



Mercury

Karol Krawczykiewicz, Grzegorz Rogoziński, Jan Król, Piotr Maszczak

Październik 2023 - Styczeń 2024

1 Informacje ogólne

Nazwa projektu: Mercury

Technologie: Express, MongoDB, Neo4j, React, Redux, Sockets, TailwindCSS, TypeScript, WebRTC

Protokoły: UDP, ICE, SDP

Repozytorium GitHub: <https://github.com/Karol-2/Mercury-Project>

2 Opis projektu

Mercury jest aplikacją webową, która zapewnia komunikację tekstową oraz na żywo. Aplikacja jest dostosowana do przeglądarek internetowych na komputerach i telefonach, umożliwiając użytkownikom współpracę na różnych urządzeniach. Ponadto, system zapewnia możliwości wyszukiwania, dodawania i usuwania znajomych.

3 Opis działania

3.1 Połączenia

Aplikacja React łączy się z API backendu. Ten zaś, posiada zabezpieczone połączenie z 2 bazami danych. System składa się z 4 elementów:

1. Frontend - działa na `http://localhost:5173`
2. Backend - działa na `http://localhost:5000`
3. Baza Neo4j - przechowuje dane użytkowników, działa na porcie 7687
4. Baza MongoDB - przechowuje dane o chatach, działa na porcie 27017

API Backendu udostępnia ścieżki dotyczące autoryzacji, obsługi czatów, edycji relacji i obsługi użytkowników. Zapytania są zabezpieczone tokenami.

3.2 Uruchomienie

Aplikację można uruchomić na dwa sposoby:

1. Uruchamiając backend i frontend lokalnie, używając narzędzia npm, i łącząc z lokalnymi wersjami baz danych.
2. Używając kontenerów Docker, dzięki plikowi **compose.yml** z konfiguracją.

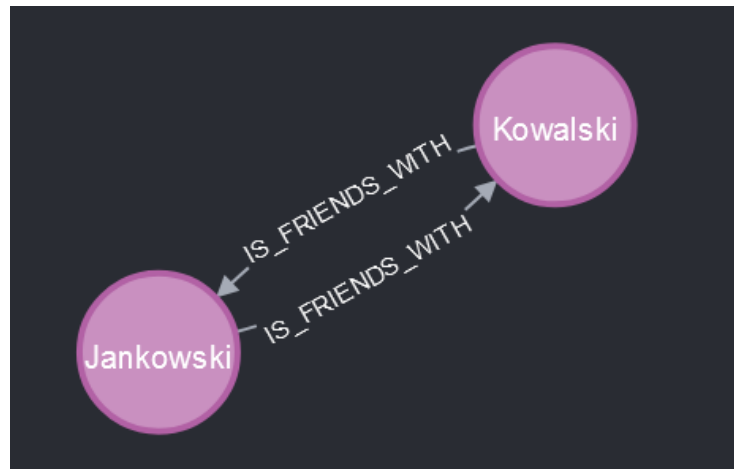
4 Modele baz danych

4.1 Model użytkownika

[illegible]

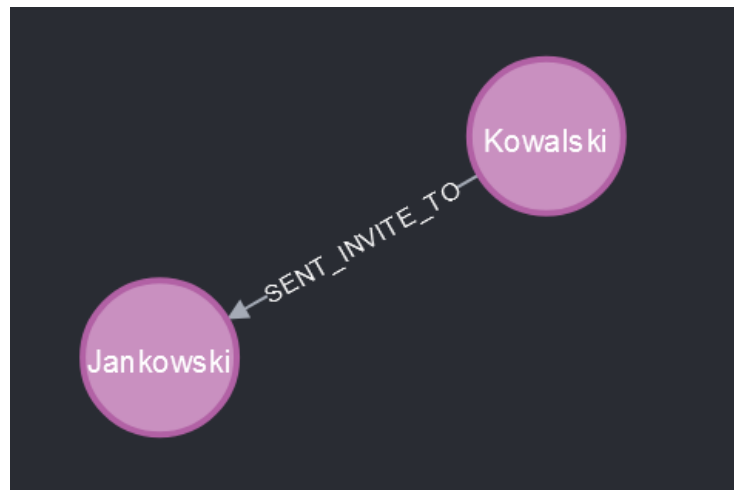
Model użytkownika składa się z ośmiu pól: **id** (identyfikator), **socketId** (identyfikator WebSocket), **first_name** (imię), **last_name** (nazwisko), **country** (dwu-literowy kod państwa), **profile_picture** (obraz zaszyfrowany base64), **mail** (adres email) i **password** (zaszyfrowane hasło).

4.2 Relacje między użytkownikami



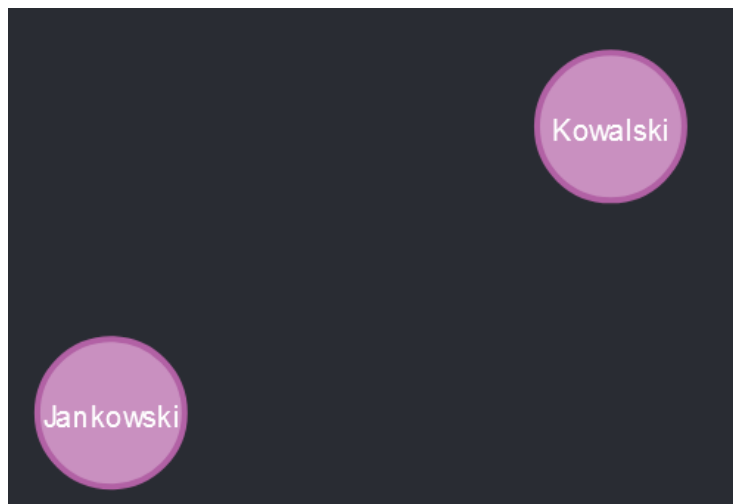
Użytkownicy z dwustronną relacją

Jeżeli dwaj użytkownicy są przyjaciółmi, istnieje między nimi relacja **IS_FRIENDS_WITH**. Obowiązuje ona w dwie strony.



Użytkownicy z jednostronną relacją

W sytuacji gdy jeden użytkownik wyśle zaproszenie do drugiego, nawiązuje się między nimi relacja **SENT_INVITE_TO**. Osoba, od której wychodzi strzałka, wysłała prośbę o dodanie do osoby, przy której jest grot strzałki. Gdy zostaje ono zaakceptowane, usuwane są dotychczasowe relacje i nadawana jest dwustronna relacja **IS_FRIENDS_WITH**.



Użytkownicy bez relacji

W przypadku anulowania prośby lub usunięcia ze znajomych, wszystkie relacje między dwoma użytkownikami zostają usunięte.

4.3 Model chatu

```
{
  _id: ObjectId('65a178a4800d927bf5b414d5'),
  authorId: '6e444ac6-4cea-4462-9134-7301445a6e28',
  receiverId: '5d16d302-ca25-498a-9865-bb9b867234b2',
  content: 'Hello World!',
  created_date: ISODate('2024-01-12T17:36:36.085Z'),
  __v: 0
}
```

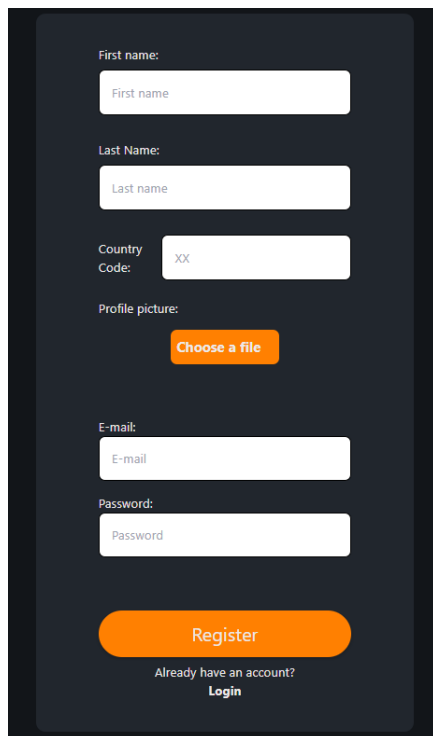
Model chatu zawiera dane o wiadomościach tekstowych przesyłanych między użytkownikami. Składają się na niego pola: **id** (id wiadomości), **authorId** (id nadawcy), **receiverId** (id odbiorcy), **content** (wiadomość), **created_date** (data wysłania).

5 Najważniejsze systemy

5.1 System rejestracji i logowania

Rejestracja nowego użytkownika odbywa się po wybraniu opcji Register na ekranie głównym, lub poprzez wejście na odpowiedni endpoint:

`http://localhost:5173/register`

A registration form with a dark background. It contains several input fields: 'First name' with a placeholder 'First name', 'Last Name' with a placeholder 'Last name', 'Country Code' with a placeholder 'XX', 'E-mail' with a placeholder 'E-mail', and 'Password' with a placeholder 'Password'. There is an orange button labeled 'Choose a file' for the profile picture. At the bottom, there is a large orange 'Register' button and a link 'Already have an account? Login'.

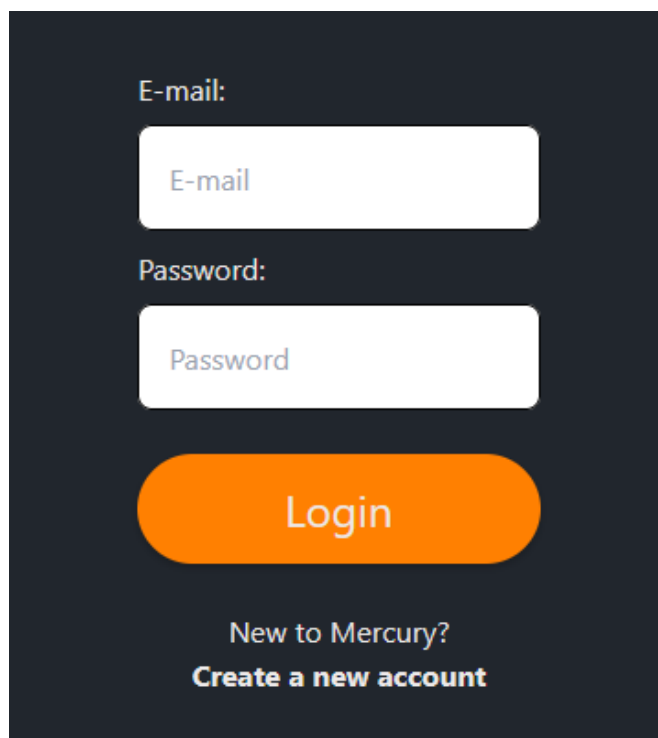
Formularz rejestracyjny

Użytkownik, aby móc się zarejestrować, musi podać swoje dane, które są w odpowiedni sposób walidowane. Reguły w formularzu rejestracji:

1. Imię i nazwisko z przynajmniej 2 znakami
2. Kod państwa o dokładnie 2 literach
3. Brak zajętego konta o tym samym mailu
4. Email o poprawnym formacie
5. Hasło z przynajmniej 8 znakami

Zdjęcie profilowe nie jest wymagane. W przypadku braku wprowadzenia własnego zdjęcia, system sam wprowadza domyślne zdjęcie. Wybór zdjęcia jest widoczny jako miniaturka, również z możliwością usunięcia.

Dane z formularza są walidowane w czasie rzeczywistym przy użyciu biblioteki Zod i React hooka useForm. Po wypełnieniu formularza i wciśnięciu przycisku 'Register' przesłany obrazek (lub w przypadku braku, to obrazek domyślny) jest przekodowywany na format base64. Hasło jest szyfrowane za pomocą biblioteki Bcrypt. Następnie sprawdzane jest, czy podany maila z formularza nie jest zajęty w systemie. Jeśli jest zajęty, to wyświetlony jest komunikat z tym związany, formularz nie jest czyszczony w celu poprawienia maila. W przypadku poprawności danych, system zapisuje nowego użytkownika oraz następuje zmiana strony na logowanie na endpointcie: `http://localhost:5173/login`



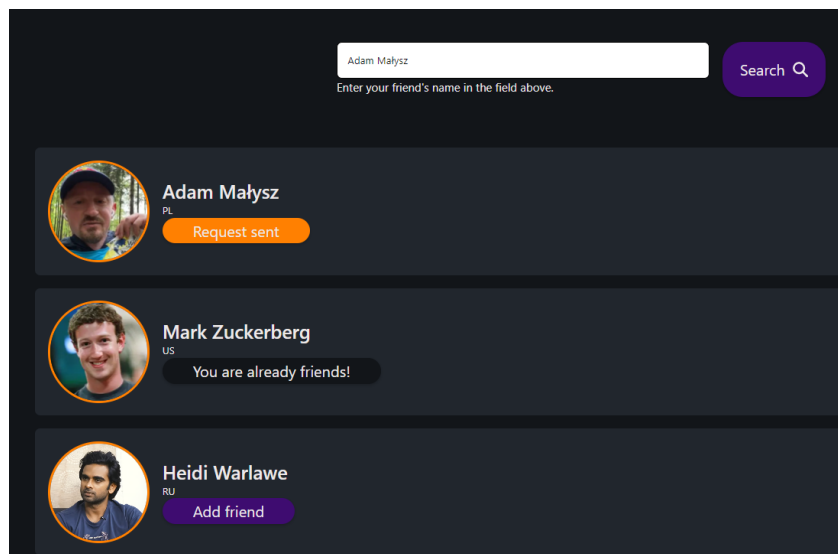
The image shows a login form on a dark background. It contains two input fields: 'E-mail' and 'Password', both with placeholder text. Below the fields is an orange 'Login' button. At the bottom, there is a link 'New to Mercury? Create a new account'.

Formularz logowania

Formularz logowania wymaga podania maila oraz hasła. Po poprawnym podaniu danych jest tworzona sesja z tokenem przy użyciu JWT oraz ciasteczek z js-cookie. Strona zmienia się na endpoint `http://localhost:5173/profile` do którego ma się tylko dostęp podczas sesji. Zawiera on wszystkie dane, które zostały wcześniej podane w rejestracji.

5.2 System szukania znajomych

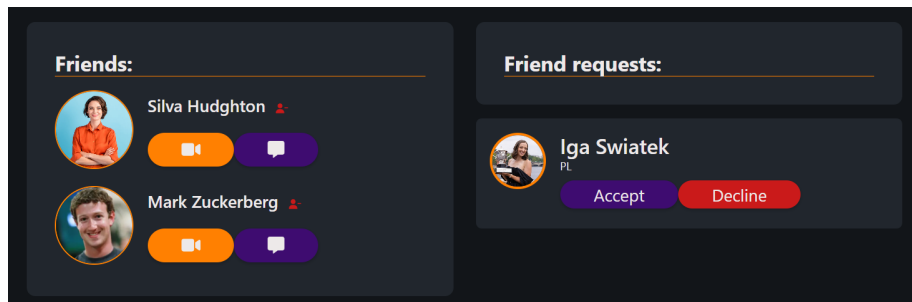
Wyszukiwanie znajomych odbywa się na endpointcie `http://localhost:5173/search`. Po wpisaniu w pasek wyszukiwania danej frazy i zatwierdzeniu jest pokazywane 10 pierwszych wyników, które są najbardziej zbliżone do tej frazy na podstawie imienia i nazwiska. W celu wyszukiwania podobieństwa jest wykorzystywany własny algorytm oparty o algorytmie Odległości Levenshteina. Każda osoba na liście zawiera przycisk informujący relację z danym kontaktem.



Okno wyszukiwania znajomych

5.3 System dodawania do znajomych

Po wybraniu kontaktu, który ma zostać dodany do sieci znajomych, wysyłany jest Request i tworzona relacja **SENT_INVITE_TO** w bazie Neo4j. Przychodzące zaproszenia i znajomych można sprawdzić w zakładce *Friends* lub na endpointcie `http://localhost:5173/friends`



Ekran listy znajomych i przychodzących zaproszeń

Zaproszenia do znajomych są wczytywane cały czas na żywo, poprzez wysyłanie requesta do bazy danych o sprawdzeniu relacjach z zalogowanym użytkownikiem.

5.4 System wideo rozmowy

Użycie WebRTC (*Web Real-Time Communication*) pozwala nawiązać połączenie typu *peer to peer* między dwoma użytkownikami i wymieniać się na bieżąco sygnałem audio i wideo. To rozwiązanie nie potrzebuje serwera, co zmniejsza opóźnienie między użytkownikami. WebRTC transportuje dane z użyciem protokołów UDP, który jest ceniony za swoją szybkość

Schemat komunikacji

1. Gdy Użytkownik1 chce połączyć się z Użytkownikiem2, wysyła on odpowiednią wiadomość o chęci dołączenia.
2. Użytkownik2 chcąc połączyć się, akceptuje 'ofertę' i wysyła Użytkownikowi1 swoje informacje.
3. Po tym, gdy użytkownicy wymieniają się danymi, nawiązuje się połączenie, każdy użytkownik zna SDP drugiego.
4. Używając metody ICE, każdy użytkownik uderza do *stun server* by uzyskać swój publiczny adres IP.
5. Gdy *stun server* odpowie, odebrane dane są poprzez użytkownika transportowane do drugiego. Tak samo działa to u drugiego użytkownika.

6. Kiedy dane znajdą wspólną drogę komunikacji, połączenie jest już gotowe i mogą być przesyłane informacje w obie strony.

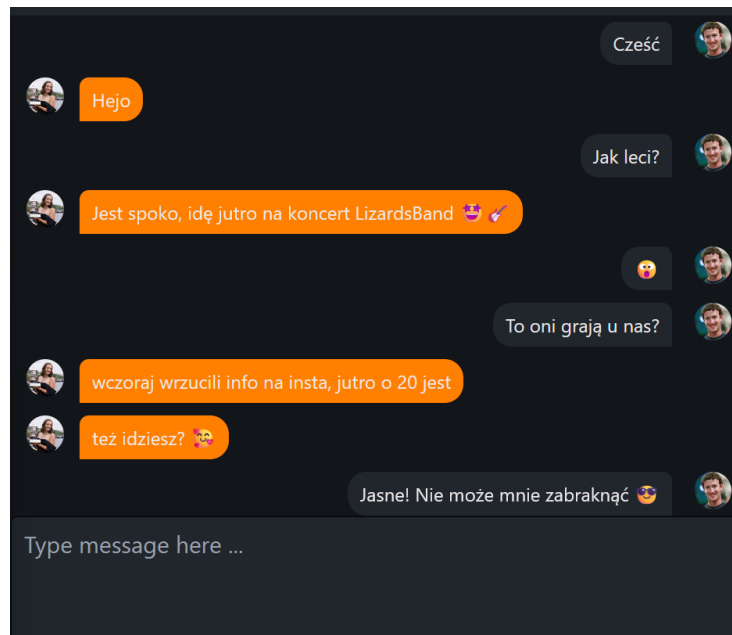
Wysyłane jest organizowane za pomocą procesu o nazwie ICE (*Interactive Connectivity Establishment*) signaling.

W danych, które są zawierane podczas dołączenia do rozmowy między użytkownikami, znajduje się protokół SDP (*Session Description Protocol*), który zawiera informacje typu: kodek, adres, typ nośnika, dane audio, dane wideo. Użytkownicy również wymieniają się '*ICE candidates*', którym jest publiczny adres IP i port, który przyjmuje dostarczane dane.

5.5 System chatu

Chat ze znajomymi jest dostępny poprzez wybranie zielonego przycisku chatu obok znajomego w liście dostępnych znajomych. Nowy chat jest tworzony wraz z przypisanym id. Aby wysłać wiadomość, należy wcisnąć Enter.

Wiadomości wysyłane są w czasie rzeczywistym, a cała konwersacja pomiędzy użytkownikami jest zapisywana do bazy MongoDB. Do komunikacji wykorzystywane są gniazda WebSocket. Historia wiadomości jest dostępna pod endpointem: `/chat/userId/friendId`



Przykładowy ekran chatu

6 Źródła

- Dokumentacja Framer-motion: <https://www.framer.com/motion/>
- Dokumentacja Zod: <https://zod.dev/>
- Dokumentacja Bcrypt: <https://www.npmjs.com/package/bcrypt>
- Dokumentacja Express: <https://expressjs.com/en/api.html>
- Artykuł Praca Zespołowa na Github:
<https://www.freecodecamp.org/news/how-to-use-git-and-github-in-a-team-like-a-pro/>
- Kurs WebRTC React:
<https://www.udemy.com/course/mastering-webrtc-part-2-real-time-video-and-screen-share/>
- Kurs WebRTC: <https://www.youtube.com/watch?v=QsH8FL0952k>