POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA
SPECJALNOŚĆ: INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH

PRACA DYPLOMOWA INŻYNIERSKA

Platforma internetowa zrzeszająca zawodników uprawiających amatorsko sporty zespołowe

Web platform for players practicing amateur team sports

AUTOR:
Bartosz Pogoda

PROWADZĄCY PRACĘ:
dr inż. Marek Piasecki, W4/K-9

OCENA PRACY:

WROCŁAW, 2018

Opracował: Tomasz Kubik <tomasz.kubik@pwr.edu.pl>
Data: maj 2016

Spis treści

Spis rysunków

Spis tabel

Rozdział 1

Wstęp

1.1. Wprowadzenie

Sport jest dziedziną towarzyszącą ludzkości od najdawniejszych czasów. Sprawność fizyczna była niezwykle ważną cechą już dla ludzi pierwotnych, dla których niejednokrotnie mogła ona być czynnikiem decydującym o przetrwaniu. W dalszych dziejach ludzkości duży wpływ na narodziny oraz rozwój kultury fizycznej miały starożytne państwa, które organizowały Igrzyska Sportowe.

Ewolucja sportu trwa nadal. Badania wykazują duży wpływ aktywnego trybu życia na zdrowie fizyczne oraz psychiczne człowieka. Aktywność fizyczna jest nieustannie promowana przez lekarzy oraz inne instytucje. Ogromny wpływ na popularyzację zdrowego trybu życia mają również wszelkie organizowane zawody sportowe, które są bardzo popularne w mediach.

Szczególną dziedziną sportu są sporty zespołowe, w których po za indywidualnymi zdolnościami zawodników, ogromne znaczenie ma współpraca. Umiejętność współdziałania w celu osiągnięcia wspólnego celu jest niezwykle istotną cechą, przydatną w wielu życiowych sytuacjach. Wiele sportów zespołowych swój fenomen opiera również na rywalizacji, która daje zawodnikom dodatkową motywację do samorozwoju.

Sporty zespołowe są od wielu lat promowane poprzez organizację dużych imprez takich jak Mistrzostwa Świata, Mistrzostwa Europy. Wraz z popularnością sportów zespołowych rośnie liczba osób, które uprawiają sporty zespołowe amatorsko[1]. Osoby takie poprzez wspólną grę oraz rywalizację mogą poprawić swoją sprawność fizyczną aktywnie spędzając czas, jak również nawiązać nowe znajomości.

Popularyzacja aktywnego trybu życia w społeczeństwie stanowi motywację oraz uzasadnienie zapotrzebowania na rozwiązania, które przy użyciu technologii dostępnych w dzisiejszych czasach, wspierałyby komunikację pomiędzy osobami uprawiającymi amatorsko sporty zespołowe.

W celu zapewnienia dużej dostępności rozwiązania bez względu na sprzęt oraz położenie użytkownika naturalnym wyborem jest umieszczenie platformy w Internecie, do którego dostęp ma znaczna większość populacji. Istniejące platformy internetowe takie jak Facebook, YouTube sukcesywnie wspomagają budowanie społeczności ludzi o wspólnych zainteresowaniach, w dużej mierze ze względu na swoją szeroką dostępność.

[1] Amatorsko czyli traktując sport jako hobby, dodatek do życia, a nie jego główny kierunek i źródło utrzymania

1.2. Cel i zakres pracy

Celem niniejszej pracy jest zaprojektowanie oraz implementacja prototypu platformy internetowej, która zrzeszałaby osoby uprawiające sporty zespołowe w sposób amatorski poprzez:

- utrzymywanie bazy zawodników,
- utrzymywanie bazy drużyn,
- utrzymywanie bazy obiektów sportowych,
- wspomaganie poszukiwania rywali,
- wspomaganie umawiania się na rozgrywkę.

Do zakresu pracy należy projekt ogólnego rozwiązania, które mogłoby z powodzeniem być zastosowane dla różnych dyscyplin sportów zespołowych. Implementacja części klienckiej zostanie jednak zawężona do jednej dyscypliny sportu zespołowego - koszykówki 3 na 3. Wybór padł na tę dyscyplinę ze względu na jej rosnącą popularność oraz brak istniejących rozwiązań ukierunkowanych na organizację rozgrywek w tej dyscyplinie.

Projekt został nazwany Team Challenge. Nazwa ta nawiązuje do głównej funkcjonalności projektowanego systemu, czyli do wspierania rywalizacji pomiędzy drużynami poprzez rzucanie wyzwań.

1.3. Układ pracy

Poniżej wymieniono kolejne rozdziały pracy wraz z krótkimi opisami ich zawartości.

- **Istniejące rozwiązania** - Przegląd istniejących platform, które realizują cele zbliżone do projektowanego systemu.

- **Koncepcja systemu Team Challenge** - Przedstawienie ogólnej koncepcji systemu. Przybliżenie sposobów w jaki system będzie realizował wymienione wyżej cele.

- **Projekt systemu Team Challenge** - Techniczny projekt systemu zrealizowany zgodnie ze standardami języka UML oraz uzupełniony o diagram związków encji.

- **Architektura i technologie** - Architektura systemu oraz przegląd wykorzystanych technologii wraz z motywacjami ich użycia.

- **Implementacja i działanie systemu Team Challenge** - Rozdział, w którym zawarto opis przebiegu implementacji oraz wybrane szczegóły działania systemu.

- **Ocena użyteczności** - Rozdział poświęcony testom użyteczności, którym zostało poddane zaimplementowane rozwiązanie. Przedstawienie metodyki badań oraz ich wyników.

- **Perspektywy rozwoju** - Nakreślenie pomysłów na dalsze kierunki prac nad systemem.

- **Podsumowanie** - Ostatni rozdział pracy, zawierający przemyślenia na temat projektu oraz wnioski wysnute podczas jego realizacji.

Rozdział 2

Istniejące rozwiązania

W niniejszym rozdziale przedstawione zostaną wybrane z istniejących rozwiązań wspierających komunikację pomiędzy zawodnikami. Dla poszczególnych platform zostaną wyszczególnione ich główne założenia oraz funkcjonalności.

2.1. Facebook

Facebook jest serwisem społecznościowym zrzeszającym prawie 2 miliardy osób z całego świata <odw wikipedia?>. Główną misją Facebooka jest zbliżanie do siebie ludzi poprzez umożliwianie budowy społeczności.

Społeczność osób uprawiających sporty zespołowe nie stanowi tutaj wyjątku. Na Facebooku istnieje bardzo dużo grup tematycznych, których celem jest gromadzenie osób uprawiających pewną dyscyplinę sportu w określonym regionie. Osoby szukające osób do gry bardzo często tworzą posty podając takie informacje jak miejsce oraz termin spotkania, preferowany wiek oraz poziom umiejętności. Chętne osoby zgłaszają się pod postem lub poprzez wiadomość prywatną.

Szukanie osób do gry poprzez portal Facebook jest bardzo często wybieranym rozwiązaniem głownie ze względu na dużą ilość użytkowników oraz szeroką dostępność serwisu. W tabeli XX przedstawiono zestawienie wybranych z publicznych grup w mieście Wrocław

Tab. 2.1: Przykładowe grupy dla zawodników na Facebook - stan z dnia 20.11.2018r

Nazwa grupy	Liczba członków
Piłka nożna Wrocław	4689
Siatkówka Wrocław	4460
Koszykówka Wrocław	1686
Piłka nożna Wrocław - dla PWR	273

>Coś tutaj więcej...Może o jakichś wadach takiego zastosowania

2.2. Playarena.pl

Playarena jest portalem skierowanym do drużyn piłki nożnej 6 osobowej.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

O playarena (http://playarena.pl/corobimy)

2.3. SportsMatchMaker.com.au

O Australinskim Sportsmatchmaker (http://www.sportsmatchmaker.com.au/aboutus.aspx)

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Rozdział 3

Koncepcja systemu Team Challenge

W niniejszym rozdziale zostanie przedstawiony koncept głównych funkcjonalności projektowanego systemu. Podczas pracy koncepcyjnej autor pracy miał na względzie główny cel systemu jakim jest zrzeszanie zawodników. Proponowane funkcjonalności mają umożliwić osiągnięcie tego celu. Tworząc koncept autor kierował się znajomością potrzeb grupy docelowej, której sam jest częścią, a niejednokrotnie swoje pomysły weryfikował z innymi zawodnikami.

3.1. Baza wiedzy

W celu umożliwienia zrzeszania zawodników sportów zespołowych konieczne jest przechowywanie w systemie bazy wiedzy na temat zawodników, drużyn oraz obiektów sportowych. W poniższych sekcjach zostaną przybliżone główne założenia dotyczące przechowywania poszczególnych danych.

3.1.1. Baza zawodników

Zawodnicy uprawiający amatorsko pewną dyscyplinę sportu są podstawową grupą docelową projektowanego systemu oraz elementem budującym społeczność. Podstawową funkcjonalnością systemu musi być rejestracja zawodników. Podczas procesu rejestracji od użytkownika powinny zostać pobrane dane niezbędne do funkcjonowania systemu, jak również informacje potrzebne do realizacji dalszych założeń, co będzie opisane w kolejnych podpunktach.

3.1.2. Baza drużyn

Zawodnicy po utworzeniu profilu będą mogli założyć drużynę lub dołączyć do już istniejącej drużyny poprzez otrzymanie oraz akceptację zaproszenia. Zawodnik zakładający drużynę otrzymuje specjalną rolę - kapitana. Kapitanem drużyny powinna być osoba reprezentatywna oraz posiadająca dobry kontakt z pozostałymi członkami zespołu, ponieważ to kapitan będzie zajmował się poszukiwaniem przeciwników oraz umawianiem spotkań. Warto zaznaczyć, że kapitan drużyny dalej pozostaje zawodnikiem i może brać czynny udział w rozgrywkach.

Punkt macierzysty

Drużyna powinna wybrać lokalizację, która na potrzeby tej pracy oraz systemu nazwana została "punktem macierzystym". Punkt ten w obrębie systemu będzie służył jako punkt referencyjny

dla porównywania odległości pomiędzy drużynami, jak również do oceny odległości od obiektów sportowych. Punktem macierzystym może być na przykład ulubione boisko zawodników lub częste miejsce spotkań pobliskie zawodnikom. W przypadku trudności wyboru domyślną lokalizacją jest centrum regionu, w którym została utworzona drużyna.

Aktywność drużyny

Dostęp do kluczowych funkcjonalności takich jak szukanie przeciwników oraz rzucanie wyzwań wymaga posiadania przez drużynę minimalnej liczby zawodników zdefiniowanej dla konkretnej dyscypliny sportu. Drużyna spełniająca powyższy wymóg określana jest mianem drużyny aktywnej.

3.1.3. Baza obiektów sportowych

Mając na uwadze docelową grupę docelową projektowanego systemu jaką są zawodnicy grający amatorsko - system powinien dostarczać bazę obiektów ogólnodostępnych, a przede wszystkim nie wymagających wkładu finansowego. Głównym konceptem w tym zakresie jest umożliwienie zawodnikom zgłaszania oraz weryfikacji obiektów.

3.2. Wspomaganie poszukiwania rywali

Kluczową funkcjonalnością systemu jest wspomaganie poszukiwania rywali do gry. Projektując tę funkcjonalność autor pracy miał na względzie, że najważniejszym ogniwem w systemie jest korzystający z niego człowiek. Z tego względu system nigdy będzie podejmował decyzji o wyborze przeciwnika samodzielnie. Celem systemu będzie wspieranie tego procesu poprzez dostarczanie kapitanowi propozycji przeciwników na podstawie zdefiniowanych przez niego preferencji.

3.2.1. Kryteria dopasowania

Podstawowym kryterium dopasowywania drużyn będzie maksymalizacja satysfakcji z gry. Poziom satysfakcji stanowi jednak kryterium, które jest wręcz niemożliwe do ustalenia wprost. Z tego powodu zostały zdefiniowane przesłanki, które mogą wpływać na większą satysfakcję drużyn z rozgrywki:

- przybliżony wiek zawodników,
- przybliżony poziom umiejętności zawodników,
- przybliżona forma zawodników,
- zachowanie fair-play drużyny przeciwnej,
- dobre wspomnienia po poprzednich rozgrywkach.

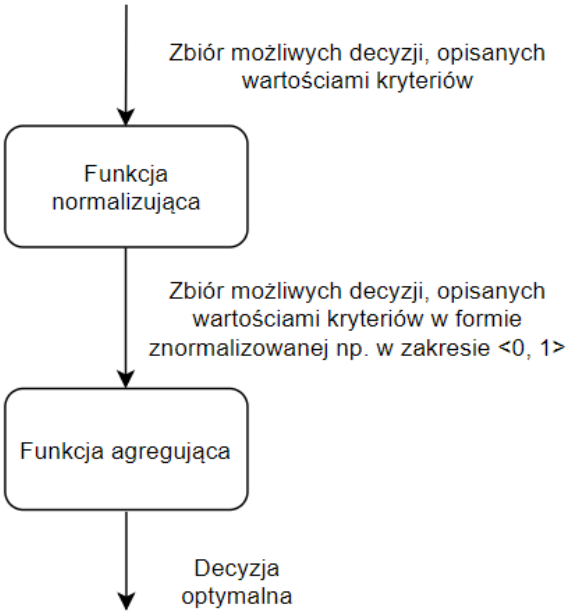
Istotne z punktu widzenia kapitana szukającego przeciwników mogą być również czynniki wpływające na możliwość umówienia spotkania. Mogą być to na przykład informacje dotyczące aktywności drużyny przeciwnej, czy na przykład jej preferowane godziny gry.

Przytoczone kryteria powinny zostać uwzględnione w projektowanym mechanizmie wspomaganego wyszukiwania przeciwników. Mechanizm powinien zostać zaimplementowany w sposób rozszerzalny, aby możliwe było definiowanie nowych kryteriów wraz z potrzebami rozwojowymi systemu.

3.2.2. Problem optymalizacji wielokryterialnej

Problem optymalizacji wielokryterialnej jest rozszerzeniem problemu optymalizacji jednokryterialnej, gdzie poszukiwana jest decyzja optymalna, ze zbioru możliwych decyzji na podstawie jednego kryterium. Problem ten sprowadza się do poszukiwania maksimum (bądź minimum) funkcji oceny danego kryterium [2]. Jeżeli kryterium jest ilościowe, np. maksymalna prędkość samochodu, to podejmując decyzje o zakupie jedynie ze względu na to kryterium naturalnie wybierzemy samochód, który osiąga największą prędkość.

Często jednak podjecie decyzji może być uwarunkowane większą liczbą czynników, na przykład, w przypadku samochodu może to być jego cena, koszt utrzymania, czy czynniki nie ilościowe takie jak jego kolor (w zależności od preferencji kupca). Podejmując decyzje należy rozpatrzyć wiele kryteriów oraz relacje miedzy nimi. Decyzja, która jest optymalna ze względu na jedno z kryteriów, nie musi być optymalna ze względu na pozostałe - z reguły tak nie jest.



Rys. 3.1: Ogólny schemat metody rozwiązywania problemu optymalizacji wielokryterialnej

Tradycyjnym sposobem rozwiązania problemu jest w pierwszej kolejności znormalizowanie wartości kryteriów do przedziału wartości <0,1>, a następnie dokonanie wyboru korzystając z wybranej funkcji agregującej [2]. Ogólny schemat został przedstawiony na rysunku 4.5

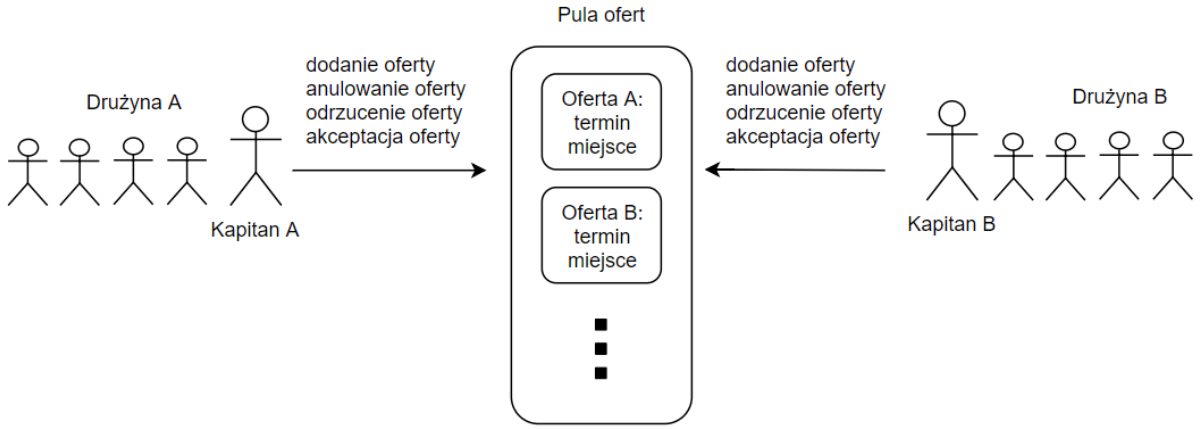
Problem podobnej natury występuje w projektowanym systemie. Poszukiwanie rywali do gry można sprowadzić do poszukiwania najlepszych decyzji wyboru drużyny spośród kwalifikujących się drużyn zdefiniowanych w systemie. Przykładowe kryteria wpływające na wartość funkcji oceny dla poszczególnych decyzji (rywali) zostały zdefiniowane w poprzednim podrozdziale.

3.3. Wspomaganie umawiania się na rozgrywkę

Kolejną bazową funkcjonalnością systemu jest wspieranie przebiegu umawiania spotkań. Kapitan aktywnej drużyny może rzucić wyzwanie innej aktywnej drużynie. Kapitan wyzwanej drużyny może podjąć decyzję o odrzuceniu wyzwania. W przeciwnym wypadku rozpoczyna się proces negocjacji terminu oraz miejsca spotkania.

3.3.1. Negocjacja terminu oraz miejsca spotkania

Kapitanowie drużyn, komunikując się ze swoimi zawodnikami, mogą dodawać do puli ofert swoje propozycje składające się z daty, godziny oraz miejsca spotkania. Miejscem spotkania może być dowolny obiekt sportowy zgłoszony w tym samym regionie co drużyny. System mógłby jednak proponować obiekty położone blisko punktów macierzystych obydwu drużyn. Negocjacje spotkania kończą się w momencie gdy jeden z kapitanów zaakceptuje propozycję drużyny przeciwnej. Termin oraz miejsce zawarte w zaakceptowanej ofercie stają się planowanym terminem oraz miejscem spotkania, a wyzwanie otrzymuje status zaakceptowanego.



Rys. 3.2: Konceptualny diagram negocjacji przy użyciu puli ofert

3.3.2. Wyniki spotkań

Wyzwanie uznaje się za zakończone gdy nastąpi wprowadzenie do systemu jego wyniku. Wynik może zostać wprowadzony przez dowolną z drużyn biorących udział w wyzwaniu. Wprowadzony wynik powinien zostać poddany weryfikacji przez drugiego z kapitanów, który może go potwierdzić, bądź w przypadku niezgodności, odrzucić. Na wynik powinny składać się informacje takie jak: ilość punktów zdobytych przez poszczególne drużyny. W przypadku gdy są to dwie różne liczby wyzwanie kończy się zwycięstwem drużyny, która zdobyła ich więcej. W przeciwnym razie wynikiem wyzwania jest remis.

3.3.3. Ocena drużyn

Kapitanowie drużyn powinni być w stanie dokonać oceny drużyny rywali po zakończonym spotkaniu. Wypełnienie formularza ma na celu dostarczenie do systemu danych, które mogą poprawić jakość wyszukiwania drużyn. Przykładowymi pytaniami mogą być tutaj: ocena poziomu fair-play drużyny oraz deklaracja chęci ponownej rozgrywki w przyszłości. Ze względu na umożliwienie swobodnej oraz szczerej odpowiedzi oceny nie mogą być widoczne dla ocenianych drużyn.

Rozdział 4

Projekt systemu Team Challenge

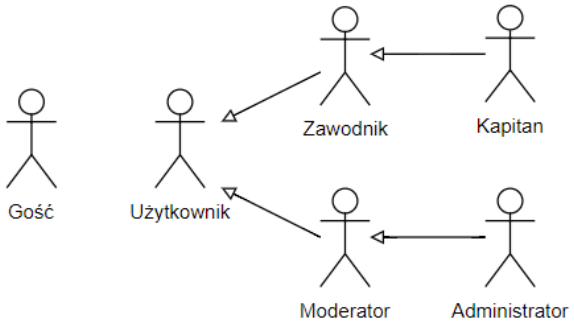
W niniejszym rozdziale przedstawiony został projekt systemu sporządzony na podstawie ogólnej koncepcji opisanej w poprzedniej części. W ramach projektu wydzielono role aktorów na stronie, następnie sporządzono specyfikację wymagań funkcjonalnych oraz niefunkcjonalnych. Na podstawie wymagań funkcjonalnych zostały zamodelowane encje występujące w systemie, co zostało zademonstrowane na diagramie związków encji. W ostatniej części rozdziału przedstawiono diagram stanów dla rzucanego wyzwania.

Diagramy przypadków użycia, związków encji oraz stanów zostały sporządzone przy użyciu narzędzia Visual Paradigm w wersji 15.1. Diagram zależności między grupami w systemie uzyskano przy pomocy internetowego narzędzia draw.io.

4.1. Role w systemie

W systemie zostały wyróżnione następujące role.

- **Gość** - Niezalogowana osoba odwiedzająca system.
- **Użytkownik** - Zarejestrowany oraz zalogowany użytkownik systemu.
- **Zawodnik** - Użytkownik posiadający profil zawodnika.
- **Kapitan** - Zawodnik, który zarządza własną drużyną.
- **Moderator** - Użytkownik posiadający specjalne uprawnienia. Nadzoruje jakość funkcjonalności dostarczanych przez system. Moderuje wprowadzane obiekty sportowe.
- **Administrator** - Specjalny użytkownik posiadający największe uprawnienia. Ma dostęp do wszystkich danych w systemie. Nadzoruje prawidłowe działanie systemu pod względem technicznym.



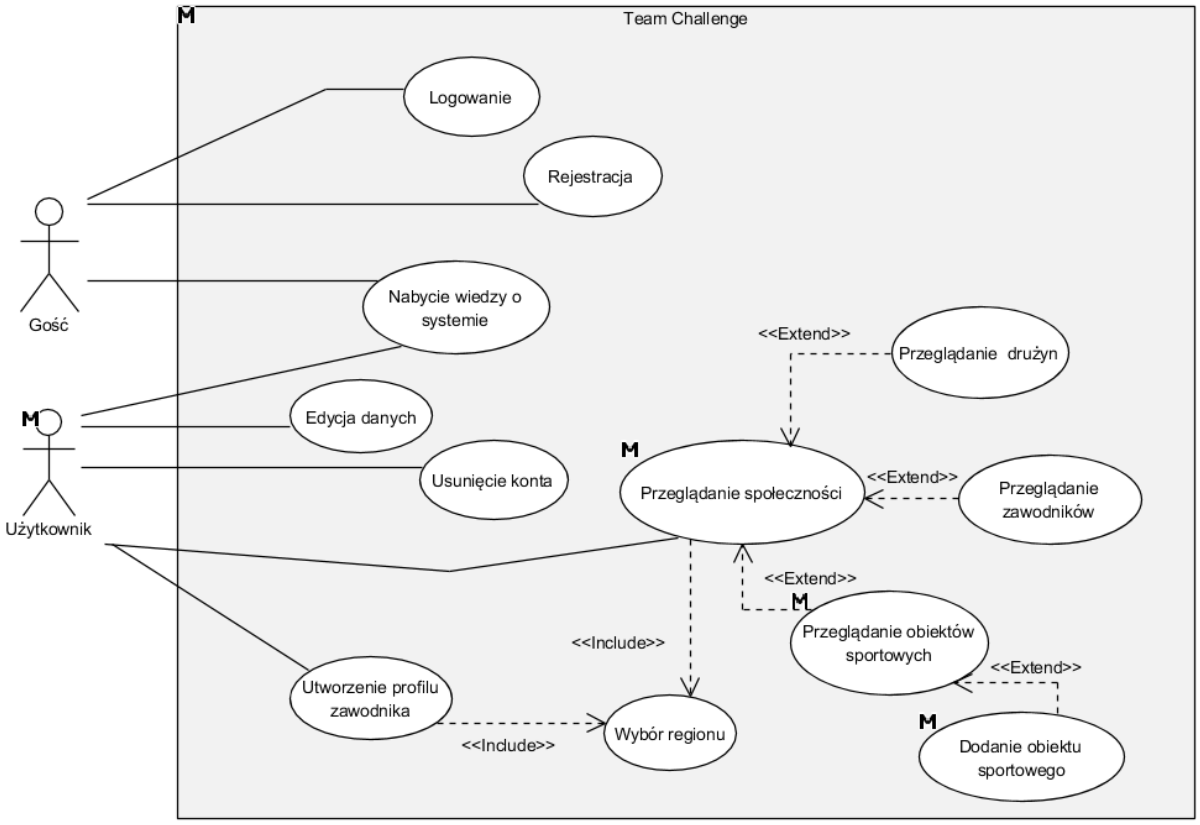
Rys. 4.1: Diagram zależności między grupami w systemie

4.2. Wymagania funkcjonalne

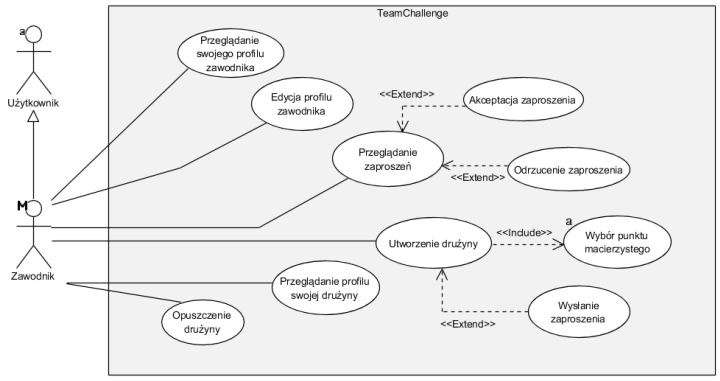
Wymagania funkcjonalne zostały zdefiniowane, aby w jasny sposób opisać oczekiwania wobec systemu oraz jego zakres odpowiedzialności. Dokładne definiowanie wymagań pozwala na zidentyfikowanie możliwych błędów koncepcyjnych jeszcze przed rozpoczęciem implementacji [3]. Specyfikacja wymagań za pomocą przypadków użycia w sposób czytelny przedstawia możliwe interakcje poszczególnych aktorów z systemem [4].

W poniższych sekcjach przedstawiono diagramy przypadków użycia. W celu poprawienia czytelności, przypadki użycia dla niektórych aktorów zostały przedstawione na oddzielnych diagramach. Dodatkowo dla wybranych spośród najbardziej istotnych przypadków zostały sporządzone szczegółowe opisy.

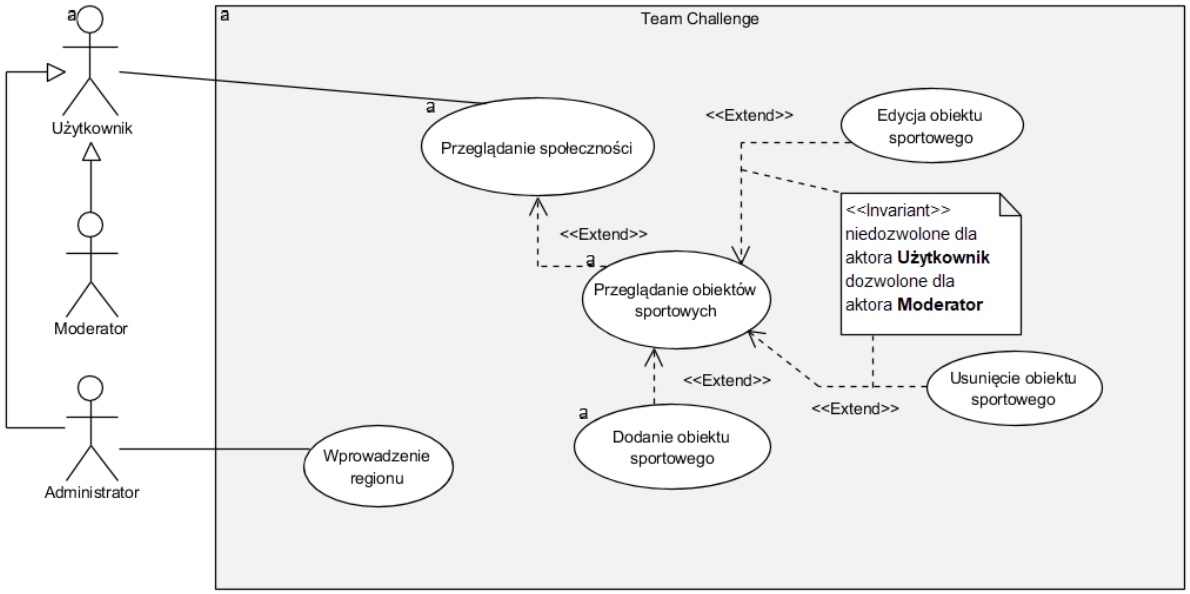
4.2.1. Diagramy przypadków użycia



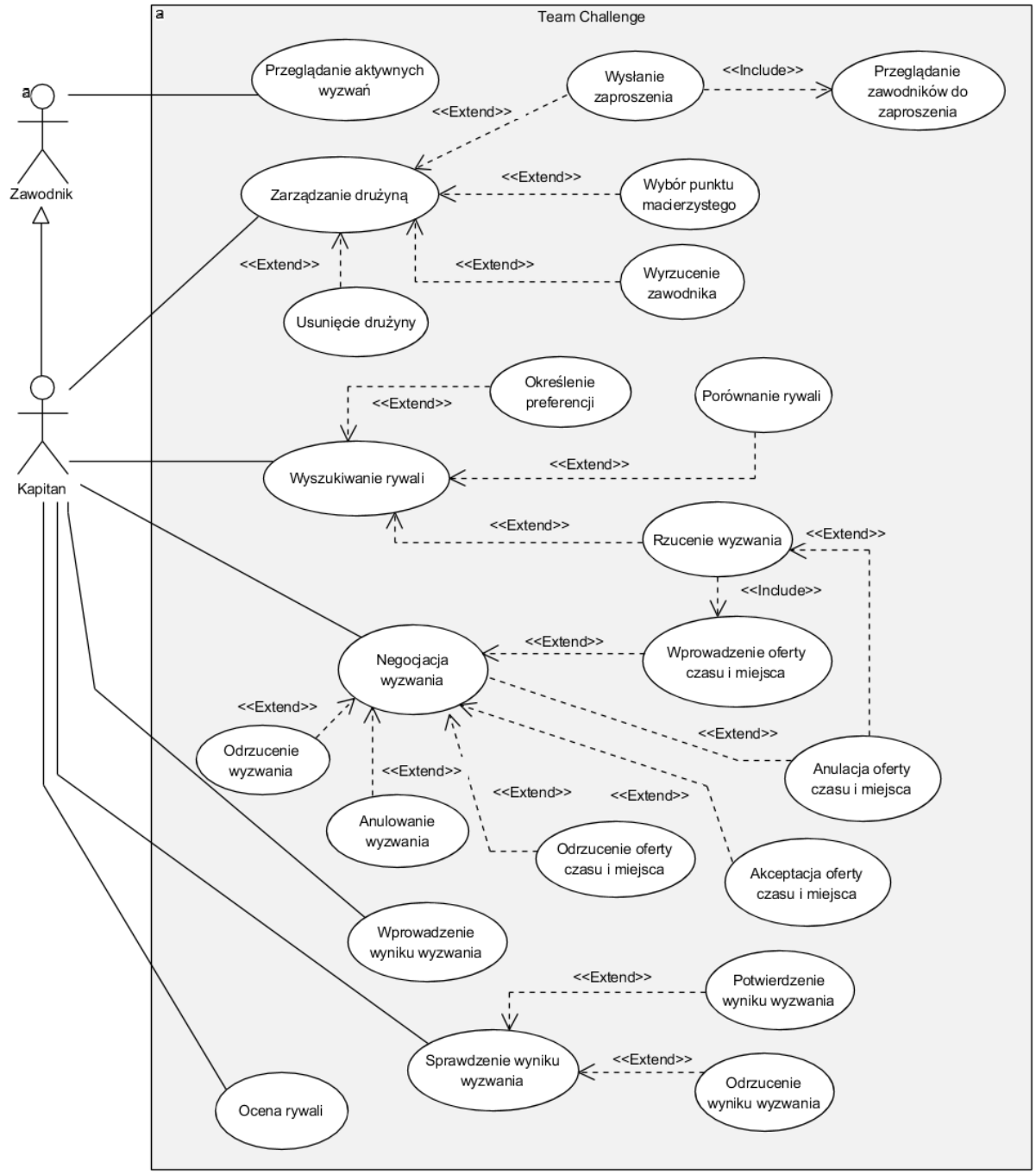
Rys. 4.2: Diagram przypadków użycia - gość oraz użytkownik



Rys. 4.3: Diagram przypadków użycia - zawodnik



Rys. 4.4: Diagram przypadków użycia - moderator oraz administrator



Rys. 4.5: Diagram przypadków użycia - kapitan

4.2.2. PU Rejestracja

Warunki początkowe

Wywołanie przez gościa.

Warunki końcowe

Konto utworzone w systemie. Możliwe zalogowanie za pomocą podanego adresu e-mail oraz hasła.

Przebieg

1. System wyświetla formularz rejestracyjny.
2. Gość wypełnia formularz podając następujące dane: adres e-mail, imię i nazwisko, hasło (dwukrotnie), datę urodzenia.
3. Użytkownik wysyła formularz za pomocą przycisku Źarejestruj się".
   (a) W przypadku błędnie wypełnionego formularza system wyróżnia niepoprawnie wypełnione pola. Powrót do kroku 2.
4. System zapisuje informacje o nowo zarejestrowanym użytkowniku.
5. System informuje gościa o sukcesie oraz możliwości zalogowania.

4.2.3. PU Utworzenie profilu zawodnika

Warunki początkowe

Wywołanie przez użytkownika nie posiadającego profilu zawodnika.

Warunki końcowe

Zapisanie informacji o nowym zawodniku. Użytkownik staje się zawodnikiem oraz uzyskuje dostęp do nowych funkcjonalności.

Przebieg

1. System wyświetla pierwszy krok kreatora zawodnika - podstawowe dane.
2. Użytkownik wybiera region, w którym przebywa - wywołanie **PU Wybór regionu**.
3. Użytkownik wprowadza swój wzrost.
   (a) W przypadku nieprawidłowej wartości system niezwłocznie informuje o tym użytkownika.
4. Użytkownik przechodzi do następnego kroku kreatora za pomocą przycisku "Dalej".
   (a) W przypadku nie wypełnienia wymaganego pola dotyczącego wzrostu system informuje o tym użytkownika wyróżniając to pole. Powrót do kroku 3.
5. System wyświetla drugi krok kreatora zawodnika - deklaracja poziomu umiejętności.
6. Użytkownik definiuje swój poziom umiejętności dopasowując się do jednego z wymienionych oraz opisanych profili umiejętności (początkujący, średnio-zaawansowany itp.).
7. Użytkownik określa częstość swojej gry spośród skończonej liczby możliwości przedstawionych przez system.
8. Użytkownik tworzy profil za pomocą przycisku "Utwórz profil".
9. System informuje użytkownika o pomyślnym utworzeniu profilu.
10. System wyświetla użytkownikowi opcjonalny krok formularza pozwalający na wgranie zdjęcia profilowego.
    (a) Użytkownik może wybrać zdjęcie oraz je przesłać.
    (b) Użytkownik może pominąć krok używając przycisku "Pomiń krok".

4.2.4. PU Utworzenie drużyny

Warunki początkowe

Wywołanie przez zawodnika, który nie jest członkiem żadnej drużyny.

Warunki końcowe

Zapisanie informacji o nowej drużynie w regionie zawodnika. Odrzucenie zaproszeń zawodnika do innych drużyn. Zawodnik staje się członkiem nowo utworzonej drużyny oraz jej kapitanem. Nabycie dostępu do dodatkowych funkcjonalności związanych z rolą kapitana.

Przebieg

1. System wyświetla pierwszy krok kreatora drużyny - podstawowe dane.
2. Zawodnik wprowadza nazwę drużyny.
   (a) W przypadku nieprawidłowej wartości system niezwłocznie informuje o tym zawodnika.
3. Zawodnik przechodzi do następnego kroku kreatora za pomocą przycisku "Dalej".
   (a) W przypadku nie wypełnienia wymaganego pola dotyczącego nazwy drużyny system informuje o tym zawodnika wyróżniając to pole. Powrót do kroku 2.
4. System wyświetla użytkownikowi opcjonalny krok formularza pozwalający na wgranie zdjęcia profilowego.
   (a) Użytkownik może wybrać zdjęcie oraz je przesłać.
   (b) Użytkownik może pominąć krok używając przycisku "Pomiń krok".
5. System wyświetla kolejny krok kreatora drużyny - wybór punktu macierzystego.
6. Użytkownik wybiera punkt macierzysty za pomocą **PU Wybór punktu macierzystego**. A następnie potwierdza wybór za pomocą przycisku "Dalej".
7. System informuje użytkownika o pomyślnym utworzeniu drużyny.
8. System wyświetla użytkownikowi opcjonalny krok formularza pozwalający na zaproszenie zawodników do drużyny.
   (a) Użytkownik może zaprosić zawodników do drużyny - wywołanie **PU Wysłanie zaproszenia**.

4.2.5. PU Wyszukiwanie rywali

Warunki początkowe

Wywołanie przez kapitana, którego drużyna jest aktywna, czyli kwalifikuje się do rozgrywek. Wymogiem jest tutaj spełnienie przez drużynę warunku minimalnej liczby graczy dla danej dyscypliny.

Warunki końcowe

Prezentacja potencjalnych rywali wraz z wskaźnikami umożliwiającymi kapitanowi wnioskowanie na temat poszczególnych propozycji.

Przebieg

1. Wywołanie **PU Określenie preferencji**.
2. System na podstawie preferencji kapitana wyznacza oraz wyświetla kapitanowi listę drużyn kwalifikujących się do rozgrywek w kolejności uwarunkowanej stopniem dopasowania. Dodatkowo dla każdej propozycji system prezentuje wskaźniki pomocne w podjęciu decyzji - poziomy dopasowań poszczególnych kryteriów.
3. Kapitan przegląda propozycje rywali.

(a) Kapitan może porównać dwie lub trzy drużyny - wywołanie **PU Porównanie rywali**.
(b) Kapitan wnioskując na podstawie dostarczonych informacji może wybrać drużynę, której chciałby rzucić wyzwanie - wywołanie **PU Rzucenie wyzwania**.

4.2.6. PU Określenie preferencji

Warunki początkowe

Wywołanie z **PU Wyszukiwanie rywali**.

Warunki końcowe

System otrzymuje informacje dotyczące preferencji kapitana odnośnie poszukiwanych rywali.

Przebieg

1. System wyświetla formularz deklaracji preferencji.
2. System wyświetla podstawowe instrukcje dotyczące działania funkcjonalności.
3. Kapitan za pomocą suwaków oraz kontrolek typu checkbox definiuje swoje preferencje odnośnie poszczególnych cech rywali.
4. Kapitan przesyła formularz za pomocą przycisku “Szukaj“.
5. System uwzględnia przesłany formularz w dalszym przetwarzaniu.

4.2.7. PU Rzucenie wyzwania

Warunki początkowe

Wywołanie z **PU Wyszukiwanie rywali**. Drużyna rzucająca wyzwanie oraz wyzywana spełniają wymogi rozgrywek - są aktywne. Nie istnieje aktualne wyzwanie w toku pomiędzy tymi drużynami.

Warunki końcowe

Wyzwanie zostaje utworzone w systemie. Jest widoczne dla obydwu drużyn. Możliwe są negocjacje terminu oraz miejsca spotkania.

Przebieg

1. System wyświetla podstawowe instrukcje dotyczące funkcjonalności.
2. System wyświetla mapę, na której widoczne są punkty macierzyste drużyny kapitana oraz drużyny, której rzucane jest wyzwanie. Na mapie widoczne są również obiekty sportowe, dla których zostały utworzone oferty czasu i miejsca spotkania.
3. System wyświetla pustą pulę ofert wraz z przyciskiem umożliwiającym dodanie oferty.
4. Kapitan dodaje co najmniej jedną ofertę czasu i miejsca spotkania - wywołanie **PU Wprowadzenie oferty czasu i miejsca**.
   (a) Kapitan może anulować wprowadzone oferty - wywołanie **PU Anulacja oferty czasu i miejsca**.
5. Gdy warunek istnienia co najmniej jednej oferty miejsca oraz czasu jest spełniony, system uaktywnia przycisk “Rzuć wyzwanie“.
6. Kapitan rzuca wyzwanie używając przycisku “Rzuć wyzwanie“.
7. System rejestruje wyzwanie w systemie, informuje użytkownika o sukcesie oraz wyświetla mu widok na którym widoczne jest utworzone wyzwanie.

4.2.8. PU Wprowadzenie oferty czasu i miejsca

Warunki początkowe

Wywołanie w kontekście rzucanego wyzwania bądź wyzwania będącego w trakcie negocjacji.

Warunki końcowe

Dodanie nowej propozycji czasu i miejsca spotkania do puli dla konkretnego wyzwania.

Przebieg

1. System wyświetla mapę, na której widoczne są punkty macierzyste obydwu drużyn biorących udział w wyzwaniu oraz obiekty sportowe dla regionu, w którym istnieją drużyny.
2. System wyświetla odpowiedni formularz.
3. Kapitan wybiera datę spotkania korzystając z kalendarza. Możliwe do wyboru są jedynie daty w przyszłości.
4. Kapitan wybiera godzinę spotkania.
5. Kapitan wybiera miejsce spotkania wciskając wybrany znacznik obiektu sportowego na mapie.
6. Kapitan przesyła ofertę używając przycisku "Dodaj do puli".
   (a) Kapitan może również anulować wprowadzanie oferty za pomocą przycisku Ąnuluj".
7. System zapisuje nową ofertę oraz wyświetla kapitanowi zaktualizowaną pulę zawierającą nowo dodany element.

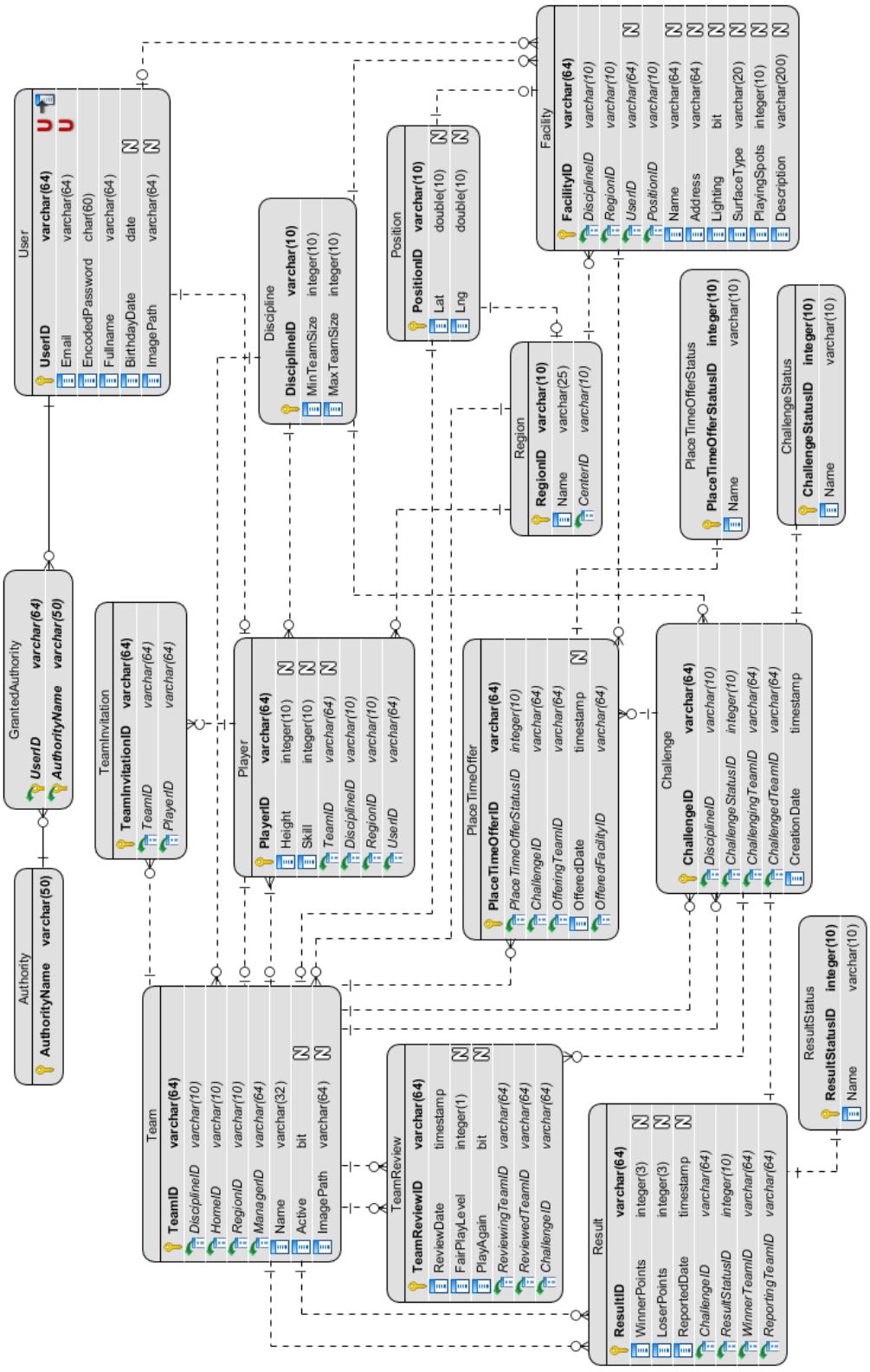
4.3. Wymagania niefunkcjonalne

Po za wymaganiami funkcjonalnymi, które opisują funkcje wykonywane przez system, zdefiniowane zostały również następujące ograniczenia odnośnie procesu implementacji systemu oraz jego działania:

1. Zastosowane technologie muszą być wspierane oraz posiadać dokładną dokumentację techniczną.
2. Implementacja systemu musi przebiegać w sposób iteracyjny, z zastosowaniem systemu kontroli wersji.
3. Kod powstały podczas implementacji systemu powinien być czytelny oraz zrozumiały. Fragmenty mogące budzić niejasności powinny być opisane za pomocą komentarzy.
4. System powinien być szeroko dostępny. Powinien udostępniać swoje funkcje niezależnie od platformy z jakiej korzysta użytkownik.
5. Interfejs systemu powinien być intuicyjny oraz prosty w obsłudze.
6. Operacje wykonywane przez system nie mogą trwać dłużej niż parę sekund. W przypadku dłuższych operacji system musi prezentować użytkownikowi ich postęp.
7. System powinien informować użytkownika o rezultatach podejmowanych akcji - sukcesach oraz błędach.

4.4. Diagram związków encji

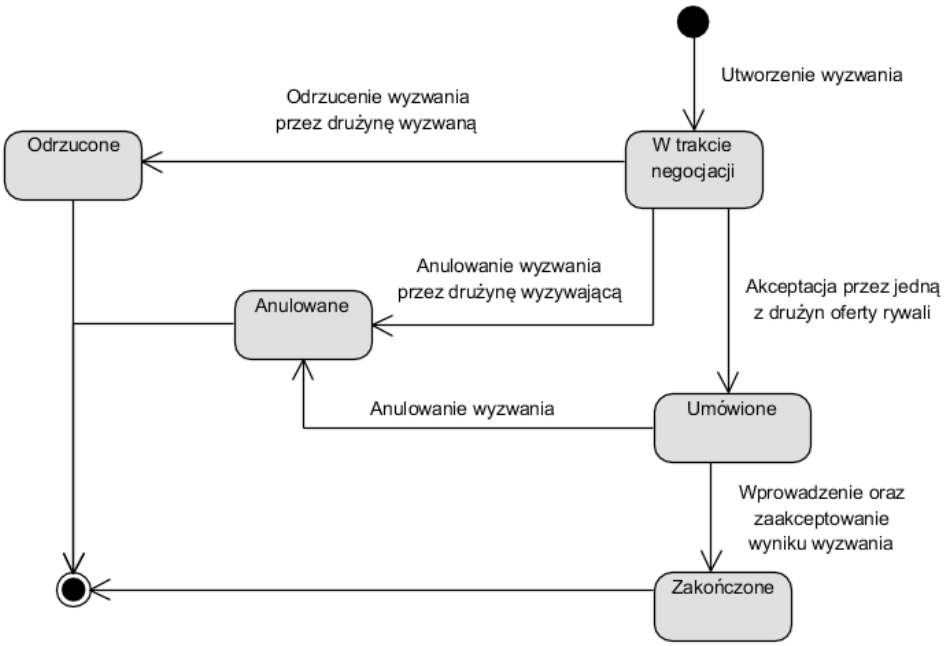
Na podstawie wymagań dotyczących funkcjonowania systemu zidentyfikowane zostały encje. Encje wraz ze swoimi atrybutami oraz powiązaniami zostały przedstawione w sposób graficzny na rysunku 4.6 za pomocą diagramu ERD.



Rys. 4.6: Diagram związków encji

4.5. Diagram stanów - Wyzwanie

Jedną z najważniejszych funkcjonalności systemu jest możliwość rzucania wyzwań innym drużynom. W celu ukazania możliwych stanów w jakim może znajdować się wyzwanie sporządzono diagram stanów. Diagram został przedstawiony na rysunku 4.7.



Rys. 4.7: Diagram możliwych stanów wyzwań

Rozdział 5

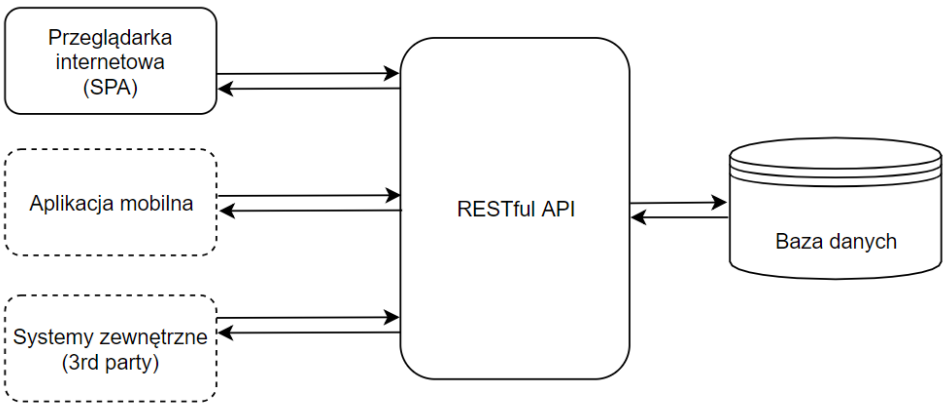
Architektura i technologie

Niniejszy rozdział został poświęcony ogólnemu opisowi architektury systemu oraz zastosowanych technologii. Dla poszczególnych wyborów zostały przedstawione przesłanki, którymi kierował się autor tej pracy.

Istotnym czynnikiem mającym wpływ na wybór technologii oraz wzorców architektonicznych była chęć poszerzenia wiedzy autora niniejszej pracy w ich zakresie. Cechą wspólną wszystkich zastosowanych rozwiązań jest szeroki dostęp do materiałów w postaci dokumentacji oraz pozycji książkowych.

5.1. Architektura systemu

Ze względu na potrzebę szerokiej dostępności platformy została ona zrealizowana jako system webowy w architekturze klient - serwer. Interfejsem użytkownika końcowego jest aplikacja kliencka typu SPA - Single Page Application uruchamiana w przeglądarce internetowej. Aplikacja ta komunikuje się z API wystawionym przez aplikację backendową umieszczoną na serwerze w sieci. Aplikacja backendowa z kolei komunikuje się z bazą danych w celu odczytu oraz zapisu informacji.



Rys. 5.1: Diagram ogólnej architektury systemu

Elementy otoczone linią kreskowaną na diagramie nie są przedmiotem tej pracy, jednak podkreślają uniwersalność API oraz wskazują możliwości rozwojowe oraz integracyjne systemu.

5.1.1. RESTful API

API wystawiane przez część serwerową zostało zaprojektowane w oparciu styl architektoniczny REST, który zakłada komunikację klient-serwer z uwzględnieniem następujących zasad:

- użycie podstawowych metody protokołu HTTP czyli - GET, PUT, POST oraz DELETE,
- identyfikacja zasobów poprzez URL,
- komunikacja bezstanowa (brak sesji).

Użycie podstawowych metod protokołu HTTP pozytywnie wpływa na czytelność oraz intuicyjność API. Projektowanie z uwzględnieniem powyższych zasad pozwala również zminimalizować powiązania pomiędzy serwerem oraz klientem, API staje się uniwersalne. Otwiera to możliwości rozwoju systemu na inne platformy, np. utworzenie aplikacji klienckich dla systemów mobilnych Android oraz iOs. Możliwości rozwoju systemu zostały przedstawione za pomocą zakreskowanych bloków na rysunku 5.1

5.2. Stos technologiczny

5.2.1. Java

Wersje tez

5.2.2. Spring Boot

Aplikacja backendowa została zaimplementowana w języku Java (w wersji 1.8) z użyciem frameworka Spring Boot (w wersji 2.0.3). Technologie te zostały wybrane ze względu na następujące czynniki:

- rozwiązania open source,
- duże grono użytkowników oraz baza materiałów w sieci,
- dobra dokumentacja,
- duża ilość dostępnych modułów Springa np. do komunikacji z bazami danych,
- chęć poszerzenia wiedzy na temat tych technologii.

5.2.3. Swagger

Swagger został wykorzystany do dostarczenia dokumentacji RESTowego API bla bla.

5.2.4. JWT

Technologia Json Web Token została wybrana jako sposób realizacji autoryzacji zapytań do API systemu. JWT jest technologią autoryzacji bezstanowej, przez co bardzo często jest używane do zabezpieczania końcówek REST'owych.

5.2.5. MySQL

Relacyjna baza danych została wybrana ze względu na przewidywaną dużą ilość powiązań między encjami w systemie. MySQL od firmy Oracle jest darmowym, bezpiecznym oraz wydajnym systemem zarządzania bazą danych. Istotnym uzasadnieniem wyboru tej technologii jest również bardzo dobra integracja z frameworkiem Spring.

5.2.6. Angular

Główną technologią wykorzystywaną po stronie front endu będzie framework do tworzenia SPA rozwijany przez Google - Angular (w wersji 6.1.0). Framework ten ułatwia budowę skalowalnych i szybkich aplikacji z bogatym interfejsem użytkownika.

W celu usprawnienia procesu rozwoju aplikacji zostanie wykorzystana biblioteka ngrx (w wersji 6.1.0), wspomagająca zarządzanie stanem aplikacji. Wykorzystanie tej biblioteki znacznie ułatwia analizę działania aplikacji oraz diagnozowanie błędów.

A jakby tak opisac czemu nie react? Ze angular bardzo dobrze sprwadza sie przy duzej ilosci formularzy, react bardziej rendering framework?

5.2.7. AntDesign - NgZorro

5.2.8. Heroku

Rozdział 6

Implementacja i działanie systemu Team Challenge

Lorem ipsum

6.1. Środowisko implementacji

Implementacja systemu odbywała się przy użyciu komputera wyposażonego w 8GB pamięci fizycznej oraz cztero-rdzeniowy procesor Intel Core i5-6300HQ (taktowanie 2.30GHz). Sprzęt o przytoczonych parametrach okazał się w pełni wystarczający dla przebiegu implementacji oraz lokalnego uruchamiania serwera aplikacji. Systemem operacyjnym używanym przy implementacji był Windows w wersji 10 Education. Wszystkie użyte programy i narzędzia dobrze współpracują z tym systemem.

Tab. 6.1: Zestawienie narzędzi wykorzystywanych podczas implementacji systemu

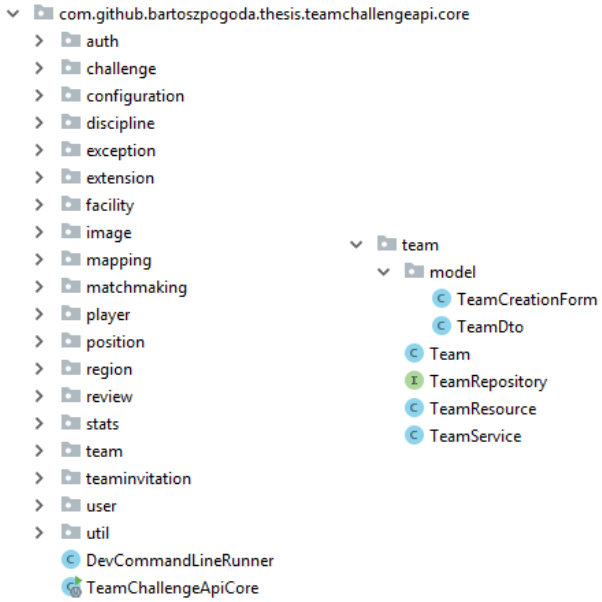
Nazwa programu	Wersja	Producent	Cel
IntelliJ IDEA	2018.2	Jetbrains	Implementacja aplikacji serwerowej (Java)
Webstorm	2018.2	Jetbrains	Implementacja aplikacji klienckiej (Angular)
Postman	6.5.2	Postman	Testowanie końcówek RESTowych aplikacji serwerowej
Google Chrome	70	Google	Testowanie aplikacji klienckiej
Git	2.18.0	-	Kontrola wersji

6.2. Implementacja aplikacji serwerowej

6.2.1. Struktura projektu

Podstawowy szkielet projektu został utworzony przy użyciu Spring Initializr. Jako narzędzie służące do zarządzania zależnościami oraz budowy projektu wybrany został Apache Maven.

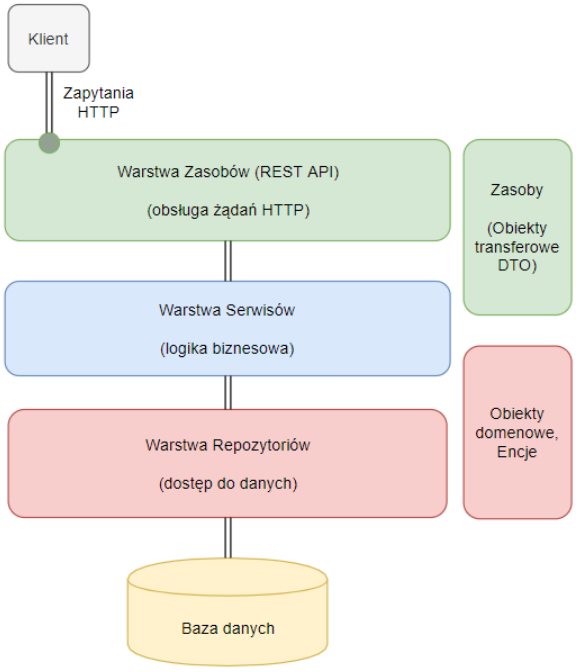
Klasy tworzące aplikację serwerową zostały podzielone na pakiety pod względem tematycznym, co zaprezentowano na rysunku 6.1. Podział taki pozwala na utrzymanie płaskiej struktury katalogów, a w związku z tym umożliwia szybkie odnajdywanie pożądanych plików.



Rys. 6.1: Fragment struktury pakietów

6.2.2. Architektura wielowarstwowa

Podczas projektowania architektury aplikacji ważne jest aby była ona przejrzysta oraz otwarta na rozszerzanie. Jedną z cech, którą powinny posiadać komponenty dobrze zaprojektowanego oprogramowania zorientowanego obiektowo jest ograniczenie odpowiedzialności. Jedną z metod rozdzielania odpowiedzialności jest wyszczególnienie w projekcie warstw komponentów. Aplikacja serwerowa będąca przedmiotem niniejszej pracy została podzielona na warstwy zgodnie ze standardami zdefiniowanymi dla szkieletu Spring. Wyszczególnione warstwy wraz z kierunkami komunikacji zostały przedstawione na rysunku 6.2.



Rys. 6.2: Warstwy aplikacji serwerowej

6.2.3. Warstwa repozytoriów

Repozytoria w frameworku Spring stanowią mechanizm dostępu do danych. Warstwa ta jest abstrakcją ukrywającą przed programistą szczegóły komunikacji z bazą danych takie jak: nawiązywanie oraz utrzymywanie połączenia, konstrukcja i wykonywanie zapytań, mapowanie wyników. Deklaracja fizycznych powiązań między aplikacją a bazą danych odbywa się za pomocą adnotacji. Na listingu 6.1 przedstawiono powiązanie encji Team z fizyczną tabelą o nazwie Teams oraz mapowania przykładowych kolumn.

Listing 6.1: Fragment przykładowej encji

```
@Entity
@Table(name = "Teams")
@Data
@Builder
public class Team {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "TeamID")
    private String id;

    @OneToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "ManagerID")
    private Player manager;

    @OneToMany(mappedBy = "team", cascade = CascadeType.ALL)
    private List<Player> players;

    // other fields with their mappings...
}
```

Dostęp do danych zapewniają repozytoria, czyli interfejsy oznaczone adnotacją @Repository. Wygodne jest rozszerzanie dostarczonego przez moduł Spring Data interfejsu CrudRepository, który definiuje podstawowe metody dostępu takie jak dodawanie, czytanie, modyfikacja oraz usuwanie encji. Pozostałe metody potrzebne do realizacji logiki biznesowej można definiować poprzez tworzenie metod o nazwach specyfikujących ich działanie. Fizyczne zapytania do bazy danych wyznaczane są na podstawie nazwy metody. Przykładową definicję repozytorium przedstawiono na listingu 6.2.

Listing 6.2: Definicja przykładowego repozytorium

```
@Repository
public interface TeamRepository
extends CrudRepository<Team, String>, JpaSpecificationExecutor<Team> {

    Optional<Team> findById(String id);

    List<Team> findByRegionIdAndDisciplineIdAndActiveIsTrue(String regionId,
     String disciplineId);
}
```

Dostarczenie interfejsu repozytorium rozszerzającego JpaSpecificationExecutor pozwala na wykonywanie bardziej zaawansowanych zapytań. Mechanizm ten został zastosowany przy tworzeniu zapytań, gdzie lista predykatów była ustalana w zależności od parametrów zapytania HTTP w sposób dynamiczny. Do budowy kwerend użyto klas Specification oraz Predicate pochodzących z modułu Sring Data JPA.

6.2.4. Warstwa serwisów

Podstawowym zadaniem serwisów jest przetwarzanie danych zgodnie z regułami biznesowymi. W zaimplementowanym systemie na poziomie serwisów również sprawdzane są prawa dostępu do zasobów. W Springu serwisy oznaczane są adnotacje @Service. Framework sam zajmuje się tworzeniem instancji serwisów, które mogą być użyte w pozostałych komponentach systemu. Przykładowy serwis został przedstawiony na listingu 6.3.

Listing 6.3: Fragment przykładowego serwisu

```
@Service
public class TeamService {

    private TeamRepository teamRepository;
    private PositionService positionService;
    // other fields...

    @Transactional
    public Position setHome(String id, PositionDto positionDto)
        throws TeamNotFoundException, AccessForbiddenException {

        Team team = teamRepository.findById(id)
        .orElseThrow(TeamNotFoundException::new);

        if(!isManagedByCurrentUser(team)) {
            throw new AccessForbiddenException();
        }

        Position position = this.positionService.save(positionDto);
        team.setHome(position);

        return position;
    }

    // other methods...
}
```

6.2.5. Warstwa zasobów

Warstwa zasobów jest szczególna, z tego względu, że stanowi interfejs sytemu dla świata zewnętrznego. W ramach tej warstwy działa dostarczony przez szkielet Spring Dispatcher Servlet. Jest to komponent obsługujący żądania HTTP przychodzące do aplikacji. W ramach obsługi żądania są one przekazywane do konkretnych "kontrolerów", wybranych na podstawie URL żądania. Listing 6.6. przedstawia rejestracje kontrolera za pomocą adnotacji @RestController oraz @RequestMapping. Dispatcher Servlet w przypadku tak skonfigurowanej klasy będzie kierował zapytania o adresie /teams do kontrolera TeamResource. Zapytanie /teams/4 zostanie przekazane konkretnie do metody getTeam z argumentem wywołania równym 4.

Na poziomie warstwy zasobów odbywa się również walidacja przychodzących danych.

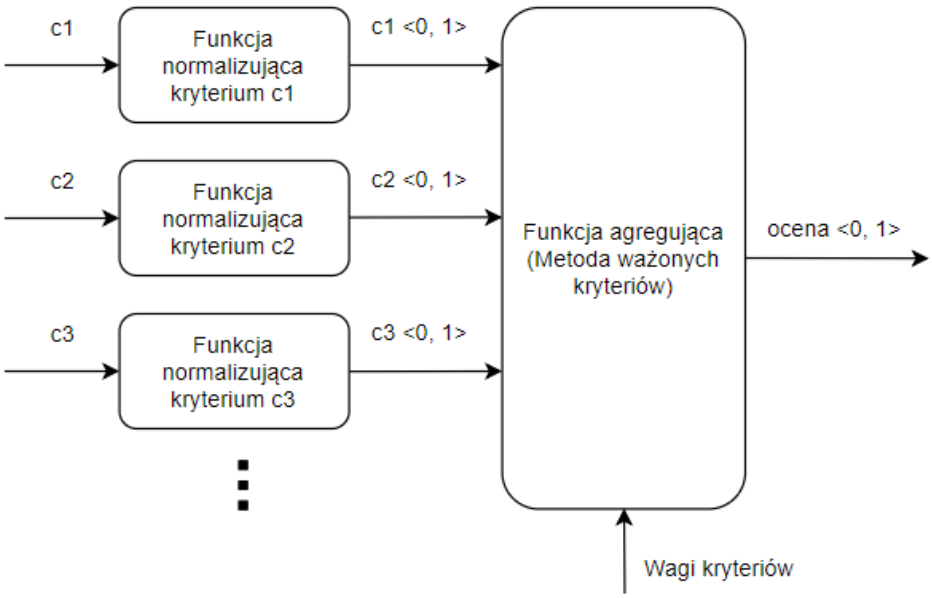
Listing 6.4: Przykładowa rejestracja kontrolera

```
@RestController
@RequestMapping("/teams")
public class TeamResource {

    private TeamService teamService;

    private DtoMappingService mappingService;

    @GetMapping("/{id}")
    public ResponseEntity<TeamDto> getTeam(@PathVariable String id)
     throws ApiException {
        return teamService.findById(id)
                .map(mappingService::mapToDto)
                .map(ResponseEntity::ok)
                .orElseThrow(TeamNotFoundException::new);
    }

    // other methods ...
}
```

Pełna specyfikacja REST API została przedstawiona w załączniku X.

6.2.6. Algorytm poszukiwania rywali

Algorytm poszukiwania rywali został zaimplementowany w oparciu o zagadnienie optymalizacji wielokryterialnej, które zostało przybliżone w trzecim rozdziale niniejszej pracy. Podczas prac nad algorytmem uwzględniono domenę problemu oraz charakter danych, na jakich będzie on operował. W związku z tym tradycyjny schemat oceny decyzji został rozszerzony, co zostało opisane w dalszej części tego rozdziału. Rysunek 6.3 przedstawia uproszczony schemat działania algorytmu zrealizowanego w systemie.



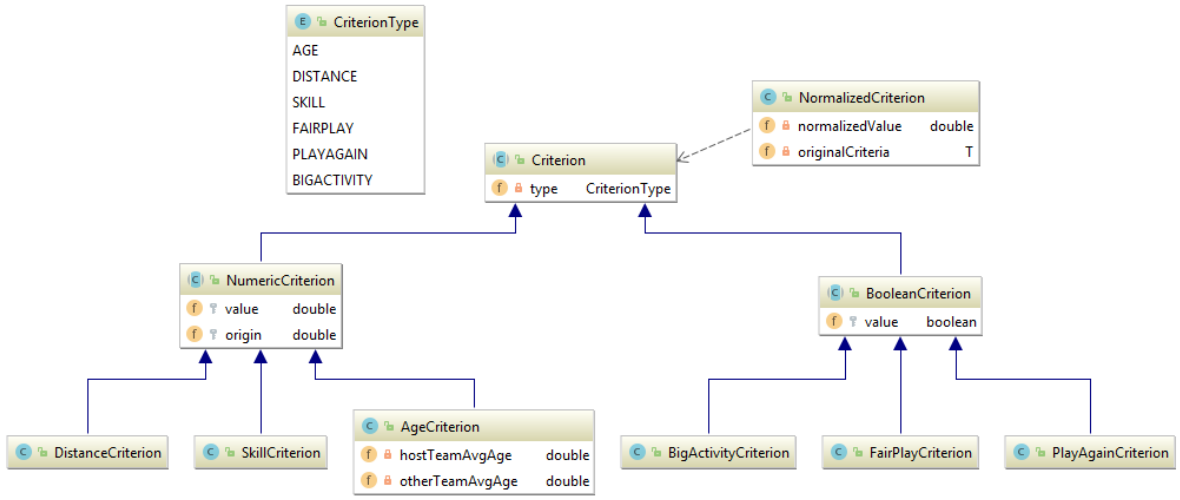
Rys. 6.3: Schemat działania algorytmu oceny decyzji

Kryteria

W systemie zostały wyszczególnione dwa główne rodzaje kryteriów - numeryczne oraz logiczne. Kryteria numeryczne obejmują kryteria, które da się zmierzyć i wyrazić w postaci liczbowej. Uwzględniono tutaj takie kryteria jak: różnica wieku, różnica umiejętności oraz odległość między drużynami. Kryteria logiczne wyrażają dodatkowe wskaźniki, które mają wpływ na dobre dopasowanie drużyn, jednak nie są policzalne. Kryteriom tym przypisane są wartości logiczne, które oznaczają czy dane kryterium zostało spełnione. Przykładowo wartość logiczna kryterium dotyczącego poziomu fair play będzie ustawiona jeżeli średnia ocen fair play potencjalnych rywali jest większa lub równa 4 (maksymalnie 5).

Dodatkowo wyszczególniono generyczną klasę, która opakowuje dowolne kryterium dodając informację o znormalizowanej wartości liczbowej.

Zastosowaną hierarchię klas przedstawiono na rysunku 6.12. Rozbudowa funkcjonalności o nowe kryteria sprowadza się do utworzenia dodatkowej implementacji w odpowiednim miejscu hierarchii.



Rys. 6.4: Hierarchia klas: kryteria

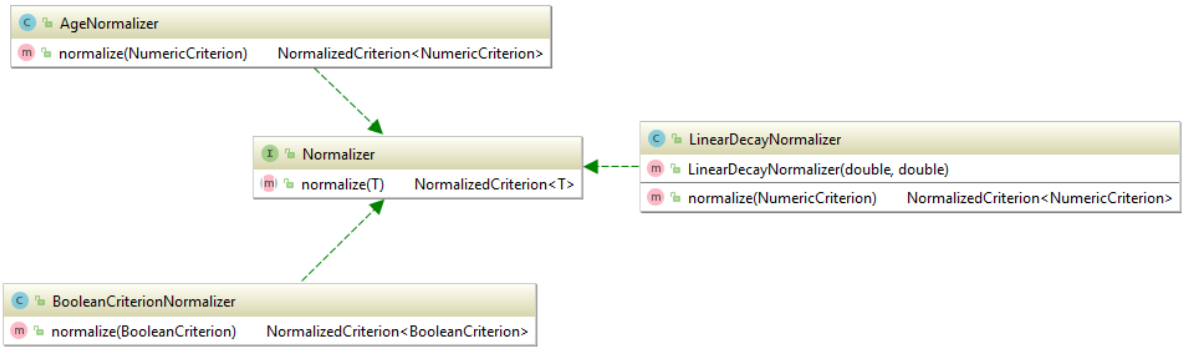
Kryteria dla poszczególnych decyzji generowane są na podstawie zgromadzonych danych o drużynach, zawodnikach oraz wynikach spotkań. Przykładową metodę przedstawiono na listingu 6.5.

Listing 6.5: Generacja kryterium różnicy wieku między dwoma drużynami

```
public AgeCriterion ageCriteria(Team hostTeam, Team otherTeam) {
  double averageAgeHostTeam = hostTeam.getPlayers().stream()
                .mapToDouble(playerService::getAge).average().orElse(0);
  double averageAgeOtherTeam = otherTeam.getPlayers().stream()
                .mapToDouble(playerService::getAge).average().orElse(0);

  return new AgeCriterion(
    averageAgeOtherTeam - averageAgeHostTeam,
    averageAgeHostTeam,
    averageAgeOtherTeam
  );
}
```

Normalizacja kryteriów

Normalizacja poszczególnych kryteriów przebiega przy użyciu różnych funkcji, dopasowanych do charakteru danych. Przykładem uzasadniającym konieczność zastosowania takiej modyfikacji może być porównanie dwóch kryteriów: odległości drużyn oraz średniego wieku zawodników. O ile kryterium odległości może być normalizowane w pełni liniowo, o tyle dla różnicy wieku taka metoda normalizacji jest błędna. Różnica wieku między dwoma zawodnikami, którzy mają 15 oraz 20 lat jest znacznie bardziej istotna aniżeli różnica między zawodnikami w wieku 30 oraz 35 lat - nie można tutaj zastosować operatora w pełni liniowego. Dodatkowo w domenie problemu wyróżniono kryteria nieliczbowe - cechy drużyny, które mogą mieć duży wpływ na jakość dopasowania, np. zadeklarowana chęć ponownej gry z daną drużyną.



Rys. 6.5: Hierarchia klas: normalizacja

Metoda ważonych kryteriów

Metoda ważonych kryteriów polega na opisaniu funkcji oceny decyzji jako sumy ważonej ocen poszczególnych kryteriów [1]. Konieczne jest przyporządkowanie wagi dla każdego z kryteriów. Ocena poszczególnych decyzji obliczana jest według wzoru:

$$F(x) = \sum_{i=1}^{k} w_i f_i(x) \tag{6.1}$$

gdzie k - ilość kryteriów, x - wektor rozwiązań, w_i - wagi takie że:

$$w \in [0,1] \text{ oraz } \sum_{i=1}^{k} w_i = 1$$

Metoda ta została wybrana ze względu na możliwość dynamicznego doboru wag poszczególnych kryteriów. Niektóre z tych wag będą dobierane przez kapitana zgodnie z preferencjami jego drużyny.

Listing 6.6: Przykładowa rejestracja kontrolera

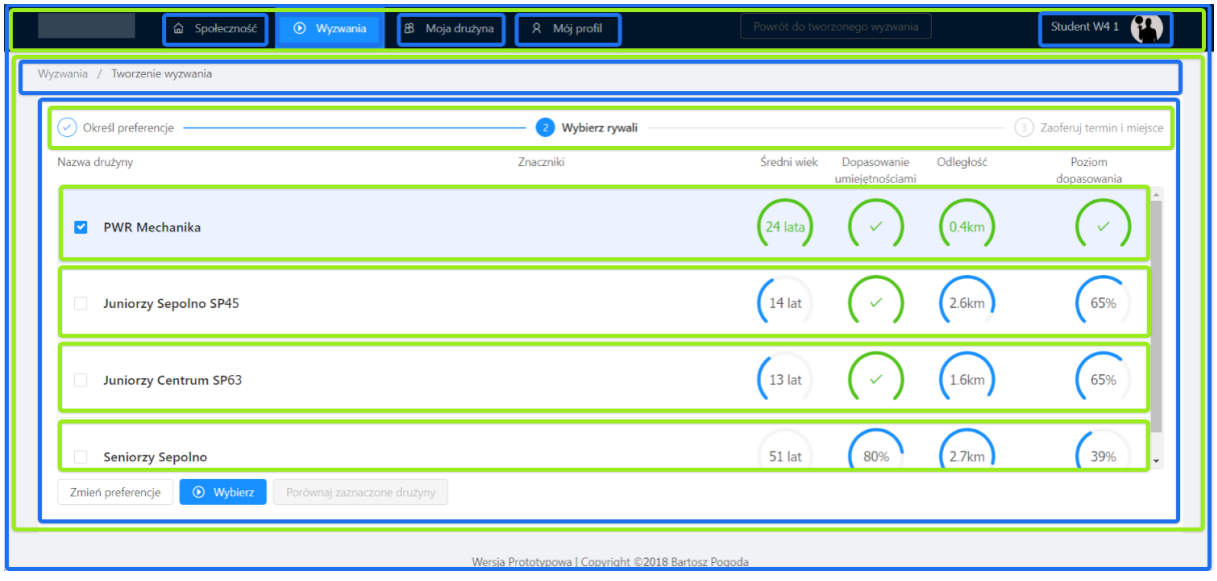
```
@Service
public class WeightedCriteriaAggregator {

    public double aggregate(List<WeightedCriteria> weightedCriteria) {
      return aggregate(weightedCriteria.stream());
    }

    public double aggregate(Stream<WeightedCriteria> stream) {
      return stream
      .mapToDouble(
       crit -> crit.getWeight() * crit.getCriteria().getNormalizedValue()
       )
      .sum();
    }

}
```

6.3. Implementacja aplikacji klienckiej

6.3.1. Modularność

Podobnie jak aplikacja serwerowa, aplikacja kliencka została zrealizowana w architekturze ograniczającej odpowiedzialność poszczególnych komponentów. W technologii Angular modularność uzyskuję się poprzez podział na komponenty oraz moduły, które je agregują w zbiory [X].

Istotnym wzorcem jaki został zastosowany w projekcie jest dodatkowy podział komponentów na komponenty prezentacyjne oraz kontenery. Komponenty prezentacyjne zajmują się jedynie wyświetlaniem danych przekazanych im na wejściach oraz pobieraniem danych od użytkownika i przekazywaniem ich na wyjścia. Kontenery posiadają większy zakres odpowiedzialności - znają stan aplikacji oraz mogą na niego wpływać. Kontenery używają komponentów prezentacyjnych do interakcji z użytkownikiem [X]. Zastosowanie takiego podziału usprawniło rozwój aplikacji oraz pozwoliło na używanie raz zaimplementowanych komponentów prezentacyjnych w różnych miejscach aplikacji.

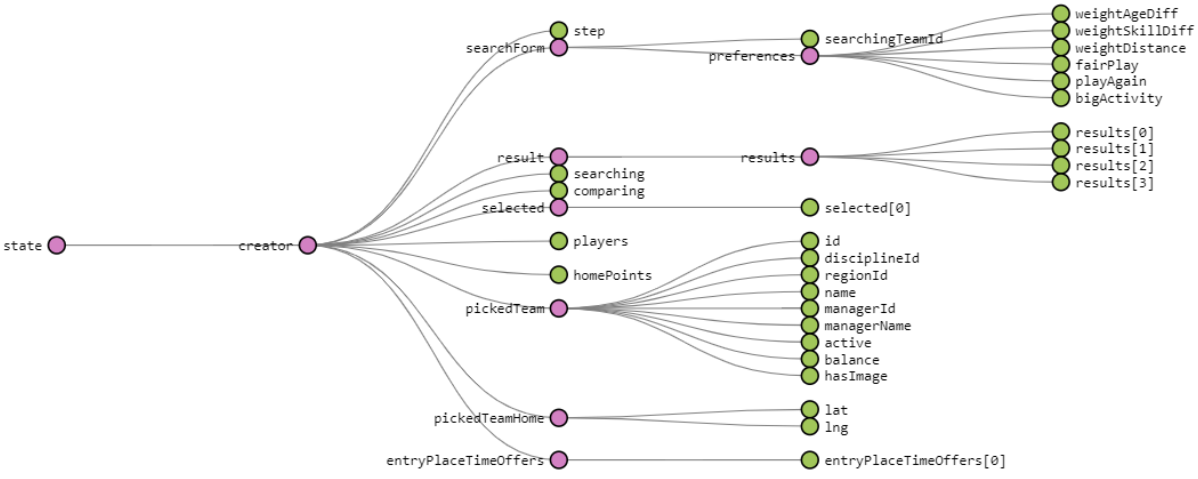


Rys. 6.6: Widok wyników poszukiwania rywali: Podział na komponenty

6.3.2. Stan aplikacji

Jednym z wyzwań podczas implementacji aplikacji działających w przeglądarce jest zarządzanie ich stanem. Tradycyjnym podejściem dla aplikacji w technologii Angular jest przetrzymywanie danych w komponentach oraz serwisach. Sposób ten działa bez zarzutów dla bardzo małych aplikacji, jednak w miarę rozwoju pojawiają się problemy takie jak: przekazywanie danych pomiędzy komponentami, które nie zależą bezpośrednio od siebie oraz problemy z synchronizacją danych. Drugi problem dotyczy sytuacji, w której dwa komponenty korzystają z tych samych obiektów, a w pewnym momencie okazuje się, że obiekty te są różne. Ciężko jest stwierdzić, który komponent zawiera referencje do obiektu prawdziwego - aktualnego. Rozwiązaniem opisanych problemów jest wdrożenie do aplikacji koncepcji *Store*. Podejście to polega na przetrzymywaniu stanu aplikacji w jednym obiekcie. Eliminuje to problem konfliktów, w aplikacji występuje tylko jedno źródło prawdy (ang. *Source of truth*). Korzystanie przez różne komponenty z tych samych danych również staje się proste ze względu na przetrzymywanie ich w centralnym miejscu dostępnym z każdego miejsca aplikacji.

Warto również dodać, że stan aplikacji jest obiektem niemodyfikowalnym. Jedynym sposobem na zmianę stanu jest zmiana referencji na nowy obiekt. W opisywanej architekturze zajmują się tym specjalne funkcje redukujące. Funkcje te na podstawie zgłaszanych akcji dokonują aktualizacji stanu. Akcje są zgłaszane w momencie zdarzeń wygenerowanych przez użytkownika, zewnętrzne systemy bądź wewnętrznie w aplikacji. Jedną z głównych zalet takiego sposobu zmian stanu jest zwiększenie wydajności aplikacji w technologii *Angular*. Komponenty mogą korzystać z bardzo wydajnego trybu odświeżania, który aktualizuje widok jedynie w momencie zmiany referencji podanych na jego wejścia - co ma miejsce przy aktualizacjach w zastosowanej architekturze. Kolejną równie ważną zaletą jest znaczne ułatwienie wykrywania błędów oraz ich przyczyn. Istnieją narzędzia deweloperskie pozwalające na śledzenie stanu oraz jego zmian podczas działania aplikacji. Przykładowy podgląd części stanu w formie drzewa przedstawiono na rysunku 6.7. Narzędzia umożliwiają również podgląd obiektu stanu w formacie *JSON*. Przykładową sekwencję akcji generowanych oraz obsługiwanych przy logowaniu do aplikacji ukazano na rysunku 6.8.



Rys. 6.7: Przykładowy fragment drzewa stanu aplikacji

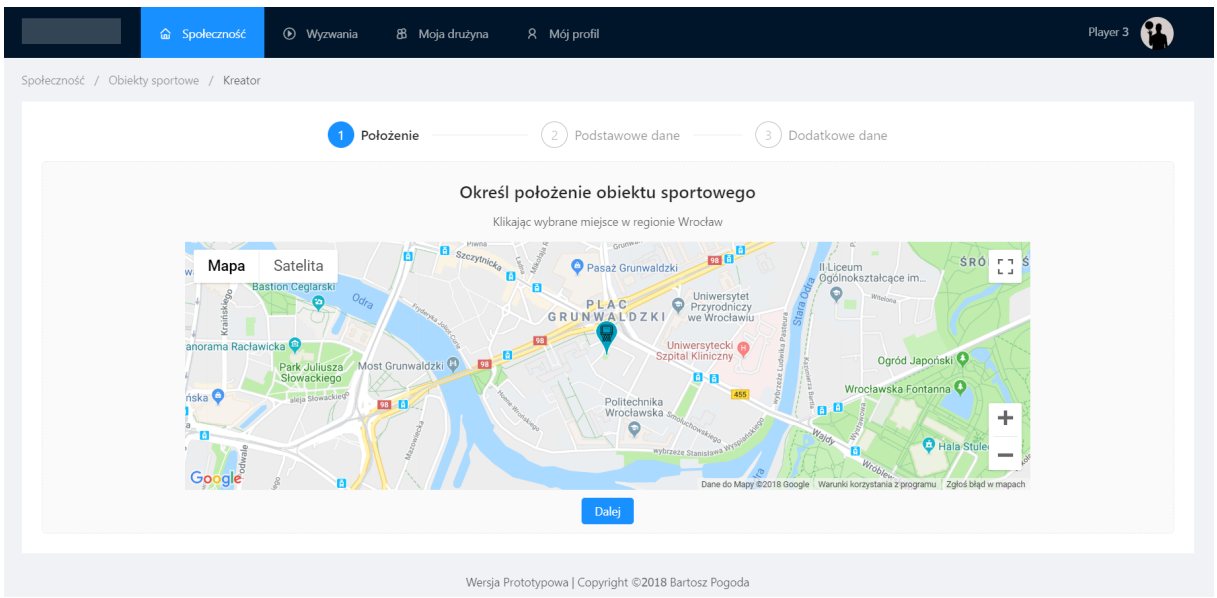
[Auth] Login	+00:06.63
[Auth] Generate Token Success	+00:00.46
[Auth] Decode Token	+00:00.01
[Auth] Decode Token Success	+00:00.01
[Auth] Login Success	+00:00.01
[Player] Load Current	+00:00.01
ROUTER_NAVIGATION	+00:00.01
[Player] Load Current Success	+00:00.17
[My Team] Load Current	+00:00.02
[My Team] Load Current Success	+00:00.17
[My Team] Update Is Manager	+00:00.01
[My Team] Load Home	+00:00:00
[My Team] Load Players	+00:00:00
[My Challenges] Load Active Challenges	+00:00:00
[My Challenges] Load Facilities	+00:00:00

Rys. 6.8: Przykładowa sekwencja akcji przy logowaniu do aplikacji

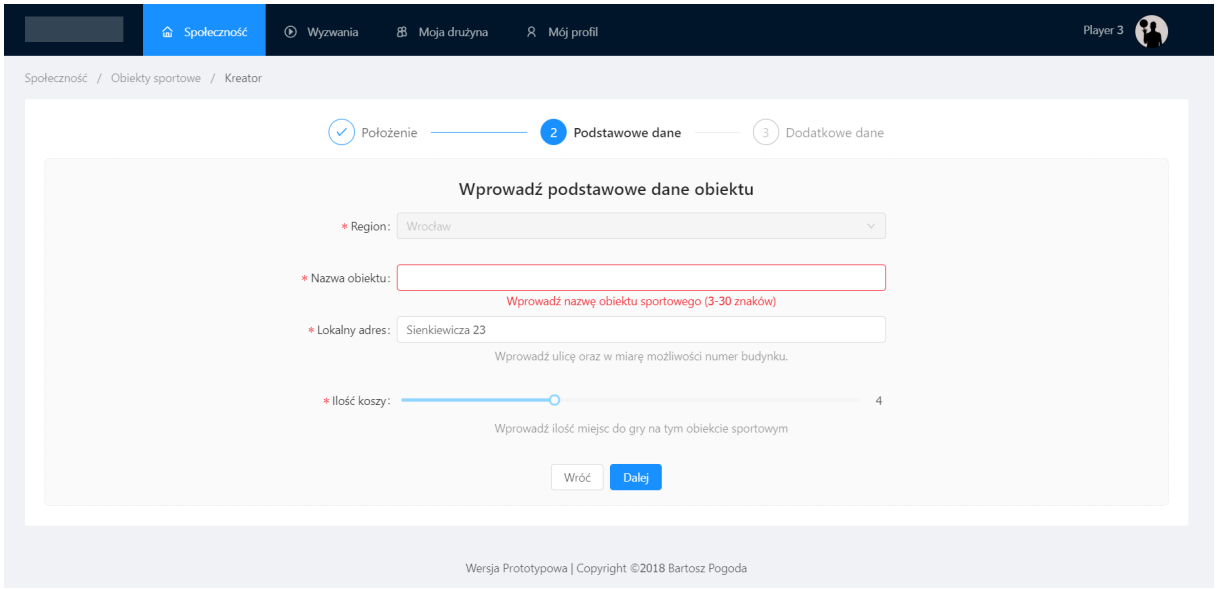
6.3.3. Formularze

Wiele funkcjonalności Team Challenge opiera się na pobraniu danych od użytkowników. W technologii *Angular* wyróżnia się dwa główne sposoby budowy formularzy - sterowane znacznikami *HTML* oraz reaktywne (ang. *reactive*). Podczas implementacji systemu użyto formularzy reaktywnych. Są one zalecane przez twórców szkieletu *Angular* ze względu na większą skalowalność oraz możliwości wielokrotnego użytku.

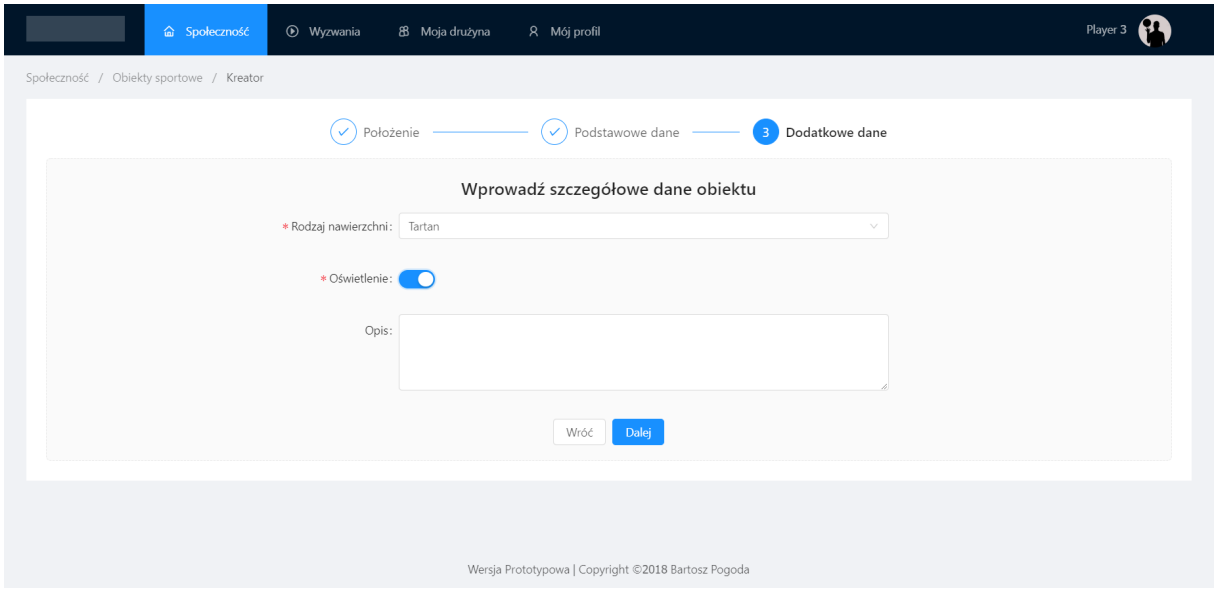
Komponenty wizualne takie jak przyciski, suwaki oraz pola tekstowe zostały dostarczone przez bibliotekę *NgZorro*.



Rys. 6.9: Formularz wprowadzania obiektu sportowego - Krok pierwszy



Rys. 6.10: Formularz wprowadzania obiektu sportowego - Krok drugi



Rys. 6.11: Formularz wprowadzania obiektu sportowego - Krok trzeci

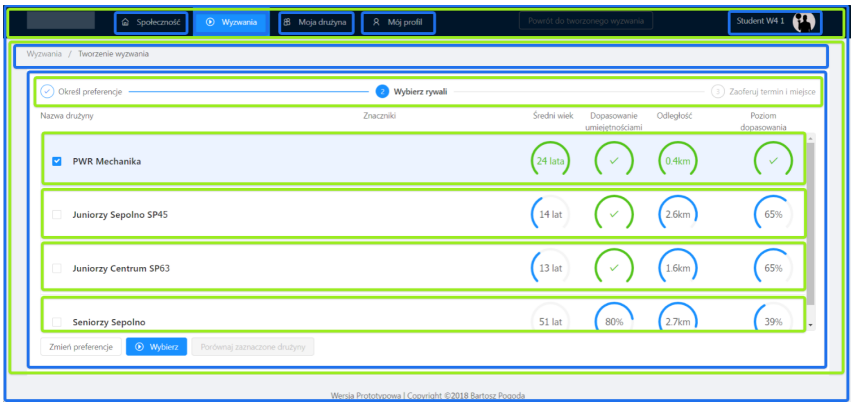
6.4. Bezpieczeństwo

Szczególną uwagę poświęcono implementacji zabezpieczeń systemu oraz zgromadzonych danych użytkowników. Funkcjonalności związane z bezpieczeństwem zostały zrealizowane bazując na module Spring Security, który dostarcza wiele przydatnych mechanizmów oraz możliwości ich konfiguracji.

Hasła użytkowników przechowywane są w formie zaszyfrowanej za pomocą funkcji skrótu BCrypt. Spring Security dostarcza klasę implementującą ten algorytm - BCryptPasswordEncoder. Główną zaletą takiego sposobu szyfrowania jest generowanie losowego ciągu znaków, tak zwanej soli, który dodatkowo wzmacnia bezpieczeństwo hasła. Zastosowanie soli przy haszowaniu znacznie utrudnia złamanie hasła przez ataki przy użyciu tęczowych tabel. W przypadku algorytmu bcrypt wygenerowana sól jest przechowywana razem z zaszyfrowanym hasłem w

bazie danych. W przypadku prób logowania jest ona wyciągana z bazy oraz używana do przetworzenia podanego hasła w celu porównania zgodności.

Autoryzacja zapytań (JWT)



Rys. 6.12: Hierarchia klas: normalizacja

Rozdział 7

Ocena użyteczności

Coś ogólnie o tym że ważna jest weryfikacja a w systemach webowych ważne jest badanie użyteczności interfejsu no albo to niżej że poddany dwokrotnie to tutaj opisać że nastąpiła taka potrzeba

7.1. Metodyka badań

System został dwukrotnie poddany weryfikacji środowiskowej. Pierwszy etap badań odbył się po ukończeniu implementacji pierwszego prototypu systemu. Drugi etap odbył się po ukończeniu drugiego prototypu, który poprawiał niedoskonałości odkryte podczas pierwszego badania.

Do weryfikacji wytypowane zostały dwie funkcjonalności istotne dla działania systemu. Pierwszą z nich jest proces, który musi przejść każdy nowy użytkownik systemu, czyli tworzenie profilu zawodnika. Drugą z wybranych funkcjonalności jest dopasowywanie rywali oraz rzucanie im wyzwania. Jest to najważniejsza funkcja systemu zawierająca elementy interfejsu wymagające weryfikacji użyteczności.

Badania odbyły się w formie badań moderowanych[?], czyli z udziałem osoby nadzorującej ich przebieg. Moderatorem w przypadku wszystkich badań był autor tej pracy. Rolą moderatora w przypadku takich badań jest obserwacja akcji podejmowanych przez osobę wykonującą zadania w systemie oraz sporządzanie notatek.

Do badań zostały wybrane osoby potencjalnie zainteresowane tematem, czyli osoby uprawiające sporty zespołowe. W celu zbadania użyteczności systemu dla różnych grup wiekowych zaproszono osoby w przedziale od 15 do 35 lat. Wśród ankietowanych były osoby profesjonalnie zajmujące się systemami webowymi jak również osoby spoza tej branży. Zgodnie z zaleceniami odnalezionymi w literaturze, w każdym z etapów badań wzięło udział pięciu respondentów[?].

Dla uczestników badania została przygotowana ankieta składająca się z trzech części. Pierwsza część była ankietą wstępną uzupełnianą przed badaniami. Zawierała ona krótkie wprowadzenie do systemu oraz pytania<>. Kolejne dwie części zawierały opisy zadań przygotowanych dla użytkowników oraz pytania kontrolne dotyczące intuicyjności poszczególnych procesów.

Uczestnicy badania zostali poinformowani o tym, że obiektem badania jest interfejs systemu, a nie oni sami. Ze względu na obecność moderatora osoby uczestniczące w badaniu zostały poproszone o głośne myślenie oraz wyrażanie uwag. Moderator podczas badań na bieżąco notował istotne akcje podejmowane przez użytkowników oraz ich uwagi. Po wykonaniu zadań miała miejsce krótka dyskusja na temat napotkanych problemów w obsłudze oraz potencjalnych usprawnień. Na podstawie notatek moderatora zostały sporządzone protokoły (odwołanie do załącznika) oraz wyciągnięte wnioski dotyczące dalszego rozwoju interfejsu.

7.2. Przebieg i wyniki badań

W pierwszej kolejności do systemu zostały wprowadzone przykładowe (zmyślone) dane zawodników, drużyn oraz obiekty sportowe. Umożliwiło to wykonanie testów wymagających interakcji z innymi obiektami w systemie.

W poniższych sekcjach został opisany przebieg oraz wyniki badań poszczególnych funkcjonalności systemu. Szczegółowe wyniki w postaci złożonych ankiet oraz sporządzonych protokołów można znaleźć w załączniku (odniesienie).

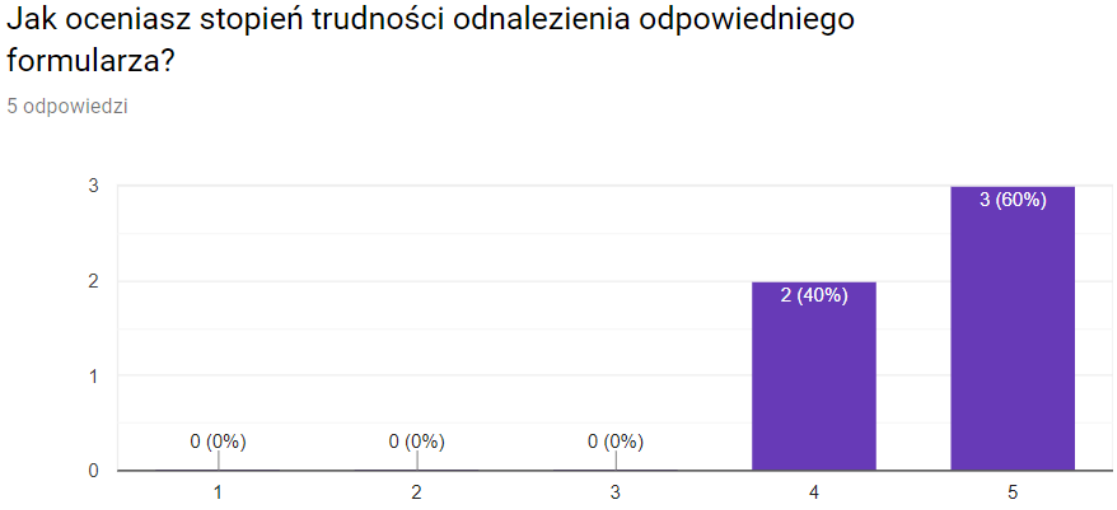
7.2.1. Tworzenie profilu zawodnika

W ramach pierwszego zadania ankietowani zostali zalogowani na przygotowane konta użytkowników w systemie, a następnie poproszeni utworzenie profilu zawodnika. Krokami prowadzącymi do wykonania zadania było odnalezienie odpowiedniego formularza a następnie wypełnienie go.

Poniżej zostaną opisane czynności, na których użyteczność moderator zwracał szczególną uwagę podczas badania.

Odnalezienie funkcjonalności

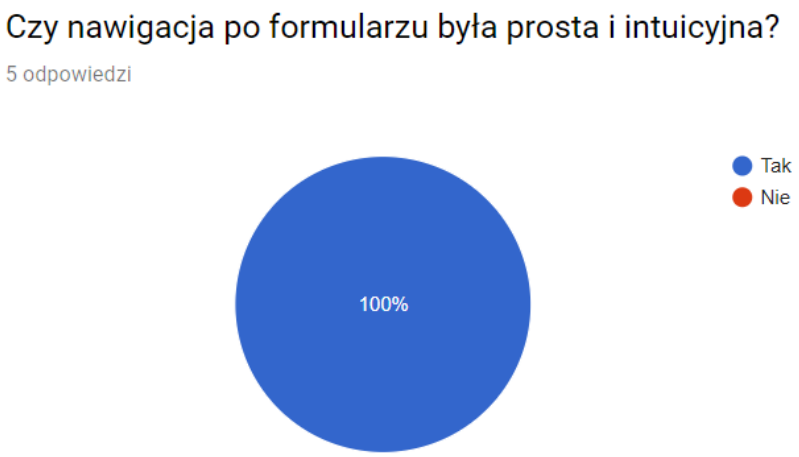
Użytkownicy nie mieli problemów z odnalezieniem kreatora zawodnika. Dwie osoby zasugerowały, że formularz mógłby pokazywać się od razu po pierwszym logowaniu do systemu. Uwaga ta została wzięta pod uwagę podczas implementacji drugiego prototypu.



Rys. 7.1: Zestawienie ocen trudności odnalezienia formularza tworzenia zawodnika po pierwszym etapie badań (1 - bardzo trudne, 5 - bardzo łatwe)

Nawigacja po formularzu

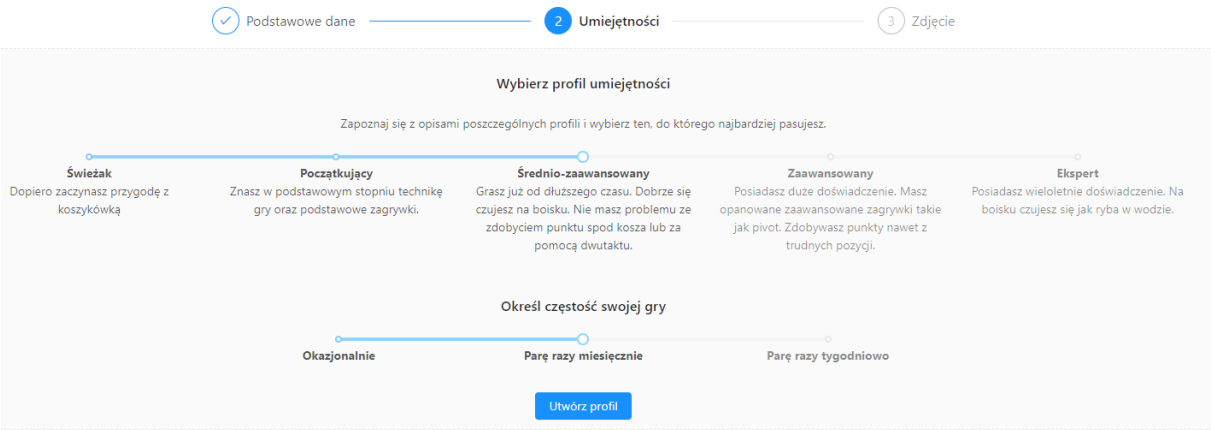
Nawigacja po formularzu składającym się z paru kroków nie sprawiła żadnych problemów respondentom.



Rys. 7.2: Aaaaaa

Deklaracja poziomu umiejętności

Również krok polegający na deklaracji poziomu umiejętności nie stanowił żadnych trudności. Każdy z ankietowanych był w stanie czytając opisy poszczególnych profili umiejętności dopasować się do jednego z nich. Dokonywanie wyborów poprzez przesunięcie suwaków było w pełni intuicyjne.



Rys. 7.3: Aaaaaa

7.2.2. Szukanie przeciwników i rzucanie wyzwania

Drugie zadanie

Odnalezienie funkcjonalności

O tym ze moglby byc odnosnik na stronie spolecznosci. O tym ze malo kto zwracal uwage na przycisk Szukaj rywaliźe wzgledu na jego pozycje.

Deklaracja preferencji wyszukiwania

O suwaczkach i ich problemach. O tym ze dodatkowe preferencje bez problemu.

Wyniki wyszukiwania

O tym ze jest ok ale wiek moglby byc bezwzgledny.

Oferty terminu oraz miejsca spotkania

Nawigacja

> Co było badane, > Widoki, > na co była zwracana uwaga > wyniki 1 iteracji > wnioski > co poprawione > wyniki 2 iteracji

Rozdział 8

Perspektywy rozwoju

Podczas trwania prac nad projektem narodziło się wiele pomysłów mogących urozmaicić działanie platformy. W niniejszym rozdziale zostaną przedstawione wybrane kierunki dalszego rozwoju.

8.1. Dalszy rozwój interfejsu użytkownika

O pomyśle na dalsze badania itp.

8.2. Aplikacja mobilna

Naturalnym kierunkiem rozwoju w przypadku platformy internetowej jest dostarczenie użytkownikom aplikacji mobilnej. Rozwiązanie takie ułatwiłoby użytkownikom dostęp do systemu z różnych lokalizacji. Aplikacja również mogłaby za pomocą powiadomień informować użytkowników o nowych zdarzeniach dotyczących umawianych wyzwań.

8.3. Partnerskie obiekty sportowe

Kolejną perspektywą jest nawiązanie współpracy z partnerami w postaci zarządców obiektów sportowych z ograniczonym dostępem. Ideą współpracy byłaby reklama obiektu sportowego w zamian za udostępnienie obiektu w określonych godzinach dla drużyn Team Challenge. Do systemu mógłby zostać wprowadzony nowy typ obiektów sportowych - obiekty partnerskie, wymagające uprzedniej rezerwacji. Należałoby również ograniczać dostęp do takich obiektów, na przykład poprzez wprowadzenie wirtualnej wewnątrz-systemowej waluty w postaci tokenów, którymi drużyny płaciłyby za rezerwację obiektów. Drużyny mogłyby otrzymywać tokeny co jakiś czas oraz za zakończone wyzwania.

8.4. System rankingowy

Interesującym kierunkiem rozwoju jest zdefiniowanie systemu rankingowego drużyn. Wyniki rozgrywek pomiędzy drużynami mogłyby wpływać na pozycję rankingową drużyny. Pozycja rankingowa drużyny mogłaby stanowić nowe kryterium dla algorytmu podczas poszukiwania rywali. Przykładowym systemem rankingowym, który mógłby znaleźć tutaj zastosowanie jest system ELO, używany w rozgrywkach szachowych oraz wielu grach komputerowych online.

Rozdział 9

Podsumowanie

Literatura

[1] dr hab. Mieczysław Połoński prof. SGGW. Analiza wielokryterialna –wstęp do zagadnienia. http://mieczyslaw_polonski.users.sggw.pl/Analiza%20wielokryter%20wstep1.pdf. [Online; dostęp 2 listopada 2018].

[2] mgr inż. Leszek Siwik, K. Cichosz, T. Borek. Wprowadzenie do optymalizacji wielokryterialnej. https://www.cs.put.poznan.pl/mkomosinski/materialy/optymalizacja/WieleKryteriow.ppt. [Online; dostęp 2 listopada 2018].

[3] O. Semusheva. Requirements. Why is it important? https://steelkiwi.com/blog/requirements-why-it-important/. [Online; dostęp 15 listopada 2018].

[4] S. Swafford. Use Cases and Their Importance. http://aspalliance.com/765_Use_Cases_and_Their_Importance, 2006. [Online; dostęp 15 listopada 2018].

Dodatek A

Instrukcja obsługi

Dodatek B

Opis załączonej płyty CD/DVD

Tutaj jest miejsce na zamieszczenie opisu zawartości załączonej płyty. Należy wymienić, co zawiera.