Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

System rozgrywek turniejowych

Uniwersalna platforma do tworzenia i śledzenia różnego rodzaju turniejów sportowych i planszowych

Implementacja i testy (5-IiT)

Zespół projektowy:

Piotr Maj – Menadżer projektu/Lider techniczny

Bartosz Jędryka – Programista

Aleksandra Kuś – Tester

Wojciech Pędziwiatr – Programista

Adrian Suchenia – Programista

Kontakt (Menadżer projektu): pm311399@student.polsl.pl

Katowice, 27.0.1.2026

1. Wprowadzenie

1.1. Cel projektu

Celem projektu jest zaprojektowanie i implementacja uniwersalnej platformy do organizacji i zarządzania turniejami sportowych i planszowych. System umożliwia tworzenie turniejów dla różnych dyscyplin (m.in. e-sport, szachy, piłka nożna), automatyczne generowanie drabinek, rejestrację uczestników oraz wprowadzanie i prezentację wyników.

1.2. Interesariusze

- **Organizatorzy turniejów** — tworzą turnieje, zarządzają konfiguracją, dodają moderatorów.
- **Moderatorzy/Sędziowie** — wprowadzają i zatwierdzają wyniki.
- **Uczestnicy (gracze / drużyny)** — biorą udział i przeglądają swoje mecze.
- **Widzowie/publiczność** — przeglądają publiczne turnieje i wyniki.

1.3. Korzyści wdrożenia

- Automatyzacja procesu tworzenia drabinek i parowań.
- Centralizacja wyników i historii rozgrywek.
- Wsparcie dla wielu systemów rozgrywek (Eliminacje, Kołowy, Szwajcarski).
- Możliwość dostosowania punktacji (szablony gier).

2. Analiza dziedziny

2.1. Przegląd istniejących rozwiązań

W analizie porównawczej uwzględniono popularne rozwiązania rynkowe, ich kluczowe cechy, zalety i ograniczenia.

1) Challonge:

   - **Zalety:** prosty interfejs do tworzenia turniejów, wiele typów bracketów, API integracyjne.
   - **Wady:** ograniczone możliwości personalizacji zaawansowanych reguł punktacji i obsługi gier wieloosobowych, płatne funkcje zaawansowane.
   - **Pozycjonowanie:** platforma dla dużych, kompleksowych zawodów e-sportowych.

2) Toornament
   - **Zalety:** rozbudowane wsparcie dla e-sportu, zaawansowane opcje turniejowe i integracje.
   - **Wady:** stosunkowo złożony interfejs, konfiguracja wymaga czasu.
   - **Pozycjonowanie:** platforma dla dużych, kompleksowych zawodów e-sportowych.

3) Battlefy
   - **Zalety:** duże możliwości konfiguracji, target e-sport, zarządzanie drużynami.
   - **Wady:** ograniczona elastyczność dla nietypowych gier planszowych.
   - **Pozycjonowanie:** ukierunkowany na ligowe rozgrywki e-sportowe.

4) Tournament Software (rozwiązanie sportowe)
   - **Zalety:** dobre raporty, dopasowane do potrzeb sportów tradycyjnych.
   - **Wady:** często desktopowe lub zamknięte, mniej przyjazne dla integracji webowych.
   - **Pozycjonowanie:** organizacje sportowe i stowarzyszenia.

3. Opis i wizja systemu

3.1. Aktorzy systemu (role)

- Organizator: Tworzy i konfiguruje turniej, zarządza moderatorami i rejestracją.
- Moderator/Sędzia: Wprowadza wyniki meczów, edytuje błędne wpisy.
- Uczestnik: Rejestruje się do turnieju, przegląda swoje mecze.
- Gość: Przegląda publiczne turnieje i wyniki.

4. Wymagania systemowe

4.1. Wymagania funkcjonalne (RF)

ID	Nazwa	Opis
RF-01	Rejestracja i logowanie użytkownika	Rejestracja użytkownika (email/hasło) oraz logowanie.
RF-02	Tworzenie turnieju	Zalogowany użytkownik może utworzyć turniej i zostać jego organizatorem.
RF-03	Szablony gier i punktacja	Wybór szablonu gry (np. Szachy, Piłka nożna) oraz konfiguracja punktacji (zwycięstwo/remis/przegrana).
RF-04	Rejestracja do turnieju	Obsługa trybów: Otwarta i Wymagająca Akceptacji. Limity uczestników.
RF-05	Przegląd turniejów	Lista nadchodzących turniejów z możliwością zapisu.

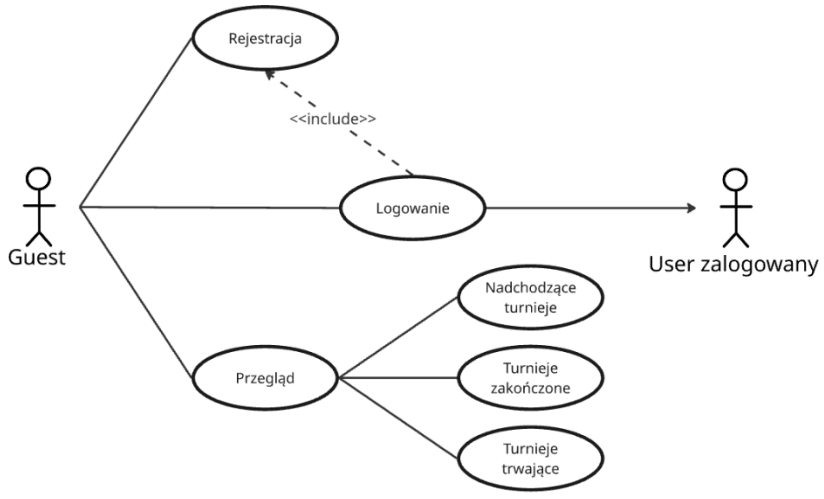
RF-06	Moderatorzy	Organizator może dodawać moderatorów do wprowadzania wyników.
RF-07	Systemy rozgrywek	Generowanie drabinek: Single/Double Elimination, Round Robin, Swiss.
RF-08	Wprowadzanie wyników	Moderator wprowadza wyniki meczów.
RF-09	Historia i Archiwum	Przegląd zakończonych turniejów i wyników.

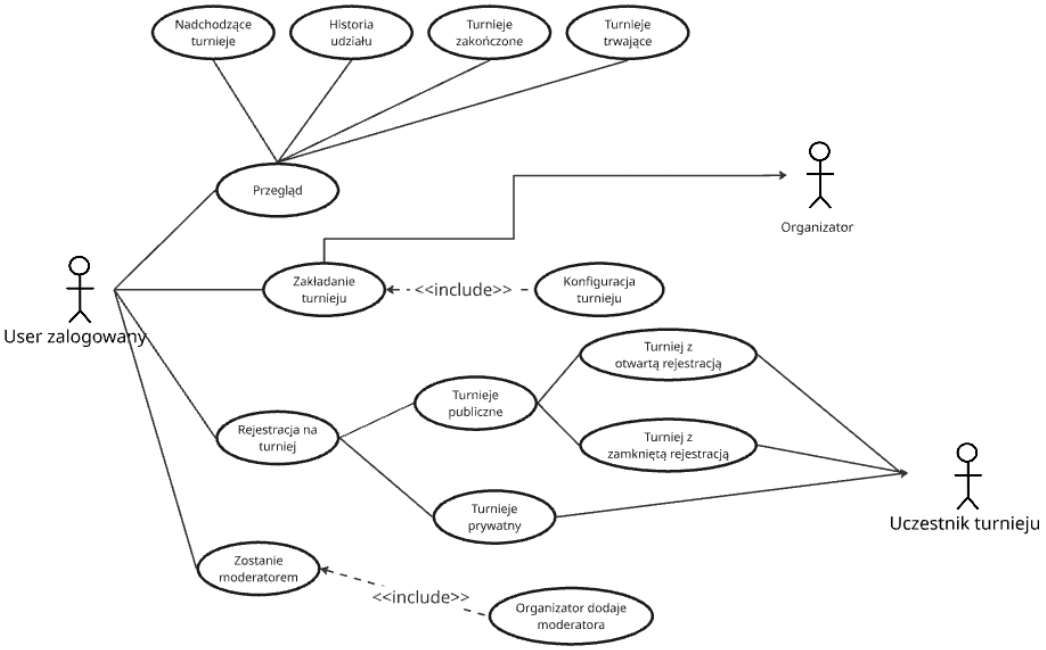
4.2. Wymagania niefunkcjonalne

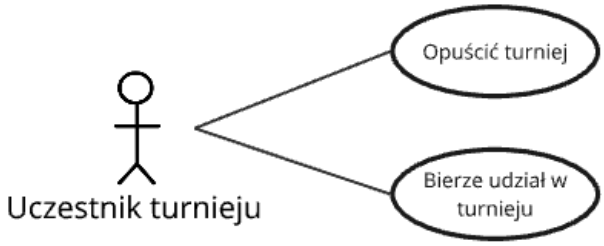
ID	Nazwa	Opis
RNF-01	Responsywność	Interfejs dostosowany do urządzeń mobilnych i desktopowych
RNF-02	Bezpieczeństwo	Uwierzytelnianie (JWT) i autoryzacja zasobów.
RNF-03	Użyteczność	Prosty formularz tworzenia turnieju

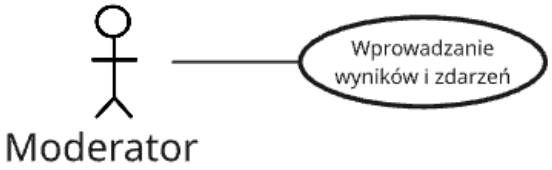
5. Modelowanie przypadków użycia (2-MPU)

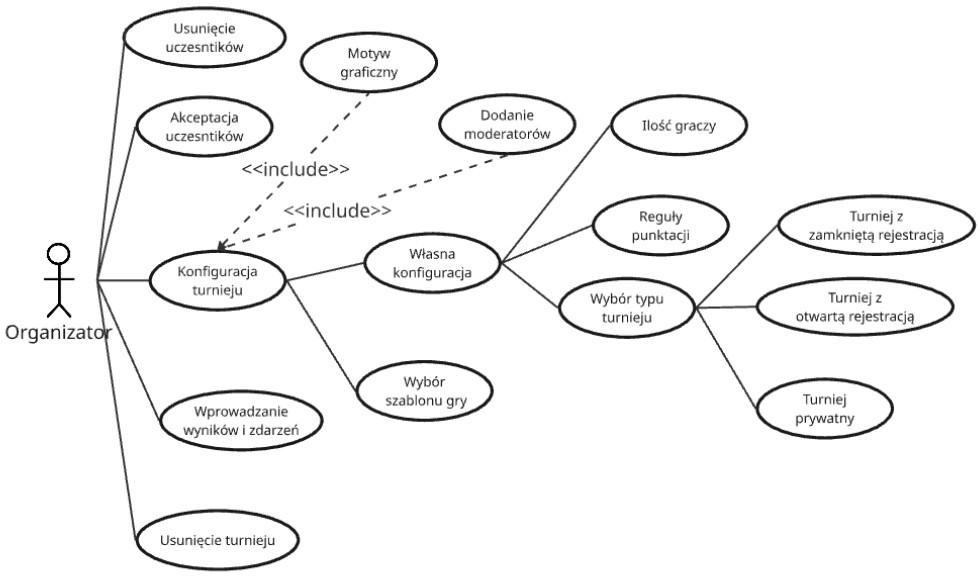
Zaprezentowanie diagramu graficznego reprezentującego funkcjonalność systemu oraz interakcje aktorów przy użyciu notacji UML za pomocą diagramu przypadków użycia.




Nadchodzące turnieje
Historia udziału
Turnieje zakończone
Turnieje trwające
Przegląd
Organizator
Zakładanie turnieju
<<include>>
Konfiguracja turnieju
User zalogowany
Turniej z otwartą rejestracją
Turnieje publiczne
Rejestracja na turniej
Turniej z zamkniętą rejestracją
Turnieje prywatny
Uczestnik turnieju
Zostanie moderatorem
<<include>>
Organizator dodaje moderatora


Opuścić turniej
Uczestnik turnieju
Bierze udział w turnieju


Wprowadzanie wyników i zdarzeń
Moderator



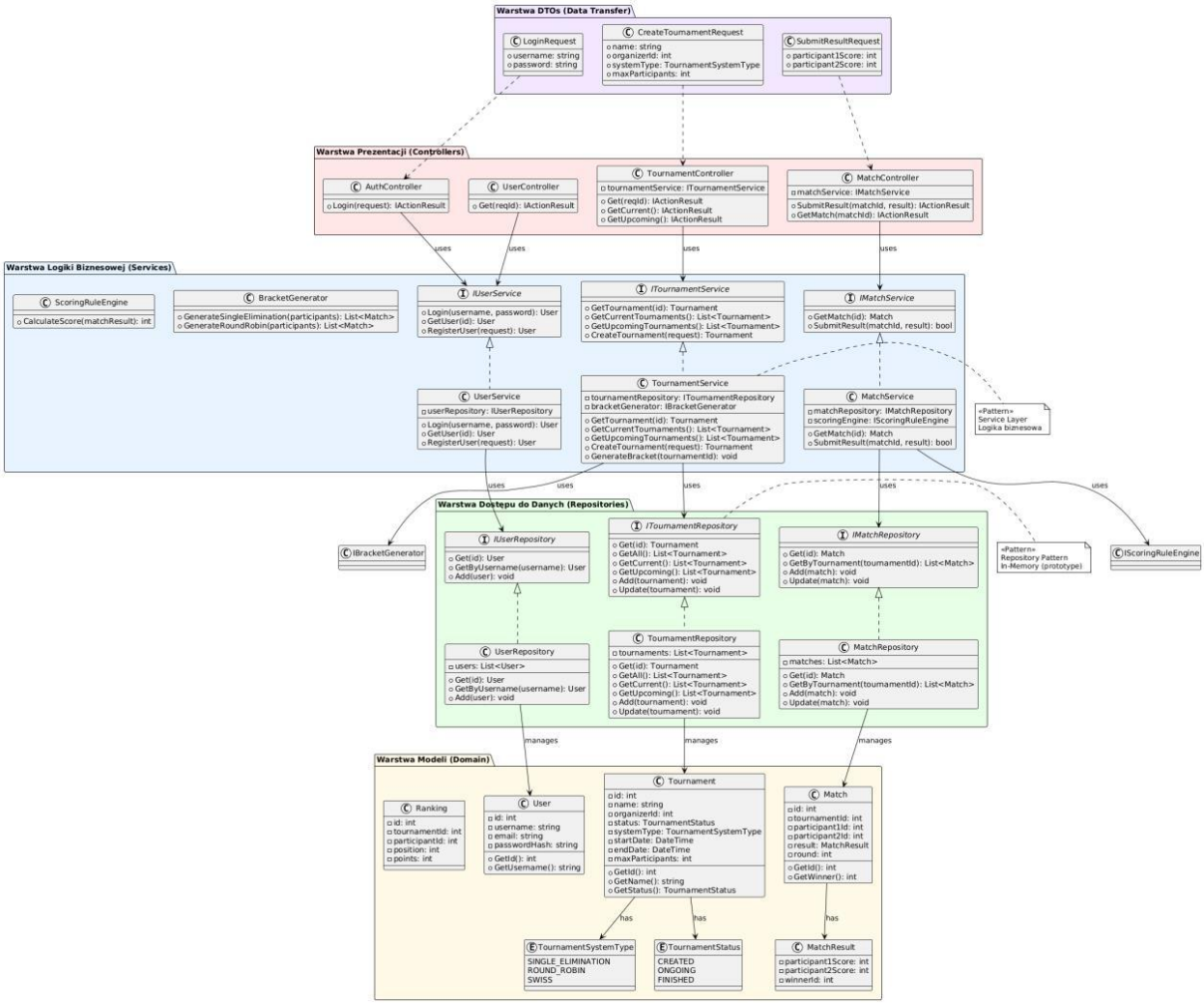
6. Modelowanie struktury (3-MS-P1)

6.1. Schemat UML

Diagram klas przedstawia architekturę systemu G-Flow w modelu warstwowym (3-tier architecture). Struktura obejmuje cztery główne warstwy:

- **Warstwa Prezentacji (Controllers)** – zawiera kontrolery obsługujące żądania HTTP: AuthController, UserController, TournamentController i MatchController. Każdy kontroler komunikuje się z odpowiednim serwisem.
- **Warstwa Logiki Biznesowej (Services)** – implementuje interfejsy serwisów (ITournamentService, IUserService, IMatchService) odpowiadające za logikę biznesową aplikacji. Serwisy zawierają komponenty pomocnicze takie jak BracketGenerator (generowanie parowań) i ScoringRuleEngine (obliczanie punktów).
- **Warstwa Dostępu do Danych (Repositories)** – abstrakcja dostępu do bazy danych poprzez interfejsy repozytoriów (ITournamentRepository, IUserRepository, IMatchRepository). Warstwa umożliwia wymianę implementacji bez zmian w wyższych warstwach.
- **Warstwa Modeli (Domain Models)** – definiuje obiekty domenowe: User, Tournament, Match, MatchResult, Ranking oraz enumeracje statusów (TournamentStatus, TournamentSystemType).

Diagram ilustruje zastosowanie wzorców: **Repository Pattern** (abstrakcja danych), **Service Layer Pattern** (logika biznesowa), **Strategy Pattern** (różne algorytmy generowania drabinek).



7. Modelowanie zachowania (4-MZ-P2)- Diagramy sekwencji

Opis Przypadku Użycia: Logowanie użytkownika

Cel: Autoryzacja użytkownika w systemie i nadanie uprawnień do korzystania z platformy poprzez wygenerowanie tokena JWT.

Przebieg procesu:

- **Inicjacja:** Użytkownik wprowadza login oraz hasło w formularzu przeglądarki (HTML/JS).
- **Żądanie API:** Przeglądarka przesyła dane za pomocą metody POST /api/login do bramki API, która deleguje zadanie do UserService.
- **Dostęp do danych:** UserService komunikuje się z UserRepo, aby pobrać dane użytkownika z bazy danych za pomocą zapytania SQL SELECT password FROM users....
- **Weryfikacja:** Baza danych zwraca zahaszowane hasło, a UserService wykonuje wewnętrzną metodę verifyPassword(), aby porównać dane wprowadzone przez użytkownika z tymi zapisanymi w systemie.

Scenariusze wynikowe (Blok ALT):

1) Logowanie pomyślne (Poprawne dane):
   - System generuje token autoryzacyjny JWT.
   - API zwraca status 200 OK [token].
   - Przeglądarka wykonuje operację zapisu tokenu (np. w Local Storage), co umożliwia dalszą pracę w systemie.
2) Błąd autoryzacji (Błędne dane):
   - UserService zwraca informację o niepowodzeniu (auth failed).
   - API przesyła do użytkownika komunikat o błędzie ze statusem 401 Unauthorized.

Opis Przypadku Użycia: Tworzenie nowego turnieju

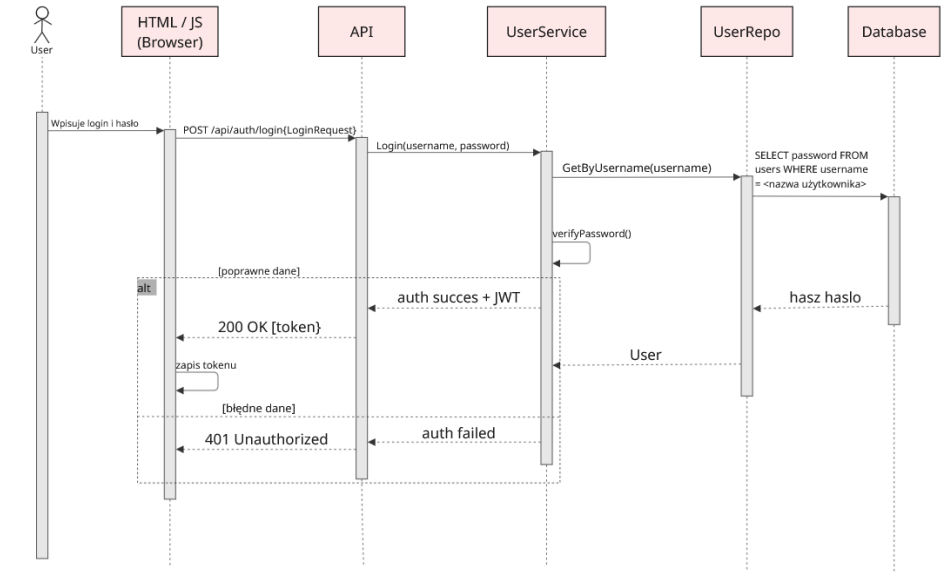
Cel: Umożliwienie zalogowanemu organizatorowi utworzenia nowej instancji turnieju w systemie wraz z zapisem jego konfiguracji w bazie danych.

Opis przebiegu (scenariusz główny):

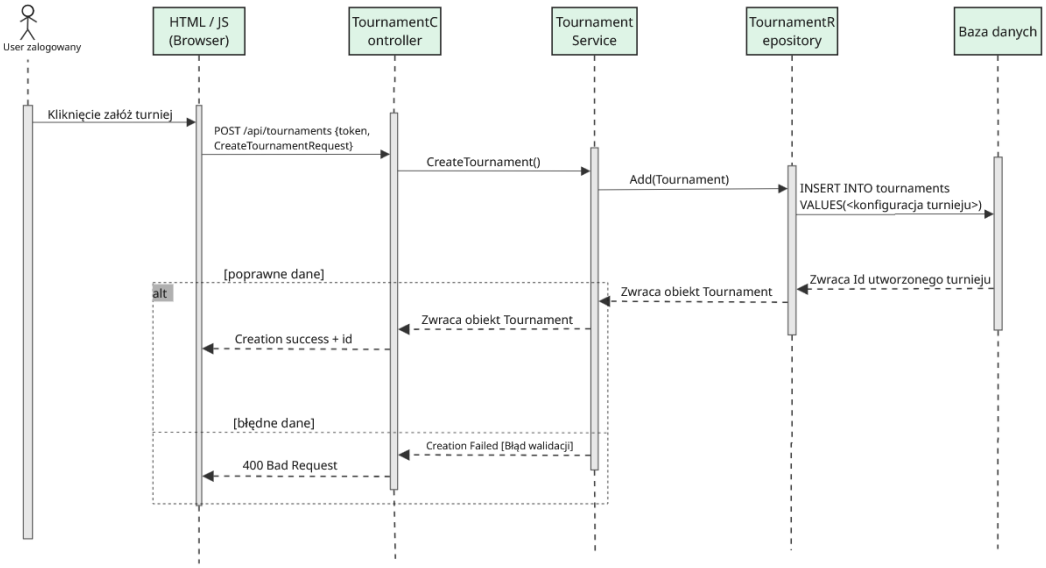
1) **Inicjacja:** Zalogowany użytkownik wypełnia formularz i klika przycisk "Załóż turniej" w interfejsie przeglądarki (HTML/JS).
2) **Przesłanie danych:** Warstwa front-endowa wysyła żądanie POST /api/tournaments, przekazując w nagłówku token autoryzacyjny oraz obiekt CreateTournamentRequest w formacie JSON do Tournament Controller.
3) **Logika biznesowa:** Kontroler deleguje zadanie do warstwy logiki (Tournament Service) poprzez metodę CreateTournament(). Serwis odpowiada za walidację danych wejściowych (np. sprawdzenie poprawności dat i nazw).
4) **Persystencja:** Po pomyślnej walidacji, serwis wywołuje metodę Add(Tournament) w warstwie dostępu do danych (Tournament Repository).
5) **Zapis w bazie:** Repozytorium wykonuje zapytanie SQL INSERT INTO tournaments do Bazy danych. Baza zwraca unikalny identyfikator (ID) nowo utworzonego rekordu.
6) **Potwierdzenie:** Informacja o sukcesie i nowym obiekcie jest przekazywana z powrotem przez wszystkie warstwy. Kontroler zwraca do interfejsu odpowiedź 201 Created (na diagramie jako Creation success) wraz z nadanym ID.

Scenariusz alternatywny (Błąd walidacji):

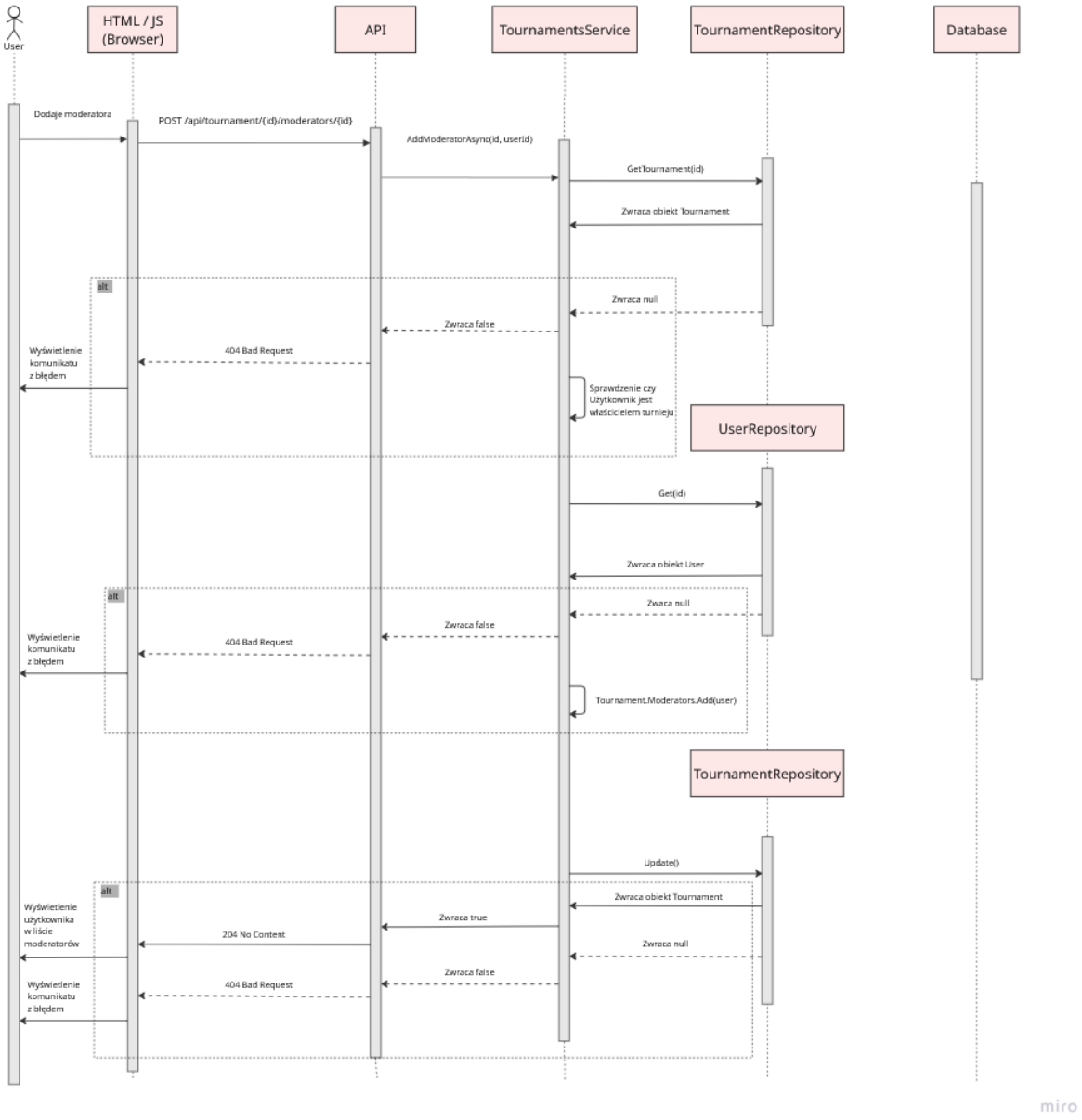
- W przypadku, gdy dane w Tournament Service nie przejdą walidacji, proces zapisu w bazie zostaje przerwany.
- System zwraca informację o błędzie (Creation Failed) do kontrolera, który wysyła do użytkownika odpowiedź 400 Bad Request.


User
HTML / JS (Browser)
API
UserService
UserRepo
Database
Wpisuje login i hasło
POST /api/auth/login{LoginRequest}
Login(username, password)
GetByUsername(username)
SELECT password FROM users WHERE username = <nazwa użytkownika>
verifyPassword()
[poprawne dane]
alt
auth succes + JWT
hasz haslo
200 OK [token}
User
zapis tokenu
[błędne dane]
401 Unauthorized
auth failed

miro


User zalogowany
HTML / JS (Browser)
TournamentController
Tournament Service
TournamentRepository
Baza danych
Kliknięcie załóż turniej
POST /api/tournaments {token, CreateTournamentRequest}
CreateTournament()
Add(Tournament)
INSERT INTO tournaments VALUES(<konfiguracja turnieju>)
[poprawne dane]
alt
Zwraca Id utworzonego turnieju
Zwraca obiekt Tournament
Zwraca obiekt Tournament
Creation success + id
[błędne dane]
Creation Failed [Błąd walidacji]
400 Bad Request

miro



8. Strategia testowania

W projekcie przyjęto hybrydowe podejście do zapewnienia jakości, łączące automatyczne testy jednostkowe warstwy biznesowej z manualnymi testami interfejsu użytkownika.

Zastosowane rodzaje testów:

1. **Testy Jednostkowe (Backend)**:
    - Wykorzystano framework **xUnit** do testowania logiki domenowej i usług aplikacji.

- Kluczowe obszary objęte testami:
    - Algorytmy generowania drabinek (Single/Double Elimination, Round Robin, Swiss, Tie-breakery).
    - Logika przyznawania punktów i walidacja reguł biznesowych (np. limity uczestników).
    - Serwisy aplikacyjne (TournamentService, MatchEventService, AuthService).

2. **Testy Manualne (Frontend)**:
    - Weryfikacja responsywności (RWD) na urządzeniach mobilnych i desktopowych.
    - Testy scenariuszowe "End-to-End" wykonywane ręcznie, sprawdzające integrację między frontendem a backendem (np. przepływ tworzenia turnieju).
    - Walidacja formularzy i poprawności wyświetlania złożonych danych (drabinki, tabele).

Przykładowe scenariusze testowe:

- **SC-01: Tworzenie Turnieju**:
    1. Użytkownik loguje się do systemu.
    2. Wybiera opcję "Utwórz turniej".
    3. Wypełnia formularz (Nazwa, Gra: Szachy, System: Swiss).
    4. Zatwierdza formularz.
    5. **Oczekiwany rezultat**: Użytkownik przekierowany do panelu turnieju, turniej widoczny na liście.
- **SC-02: Generowanie Drabinki (Single Elimination)**:
    1. Organizator dodaje 4 uczestników.
    2. Klika "Rozpocznij turniej".
    3. **Oczekiwany rezultat**: System generuje drabinkę z 2 półfinałami.

- **SC-03: Wprowadzanie Wyniku**:
    1. Moderator wchodzi w szczegóły meczu.
    2. Wprowadza wynik 1:0.
    3. Zatwierdza.
    4. **Oczekiwany rezultat**: Zwycięzca awansuje w drabince, status meczu "Zakończony".

9. Wnioski końcowe

Podsumowanie projektu Udało się zrealizować kluczowe założenia systemu, dostarczając funkcjonalną platformę do zarządzania turniejami (e-sport i planszowe). Aplikacja automatyzuje proces parowania i centralizuje wyniki, a interfejs został dostosowany do wymogów responsywności.

Wyzwania techniczne i projektowe:

- **Złożoność algorytmów**: Implementacja systemów Swiss i Double Elimination wymagała precyzyjnej logiki parowań i obsługi "wolnych losów" (byes).
- **UX na mobile**: Prezentacja szerokich tabel i drabinek na małych ekranach wymagała kompromisów i wdrożenia widoków wertykalnych.
- **Spójność danych**: Synchronizacja stanu między edycją (Organizator) a podglądem (Uczestnik) wymagała starannego projektowania API.

Zdobyte doświadczenia:

- Zrozumienie specyfiki różnych formatów rozgrywek (od prostych pucharowych po złożone systemy szwajcarskie).
- Praktyczne zastosowanie wzorców projektowych w C# do obsługi wymiennej logiki turniejowej.

Kierunki dalszego rozwoju:

- **Turnieje prywatne**: Pełne wdrożenie mechanizmów kontroli dostępu.
- **Eksport danych**: Generowanie raportów PDF/CSV dla organizatorów.
- **WebSockety**: Wprowadzenie komunikacji w czasie rzeczywistym dla wyników live.
- **Personalizacja**: Rozbudowa systemu motywów o customowe grafiki użytkowników.

10. Słownik pojęć

- **Audyt (Audit)**: Zapis historyczny operacji krytycznych w systemie, w szczególności zmian wyników meczów wraz z identyfikacją użytkownika i czasem zmiany.
- **Bye (Wolny los)**: Sytuacja w turnieju, w której uczestnik automatycznie przechodzi do kolejnej rundy (lub otrzymuje punkty) z powodu braku przeciwnika.
- **Drabinka (Bracket)**: Graficzna reprezentacja parowań i awansów uczestników w systemie eliminacyjnym.
- **Game Template (Szablon gry)**: Predefiniowany zestaw parametrów dla konkretnej gry (liczba graczy w meczu, typ punktacji, ikona), wykorzystywany przy tworzeniu turniejów.
- **Mecz (Match)**: Pojedyncze spotkanie między uczestnikami w ramach turnieju. Posiada status (np. zaplanowany, w toku, zakończony), wynik i opcjonalnie zdarzenia (eventy).
- **Moderator / Sędzia**: Użytkownik z uprawnieniami do wprowadzania i edycji wyników w przypisanym turnieju.
- **Organizator (Organizer)**: Użytkownik tworzący turniej, definiujący jego parametry oraz zarządzający dostępem i moderatorami.
- **Private tournament (Turniej prywatny)**: Turniej widoczny wyłącznie dla uczestników i przypisanych moderatorów; dostęp publiczny jest zablokowany.
- **Ranking / Standing**: Uporządkowana lista uczestników na podstawie zgromadzonych punktów i tie-breakerów.
- **Round Robin (System kołowy)**: System rozgrywek, w którym każdy uczestnik gra z każdym innym przynajmniej raz (każdy z każdym).
- **Scoring Rule (Reguła punktacji)**: Zestaw zasad określających, ile punktów przypada za zwycięstwo, remis lub przegraną (np. 3/1/0 w piłce nożnej).
- **Swiss System (System szwajcarski)**: System rozgrywek, w którym gracze są dobierani w pary w kolejnych rundach na podstawie zbliżonego bilansu punktowego.
- **System rozgrywek (Tournament System)**: Algorytm określający zasady parowań i awansów (np. Single Elimination, Double Elimination, Swiss, Round Robin).
- **Tie-breaker**: Dodatkowe kryterium decydujące o kolejności w rankingu w przypadku równej liczby punktów (np. wynik bezpośredniego meczu, Buchholz).
- **Uczestnik (Participant)**: Osoba lub drużyna biorąca udział w turnieju.
- **Użytkownik (User)**: Każda osoba posiadająca konto w systemie. Może pełnić różne role (organizator, moderator, gracz) w zależności od kontekstu.