Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Kierunek: Teleinformatyka

PROJEKT Z APLIKACJI MOBILNYCH
NA TEMAT
“SYSTEM STATYSTYK SPORTOWYCH DLA SIATKÓWKI”
DOKUMENTACJA

Adam Sołtysiak 259722

Artur Chrapek 259710

Adam Gołecki 259696

Nataliia Belinska 257748

Jan Świergoń 250167

Kurs: Aplikacje mobilne – projekt

Grupa: K01-20e (wtorek 13:15 TP)

Wrocław, 2023

Spis treści

1 Zaprezentowanie rynkowego opisu projektu wraz z planem komercjalizacji

Ten rozdział zawiera słowny opis, czego dany projekt dotyczy, a także – porównanie proponowanych rozwiązań z już istniejącymi na rynku.

1.1 Rynkowy opis projektu

REST API pozwalające na zapytanie o szeroką bazę statystyk z polskich lig siatkowych (wyniki drużyn, turnieje itp.). Dane generowane losowo w celach prezentacji działania aplikacji z wykorzystaniem gem'a "Faker".

1.2 Plan komercjalizacji

Sprzedaż tokenów dostępu – czasowych, bądź ilościowych (po zapytaniach).

W przypadku tokenów czasowych to np. sprzedaż tokenu na X dni (z ograniczeniem X zapytań dziennie).

Możliwość zapłaty zewnętrznym deweloperom pomagającym zbierać dane statystyczne na żywo z meczów.

1.3 Czego dotyczy projekt

Stworzenia REST API pozwalającego na zwracanie statystyk siatkowych takich jak:

- statystyki wygranych/przegranych meczy,
- wyniki setów,
- ilość rozegranych setów,
- turnieje, sezony, rundy w turnieju,
- drużyny na turniejach,
- miejsce drużyny w tabeli,
- kompletne statystyki meczów drużyny.

1.4 Rozwiązania już istniejące na rynku

Szeroko rozwinięta branża – duża ilość podobnych aplikacji, każda oferująca podobne statystyki:

- historyczne wyniki i statystyki meczów,
- punktacja meczów na żywo,
- statystyki H2H poszczególnych drużyn.

Istnieją również API pozwalające zpushować betowania oraz wyniki zakładów.

Ze stron internetowych – również wiele podobnych oferujących wręcz identyczne statystyki:

- Volleyball Data API Documentation (BroadageSports) | RapidAPI

- API Explorer (geniussports.com)
- Volleyball (Indoor) v2 | Sportradar US API Portal

Ceny istniejących rozwiązań dostępnych na rynku:

Objects	Basic $0.00 / mo Subscribe	Pro $10.00 / mo Subscribe	Recommended Ultra $15.00 / mo Subscribe	Mega $20.00 / mo Subscribe
Requests	**100 / day** + 0,0001 USD each other	**7500 / day** Hard Limit	**75 000 / day** Hard Limit	**150 000 / day** Hard Limit
Features				
Seasons	✓	✓	✓	✓
Leagues	✓	✓	✓	✓
Countries	✓	✓	✓	✓
Teams	✓	✓	✓	✓
Statistics	✓	✓	✓	✓
Historical Data	✓	✓	✓	✓
Games	✓	✓	✓	✓
Standings	✓	✓	✓	✓
Odds	✓	✓	✓	✓
Livescore	✓	✓	✓	✓
Rate Limit	10 requests per minute	300 requests per minute	450 requests per minute	900 requests per minute

Rysunek 1. Ceny istniejących REST API według źródła internetowego (https://docs.rapidapi.com/).

2 Zdefiniowanie zasobów projektu, przedstawienie schematu ERD bazy danych wraz z wyznaczeniem zasobów pośrednich oraz deklaracja reprezentacji zasobów

2.1 Zdefiniowanie zasobów projektu

Zasoby projektu:

- Tournament:
  - Tournament ID – klucz główny tabeli, wykorzystywany do formułowania zapytań oraz formowania relacji w bazie danych,
  - Tournament name – pełna nazwa turnieju, np. “Turniej młodzików 2022”,
  - Season ID – klucz obcy z tabeli Sezon;
- Season:
  - Season ID,
  - Season name – przykładowo “2022/2023”,
  - Shortened season name – skrócona nazwa sezonu, np. “22/23”;
- Tournament stage:
  - Stage ID,
  - Stage name – nazwa etapu turnieju, np. “Kwalifikacje”,
  - Tournament ID – turniej, w którym jest dany etap;
- Stage round:
  - Round ID,
  - Round name – nazwa rundy etapu (np. runda 1, runda 2…),
  - Stage ID – etap w którym jest dana runda;
- Team:
  - Team ID,
  - Team name – pełna nazwa drużyny, np. “Orły Wrocław”,
  - Shortened team name – wykorzystywane w scoreboardach (3 znaki), np. “OWR”,
- Player:
  - Player ID,
  - Name – imię zawodnika, np. “Bartosz”,
  - Surname – nazwisko zawodnika, np. “Kurek”,
  - Team ID – drużyna do której należy zawodnik;
- Match:
  - Match ID,
  - Match name – nominalny tytuł meczu, np. “Orły Wrocław vs Jastrzębie Opole”,
  - Round ID – runda, w której rozgrywany jest mecz,
  - Match date – data rozgrywania meczu, np. 2022-01-20,
  - Team1 ID – drużyna 1 (zazwyczaj gospodarz), np. “Orły Wrocław”,
  - Team2 ID – drużyna 2 (zazwyczaj goście), np. “Jastrzębie Opole”,
  - Result – wynik meczu w setach, np. “3-2”;

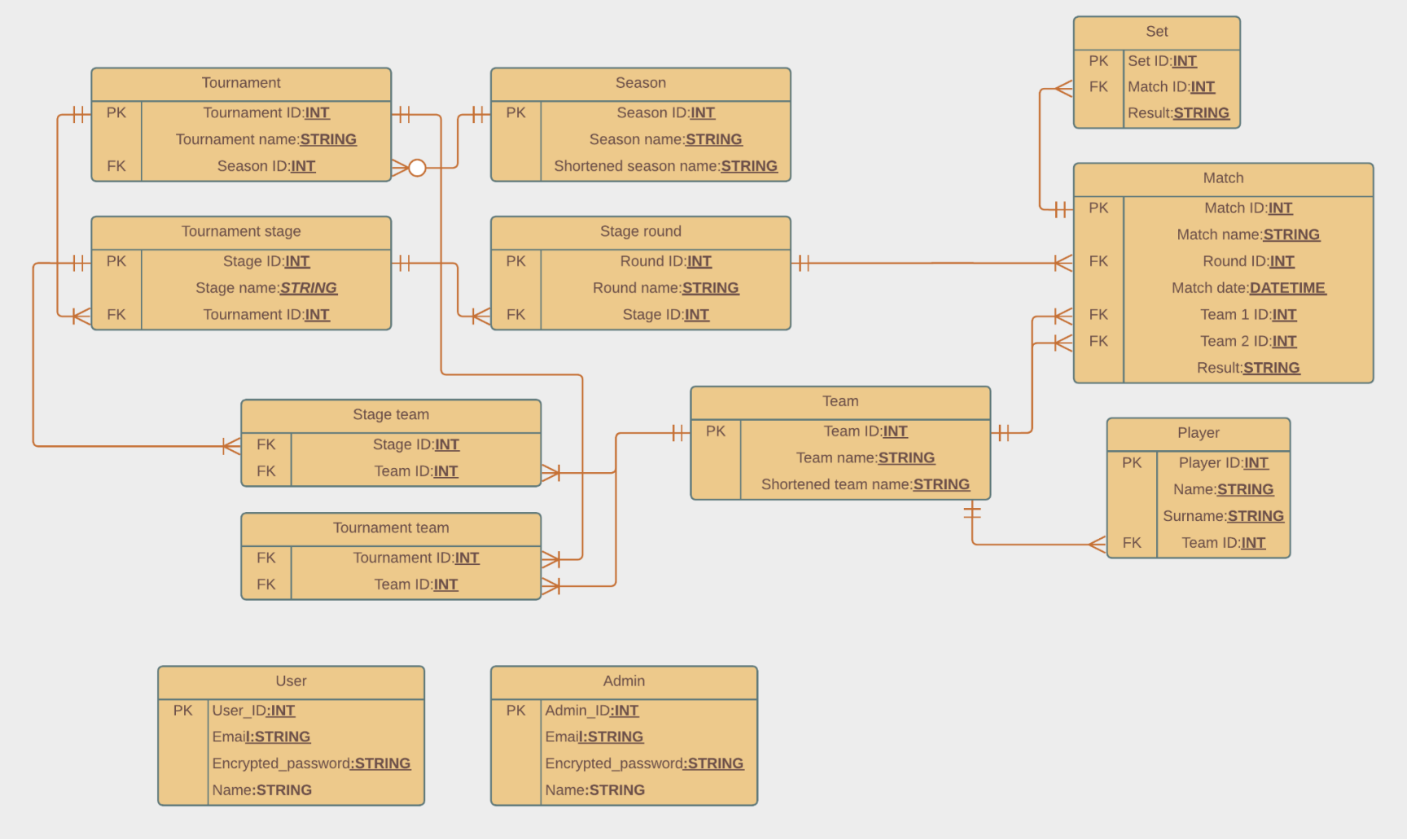
- Set:
  - Set ID,
  - Match_ID – mecz, w którym zagrano set,
  - Set_number – numer setu,
  - Result – wynik setu w punktach, np.: “22-20”.

Zasoby pośrednie:

- Tournament teams – relacja wiele do wielu tabela turniej z tabelą drużyna:
  - Tournament ID,
  - Team ID;
- Stage teams – relacja wiele do wielu tabela etap z tabelą drużyna:
  - Etap ID,
  - Team ID.

Dwie tabele użytkowników/administratorów wykorzystywane w celach autentykacji, generowane automatycznie przez gem’a zajmującego się autentykacją (devise_token_auth).

2.2 Projekt bazy danych (schemat ERD)



2.3 Deklaracja reprezentacji zasobów

Przykładowe zapytania i ich zwracane obiekty JSON (dla każdej tabeli):

Season info:

```
“Data”: [
{
“Season ID”: 1,
“Season name”: ”Sezon 2022/2023”,
“Shortened season name”: ”22/23”
},
{...}...]
```

Tournament info:

```
“Data”: [
{
“Tournament ID”: 1,
“Tournament name”: “Turniej młodzików Bydgoszcz”,
“Season”:
      {
      “Season ID”: 1,
      “Season name”: “2022/2023”,
      “Shortened season name”: “22/23”
      }
},
{...},...]
```

Tournament stage info:

```
“Data”: [
{
“Stage ID”: 1,
“Stage name”: “Kwalifikacje”,
“Tournament ID”: 1
},
{...}...]
```

Round info:

```
“Data”: [
{
“Round ID”: 1,
“Round name”: “Runda 1”,
“Stage ID”: 1
},
{...}...]
```

Stage teams info:

```
“Data”: [
{
“Stage ID”: 1,
“Team”:
      {
      “Team ID”: 1,
      “Team name”: “Orły Bydgoszcz”,
      “Shortened team name”: “OBD”
      }
,
},
{...}...]
```

Tournament teams info:

```
“Data”: [
{
“Tournament ID”: 1,
“Team”:
{
“Team ID”: 1,
“Team name”: “Orły Bydgoszcz”,
“Shortened team name”: “OBD”
}
,
},
{...}...]
```

Teams info:

```
“Data”: [
{
“Team ID”: 2,
“Team name”: “Asseco Resovia Rzeszów”,
“Shortened team name”: „ARR”
},
{...}...]
```

Players info:

```
“Data”: [
{
“Player ID”: 1,
“Name”: “Bartosz”,
“Surname”: “Kurek”,
“Team ID”: 1
},
```

```
{...}...]
```

Match info:

```
“Data”: [
{
“Match ID”: 1,
“Match name”: “Niemcy - Portugalia”,
“Round ID”: 1,
“Match date”: “23.10.2022 19:00:00”,
“Team 1 ID”: 3,
“Team 2 ID”: 4,
“Result”: “2-3”
},
{...}...]
```

Set info:

```
“Data”: [
{
“Set ID”: 1,
“Match ID”: 1,
“Result”: “22-20”
},
{...}...]
```

Zwracane dane dla przykładowego zapytania po funkcyjnym przetworzeniu danych:

Team statistics for Team ID=1 and Tournament ID=1


```
{
   "status": "SUCCESS",
   "message": "Loaded team stats",
   "data": {
       "Position": 9,
       "Matches played": 38,
       "Matches won": 14,
       "Matches lost": 24,
       "Match": [...],
       "Sets won": 56,
       "Sets lost": 54,
       "Points won": 3355,
       "Points lost": 3308,
       "Small points won": 166,
       "Small points lost": 119
   }
```

3 Specyfikacja interfejsu programistycznego oraz określenie sposobu autentykacji

3.1 Specyfikacja interfejsu programistycznego

Seasons collection [/seasons]

Seasons Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Season ID": 2223,
                "Season name": "Sezon 2022/2023",
                "Shortened season name": "22/23"
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Create a New Season [POST]

```
+ Request (application/json)
      {
            "Season name": "2022/2023",
            "Shortened season name": "22/23"
      }
+ Response 201 (application/json)
      + Headers
            Location: /seasons/2223
      + Body
            {
                  "Season ID": 1,
                  "Season name": "2022/2023",
                  "Shortened season name": "22/23"
            }
```

Delete a Season [DELETE]

```
+ Request (application/json)
      {
            "Season ID": 1
      }
+ Response 204 (application/json)
```

Update a Season [PATCH]

```
+ Request (application/json)
      {
                  "Season name": "2021/2022",
                  "Shortened season name": "21/22"
      }
+ Response 200 (application/json)
            {
                  "Season ID": 1,
                  "Season name": "2021/2022",
                  "Shortened season name": "21/22"
            }
```

Tournaments collection [/tournaments]

Tournaments Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Tournament ID": 1,
                "Tournament name": "Turniej młodzików Bydgoszcz",
                "Season":
                {
                    "Season ID": 1,
                    "Season name": "Sezon 2022/2023",
                    "Shortened season name": "22/23"
                }
            }
        ]
        + Response 404 (application/json)

      {
            "message": "NOT FOUND"
      }
```

Create a New Tournament [POST]

```
+ Request (application/json)
      {
            "Tournament name": "Katowice Open",
            "Season ID": 1,
      }
+ Response 201 (application/json)
      + Headers
            Location: /tournaments/1
      + Body
            {
                  "Tournament ID": 1,
                  "Tournament name": "Katowice Open",
                  "Season ID": 1
            }
```

Delete a Tournament [DELETE]

```
+ Request (application/json)
      {
            "Tournament ID": 1
      }
+ Response 204 (application/json)
```

Update Tournament [PATCH]

```
+ Request (application/json)
      {
            "Tournament name": "Wrocław Open",
            "Season ID": 1,
      }
+ Response 200 (application/json)
            {
                  "Tournament ID": 1,
                  "Tournament name": "Wrocław Open",
                  "Season ID": 1
            }
```

Stages collection [/stages]

Stages Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Stage ID": 1,
                "Stage name": "Kwalifikacje",
                "Tournament":
                {
                    "Tournament ID": 1,
                    "Tournament name": "Turniej młodzików Bydgoszcz",
                    "Season ID": 1
                }
            }
        ]
        + Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Create a New Tournament Stage [POST]

```
+ Request (application/json)
      {
            "Stage name": "Kwalifikacje",
            "Tournament ID": 1
      }
+ Response 201 (application/json)
      + Headers
            Location: /stages/1
      + Body
            {
                  "Stage ID": 1,
                  "Stage name": "Kwalifikacje",
                  "Tournament ID": 1
            }
```

Delete a Stage [DELETE]

```
+ Request (application/json)
      {
            "Stage ID": 1,
      }
+ Response 204 (application/json)
```

Update Tournament Stage [PATCH]

```
+ Request (application/json)
      {
            "Stage name": "Faza pucharowa",
            "Tournament ID": 1
      }
+ Response 200 (application/json)
            {
                  "Stage ID": 1,
                  "Stage name": "Faza pucharowa",
                  "Tournament ID": 1
            }
```

Rounds collection [/rounds]

Rounds Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Round ID": 1,
                "Round name": "Runda 1",
                "Stage ID":1
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Create a New Round [POST]

```
+ Request (application/json)
      {
            "Round name": "Runda 1",
            "Stage ID": 1
      }
+ Response 201 (application/json)
      + Headers
            Location: /roundinfo/1
      + Body
            {
                  “Round ID”: 1,
                  “Round name”: ”Runda 1”,
                  “Stage ID”: 1
            }
```

Delete a Round [DELETE]

```
+ Request (application/json)
      {
            "Round ID": 1,
      }
+ Response 204 (application/json)
```

Update Round [PATCH]

```
+ Request (application/json)
      {
            "Round name": "Runda 2",
            "Stage ID": 1
      }
```

```
+ Response 200 (application/json)
            {
                  “Round ID”: 1,
                  “Round name”: ”Runda 2”,
                  “Stage ID”: 1
            }
```

Stage teams collection [/steams]

###Stage Teams info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Stage ID": 1,
                "Team":
                {
                    "Team ID": 1,
                    "Team name": "Orły Bydgoszcz",
                    "Shortened team name": "OBD"
                }
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Bind a Team to a Stage [POST]

```
+ Request (application/json)
      {
            "Stage ID": 1,
            "Team ID": 2,
      }
+ Response (application/json)
      + Headers
            Location: /steams
      + Body
            {
                  "Stage ID": 1,
                  "Team ID": 2,
            }
```

Tournament teams collection [/tteams]

###Tournament teams Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Tournament ID": 1,
                "Team":
                {
                    "Team ID": 1,
                    "Team name": "Orły Bydgoszcz",
                    "Shortened team name": "OBD"
                }
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Bind a Team to a Tournament [POST]

```
+ Request (application/json)
      {
            "Tournament ID": 1,
            "Team ID": 2,
      }
+ Response (application/json)
      + Headers
            Location: /tteams/1
      + Body
            {
                  "Tournament ID": 1,
                  "Team ID": 2,
            }
```

Teams collection [/teams]

Teams Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Team ID": 2,
                "Team name": "Asseco Resovia Rzeszów",
                "Shortened team name": "ARR"
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Create a New Team [POST]

```
+ Request (application/json)
      {
            "Team name": "Orły Bydgoszcz",
            "Shortened team name": "OBD"
      }
+ Response 201 (application/json)
      + Headers
            Location: /teams/1
      + Body
            {
                  "Team ID": "1",
                  "Team name": "Orły Bydgoszcz",
                  "Shortened team name": "OBD"
            }
```

Delete a Team [DELETE]

```
+ Request (application/json)
      {
            "Team ID": 1,
      }
+ Response 204 (application/json)
```

Update Team [PATCH]

```
+ Request (application/json)
      {
            "Team name": "Orły Wrocław",
            "Shortened team name": "OWR"
      }
```

```
+ Response 200 (application/json)
            {
                  "Team ID": "1",
                  "Team name": "Orły Wrocław",
                  "Shortened team name": "OWR"
            }
```

Players collection [/players]

Players Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Player ID": 1,
                "Name": "Bartosz",
                "Surname": "Kurek",
                "Team ID": 1
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Create a New Player [POST]

```
You may add a new player using this action. It takes a JSON
object containing the players name, surname and the ID of the team he
belongs to.
+ Request (application/json)
      {
            “Name”: ”Bartosz”,
            “Surname”: ”Kurek”,
            “Team ID”: "1"
      }
+ Response 201 (application/json)
      + Headers
            Location: /players/1
      + Body

            {
                  “Player ID”: 1,
                  “Name”: ”Bartosz”,
                  “Surname”: ”Kurek”,
                  “Team ID”: "1"
            }
```

Delete a Player [DELETE]

```
+ Request (application/json)
      {
            “Player ID”: “1”
      }
+ Response 204 (application/json)
```

Update Player [PATCH]

```
+ Request (application/json)
      {
            “Name”: ”Karol”,
            “Surname”: ”Kłos”,
            “Team ID”: "2"
      }
+ Response 201 (application/json)
      + Headers
            Location: /players/1
      + Body
            {
                  “Player ID”: 1,
                  “Name”: ”Karol”,
                  “Surname”: ”Kłos”,
                  “Team ID”: "2"
            }
```

Matches collection [/match]

Match Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Match ID":1,
                "Match name": "Niemcy - Portugalia",
                "Round ID": 1,
                "Match date": "23.10.2022 19:00:00",
                "Team 1 ID": 3,
                "Team 2 ID": 4,
                "Result": "2-3"
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Create a New Match [POST]

```
+ Request (application/json)
      {
            "Match name": "Niemcy - Portugalia",
            "Round ID": 1,
            "Match date": "23.10.2022 19:00:00",
            "Team 1 ID": 1,
            "Team 2 ID": 2,
            "Result": "22-20"
      }
+ Response 201 (application/json)
      + Headers
            Location: /match/1
      + Body
            {
                  "Match ID": 1,
                  "Match name": "Niemcy - Portugalia",
                  "Round ID": 1,
                  "Match date": "23.10.2022 19:00:00",
                  "Team 1 ID": 1,
                  "Team 2 ID": 2,
                  "Result": "22-20"
            }
```

Delete a Match [DELETE]

```
+ Request (application/json)
      {
            "Match ID": 1,
```

```
      }
+ Response 204 (application/json)
```

Update Match [PATCH]

```
+ Request (application/json)
      {
            "Match name": "Niemcy - Hiszpania",
            "Round ID": 1,
            "Match date": "05.11.2022",
            "Team 1 ID": 1,
            "Team 2 ID": 2,
            "Result": "22-20"
      }
+ Response 200 (/application/json)
            {
                  "Match ID": 1,
                  "Match name": "Niemcy - Hiszpania",
                  "Round ID": 1,
                  "Match date": "05.11.2022",
                  "Team 1 ID": 1,
                  "Team 2 ID": 2,
                  "Result": "22-20"
            }
```

Sets collection[/sets]

Match set Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "Set ID":1,
                "Match ID": 1,
         "Set number": 1,
                "Result": "22-20"
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Create a New Set [POST]

```
+ Request (application/json)
      {
            "Match ID": 1,
            "Set number": 1,
             "Result": "22-20"
      }
+ Response 201 (application/json)
      + Headers
            Location: /setinfo/1
      + Body
            {
                  "Set ID": 1,
                  "Match ID": 1,
                  "Set number": 1,
                  "Result": "22-20"
            }
```

Delete a Set [DELETE]

```
+ Request (application/json)\
      {
            "set ID": 1,
      }
+ Response 204 (application/json)
```

Update Set [PATCH]

```
+ Request (application/json)
      {
            "Match ID": 1,
            "Set number": 2,
```

```
            "Result": "22-20"
      }
+ Response 200 (application/json)
           {
                 “Set ID”: 1,
                 "Match ID": 1,
                 “Set number”: 2,
                  "Result": "22-20"
           }
```

Team stats info [/team_stats]

List Team Stats Info [GET]

```
+ Response 200 (application/json)
        {
    "status": "SUCCESS",
    "message": "Loaded team stats",
    "data": {
        "Position": 9,
        "Matches played": 38,
        "Matches won": 14,
        "Matches lost": 24,
        "Match": [...],
        "Sets won": 56,
        "Sets lost": 54,
        "Points won": 3355,
        "Points lost": 3308,
        "Small points won": 166,
        "Small points lost": 119
    }

+Response 401 (application/json)
      {
          "status": "Unauthorized",
          "message": "Please sign in or sign up",
          "data": 401
      }
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

3.2 Określenie sposobu autentykacji

Logowanie użytkowników w systemie zwraca w headerze odpowiedzi access-token, który należy zawrzeć w headerze zapytania, aby system odpowiednio odpowiedział. Tokeny mają określoną datę ważności oraz są regenerowane po każdym zapytaniu.

Dzielimy użytkowników na zwykłych użytkowników oraz administratorów. Zwykli użytkownicy mogą się sami rejestrować zapytaniem POST, oraz sami mogą zmieniać dane lub usuwać swoje konta. Konta użytkowników mogą być wykorzystane do wykonania prostych zapytań **GET**.

Aby mieć dostęp do zapytań DEL, POST, PUT, należy zalogować się jako administrator.

Administracja musi być zarejestrowana administracyjnie bezpośrednio przez bazę danych w celu zwiększenia bezpieczeństwa aplikacji.

Przy utracie połączenia (nawet chwilowej) należy wygenerować nowy token.

4 Projekt architektury aplikacji internetowej

4.1 Stos technologiczny

Ruby – język programowania.

Ruby on rails – framework do zbudowania REST API.

Git – VCS (version control system).

Github – serwer VCS.

XAMPP – serwer bazy danych (MySQL, phpmyadmin). Wykorzystanie gem'a mysql2, żeby połączyć bazę z Ruby on Rails.

Postman – testowanie zapytań REST'owych.

4.2 Aplikacja kliencka

Prosta aplikacja terminalowa umożliwiająca wykonywanie zapytań do naszego API.