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:

- tournaments:
  - 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;
- seasons:
  - id,
  - Season_name – przykładowo “2022/2023”,
  - Shortened_season_name – skrócona nazwa sezonu, np. “22/23”;
- tournament_stages:
  - id,
  - stage_name – nazwa etapu turnieju, np. “Kwalifikacje”,
  - tournament_id – turniej, w którym jest dany etap;
- stage_rounds:
  - id,
  - round_name – nazwa rundy etapu (np. runda 1, runda 2…),
  - tournament_stage_id – etap w którym jest dana runda;
- teams:
  - id,
  - Team_name – pełna nazwa drużyny, np. “Orły Wrocław”,
  - Shortened_team_name – wykorzystywane w scoreboardach (3 znaki), np. “OWR”,
- players:
  - id,
  - name – imię zawodnika, np. “Bartosz”,
  - surname – nazwisko zawodnika, np. “Kurek”,
  - team_id – drużyna do której należy zawodnik;
- matches:
  - 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”;

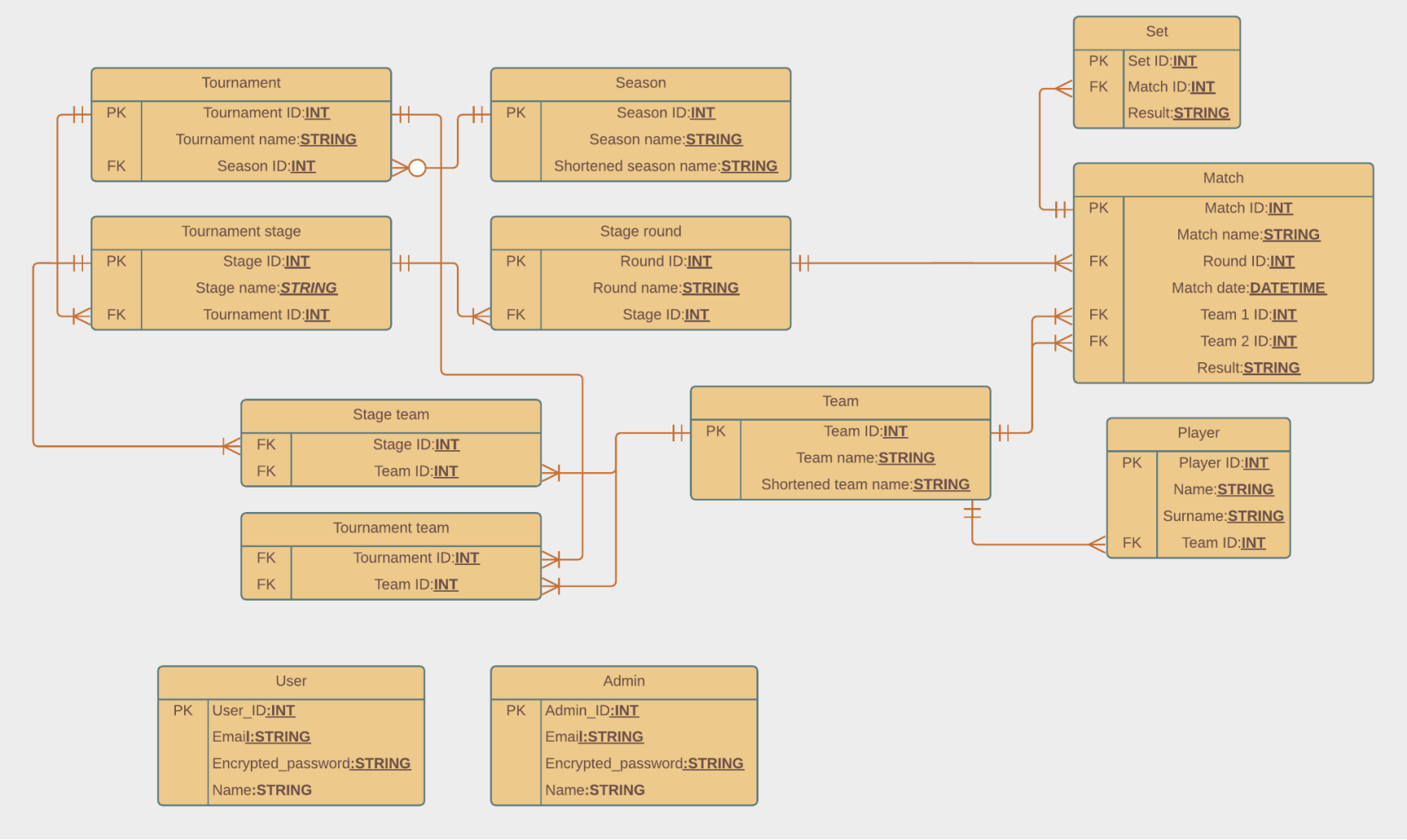
- match_sets:
  - id,
  - match_id – mecz, w którym zagrano set,
  - set_number – numer setu,
  - result – wynik setu w punktach, np.: “22-20”.

Zasoby pośrednie:

- tournament_teams – relacja wiele do wielu tabela turniej z tabelą drużyna:
  - id,
  - tournament_id,
  - team_id;
- stage_teams – relacja wiele do wielu tabela etap z tabelą drużyna:
  - id,
  - team_id,
  - tournament_stage_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”: ]
{
“id”: 1,
“Season_name”: ”Sezon 2022/2023”,
“Shortened_season_name”: ”22/23”
},
{...}...]
```

Tournament info:

```
“data”: [
{
“id”: 1,
“tournament_name”: “Turniej młodzików Bydgoszcz”,
“season_id”: 1
},
{...},...]
```

Tournament stage info:

```
“data”: [
{
“stage_id”: 1,
“stage_name”: “Kwalifikacje”,
“tournament_id”: 1
},
{...}...]
```

Round info:

```
“data”: [
{
“id”: 1,
“round_name”: “Runda 1”,
“stage_id”: 1
},
{...}...]
```

Stage teams info:

```
“data”: [
{
“id”: 1,
```

```
“team_id”: 1,
“tournament_stage_id”: 1
},
{...}...]
```

Tournament teams info:

```
“data”: [
{
“id”: 1,
“tournament_id”: 1,
“team_id”: 1
},
{...}...]
```

Teams info:

```
“data”: [
{
“id”: 2,
“team_name”: “Asseco Resovia Rzeszów”,
“shortened_team_name”: „ARR”
},
{...}...]
```

Players info:

```
“data”: [
{
“id”: 1,
“name”: “Bartosz”,
“surname”: “Kurek”,
“team_id”: 1
},
{...}...]
```

Match info:

```
“data”: [
{
“id”: 1,
“match_name”: “Niemcy - Portugalia”,
“round_id”: 1,
“match_date”: “23.10.2022 19:00:00”,
“team1_id”: 3,
“team2_id”: 4,
“result”: “2-3”
},
```

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

Match set info:

```
“data”: [
{
“id”: 1,
“match_id”: 1,
“set_number”: 5,
“result”: “22-20”
},
{...}...]
```

Zwracane dane dla przykładowego zapytania po funkcyjnym przetworzeniu danych:

Team statistics for id=1 and 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)
        [
            {
                "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
            {
                  "id": 1,
                  "Season_name": "2022/2023",
                  "Shortened_season_name": "22/23"
            }
```

Delete a Season [DELETE]

```
+ Request (application/json)
      {
            "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)
            {
                  "id": 1,
                  "Season_name": "2021/2022",
                  "Shortened_season_name": "21/22"
            }
```

Tournaments collection [/tournaments]

Tournaments Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "id": 1,
                "tournament_name": "Turniej młodzików Bydgoszcz",
                "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",
            "id": 1,
      }
+ Response 201 (application/json)
      + Headers
            Location: /tournaments/1
      + Body
            {
                  "id": 1,
                  "tournament_name": "Katowice Open",
                  "season_id": 1
            }
```

Delete a Tournament [DELETE]

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

Update Tournament [PATCH]

```
+ Request (application/json)
      {
            "tournament_name": "Wrocław Open",
            "season_id": 1,
      }
+ Response 200 (application/json)
            {
                  "id": 1,
                  "tournament_name": "Wrocław Open",
                  "season_id": 1
            }
```

Tournament stages collection [/tournament_stages]

Stages Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "stage_id": 1,
                "stage_name": "Kwalifikacje",
                "Tournament":
                {
                    "id": 1,
                    "tournament_name": "Turniej młodzików Bydgoszcz",
                    "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)
      {
            "id": 1,
      }
+ Response 204 (application/json)
```

Update Tournament Stage [PATCH]

```
+ Request (application/json)
      {
            "stage_name": "Faza pucharowa",
            "tournament_id": 1
      }
+ Response 200 (application/json)
            {
                  "id": 1,
                  "stage_name": "Faza pucharowa",
                  "tournament_id": 1
            }
```

Stage rounds collection [/stage_rounds]

Rounds Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "id": 1,
                "round_name": "Runda 1",
                "tournament_stage_id":1
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Create a New Round [POST]

```
+ Request (application/json)
      {
            "round_name": "Runda 1",
            "tournament_stage_id": 1
      }
+ Response 201 (application/json)
      + Headers
            Location: /roundinfo/1
      + Body
            {
                  “id”: 1,
                  “round_name”: ”Runda 1”,
                  “tournament_stage_id”: 1
            }
```

Delete a Round [DELETE]

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

Update Round [PATCH]

```
+ Request (application/json)
      {
            "round_name": "Runda 2",
            "tournament_stage_id": 1
      }
```

```
+ Response 200 (application/json)
          {
               “id”: 1,
               “round_name”: ”Runda 2”,
               “tournament_stage_id”: 1
          }
```

Stage teams collection [/stage_teams]

###Stage Teams info [GET]

```
+ Response 200 (application/json)
        [
            {
                "id": 1,
                "team_id": 2,
                "tournament_stage_id": 1
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Bind a Team to a Stage [POST]

```
+ Request (application/json)
      {
            "team_id": 2,
            "tournament_stage_id": 1
      }
+ Response (application/json)
      + Headers
            Location: /steams
      + Body
            {
                  "id": 1,
                  "team_id": 2,
                  "tournament_stage_id": 1

            }
```

Tournament teams collection [/tournament_teams]

###Tournament teams Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "id": 1,
                "tournament_id": 2,
                "team_id": 1
            }
        ]
+ Response 404 (application/json)
      {
            "message": "NOT FOUND"
      }
```

Bind a Team to a Tournament [POST]

```
+ Request (application/json)
      {
            "tournament_id": 2,
            "team_id": 1

      }
+ Response (application/json)
      + Headers
            Location: /tteams/1
      + Body
            {
                  "id": 1,
                  "tournament_id": 2,
                  "team_id": 1
            }
```

Teams collection [/teams]

Teams Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "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
            {
                  "id": "1",
                  "team_name": "Orły Bydgoszcz",
                  "shortened_team_name": "OBD"
            }
```

Delete a Team [DELETE]

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

Update Team [PATCH]

```
+ Request (application/json)
      {
            "team_name": "Orły Wrocław",
            "shortened_team_name": "OWR"
      }
```

```
+ Response 200 (application/json)
            {
                  "id": "1",
                  "team_name": "Orły Wrocław",
                  "shortened_team_name": "OWR"
            }
```

Players collection [/players]

Players Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "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

            {
                  “id”: 1,
                  “name”: ”Bartosz”,
                  “surname”: ”Kurek”,
                  “team_id”: "1"
            }
```

Delete a Player [DELETE]

```
+ Request (application/json)
      {
            “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
            {
                  “id”: 1,
                  “name”: ”Karol”,
                  “surname”: ”Kłos”,
                  “team_id”: 2
            }
```

Matches collection [/matches]

Match Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "id":1,
                "match_name": "Niemcy - Portugalia",
                "round_id": 1,
                "match_date": "23.10.2022 19:00:00",
                "team1_id": 3,
                "team2_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",
            "team1_id": 1,
            "team2_id": 2,
            "result": "22-20"
      }
+ Response 201 (application/json)
      + Headers
            Location: /match/1
      + Body
            {
                  "id": 1,
                  "match_name": "Niemcy - Portugalia",
                  "round_id": 1,
                  "match_date": "23.10.2022 19:00:00",
                  "team1_id": 1,
                  "team2_id": 2,
                  "result": "22-20"
            }
```

Delete a Match [DELETE]

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

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

Update Match [PATCH]

```
+ Request (application/json)
      {
            "match_name": "Niemcy - Hiszpania",
            "round_id": 1,
            "match_date": "05.11.2022",
            "team1_id": 1,
            "team2_id": 2,
            "result": "22-20"
      }
+ Response 200 (/application/json)
            {
                  "id": 1,
                  "match_name": "Niemcy - Hiszpania",
                  "round_id": 1,
                  "match_date": "05.11.2022",
                  "team1_id": 1,
                  "team2_id": 2,
                  "result": "22-20"
            }
```

Match sets collection[/match_sets]

Match set Info [GET]

```
+ Response 200 (application/json)
        [
            {
                "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
            {
                  “id”: 1,
                  "match_id": 1,
                  “set_number”: 1,
                  "result": "22-20"
            }
```

Delete a Set [DELETE]

```
+ Request (application/json)\
      {
            "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)
            {
                  “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.

3.3 Uwagi

W trakcie realizacji projektu uległy zmianie nazwy niektórych zmiennych, np. zaczynające się z dużej litery na małą, w celu większego dopasowania do domyślnej składni Ruby on Rails. Z powodu zmian w modelach niektórych relacji zaktualizowano zapytania na zgodne z rzeczywistymi.

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.