DOKUMENTACJA REST API PORTALU ELOVE

1. Modele danych

User – klasa użytkownika, wykorzystywana przy autoryzacji.
Preferences – klasa zawierająca informacje o preferencjach użytkownika.
Settings– klasa zawierająca informacje o ustawieniach użytkownika.
Image – klasa będąca reprezentacją zdjęć w systemie.
Friend – klasa przechowująca informację o relacji pomiędzy użytkownikami.
Blacklist – klasa przechowująca informację o zablokowanych użytkownikach przez innego użytkownika.
Like – klasa przechowująca informację o polubieniu użytkownika przez innego użytkownika.
Report – klasa będąca reprezentacją zgłoszenia naruszenia regulaminu.
Verify – klasa przechowująca informacje o stanie konta (zweryfikowane lub niezweryfikowane).
BannedIp – klasa przechowująca informacje o zbanowanym użytkowniku.
Message – klasa będąca reprezentacją wiadomości w systemie.

2. Obsługiwane zapytania

2.1. Tworzenie konta użytkownika i autoryzacja

2.1.1. Rejestracja

- Routing: api/user/register
- Metoda: POST
    - Dane:
        - username
        - email
        - location
        - birthday
        - sex
        - password
        - password2

2.1.2. Logowanie

- Routing: api/user/login
- Metoda: POST
    - Dane:
        - username
        - password

2.1.3. Wylogowanie

- Routing: api/user/logout
- Metoda: POST
    - Wymaga uwierzytelnienia tokenem

2.1.4. Usuwanie konta

- Routing: api/user/delete
- Metoda: DELETE
    - Dane:
        - password

- Wymaga uwierzytelnienia tokenem

2.1.5. Zmiana hasła

- Routing: api/user/change-password
- Metoda: PATCH
  - Dane:
    - password
    - new_password1
    - new_password2
  - Wymaga uwierzytelnienia tokenem

2.1.6. Odzyskiwanie hasła

- Routing: api/user/restore-password
- Metoda: PATCH
  - Dane:
    - email

2.1.7. Weryfikacja adresu e-mail

- Routing: api/user/account-verify
- Metoda: PATCH
  - Dane:
    - verify_code

2.2. Zarządanie kontem użytkownika

2.2.1. Informacje o zalogowanym użytkowniku

- Routing: api/user/properties
- Metoda: GET
  - Wymaga uwierzytelnienia tokenem
- Metoda: PATCH
  - Dane:
    - email
    - name
    - surname
    - location
    - sex
    - hair_color
    - body_type
    - growth
    - weight
    - description
    - is_smoking
    - is_drinking_alcohol
    - orientation
    - eye_color
    - hair_length
    - status
    - education

2.2.2. Preferencje zalogowanego użytkownika

- Routing: api/user/preferences
- Metoda: GET
    - Wymaga uwierzytelnienia tokenem
- Metoda: PATCH
    - Dane:
        - hair_color_blonde_preference
        - hair_color_brunette_preference
        - hair_color_red_preference
        - growth_preference
        - weight_preference
        - body_type_preference
        - is_smoking_preference
        - is_drinking_alcohol_preference
        - age_preference_min
        - age_preference_max
    - Wymaga uwierzytelnienia tokenem

2.2.3. Ustawienia zalogowanego użytkownika

- Routing: api/user/settings
- Metoda: GET
    - Wymaga uwierzytelnienia tokenem
- Metoda: PATCH
    - Dane:
        - dark_theme
        - messages_privacy
        - search_privacy
        - comments_privacy
        - hide_age
    - Wymaga uwierzytelnienia tokenem

2.2.4. Zdjęcia zalogowanego użytkownika

- Routing: api/user/images
- Metoda: GET
    - Wymaga uwierzytelnienia tokenem
    - Response:
        - 401 – brak uprawnień / użytkownik niezalogowany
        - 404 – użytkownik nie istnieje
        - 200 – sukces
- Metoda: POST
    - Dane:
        - image – zdjęcie w formacie jpg, jpeg, png
    - Wymaga uwierzytelnienia tokenem
    - Response:
        - 401 – brak uprawnień / użytkownik niezalogowany
        - 404 – użytkownik nie istnieje
        - 400 – błędny typ pliku lub jego brak
        - 201 – zdjęcie zostało dodane pomyślnie

- Metoda: DELETE
    - Dane:
        - pk – primary key usuwanego zdjęcia
    - Wymaga uwierzytelnienia tokenem
    - Response:
        - 401 – brak uprawnień / użytkownik niezalogowany
        - 404 – użytkownik lub zdjęcie nie istnieje
        - 200 – zdjęcie zostało usunięte pomyślnie

2.2.5. Zdjęcie profilowe zalogowanego użytkownika

- Routing: api/user/profile-image
- Metoda: GET
    - Wymaga uwierzytelnienia tokenem
    - Response:
        - 401 – brak uprawnień / użytkownik niezalogowany
        - 404 – użytkownik nie istnieje
        - 200 –zostaje zwrócony adres zdjęcia profilowego
- Metoda: PATCH
    - Dane:
        - image – zdjęcie w formacie jpg, jpeg, png
    - Wymaga uwierzytelnienia tokenem
    - Response:
        - 401 – brak uprawnień / użytkownik niezalogowany
        - 404 – użytkownik nie istnieje
        - 400 – brak danych
        - 200 – zdjęcie profilowe zostało zmienione pomyślnie

2.3. Użytkownicy i interakcje pomiędzy nimi

2.3.1. Lista użytkowników

- Routing: api/user/users
- Metoda: GET
    - Parametry:
        - name (opcjonalnie)
        - surname (opcjonalnie)
        - sex (opcjonalnie)
        - location (opcjonalnie)
        - hair_length (opcjonalnie)
        - hair_color (opcjonalnie)
        - growth (opcjonalnie)
        - body_type (opcjonalnie)
        - is_smoking (opcjonalnie)
        - is_drinking_alcohol (opcjonalnie)
        - orientation (opcjonalnie)
        - eye_color (opcjonalnie)
        - age_preference_min (opcjonalnie)
        - age_preference_max (opcjonalnie)

        - Paginacja: api/user/users?page=numer_strony
        - Wymaga uwierzytelnienia tokenem

2.3.2. Informacje o użytkowniku

- Routing: api/user/users/<int:pk>
- Metoda: GET
    - Parametry zawarte w routing:
        - pk
    - Wymagane uwierzytelnienie tokenem

2.3.3. Zdjęcia użytkownika

- Routing: api/user/users/<int:pk>/images
- Metoda: GET
    - Parametry zawarte w routing:
        - pk
    - Wymagane uwierzytelnienie tokenem

2.3.4. Polubienie użytkownika

- Routing: api/user/create-like
- Metoda: POST
    - Dane:
        - value
        - pk
    - Wymagane uwierzytelnienie tokenem

2.3.5. Usunięcie polubienia

- Routing: api/user/delete-like
- Metoda: DELETE
    - Dane:
        - pk
    - Wymagane uwierzytelnienie tokenem

2.3.6. Lista użytkowników, którzy zostali polubieni przez użytkownika o podanym primary key

- Routing: api/user/get-users-are-liked/<int:pk>
- Metoda: GET
    - Parametry zawarte w routing:
        - pk
    - Wymagane uwierzytelnienie tokenem

2.3.7. Lista użytkowników, którzy polubili użytkownika o podanym primary key

- Routing: api/user/get-users-liked/<int:pk>
- Metoda: GET
    - Parametry zawarte w routing:
        - pk
    - Wymagane uwierzytelnienie tokenem

2.3.8. Lista użytkowników, którzy zostali polubieni przez zalogowanego użytkownika

- Routing: api/user/get-user-are-liked
- Metoda: GET

    - Dane:
        - pk
    - Wymagane uwierzytelnienie tokenem

2.3.9. Lista użytkowników, którzy polubili zalogowanego użytkownika

- Routing: api/user/get-user-liked
- Metoda: GET
    - Dane:
        - pk
    - Wymagane uwierzytelnienie tokenem

2.3.10. Blokowanie użytkowników

- Routing: api/user/blacklist
- Metoda: POST
    - Dane:
        - pk
    - Wymagane uwierzytelnienie tokenem
- Metoda: DELETE
    - Dane:
        - pk
    - Wymagane uwierzytelnienie tokenem
- Metoda: GET
    - Parametry:
        - pk
    - Wymagane uwierzytelnienie tokenem

2.3.11. Przyjaciele

- Routing: api/user/friendlist
- Metoda: POST
    - Dane:
        - pk
    - Wymagane uwierzytelnienie tokenem
- Metoda: DELETE
    - Dane:
        - pk
    - Wymagane uwierzytelnienie tokenem
- Metoda: GET
    - Parametry:
        - pk
    - Wymagane uwierzytelnienie tokenem
- Metoda: PATCH
    - Dane:
        - pk
    - Wymagane uwierzytelnienie tokenem

2.4. Czat

2.4.1. Informacja czy użytkownik jest obecnie aktywny

- Routing: api/chat/<str:username>/is-active
- Metoda: GET
  - Parametry zawarte w routingu:
    - username
  - Wymagane uwierzytelnienie tokenem
  - Response:
    - 401 – brak uprawnień / użytkownik niezalogowany
    - 404 – użytkownik nie istnieje
    - 200 – sukces, informacja o aktywności

2.4.2. Wysyłanie wiadomości

- Routing: api/chat/<str:username>/send-msg
- Metoda: POST
  - Dane:
    - text_message
  - Parametry zawarte w routingu:
    - username
  - Wymagane uwierzytelnienie tokenem
  - Response:
    - 401 – brak uprawnień / użytkownik niezalogowany
    - 404 – użytkownik nie istnieje
    - 200 – sukces

2.4.3. Pobranie wszystkich wiadomości pomiędzy zalogowanym użytkownikiem, a użytkownikiem o podanym username

- Routing: api/chat/<str:username>/get-all-msgs
- Metoda: GET
  - Parametry zawarte w routingu:
    - username
  - Wymagane uwierzytelnienie tokenem
  - Response:
    - 401 – brak uprawnień / użytkownik niezalogowany
    - 404 – użytkownik nie istnieje
    - 200 – sukces

2.4.4. Pobranie ostatnich x wiadomości pomiędzy zalogowanym użytkownikiem, a użytkownikiem o podanym username

- Routing: api/chat/<str:username>/get-last-x-msgs
- Metoda: GET
  - Dane:
    - x
  - Parametry zawarte w routingu:
    - username
  - Wymagane uwierzytelnienie tokenem

    - Response:
        - 401 – brak uprawnień / użytkownik niezalogowany
        - 404 – użytkownik nie istnieje
        - 200 – sukces

2.4.5. Pobranie nowszych niż podana data wiadomości pomiędzy zalogowanym użytkownikiem, a użytkownikiem o podanym username

- Routing: api/chat/<str:username>/get-new-msgs
- Metoda: GET
    - Dane:
        - last_date
    - Parametry zawarte w routingu:
        - username
    - Wymagane uwierzytelnienie tokenem
    - Response:
        - 401 – brak uprawnień / użytkownik niezalogowany
        - 404 – użytkownik nie istnieje
        - 200 – sukces

2.4.6. Pobranie wszystkich wiadomości wysłanych przez użytkownika o podanym pk

- Routing: api/chat/get-all-messages-sent-by-user-pk
- Metoda: GET
    - Wymagane uwierzytelnienie tokenem
    - Response:
        - 401 – brak uprawnień / użytkownik niezalogowany
        - 404 – użytkownik nie istnieje
        - 200 – sukces

2.4.7. Pobranie wszystkich wiadomości otrzymanych przez użytkownika o podanym pk

- Routing: api/chat/get-all-messages-received-by-user-pk
- Metoda: GET
    - Wymagane uwierzytelnienie tokenem
    - Response:
        - 401 – brak uprawnień / użytkownik niezalogowany
        - 404 – użytkownik nie istnieje
        - 200 – sukces