

Uniwersytet Wrocławski  
Wydział Fizyki i Astronomii

Jakub Jędros

Tworzenie aplikacji mobilnych wykorzystujących  
zapis danych w chmurze oraz Google Maps Platform, na  
przykładzie aplikacji służącej do oznaczania miejsc i  
planowania wycieczek we Wrocławiu.

Creating mobile applications using cloud database and  
Google Maps API on the example of application allowing for  
saving places and planning trip in Wroclaw.

Praca inżynierska na kierunku  
Informatyka Stosowana i Systemy Pomiarowe

Opiekun  
prof. dr hab. Zbigniew Koza

Wrocław, 25 kwietnia 2022

# Spis treści

<b>1 Wstęp</b>	<b>5</b>
<b>2 Struktura aplikacji</b>	<b>7</b>
2.1 Chmura obliczeniowa Google . . . . .	8
2.1.1 Firebase . . . . .	9
2.1.2 Firebase Authentication . . . . .	9
2.1.3 Firebase Cloud Firestore . . . . .	10
2.1.4 Google Maps Platform . . . . .	12
2.2 Aplikacja mobilna . . . . .	13
2.2.1 Kotlin i Android Studio . . . . .	13
2.2.2 Model danych . . . . .	14
<b>3 Implementacja</b>	<b>16</b>
3.1 Interfejs użytkownika . . . . .	16
3.2 Integracja z Firebase . . . . .	19
3.2.1 Implementacja Firebase Authentication . . . . .	19
3.2.2 Implementacja Firebase Cloud Firestore . . . . .	20
3.3 Integracja z Google Maps Platform . . . . .	23
3.3.1 Implementacja Maps SDK for Android . . . . .	23
3.3.2 Implementacja Directions API . . . . .	24
<b>4 Prezentacja aplikacji</b>	<b>27</b>
4.1 Ekrany autoryzacji użytkownika . . . . .	27
4.1.1 Ekran powitalny . . . . .	27
4.1.2 Ekran rejestracji . . . . .	28
4.1.3 Ekran logowania za pomocą konta Twitter . . . . .	28
4.2 Ekrany główne . . . . .	30
4.2.1 Ekran mapy . . . . .	30
4.2.2 Ekran ustawień . . . . .	35
4.2.3 Listy ulubionych miejsc i wycieczek . . . . .	36
4.2.4 Listy miejsc i wycieczek użytkownika . . . . .	39
4.2.5 Lista najpopularniejszych wycieczek . . . . .	41
<b>5 Podsumowanie</b>	<b>43</b>

# Streszczenie

W niniejszej pracy dyplomowej przedstawiony zostanie proces tworzenia aplikacji mobilnej na system operacyjny Android, napisanej w języku Kotlin, na przykładzie aplikacji mobilnej dającej użytkownikom możliwość zapisu miejsc na mapie Wrocławia oraz planowania wycieczek wykorzystując zapisane lokacje. Użytkownik może zapisywać miejsca i wycieczki jako prywatne lub publiczne, które wyświetla się na mapie innych użytkowników.

Celami projektu były:

- Wykrywanie pozycji użytkownika na mapie Wrocławia.
- Pozwolenie użytkownikowi na dodanie Miejsca posiadającego nazwę, opis oraz należącego do kategorii.
- Pozwolenie użytkownikowi na tworzenie Wycieczek składających się z maksymalnie dziesięciu Miejsc.
- Pozwolenie użytkownikowi na dodawanie Etykiet do wszystkich publicznych Miejsc.
- Funkcja rysowania na mapie ścieżek prowadzących do miejsc oraz obrazujących trasy wycieczek.

Aplikacja ma służyć jako narzędzie społecznościowe pozwalające użytkownikom na dzielenie się ciekawymi miejscami oraz zapisywaniem planów wycieczek. Użytkownicy mogą zapisywać interesujące ich miejsca oraz wycieczki do list „Ulubionych”. Do każdego miejsca użytkownik może wytyczyć ścieżkę, a każda wycieczka ma możliwość rysowania trasy przechodzącej przez wszystkie jej miejsca w kolejności zadanej przez użytkownika.

Aplikacja przechowuje dane użytkowników, ich miejsc oraz wycieczek w chmurze obliczeniowej Google Cloud za pośrednictwem platformy Firebase. Komunikacja z bazą danych zachodzi za pomocą bibliotek i API tworzonych przez firmę Google z myślą o aplikacjach mobilnych i webowych.

Interfejs użytkownika zaprojektowany został z użyciem stworzonego przez Google stylu graficznego i języka projektowego Material Design.

# Abstract

This thesis describes the process of creating a mobile application for Android operating system, written in the language Kotlin, on the example of a mobile application that allows its users to save places on the map of Wroclaw and to plan trips using the aforementioned places. The user can save places as private or public, which can be seen on the map by other users. Main goals of the project were:

- Detecting the position of the user on the map of Wroclaw.
- Letting the user add a Place containing name, description and belonging to a category.
- Letting the user create Trips consisting of at most ten places.
- Letting the user add Tags to every public Place.
- Drawing the paths on the map that lead to a place or chart the path of a trip.

The application is meant as a social network, allowing the users to share interesting places and to save their trip plans. Users can save places and trips by adding them to lists of Favourites. It is possible to chart a path to every place. Every trip allows the user to chart a path through all of its places in an order arranged by the user.

The application stores the data of its users and their places and trips in Google Cloud Computing Engine using the service of the Firebase platform. Communication with the database is facilitated by libraries and API's created by Google with mobile and web applications in mind.

User interface was created using Material Design — graphic style and project language created and maintained by Google.

# 1 Wstęp

Projekt powstał z myślą o stworzeniu aplikacji społecznościowej pozwalającej użytkownikom na dzielenie się ciekawymi miejscami, które odwiedzili, oraz pozwalającej na planowanie wycieczek. Aplikacja może posłużyć jako narzędzie pomagające turystom i obcokrajowcom poznać interesujące lokacje w swojej nowej okolicy. Motywacją do stworzenia aplikacji była chęć napisania aplikacji integrującej w sobie wykorzystanie Google Maps Platform, serwisu Firebase oraz stworzenia systemu społecznościowego pozwalającego na komunikację niewykorzystującą pisemnego wkładu użytkowników, który przyjął formę Etykiet.

Po zarejestrowaniu się za pomocą adresu e-mail lub konta Twitter użytkownik powitany jest przez ekran główny zawierający na sobie mapę oraz pasek nawigacji. Za pomocą znajdującego się na środku paska nawigacji przycisku „Dodaj” użytkownik może w miejscu na mapie znajdującym się w centrum ekranu ustawić Pinezkę, dodając tym samym Miejsce do bazy danych. Użytkownik może ustalić nazwę, opis i kategorię miejsca, oraz jego prywatność. Korzystając z przycisków znajdujących się na górze ekranu, użytkownik może przeszukać okolicę widoczną na ekranie w poszukiwaniu Miejsc. Dolne menu umożliwia użytkownikowi nawigację pomiędzy widokiem mapy, ustawień konta użytkownika, listą ulubionych miejsc i wycieczek użytkownika, listą miejsc i wycieczek stworzonych przez użytkownika oraz listą najpopularniejszych wycieczek. Po zeskanowaniu okolicy w poszukiwaniu miejsc na mapie wyświetla się pinezki reprezentujące miejsca. Kliknięcie jednej z nich otworzy menu prezentujące okoliczne miejsca i ich detale. Tam użytkownik ma możliwość dodania miejsca do istniejącej już wycieczki lub stworzenia nowej, zawierającej wybrane miejsce.

Istotnym elementem projektu jest także system Etykiet (Tagów). Jest on sposobem na wprowadzenie pewnego stopnia komunikacji użytkowników bez konieczności pisemnego wkładu takiego, jakim są np. komentarze. Jest to prosty system pozwalający każdemu użytkownikowi na dodanie wybranych z listy Etykiet do każdego publicznego miejsca w bazie danych. Każde miejsce zlicza ilość nadanych przez użytkowników etykiet i wyświetla 10 najpopularniejszych z nich w menu miejsca. Pozwala to na pewien sposób oceny miejsca pod kątem jego przeznaczenia, a nawet reinterpretacji przeznaczenia założonego przez autora miejsca.

Aplikacja została stworzona za pomocą języka programowania Kotlin, który jest zalecany przez firmę Google językiem do tworzenia natywnych aplikacji na system operacyjny Android. Jest on rozwijany przez firmę JetBrains. Firma ta jest także twórcą Android Studio — Zintegrowanego Środowiska Programistycznego dedykowanego do programowania aplikacji na urządzenia z systemem Android. Android Studio posiada zintegrowane emulatory urządzeń oraz szereg narzędzi ułatwiających tworzenie aplikacji mobilnych.

Wszelkiego rodzaju dane zapisywane są przez aplikację w Chmurze Obliczeniowej Google za pośrednictwem serwisu Firebase. Firebase to serwis typu Back-end Jako

Usługa (ang. Back-end as a Service, BaaS) należący do firmy Google. Komunikacja pomiędzy Firebase a aplikacją zachodzi za pomocą dedykowanych bibliotek, których zastosowanie zostanie przybliżone w dalszej części pracy. Dane użytkowników, miejsc oraz wycieczek zapisywane są w oferowanej w usłudze serwisu Firebase nierelacyjnej bazie danych Firestore.

Google Maps Platform to obszerny zestaw wielu Interfejsów Programowania Aplikacji (ang. application programming interface, API) należących do trzech głównych kategorii: Maps (mapy), Places (miejscia) i Routes (trasy). W niniejszej pracy zastosowałem należące do pierwszej kategorii „Maps SDK for Android” oraz należące do trzeciej kategorii Directions API. Opis integracji tych interfejsów zawarty jest w dalszych częściach pracy.

## 2 Struktura aplikacji

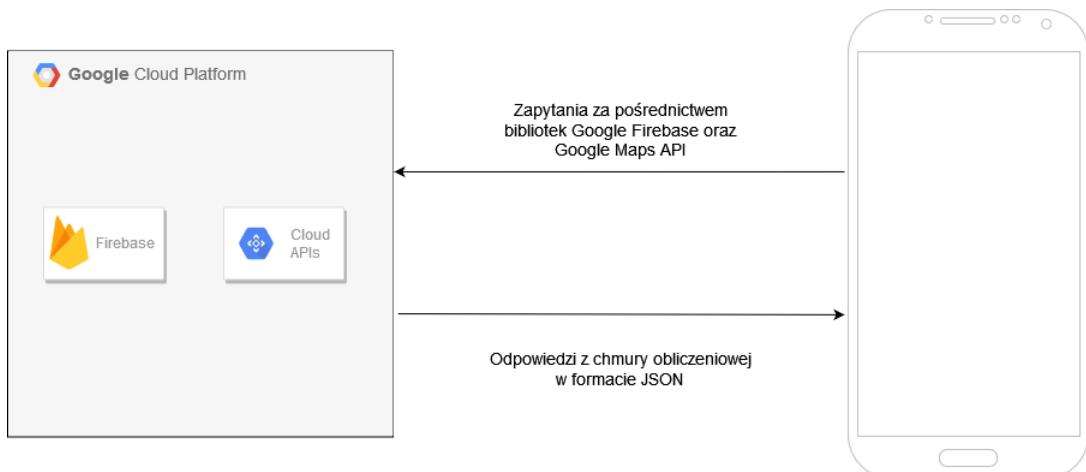
Aplikacja składa się z dwóch głównych części:

- Aplikacji mobilnej dostępczej na system Android.
- Warstwy back-end zapewnianej przez usługi serwisu Firebase.

Aplikacja mobilna stworzona została zgodnie ze wzorem architektury MVVM (ang. Model-View-Viewmodel, Model-Widok-Viewmodel). Oznacza to, że struktura aplikacji podzielona jest na trzy główne elementy:

- Model — reprezentacja obiektów przechowywanych w bazie danych, warstwa danych.
- Widok — warstwa graficzna, wizualna, obsługująca interakcję użytkownika. Reprezentacja modelu w poszczególnych elementach interfejsu użytkownika.
- Viewmodel — abstrakcyjny obiekt separujący warstwę danych od warstwy widoku. Zawiera w sobie logikę komunikacji aplikacji z bazą danych, oddzielając elementy interfejsu od logiki tej komunikacji.

Firebase [23] jest platformą zapewniającą twórcom aplikacji mobilnych i sieciowych różnorodność narzędzi ułatwiających rozwój aplikacji. Platforma jest ściśle powiązana z Google Cloud Platform — główną usługą chmur obliczeniowych Google. Warstwa back-end wykorzystuje dwa produkty oferowane przez Firebase: Cloud Firestore — nierelacyjną bazę danych mającą formę drzewa w pliku o formacie JSON oraz Authentication — narzędzie pozwalające na bezpieczne uwierzytelnianie użytkowników w serwisie zbudowanym na podstawie Firebase.

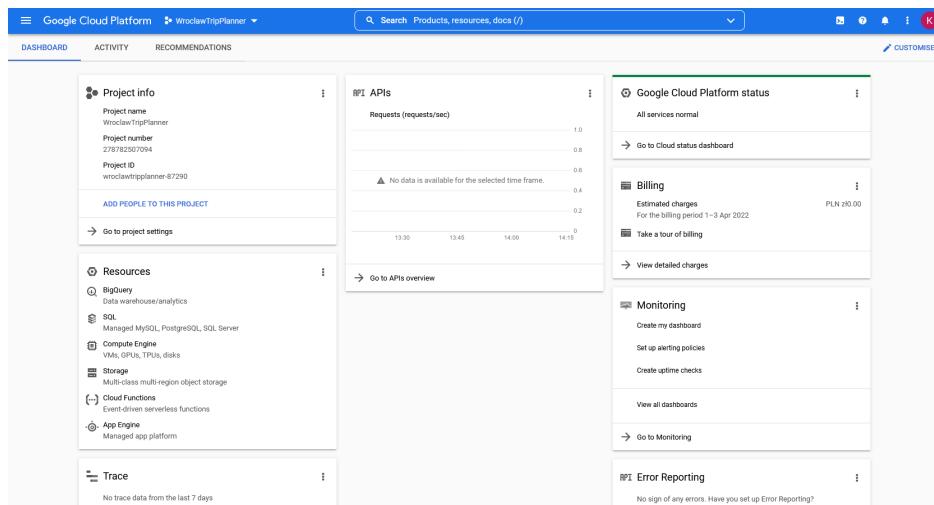


Rysunek 1: Komunikacja aplikacji z Chmurą Obliczeniową Google.

## 2.1 Chmura obliczeniowa Google

Usługi Chmury Obliczeniowej Google oferowane są za pośrednictwem serwisu Google Cloud Platform. Google Cloud Platform [20] jest kompletem narzędzi i usług pozwalających deweloperom budować aplikacje korzystające z tej samej infrastruktury, której firma Google używa do budowania swoich produktów konsumenckich, takich jak Google Search, Gmail, czy YouTube. Google Cloud Platform zapewnia użytkownikom szereg usług typu Infrastruktura Jako Usługa (ang. Infrastructure as a Service, IaaS), Platforma Jako Usługa (ang. Platform as a Service, PaaS) itp. Oznacza to, że Google Cloud Platform jako chmura obliczeniowa ofiaruje swoim użytkownikom sprzęt komputerowy zoptymalizowany do konkretnych rozwiązań i pokryty warstwą abstrakcji, jaką jest interfejs chmury obliczeniowej [1].

Wśród licznych produktów dostępnych na Google Cloud Platform [18] można znaleźć m.in. App Engine — Platforma Jako Usługa pozwalająca na wdrażanie aplikacji w językach takich jak Java, PHP, czy Python, Compute Engine — Infrastruktura Jako Usługa służąca do tworzenia maszyn wirtualnych, a także wiele narzędzi do przechowywania różnego rodzaju danych, wspomagających aplikacje wykorzystujące machine learning, usługi bezpieczeństwa czy liczne narzędzia do zarządzania projektem i analityki. Ponadto w ofercie Google Cloud Platform znajduje się kategoria usług **API Platform**. Daje ona dostęp do obszernej biblioteki Interfejsów Programowania Aplikacji. Należą do niej zarówno API tworzone przez Google, takie jak Cloud Translation API, Google Drive API, czy wykorzystane w niniejszej pracy Maps SDK for Android i Directions API, ale także interfejsy tworzone przez firmy postronne. Projekt stworzony na platformie Firebase jest równocześnie projektem Google Cloud platform, co pozwala na proste korzystanie z usług zapewnianych przez oba serwisy.



Rysunek 2: Ekran główny konsoli Google Cloud Platform.

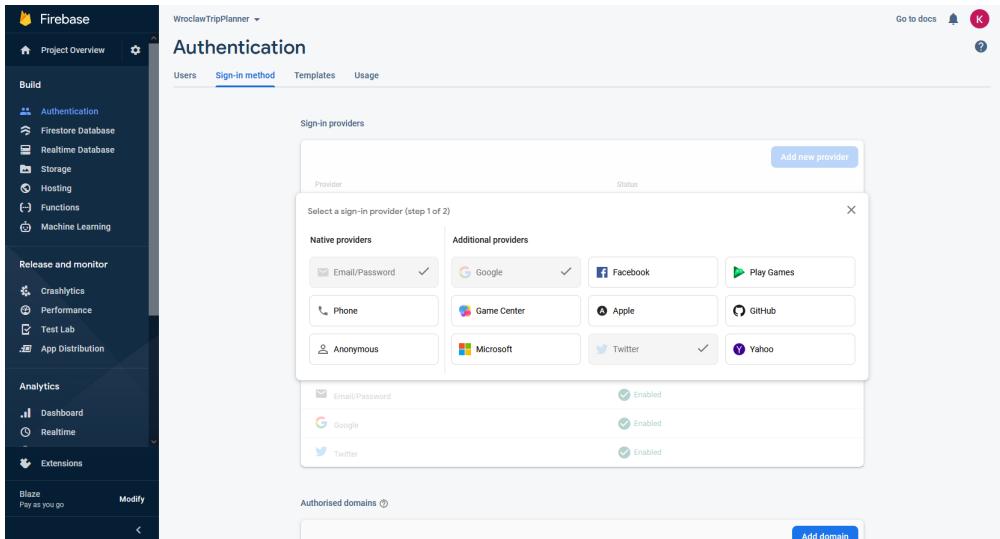
### **2.1.1 Firebase**

Firebase to rozwijana przez firmę Google platforma dedykowana do tworzenia i rozwijania aplikacji sieciowych i mobilnych. Firebase oferuje swoje narzędzia w formule Back-end Jako Usługa. Głównymi usługami Firebase [13], które zostały wykorzystane w omawianej aplikacji to Authentication (pol. Autoryzacja) oraz Realtime Database (pol. Baza danych w czasie rzeczywistym). Poza tymi produktami, których działanie omówione będzie w następnych punktach, Firebase oferuje także takie narzędzia jak Cloud Messaging, pozwalające na proste tworzenie aplikacji do przesyłania komunikatów, Analytics, czyli obszerne narzędzie pozwalające na analizę danych dotyczących zachowania użytkowników, czy Performance, pozwalające na dogłębne monitorowanie wydajności aplikacji.

### **2.1.2 Firebase Authentication**

Firebase Authentication [12] to jedno z głównych narzędzi Firebase pozwalające twórcy aplikacji na zbudowanie bezpiecznego systemu autoryzacji użytkowników.

Narzędzie to pozwala na zapewnienie użytkownikowi wygodnego systemu logowania, dającego możliwość autoryzacji zarówno za pomocą adresu e-mail, jak i zewnętrznych kont na serwisach takich jak Gmail, Twitter, czy Facebook, które mogą posłużyć za wiarygodne źródło autoryzacji. Firebase Authentication pozwala na zaawansowane dostosowanie systemu logowania za pomocą e-mail np. poprzez stworzenie autoryzacji kodem wysłanym do użytkownika za pomocą SMS lub e-mail. W niniejszej pracy zastosowana została funkcjonalność logowania poprzez e-mail oraz konto Twitter, jako przykład wykorzystania zewnętrznego konta. Jako że aplikacja nie zapisuje wrażliwych danych użytkownika, takich jak np. historia lokalizacji, nie zostały zastosowane żadne dodatkowe formy autoryzacji takie jak link potwierdzający wysłany na adres e-mail użytkownika.



Rysunek 3: Ekran wyboru sposobów autoryzacji w konsoli Firebase.

### 2.1.3 Firebase Cloud Firestore

Cloud Firestore [14] to produkt Firebase zapewniający twórcy aplikacji skalowalną, nie-relacyjną bazę danych pozwalającą na zapisywanie oraz sczytywanie danych w czasie rzeczywistym. Firestore powstał z myślą o zapewnieniu twórcom aplikacji mobilnych i webowych elastycznego narzędzia do tworzenia prostych w rozwoju baz danych.

Do głównych zalet Firestore należą [8]:

**Elastyczność** — Model danych Firestore zapewnia elastyczną hierarchię i strukturę danych.

**Skalowalność** — Wykorzystując rozbudowaną infrastrukturę chmury obliczeniowej Google, Firestore jest w stanie dynamicznie zapewniać użytkownikowi m.in. odpowiednie zasoby wymagane do bezproblemowego funkcjonowania bazy danych, niezależnie od ilości danych, replikację danych pomiędzy regionami świata, czy gwarancję spójności danych. [21]

**Wsparcie off-line** — Aplikacje wykorzystujące bazę danych Firestore na bieżąco zapisują stan bazy danych w pamięci podręcznej. Dzięki temu nawet bez dostępu do internetu aplikacja jest w stanie sczytywać i zapisywać informacje, które zostaną zsynchronizowane z chmurą obliczeniową kiedy urządzenie ponownie uzyska dostęp do sieci internetowej.

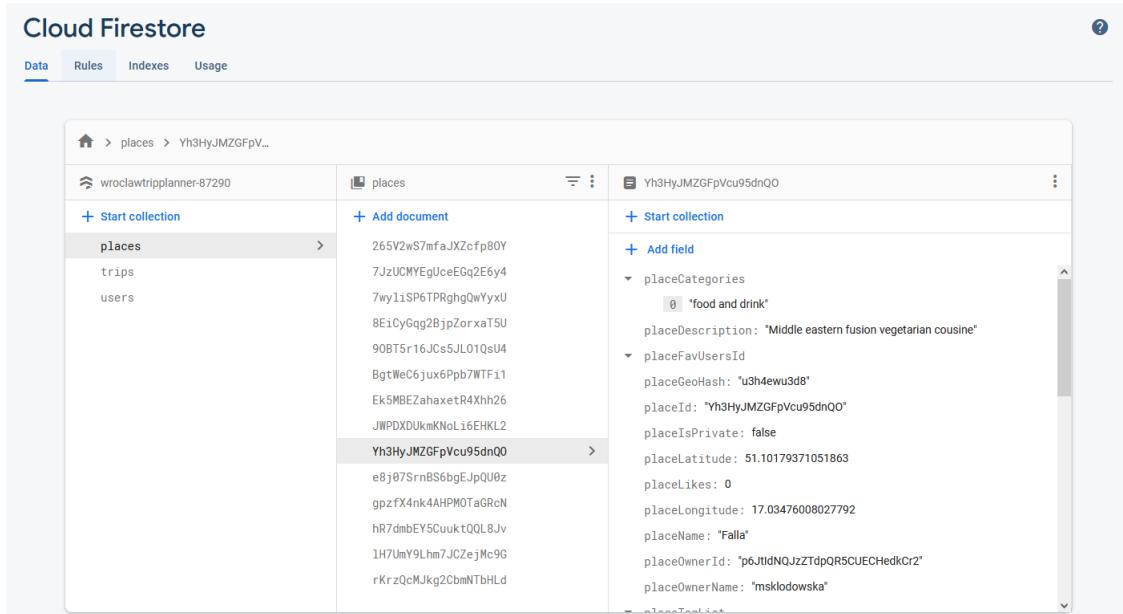
**Aktualizacje w czasie rzeczywistym** — odczyt i zapis do bazy danych Firestore następuje w czasie rzeczywistym. Pozwala to aplikacjom wykorzystującym Firestore na nasłuchiwanie zmian w bazie danych i umożliwia natychmiastowe reakcje na zachodzące w niej zmiany.

**Wyraziste zapytania** — Firestore pozwala na dokonywanie detalicznych zapytań pozwalających na skomplikowane kombinacje filtrowania i sortowania. Ponadto, baza

danych posiada system indeksowania zaprojektowany przez Google tak, aby wydajność zapytań była proporcjonalna do rozmiaru wynikowego zestawu, a nie samej bazy danych.

Zgodnie z nierelacyjnym modelem danych NoSQL struktury danych są reprezentowane w Cloud Firestore jako Dokumenty [8]. Dokument zawiera w sobie pola mapowane do wartości. System ten wspiera liczne typy danych, od prostych łańcuchów znaków i liczb, po bardziej złożone, zagnieżdżone obiekty. Kontenerem zawierającym w sobie dokumenty jest Kolekcja. Firestore pozwala także na tworzenie podkolekcji wewnętrz dokumentów. Pozwala to twórcy aplikacji na elastyczne budowanie struktury i hierarchii danych.

Baza danych Firestore umożliwia tworzenie skomplikowanych, ale zarazem wydajnych zapytań dzięki systemowi indeksowania. Indeks bazy danych to struktura danych, często w formie B-drzewa [3], mająca na celu optymalizację prędkości wyszukiwania danych w bazie danych kosztem dodatkowych zapisów danych i miejsca w pamięci, które przechowuje strukturę indeksów. Zamysłem tej struktury jest to, aby uniknąć procesu przeszukiwania każdego wiersza bazy danych w poszukiwaniu wymaganej informacji. Cloud Firestore automatycznie generuje indeksy [15] zapewniające szybki dostęp do prostych zapytań, np. zawierających wyszukiwanie i sortowanie względem jednego pola dokumentu. Dla bardziej skomplikowanych zapytań Firestore oferuje narzędzie dodawania złożonych indeksów. W wypadku kiedy z poziomu aplikacji wyślemy zapytanie wymagające złożonego indeksu, w treści błędu wygenerowany zostanie link automatycznie tworzący pożądaną przez twórcę aplikacji indeks.



Rysunek 4: Widok bazy danych Cloud Firestore w konsoli Firebase.

#### **2.1.4 Google Maps Platform**

Google Maps to internetowa aplikacja rozwijana przez Google oferująca szeroki zakres usług nawigacyjnych. Do oferty Google Maps należą m.in. zdjęcia satelitarne, mapy ulic, interaktywne, panoramiczne zdjęcia ulic w 360°, informacje o stanie ruchu drogowego przekazywane w czasie rzeczywistym i system nawigacyjny powiązany z systemem GPS. Satelitarne zdjęcia Google Maps prezentują świat z „widoku ptaka”, a wysokiej rozdzielczości zdjęcia zapewnione są głównie przez fotografię lotniczą. Zdjęcia są aktualizowane przez Google tak często, jak jest to możliwe. Google Maps jest także narzędziem, które pozwala na udostępnienie swojego biznesu, np. restauracji, czy innego rodzaju firmy. Dodane do bazy danych Google miejsce będzie wyświetlać się użytkownikom map, a korzystając z usług reklamowych firmy Google, klient może zapewnić sobie lepsze pozycjonowanie w wyszukiwaniu. Miejsca znajdujące się na mapie Google mogą być oceniane i komentowane przez użytkowników.

Google Maps Platform [24] to rozwijany przez Google zestaw licznych API i Zestawów Narzędzi Programistycznych (ang. Software Development Kit, SDK). Pozwalały one deweloperom na wbudowanie Google Maps w swoją aplikację mobilną bądź internetową oraz na otrzymywanie danych z Google Maps. Google Maps Platform składa się z trzech głównych produktów, w których pogrupowane są udostępniane przez firmę API [19].

Są to:

Maps (Mapy) — Zestaw zawierający w sobie główną funkcjonalność Google Maps Platform — możliwość wyświetlania interaktywnej mapy świata. Do tego zestawu należą: Maps JavaScript API, które pozwala na załączenie mapy w aplikacji internetowej, Maps SDK dla systemów Android i iOS, SDK pozwalające na integrację map w aplikacjach mobilnych na obu wymienionych platformach, Street View Imagery — API pozwalające na wykorzystanie wykonanych w 360° zdjęć ulic z Google Street View, oraz inne narzędzia związane z widokiem mapy.

Places (Miejsca) — Zestaw wykorzystujący bazę danych miejsc zawartych w Google Maps. Znajdują się w nich takie narzędzia jak Places API pozwalające na pobieranie opisów, zdjęć i adresów miejsc w bazie danych Google, Geocoding, pozwalający na dwustronną konwersję pomiędzy koordynatami a adresami lub Autocomplete, przeszukujący bazę miejsc w poszukiwaniu takich, które będą pasować do wyszukiwania użytkownika i zostaną zasugerowane w trakcie procesu wyszukiwania.

Routes (Trasy) — Zestaw narzędzi pomagających wytyczać trasy do pożądanych miejsc. Należą do niego: Directions API, pozwalające na rysowanie na ulicach mapy tras, wskazujących kierunek do danych koordynatów, czy Distance Matrix API pozwalające na obliczanie czasu podróży i dystansu pomiędzy wieloma celami podróży.

## 2.2 Aplikacja mobilna

Aplikacja mobilna stworzona została z myślą o działaniu na systemie operacyjnym Android. Jest to system operacyjny na urządzenia mobilne, oparty na zmodyfikowanym jądrze Linuxa. Jest rozwijany przez sojusz biznesowy Open Handset Alliance [29]. Android tworzony jest w modelu otwartego oprogramowania, finansowanego w znacznej mierze przez Google. Według badań prowadzonych przez firmę Statista, wraz ze styczniem 2022 Android jest najpopularniejszym systemem operacyjnym na urządzenia mobilne, posiadając prawie 70% udziału w rynku urządzeń mobilnych [28]. Ponadto system ten jest także wykorzystywany jako system operacyjny samochodowy (Android Auto), telewizyjny (Android TV) oraz na urządzenia mobilne typu Smartwatch (Wear OS).

### 2.2.1 Kotlin i Android Studio

Aplikacja mobilna została napisana w języku programowania Kotlin [27]. Jest to wieloplatformowy, statycznie typowany język programowania stworzony i rozwijany przez firmę JetBrains. Od 2019 roku jest on zalecany przez Google jako podstawowy język programowania na system operacyjny Android [5]. Język ten jest zaprojektowany z myślą o pełnym współdziałaniu z językiem Java, poprzednikiu Kotlinia jako głównego języka w programowaniu na Androide. Poza językiem Kotlin wykorzystany został także język znaczników XML, służący do projektowania warstwy graficznej aplikacji, oraz Gradle, narzędzie do automatyzacji budowania projektów, wspierane jako oficjalny system budowania projektów na system Android.

Aplikacja została napisana, wykorzystując Zintegrowane Środowisko Deweloperskie (ang. Integrated Development Environment, IDE) Android Studio [17], Jest to oficjalne IDE systemu operacyjnego Android stworzone we współpracy firm Google i JetBrains. Oferuje ono wiele narzędzi wspierających budowanie aplikacji na system Android. Android Studio zapewnia m.in. inteligentny edytor kodu uzupełniający kod pisany w Kotlinie, Javie lub C/C++, graficzny edytor wspomagający tworzenie wizualnej części aplikacji, wbudowany, prosty w obsłudze emulator urządzenia mobilnego z systemem Android, czy zintegrowany system budowania projektów Gradle.

## 2.2.2 Model danych

W aplikacji wykorzystane są cztery klasy danych mające reprezentować obiekty umieszczone w bazie danych.

Są to:

User (Użytkownik) — jest to klasa zawierająca pseudonim użytkownika, jego unikalny identyfikator oraz adres e-mail.

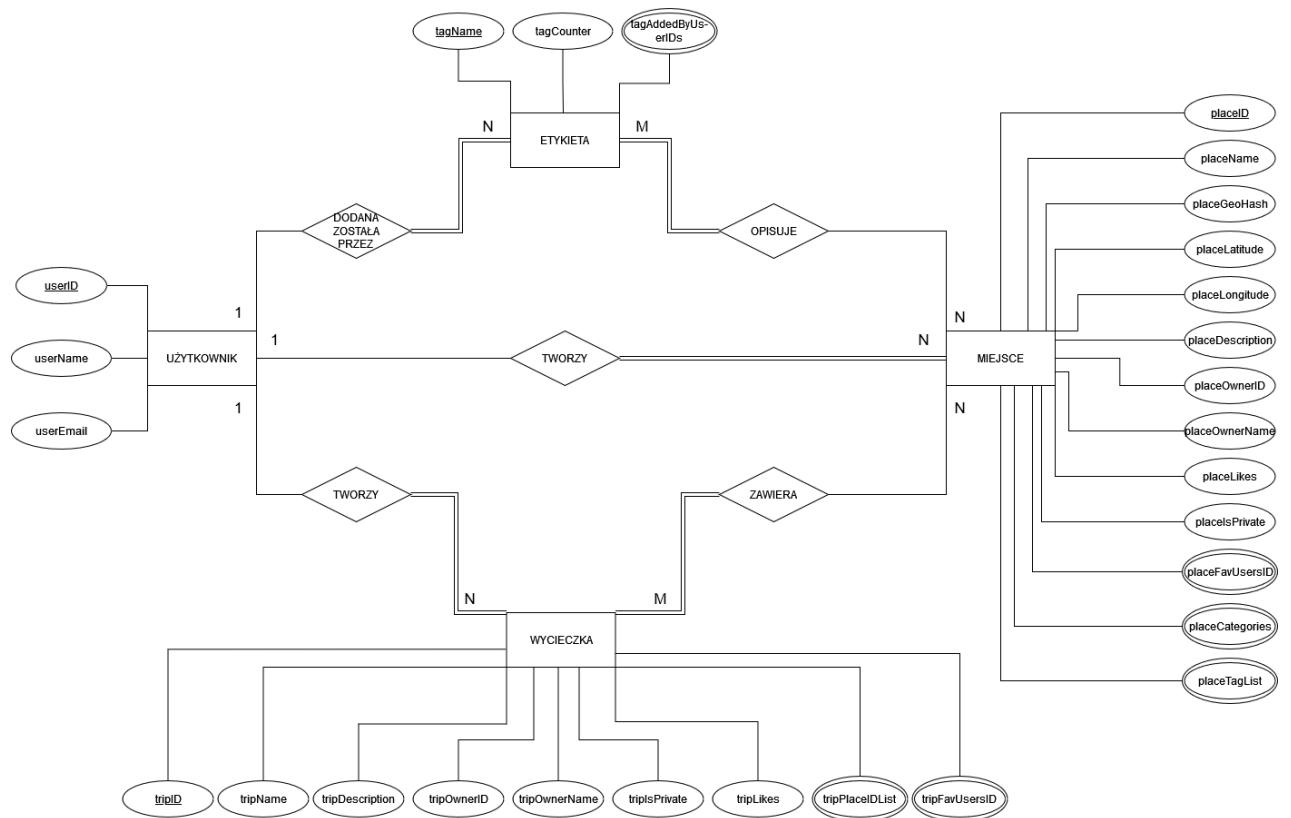
Place (Miejsce) — najbardziej istotna klasa danych w aplikacji, reprezentująca miejsca dodawane na mapie przez użytkowników. Użytkownik dodający miejsce ma możliwość ustalenia jego Nazwy, Opisu, Kategorii (do wyboru z pięciu gotowych) oraz prywatność (prywatne miejsce nie zostanie wyświetcone na mapie podczas wyszukiwania). Ponadto, miejsce zawiera w sobie: unikalny identyfikator, długość i szerokość geograficzną (sczytane z położenia kurSORA w trakcie dodawania miejsca), GeoHash (typ danych, będący zaszyfrowanymi współrzędnymi geograficznymi miejsca, mający usprawnić wydajność geograficznych zapytań do bazy danych [11]), identyfikator i pseudonim autora miejsca, listę kategorii (ponieważ miejsce może należeć do więcej niż jednej kategorii na raz), listę etykiet przypisanych miejscu przez wszystkich użytkowników, licznik polubień danego miejsca, listę identyfikatorów użytkowników, którzy polubili dane miejsce oraz flagę określającą prywatność miejsca.

Tag (Etykieta) — prosta klasa tworząca system etykiet. System ten inspirowany jest systemem etykiet wykorzystanym przez firmę Valve w platformie dystrybucji cyfrowej Steam. Jest to platforma, która łączy w sobie funkcje platformy dystrybucji programów komputerowych (głównie gier komputerowych i oprogramowania z nimi związanego) oraz serwisu społecznościowego. System etykiet stworzony przez Valve pozwala użytkownikom na dodawanie do znajdujących się w ofercie programów etykiet w postaci kilku słów np. „łamigłówki”, „trudna”, „gra z otwartym światem”. Znajdujących się w bazie danych etykiet można używać, aby filtrować wyniki wyszukiwania w ofercie platformy Steam [2]. System znaczników obecny w aplikacji mobilnej podobnie jak system Valve pozwala każdemu użytkownikowi na dodanie etykiety do każdego publicznego miejsca. Z uwagi na rozmiar projektu i ograniczenia techniczne, system ten nie umożliwia użytkownikom tworzenia nowych etykiet, a jedynie na korzystanie z gotowego zbioru etykiet prezentowanego w menu dodawania etykiet. Klasa etykiety składa się z nazwy etykiety, licznika dodań, oraz listy użytkowników, którzy dodali etykietę do miejsca. W menu wyświetlającym szczegóły miejsca znajduje się lista wyświetlająca najpopularniejsze etykiety. Każdy użytkownik może dodać konkretną etykietę tylko raz, a w bazie danych zliczane jest, ile razy dana etykieta została dodana do danego miejsca. To pozwala na wyświetlenie w menu przedstawiającym szczegóły miejsca, dziesięciu najpopularniejszych etykiet danego miejsca. Pozwala to użytkownikom na określanie tego, jakie aktywności można w danym miejscu wykonywać np. „muzeum”, „pomnik”, „bar”, „kino” itp. Najpopularniejsze etykiety powinny najlepiej reprezentować przeznaczenie danego miejsca.

Trip (Wycieczka) — jest to klasa reprezentująca zbiór miejsc, wykorzystywana do two-

rzenia zapytań o trasę w Directions API. Użytkownik tworzący wycieczkę może ustalić jej nazwę, opis i prywatność, a następnie może dodawać do niej miejsca. Każda wycieczka może zawierać w sobie do 10 miejsc, z uwagi na koszt odpowiedzi na bardziej złożone zapytania w Directions API. Poza polami ustalonymi przez użytkownika wycieczka zawiera także unikalny identyfikator, identyfikator i pseudonim autora, licznik polubień danej wycieczki oraz listę identyfikatorów użytkowników, którzy polubili daną wycieczkę.

Dokładniejsza reprezentacja klas danych oraz zachodzących pomiędzy nimi relacji przedstawiona jest na poniższym diagramie (patrz Rysunek 5).



Rysunek 5: Diagram przedstawiający relacje w bazie danych.

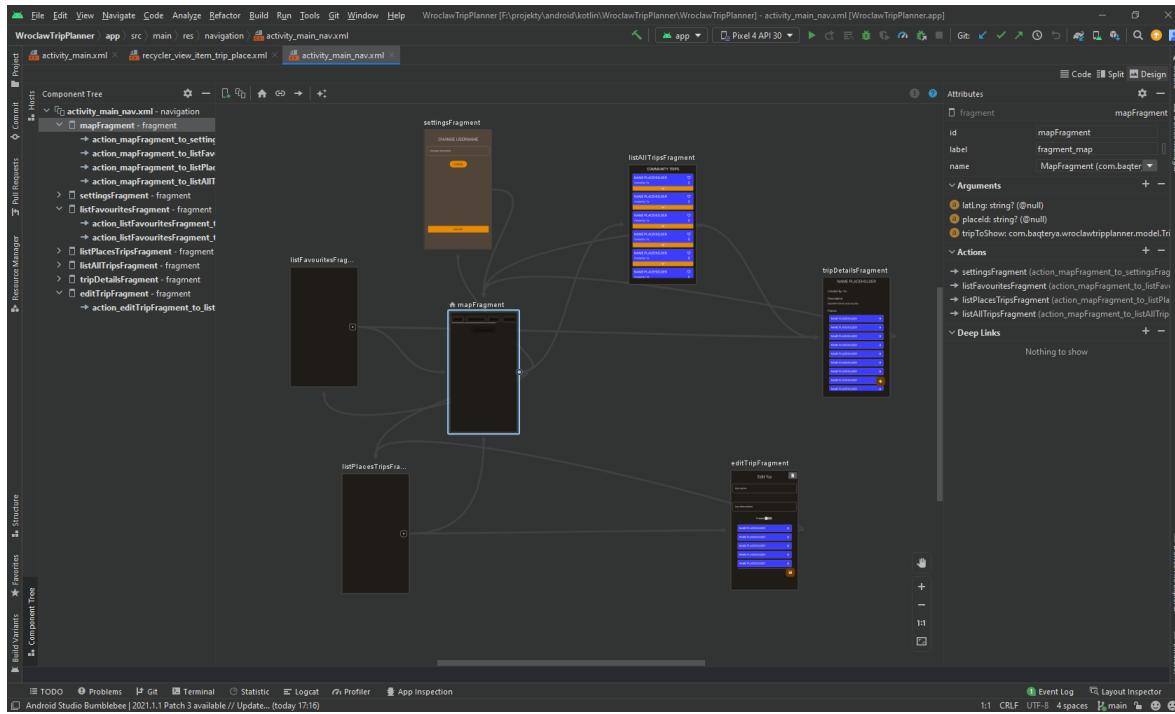
## 3 Implementacja

### 3.1 Interfejs użytkownika

Bazowym sposobem budowy interfejsów użytkownika w aplikacjach na system operacyjny Android jest tworzenie ich za pomocą języka znaczników XML. Interfejs składany jest ze znaczników zapewnionych przez wbudowane czy też zewnętrzne biblioteki. Podstawowym elementem interfejsu jest Widok (ang. View), bazowa klasa każdego interaktywnego komponentu interfejsu użytkownika. Widoki pogrupowane są w Grupy Widoków (ang. ViewGroup) zwane zazwyczaj Layoutami [9].

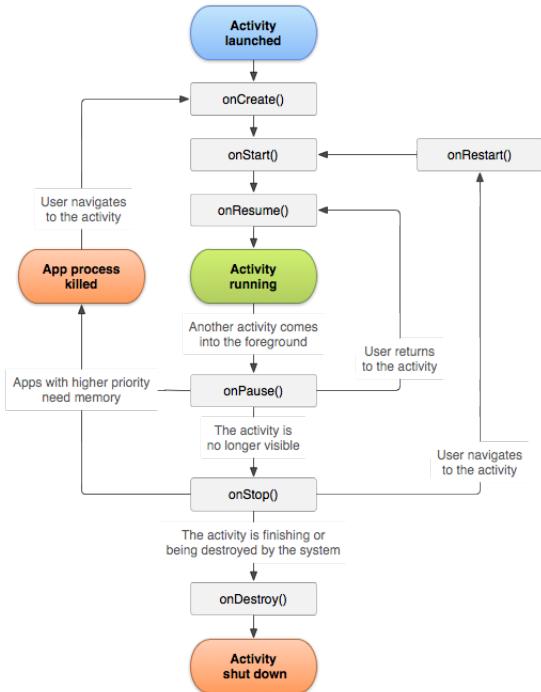
Za wizualną oprawę aplikacji odpowiedzialne są głównie elementy interfejsu oferowane przez Material Design. Jest to zbiór elementów interfejsu i język designu stworzony i rozwijany przez Google z myślą o tworzeniu aplikacji mobilnych. W procesie tworzenia aplikacji wykorzystane zostały takie elementy jak Bottom App Bar, pozwalający na wygodną nawigację pomiędzy ekranami aplikacji, czy Card View, zapewniające elegancki sposób wyświetlania elementów list. Najbardziej warto skupić się jednak na elemencie nazwanym Chip [7]. Są to małe elementy interfejsu, na które składa się przede wszystkim obwódka oraz wyświetlany na nich tekst. Z pozoru mogą wydawać się typem zwykłego przycisku, ale różni je stojąca za nimi idea projektowa. Przycisk jest elementem, od którego oczekiwana jest konsekwencja, zarówno wizualna, w wyglądzie, jak i w działaniu, wywołując oczekiwana przez użytkownika akcję. Chip tym różni się od przycisku, że jego cel jest bardziej dynamiczny i często reprezentuje grupę interaktywnych elementów [6]. Chipy można podzielić na cztery grupy: Pomocnicze (ang. Assist), najbliższe w swoim działaniu do przycisków, powinny pojawiać się w interfejsie dynamicznie, i kontekstowo, Filtrujące, za pomocą zmiany koloru i pojawienia się ikony obrazują wybór użytkownika, Wprowadzające (ang. Input), zastępujące dyskretne dane wprowadzone przez użytkownika, oraz Sugerujące (ang. Suggestion), mające graficznie reprezentować dynamicznie generowane podpowiedzi. W aplikacji mobilnej zastosowane zostały Chipy Pomocnicze, w konstrukcji przejrzystego interfejsu komponującego się z wyświetlana w tle mapą, oraz Filtrujące zapewniające użytkownikowi responsywny system wyboru kategorii, oraz etykiet dla Miejsc. Aplikacje projektowane na system Android posiadają także ujednolicony system tworzenia motywów kolorystycznych. W aplikacji zastosowany został motyw stworzony za pomocą Material Theme Builder, narzędzia wspomagającego poprawny dobór kolorów oferowanego przez Material Design [16].

Tworzenie aplikacji na system Android polega także w dużej mierze na korzystaniu z rozwijanego przez Google obszernego zestawu bibliotek Android Jetpack [22]. Zawiera on w sobie podstawowe komponenty aplikacji mobilnych, takich jak np. Aktywności (pojedynczy ekran interfejsu użytkownika, analogiczny do okna w aplikacji komputerowej), czy biblioteki obsługujące Material Design. Oferuje on także szereg rozwiązań usprawniających projektowanie aplikacji takich jak np. Navigation Component, ułatwiający poruszanie się pomiędzy ekranami aplikacji i oferujący wizualną reprezentację akcji nawigacyjnych, które mogą zostać podjęte przez użytkownika (patrz Rysunek 6).



Rysunek 6: Akcje nawigacyjne reprezentowane na diagramie Naviagtion Component.

W odróżnieniu od standardowego stylu tworzenia aplikacji, w którym kod inicjowany jest za pomocą metody *main()* aplikacje na system Android wykorzystują Aktywności. Aktywność reprezentuje pojedynczy ekran interfejsu użytkownika, na którym wyświetlane są interaktywne widoki. To w nich zawiera się kod odpowiadający za obsługę wszelkich funkcji aplikacji takich jak np. efekt wcisnięcia przez użytkownika danego przycisku, wprowadzenia tekstu itp. Wizualna warstwa Aktywności obsługiwana jest przez plik XML zawierający umiejscowienie i właściwości elementów interfejsu użytkownika. Aktywności posiadają określony cykl życia, który określa jej aktualny stan, oraz stan danych zadeklarowanych w konkretnej aktywności (patrz Rysunek 7). Informacja wprowadzona przez użytkownika w jednej aktywności zostanie skasowana z pamięci telefonu w momencie zatrzymania się aktywności, więc twórca aplikacji musi zapewnić poprawny przesył danych pomiędzy nimi.



Rysunek 7: Cykl życia Aktywności [25].

Kolejnym istotnym elementem interfejsu użytkownika jest klasa Fragment, stanowiąca część interfejsu przeznaczona do wielokrotnego użytku. Każdy fragment musi zawierać się w aktywności, a każda aktywność może zarządzać dowolną ilością fragmentów. Są one łatwe w modulacji i pomagają dyskretnie dzielić interfejs użytkownika. W trakcie działania aktywności fragmenty przechodzą swój własny, analogiczny, ale oddzielny cykl życia. Dużą zaletą korzystania z fragmentów jest to, jak szerokie jest wsparcie dla aplikacji opartych głównie na fragmentach. Dobrym przykładem jest wspomniany wcześniej Navigation Component. Pojedynczy komponent tego typu implementowany jest w konkretnej aktywności. Niniejsza aplikacja zawiera w sobie dwie aktywności: jedną odpowiadającą za procesy logowania i rejestracji oraz drugą, zawierającą główną, użytkową część aplikacji. Dla przykładu, w pierwszej z nich użytkownik może nawigować pomiędzy trzema fragmentami odpowiedzialnymi za wyświetlenie ekranu powitalnego, ekranu logowania, oraz ekranu rejestracji. Z punktu widzenia użytkownika nie ma różnicy pomiędzy użyciem fragmentu lub aktywności, jednak dla twórcy aplikacji rozwiązanie to oferuje znacznie większą elastyczność i wygodę.

## 3.2 Integracja z Firebase

Tworzenie aplikacji korzystającej z funkcji oferowanych przez Firebase należy rozpocząć od utworzenia nowego projektu w witrynie internetowej Firebase. W trakcie tego procesu wyświetlane są instrukcje przedstawiające kroki, które należy podjąć, aby połączyć aplikację z projektem chmury obliczeniowej. Sprowadza się to przede wszystkim do umieszczenia w folderze zawierającym kod źródłowy, pliku *google-services.json*, automatycznie wygenerowanego w trakcie procesu. Zawiera on w sobie informacje o projekcie oraz klucze API pozwalające na połączenie z chmurą obliczeniową. W następnej kolejności należy do aplikacji mobilnej dołączyć biblioteki, na które składa się Firebase SDK for Android. Odbywa się to za pomocą systemu budowania projektów Gradle (patrz Listing 1).

```
1 //Firebase
2 implementation platform('com.google.firebaseio:bom:29.0.0')
3 implementation 'com.google.firebaseio:firebase-analytics'
4 implementation 'com.google.firebaseio:firebase-auth-ktx:21.0.2'
5 implementation 'com.google.firebaseio:firebase-firebase-ktx:24.0.2'
6
```

Listing 1: Biblioteki Firebase SDK zaimplementowane w aplikacji

### 3.2.1 Implementacja Firebase Authentication

Pierwszą powiązaną z Firebase funkcjonalnością aplikacji, z którą zetknie się nowy użytkownik aplikacji, jest Firebase Authentication, odpowiadające za rejestrację oraz logowanie użytkowników. W aplikacji zaimplementowana została możliwość tworzenia konta za pomocą adresu e-mail oraz logowanie się za pośrednictwem konta na platformie Twitter. Z poziomu SDK dostęp do funkcji autoryzacyjnych otrzymujemy za pomocą obiektu *Firebase.auth*. To na nim wywoływane są metody tworzące nowe konto użytkownika oraz pozwalające na zalogowanie. Na Listingu 2 zobaczyć można przykład zastosowania Firebase SDK for Android do utworzenia konta użytkownika za pomocą danych (email, password, oraz username) wprowadzonych przez niego w polach tekstowych oraz metodę pozwalającą na zalogowanie za pomocą konta w serwisie Twitter. Kiedy rejestracja bądź logowanie za pomocą konta Twitter zakończy się pomyślnie, konto użytkownika zostanie zapisane w serwisie autoryzacyjnym Firebase dostępnym za pomocą konsoli Firebase. Dodatkowo dane użytkownika zostaną wraz z jego pseudonimem zapisane w bazie danych Firestore, aby ułatwić komunikację w bazie danych. Hasło użytkownika pozostaje dyskretne nawet dla twórcy aplikacji. Za bezpieczeństwo tajnych danych odpowiadają zabezpieczenia chmury obliczeniowej Google. Twórca projektu, w razie potrzeby może uzyskać zaszyfrowane wersje haseł, np. na potrzeby migracji danych na inny serwer.

```

1 auth = Firebase.auth
2
3 /**
4 * stworzenie konta za pomocą email'u i hasła wprowadzonego
5 * przez użytkownika do pol tekstowych
6 */
7 Firebase.auth.createUserWithEmailAndPassword(email, password)
8
9
10 /**
11 * kod odpowiedzialny za połaczenie z serwisem Twitter
12 * otwiera w oknie aplikacji okno przeglądarki, w której
13 * użytkownik zostanie poproszony o autoryzacje za pomocą
14 * danych konta Twitter
15 */
16 val provider = OAuthProvider.newBuilder("twitter.com")
17
18 val pendingTaskResult = auth.pendingAuthResult
19 if (pendingTaskResult == null) {
20     auth.startActivityForSignInWithProvider(
21         requireActivity(), provider.build()
22     )
23 }
24

```

Listing 2: Utworzenie konta użytkownika za pomocą *Firebase.auth*

### 3.2.2 Implementacja Firebase Cloud Firestore

Dostęp do funkcjonalności Cloud Firestore otrzymać można za pomocą obiektu *Firebase.firestore*. Za jego pomocą możemy dokonywać zapytań do bazy danych i otrzymywać odpowiedzi zwrotne. Przykład zaobserwować można w Listingu 13, który pokazuje, jak za pomocą Firebase SDK wykonujemy proste polecenie dodania nowego miejsca do kolekcji *places*.

```

1 db = Firebase.firestore
2
3 db.collection("places").add(newPlace)
4

```

Listing 3: Dodanie miejsca do bazy danych za pomocą *Firebase.firestore*

Jak zauważać można na Listingu 13 do bazy danych dodany jest obiekt o nazwie *newPlace*. Jest to obiekt klasy *Place*, który w procesie dodawania do bazy danych konwertowany jest na analogiczny słownik danych, umieszczany w drzewie JSON bazy danych. Każdy typ danych wykorzystany w aplikacji posiada klasę w Kotlinie modelującą jego pola w bazie danych. Na przykładzie Listingu 4 zaobserwować można jak modelowany, jest obiekt miejsca z poziomu aplikacji.

```

1 /**
2 * Data class model of a Place object in the database
3 *
4 * @property placeFavUsersId: List of users that added
5 * the place to their favourites
6 */
7 data class Place(
8     var placeId: String? = null,
9     var placeName: String? = null,
10    var placeGeoHash: String? = null,
11    var placeLatitude: Double? = null,
12    var placeLongitude: Double? = null,
13    var placeDescription: String? = null,
14    var placeOwnerId: String? = null,
15    var placeOwnerName: String? = null,
16    var placeCategories: ArrayList<String> = arrayListOf(),
17    var placeTagList: ArrayList<Tag> = arrayListOf(),
18    var placeFavUsersId: ArrayList<String> = arrayListOf(),
19    var placeLikes: Int = 0,
20    var placeIsPrivate: Boolean = false,
21 )
22

```

Listing 4: Klasa danych Miejsca zmodelowana w języku Kotlin

Powyższa klasa danych zostaje przez Firebase SDK przekonwertowana do formy, którą zaobserwować można na Rysunku 4. W wypadku kiedy dane z bazy danych muszą zostać wykorzystane w aplikacji, można wykorzystać konwersję w drugą stronę, która możliwa jest za pomocą funkcji *toObject()* dostępnej w Firebase SDK. Przykład pozyskania danych z bazy, aby wykorzystać je w aplikacji, zaobserwować można na Listingu 5. Obiekt JSON przekształcany jest tam do analogicznej mu klasy *Place*, aby w przejrzysty sposób uzyskiwać dostęp do danych, tutaj pozycji geograficznej, danego miejsca i umieścić jego znacznik na mapie, zatytułowany nazwą miejsca. Klasa Miejsca musi posiadać takie same nazwy pól jak klucze słownika JSON odpowiadającego Miejscu w bazie danych.

```

1 // pobranie miejsca z bazy danych
2 db.collection("places").document(placeId).get()
3 // dodanie funkcji nasluchujacej sukcesu operacji
4 .addOnSuccessListener {
5     // konwersja do obiektu klasy Place
6     val place = it.toObject(Place::class.java)!!
7     // umiejscowienie na mapie pinezki wykorzystujac
8     // dane pobrane z obiektu klasy Place
9     map.addMarker(
10         MarkerOptions()
11             .title(place.placeName)
12             .position(
13                 LatLng(
14                     place.placeLatitude!!,
15                     place.placeLongitude!!
16                 )
17             )
18             .icon(bitmapDescriptorFromVector(
19                 requireContext(), R.drawable.ic_map_pin
20             )))
21     )?.showInfoWindow()
22 }
23

```

Listing 5: Fragment kodu pobierający i wykorzystujący miejsce znajdujące się w bazie danych

Jak wspomniane zostało w rozdziale 2 aplikacja oddziela obsługę bazy danych od obsługi interfejsu korzystając ze wzorca projektowego MVVM. W aplikacji utworzona została klasa FirestoreViewModel, która odpowiada za bezpośrednią komunikację z bazą danych, oddzielając ją tym samym od części obsługującej interfejs użytkownika. Dla przykładu w Listingu 6 zobaczymy fragment kodu, który pobiera dane wprowadzone przez użytkownika w trakcie tworzenia nowej wycieczki, aby następnie przesłać je do obiektu FirestoreViewModel w celu wpisania wycieczki do bazy danych Firestore.

```

1 // deklaracja obiektu ViewModel
2 val firestoreViewModel = FirestoreViewModel()
3 // deklaracja obiektu reprezentujacego nowa wycieczke
4 val newTrip = Trip()
5
6 // program sprawdza, czy nazwa i opis wycieczki
7 // zostaly wprowadzone przez uzytkownika
8 if (inputCheck(tripName) && inputCheck(tripDescription)) {
9     // uzupełnienie pol wycieczki danymi
10    // wprowadzonymi przez uzytkownika
11    newTrip.tripName = tripName
12    newTrip.tripDescription = tripDescription
13    newTrip.tripIsPrivate = tripIsPrivate.isChecked
14
15    // wywolanie funkcji dodajacej miejsce do bazy danych
16    // z obiektu ViewModel
17    firestoreViewModel.addTripToFirestore(
18        currentPlace.placeId!!, newTrip
19    )
20 } else {
21     // wyswietlenie powiadomienia w razie nie uzupełnienia
22     // przez uzytkownika wszystkich wymaganych parametrow
23     Toast.makeText(
24         context,
25         "Please fill all fields.",
26         Toast.LENGTH_SHORT
27     ).show()
28 }
29

```

Listing 6: Fragment kodu obsługujący interfejs użytkownika i komunikujący się z bazą danych za pomocą FirestoreViewModel

### 3.3 Integracja z Google Maps Platform

Głównym elementem aplikacji jest zastosowanie w niej funkcji oferowanych przez Google Maps Platform. Jak wspomniane zostało w rozdziale 2.1, tworzenie projektu w Firebase jest jednoznaczne z utworzeniem projektu w Google Cloud Platform. Dzięki temu w konsoli Google Cloud można skorzystać z produktu APIs and Services, aby do projektu aplikacji Android dodać Maps SDK for Android oraz Directions API. Następnie, wygenerowany w Google Cloud Console klucz API należy dodać do aplikacji Android. Zapewnia to połączenie z usługami Google Maps Platform.

#### 3.3.1 Implementacja Maps SDK for Android

Google Maps SDK for Android zawiera w sobie podstawowe funkcjonalności związane z wyświetlaniem dynamicznej mapy i interakcji z nią. Głównym ekranem aplikacji jest ekran wyświetlający mapę. Implementuje on interfejs programistyczny *OnMapReadyCallback*, który umożliwia wprowadzenie do aplikacji metody nasłuchującej momentu, w którym mapa będzie gotowa do wyświetlenia. To właśnie tam konfigurowane są

ustawienia mapy i wywoływana jest funkcja znajdująca lokację urządzenia użytkownika. Na Listingu 7 zaobserwować można proces konfiguracji ustawień mapy. W procesie wyszukiwania miejsc na mapie wyświetcone zostają pinezki oznaczające konkretne miejsca. Za pomocą funkcjonalności zaoferowanej przez Maps SDK możliwe jest elastyczne dostosowanie sposobu wyświetlania obiektów na mapie. Proces wyświetlania na mapie pinezki reprezentującej konkretne miejsce można zaobserwować w Listingu 5.

Aby aplikacja mogła wykorzystywać w swoim działaniu informacje o położeniu urządzenia użytkownika, musi on udzielić aplikacji odpowiednich uprawnień.

```
1 override fun onMapReady(map: GoogleMap) {
2     // włączenie wyświetlania lokacji urządzenia
3     // na mapie
4     map.isMyLocationEnabled = true
5     map.uiSettings.isMyLocationButtonEnabled = true
6     map.uiSettings.isCompassEnabled = false
7     map.isBuildingsEnabled = true
8     // ustawienie stylu graficznego mapy zdefiniowanego
9     // w map_style.json
10    map.setMapStyle(MapStyleOptions.loadRawResourceStyle(
11        requireContext(), R.raw.map_style
12    ))
13    locationButtonSettings()
14
15    // ograniczenie możliwości przewijania mapy
16    // do obrebu Wrocławia
17    val latLngBounds = LatLngBounds(
18        LatLng(51.047, 16.936),
19        LatLng(51.143, 17.1)
20    )
21    map.setLatLngBoundsForCameraTarget(latLngBounds)
22    // ograniczenie możliwości oddalania i przyblizania
23    // kamery do zadanych wartości
24    map.setMinZoomPreference(MIN_ZOOM)
25    map.setMaxZoomPreference(MAX_ZOOM)
26}
```

Listing 7: Konfiguracja ustawień mapy w funkcji *OnMapReadyCallback*

### 3.3.2 Implementacja Directions API

W przeciwieństwie do API wyświetlającego samą mapę Directions API nie posiada biblioteki dedykowanej dla tworzenia aplikacji na system Android. Oznacza to, że należy skorzystać z innej metody wysyłania zapytań do API. W aplikacji została wykorzystana biblioteka Volley. Jest to tworzona przez Google biblioteka przeznaczona do wykonywania zapytań HTTP i łączenia z różnego rodzaju usługami sieciowymi [26]. Directions API wykorzystane jest w aplikacji w celu rysowania trasy do wybranego przez użytkownika miejsca lub aby narysować trasę wycieczki, przechodzącą przez wszystkie zawarte w niej miejsca. Za pomocą Volley wysyłane jest zapytanie do Directions API, a następnie po otrzymaniu odpowiedzi w postaci drzewa JSON wydobywane są in-

strukcje potrzebne do narysowania wytyczonej ścieżki na widzianej przez użytkownika mapie. Przykładową funkcję wysyłającą zapytanie do API i rysującą ścieżkę prowadzącą z lokacji użytkownika do wybranego miejsca zobaczyć można na Listingu 8. Directions API wysyła odpowiedź w formie drzewa JSON zawierającego dokładne instrukcje podróży od startu do celu. W zapytaniu można sprecyzować także przystanki, przez które przechodzić ma trasa, co zastosowane zostało w przypadku rysowania trasy Wycieczki. Odpowiedź z Directions API składa się z hierarchii elementów: trasy (ang. routes), najwyższy element, reprezentuje trasę pomiędzy początkiem a celem, odcinki (ang. legs), trasa pomiędzy przystankami trasy, kroki (ang. steps), najniższe elementy trasy, zawierają konkretne, dokładne instrukcje kierujące [10].

```

1 fun drawPathToPlace(
2     placeLatitude: Double, placeLongitude: Double, placeId: String
3 ) {
4     // tablica przechowujaca w sobie pozycje przez ktore musi
5     // przejsc sciezka prowadzaca do miejsca
6     val path: MutableList<List<LatLng>> = ArrayList()
7
8     // zapytanie wysylane do Directions API
9     // zawiera w sobie miejsce rozpoczęcia i konca trasy
10    // oraz klucz API laczacy aplikacje z Directions API
11    val urlDirections =
12        "https://maps.googleapis.com/maps/api/directions/json?" +
13            "origin=${map.myLocation.latitude}, " +
14            "${map.myLocation.longitude}&" +
15            "destination=${placeLatitude},${placeLongitude}&" +
16            "key=AIzaSyCNYAkT-Zg-NY4md_kespvcX7fV_ff8KQw"
17
18    // wysyłanie zapytania i nasluchiwanie odpowiedzi obsługiwane
19    // jest przez oferowany przez Volley obiekt StringRequest
20    val directionsRequest = object : StringRequest(
21        Method.GET,
22        urlDirections,
23        Response.Listener { response ->
24            // z obiektu JSON odebranego z API wyciągane są
25            // routes - trasa pomiędzy wybranym początkiem i końcem
26            val responseJSON = JSONObject(response)
27            val routes = responseJSON.getJSONArray("routes")
28
29            // legs - składowe routes; w jednym obiekcie routes
30            // znajduje się leg dla każdego z przystanków, w tym
31            // przypadku jedno.
32            val legs = routes.getJSONObject(0).getJSONArray("legs")
33
34            // steps - najniższa jednostka trasy, określająca
35            // pojedynczą instrukcję trasy
36            // pozycje każdego step dodawane są do sciezki, która
37            // zostanie narysowana
38            val steps = legs.getJSONObject(0).getJSONArray("steps")
39            for (i in 0 until steps.length()) {
40                val points = steps.getJSONObject(i)
41                    .getJSONObject("polyline").getString("points")
42                path.add(PolyUtil.decode(points))
43            }
44
45            // pętla nanosząca linie sciezki na mapę
46            for (i in 0 until path.size) {
47                map.addPolyline(PolylineOptions().addAll(path[i])
48                    .color(Color.rgb(230, 138, 0)))
49            }
50        },
51
52    val requestQueue = Volley.newRequestQueue(requireContext())
53    requestQueue.add(directionsRequest)
54 }

```

Listing 8: Funkcja pobierająca trasę z Directions API i rysująca ją na mapie

## 4 Prezentacja aplikacji

Aplikację mobilną podzielić można na dwa główne widoki. Pierwszy, z którym zetknie się użytkownik, to ekran autoryzacji. Tam możliwe jest dla niego stworzenie nowego konta lub zalogowanie się na istniejące już konto. Drugi to ekran główny, w którym odbywają się wszystkie kluczowe aktywności związane z głównym celem aplikacji, jakim jest stworzenie narzędzia do dzielenia się ciekawymi miejscami i planami wycieczek.

### 4.1 Ekrany autoryzacji użytkownika

#### 4.1.1 Ekran powitalny

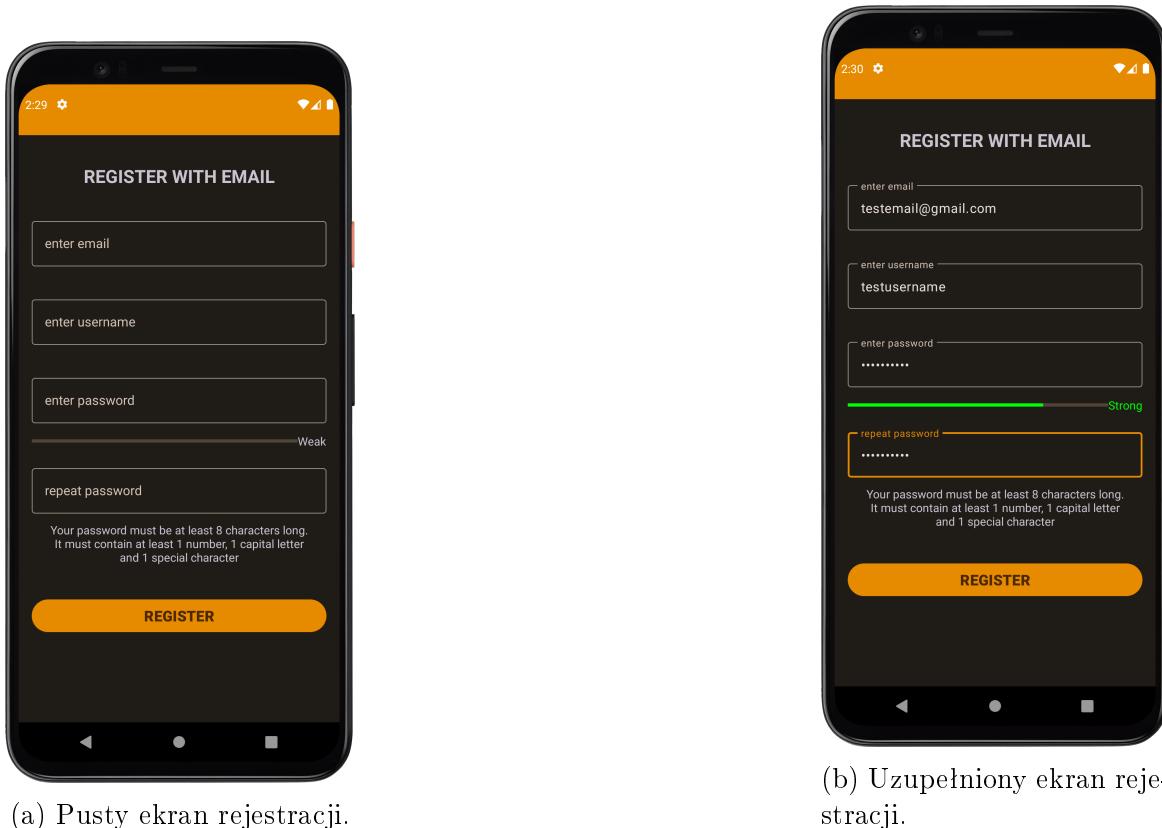
Pierwszym ekranem, który zobaczy zostanie przez nowego użytkownika to ekran powitalny, widoczny na rysunku 8. Na górze ekranu powitalnego widoczne jest zdjęcie panoramy Wrocławia widoczne z kościoła p.w. św. Elżbiety [4]. Poniżej znajduje się tekst powitalny. Na samym dole, w wygodnym w dostępie miejscu ułożone są przyciski wyboru akcji. Pierwszy z nich, zatytułowany „SIGN UP” przekieruje użytkownika do okna rejestracji. Drugi z nich otworzy okno przeglądarki, w którym użytkownik będzie mógł zalogować się za pomocą konta Twitter. Na samym dole znajduje się widok tekstowy dający powracającemu użytkownikowi możliwość zalogowania się za pomocą adresu e-mail. Każda pomyślana próba autoryzacji zakończy się przeniesieniem użytkownika z aktywności autoryzacji do głównej aktywności.



Rysunek 8: Ekran powitalny aplikacji.

#### 4.1.2 Ekran rejestracji

Ekran rejestracji, widoczny na rysunku 9, składa się przede wszystkim z pól tekstowych, do których użytkownik może wprowadzić pożądane informacje. Poprawność wprowadzanych danych jest w trakcie procesu rejestracji weryfikowana. Adres e-mail musi być odpowiednio sformatowany, pseudonim musi być unikalny, a hasło musi spełniać odpowiednie kryteria bezpieczeństwa. Poziom bezpieczeństwa hasła ukazywany jest za pomocą paska postępu, zaczerpniętego z biblioteki Material Design. Za pomocą kolorów i tekstu sygnalizuje on użytkownikowi to, czy hasło spełnia odpowiednie warunki. Aby móc utworzyć konto, hasło musi spełniać warunki opisane w instrukcji znajdującej się u dołu ekranu. Na samym dole znajduje się przycisk „REGISTER”. Wciśnięcie go powoduje weryfikację danych. W przypadku, jeśli któryś z pól wypełnione jest niepoprawnie, wyświetlane jest powiadomienie, a kursor użytkownika automatycznie przemieszcza się w wymagające uwagi pole.

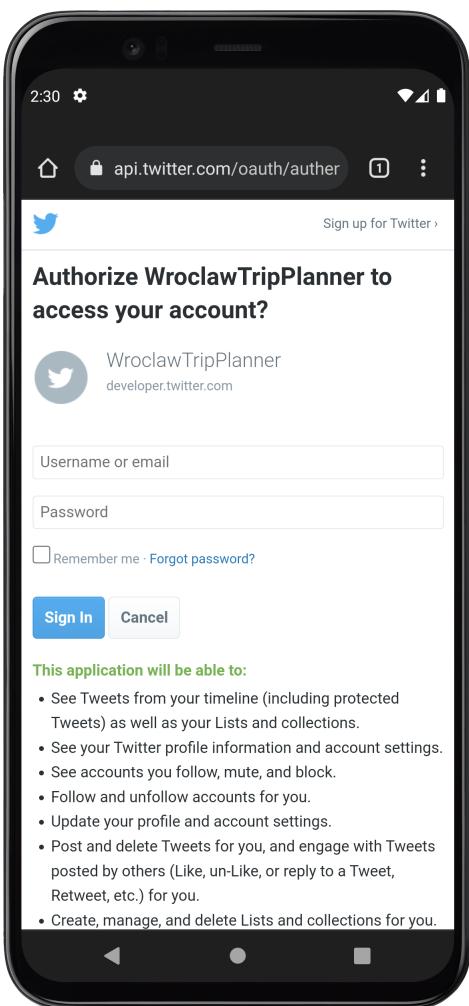


Rysunek 9: Ekran rejestracji użytkownika.

#### 4.1.3 Ekran logowania za pomocą konta Twitter

Użytkownik może uzyskać dostęp do aplikacji, logując się za pomocą konta Twitter. Reprezentujący tę opcję przycisk przekieruje użytkownika do przeglądarki, gdzie poproszony on zostanie o autoryzację dostępu aplikacji do konta Twitter (patrz rys. 10).

W obecnych czasach popularne jest oferowanie w aplikacjach mobilnych i internetowych opcji logowania za pomocą strony trzeciej, ponieważ wielu użytkowników może nie być chętnymi do tworzenia nowego konta w danym serwisie. Dodanie tego typu opcji niweluje ten problem, pozwalając użytkownikowi na korzystanie z konta istniejącego już w innym serwisie. Przy tego typu rejestracji pseudonimem użytkownika z serwisu Twitter zostanie ustawiony jako pseudonim w aplikacji. W wypadku, jeśli pseudonim ten jest już zajęty, użytkownik zostanie poproszony o wybór nowego, unikalnego pseudonimu.



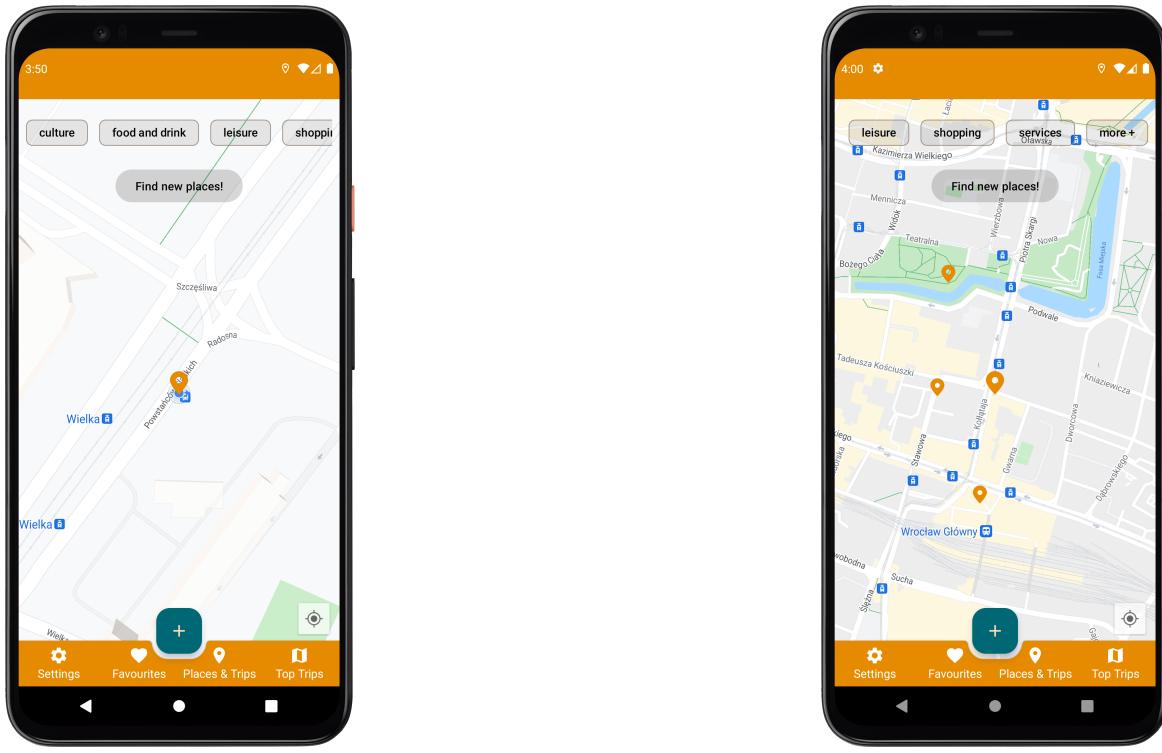
Rysunek 10: Okno autoryzacji za pomocą konta Twitter.

## 4.2 Ekrany główne

Główną część aplikacji stanowi aktywność główna, w skład której wchodzą fragmenty obsługujące interakcję z mapą oraz zamieszonymi na niej miejscami. Nawigacja pomiędzy jego fragmentami następuje za pomocą dolnego paska nawigacyjnego należącego do biblioteki Material Design.

### 4.2.1 Ekran mapy

Głównym ekranem, w którym użytkownik wykonuje najbardziej istotne akcje, jest ekran mapy. W nim widoczna jest, pobierana za pomocą Maps SDK for Android, mapa Google. Wygląd ekranu mapy zaprezentowany jest na rys. 11a. W prawym dolnym rogu mapy znajduje się przycisk centrujący ekran na pozycji użytkownika. Na górze zlokalizowane są przyciski wyszukiwania miejsc. Przycisk opisany „Find new places!” (pol. Odkryj nowe miejsca) przeszukuje bazę danych w poszukiwaniu wszystkich miejsc znajdujących się w promieniu trzystu metrów od środka ekranu. Przykład mapy z zaznaczonymi za pomocą przycisku „Find new places!” miejscami widoczny jest na rys. 11b. Na samej górze znajduje się grupa chipów opisanych nazwami kategorii. Przyciśnięcie chipa z napisem „shopping” (pol. zakupy) jest analogiczne do przycisku „Find new places!”, ale pobrane zostaną tylko miejsca o kategorii „shopping”.

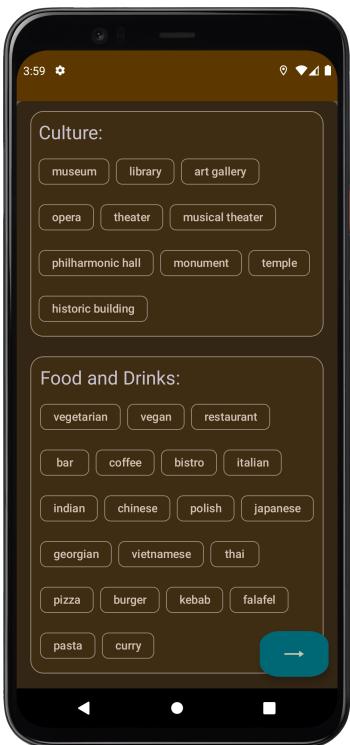


(a) Pozycja użytkownika widoczna na mapie.

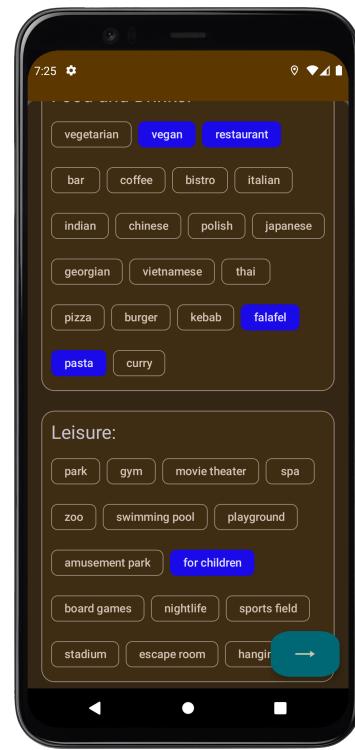
(b) Wyszukane miejsca oznaczone na mapie.

Rysunek 11: Ekran mapy

Ostatni z chipów, opisany „more +” wysuwa dolny arkusz (ang. bottom sheet) zawierający listę etykiet. Za pomocą tego ekranu użytkownik może wybrać do dziesięciu pożądanych przez siebie etykiet. Po wcisnięciu znajdującego się w prawym dolnym rogu ekranu, przycisku szukaj oznaczonego strzałką, na ekranie mapy wyświetlane zostaną miejsca w sposób analogiczny dla przycisku „Find new places！”, lecz pobrane zostaną tylko takie, które opisane są wybranymi przez użytkownika etykietami. Na rys. 12a zobaczyć można arkusz znaczników. Na rys. 12b widać przykład tego samego ekranu z zaznaczonymi przez użytkownika etykietami. Arkusz etykiet składa się, ze wspomnianych w rozdziale 3.1, chipów wyboru, które przejrzyście obrazują wybór etykiet.



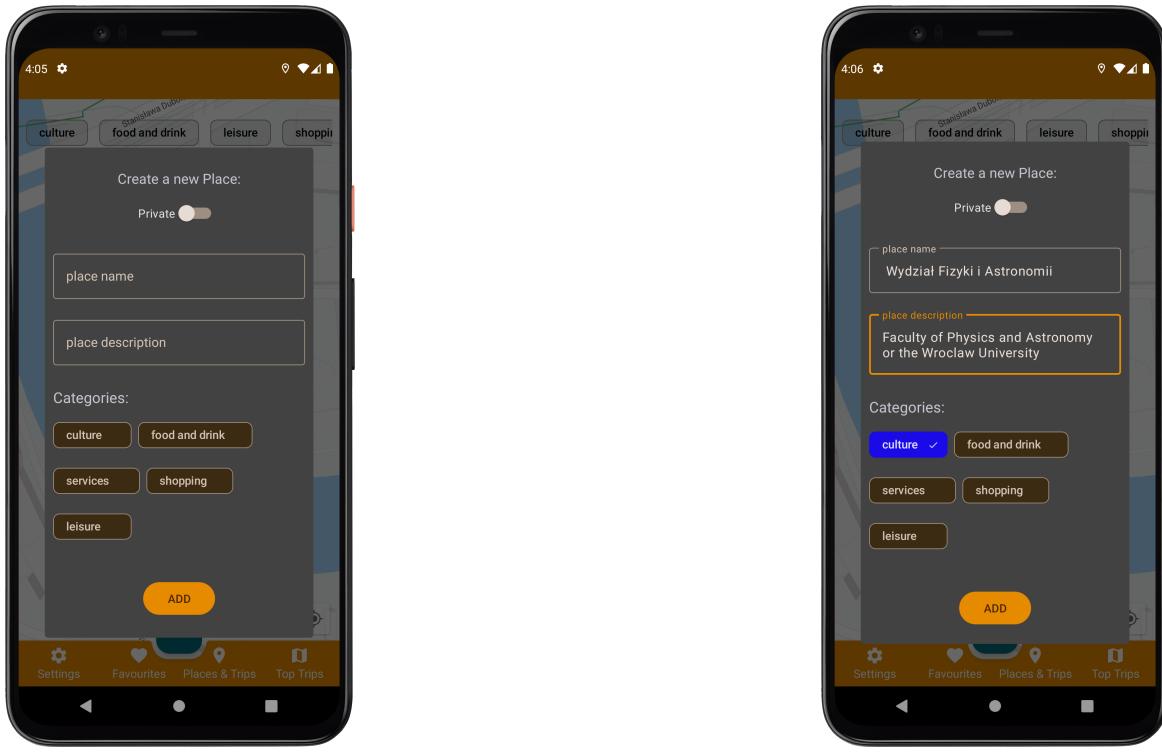
(a) Czysty arkusz etykiet.



(b) Arkusz etykiet z wybranymi użytkownika.

Rysunek 12: Dolny arkusz etykiet.

Na środku ekranu znajduje się duża pomarańczowa pinezka, której igła zaznacza środek ekranu. Służy ona za kursor w trakcie dodawania miejsca. Na środkowym dole ekranu znajduje się oznaczony plusem przycisk dodania miejsca. Rozpoczyna on proces dodawania miejsca, znajdującego się w miejscu wskazanym przez kursor. Po wcisnięciu na ekranie pojawia się okno dialogowe z polami, które uzupełnić musi użytkownik, aby dodać miejsce do bazy danych. Użytkownik musi sprecyzować jego nazwę, opis oraz wybrać co najmniej jedną z pięciu możliwych kategorii. Dialog tworzenia miejsca widoczny jest na rys. 13a. Przykład poprawnie wypełnionego ekranu tworzenia miejsca widoczny jest na rys. 13b

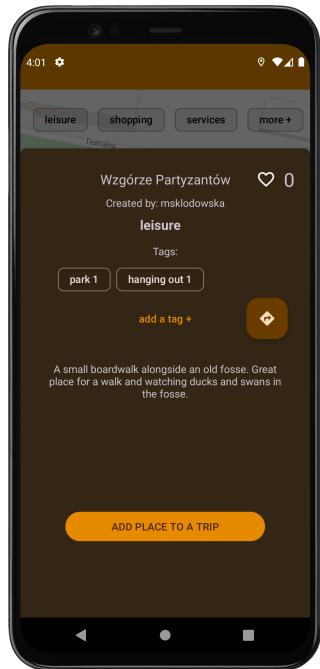


(a) Pusty dialog.

(b) Wypełniony dialog.

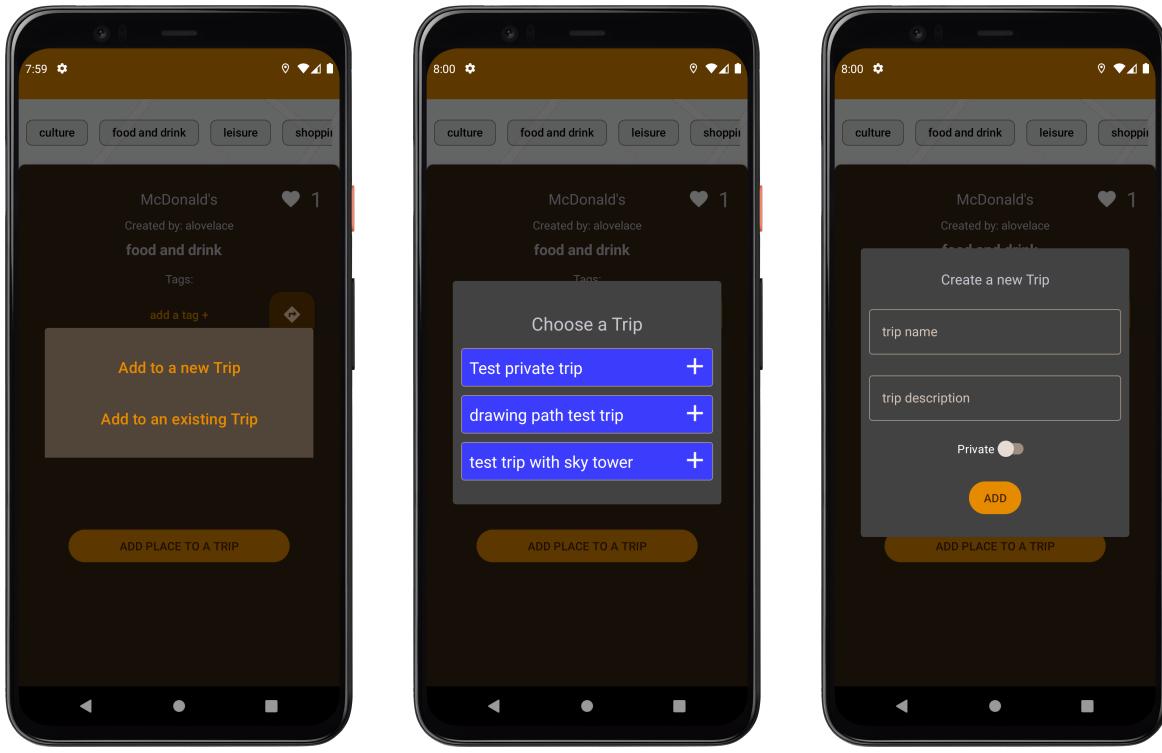
Rysunek 13: Dialog tworzenia miejsca.

Jak zauważyc można na rys. 11b, wyszukane miejsca pojawiają się na mapie w formie pinezek. Kliknięcie pinezki otwiera dolny arkusz przedstawiający detale miejsca. Na górze ekranu widoczne są nazwa miejsca, jego autor i wybrane przez niego kategorie. W prawym górnym rogu znajduje się licznik polubień i przycisk w kształcie serca, za pomocą którego użytkownik może dodać miejsce do swoich ulubionych. Poniżej kategorii widoczna jest lista etykiet z licznikami dodań. Pod nimi znajduje się przycisk dodania etykiet. Otwiera on arkusz widoczny wcześniej podczas wyszukiwania na rys. 12. Tym razem na miejscu przycisku wyszukania widoczny jest przycisk dodania oznaczony plusem sugerujący zmianę akcji na dodawanie etykiet. Na prawo od przycisku dodawania etykiet znajduje się oznaczony obrazkiem znaku drogowego przycisk rysowania drogi. Zamyka on arkusz miejsca i rysuje na mapie trasę rozpoczynającą się w aktualnej pozycji użytkownika, a kończącą w wybranym miejscu. Przesuwanie palcem w prawo lub w lewo zaskutkuje zmianą aktualnie wyświetlanego miejsca na inne znajdujące się w jego okolicy. Arkusz detali miejsca widoczny jest na rys. 14.



Rysunek 14: Widok szczegółów miejsca.

Na samym dole arkusza miejsca użytkownik ma możliwość dodania go do wycieczki. Wciśnięcie przycisku wywoła ekran dialogowy, widoczny na rysunku 15a dający użytkownikowi wybór pomiędzy dodaniem miejsca do istniejącej wycieczki lub utworzeniem nowej wycieczki zawierającej wybrane miejsce. Opcja istniejącej wycieczki przedstawi użytkownikowi listę wszystkich utworzonych przez niego wycieczek widoczną na rys. 15b. Wciskając element listy, użytkownik doda do niej wybrane miejsce i zakończy proces. Druga z opcji otworzy formularz tworzenia nowej wycieczki, w którym użytkownik może ustalić jej prywatność oraz nadać jej nazwę i opis.



(a) Wybór akcji dodawania wycieczki.

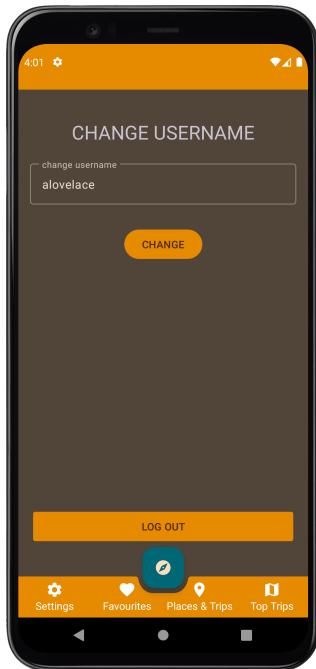
(b) Lista istniejących wycieczek.

(c) Dialog tworzenia wycieczki.

Rysunek 15: Proces dodania miejsca do wycieczki.

#### 4.2.2 Ekran ustawień

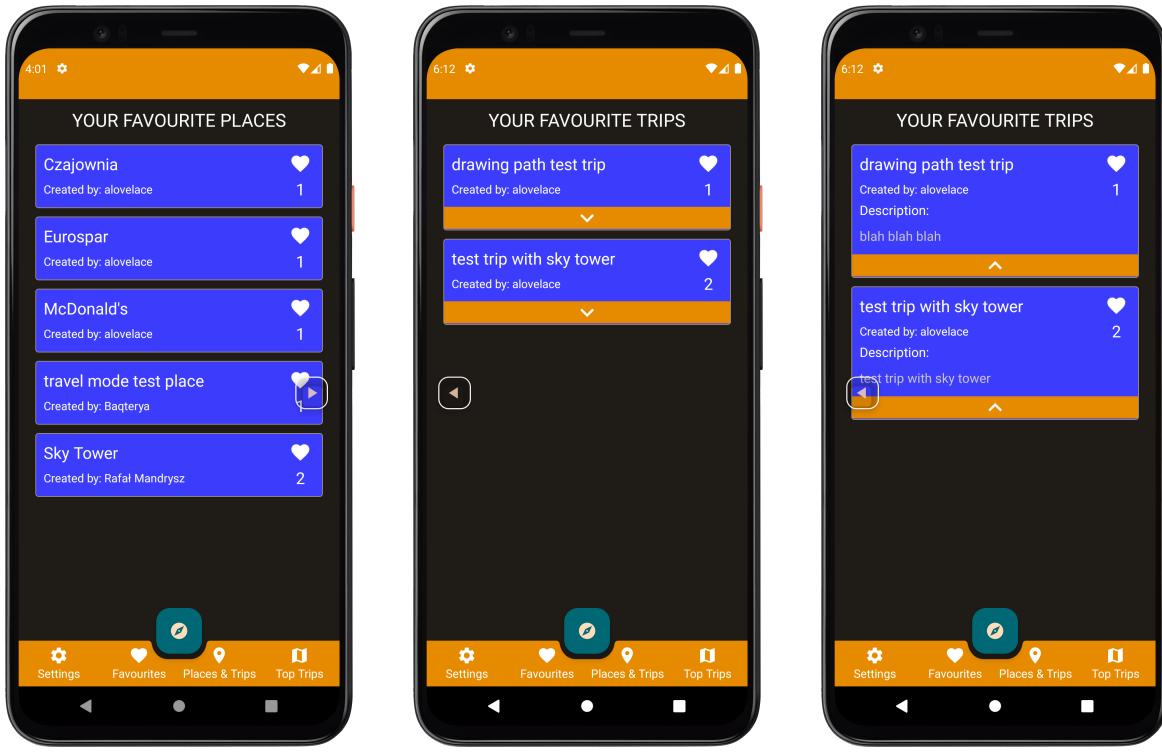
Pierwszy od lewej przycisk na dolnym pasku kieruje użytkownika do ekranu ustawień. Znajduje się w nim jedynie opcja zmiany pseudonimu oraz opcja wylogowania się. Dzięki temu, że zmiany w Cloud Firestore zachodzą w czasie rzeczywistym, zmiana pseudonimu skutkuje w natychmiastowej zmianie nazwy autora widocznej m.in. w arkuszu szczegółów miejsca. Przycisk wylogowania przenosi użytkownika do ekranu powitalnego. Ekran ustawień widoczny jest na rys. 16. Przy zmianie ekranu na inny niż ekran mapy zauważać można, że przycisk dodania miejsca został zastąpiony przez przycisk przedstawiający kompas. Wciśnięcie go powoduje powrót do ekranu mapy.



Rysunek 16: Widok ustawień.

#### 4.2.3 Listy ulubionych miejsc i wycieczek

Następny na pasku dolnym jest oznaczony kształtem serca przycisk zatytułowany „Favourites” (pol. Ulubione). Przekierowuje on użytkownika do ekranu, na którym wylistowane są polubione przez użytkownika miejsca oraz wycieczki. Znajdują się one na osobnych listach, które przełączać można, przesuwając palcem w prawo lub w lewo, zgodnie z wyświetlonymi na ekranie strzałkami. Na rys. 17a zobaczyć można listę ulubionych miejsc użytkownika. Wciśnięcie elementu listy przeniesie użytkownika na ekran mapy i wyświetli na niej pinezkę tego miejsca. Wciśnięcie przycisku polubienia usunie miejsce z ulubionych i w czasie rzeczywistym zniknie z listy. Przesunięcie ekranu w lewo wyświetli listę ulubionych wycieczek widoczną na rys. 17b. Każdy element listy posiada oznaczony strzałką przycisk rozwinienia pozwalający zobaczyć podgląd opisu danej wycieczki, co widoczne jest na rys. 17c.



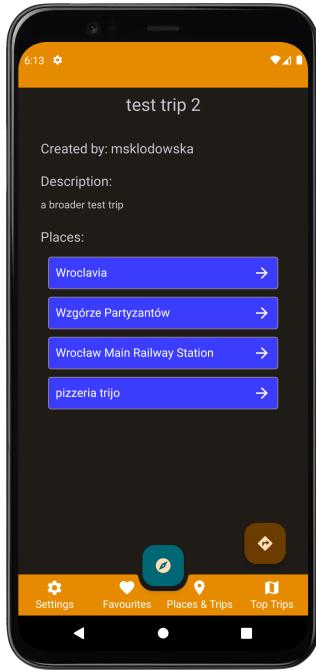
(a) Lista ulubionych miejsc użytkownika.

(b) Lista ulubionych wycieczek.

(c) Rozwinięty opis wycieczek.

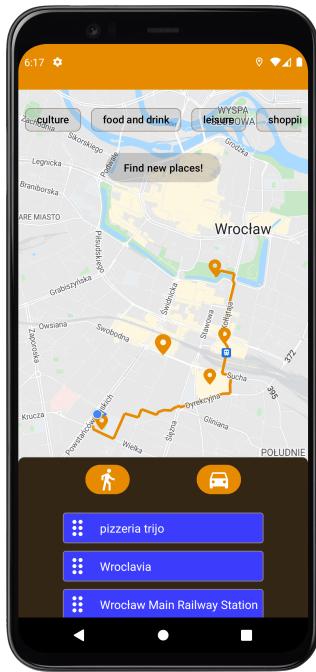
Rysunek 17: Ekran ulubionych miejsc i wycieczek.

Wciśnięcie elementu listy wycieczek przeniesie użytkownika do widocznego na rys 18 ekranu szczegółów wycieczki. Na samej górze ekranu widoczna jest nazwa wycieczki. Pod nią mieścią się pseudonim autora, opis oraz lista miejsc zawierających się w wycieczce. Wciśnięcie elementu listy miejsc przekieruje użytkownika do ekranu mapy i wyświetli na niej pinezkę przedstawiającą wybrane miejsce.



Rysunek 18: Ekran szczegółów wycieczki.

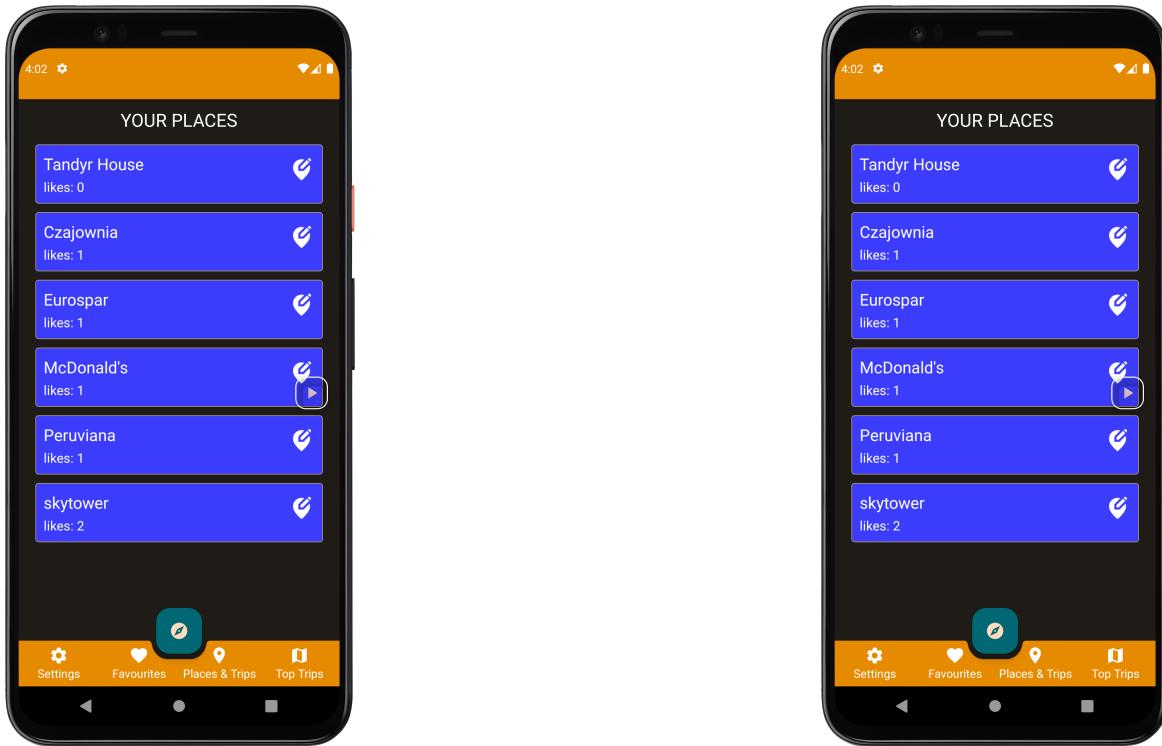
W prawym dole ekranu szczegółów wycieczki widoczny jest oznaczony znakiem drogowym przycisk rysowania trasy wycieczki. Przekierowuje on użytkownika na ekran mapy, gdzie zaczyna się proces rysowania trasy. Rozpoczyna się ona w aktualnej pozycji użytkownika, a kończy w ostatnim miejscu na liście. Po przekierowaniu na ekranie widoczny jest dolny arkusz pozwalający użytkownikowi na wybór pomiędzy pieszą a samochodową trasą oraz listą miejsc. Pozycje na liście oznaczone są ikoną mającą sugerować użytkownikowi, że może zmieniać kolejność elementów na liście, wybierając w ten sposób kolejność odwiedzania miejsc wycieczki. Widok ten przedstawiony jest na rys. 19.



Rysunek 19: Proces planowania trasy wycieczki.

#### 4.2.4 Listy miejsc i wycieczek użytkownika

Trzecia z opcji widocznych na pasku dolnym opisana jest „Places and Trips” i przedstawia ona listy miejsc i wycieczek stworzonych przez użytkownika. Ekran ten widoczny jest na rys. 20. Poruszanie się pomiędzy listą miejsc i wycieczek działa w ten sam sposób, co w opisanym wcześniej ekranie „Favourites”. Widoczna na rys. 20a lista przedstawia wszystkie dodane przez użytkownika miejsca wraz z liczbą ich polubień przez wszystkich użytkowników. Podobnie skonstruowana jest widoczna na rys. 20b lista wycieczek stworzonych przez użytkownika.

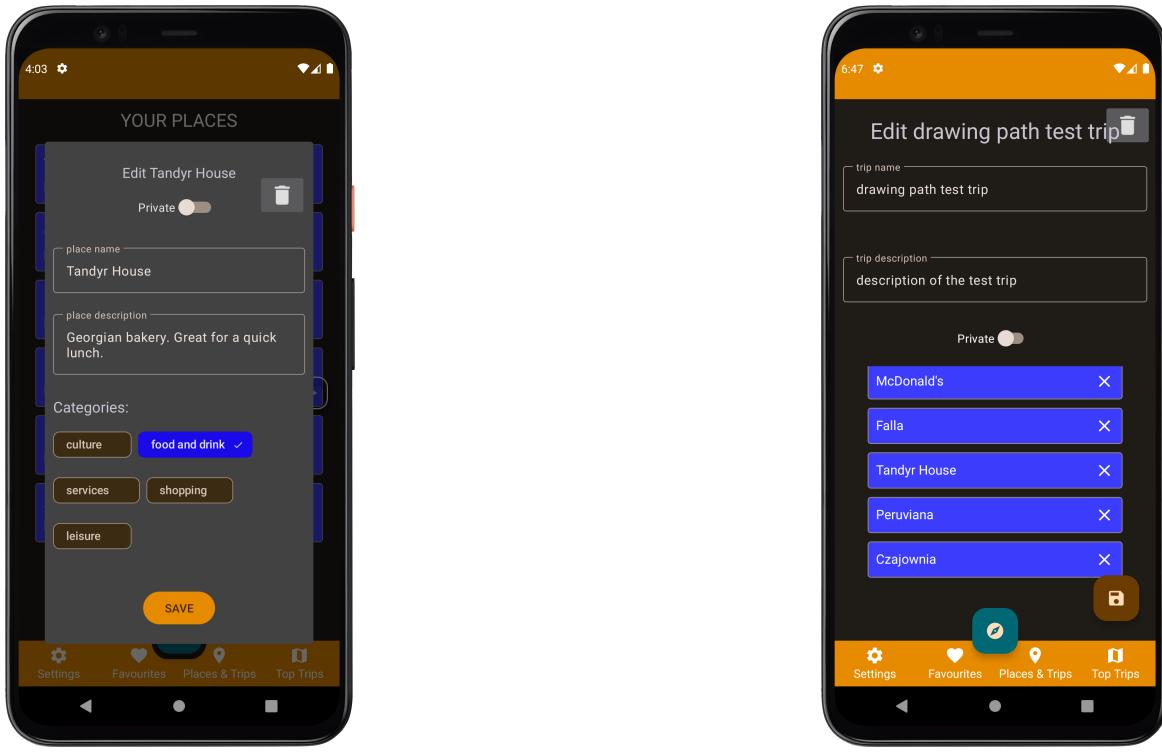


(a) Lista miejsc użytkownika.

(b) Lista wycieczek użytkownika.

Rysunek 20: Ekran miejsc i wycieczek użytkownika.

Wciśnięcie elementu listy miejsc stworzonych przez użytkownika przekierowuje użytkownika do ekranu mapy, na której wyświetlane zostaje wybrane miejsce. Wciśnięcie ikony edycji miejsca skutkuje wyświetlением dialogu edycji miejsca widocznego na rys. 21a. Jest on analogiczny do przedstawionego wcześniej ekranu dialogowego służącego do utworzenia miejsca, z tą różnicą, że jest on uzupełniony aktualnymi danymi miejsca oraz obok nazwy miejsca widoczny jest przedstawiający ikonę kosza przycisk usunięcia miejsca. Powoduje on, poprzedzone dialogiem ostrzegawczym, bezpowrotnie usunięcie miejsca z bazy danych. Znajdujący się na dole ekranu dialogowego przycisk „Save” (pol. Zapisz) zapisuje zmiany wprowadzone przez użytkownika. Wciśnięcie elementu listy wycieczek użytkownika przenosi użytkownika do ekranu edycji wycieczki. Widoczne są w nim pola do edycji nazwy, opisu i prywatności wycieczki, przycisk usunięcia, działający jednakowo do przycisku usunięcia miejsca, oraz lista zawartych w wycieczce miejsc. Elementy listy miejsc wycieczki opatrzone są ikoną „X”, której wcisnięcie skutkuje usunięciem miejsca z wycieczki. Ekran edycji wycieczki przedstawiony jest na rys. 21b.



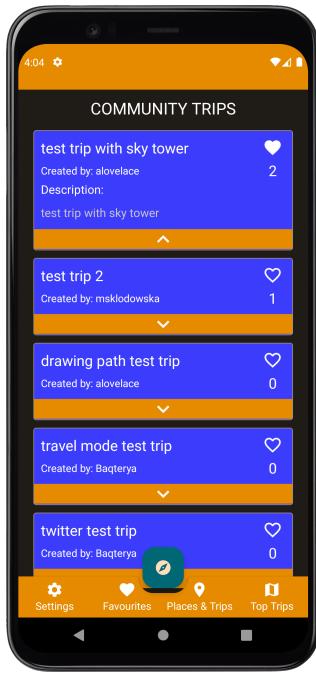
(a) Dialog edycji miejsca.

(b) Ekran edycji wycieczki.

Rysunek 21: Edycja miejsc i wycieczek użytkownika.

#### 4.2.5 Lista najpopularniejszych wycieczek

Ostatni z dostępnych w aplikacji ekranów to ekran przedstawiający listę stu nieprywatnych wycieczek stworzonych przez użytkowników aplikacji o największej ilości polubień. Są one posortowane względem polubień. Jest to sposób na to, aby użytkownicy mogli odkrywać stworzone przez innych wycieczki i poznawać popularne trasy. Elementy listy przedstawiają nazwę, twórcę oraz ilość polubień wycieczek, oraz przycisk pozwalający rozwinąć ich opis. Wciśnięcie przycisku polubienia doda miejsce do ulubionych miejsc użytkownika. Wciśnięcie elementu listy przeniesie użytkownika do widocznego na rys. 15 ekranu szczegółów miejsca. Ekran listy najpopularniejszych wycieczek widoczny jest na rys. 22.



Rysunek 22: Ekran najpopularniejszych wycieczek.

## 5 Podsumowanie

Zamysłem niniejszej pracy było stworzenie narzędzia pozwalającego użytkownikom na zapisywaniu ciekawych miejsc we Wrocławiu i na dzieleniu się nimi z innymi użytkownikami. Głównymi celami w tworzeniu aplikacji były autoryzacja użytkowników, możliwość zapisywania miejsc oraz wycieczek do bazy danych Cloud Firestore, oraz zapewnienie użytkownikowi usług nawigacyjnych: wskazywania jego pozycji, oraz rysowania ścieżek prowadzących do miejsc lub obrazujących przebieg wycieczki. Cele te zostały zrealizowane.

Aplikacja może stanowić podstawę do rozwoju rozbudowanej aplikacji nawigacyjno-społecznościowej. W związku z wykorzystaniem usług Firebase jest baza danych otwarta, jest na rozwój i skalowanie. Dzięki wydajności Cloud Firestore nie będzie przeszkodą wzrost liczby użytkowników ani wprowadzanych przez nich do bazy danych Cloud Firestore danych. Ponadto, aplikacja próbuje poruszyć istotny temat w projektowaniu aplikacji społecznościowych, jakim jest aktywizacja użytkowników. System znaczników zaimplementowany w aplikacji stanowi elastyczną podstawę na bazie, której zbudować można by zaawansowany system komunikacji użytkowników. Wymagają one od nich zdecydowanie mniej inicjatywy niż pisanie tradycyjnych komentarzy, dzięki czemu można liczyć na większy udział użytkowników w używaniu tej funkcji.

Proces tworzenia aplikacji nie odbył się bez problemów. Zastosowanie Material Design pozwala na uzyskanie przejrzystego, estetycznego i responsywnego interfejsu, lecz w paru miejscach ustawienie przycisków nie jest dostatecznie dobrze przemyślane. Niektóre funkcjonalności interfejsu, takie jak przesuwanie na boki miejsc w arkuszu szczegółów miejsc mogą być nieintuicyjne. W wypadku przyszłego rozwoju aplikacji nacisk powinien zostać położony na rozbudowanie interfejsu rysowania trasy wycieczki, który, chociaż funkcjonalny, odstaje w kwestii wygody obsługi. Rozbudowie powinien ulec także ekran ustawień, który w obecnym stanie jest dość ubogi. W przyszłym rozwoju aplikacji powinny zagościć w nim np. takie opcje jak zmiana języka aplikacji, czy zaimplementowanie zmiany motywu pomiędzy jasnym a ciemnym.

Mimo wszystko tworzenie aplikacji odbyło się bez większych problemów technicznych. Można zawdzięczać to wygodzie w używaniu chmury obliczeniowej Firebase i oferowanych przez Google bibliotekom i narzędziom. Istotny jest też fakt tego, że wykorzystane w pracy narzędzia są narzędziami współczesnymi i wciąż rozwijanymi. Dzięki temu dostępna jest obszerna i przystępna dokumentacja. Proces budowy aplikacji dowodzi, że Firebase jest potężnym i przystępnym w obsłudze narzędziem, dzięki któremu nawet jednoosobowy zespół jest w stanie w krótkim czasie wdrożyć aplikację wykorzystującą rozbudowaną bazę danych i szereg bezpiecznych i wydajnych rozwiązań sieciowych takich jak autoryzacja użytkowników.

## Bibliografia

- [1] Alex Amies i in. *Developing and Hosting Applications on the Cloud*. IBM Press, 2012.
- [2] Valve Corporation. *Browse Steam Your Way, Introducing Steam Tags, A Powerful New Way to Shop For Games*. URL: <https://store.steampowered.com/tag/> (term. wiz. 11.04.2022).
- [3] Garcia-Molina i in. *Database Systems: The Complete Book*. Sty. 2002.
- [4] Sławek Iłski. *Wrocław — Rynek — Panorama z kościoła p.w. św. Elżbiety*. (term. wiz. 16.04.2022). 27 lip. 2012. URL: <https://web.archive.org/web/20161024173922/http://www.panoramio.com/photo/76098529>.
- [5] Google Inc. *Android's Kotlin-first approach*. URL: <https://developer.android.com/kotlin/first> (term. wiz. 07.04.2022).
- [6] Google Inc. *Chips, guidelines*. URL: <https://m3.material.io/components/chips/guidelines> (term. wiz. 13.04.2022).
- [7] Google Inc. *Chips, overview*. URL: <https://m3.material.io/components/chips/overview> (term. wiz. 13.04.2022).
- [8] Google Inc. *Dokumentacja Cloud Firestore*. URL: <https://firebase.google.com/docs/firestore> (term. wiz. 07.04.2022).
- [9] Google Inc. *Dokumentacja klasy View*. URL: <https://developer.android.com/reference/android/view/View> (term. wiz. 13.04.2022).
- [10] Google Inc. *Elementy składowe obiektu DirectionsResult*. URL: <https://developers.google.com/maps/documentation/javascript/directions#Routes> (term. wiz. 15.04.2022).
- [11] Google Inc. *Geo queries*. URL: <https://firebase.google.com/docs/firestore/solutions/geoqueries> (term. wiz. 11.04.2022).
- [12] Google Inc. *Google Firebase Authentication*. URL: <https://firebase.google.com/products/auth> (term. wiz. 07.04.2022).
- [13] Google Inc. *Google Firebase Build Products*. URL: <https://firebase.google.com/products-build> (term. wiz. 07.04.2022).
- [14] Google Inc. *Google Firebase Cloud Firestore*. URL: <https://firebase.google.com/products/firestore> (term. wiz. 07.04.2022).
- [15] Google Inc. *Indeksowanie w Cloud Firestore*. URL: <https://firebase.google.com/docs/firestore/query-data/indexing> (term. wiz. 07.04.2022).
- [16] Google Inc. *Material Theme Builder*. URL: <https://material-foundation.github.io/material-theme-builder/#/custom> (term. wiz. 13.04.2022).
- [17] Google Inc. *Meet Android Studio*. URL: <https://developer.android.com/studio/intro> (term. wiz. 07.04.2022).
- [18] Google Inc. *Produkty Google Cloud*. URL: <https://cloud.google.com/products> (term. wiz. 07.04.2022).
- [19] Google Inc. *Produkty Google Maps Platform*. URL: <https://mapsplatform.google.com/maps-products/> (term. wiz. 07.04.2022).

- [20] Google Inc. *Przegląd Google Cloud*. URL: <https://cloud.google.com/docs/overview> (term. wiz. 07.04.2022).
- [21] Google Inc. *Skalowanie w Cloud Firestore*. URL: <https://cloud.google.com/architecture/building-scalable-apps-with-cloud-firebase> (term. wiz. 07.04.2022).
- [22] Google Inc. *Strona główna Android Jetpack*. URL: <https://developer.android.com/jetpack> (term. wiz. 13.04.2022).
- [23] Google Inc. *Strona główna Google Firebase*. URL: <https://firebase.google.com/> (term. wiz. 07.04.2022).
- [24] Google Inc. *Strona główna Google Maps Platform*. URL: <https://mapsplatform.google.com/> (term. wiz. 07.04.2022).
- [25] Google Inc. *The Activity Lifecycle*. URL: <https://developer.android.com/guide/components/activities/activity-lifecycle> (term. wiz. 13.04.2022).
- [26] Google Inc. *Volley*. URL: <https://google.github.io/volley/> (term. wiz. 15.04.2022).
- [27] JetBrains s.r.o. *Strona główna języka Kotlin*. URL: <https://kotlinlang.org/> (term. wiz. 07.04.2022).
- [28] Statista. *Mobile operating systems' market share worldwide from January 2012 to January 2022*. URL: <https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/> (term. wiz. 07.04.2022).
- [29] *Strona główna Open Handset Alliance*. URL: <https://www.openhandsetalliance.com/index.html> (term. wiz. 07.04.2022).