Dokument SRS

System wspomagający organizację ligi/turnieju sportowego.

12.01.2019

Jakub Pranica, Sebastian Donchór
Studenci Politechniki Krakowskiej

Spis treści

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.

Słownik

- **SWOLTS** - System wspomagający organizację ligi/turnieju sportowego.
- **GUI** - (od angielskiego graphical user interface) - inaczej interfejs graficzny. Określa sposób prezentacji informacji przez komputer oraz interakcji z użytkownikiem, poprzez wyświetlane obiekty.
- **Ranking Elo** – metoda obliczania relatywnej siły gry szachistów w punktacji elo. Nazwa „elo” pochodzi od nazwiska Arpada Elo – amerykańskiego naukowca węgierskiego pochodzenia, którego prace ukształtowały szachowy system rankingowy oparty na naukowych podstawach. System został zaimplementowany do wielu innych gier np. piłki nożnej.

- **Terminarz** - w kontekście aplikacji rozumiany w dwóch formach:
  1. Lista wszystkich niezakończonych, ale zaplanowanych meczów w federacji.
  2. Karta pozwalająca na wprowadzanie daty i areny wybranego, jeszcze niezaplanowanego meczu.
- **Federacja** - organizacja zajmująca się zarządzaniem i kontrolą wydarzeń związanych z pewną dyscypliną sportu/rywalizacji. Federacje często gromadzą informacje o uczestnikach lig i turniejów, których są patronatem np. ranking zawodników.

Następujące pojęcia dotyczą sposobów rozgrywania zawodów sportowych w dyscyplinach, w których rywalizacja polega na bezpośrednich pojedynkach między uczestnikami:

- **System pucharowy** – Uczestnicy rozgrywają mecze (gry) pomiędzy sobą, po których zwycięzca kwalifikuje się do dalszych gier, pokonany odpada z rywalizacji.
- **System szwajcarski** – W tym systemie z góry określa się liczbę rund, które należy rozegrać - w jeden rundzie każdy zawodnik rozgrywa jedno spotkanie. Za zwycięstwo w grze uczestnik otrzymuje jeden punkt, za remis pół punktu. Dobór par przeciwników w kolejnych rundach zależy od wyników uzyskanych w poprzednich. Pary dobiera się w miarę możliwości spośród tych uczestników, którzy dotychczas zdobyli jednakową liczbę punktów.
- **System McMahona** – odmiana systemu szwajcarskiego wykorzystująca system rankingowy. Przed rozpoczęciem rozgrywek gracze są szeregowani na podstawie rankingu, a następnie dzieleni na grupy w zależności od siły. W każdej grupie gracze otrzymują pewną liczbę punktów początkowych (najsłabsi dostają po 0 punktów, nieco silniejsi - po 1 punkt, następni - po 2 itd.). W naszym przypadku przyjmujemy 1 punkt za każde 100 punktów ELO.
- **System kołowy** – W tym systemie każdy uczestnik gra kolejno ze wszystkimi przeciwnikami. Ten sposób rozgrywek nazywany jest również systemem każdy z każdym, w anglojęzycznej literaturze round-robin.

Poniższy tekst opisuje pojęcia związane z systemem określonym jako **“Wieloklasowa liga”**:

Rozgrywki ligowe organizuje się regularnie – zazwyczaj raz do roku – a pełny cykl rozgrywkowy nazywa się **sezonem**. Zespoły uczestniczące w rozgrywkach spotykają się według określonego terminarza – jedna seria meczów nazywana jest **kolejką**. Na podstawie wyników meczów w poszczególnych kolejkach tworzone są **raporty** – lista zdobytych punktów przez poszczególnych uczestników. Mistrzem z reguły zostaje drużyna, która po rozegraniu wszystkich zaplanowanych kolejek posiada największą ilość punktów.

W przypadku większej liczby drużyn uczestniczących w rozgrywkach ligowych zachodzi konieczność ich zhierarchizowania, czyli podzielenia na poszczególne poziomy ligowe (szczeble ligowe), nazywane potocznie **„ligami”**, bądź **„klasami rozgrywkowymi”**. Przynależność drużyny do określonej ligi, bądź klasy wynika zazwyczaj z poziomu sportowego prezentowanego w poprzednim sezonie lub – rzadziej – od warunków finansowych, infrastruktury i innych czynników. W niemal wszystkich systemach ligowych

obowiązuje zasada **spadków i awansów**, czyli cosezonowa aktualizacja składów klas rozgrywkowych.

Rozgrywki ustala się zazwyczaj zgodnie z **systemem kołowym**, czyli w ciągu sezonu każda z drużyn spotyka się z każdą inną, występującą na tym samym szczeblu ligowym.

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 funkcji 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.

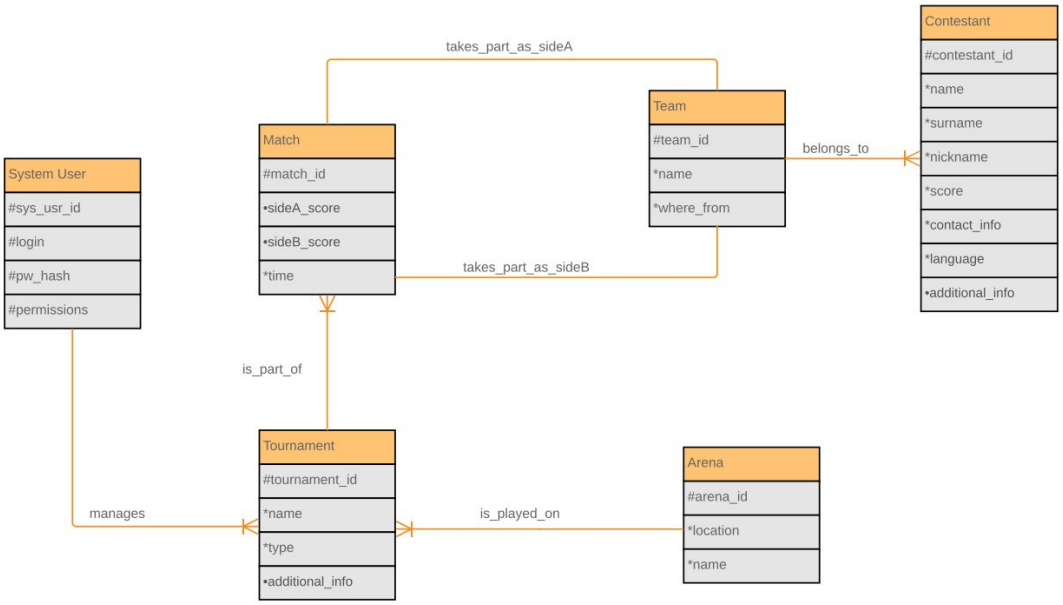
Proponowane rozwiązanie

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

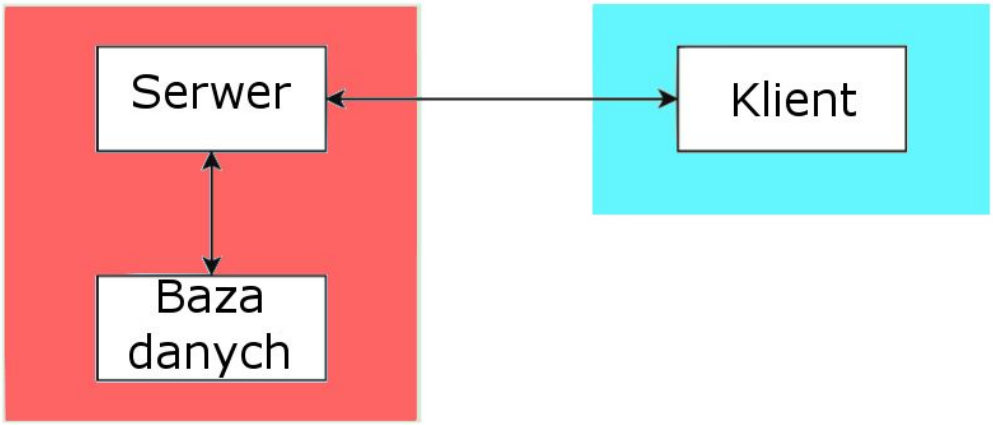
Organizacja korzystająca z aplikacji będzie miała podpiętą do systemu bazę danych przechowującą dane dotyczące zawodników i ich punktacji w wydarzeniach.

Entity Relationship Diagram (wersja prototypowa)



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.

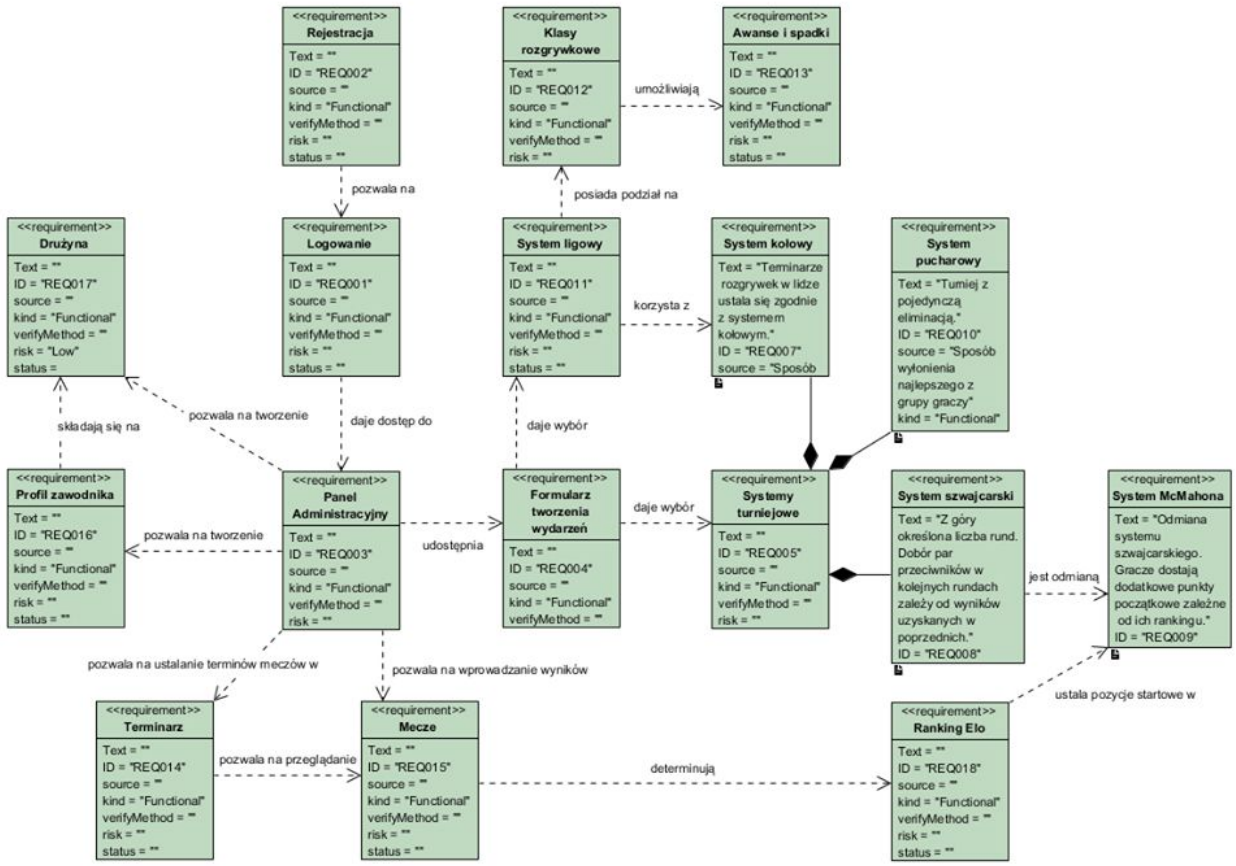
Technologia

Dwie aplikacje w Javie - jedna dla serwera, druga dla klienta. GUI za pomocą JavaFX.

Wymagania i funkcjonalność produktu

Wymagania funkcjonalne

Diagram wymagań funkcjonalnych



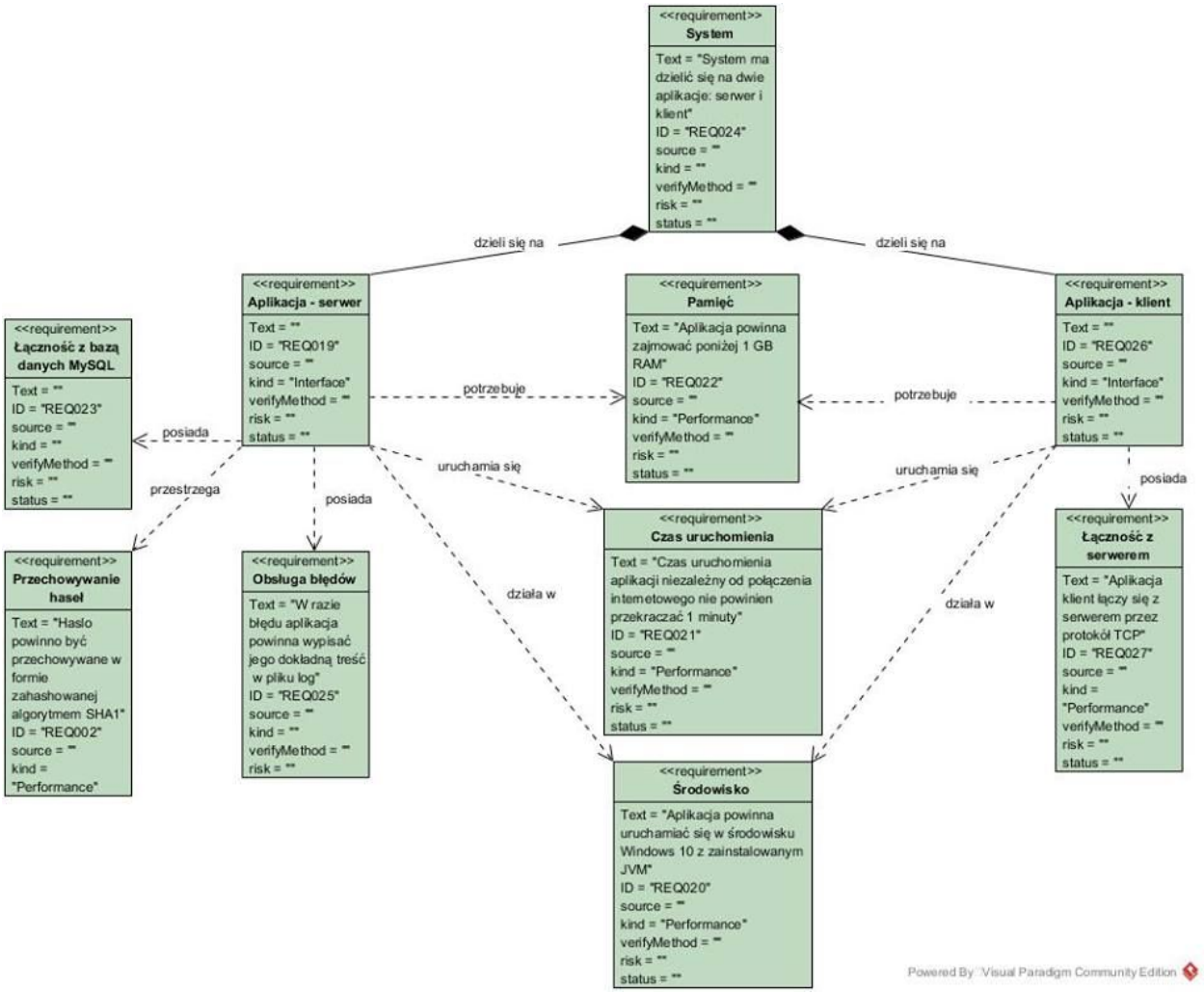
1. **Logowanie** - możliwość zalogowania się do panelu administracyjnego.
2. **Panel Administracyjny** pozwala na zarządzanie systemem zgodnie z posiadanymi uprawnieniami.
    2.1. **Zarządzanie federacją**
        2.1.1. **Rejestracja** - główny administrator systemu może rejestrować konta dla organizatorów turniejów.
        2.1.2. **Tworzenie i edycja profili zawodników** - wprowadzenie podstawowych informacji o zawodnikach.

2.1.3. **Tworzenie i edycja drużyn** - wprowadzenie podstawowych informacji o drużynach.
2.1.4. **Tworzenie wydarzenia** - formularz pozwalający na utworzenie turnieju według wybranego systemu.
2.2. **Zarządzanie wydarzeniem**
2.2.1. **Dodawanie i usuwanie uczestników** w trakcie zapisów (przed rozpoczęciem turnieju lub sezonu).
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 zaplanowanych meczów.
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.
8.2. **Lista zawodników pozostałych w turnieju** - generowana w formie raportu, po każdej rundzie.
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. **Wyniki w formie raportów** - punktacja zawodników po każdej rundzie.
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. **Wyniki w formie raportów** - punktacja zawodników po każdej kolejce.
11. **System ligowy** - implementacja systemu ligowego korzystającego z systemu kołowego.
11.1. **Podział na klasy rozgrywkowe** - przedstawienie wyników osobno dla każdej grupy.
11.1.1. **Spadki i awanse** - po zakończeniu sezonu następuje możliwość aktualizacji przydziału uczestników do poszczególnych klas.

Wymagania funkcjonalne

Diagram wymagań niefunkcjonalnych



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

Aplikacja jest skierowana przede wszystkim do organizatorów turniejów w ramach jednej federacji wykorzystującej nasz system, ale może być również wykorzystywana przez innych członków organizacji w celu sprawdzania wyników zawodników.

Rozróżniamy następujące rodzaje dostępu do aplikacji:

Główny administrator systemu

Użytkownik ten może rejestrować konta dla organizatorów turniejów. 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 oraz inne obiekty, zaplanować terminy meczów, zatwierdzać wyniki, 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.

Baza danych

Baza danych została wykonana w systemie MySQL ze względu na jego prostotę obsługi oraz popularność, co przekłada się na jego dostępność w serwisach hostingowych. W celu testów optymalizacyjnych baza została wypełniona danymi ze strony filldb.info. Ilość wypełnionych rekordów została dobrana tak, aby zasymulować realne użycie aplikacji.

Table	Rows
arenas	1000
contestants	1000
matches	1000
system_users	100
teams	500
tournaments	300

Przykładowe zapytania i ich plany wykonania

Wyświetlenie nadal nierozegranych meczy w kolejności od najwcześniejszego.

*SELECT * FROM `matches` WHERE sideA_score IS NULL ORDER BY time DESC*

Jeśli punkty strony A ustawione są jako NULL, to punkty strony B z założenia także są NULL. Oznacza to, że mecz jeszcze się nie odbył.

Wykonanie zapytania trwało 0.0011 sekundy.

Plan zapytania

select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
SIMPLE	matches	ALL	*NULL*	*NULL*	*NULL*	*NULL*	1000	Using where; Using filesort

---

Wyszukiwanie zawodników po nazwisku

*SELECT * FROM `contestants` WHERE surname = 'Abernathy'*

Zapytanie zwraca zawodników o nazwisku Abernathy.

Wykonanie zapytania trwało 0.0007 sekundy.

Plan zapytania

select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
SIMPLE	contestants	ref	surname,surname_2	surname	202	const	1	Using index condition

---

Sprawdzenie, ile jest aren w danej miejscowości

SELECT COUNT(arena_id) as "Number of arenas" FROM arenas WHERE location = "West Travon" GROUP BY location

Zapytanie zlicza ilość aren w miejscowości West Travon.

Plan zapytania

select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
SIMPLE	arenas	ALL	*NULL*	*NULL*	*NULL*	*NULL*	1000	Using where

---

Wyświetlenie, zestawienia zawodnik - drużyna

SELECT C.name, C.nickname, C.surname, T.name FROM contestants C LEFT JOIN teams T ON C.team_id=T.team_id

Zapytanie wyświetla imię, pseudonim, nazwisko gracza oraz nazwę drużyny do której on należy.

Wykonanie zapytania trwało 0.0006 sekund(y

Plan zapytania

select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
SIMPLE	C	ALL	*NULL*	*NULL*	*NULL*	*NULL*	1000	
SIMPLE	T	eq_ref	PRIMARY	PRIMARY	4	sdonchor_SWOTLS-DB.C.team_id	1	Using where

---

Wyświetlanie wyników zakończonych meczy

SELECT T1.name, T2.name, M.sideA_score, M.sideB_score, M.time FROM matches M JOIN teams T1 ON M.sideA=T1.team_id JOIN teams T2 ON M.sideB=T2.team_id WHERE M.sideA_score IS NOT NULL

Zapytanie wyświetla nazwy obu drużyn biorących udział w meczu, ich wyniki oraz czas meczu. Mecze które się jeszcze nie odbyły są ignorowane.

Wykonanie zapytania trwało 0.0028 sekundy.

Plan zapytania

select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
SIMPLE	T1	ALL	PRIMARY	*NULL*	*NULL*	*NULL*	500	
SIMPLE	M	ref	sideA,sideB	sideA	4	sdonchor_SWOTLS-DB.T1.team_id	1	Using where
SIMPLE	T2	eq_ref	PRIMARY	PRIMARY	4	sdonchor_SWOTLS-DB.M.sideB	1	

Zabiegi optymalizacyjne

W celu przyśpieszenia działania bazy danych zostały wykonane następujące zabiegi optymalizacyjne. Profile dotyczą prostych zapytań z użyciem indeksów (np. *SELECT * FROM contestants WHERE nickname = "sdonchór"*)

Indeksy unikalne

Nałożono indeksy unikalne na kolumny *TEAMS.name, CONTESTANTS.nickname, ARENAS.name* , ponieważ zakładamy, że nie wystąpią dwie drużyny i areny o takiej samej nazwie, ani zawodnicy o takich samych pseudonimach.

Profil zapytania przed zastosowaniem indeksów

Szczegółowy profil

Kolejność	Stan	Czas
1	Starting	28 µs
2	Checking Permissions	6 µs
3	Opening Tables	19 µs
4	After Opening Tables	5 µs
5	System Lock	4 µs
6	Table Lock	4 µs
7	Init	15 µs
8	Optimizing	7 µs
9	Statistics	19 µs
10	Preparing	9 µs
11	Executing	3 µs
12	Sending Data	671 µs
13	End	9 µs
14	Query End	4 µs
15	Closing Tables	3 µs
16	Unlocking Tables	8 µs
17	Freeing Items	4 µs
18	Updating Status	29 µs
19	Cleaning Up	4 µs

Podsumowanie według stanu

Stan	Całkowity czas	% Czas	Połączenia	ø Czas
Sending Data	671 µs	78.85%	1	671 µs
Updating Status	29 µs	3.41%	1	29 µs
Starting	28 µs	3.29%	1	28 µs
Opening Tables	19 µs	2.23%	1	19 µs
Statistics	19 µs	2.23%	1	19 µs
Init	15 µs	1.76%	1	15 µs
Preparing	9 µs	1.06%	1	9 µs
End	9 µs	1.06%	1	9 µs
Unlocking Tables	8 µs	0.94%	1	8 µs
Optimizing	7 µs	0.82%	1	7 µs
Checking Permissions	6 µs	0.71%	1	6 µs
After Opening Tables	5 µs	0.59%	1	5 µs
System Lock	4 µs	0.47%	1	4 µs
Table Lock	4 µs	0.47%	1	4 µs
Query End	4 µs	0.47%	1	4 µs
Freeing Items	4 µs	0.47%	1	4 µs
Cleaning Up	4 µs	0.47%	1	4 µs
Executing	3 µs	0.35%	1	3 µs
Closing Tables	3 µs	0.35%	1	3 µs

Profil zapytania po zastosowaniu indeksów

Szczegółowy profil

Kolejność	Stan	Czas
1	Starting	40 µs
2	Checking Permissions	10 µs
3	Opening Tables	34 µs
4	After Opening Tables	11 µs
5	System Lock	6 µs
6	Table Lock	21 µs
7	Init	28 µs
8	Optimizing	14 µs
9	Statistics	66 µs
10	Preparing	10 µs
11	Unlocking Tables	13 µs
12	Executing	16 µs
13	Query End	11 µs
14	Closing Tables	5 µs
15	Unlocking Tables	18 µs
16	Freeing Items	9 µs
17	Updating Status	45 µs
18	Cleaning Up	7 µs

Podsumowanie według stanu

Stan	Całkowity czas	% Czas	Połączenia	ø Czas
Statistics	66 µs	18.13%	1	66 µs
Updating Status	45 µs	12.36%	1	45 µs
Starting	40 µs	10.99%	1	40 µs
Opening Tables	34 µs	9.34%	1	34 µs
Init	28 µs	7.69%	1	28 µs
Table Lock	21 µs	5.77%	1	21 µs
Executing	16 µs	4.40%	1	16 µs
Optimizing	14 µs	3.85%	1	14 µs
Unlocking Tables	13 µs	3.57%	1	13 µs
After Opening Tables	11 µs	3.02%	1	11 µs
Query End	11 µs	3.02%	1	11 µs
Checking Permissions	10 µs	2.75%	1	10 µs
Preparing	10 µs	2.75%	1	10 µs
Freeing Items	9 µs	2.47%	1	9 µs
Cleaning Up	7 µs	1.92%	1	7 µs
System Lock	6 µs	1.65%	1	6 µs
Closing Tables	5 µs	1.37%	1	5 µs

Jest widoczna duża poprawa w czasie wykonania zapytania.

Indeksy nieunikalne

Aplikacja będzie często wyszukiwać zawodników na podstawie ich nazwisk. Nazwiska mogą się powtarzać, dlatego też nałożenie na kolumnę **CONTESTANTS.surname** indeksu nieunikalnego jest najoptymalniejsze.

Profil zapytania przed zastosowaniem indeksu

Szczegółowy profil

Kolejność	Stan	Czas
1	Starting	31 µs
2	Checking Permissions	4 µs
3	Opening Tables	25 µs
4	After Opening Tables	5 µs
5	System Lock	4 µs
6	Table Lock	4 µs
7	Init	15 µs
8	Optimizing	7 µs
9	Statistics	8 µs
10	Preparing	9 µs
11	Executing	3 µs
12	Sending Data	729 µs
13	End	8 µs
14	Query End	4 µs
15	Closing Tables	3 µs
16	Unlocking Tables	8 µs
17	Freeing Items	4 µs
18	Updating Status	27 µs
19	Cleaning Up	4 µs

Podsumowanie według stanu

Stan	Całkowity czas	% Czas	Połączenia	ø Czas
Starting	31 µs	3.44%	1	31 µs
Checking Permissions	4 µs	0.44%	1	4 µs
Opening Tables	25 µs	2.77%	1	25 µs
After Opening Tables	5 µs	0.55%	1	5 µs
System Lock	4 µs	0.44%	1	4 µs
Table Lock	4 µs	0.44%	1	4 µs
Init	15 µs	1.66%	1	15 µs
Optimizing	7 µs	0.78%	1	7 µs
Statistics	8 µs	0.89%	1	8 µs
Preparing	9 µs	1.00%	1	9 µs
Executing	3 µs	0.33%	1	3 µs
Sending Data	729 µs	80.82%	1	729 µs
End	8 µs	0.89%	1	8 µs
Query End	4 µs	0.44%	1	4 µs
Closing Tables	3 µs	0.33%	1	3 µs
Unlocking Tables	8 µs	0.89%	1	8 µs
Freeing Items	4 µs	0.44%	1	4 µs
Updating Status	27 µs	2.99%	1	27 µs
Cleaning Up	4 µs	0.44%	1	4 µs

Profil zapytania po zastosowaniu indeksu

Szczegółowy profil

Kolejność	Stan	Czas
1	Starting	29 µs
2	Checking Permissions	5 µs
3	Opening Tables	18 µs
4	After Opening Tables	5 µs
5	System Lock	4 µs
6	Table Lock	11 µs
7	Init	16 µs
8	Optimizing	7 µs
9	Statistics	37 µs
10	Preparing	12 µs
11	Executing	10 µs
12	Query End	6 µs
13	Closing Tables	3 µs
14	Unlocking Tables	6 µs
15	Freeing Items	5 µs
16	Updating Status	31 µs
17	Cleaning Up	4 µs

Podsumowanie według stanu

Stan	Całkowity czas	% Czas	Połączenia	ø Czas
Statistics	37 µs	17.70%	1	37 µs
Updating Status	31 µs	14.83%	1	31 µs
Starting	29 µs	13.88%	1	29 µs
Opening Tables	18 µs	8.61%	1	18 µs
Init	16 µs	7.66%	1	16 µs
Preparing	12 µs	5.74%	1	12 µs
Table Lock	11 µs	5.26%	1	11 µs
Executing	10 µs	4.78%	1	10 µs
Optimizing	7 µs	3.35%	1	7 µs
Query End	6 µs	2.87%	1	6 µs
Unlocking Tables	6 µs	2.87%	1	6 µs
Checking Permissions	5 µs	2.39%	1	5 µs
After Opening Tables	5 µs	2.39%	1	5 µs
Freeing Items	5 µs	2.39%	1	5 µs
System Lock	4 µs	1.91%	1	4 µs
Cleaning Up	4 µs	1.91%	1	4 µs
Closing Tables	3 µs	1.44%	1	3 µs

W porównaniu do indeksu unikalnego, poprawa czasu wykonania zachodzi, ale nie jest tak znaczna.

Minimalna ilość kolumn

Aby zmniejszyć złożoność czasową i pamięciową zapytań zastosowano minimalną ilość kolumn w tabelach. Dane które mogą zostać wywnioskowane z pozostałych są wyprowadzane w aplikacji lub w zapytaniu SQL. Przykładem jest tutaj tabela **MATCHES.** Aby dowiedzieć się, czy mecz już się odbył nie jest potrzebne dodatkowe pole typu **BOOLEAN**. Wystarczy sprawdzić, czy zachodzi warunek **NOW()>MATCHES.time** oraz **MATCHES.sideA_score IS NOT NULL.** Jeśli tak - mecz już się odbył, jeśli nie - nie odbył się, lub został unieważniony (w przypadku gdy **NOW()>MATCHES.time AND MATCHES.sideA_SCORE IS NULL**).

Podział zadań

Jakub Pranica

- GUI
- logika systemów turniejowych
- system rankingowy

Sebastian Donchór

- komunikacja między modułami (klient, serwer, baza danych)
- baza danych i jej obsługa
- weryfikowanie zapytań klienta i logowania
- przetwarzanie danych

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. Implementacja podstawowych systemów turniejowych

VIII. Implementacja rankingu Elo

IX. Implementacja systemów korzystających z Elo

X. System ligowy

XI. Dokumentacja użytkowa

Odwołania do literatury

- **Kodeks Szachowy** cz. B.III oraz B.IV (wydanie 2002)
- **W.Litmanowicz, J.Giżycki, "Szachy od A do Z"**, tom II, Warszawa 1987, str. 1066 (rundowy system) oraz 1211 (tabela kojarzeń)
- **Tadeusz Czarnecki, "ABC Szachisty"**. Warszawa: Sport i Turystyka, 1966.
- https://pl.wikipedia.org/wiki/System_pucharowy
- https://pl.wikipedia.org/wiki/System_szwajcarski
- https://pl.wikipedia.org/wiki/System_ko%C5%82owy
- https://pl.wikipedia.org/wiki/Ranking_szachowy
- https://pl.wikipedia.org/wiki/System_McMahona