

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:

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



Szablon jest udostępniany na licencji Creative Commons: *Uznanie autorstwa – Użycie niekomercyjne – Na tych samych warunkach, 3.0 Polska*, Wrocław 2016.

Oznacza to, że wszystkie zawarte nim treści można kopiować i wykorzystywać do celów niekomercyjnych, a także tworzyć na ich podstawie utwory zależne pod warunkiem podania autora i nazwy licencjodawcy oraz udzielania na utwory zależne takiej samej licencji. Tekst licencji jest dostępny pod adresem: <http://creativecommons.org/licenses/by-nc-sa/3.0/pl/>.

Spis treści

Spis rysunków	5
Spis tabel	7
Spis fragmentów kodu	8
1. Wstęp	9
1.1. Wprowadzenie	9
1.2. Cel i zakres pracy	10
1.3. Układ pracy	10
2. Istniejące rozwiązania	11
2.1. Playarena.pl	11
2.2. SportsManago	12
2.3. SportsMatchMaker.com.au	12
2.4. Facebook	13
3. Koncepcja systemu Team Challenge	14
3.1. Baza wiedzy	14
3.1.1. Baza zawodników	14
3.1.2. Baza drużyn	14
3.1.3. Baza obiektów sportowych	15
3.2. Wspomaganie poszukiwania rywali	15
3.2.1. Kryteria dopasowania	15
3.2.2. Problem optymalizacji wielokryterialnej	16
3.3. Wspomaganie umawiania się na rozgrywkę	16
3.3.1. Negocjacja terminu oraz miejsca spotkania	17
3.3.2. Wyniki spotkań	17
3.3.3. Ocena drużyn	17
4. Projekt systemu Team Challenge	18
4.1. Role w systemie	18
4.2. Diagramy przypadków użycia	19
4.3. Szczegółowe opisy przypadków użycia	21
4.4. Wymagania niefunkcjonalne	25
4.5. Diagram związków encji	26
4.6. Diagram stanów - Wyzwanie	27
5. Architektura i technologie	28
5.1. Architektura systemu	28

5.1.1. RESTful API	29
5.2. Stos technologiczny	29
5.2.1. Java	29
5.2.2. Spring Boot	29
5.2.3. JWT	29
5.2.4. MySQL	30
5.2.5. Angular	30
5.2.6. Heroku	30
6. Implementacja systemu Team Challenge	31
6.1. Środowisko implementacji	31
6.2. Implementacja aplikacji serwerowej	31
6.2.1. Struktura projektu	31
6.2.2. Architektura wielowarstwowa	32
6.2.3. Warstwa repozytoriów	33
6.2.4. Warstwa serwisów	34
6.2.5. Warstwa zasobów	34
6.2.6. Algorytm poszukiwania rywali	35
6.3. Implementacja aplikacji klienckiej	38
6.3.1. Modularność	38
6.3.2. Stan aplikacji	39
6.3.3. Formularze	40
6.4. Bezpieczeństwo	42
6.5. Weryfikacja implementacji	42
7. Ocena użyteczności	43
7.1. Metodyka testów	43
7.2. Przebieg i wyniki badań	44
7.2.1. Tworzenie profilu zawodnika	44
7.2.2. Szukanie przeciwników i rzucanie wyzwania	45
8. Prezentacja działania systemu Team Challenge	48
8.1. Rejestracja i logowanie	48
8.2. Tworzenie profilu zawodnika	49
8.3. Tworzenie drużyny	51
8.4. Wyszukiwanie rywali i rzucanie wyzwań	53
8.5. Umawianie spotkań i wprowadzanie wyników	57
9. Perspektywy rozwoju	61
9.1. Kontynuacja testów użyteczności	61
9.2. Aplikacja mobilna	61
9.3. Partnerskie obiekty sportowe	61
9.4. System rankingowy	61
9.5. Zwiększenie możliwości zawodników	62
9.6. Czat	62
10. Podsumowanie	63
Literatura	64
A. Opis załączonej płyty CD	66

Spis rysunków

2.1. Przykładowy widok serwisu Playarena.pl	11
2.2. Przykładowy widok systemu SportsMatchMaker.com.au	12
2.3. Przykładowy post użytkownika szukającego towarzyszy do gry na portalu Facebook	13
3.1. Ogólny schemat metody rozwiązywania problemu optymalizacji wielokryterialnej	16
3.2. Koncepcyjny diagram negocjacji przy użyciu puli ofert	17
4.1. Diagram zależności między grupami w systemie	18
4.2. Diagram przypadków użycia - gość oraz użytkownik	19
4.3. Diagram przypadków użycia - zawodnik	20
4.4. Diagram przypadków użycia - moderator oraz administrator	20
4.5. Diagram przypadków użycia - kapitan	21
4.6. Diagram związków encji	26
4.7. Diagram możliwych stanów wyzwań	27
5.1. Diagram ogólnej architektury systemu	28
6.1. Fragment struktury pakietów	32
6.2. Warstwy aplikacji serwerowej	32
6.3. Schemat działania algorytmu oceny decyzji	35
6.4. Hierarchia klas - kryteria	36
6.5. Hierarchia klas - normalizacja	37
6.6. Widok wyników poszukiwania rywali - podział na komponenty	38
6.7. Przykładowy fragment drzewa stanu aplikacji	39
6.8. Przykładowa sekwencja akcji przy logowaniu do aplikacji	40
6.9. Formularz wprowadzania obiektu sportowego - krok pierwszy	40
6.10. Formularz wprowadzania obiektu sportowego - krok drugi	41
6.11. Formularz wprowadzania obiektu sportowego - krok trzeci	41
7.1. Zestawienie ocen trudności odnalezienia formularza tworzenia zawodnika po pierwszym etapie badań (1 - bardzo trudne, 5 - bardzo łatwe)	44
7.2. Widok po zalogowaniu się kapitana do systemu	45
7.3. Formularz deklaracji preferencji przed zmianami - I prototyp	46
7.4. Formularz deklaracji preferencji po zmianach - II prototyp	46
7.5. Zestawienie ocen intuicyjności deklaracji preferencji wyszukiwania (1 - nie intuicyjne, 5 - w pełni intuicyjne)	47
8.1. Widok strony głównej systemu	48
8.2. Widok formularza rejestracji	49
8.3. Widok formularza logowania	49
8.4. Widok kreatora zawodnika - krok pierwszy	49
8.5. Widok kreatora zawodnika - krok drugi	50

8.6. Widok kreatora zawodnika - krok trzeci	50
8.7. Widok profilu zawodnika	51
8.8. Widok zakładki "Moja drużyna"	51
8.9. Widok kreatora drużyny - krok pierwszy	51
8.10. Widok kreatora drużyny - krok drugi	52
8.11. Widok kreatora drużyny - krok trzeci	52
8.12. Widok kreatora drużyny - krok czwarty	53
8.13. Widok zakładki "Społeczność"	53
8.14. Widok formularza deklaracji preferencji drużyny	54
8.15. Widok wyników wyszukiwania rywali	54
8.16. Widok porównania potencjalnych rywali	55
8.17. Widok puli wstępnych ofert czasu oraz miejsca spotkania	55
8.18. Widok formularza tworzenia nowej oferty czasu i miejsca spotkania	56
8.19. Widok zaktualizowanej puli wstępnych ofert czasu oraz miejsca spotkania	56
8.20. Widok obecnych wyzwań	56
8.21. Widok zakładki "Wyzwania"	57
8.22. Widok szczegółów wybranego wyzwania w trakcie negocjacji	57
8.23. Widok negocjacji miejsca oraz czasu spotkania - w trakcie negocjacji	58
8.24. Widok negocjacji miejsca oraz czasu spotkania - po zaakceptowaniu	58
8.25. Widok szczegółów zaakceptowanego wyzwania	59
8.26. Widok formularza wprowadzania wyniku wyzwania	59
8.27. Widok zakładki "Wynik" w przypadku niezaakceptowanego wyniku	59
8.28. Widok szczegółów zakończonego wyzwania	60
8.29. Widok formularza oceny rywali	60

Spis tabel

2.1. Przykładowe grupy dla zawodników na Facebook - stan z dnia 20.11.2018r	13
6.1. Zestawienie narzędzi wykorzystywanych podczas implementacji systemu	31

Spis fragmentów kodu

6.1.	Fragment deklaracji klasy encyjnej Team	33
6.2.	Definicja repozytorium TeamRepository	33
6.3.	Fragment serwisu TeamService	34
6.4.	Fragment kontrolera TeamResource	35
6.5.	Generacja kryterium różnicy wieku między dwoma drużynami	36
6.6.	Klasa agregująca za pomocą metody ważonych kryteriów	38

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 poza indywidualnymi zdolnościami zawodników, ogromne znaczenie ma współpraca. Umiejętność współdziałania dla 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. Współzawodnictwo uczy przegrywać, wygrywać oraz szanować przeciwnika.

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, czyli traktując sport jako hobby, dodatek do życia, a nie jego główny kierunek i źródło utrzymania. 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, wspierają komunikację pomiędzy osobami uprawiającymi amatorsko sporty zespołowe.

W celu zapewnienia dużej dostępności rozwiązań, 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.

1.2. Cel i zakres pracy

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

- budowę oraz utrzymanie bazy zawodników,
- budowę oraz utrzymanie bazy drużyn,
- budowę oraz utrzymanie 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 zauważona 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, jaką jest wspieranie rywalizacji pomiędzy drużynami poprzez rzucańe 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 (ang. *Unified Modeling Language*) oraz uzupełniony o diagram związków encji.
- **Architektura i technologie** - Architektura systemu oraz przegląd wykorzystanych technologii wraz z motywacjami ich użycia.
- **Implementacja systemu Team Challenge** - Rozdział, w którym zawarto opis przebiegu implementacji systemu.
- **Ocena użyteczności** - Rozdział poświęcony testom użyteczności, którym został poddany system podczas rozwoju. Przedstawienie metodyki badań oraz ich wyników.
- **Prezentacja działania systemu Team Challenge** - Przybliżenie działania systemu z punktu widzenia użytkownika.
- **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 przedstawiono wybrane z istniejących rozwiązań wykorzystywanych w obszarze sportów zespołowych. Dla poszczególnych platform zostały wyszczególnione ich główne założenia oraz funkcjonalności wpływające na usprawnienie komunikacji pomiędzy sportowcami.

2.1. Playarena.pl

Rozwiązaniem, które w najbliższy sposób realizuje cele zdefiniowane we wstępnie tej pracy jest polski portal Playarena. Playarena.pl jest specjalistycznym portalem skierowanym do drużyn piłki nożnej 6-osobowej. Celem jego twórców jest zachęcanie ludzi do gry, dostarczanie możliwości rywalizacji, rozwoju oraz zabawy [3].

Po rejestracji w systemie użytkownicy mogą zakładać swoje drużyny lub dołączać do istniejących poprzez system aplikacji oraz zaproszeń. Głównym obszarem działania portalu są mecze ligowe, w których uczestniczyć mogą skompletowane drużyny. Kapitanowie drużyn mają możliwość rzucania wyzwań innym drużynom w swojej lidze. Najlepsze drużyny awansują do wyższych lig, a zawodnicy wypatreni przez łowców talentów mogą otrzymać powołanie do Reprezentacji Polski w piłce nożnej 6-osobowej. Przykładową tabelę ligową przedstawiono na rysunku 2.1.

Rozgrywki odbywają się na obiektach sportowych, które są zgłaszcane oraz weryfikowane przez użytkowników systemu.

The screenshot shows the Playarena.pl website interface. At the top, there is a navigation bar with the logo 'playarena.pl', a search bar, and various links like 'ZAWODNIKA', 'PROMOCJA R-GOL.com', and user icons. Below the header, the title 'Wrocław - 1 Liga' is displayed above a table of league results. The table has columns for 'Pozycja' (Position), 'Nazwa drużyny' (Team Name), 'M' (Wins), 'Z' (Draws), 'R' (Losses), 'P' (Points), 'Bramki' (Goals), and 'Punkty' (Points). The teams listed are: 1. Alkopoliagamia (12 wins, 9 draws, 1 loss, 2 points, 178:99 goals, 28 points); 2. Soccer Punch (12 wins, 9 draws, 1 loss, 2 points, 132:93 goals, 28 points); 3. Przyjaciele z boiska (11 wins, 8 draws, 1 loss, 2 points, 133:88 goals, 25 points); 4. Orły Wrocław (8 wins, 8 draws, 0 losses, 0 points, 96:41 goals, 24 points); and 5. FC 78 WROCŁAW (11 wins, 7 draws, 2 losses, 2 points, 92:67 goals, 23 points). To the right of the table, there is a sidebar with a banner for 'SEZON12 - NOWY KOD RAB...', a message about a coupon code, and a timer indicating the offer ends at 01.09.2018 00:00.

Pozycja	Nazwa drużyny	M	Z	R	P	Bramki	Punkty
1.	Alkopoliagamia	12	9	1	2	178 : 99	28
2.	Soccer Punch	12	9	1	2	132 : 93	28
3.	Przyjaciele z boiska	11	8	1	2	133 : 88	25
4.	Orły Wrocław	8	8	0	0	96 : 41	24
5.	FC 78 WROCŁAW	11	7	2	2	92 : 67	23

Rys. 2.1: Przykładowy widok serwisu Playarena.pl

2.2. SportsManago

Na polskim rynku istnieją również rozwiązania wspomagające komunikację w obrębie amatorskich klubów sportowych oraz zarządzanie nimi. Jednym z takich rozwiązań jest *SportsManago*, czyli wielofunkcyjne narzędzie usprawniające przepływ informacji między menedżerami, trenerami, zawodnikami oraz ich rodicami. Pomyśłodawcą i wykonawcą projektu jest Witold Dyrur - Absolwent Wydziału Elektroniki Politechniki Wrocławskiej.

Narzędzie umożliwia gromadzenie informacji o klubie, kartotek jego zawodników, historii meczy wraz ze szczegółowymi statystykami. *SportsManago* wspiera między innymi:

- sprawdzanie obecności na treningach,
- prowadzenie naborów do klubu,
- zarządzanie obozami,
- zarządzanie zawodami.

Dużym usprawnieniem jest zastosowany system powiadomień SMS (ang. *Short Message Service*), który pozwala na szybkie dostarczanie informacji na temat treningów, meczów i innych wydarzeń.

2.3. SportsMatchMaker.com.au

Przykładem zagranicznego rozwiązania jest *SportsMatchMaker*, czyli Australijska sieć służąca do komunikacji pomiędzy sportowcami [4].

Działanie systemu przypomina tablicę ogłoszeń. Użytkownicy mogą deklarować się jako poszukujący osób do wspólnego uprawiania sportów. Ogłoszenie zawiera takie informacje jak: dyscyplina sportu, poziom zaawansowania oraz lokalizacja. Przeglądanie ofert polega na wyszukiwaniu z możliwością zawężenia wyników do określonej dyscypliny oraz lokalizacji. Przykładowe wyniki wyszukiwania osób grających w koszykówkę zostały przedstawione na rysunku 2.2. Nawiązanie kontaktu odbywa się za pomocą wiadomości prywatnych.

The screenshot shows a search results page for "Basketball" across "Australia wide". There are 2196 profiles found. The interface includes a search bar with filters for "Basketball", "Wpisz lokalizację", and "within 25 kms". Below the search bar, there's a "Search For People" button. The main area displays two profiles:

- Lucas #####**
Basketball -Advanced
Male, 24
28-Nov-18
Mernda VIC
Message preview: Hi I've been playing basketball for a long time Just come back from college ball in America. Stopped playing high level, would like to play regularly on Thursday night
- Michael #####**
Basketball -Intermediate
Male, 34
27-Nov-18
Melbourne VIC
Message preview: Have played basketball on and off for roughly 20 years. Used to play multiple times a week. My strengths would probably be defence and passing. Normally play a guard type role I'm 5'10 33 year old Looking to get back into playing again.. somewhere around Melbourne area.. happy to venture out a little if required. Looking for permanent spot on any night. Hoping to get at least 1-2 games a week.

Rys. 2.2: Przykładowy widok systemu SportsMatchMaker.com.au

2.4. Facebook

Poza specjalistycznymi systemami, których głównym celem jest wspieranie sportowców, istnieją również platformy, których głównych założenia mogą wydawać się odległe. *Facebook* jest serwisem społecznościowym zrzeszającym prawie 2 miliardy osób z całego świata. Główną misją *Facebook-a* jest zbliżanie do siebie ludzi poprzez umożliwianie budowy społeczności [15].

Społeczność osób uprawiających sporty zespołowe nie stanowi tutaj wyjątku. Na *Facebook-u* istnieje bardzo dużo grup tematycznych, których celem jest gromadzenie osób uprawiających pewną dyscyplinę sportu w określonym regionie. Osoby organizujące rozgrywki bardzo często tworzą posty podając takie informacje jak miejsce oraz termin spotkania, preferowany wiek oraz poziom umiejętności. Chętne osoby zgłaszały się pod postem lub poprzez wiadomość prywatną. Przykładowy post został przedstawiony na rysunku 2.3.



Rys. 2.3: Przykładowy post użytkownika szukającego towarzyszy do gry na portalu Facebook

Szukanie osób do gry poprzez portal *Facebook* jest bardzo często wybieranym rozwiązaniem głównie ze względu na dużą ilość użytkowników oraz szeroką dostępność serwisu. W tabeli 2.1 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

Rozdział 3

Koncepcja systemu Team Challenge

W niniejszym rozdziale została przybliżona koncepcja głównych funkcjonalności platformy *Team Challenge*. 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 zbierania 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 będzie rejestracja zawodników. Podczas procesu rejestracji od użytkownika zostaną pobrane dane niezbędne do funkcjonowania systemu.

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 założeniem w tym zakresie jest umożliwienie zawodnikom zgłaszania obiektów.

3.2. Wspomaganie poszukiwania rywali

Kluczową funkcjonalnością systemu będzie 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 oraz forma zawodników,
- mała odległość między punktami macierzystymi drużyn,
- zachowanie fair-play drużyny przeciwej,
- dobre wspomnienia po poprzednich rozgrywkach.

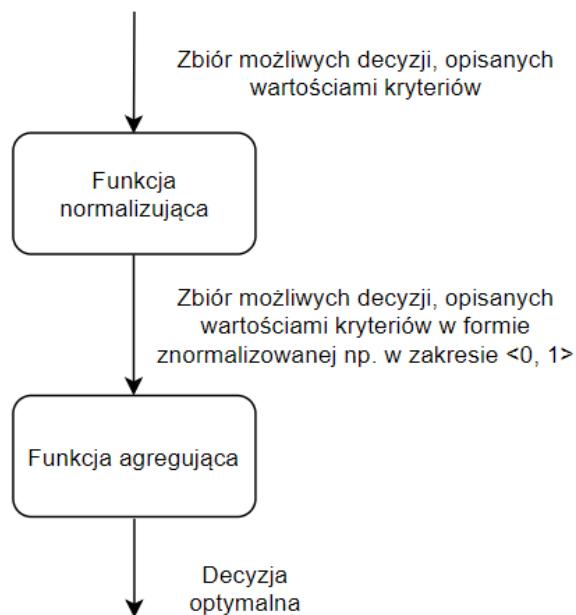
Istotne z punktu widzenia kapitana szukającego przeciwników mogą być również czynniki wpływające na możliwość umówienia spotkania. Mogą być to na przykład informacje dotyczące aktywności drużyny przeciwej, czy na przykład jej preferowane godziny gry.

Przytoczone kryteria powinny zostać uwzględnione w projektowanym mechanizmie wspomaganej 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 [8]. Jeżeli kryterium jest ilościowe, np. maksymalna prędkość samochodu, to podejmując decyzje o zakupie jedynie ze względu na to kryterium naturalnie wybierzymy samochód, który osiąga największą prędkość.

Często jednak podjęcie 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 między 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 rozwiązywania problemu jest w pierwszej kolejności znormalizowanie wartości kryteriów do przedziału wartości $<0,1>$, a następnie dokonanie wyboru korzystając z wybranej funkcji agregującej [8]. Ogólny schemat został przedstawiony na rysunku 3.1.

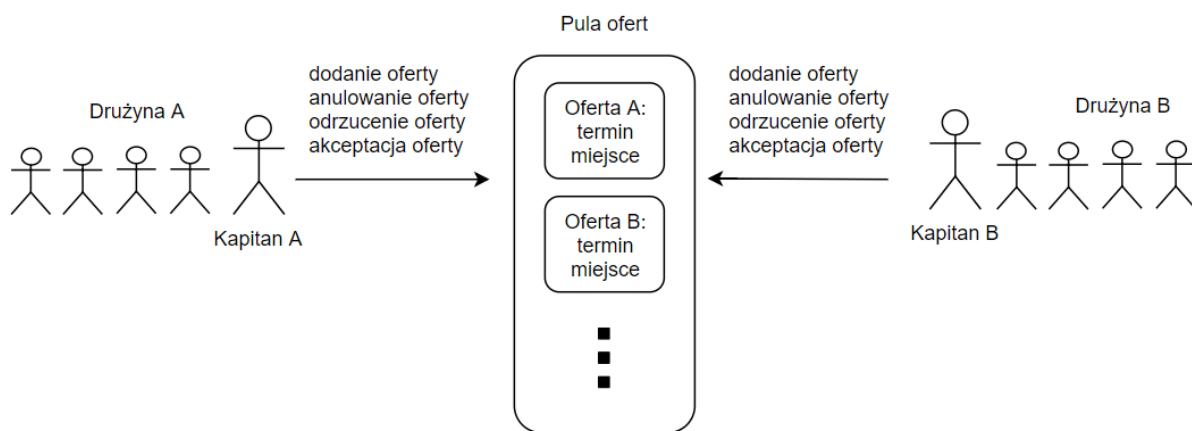
Problem podobnej natury występuje w projektowanym systemie. Poszukiwanie rywali do gry można sprowadzić do poszukiwania najlepszych decyzji wyboru drużyny spośród kwalifikujących się drużyn zdefiniowanych w systemie. Przykładowe kryteria wpływające na wartość funkcji oceny dla poszczególnych decyzji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

3.3. Wspomaganie umawiania się na rozgrywkę

Kolejną bazową funkcjonalnością systemu będzie wspieranie przebiegu umawiania spotkań. Kapitan aktywnej drużyny będzie miał możliwość rzucenia wyzwania innej aktywnej drużynie. Kapitan wyzwanej drużyny będzie mógł podjąć decyzję o odrzuceniu wyzwania. W przeciwnym wypadku rozpocznie się proces negocjacji terminu oraz miejsca spotkania.

3.3.1. Negocjacja terminu oraz miejsca spotkania

Kapitanowie drużyn, komunikując się ze swoimi zawodnikami, będą mogli 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. Negocjacje spotkania będą uznane za zakończone w momencie gdy jeden z kapitanów zaakceptuje propozycję drużyny przeciwnej. Termin oraz miejsce zawarte w zaakceptowanej ofercie staną się planowanym terminem oraz miejscem spotkania, a wyzwanie otrzyma status zaakceptowanego.



Rys. 3.2: Koncepcyjny diagram negocjacji przy użyciu puli ofert

3.3.2. Wyniki spotkań

Wyzwanie zostanie uznane za zakończone gdy nastąpi wprowadzenie do systemu jego wyniku. Wynik będzie 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. Wynikiem jest ilość punktów zdobytych przez poszczególne drużyny. W przypadku gdy są to dwie różne liczby wyzwanie zakończy się zwycięstwem drużyny, która zdobyła ich więcej. W przeciwnym razie wynikiem wyzwania będzie remis.

3.3.3. Ocena drużyn

Kapitanowie drużyn będą mieli możliwość 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

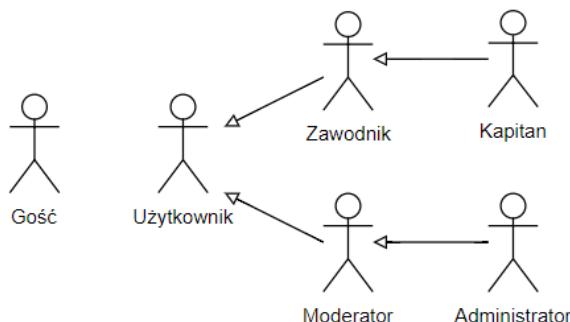
W niniejszym rozdziale przedstawiony został projekt systemu sporządzony na podstawie ogólnej koncepcji opisanej w poprzedniej części. W ramach projektu wydzielono role aktorów na stronie, następnie sporządzono specyfikację wymagań funkcjonalnych oraz niefunkcjonalnych. Na podstawie wymagań funkcjonalnych 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.

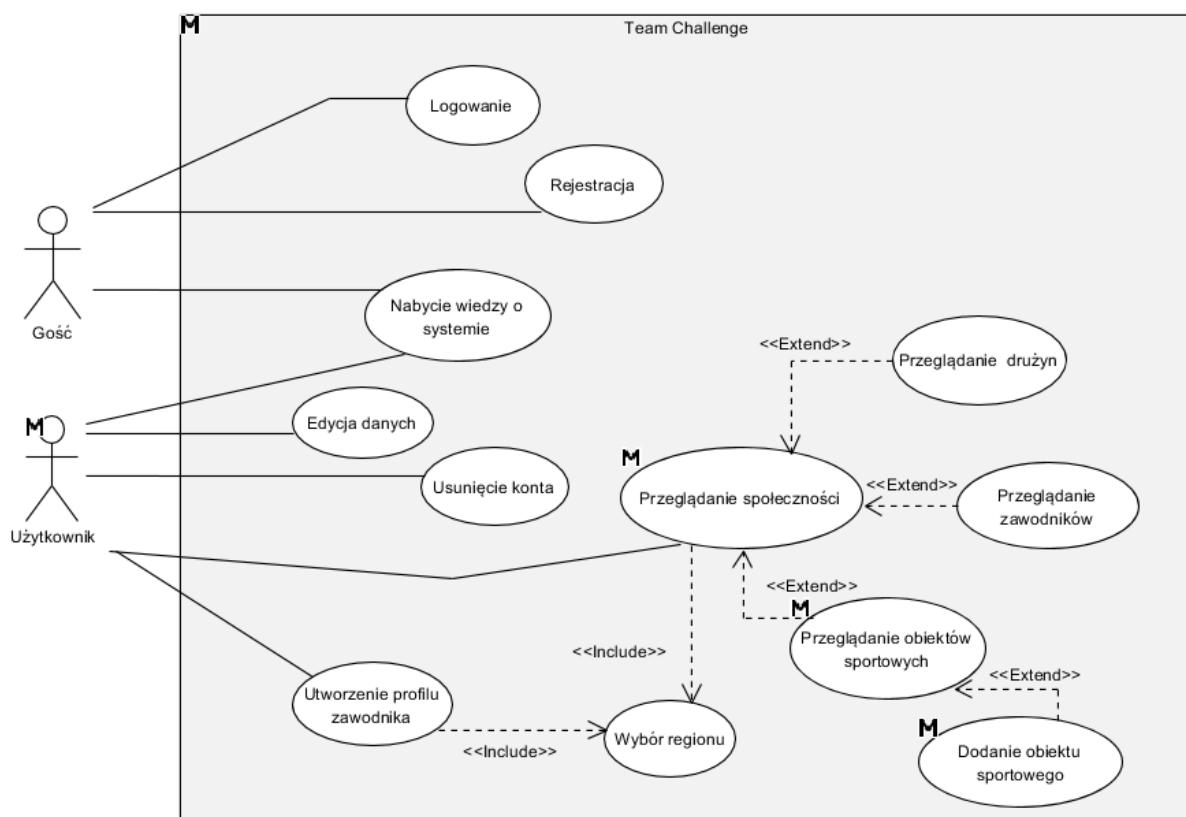


Rys. 4.1: Diagram zależności między grupami w systemie

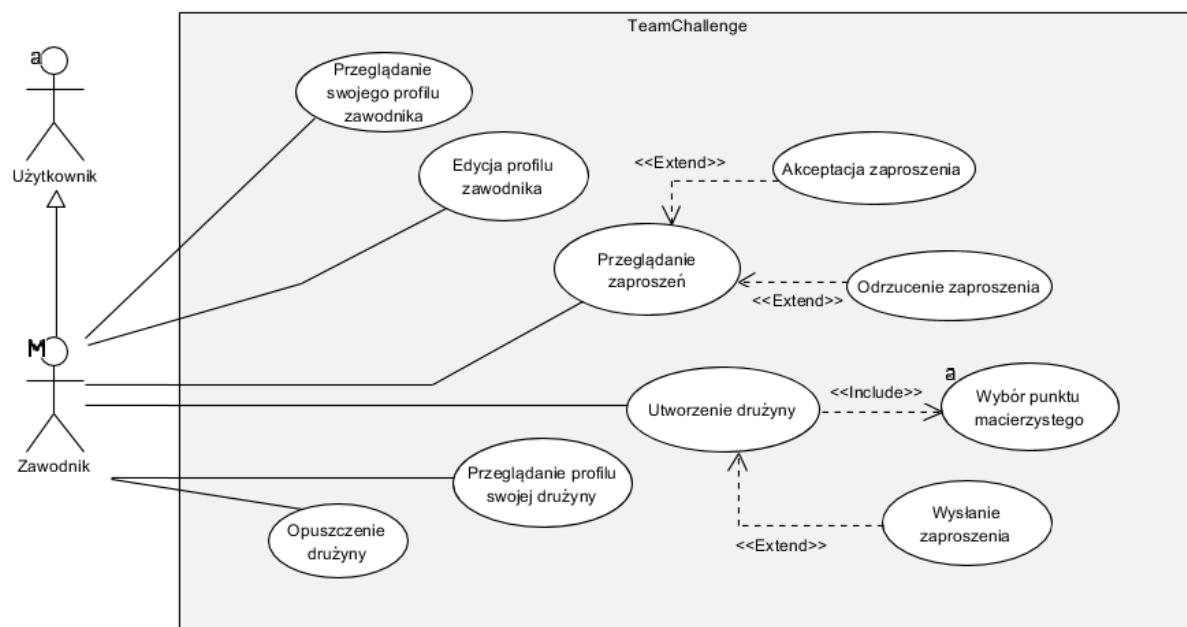
4.2. Diagramy przypadków użycia

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 [10]. Specyfikacja wymagań za pomocą przypadków użycia w sposób czytelny przedstawia możliwe interakcje poszczególnych aktorów z systemem [12].

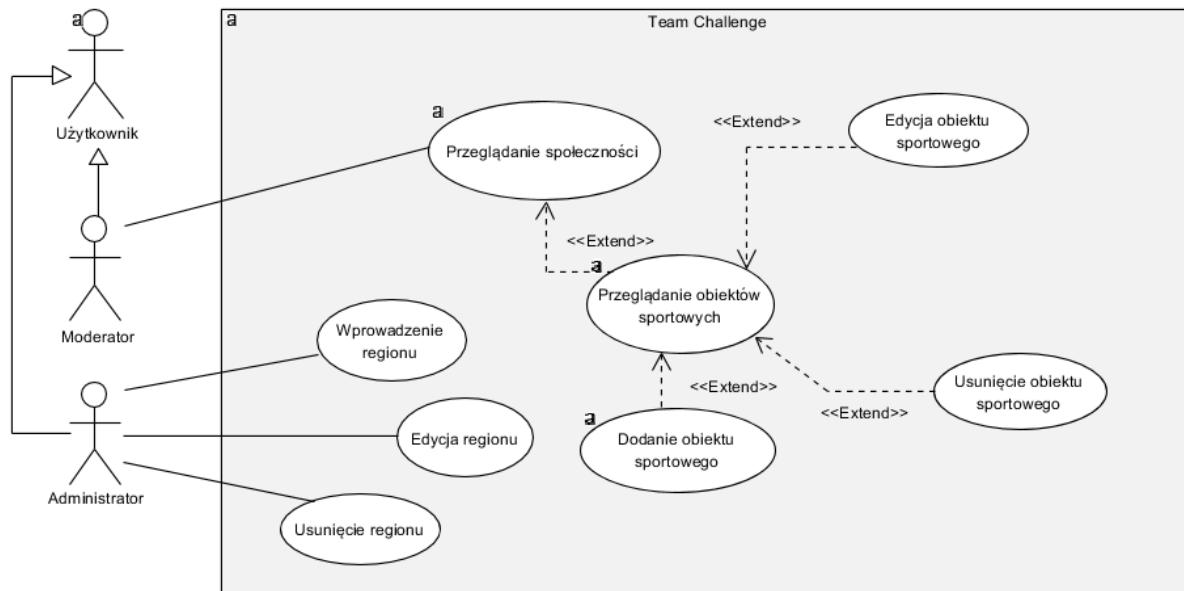
Poniżej przedstawiono diagramy przypadków użycia. W celu poprawienia czytelności, przypadki użycia dla niektórych aktorów zostały przedstawione na oddzielnych diagramach.



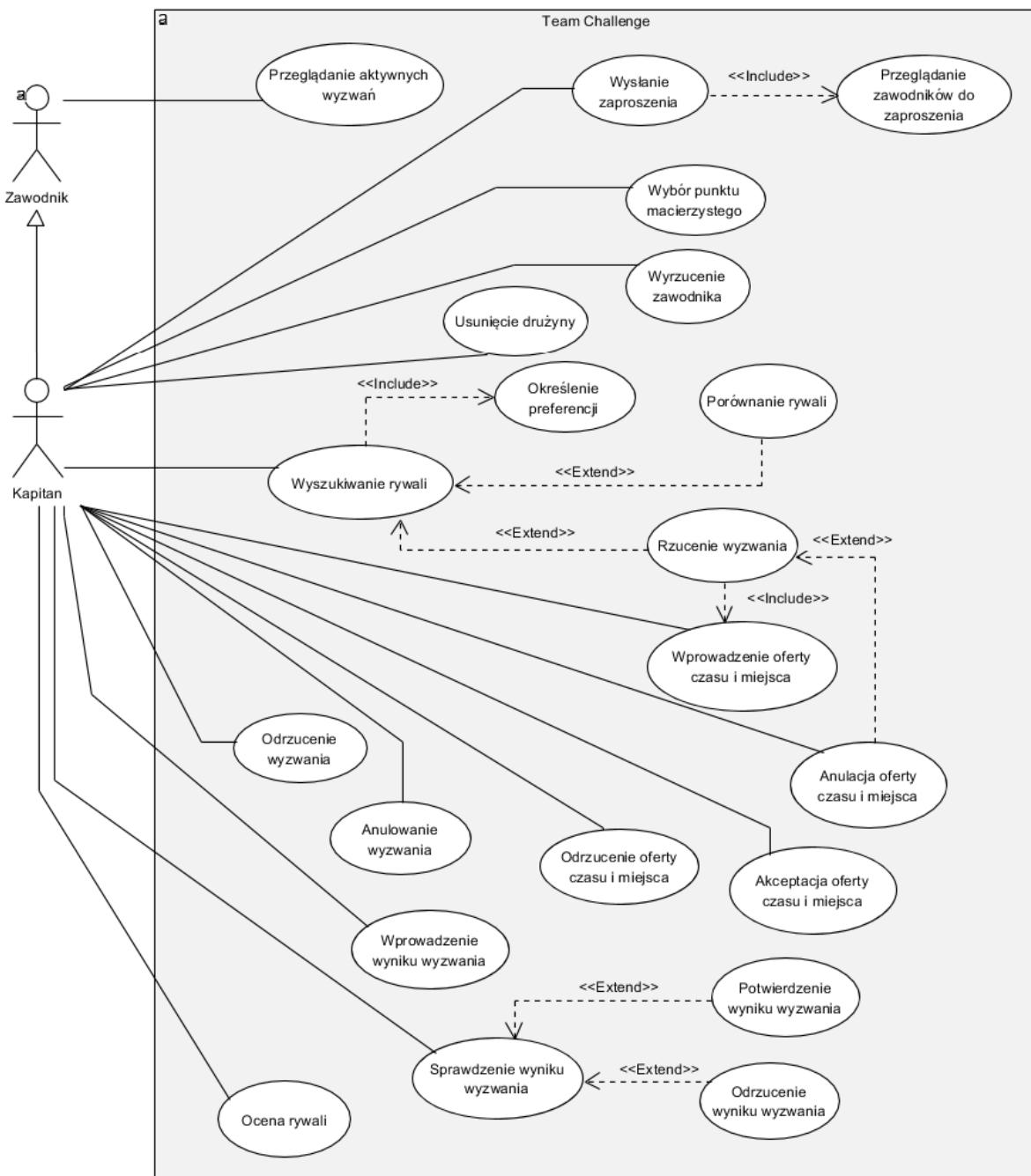
Rys. 4.2: Diagram przypadków użycia - gość oraz użytkownik



Rys. 4.3: Diagram przypadków użycia - zawodnik



Rys. 4.4: Diagram przypadków użycia - moderator oraz administrator



Rys. 4.5: Diagram przypadków użycia - kapitan

4.3. Szczegółowe opisy przypadków użycia

Dla wybranych spośród najbardziej istotnych przypadków użycia zostały sporządzone szczegółowe opisy zawierające warunki początkowe, końcowe oraz przebieg.

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 "Zarejestruj 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.

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 określaczęstość 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".

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**.

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**.

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.

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.

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 "Anuluj".
7. System zapisuje nową ofertę oraz wyświetla kapitanowi zaktualizowaną pulę zawierającą nowo dodany element.

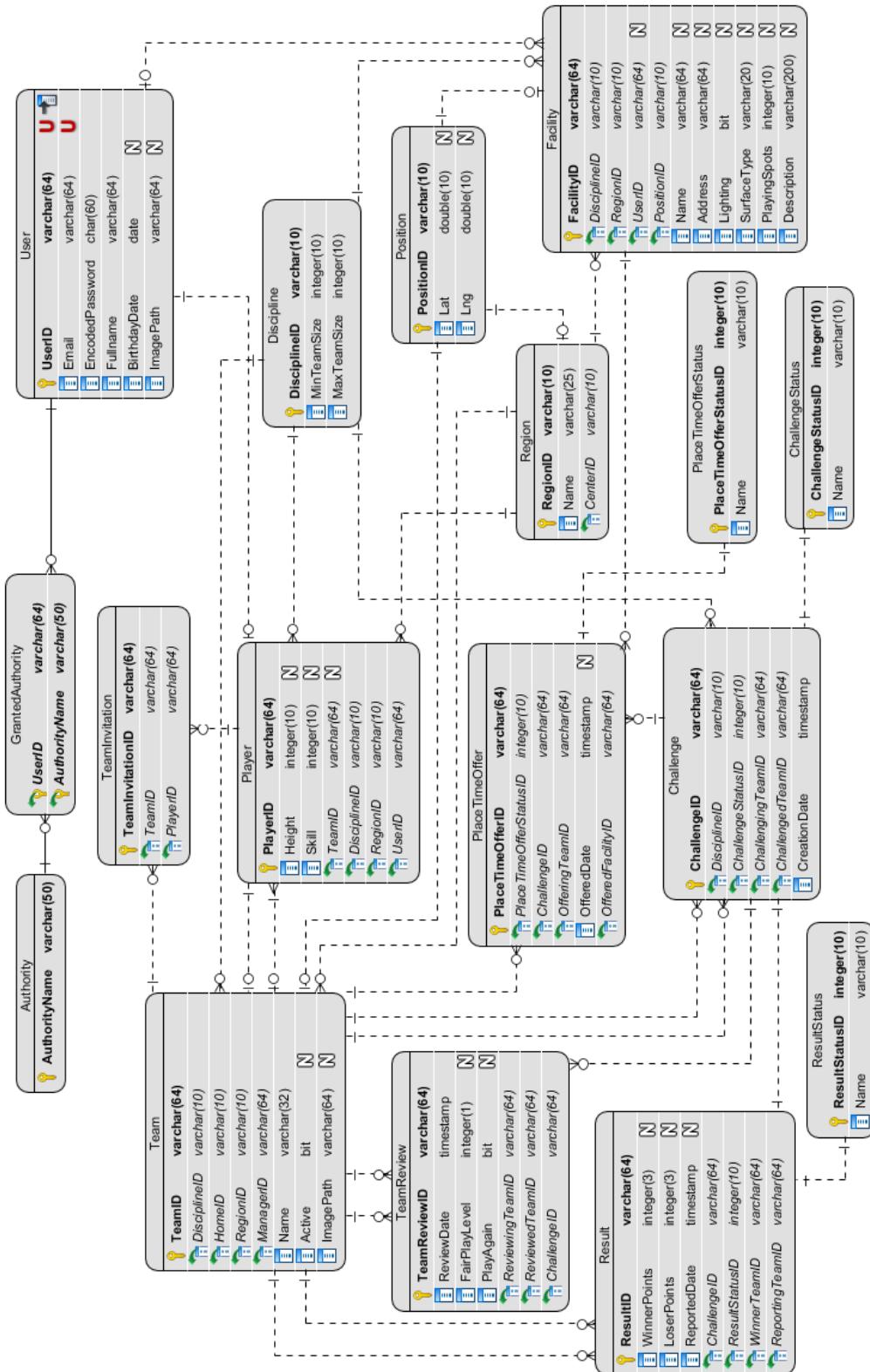
4.4. Wymagania niefunkcjonalne

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

1. Zastosowane technologie muszą 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 zrozumiałym. Fragmenty mogące budzić niejasności powinny być opisane za pomocą komentarzy.
4. System powinien być szeroko dostępny. Powinien udostępniać swoje funkcje użytkownikom korzystających z różnych platform wyposażonych w przeglądarkę internetową.
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.

4.5. Diagram związków encji

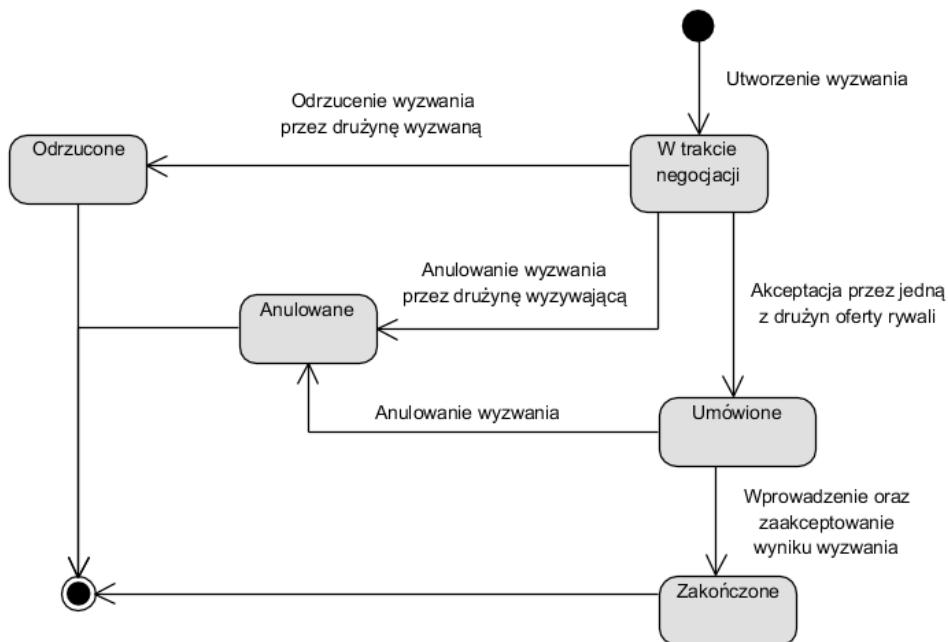
Na podstawie wymagań dotyczących funkcjonowania systemu zidentyfikowane zostały encje. Encje wraz ze swoimi atrybutami oraz powiązaniemi zostały przedstawione w sposób graficzny na rysunku 4.6 za pomocą diagramu ERD (ang. *Entity Relationship Diagram*).



Rys. 4.6: Diagram związków encji

4.6. 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

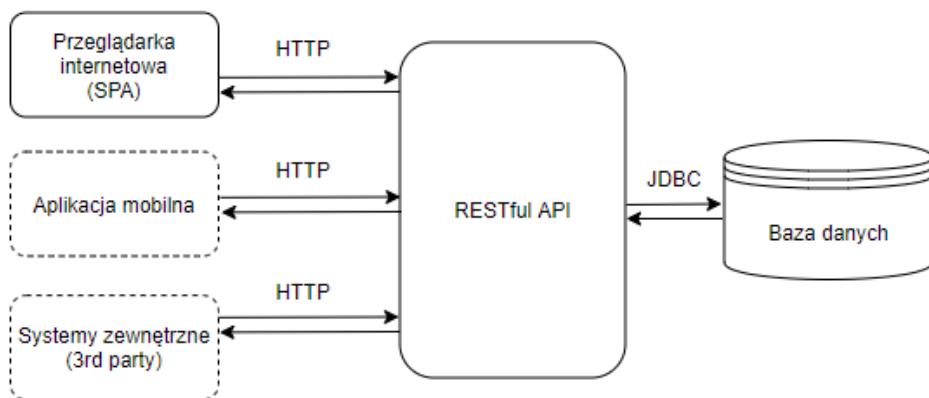
Architektura i technologie

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 (ang. *Single Page Application*) uruchamiana w przeglądarce internetowej. Aplikacja ta komunikuje się z API (ang. *Application Programming Interface*) aplikacji serwerowej za pomocą zapytań protokołu HTTP (ang. *Hypertext Transfer Protocol*). Aplikacja serwerowa z kolei komunikuje się z bazą danych w celu odczytu oraz zapisu informacji za pomocą interfejsu JDBC (ang. *Java DataBase Connectivity*).



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 (ang. *Representational State Transfer*), który zakłada komunikację klient-serwer z uwzględnieniem następujących zasad:

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

Użycie podstawowych metod protokołu HTTP pozytywnie wpływa na czytelność oraz intuicyjność API. Projektowanie z uwzględnieniem powyższych zasad pozwala również zminimizować 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

W poniższych sekcjach zostały przybliżone wybrane z zastosowanych technologii.

5.2.1. Java

Aplikacja serwerowa została zaimplementowana w języku *Java 1.8* [5]. *Java* jest bardzo często wybieranym językiem do budowy systemów internetowych, ze względu na duże wsparcie w postaci rozbudowanej biblioteki podstawowej, jak również dużą ilość darmowych narzędzi oraz bibliotek dostarczanych przez firmy trzecie. Język ten został wybrany również ze względu na doświadczenie akademickie oraz komercyjne autora niniejszej pracy, a także chęć dalszego rozwoju.

Wersja 1.8 została wybrana ze względu na jej stabilność oraz dostarczone usprawnienia w postaci wyrażeń *lambda*, przetwarzania strumieniowego oraz typu *Optional*. Wymienione mechanizmy wpłynęły na zwiększenie wydajności procesu programowania oraz ogólnej jakości uzyskanego kodu.

5.2.2. Spring Boot

Podstawą dla rozwoju aplikacji serwerowej był szkielet *Spring Boot* w wersji 2.0.3 [13]. Główną zaletą budowy aplikacji sieciowej w oparciu o gotowy szkielet jest możliwość koncentracji na implementacji logiki biznesowej. Wiele mechanizmów związanych z obsługą zapytań, niezawodnością, bezpieczeństwem, dostępem do danych staje się odpowiedzialnością szkieletu aplikacji, a nie programisty. Programista ma możliwość konfiguracji oraz nadpisywania poszczególnych zachowań, jednak w wielu zastosowaniach okazuje się to być zbędne.

5.2.3. JWT

Technologia JWT (ang. *Json Web Token*) została zastosowana jako sposób autoryzacji zapytań do API systemu. JWT jest technologią autoryzacji bezstanowej, przez co bardzo dobrze współgra z architekturą aplikacji REST-owych, dla których brak stanu jest jednym z głównych założeń.

5.2.4. MySQL

Relacyjna baza danych została wybrana ze względu na 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 ze szkieletem *Spring*.

5.2.5. Angular

Główną technologią wykorzystywaną po stronie front endu jest szkielet do tworzenia SPA (ang. *Single Page Application*) rozwijany przez firmę *Google - Angular* (w wersji 6.1.0) [11]. Szkielet ten ułatwia budowę skalowalnych i szybkich aplikacji z bogatym interfejsem użytkownika. Dużą zaletą *Angular-a* jest wsparcie dla programowania w języku *TypeScript*, co usprawnia rozwój aplikacji poprzez kontrolę typów.

Dodatkowo w celu usprawnienia procesu rozwoju aplikacji została wykorzystana biblioteka *ngrx* (w wersji 6.1.0), wspomagająca zarządzanie stanem aplikacji. Wykorzystanie tej biblioteki znacznie ułatwiło analizę działania aplikacji oraz diagnozowanie błędów.

AntDesign - NgZorro

NgZorro zostało użyte jako główna biblioteka dostarczająca szeroką gamę komponentów graficznych do aplikacji w technologii *Angular* [2]. Wybór padł na to rozwiązanie ze względu na duże wsparcie dla przeglądarek oraz bardzo dobrą dokumentację z przykładami. Użyte komponenty również cechują się atrakcyjnym wyglądem, który ma duży wpływ na odbiór systemu przez użytkownika.

5.2.6. Heroku

Heroku zapewnia duże wsparcie dla systemów zbudowanych w oparciu o język Java. Rozwiązanie to zostało wybrane ze względu na darmowe plany użytkowania, dobrą dokumentację, jak również doświadczenie autora pracy w zakresie wdrożeń na tę platformę.

Wdrożenie na *Heroku* aplikacji w technologiach *Spring Boot* oraz *Anuglar* nie stanowi żadnego problemu. Platforma dostarcza również dodatek w postaci bazy *MySQL*. W przypadku darmowego planu pojemność bazy jest bardzo ograniczona, jednak wystarczająca w celach rozwojowych oraz weryfikacyjnych systemu.

Rozdział 6

Implementacja systemu Team Challenge

W niniejszym rozdziale został opisany przebieg procesu implementacji, jak również sposób w jaki przy użyciu dostępnych technologii zrealizowano funkcjonalności systemu.

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

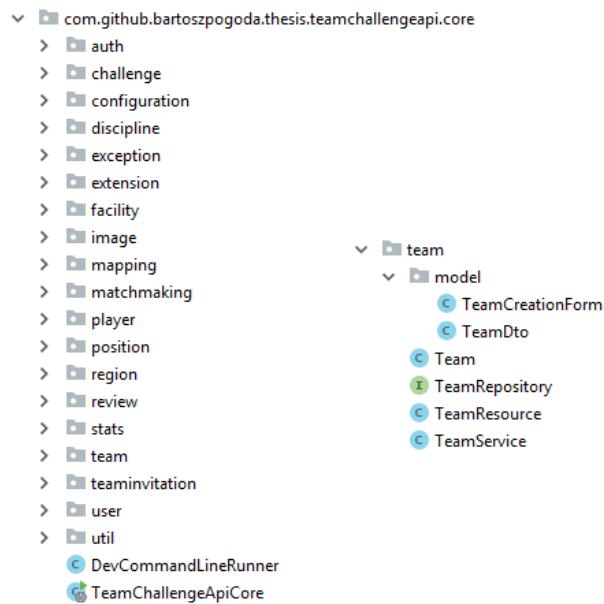
Nazwa programu	Wersja	Producent	Cel
IntelliJ IDEA	2018.2	Jetbrains	Implementacja aplikacji serwerowej (Java)
Webstorm	2018.2	Jetbrains	Implementacja aplikacji klienckiej (Angular)
Postman	6.5.2	Postman	Testowanie końcówek RESTowych aplikacji serwerowej
Google Chrome	70	Google	Testowanie aplikacji klienckiej
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*.

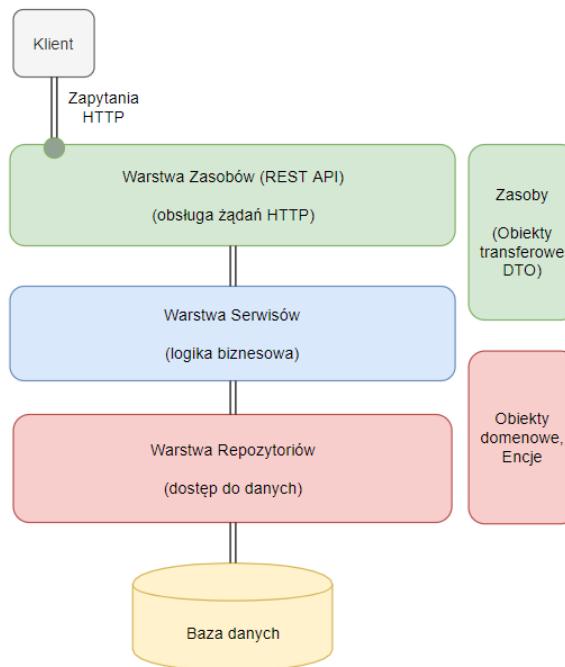
Klasy tworzące aplikację serwerową zostały podzielone na pakiety pod względem tematycznym, co zaprezentowano na rysunku 6.1. Podział taki pozwala na utrzymanie płaskiej struktury katalogów, a w związku z tym umożliwia szybkie odnajdywanie pożądanych plików.



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 szkielecie *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. We fragmencie kodu 6.1 przedstawiono powiązanie encji *Team* z fizyczną tabelą o nazwie *Teams* oraz mapowania przykładowych kolumn.

Fragment kodu 6.1: Fragment deklaracji klasy encyjnej *Team*

```
@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 we fragmencie kodu 6.2.

Fragment kodu 6.2: Definicja repozytorium *TeamRepository*

```
@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 *Spring 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 *Spring-u* serwisy oznaczane są adnotacją `@Service`. Szkielet sam zajmuje się tworzeniem instancji serwisów, które mogą być użyte w pozostałych komponentach systemu. Przykład realizacji logiki biznesowej w postaci ustawiania punktów macierzystych drużyn został przedstawiony we fragmencie kodu 6.3.

Fragment kodu 6.3: Fragment serwisu TeamService

```
@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 systemu 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 żądań są one przekazywane do konkretnych "kontrolerów", wybranych na podstawie URL żądania. Fragment kodu 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.

Fragment kodu 6.4: Fragment kontrolera TeamResource

```

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

    private TeamService teamService;

    private DtoMappingService mappingService;

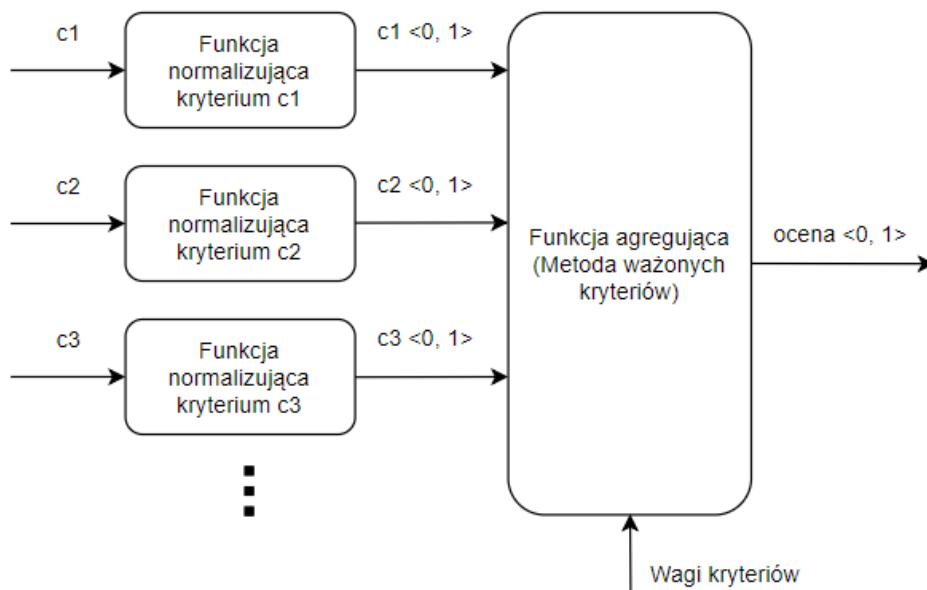
    @GetMapping("/{id}")
    public ResponseEntity<TeamDto> getTeam(@PathVariable String id)
        throws ApiException {
        return teamService.findById(id)
            .map(mappingService::mapToDto)
            .map(ResponseEntity::ok)
            .orElseThrow(TeamNotFoundException::new);
    }

    // other methods ...
}

```

6.2.6. Algorytm poszukiwania rywali

Algorytm poszukiwania rywali został zaimplementowany w oparciu o zagadnienie optymalizacji wielokryterialnej, które zostało przybliżone w rozdziale opisującym koncepcję. 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.



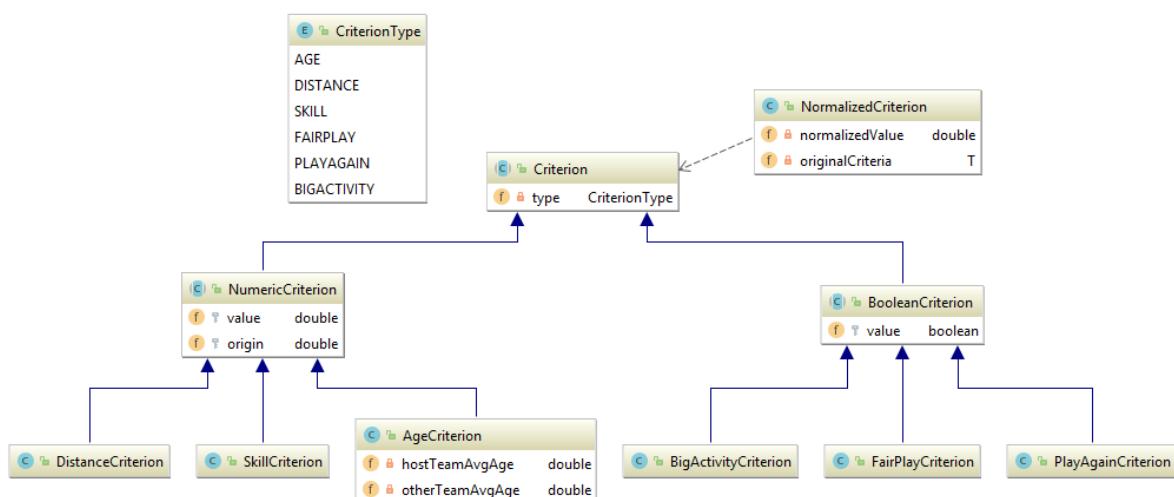
Rys. 6.3: Schemat działania algorytmu oceny decyzji

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 6.6. 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.

Fragment kodu 6.5: Generacja kryterium różnicy wieku między dwoma drużynami

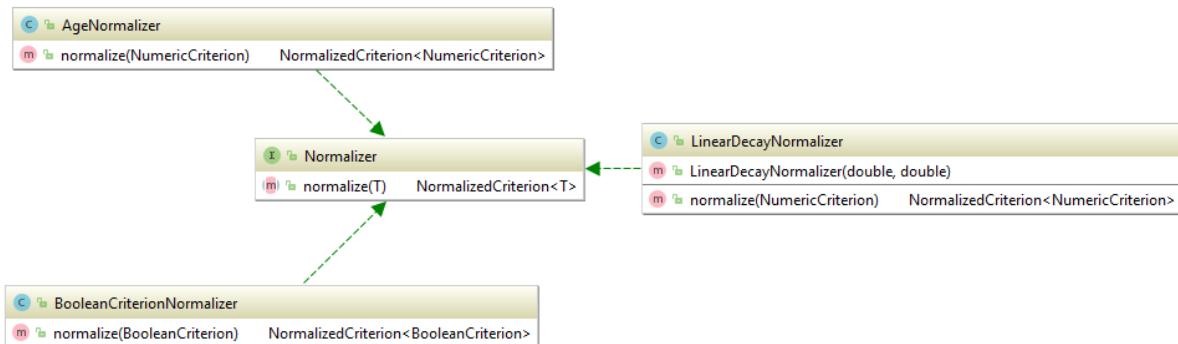
```

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 [6]. Konieczne jest przyporządkowanie wagi dla każdego z kryteriów. Ocena poszczególnych decyzji obliczana jest według wzoru:

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

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

$$w \in [0, 1] \text{ oraz } \sum_{i=1}^k w_i = 1 \quad (6.2)$$

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.

Fragment kodu 6.6: Klasa agregująca za pomocą metody ważonych kryteriów

```

@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();
    }

}

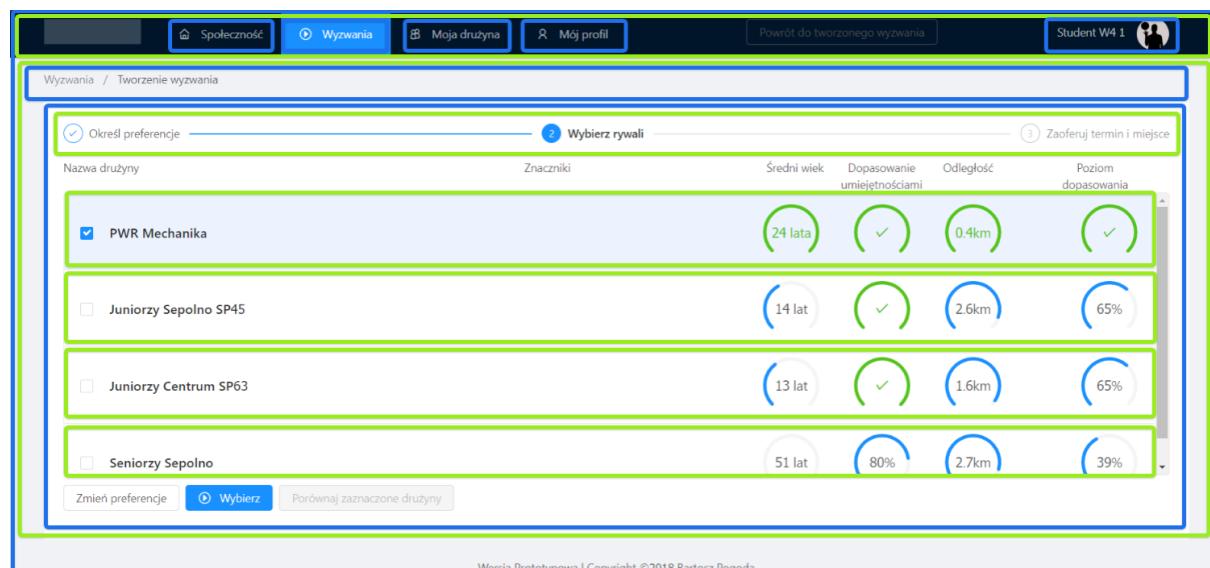
```

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 uzyskuje się poprzez podział na komponenty oraz moduły, które je agregują w zbiory [1].

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 [14]. 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 zgłoszonych akcji dokonują aktualizacji stanu. Akcje są zgłoszane 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 (ang. *JavaScript Object Notation*). 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. W celu sprawdzania poprawności wprowadzonych danych już po stronie aplikacji klienckiej wykorzystano wsparcie wbudowane w module *ReactiveFormsModule*. Przykładowy rezultat walidacji przedstawiono na rysunku 6.10.

Komponenty wizualne takie jak przyciski, suwaki, pola wyboru oraz pola tekstowe zostały dostarczone przez bibliotekę *NgZorro*. Elementy formularza wykorzystujące mapy zostały zbudowane przy użyciu modułu *@ngui/map* oraz *Google Maps API*. Przykładowy krok formularza z widokiem mapy został przedstawiony na rysunku 6.9.

Rys. 6.9: Formularz wprowadzania obiektu sportowego - krok pierwszy

6.3. Implementacja aplikacji klienckiej

The screenshot shows the second step of a three-step form for creating a sports facility. The top navigation bar includes links for 'Społeczność', 'Wyzwania', 'Moja drużyna', 'Mój profil', and a user icon labeled 'Player 3'. The main content area has a header 'Wprowadź podstawowe dane obiektu' (Enter basic facility data). It contains fields for 'Region' (selected: Wrocław), 'Nazwa obiektu' (Name of the facility), 'Lokalny adres' (Local address: Sienkiewicza 23), and 'Ilość koszy' (Number of basketball hoops: 4). Below these fields is a note: 'Wprowadź ulicę oraz w miarę możliwości numer budynku.' (Enter street name and building number if possible). At the bottom are 'Wróć' (Back) and 'Dalej' (Next) buttons. The footer indicates it's a prototype version from 2018.

Rys. 6.10: Formularz wprowadzania obiektu sportowego - krok drugi

Większość z utworzonych formularzy podzielono na mniejsze etapy - kroki. Podejście takie pozwala na ograniczenie ilości elementów, nad którymi użytkownik musi się skupić w danym momencie. Cały formularz wydaje się lżejszy i bardziej atrakcyjny dla użytkownika. W celu graficznej prezentacji aktualnego kroku formularza oraz postępu użyto komponentu *nz-steps* dostarczonego przez *NgZorro*. Ostatni krok formularza wprowadzania obiektu sportowego przedstawiono na rysunku 6.11.

The screenshot shows the third step of the three-step form. The top navigation bar is identical to the previous screenshot. The main content area has a header 'Wprowadź szczegółowe dane obiektu' (Enter detailed facility data). It contains fields for 'Rodzaj nawierzchni' (Surface type: selected Tartan) and 'Oświetlenie' (Lighting: toggle switch turned on). Below these is a 'Opis:' (Description:) text area. At the bottom are 'Wróć' and 'Dalej' buttons. The footer indicates it's a prototype version from 2018.

Rys. 6.11: Formularz wprowadzania obiektu sportowego - krok trzeci

Walidacja poprawności danych w systemie została zrealizowana zarówno po stronie serwerowej, jak i klienckiej. Wypełniane przez użytkownika formularze są na bieżąco sprawdzane pod względem poprawności. Użytkownik jest niezwłocznie informowany o wszelkich błędach jak np. braku wypełnienia wymaganych pól lub błędów w formacie. Powyżej opisane funkcjonalności zostały zrealizowane za pomocą komponentów wchodzących w skład modułu *ReactiveFormsModule*.

6.4. Bezpieczeństwo

Szczególną uwagę poświęcono implementacji zabezpieczeń systemu oraz zgromadzonych danych użytkowników. Funkcjonalności związane z bezpieczeństwem zostały zrealizowane biorąc 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 hashowaniu znacznie utrudnia złamanie hasła przez ataki przy użyciu tyczowych 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 została zaimplementowana w oparciu o technologię JWT. Formularz logowania przesyłany jest z aplikacji klienckiej do aplikacji serwerowej, gdzie jest przechwytywany przez kontroler *TokenResource*. W przypadku pomyślnej autentykacji generowany jest token JWT zawierający podstawowe informacje o użytkowniku, jego rolach oraz dacie wygenerowania oraz wygaśnięcia tokenu. Token jest podpisany za pomocą algorytmu HS256 specjalnym kluczem, co praktycznie uniemożliwia jego modyfikację bądź podrobienie bez znajomości klucza. Wygenerowany token wraca do klienta, gdzie jest zapisywany w stanie aplikacji oraz dekodowany w celu odczytania otrzymanych informacji. Token jest używany do autoryzacji dalszych zapytań do aplikacji serwerowej, w tym celu dodawany jest jako nagłówek każdego żądania HTTP. Serwer otrzymując żądania w pierwszej kolejności sprawdza obecność nagłówka oraz weryfikuje zawarty w nim token pod względem poprawności, aktualności oraz praw dostępów do konkretnych zasobów.

Ze względu na ograniczony czas ważności tokenu aplikacja kliencka zmuszona jest co pewien czas odnawiać token. W tym celu wysyła zapytanie na specjalny adres *POST /api/token/renewal*. Serwer w przypadku pomyślnej weryfikacji aktualnego tokenu generuje nowy o nowej dacie ważności. W przypadku niepomyślnej weryfikacji serwer zwraca kod błędu, w wyniku czego następuje wylogowanie użytkownika z informacją o wygaśnięciu sesji i konieczności nowego logowania.

6.5. Weryfikacja implementacji

Część funkcjonalności systemu została pokryta testami jednostkowymi, które uruchamiane były każdorazowo po wprowadzonych zmianach w kodzie. Testy te pomagały wykrywać problemy związane z działaniem pojedynczych komponentów w izolacji. Głównym narzędziem wykorzystywanym do tworzenia testów oraz ich wykonywania był *JUnit 4*. Dodatkowo zostały użyte biblioteki *Hamcrest 1.3* oraz *Mockito 2.15.0*. Wykorzystanie *Hamcrest* wpłynęło na czytelność asercji. Biblioteka *Mockito* pozwoliła na wykonywanie testów w pełnej izolacji poprzez wykorzystywanie w testach sztucznych implementacji zależności testowanej klasy. Dzięki temu w poszczególnych testach sprawdzana była poprawność implementacji jednej klasy, a nie innych klas, od których ona zależy.

Weryfikacja działania systemu jako całość, pod kątem zgodności z założeniami, czyli testy e2e (ang. *end-to-end*) wykonywana była w sposób manualny przez autora pracy.

Rozdział 7

Ocena użyteczności

System został dwukrotnie poddany walidacji. Pierwszy etap testów odbył się po ukończeniu implementacji pierwszego prototypu systemu. Drugi etap odbył się po ukończeniu drugiego prototypu, który poprawiał niedoskonałości odkryte podczas pierwszego etapu. Głównym celem przeprowadzonego procesu walidacji było sprawdzenie, czy implementowane rozwiązanie spełnia oczekiwania potencjalnych odbiorców.

Do walidacji 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 poszukiwanie rywali oraz rzucanie im wyzwania. Jest to najważniejsza funkcja systemu zawierająca elementy interfejsu wymagające sprawdzenia użyteczności.

7.1. Metodyka testów

Testy odbyły się w formie badań moderowanych, czyli z udziałem osoby nadzorującej ich przebieg [7]. 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 17 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 [9].

Dla uczestników badania została przygotowana ankieta składająca się z trzech części. Pierwsza część była ankietą wstępnią uzupełnianą przed badaniami. Zawierała ona krótkie wprowadzenie do systemu oraz pytania dotyczące wieku, profesji oraz umiejętności obsługi komputera osoby ankietowanej. 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 zalet interfejsu, napotkanych problemów w obsłudze oraz potencjalnych usprawnień. Na podstawie notatek moderatora oraz ankiet zostały 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 notatek można znaleźć na załączonej płycie.

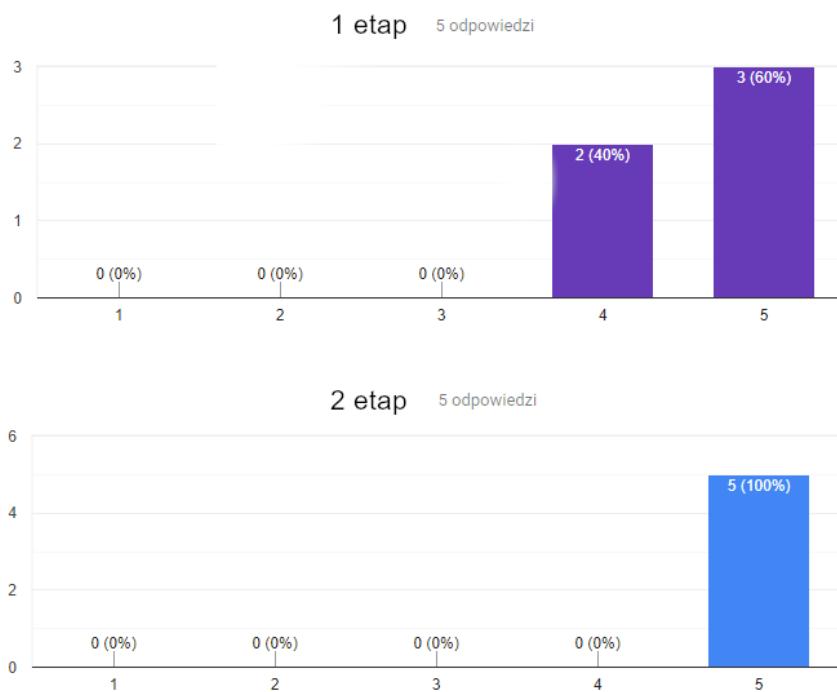
7.2.1. Tworzenie profilu zawodnika

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.

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 uwzględniona w implementacji drugiego prototypu.

Jak oceniasz stopień trudności odnalezienia odpowiedniego formularza?



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.

Deklaracja poziomu umiejętności

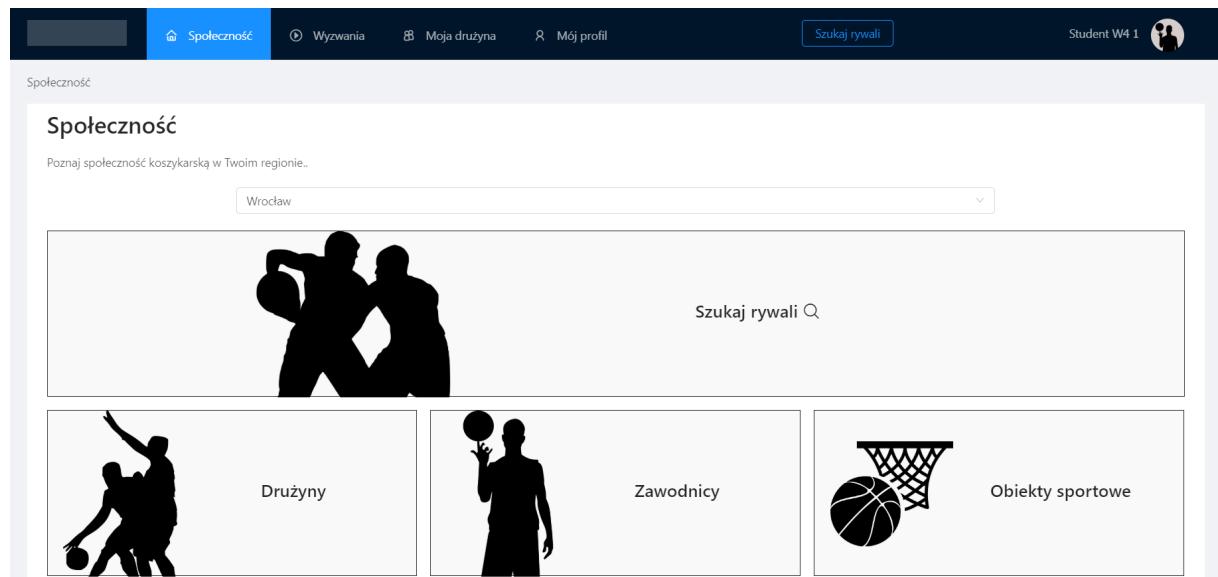
Krok polegający na deklaracji poziomu umiejętności również nie stanowił żadnych trudności. Każdy z ankietowanych był w stanie czytając opisy poszczególnych profili umiejętności dopasować się do jednego z nich. Dokonywanie wyborów poprzez przesunięcie suwaków było w pełni intuicyjne.

7.2.2. Szukanie przeciwników i rzucanie wyzwania

Drugie zadanie miało na celu sprawdzenie intuicyjności poszukiwania przeciwników oraz tworzenia wyzwania. Respondenci zostali zalogowani na przygotowane wcześniej konto kapitana jednej z drużyn i poproszeni o odnalezienie przeciwników o podobnym wieku w okolicy oraz rzucenie im wyzwania.

Odnalezienie funkcjonalności

Respondenci używający pierwszego prototypu odnaleźli odpowiedni formularz, jednak dla niektórych wymagało to trochę czasu. W drugim prototypie odnośnik umieszczono w centrum strony społeczności widocznej po zalogowaniu do systemu.



Rys. 7.2: Widok po zalogowaniu się kapitana do systemu

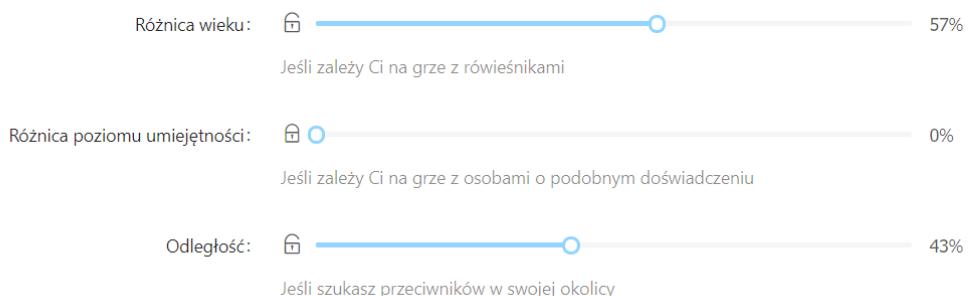
Deklaracja preferencji wyszukiwania

Deklaracja preferencji odnośnie rywali okazała się być problematyczna dla części użytkowników. Zaproponowany w ramach pierwszego prototypu podział procentowy wprowadzał użytkowników w błąd. Z punktu widzenia algorytmu im bardziej znaczące jest dane kryterium tym większą wartość (wagę) powinno ono otrzymać. Użytkownicy jednak wartość 0% utożsamiali z brakiem różnicy a więc dobrym dopasowaniem. Z tego względu niektórzy respondenci ustawiли suwaki odwrotnie niż było to wskazane. Interfejs przedstawiono na rysunku 7.3

Dużą część uwagi podczas przygotowywania drugiego prototypu poświęcono poprawie intuicyjności tej funkcjonalności. W pierwszej kolejności przygotowano makietę nowego wyglądu i opisu funkcjonalności, która została przedstawiona respondentom i spotkała się z ich aprobatą. Następnie planowane zmiany zostały zaimplementowane i zweryfikowane podczas testów drugiego prototypu. Głównym usprawnieniem było zrezygnowanie z wartości procentowych na rzecz abstrakcji - punktów preferencji. Dodatkowo dodano symbole graficzne, dokładniejsze opisy poszczególnych wyborów oraz panel z pomocą odnośnie rozdzielania punktów preferencji. Interfejs po zmianach przedstawiono na rysunku 7.4.

Dostosuj wagę poszczególnych kryteriów

Dostosuj wartościowanie poszczególnych kryteriów z uwzględnieniem preferencji Twojej drużyny.

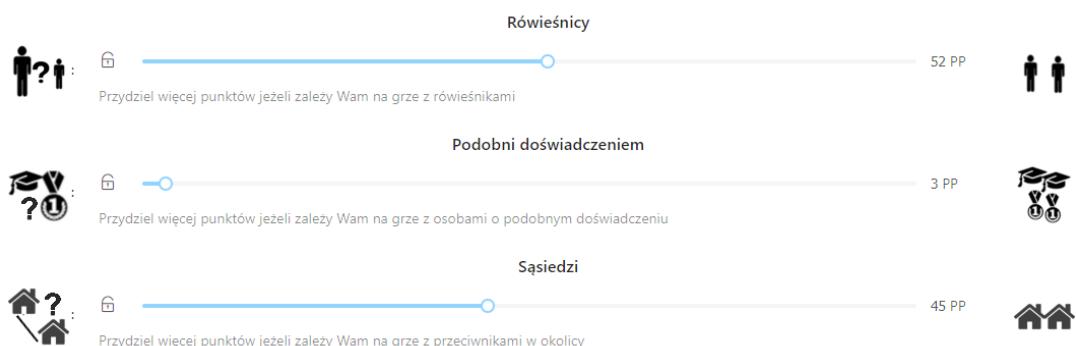


Rys. 7.3: Formularz deklaracji preferencji przed zmianami - I prototyp

Rozdysponuj punkty preferencji pomiędzy poszczególne cechy

[Jak rozdać punkty preferencji?](#)

Do dyspozycji masz 100 PP.

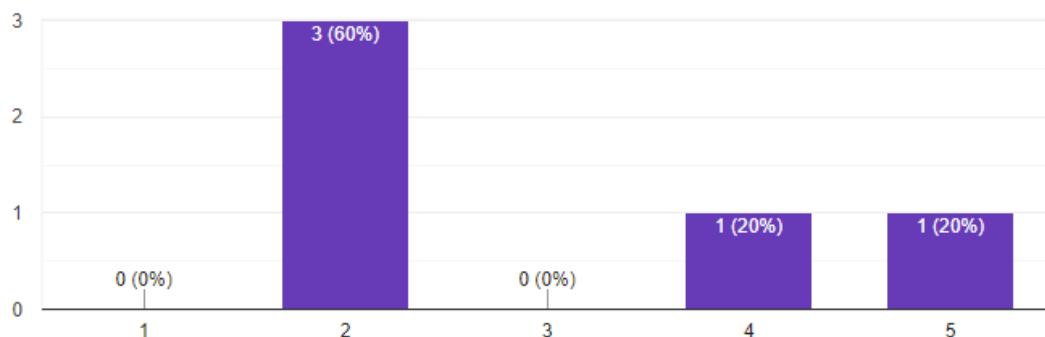


Rys. 7.4: Formularz deklaracji preferencji po zmianach - II prototyp

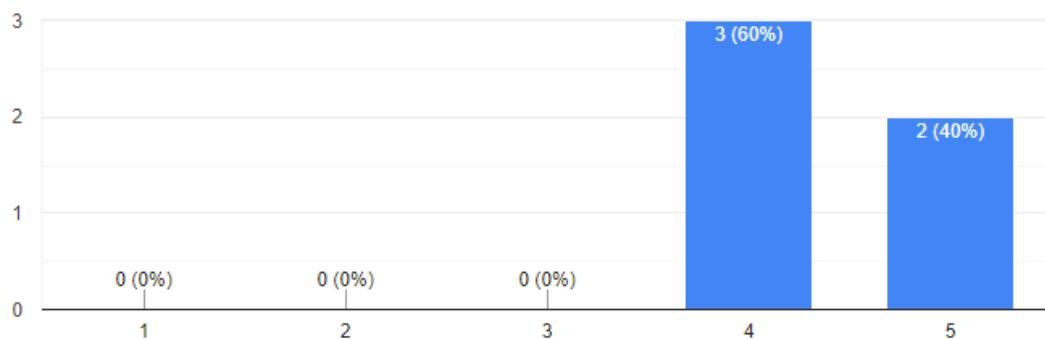
Wprowadzone usprawnienia przełożyły się na znaczne lepsze zrozumienie funkcjonalności przez respondentów. Rysunek 7.5 przedstawia zestawienie odpowiedzi osób ankietowanych przed oraz po wdrożeniu zmian.

Czy dobieranie wartościowania kryteriów przy poszukiwaniu było intuicyjne? (trzy suwaki)

1 etap 5 odpowiedzi



2 etap 5 odpowiedzi



Rys. 7.5: Zestawienie ocen intuicyjności deklaracji preferencji wyszukiwania (1 - nie intuicyjne, 5 - w pełni intuicyjne)

Wyniki wyszukiwania oraz rzucanie wyzwania

Sposób przedstawienia wyników wyszukiwania - proponowanych rywali okazał się być bardzo przystępny dla użytkowników. Respondenci byli w stanie na podstawie dostarczonych informacji wnioskować na temat dopasowania poszczególnych drużyn, a następnie rzucić wyzwanie jednej z nich. Proponowanie wstępnych ofert miejsca oraz czasu spotkania było zrozumiałe. W ramach tej części funkcjonalności w drugim prototypie zostały wprowadzone jedynie drobne zmiany.

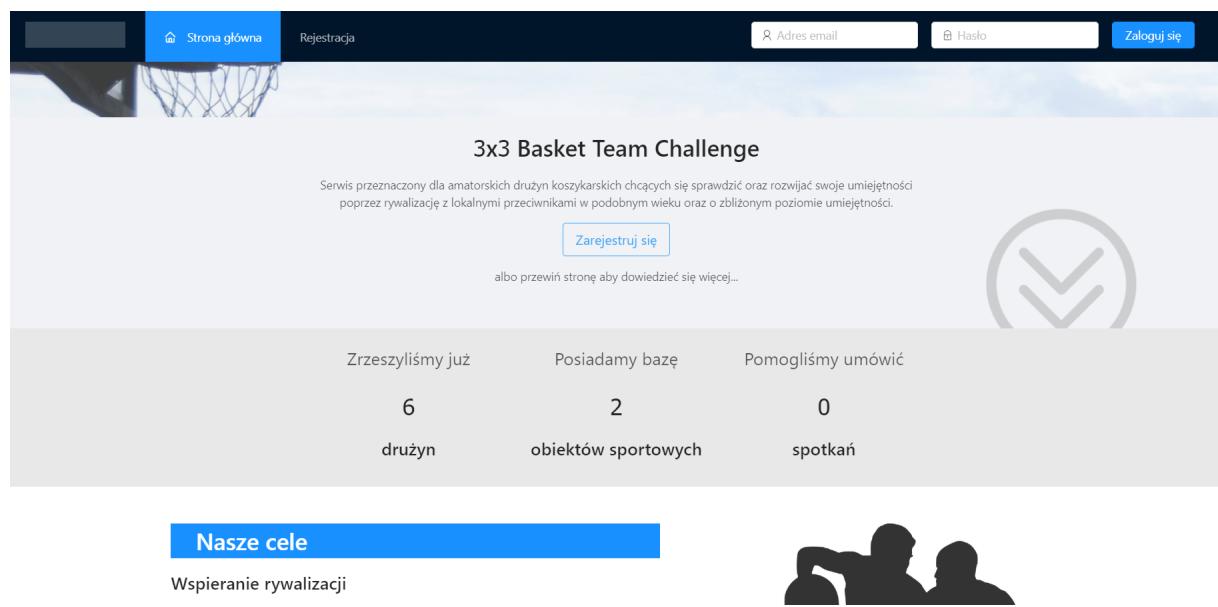
Rozdział 8

Prezentacja działania systemu Team Challenge

W poniższym rozdziale zostało przybliżone działanie systemu *Team Challenge* z punktu widzenia użytkownika końcowego. Dla wybranych funkcjonalności systemu został przedstawiony sposób realizacji oraz przykładowe widoki. Zrzuty ekranu zostały wykonane dla drugiego prototypu aplikacji, wdrożonego na platformę *Heroku* www.team-challenge.herokuapp.com.

8.1. Rejestracja i logowanie

Pierwszą funkcjonalnością z jaką ma styczność każdy nowy użytkownik jest rejestracja w systemie. W celu utworzenia konta należy wcisnąć jeden z odnośników znajdujących się na stronie głównej systemu, którą przedstawiono na rysunku 8.1.



Rys. 8.1: Widok strony głównej systemu

Formularz rejestracyjny został przedstawiony na rysunku 8.2. Podany przez użytkownika adres e-mail będzie jego loginem do systemu. Hasło pobierane jest dwukrotnie w celu zmniejszenia szansy na omyłkowe zadeklarowanie błędного hasła. Dodatkowo hasło ze względów bezpieczeństwa musi składać się z co najmniej 7 znaków.

Dołącz do nas już dzisiaj!

Jesteś gotowy podjąć wyzwanie? Zarejestruj się korzystając z poniższego formularza.

* Adres email: student.elektroniki@mail.com

* Hasło: ██████████

* Potwierdź hasło: ██████████
Hasła nie są zgodne!

* Imię i nazwisko: Student Elektroniki

* Data urodzenia: 1996/05/13

Zarejestruj się

Rys. 8.2: Widok formularza rejestracji

Po utworzeniu konta użytkownik może się zalogować do systemu za pomocą formularza umieszczonego na pasku nawigacyjnym. Fragment widoku został przedstawiony na rysunku 8.3.



Rys. 8.3: Widok formularza logowania

8.2. Tworzenie profilu zawodnika

Po pierwszym logowaniu nowego użytkownika, zostaje on przekierowany do kreatora zawodnika. Jest to formularz pozwalający na uzupełnienie swojego profilu zawodnika o informacje konieczne dla funkcjonowania systemu *Team Challenge*. Pierwszy krok formularza przedstawiono na rysunku 8.4.

Mój profil / Kreator

Kreator zawodnika

Nie posiadasz jeszcze profilu zawodnika. Aby móc dołączyć do rozgrywek załóż profil korzystając z poniższego formularza.

1 Podstawowe dane 2 Umiejętności 3 Zdjęcie

* Region: Wrocław

* Wzrost (cm): 184

Dalej

Rys. 8.4: Widok kreatora zawodnika - krok pierwszy

Region jest wybierany spośród zdefiniowanych przez administratora. W prototypowej wersji systemu są to Wrocław oraz Kłodzko. Zawodnik proszony jest o wprowadzenie swojego wzrostu, jednak ta cecha nie ma wpływu na działanie systemu. Najistotniejsze informacje z punktu widzenia mechanizmu dopasowywania pobierane są w drugim kroku formularza, który przedstawiono na rysunku 8.5.

Nie posiadasz jeszcze profilu zawodnika. Aby móc dołączyć do rozgrywek załącz profil korzystając z poniższego formularza.

Wybierz profil umiejętności

Zapoznaj się z opisami poszczególnych profili i wybierz ten, o którego najbardziej pasujesz.

Świeżak	Początkujący	Średnio-zaawansowany	Zaawansowany	Ekspert
Dopiero zaczynasz przygodę z koszykówką	Znasz w podstawowym stopniu technikę gry oraz podstawowe zagrywki.	Grasz już od dłuższego czasu. Dobrze się czujesz na boisku. Nie masz problemu ze zdobyciem punktu spod kosza lub za pomocą dwutaktu.	Posiadasz duże doświadczenie. Masz opanowane zaawansowane zagrywki takie jak pivot. Zdobywasz punkty nawet z trudnych pozycji.	Posiadasz wieloletnie doświadczenie. Na boisku czujesz się jak ryba w wodzie.

Określ częstość swojej gry

Okazjonalnie Parę razy miesięcznie Parę razy tygodniowo

Utwórz profil

Rys. 8.5: Widok kreatora zawodnika - krok drugi

Użytkownik deklaruje swój stopień zaawansowania oraz częstość gry. Na podstawie tych dwóch wartości system przypisuje zawodnikowi wartość w zakresie 0 do 10 określającą jego umiejętności z uwzględnieniem aktualnej formy. Wartość ta steruje kryterium bliskiego poziomu umiejętności zawodników podczas dopasowywania. Ostatni krok formularza jest w pełni opcjonalny, pozwala on na wgranie do systemu zdjęcia lub awatara - rysunek 8.6.

Mój profil / Kreator

Kreator zawodnika

Dodaj zdjęcie

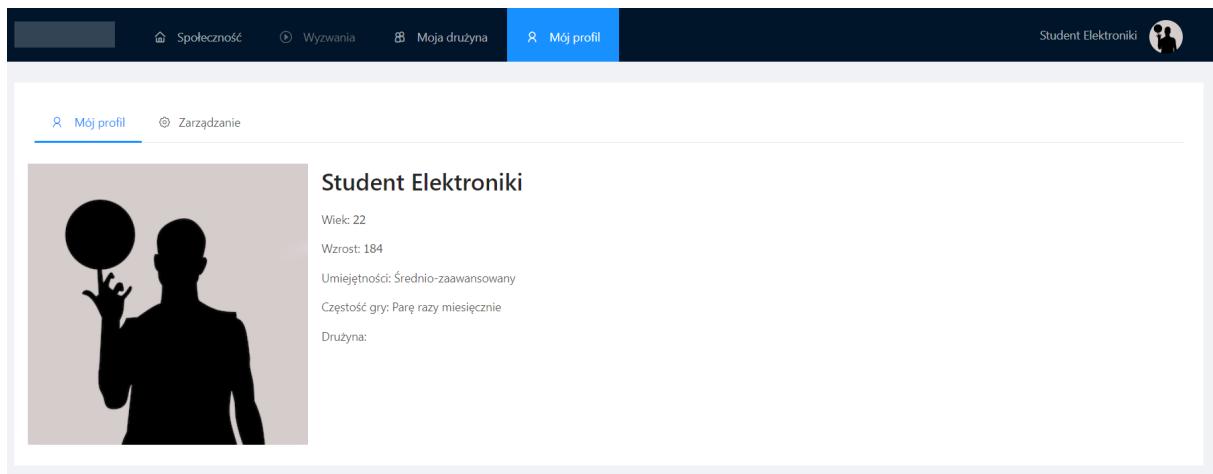
Aby inni mogli Cię rozpoznać.

Wybierz...

Pomiń krok

Rys. 8.6: Widok kreatora zawodnika - krok trzeci

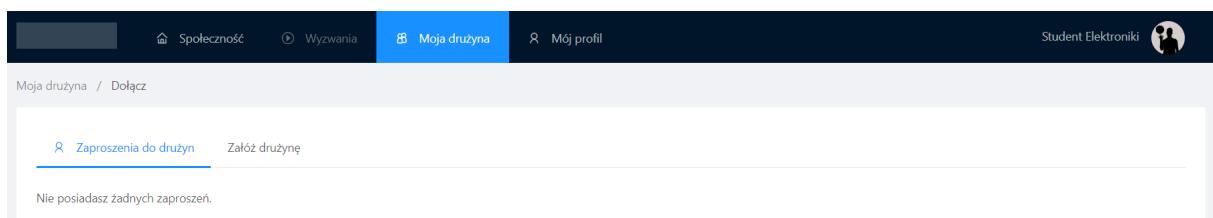
Na rysunku 8.7 przedstawiono utworzony profil zawodnika. Ukażany awatar jest domyślny dla wszystkich zawodników, którzy nie wgrali własnego zdjęcia.



Rys. 8.7: Widok profilu zawodnika

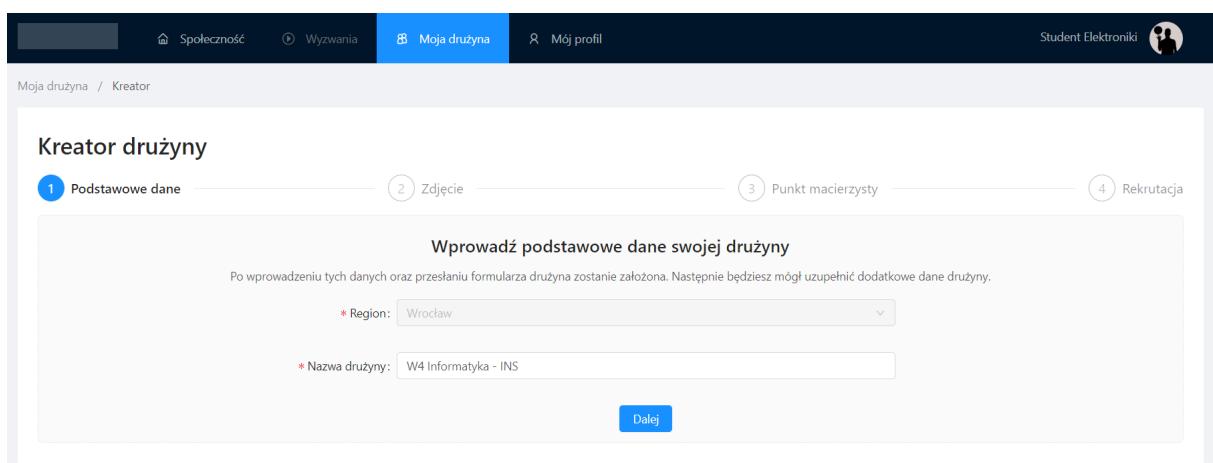
8.3. Tworzenie drużyny

Funkcjonalność tworzenia drużyny znajduje się pod przyciskiem “Załącz drużynę” widocznym w zakładce “Moja drużyna” gdy zawodnik nie należy do żadnej drużyny, co ukazano na rysunku 8.8.



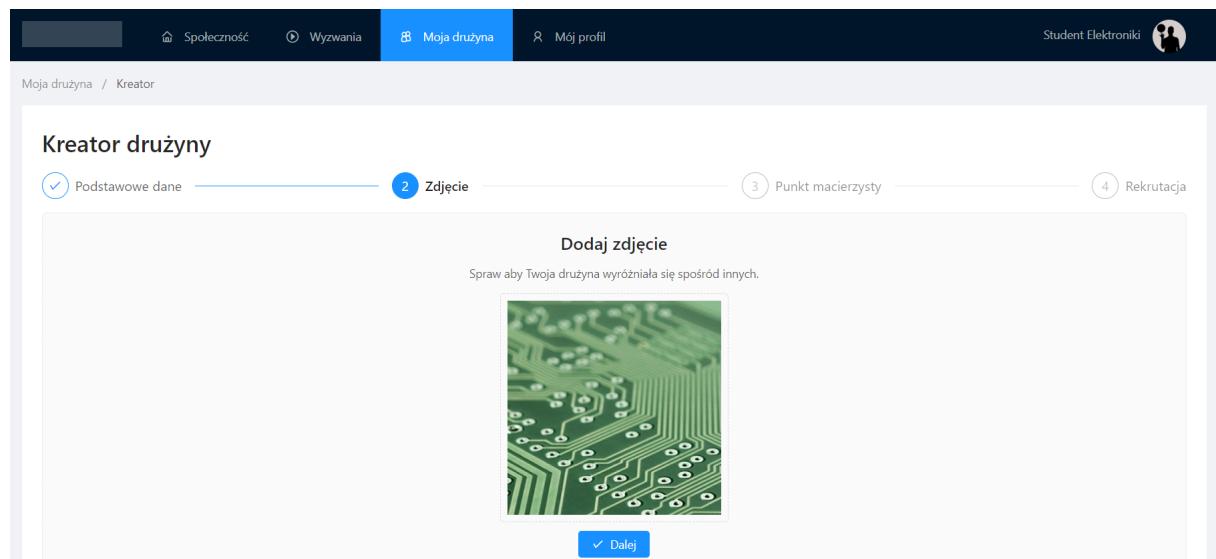
Rys. 8.8: Widok zakładki “Moja drużyna”

Pierwszym krokiem jest zdefiniowanie nazwy drużyny. Nazwa ta będzie reprezentować drużynę w systemie. Region drużyny wybierany jest automatycznie na podstawie regionu wybranego przez zawodnika podczas tworzenia profilu.



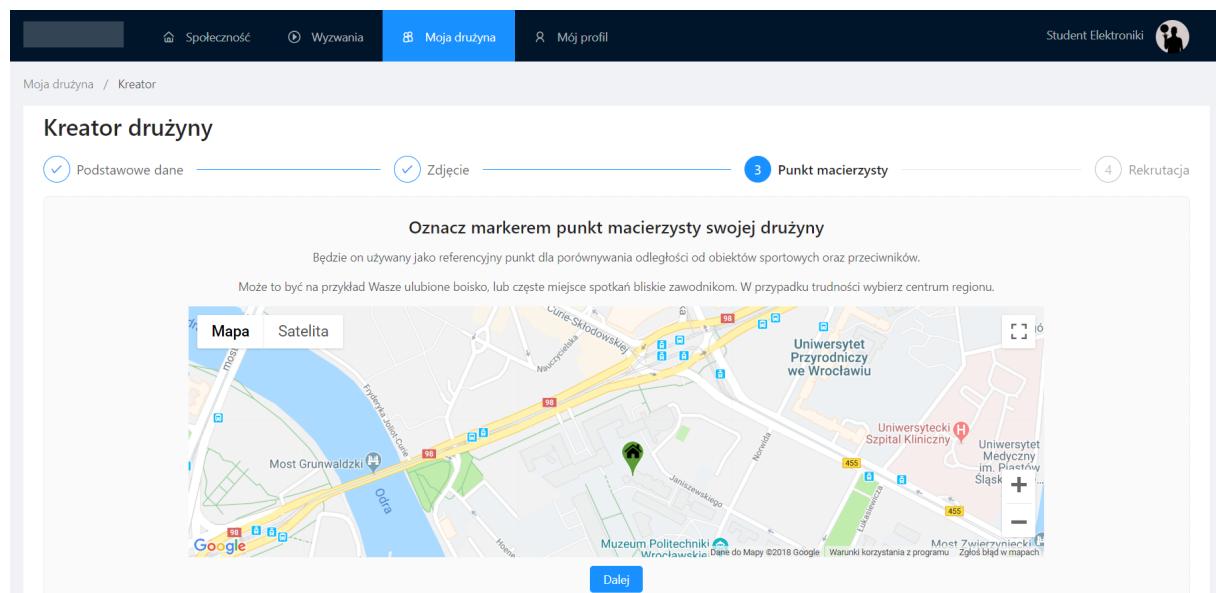
Rys. 8.9: Widok kreatora drużyny - krok pierwszy

Podobnie jak w przypadku profilu zawodnika, kapitan ma możliwość wgrania zdjęcia drużyny. Widok po wgraniu przykładowego zdjęcia przedstawiono na rysunku 8.10.



Rys. 8.10: Widok kreatora drużyny - krok drugi

Kapitan może wybrać punkt macierzysty drużyny, czyli lokalizację służącą do porównań odległości między drużynami. W tym celu użytkownik może przemieszczać zielony marker na mapie poprzez wciskanie docelowych punktów. Domyślnie jest to centrum regionu, jednak zalecane jest ustawnienie miejsca, które jest pobliskie zawodnikom. W przypadku studentów elektroniki mieszkających w okolicy kampusu może to być budynek wydziału. Przykładowe ustawienie zaprezentowano na rysunku 8.11.



Rys. 8.11: Widok kreatora drużyny - krok trzeci

Ostatnim krokiem kreatora jest widok rekrutacji, który umożliwia wysłanie wstępnych zaproszeń zawodników do drużyny. Zawodnicy są przedstawieni w tabeli z możliwością ich wyszukiwania po imieniu oraz nazwisku. Tabela przedstawia jedynie zawodników, których można zaprosić, czyli zdefiniowanych dla tego samego regionu co drużyna oraz nie będący w innym zespole. Widok przedstawiono na rysunku 8.12.

Rys. 8.12: Widok kreatora drużyny - krok czwarty

8.4. Wyszukiwanie rywali i rzucanie wyzwań

Funkcjonalność wyszukiwania rywali oraz umawiania spotkań staje się dostępna dla kapitana drużyny w momencie gdy zbierze on co najmniej trzech zawodników. Jest to minimalny rozmiar drużyny dla koszykówki 3 na 3. Kapitan aktywnej drużyny ma dostęp do przycisku "Szukaj rywali" na pasku nawigacji. Przycisk ten służy do szybkiej nawigacji do funkcjonalności z każdego miejsca aplikacji. Kapitan może również użyć zakładki "Szukaj rywali" w widoku społeczności, który widoczny jest na rysunku 8.13.

Rys. 8.13: Widok zakładki "Społeczność"

Pierwszym etapem wyszukiwania rywali jest określenie preferencji kapitana. Formularz został przedstawiony na rysunku 8.14. Zadaniem kapitana jest rozdysponowanie 100 punktów preferencji pomiędzy cechy rywali takie jak: podobny wiek, podobne doświadczenie oraz mała odległość między punktami macierzystymi. Podział pomiędzy te trzy cechy odbywa się za pomocą suwaków, które zostały skonstruowane tak, aby ich wartości w sumie zawsze dawały 100.

Oznacza to, że zwiększając wartość jednego suwaka, zmniejszamy wartość dwóch pozostałych. Kapitan może zablokować wybrany suwak na określonej wartości za pomocą symbolu kłódki znajdującej się po lewej stronie. Po prawej stronie ekranu znajdują się także dodatkowe preferencje takie jak poziom fair-play, ponowne spotkanie oraz duża aktywność przeciwników. Na podstawie wypełnionego formularza obliczane są wagi poszczególnych kryteriów dla algorytmu optymalizacji wielokryterialnej. Proporcje między wartościami punktów preferencji dla poszczególnych cech przekształcane są na wagi kryteriów liczbowych, a zadeklarowane dodatkowe preferencje powodują uwzględnianie w ocenie poszczególnych drużyn wybranych kryteriów logicznych (rozdział 6.2.6).

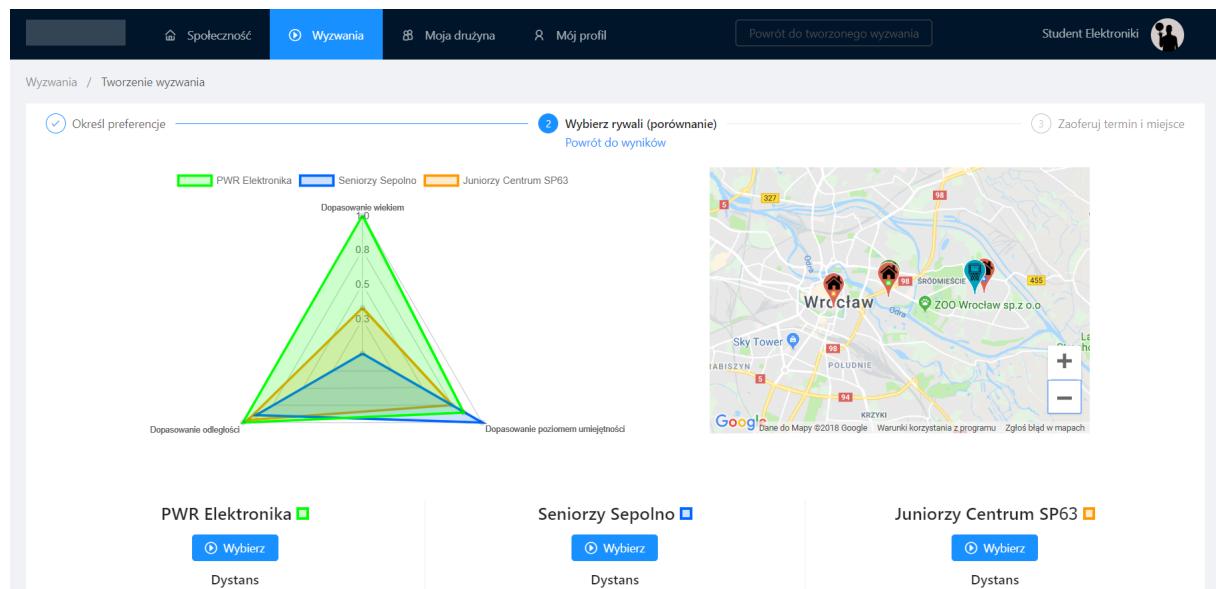
Rys. 8.14: Widok formularza deklaracji preferencji drużyny

Wyniki wyszukiwania przedstawione są w postaci listy posortowanej według poziomu dopasowania. Poziom dopasowania jest oceną ważoną poszczególnych drużyn z uwzględnieniem preferencji kapitana. Dla każdej z drużyn wyświetlane są znaczniki, które odpowiadają cechom związanym z dodatkowymi preferencjami oraz poziomy dopasowań pod względem wieku, umiejętności oraz odległości. Przykładowy widok przedstawiono na rysunku 8.15.

Nazwa drużyny	Znaczniki	Średni wiek	Dopasowanie umiejętnościami	Odległość	Poziom dopasowania
<input checked="" type="checkbox"/> PWR Elektronika	Fair play, Duża aktywność	22 lata	85%	0.1km	96%
<input type="checkbox"/> PWR Mechanika		24 lata	90%	0.4km	87%
<input type="checkbox"/> Seniorzy Sepolno	Fair play	51 lat	✓	2.7km	66%
<input type="checkbox"/> Juniorzy Centrum SP63		13 lat	75%	1.6km	61%

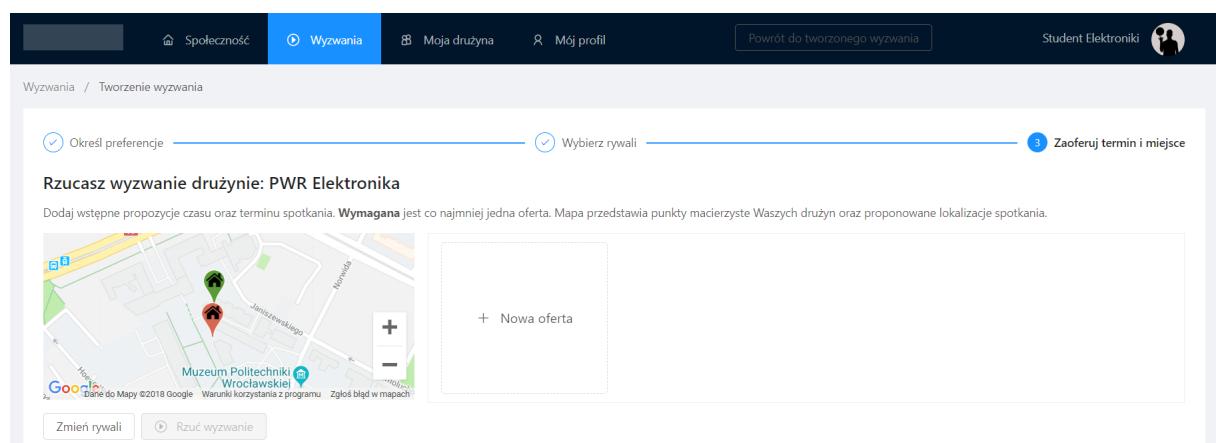
Rys. 8.15: Widok wyników wyszukiwania rywali

Kapitan wnioskując na podstawie wskaźników może od razu wybrać rywali albo zaznaczyć dwie lub trzy drużyny w celu porównania. W momencie zaznaczenia odpowiedniej liczby elementów uaktywnia się przycisk "Porównaj zaznaczone drużyny", który przenosi użytkownika do widoku porównania przedstawionego na rysunku 8.16. Wybrane drużyny zostają zestawione w sąsiednich kolumnach w celu ułatwienia podjęcia decyzji kapitanowi. Wykres przedstawia dopasowanie poszczególnych drużyn, a na mapie znajdują się punkty macierzyste zespołu kapitana oraz pozostałych biorących udział w porównaniu. W dolnej części widoku nie ujętej na zrzucie ekranu znajdują się informacje w formie tekstopisowej oraz przedstawieni są zawodnicy poszczególnych zespołów.



Rys. 8.16: Widok porównania potencjalnych rywali

Na potrzeby przykładu wybrana została drużyna PWR Elektronika, która jest bardzo dobrze dopasowana do drużyny poszukującej. Następnym krokiem jest rzucenie wyzwania za pomocą widoku przedstawionego na rysunku 8.17. Kapitan musi dodać co najmniej jedną ofertę czasu oraz miejsca spotkania za pomocą bloku "+ Nowa oferta". Formularz wprowadzania oferty przedstawiono na rysunku 8.18.



Rys. 8.17: Widok puli wstępnych ofert czasu oraz miejsca spotkania

8.4. Wyszukiwanie rywali i rzucanie wyzwań

Nowa oferta

Data: 2018-12-13

Czas: 14:00

Miejsce: Boisko przy Szkole Podstawowej nr 45

Adres: Ulica Oświetlenie: Nie Nawierzchnia: Beton Miejsca do gry: 4

Zmien rywali Rzuć wyzwanie

Anuluj Dodaj do puli

Rys. 8.18: Widok formularza tworzenia nowej oferty czasu i miejsca spotkania

Wprowadzone oferty widoczne są w puli, jak na rysunku 8.18. Kapitan ma możliwość cofania ofert przy użyciu przycisku “Anuluj”. W momencie wprowadzenia co najmniej jednej oferty przycisk “Rzuć wyzwanie” uaktywnia się.

Wyzwania / Tworzenie wyzwania

Określ preferencje Wybierz rywali Zaofuruj termin i miejsce

Rzucasz wyzwanie drużyny: PWR Elektronika

Dodaj wstępne propozycje czasu oraz terminu spotkania. **Wymagana** jest co najmniej jedna oferta. Mapa przedstawia punkty macierzyste Waszych drużyn oraz proponowane lokalizacje spotkania.

Zmień rywali Rzuć wyzwanie

Anuluj

Zaofurano
Boisko przy Szkole Po...
Adres: Ulica Data: 13 Grudnia 2018 Czas: 14:00

+ Nowa oferta

Rys. 8.19: Widok zaktualizowanej puli wstępnych ofert czasu oraz miejsca spotkania

W wyniku rzucenia wyzwania zostaje ono zarejestrowane w systemie oraz od tej pory jest widoczne w zakładce “Wyzwania”. Początkowym stanem każdego nowego wyzwania są negocjacje. Szczegóły dotyczące tego procesu zostaną opisane w następnej części tego rozdziału. Widok obecnych wyzwań został przedstawiony na rysunku 8.20.

Wyzwania

Nowe wyzwanie

Obecne wyzwania

Nazwa drużyny	Menedżer	Status	Miejsce	Data
PWR Elektronika	Student W4 1	W trakcie negocjacji	Nie ustalone	Nie ustalone

Szukaj rywali

Wyzwanie zostało utworzone.

Rys. 8.20: Widok obecnych wyzwań

8.5. Umawianie spotkań i wprowadzanie wyników

Umawianie spotkań jest ważną funkcjonalnością prowadzącą do realizacji celów systemu, czyli zrzeszania zawodników. Widok obecnych oraz zakończonych wyzwań przedstawiono na rysunku 8.21. Aktualne wyzwanie jest w stanie negocjacji. Dodatkowo widać informacje o istnieniu ofert czasu i miejsca od rywali, które nie zostały rozpatrzone - odrzucone bądź zaakceptowane.

Nazwa drużyny	Menedżer	Status	Miejsce	Data
W4 Informatyka - INS	Student Elektroniki	W trakcie negocjacji Aktywne oferty od rywali	Nie ustalone	Nie ustalone

Nazwa drużyny	Menedżer	Status	Miejsce	Data
Juniorzy Sepolno SP45	Junior Sepolno 1	Zakończone	Boisko przy Szkole Podstawowej nr 45	9 Grudnia 2018 22:15
Seniorzy Sepolno	Senior Sepolno 1	Zakończone	Boisko przy Szkole Podstawowej nr 45	10 Grudnia 2018 22:15
PWR Mechanika	Student W10 1	Zakończone	Boisko przy Szkole Podstawowej nr 45	31 Grudnia 2018 22:35

Rys. 8.21: Widok zakładki “Wyzwania”

Użytkownik może wejść w szczegóły poszczególnych wyzwań klikając na nie. Po wykonaniu tej czynności ukazuje się widok przedstawiony na rysunku 8.22. Widok ten prezentuje dodatkowe informacje na temat wyzwania, a także umożliwia podejmowanie akcji.

Rys. 8.22: Widok szczegółów wybranego wyzwania w trakcie negocjacji

Po utworzeniu wyzwania szczególnie ważny jest widok negocjacji, który został przedstawiony na rysunku 8.23. Po lewej stronie znajdują się oferty czasu i miejsca spotkania aktywnej

drużyny oraz jej rywali. Kapitan może wprowadzać nowe oferty, anulować swoje, bądź odrzucać i akceptować oferty rywali. Akceptacja oferty rywali równoznaczna jest ze zgodą obydwu drużyn na powiązane z nią miejsce oraz czas spotkania.

The screenshot shows a user interface for managing meetings. At the top, there are tabs: 'Społeczność' (Community), 'Wyzwania' (Challenges), 'Moja drużyna' (My team), 'Mój profil' (My profile), 'Szukaj rywali' (Search for opponents), and a user icon 'Student W4 1'. Below the tabs, the path 'Wyzwania / W4 Informatyka - INS' is shown. Underneath, there are two main sections: 'Wasze oferty' (Your offers) and 'Oferty rywali' (Opponent's offers). In 'Wasze oferty', there is one offer: 'Boisko przy Szkole Po...' with details: Adres: Ulica, Data: 11 Grudnia 2018, Czas: 14:10. It has a status 'Anulowana' (Cancelled) and a button 'Anuluj' (Cancel). In 'Oferty rywali', there is one offer: 'Boisko przy Szkole Po...' with details: Adres: Ulica, Data: 12 Grudnia 2018, Czas: 17:00. It has a status 'Zaoferedano' (Offered) and a button 'Anuluj' (Cancel). To the right of these sections is a map of Wrocław, Poland, showing various neighborhoods and roads. A green marker indicates the location of the meeting offer from the opponent.

Rys. 8.23: Widok negocjacji miejsca oraz czasu spotkania - w trakcie negocjacji

Na rysunku 8.24 przedstawiono widok po uzyskaniu zgody obydwu drużyn. Pozostałe oferty są automatycznie odrzucane, a proces negocjacji kończy się. Widok zaktualizowanych szczegółów spotkania wraz z jego ustaloną datą oraz miejscem przedstawiono na rysunku 8.25.

This screenshot shows the same interface after the opponent's offer has been accepted. The 'Wasze oferty' section is empty. In the 'Oferty rywali' section, the previous offer is now marked as 'Zaakceptowana' (Accepted) and has a green background. The details remain the same: 'Boisko przy Szkole Po...' with Adres: Ulica, Data: 12 Grudnia 2018, Czas: 17:00. The map on the right shows the same location in Wrocław, with a blue marker indicating the accepted meeting point.

Rys. 8.24: Widok negocjacji miejsca oraz czasu spotkania - po zaakceptowaniu

8.5. Umawianie spotkań i wprowadzanie wyników

The screenshot shows a challenge interface. At the top, there are navigation links: 'Społeczność', 'Wyzwania' (highlighted), 'Moja drużyna', 'Mój profil', 'Szukaj rywali', and a user icon. Below this, the breadcrumb path 'Wyzwania / PWR Elektronika' is shown. The main content area displays two teams: 'W4 Informatyka - INS' and 'PWR Elektronika', separated by a 'vs.' symbol. Each team has a silhouette icon and a row of small icons representing players. A section titled 'Status spotkania' shows a green button labeled 'Zaakceptowane' (Accepted). Other details include the date and time of the meeting ('12 Grudnia 2018 17:00') and its location ('Boisko przy Szkole Podstawowej nr 45'). Buttons at the bottom allow users to cancel or add results.

Rys. 8.25: Widok szczegółów zaakceptowanego wyzwania

W celu sfinalizowania wyzwania po jego rozegraniu drużyny powinny wprowadzić do systemu wynik meczu. Obojętnie który z kapitanów może w tym celu użyć przycisku "Wprowadź wynik" dostępnego dla wyzwań w stanie "Zaakceptowane". Wyświetlane jest okno dialogowe przedstawione na rysunku 8.26. Kapitan wprowadza punkty zdobyte przez poszczególne drużyny. Symbol pucharu umieszczany jest automatycznie nad drużyną, która zdobyła więcej punktów.

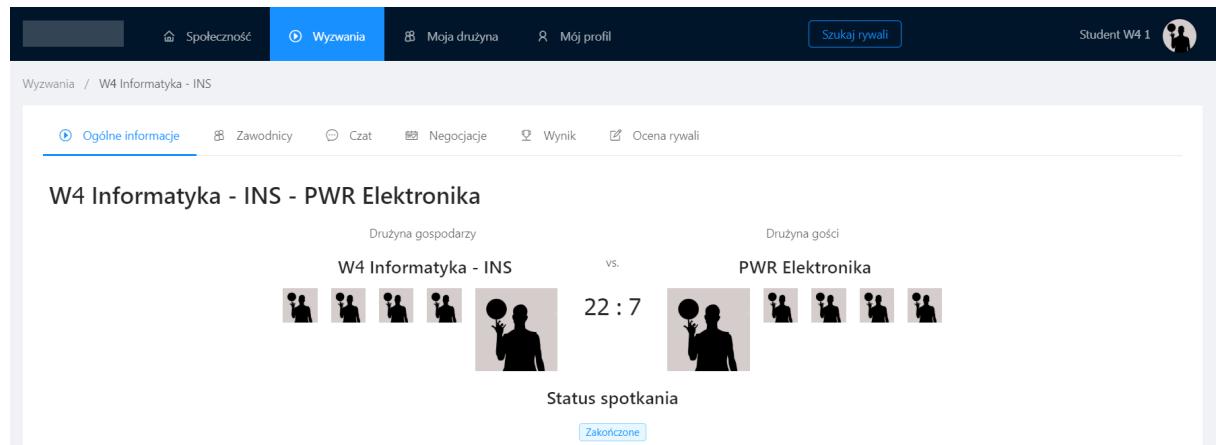
The screenshot shows a modal dialog box titled 'Wprowadzenie wyniku'. It displays the challenge details: 'W4 Informatyka - INS' vs 'PWR Elektronika'. Below the teams, there are two input fields with the numbers '22' and '7' respectively, with up and down arrows for adjustment. At the bottom of the dialog are 'Anuluj' (Cancel) and 'Wprowadź' (Enter) buttons. The background shows the same challenge interface as in Figure 8.25.

Rys. 8.26: Widok formularza wprowadzania wyniku wyzwania

The screenshot shows a challenge interface with the 'Wynik' (Result) tab selected. The challenge details are the same as in previous figures. A message at the top states: 'Wynik został zgłoszony przez drużynę rywali. W celu sfinalizowania wyzwania potwierdź ten wynik. W przypadku niezgodności możesz go odrzucić, co spowoduje to odrzucenie wyzwania.' Below this, the result is displayed as 'W4 Informatyka - INS' 22 - 7 'PWR Elektronika'. At the bottom are 'Odrzuć wynik' (Reject result) and 'Potwierdź wynik' (Confirm result) buttons.

Rys. 8.27: Widok zakładki "Wynik" w przypadku niezaakceptowanego wyniku

Wprowadzony wynik musi zostać potwierdzony przez drugiego z kapitanów lub odrzucony w przypadku niezgodności, co przedstawiono na rysunku 8.27. W momencie uzyskania zgody obydwu kapitanów odnośnie rezultatu spotkania zostaje ono oznaczone jako zakończone. Widok zaktualizowanego wyzwania przedstawiono na rysunku 8.28.



Rys. 8.28: Widok szczegółów zakończonego wyzwania

Ostatnim krokiem jaki kapitanowie drużyn mogą podjąć w ramach wyzwania jest ocenie-
nie rywali. Formularz oceny został przedstawiony na rysunku 8.29. Przesłanie formularza jest opcjonalne, jednak ma duży wpływ na działanie systemu. Drużyny oceniane wysoko pod względem poziomu fair-play są wyróżniane w wynikach wyszukiwania specjalnym znacznikiem. W przypadku deklaracji chęci ponownego spotkania drużyna będzie oznaczona znacznikiem ‘‘Zagraj ponownie’’, ale tylko dla drużyny, która wyraziła taką chęć.

The screenshot shows the 'Ocena rywali' (Opponent Evaluation) form for the match between 'W4 Informatyka - INS' and 'PWR Elektronika'. The user is asked to rate the opponent's fair play level on a scale from 1 to 5 stars. A note above the rating scale says 'Oceń poziom fair play swoich przeciwników.' Below the rating scale is a question 'Czy chcesz dodatkowo promować drużynę w swoich wynikach wyszukiwania w przyszłości?' (Do you want to promote the team in your search results in the future?). There is a checked checkbox labeled 'Tak' (Yes). A tooltip for this checkbox explains: 'Wysokość oceny poziomu fair play drużyny podnosi jej pozycję w wynikach wyszukiwania. Drużyna będzie oznaczona tagiem "Zagraj ponownie"'. At the bottom is a 'Zapisz' (Save) button.

Rys. 8.29: Widok formularza oceny rywali

Rozdział 9

Perspektywy rozwoju

Podczas trwania prac nad projektem narodziło się wiele pomysłów mogących urozmaicić działanie platformy. W niniejszym rozdziale zostały przedstawione wybrane kierunki, które mogłyby wpłynąć na atrakcyjność oraz jakość usług dostarczanych przez system.

9.1. Kontynuacja testów użyteczności

W ramach dalszego rozwoju systemu mogłyby zostać podjęta kontynuacja testów walidacyjnych. Dotychczas przeprowadzone badania moderowane sprawdzały intuicyjność poszczególnych procesów, a kryterium oceny była poprawność podjętych przez użytkownika działań. Kolejny etap testów mógłby dodatkowo sprawdzać ile czasu zajmuje użytkownikom wykonanie poszczególnych akcji. Analiza wyników czasowych mogłyby posłużyć identyfikacji obszarów, które wymagają przebudowy w celu przyśpieszenia interakcji.

9.2. Aplikacja mobilna

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

9.3. 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. Należałyby również ograniczać dostęp do takich obiektów, na przykład poprzez wprowadzenie wirtualnej wewnętrz-systemowej waluty w postaci tokenów, którymi drużyny płaciłyby za rezerwację obiektów. Drużyny mogłyby otrzymywać tokeny co jakiś czas oraz za zakończone wyzwania.

9.4. System rankingowy

Interesującym kierunkiem rozwoju jest zdefiniowanie systemu rankingowego drużyn. Wyniki rozgrywek pomiędzy drużynami mogłyby wpływać na pozycję rankingową drużyny. Pozycja

rankingowa drużyny mogłyby 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. Rywalizacja o miejsca rankingowe mogłyby stanowić dodatkową motywację do rozwoju drużyn.

9.5. Zwiększenie możliwości zawodników

W zaimplementowanym systemie większość akcji dotyczących poszukiwania przeciwników, negocjacji spotkania, wprowadzania wyników oraz ocen rywali odbywa się za pośrednictwem kapitana. Po założeniu profilu zawodnika oraz dołączeniu do drużyny funkcjonalności zawodników, nie będących kapitanami, ograniczają się do przeglądania społeczności, przeglądania wyzwań oraz wprowadzania obiektów sportowych. Wpływ zawodników na poszczególne funkcjonalności mógłby zostać zwiększyony poprzez:

- umożliwienie zawodnikom szukania rywali oraz proponowania spotkań do akceptacji przez kapitanów,
- umożliwienie zawodnikom głosowania na poszczególne oferty czasu oraz miejsca spotkania wprowadzane przez kapitanów.

9.6. Czat

W celu usprawnienia komunikacji pomiędzy drużynami umawiającymi się na mecz mógłby zostać wprowadzony dodatkowy system wymiany informacji w formie czatu. Luźna wymiana wiadomości mogłyby również posłużyć wstępemu zapoznaniu się drużyn, a co za tym idzie zwiększeniu komfortu oraz pewności pierwszego spotkania.

Rozdział 10

Podsumowanie

Przeprowadzenie projektu pozwoliło na uzyskanie efektu końcowego, w postaci zaawansowanego prototypu systemu pozwalającego na budowę społeczności sportowców, poszukiwanie dobrze dopasowanych rywali oraz umawianie meczów.

Praca nad projektem została podzielona na etapy: analizy problemu, analizy istniejących rozwiązań, pracy koncepcyjnej, wykonania projektu technicznego, implementacji rozwiązania, weryfikacji rozwiązania oraz jego validacji. Ostatnie trzy etapy odbywały się jednocześnie.

Podczas poszczególnych etapów projektu korzystano ze sprawdzonych standardów, technologii oraz metodyk działania. Miało to duży wpływ na sprawny rozwój projektu. W przypadku napotkanych problemów oraz niejasności bardzo przydatna okazywała się literatura w postaci dokumentacji oraz artykułów naukowych.

Aplikacja została zrealizowana z myślą o rozwoju. Część serwerowa umożliwia dynamiczne wprowadzanie nowych dyscyplin sportowych oraz regionów działania. Poszczególne komponenty systemu zostały zaimplementowane zgodnie z zasadą OCP (ang. *Open/closed principle*). Wytworzone oprogramowanie jest otwarte na rozszerzenia wynikające ze zmieniających się oczekiwani grupy docelowej. Przykładem może być zaimplementowany algorytm optymalizacji wielokryterialnej, którego struktura klas pozwala na łatwe wprowadzanie nowych kryteriów. Duży wpływ na możliwości rozwojowe ma również rozdzielenie części klienckiej oraz serwerowej. Aplikacja serwerowa została zrealizowana jako niezależny oraz kompletny komponent, który realizuje całą logikę biznesową systemu udostępniając interfejs w postaci REST API. Po stronie serwerowej nie istnieją żadne powiązania do części klienckiej. Serwer jest gotowy na obsługę różnych klientów bez względu na platformę, jedynym wymogiem jest komunikacja za pomocą protokołu HTTP.

Przeprowadzone testy użyteczności rozwiązania pozwoliły na zidentyfikowanie problemów związanych z intuicyjnością interfejsu oraz wprowadzenie usprawnień. Konfrontacja systemu we wczesnym etapie rozwoju z potencjalnymi użytkownikami pozwoliła również na lepsze zrozumienie ich oczekiwani oraz przystosowanie kierunku rozwoju platformy.

Uzyskany produkt końcowy może stanowić wartą rozpatrzenia alternatywę dla istniejących rozwiązań. Głównym elementem wyróżniającym system *Team Challenge* spośród innych dostępnych propozycji jest nowatorskie wspomaganie poszukiwania rywali, które zostało zrealizowane w oparciu o zagadnienie optymalizacji wielokryterialnej. System dostarcza wiele informacji dotyczących proponowanych rywali, co wspomaga wybór dobrze dopasowanych przeciwników, a ostatecznie prowadzi do satysfakcjonujących spotkań.

Praca nad platformą była dla autora niniejszej pracy ciekawym doświadczeniem. Wynikało to z dużego zainteresowania tematem sportów zespołowych oraz chęcią rozwoju w obrębie systemów internetowych. Rozwiązywanie napotkanych problemów pozwoliło na poszerzenie wiedzy na temat wykorzystywanych technologii oraz wiedzy ogólnej dotyczącej inżynierii systemów informatycznych.

Literatura

- [1] Angular docs: Architecture overview.
<https://angular.io/guide/architecture>. [Online; dostęp 11 listopada 2018].
- [2] NG-ZORRO - Getting Started.
<https://ng.ant.design/docs/getting-started/en>. [Online; dostęp 12 października 2018].
- [3] Playarena.pl - Nasza misja.
<http://playarena.pl/corobimy>. [Online; dostęp 24 października 2018].
- [4] SportsMatchMaker.com.au - About us.
<http://www.sportsmatchmaker.com.au/aboutus.aspx>. [Online; dostęp 25 października 2018].
- [5] J. Bloch. *Effective Java, 3rd Edition*. Pearson Education, 2017.
- [6] 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].
- [7] B. Kern, L. Bahl. What is moderated testing?
<https://www.usertesting.com/blog/moderated-testing-101/>. [Online; dostęp 11 października 2018].
- [8] 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 3 listopada 2018].
- [9] J. Nielsen. How many test users in a usability study?
<https://www.nngroup.com/articles/how-many-test-users/>, 2012. [Online; dostęp 17 października 2018].
- [10] O. Semusheva. Requirements. Why is it important?
<https://steekiwi.com/blog/requirements-why-it-important/>. [Online; dostęp 15 listopada 2018].
- [11] A. Shukla. *Building Web Apps with Spring 5 and Angular 4*. Packt Publishing, 2017.
- [12] S. Swafford. Use Cases and Their Importance.
http://aspalliance.com/765_Use_Cases_and_Their_Importance, 2006. [Online; dostęp 15 listopada 2018].
- [13] G. L. Turnquist. *Learning Spring Boot 2.0 - Second Edition*. Packt Publishing, 2017.

- [14] A. University. Smart components vs presentational components.
<https://blog.angular-university.io/>. [Online; dostęp 11 listopada 2018].
- [15] M. Zuckerberg. Bringing the World Closer Together.
<https://www.facebook.com/zuck/posts/10154944663901634>, 2017. [Online; dostęp 3 listopada 2018].

Dodatek A

Opis załączonej płyty CD

W poniższych podpunktach opisano zawartość poszczególnych katalogów, które można odnaleźć na załączonej płycie CD.

- **1-dokument** - Katalog zawierający niniejszą pracę dyplomową w formacie PDF (ang. *Portable Document Format*).
- **2-kod-aplikacji-serwerowej** - Katalog zawierający kompletny kod aplikacji serwerowej wraz z metadanymi systemu kontroli wersji *GIT*.
- **3-skrypty-baza** - Katalog zawierający skrypty SQL służące do wdrożenia schematu bazy oraz wgrania przykładowych danych.
- **4-kod-aplikacji-klienckiej** - Katalog zawierający kompletny kod aplikacji klienckiej wraz z metadanymi systemu kontroli wersji *GIT*.
- **5-zrzuty-ekranu** - Katalog zawierający zrzuty ekranu prezentujące działanie zaimplementowanego systemu.
- **6-ankiety** - Katalog zawierający ankiety wypełnione przez uczestników badań moderowanych oraz notatki moderatora.
- **7-dostepy** - Katalog zawierający plik tekstowy z dostępami w postaci adresu wdrożonego systemu oraz danych dostępowych użytkowników testowych.