą o łatwym dostępie do wyników meczów, statystyk drużyn oraz osiągnięć indywidualnych zawodników. System umożliwia wygodne przeglądanie danych, filtrowanie informacji i analizowanie rozgrywek w przejrzysty i intuicyjny sposób.

Dzięki nowoczesnym technologiom, takim jak Spring Boot i PostgreSQL, aplikacja działa sprawnie, oferując wysoką wydajność i możliwość łatwego rozwoju. System pozwala na dynamiczne aktualizowanie wyników, generowanie rankingów zawodników oraz zarządzanie drużynami, sędziami i stadionami.

Dzięki swojej elastyczności i przejrzystej architekturze, Liga Piłkarska może być wykorzystywana zarówno przez kibiców, jak i osoby zarządzające rozgrywkami. To kompleksowe narzędzie do monitorowania wyników i analizy statystyk, które ułatwia śledzenie rywalizacji piłkarskiej w sposób szybki i efektywny.

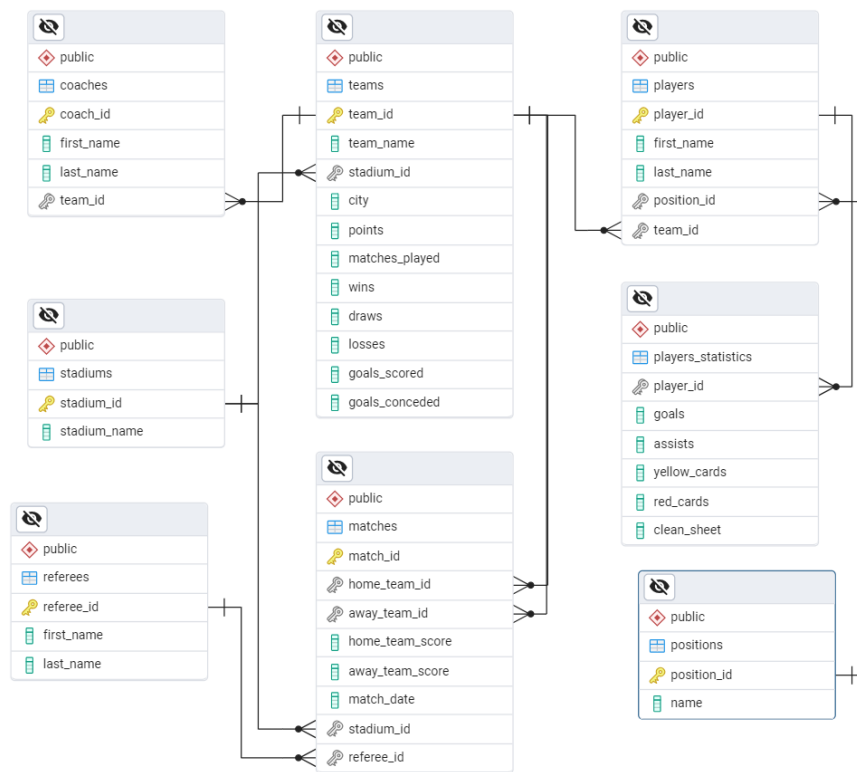
3. Opis struktury projektu

Poniżej przedstawiono informacje dotyczące struktury projektu. Obejmują one używane środowisko programistyczne, narzędzia, minimalne wymagania sprzętowe, hierarchie klas oraz strukturę przechowywanych danych.

- Środowisko programistyczne:
 - Oprogramowanie: IntelliJ IDEA 2024.3.2
 - Język programowania: Java 23
 - Implementacja Javy: OpenJDK 23.0.1
 - Framework backendowy: Spring Boot 3.4.1
 - Baza danych: PostgreSQL 17
 - Architektura aplikacji: REST API
- Narzędzia i biblioteki:
 - Backend:
 - Spring Boot 3.4.1 – framework do budowy aplikacji webowych
 - Spring Boot Starter Web – obsługa API REST
 - Spring Boot Starter Data JPA – integracja z bazą danych za pomocą Hibernate
 - Spring Boot Starter JDBC – obsługa połączeń do bazy danych
 - PostgreSQL JDBC Driver (42.7.4) – sterownik do komunikacji z bazą danych PostgreSQL
 - Zarządzanie zależnościami:
 - Maven – system zarządzania pakietami i budowania aplikacji
 - Kontrola wersji.
 - Git – repozytorium kodu źródłowego
 - Maven Plugins.
 - Maven Compiler Plugin 3.13.0 – kompilacja kodu źródłowego
 - Maven Clean Plugin 3.4.0 – czyszczenie katalogu build
 - Maven Deploy Plugin 3.1.3 – wdrażanie aplikacji
 - Maven Install Plugin 3.1.3 – instalowanie pakietów lokalnie
 - Maven Resources Plugin 3.3.1 – zarządzanie zasobami aplikacji
- Rekomendowane wymagania sprzętowe:
 - Procesor: Intel Core i5 / Ryzen 5 lub równoważny
 - Pamięć RAM: Minimum 4 GB (zalecane 8 GB dla płynnej pracy)
 - Dysk twardy: Minimum 10 GB wolnej przestrzeni na pliki projektu i bazę danych
 - System operacyjny: Windows 11 / macOS 14 / Ubuntu 24.04
 - Baza danych: PostgreSQL 17

Struktura bazy danych:

Na Rys. 1 przedstawiono Diagram ERD bazy danych wymaganej do poprawnego działania aplikacji.



Rys. 1. Diagram ERD bazy danych

Baza danych składa się z ośmiu tabel:

- Tabela „teams”

Przechowuje informacje o drużynach uczestniczących w rozgrywkach.

Kolumna	Typ danych	Opis
team_id	INT Serial(PK)	Unikalny identyfikator drużyny
team_name	VARCHAR	Nazwa drużyny
stadium_id	INT (FK)	Powiązanie ze stadionem
city	VARCHAR	Miasto drużyny
points	INT	Liczba punktów w sezonie
matches_played	INT	Liczba rozegranych meczów
wins	INT	Liczba wygranych meczów
draws	INT	Liczba remisów
losses	INT	Liczba przegranych meczów
goals_scored	INT	Liczba zdobytych bramek
Goals_conceded	INT	Liczba straconych bramek

Relacje:

„stadium_id” jest kluczem obcym (FK) wskazującym na tabelę „stadiums”.

- Tabela „players”:

Przechowuje informacje o zawodnikach grających w drużynach.

Kolumna	Typ danych	Opis
player_id	INT Serial (PK)	Unikalny identyfikator zawodnika
first_name	VARCHAR	Imię zawodnika
last_name	VARCHAR	Nazwisko zawodnika
position_id	INT (FK)	Pozycja zawodnika na boisku
team_id	INT (FK)	Drużyna do której należy zawodnik

Relacje:

„team_id” jest kluczem obcym (FK) wskazującym na tabelę „teams”.

„position_id” jest kluczem obcym (FK) wskazującym na tabelę „positions”.

- Tabela „positions”

Przechowuje nazwy pozycji, na których mogą grać zawodnicy.

Kolumna	Typ danych	Opis
position_id	INT Serial (PK)	Unikalny identyfikator pozycji
name	VARCHAR	Nazwa pozycji

Relacje:

„position_id” jest używane w tabeli „players” jako klucz obcy (FK).

- Tabela „matches”:

Przechowuje informacje o rozegranych meczach

Kolumna	Typ danych	Opis
match_id	INT Serial (PK)	Unikalny identyfikator meczu
home_team_id	INT (FK)	Identyfikator drużyny gospodarza
away_team_id	INT (FK)	Identyfikator drużyny gości
home_team_score	INT	Liczba bramek gospodarzy
away_team_score	INT	Liczba bramek gości
match_date	DATE	Data meczu
stadium_id	INT (FK)	Stadion, na którym odbył się mecz
referee_id	INT (FK)	Sędzia prowadzący mecz

Relacje:

„home_team_id” i „away_team_id” są kluczami obcymi (FK) wskazującymi na tabelę „teams”.

„stadium_id” jest kluczem obcym (FK) wskazującym na tabelę „stadiums”.

„referee_id” jest kluczem obcym (FK) wskazującym na tabelę „referees”.

- Tabela „player_statistics”:

Przechowuje szczegółowe statystyki indywidualnych zawodników.

Kolumna	Typ danych	Opis
player_id	INT (FK)	Identyfikator zawodnika
goals	INT	Liczba zdobytych bramek
assists	INT	Liczba asyst
yellow_cards	INT	Liczba żółtych kartek
red_cards	INT	Liczba czerwonych kartek
clean_sheet	INT	Liczba czystych kont bramkarza

Relacje:

„player_id” jest kluczem głównym i jednocześnie kluczem obcym (FK), wskazującym na tabelę „players”.

- Tabela „coaches”:

Przechowuje informacje o trenerach drużyn.

Kolumna	Typ danych	Opis
coach_id	INT Serial (PK)	Unikalny identyfikator trenera
first_name	VARCHAR	Imię trenera
last_name	VARCHAR	Nazwisko trenera
team_id	INT (FK)	Identyfikator drużyny, którą prowadzi

Relacje:

„team_id” jest kluczem obcym (FK) wskazującym na tabelę „teams”.

- Tabela „referees”:

Przechowuje dane o sędziach prowadzących mecze.

Kolumna	Typ danych	Opis
referee_id	INT Serial (PK)	Unikalny identyfikator sędziego
first_name	VARCHAR	Imię sędziego
last_name	VARCHAR	Nazwisko sędziego

Relacje:

„referee_id” jest używane w tabeli „matches” jako klucz obcy (FK).

- Tabela „stadiums”:

Przechowuje informacje o stadionach, na których odbywają się mecze.

Kolumna	Typ danych	Opis
stadium_id	INT Serial (PK)	Unikalny identyfikator stadionu
stadium_name	VARCHAR	Nazwa stadionu

Relacje:

„stadium_id” jest używane jako klucz obcy (FK) w tabelach „matches” oraz „teams”.

- Relacje pomiędzy tabelami:
 - Jeden trener „coaches” → jedna drużyna „teams”
 - Jedna drużyna „teams” → wielu zawodników „players”
 - Jedna pozycja „positions” → wielu zawodników „players”
 - Jeden mecz „matches” → dwie drużyny „teams”
 - Jeden stadion „stadiums” → wiele meczów „matches”
 - Jeden sędzia „referees” → wiele meczów „matches”
 - Jeden zawodnik „players” → jedna statystyka „players_statistics”

Do poprawnego działania projektu, konieczne jest połączenie z bazą danych. Plik .sql zawierający strukturę bazy danych znajduje się w katalogu głównym repozytorium pod nazwą LigaPilkarska.sql. Przed rozpoczęciem korzystania z aplikacji, wymagane jest zaimportowanie tego pliku do systemu zarządzania bazami danych.

Zarządzanie danymi: Procedury

- Zawodnicy
 - `add_player(IN first_name_param CHARACTER, IN last_name_param CHARACTER, IN name_param CHARACTER, IN team_name_param CHARACTER VARYING)`
Dodaje nowego zawodnika do bazy danych, przypisując go do wskazanej drużyny.
 - `delete_player(IN player_id_param INTEGER)`
Usuwa istniejącego zawodnika z bazy danych na podstawie podanego identyfikatora.
 - `edit_player(IN p_player_id BIGINT, IN p_first_name CHARACTER VARYING, IN p_last_name CHARACTER VARYING, IN p_position_name CHARACTER VARYING, IN p_team_name CHARACTER VARYING)`
Aktualizuje dane zawodnika (imię, nazwisko, pozycję, drużynę) na podstawie podanego identyfikatora.
- Statystyki zawodników
 - `add_players_statistic(IN p_player_id INTEGER, IN p_goals INTEGER, IN p_assists INTEGER, IN p_yellow_cards INTEGER, IN p_red_cards INTEGER, IN p_clean_sheet INTEGER)`
Dodaje nowe statystyki (bramki, asysty, żółte/czerwone kartki, czyste konta) dla wybranego zawodnika.
 - `delete_players_statistic(IN p_player_statistic_id INTEGER)`
Usuwa rekord statystyk danego zawodnika na podstawie jego identyfikatora statystyk.
 - `edit_player_statistics(IN player_id INTEGER, IN new_goals INTEGER, IN new_assists INTEGER, IN new_yellow_cards INTEGER, IN new_red_cards INTEGER, IN new_clean_sheets INTEGER)`
Edytuje istniejące statystyki zawodnika, np. liczbę bramek, asyst czy kartek.

- Trenerzy
 - `add_coach(IN p_first_name TEXT, IN p_last_name TEXT, IN p_team_id INTEGER)`
Dodaje nowego trenera do bazy danych, przypisując go do drużyny na podstawie jej identyfikatora.
 - `add_coach_by_team_name(IN p_first_name CHARACTER VARYING, IN p_last_name CHARACTER VARYING, IN p_team_name CHARACTER VARYING)`
Dodaje nowego trenera do bazy danych, przypisując go do drużyny na podstawie nazwy drużyny (zamiast jej identyfikatora).
 - `delete_coach(IN p_coach_id INTEGER)`
Usuwa trenera z bazy danych na podstawie podanego identyfikatora.
 - `edit_coach(IN p_coach_id INTEGER, IN p_first_name TEXT, IN p_last_name TEXT, IN p_team_id INTEGER)`
Aktualizuje dane istniejącego trenera (imię, nazwisko, przypisana drużyna) w bazie danych.
- Mecze
 - `add_match(IN p_home_team_name CHARACTER VARYING, IN p_away_team_name CHARACTER VARYING, IN p_home_team_score INTEGER, IN p_away_team_score INTEGER, IN p_match_date DATE, IN p_stadium_name CHARACTER VARYING, IN p_referee_name CHARACTER VARYING)`
Dodaje nowy mecz do bazy, określając drużyny, wynik, datę, stadion i sędziego.
 - `delete_match(IN p_match_id INTEGER)`
Usuwa mecz z bazy danych na podstawie identyfikatora meczu.
 - `edit_match_by_team_names(IN p_match_id INTEGER, IN p_home_team_name TEXT, IN p_away_team_name TEXT, IN p_home_team_score INTEGER, IN p_away_team_score INTEGER, IN p_match_date DATE, IN p_stadium_name TEXT, IN p_referee_name TEXT)`
Edytuje dane istniejącego meczu (nazwy drużyn, wyniki, datę, stadion, sędziego) na podstawie identyfikatora meczu.
- Sędziowie
 - `add_referee(IN first_name_param CHARACTER VARYING, IN last_name_param CHARACTER VARYING)`
Dodaje nowego sędziego do bazy danych.
 - `delete_referee(IN referee_id_param INTEGER)`
Usuwa sędziego z bazy danych na podstawie podanego identyfikatora.
 - `edit_referee(IN referee_id_param INTEGER, IN first_name_param CHARACTER VARYING, IN last_name_param CHARACTER VARYING)`
Aktualizuje dane osobowe sędziego (imię, nazwisko) w bazie danych.

- **Stadiony**
 - `add_stadium(IN stadium_name_param CHARACTER VARYING)`
Dodaje nowy stadion do bazy danych.
 - `delete_stadium(IN stadium_id_param INTEGER)`
Usuwa stadion z bazy danych na podstawie identyfikatora stadionu.
 - `edit_stadium(IN stadium_id_param INTEGER, IN stadium_name_param CHARACTER VARYING)`
Aktualizuje dane istniejącego stadionu (np. jego nazwę) w bazie danych.
- **Drużyny**
 - `add_team(IN team_name_param CHARACTER, IN stadium_name_param CHARACTER VARYING, IN city_param CHARACTER VARYING)`
Dodaje nową drużynę, przypisując jej nazwę, stadion oraz miasto.
 - `delete_team(IN team_id_param BIGINT)`
Usuwa istniejącą drużynę z bazy danych na podstawie identyfikatora drużyny.
 - `edit_team(IN p_team_id BIGINT, IN p_team_name CHARACTER VARYING, IN p_stadium_name CHARACTER VARYING, IN p_city CHARACTER VARYING)`
Edytuje dane wybranej drużyny (nazwę, stadion, miasto) w bazie danych.
- **Pozycje**
 - `add_position(IN p_name CHARACTER)`
Dodaje nową pozycję zawodnika (np. „obrońca”, „pomocnik”) do bazy danych.
 - `delete_position(IN p_position_id INTEGER)`
Usuwa istniejącą pozycję z bazy danych na podstawie identyfikatora.
 - `update_position(IN position_id INTEGER, IN name CHARACTER)`
Aktualizuje nazwę istniejącej pozycji zawodnika w bazie danych.

Zarządzanie danymi: Funkcje

- `get_all_matches_by_team(team_id_param BIGINT)`
Zwraca listę wszystkich meczów wybranej drużyny (m.in. data, wynik, rywal) na podstawie jej identyfikatora.
- `get_all_referees()`
Zwraca listę wszystkich sędziów dostępnych w bazie danych (np. imię, nazwisko).
- `get_all_stadiums()`
Zwraca listę wszystkich stadionów (np. nazwa, lokalizacja) zapisanych w bazie danych.
- `get_all_teams()`
Zwraca listę wszystkich drużyn (m.in. nazwa, miasto, stadion).
- `get_coaches_info()`
Zwraca informacje o wszystkich trenerach (np. imię, nazwisko, przypisana drużyna).
- `get_matches_by_date(match_day DATE)`
Zwraca zestaw meczów rozgrywanych w podanym dniu.
- `get_matches_info()`
Zwraca szczegółowe dane o wszystkich meczach (m.in. drużyny, wynik, data, sędzia).
- `get_player_statistics()`
Zwraca statystyki wszystkich zawodników (np. bramki, asysty, kartki).
- `get_players_info()`
Zwraca podstawowe informacje o wszystkich zawodnikach (m.in. imię, nazwisko, drużyna, pozycja).
- `get_recent_matches_by_team(team_id_param BIGINT)`
Zwraca listę ostatnich meczów wybranej drużyny.
- `get_team_details(team_id_param BIGINT)`
Zwraca szczegółowe informacje o drużynie (m.in. nazwa, miasto, stadion, trener).
- `get_team_id_by_name(p_team_name CHARACTER VARYING)`
Umożliwia pobranie identyfikatora drużyny na podstawie jej nazwy.
- `get_team_players(team_id_param BIGINT)`
Zwraca listę zawodników przypisanych do wybranej drużyny.
- `get_top_assisters()`
Zwraca zestawienie zawodników z największą liczbą asyst.
- `get_top_goalkeepers()`
Zwraca zestawienie najlepszych bramkarzy (np. wg liczby czystych kont).
- `get_top_players_by_team(team_id_param BIGINT)`
Zwraca najlepszych zawodników w drużynie o podanym identyfikatorze (np. według goli, asyst).
- `get_top_scorers()`
Zwraca listę zawodników z największą liczbą strzelonych bramek.
- `getallteamsranking()`
Zwraca ogólny ranking wszystkich drużyn (np. według liczby punktów).

- `getaveragepointspermatch()`
Zwraca średnią liczbę punktów na mecz dla każdej drużyny.
- `getbestattackingteams()`
Zwraca drużyny z najsukuteczniejszym atakiem (najwięcej strzelonych bramek).
- `getbestdefensiveteams()`
Zwraca drużyny z najlepszą obroną (najmniej straconych bramek).
- `gethomeawayperformance()`
Zwraca porównanie wyników drużyn w meczach u siebie i na wyjeździe (zwycięstwa, porażki, bilans bramek).
- `getmostcommonmatchresults()`
Zwraca najczęściej występujące wyniki meczów (np. 0:0, 1:1).
- `getteamranking(p_team_id BIGINT)`
Zwraca aktualną pozycję wskazanej drużyny w tabeli ligowej.

Zarządzanie danymi: Triggery i ich funkcje

W projekcie zastosowano kilka wyzwalaczy (triggers), które w sposób automatyczny aktualizują powiązane tabele, dzięki czemu baza danych zawsze odzwierciedla bieżący stan statystyk drużyn oraz zawodników. Poniżej znajduje się krótki opis każdego z wyzwalaczy i wywoływanych przez nie funkcji:

- `trigger_add_player_statistics`
 - Funkcja: `add_player_statistics()`
 - Zadanie: Po dodaniu nowego zawodnika do tabeli `players`, automatycznie tworzy dla niego wpis w tabeli `players_statistics` z początkowymi wartościami (np. 0 bramek, 0 asyst).
- `trigger_delete_player_statistics`
 - Funkcja: `delete_player_statistics()`
 - Zadanie: Po usunięciu zawodnika z tabeli `players`, usuwa również powiązany rekord ze statystykami zawodnika z tabeli `players_statistics`.
- `trigger_update_matches_played`
 - Funkcja: `update_matches_played()`
 - Zadanie: Po wstawieniu nowego meczu do tabeli `matches` (INSERT) zwiększa licznik rozegranych spotkań (`matches_played`) drużyn gospodarz i gości. Natomiast przy usunięciu meczu (DELETE) zmniejsza ten licznik odpowiednio.
- `trigger_update_team_goals`
 - Funkcja: `update_team_goals()`
 - Zadanie: Po dodaniu (INSERT) lub usunięciu (DELETE) meczu aktualizuje liczbę strzelonych (`goals_scored`) oraz straconych (`goals_conceded`) bramek drużynom biorącym udział w meczu. Przykładowo, gdy gospodarze wygrywają 2:1, zwiększa się `goals_scored` gospodarzy o 2, a gości o 1, i odpowiednio `goals_conceded` każdej drużynie.
- `trigger_update_team_points`
 - Funkcja: `update_team_points()`
 - Zadanie: Po wstawieniu nowego meczu przydziela odpowiednią liczbę punktów (3 za zwycięstwo, 1 za remis, 0 za porażkę) drużynom gospodarzy i gości. W przypadku usuwania meczu wycofuje te punkty. Umożliwia to automatyczne prowadzenie klasyfikacji punktowej w tabeli.
- `trigger_update_team_results`
 - Funkcja: `update_team_results()`
 - Zadanie: Podobnie jak w przypadku punktów, ten wyzwalacz zajmuje się kolumnami `wins`, `draws` i `losses`. Po dodaniu meczu zwiększa odpowiednią statystykę (wygrane, remisy lub porażki) obu drużyn, a po usunięciu meczu cofa te wartości do stanu przed rozegraniem spotkania.

Analiza i kroki niezbędne do transponowania bazy relacyjnej na model nierelacyjny

- **Wprowadzenie**
Niniejsza sekcja zawiera analizę i opis działań niezbędnych do przeniesienia relacyjnej bazy danych zdefiniowanej w ramach projektu na model nie relacyjny (NoSQL). W wyniku tej analizy przedstawiono kroki, które należy podjąć, aby dokonać transponowania danych.
- **Wybór modelu nie relacyjnego**
 - Wybrany model nie relacyjny: Model dokumentowy (np. MongoDB)
- **Analiza struktury danych**
 - Struktura danych w relacyjnej bazie danych zostanie przeniesiona do jednego dokumentu JSON, który będzie zawierał wszystkie powiązane dane.
- **Mapowanie tabel na dokument JSON**
 - W modelu dokumentowym cała baza danych zostanie zamieniona na jeden dokument JSON, zawierający wszystkie niezbędne informacje.
- **Migracja danych**

Przeprowadzenie migracji danych z relacyjnej bazy danych do modelu nie relacyjnego:

- Ekstrakcja danych z relacyjnej bazy danych.
- Transformacja danych do formatu JSON.
- Załadowanie dokumentu JSON do bazy NoSQL (np. MongoDB).

Proces migracji do MongoDB Compass

- Wygenerowanie pliku JSON z PostgreSQL

Aby wyeksportować dane w odpowiednim formacie JSON, należy wykonać zapytanie SQL:

```
SELECT jsonb_pretty(
  jsonb_build_object(
    'teams', jsonb_agg(
      jsonb_build_object(
        'team_id', t.team_id,
        'team_name', t.team_name,
        'stadium', jsonb_build_object(
          'stadium_name', s.stadium_name,
          'city', trim(t.city)
        ),
        'points', t.points,
        'matches_played', t.matches_played,
        'wins', t.wins,
        'draws', t.draws,
        'losses', t.losses,
        'goals_scored', t.goals_scored,
        'goals_conceded', t.goals_conceded,
        'coach', jsonb_build_object(
          'first_name', c.first_name,
          'last_name', c.last_name
        ),
        'players', (
          SELECT jsonb_agg(
            jsonb_build_object(
              'player_id', p.player_id,
              'first_name', trim(p.first_name),
              'last_name', trim(p.last_name),
              'position', trim(pos.name),
              'statistics', jsonb_build_object(
                'goals', ps.goals,
                'assists', ps.assists,
                'yellow_cards', ps.yellow_cards,
                'red_cards', ps.red_cards,
                'clean_sheets', ps.clean_sheet
              )
            )
          ) FROM players p
          LEFT JOIN positions pos ON p.position_id = pos.position_id
          LEFT JOIN players_statistics ps ON p.player_id = ps.player_id
          WHERE p.team_id = t.team_id
        ),
        'matches', (
          SELECT jsonb_agg(
```

```

        jsonb_build_object(
            'match_id', m.match_id,
            'home_team', home_team.team_name,
            'away_team', away_team.team_name,
            'home_team_score', m.home_team_score,
            'away_team_score', m.away_team_score,
            'match_date', m.match_date,
            'stadium', st.stadium_name,
            'referee', jsonb_build_object(
                'first_name', r.first_name,
                'last_name', r.last_name
            )
        )
    ) FROM matches m
    LEFT JOIN teams home_team ON m.home_team_id = home_team.team_id
    LEFT JOIN teams away_team ON m.away_team_id = away_team.team_id
    LEFT JOIN stadiums st ON m.stadium_id = st.stadium_id
    LEFT JOIN referees r ON m.referee_id = r.referee_id
    WHERE m.home_team_id = t.team_id OR m.away_team_id = t.team_id
    )
    )
    )
    ) FROM teams t
    LEFT JOIN coaches c ON t.team_id = c.team_id
    LEFT JOIN stadiums s ON t.stadium_id = s.stadium_id;

```

Następnie wynik zapytania należy zachować jako plik JSON (teams.json).

- Import pliku JSON do MongoDB Compass
 - Otworzyć MongoDB Compass.
 - Połączyć się z instancją MongoDB.
 - Utworzyć nową bazę danych (LigaPiłkarska).
 - Utworzyć kolekcję.
 - Wybrać opcję "Import JSON" i wskazać plik teams.json.
 - Po imporcie sprawdzić w MongoDB Compass, czy dokument został poprawnie zapisany.

Podsumowanie

- Wszystkie dane są przechowywane w jednej kolekcji.
- Jeden dokument przechowuje całą bazę danych w formie hierarchicznej.
- Proces migracji wymaga tylko eksportu JSON i zaimportowania do MongoDB Compass.
- Po imporcie cała baza jest dostępna w formie jednego dużego dokumentu JSON.

Backend

Backend aplikacji "Liga Piłkarska" został zbudowany z wykorzystaniem Java Spring Boot oraz Spring Data JPA, co zapewnia wydajność, czytelność kodu i wygodne zarządzanie danymi w bazie danych. Poniżej znajduje się szczegółowy opis każdej użytej technologii i powody jej wyboru.

Technologie i frameworki

- **Java Spring Boot**

Opis:

- Spring Boot to framework do tworzenia aplikacji webowych w Javy, który upraszcza konfigurację i automatyzuje wiele aspektów programowania.

Powody wyboru:

- Uproszczona konfiguracja aplikacji w porównaniu do tradycyjnego Spring Framework.
- Wbudowany serwer, co eliminuje potrzebę konfiguracji zewnętrznych serwerów aplikacyjnych.
- Obsługa REST API w sposób natywny, co ułatwia tworzenie interfejsów dla frontendowych aplikacji.

- **Spring Data JPA**

Opis:

- Spring Data JPA to moduł Springa, który umożliwia łatwe zarządzanie bazą danych poprzez ORM (Object-Relational Mapping) bez konieczności pisania dużej ilości kodu SQL.

Powody wyboru:

- Automatyczna obsługa operacji CRUD (Create, Read, Update, Delete).
- Integracja z Hibernate, co pozwala na użycie mechanizmów mapowania obiektowo-relacyjnego.
- Redukcja ilości kodu potrzebnego do obsługi baz danych.

- **Spring Boot REST**

Opis:

- Moduł Spring Boot REST ułatwia tworzenie API RESTful, zapewniając mechanizmy do obsługi żądań HTTP (GET, POST, PUT, DELETE).

Powody wyboru:

- Wbudowana obsługa JSON-a poprzez Jackson.
- Integracja z innymi modułami Spring Boot.
- Obsługa kontrolerów REST poprzez adnotacje (`@RestController`, `@RequestMapping`)

- Spring Boot JDBC & JdbcTemplate

Opis:

- Narzędzie do bezpośredniego wykonywania zapytań SQL w aplikacji.

Powody wyboru:

- Zapewnia bezpośrednią kontrolę nad zapytaniami SQL.
- Wspiera dynamiczne generowanie zapytań i ich parametryzowanie.

- Hibernate (JPA)

Opis:

- Hibernate to framework ORM dla Javy, który pozwala na mapowanie obiektów do relacyjnej bazy danych.

Powody wyboru:

- Redukuje ilość kodu SQL, który trzeba napisać.
- Obsługuje cache'owanie zapytań, co poprawia wydajność aplikacji.
- Automatyczne generowanie schematu bazy danych.

- Jakarta Persistence API (JPA)

Opis:

- Specyfikacja API do mapowania obiektowo-relacyjnego w Javie, wykorzystywana przez Hibernate i Spring Data JPA.

Powody wyboru:

- Standardowy interfejs do zarządzania danymi w aplikacjach Javy.
- Eliminuje potrzebę pisania kodu SQL dla operacji CRUD.
- Łatwa integracja z Spring Boot.

- Spring Boot Actuator

Opis:

- Narzędzie do monitorowania aplikacji Spring Boot i jej wydajności.

Powody wyboru:

- Umożliwia zbieranie metryk systemowych.
- Monitoruje zasoby aplikacji.
- Zapewnia gotowe punkty dostępowe do monitorowania API.

Frontend

Frontend aplikacji "Liga Piłkarska" został zbudowany z wykorzystaniem HTML, CSS i JavaScript, co zapewnia czytelną strukturę, estetyczny wygląd i dynamiczne interakcje użytkownika. Poniżej znajduje się szczegółowy opis każdej użytej technologii i powody jej wyboru.

- HTML (HyperText Markup Language)
 - definiuje strukturę strony internetowej, umożliwiając organizację treści, takich jak formularze, tabele wyników oraz interaktywne elementy użytkownika index.
- CSS (Cascading Style Sheets)
 - odpowiada za wygląd aplikacji, w tym stylizację tekstu, układ strony oraz responsywność, co zapewnia estetyczne i spójne wrażenia wizualne.
- JavaScript (Vanilla JS)
 - umożliwia dynamiczne pobieranie danych z backendu oraz interaktywność, np. wyświetlanie wyników meczów w czasie rzeczywistym, obsługę formularzy i animacjescrpts.

Opis dostępnych endpointów API aplikacji "Liga Piłkarska"

- Endpointy dotyczące graczy (Players)
 - GET /api/players/top-scorers
Opis: Zwraca listę najlepszych strzelców (zawodników z największą liczbą bramek).
 - GET /api/players/top-assisters
Opis: Zwraca listę najlepszych asystentów (zawodników z największą liczbą asyst).
 - GET /api/players/top-goalkeepers
Opis: Zwraca listę najlepszych bramkarzy (zawodników z największą liczbą czystych kont).
 - GET /api/players/details
Opis: Pobiera listę wszystkich zawodników z dodatkowymi informacjami, np. drużyna i pozycja.
 - POST /api/players/details
Opis: Dodaje nowego zawodnika.
 - PUT /api/players/details/{id}
Opis: Aktualizuje dane zawodnika o podanym id.
 - DELETE /api/players/details/{id}
Opis: Usuwa zawodnika o podanym id z bazy danych.
- Endpointy dotyczące drużyn (Teams)
 - GET /api/teams
Opis: Pobiera listę wszystkich drużyn.
 - GET /api/teams/ranking
Opis: Pobiera ranking drużyn na podstawie wyników w lidze.
 - GET /api/teams/raw
Opis: Pobiera drużyny bezpośrednio z bazy danych, zwracając ich nazwy.
 - POST /api/teams
Opis: Dodaje nową drużynę do bazy danych.
 - PUT /api/teams/{teamId}
Opis: Aktualizuje informacje o drużynie o podanym teamId.
 - DELETE /api/teams/{teamId}
Opis: Usuwa drużynę o podanym teamId.

- Endpointy dotyczące meczów (Matches)
 - GET /api/matches
Opis: Pobiera wszystkie mecze z bazy danych.
 - GET /api/matches/by-date?date={ date }
Opis: Pobiera listę meczów rozegranych w określonym dniu podanym jako parametr date.
 - GET /api/matches/dates
Opis: Pobiera unikalne daty meczów dostępne w bazie danych.
 - POST /api/matches
Opis: Dodaje nowy mecz do bazy danych.
 - PUT /api/matches/{ id }
Opis: Aktualizuje istniejący mecz o podanym id.
 - DELETE /api/matches/{ id }
Opis: Usuwa mecz o podanym id.
- Endpointy dotyczące szczegółów meczów (Match Details)
 - GET /api/match-details/{ teamId }/recent
Opis: Pobiera listę ostatnich meczów drużyny o teamId.
 - GET /api/match-details/{ teamId }/all
Opis: Pobiera wszystkie mecze drużyny o teamId.
- Endpointy dotyczące trenerów (Coaches)
 - GET /api/coaches
Opis: Pobiera listę wszystkich trenerów dostępnych w bazie danych.
 - POST /api/coaches
Opis: Dodaje nowego trenera do bazy danych.
 - PUT /api/coaches/{ id }
Opis: Aktualizuje dane trenera o podanym id.
 - DELETE /api/coaches/{ id }
Opis: Usuwa trenera o podanym id z bazy danych.
- Endpointy dotyczące statystyk zawodników (Player Statistics)
 - GET /api/player-statistics
Opis: Pobiera statystyki wszystkich zawodników.
 - PUT /api/player-statistics/{ playerId }
Opis: Edytuje statystyki zawodnika o podanym playerId.
- Endpointy dotyczące sędziów (Referees)
 - GET /api/referees
Opis: Pobiera listę wszystkich sędziów.
 - POST /api/referees
Opis: Dodaje nowego sędziego do bazy danych.
 - PUT /api/referees/{ refereeId }
Opis: Aktualizuje dane sędziego o podanym refereeId.
 - DELETE /api/referees/{ refereeId }
Opis: Usuwa sędziego o podanym refereeId z bazy danych.

- Endpointy dotyczące stadionów (Stadiums)
 - GET /api/stadiums
Opis: Pobiera listę wszystkich stadionów.
 - POST /api/stadiums
Opis: Dodaje nowy stadion do bazy danych.
 - PUT /api/stadiums/{stadiumId}
Opis: Aktualizuje dane stadionu o podanym stadiumId.
 - DELETE /api/stadiums/{stadiumId}
Opis: Usuwa stadion o podanym stadiumId z bazy danych.
- Endpointy dotyczące szczegółów drużyn (Team Details)
 - GET /api/team-details/{teamId}
Opis: Pobiera szczegóły drużyny o podanym teamId.
 - GET /api/team-details/{teamId}/players
Opis: Pobiera listę zawodników drużyny o podanym teamId.
- Endpoint dotyczący rankingu drużyn (Team Ranking)
 - GET /api/team-ranking/{teamId}
Opis: Pobiera pozycję drużyny o teamId w rankingu.
- Endpointy dotyczące statystyk drużyn (Team Stats)
 - GET /api/team-stats/ranking
Opis: Pobiera ranking drużyn na podstawie wyników.
 - GET /api/team-stats/attack
Opis: Pobiera listę drużyn z najlepszym atakiem.
 - GET /api/team-stats/defense
Opis: Pobiera listę drużyn z najlepszą obroną.
 - GET /api/team-stats/average-points
Opis: Pobiera średnią liczbę punktów zdobywanych przez drużyny.
 - GET /api/team-stats/home-away
Opis: Pobiera statystyki wyników domowych i wyjazdowych drużyn.
 - GET /api/team-stats/most-common-results
Opis: Pobiera najczęściej występujące wyniki meczów.

Wnioski z realizacji projektu

- Najważniejsze osiągnięcia projektu
 - Stworzenie aplikacji Liga Piłkarska, która umożliwia zarządzanie danymi o drużynach, zawodnikach, trenerach oraz meczach.
 - Implementacja API REST z wykorzystaniem Spring Boot, co zapewniło wydajność i łatwość skalowania aplikacji.
 - Integracja z bazą danych poprzez Spring Data JPA, co umożliwia sprawne zarządzanie danymi drużyn, zawodników i trenerów.
 - Implementacja mechanizmu pobierania najlepszych strzelców, asystentów oraz bramkarzy z bazy danych.
 - Wdrożenie funkcjonalności filtrowania meczów według daty i wyświetlania wyników w interfejsie użytkownika.
 - Stworzenie czytelnego i responsywnego interfejsu użytkownika z użyciem HTML, CSS oraz JavaScript.
 - Obsługa błędów i odpowiednie zarządzanie wyjątkami w kodzie backendu.
- Problemy napotkane podczas realizacji i sposoby ich rozwiązania
 - Błędy w mapowaniu encji JPA
 - W początkowej fazie występowały problemy z adnotacjami JPA (np. błędne mapowanie relacji w klasie Team i Player). Zostały one rozwiązane poprzez poprawne użycie adnotacji @Entity, @Id, @GeneratedValue i konfigurację połączenia z bazą danych.
 - Obsługa błędów zapytań SQL w kontrolerach
 - Podczas implementacji metod zwracających najlepszych zawodników oraz mecze dla danej daty, występowały błędy SQL. Poprawiono zapytania i dodano obsługę wyjątków w try-catch.
 - Problemy z wyświetlaniem danych w interfejsie użytkownika
 - Po stronie frontendu dane nie były poprawnie renderowane. Rozwiązano ten problem poprzez debugowanie i użycie JavaScript (fetch API) do dynamicznego pobierania i aktualizowania treści strony.

Możliwe przyszłe usprawnienia i rozwój projektu

- Sugestie dotyczące przyszłych usprawnień
 - Rozszerzenie API o dodatkowe statystyki – dodanie szczegółowych statystyk dla zawodników, takich jak skuteczność strzałów, liczba podań czy dystans przebiegnięty w meczu.
 - Personalizacja interfejsu użytkownika – umożliwienie użytkownikom dostosowywania wyglądu strony (ciemny motyw, różne układy tabel itp.).
 - Logowanie i role użytkowników – wdrożenie systemu autoryzacji, gdzie administratorzy mogą zarządzać danymi, a użytkownicy mają tylko dostęp do odczytu.
- Plany na dalszy rozwój projektu
 - Integracja z zewnętrznymi API – np. pobieranie danych z FIFA API lub innych źródeł w celu uzyskania rzeczywistych statystyk meczowych.
 - Rozwój systemu powiadomień – wysyłanie użytkownikom powiadomień o nadchodzących meczach ich ulubionych drużyn.
 - Wdrożenie aplikacji mobilnej – utworzenie wersji mobilnej aplikacji w React Native, aby użytkownicy mogli łatwo przeglądać wyniki i statystyki na telefonach.
 - Dodanie modułu analizy taktycznej – narzędzie do wizualizacji i analizy przebiegu meczów oraz statystyk drużyn.
 - Dodanie funkcjonalności umożliwiającej śledzenie, w której minucie padła bramka, kto był strzelcem oraz kto asystował. System będzie zapisywał te informacje dla każdego meczu, co pozwoli na dokładniejsze analizowanie przebiegu spotkania.
 - Dodatkowo zostanie wdrożona wizualizacja osi czasu meczu, na której będą zaznaczone kluczowe wydarzenia, takie jak bramki, kartki oraz zmiany zawodników. Użytkownicy będą mogli przeglądać te informacje zarówno w widoku szczegółowym meczu, jak i w zestawieniach statystycznych.
 - Funkcjonalność ta zostanie zautomatyzowana poprzez przypisanie wydarzeń do konkretnego gracza, co umożliwi dynamiczne obliczanie punktów i statystyk zawodników. System będzie automatycznie aktualizował klasyfikację strzelców, najlepszych asystentów oraz ranking drużyn na podstawie wyników meczów.

4. Harmonogram realizacji projektu

Projekt realizowany był z wykorzystaniem systemu kontroli wersji Git, wszystkie pliki źródłowe projektu znajdują się pod adresem:

<https://github.com/SebastianKuzyk/LigaPilkarska> i będą dostępne do 31.12.2025.

5. Prezentacja warstwy użytkowej projektu

Na Rys. 2 przedstawiono Widok strony głównej użytkownika który przedstawia:

Wybór daty:

- Pozwala użytkownikowi na znalezienie wyników z danego dnia

Tabela z kluczowymi statystykami:

- Najlepsi Strzelcy: Zestawienie zawodników, którzy zdobyli najwięcej bramek, z podaniem ich imienia, nazwiska, drużyny i liczby goli.
- Najlepsi Asystenci: Lista zawodników z największą liczbą asyst w meczach, wraz z ich drużyną i liczbą asyst.
- Najlepsi Bramkarze: Zawodnicy grający na pozycji bramkarza, posortowani według liczby czystych kont
- Ranking Drużyn: Lista drużyn uszeregowanych według ich pozycji w lidze, z wyróżnieniem miejsca oraz nazwą drużyny.

Liga Piłkarska

[Zobacz statystyki drużyn](#)

Wybierz datę: 11.09.2024

Brentford 2 - 2 Tottenham

Chelsea 1 - 1 Liverpool

Najlepsi Strzelcy				Najlepsi Asystenci				Najlepsi Bramkarze				Ranking Drużyn	
Imię	Nazwisko	Zespół	Bramki	Imię	Nazwisko	Zespół	Asysty	Imię	Nazwisko	Zespół	Czyste Konta	Pozycja	Nazwa Drużyny
Can	Burn	Newcastle	6	Tariq	Lamprey	Brighton	7	Bernd	Leno	Fulham	3	1	Arsenal
James	Milner	Brighton	6	Kieran	Trippier	Newcastle	5	David	Raya	Arsenal	3	2	Aston Villa
Thiago	Alcantara	Liverpool	6	Phil	Foden	Manchester City	4	Dean	Henderson	Crystal Palace	3	3	Manchester City
Matty	Cash	Aston Villa	5	Pierre-Emile	Højbjerg	Tottenham	4	Nick	Pope	Newcastle	3	4	Liverpool
Jan	Paul van Hecke	Brighton	5	Bruno	Guimarães	Newcastle	4	Djordje	Petrovic	Chelsea	3	5	Newcastle
Douglas	Lutz	Aston Villa	5	James	Milner	Brighton	4	Ederison	Moraes	Manchester City	3	6	Brighton
Bukayo	Saka	Arsenal	5	Kieran	Lewis-Potter	Brentford	3	Emiliano	Martinez	Aston Villa	2	7	Tottenham
Pontus	Jansson	Brentford	4	Martin	Odegaard	Arsenal	3	Guglielmo	Vicario	Tottenham	2	8	West Ham
Virgil	van Dijk	Liverpool	4	Darwin	Nunez	Liverpool	3	Andre	Orsma	Manchester United	2	9	Manchester United
Tariq	Lamprey	Brighton	4	Ollie	Watkins	Aston Villa	3	David	Raya	Brentford	1	10	Birmingham Forest
												11	Chelsea
												12	Crystal Palace
												13	Brentford

Rys. 2. Widok strony głównej

Po wybraniu opcji „Zobacz statystyki drużyn” zostanie wyświetlona podstrona z ze wszystkimi najważniejszymi statystykami (Rys. 3):

Liga Piłkarska					
Ranking Drużyn					
Pozycja	Drużyna	Punkty	Różnica bramek	Żółte kartki	Czerwone kartki
1	Arsenal	12	4	8	1
2	Aston Villa	9	11	6	1
3	Manchester City	9	7	5	1
4	Liverpool	9	7	7	1
5	Newcastle	9	6	7	1
6	Brighton	9	5	4	2
7	Tottenham	9	4	6	0
8	West Ham	6	2	7	1
9	Manchester United	6	-1	4	1
10	Nottingham Forest	5	1	5	1
11	Chelsea	4	1	3	1
12	Crystal Palace	4	0	3	0
13	Brentford	3	-2	5	1
14	Fulham	3	-4	3	1
15	Wolves	3	-4	5	2
16	Bournemouth	2	-6	4	1
17	Luton Town	1	-2	3	0
18	Sheffield United	1	-10	3	1
19	Everton	0	-4	2	0
20	Burnley	0	-7	2	1

Najlepsza Ofensywa i Defensywa

Rys. 3 Statystyki

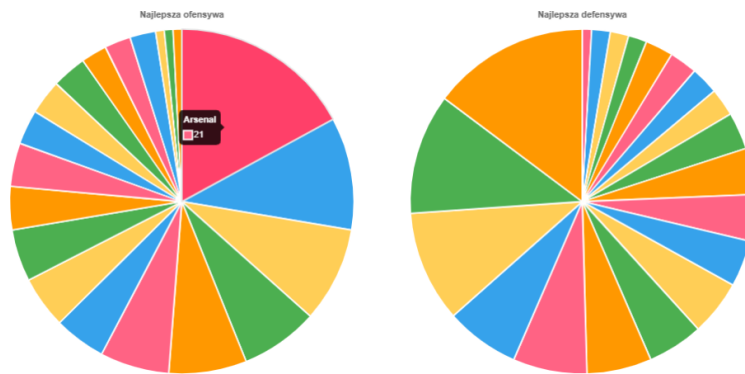
- Ranking drużyn (Rys. 4) – drużyny sortowane są według liczby zdobytych punktów.

Ranking Drużyn					
Pozycja	Drużyna	Punkty	Różnica bramek	Żółte kartki	Czerwone kartki
1	Arsenal	12	4	8	1
2	Aston Villa	9	11	6	1
3	Manchester City	9	7	5	1
4	Liverpool	9	7	7	1
5	Newcastle	9	6	7	1
6	Brighton	9	5	4	2
7	Tottenham	9	4	6	0
8	West Ham	6	2	7	1
9	Manchester United	6	-1	4	1
10	Nottingham Forest	5	1	5	1
11	Chelsea	4	1	3	1
12	Crystal Palace	4	0	3	0
13	Brentford	3	-2	5	1
14	Fulham	3	-4	3	1
15	Wolves	3	-4	5	2
16	Bournemouth	2	-6	4	1
17	Luton Town	1	-2	3	0
18	Sheffield United	1	-10	3	1
19	Everton	0	-4	2	0
20	Burnley	0	-7	2	1

Rys. 4 Ranking drużyn

- Najlepsza Ofensywa i Defensywa (Rys. 5) – generowane są 2 diagramy kołowe dla najlepszej ofensywy i defensywy po najejchaniu na dany kolor wyświetlają się szczegóły:
 - Najlepsza ofensywa – pokazuje drużyny posortowane według ilości strzelonych bramek.
 - Najlepsza defensywa – pokazuje drużyny posortowane według ilości obronionych bramek.

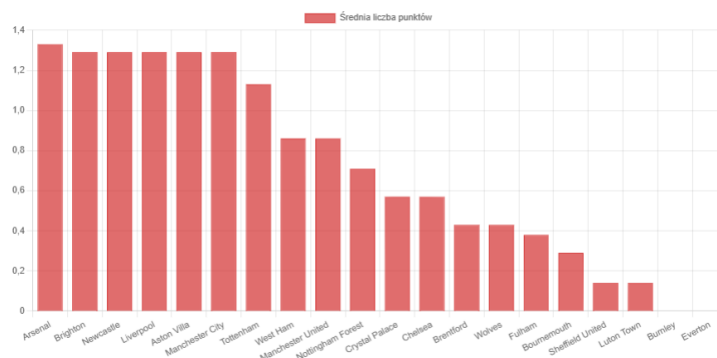
 Najlepsza Ofensywa i Defensywa



Rys. 5 Najlepsza Ofensywa i Defensywa

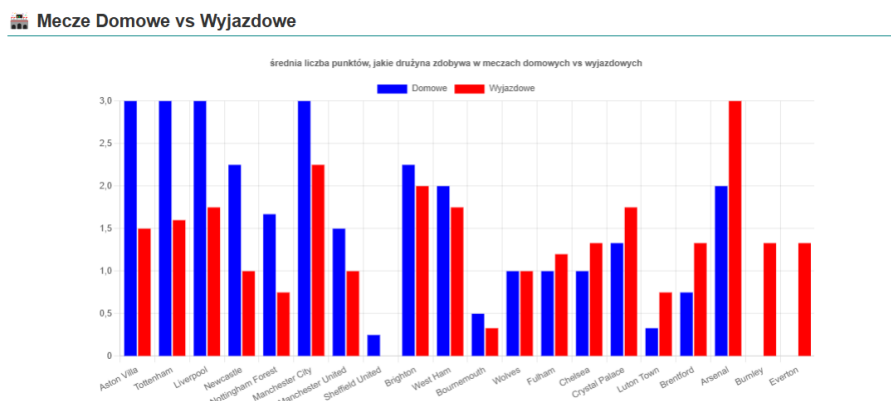
- Średnia Liczba Punktów na Mecz (Rys. 6) – pokazuje średnią liczbę punktów uzyskanych w meczu przez każdą drużynę

 Średnia Liczba Punktów na Mecz



Rys. 6 Średnia Liczba Punktów na Mecz

- Mecze Domowe vs Wyjazdowe (Rys. 7) – pokazuje średnią liczbę punktów, jaką dana drużyna zdobyła w meczach domowych a wyjazdowych.



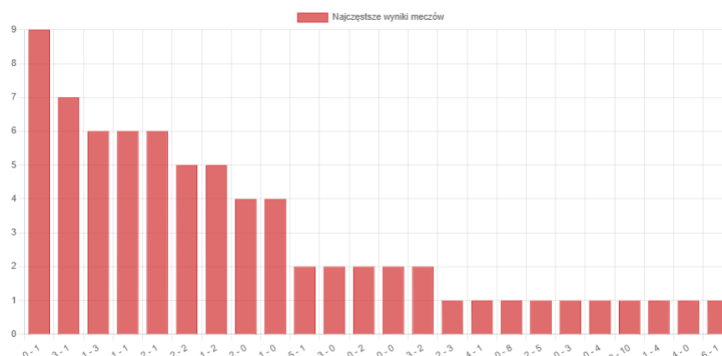
Rys. 7 Mecze Domowe vs Wyjazdowe

- Najczęstsze wyniki meczów – generuje tabele (Rys. 8) oraz wykres (Rys. 9) pokazują one jakimi wynikami mecze kończą się najczęściej.

Najczęstsze wyniki meczów

Wynik	Liczba meczów	Procent
0 - 1	9	12.5%
3 - 1	7	9.72%
1 - 3	6	8.33%
1 - 1	6	8.33%
2 - 1	6	8.33%
2 - 2	5	6.94%
1 - 2	5	6.94%
2 - 0	4	5.56%
1 - 0	4	5.56%
5 - 1	2	2.78%
3 - 0	2	2.78%
0 - 2	2	2.78%
0 - 0	2	2.78%
3 - 2	2	2.78%
2 - 3	1	1.39%
4 - 1	1	1.39%
0 - 8	1	1.39%
2 - 5	1	1.39%
0 - 3	1	1.39%
0 - 4	1	1.39%
10 - 10	1	1.39%
1 - 4	1	1.39%
4 - 0	1	1.39%
6 - 1	1	1.39%

Rys. 8 Najczęstsze wyniki meczów - tabela



Rys. 9 Najczęstsze wyniki meczów – wykres

Po wybraniu jednej z drużyn z Rankingu Drużyn na stronie głównej projektu przechodzimy na podstronę tej drużyny (Rys. 10). W tym widoku otrzymujemy takie informacje jak nazwa oraz trener drużyny, informacje o stadionie i lokalizacji. Następnie przedstawione są statystyki najlepszych zawodników: Najwięcej goli i asyst, najlepszy bramkarz oraz zawodników którzy mają najwięcej żółtych i czerwonych kartek. Pod spodem znajduje się przycisk który wyświetla pełną listę zawodników drużyny. Następnie wyświetlana jest lista ostatnich meczy oraz przycisk Pokaż pełną listę meczów.

Arsenal				
Trener: Mikel Arteta				
Aktualna pozycja w rankingu: 1				
Informacje o drużynie				
Stadion: Emirates Stadium				
Miasto: London				
Statystyki zawodników				
<ul style="list-style-type: none"> Najwięcej goli: Bukayo Saka (5) Najwięcej asyst: Bukayo Saka (3) Najlepszy bramkarz: David Raya (3) Najwięcej żółtych kartek: Kai Havertz (3) Najwięcej czerwonych kartek: Gabriel Jesus (1) 				
Pokaż pełną listę zawodników				
Ostatnie mecze				
Gospodarz	Wynik	Gość	Stadion	Rezultat
Arsenal	10 - 10	Fulham	Villa Park	u siebie
Arsenal	2 - 1	Tottenham	Villa Park	u siebie
Arsenal	2 - 2	Tottenham	Emirates Stadium	u siebie
Arsenal	3 - 1	Manchester United	Emirates Stadium	u siebie
Arsenal	2 - 2	Fulham	Emirates Stadium	u siebie
Bournemouth	0 - 4	Arsenal	Vitality Stadium	na wyjeździe
Everton	0 - 1	Arsenal	Goodison Park	na wyjeździe
Crystal Palace	0 - 1	Arsenal	Selhurst Park	na wyjeździe
Pokaż pełną listę meczów				
→ 100%				

Rys. 10 Podstrona danej drużyny

- Po wybraniu opcji pokaż pełną listę zawodników wyświetli się podstrona (Rys. 11) zawierająca szczegółowe informacje na temat wszystkich zawodników.

Arsenal							
Trener: Mikel Arteta							
Aktualna pozycja w rankingu: 1							
Lista zawodników							
Imię	Nazwisko	Pozycja	Bramki	Asysty	Czyste konto	Żółte kartki	Czerwone kartki
Bukayo	Saka	Forward	5	3	0	0	0
Declan	Rice	Midfielder	0	1	0	2	0
Martin	Odegaard	Midfielder	2	3	0	0	0
Gabriel	Jesus	Forward	0	0	0	0	1
Kai	Havertz	Midfielder	2	0	0	3	0
William	Saliba	Defender	0	0	0	0	0
Ben	White	Defender	0	0	0	1	0
Gabriel	Magnhairs	Defender	0	0	0	0	0
Aaron	Ramsdale	Goalkeeper	0	0	0	0	0
Leandro	Trossard	Forward	3	2	0	0	0
Jorginho	Friedo	Midfielder	1	0	0	0	0
Thomas	Partey	Midfielder	0	0	0	0	0
Takhiro	Tomiyasu	Defender	0	0	0	2	0
Oleksandr	Zinchenko	Defender	0	0	0	0	0
Jakub	Kiwior	Defender	1	1	0	0	0
Eddie	Nketiah	Forward	0	0	0	0	0
Ross	Nelson	Forward	1	0	0	0	0
David	Raya	Goalkeeper	0	0	3	0	0
— Wst							

Rys. 11 Lista zawodników

- Po wybraniu opcji pokaż pełną listę meczów wyświetli się podstrona (Rys. 12) zawierająca szczegółowe informacje o wszystkich meczach drużyny.

Arsenal

Trener: Mikel Arteta

Aktualna pozycja w rankingu: 1

Lista wszystkich meczów drużyny

Data	Gospodarz	Gość	Wynik	Stadium	Rodzaj
2024-12-04	Arsenal	Fulham	10 - 10	Villa Park	u siebie
2023-12-04	Arsenal	Tottenham	2 - 1	Villa Park	u siebie
2024-09-24	Arsenal	Tottenham	2 - 2	Emirates Stadium	u siebie
2024-09-03	Arsenal	Manchester United	3 - 1	Emirates Stadium	u siebie
2024-08-26	Arsenal	Fulham	2 - 2	Emirates Stadium	u siebie
2024-08-12	Arsenal	Nottingham Forest	2 - 1	Emirates Stadium	u siebie
2024-09-30	Bournemouth	Arsenal	0 - 4	Vitality Stadium	na wyjeździe
2024-09-17	Everton	Arsenal	0 - 1	Goodison Park	na wyjeździe
2024-08-21	Crystal Palace	Arsenal	0 - 1	Selhurst Park	na wyjeździe

— Wst

Rys. 12 Lista meczów

Po wejściu na stronę administratora użytkownik widzi Panel Administratora (Rys. 13), który umożliwia zarządzanie ligą piłkarską. W sekcji Trenerzy wyświetlana jest tabela zawierająca listę trenerów wraz z ich imieniem, nazwiskiem i zespołem, który prowadzą.

Administrator ma możliwość dodawania nowych trenerów za pomocą przycisku "Dodaj Trenera", a także edycji i usuwania istniejących rekordów poprzez odpowiednie przyciski w kolumnie "Akcje".

Nawigacja w górnej części strony pozwala na przełączanie się między różnymi sekcjami, takimi jak Mecze, Gracze, Statystyki Graczy, Sędziowie, Stadiony i Zespoły, co zapewnia łatwy dostęp do wszystkich funkcji zarządzania ligą.

Panel Administratora			
Trenerzy Mecze Gracze Statystyki Graczy Sędziowie Stadiony Zespoły			
Trenerzy			
Dodaj Trenera			
Imię	Nazwisko	Zespół	Akcje
Mikel	Arteta	Arsenal	Edytuj Usuń
Unai	Emery	Aston Villa	Edytuj Usuń
Andoni	Isola	Bournemouth	Edytuj Usuń
Thomas	Frank	Brentford	Edytuj Usuń
Roberto	De Zeeb	Brighton	Edytuj Usuń
Vincent	Kompany	Burnley	Edytuj Usuń
Mauricio	Pochettino	Chelsea	Edytuj Usuń
Roy	Hodgson	Crystal Palace	Edytuj Usuń
Sean	Dyche	Everton	Edytuj Usuń
Marco	Silva	Fulham	Edytuj Usuń
Jurgen	Klopp	Liverpool	Edytuj Usuń
Rob	Edwards	Luton Town	Edytuj Usuń
Pep	Guardiola	Manchester City	Edytuj Usuń
Erik	ten Hag	Manchester United	Edytuj Usuń
Eddie	Howe	Newcastle	Edytuj Usuń
Steve	Cooper	Nottingham Forest	Edytuj Usuń

Rys. 13 Panel Administratora

6. Podsumowanie

Realizacja projektu "Liga Piłkarska" przebiegła pomyślnie, a stworzony system działa zgodnie z założeniami. Udało się wdrożyć wszystkie kluczowe funkcjonalności, zapewniając intuicyjną i funkcjonalną aplikację do zarządzania rozgrywkami piłkarskimi. System umożliwia efektywne zarządzanie drużynami, zawodnikami, trenerami oraz wynikami meczów.

W przyszłości projekt można rozbudować o dodatkowe funkcje, takie jak bardziej zaawansowane statystyki zawodników, analizy wydajności drużyn, system rankingowy czy integracja z zewnętrznymi bazami danych sportowych. Dodatkowo możliwe jest wdrożenie funkcjonalności umożliwiających śledzenie transmisji na żywo oraz powiadomienia o wynikach meczów w czasie rzeczywistym.

Te rozszerzenia mogą znacząco zwiększyć funkcjonalność i atrakcyjność systemu, dostarczając użytkownikom jeszcze więcej narzędzi do analizy i monitorowania rozgrywek. Projekt stanowi solidną podstawę, a jego dalszy rozwój może dodatkowo podnieść jego wartość i dostosować go do rosnących potrzeb użytkowników i entuzjastów piłki nożnej.

7. Literatura

1. <https://www.udemy.com/course/spring-boot-dla-poczatkujacych/>
2. <https://justjoin.it/blog/komunikacja-frontend-www-z-backend-w-javie>
3. <https://www.samouczekprogramisty.pl/kurs-sql/>
4. <https://www.premierleague.com/>