SureTime

Dokumentacja

1. Ogólne informacje

Opis projektu

W branży lekkiej atletyki do zarządzania zawodami używa się praktycznie jednego systemu. Firma która go stworzyła aktualnie posiada monopol w tej dziedzinie. Motywacją do stworzenia takiego projektu było stworzenie systemu, który w przyszłości mógłby stworzyć konkurencję wyżej wymienionej firmie.

SureTime to wielozadaniowy system do organizowania, zarządzania zawodami sportowymi jak i klubami sportowymi. Można podzielić go na dwie główne części, zarządzanie zawodami oraz zarządzanie klubami / zawodnikami.

Zarządzanie klubami oraz zawodnikami polega na stworzeniu możliwości tworzenia kont zawodniczych, przypisywaniu danych takich jak adres, trener. Stworzenie klubów pozwala na przypisanie zawodników, dodanie trenerów.

Zarządzanie zawodami polega na możliwości stworzenia zawodów, dodanie do nich typów konkurencji (np. bieg na 800 metrów czy rzut młotem), przypisanie zawodników do danej konkurencji. Istnieje też możliwość podzielić konkurencję na kilka serii (Np w wypadku gdy w jednej konkurencji wystartować chce więcej niż jest to możliwe).

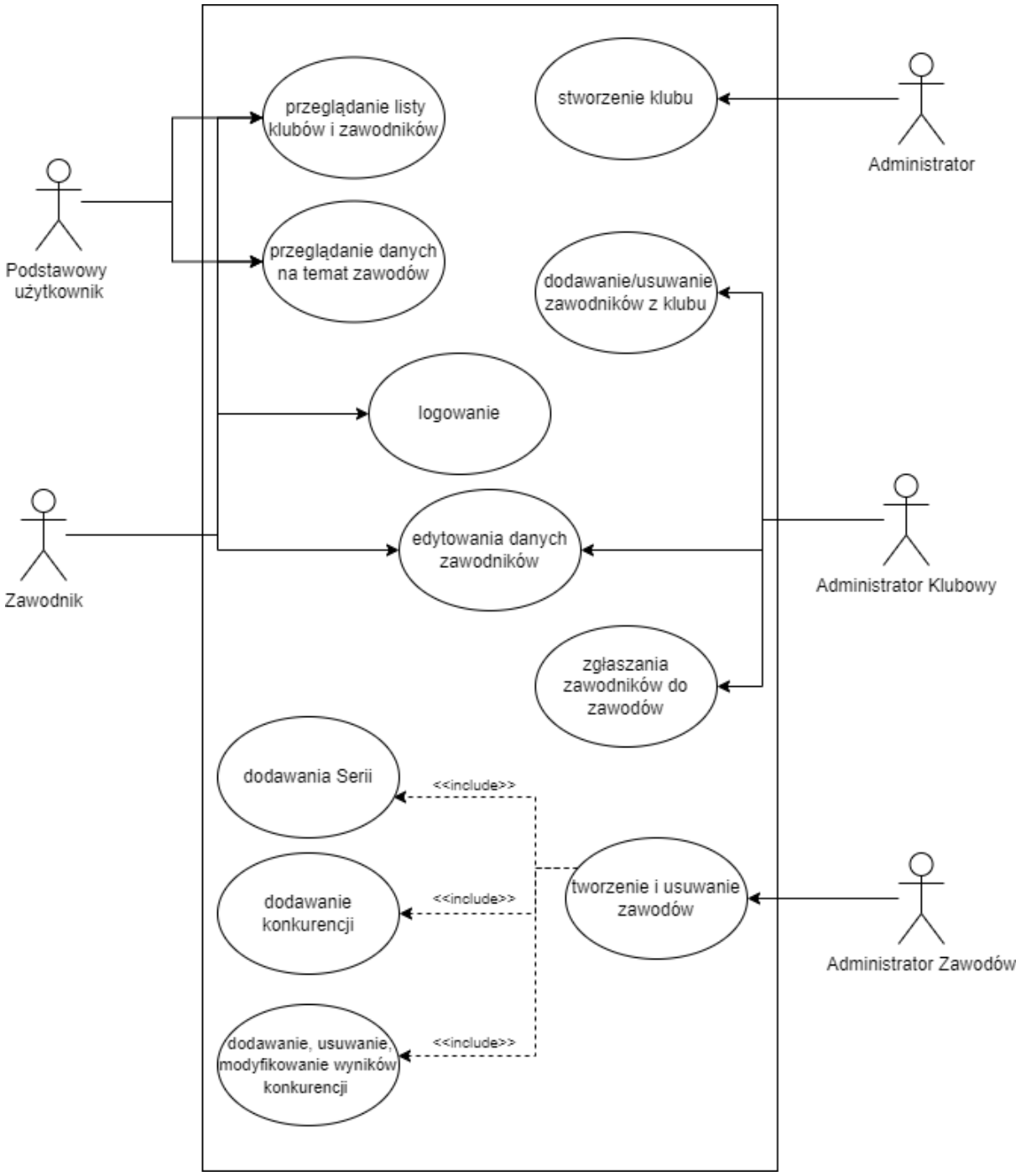
Członkowie zespołu

- Szymon Sala - Lider, BackEnd
- Przemek Sałek - FrontEnd
- Aleksander Pitucha - BackEnd
- Szymon Żywko - BackEnd
- Dorota Zaremba - Dokumentacja

Jak uruchomić

Nasz projekt składa się z 5 repozytoriów git, które zawierają się w projekcie pod https://github.com/SureTime-UMCS. Pod każdym projektem jest opis jak daną aplikację (kod) uruchomić. **Ważne**: **Wszystkie pobrane repozytoria muszą znajdować się w tym samym katalogu.** Wynika to z faktu, że plik konfiguracyjny docker-compose.yaml wymaga repozytorium suretime-fe.

Diagramy/Przypadki użycia



2. Backend

Wykorzystane technologie

- swagger
- glory of rest (hateoas)
- gateway
- JWT
- caffeine cache
- mongoDB
- eureka

Dobre praktyki

1. Wykorzystanie uuid - zasoby są rozpoznawane po uuid dzięki czemu nie ma możliwości odwołania się do zasobu nie posiadając id. Umożliwiają swobodną edycję zasobu, który nie jest identyfikowany po żadnej z jego wartości.
2. Zastosowanie interfejsów i dziedziczenia w przypadku serwisów - umożliwia większą swobodę w przyszłych modyfikacjach kodu i nie zamyka aplikacja na rozwój
3. Podział struktury projektu na *domain* i *application* umożliwiający łatwiejsze nawigowanie po plikach i utrzymanie przejrzystości.
4. Wersjonowanie api za pomocą oznaczeń w url endpointu. Np. /api/v1/events

User Roles

- Admin [“admin”]
    - Tworzenie klubu.

- Administrator zawodów [“competition_admin”]
    - Zarządzanie zawodami
        - Stworzenie/Edycja/Usunięcie zawodów
        - Dodawanie konkurencji
        - Dodanie serii
        - Dodawanie/Edycja/Usunięcie wyników

- Administrator klubowy [“club_admin”] (wymagane zalogowanie)
    - Stworzenie klubu
    - Zarządzanie profilami zawodników
        - Stworzenie/Edycja/Usunięcie profilu zawodników
    - Zgłaszanie zawodników do zawodów

- Zawodnik [“contestant”/”player”]
    - Wyświetlanie swojego profilu zawodnika

    - Wyświetlanie porównania z innymi zawodnikami
    - Wyświetlanie historii uczestnictwa w zawodach
    - Zarządzanie profilem
        - Edycja danych

- Podstawowy użytkownik [“basic_user”] (może być anonymous - bez logowania)
    - Wyświetlanie zawodów
        - kto uczestniczył
        - wyniki konkretnych konkurencji
        - ogólne wyniki zawodów
        - konkurencje
    - Wyświetlenie profilu zawodnika
    - Wyświetlenie listy klubów
    - Wyświetlenie konkretnego klubu
        - wyniki
        - zawodnicy

Endpoint

- **Auth-controller**
    - POST/api/auth/login
    - POST/api/auth/register

- **Club-controller**
    - GET/api/v1/clubs
    - POST/api/v1/clubs
    - POST/api/v1/clubs/{clubName}/moderators
    - GET/api/v1/clubs/{id}
    - GET/api/v1/clubs/{name}
    - PUT/api/v1/clubs/{name}
    - DELETE/api/v1/clubs/{name}
    - POST/api/v1/clubs/{name}/person

- **Competition-controller**
    - GET /api/v1/competitions
    - GET /api/v1/competitions/{id}
    - PUT /api/v1/competitions/{id}
    - DELETE /api/v1/competitions/{id}
    - POST /api/v1/competitions/{id}/competitor/{email}
    - DELETE /api/v1/competitions/{id}/competitor/{email}
    - POST /api/v1/competitions/{id}/event/{event_id}
    - DELETE /api/v1/competitions/{id}/event/{event_id}

- **Event-controller**
  - GET /api/v1/events
  - GET /api/v1/events/{id}
  - PUT /api/v1/events/{id}
  - DELETE /api/v1/events/{id}
  - POST /api/v1/events/{id}/competitor/{email}
  - DELETE /api/v1/events/{id}/competitor/{email}
  - POST /api/v1/events/{id}/heat/{heat_id}
  - DELETE /api/v1/events/{id}/heat/{heat_id}

- **Heat-controller**
  - GET /api/v1/heats
  - GET /api/v1/heats/{id}
  - PUT /api/v1/heats/{id}
  - DELETE /api/v1/heats/{id}
  - POST /api/v1/heats/{id}/competitors
  - DELETE /api/v1/heats/{id}/competitors
  - POST /api/v1/heats/{id}/results
  - DELETE /api/v1/heats/{id}/results

- **Person-controller**
  - GET /api/v1/persons
  - POST /api/v1/persons
  - PUT /api/v1/persons
  - GET /api/v1/persons/{uuid}
  - DELETE /api/v1/persons/{uuid}
  - GET /api/v1/persons/me
  - PUT /api/v1/persons/roles/{uuid}

3. Frontend

Frontend został wykonany w React. Do ułatwienia stylizowania strony wykorzystaliśmy Tailwind. Frontend wykorzystuje tylko część endpointów:

- Rejestrację
- Logowanie (wraz z przypisaniem jwt token do ciasteczka)
- Przegląd użytkowników
- Przegląd danych poszczególnego użytkownika

4. Baza Danych

Models overview

- **Address** - Reprezentacja adresu.
- **Club** - Za zadanie ma reprezentować klub w którym są zrzeszone osoby.
- **Competition** - Reprezentacja wydarzenia sportowego.
- **Event** - Reprezentacja poszczególnej konkurencji np. bieg na 1500 metrów.
- **Person** - Reprezentacja osoby.
- **Heat (**Heat - seria) - Model ten ma za zadanie reprezentować jedną serię w konkurencji np. bieg na 100 metrów zgłoszonych jest 15 osób. Bieżnia (zazwyczaj) ma osiem torów. Dlatego trzeba tą konkurencje podzielić na co najmniej dwie serie.

5. Dalszy rozwój

1. Konfiguracja polityki CORS, powinna być możliwość strzelania na aplikację BE tylko przez gateway.
2. Postawienie środowiska (UAT)
3. Poprawka dockeryzacji aplikacji FE.
4. Rozbicie aplikacji na mikroserwisy.