



**Politechnika
Śląska**

PROJEKT INŻYNIERSKI

Aplikacja webowa pomagająca nawiązywanie nowych znajomości

Filip MIERA

Nr albumu: 295710

Kierunek: Informatyka

Specjalność: Grafika Komputerowa i Oprogramowanie

PROWADZĄCY PRACĘ

Dr hab. inż. Adam Domański

KATEDRA Systemów Rozproszonych i Urządzeń Informatyki

Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2024

Tytuł pracy

Aplikacja webowa pomagająca nawiązywanie nowych znajomości

Streszczenie

Celem pracy jest utworzenie i wdrożenie aplikacji internetowej, która umożliwia nawiązywanie nowych znajomości oraz komunikację między użytkownikami. Aplikacja powinna umożliwiać użytkownikowi utworzenie swojego profilu, dodawanie zdjęć i opisu oraz przeglądanie profili innych użytkowników. Dodatkowo, użytkownik powinien mieć możliwość dostosowania preferencji dotyczących odkrywania profili innych użytkowników, takich jak maksymalna odległość oraz zakres wieku

Słowa kluczowe

znajomości, komunikacja, preferencje

Thesis title

Web application that helps you make new social connections

Abstract

The work aims to create and implement a web application that enables making new friends and communicating between users. The application should allow users to create profiles, add photos and descriptions, and view other users' profiles. Additionally, the user should be able to adjust preferences for discovering other users' profiles, such as maximum distance and age range

Key words

connections, communication, preferences

Spis treści

1 Wstęp	1
1.1 Wprowadzenie w problem	1
1.2 Problem w dziedzinie	1
1.3 Cel pracy	2
1.4 Zakres pracy	2
1.5 Krótka charakterystyka rozdziałów	2
2 Analiza tematu	5
2.1 Sformułowanie problemu	5
2.2 Osadzenie tematu w kontekście aktualnego stanu wiedzy	5
2.3 Opis znanych rozwiązań	5
3 Wymagania i narzędzia	9
3.1 Wymagania funkcjonalne	9
3.2 Wymagania niefunkcjonalne	10
3.3 Opis narzędzi	10
4 Specyfikacja zewnętrzna	13
4.1 Wymagania sprzętowe i programowe aplikacji	13
4.2 Sposób obsługi	14
4.3 Przykład działania	16
5 Specyfikacja wewnętrzna	29
5.1 Architektura	29
5.2 Ważniejsze klasy	30
5.2.1 Frontend	30
5.2.2 Backend	32
5.3 Ważniejsze algorytmy	33
5.4 Baza danych	34
5.5 Przypadki użycia	36

6 Weryfikacja i walidacja	39
6.1 Sposób testowania	39
6.2 Wykryte i usunięte błędy	39
7 Podsumowanie i wnioski	41
7.1 Kierunki dalszej rozbudowy	41
7.1.1 Zaawansowany algorytm dopasowywania	41
7.1.2 Odblokowywanie zablokowanych użytkowników	41
7.1.3 System płatnych usług	42
7.1.4 Bezpieczeństwo	42
7.2 Napotkane problemy	43
7.2.1 Problemy przy tworzeniu frontendu	43
7.2.2 Problemy przy tworzeniu backendu	43
Bibliografia	46
Spis skrótów i symboli	47
Źródła	51
Spis rysункów	62

Rozdział 1

Wstęp

1.1 Wprowadzenie w problem

Środowisko cyfrowe stało się integralną częścią życia wielu ludzi, zwłaszcza młodszych pokoleń. Coraz więcej aspektów naszego codziennego życia przenosi się do internetu, co może wpływać na to, jak się komunikujemy, pracujemy lub uczymy. Młodzi ludzie często czują się bardziej komfortowo w internecie, gdzie mogą łatwo nawiązywać kontakty, dzielić się zainteresowaniami i tworzyć społeczności.

Jednakże długotrwała izolacja społeczna, wynikająca na przykład z pandemii, może prowadzić do poważnych problemów zdrowia psychicznego. Brak bezpośrednich kontaktów z innymi ludźmi może sprzyjać uczuciu samotności, a to z kolei może prowadzić do lęku i depresji. Zostało to opisane w tym artykule [1].

Zmiana miejsca zamieszkania stanowi dodatkowe wyzwania, zwłaszcza jeśli chodzi o nawiązywanie nowych znajomości. Przeniesienie się do zupełnie nowego otoczenia, związane jest z koniecznością dostosowania się do nowej społeczności, co może być trudne i czasochłonne. Proces budowania relacji poza światem wirtualnym staje się wyzwaniem, które wymaga nie tylko czasu, ale także odwagi do otwarcia się na nowe doświadczenia społeczne.

1.2 Problem w dziedzinie

W dzisiejszym dynamicznym świecie ludziom często trudno jest poznać nowych ludzi poza ich obecnymi kręgami społecznymi. Ten problem wynika z różnych czynników, a jego konsekwencje mogą obejmować ograniczone możliwości poszerzania sieci kontaktów i nawiązywania różnorodnych przyjaźni.

Istniejące platformy mediów społecznościowych mogą priorytetowo traktować istniejące przyjaźnie, ta aplikacja nakłania użytkowników do poznawania nowych osób i zawiązywania nowych znajomości przez internet, co zostało przedstawione w

pozytywnym świetle w tym artykule [2].

Ludzie często mają różne zainteresowania i preferencje dotyczące kontaktów towarzyskich. Istniejące platformy mogą nie zaspokajać różnorodnych potrzeb osób poszukujących różnego rodzaju przyjaźni lub interakcji społecznych.

1.3 Cel pracy

Celem pracy jest opracowanie i wdrożenie aplikacji internetowej, mającej na celu ułatwienie nawiązywania nowych znajomości oraz umożliwienie komunikacji między użytkownikami. Użytkownik będzie miał możliwość zobaczenia innych użytkowników, dostosowując filtry według płci, wieku i odległości. Prezentowane profile będą zawierać informacje dotyczące zainteresowań oraz preferencji. Ta aplikacja może przyczynić się do poprawy samopoczucia użytkowników, ułatwiając nawiązywanie nowych przyjaźni w dobrze im znanej przestrzeni internetowej, nawet jeśli jest się tysiące kilometrów od poznawanej osoby.

Może również umożliwić rozwój społeczności internetowych, w których osoby o wspólnych zainteresowaniach i celach mogą znaleźć przyjaciół o podobnych poglądach, co sprzyja poczuciu przynależności i koleżeństwa.

1.4 Zakres pracy

Użytkownik ma duże możliwości opisania prawdziwego siebie przy tworzeniu lub edycji swojego profilu, aplikacja umożliwia na przykład wybranie swoich zainteresowań, preferencji względem posiadania dzieci lub zwierząt, chodzenia na siłownię oraz utrzymywania diety. Aspekt wizualny strony, był bazowany na najpopularniejszej aplikacji z tej dziedziny: 'Tinder'. Wygląd ten jest bardzo schludny, przyjemny dla oka oraz bardzo intuicyjny.

1.5 Krótka charakterystyka rozdziałów

Wstęp: Rozdział otwierający pracę, w którym przedstawione zostały główne założenia i cele aplikacji służącej do nawiązywania nowych znajomości. Wskazane jest również wprowadzenie do istoty problematyki związanej z tym obszarem.

Analiza tematu: Został dokonany przegląd istniejących rozwiązań związanych z nawiązywaniem znajomości dostępnych na rynku.

Wymagania i narzędzia: Ten rozdział koncentruje się na określeniu wymagań funkcjonalnych i niefunkcjonalnych aplikacji. Zawarte są także informacje dotyczące narzędzi, które zostały wykorzystane w procesie tworzenia.

Specyfikacja zewnętrzna: Zaprezentowano zewnętrzne aspekty aplikacji, skupiające się na interakcjach z użytkownikami, interfejsie użytkownika i ogólnym doświadczeniu korzystania z aplikacji.

Specyfikacja wewnętrzna: Rozdział poświęcony jest wewnętrznym detalem aplikacji, obejmującym strukturę bazy danych, algorytmy oraz inne kluczowe elementy projektu.

Weryfikacja i validacja: Przedstawione zostały metody testowania oraz opisano wykryte błędy.

Podsumowanie i wnioski: W zakończeniu pracy opisano potencjalne kierunki rozwoju projektu oraz napotkane problemy podczas procesu tworzenia.

Rozdział 2

Analiza tematu

2.1 Sformułowanie problemu

W ostatnich dziesięcioleciach nastąpił dynamiczny rozwój komunikacji cyfrowej, co znaczaco wpłyneło na nasze życie społeczne. Wiele osób boryka się z trudnościami w nawiązywaniu nowych znajomości poza swoimi ustalonymi kręgami społecznymi, co może wpływać negatywnie na stan psychiczny. Celem tej pracy inżynierskiej jest stworzenie platformy ułatwiającej nawiązywanie trwałych przyjaźni lub nawet znalezienie potencjalnych partnerów życiowych.

2.2 Osadzenie tematu w kontekście aktualnego stanu wiedzy

Platformy mediów społecznościowych, pomimo umożliwiania łatwego nawiązywania kontaktów online, często nie są w stanie zachować prawdziwych relacji. Znane są również dobrze dokumentowane przypadki wzrostu poczucia osamotnienia. Niniejsza praca opiera się na tej wiedzy i ma za zadanie wykorzystanie technologii w celu stworzenia rozwiązania, które skonfrontuje się z obecnymi wyzwaniami budowania autentycznych połączeń w erze cyfrowej.

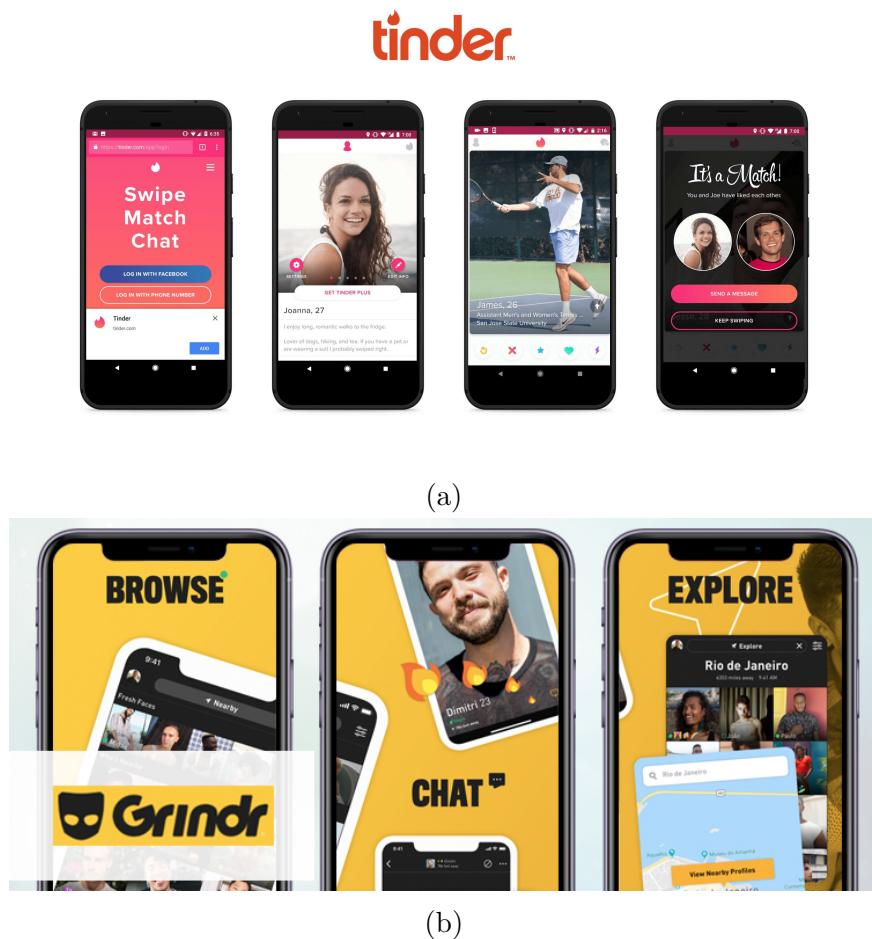
2.3 Opis znanych rozwiązań

Obecne możliwości nawiązywania nowych znajomości koncentrują się głównie na platformach społecznościowych i aplikacjach randkowych. Mimo że te platformy łączą jednostki na podstawie różnorodne kryteria, często brakuje im swobody w wyrażaniu prawdziwej tożsamości. Projektowana aplikacja internetowa ma na celu wyróżnienie się poprzez zapewnienie użytkownikom pełnej swobody w korzystaniu z niej, bez żadnych ograniczeń czy moderacji działań. Nikt nie będzie kontrolował, co dodajesz, piszesz,

ani nie będzie podejmować działań karzących za wyrażanie swoich przekonań czy opisywanie siebie. Oto opis kilku podobnych istniejących aplikacji:

1. Tinder: - Tinder ukazany na rysunku 2.1a, nazywany często aplikacją randkową, zdobył popularność dzięki swojemu innowacyjnemu podejściu do poznawania nowych ludzi. Działa na zasadzie przeglądania zdjęć użytkowników, gdzie można przeciągać w prawo, aby wyrazić zainteresowanie, lub w lewo, aby odrzucić daną osobę. Jeśli dwie osoby wyrażą wzajemne zainteresowanie, tworzy się "match", umożliwiający im rozpoczęcie rozmowy. - Oprócz zdjęć, użytkownicy mogą dodawać krótki opis i preferencje związane z wiekiem, lokalizacją i zainteresowaniemi. - Algorytmy Tinder analizują dane, aby proponować użytkownikom osoby, które mogą być dla nich interesujące.

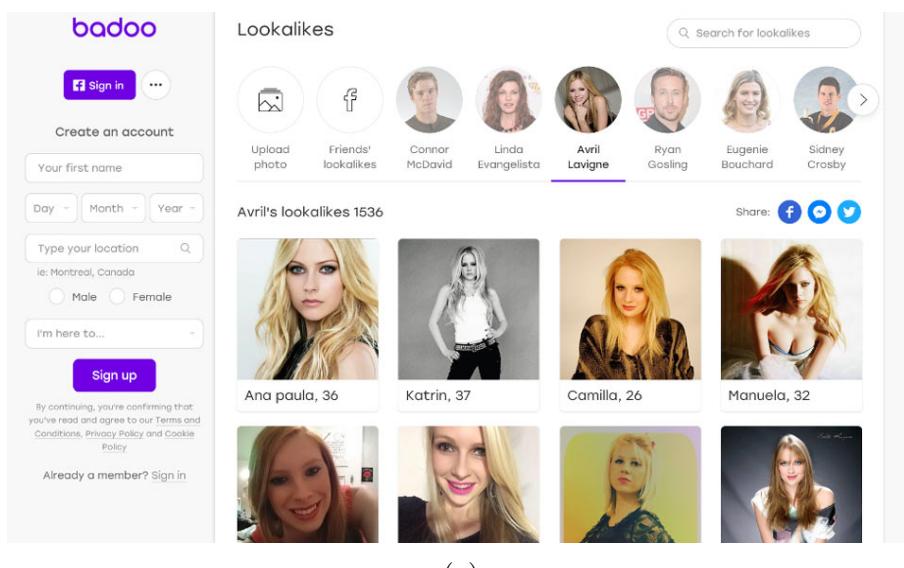
2. Grindr: - Grindr pokazany na obrazku 2.1b jest specjalnie stworzoną aplikacją dla społeczności LGBT. Skoncentrowana na geolokalizacji, umożliwia użytkownikom nawiązywanie kontaktów zarówno romantycznych, jak i przyjacielskich. - Oprócz informacji o orientacji seksualnej, profil zawiera dodatkowe elementy, takie jak preferencje związane z rolą w związku. - Geolokalizacja pozwala użytkownikom na spotkanie osób w ich okolicy. - Opcje filtrów pozwalają spersonalizować wyszukiwanie.



Rysunek 2.1: a. Tinder. [3], b. Grindr. [4]

3. Badoo: - Badoo ukazano na grafice 2.2a, łączy funkcje społecznościowe i randkowe, oferując użytkownikom możliwość budowania zarówno romantycznych, jak i przyjacielskich relacji. - Profile zawierają zdjęcia, krótki opis, a także informacje o zainteresowaniach i preferencjach. - Funkcja "Zbliżający się" informuje o użytkownikach w okolicy. - Badoo oferuje także opcję gier, takich jak "Gra na randki", co dodaje element zabawy.

4. Hinge: - Widoczna na zdjęciu 2.2b aplikacja jest typowo randkowa, która stawia sobie za cel stworzenie trwałych relacji poprzez bardziej szczegółowe profile i personalizowane pytania, które użytkownicy odpowiadają, aby pokazać swoją osobowość. - Profil obejmuje zdjęcia, a także odpowiedzi na specjalnie dobrane pytania, które ułatwiają lepsze poznanie osoby. - Użytkownicy mogą komentować i polubić konkretne elementy w profilu, co tworzy punkty wspólne. - Hinge ogranicza dzienną liczbę "lubie" dla lepszej jakości interakcji.



(a)



(b)

Rysunek 2.2: a Badoo. [5], b Hinge. [6]

Rozdział 3

Wymagania i narzędzia

3.1 Wymagania funkcjonalne

Przed rozpoczęciem projektu zdefiniowano pewne wymagania funkcjonalne:

- Uwierzytelnianie użytkowników: System powinien umożliwiać użytkownikowi bezpieczną rejestrację oraz logowanie.
- Edytowanie profilu: System powinien umożliwiać użytkownikowi edycję wszystkich danych wyświetlanych na jego profilu oprócz wieku.
- Wysyłanie i odbieranie wiadomości: Użytkownik musi mieć możliwość komunikacji z innymi użytkownikami platformy, za pomocą wiadomości tekstowych wysyłanym bezpośrednio do wybranej osoby.
- Lokalizacja użytkownika: System jest zmuszony do pobierania lokalizacji użytkownika.
- Polubienia: System musi zliczać ilość wykorzystanych polubień i super polubień.
- Serwis działający w tle: System jest zmuszony do posiadania serwisu działającego w tle, który będzie wykonywał odpowiednie zadania co określony czas.
- Powiadomienia o nowych czacie: System powinien informować użytkownika o powstaniu nowego chatu.
- Filtrowanie użytkowników: System musi posiadać funkcję umożliwiającą użytkownikowi filtrowanie innych użytkowników w zależności od wieku, płci oraz dystansu, który ich dzieli.

3.2 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne zatwierdzone przed utworzeniem projektu:

- Wydajność: witryna powinna załadować się w ciągu maksymalnie 2 sekundy w typowych warunkach użytkownika.
- Bezpieczeństwo: witryna powinna wdrożyć praktyki bezpiecznego kodowania i szyfrować wrażliwe dane.
- Użyteczność: interfejs użytkownika powinien być intuicyjny i prosty w użyciu.

3.3 Opis narzędzi

- Frontend¹: realizacja tej części projektu opierała się na użyciu frameworka² Angular³, z wykorzystaniem kluczowych funkcji budowania aplikacji takich jak routing oraz formularze, co przedstawia artykuł [7]. Wybór ten wynika głównie z jego czytelnej struktury plików projektu oraz z zestawu doskonale zintegrowanych bibliotek, obejmujących szeroką gamę funkcji, takich jak przekierowywanie po stronie, zarządzanie formularzami oraz komunikacja klient-serwer. W trakcie pracy wykorzystano język Typescript⁴, który jest z nim silnie zintegrowany.
- Stylizacja strony: wizualny aspekt strony, został opisany za pomocą HTML⁵ oraz SCSS (Syntactically Awesome Stylesheet)⁶, który pozwala nam w prosty sposób rozszerzyć możliwości zwykłego CSS.
- .NET⁷: zintegrowane środowisko opracowane przez firmę Microsoft. Framework ten został stworzony w celu ułatwienia tworzenia aplikacji o różnym przeznaczeniu, od aplikacji internetowych po oprogramowanie na urządzenia mobilne.
- Baza danych⁸: do przechowywania i wyszukiwania danych został wybrany Microsoft SQL Server⁹, wraz z Microsoft SQL Server Management Studio¹⁰ do łatwego modyfikowania danych bezpośrednio w bazie

¹https://futurecollars.com/slownik_it/frontend-definicja/

²<https://pixlab.pl/framework-co-to-jest-i-do-czego-mozna-go-wykorzystac-w-programowaniu>

³<https://angular.io/>

⁴<https://www.typescriptlang.org/>

⁵<https://www.smart-agency.pl/html-co-to-wlasciwie-jest/>

⁶<https://sass-lang.com/>

⁷<https://learn.microsoft.com/pl-pl/dotnet/core/introduction>

⁸<https://support.microsoft.com/pl-pl/topic/podstawowe-informacje-o-bazach-danych-a849ac16-07c7-4a31-9948-3c8c94a7c204>

⁹<https://www.microsoft.com/pl-pl/sql-server>

¹⁰<https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>

- Kontrola wersji: w projekcie został wykorzystany system kontroli wersji Git¹¹, do śledzenia zmian w kodzie.
- Visual Studio Code¹²: to lekki, ale potężny edytor kodu stworzony przez Microsoft. Oferuje funkcje takie jak kolorowanie składni, podpowiedzi kodu oraz integrację z Git. Posiada również ogromną społeczność, dostarczającą bardzo przydatne rozszerzenia dla wielu języków, frameworków i narzędzi.
- Biblioteki frontendowe:
 1. angular/material¹³: zestaw komponentów do projektowania interfejsu użytkownika dla aplikacji internetowych opartych na frameworku Angular.
 2. angular/material-moment-adapter¹⁴: adapter dostarczany w pakiecie Angular Material, który umożliwia integrację biblioteki moment.js z formularzami dat w Angularze.
 3. ctrl/ngx-emoji-mart¹⁵: aby umożliwić użytkownikowi przesyłanie emotikonów w formie wiadomości, skorzystano z biblioteki.
 4. jwt-decode¹⁶: do dekodowania JWT (JSON Web Token) wykorzystana została biblioteka
 5. microsoft/signalr¹⁷: biblioteka, która umożliwia tworzenie aplikacji webowych wykorzystując WebSocket. Zapewnia prosty sposób na wprowadzenie komunikacji w czasie rzeczywistym między klientem a serwerem, co jest szczególnie przydatne w przypadku aplikacji posiadającej czat czy wymagającej natychmiastowej synchronizacji danych.
 6. rxjs¹⁸: biblioteka programistyczna, która pozwala na programowanie reaktywne. Jest bardzo często używana w połączeniu z frameworkm Angular. RxJS (Reactive Extensions for JavaScript) umożliwia programowanie z użyciem obserwowań sekwencji danych, co pozwala na bardziej efektywny sposób zarządzania operacjami asynchronicznymi, obsługą zdarzeń i manipulacją strumieniami danych, co zostało lepiej opisane w tym artykule [8].
 7. date-fns¹⁹: biblioteka, która specjalizuje się w operacjach związanych z manipulacją dat i czasu.

¹¹<https://git-scm.com/>

¹²<https://code.visualstudio.com/>

¹³<https://www.npmjs.com/package/@angular/material>

¹⁴<https://www.npmjs.com/package/@angular/material-moment-adapter>

¹⁵<https://www.npmjs.com/package/@ctrl/ngx-emoji-mart>

¹⁶<https://www.npmjs.com/package/jwt-decode>

¹⁷<https://www.npmjs.com/package/@microsoft/signalr>

¹⁸<https://www.npmjs.com/package/rxjs>

¹⁹<https://www.npmjs.com/package/date-fns>

8. lodash²⁰: bardzo popularna biblioteka, która dostarcza funkcje narzędziowe do manipulacji danymi. Jest znana ze swojej uniwersalności i efektywności w operacjach na kolekcjach danych.

²⁰<https://www.npmjs.com/package/lodash>

Rozdział 4

Specyfikacja zewnętrzna

4.1 Wymagania sprzętowe i programowe aplikacji

Zaleca się, aby urządzenie, na którym ma być uruchomiona aplikacja, spełniało minimalne parametry:

- Procesor o minimalnej mocy 1GHz
- 2GB pamięci RAM

Ponadto rekomenduje się, aby na urządzeniu była zainstalowana przynajmniej jedna z wymienionych przeglądarek internetowych w wersji 119 lub nowszej:

- Opera
- Microsoft Edge
- Google Chrome
- Brave

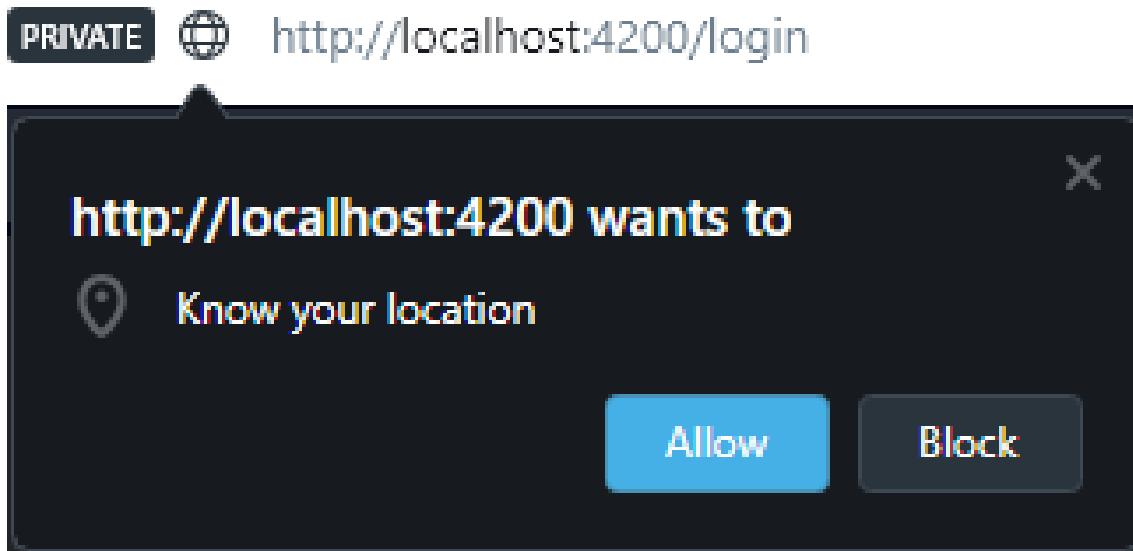
Dodatkowo wymagane są odpowiednie narzędzia do poprawnego skompilowania i działania aplikacji:

- .NET 7.0.5
- Node.js 16.14.0
- Entity Framework Core 7.0.10
- Visual Studio Code
- MS SQL Server 2019
- Angular 16

Zalecanym systemem operacyjnym jest Microsoft Windows 11. Chociaż wszystkie wymienione powyżej narzędzia są dostępne i kompatybilne z systemami Microsoft Windows 10 oraz Linux, jednakże aplikacja nie została na nich przetestowana ani do nich dostosowana.

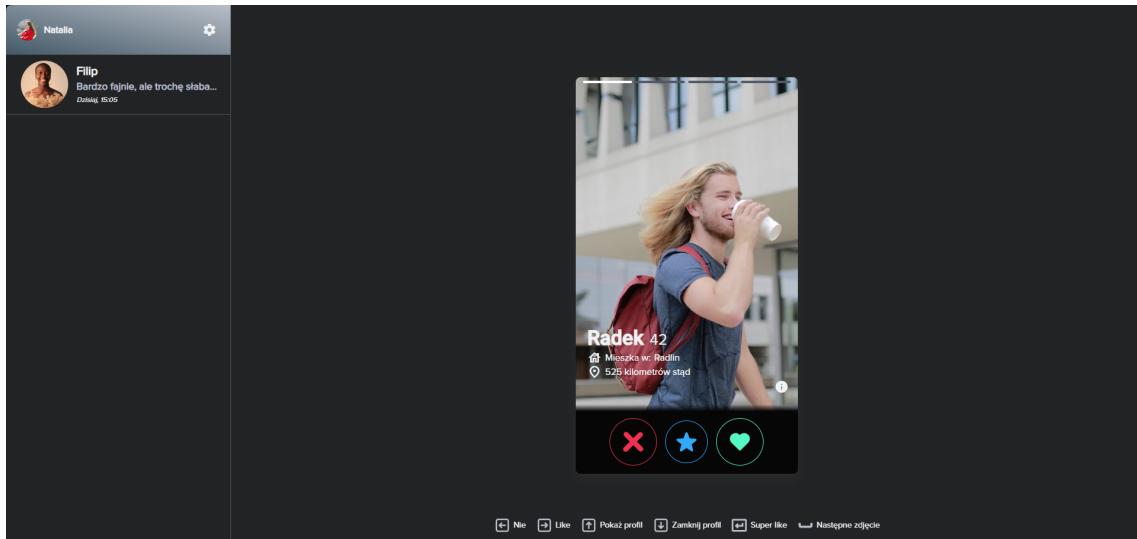
4.2 Sposób obsługi

Każdy użytkownik tworzący konto, ma podstawową i jedyną rolę zwykłego użytkownika. Ma dostęp do takich funkcji jak, edycja swojego profilu, dawanie polubień innym użytkownikom i pisanie wiadomości. Przy pierwszym wejściu na stronę, użytkownikowi pokaże się komunikat odnośnie udostępnienia swojej lokalizacji. Pojawiająca się informacja została przedstawiona na rysunku 4.1. Jeżeli zgoda zostanie przyznana, to lokalizacja użytkownika będzie ustalana na podstawie jego aktualnego miejsca pobytu, w przeciwnym razie będzie ona wyciągana z położenia miasta, wybranego jako miejsce zamieszkania.



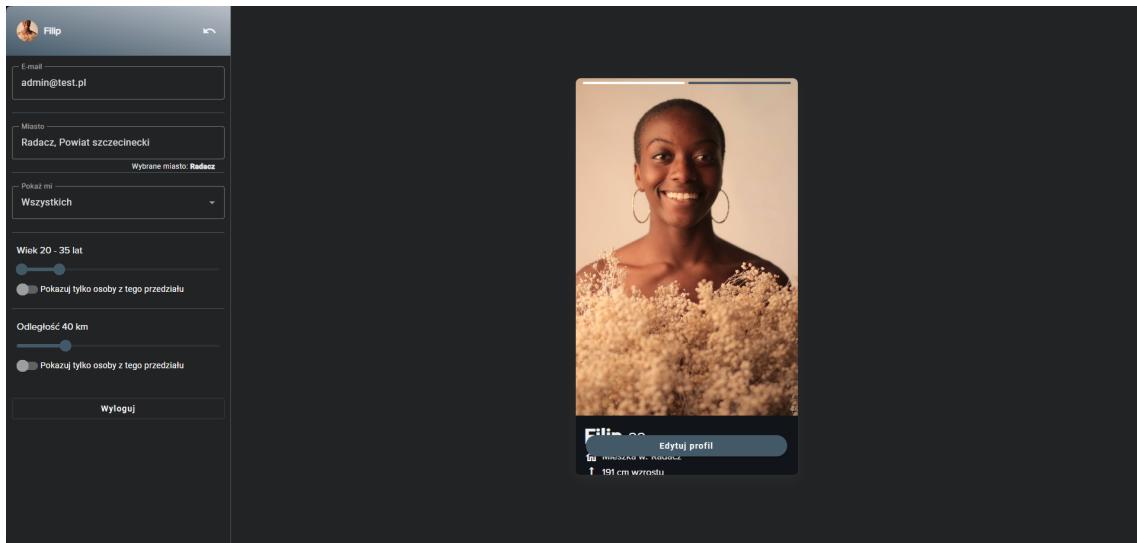
Rysunek 4.1: Lokalizacja użytkownika.

Po utworzeniu konta i zalogowaniu się użytkownik trafia na główną stronę co jest widoczne na ilustracji 4.2.



Rysunek 4.2: Główna strona.

Na środku jest pokazany znaleziony użytkownik. Z lewej strony są widoczne aktualne czaty, a nad nimi znajduje się przycisk, po którego kliknięciu zostanie on przekierowany do ustawień widocznych na obrazku 4.3.

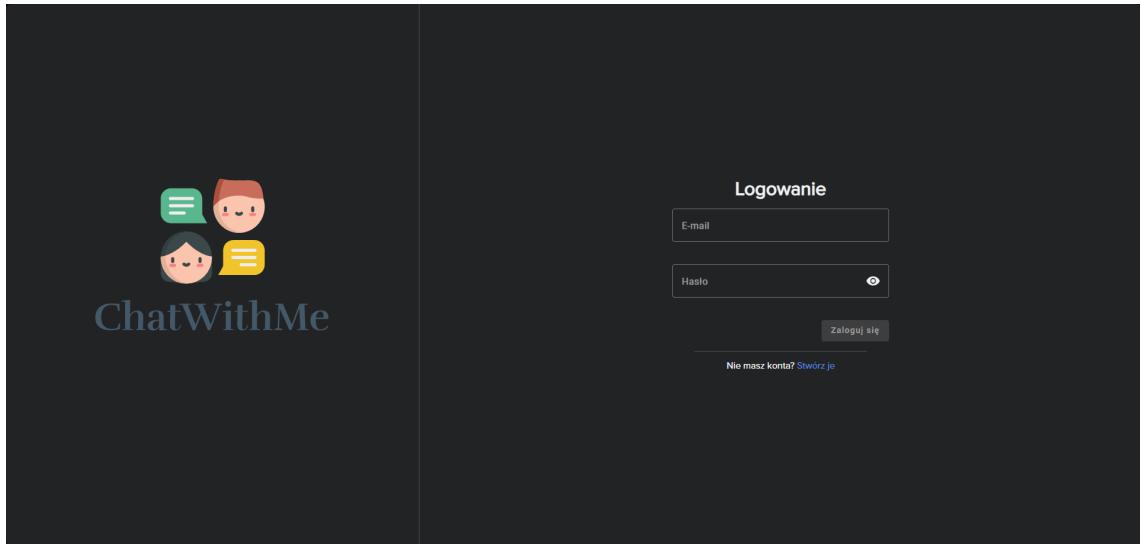


Rysunek 4.3: Strona ustawień.

W tym miejscu może modyfikować swoje preferencje odnośnie wyświetlanych użytkowników oraz informacje o sobie.

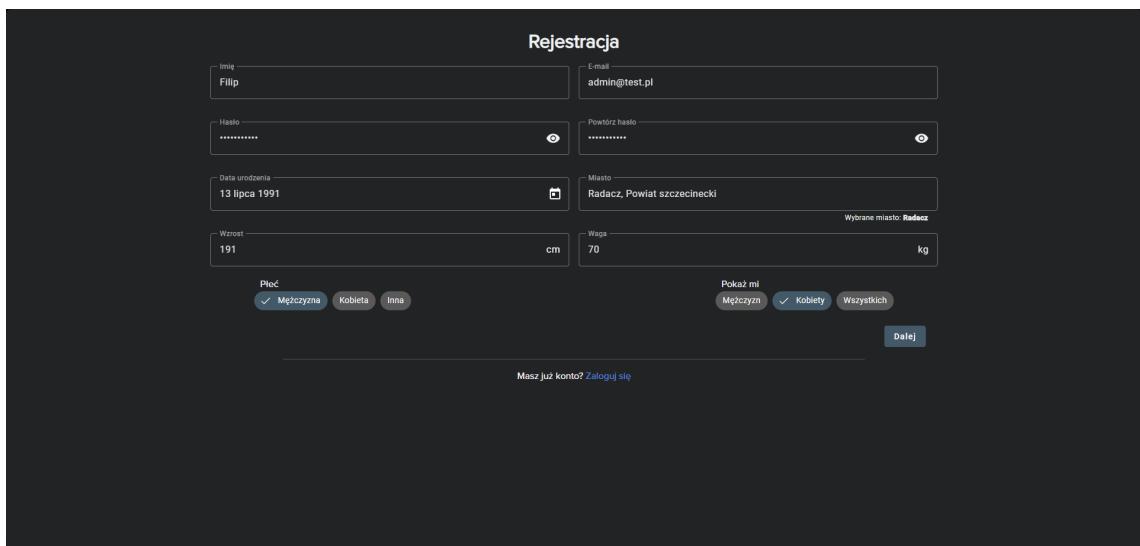
4.3 Przykład działania

Po pomyślnym uruchomieniu aplikacji użytkownik zostanie przekierowany na stronę logowanie, przedstawioną na rysunku 4.4, nie da się korzystać z aplikacji bez utworzonego konta.



Rysunek 4.4: Strona logowania.

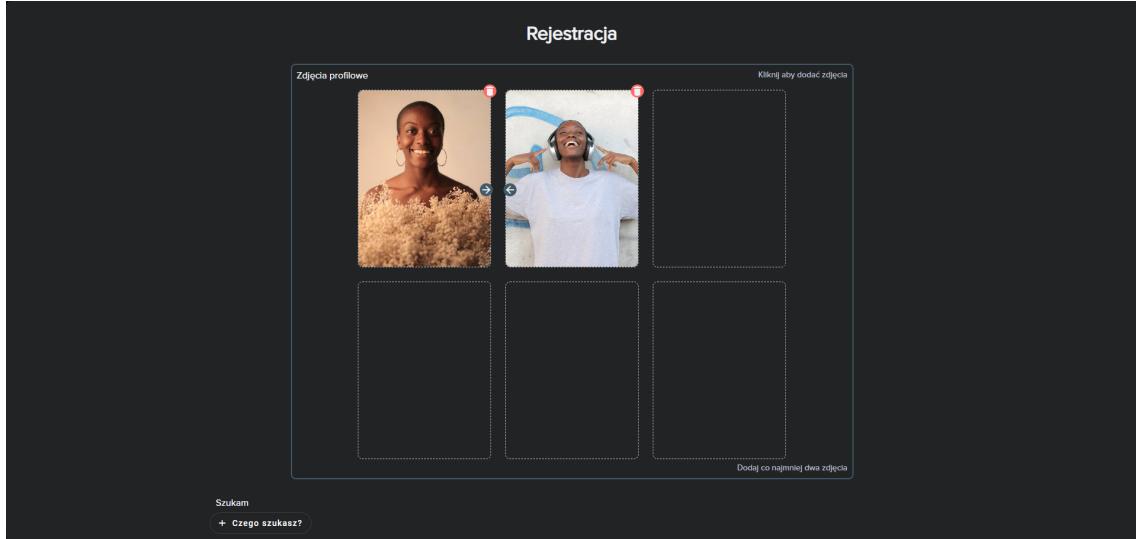
Ta strona, pozwala na zalogowanie się użytkownika, który ma już utworzone konto. Jeżeli takowego nie posiada, to może przejść do rejestracji, klikając przycisk "Stwórz je", co spowoduje przekierowanie na stronę przedstawioną na zdjęciu 4.5.



Rysunek 4.5: Pierwsza strona rejestracji.

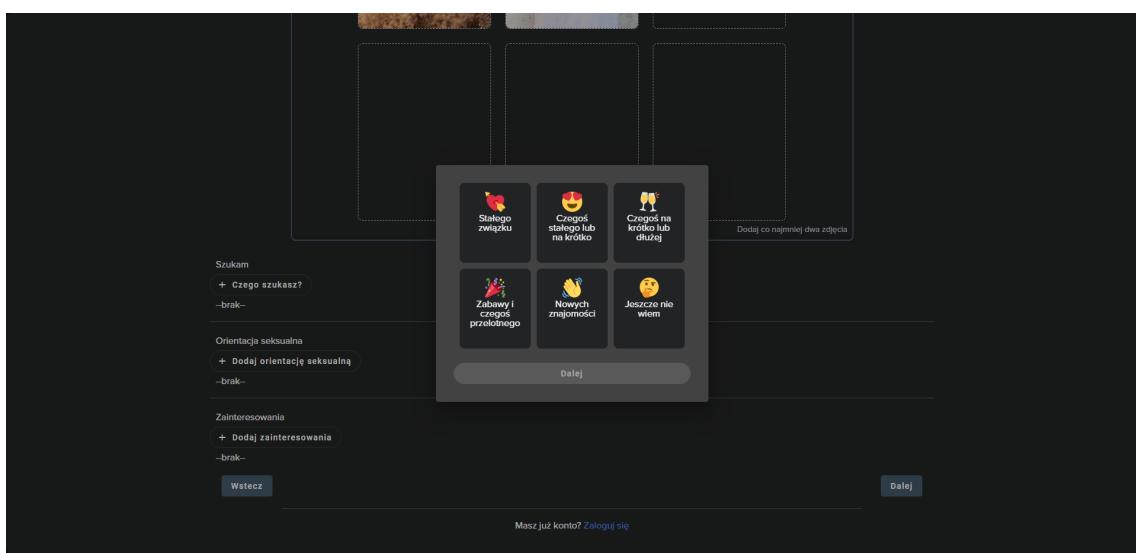
Na tej stronie, użytkownik wprowadza podstawowe dane, takie jak: email, hasło, które są walidowane wyrażeniem regularnym. Email musi mieć formę standardową, a hasło musi mieć minimalnie 8 znaków w tym 1 cyfrę. Data urodzenia, która musi

określać wiek użytkownika większy niż 18 lat. Miasto, z jakiego pochodzi musi istnieć w rejestrze geonames.org, oraz niewalidowane pola takie jak imię, wzrost, waga, płeć własna oraz osoby, z którą preferowałby konwersować. Na następnej stronie została udostępniona możliwość dodawania zdjęć co zostało pokazane na rysunku 4.6.



Rysunek 4.6: Dodanie zdjęć.

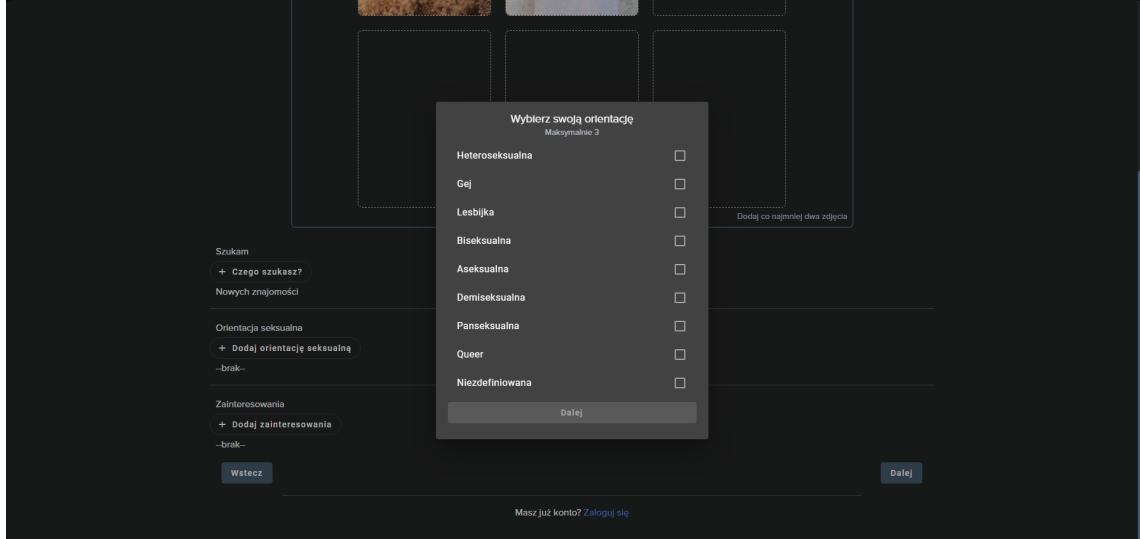
Minimalna ilość zdjęć to 2, a maksymalnie można dodać 6, mając jednocześnie możliwość usunięcia istniejących oraz zmiany ich kolejności wyświetlanego na profilu. Zdjęcie 4.7 przedstawia kolejnym krokiem tworzenia konta, którym jest określenie swojego nastawienia odnośnie innych użytkowników na tej platformie.



Rysunek 4.7: Określenie nastawienia użytkownika.

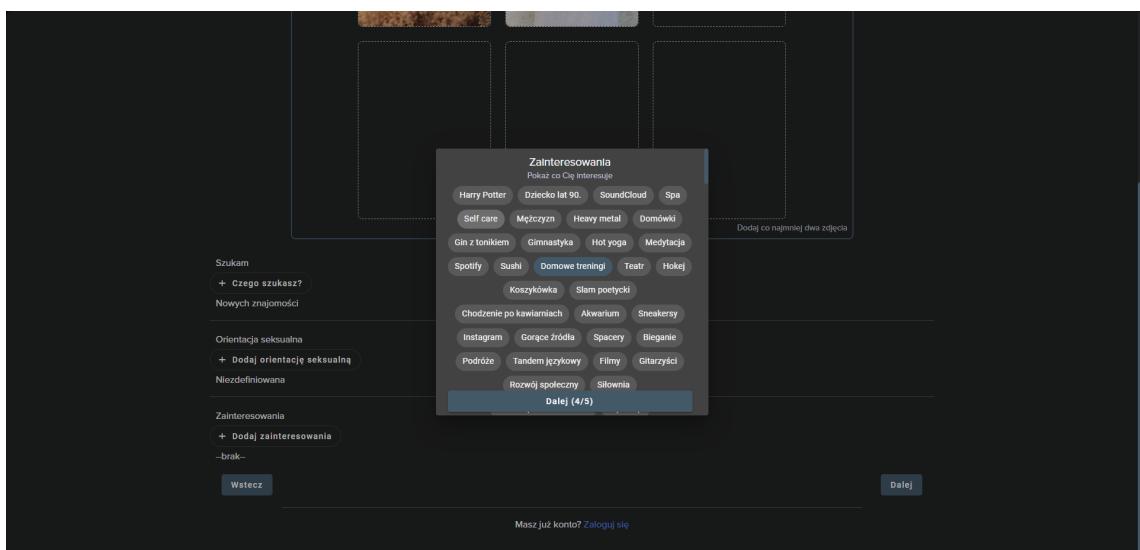
Przy rejestracji na platformie, użytkownikowi zostaje przedstawiona sekcja dotycząca określenia swojej orientacji seksualnej. Proces ten ma na celu stworzenie bardziej spersonalizowanego i zindywidualizowanego doświadczenia na platformie. Użytkownik

ma możliwość wyboru od jednej do trzech opcji, które najlepiej odzwierciedlają jego lub jej orientację seksualną co można zobaczyć na rysunku 4.8.



Rysunek 4.8: Wybór orientacji seksualnej.

W kolejnym kroku użytkownik jest zachęcany do dodatkowego zindywidualizowania swojego profilu poprzez wybór zainteresowań. Ta sekcja umożliwia użytkownikowi określenie obszarów, które go fascynują, oraz nawiązanie więzi z osobami, które mają podobne pasje. Proces ten ma na celu stworzenie bardziej ukierunkowanego doświadczenia społecznościowego, gdzie użytkownicy mogą łatwiej odnaleźć osoby o wspólnych zainteresowaniach. Na zdjęciu 4.9 są przedstawione przykładowe zainteresowania, które obejmują szeroki zakres dziedzin, takich jak sport, sztuka, podróże, literatura, technologia, muzyka czy filmy.



Rysunek 4.9: Wybór zainteresowań.

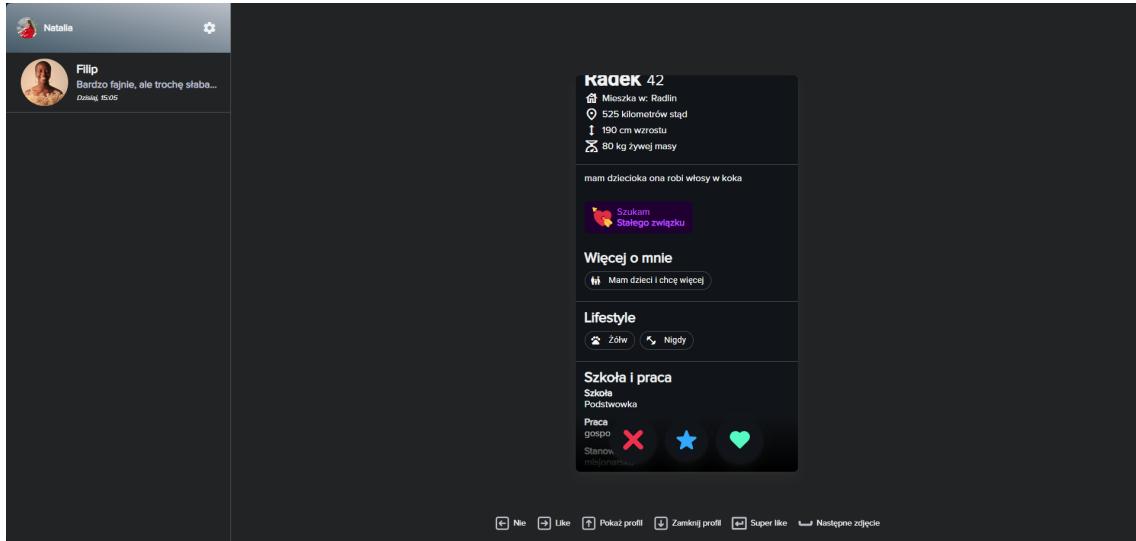
Ostatnim etapem procesu tworzenia nowego konta jest możliwość uzupełnienia dodatkowych informacji, które są całkowicie opcjonalne. To idealna okazja, aby użytkownik podzielił się informacjami o sobie i stworzył bardziej kompletny profil. Można dodać opis, podać swój znak zodiaku, podkreślić swoje wykształcenie oraz aktualne miejsce pracy i stanowisko. Ponadto istnieje możliwość ujawnienia informacji o szkole, do której się uczeszczało, oraz odpowiedzi na pytania dotyczące preferencji związanych z piciem alkoholu, paleniem papierosów, aktywności fizycznej, diety oraz planów dotyczących posiadania dzieci. Jest to w pełni opcjonalne, ale może się przyczynić do lepszego poznania przez inne osoby 4.10.

Rysunek 4.10: Dodatkowe informacje o użytkowniku.

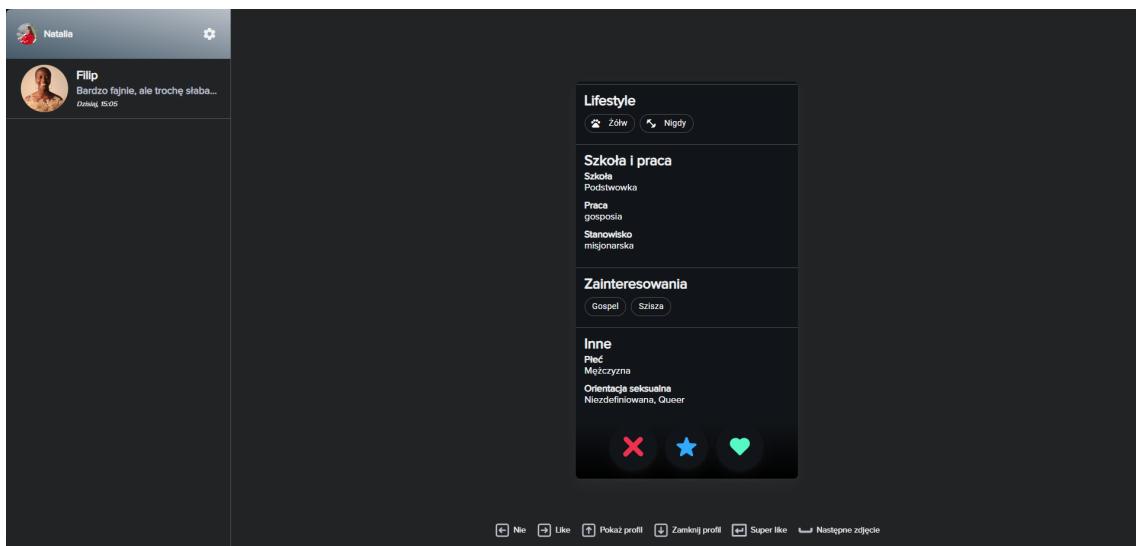
Po utworzeniu konta, pokazany zostanie ekran logowania widoczny na rysunku 4.4, tylko tym razem z wypełnionymi polami jak przy rejestracji. Po pomyślnym zalogowaniu się zostanie wyświetlona główna strona aplikacji prezentowana na rysunku 4.2. Na środku strony jest pokazany znaleziony użytkownik, mieszczący się w kryteriach wybranych przez użytkownika (płeć, wiek, odległość), któremu może dać polubienie, super polubienie lub go odrzucić, w zależności od własnych preferencji. Można to zrobić za pomocą myszy, klikając lewy przycisk na odpowiednią akcję znajdująca się na widocznym profilu lub klawiatury wykorzystując strzałki, przycisk enter oraz spację. Instrukcja działania przycisków jest widoczna na dole ekranu, pod wyświetlonym profilem. Na zdjęciu użytkownika są widoczne tylko podstawowe dane, aby zobaczyć wszystkie, trzeba nacisnąć ikonkę "Informacja". Po wykonaniu tej czynności profil zostanie rozwinięty i zjeżdżając w dół, można zobaczyć wszystkie informacje widoczne na obrazkach 4.11 oraz 4.12, które podał.

Jeżeli posiada istniejące czaty, to z lewej strony będą one widoczne, razem z podstawowymi informacjami o osobie konwersującej takie jak: imię, główne zdjęcie, ostatnia wiadomość z czatu razem z godziną wysłania oraz czy została ona wyświetlona.

Filip Miera



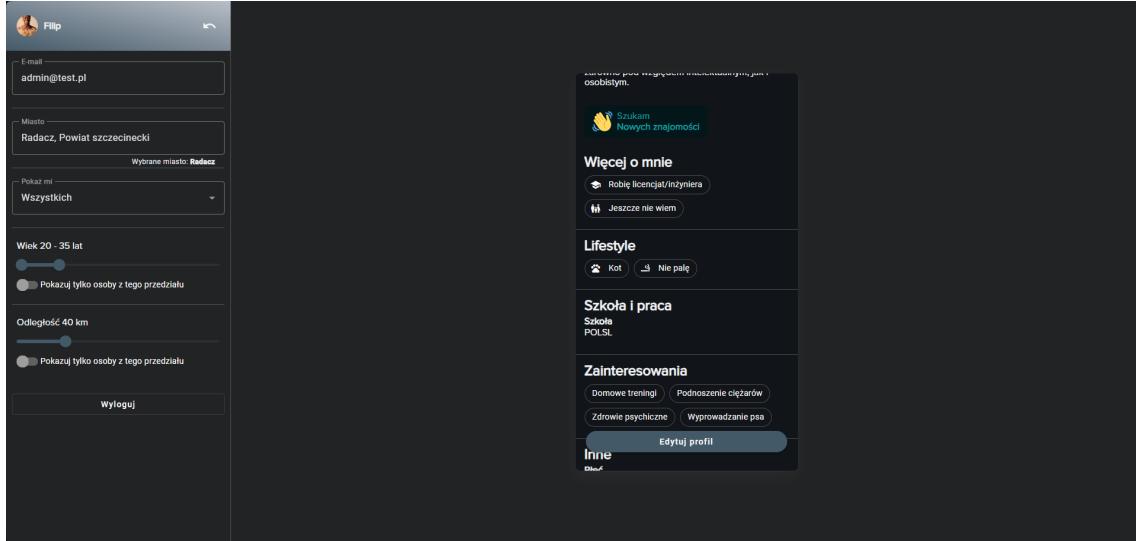
Rysunek 4.11: Więcej informacji cz.1.



Rysunek 4.12: Więcej informacji cz.2.

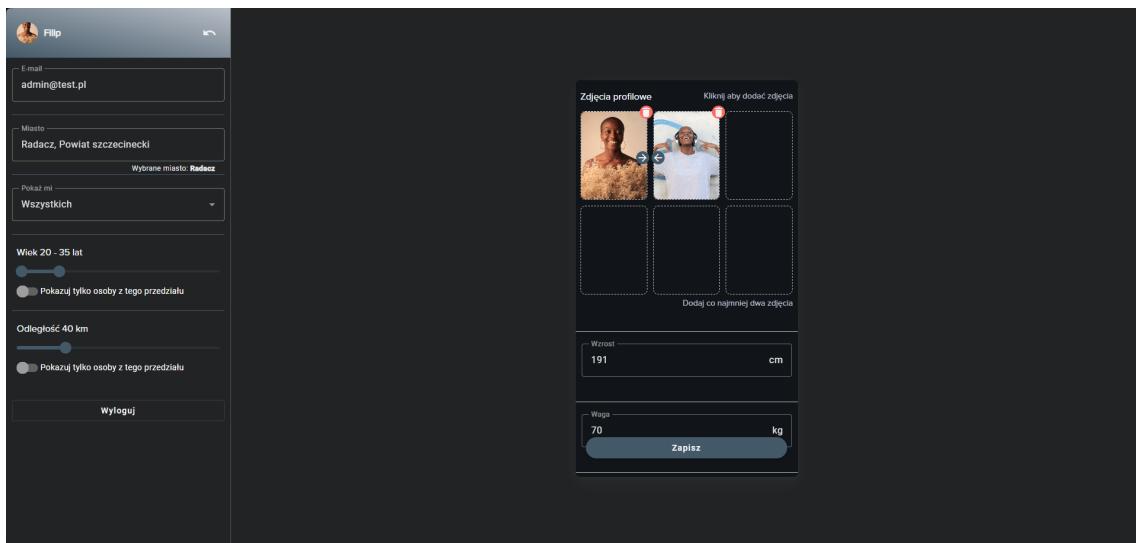
W innym przypadku będzie widoczna informacja o braku istniejących konwersacji. Wyżej znajduje się przycisk z imieniem aktualnego użytkownika oraz jego głównym zdjęciem, po kliknięciu go zostanie on przekierowany do strony ustawień widocznej na ilustracji 4.3.

Na tej stronie, użytkownik może wylogować się ze swojego konta i modyfikować swoje preferencje odnośnie wyświetlanych mu użytkowników, takie jak płeć, odległość czy wiek. Na środku zdjęcia 4.13 jest widoczny profil zalogowanego użytkownika z jego wszystkimi informacjami.

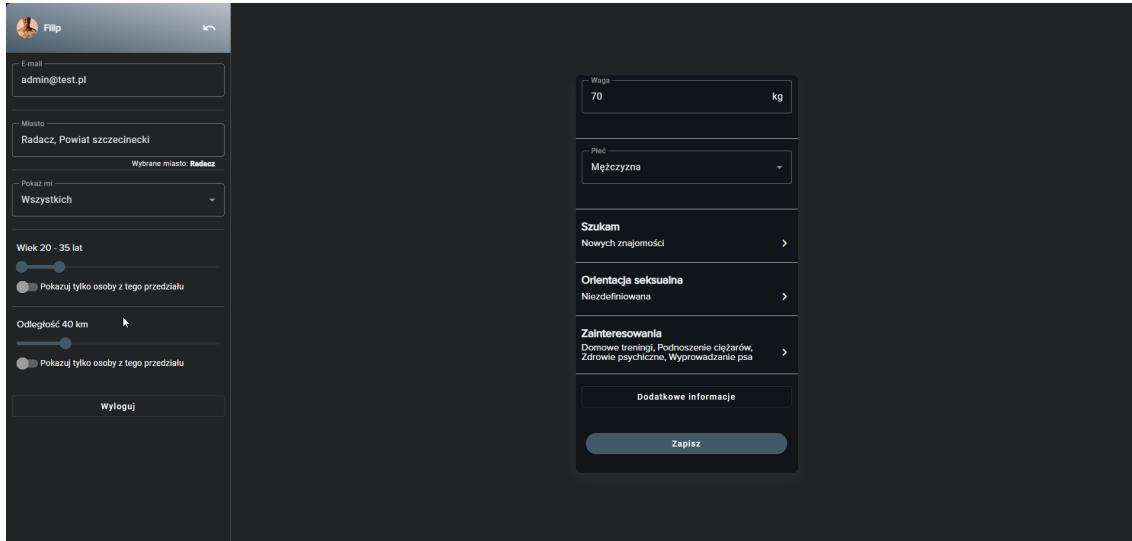


Rysunek 4.13: Więcej informacji aktualnego użytkownika.

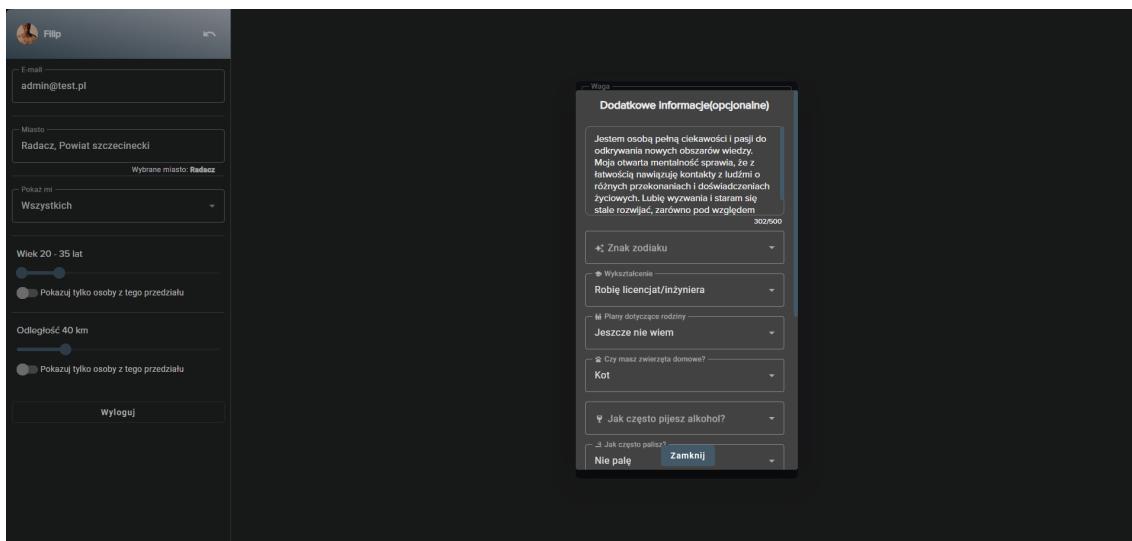
Aby je edytować, musi użyć przycisku "Edytuj profil", który pozwoli na edycję informacji, które są widoczne na jego profilu co jest pokazane na poniższych zdjęciach 4.14 4.15 4.16. Aby wrócić na stronę główną, należy wcisnąć ten sam przycisk, który przenosi do ustawień.



Rysunek 4.14: Edycja profilu cz.1.

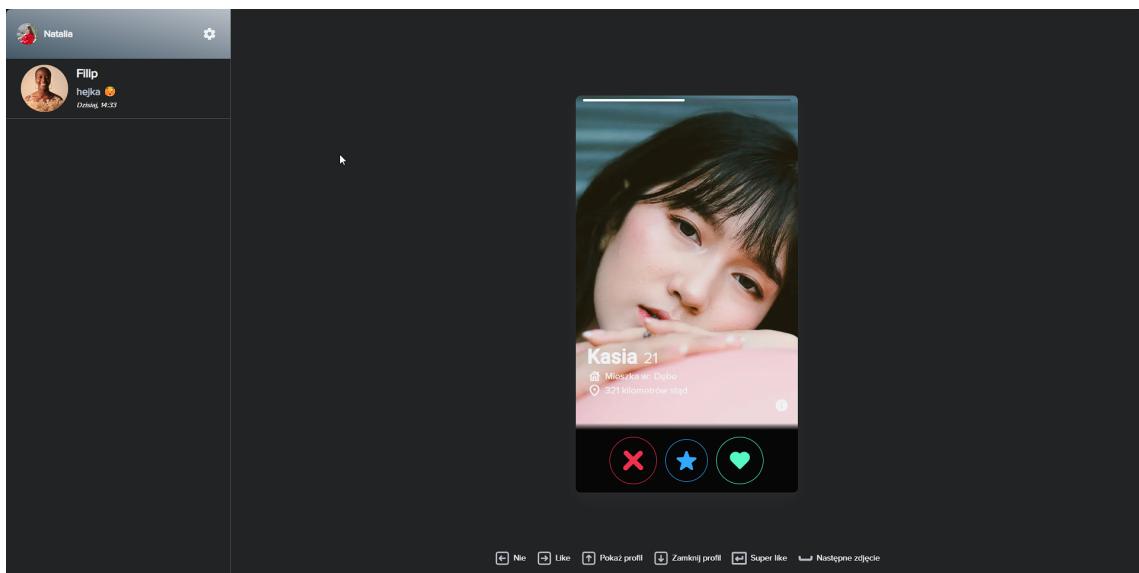


Rysunek 4.15: Edycja profilu cz.2.

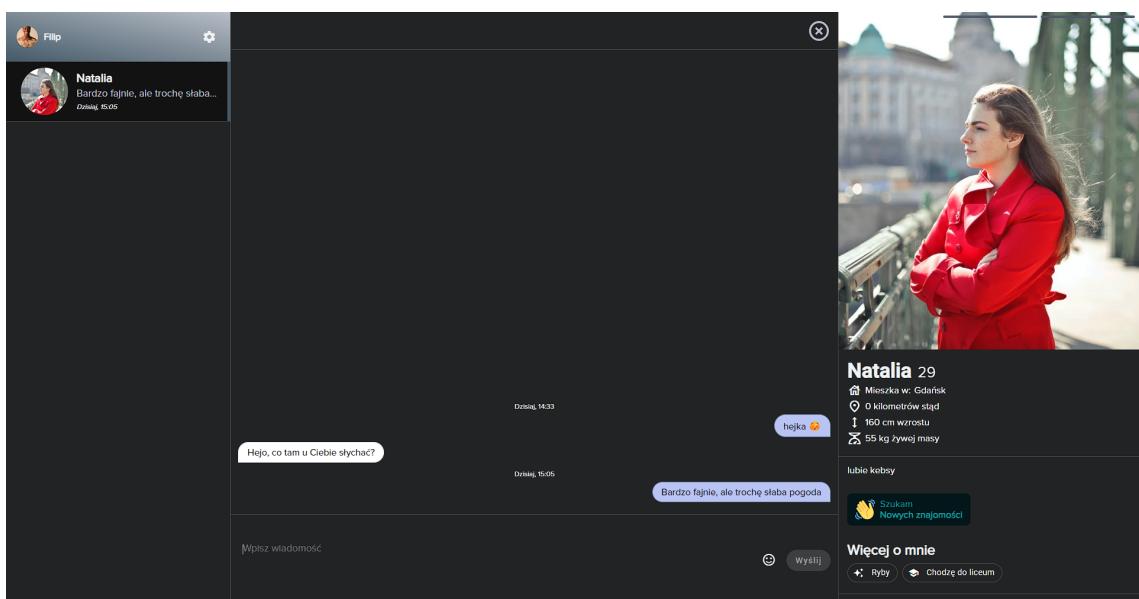


Rysunek 4.16: Edycja profilu cz.3.

Z lewej strony są widoczne wszystkie czaty. Jeśli użytkownik posiada kilka czatów i otrzyma nową wiadomość, wszystkie czaty zostaną posortowane według najnowszej wiadomości. Dodatkowo nieprzeczytane wiadomości będą oznaczone pogrubionym tekstem, aby łatwo było zauważyc, które wiadomości są nowe co można zobaczyć na grafice 4.17. Po kliknięciu czatu zostanie on otwarty co pokazuje zdjęcie 4.18. Po wykonaniu tej czynności zostanie on, automatycznie oznaczony jako "przeczytany" i zostanie wyświetlony widok wybranej konwersacji. Otwarty czat, zostaje wyróżniony w liście, poprzez zmianę tła oraz dodanie ramki z prawej strony. W głównej sekcji ekranu znajduje się czat między użytkownikami.

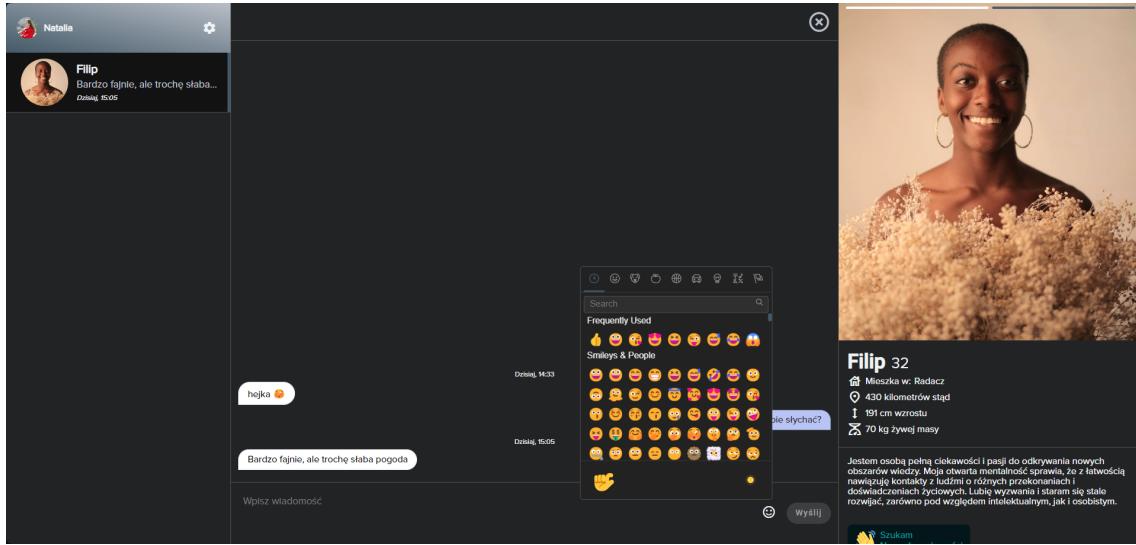


Rysunek 4.17: Cza zamknięty, widok użytkownika Natalia.



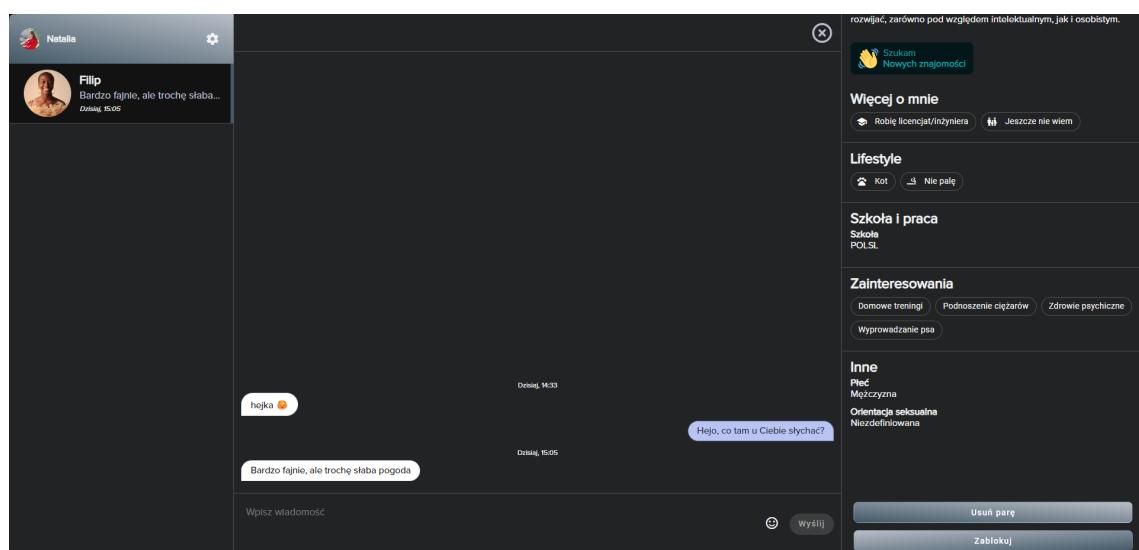
Rysunek 4.18: Cza otwarty, widok użytkownika Filip.

Wiadomości widoczne z prawej strony, są komunikatami wysłanymi przez aktualnie zalogowanego użytkownika. Jeżeli, między kolejnymi wiadomościami jest różnica czasowa, większa niż 15 minut, to na środku zostanie pokazana godzina wysłania. Dodatkowo po najechaniu na tekst, pojawi się dokładny czas jej wysłania. Obok przycisku wysyłania wiadomości, znajduje się guzik w postaci uśmiechniętej twarzy, po naciśnięciu którego wyświetla się menu widoczne na obrazie 4.19, z którego można wybrać najróżniejsze emotikony, aby urozmaicić i wzrobić rozmowę.



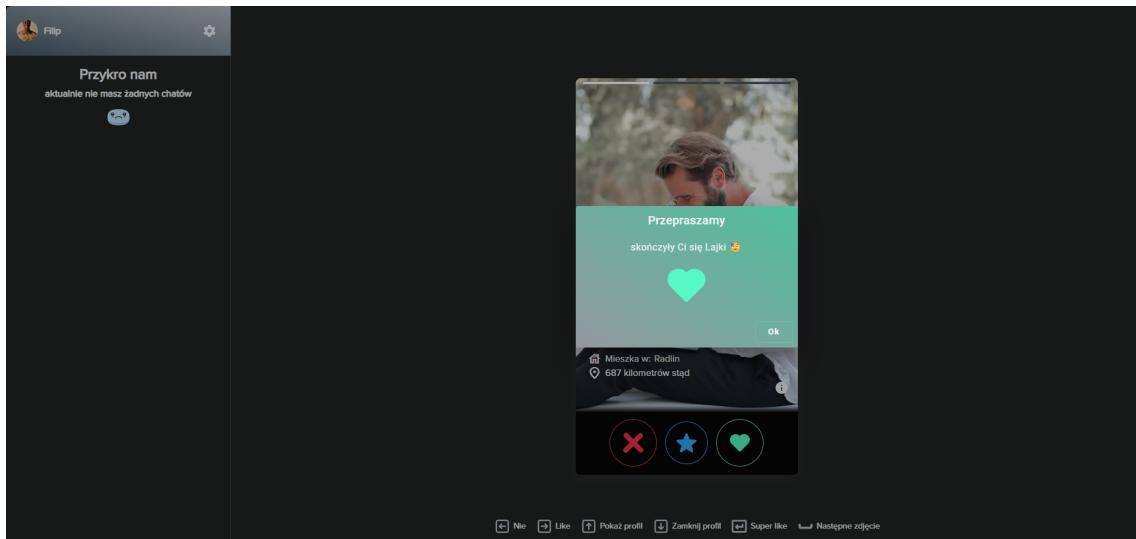
Rysunek 4.19: Czat, emotikony.

W prawej części ekranu znajdują się wszystkie informacje, które rozmówca ma udostępnione na swoim profilu. Na samym dole znajdują się 2 przyciski, jeden do usuwania pary, który chowa czat z tą osobą i dodaje ją do bazy danych jako osobę, której się nie polubiło. W tym przypadku istnieje możliwość, aby po raz kolejny dać polubienie tej osobie. Natomiast drugi przycisk z napisem "Zablokuj", całkowicie ukrywa tę osobę przed użytkownikiem i nigdy nie będzie można jej polubić. Te opcje są widoczne na rysunku 4.20.

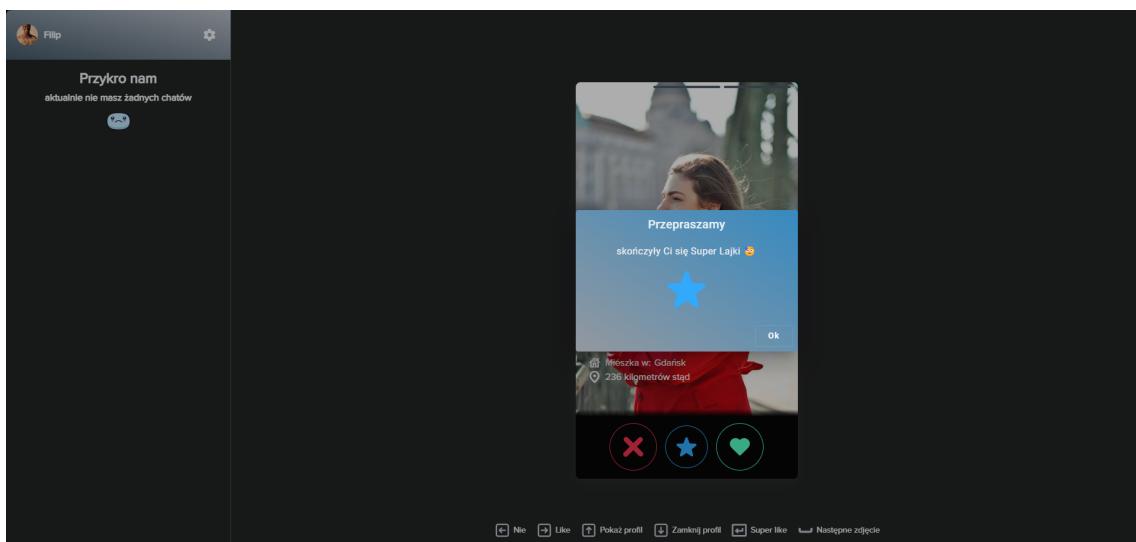


Rysunek 4.20: Czat, blokowanie i usuwanie pary.

Każdy użytkownik, ma określoną ilość polubień (30) oraz super polubień (2) na cykl. Cykl w aplikacji trwa 4 godziny. Przy każdym przejściu ilość polubień oraz super polubień resetuje się do wartości początkowej, oraz usuwają się wszystkie niepolubione osoby. Po przekroczeniu któregoś limitu pojawi się odpowiedni komunikat dla polubień lub super polubień przedstawiony na grafikach 4.21 i 4.22. Osoba, której użytkownik dał polubienie przed pojawieniem się komunikatu, zostanie po prostu dodana znów do kolejki, bez żadnych konsekwencji.

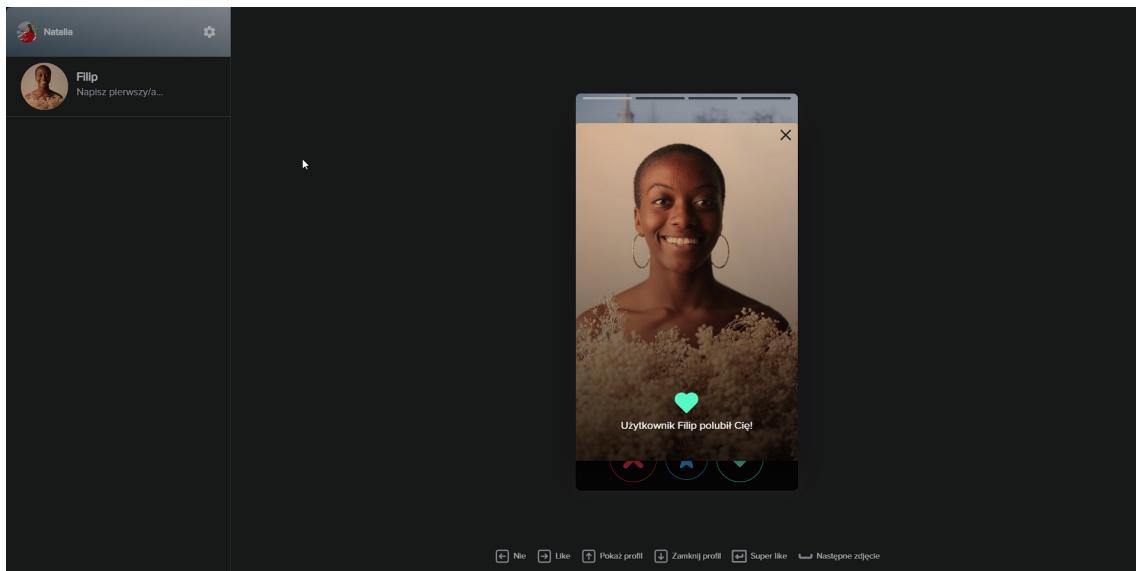


Rysunek 4.21: Informacja o braku polubień.

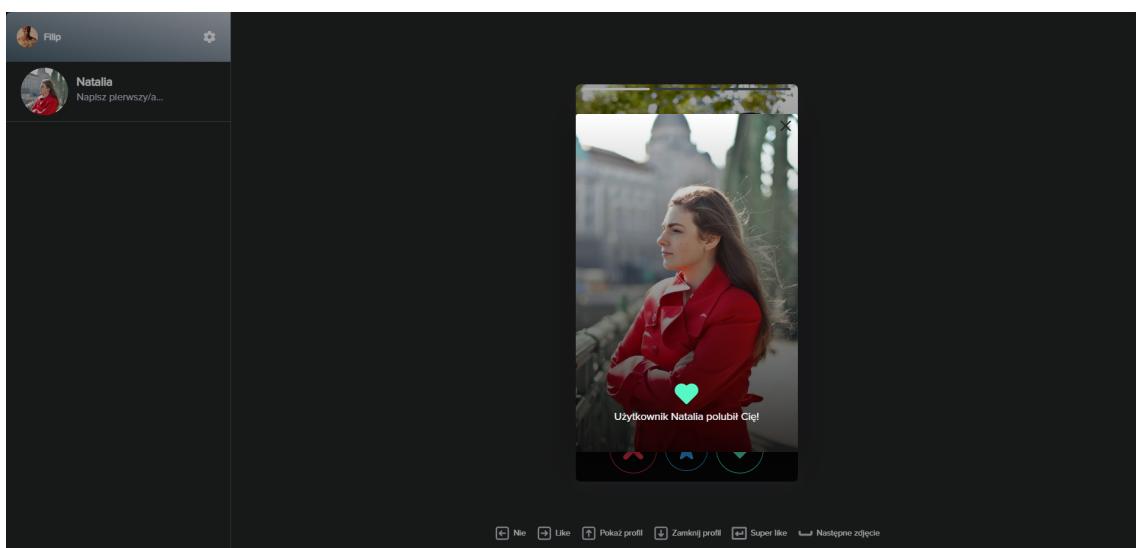


Rysunek 4.22: Informacja o braku super polubień.

Podczas procesu polubiania użytkowników, system sprawdza, czy drugi użytkownik wcześniej polubił już aktualnego użytkownika. Jeżeli nie, to nic się nie dzieje. Jednak, jeżeli tak, obu użytkownikom zostaje wyświetlony dialog informacyjny przedstawiony na ilustracjach 4.23 oraz 4.24, zawierający informacje o tym, z kim utworzona została konwersacja, oraz czy ten drugi użytkownik udzielił polubienia, czy super polubienia. Powiadomienie to pojawia się wyłącznie w momencie tworzenia konwersacji. Jeżeli drugi użytkownik w danym momencie nie jest zalogowany, to system automatycznie tworzy nowy czat bez dodatkowego powiadamiania.



Rysunek 4.23: Informacja o nowej konwersacji, widok użytkownika Natalia.



Rysunek 4.24: Informacja o nowej konwersacji, widok użytkownika Filip.

Rozdział 5

Specyfikacja wewnętrzna

5.1 Architektura

Struktura systemu obejmuje zarówno interfejs użytkownika, jak i warstwę serwerową. Architektura systemu została zoptymalizowana zgodnie ze standardem REST (Representational state transfer)¹ API (Application programming interface)², który określa zasady komunikacji z systemem. Do budowy serwera wykorzystano technologię ASP.NET, która umożliwia tworzenie aplikacji zgodnych z zasadami REST API, szerzej opisanymi w tym artykule [9].

Interfejs użytkownika został stworzony na podstawie technologii Angular i języka TypeScript, wykorzystując głównie bibliotekę Angular Material. Utworzony interfejs umożliwia interakcję z użytkownikiem, wstępną walidację wprowadzanych danych oraz komunikację z warstwą serwera poprzez odpowiednie punkty końcowe.

¹<https://zaprogramujzycie.pl/czym-jest-rest/>

²<https://ks.pl/slownik/co-to-jest-api>

5.2 Ważniejsze klasy

W kolejnych fragmentach podane zostaną szczegółowe opisy kluczowych klas, które pełnią istotną rolę w strukturze i funkcjonowaniu całej aplikacji. Przedstawienie tych klas pozwoli na lepsze zrozumienie ich roli oraz wzajemnych relacji, co stanowi kluczowy element w zrozumieniu architektury aplikacji.

5.2.1 Frontend

W tym interceptorze, do każdego żądania dodawany jest do nagłówka token uwierzytelniający przypisany do danego użytkownika. Dzięki temu serwer jest w stanie identyfikować wykonawcę danej funkcji i odpowiednio reagować na nią, co przedstawia kod na rysunku 5.1.

```
1  @Injectable()
2  export class AuthInterceptor implements HttpInterceptor {
3
4      constructor(private userService: UserService) { }
5
6      intercept(request: HttpRequest<unknown>, next: HttpHandler
7          ): Observable<HttpEvent<unknown>> {
8          if (!request.url.includes('geonames'))
9              request = request.clone({
10                  setHeaders: {
11                      'Authorization': `bearer ${this.userService.
12                          getUserToken()}`,
13                  }
14              });
15          return next.handle(request);
16      }
17  }
```

Rysunek 5.1: Interceptor autoryzacyjny.

W przypadku otrzymania przez interceptor błędu 401, co oznacza 'Unauthorized', oraz gdy token użytkownika nie jest przeterminowany, interceptor podejmie próbę ponownego zalogowania użytkownika. Wykorzystuje do tego celu mechanizm odświeżania logowania (Refresh Token), który został zaimplementowany w przedstawiony sposób 1.

Aby strona była dostępna jedynie dla zalogowanych użytkowników, konieczne jest użycie tego guarda. Jego rola polega na uniemożliwieniu niezalogowanym osobom dostępu do głównej strony poprzez bezpośrednie użycie adresu URL (Uniform Resource Locator). Tę funkcjonalność oferuje przedstawiona fukncja na grafice 5.2.

```

1  export const userAuthenticated = () => {
2      const authService = inject(UserService);
3      const router = inject(Router);
4      if (authService.isAuthenticated())
5          return true;
6
7      void router.navigateByUrl(RoutesPath.LOGIN);
8  };

```

Rysunek 5.2: Guard blokujący stronę główną.

W celu usprawnienia funkcjonalności i intuicyjności aplikacji, dodany został guard, którego zadaniem jest automatyczne przekierowywanie użytkownika, który nie został wylogowany ani którego token nie wygasł, na stronę główną, gdy próbuje uzyskać dostęp do strony logowania lub rejestracji. To działanie pokazuje ilustacją 5.3.

```

1  export const userNotAuthenticated = () => {
2      const authService = inject(UserService);
3      const router = inject(Router);
4      if (!authService.isAuthenticated())
5          return true;
6
7      void router.navigateByUrl(RoutesPath.HOME);
8  };

```

Rysunek 5.3: Guard blokujący stronę logowania.

Serwis silnie wykorzystujący bibliotekę signalr, odpowiedzialny wyłącznie za obsługę zapytań związanych z czatem otrzymywanych przez połączenie websocketowe, obejmuje funkcje takie jak pobieranie czatów, pobieranie wiadomości, otrzymywanie nowych wiadomości, tworzenie nowych konwersacji oraz blokowanie/odrzucanie wcześniej polubionego użytkownika co widoczne na obrazie 3.

Serwis pełniący rolę zarządzania stanem aplikacji oraz przechowywania danych, które nie wymagają pobierania przy każdym nowym wyświetleniu. Przykładowe dane przechowywane przez ten serwis, to informacje o aktualnym użytkowniku w ustawieniach. Zestawiono to na rysunku 6.

5.2.2 Backend

Do automatycznego wykonywania akcji został wykorzystany BackgroundService widoczny na ilustracji 9, wykonuje się on co 4 godziny. Posiada funkcje resetowania ilości polubień, super polubień i usuwania niepolubień użytkowników z bazy danych, aby znów można było ich zobaczyć. Usługa działa w tle, dbając o porządek danych bez potrzeby ingerencji użytkownika.

Klasa zobrazowana na rysunku 5.4 dostarcza informacje o bieżącym użytkowniku. Korzysta z httpContextAccessor do odczytania tokenu bezpieczeństwa (JWT) z żądania HTTP (Hypertext Transfer Protocol). Następnie ekstrahuje i przypisuje do ogólnodostępnej zmiennej identyfikator użytkownika z tego tokenu.

```
1  public class CurrentUserService : ICurrentUserService
2  {
3      private readonly IHttpContextAccessor _httpContextAccessor
4      ;
5
6      public CurrentUserService(IHttpContextAccessor
7          httpContextAccessor)
8      {
9          _httpContextAccessor = httpContextAccessor;
10     }
11
12     public JwtSecurityToken? UserToken => Reader.ReadToken(
13         _httpContextAccessor.HttpContext);
14     public string? UserId => UserToken?.FindOrDefault(
15         ClaimNames.Id);
16 }
```

Rysunek 5.4: Klasa przechowująca id użytkownika.

Klasy ChatHub i ChatService współpracują tworząc kompleksową infrastrukturę obsługi czatu. ChatHub działa jako mediator, używając technologii SignalR do obsługi połączeń WebSocket między frontendem a serwerem. Jako punkt centralny komunikacji, umożliwia otrzymywanie poleceń od klientów i wysyłanie danych asynchronicznie w czasie rzeczywistym do wszystkich połączonych klientów. ChatService pełni rolę dostawcy danych, dostarczając operacje dostępu do danych związanych z czatem. Wykorzystywany przez ChatHub, umożliwia pobieranie informacji z bazy danych. Ta współpraca zapewnia płynną i efektywną komunikację w czasie rzeczywistym w systemie czatu.

5.3 Ważniejsze algorytmy

Do obliczenia odległości między dwoma użytkownikami na podstawie ich współrzędnych geograficznych (szerokości i długości geograficznej), zastosowano wzór Haversine'a ujęty na rysunku 5.5. Wzór ten jest matematyczną metodą używaną do obliczania odległości pomiędzy dwoma punktami na powierzchni sfery. Wzór Haversine'a jest wyrażony równaniem trigonometrycznym, a jego podstawą jest zastosowanie funkcji sinus do obliczeń. Pozwala on uwzględnić zakrzywienie ziemi i dostarcza dokładniejsze wyniki w porównaniu do prostszych metodyk obliczeń na płaskiej powierzchni.

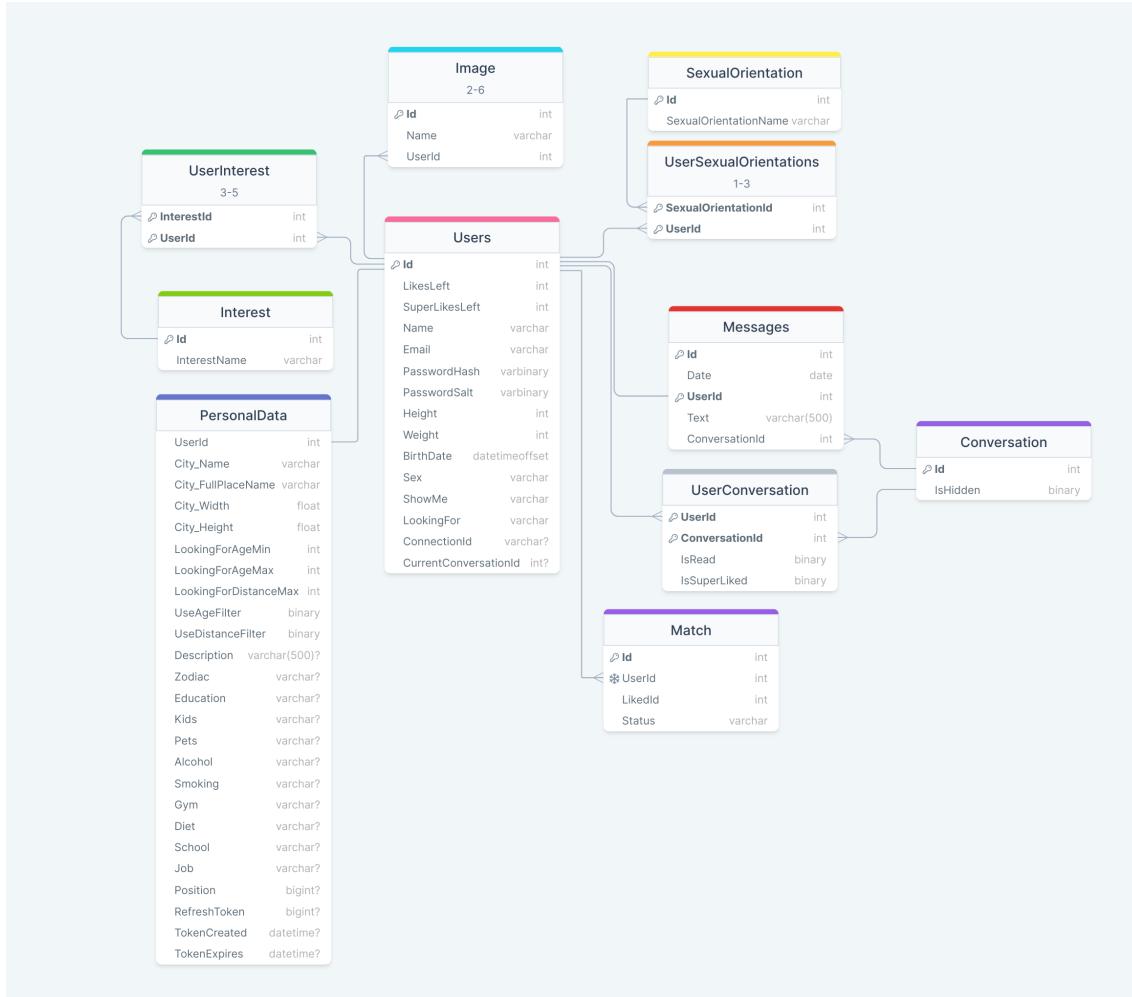
```

1  private const double EarthRadiusKm = 6371.0;
2
3  private double CalculateDistance(double lat1, double lon1,
4      double lat2, double lon2)
5  {
6      //Konwersja stopni na radiany
7      var dLat = DegreeToRadian(lat2 - lat1);
8      var dLon = DegreeToRadian(lon2 - lon1);
9
10     //Obliczanie odległości za pomocą wzoru Haversine
11     var a = Math.Sin(dLat / 2) * Math.Sin(dLat / 2) +
12         Math.Cos(DegreeToRadian(lat1)) * Math.Cos(
13             DegreeToRadian(lat2)) *
14             Math.Sin(dLon / 2) * Math.Sin(dLon / 2);
15     var c = 2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));
16     var distanceKm = EarthRadiusKm * c;
17
18     return distanceKm;
19 }
20
21 private double DegreeToRadian(double angle)
22 {
23     return Math.PI * angle / 180.0;
24 }
```

Rysunek 5.5: Obliczanie odległości.

5.4 Baza danych

W aplikacji wszystkie dane, są przechowywane w relacyjnej bazie danych. Szczegółowy opis tabel i ich atrybutów zostały przedstawione na schemacie bazy danych przedstawionym na rysunku 5.6.



Rysunek 5.6: Schemat bazy danych.

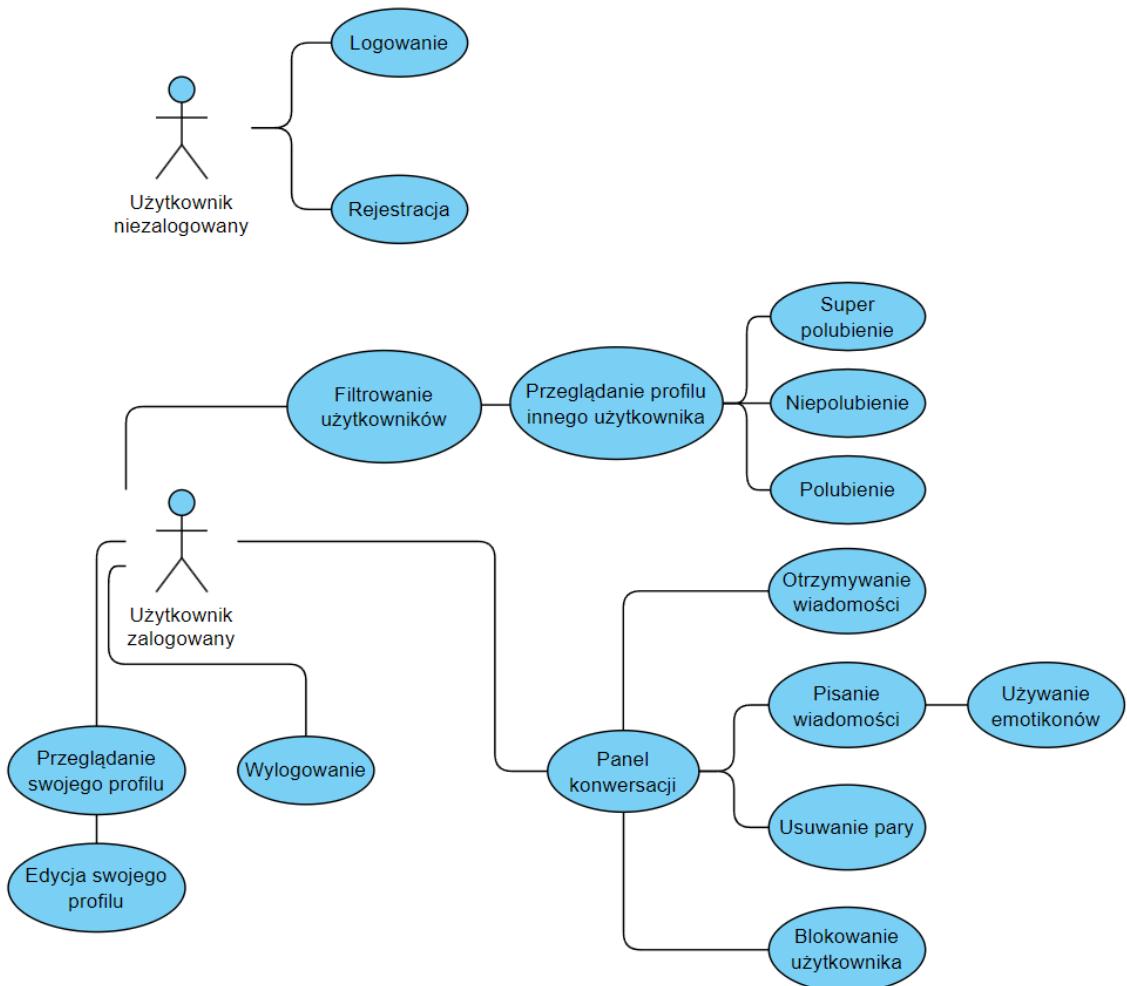
Poniżej znajduje się opis poszczególnych tabel aplikacji:

- **Users**: Przechowuje najistotniejsze dane użytkownika takie jak zaszyfrowane hasło, nazwę użytkownika, email, pozostałe polubienia, identyfikator połączenia czatu oraz identyfikator aktualnie otwartej konwersacji.
- **PersonalData**: Opcjonalne dane użytkownika, jego filtry, miejsce zamieszkania oraz dane tokenu (data utworzenia, data wygaśnięcia, token odnawiający).
- **Image**: Przechowywuje zdjęcia użytkowników w formacie base64
- **UserInterests**: Tabela wiążąca relację many-to-many między tabelami użytkownika oraz zainteresowań.

- **Interest:** Przechowuje wszystkie możliwe do wyboru zainteresowania użytkowników, ma relację many-to-many ponieważ jeden użytkownik może mieć wiele zainteresowań i jedno zainteresowanie może mieć wielu użytkowników.
- **UserSexualOrientations:** Struktura łącząca użytkowników z ich preferencjami seksualnymi w relacji many-to-many pomiędzy tabelami użytkowników a preferencji seksualnych.
- **SexualOrientation:** Przechowuje dostępne opcje preferencji seksualnych użytkowników. Ma relację many-to-many, co oznacza, że jeden użytkownik może mieć różne preferencje seksualne, a jedna preferencja seksualna może być współdzielona przez wielu użytkowników.
- **UserConversation:** Tabela łącząca użytkowników z ich konwersacjami, utrzymująca relację many-to-many między tabelami użytkowników a rozmów. Dodatkowo zawiera pola isRead używane do pokazywania czy użytkownik przeczytał wiadomość z tej konwersacji i isSuperLiked wykorzystywane do wyświetlenia odpowiedniej ikony polubienia po wejściu w konwersację.
- **Conversation:** Przechowuje informacje o wszystkich dostępnych rozmowach. Ma relację many-to-many, co oznacza, że jeden użytkownik może uczestniczyć w wielu rozmowach, a jedna rozmowa może obejmować wielu użytkowników. Dodatkowo zawiera pole isHidden które pomaga ukryć konwersację jeżeli któryś użytkownik zablokował albo usunął polubienie drugiego użytkownika.
- **Messages:** Tabela zawierająca pola wyłącznie jednej konwersacji, posiada treść wiadomości wysłanej przez użytkownika, dokładnej daty i godziny.
- **Match:** Tabela przechowująca polubienia, super polubienia, niepolubienia oraz identyfikatory obu użytkowników.

5.5 Przypadki użycia

Niezalogowani użytkownicy są ograniczeni do dwóch podstawowych opcji: mogą zalogować się na istniejące konto lub utworzyć nowe. Nie posiadają dostępu do pełnej funkcjonalności aplikacji i są zmuszeni do zalogowania się lub rejestracji. Jednak po utworzeniu konta i zalogowaniu, użytkownik przechodzi do pełnej palety funkcji. Po zalogowaniu użytkownik ma możliwość przeglądania profili innych użytkowników. System umożliwia filtrowanie wyników według preferencji, takich jak wiek, odległość i płeć. Co więcej, użytkownik może wyrażać swoje zainteresowanie poprzez polubienia lub super polubienia. Polubienie lub super polubienie jest kluczowe, ponieważ otwiera potencjalną możliwość nawiązania czatu z drugą osobą, pod warunkiem, że zainteresowanie jest odwzajemnione. Natomiast niepolubienie odrzuca danego użytkownika, który nie będzie już widoczny do czasu cyklu resetowania, obsługiwanego przez serwis backendowy pracujący w tle. Po utworzeniu połączenia między użytkownikami, otrzymują oni dostęp do funkcji czatu, gdzie mogą wymieniać się wiadomościami z innymi użytkownikami. Znajdujące się tam dedykowane menu emotikonów pozwala na dodawanie większych emocji do wiadomości, co może zwiększyć komunikatywność i przyjemność korzystania z aplikacji. Z poziomu widoku czatu, użytkownik ma także możliwość usunięcia pary (jeśli czat nie spełnia oczekiwania) lub całkowitego zablokowania danego użytkownika, co gwarantuje elastyczne zarządzanie relacjami online. W sekcji ustawień użytkownik ma kontrolę nad swoimi preferencjami filtrowania, umożliwiając dostosowanie wyników wyszukiwania do indywidualnych upodobań. Ponadto, użytkownik może edytować swoje dane wyświetlane na profilu oraz śledzić, jak wygląda jego profil z perspektywy innych użytkowników. Całość tego złożonego procesu ma na celu stworzenie interaktywnego, spersonalizowanego środowiska, w którym użytkownik ma pełną kontrolę nad swoim doświadczeniem w aplikacji. Opisane wcześniej elementy zostały zilustrowane w diagramie przypadków użycia, widocznym na rysunku 5.7. Ta graficzna reprezentacja ukazuje funkcje i dostępne możliwości dla różnych aktorów na platformie, obejmując różne scenariusze korzystania z aplikacji, zależnie od roli danego użytkownika.



Rysunek 5.7: Schemat przypadków użycia.

Rozdział 6

Weryfikacja i walidacja

6.1 Sposób testowania

Podczas etapu tworzenia nowych funkcjonalności, skupiono się na aktywnym debugowaniu kodu. Ten proces umożliwił wykrywanie i naprawę ewentualnych problemów już na etapie rozwoju. W procesie testowania aplikacji webowej zastosowano dwustopniowy podejście. Przeprowadzono również szczegółowe przeklikanie interfejsu, skupiając się na każdym jego aspekcie. Ten krok pozwolił na wykrycie potencjalnych nieprawidłowości w interakcji z użytkownikiem i zoptymalizowanie funkcji.

6.2 Wykryte i usunięte błędy

Jeżeli użytkownik posiadał więcej niż jeden czat, to niezależnie od tego, z którego z nich otrzymywał nową wiadomość, ta automatycznie pojawiała się w aktualnie otwartym czacie. Taka funkcjonalność mogła prowadzić do dezorganizacji i utrudniać skutecną obsługę wielu konwersacji jednocześnie.

Po przejściu do ustawień i powrocie na główną stronę, zapytanie o nowego użytkownika otrzymywało identyfikator osoby widocznej przed wejściem do ustawień. To było błędne zachowanie aplikacji. Taka sytuacja mogła prowadzić do mylących informacji i utrudniać poprawne odwzorowanie preferencji użytkownika. Poprawienie tego błędu było kluczowe dla zachowania spójności informacji i poprawnego funkcjonowania funkcji dotyczących użytkowników w aplikacji.

Po utworzeniu nowego czatu, użytkownicy nie otrzymywali powiadomienia, a nowy czat stawał się widoczny dopiero po odświeżeniu strony. Taka sytuacja mogła prowadzić do opóźnień w komunikacji i utrudniać użytkownikom natychmiastowy dostęp do nowych czatów. Poprawa tej funkcji była istotna dla zapewnienia płynności interakcji i natychmiastowej świadomości użytkowników o nowych wiadomościach czy czatach.

Podczas tworzenia nowego czatu, w bazie danych na odwrót zapisywało się, które

polubienie użytkownicy wzajemnie sobie dali. Taka odwrotna logika wprowadzała błędy w przechowywaniu informacji o polubieniach, co mogło prowadzić do dezaktualizacji danych i nieprawidłowego odzwierciedlenia relacji między użytkownikami. Poprawa tego procesu była kluczowa dla utrzymania spójności danych i prawidłowego funkcjonowania funkcji związanych z ocenami i relacjami między użytkownikami.

Rozdział 7

Podsumowanie i wnioski

7.1 Kierunki dalszej rozbudowy

7.1.1 Zaawansowany algorytm dopasowywania

Taki zaawansowany algorytm mógłby być używany do precyzyjnego doboru profili użytkowników, uwzględniając ich zainteresowania, styl życia i preferencje. Taka innowacyjna funkcjonalność miałaby na celu stworzenie bardziej satysfakcjonującego doświadczenia w wyszukiwaniu potencjalnych przyjaciół lub partnerów. Algorytm mógłby analizować wszystkie aspekty profilu użytkownika i na podstawie tych danych system mógłby identyfikować wspólne zainteresowania i wartości, co pozwoliłoby na bardziej trafne dopasowania między użytkownikami. Dodatkowo, taki zaawansowany algorytm mógłby uwzględnić ewentualne preferencje użytkowników dotyczące poziomu prywatności czy stopnia otwartości na nowe relacje. To podejście umożliwiałoby bardziej spersonalizowane i zindywidualizowane doświadczenia.

7.1.2 Odblokowywanie zablokowanych użytkowników

Wprowadzenie opcji odblokowywania zablokowanych profili stanowi innowacyjny krok w usprawnianiu kontroli nad interakcjami między użytkownikami. Ta nowa funkcja pozwoliłaby użytkownikom na ponowne skontaktowanie się z osobami, które zostały wcześniej zablokowane, nadając im większą elastyczność w zarządzaniu swoimi relacjami online. Dzięki możliwości odblokowania wcześniej zablokowanych profili, użytkownicy zyskują możliwość ponownego nawiązania kontaktu, rozwiązania konfliktów czy nawet otwarcia się na nowe perspektywy. Ta funkcja nie tylko umożliwia elastyczniejsze zarządzanie interakcjami, ale także sprzyja budowaniu zdrowszego środowiska online, wspierając dialog i pojednanie między użytkownikami. Odblokowywanie profili staje się zatem narzędziem, które pozwala na skuteczniejszą komunikację i lepsze zrozumienie w świecie wirtualnych relacji.

7.1.3 System płatnych usług

Wprowadzenie systemu płatnych usług, który umożliwi użytkownikom zakup subskrypcji, eliminując limity w korzystaniu z funkcji takich jak odświeżanie ilości polubień i super polubień. Dodatkowo dodanie funkcji umożliwiającej powrót do poprzedniego profilu, umożliwiając bezpłatne korzystanie z tej opcji dla jeszcze nieokreślonej ilości profili, a następnie wprowadzenie opłaty za dodatkowe możliwości. Ta funkcja zapewni użytkownikom większą swobodę w przeglądaniu profili.

Kolejnym elementem byłaby funkcja pokazywania innej osobie, co w jej profilu podoba się najbardziej. Ta opcja ma na celu stworzenie dodatkowego interaktywnego elementu, który pozwala użytkownikom na wyrażenie swojego zainteresowania w sposób bardziej szczegółowy i spersonalizowany. Ten model biznesowy nie tylko generuje dodatkowe przychody, ale także zwiększa zaangażowanie użytkowników w korzystanie z aplikacji.

7.1.4 Bezpieczeństwo

Wprowadzenie szeregu kluczowych funkcji bezpieczeństwa w aplikacji jest niezwykle istotne w kontekście zapewnienia większego bezpieczeństwa oraz ochrony prywatności użytkowników. Te elementy stanowią integralną część strategii zabezpieczającej konto użytkownika. Poniżej znajduje się opis planowanych do wdrożenia, kluczowych funkcji:

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart)¹, jest rozwiązaniem mającym na celu zabezpieczenie aplikacji przed automatycznymi atakami botów. Wdrażając CAPTCHA, aplikacja wymusza na użytkownikach rozwiązanie krótkiego zadania, które jest łatwe dla ludzi, ale trudne do zautomatyzowania przez programy komputerowe. Dzięki temu mechanizmowi, aplikacja może skutecznie identyfikować, czy osoba korzystająca z niej jest prawdziwym użytkownikiem, czy może automatycznym programem, co z kolei minimalizuje ryzyko spamu, fałszywych rejestracji oraz innych potencjalnie szkodliwych działań.

Funkcja potwierdzania adresu e-mail stanowi kluczowy krok w procesie weryfikacji tożsamości użytkownika. Po zarejestrowaniu się w aplikacji, użytkownik otrzymuje wiadomość e-mail z unikalnym linkiem aktywacyjnym. Kliknięcie w ten link potwierdza prawdziwość podanego adresu e-mail, co przeciwdziała tworzeniu fałszywych kont i chroni przed niepożądanym zachowaniem.

Możliwość resetowania hasła jest kluczowa dla bezpieczeństwa konta użytkownika. W przypadku zapomnienia hasła lub potrzeby jego zmiany z powodów bezpieczeństwa, użytkownik może skorzystać z opcji resetowania hasła. Aby potwierdzić tożsamość, aplikacja wyśle link resetujący na zarejestrowany adres e-mail. Ten proces zabezpiecza

¹<https://support.google.com/a/answer/1217728?hl=pl>

konto przed nieuprawnionym dostępem, a także umożliwia użytkownikowi odzyskanie dostępu do swojego konta w sytuacji utraty hasła.

Wdrożenie tych kluczowych funkcji bezpieczeństwa nie tylko minimalizuje ryzyko ataków, ale także buduje zaufanie użytkowników do aplikacji. Użytkownicy mają pewność, że ich dane są odpowiednio chronione, a procesy związane z rejestracją i zarządzaniem kontem są bezpieczne. Dzięki temu podejściu, aplikacja staje się bardziej niezawodna i atrakcyjna dla użytkowników, co wpływa pozytywnie na ich doświadczenie korzystania z niej.

7.2 Napotkane problemy

7.2.1 Problemy przy tworzeniu frontendu

Natychmiastowe logowanie użytkownika po zakończonej rejestracji generowało błędy, ponieważ, szybkość tej akcji powodowała, że logujący się użytkownik nie istniał jeszcze w bazie danych, co generowało błędy i utrudniało płynne przejście między etapem rejestracji, a zalogowaniem. Poprawa tego procesu jest kluczowa dla zapewnienia płynnej nawigacji użytkownika oraz uniknięcia problemów związanych z niepełnym zapisaniem danych w bazie po rejestracji.

Nieprzemyślana struktura frontendu skomplikowała proces tworzenia responsywnego interfejsu. Konieczność wprowadzenia zmian do stworzenia responsywności wymagałaby praktycznie przepisania większości kodu od nowa. Optymalizacja struktury frontendu jest kluczowa, aby ułatwić dostosowywanie interfejsu do różnych urządzeń i minimalizować konieczność przepisywania dużej części kodu przy wprowadzaniu zmian.

Brak umiejętności graficznych utrudniał projektowanie atrakcyjnego wyglądu aplikacji i wybór odpowiednich kolorów strony. Estetyka jest kluczowa dla przyciągania użytkowników, co stanowiło istotny problem. Wprowadzenie narzędzi graficznych lub współpraca z profesjonalistami mogłaby poprawić wizualny wygląd aplikacji, zwiększając jej atrakcyjność.

7.2.2 Problemy przy tworzeniu backendu

Trudności w uzyskaniu dostępu do nagłówka zapytania uniemożliwiły wyciągnięcie tokenu użytkownika. Brak tej informacji wpłynąłby negatywnie na zdolność identyfikacji użytkownika, co jest kluczowym elementem w procesie autentykacji. Skuteczny dostęp do nagłówka zapytania jest niezbędny do prawidłowej weryfikacji tożsamości użytkownika i zapewnienia bezpieczeństwa procesu autentykacji.

Utworzenie nowego serwisu pracującego w tle i wykonującego okresowe zadania było bardziej problematyczne niż na początku zakładano. Dostępne w internecie gotowe rozwiązania nie spełniały oczekiwania, co wymagało dodatkowego czasu na znalezienie

skutecznego i niezawodnego podejścia do tego problemu. Osłabiona dostępność adekwatnych rozwiązań online skomplikowała proces implementacji serwisu do obsługi zadań w tle.

Mimo że pisanie WebSocketów w bibliotece SignalR nie było bardzo skomplikowane, różnice w stosunku do tradycyjnych zapytań wprowadziły pewne trudności. Konieczność dostosowania się do specyfiki tej biblioteki wymagała dodatkowego wysiłku i eksperymentacji. Integracja z WebSocketami w SignalR, wymagała bardziej zindywidualizowanego podejścia, co przekładało się na potrzebę dogłębnego zrozumienia specyfiki tej technologii i dostosowania istniejącego kodu.

Bibliografia

- [1] Iga Rybarczyk i Tytus Koweszko. „Post-traumatic stress disorder, suicide risk, feeling of loneliness and life satisfaction in the general population during the COVID-19 pandemic”. W: *Psychiatria* 19.3 (2022), s. 183 –193. ISSN: 1733-4594. DOI: {}. URL: <https://journals.viamedica.pl/psychiatria/article/view/PSYCH.a2021.0044>.
- [2] Adam Briggle. „Real friends: how the Internet can foster friendship”. W: *Ethics and Information Technology* 10.1 (2008), s. 71–79. ISSN: 1572-8439. DOI: 10.1007/s10676-008-9160-z. URL: <https://doi.org/10.1007/s10676-008-9160-z>.
- [3] *Tinder*. 2024. URL: <https://tinder.com/> (term. wiz. 04.01.2024).
- [4] *Grindr*. 2024. URL: <https://www.grindr.com/> (term. wiz. 04.01.2024).
- [5] *Badoo*. 2024. URL: <https://badoo.com/> (term. wiz. 04.01.2024).
- [6] *Hinge*. 2024. URL: <https://hinge.co/> (term. wiz. 04.01.2024).
- [7] Aristeidis Bampakos i Pablo Deleman. 2023.
- [8] Lamis Chebbi. 2022.
- [9] Valerio De Sanctis. 2023.
- [10] Maxim Koretskyi. *Everything you need to know about the ‘ExpressionChangedAfterItHasBeenCheckedError’ error*. 2017. URL: <https://hackernoon.com/everything-you-need-to-know-about-the-expressionchangedafterithasbeencheckederror-error-e3fd9ce7dbb4> (term. wiz. 01.07.2017).
- [11] Mateusz Dobrowolski. *RxJS w Angularze – wiedza w pigulce*. 2022. URL: <https://www.angular.love/2022/01/25/rxjs-w-angularze-wiedza-w-pigulce/> (term. wiz. 25.01.2022).
- [12] Aqeel Abbas. *Understanding Angular Lifecycle Hooks with Examples*. 2023. URL: <https://medium.com/@aqeelabbas3972/understanding-angular-lifecycle-hooks-with-examples-472814158836> (term. wiz. 04.09.2023).
- [13] Adam Freeman. *Angular. Profesjonalne techniki programowania. Wydanie IV*. Gliwice: Helion, 2020. ISBN: 978-83-283-7543-7.

- [14] Joel Murach. *Murach's Asp.net Core Mvc.* Fresno, CA: Mike Murach i Associates, Inc., 2022. ISBN: 978-1943873029.

Spis skrótów i symboli

JWT Token sieciowy JSON (ang. *JSON Web Token*)

JSON Notacja obiektów JavaScript (ang. *JavaScript Object Notation*)

HTTP Protokół transferu hipertekstu (ang. *Hypertext Transfer Protocol*)

SCSS Syntaktyczne niesamowity arkusz stylów (ang. *Syntactically Awesome Stylesheet*)

URL Ujednolicony lokalizator zasobów (ang. *Uniform Resource Locator*)

RxJS Reaktywne rozszerzenia dla JavaScriptu (ang. *Reactive Extensions for JavaScript*)

REST Reprezentacyjny transfer stanu (ang. *Representational state transfer*)

API Interfejs programowania aplikacji (ang. *Application programming interface*)

CAPTCHA Całkowicie zautomatyzowany publiczny test Turinga pozwalający odróżnić komputery od ludzi (ang. *Completely Automated Public Turing test to tell Computers and Humans Apart*)

Dodatki

Źródła

```
1  @Injectable()
2  export class RefreshTokenInterceptor implements
3      HttpInterceptor {
4
5      private isRefreshing = false;
6      private refreshTokenSubject: BehaviorSubject<string> = new
7          BehaviorSubject<string>(' ');
8
9      constructor(private userService: UserService) { }
10
11     intercept(request: HttpRequest<unknown>, next: HttpHandler
12         ): Observable<HttpEvent<unknown>> {
13
14         if (this.userService.isTokenExpired())
15             this.userService.logout();
16
17         return next.handle(request).pipe(
18             catchError((error: HttpErrorResponse) => {
19                 if (error instanceof HttpErrorResponse && error.
20                     status === ErrorStatus.UNAUTHORIZED) {
21                     return this.handle401Error(request, next);
22                 } else {
23                     return throwError(error);
24                 }
25             })
26         );
27     }
28 }
```

Rysunek 1: Interceptor odświeżający token cz.1.

```
1  private handle401Error(request: HttpRequest<unknown>, next
2    : HttpHandler) {
3    if (!this.isRefreshing) {
4      this.isRefreshing = true;
5      this.refreshTokenSubject.next(' ');
6
7      return this.userService.refreshToken().pipe(
8        switchMap((result) => {
9          this.isRefreshing = false;
10         this.refreshTokenSubject.next(result.accessToken);
11         return next.handle(this.addToken(request, result.
12           accessToken));
13       })
14     );
15   }
16
17   return this.refreshTokenSubject.pipe(
18     filter(token => token !== ''),
19     take(1),
20     switchMap((token) => {
21       return next.handle(this.addToken(request, token));
22     })
23   );
24
25   private addToken(request: HttpRequest<unknown>, token:
26     string): HttpRequest<unknown> {
27     return request.clone({
28       setHeaders: {
29         'Authorization': 'bearer ${token}',
30       }
31     });
32   }
33 }
```

Rysunek 2: Interceptor odświeżający token cz.2.

```

1  @Injectable({
2      providedIn: 'root',
3  })
4  export class MessengerService {
5
6      private hubConnection: signalR.HubConnection;
7      private authorId: string;
8      private authorName: string;
9      private receiverSet = false;
10
11     private isConnected$ = new BehaviorSubject<boolean>(false)
12         ;
13
14     constructor(
15         private userService: UserService,
16         private userChatService: UserChatService,
17     ) { }
18
19     buildConnection(): void {
20         this.hubConnection = new HubConnectionBuilder()
21             .configureLogging(LogLevel.None).withUrl('http://'
22                 localhost:5207/chat').build();
23
24         this.authorId = this.userService.userAuthorization().user
25             id;
26         this.authorName = this.userService.userAuthorization().name;
27
28         this.startConnection();
29     }
30
31     //Wyjście z zakładki czatu
32     leaveChatTab() {
33         this.hubConnection.invoke("LeaveChat", this.authorId)
34             .catch((err) => console.log(err));
35     }
36
37     //Wysyłanie wiadomości
38     sendMessage(message: string, chatId: number) {
39         this.hubConnection.invoke("SendMessage", chatId, this.
40             authorId, this.authorName, message)
41             .catch((err) => console.log(err));
42     }

```

Rysunek 3: Serwis przetwarzający polecenia WebSocketu cz.1.

```
1  //Historia chatów
2  restoreAllConversations() {
3      this.hubConnection.invoke("RestoreAllConversations",
4          this.authorId)
5          .catch((err) => console.log(err));
6
7  //Przywrócenie wszystkich wiadomości z danego chatu
8  restoreMessages(chatId: number) {
9      this.hubConnection.invoke("RestoreMessages", chatId,
10         this.authorId)
11        .catch((err) => console.log(err));
12
13  get isConnectedObservable(): Observable<boolean> {
14      return this.isConnected$.asObservable();
15  }
16
17  get isConnectedRaw(): boolean {
18      return this.isConnected$.value;
19  }
20
21  private startConnection() {
22      this.hubConnection.start()
23          .then(() => {
24              this.hubConnection.invoke("Join", this.authorId)
25                  .then(() => this.isConnected$.next(true))
26                  .catch((err) => console.log(err));
27          })
28          .catch((err) => console.log(err));
29      if (!this.receiverSet)
30          this.setReceiver();
31  }
32
33  private setReceiver(): void {
34      this.receiverSet = true;
35
36      //Otrzymanie wiadomości
37      this.hubConnection.on("ReceiveMessage", (chatId: number,
38          authorId: number, authorName: string, date: Date,
39          content: string) => {
40          this.userChatService.setMessageStream({ chatId,
41              authorId, authorName, date, content,
42              fromCurrentUser: authorId === Number(this.authorId)
43          });
44      });
45  }
```

Rysunek 4: Serwis przetwarzający polecenia WebSocketu cz.2.

```
1 //Przywrócenie wszystkich konwersacji pojedynczo
2 this.hubConnection.on("ReceiveConversation", (id: number
3     , withUser: string , withUserId: number , date: Date ,
4     content: string ,
5     isRead: boolean , userImage: string , isSuperLiked:
6         boolean) => {
7     this.userChatService.setPreviousChatsStream({ id ,
8         withUser , withUserId , date , content , isRead ,
9         userImage , isSuperLiked });
10    });
11
12
13
14
15
16
17
18 //Informacja o nowej konwersacji
19 this.hubConnection.on("ConversationCreated", (
20     firstUserId: number , firstUserName: string ,
21     firstUserImage: string ,
22     firstUserSuperLiked: boolean , secondUserId: number ,
23     secondUserName: string , secondUserImage: string ,
24     secondUserSuperLiked: boolean ) => {
25     this.userChatService.setNewConversationCreated({
26         firstUserId , firstUserName , firstUserImage ,
27         firstUserSuperLiked , secondUserId , secondUserName ,
28         secondUserImage , secondUserSuperLiked
29     });
30    });
31
32
33
34
35
36
37
38 //Unmatch albo zablokowanie
39 this.hubConnection.on("ConversationToHide", (chatId:
40     number) => {
41     this.userChatService.setConversationToHide(chatId);
42    });
43 }
```

Rysunek 5: Serwis przetwarzający polecenia WebSocketu cz.3.

```
1  @Injectable({
2    providedIn: 'root',
3  })
4  export class ControllerService {
5
6    //Dane użytkownika przenoszone z rejestracji do input'ów w logowaniu
7    private userAuthentication$ = new BehaviorSubject<
8      UserAuthentication>(null);
9
10   set userAuthentication(userAuthentication: UserAuthentication) {
11     this.userAuthentication$.next(userAuthentication);
12   }
13
14   get userAuthentication(): UserAuthentication {
15     return this.userAuthentication$.value;
16   }
17
18   //Cache'owanie danych(zdjęcie, imię)
19   private cachedUserBasicInfo$ = new BehaviorSubject<
20     UserBasicData>(null);
21
22   set cachedUserBasicInfo(cachedUserBasicInfo: UserBasicData) {
23     this.cachedUserBasicInfo$.next(cachedUserBasicInfo);
24   }
25
26   get cachedUserBasicInfo(): UserBasicData {
27     return this.cachedUserBasicInfo$.value;
28   }
29
30   //Cache'owanie danych(cały user)
31   private cachedUserMainInfo$ = new BehaviorSubject<Match>(
32     null);
33
34   set cachedUserMainInfo(cachedUserMainInfo: Match) {
35     this.cachedUserMainInfo$.next(cachedUserMainInfo);
36   }
37
38   get cachedUserMainInfo(): Match {
39     return this.cachedUserMainInfo$.value;
40   }
```

Rysunek 6: Serwis do zarządzania stanem aplikacji cz.1.

```
1 //Cache'owanie danych(dane na lewym pasku oprócz header'a)
2 private cachedUserSideInfo$ = new BehaviorSubject<SideData>(null);
3
4     set cachedUserSideInfo(cachedUserSideInfo: SideData) {
5         this.cachedUserSideInfo$.next(cachedUserSideInfo);
6     }
7
8     get cachedUserSideInfo(): SideData {
9         return this.cachedUserSideInfo$.value;
10    }
11
12    getCachedUserSideInfoObservable(): Observable<SideData> {
13        return this.cachedUserSideInfo$.asObservable();
14    }
15
16    //Listener do odświeżania zdjęcia(w header) po aktualizacji danych aktualnego
17    // użytkownika
18    private refreshImage$ = new BehaviorSubject<void>(null);
19
20        refreshImage(): void {
21            this.refreshImage$.next();
22        }
23
24        refreshImageObservable(): Observable<void> {
25            return this.refreshImage$.asObservable();
26        }
27
28    //Wyczyszczenie chatu(input'a) po zmianie na inny chat
29    private clearChatInput$ = new BehaviorSubject<void>(null);
30
31        setClearChat(): void {
32            this.clearChatInput$.next();
33        }
34
35        getClearChatObservable(): Observable<void> {
36            return this.clearChatInput$.asObservable();
37        }
```

Rysunek 7: Serwis do zarządzania stanem aplikacji cz.2.

```
1 //Jeżeli są dane to pokazuje się chat, ten subject posiada Id chatu i
2     isSuperLiked
3     private currentChatData$ = new BehaviorSubject<ChatData>(
4         null);
5
6     setCurrentChatData(value: ChatData): void {
7         this.currentChatData$.next(value);
8     }
9
10    getCurrentChatDataObservable(): Observable<ChatData> {
11        return this.currentChatData$.asObservable();
12    }
13
14    getCurrentChatDataRaw(): ChatData {
15        return this.currentChatData$?.value;
16    }
17
18 //Ustawia aktualną lokalizację użytkownika(jeżeli wyraził zgodę)
19     private userLocation$ = new BehaviorSubject<Location>(null
20         );
21
22     setUserLocation(value: Location): void {
23         this.userLocation$.next(value);
24     }
25
26     getUserLocationRaw(): Location {
27         return this.userLocation$.value;
28     }
29
30 //Zapisuje Id match'a który jest aktualnie widoczny
31     private currentMatchId$ = new BehaviorSubject<number>(null
32         );
33
34     setCurrentMatchId(value: number): void {
35         this.currentMatchId$.next(value);
36     }
37 }
```

Rysunek 8: Serwis do zarządzania stanem aplikacji cz.3.

```
1  public sealed class ClearDislikesWorkerService :
2      BackgroundService
3  {
4      private readonly IServiceProvider _serviceProvider;
5
6      public ClearDislikesWorkerService(IServiceProvider
7          serviceProvider)
8      {
9          _serviceProvider = serviceProvider;
10     }
11
12     protected override async Task ExecuteAsync(
13         CancellationToken stoppingToken)
14     {
15         while (!stoppingToken.IsCancellationRequested)
16         {
17             using var scope = _serviceProvider.CreateScope()
18                 ;
19             var dbContext = scope.ServiceProvider.
20                 GetRequiredService<AppDbContext>();
21
22             var matches = dbContext
23                 .Match
24                 .Where(u => u.Status == LikeStatus.Dislike)
25                 .ToList();
26
27             var users = dbContext.Users.ToList();
28
29             foreach (var user in users)
30             {
31                 user.LikesLeft = 30;
32                 user.SuperLikesLeft = 2;
33             }
34             dbContext.Match.RemoveRange(matches);
35             dbContext.SaveChanges();
36
37             await Task.Delay(TimeSpan.FromHours(4),
38                             stoppingToken);
39         }
40     }
41 }
```

Rysunek 9: Serwis wywołujący się co określony czas.

Spis rysunków

2.1 a Tinder. [3], b Grindr. [4]	6
2.2 a Badoo. [5], b Hinge. [6]	7
4.1 Lokalizacja użytkownika.	14
4.2 Główna strona.	15
4.3 Strona ustawień.	15
4.4 Strona logowania.	16
4.5 Pierwsza strona rejestracji.	16
4.6 Dodanie zdjęć.	17
4.7 Określenie nastawienia użytkownika.	17
4.8 Wybór orientacji seksualnej.	18
4.9 Wybór zainteresowań.	18
4.10 Dodatkowe informacje o użytkowniku.	19
4.11 Więcej informacji cz.1.	20
4.12 Więcej informacji cz.2.	20
4.13 Więcej informacji aktualnego użytkownika.	21
4.14 Edycja profilu cz.1.	21
4.15 Edycja profilu cz.2.	22
4.16 Edycja profilu cz.3.	22
4.17 Czat zamknięty, widok użytkownika Natalia.	23
4.18 Czat otwarty, widok użytkownika Filip.	23
4.19 Czat, emotikony.	24
4.20 Czat, blokowanie i usuwanie pary.	25
4.21 Informacja o braku polubień.	26
4.22 Informacja o braku super polubień.	26
4.23 Informacja o nowej konwersacji, widok użytkownika Natalia.	27
4.24 Informacja o nowej konwersacji, widok użytkownika Filip.	27
5.1 Interceptor autoryzacyjny.	30
5.2 Guard blokujący stronę główną.	31
5.3 Guard blokujący stronę logowania.	31

5.4	Klasa przechowująca id użytkownika.	32
5.5	Obliczanie odległości.	33
5.6	Schemat bazy danych.	34
5.7	Schemat przypadków użycia.	37
1	Interceptor odświeżający token cz.1.	51
2	Interceptor odświeżający token cz.2.	52
3	Serwis przetwarzający polecenia WebSocketu cz.1.	53
4	Serwis przetwarzający polecenia WebSocketu cz.2.	54
5	Serwis przetwarzający polecenia WebSocketu cz.3.	55
6	Serwis do zarządzania stanem aplikacji cz.1.	56
7	Serwis do zarządzania stanem aplikacji cz.2.	57
8	Serwis do zarządzania stanem aplikacji cz.3.	58
9	Serwis wywołujący się co określony czas.	59