

# System wspomagający organizację ligi/turnieju sportowego.

02.11.2018

Jakub Pranica, Sebastian Donchór

Studenci Politechniki Krakowskiej

## Przegląd

Aplikacja ma za zadanie ułatwiać organizację ligi bądź turnieju w dowolną grę jeden-na-jeden (drużyna-na-drużynę) według wybranego systemu zaimplementowanego w aplikacji. Osoba zalogowana z uprawnieniami organizatora może tworzyć wydarzenia i wprowadzać wyniki, natomiast osoba z uprawnieniami zawodnika może przeglądać swoją punktację w konkretnych wydarzeniach lub ogólny ranking w danej federacji.

## Kontekst

Organizacja wielodrużynowej rywalizacji sportowej to bardzo wiele obowiązków, od których zależy sukces przedsięwzięcia. Jednym z problemów jest gromadzenie i przetwarzanie informacji na temat wydarzeń, zawodników, wyników spotkań itp.

Wiele znanych gier rozgrywanych jest w formie spotkań dwóch rywalizujących ze sobą stron, których wynikiem jest zwycięstwo jednej z nich lub remis. Oznacza to, że w jednym momencie, na jednym obszarze gry tylko dwóch przeciwników może porównać swoje umiejętności. Przez lata powstało wiele systemów mających na celu wyłonienie najlepszego z grupy zawodników poprzez rozegranie serii takich spotkań. Głównymi problemami w stosowaniu tych systemów są:

- dobór par, które powinny rozegrać mecz między sobą w kolejnych rundach
- złożoność tych systemów i natłok informacji, co utrudnia ich ręczne nadzorowanie

Jeszcze innym problemem, do którego chcemy nawiązać jest długoterminowość - wiele organizacji chce aby wyniki w jednym wydarzeniu wpływały na pozycję startową w kolejnym.

## Cele

Celem projektu jest stworzenie systemu pozwalającego na zautomatyzowanie procesów koniecznych do sprawnej organizacji wydarzeń sportowych, głównie na płaszczyźnie punktacji i dobierania par meczowych.

Organizator powinien mieć wybór pomiędzy kilkoma systemami rozgrywania turniejów:

1. Implementacja systemu pucharowego.
2. Implementacja systemu szwajcarskiego.
3. Implementacja systemu kołowego.

W celu zaadresowania problemu długoterminowości i dobierania par startowych wprowadzimy ranking pozwalający na ułożenie graczy według ich umiejętności:

4. Implementacja rankingu Elo.
5. Implementacja systemu McMahona na podstawie Elo.

System powinien zawierać również wsparcie dla długoterminowych lig podzielonych na sezony:

6. Implementacja systemu ligowego - wsparcie dla wieloklasowych lig (szczeble ligowe, spadki, awanse).

Po utworzeniu wydarzenia organizator powinien mieć możliwość ustalenia daty spotkań i wprowadzania wyników:

7. Stworzenie terminarza pozwalającego na planowanie meczów na konkretny dzień i godzinę.
8. Interfejs pozwalający na przeglądanie i wprowadzanie wyników.

Osoba niezalogowana będzie mogła przeglądać wyniki meczów i punktację w konkretnych wydarzeniach oraz ogólny ranking w federacji.

## Nie-cele (Non-Goals)

Sekcja ta ma za zadanie opisać problemy, których nie będziemy poruszać, tak aby wszyscy zainteresowani byli po tej samej stronie.

1. Chcemy skupić się na grach, których wynikiem jest wygrana jednej z dwóch stron lub remis. Odpadają więc zawody gdzie większa ilość przeciwników rywalizuje jednocześnie lub wyniki indywidualnie są porównywane ze wszystkimi (skoki narciarskie, lekkoatletyka).

2. Zapisy będą odbywać się poza aplikacją - to organizator tworzy profile zawodników. Nie chcemy zmuszać zawodników do samodzielnego rejestrowania się w systemie - byłoby to problematyczne również dla organizatorów, którzy musieliby czekać na działanie graczy i podejmować ich weryfikację.
3. Informacje związane z wydarzeniami będą publiczne, więc nie ma potrzeby by zawodnicy logowali się do systemu. Funkcja logowania będzie przeznaczona dla organizatorów chcących uzyskać dostęp do panelu zarządzania. Dostęp do rejestracji kont dla organizatorów będzie posiadała jedynie osoba zalogowana jako główny administrator systemu.

## Obecne rozwiązanie

Obecnie wciąż wiele organizacji korzysta z liczenia i przechowywania punktacji w programach typu Excel czy nawet na kartce papieru. Organizatorzy decydujący się na "ręczne" rozwiązanie przedstawionych problemów muszą liczyć się z czasochłonnością, pomyłkami, brakiem integracji powiązanych ze sobą procesów i informacji.

Aktualnie istniejące rozwiązania na rynku skupiają się na pojedynczych wydarzeniach i systemach, rozwiązując tylko część powiązanych ze sobą problemów.

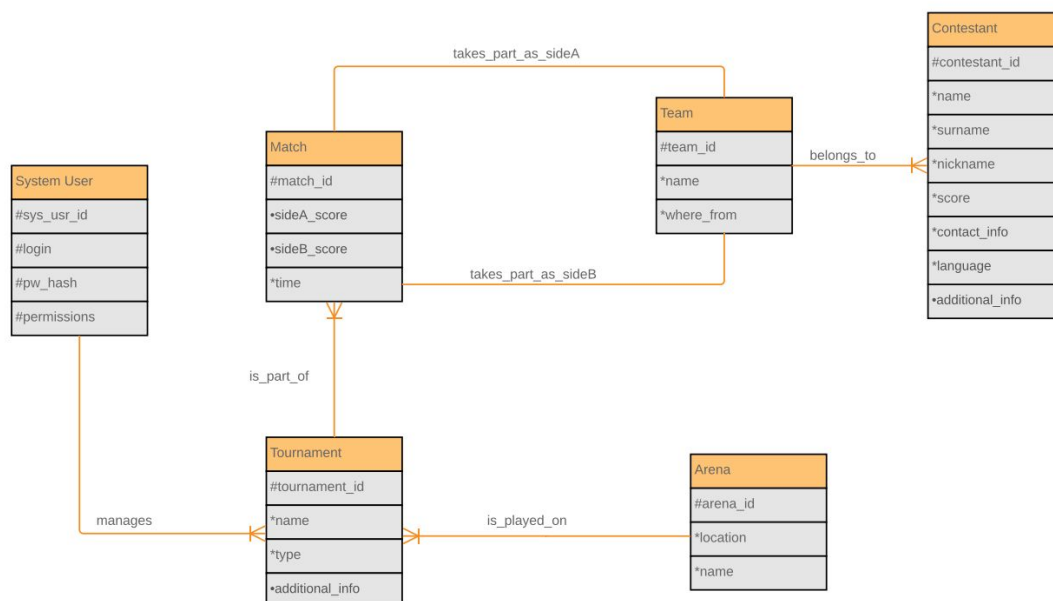
## Proponowane rozwiązanie

My natomiast proponujemy rozwiązanie łączące ze sobą system rankingowy wraz z systemami turniejowymi i ligowymi co ułatwi pracę federacji organizującej wiele wydarzeń w ciągu sezonu.

## Dane techniczne

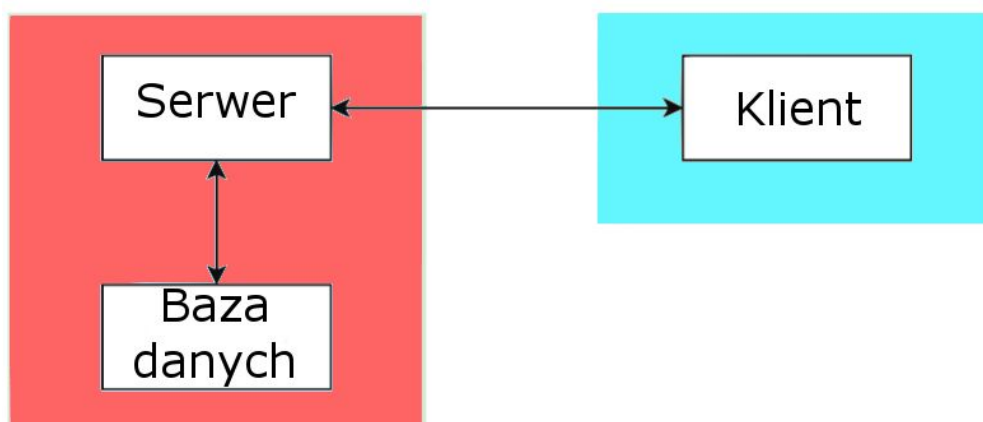
Organizacja korzystająca z aplikacji będzie miała podpiętą do systemu bazę danych przechowującą dane dotyczące zawodników i ich punktacji w wydarzeniach.

### Entity Relationship Diagram (wersja prototypowa)



## Architektura systemu

Baza danych zarządzana będzie przez aplikację serwerową obsługującą zapytania od Klienta. Aplikacja kliencka ma się skupiać na przyjmowaniu poleceń od użytkownika, wysyłaniu ich do serwera i wyświetlaniu wyników po otrzymaniu odpowiedzi. Użytkownik z uprawnieniami organizatora będzie miał możliwość tworzenia profili zawodników, lig, turniejów oraz możliwość zaplanowania meczów w terminarzu. Wszystkie wydarzenia będą odbywały się w ramach jednej federacji prowadzącej automatycznie ranking Elo zawodników, co pozwoli na implementację dobierania par meczowych według umiejętności.



Architektura typu klient-serwer pozwoli na jednoczesny dostęp wielu użytkowników z różnymi uprawnieniami.

## Technologia

Dwie aplikacje w Javie - jedna dla serwera, druga dla klienta. GUI za pomocą JavaFX. Baza danych MySQL lokalna lub zewnętrzna.

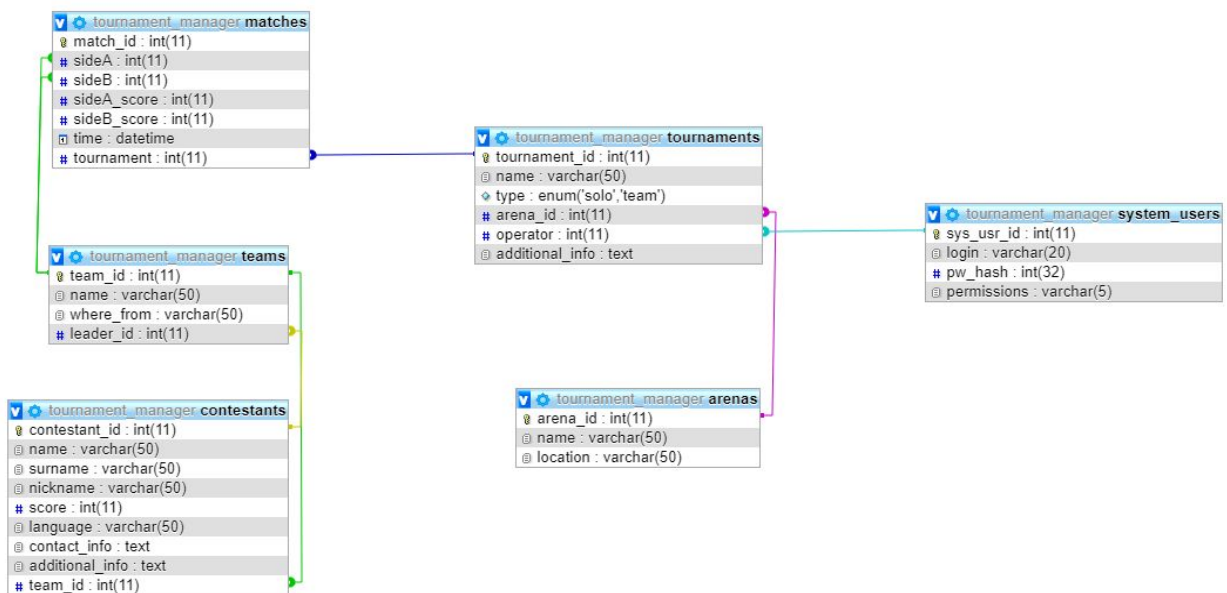
## Opis bazy danych w języku SQL

```
CREATE TABLE `arenas` (`arena_id` int(11) NOT NULL, `name` varchar(50) COLLATE utf16_polish_ci NOT NULL, `location` varchar(50) COLLATE utf16_polish_ci NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf16 COLLATE=utf16_polish_ci;
CREATE TABLE `contestants` (`contestant_id` int(11) NOT NULL, `name` varchar(50) COLLATE utf16_polish_ci NOT NULL, `surname` varchar(50) COLLATE utf16_polish_ci NOT NULL, `nickname` varchar(50) COLLATE utf16_polish_ci NOT NULL, `score` int(11) NOT NULL, `language` varchar(50) COLLATE utf16_polish_ci NOT NULL, `contact_info` text COLLATE utf16_polish_ci NOT NULL, `additional_info` text COLLATE utf16_polish_ci, `team_id` int(11) DEFAULT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf16 COLLATE=utf16_polish_ci;
CREATE TABLE `matches` (`match_id` int(11) NOT NULL, `sideA` int(11) NOT NULL, `sideB` int(11) NOT NULL, `sideA_score` int(11) DEFAULT NULL, `sideB_score` int(11) DEFAULT NULL, `arena_id` int(11) NOT NULL, `time` datetime NOT NULL, `tournament` int(11) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf16 COLLATE=utf16_polish_ci;
CREATE TABLE `system_users` (`sys_usr_id` int(11) NOT NULL, `login` varchar(20) COLLATE utf16_polish_ci NOT NULL, `pw_hash` varchar(32) NOT NULL, `permissions` varchar(5) COLLATE utf16_polish_ci DEFAULT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf16 COLLATE=utf16_polish_ci;
CREATE TABLE `teams` (`team_id` int(11) NOT NULL, `name` varchar(50) COLLATE utf16_polish_ci NOT NULL, `where_from` varchar(50) COLLATE utf16_polish_ci NOT NULL, `leader_id` int(11) DEFAULT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf16 COLLATE=utf16_polish_ci;
CREATE TABLE `tournaments` (`tournament_id` int(11) NOT NULL, `name` varchar(50) COLLATE utf16_polish_ci NOT NULL, `type` enum('solo','team') COLLATE utf16_polish_ci NOT NULL, `operator` int(11) NOT NULL, `additional_info` text COLLATE utf16_polish_ci) ENGINE=InnoDB DEFAULT CHARSET=utf16 COLLATE=utf16_polish_ci;
ALTER TABLE `arenas`
  ADD PRIMARY KEY (`arena_id`);
ALTER TABLE `contestants`
  ADD PRIMARY KEY (`contestant_id`),
  ADD KEY `team_id` (`team_id`);
ALTER TABLE `matches`
  ADD PRIMARY KEY (`match_id`),
  ADD KEY `sideA` (`sideA`),
  ADD KEY `arena_id` (`arena_id`),
  ADD KEY `sideB` (`sideB`),
  ADD KEY `tournament` (`tournament`);
ALTER TABLE `system_users`
  ADD PRIMARY KEY (`sys_usr_id`);
ALTER TABLE `teams`
  ADD PRIMARY KEY (`team_id`),
  ADD KEY `leader_id` (`leader_id`);
```

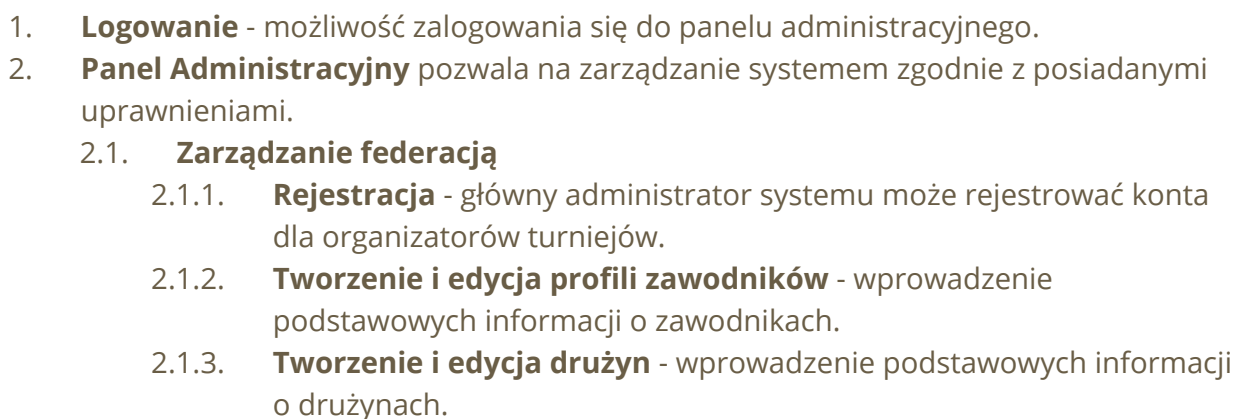
```


ALTER TABLE `tournaments`
  ADD PRIMARY KEY (`tournament_id`),
  ADD KEY `operator` (`operator`);
ALTER TABLE `contestants`
  MODIFY `contestant_id` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `arenas`
  MODIFY `arena_id` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `tournaments`
  MODIFY `tournament_id` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `system_users`
  MODIFY `sys_usr_id` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `matches`
  MODIFY `match_id` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `teams`
  MODIFY `team_id` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `contestants`
  ADD CONSTRAINT `contestants_ibfk_1` FOREIGN KEY (`team_id`) REFERENCES `teams` (`team_id`);
ALTER TABLE `matches`
  ADD CONSTRAINT `matches_ibfk_1` FOREIGN KEY (`sideA`) REFERENCES `teams` (`team_id`),
  ADD CONSTRAINT `matches_ibfk_2` FOREIGN KEY (`sideB`) REFERENCES `teams` (`team_id`),
  ADD CONSTRAINT `matches_ibfk_4` FOREIGN KEY (`arena_id`) REFERENCES `arenas` (`arena_id`),
  ADD CONSTRAINT `matches_ibfk_3` FOREIGN KEY (`tournament`) REFERENCES `tournaments`
  (`tournament_id`);
ALTER TABLE `teams`
  ADD CONSTRAINT `teams_ibfk_1` FOREIGN KEY (`leader_id`) REFERENCES `contestants` (`contestant_id`);
ALTER TABLE `tournaments`
  ADD CONSTRAINT `tournaments_ibfk_2` FOREIGN KEY (`operator`) REFERENCES `system_users`
  (`sys_usr_id`);

```



## Diagram wymagań funkcjonalnych

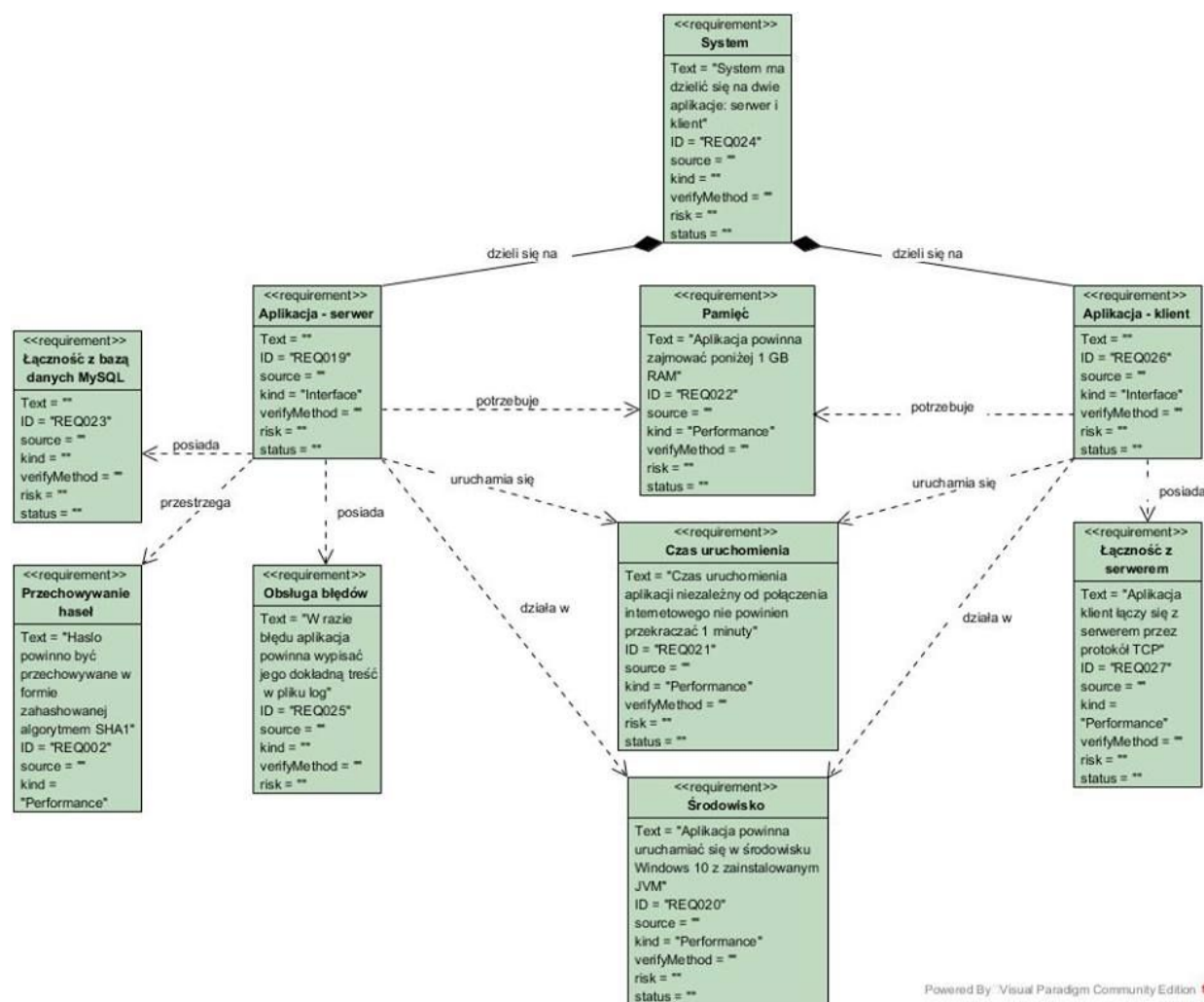


- 
- 2.1.4. **Tworzenie wydarzenia** - formularz pozwalający na utworzenie turnieju według wybranego systemu.
  - 2.2. **Zarządzanie wydarzeniem**
    - 2.2.1. **Dodawanie i usuwanie uczestników**
    - 2.2.2. **Ustalanie terminów meczów**
    - 2.2.3. **Wprowadzanie wyników meczów**
  3. **Obliczanie rankingu Elo** - po wprowadzeniu wyniku meczu, powinna nastąpić aktualizacja rankingu dla obu stron.
  4. **Wyświetlanie profili zawodników** - wyświetlenie informacji o danym zawodniku (w tym jego rankingu).
  5. **Wyświetlanie profili drużyn** - wyświetlenie informacji o danej drużynie.
  6. **Przegląd terminarza** - wyświetlanie zbioru wydarzeń wraz z ich datami.
  7. **Wyświetlanie profili meczów** - wyświetlenie informacji o danym meczu.
  8. **Implementacja systemu pucharowego** - turniej pojedynczej eliminacji.
    - 8.1. **Podział turnieju na rundy** - lista meczów do zaplanowania w każdej rundzie - liczba spotkań zmniejsza się o połowę wraz z każdą następną rundą.
  9. **Implementacja systemu szwajcarskiego**
    - 9.1. **Podział turnieju na rundy** - lista meczów do zaplanowania w każdej rundzie - liczba rund jest z góry określona.
    - 9.2. **Dobór par przeciwników** w kolejnych rundach zależy od wyników uzyskanych w poprzednich.
    - 9.3. **Tabela wyników**
    - 9.4. **Możliwość zastosowania odmiany McMahona** - gracze otrzymują dodatkowe punkty początkowe zależnie od ich rankingu.
  10. **Implementacja systemu kołowego**
    - 10.1. **Podział turnieju na kolejki** - lista meczów do zaplanowania w każdej rundzie - powinno być ich tyle aby każdy zagrał z każdym.
    - 10.2. **Tabela wyników**
  11. **System ligowy** - implementacja systemu ligowego korzystającego z systemu kołowego.
    - 11.1. **Podział na klasy rozgrywkowe** - osobne tabele wyników dla każdej grupy.
      - 11.1.1. **Spadki i awanse** - po zakończeniu sezonu następuje aktualizacja przydziału do poszczególnych klas.



## Wymagania funkcjonalne

### Diagram wymagań niefunkcyjnych



#### 1. Aplikacja serwerowa

- 1.1. **Przechowywanie haseł** - w formie zahashowanej SHA1.
- 1.2. **Łączność z bazą danych** - aplikacja posiada łączność z lokalnym lub zewnętrznym serwerem bazy danych MySQL.
- 1.3. **Obsługa błędów** - ewentualne błędy są jasno opisywane i zapisywane w pliku log.

#### 2. Aplikacja kliencka

- 2.1. **Łączność z serwerem** - bez połączenia z aplikacją serwerową na maszynie lokalnej lub zewnętrznej, aplikacja kliencka jest bezużyteczna.

#### 3. Obie aplikacje

- 3.1. **Środowisko** - uruchamianie przynajmniej w jednym systemie operacyjnym - Windows 10.

- 3.2. **Czas uruchomienia** - część aplikacji niezależna od połączenia internetowego powinna uruchamiać się poniżej 1 minuty.
- 3.3. **Pamięć operacyjna** - aplikacja nie powinna zajmować więcej niż 1GB pamięci operacyjnej komputera.

## Charakterystyka użytkowników

### Główny administrator systemu

Użytkownik ten może rejestrować konta dla organizatorów turniejów i zarządzać nim. Główny administrator posiada również uprawnienia organizatora.

### Organizator turnieju

Użytkownik ten po uzyskaniu dostępu od głównego administratora może tworzyć wydarzenia turniejowe, przeglądać i edytować dane dotyczące zarządzanych przez siebie wydarzeń, dodawać i usuwać uczestników, zaplanować terminy meczów, zmienić swoje hasło.

### Użytkownik niezalogowany

Może przeglądać wyniki meczów i punktację w konkretnych wydarzeniach oraz ogólny ranking w federacji.

## Alternatywne rozwiązania

Innym rozwiązaniem, nad którym myśleliśmy było stworzenie platformy łączącej zawodników z wieloma federacjami. Zawodnicy samodzielnie tworzyliby ogólne konta (do każdej dyscypliny) i sami zapisywaliby się do wydarzenia. W takim przypadku byłaby jedna wspólna baza danych dla wielu federacji, korzystających z naszej aplikacji.

Takie rozwiązanie pozbawiłoby jednak organizatorów dużej części kontroli. Wymagałoby to wysyłania próśb do zawodników, aby zarejestrowali się w systemie i wyszukali jedno z wielu wydarzeń. Pozwoliłoby to na stworzenie globalnego rankingu dla jednej dyscypliny, jednak nie każda federacja chciałaby współdzielić dane na temat zawodników z konkurencją.

Dlatego zdecydowaliśmy się na rozwiązanie dla pojedynczych organizacji opisane wyżej.

Inną kwestią, nad którą się zastanawialiśmy, było:

- Czy zawodnicy powinni mieć możliwość/konieczność zalogowania się?  
Jeżeli nie, to prawie wszystkie informacje muszą być publiczne.  
Jeżeli tak, to problemem staje się tworzenie kont:
  - Czy zawodnicy mają tworzyć konta sami czy tworzy je organizator?  
Jako iż skłaniamy się ku rozwiązaniu by zapisy odbywały się poza aplikacją, tworzenie kont przez użytkowników nie miałoby sensu, ponieważ i tak wymagałoby to weryfikacji przez administratora i przypisania przez niego konta do odpowiedniego profilu.  
Natomiast tworzenie kont dostępowych przez organizatora również jest problemowe z powodu uzgadniania i udostępniania haseł.

Ostatecznie zdecydowaliśmy, że zawodnicy nie będą musieli się logować, a samo logowanie będzie służyć dostępowi do panelu zarządzania.

## Oś czasu

### 05.11.2018

- Opis problemu, cel projektu, założenia projektowe, zakres produktu;
- Prototypowa wersja diagramu ERD;

### 12.11.2018

- Wymagania projektowe (funkcjonalne, нефunkcjonalne, interfejsy);
- Charakterystyka użytkowników;
- Główne funkcje produktu;

### 16.11.2018

- Podział projektu na moduły - podział osobowy;

### 26.11.2018

- Uaktualniona wersja diagramu ERD;
- Diagram Klas;
- 30% aplikacji;

### 03.12.2018

- Opis bazy danych i jej projektowania;
- Architektura systemu;

### 17.12.2018

- Diagram przepływu danych (DFD);

- Diagram stanu (STD);

**04.01.2019**

- Raport z działania bazy danych:
  - Przykładowe zapytania do systemu z wyjaśnieniem ich celu w języku polskim;
  - Przykłady planów wykonania zapytań i optymalizacji bazy danych w celu poprawienia czasu realizacji zapytań;
- Dokumentacja Użytkownika;

## Kamienie milowe

### I. Uzupełnienie specyfikacji projektu

- Wymagania projektowe
- Sprecyzowanie głównych funkcji produktu
- ER-diagram
- Opisy procesów biznesowych
- Charakterystyka użytkowników
- Podział projektu na moduły - podział osobowy

### II. Podstawowe wersje aplikacji klienckiej i serwerowej

W wersji podstawowej aplikacja serwerowa zawierałaby możliwość połączenia się z zewnętrznym serwerem bazodanowym MySQL, wprowadzania danych do tabel, odpowiadania na połączenia TCP od aplikacji klienckiej i przyjmowania danych po zweryfikowaniu logowania. Aplikacja kliencka zawierałaby tymczasową możliwość wysyłania MySQL Query do aplikacji serwerowej.

### III. System logowania

Hasła przechowywane w formie zahashowanej.

### IV. Prototypowa wersja GUI

### V. Edytor profili zawodników

### VI. Edytor wydarzeń

### VII. Terminarz



VIII. Implementacja podstawowych systemów turniejowych

IX. Implementacja rankingu Elo

X. Implementacja systemów korzystających z Elo

XI. System wieloklasowych lig