Temat: System statystyk sportowych dla siatkówki
Adam Sołtysiak 259722
Artur Chrapek 259710
Adam Gołecki 259696
Nataliia Belinska 257748
Jan Świergoń 250167
Dokumentacja projektu (poszczególne etapy złączone)

Zaprezentowanie rynkowego opisu projektu wraz z planem komercjalizacji. Opisać słownie, czego ma dotyczyć projekt, porównać proponowane rozwiązanie z już istniejącymi na rynku.

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" .

2. **Plan komercjalizacji:**

Sprzedaż tokenów dostępu (czasowy/ilościowo po zapytaniach).
Jak czasowy to np. sprzedaż tokenu na X dni (z ograniczeniem X zapytań dziennie).
Możliwość zapłaty zewnętrznym deweloperom pomagającym zbierać dane statystyczne/ na żywo z meczów.

3. **Czego dotyczy projekt:**

Statystyki siatkowe:
Statystyki wygranych/przegranych meczy
Wyniki setów
Ilość rozegranych setów
Turnieje
Drużyny na turniejach
Skład drużyn

4. **Rozwiązania już istniejące na rynku:**

Szeroko rozwinięta branża - duża ilość podobnych aplikacji każda oferująca podobne statystyki
-historyczne wyniki i statystyki setów
-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

https://rapidapi.com/BroadageSports/api/volleyball-data

https://developer.geniussports.com/warehouse/rest/index_volleyball.html?fbclid=IwAR0bKU2-Vroa2COpnS2z2sUzzcqHfBrN4-o_7-RsaO7pPb3PSf8B6M3ZbE8#

https://developer.sportradar.com/docs/read/baseline_sports_coverage/Volleyball_Indoor_v2?fbclid=IwAR2eEkg3Inse_oBoXdmBYk7S-kzUd-yqyHfICGhSm83rxvG_QvEAnqyujk8#indoor-volleyball-api-overview

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

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

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

1. Zdefiniowanie zasobów projektu

Zasoby projektu:

a. Tournament
   i. Tournament ID - klucz główny tabeli, wykorzystywany do formułowania zapytań oraz formowania relacji w bazie danych.
   ii. Tournament name -pełna nazwa turnieju
   iii. Season ID - klucz obcy z tabeli Sezon
b. Season
   i. Season ID
   ii. Season name - przykładowo 2022/2023
   iii. Shortened season name - skrócona nazwa sezonu np 22/23
c. Tournament stage
   i. Stage ID
   ii. Stage name - nazwa etapu turnieju
   iii. Tournament ID - turniej w którym jest dany etap
d. Stage round
   i. Round ID
   ii. Round name - nazwa rundy etapu (zazwyczaj po prostu runda 1, runda 2…)
   iii. Stage ID - etap w którym jest dana runda
e. Team
   i. Team ID
   ii. Team name - pełna nazwa drużyny
   iii. Shortened team name - do wykorzystania w scoreboardach (3 znaki)
f. Player
   i. Player ID
   ii. Name - imię zawodnika
   iii. Surname - nazwisko zawodnika
   iv. Team ID - drużyna do której należy zawodnik

g. Match
   i. Match ID
   ii. Match name - nominalny tytuł meczu
   iii. Round ID - runda w której rozgrywany jest mecz
   iv. Match date - data rozgrywania meczu
   v. Team 1 ID - drużyna 1 (zazwyczaj gospodarz)

vi. Team 2 ID - drużyna 2 (zazwyczaj goście)
vii. Result - wynik meczu w setach np 3-2

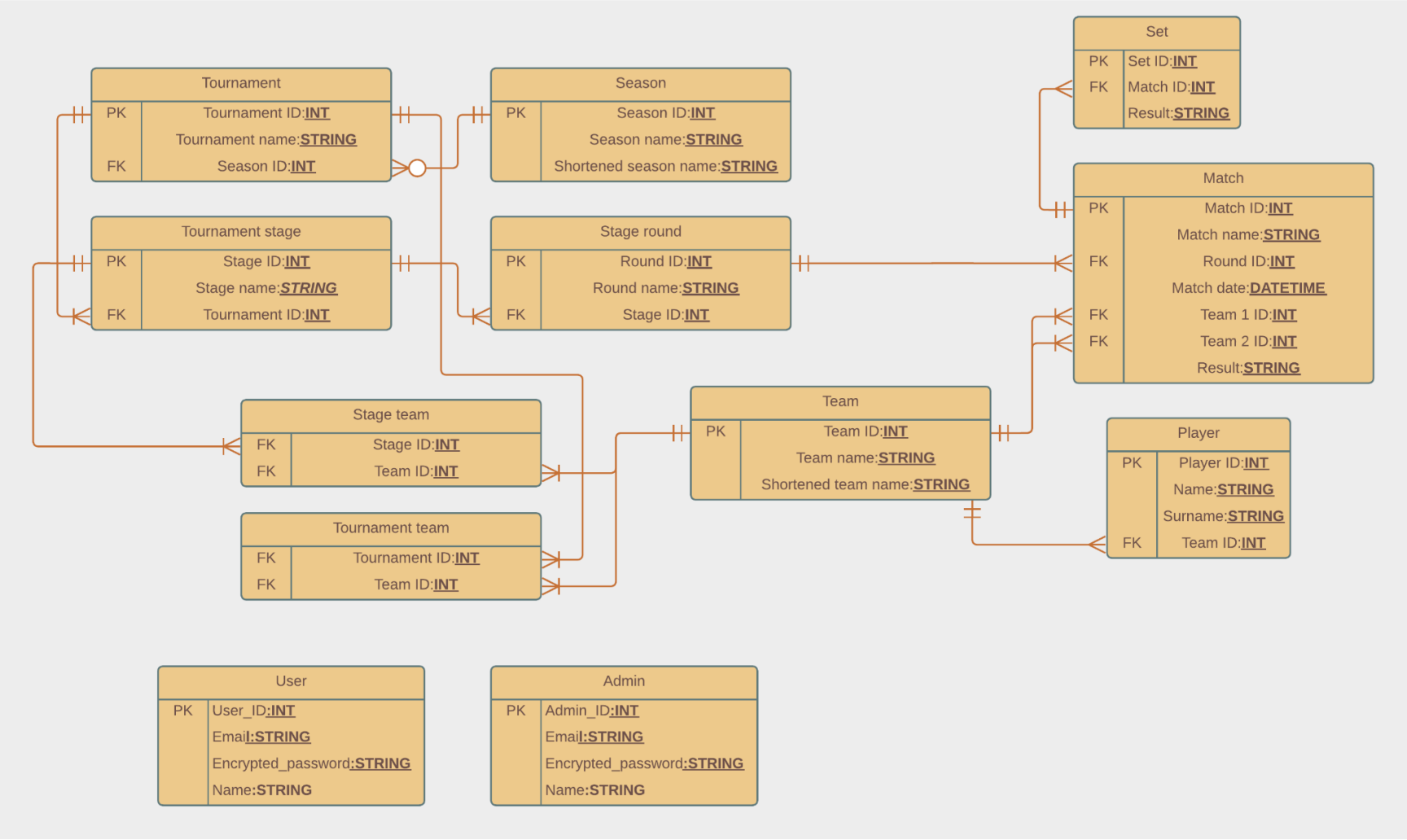
h. Set
   i. Set ID
   ii. Match ID
   iii. Result - wynik setu w punktach np: 22-20

Zasoby pośrednie:

a. Tournament teams - relacja wiele do wielu tabela turniej z tabelą drużyna
   i. Tournament ID
   ii. Team ID
b. Stage teams - relacja wiele do wielu tabela etap z tabelą drużyna
   i. Etap ID
   ii. Team ID

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

2. Projekt bazy danych (schemat ERD)



3. Deklaracja reprezentacji zasobów

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

Season info:

“Data”: [
{
“Season ID”:0001,
“Season name”:”Sezon 2022/2023”,
“Shortened season name”:”22/23”
},
{...}...]

Tournament info:

“Data”: [
{
“Tournament ID”:0001,
“Tournament name”:”Turniej młodzików Bydgoszcz”,
“Season”:
{
“Season ID”:0001,
“Season name”:”2022/2023”,
“Shortened season name”:”22/23”
}
},
{...},...]

Tournament stage info:

“Data”: [
{
“Stage ID”:0001,
“Stage name”:”Kwalifikacje”,
“Tournament ID”:0001
},
{...}...]

Round info:

“Data”: [
{
“Round ID”:0001,
“Round name”:”Runda 1”,
“Stage ID”:0001
},
{...}...]

Stage teams info:

```
“Data”: [
{
“Stage ID”: 0001,
“Team”:
        {
        “Team ID”:0001,
        “Team name”:” Orły Bydgoszcz”,
        “Shortened team name”:”OBD”
        }
,
},
{...}...]
```

Tournament teams info:

```
“Data”: [
{
“Tournament ID”: 0001,
“Team”:
        {
        “Team ID”:0001,
        “Team name”:” Orły Bydgoszcz”,
        “Shortened team name”:”OBD”
        }
,
},
{...}...]
```

Teams info:

```
“Data”: [
{
“Team ID”:0002,
“Team name”:”Asseco Resovia Rzeszów”,
“Shortened team name”:”ARR”
},
{...}...]
```

Players info:

“Data”: [
{
“Player ID”:0001,
“Name”:”Bartosz”,
“Surname”:”Kurek”,
“Team ID”: 0001
},
{...}...]

Match info:

“Data”: [
{
“Match ID”:0001,
“Match name”:”Niemcy - Portugalia”,
“Round ID”:0001,
“Match date”:”23.10.2022 19:00:00”,
“Team 1 ID”:0003,
“Team 2 ID”:0004,
“Result”:”2-3”
},
{...}...]

Set info:

“Data”: [
{
“Set ID”:0001,
“Match ID”: 0001,
“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
  }
```

Specyfikacja interfejsu programistycznego oraz określenie sposobu autentykacji.

4.1 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": 0001,
                "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": 0001,
            "Season name": "2021/2022",
            "Shortened season name": "21/22"
        }
```

Tournaments collection [/tournaments]

Tournaments Info [GET]

+ Response 200 (application/json)

```
[
   {
      "Tournament ID": 0001,
      "Tournament name": "Turniej młodzików Bydgoszcz",
      "Season":
      {
         "Season ID": 0001,
         "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": 0001,
   }
```

+ Response 201 (application/json)

   + Headers

   Location: /tournaments/0001

   + Body

```
   {
            "Tournament ID": 0001,
            "Tournament name": "Katowice Open",
            "Season ID": 0001
   }
```

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": 0001,
        }

+ Response 200 (application/json)

        {
                "Tournament ID": 0001,
                "Tournament name": "Wrocław Open",
                "Season ID": 0001
        }

Stages collection [/stages]

Stages Info [GET]

+ Response 200 (application/json)

```
[
   {
      "Stage ID": 0001,
      "Stage name": "Kwalifikacje",
      "Tournament":
      {
         "Tournament ID": 0001,
         "Tournament name": "Turniej młodzików Bydgoszcz",
         "Season ID": 0001
      }

   }
]
```

+ Response 404 (application/json)

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

Create a New Tournament Stage [POST]

+ Request (application/json)

```
   {
   "Stage name": "Kwalifikacje",
   "Tournament ID": 0001
   }
```

+ Response 201 (application/json)

   + Headers

   Location: /stages/0001

   + Body

```
   {
            "Stage ID": 0001,
            "Stage name": "Kwalifikacje",
            "Tournament ID": 0001
   }
```

Delete a Stage [DELETE]

+ Request (application/json)

        {
        "Stage ID": 0001,
        }

+ Response 204 (application/json)

Update Tournament Stage [PATCH]

+ Request (application/json)

        {
        "Stage name": "Faza pucharowa",
        "Tournament ID": 0001
        }

+ Response 200 (application/json)

        {
                "Stage ID": 0001,
                "Stage name": "Faza pucharowa",
                "Tournament ID": 0001
        }

Rounds collection [/rounds]

Rounds Info [GET]

+ Response 200 (application/json)

```
    [
        {
            "Round ID": 0001,
            "Round name": "Runda 1",
            "Stage ID":0001
        }
    ]
```

+ Response 404 (application/json)

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

Create a New Round [POST]

+ Request (application/json)

```
        {
        "Round name": "Runda 1",
        "Stage ID": 0001
        }
```

+ Response 201 (application/json)

        + Headers

        Location: /roundinfo/0001

        + Body

```
        {
                “Round ID”: 0001,
                “Round name”: ”Runda 1”,
                “Stage ID”: 0001
        }
```

Delete a Round [DELETE]

+ Request (application/json)

```
        {
        "Round ID": 0001,
```

```
        }
```

+ Response 204 (application/json)

Update Round [PATCH]

+ Request (application/json)

```
        {
        "Round name": "Runda 2",
        "Stage ID": 0001
        }
```

+ Response 200 (application/json)

```
        {
                “Round ID”: 0001,
                “Round name”: ”Runda 2”,
                “Stage ID”: 0001
        }
```

Stage teams collection [/steams]

###Stage Teams info [GET]

+ Response 200 (application/json)

```
    [
        {
            "Stage ID": 0001,
            "Team":
            {
                "Team ID": 0001,
                "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": 0001,
        "Team ID": 0002,
        }
```

+ Response (application/json)

        + Headers

        Location: /steams

        + Body

```
        {
                "Stage ID": 0001,
                "Team ID": 0002,
        }
```

Tournament teams collection [/tteams]

###Tournament teams Info [GET]

+ Response 200 (application/json)

```
    [
        {
            "Tournament ID": 0001,
            "Team":
            {
                "Team ID": 0001,
                "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": 0001,
        "Team ID": 0002,
        }
```

+ Response (application/json)

        + Headers

        Location: /tteams/0001

        + Body

```
        {
                "Tournament ID": 0001,
                "Team ID": 0002,
        }
```

Teams collection [/teams]

Teams Info [GET]

+ Response 200 (application/json)

```
    [
        {
            "Team ID": 0002,
            "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/0001

        + Body

```
        {
                "Team ID": "0001",
                "Team name": "Orły Bydgoszcz",
                "Shortened team name": "OBD"
        }
```

Delete a Team [DELETE]

+ Request (application/json)

```
        {
        "Team ID": 0001,
        }
```

+ Response 204 (application/json)

Update Team [PATCH]

+ Request (application/json)

```
        {
        "Team name": "Orły Wrocław",
        "Shortened team name": "OWR"
        }
```

+ Response 200 (application/json)

```
        {
                "Team ID": "0001",
                "Team name": "Orły Wrocław",
                "Shortened team name": "OWR"
        }
```

Players collection [/players]

Players Info [GET]

+ Response 200 (application/json)

```
    [
      {
        "Player ID": 0001,
        "Name": "Bartosz",
        "Surname": "Kurek",
        "Team ID": 0001
      }
    ]
```

+ 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”: "0001"
      }
```

+ Response 201 (application/json)

      + Headers

      Location: /players/0001

      + Body

```
      {
            “Player ID”: 0001,
            “Name”: ”Bartosz”,
            “Surname”: ”Kurek”,
            “Team ID”: "0001"
      }
```

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”: "0002"
}
```

+ Response 201 (application/json)

+ Headers

Location: /players/0001

+ Body

```
{
	“Player ID”: 0001,
	“Name”: ”Karol”,
	“Surname”: ”Kłos”,
	“Team ID”: "0002"
}
```

Match collection [/match]

Match Info [GET]

+ Response 200 (application/json)

```
    [
        {
            "Match ID":0001,
            "Match name": "Niemcy - Portugalia",
            "Round ID": 0001,
            "Match date": "23.10.2022 19:00:00",
            "Team 1 ID": 0003,
            "Team 2 ID": 0004,
            "Result": "2-3"

        }
    ]
```

+ Response 404 (application/json)

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

Create a New Match [POST]

+ Request (application/json)

```
        {
        "Match name": "Niemcy - Portugalia",
        "Round ID": 0001,
        "Match date": "23.10.2022 19:00:00",
        "Team 1 ID": 0001,
        "Team 2 ID": 0002,
        "Result": "22-20"
        }
```

+ Response 201 (application/json)

        + Headers

        Location: /match/0001

        + Body

```
        {
                "Match ID": 0001,
                "Match name": "Niemcy - Portugalia",
```

```
        "Round ID": 0001,
        "Match date": "23.10.2022 19:00:00",
        "Team 1 ID": 0001,
        "Team 2 ID": 0002,
        "Result": "22-20"
    }
```

Delete a Match [DELETE]

+ Request (application/json)

```
    {
    "Match ID": 0001,
    }
```

+ Response 204 (application/json)

Update Match [PATCH]

+ Request (application/json)

```
    {
    "Match name": "Niemcy - Hiszpania",
    "Round ID": 0001,
    "Match date": "05.11.2022",
    "Team 1 ID": 0001,
    "Team 2 ID": 0002,
    "Result": "22-20"
    }
```

+ Response 200 (/application/json)

```
    {
        "Match ID": 0001,
        "Match name": "Niemcy - Hiszpania",
        "Round ID": 0001,
        "Match date": "05.11.2022",
        "Team 1 ID": 0001,
        "Team 2 ID": 0002,
        "Result": "22-20"
    }
```

Sets collection[/sets]

Set Info [GET]

+ Response 200 (application/json)

```
    [
        {
            "Set ID":0001,
            "Match ID": 0001,
            "Result": "22-20"
        }
    ]
```

+ Response 404 (application/json)

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

Create a New Set [POST]

+ Request (application/json)

```
        {
        "Match ID": 0001,
        "Result": "22-20"
        }
```

+ Response 201 (application/json)

        + Headers

        Location: /setinfo/0001

        + Body

```
        {
                “Set ID”: 0001,
                "Match ID": 0001,
                "Result": "22-20"
        }
```

Delete a Set [DELETE]

+ Request (application/json)\

```
        {
        "set ID": 0001,
```

```
        }

+ Response 204 (application/json)
### Update Set [PATCH]

+ Request (application/json)


        {
        "Match ID": 0001,
        "Result": "25-20"
        }

+ Response 200 (application/json)

        {
                “Set ID”: 0001,
                "Match ID": 0001,
                "Result": "25-20"
        }
```

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"
        }
```


4.2 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.

Projekt architektury aplikacji internetowej.

5.1 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), w rails jest też wbudowany serwer bazy danych ale będzie nam łatwiej na GUI, wykorzystujemy gema (plugin w ruby) mysql2 żeby tą bazę z RoR(ruby on rails) połączyć.
Postman - testowanie zapytań REST'owych.

5.2 Aplikacja kliencka

Prosta aplikacja terminalowa umożliwiająca wykonywanie zapytań do naszego API.