Spis treści

1. Wstęp

Wraz z rozwojem technologii, tworzenie aplikacji mobilnych staje się co raz szybsze i prostsze. W dzisiejszych czasach, w krajach rozwiniętych, z Internetu korzysta powyżej 80% społeczeństwa[1]. Co więcej, średnia czasu spędzana przez użytkownika urządzenia mobilnego w sieci to aż 2.8 godziny dziennie[2]. Internet swoją popularnością przewyższył najpopularniejszy dotychczas kanał informacyjny – telewizję, a media społecznościowe typu Facebook[3], Instagram[4] czy Twitter[5] wyznaczają dzisiejsze trendy oraz kształtują młodsze pokolenia. Dana sytuacja daje nam możliwość rozwiązywania problemów społecznych nowoczesnymi metodami, między innymi dzięki aplikacjom mobilnym.

Problemem, który porusza dana praca jest zanikanie relacji międzyludzkich na rzecz rozwoju technologicznego[6]. Jej dynamiczny rozwój powoduje, że większość użytkowników rezygnuje ze spotkań w świecie realnym, zamieniając je na godziny spędzane przed urządzeniami elektronicznymi. Ludzie powoli odzwyczajają się od zwykłych rozmów, a postawieni w sytuacji bezpośredniego kontaktu z drugim człowiekiem czują się niekomfortowo. Dany problem wydaje się pogłębiać wraz z wynajdywaniem nowych, co raz bardziej odzwierciedlających rzeczywistość rozwiązań technologicznych.

Daną sytuację można jednak rozwiązać również przy pomocy technik internetowych[7]. Jednym z przykładów istniejących tego typu metod jest Tinder[8] – popularna aplikacja mająca na celu ułatwianie spotkań. Problemem, który istnieje we wspomnianym rozwiązaniu jest fakt, że pary dobierane są tylko na podstawie wyglądu[9]. Osoby połączone w ten sposób często doznają zawodów powodowanych brakiem wspólnych tematów do rozmów. Sprawia to, że kończą spotkania jeszcze bardziej zniechęcone do kontaktów społecznych niż przed nimi.

1.1. Cel pracy

Celem pracy jest stworzenie aplikacji mobilnej ułatwiającej wspólne spędzanie czasu użytkownikom. Dany projekt ma być alternatywą dla wyżej wymienionej platformy rozwiązującą problem braku kontekstu podczas spotkań. Najważniejszą różnicą pomiędzy danymi rozwiązaniami ma być sposób w jaki aplikacja będzie łączyła użytkowników. W przypadku opisywanego projektu, czynnikiem warunkującym dobór partnerów będzie wykonywana przez nich wspólna aktywność. Celem tego typu podejścia jest nadanie kontekstu spotkaniu i zapewnienie, że ich członków będzie łączyło coś więcej niż tylko zainteresowanie aparycją drugiego użytkownika.

1.2. Układ pracy

Praca została podzielona na 5 rozdziałów:

- Rozdział 1 zawiera motywację oraz cel pracy.
- Rozdział 2 został poświęcony specyfikacji wymagań.
- Rozdział 3 opisuję technologię użytą przy tworzeniu aplikacji.
- Rozdział 4 zawiera opis poszczególnych ekranów aplikacji.
- Rozdział 6 to podsumowanie pracy.

2. Wymagania

2.1. Funkcjonalne

Autoryzacja i uwierzytelnienie:

- Użytkownik powinien móc się uwierzytelnić.
- Użytkownik powinien móc się autoryzować.
- Do poprawnej autoryzacji użytkownik powinien podać: e-mail, login, hasło.
- Do poprawnego uwierzytelnienia użytkownik powinien podać e-mail oraz hasło.
- Jeśli po użytkowaniu aplikacji użytkownik się nie wylogował, przy następnym uruchomieniu aplikacji powinien być automatycznie zalogowany.
- Użytkownik powinien móc się uwierzytelnić używając danych z Facebooka.
- Po uruchomieniu aplikacji użytkownik powinien mieć możliwość wyboru autoryzacji lub uwierzytelnienia.
- W przypadku niepoprawnego podanego loginu lub hasła podczas uwierzytelniania, powinna zostać wyświetlona informacja o błędzie, pola tekstowe powinny zostać wyczyszczone, a użytkownik powinien pozostać na ekranie uwierzytelniania.
- Użytkownik powinien być w stanie się wylogować.

Spotkania:

- Lista pobliskich spotkań powinna być wyświetlana na ekranie głównym i powinna zawierać: nazwę, datę oraz odległość użytkownika do miejsca spotkania.
- W przypadku niewypełnienia pola wymaganego do utworzenia spotkania, użytkownikowi powinien zostać wyświetlony komunikat o błędzie.
- Po wybraniu pozycji z listy pobliskich spotkań na ekranie głównym, użytkownik powinien być przeniesiony do ekranu z jego szczegółowymi informacjami.
- Użytkownik powinien być w stanie dołączać do spotkań.
- Nowe spotkanie powinno być tworzone z ekranu głównego.
- Użytkownik powinien być w stanie przejść do listy spotkań z ekranu głównego
- Użytkownik powinien być ws tanie wyszukiwać spotkania z ekranu.
- Podczas wyświetlania ekranu ze spotkaniami na mapie, użytkownik powinien być w środka okręgu obrazującego odległość użytkownika od spotkania.
- Użytkownik powinien być w stanie regulować dany promień.
- Użytkownik powinien być w stanie wyszukiwać spotkania na mapie.
- Aby utworzyć spotkanie użytkownik powinien podać: nazwę, opis, lokalizację, promień, termin oraz maksymalną ilość uczestników spotkania.
- Użytkownik powinien być w stanie wybrać lokalizację na mapie.
- Lista spotkań powinna być podzielona na spotkania utworzone przez użytkownika i na spotkania w których uczestniczy.

- Po wybraniu pozycji z listy spotkań, użytkownikowi powinna zostać wyświetlona karta ze szczegółami spotkania.
- Użytkownik powinien być w stanie usunąć stworzone spotkanie.
- W przypadku usunięcia spotkania, jego członkowie powinni o tym zostać poinformowani powiadomieniem.
- Użytkownik powinien być w stanie opuścić spotkanie.
- W przypadku opuszczenia spotkania przez jednego z jego uczestników, organizator powinien zostać o tym poinformowany powiadomieniem.
- W przypadku, gdy nowy użytkownik dołączy do spotkania, jego założyciel powinien zostać o tym poinformowany powiadomieniem.

Pozostali użytkownicy i przyjaciele:

- Użytkownik powinien móc wyświetlić listę wszystkich użytkowników.
- Użytkownik powinien być w stanie zapraszać innych użytkowników do znajomych.
- użytkownik powinien być w stanie usuwać znajomych.
- W przypadku zaproszenia użytkownika do znajomych, powinien on zostać poinformowany o tym powiadomieniem.
- W przypadku w którym użytkownik wysłał zaproszenie do znajomych, a ono nie zostało jeszcze zaakceptowane, użytkownik powinien móc je anulować.
- Użytkownik powinien móc wyświetlić listę swoich przyjaciół.
- Po wybraniu pozycji z listy przyjaciół użytkownik powinien być przekierowany do ekranu z którego będzie mógł ją z niej usunąć lub rozpocząć konwersację.

Profil użytkownika:

- Użytkownik powinien móc edytować swój profil.
- Po utworzeniu konta użytkownikowi powinien zostać ustawiony domyślny opis oraz zdjęcie profilowe.
- Podczas edycji profilu użytkownik powinien być w stanie zmienić zdjęcie profilowe oraz opis.
- Użytkownik powinien móc edytować swój opis poprzez kliknięcie na niego na ekranie przeznaczonym do edycji profilu.
- Po wybraniu pozycji z listy wszystkich użytkowników powinien być wyświetlony dany profil.
- Użytkownik powinien móc zmienić swoje zdjęcie profilowe poprzez kliknięcie na nie na ekranie przeznaczonym do edycji profilu.
- Użytkownik powinien móc wybrać zdjęcie profilowe ze zdjęć znajdujących się w jego galerii lub na dysku Google.

Czat:

- Użytkownik powinien móc prowadzić konwersacje z osobami będącymi na liście jego przyjaciół
- W oknie konwersacji, użytkownikowi powinny być wyświetlane wszystkie poprzednie wiadomości z daną osobą.
- Użytkownik powinien być informowany o nowej wiadomości od przyjaciela tylko w przypadku, w którym przeczytał wszystkie poprzednie.
- Użytkownik powinien być w stanie wyświetlić listę swoich konwersacji.
- Lista konwersacji użytkownika powinna składać się z ostatniej wiadomości oraz loginu użytkownika z którym jest prowadzona.

2.2. Niefunkcjonalne

Ogólne:

- Interfejs użytkownika powinien być intuicyjny. Użytkownik powinien być w stanie korzystać z aplikacji bez ówczesnego zapoznania się z instrukcją.
- Aplikacja powinna działać płynnie.
- Aplikacja powinna działać na systemach Android oraz IOS
- Aplikacja powinna być zależna od wielkości ekrany użytkownika
- Działanie aplikacji powinno być niezakłócone przez aktywność innych programów na urządzeniu. W przypadku np. konieczności odebrania połączenia telefonicznego w czasie korzystania z aplikacji, użytkownik powinien móc wrócić do poprzedniego stanu po zakończonej rozmowie.
- Aplikacja powinna być łatwa w utrzymaniu.
- Aplikacja powinna być łatwa do instalacji na serwerze.
- Zmiany, w bazie danych dotyczących aplikacji, powinny być łatwe do implementacji.
- Intymne dane dotyczące użytkownika powinny być zapisywane do bazy danych dopiero po zaszyfrowaniu.
- Wszystkie tokeny identyfikujące użytkownika powinny być zapisywane dla porównania na urządzeniu lokalnym.
- W przyszłości, aplikacja powinna móc być udostępniona na Google Play Store oraz Apples's App Store
- Po zalogowaniu, użytkownik nie powinien czekać na wyświetlenie ekranu głównego dłużej niż 5 sekund.

Autoryzacja i uwierzytelnienie:

- Po poprawnym wypełnieniu formularza autoryzacji i jego zaakceptowaniu, dane o użytkowniku powinny zostać zapisane w bazie danych.
- Przechowywane w bazie danych hasło użytkownika powinno zostać zaszyfrowane.

- Token z id użytkownika powinien być przechowywany w pamięci catche urządzenia.
- Po wylogowaniu, token z id użytkownika powinien zostać usunięty z pamięci catche urządzenia.
- Długość nazwy użytkownika powinna znajdować się w zakresie od 3 do 20.
- Długość hasła użytkownika powinna znajdować się w zakresie od 5 do 30.
- Długość nazwy użytkownika powinna znajdować się w zakresie od 5 do 35.
- E-mail musi posiadać dwie części domeny np. gmail.com.
- Podczas wysyłania formularza do rejestracji należy sprawdzić czy podany przez użytkownika e-mail nie został wcześniej użyty. W przeciwnym wypadku, wyświetlić informację o konieczności podania innego adresu e-mail.
- Podczas uruchamiania aplikacji należy sprawdzić czy na urządzeniu zapisany jest token z id użytkownika. Jeśli tak, użytkownik powinien być zalogowany automatycznie, jeśli nie, użytkownik powinien być przekierowany do ekranu logowania.
- Podczas próby logowania, należy sprawdzić czy podany e-mail istnieje w bazie danych, jeśli tak, sprawdzić czy przypisane do niego hasło jest zgodne z podanym przez użytkownika.
- Po zalogowaniu, serwer powinien wysłać zapytanie do serwisu Expo z prośbą o token potrzebny do wysyłania powiadomień.

Spotkania:

- Spotkania z listy wyświetlanej na stronie głównej powinny być w odległości nie większej niż 3000 metrów od użytkownika.
- Odległość podawana na liście spotkań znajdującej się na ekranie głównym powinna być określona z przybliżeniem do jednego metra.
- Długość nazwy spotkania powinna znajdować się w zakresie od 3 do 20 znaków.
- Długość nazwy spotkania powinna znajdować się w zakresie od 5 do 40 znaków.
- Wartość pola o maksymalna liczba członków powinna być liczbą, a jej maksymalna wartość powinna wynosić 20.
- Termin spotkania powinien być wartością w formacie YYYY-MM-DD.
- Promień spotkania powinien być wartością liczbową, nie większą niż 30000.
- Przed wyświetleniem mapy, należy pobrać dokładną lokalizację użytkownika.
- Dane na temat spotkania powinny być zapisywane do bazy danych dopiero po zaakceptowaniu formularza przez użytkownika.
- Minimalna zmiana wartości suwaka określająca promień okręgu w okół użytkownika podczas wyszukiwania spotkań powinna wynosić 500 metrów. Zakres jego wartości powinien znajdować się w przedziale od 1 do 30 kilometrów.
- Wyświetlając pinezki na mapie powinniśmy pobrać ich lokalizacje z bazy danych.
- Wyświetlając użytkownikowi listę dostępnych spotkań, powinny zostać wyświetlone tylko te, których ilość członków jest mniejsza od maksymalnej.

- Po usunięciu spotkania należy zaktualizować bazę danych.
- Po opuszczeniu spotkania przez użytkownika należy zaktualizować bazę danych.
- Spotkania, których termin jest późniejszy niż obecna data powinny być usuwane z bazy danych.
- Usuwanie przedawnionych spotkań z baz danych powinno odbywać się co minutę.
- Informację wyświetlane na ekranie ze szczegółami spotkania powinny być pobierane z bazy danych na podstawie id spotkania.
-

Pozostali użytkownicy i przyjaciele:

- Lista przyjaciół użytkownika powinna składać się ze zdjęcia oraz loginu.
- Lista wszystkich użytkowników powinna składać się tylko z ich loginów.
- Ilość przyjaciół użytkownika powinna być nieograniczona.

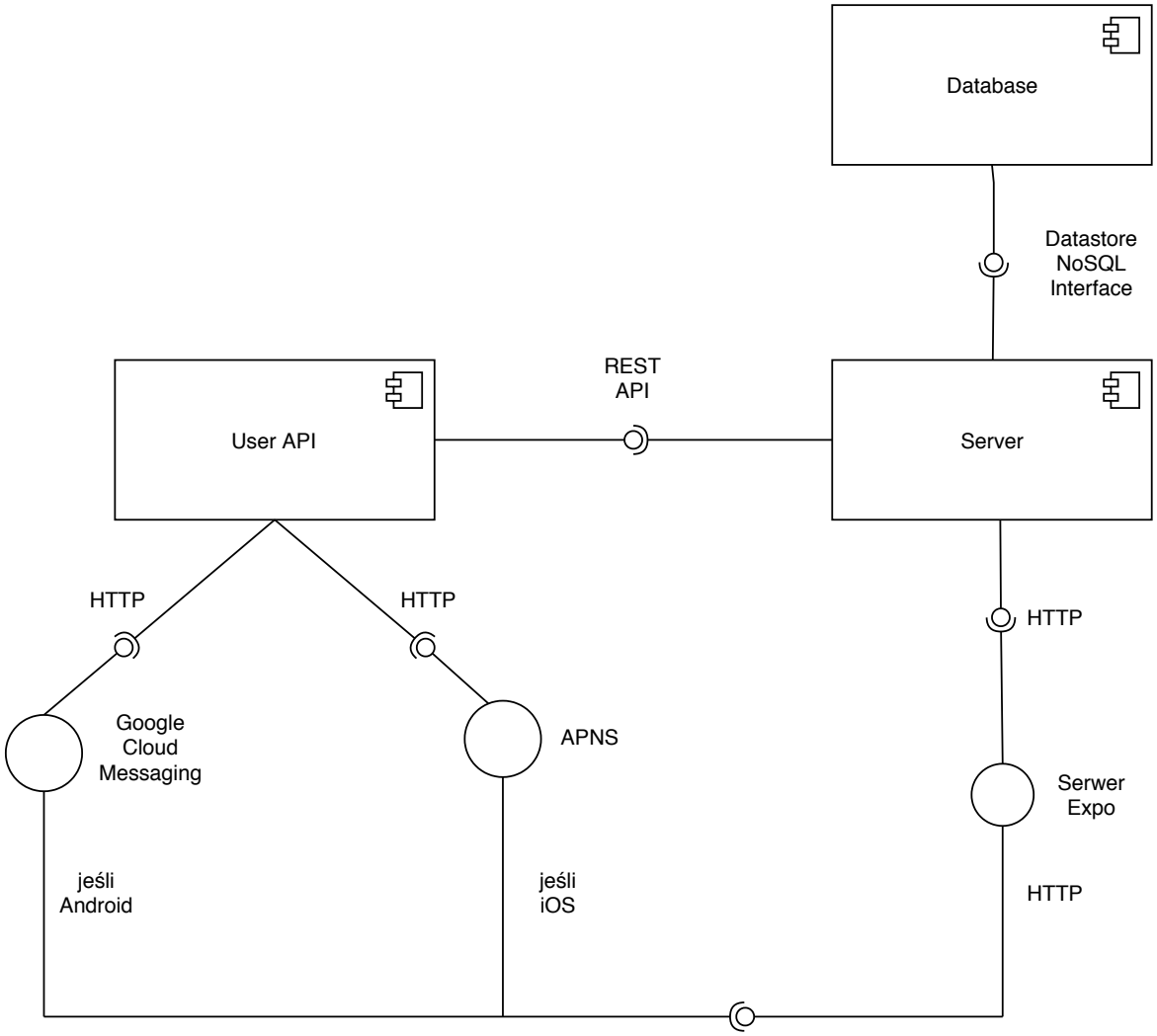
Profil użytkownika:

- Zdjęcie profilowe użytkownika powinno być wczytywane z serwera.
- Do użytkownika powinno być przypisane tylko jedno zdjęcie.
- Opis użytkownika powinien być przechowywany na serwerze.
- Długość opisu nie powinna przekraczać 30 znaków.
- Przed wysłaniem zdjęcia na serwer, jego jakość powinna być pięciokrotnie zmniejszana.

Czat:

- Wiadomości użytkownika powinny być przechowywane w bazie danych.
- Użytkownik powinien być w stanie jednoznacznie odróżnić wiadomości wysłane od odebranych.

3. Architektura aplikacji



Rysunek 1. Architektura aplikacji

Aplikacja mobilna

Ze względu na wymaganą kompatybilność z urządzeniami korzystającymi zarówno z systemów Android jak i iOS została napisana w frameworku o nazwie React Native. Jego podstawowe zasady działania są wirtualnie identyczne do zasad Reacta[10] poza faktem, że React Native nie manipuluje DOMem[11] poprzez wirtualny DOM[12]. Fakt ten sprawia, że w razie konieczności stworzenia aplikacji internetowej, większość kodu będzie mogła zostać użyta ponownie.

Wspomniany wyżej framework działa w procesie znajdującym się w tle (który interpretuje JavaScript pisany przez programistów) bezpośrednio na urządzeniu końcowym i w przypadku danej aplikacji, komunikuje się z platformami poprzez asynchroniczne zapytania.

React Native nie korzysta z HTML. Zamiast tego, do manipulowania natywnymi widokami, używane są wiadomości z wątków JavaScriptu.

Serwer Node.js

Aby aplikacja nie była zależna od platform typu Firebase[13], a jej funkcjonalności były nieograniczone, zdecydowano się napisać serwer przy wsparciu biblioteki Node.js. Aplikacja komunikuje się z nim przy użyciu protokołu HTTP, używając głównie metody GET, POST oraz DELETE. Wykorzystuje do tego architekturę oprogramowania REST[14], która minimalizuje ilość ruchu w sieci, jednocześnie maksymalizując niezależność oraz skalowalność.

HTTP

Protokół aplikacji dla rozproszonych oraz kolaboracyjnych systemów informacyjnych. HTTP jest międzynarodową podstawą komunikowania się danych, gdzie dokumenty hipertekstowe posiadają hiperlinki do innych zasobów, które są łatwo dostępna dla użytkownika.[15]

REST

Jest stylem architektury oprogramowania, który definiuje zbiór ograniczeń używanych do tworzenia serwisów internetowych. Aplikacje, które dostosowują się do tej architektury są nazywane RESTful Api i zapewniają interoperacyjność z komputerami znajdującymi się w sieci. Serwisy tego typu umożliwiają systemom wysyłającym zapytania na dostęp oraz manipulację tekstowych reprezentacji zasobów internetowych.[14]

Serwer Expo

Służy do kwalifikowania push tokenów na generowane przez urządzenia Apple lub Google. Kolejnie wysyła zapytanie do serwisu Google Cloud Messaging (w przypadku Androida) lub APNS (w przypadku iOS), które następnie wysyłają powiadomienia na urządzenia użytkowników.

Google Cloud Messaging

Bezpłatny serwis stworzony przez Google, który umożliwia deweloperom aplikacji na wysyłanie powiadomień adresowanych do systemów operacyjnych Android oraz aplikacji lub wtyczek zaprojektowanych na przeglądarki Google Chrome.[16]

APNS

Apple Push Notification service jest platformą analogiczną to Google Cloud Messaging, lecz przeznaczoną dla urządzeń stworzonych przez firmę Apple.[17]

Powiadomienia PUSH

Po poprawnej autoryzacji, urządzenie wysyła swoje ID do serwisów Google Cloud Messaging lub APNS. W odpowiedzi otrzymuje token powiadomienia. Następnie jest on wysyłany do serwera Node.js i zapisywany w bazie danych.

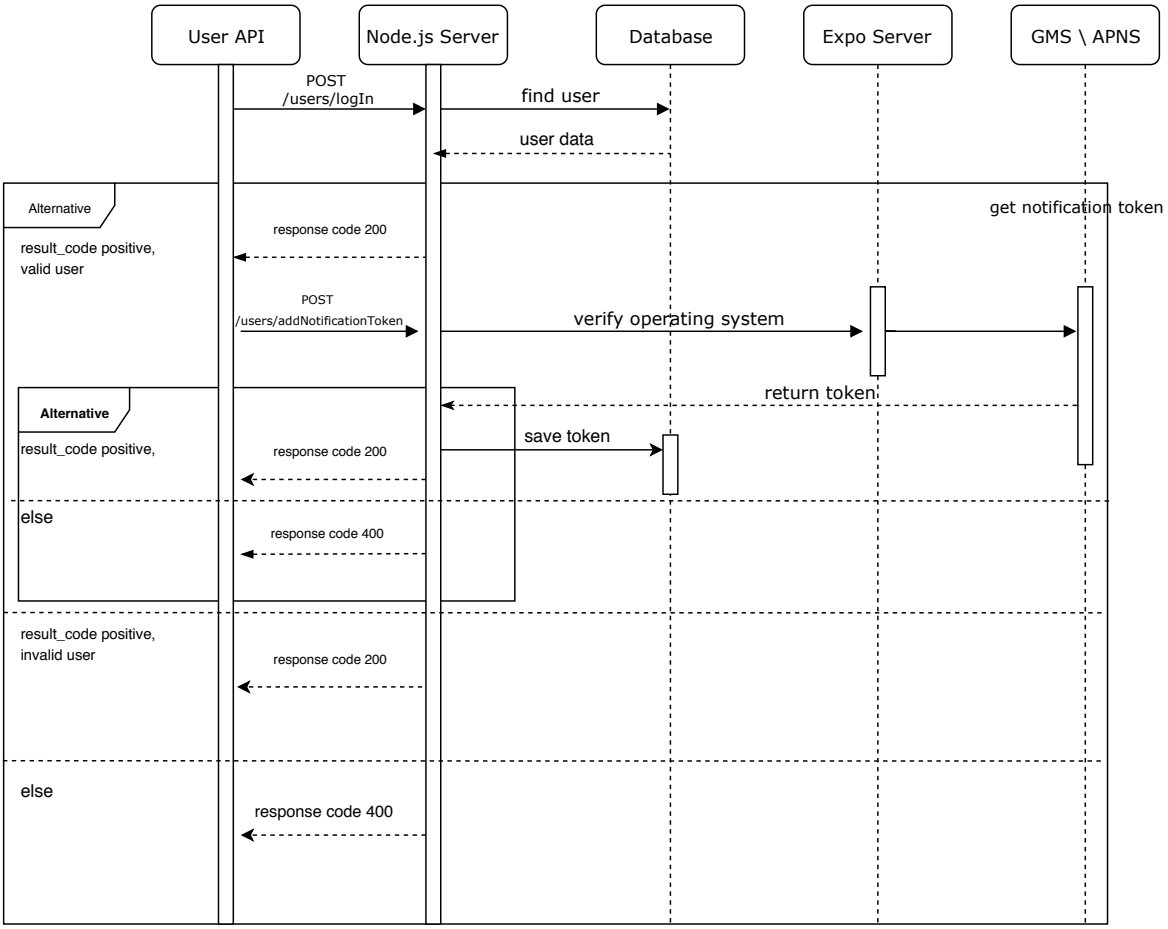
W przypadku, w którym urządzenie próbuje wysłać powiadomienie, powiązany z nim token jest pobierany z bazy danych. Kolejnym krokiem jest wysłanie danego identyfikatora do serwera Expo, który weryfikuje z jakiego oprogramowania (Android lub iOS) korzysta urządzenie wysyłające zapytanie, przekazując je odpowiednio do GCM lub APNS.

Ostatnim etapem jest przesłanie, przy pomocy protokołu HTTP, powiadomienia na urządzenie adresata. Jeśli urządzenie to w danej chwili jest nieosiągalne, powiadomienie jest dodawane do kolejki i wysyłane z opóźnieniem. Maksymalne opóźnienie w przypadku GSM to 28 dni, po tym okresie wiadomość zostaje usuwana.

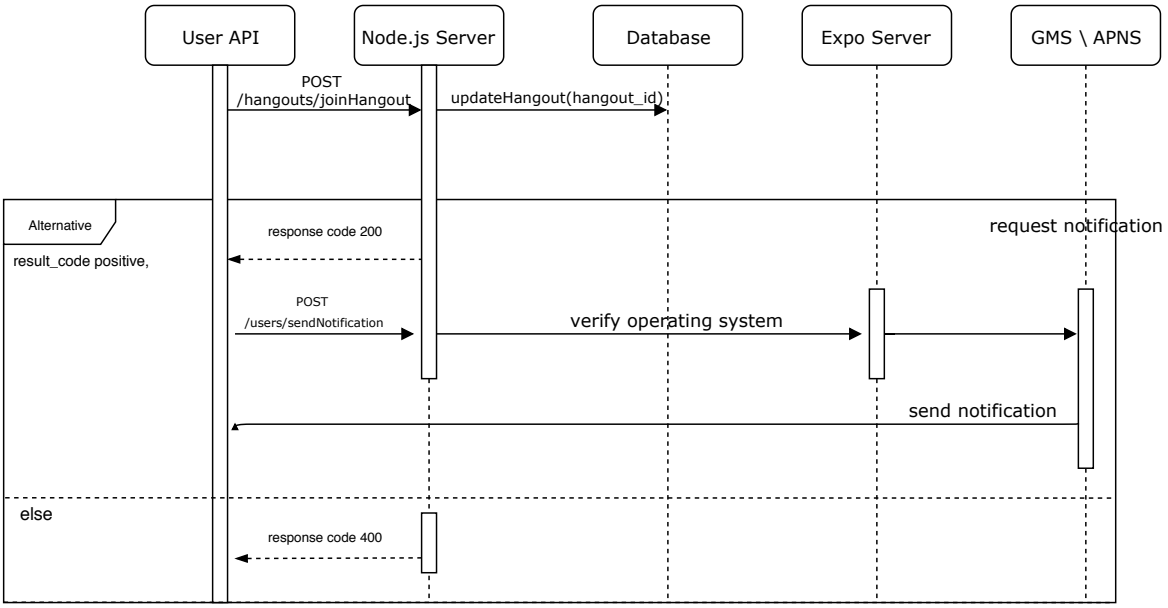
3.1. Przykładowe scenariusze użycia

Jako pierwszy został przedstawiony scenariusz logowania się przez użytkownika. Po poprawnym wypełnieniu formularza autoryzacji na urządzeniu mobilnym, na serwer przesyłane jest zapytanie, które porównuje wysłane dane ze znajdującymi się w bazie danych. W przypadku poprawnego ich podania aplikacja użytkownika komunikuję się z serwerem Expo w celu zidentyfikowania systemu operacyjnego z którego zostały wysłane. Kolejnie do serwisu GMS lub APNS wysyłane jest zapytanie z prośbą o token dla danego urządzenia. Gdy zostanie zwrócony poprawnie, token jest wysyłany na serwer i zapisywany w bazie danych. Scenariusz został przedstawiony graficznie na rysunku 2.

Na obrazku 3 pokazano przebieg scenariusza dołączania do spotkania. W przypadku pozytywnego wykonania wszystkich akcji, sekwencja zaczynana jest od wysłania zapytania na serwer, które aktualizuję rekord spotkania w bazie danych. Po otrzymaniu odpowiedzi zwrotnej z kodem 200, urządzenie mobilne wysyła zapytanie, zawierające token urządzenia założyciela spotkania, do serwera Expo w celu weryfikacji jego systemu operacyjnego. Kolejnie jest ono przekazywane do serwisu GMS lub APNS, które wysyła powiadomienie na wskazane tokenem urządzenie.



Rysunek 2. Scenariusz autoryzacji użytkownika



Rysunek 3. Scenariusz dołączania do spotkania

3.2. Struktura plików

3.2.1. Serwer Node.js

Większość plików została pogrupowana w foldery o nazwach config oraz modules. Wyjątkami są pliki .babelrc, index.js oraz package.json. Dodatkowo znajdują się tam także foldery ńode modulesóraz "uploads", w których znajdują się odpowiednio biblioteki niezbędne do odpalenia aplikacji oraz pliki przesyłane przez użytkowników aplikacji. Graficzne przedstawienie struktury ukazuje zdjęcie 4.

Rysunek 4. Struktura plików backendowych

.babelrc

Babel służy do konwertowania najnowszej wersji JavaScriptu do wersji ES5, która może być wykonywana przez każdą przeglądarkę. Dzięki niemu możemy wszystkich udogodnień syntaktycznych dodanych w specyfikacji JavaScriptu ES6, takich jak: klasy, grube strzałki czy paro liniowe stringi.[18]

index.js

Główny plik, w którym wywoływana jest funkcja z biblioteki express tworząca serwer. W tym miejscu do aplikacji jest także dołączana baza danych, tworzone trasy oraz ustawiane porty.

package.json

W tym pliku opisane są wszystkie pakiety od których zależny jest serwer. Dzięki temu aplikacja jest reproduktywna, co ułatwia prace nad nią innym programistą.

config

Folder zawierający plik konfiguracyjny do bazy danych oraz plik definiujący oprogramowanie pośrednicze takie jak body parser[19] oraz morgan[20].

modules

Folder w którym zawarta jest cała logika serwera. Został podzielony na cztery osobne moduły (grups, hangouts, photos oraz users), składające się z następujących plików:

- controler.js - zawierającego definicję wszystkich funkcji w danym module
- routes.js - który opisuje wszystkich możliwych zapytań związanych
- model.js - definiującego schemat elementu bazy danych związanego z danym modułem
- index.js - ułatwiającego importowanie powyżej wymienionych plików

uploads

Folder w którym przechowywane są zdjęcia profilowe przesłane przez użytkowników.

3.2.2. Aplikacja React Native

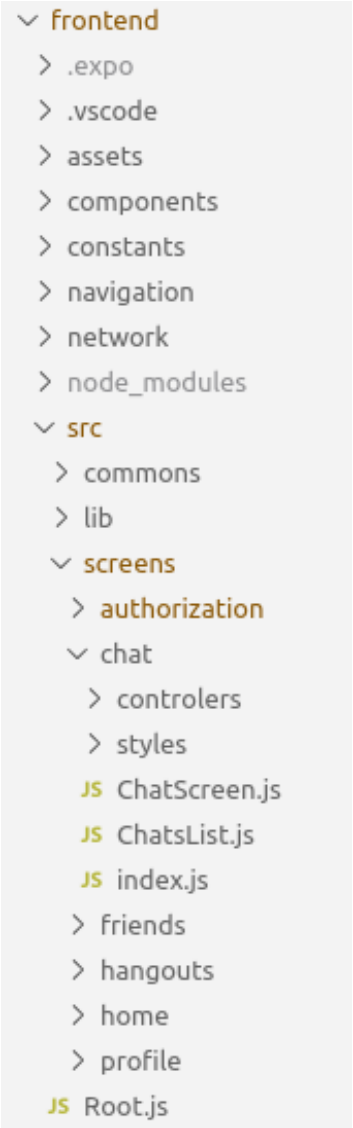
Graficzny opis struktury plików frontendowych przedstawiono na rysunku 5.

assets

Folder przechowujący czcionki, zdjęcia używane podczas uruchamiania aplikacji (logo, ekran ładowania), zmienne globalne oraz definicje funkcji asynchronicznych służące do ładowania zmiennych globalnych.

components

Pojedyncze pliki zawierające opis elementów wykorzystywanych w wielu ekranach, takie jak: HeaderIcon oraz TabBarIcon.



Rysunek 5. Struktura plików frontendowych

constants

W tym miejscu przechowywane są pliki opisujące stałe wartości: kolory, szerokość i wysokość ekranu oraz style wykorzystywane w wielu plikach.

navigation

Folder z plikami odpowiedzialnymi za nawigację pomiędzy ekranami.

network

Zawiera definicje funkcji odpowiedzialnych za asynchroniczne zapytania do serwera, występujące w wielu ekranach.

src

Folder zawierający podfoldery ćommonsóraz ścreens". Pierwszy zawierające pliki opisujące elementy powtarzające się na wielu ekranach, w tym wypadku opis ekranu wczytywania danych. W drugim znajdują się opisy ekranów, podzielone tematycznie na: autoryzację, czat, przyjaciół, spotkania, ekran główny oraz profil użytkownika. Każdy z folderów opisujących pewną grupę ekranów zawiera: plik główny z generalnym opisem ekranu, folder ćstylesópisujący style, folder ćontrolers"w którym znajdują się definicję komponentów używanych w danych grupach ekranów. Dodatkowo dołączany jest jeszcze plik index.js, który ułatwia importowanie ekranów.

App.js

Jest to główny plik aplikacji. W nim tworzona jest aplikacja oraz definiowane są czynności wykonywane przed jej uruchomieniem np. ładowanie czcionek i zmiennych globalnych.

app.json

Plik w formacie manifest opisującym między innymi wymagania, zmienne środowiskowe, dodatki oraz inne informacje dotyczące aplikacji.

package.json

W tym pliku opisane są wszystkie pakiety od których zależna jest aplikacja. Dzięki temu jest reproduktywna, co ułatwia jej rozwój innym programistą.

babel.config.js

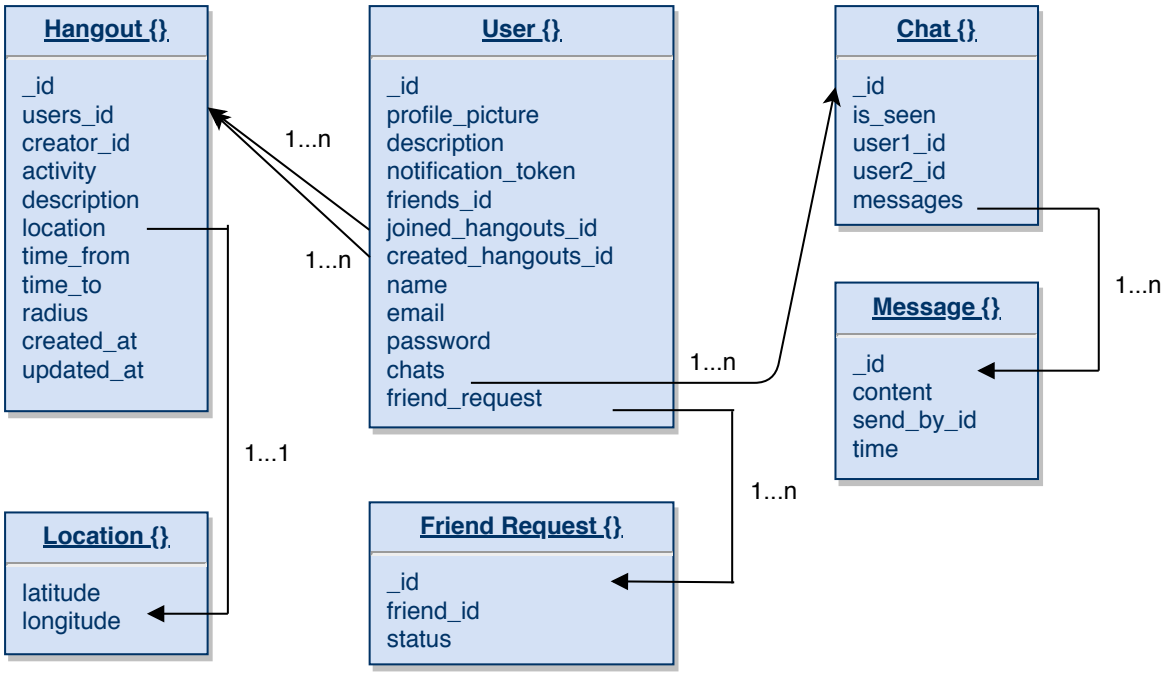
Podobnie jak w folderze z plikami dotyczącymi serwera, babel.config.js służy do konwertowania najnowszej wersji JavaScriptu do wersji ES5.

google-services.json

Plik rejestrujący aplikację w serwisie Google, umożliwiający korzystanie z usług takich jak Firebase[13].

3.3. Struktura bazy danych

Na rysunku 6 przedstawiono ogólną strukturę bazy danych. Jej głównym obiektem jest użytkownik, który posiada cztery zagnieżdżone tablicę o nazwach: joined hangouts, created hangouts, chats oraz friend requests, które na podstawie id są odpowiednio połączone z obiektami klas Hangout, Chat i Friend Request. Obiekt czatu posiada dodatkowo tablicę wiadomości.



Rysunek 6. Struktura bazy danych

4. Technologia

4.1. JavaScript

Do poprawnego funkcjonowania aplikacji wymagane jest REST Api oraz interfejs użytkownika. Jednym z niewielu języków programowania, który umożliwia napisanie obydwu aplikacji jest JavaScript. Dodatkowo, dzięki rosnącej społeczności programistów, język ten w 2019 był najpopularniejszym na serwisie Stack Overflow[3]. Poprzez wykorzystanie tylko jednego języka programowania w całej aplikacji jej dalszy rozwój nie wymaga wysokiego progu wejścia - cały kod opiera się tylko na jednej technologii. W przypadku rozwoju aplikacji o serwis internetowy wymagana będzie jedynie wiedza na temat HTML oraz CSS.

4.2. Node.js

Dzięki danemu środowisku jesteśmy w stanie stworzyć serwer dla naszej aplikacji, który jest napisany w języku JavaScript, jest wydajny i skalowalny. Dodatkowo Node.js jest rozwiązaniem cross-platformowym, które można łatwo uruchomić zarówno na Linuxie, Windowsie jak i Macu.

4.3. React

React jest biblioteką JavaScript służącą do tworzenia interfejsów użytkownika. Głównym konceptem tej technologii jest komponent - poszczególna część ekranu użytkownika np.

przycisk. Komponenty mogą być używane wielokrotnie oraz są możliwe do skomponowania - jeden komponent może składać się z wielu innych. React wykrywa, które komponenty powinny być zbudowane na nowo, a które można użyć ponownie, co pozytywnie wpływa na jego szybkość oraz popularność wśród programistów. Pomimo wszystko React nie jest oprawą dla obiektowego modelu dokumentu przeglądarki, gdyż przy jego użyciu możemy tworzyć także aplikacje mobilne.

4.4. React Native

React Native stwarza możliwość budowania aplikacji mobilnych używając Reacta oraz JavaScript. Zamiast korzystać obiektów prostych typu 'span', React Native oferuje obiekty typu 'Text', które są kolejnie interpretowane na obiekty zależne od platformy na które jest budowana aplikacja. Tak więc, dla aplikacji zaprojektowanych na systemy Android obiekt ten zwróci 'TextView', a dla aplikacji iOS zostanie zwrócony 'UIView'. Dzięki temu, pomimo, iż aplikacja jest napisana w JavaScripcie, jest w pełni natywna.

Podczas działania aplikacji zawsze są tworzone co najmniej dwa wątki - jeden z nich jest wątkiem głównym, który istnieje także w klasycznych aplikacjach mobilnych. Odpowiada za wyświetlanie elementów użytkownikowi oraz przetwarzanie jego gestów. Drugi wątek jest specyficzny dla React Nativa. Jego zadaniem jest wykonywanie JavaScriptu w oddzielnym silniku. Jest on odpowiedzialny za biznesową logikę aplikacji oraz definiuje strukturę oraz funkcjonalności interfejsu użytkownika. Wątki te nigdy nie komunikują się ze sobą bezpośrednio i nigdy nie blokują się wzajemnie.

Pomiędzy wątkami istnieje most umożliwiający komunikację między wątkami, który jest istotą React Nativa. Jest on asynchroniczny, przez co wątki nigdy się wzajemnie nie blokują. Dodatkowo jest on zoptymalizowany w sposób umożliwiający szybką komunikację, a wiadomości które przez niego przechodzą zostają serializowane, dzięki czemu obydwa wątki operują na tych samych danych.

Główną zaletą środowiska React Native jest fakt, iż napisane w nim aplikacje działają zarówno na platformie Android jak i iOS, przy czym są zupełnie natywne, dzięki czemu działają płynnie i w podobny sposób na obydwu platformach. Dodatkowo, głównym językiem wykorzystywanym w React Nativie jest JavaScript, więc idealnie komponuje się on z serwerową częścią aplikacji. Gdybyśmy w przyszłości chcieli poszerzyć naszą aplikację o stronę internetową, do jej napisania moglibyśmy skorzystać z analogicznego środowiska o nazwie React.

4.5. Socket.IO

Biblioteka ta podkreśla zalety wykorzystanie tego samego języka po stronie front-endu oraz back-endu, gdyż interfejsy po stronie użytkownika oraz serwera są bardzo zbliżone.

Socket.IO ułatwia stworzenie dwukierunkowego kanału komunikacji WebSocket pomiędzy aplikacją, a serwerem. W przypadku aplikacji HangOut, biblioteka ta jest użyta głównie do przesyłania zapytań do bazy danych.

4.6. Express.js

Jest najpopularniejszym frameworkiem wykorzystanym do tworzenia aplikacji typu REST. Uznanie użytkowników zawdzięcza swojemu minimalizmowi oraz elastyczności. Express.js służy do różnego typu zapytań HTTP i tworzenia zasad trasowania zapytań i ich wywołań zwrotnych.

4.7. MongoDB

Pomimo, że Node.js jest w stanie współpracować z bazą danych MySQL, perfekcyjną kombinacją są niestrukturyzowane bazy typu NoSQL. W przypadku aplikacji HangOut zdecydowano wybrać MongoDB. Zaletą takiego połączenia jest fakt, że dane, zamiast tabel połączonych kluczami, są reprezentowane jako dokumenty. Dzięki temu w bazie mogą być zapisywane dane o zmieniającym się formacie.

4.8. Expo

Expo jest zestawem narzędzi ułatwiającym pracę z React Native. Jego najważniejszym komponentem jest Expo SDK - zestaw bibliotek ułatwiających korzystanie z zaawansowanych funkcji telefonu. W przypadku powyższej pracy, zastosowanie znalazła podczas implementacji funkcji korzystających z dostępu do GPS, logowaniem za pośrednictwem portali społecznościowych oraz wysyłaniem i odbieraniem powiadomień PUSH. Co więcej, Expo jest wzbogacone w pakiet narzędzi ułatwiających kompilację i uruchamianie aplikacji. Za jej pośrednictwem jesteśmy w stanie odpalić nasz program na maszynie wirtualnej lub zainstalować go na własnym urządzeniu.

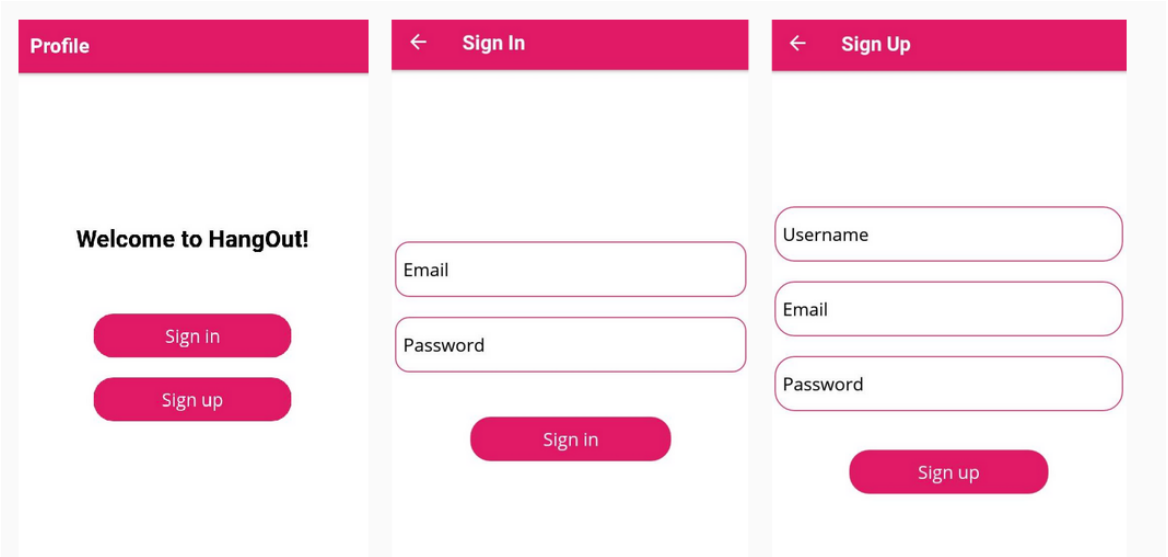
5. Opis ekranów

5.1. Autoryzacja i uwierzytelnienie

Na zdjęciu 7 zostały przedstawione 3 ekrany, które służą do logowania lub rejestracji. Jako pierwszy użytkownikowi zostaje wyświetlony ekran z dwoma przyciskami.

W przypadku wybrania przycisku 'Sign in' użytkownik zostaje przekierowany do ekranu logowania, który zawiera dwa pola. W przypadku poprawnego wypełnienia obydwu, podane dane zostają wysłane na serwer metodą GET, który sprawdza czy użytkownik o podanym adresie email istnieje w bazie danych. Jeśli tak, porównuje hasła i w przypadku w którym są zgodne, przekierowuje użytkownika do ekranu głównego aplikacji.

Podczas gdy użytkownik zdecyduje się wybrać przycisk o nazwie 'Sign up' zostaje on przeniesiony do ekranu rejestracji, w którym musi wypełnić trzy pola: nazwę użytkownika, email oraz hasło. Po niepoprawnym wypełnieniu któregokolwiek z pól, na górze ekranu zostaje wyświetlana informacja o błędzie. Gdy wszystkie pola zostaną wypełnione poprawnie, dane z formularza zostają przesłane na serwer metodą POST, która na początku sprawdza, czy podany email już istnieje. Jeśli tak, zwraca daną informację, prosząc użytkownika o wybranie innego. W przeciwnym wypadku, po uprzednim zakodowaniu hasła, dodaje dane do bazy danych, a użytkownikowi zostaje wyświetlony ekran główny.



Rysunek 7. Ekran autoryzacji i uwierzytelnienia

5.2. Ekran główny

Jeśli użytkownik logował się już wcześniej do naszej aplikacji, w cache telefonu zostało zapisane jego ID. W przypadku, w którym po ostatnim korzystaniu z aplikacji użytkownik nie

wylogował się, zostanie przekierowany na ekran główny. Jego wygląd możemy zobaczyć na rysunku 8.

Górna część składa się z trzech przycisków: 'Create Hangout', 'Find Hangout' oraz 'My Hangouts'. Kliknięcie pierwszego z nich przekieruje użytkownika do ekranu, na którym będzie mógł stworzyć nowe spotkanie. W przypadku wybrania drugiej opcji użytkownikowi zostanie wyświetlona mapa z oznaczonymi spotkaniami. Ostatni przycisk powoduje przeniesienie do ekranu na którym zostanie wyświetlona lista wszystkich spotkań użytkownika.

Podczas uruchamiania aplikacji pobierane są informację na temat szerokości oraz wysokości geograficznej użytkownika. Dodatkowo na serwer wysyłane jest zapytanie, zwracające listę spotkań znajdujących się w pobliżu użytkownika.

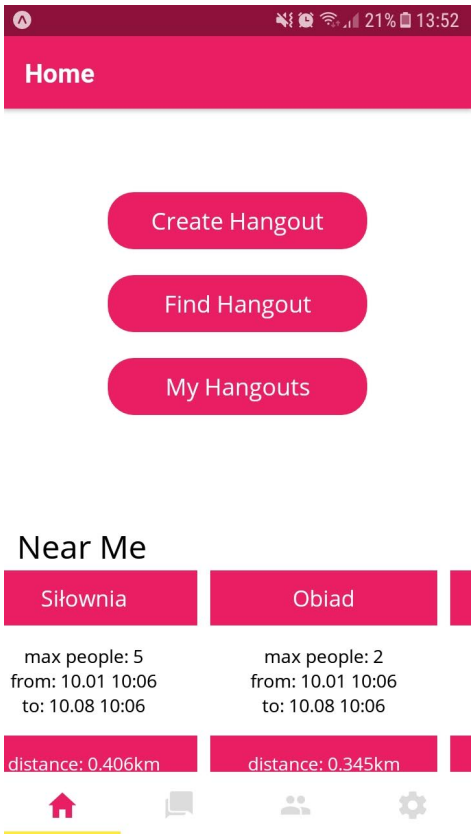
Serwer, otrzymawszy w zapytaniu dane o lokalizacji użytkownika, pobiera z bazy danych wszystkie spotkania i dla każdego z nich, na podstawie funkcji z Listingu 1, liczy odległość.

```
1  //getting distance from 2 objects in kilometers
2  function getDistanceFromLatLonInKm(lat1, lon1, lat2, lon2) {
3    var R = 6371; // Radius of the earth in km
4    var dLat = deg2rad(lat2 - lat1);
5    var dLon = deg2rad(lon2 - lon1);
6    var a =
7      Math.sin(dLat / 2) * Math.sin(dLat / 2) +
8      Math.cos(deg2rad(lat1)) *
9        Math.cos(deg2rad(lat2)) *
10       Math.sin(dLon / 2) *
11       Math.sin(dLon / 2);
12   var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
13   var d = R * c; // Distance in km
14   return d.toFixed(3);
15 }
16
17 //Changing degrees to Radians
18 function deg2rad(deg) {
19   return deg * (Math.PI / 180);
20 }
```

Listing 1. Implementacja obliczania odległości

Jeśli wartość funkcji liczona od lokalizacji użytkownika oraz spotkania, zmniejszona o promień spotkania jest nie większa niż 3 kilometry, dane spotkania zostają wysyłane jako odpowiedź serwera. Kolejnie są one wyświetlane na liście u dołu ekranu. Pozycja z listy zawiera podstawowe informacje na temat spotkania, takie jak: maksymalna liczba uczestników, okres w jakim możliwe jest spotkanie oraz dokładna odległość od użytkownika.

Po kliknięciu na wybrany element użytkownik zostaje przekierowany do karty danego spotkania.



Rysunek 8. Ekran główny

5.3. Profil użytkownika

Na ekranie z rysunku 9 możemy zmieniać podstawowe informacje na temat użytkownika. Zdecydowano ograniczyć je do minimum, gdyż nie jest to kluczowy element aplikacji. Przed załadowaniem ekranu wysyłane jest zapytanie na serwer, które zwraca nazwę oraz opis użytkownika. Dane te są później wyświetlane w API użytkownika. W tym miejscu można zmienić zdjęcie profilowe i opis.

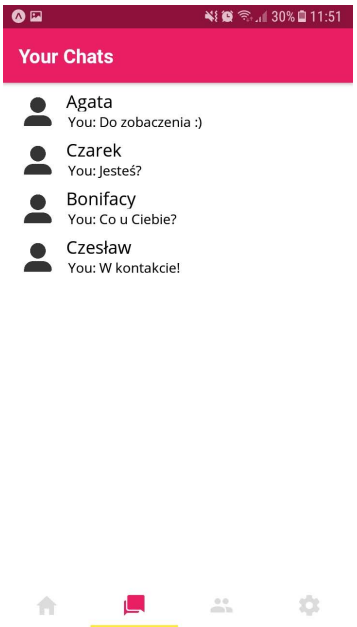
Chcąc zmienić obraz, należy na niego kliknąć. Aplikacja przeniesie użytkownika do kolejnego ekranu, na którym zostanie wyświetlona lista zdjęć z galerii. Po wybraniu zdjęcia, zostanie przekierowany do kolejnego ekranu, na którym będzie mógł określić fragment zdjęcia, który chce ustawić jako profilowe. Po zaakceptowaniu wybranej wielkości, jakość zostaje zmniejszona pięciokrotnie, a zdjęcie zostaje zapisane na serwerze. W przypadku, w którym użytkownik nie przesłał jeszcze żadnego zdjęcia, ustawiane jest mu domyślne.

Aby edytować opis, użytkownik musi na niego kliknąć i wprowadzić zmiany. Po zakończonej aktywności aplikacja wysyła zapytanie POST z nowym opisem, które aktualizuję bazę danych.

Rysunek 9. Profil użytkownika

5.4. Lista czatów

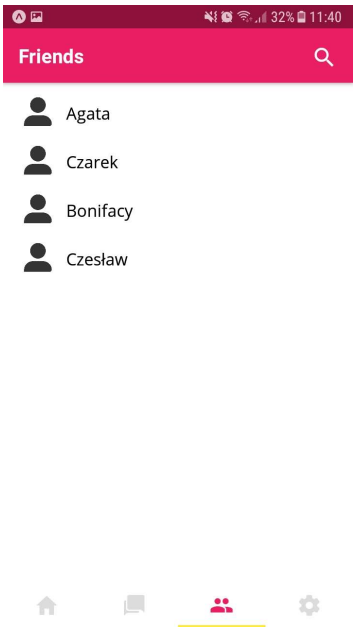
Ekran z rysunku 10 służy do wyświetlenia listy czatów z przyjaciółmi. Przed jego załadowaniem jest wysyłane zapytanie na serwer, które zwraca listę konwersacji wzbogaconą o nazwy przyjaciół oraz ostatnie wiadomości, które są później wyświetlane użytkownikowi. Po kliknięciu na wybrany obiekt aplikacja przekierowuje do ekranu danej konwersacji.



Rysunek 10. Lista czatów

5.5. Lista przyjaciół

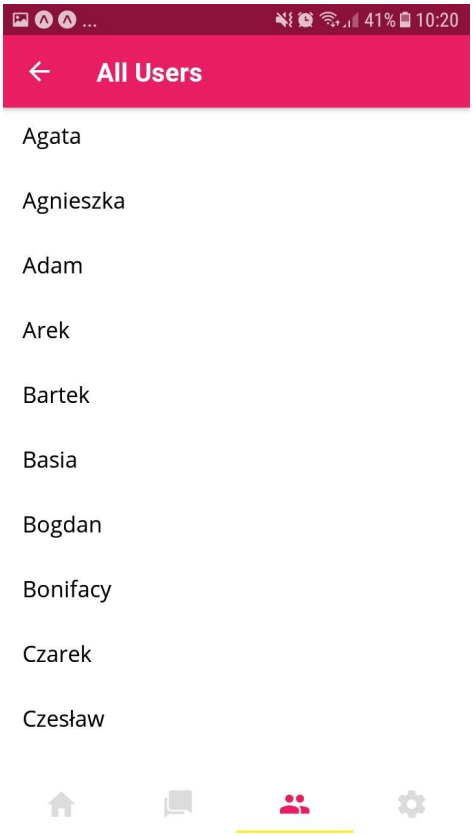
Ekran znajdujący się na rysunku 11 służy do wyświetlenia listy przyjaciół. Przed załadowaniem, na serwer zostaje wysłane zapytanie z id użytkownika, które zwraca listę identyfikatorów oraz nazw jego przyjaciół. Po kliknięciu na wybrany obiekt użytkownik zostaje przekierowany do ekranu z profilem przyjaciela. W prawym górnym rogu znajduje się przycisk przenoszący użytkownika do listy wszystkich użytkowników aplikacji.



Rysunek 11. Lista przyjaciół

5.6. Lista użytkowników

Ekran znajdujący się na zdjęciu 12 służy do wyświetlenia listy wszystkich użytkowników aplikacji. Przed jego załadowaniem, na serwer wysyłane jest zapytanie na serwer, które zwraca listę identyfikatorów oraz nazw wszystkich użytkowników aplikacji. Po kliknięciu na wybrany obiekt z listy, użytkownik zostaje przekierowany do ekranu z profilem danej osoby.



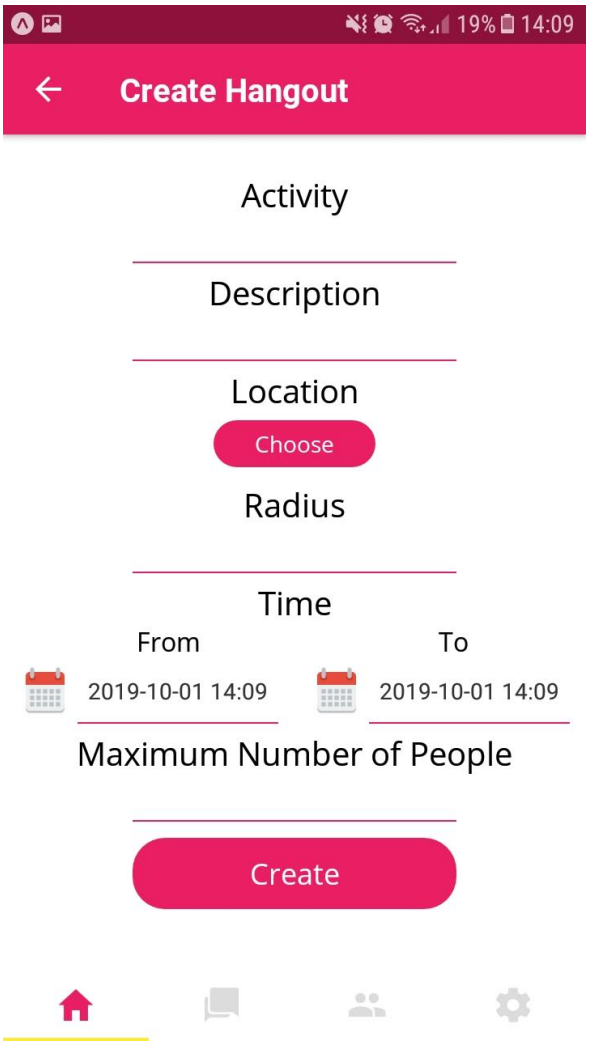
Rysunek 12. Lista użytkowników

5.7. Tworzenie spotkania

Ekran przedstawiony na obrazku 13 służy do tworzenia spotkań. Informacje wymagane do poprawnego wypełnienia formularza to:

- nazwa (activity)
- opis (description)
- lokalizacja (location)
- odległość, od wyznaczonej początkowo lokalizacji, jaką użytkownik jest w stanie maksymalnie się przemieścić (radius)
- okres w jakim użytkownik może uczestniczyć w spotkaniu (time)
- maksymalna liczba uczestników spotkania (maximum number of people)

Aby określić lokalizację spotkania użytkownik musi kliknąć przycisk 'Choose', który przekieruje go do ekranu z mapą. Po wypełnieniu formularza należy go zaakceptować przyciskiem 'Create'. W przypadku niepoprawnego wypełnienia któregoś z pól, użytkownik zostaje o tym poinformowany komunikatem na górze ekranu. Jeśli formularz został wypełniony poprawnie, na serwer zostaje wysyłane zapytanie, które zapisuje go w bazie danych. Następnie użytkownik zostaje przeniesiony do ekranu głównego.



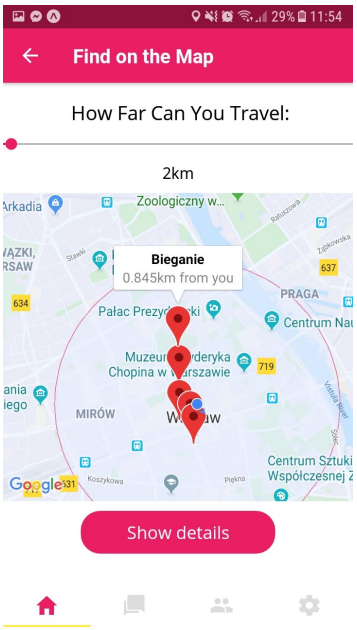
Rysunek 13. Tworzenie spotkania

5.8. Wybieranie lokalizacji spotkania

Ekran z rysunku 14 służy do wybrania lokalizacji swojego spotkania. Przed jego pokazaniem podbierane są informacje na temat lokalizacji użytkownika. Na ich podstawie wyświetlany jest obszar mapy w pobliżu użytkownika. Aby określić miejsce należy zaznaczyć pinezkę na mapie oraz wcisnąć przycisk 'Save'. Następnie użytkownik ponownie zostaje przekierowany do poprzedniego ekranu.

Rysunek 14. Wybieranie lokalizacji spotkania

5.9. Wyszukiwanie spotkań

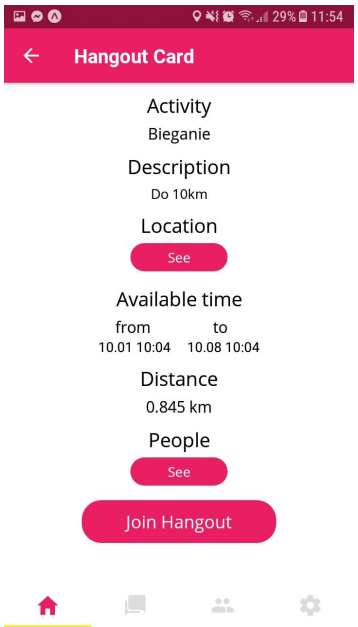


Rysunek 15. Wyszukiwanie spotkań

Na ekranie ukazanym na zdjęciu 15 użytkownikowi zostaje wyświetlona mapa ze wszystkimi dostępnymi spotkaniami. Przed jego załadowaniem, na serwer jest wysyłane zapytanie, które w odpowiedzi zwraca podstawowe informację na temat spotkań znajdujących się w obszarze okręgu o promieniu ustalonym przy pomocy suwaka. Zmieniając jego wartość, w górnej części ekranu, można ustalić promień koła wyświetlanego na mapie.

Dzięki temu użytkownik może przefiltrować wyświetlane spotkania i wziąć pod uwagę tylko te, które znajdują się w odległości nie większej niż ta wybrana na suwaku. Dodatkowo, w celu wyświetlenia odpowiedniego obszaru mapy, z serwera pobierana jest lokalizacja użytkownika. Po kliknięciu na pinezkę zostaje wyświetlona nazwa danego spotkania oraz dokładna odległość. Jeśli użytkownik dodatkowo kliknie przycisk Śhow detailsźostanie przekierowany do ekranu ze szczegółowymi informacjami na temat spotkania. W tym miejscu będzie mógł też do niego dołączyć.

5.10. Szczegóły spotkania

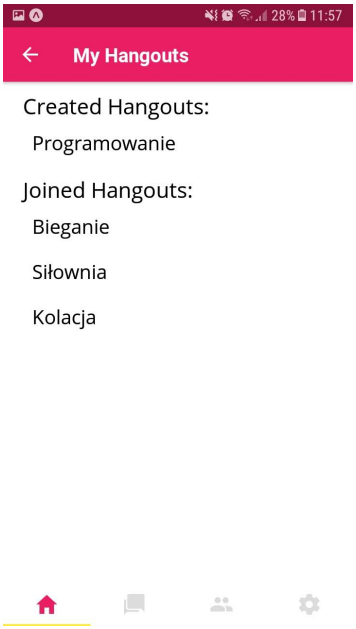


Rysunek 16. Szczegóły spotkania

Ekran, na którym są wyświetlane szczegółowe informacje na temat spotkania jest przedstawiony na rysunku 16. Zanim zostanie załadowany, na serwer przesyłane jest id spotkania, na podstawie którego są zwracane szczegółowe informację, które go dotyczą. Po kliknięciu przycisku z pierwszym napisem Śee", użytkownik zostanie przekierowany do ekranu z mapą na której pokazana jest dokładna lokalizacja spotkania. Kliknięcia drugiego przycisku przekierowuje do ekranu z listą uczestników spotkania. Aby dołączyć, należy wcisnąć ostatni przycisk o nazwie "Join hangout". Gdy użytkownik to zrobi, zostanie przeniesiony na ekran główny, a założyciel spotkania zostanie o tym poinformowany powiadomieniem PUSH. Dodatkowo na serwer jest wysyłane zapytanie, które powoduje zapisanie nowego spotkania do bazy danych.

5.11. Lista spotkań użytkownika

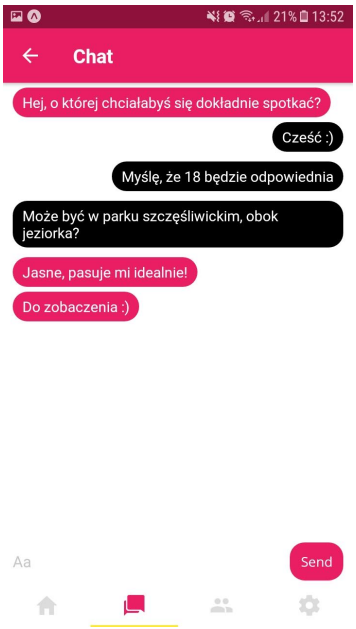
Ekran z listą wszystkich spotkań użytkownika ukazuje obrazek 17. Aby poprawnie wyświetlić zawarte na nim informację, API wysyła zapytanie do serwera, które zwraca listę spotkań utworzonych przez użytkownika oraz listę spotkań do których użytkownik dołączył. Informację są przedstawiane na ekranie również w formie dwóch oddzielnych list, które są odpowiednio podpisane. Po kliknięciu na wybraną pozycję z jednej z nich, użytkownik zostaje przekierowany do ekranu ze szczegółami spotkania, gdzie będzie mógł je opuścić lub usunąć.



Rysunek 17. Lista spotkań użytkownika

5.12. Czat

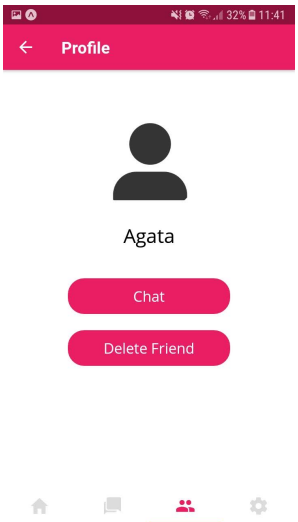
Ekran wybranej konwersacji został ukazany na ilustracji 18. Przed jego wyświetleniem do serwera zostaje wysyłane zapytanie, które zwraca z bazy danych daną konwersację, podzieloną na wiadomości. Dodatkowo, każda z nich jest podpisana id użytkownika, który ją napisał. Dane wiadomości, w przypadku nadania przez użytkownika są wyświetlane na różowym tle. Gdy zostały wysłane przez drugą osobę, wyświetlane będą na tle czarnym. Po wprowadzeniu wiadomości do pola na dole ekranu i zaakceptowaniu jej przyciskiem po lewej stronie zostaje wysyłane zapytanie na serwer, które dodaje wiadomość do bazy danych. Informacja ta zostaje zapisana w dwóch dokumentach - powiązanym osobno z nadawcą i odbiorcą. Adresat dodatkowo zostaje poinformowany powiadomieniem PUSH, które jest wysyłane tylko w przypadku, w którym odczytał wszystkie poprzednie wiadomości od danego użytkownika. Takie rozwiązanie zapobiega wysyłaniu nadmiarowych notyfikacji.



Rysunek 18. Czat

5.13. Profil przyjaciela

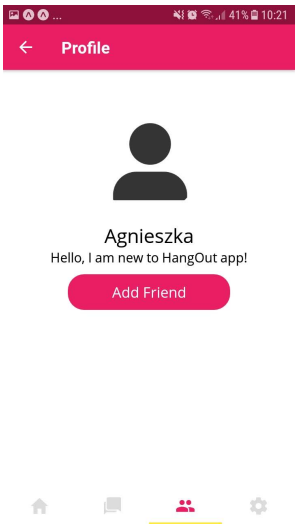
Ekran z rysunku 19 służy do wyświetlenia konwersacji z przyjacielem lub do jego usunięcia. Informację o nazwie użytkownika są przekazywane do tego ekranu z poprzedniego. Naciśnięcie pierwszego przycisku przez użytkownika przeniesie go do ekranu czatu. Wybranie drugiego spowoduje wysłanie zapytania na serwer, które zaktualizuję listę przyjaciół użytkowników wzajemnie ich z niej usuwając. Po danej akcji, nazwa przycisku z 'Delete Friend' zostaje zmieniana na 'Add Friend'.



Rysunek 19. Profil przyjaciela

5.14. Profil użytkownika

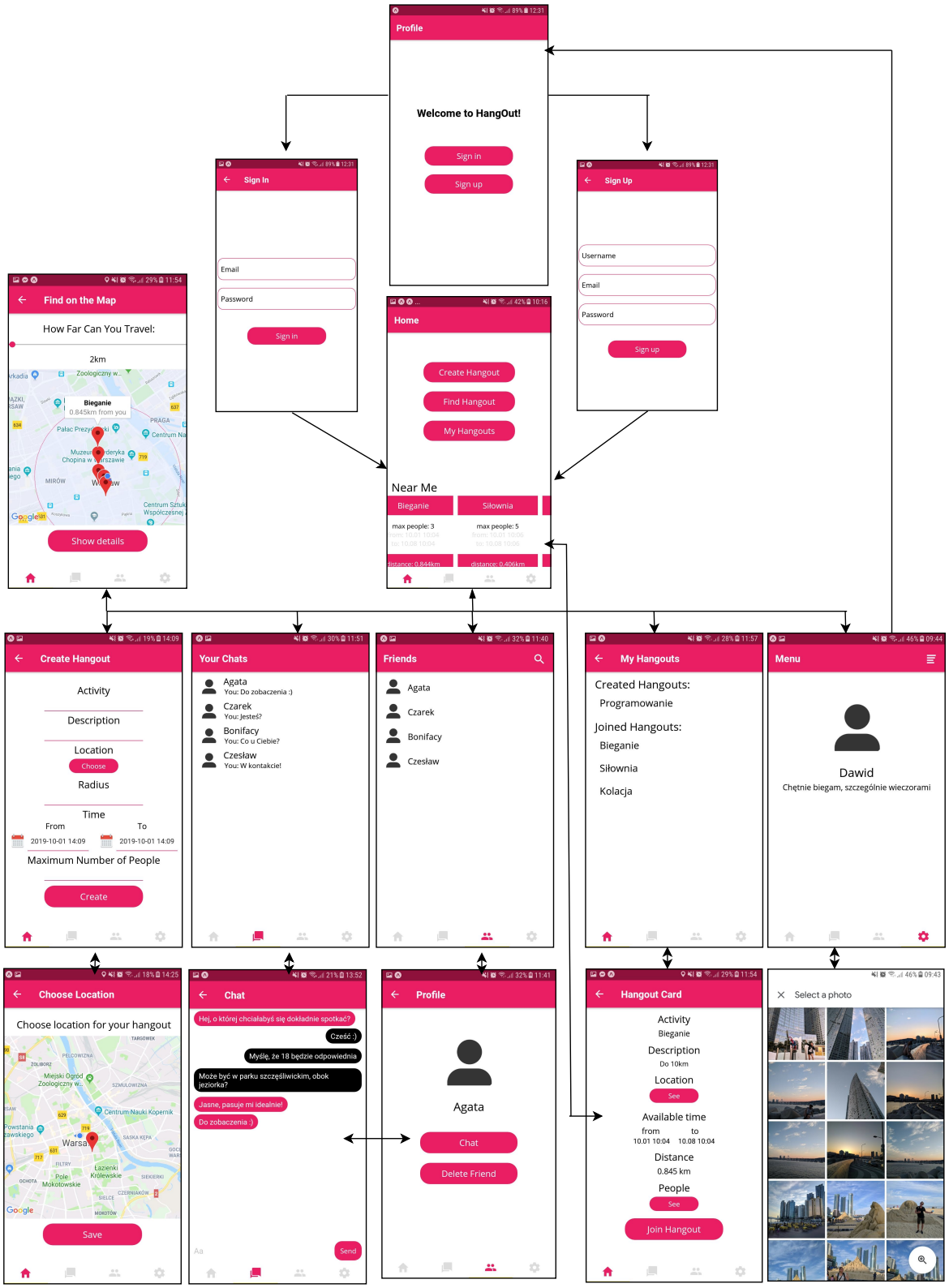
Na ekranie przedstawionym na zdjęciu 20 przedstawiającym profil użytkownika. Po wciśnięciu przycisku o nazwie Ądd Friend"do danej osoby zostanie wysłane powiadomienie PUSH z informacją o propozycji zawarcia przyjaźni. Po otwarciu powiadomienia, zostaje wyświetlony profil użytkownika wysyłającego propozycję, z możliwością jej przyjęcia. Dodatkowo na serwer zostaje wysłane zapytanie, które powoduje dodanie zapraszanej osoby do listy zaproszonych przez użytkownika, a nazwa przycisku z 'Add Friend' zostaje zmieniona na 'Cancel Request'.



Rysunek 20. Profil użytkownika

5.15. Architektura ekranów

Na rysunku 21 przedstawiono możliwości przemieszczania się pomiędzy ekranami.



Rysunek 21. Architektura ekranów

6. Podsumowanie

Podczas pracy inżynierskiej udało się spełnić wszystkie wymagania opisane w drugim rozdziale. Stworzona aplikacja umożliwia płynne tworzenie i dołączanie do spotkań, a użytkownicy mają możliwość komunikacji poprzez czat. Dodatkowo została zaimplementowana opcja edycji profilu, możliwość dodawania użytkowników do przyjaciół oraz powiadomienia PUSH.

Całość została podzielona na dwa osobne moduły - serwer, głównie obsługujący zapytania do bazy danych oraz API użytkownika odpowiedzialne za wyświetlanie i pobieranie informacji na urządzeniach mobilnych.

Obydwie części pracy zostały oparte na języku JavaScript, dzięki czemu dalszy rozwój aplikacji jest bardzo prosty. Co więcej serwer został stworzony niezależnie od platform typu Firebase, co umożliwia dużą swobodę w implementacji dalszych rozwiązań.

6.1. Możliwości rozwoju

Z racji na wybór środowiska w jakim zostało napisane API użytkownika (React Native) aplikacja działa na platformie Android oraz iOS. W danym wypadku, kolejnym etapem rozwoju aplikacji powinno być umożliwienie użytkownikom pobrania jej z autoryzowanych sklepów takich jak Google Play oraz App Store.

Kolejną propozycją na rozwój jest serwis internetowy, pozwalający na korzystanie z aplikacji za pomocą przeglądarek. Dużym uproszczeniem w przypadku próby implementacji jest fakt, że serwer z którego korzysta aplikacji mobilna może zostać wykorzystany również do obsługi API webowego. Co więcej, decydując się na wybór środowiska React do napisania strony frontendowej, programista będzie mógł ponownie użyć większość kodu napisanego w React Nativie.

Bibliografia

[1] K. N. Hampton, „Social Media and Change in Psychological Distress Over Time: The Role of Social Causation”, *Journal of Computer-Mediated Communication*, czer. 2019.

[2] D. Derks, D. van Duin, M. Tims i A. B. Bakker, „Smartphone use and work–home interference: The moderating role of social norms and employee work engagement”, *Journal of Occupational and Organizational Psychology*, 2015.

[3] `https://www.facebook.com/`.

[4] `https://www.instagram.com/`.

[5] `https://twitter.com/`.

[6] C Sanders, T Field, M Diego i M Kaplan, „The Relationship of Internet Use to Depression and Social Isolation Among Adolescents”, *Adolescence*, t. 35, s. 237–42, lut. 2000.
[7] I. W. Telecommunication, „Individuals using the Internet 2005 to 2015”, maj 2015.
[8] `https://tinder.com/`.
[9] G. Ranzini i C. Lutz, „Love at first swipe? Explaining Tinder self-presentation and motives”, *Mobile Media & Communication*, 2017.
[10] `https://reactjs.org/`.
[11] `https://en.wikipedia.org/wiki/Document_Object_Model`.
[12] `https://en.wikipedia.org/wiki/React_(web_framework)#Virtual_DOM`.
[13] `https://firebase.google.com/`.
[14] „Web Services Architecture”, *World Wide Web Consortium*, wrz. 2016.
[15] J. C. N. H. F. M. L. L. P. J. B.-L. T. Gettys James; Mogul, „Hypertext Transfer Protocol – HTTP/1.1”, czer. 1999.
[16] `https://en.wikipedia.org/wiki/Google_Cloud_Messaging`.
[17] `https://en.wikipedia.org/wiki/Apple_Push_Notification_service`.
[18] `https://babeljs.io/docs/en/`.
[19] `https://www.npmjs.com/package/body-parser`.
[20] `https://www.npmjs.com/package/morgan`.

Wykaz symboli i skrótów

1. **EiTI** – Wydział Elektroniki i Technik Informacyjnych
2. **PW** – Politechnika Warszawska

Spis rysunków

Spis tabel

Spis załączników