Architektura i komunikacja między systemami i bazami danych

Mgr inż. Krzysztof Bożek

|  |  |
| --- | --- |
| **Skład zespołu:** |  |
| **Kacper Majgier** | 49513 |
| **Paweł Pawiłojć** | 49536 |
| **Maciej Suchecki** | 47857 |

Temat:

bikeatize - Aplikacja webowa na urządzenia mobilne   
do grywalizacji jeżdżenia na rowerze   
w celu zwiększenia liczby przejechanych przez użytkowników kilometrów.

Spis treści

[**1.** **Wstęp** 3](#_Toc199567502)

[**2. Wykorzystane technologie 3**](#_Toc199567503)

[**3. Instalacja i konfiguracja 4**](#_Toc199567504)

[**4. Zrzuty ekranu 5**](#_Toc199567505)

[**5. Architektura systemu 7**](#_Toc199567506)

[8](#_Toc199567507)

[8](#_Toc199567508)

[**6.** **Testowanie funkcjonalności API poprzez interaktywną dokumentację Swagger UI** 10](#_Toc199567509)

[**Proces rejestracji użytkownika** 10](#_Toc199567510)

[**POST /api/auth/register** 10](#_Toc199567511)

[**Proces logowania się użytkownika** 11](#_Toc199567512)

[**POST /api/auth/login** 11](#_Toc199567513)

[**Odświeżanie sesji – refresh token** 12](#_Toc199567514)

[**POST /api/auth/refresh** 12](#_Toc199567515)

[**Geolocation API – Pobranie lokalizacji użytkownika** 13](#_Toc199567516)

[**POST /api/geolocation/location/update** 13](#_Toc199567517)

[**Geolocation API – Pobranie lokalizacji użytkownika** 14](#_Toc199567518)

[**GET /api/distance/distance/{user\_id}** 14](#_Toc199567519)

[**Distance – Obliczanie przebytego dystansu, Algorytm Haversine** 15](#_Toc199567520)

[**GET /api/distance/distance/{user\_id}** 15](#_Toc199567521)

[**Trips – API stworzone do rejestrowania przejazdów użytkowników** 16](#_Toc199567522)

[**POST /api/trips/start\_trip/{user\_id}** 16](#_Toc199567523)

[**Trips – Zapisanie punktu z lokalizacją użytkownika** 17](#_Toc199567524)

[**POST /api/trips/update\_location/{trip\_id}** 17](#_Toc199567525)

[**Trips – Zakończenie przykładowej trasy** 18](#_Toc199567526)

[**POST /api/trips/stop\_trip/{trip\_id}** 18](#_Toc199567527)

[**Trips – Pobranie historii tras użytkownika** 19](#_Toc199567528)

[**Trips – Pobranie informacji rankingowych** 20](#_Toc199567529)

[**Trips – Usunięcie trasy** 21](#_Toc199567530)

[**Trips – Dystans całkowity jaki przebył wybrany użytkownik** 22](#_Toc199567531)

[**CRUD – Dodanie użytkownika** 23](#_Toc199567532)

[**CRUD – Zaktualizowanie informacji o użytkowniku** 24](#_Toc199567533)

[**CRUD – Usunięcie użytkownika** 25](#_Toc199567534)

# **Wstęp**

**O aplikacji**

Bikeatize to aplikacja webowa stworzona dla entuzjastów jazdy na rowerze, pozwalająca na kompleksowe śledzenie aktywności użytkownika. Dzięki dynamicznemu systemowi punktów oraz rang, użytkownicy mogą rywalizować i zdobywać osiągnięcia, jednocześnie monitorując swoje trasy w czasie rzeczywistym.

**Link do repozytorium:** [**https://github.com/dsw49513/Bikeatize**](https://github.com/dsw49513/Bikeatize)

**Główne funkcjonalności Bikeatize**

* **Śledzenie tras i lokalizacji** – aplikacja rejestruje przejazdy rowerowe w czasie rzeczywistym.
* **System punktów** – każdy użytkownik zdobywa punkty za pokonany dystans
* **Zarządzanie użytkownikami** – pełna obsługa rejestracji i logowania.
* **Wizualizacja tras** – możliwość wyświetlania aktualnej pozycji na mapie.
* **Analiza historii przejazdów** – użytkownicy mogą sprawdzać swoje poprzednie trasy i usuwać wybrane przejazdy.

# Wykorzystane technologie

**Backend**

* FastAPI – framework do budowy szybkich i asynchronicznych aplikacji webowych w Pythonie.
* SQLAlchemy + Alembic – ORM do zarządzania bazą danych oraz system migracji schematu.
* MySQL

**Frontend**

* React – frontend jest napisany w bibliotece React
* Vite – narzędzie do budowania i uruchamiania projektu React
* Material UI – ikony i elementy interfejsu

**API i integracje**

* FastAPI Swagger UI – automatyczna dokumentacja API dostępna pod http://127.0.0.1:8000/docs.
* RESTful API – aplikacja korzysta z metod HTTP (GET, POST, PUT, DELETE) do komunikacji między klientem a serwerem.
* Openweathermap – API pogody dla danej aktualizacji
* Leaflet – mapa

**Bezpieczeństwo**

* JWT (JSON Web Token) – mechanizm uwierzytelniania użytkowników i zabezpieczania sesji.
* bcrypt – szyfrowanie haseł użytkowników przed zapisaniem ich w bazie.

# **Instalacja i konfiguracja**

**Wymagania:**

* Python 3.10+
* Node.js (frontend)
* MySQL 8 (baza danych)
* Docker (opcjonalnie do uruchomienia bazy)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Uruchomienie projektu:**

**1. Baza danych (Docker):**

docker run --name biketize-db \

-e MYSQL\_ROOT\_PASSWORD=secret \

-e MYSQL\_DATABASE=bikeatize \

-p 3306:3306 \

-d mysql:8

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Backend (FastAPI):**

cd backend

python3 -m venv venv

source venv/bin/activate

pip install -r ../requirements.txt

uvicorn main:app --reload

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**3. Frontend (React):**

cd frontend

npm install

npm run dev

# Zrzuty ekranu

Obraz zawierający tekst, zrzut ekranu, projekt graficzny, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna. Obraz zawierający tekst, rower, zrzut ekranu, logo

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna. Obraz zawierający tekst, rower, zrzut ekranu, logo

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, mapa, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna. Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna. Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

# Architektura systemu

Obraz zawierający tekst, menu, dokument, Czcionka

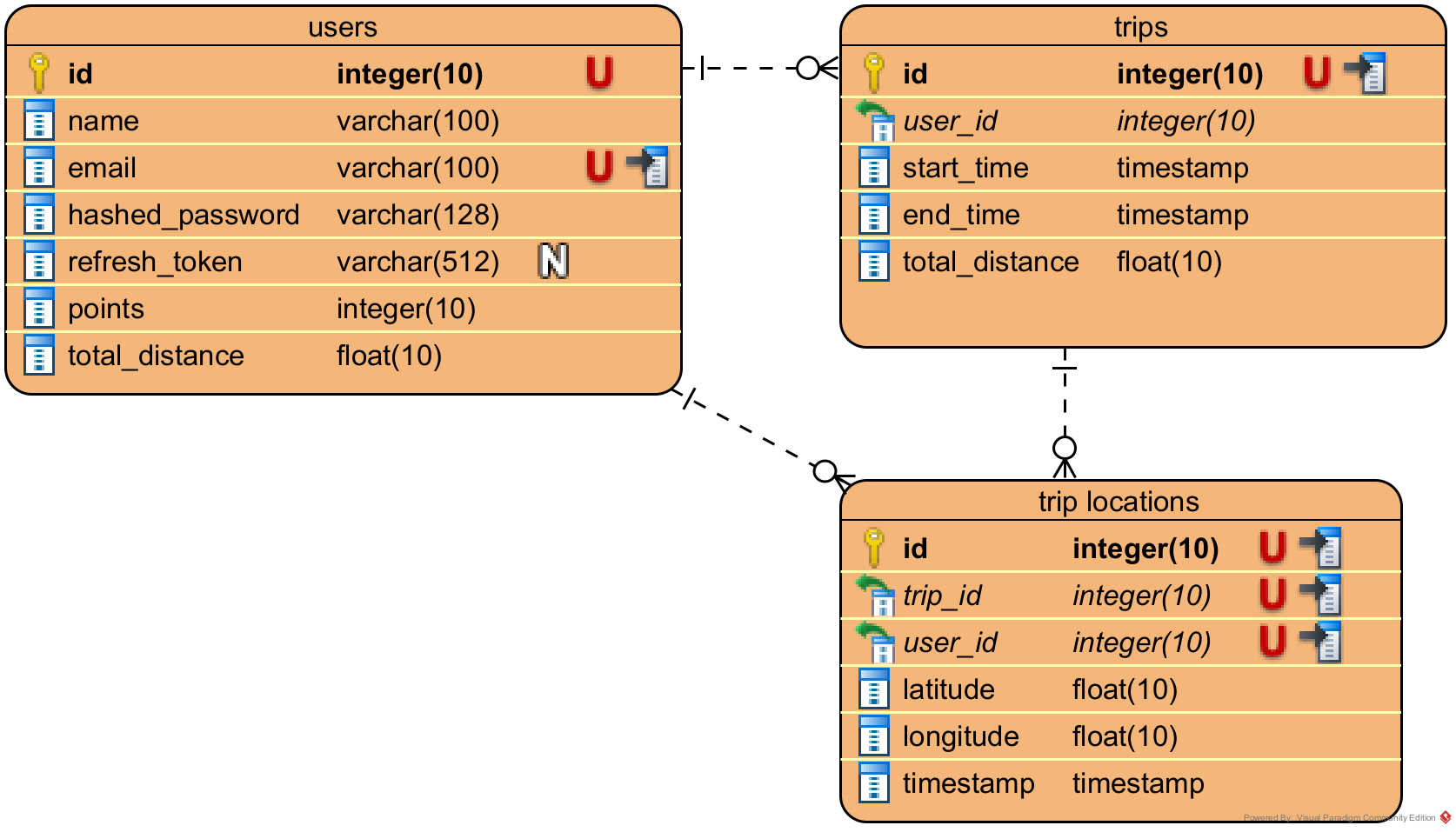
Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

. Struktura projektu

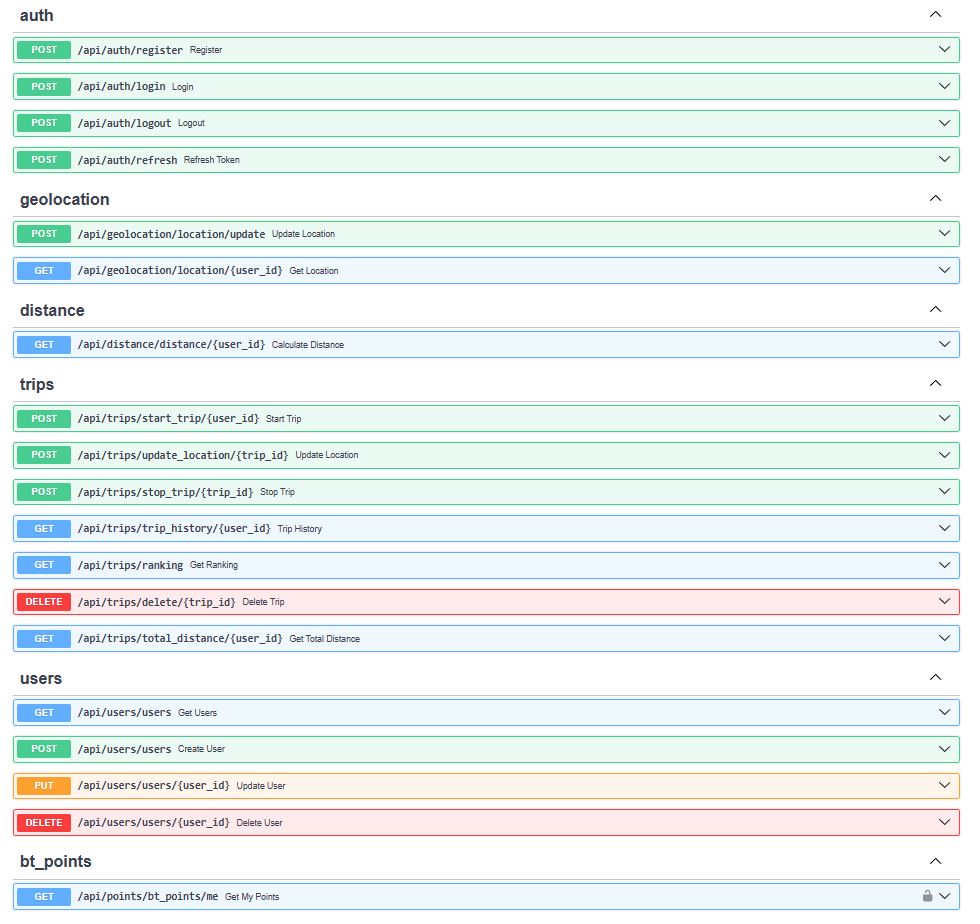
Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

. Model bazy danych stworzony w dbdiagram



3. Model bazy danych stworzony w visual paradigm

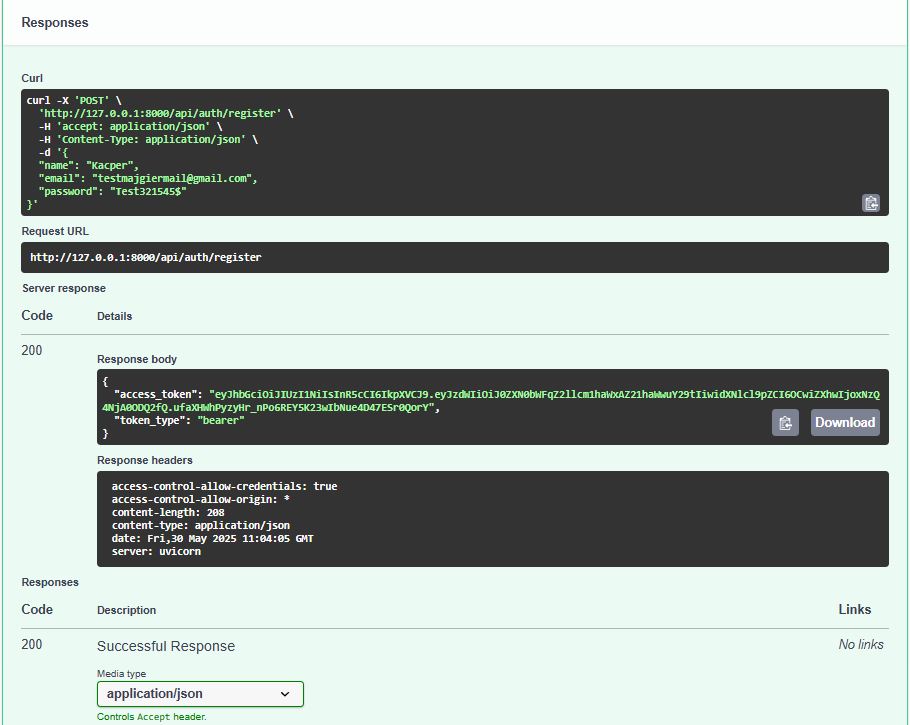


Rysunek 3 Lista endpointów

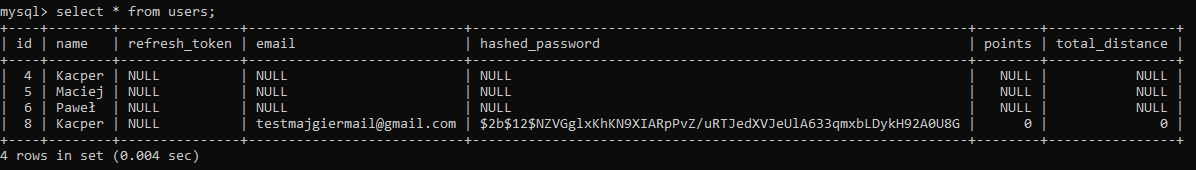
# **Testowanie funkcjonalności API poprzez interaktywną dokumentację Swagger UI**

# **Proces rejestracji użytkownika**

# **POST** [**/api/auth/register**](http://127.0.0.1:8000/docs#/auth/register_api_auth_register_post)



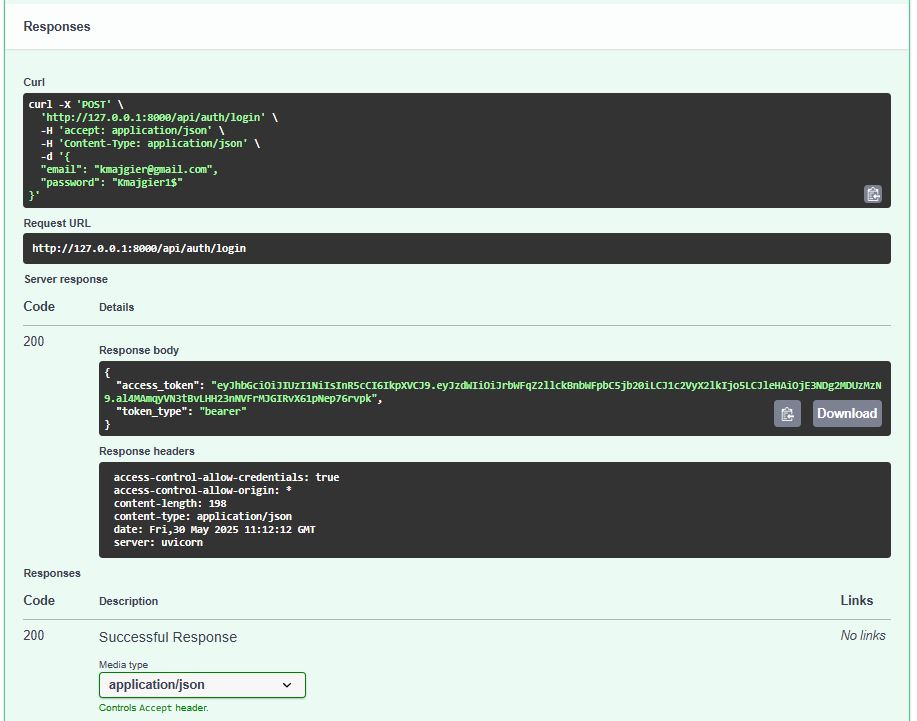
. Rejestracja użytkowników widok w Swagger UI



. Rejestracja użytkowników widok z bazy danych

# **Proces logowania się użytkownika**

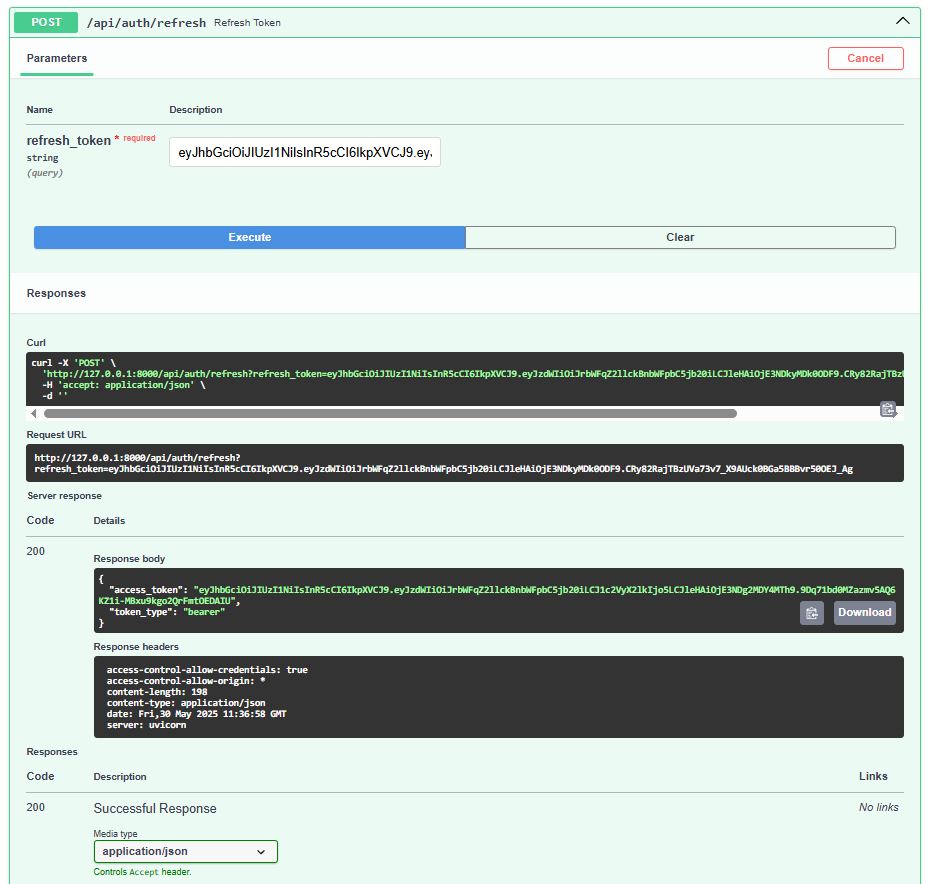
# **POST** [**/api/auth/login**](http://127.0.0.1:8000/docs#/auth/login_api_auth_login_post)



. Prawidłowe zalogowanie się użytkownika widok w Swagger UI

# **Odświeżanie sesji – refresh token**

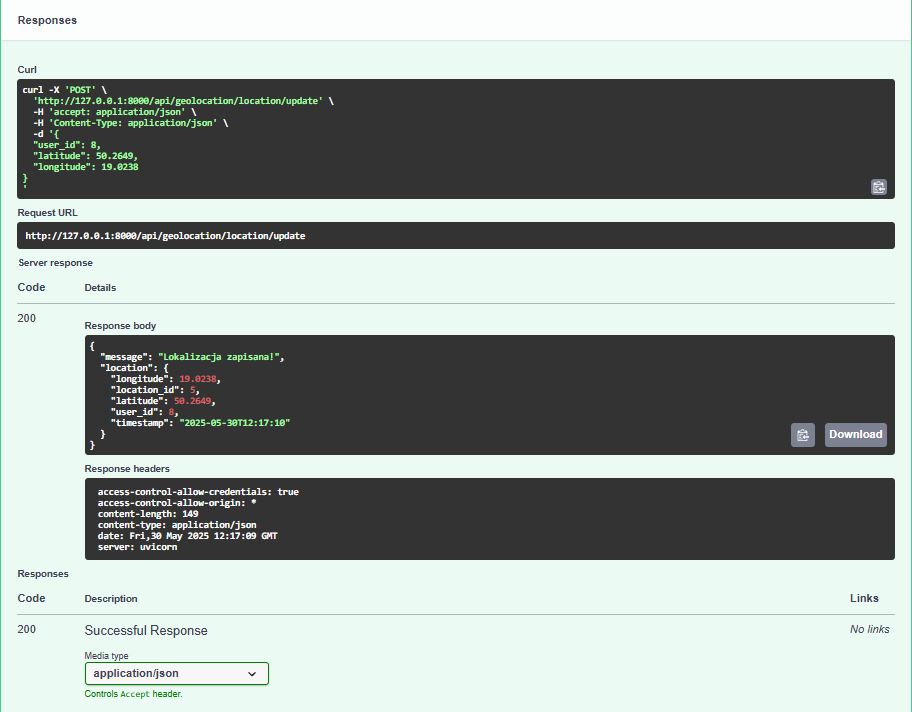
# **POST** [**/api/auth/refresh**](http://127.0.0.1:8000/docs#/auth/refresh_token_api_auth_refresh_post)



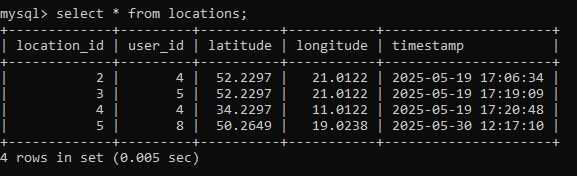
. Prawidłowe odświeżenie sesji - Refresh token, widok w Swagger UI

# **Geolocation API – Pobranie lokalizacji użytkownika**

# **POST** [**/api/geolocation/location/update**](http://127.0.0.1:8000/docs#/geolocation/update_location_api_geolocation_location_update_post)



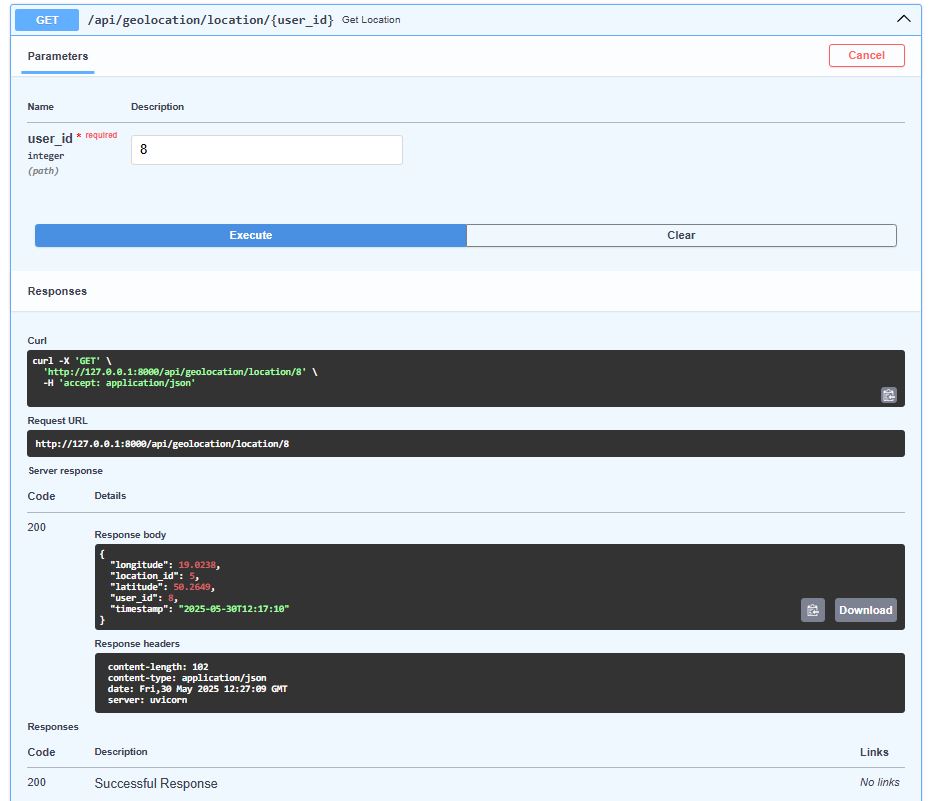
8. Pobranie lokalizacji użytkownika, widok w Swagger UI



9. Widok zapisanej lokalizacji dla użytkownika o id 8

# **Geolocation API – Pobranie lokalizacji użytkownika**

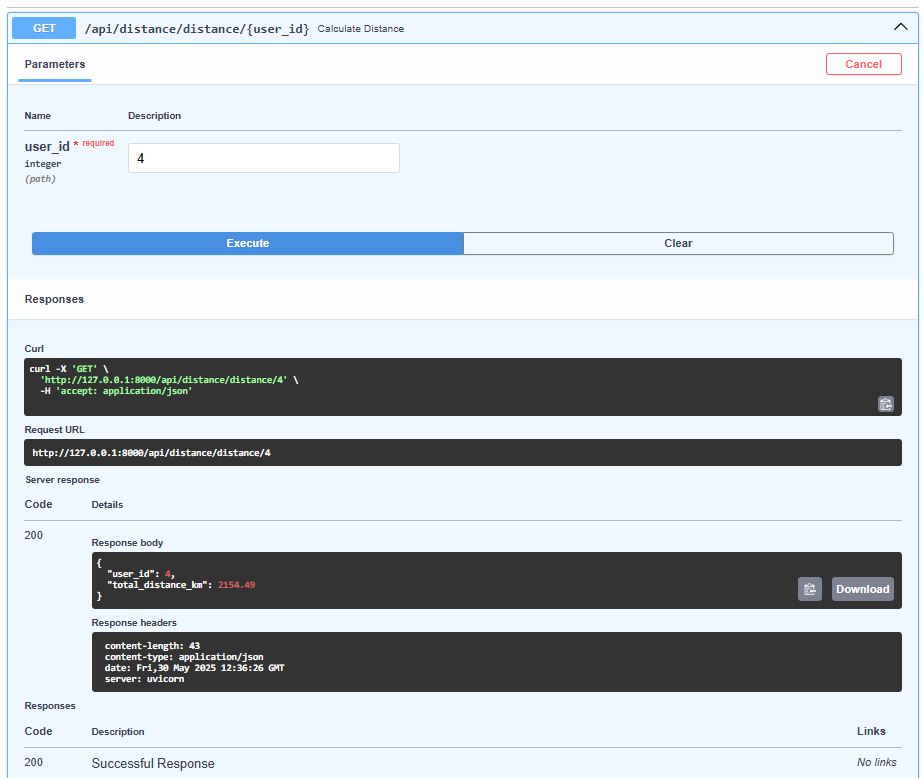
# **GET** [**/api/distance/distance/{user\_id}**](http://127.0.0.1:8000/docs#/distance/calculate_distance_api_distance_distance__user_id__get)



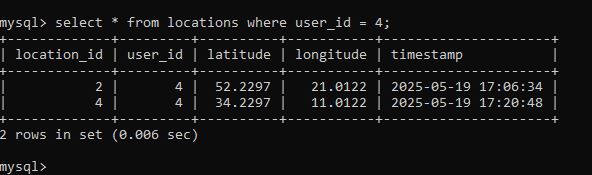
10. Pobranie aktualne lokalizacji użytkownika o id 8.

# **Distance – Obliczanie przebytego dystansu, Algorytm Haversine**

# **GET** [**/api/distance/distance/{user\_id}**](http://127.0.0.1:8000/docs#/distance/calculate_distance_api_distance_distance__user_id__get)



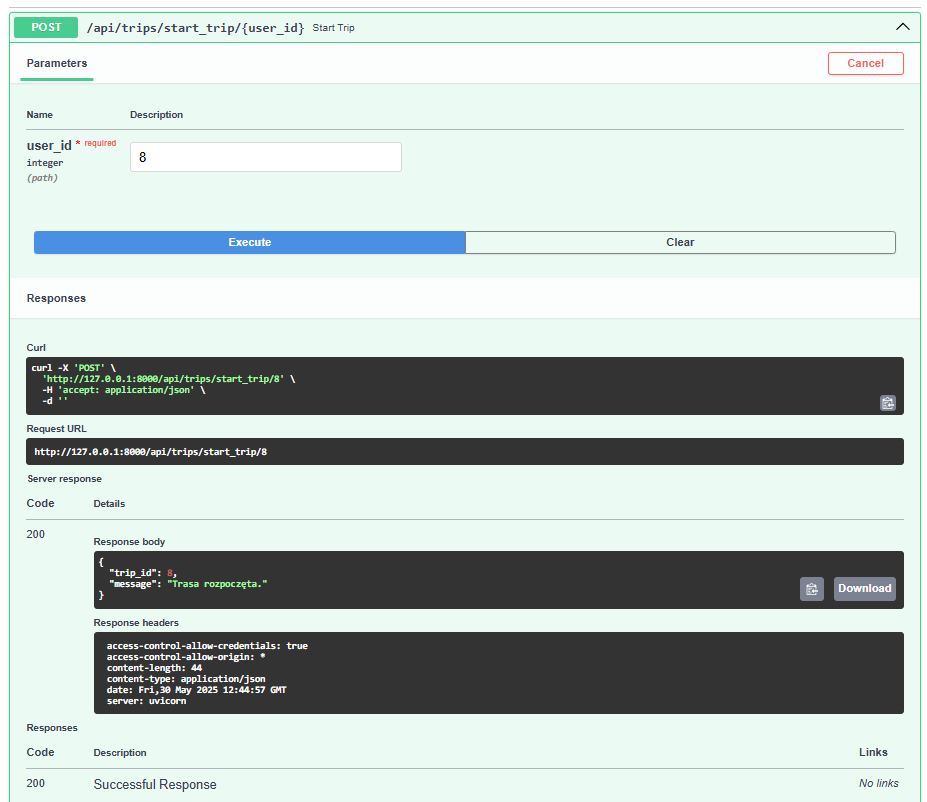
11. Obliczony dystans jaki przebył użytkownik o id 4



12. Przykładowe punkty które zostały zarejestrowane do obliczenia dystansu który przebył użytkownik

# **Trips – API stworzone do rejestrowania przejazdów użytkowników**

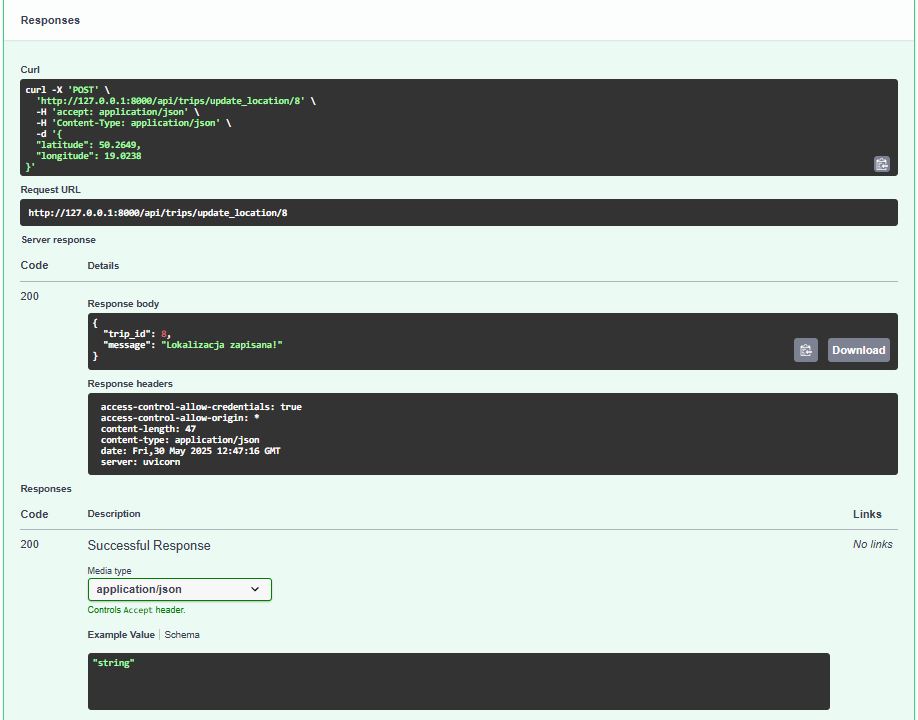
# **POST** [**/api/trips/start\_trip/{user\_id}**](http://127.0.0.1:8000/docs#/trips/start_trip_api_trips_start_trip__user_id__post)



13. Udane rozpoczęcie trasy przejazdu

# **Trips – Zapisanie punktu z lokalizacją użytkownika**

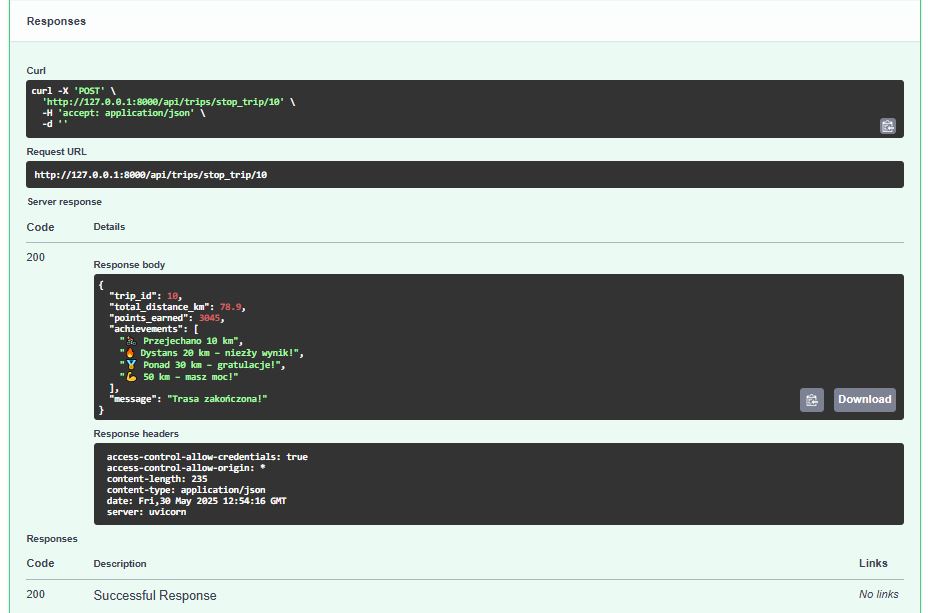
# **POST** [**/api/trips/update\_location/{trip\_id}**](http://127.0.0.1:8000/docs#/trips/update_location_api_trips_update_location__trip_id__post)



14. Aplikacja zapisuje systematycznie lokalizacje użytkownika w postaci punktów podróży, które następnie są wykorzystywane do obliczenia przebytej trasy.

# **Trips – Zakończenie przykładowej trasy**

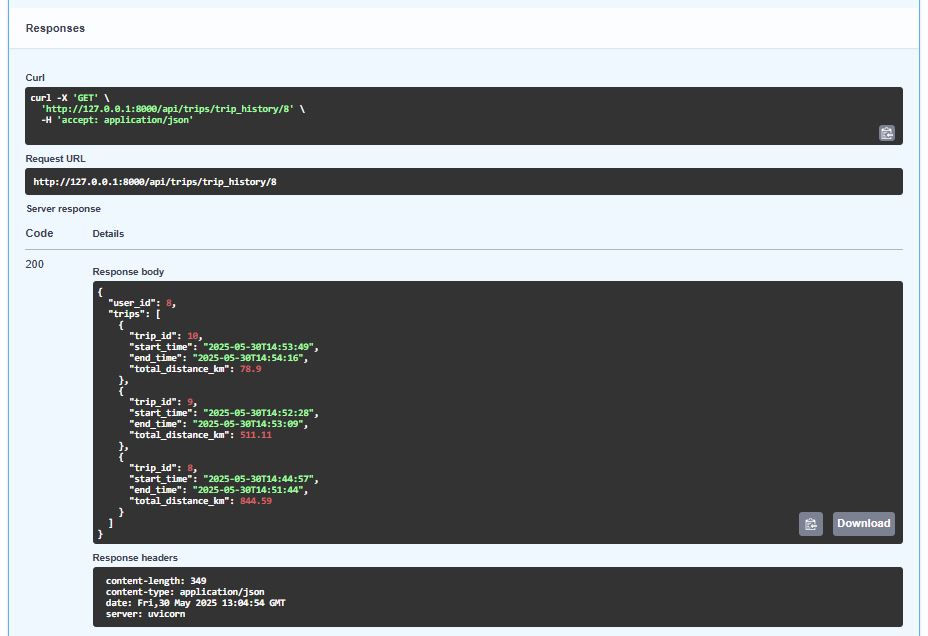
# **POST** [**/api/trips/stop\_trip/{trip\_id}**](http://127.0.0.1:8000/docs#/trips/stop_trip_api_trips_stop_trip__trip_id__post)



15. Zakończenie przykładowej trasy, obliczonej na podstawie 3 punktów: **Opole, Brzeg, Wrocław**

# **Trips – Pobranie historii tras użytkownika**

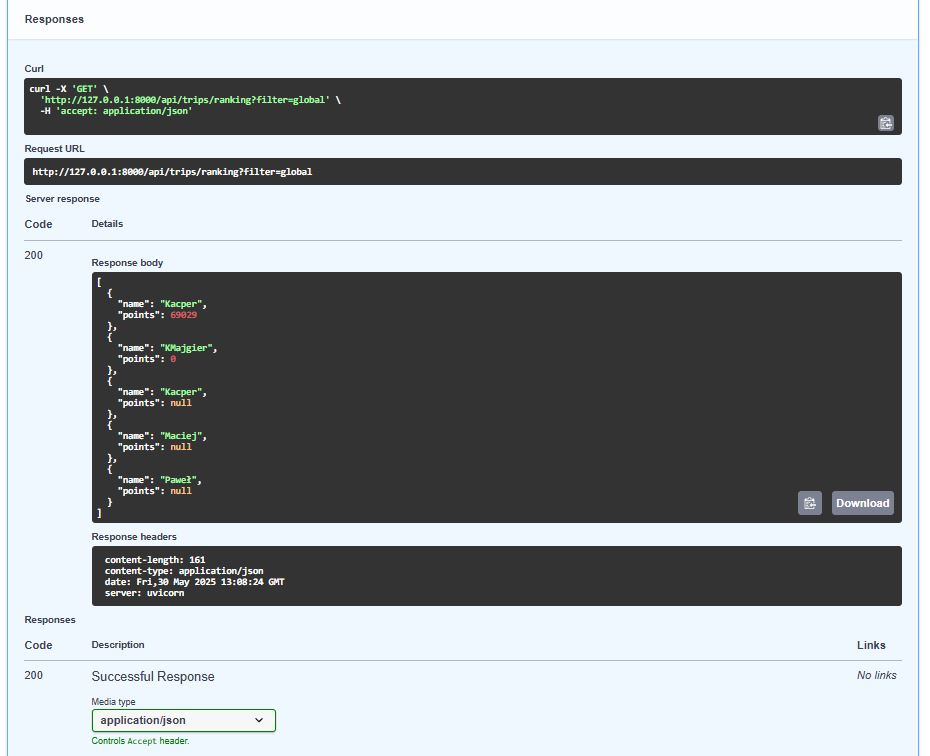
**GET** [**/api/trips/ranking**](http://127.0.0.1:8000/docs#/trips/get_ranking_api_trips_ranking_get)



16. Zakończone trasy użytkownika o id 8

# **Trips – Pobranie informacji rankingowych**

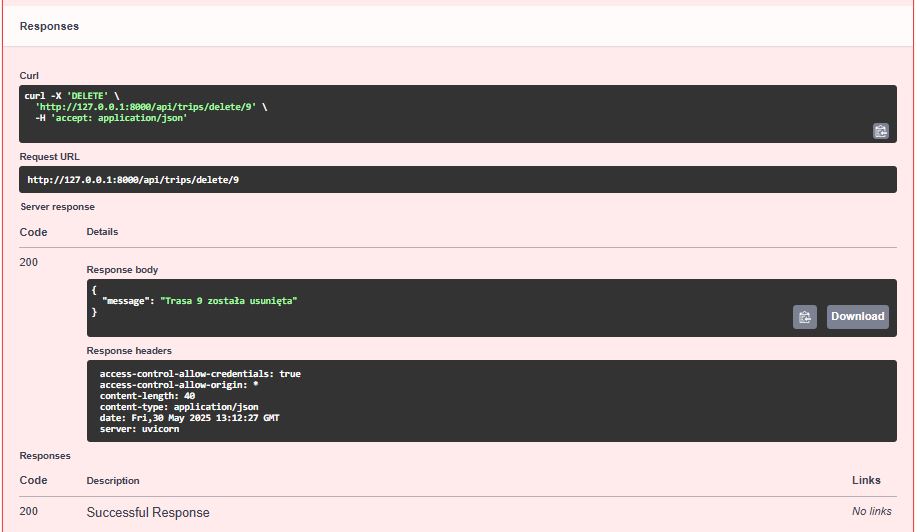
**GET** [**/api/trips/trip\_history/{user\_id}**](http://127.0.0.1:8000/docs#/trips/trip_history_api_trips_trip_history__user_id__get)



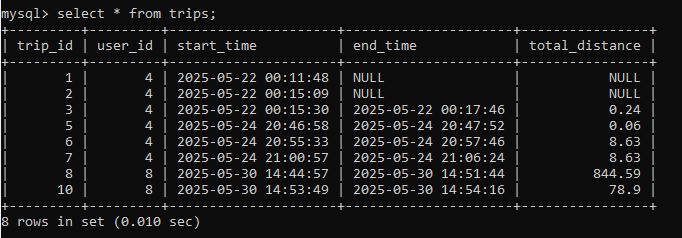
17. Lista użytkowników o najwyższej ilości punktacji za odbyte trasy

# **Trips – Usunięcie trasy**

**DELETE** [**/api/trips/delete/{trip\_id}**](http://127.0.0.1:8000/docs#/trips/delete_trip_api_trips_delete__trip_id__delete)



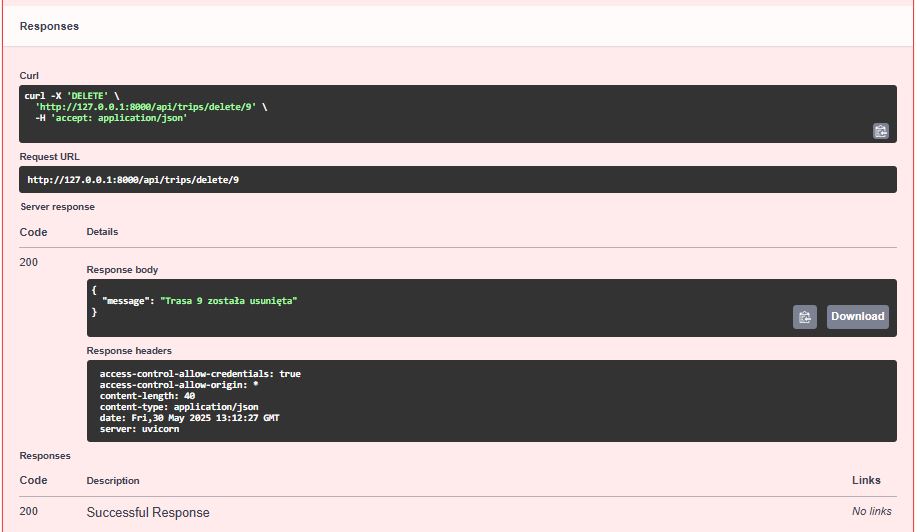
18. Usunięcie trasy id 9



19. Widok bezpośrednio z bazy danych potwierdzający usunięcie trasy 9

# **Trips – Dystans całkowity jaki przebył wybrany użytkownik**

**GET** [**/api/trips/total\_distance/{user\_id}**](http://127.0.0.1:8000/docs#/trips/get_total_distance_api_trips_total_distance__user_id__get)



18. Usunięcie trasy id 9

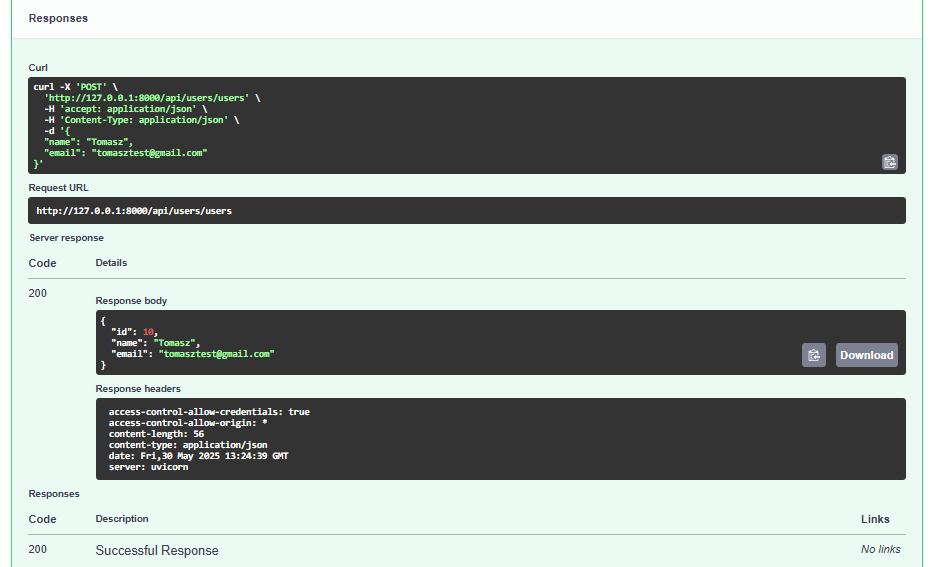
Obraz zawierający tekst, zrzut ekranu, Czcionka, czarne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

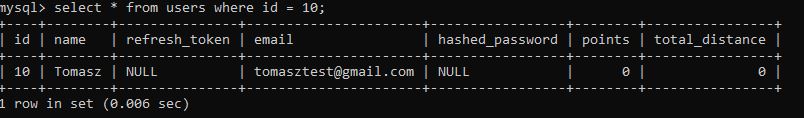
19. Widok bezpośrednio z bazy danych potwierdzający usunięcie trasy 9

# **CRUD – Dodanie użytkownika**

**POST**[**/api/users/users**](http://127.0.0.1:8000/docs#/users/create_user_api_users_users_post)



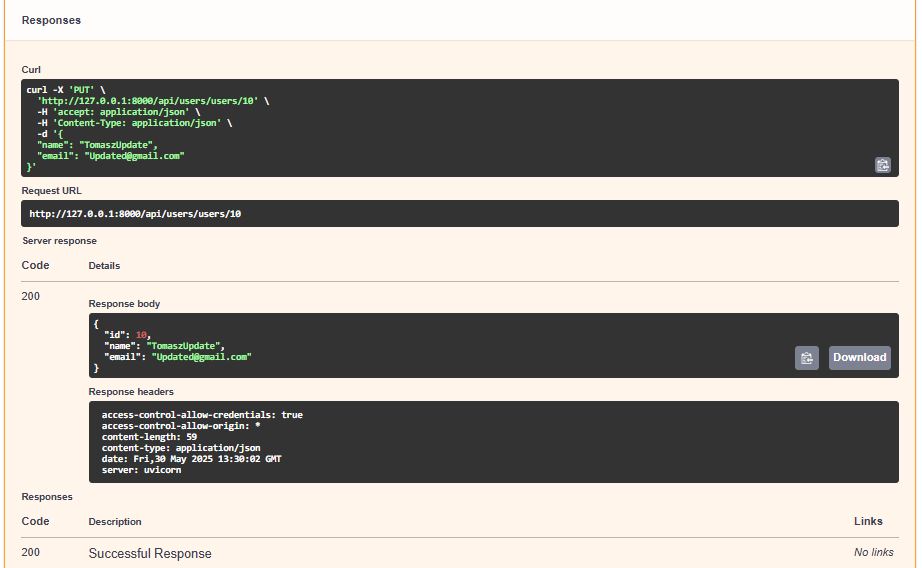
20. Dodanie użytkownika



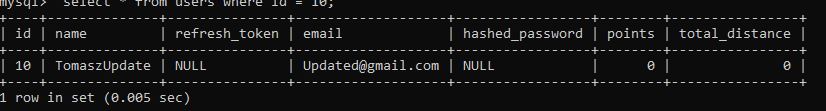
21. Widok bezpośrednio z bazy danych potwierdzający utworzenie użytkownika.

# **CRUD – Zaktualizowanie informacji o użytkowniku**

**UPDATE** [**/api/users/users/{user\_id}**](http://127.0.0.1:8000/docs#/users/update_user_api_users_users__user_id__put)



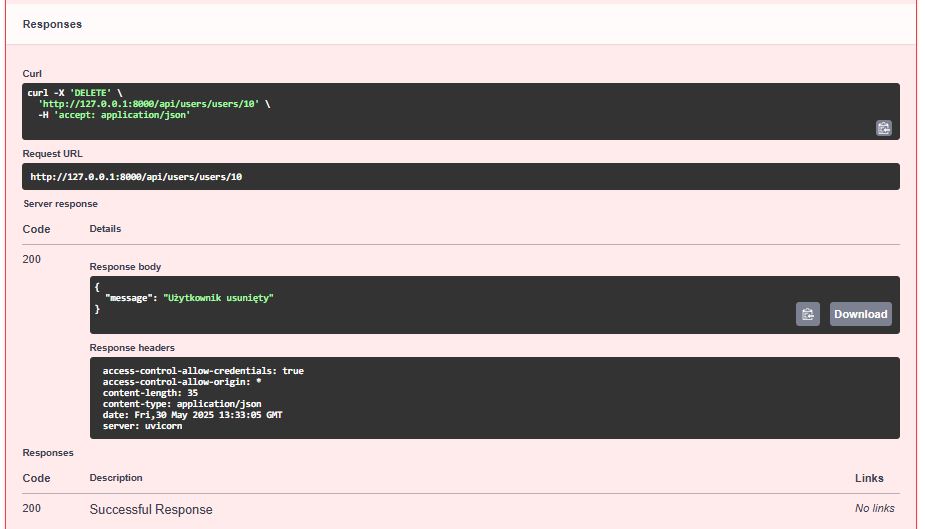
22. Dodanie użytkownika



23. Widok bezpośrednio z bazy danych potwierdzający zmianę danych użytkownika.

# **CRUD – Usunięcie użytkownika**

**DELETE** [**/api/users/users/{user\_id}**](http://127.0.0.1:8000/docs#/users/update_user_api_users_users__user_id__put)



24. Dodanie użytkownika