UNIWERSYTET GDAŃSKI
WYDZIAŁ MATEMATYKI, FIZYKI I INFORMATYKI

Przemysław Sankowski

numer albumu: 253938

Kierunek: Informatyka

Specjalność: Ogólna

Hike Mate - Aplikacja informująca o trójmiejskich szlakach pieszych

Praca licencjacka

wykonana pod kierunkiem

dr. Paweł Pączkowski

Gdańsk, rok 2021

Streszczenie pracy

W pierwszym rozdziale pracy opisano główne zastosowanie aplikacji, tj. rozpoznawanie szlaku pieszego na podstawie jego koloru i aktualnej lokalizacji użytkownika. Jest w nim też zawarty skrócony, bardzo uproszczony opis działania oraz wymagania sprzętowe, jakie urządzenie musi spełniać, aby uruchomić aplikację Hike Mate.

Drugi rozdział poświęcony jest przedstawieniu projektu, czyli diagramy klas i przypadków użycia, wymagania funkcjonalne aplikacji oraz zrzuty ekranu przedstawiające interfejs użytkownika.

Trzeci rozdział przedstawia implementację projektu od strony zarówno technicznej, jak i szczegółowe omówienie tego, co użytkownik może wykonać w każdej aktywności aplikacji. Podrozdział opisujący architekturę mówi o plikach, z których aplikacja korzysta oraz o folderach, składających się na strukturę programu. Dalej jest szczegółowe omówienie działania aplikacji dla każdej aktywności systemu Android oddzielnie, w kolejności najbardziej naturalnej dla użytkownika. W rozdziale tym jest także zestawienie użytych technologii oraz wypisanie wszystkich bibliotek z zastrzeżeniami, do czego były one używane. Jest też opis procesu przygotowania danych, na których operuje aplikacja.

Czwarty rozdział, ostatni, opisuje testy przeprowadzone w celu weryfikacji stabilności i poprawnego funkcjonowania aplikacji.

Spis treści

1 Ogólny opis aplikacji

1.1 Wprowadzenie

Podmiotem mojej pracy jest Hike Mate - aplikacja na systemy Android w wersjach od 5.0 Lollipop wzwyż.

Głównym zadaniem aplikacji jest podanie użytkownikowi informacji na temat pieszego szlaku turystycznego w obrębie Trójmiasta, na którym aktualnie się znajduje. Operacja ta jest wykonana na podstawie danych lokalizacyjnych pobranych z wbudowanego w większość urządzeń z systemem Android modułu GPS - w związku z czym, aplikacja do działania wymaga urządzenia obsługującego powyższy moduł. Dodatkowo, jeśli w urządzenie wbudowana jest kamera, aplikacja może umożliwić odczyt koloru bezpośrednio z oznaczenia - jest to jedynie opcja dodatkowa, dzięki czemu Hike Mate może działać także na urządzeniach nie posiadających możliwości robienia zdjęć. Opcja ta umożliwia wykorzystywanie aplikacji także przez ludzi cierpiących na zaburzenia widzenia barw.

1.2 Przykład zastosowania

Zwiedzając Gdańsk, Gdynie, Sopot oraz ich okolice można wielkrotnie natrafić na kolorowe oznaczenia szlaków namalowane na drzewach, budynkach czy kamieniach. Ze względu na szybkość życia w mieście zazwyczaj nie zastanawiamy się nawet, gdzie może prowadzić szlak, na którym się aktualnie, często zupełnie przypadkiem, znajdujemy. Aplikacja Hike Mate skierowana jest do ludzi żyjących miejskim trybem życia, którzy w ramach oderwania od szybkości codziennego życia lubią wędrować po ciekawych lokalizacjach. Osoba taka, nazwijmy ją Ania, mogłaby, czekając na autobus w sopockiej dzielnicy Przylesie zauważyć niebieskie oznaczenie szlaku namalowane na słupie oświetleniowym i zacząć zastanawiać się, co to za szlak. Z racji faktu, że Ania w wolnym czasie uwielbia spacerować po lesie, zainteresowała się, gdzie taki szlak może prowadzić. Jednak po wpisaniu paru haseł do wyszukiwarki internetowej, okazało się, że zlokalizo-

wanie takiego szlaku na podstawie oznaczenia obok przystanku autobusowego jest dość pracochłonne, gdyż wymaga prześledzenia przebiegów wszystkich szlaków Trójmiasta oznaczonych takim kolorem. Ania mogłaby skazać sie na niewygodę przeszukiwania internetu w telefonie jadąc do pracy lub tracić na to swój czas wolny w domu, mozolnie przeglądając wszystkie możliwe trasy. Z aplikacją Hike Mate nie ma potrzeby poświęcania czasu na zbędne poszukiwania, wystarczy zrobić zdjęcie znaku lub wybrać jego kolor z listy znajdując się w pozycji, w której owe oznaczenie szlaku się znalazło, resztę natomiast wykona już sama aplikacja, która wyświetli trasę, jaką szlak ten przebiega, jego nazwę, długość oraz profil wysokościowy. Dzięki tym informacjom Ania będzie wiedziała, czego dokładnie szukać, żeby dowiedzieć się co może spotkać na trasie - może oczywiście także pójść w tę trasę w ciemno i poznać wszystkie jej tajemnice na własnej skórze - aplikacja Hike Mate przecież ją poprowadzi, a na podstawie długości, aktualnej pozycji użytkownika i profilu wysokościowego Ania będzie wiedziała jak się na tę trasę dokładnie przygotować.

1.3 Aktualnie dostępne zamienniki

W tym momencie nie ma aplikacji na urządzenia mobilne (ani też na stacjonarne) pozwalającej na odnalezienie szlaku turystycznego, na którym osoba się znajduje. Jedyny możliwy zamiennik to manualne przeszukiwanie internetu lub przewodników PTTK. Istnieje natomiast wiele aplikacji pozwalających samemu zaplanować sobie trasę, w celu późniejszej wędrówki, jednak nie taka idea przyświeca powstaniu Hike Mate - ta aplikacja ma zadanie prowadzić ludzi po oficjalnych, oznaczonych, szlakach PTTK.

1.4 Sposób działania

Przy pierwszym uruchomieniu aplikacja zarząda pozwolenia na dostęp do danych lokalizacyjnych usługi GPS urządzenia z systemem Android. Po odmowie aplikacja po prostu się wyłączy, po zezwoleniu natomiast wyświetli się aktywność, w której użytkownik musi wybrać w jaki sposób chce zidentyfikować kolor szlaku, na którym się

znajduje - może to zrobić za pomocą wyboru jednego z pięciu kolorów z listy lub poprzez zrobienie zdjęcia, przy czym druga opcja zadziała tylko, jesli urządzenie ma kamerę - w przeciwnym wypadku użytkownik po wybraniu tej opcji zostanie poinformowany o braku takiej możliwości. Jeśli użytkownik zdecyduje się na zrobienie zdjęcia, będzie musiał zezwolić aplikacji na dostęp do kamery (jedynie przy pierwszym uruchomieniu) - w przeciwnym wypadku zostanie cofnięty do poprzedniego ekranu. Kiedy kamera zostanie już uruchomiona użytkownik poproszony będzie o zrobienie zdjęcia zawierającego oznakowanie szlaku wewnątrz oznaczonej na czerwono ramki, przy czym zewnętrzna strona ramki będzie zacieniona.

Po zrobieniu zdjęcia oznaczenia szlaku lub wybraniu koloru z listy użytkownik jest przenoszony na ekran zawierający podstawowe informacje na temat szlaku, tj. jego nazwę, kolor, długość i różnicę w wysokości pomiędzy najwyższym i najniższym punktem trasy. Na tym ekranie znajdują się trzy przyciski: jeden cofa użytkownika do początkowej aktywności aplikacji, drugi wyświetla mapę z serwisu Google Maps z naniesioną na nią czerwoną linią przedstawiającą przebieg szlaku i aktualną pozycję użytkownika, trzeci natomiast wyświetla wykres z profilem wysokościowym trasy. W przypadku, w którym użytkownik wybiera z listy kolor szlaku, nie znajdując się na nim (lub w promieniu 100m od niego) aplikacja Hike Mate wyświetla ekran z informacją o braku takiego szlaku w swojej bazie, przyciskiem pozwalającym na powrót do ekranu startowego aplikacji oraz z przyciskiem służącym do wyświetlenia mapy z zaznaczoną aktualną pozycją użytkownika. To samo dzieje się także jeśli użytkownik robi zdjęcie oznaczenia szlaku nie zaimplementowanego w aplikacji.

1.5 Wymagania sprzętowe

Ze względu na wykorzystanie modułu Google Play Services Maps dostosowanego do wersji Android API 21 lub wyższej, do działania aplikacji wymagane jest wykorzystanie urządzenia o odpowiadającej wersji, tj. Android 5.0 Lollipop, wydanego po 4. listopada 2012 roku [1]. Docelowe urządzenie musi także posiadać moduł GPS zdolny do wykry-

wania aktualnej lokalizacji użytkownia. Aplikacja może również wykorzystywać aparat, jednak nie jest to konieczne do zapewnienia pełnej funkcjonalności.

1.6 Ogólne zasady działania głównych funkcji aplikacji

- **Przygotowanie pliku mapy**

  Plik z mapą jest "wyklikany" w serwisie Google Maps w celu zdobycia dokładnego przebiegu trasy, następnie przekonwertowany na plik *.plt za pomocą procesu opisanego w 3.x. Samo przygotowanie pliku w formacie czytelnym dla aplikacji jest wykonywane przez pomocniczy program napisany w języku Python3.8 z wykorzystaniem biblioteki Pandas.

- **Rozpoznawanie koloru**

  Przy wyborze koloru z listy zmienna zawierająca kolor szlaku jest podawana do dalszej aktywności, natomiast w przypadku wykonania zdjęcia kolor jest poznawany przez określenie dominującego koloru na poziomym pasku o wymiarach 500 x 150 px (wymiar ramki wewnątrz której ma się znajdować znak to 800 x 600 px). Określenie dominującego koloru jest wykonane za pomocą biblioteki firmy Google, Android.Palette, odczytane w formie ARGB (heksadecymalnej), przekonwertowane na format HSV, z którego funkcja w prosty sposób może określić z jakim kolorem mamy do czynienia.

- **Rozpoznawanie szlaku**

  Aplikacja zawiera listy współrzędnych geograficznych względem których przebiega dany szlak (ok. 1000 punktów dla każdej ścieżki) w postaci plików *.plt znajdujących się w folderze assets wbudowanym w strukturę aplikacji. Po wybraniu koloru aplikacja tworzy listę plików dotyczących danego koloru po czym przeszukuje po kolei wszystkie pliki na danej liście. Jeśli punkt z którejś listy jest w promieniu 100m od aktualnej pozycji użytkownika zwracana jest nazwa pliku zawierającego dany szlak.

- **Rysowanie ścieżki**

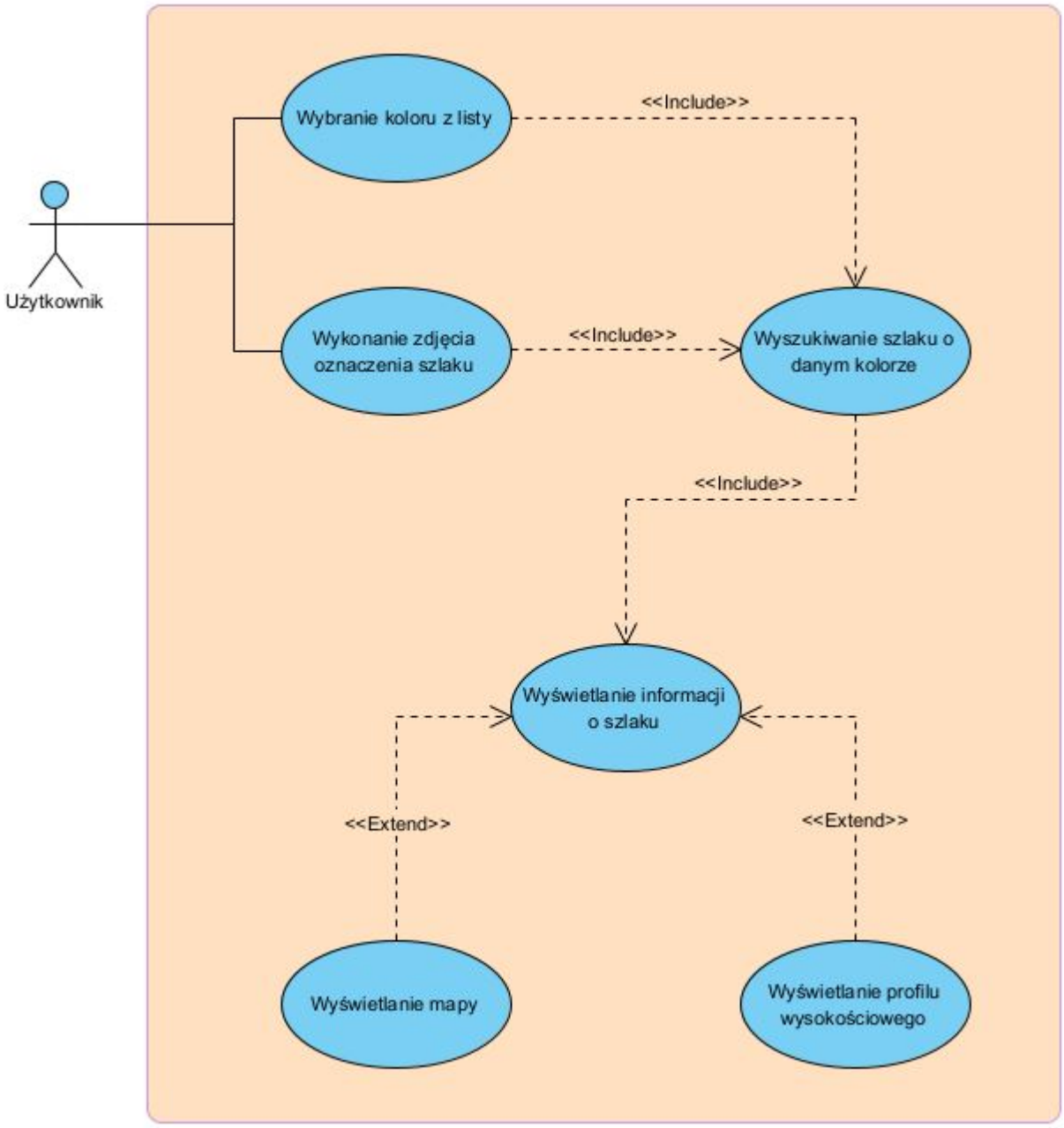
  Aplikacja dostaje od funkcji dotyczącej rozpoznawania szlaku informację o nazwie pliku zawierającego wymaganą ścieżkę, następnie na mapę Google Maps jest nakładana linia będąca połączeniem wszystkich punktów z listy współrzędnych geograficznych prostymi - lista powstaje na podstawie pliku już w momencie rozpoznawania szlaku i jest przechowywana w osobnym obiekcie.

- **Rysowanie profilu wysokościowego**

  Profil wysokościowy jest rysowany na podstawie tego samego pliku *.plt, który zawiera także wartości wysokości każdego punktu nad poziomem morza podane w metrach. Wykres liniowy przedstawiający te wartości rysowany jest przez samą aplikację w momencie wciśnięcia odpowiadającego temu zadaniu przycisku i nie jest nigdzie zapisywany.

2 Projekt i analiza

2.1 Diagram przypadków użycia



2.2 Wymagania funkcjonalne

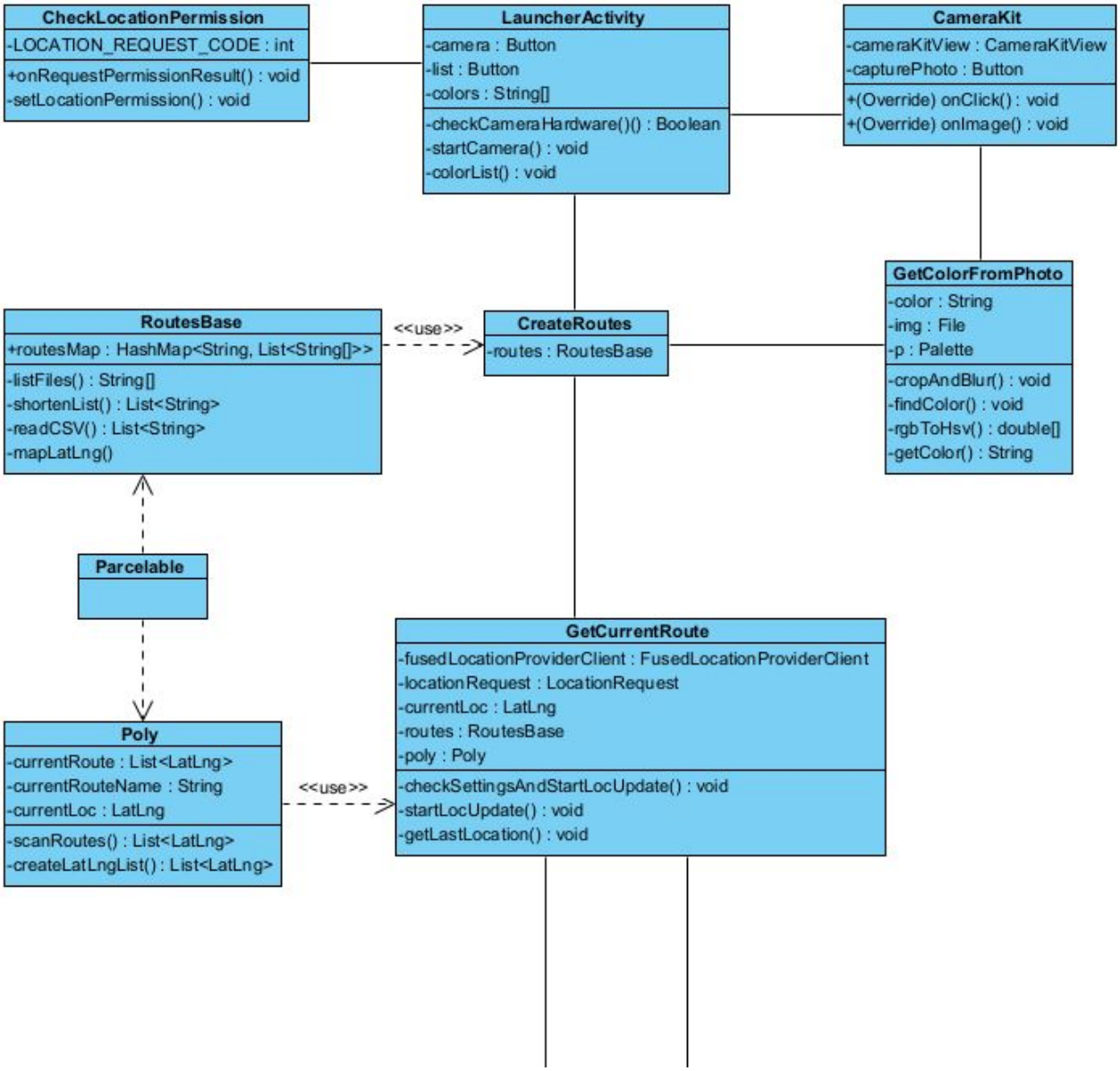
Aplikacja ma za zadanie pomagać określać szlak PTTK, na którym użytkownik w danym momencie się znajduje w najprostszy możliwy sposób. W związku z powyższym lista wymagań funkcjonalnych skupia się na prostocie i szybkości jej użytkowania.

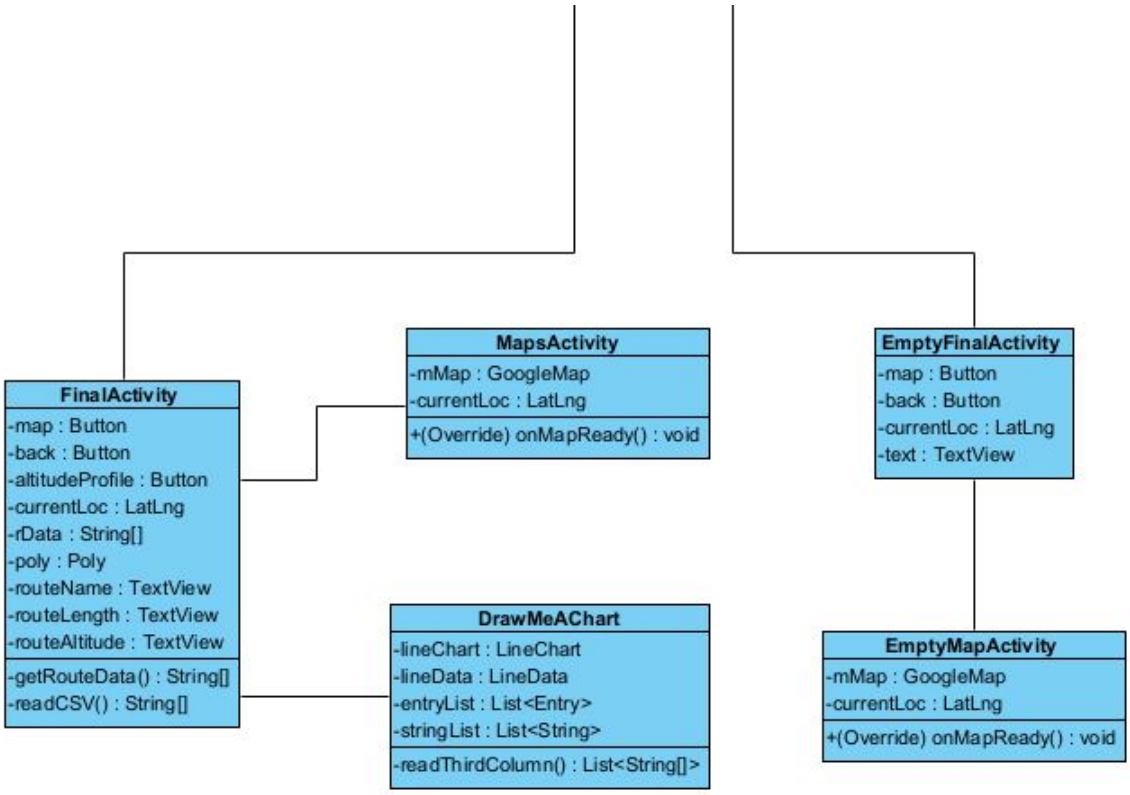
Lista wymagań funkcjonalnych:

1. Prostota użytkowania aplikacji, tak, aby każdy mógł ją po prostu zainstalować i z niej korzystać, bez konieczności czytania żadnych instrukcji pomocniczych.
2. Czytelność uzyskanych informacji - możliwość wykonywania zbliżeń na mapie oraz profilu wysokościowym.
3. Szybkość działania aplikacji.
4. Możliwość korzystania z aplikacji w trybie offline, bez konieczności łączności z internetem - w lasach, którymi przebiega wiele szlaków, zasięg sieci internetowych może być ograniczony.
5. Niezawodność aplikacji - aplikacja musi być stabilna i mieć możliwość obsłużenia użytkownika nawet w przypadku kiepskiej czytelności oznaczenia, spowodowanej starą farbą, przez którą aplikacja może mieć problem ze zidentyfikowaniem dominującego koloru (wybór koloru z listy).
6. Estetyczny wygląd na wielu urządzeniach z systemem Android - automatyczna skalowalność interfejsu.

2.3 Diagram klas

Wszystkie klasy inne niż RoutesBase oraz Poly są aktywnościami (ang. Activities), w związku z czym implementują klasę AppCompatActivity - nie jest to zaznaczone na diagramie w celu zwiększenia jego czytelności.



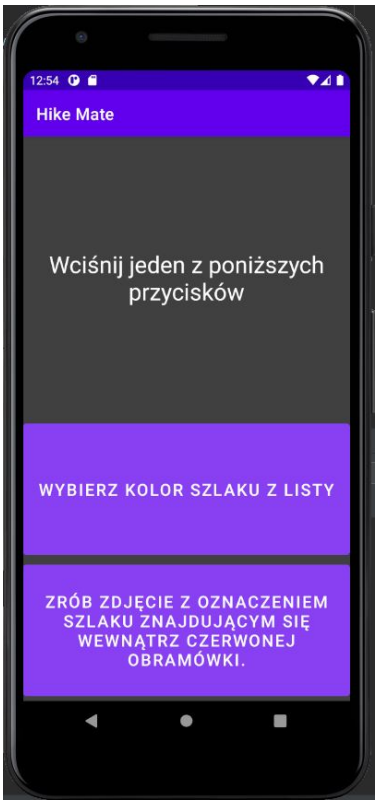

FinalActivity
-map : Button
-back : Button
-altitudeProfile : Button
-currentLoc : LatLng
-rData : String[]
-poly : Poly
-routeName : TextView
-routeLength : TextView
-routeAltitude : TextView
-getRouteData() : String[]
-readCSV() : String[]
MapsActivity
-mMap : GoogleMap
-currentLoc : LatLng
+(Override) onMapReady() : void
DrawMeAChart
-lineChart : LineChart
-lineData : LineData
-entryList : List<Entry>
-stringList : List<String>
-readThirdColumn() : List<String[]>
EmptyFinalActivity
-map : Button
-back : Button
-currentLoc : LatLng
-text : TextView
EmptyMapActivity
-mMap : GoogleMap
-currentLoc : LatLng
+(Override) onMapReady() : void

2.4 Projekt intefejsu użytkownika

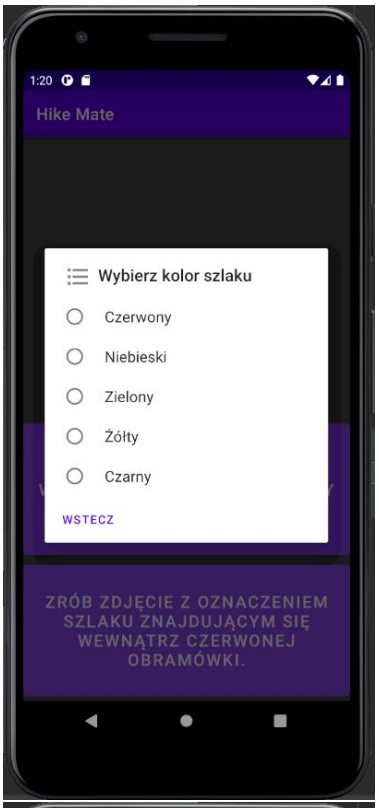
Aby uzyskać oczekiwany efekt prostoty obsługi aplikacji, intefrejs także został uproszczony do minimum. Najprostszy sposób obsługi aplikacji wymaga wciśnięcia jedynie trzech przycisków.

Poniższe grafiki przedstawiają aplikację uruchomioną na emulatorze telefonu Pixel 3a. W zależności od urządzenia fragmenty interfejsu mogą mieć inne rozmiary zachowując jednak takie same proporcje.

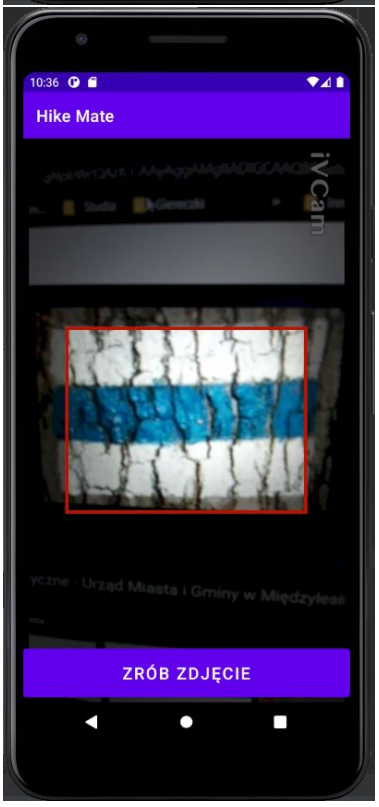
Po uruchomieniu aplikacji i nadaniu jej odpowiednich pozwoleń użytkownika wita ekran, z którego może przejść bezpośrednio do aparatu, lub określić kolor swojego szlaku z listy:



Po wciśnięciu przycisku odpowiadającego za wybór koloru z listy wyświetla się lista pojedynczego wyboru:

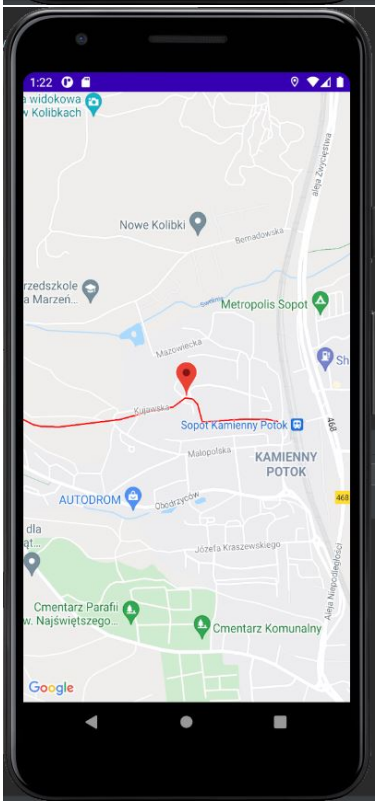


W przypadku wybrania opcji wykonania zdjęcia, aplikacja uruchomi aparat z zaznaczonym fragmentem, w którym ma się znajdować oznaczenie szlaku:



Jeśli aplikacja odnajdzie szlak o odpowiadającym kolorze, niezależnie od tego, czy za pomocą zdjęcia, czy po wyborze z listy, użytkownik zobaczy odpowiadające informacje:

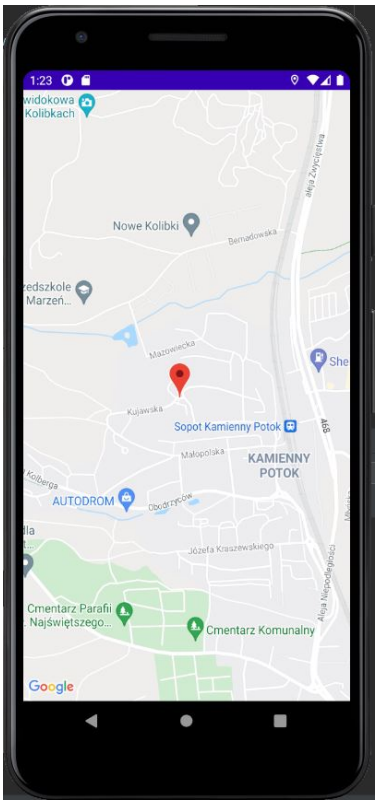
Z tego miejsca, użytkownik może włączyć mapę z narysowaną trasą, jaką przebiega szlak (istnieje możliwość przybliżania, oddalania oraz przesuwania mapy):



Użytkownik może także wybrać opcję rysowania profilu wysokościowego (przykład dla szlaku Kartuskiego; istnieje możliwość przybliżenia fragmentu wykresu):

W przypadku, gdy aplikacja z jakiegoś powodu nie odnajdzie szlaku, na którym znajduje się użytkownik (np. przy niepoprawnym wyborze koloru z listy lub złym wykonaniu zdjęcia) użytkownik zostanie o tym poinformowany:

Użytkownik wciąż może w takiej sytuacji sprawdzić swoją pozycję na mapie:



3 Implementacja

3.1 Architektura aplikacji

Aplikacja została napisana w architekturze klasycznej dla androida, tj. opierającej się na klasach aktywności (ang. Activities) i widoków (ang. Views). "Cykl życia" aktywności zakłada siedem funkcji, wykonywanych w odpowiedniej kolejności, tj. onStart(), onCreate(), onResume(), onPause(), onStop(), onRestart() oraz onDestroy() [2]. Aplikacja Hike Mate, ze względu na brak funkcji działających w tle oraz brak konieczności wykonywania czynności przy zamykaniu jej, korzysta jedynie z dwóch pierwszych funkcji tego cyklu, resztę pozostawiając puste. Funkcje te, w celu skorzystania z nich, muszą zostać nadpisane (ang. Override), w związku z czym aplikacja sama z siebie tak czy inaczej je wykonuje - zazwyczaj jako puste, nic nie robiące, funkcje.

Aplikacja, jak każda inna aplikacja napisanaw języku Java z myślą o systemie Android, zmuszona jest do wykorzystywania plików .gradle, w których znajdują się podstawowe informacje wymagane do zbudowania aplikacji i/lub uruchomienia jej na systemie Android. W aplikacji będącej podmiotem tej pracy są trzy pliki .gradle, jeden dotyczy całości projektu, zawiera informacje o repozytoriach, z których aplikacja korzysta w momencie budowania się oraz o czynności usuwania aplikacji - domyślnie, po odinstalowaniu aplikacji, usuwa także jej folder. Drugim jest plik dotyczący modułu HikeMate.app, w którym znajdują się informacje dotyczące wersji systemów Android, na których aplikacja będzie działać oraz modułów, które implementuje. Są w nim także informacje dotyczące aktualnej wersji aplikacji i kompatybilności z kompilatorami języka Java. W trzecim, ostatnim, pliku typu .gradle znajduje się konfiguracja modułu OpenCV zaimplementowanego na potrzeby aplikacji Hike Mate.

Ponad to wewnątrz aplikacji jest także częściowo automatycznie wygenerowany plik AndroidManifest.xml - są w nim podstawowe informacje na temat wykorzystywanych przez aplikację pozwoleń i funkcji systemu Android. Znajduje się tam też informacja dotycząca aktywności składających się na całość aplikacji oraz wskazująca systemowi

Android, od której aktywności zaczyna działać aplikacja.

Wszystkie pliki pomocnicze dla aplikacji systemu Android muszą znajdować się wewnątrz jednego z czterech folderów wewnątrz katalogu res obsługiwanym przez bibliotekę języka Java dostosowaną pod system Android o takiej samej nazwie. Pliki zawierające informacje o szlakach w formacie *.plt znajdują się w podkatalogu assets. Wszystkie pliki .xml zawierające kolory, wymiary figur geometrycznych, łańcuchy znaków itp. znajdują się w folderze values. Każda z wartości w tych plikach ma przydzielony automatycznie właściwy sobie sześciocyfrowy numer ID typu Integer, a aplikacja korzysta z nich jako ze stałych wartości globalnych. Figury geometryczne i ikony powinny być w folderze drawable, a pliki .xml zawierające szkielet graficznego interfejsu muszą być w folderze layout - w innym wypadku funkcja odpowiadająca za wczytanie ich zwyczajnie ich nie znajdzie.

3.2 Szczegółowe omówienie działania aplikacji

Dalsza część omówienia implementacji wyszczególnia wszystkie aktywności aplikacji w najprostszej (z punktu użytkowania) kolejności ich aktywacji.

3.2.1 Sprawdzanie pozwoleń i aktywność startowa

Aplikacja rozpoczyna swoje działanie w aktywności o nazwie CheckLocalisationPermission.java, gdzie sprawdzane jest pozwolenie na wykorzystanie danych lokalizacyjnych modułu gps wbudowanego w urządzenie - aktywność ta nie ma swojego interfejsu. Jeśli aplikacja nie otrzymała jeszcze takiego pozwolenia, użytkownik dostaje standardowe (zaimplementowane wewnątrz samego systemu operacyjnego) pytanie o zezwolenie na korzystanie z wymaganych elementów. Jeśli pozwolenie nie zostanie udzielone, aplikacja wyłącza się. Po udzieleniu pozwolenia użytkownik przenoszony jest do aktywności startowej, czyli LauncherActivity.java. To samo dzieje się także jeśli aplikacja już wcześnie otrzymała stosowne pozwolenia - w takiej sytuacji aplikacja sprawdza, czy

pozwolenie jest uzyskane i jeśli otrzyma wynik pozytywny, automatycznie przechodzi do aktywności startowej.

Wewnątrz aktywności startowej użytkownik widzi komunikat informujący o konieczności wybrania opcji, z której chce skorzystać - może wybrać kolor z listy lub zrobić zdjęcie oznaczenia szlaku. Po wciśnięciu przycisku odpowiadającego za robienie zdjęcia, aplikacja sprawdza, czy urządzenie wyposażone jest w aparat - jeśli tak, domaga się pozwolenia na korzystanie z niego, jeśli nie - użytkownik zostaje o tym poinformowany w informacji pojawiającej się na dole ekranu na kilkanaście sekund (jest to Toast, typowy dla aplikacji mobilnych - mały komunikat typu "pop up" [3]). W przypadku odmowy wykorzystania aparatu użytkownik zostaje w aktywności startowej aplikacji z tymi samymi opcjami, co chwilę wcześniej - ponownie może wybrać aparat i ponownie zostanie zapytany o pozwolenie. Może też wcisnąć drugi przycisk i wybrać kolor z listy.

3.2.2 Robienie zdjęcia lub wybór koloru z listy

W przypadku wciśnięcia przycisku odpowiadającego za wybór koloru z listy, na ekranie wyświetla się prosta lista jednorazowego wyboru typu "pop up" (tj. wyświetla się ona jako nowa warstwa, przyciemniając interfejs aktywności, znajdujący się w tle). Na liście widnieje pięć kolorów - pięć oznaczeń szlaków PTTK. Po wybraniu koloru użytkownik przenoszony jest do aktywności końcowej, a razem z nim przekazywany jest wewnątrz obiektu klasy Intent [4] (pl. Zamiar/Intencja) obiekt klasy Parcel (pl. Paczka), który zawiera obiekt klasy String (pl. Sznur, tu - łańcuch znaków typu char [6]) określający wybrany przez niego kolor.

W przypadku robienia zdjęcia natomiast, jest ono zapisywane w folderze cache i nigdy nie usuwane, co najwyżej zastępowane przez kolejne zdjęcie wykonane przez aplikację Hike Mate - nie ma potrzeby usuwania zdjęcia, ponieważ jest ono za pomocą biblioteki OpenCV przycinane do rozmiaru 500 x 150 px, przy którym to rozmiarze zajmuje pomijalne dla nowoczesnych urządzeń ilości miejsca (przy samym systemie zajmującym nawet do paru gigabajtów, obraz o wielkości rzędu 120 kilobajtów nie ma

większego znaczenia). System Android jest ponadto wyposażony w automatyczną funkcję działającą przy braku miejsca - usuwa pliki z folderów cache wszystkich aplikacji, dopóki nie uzyska miejsca lub nie skończą się pliki - zwykle ma miejsce jednak ta druga opcja, ponieważ w folderach cache rzadko kiedy przechowywane są pliki o znaczących wielkościach.

Po zrobieniu zdjęcia i wycięciu interesującego nas fragmentu określany jest dominujący kolor, który zostanie podany do aktywności końcowej. Jest to wykonywane za pomocą modułu Palette, oficjalnego, przeznaczonego na systemy Android oprogramowania. Moduł ten służy głównie do odnajdywania palety barw na zdjęciu, konkretnie, 6 odcieni dominujących na zdjęciu, kolejno Vibrant, Vibrant Dark, Vibrant Light, Muted, Muted Dark, Muted Light [7] i wykorzystywania ich w celach estetycznego wykonania interfejsu aplikacji - w przypadku Hike Mate jednak jedyną wykorzystaną jego funkcją jest odnalezienie dominującego koloru na obrazie. Odpowiednia funkcja modułu Palette zwraca kolor w postaci heksadecymalnej, trudnej do rozszyfrowania, postaci ARGB, z której inna metoda tego samego modułu może jednak wyciągnąć liczby odpowiadające systemowy RGB, odpowiednio Red, Green i Blue. Znajomość tych liczb jednak nie ułatwia zbytnio automatycznego rozpoznawania barw, dlatego też z pomocą przychodzi funkcja zamieniająca format RGB na HSV (Hue Saturation Value).

Format ten przedstawiany jest za pomocą trzech liczb - dwie z nich w zakresie 0 - 100, trzecia, 0 - 360. Dwie pierwsze oznaczają natężenie i jasność barwy w procentach, ostatnia natomiast odcień tej barwy na kole barw. Dla przykładu - centrum barwy zielonej odpowiada kąt 120°, centrum barwy niebieskiej - 240°, centrum barwy czerwonej - 0 lub 360°[8].

Posiadając wartość koloru w formacie HSV bardzo łatwo jest już określić jeden z pięciu kolorów szlaków - służy do tego prosta metoda oparta na if-else zwracająca kolor w zależności od otrzymanej wartości, np. w przypadku wartości odpowiadającej za jasność barwy wynoszącej mniej niż 20%, aplikacja wie, że mamy do czynienia z kolorem czarnym, w przypadku natomiast wartości określającej kąt wynoszącej pomiędzy 30 a 89 - otrzymujemy kolor żółty. Niestety nie jest to rozwiązanie idealne - problemem mo-

że być w tym wypadku zdjęcie wykonane w bardzo słabym oświetleniu. Jest to jednak jedyne możliwe rozwiązanie - inne, polegające na sprawdzaniu dominującego koloru za pomocą liczenia pikseli o danym odcieniu z użyciem OpenCV, wymaga zbyt wiele mocy obliczeniowej, przez co urządzenia ze słabszymi procesorami mogłyby potrzebować nawet kilku/kilkunastu minut do wykrycia koloru - mowa tutaj o urządzeniach przenośnych, a zatem zasilanych baterią, której zużycie by w tym momencie diametralnie rosło.

3.2.3 Odnajdywanie trasy na podstawie koloru i pozycji użytkownika

Po uzyskaniu koloru przez zdjęcie lub wybór z listy w aktywności CreateRoute.java tworzony jest obiekt klasy RoutesBase zawierający wszystkie dostępne w regionie Trójmiasta szlaki o danym kolorze. Obiekt ten przekazywany jest do kolejnej aktywności, GetCurrentRoute.java, w której, za pomocą modułu GPS wbudowanego w urządzenie obsługujące aplikację i odpowiednich bibliotek natywnych dla systemu Android, odczytywana jest pozycja użytkownika - na jej podstawie tworzony jest obiekt klasy Poly, który zostanie utworzony w zależności od wartości zwróconej przez zapętloną funkcję PolyUtil.isLocationOnPath, sprawdzającej, czy lokacja użytkownika jest w promieniu 100m od którejś z tras o odpowiadającym kolorze wewnątrz obiektu RoutesBase. Jeśli tak - obiekt klasy Poly jest przekazywany dalej z informacją o nazwie szlaku - w przeciwnym wypadku obiekt ten wciąż jest tworzony, jednakże wpisywane są do niego informacje o braku takiej trasy, które odczytuje kolejna aktywność. Tworzenie obiektów klas RoutesBase jak i Poly jest wykonywane w oddzielnym wątku, na który główny wątek odpowiedniej aktywności musi zaczekać - jest to zabieg służący jedynie zapewnieniu aplikacji stabilności. Lepszą metodą byłoby zastosowanie w tym celu korutyn, niestety w języku Java nie są one natywnie zaimplementowane.

Czas pracy aplikacji zależy w dużym stopniu od opisanej powyżej funkcji, dlatego też odczytanie koloru i utworzenie na tej podstawie niewielkiej bazy zawierającej jedynie szlaki o danym kolorze jest kluczowym elementem aplikacji.

3.2.4 Aktywność końcowa

Po utworzeniu obiektów klas RouteBase i Poly, użytkownik przenoszony jest do aktywności końcowej aplikacji, w której w zależności od zawartości obiektu klasy Poly ma dwie różne możliwości.

W pierwszym przypadku, jeżeli wartość zmiennej typu String nazwanej currentRouteName będzie równa "placeholder", czyli jej wartości domyślnej, użytkownik przekierowywany jest na ekran informujący go o braku trasy o danym kolorze w miejscu, w którym się aktualnie znajduje - wartość domyślna zmiennej currentRouteName zostaje zachowywana w sytuacjach, kiedy:

- użytkownik wybierze z listy kolor inny niż ten, którym oznaczony jest szlak, na którym się znajduje.
- użytkownik wybierze dowolny kolor nie znajdując się na żadnym szlaku.
- szlak, na którym znajduje się użytkownik nie znajduje się w bazie danych aplikacji (może to być szlak poza obrębem Trójmiasta lub szlak nie określony jako pieszy, tylko np. rowerowy).
- zdjęcie zostało błędnie wykonane (np. oznaczenie nie znajdowało się wewnątrz czerwonej ramki).

W tym momencie uruchamiana jest aktywność EmptyFinalActivity.java, w której poza informacją o braku trasy, użytkownik może przejść do aktywności EmptyMapActivity.java, będącej fragmentem (rodzaj UI w systemie Android - cechuje go możliwość wielokrotnego użytku [9]) zawierającym mapę Google z zaznaczoną aktualną lokalizacją.

W drugim przypadku, w którym szlak został poprawnie odnaleziony, wartość zmiennej currentRouteName jest zmieniana na nazwę pliku zawierającego dane dotyczące przebiegu szlaku, na którym znajduje się użytkownik. Tworzony jest odpowiedni Intent, za pomocą którego użytkownik przenoszony jest do aktywności FinalActivity.java,

w której na ekranie pojawiają się podstawowe informacje o szlaku i o jego przebiegu (tj. jego kolor, nazwa, długość i przewyższenie - różnica pomiędzy wysokością nad poziomem morza najwyższego i najniższego punktu na szlaku) oraz przyciski umożliwiające przygotowanie mapy Google z naniesioną na nią trasą, jaką przebiega szlak oraz narysowanie odpowiedniego wykresu przedstawiającego profil wysokościowy danego szlaku. Aktywności noszą nazwę, odpowiednio, MapsActivity.java i DrawMeAChart.java.

Zarówno w aktywności EmptyFinalActivity.java jak i w FinalActivity.java użytkownik ma możliwość wciśnięcia przycisku odpowiadającego za powrót do aktywności startowej (tj. wyboru koloru z listy lub zrobienia zdjęcia).

3.2.5 Przygotowanie i odczyt mapy Google

Pierwsza z wymienionych w powyższym punkcie aktywności, do których użytkownik może dostać się z FinalActivity.java, służy do wyświetlania mapy - w obiekcie klasy Poly o nazwie szlaku innej niż domyślna znajduje się przygotowana w momencie tworzenia obiektu lista zawierająca dane z pierwszej i drugiej kolumny odpowiedniego pliku *.plt. Są to odpowiednio szerokość i długość geograficzna (z dokładnością do 6 miejsca po przecinku, tj. w przybliżeniu, do centymetra [10]). Uruchamiany jest fragment zawierający mapę Google, na który nanoszona jest linia składająca się z połączonych ze sobą punktów w formacie LatLng (klasa pochodząca z biblioteki służącej do obsługi map w systemach Android). Na mapie zaznaczona jest także aktualna pozycja użytkownika.

3.2.6 Rysowanie i odczyt profilu wysokościowego

Druga z aktywności możliwych do wyboru w FinalActivity.java odczytuje z pliku *.plt o przekazanej wewnątrz obiektu klasy Poly nazwie trzecią kolumnę - zawierającą wysokość nad poziomem morza (w metrach) w każdym punkcie zaznaczonym na liście prezentującej trasę. Na podstawie tych danych rysuje profil wysokościowy danej trasy - samo rysowanie profilu nie wymaga zbyt dużej pracy procesora, dlatego też wykres

ten nie jest przechowywany w żadnym obiekcie, a rysowany dynamicznie przy każdym uruchomieniu tej aktywności.

3.3 Użyte technologie

W przygotowaniu aplikacji wykorzystane zostały dwa języki oraz oprogramowanie firm trzecich w wersji freeware. Proces przygotowania plików zawierających przebieg szlaków z użyciem programu OziExplorer, strony internetowej konwertującej trasę z mapy google na plik .gpx oraz skryptu w języku Python3.8 został opisany w następnym punkcie (3.4).

Sama w sobie aplikacja została w całości napisana w środowisku Android Studio 4.2.0 w języku Java w wersji SE 8 (tj. JDK 1.8 lub 8). W procesie tworzenia aplikacji zostały użyte głównie oficjalne biblioteki firmy Google dostosowane do systemu Android. Aplikacja również korzysta ze specyfikacji Androidx ułatwiającej implementację obsługiwanych przez ten system bibliotek (dotyczy jedynie poszczególnych bibliotek Google'a).

3.3.1 Zestawienie oficjalnych pakietów firmy Google wykorzystanych w budowie aplikacji

Wszystkie wymienione poniżej wersje są najnowszymi na dzień 05.06.2021:

- Android Maps Utils w wersji 2.2.3 - moduł ten jest wykorzystany jedynie ze względu na bibliotekę PolyUtils użytą do sprawdzenia, czy użytkownik znajduje się na danym szlaku.
- Google Play Services Maps 17.0.1 - moduł wykorzystany do tworzenia fragmentu zawierającego mapę Google.
- Google Play Services Location 18.0.0 - moduł wykorzystany do określenia i oznaczenia pozycji użytkownika na mapie.

- Android Appcompat 1.3.0 - podstawowy moduł Androida zawierający implementacje aktywności

- Android Palette 1.0.0 - moduł wykorzystany do odczytania dominującego na wyciętym fragmencie zdjęcia koloru

3.3.2 Zestawienie nieoficjalny pakietów języka Java wykorzystanych w budowie aplikacji

Wybrane wersje zaimplementowanych modułów są stabilne i sprawne, czego nie można w każdym przypadku powiedzieć o ich najnowszych wersjach:

- CameraKit w wersji 1.0.0-beta3.10 - moduł służący do wyświetlenia podglądu z kamery dostępnej w urządzeniu z systemem Android (jeśli ją posiada) oraz do wykonania zdjęcia. Jest to projekt otwarty na licencji MIT dostępny na stronie internetowej GitHub.com [11].

- Kotlin stdlib sdk7 oraz Kotlin Coroutines Library - część kodu modułu CameraKit jest napisana w języku Kotlin [11], zatem, o ile moduły te nie są wykorzystywane przez podmiot tej pracy bezpośrednio, są one wymagane do wykorzystania oprogramowania CameraKit. Język Kotlin jest interoperacyjny z językiem Java [12], w związku z czym wykorzystanie tej technologii w projekcie opartym na Javie nie spowalnia działania samej aplikacji.

- MPAndroidChart w wersji 3.1.0 - moduł stworzony do rysowania grafów i wykresów w systemie android, w aplikacji Hike Mate została wykorzystana z niego jedynie biblioteka dotycząca wykresów liniowych. Moduł ten oparty jest na licencji Apache License, Version 2.0 i dostępny na stronie internetowej GitHub.com [13].

- OpenCV 3.4.3 w wersji przeznaczonej na systemy Android - moduł wykorzystany do wycięcia fragmentu "interesującego" aplikację (ang. Region of interest)

zdjęcia w celu poprawnego zidentyfikowania dominującego koloru w odpowiedniej części oznaczenia szlaku. Oprogramowanie to jest oparte na licencji 3-clause BSD license.

3.4 Przygotowywanie plików zawierających dane dotyczące szlaków

Proces przygotowania samych plików *.plt jest dość prosty, lecz wymaga oprogramowania zewnętrznego w postaci strony internetowej https://mapstogpx.com/ oraz Des Newman's OziExplorer, dostępnego w wersjach shareware i freeware, przy czym funkcje wymagane do przygotowania plików potrzebnych do właściwej pracy aplikacji dostępne są w wersji darmowej.

3.4.1 Proces konwersacji trasy map Google na plik formatu *.plt

Szlaki i ich dokładne opisy dostępne są na stronie internetowej PTTK [15]. Proces służący generacji plików wymaganych do poprawnego działania aplikacji to:

1. W pierwszej kolejności należy utworzyć trasę pieszą na mapie Google poprzez manualne "wyklikanie" odpowiednich punktów - w apliakcji nie można zastosować trasy wyszukanej przez Google po wprowadzeniu listy punktów, ze względu na ograniczenie tych punktów.

2. Link zawierający trasę wyznaczoną w powyższym punkcie należy wkleić na stronę https://mapstogpx.com/, która zwróci plik w formacie *.gpx zawierający odpowiednią trasę.

3. Otrzymany plik *.gpx należy zaimportować w programie OziExplorer - na ekranie pojawi się mapa zawierająca trasę.

4. Trasę trzeba zapisać w formacie *.plt.

5. Ze względu na ograniczenie punktów wprowadzanych na mapę Google (maksymalnie 10), punkty 1-4 trzeba powtarzać do uzyskania zadowalającego efektu, tj. ułożenia całej trasy.

6. Wszystkie powstałe pliki *.plt należy skleić w odpowiedniej kolejności.

Powyższe operacje należy wykonać ręcznie.

Niektóre ze szlaków udostępnione są w formie *.plt w serwisie https://traseo.pl dla zarejestrowanych użytkowników - regulamin strony nie określa licencji na jakiej udostępniane są te pliki, są one natomiast podpisane przez oprogramowanie OziExplorer, dlatego też w w tworzeniu aplikacji zdecydowałem się wykorzystać te pliki.

3.4.2 Przygotowywanie plików z użyciem języka Python3.8

Po otrzymaniu plików *.plt w wyniku powyższego procesu lub ściągnięciu z odpowiedniego źródła są one formatowane za pomocą prostego programu pomocniczego w języku Python3.8. Program zawiera jedynie trzy metody, z której każda odczytuje pliki za pomocą biblioteki Pandas:

- Pierwsza ma za zadanie usunięcie niepotrzebnych dla aplikacji Hike Mate kolumn wygenerowanych przez oprogramowanie OziExplorer i pozostawienie jedynie tych, w których zawarte są: szerokość geograficzna, długość geograficzna i wysokość punktów nad poziomem morza.

- Druga metoda tworzy nowy plik w formacie *.csv zawierający różnicę pomiędzy odczytanymi z odpowiedniej kolumny pliku *.plt wartościami określającymi najwyższy oraz najniższy punkt na trasię (przewyższenie).

- Trzecia i ostatnia wykorzystuje bibliotekę GeoPy w celu odnalezienia całkowitej długości trasy na podstawie punktów jej przebiegu.

3.5 Całkowity wykaz użytych bibliotek

Poniższe listy przedstawiają jedynie biblioteki nie będące natywnie wbudowane w język Java w użytej wersji, tj. wymagające manualnego zaimportowania. W związku z powyższym biblioteki odpowiadające za takie klasy jak np. String, Integer, czy Thread nie są tutaj wymienione.

Podstawowe biblioteki systemu Android pozwalające na implementacje klas aktywności zgodnych z architekturą tego systemu:

- androidx.appcompat.app.AppCompatActivity;
- androidx.core.app.ActivityCompat;
- androidx.core.content.ContextCompat;
- androidx.core.content.Context;
- androidx.fragment.app.FragmentActivity;

Biblioteki służące sprawdzaniu zezwoleń aplikacji na wykonywanie swoich czynności:

- android.Manifest;
- android.content.pm.PackageManager;

Biblioteki języka Java pozwalające na odczyt/zapis plików graficznych oraz *.plt:

- java.io.IOException;
- java.io.FileOutputStream;
- java.io.File;
- java.io.BufferedReader;
- java.io.InputStream;
- java.io.InputStreamReader;

oraz na zarządzanie nimi z poziomu urządzenia z systemem Android:

- android.content.AssetManager;
- android.content.res.Resources;

Biblioteka modułu CameraKit pozwalająca na podgląd obrazu z kamery urządzenia:

- com.camerakit.CameraKitView;

Biblioteki służące do wykonywania transakcji pomiędzy poszczególnymi aktywnościami oraz przesyłanie pomiędzy nimi danych w postaci "zapakowanych" obiektów:

- android.content.Intent;
- android.content.IntentSender;
- android.os.Parcel;
- android.os.Parcelable;
- android.os.Bundle;

Biblioteki służące do stworzenia i obsługi interfejsu użytkownika:

- android.view.View;
- android.widget.Button;
- android.widget.TextView;
- android.widget.Toast;
- androidx.appcompat.app.AlertDialog;

Biblioteki pomocnicze, tj. np. biblioteki wielokrotnie używane w celu zaprogramowania logiki, pisaniu funkcji wielowątkowych, czy wyświetlania logów pomocnych przy debugowaniu:

- android.util.Log;
- androidx.annotation.NonNull;
- android.annotation.SuppressLint;
- java.util.ArrayList;
- java.util.List;
- java.util.Objects;
- java.util.Map;
- java.util.HashMap;
- android.os.Looper;

Biblioteki modułu MPAndroidChart służące do rysowania wykresów (rysowanie profilu wysokościowego):

- com.github.mikephil.charting.charts.LineChart;
- com.github.mikephil.charting.components.XAxis;
- com.github.mikephil.charting.components.YAxis;
- com.github.mikephil.charting.data.Entry;
- com.github.mikephil.charting.data.LineData;
- com.github.mikephil.charting.data.LineDataSet;

Biblioteki modułów obsługujących grafikę służących wykrywaniu koloru na wykonanym przez użytkownika zdjęciu:

- android.graphics.Color;
- org.opencv.android.OpenCVLoader;
- org.opencv.core.Mat;
- org.opencv.core.Size;
- org.opencv.imgcodecs.Imgcodecs;
- org.opencv.imgproc.Imgproc;
- android.graphics.Bitmap;
- android.graphics.BitmapFactory;
- androidx.palette.graphics.Palette;

Biblioteki służące do implementacji mapy Google i jej obsługi:

- com.google.android.gms.maps.CameraUpdateFactory;
- com.google.android.gms.maps.GoogleMap;
- com.google.android.gms.maps.OnMapReadyCallback;
- com.google.android.gms.maps.SupportMapFragment;
- com.google.android.gms.maps.model.LatLng;
- com.google.android.gms.maps.model.Marker;
- com.google.android.gms.maps.model.MarkerOptions;
- com.google.android.gms.maps.model.PolylineOptions;

- com.google.maps.android.PolyUtil;

Biblioteki służące do wykrywania pozycji użytkownika na podstawie modułu gps urządzenia:

- com.google.android.gms.common.api.ResolvableApiException;
- com.google.android.gms.location.FusedLocationProviderClient;
- com.google.android.gms.location.LocationCallback;
- com.google.android.gms.location.LocationRequest;
- com.google.android.gms.location.LocationResult;
- com.google.android.gms.location.LocationServices;
- com.google.android.gms.location.LocationSettingsRequest;
- com.google.android.gms.location.LocationSettingsResponse;
- com.google.android.gms.location.SettingsClient;
- com.google.android.gms.tasks.Task;

Biblioteki języka Python3.8 wykorzystane do restrukturyzacji plików *.plt utworzonych przez oprogramowanie OziExplorer:

- Pandas
- GeoPy

4 Testy

Testy przeprowadzono manualnie korzystając z dwóch emulatorów urządzeń z systemem Android 11.0 wbudowanych w oprogramowanie Android Studio oraz z dwóch telefonów z systemami Android w różnych wersjach.

Emulatory:

- Pixel 3a 30 x86 - emulator telefonu Google Pixel 3a w wersji z Androidem 11 (API 30) i procesorem o architekturze x86, posiadający wyświetlacz 1080 x 2200 px; 440 dpi (dpi - ang. dots per inch, tj. liczba punktów przypadających na 1 cal, w tym przypadku dotyczy pikseli [16]).

- Galaxy Nexus API 30 - emulator telefonu Samsung Galaxy nexus w wersji z Androidem 11 (API 30) i procesorem o architekturze x86, posiadający wyświetlacz 720 x 1280 px; xhdpi (ok. 320 dpi, wyświetlacz o wysokim zagęszczeniu pikseli (ang. extra-high-density screen) [16]).

Urządzenia fizyczne (telefony):

- Huawei Mate 10 Lite, wersja Android 8.0 Oreo (API 26) z procesorem Hisilicon Kirin 659 i ekranem 1080 x 2160 px [17].

- Samsung Galaxy A51, wersja Android 10 Q (API 29) z procesorem Samsung Exynos 9611 i ekranem 1080 x 2400 px [18].

Wiele testów przeprowadzanych było na etapie powstawania aplikacji za pomocą narzędzi programu Android Studio do debugowania oraz logowania potencjalnych błędów czy oczekiwanych wartości odpowiednich funkcji.

4.1 Scenariusz testowania

Poniżej opisane są jedynie testy przeprowadzane na skończonej aplikacji w celu potwierdzenia jej poprawnego funkcjonowania.

- W pierwszej kolejności, po instalacji aplikacji na docelowym urządzeniu (także, emulatorze), sprawdzane było zachowanie aplikacji w zależności od przyznawanych uprawnień do wykorzystania modułu GPS oraz kamery.

- Następnie sprawdzany był graficzny interfejs użytkownika, w celu sprawdzenia, czy wszystko wyświetla się tak jak powinno, tj. czy napisy nie są ucinane w zależności od rozdzielczości i wielkości ekranu urządzenia oraz czy przyciski nic nie przesłaniają. Także - czy ramka, wewnątrz której docelowo ma się znajdować oznaczenie szlaku w przypadku robienia zdjęcia prawidłowo się wyświetla.

- Dalej sprawdzana była funkcjonalność modułu GPS poprzez odpowiednie operacje: w przypadku emulatorów ustawiana była manualnie lokacja w niewielkiej odległości od przebiegu szlaku i testowane było, czy w odpowiednim miejscu znajduje się szlak o danym kolorze. W przypadku urządzeń fizycznych aplikacja testowana była przez manualne sprawdzanie powyższych funkcji w momencie, w którym użytkownik znajdował się (lub nie, w zależności od testu) na odpowiednim szlaku.

- Kolejna testowana była funkcjonalność kamery i rozpoznawania koloru na podstawie wykonanego przez aplikację zdjęcia. W przypadku urządzenia fizycznego testy mogą być niepełne, ze względu na ograniczony zasięg do szlaków jak i ich oznaczeń. W przypadku emulatorów natomiast, testy zostały przeprowadzone na trzech zdjęciach dla każdego możliwego koloru szlaku, a wynik funkcji sprawdzany był poprzez wypisywanie wyniku na terminalu LogCat programu Android Studio. Do wykonania zdjęć z użyciem emulatorów wykorzystano kamerkę internetową wbudowaną w laptopa

- Następnie testowany był profil wysokościowy pod kątem poprawnego wyświetlania narysowanego wykresu pod względem estetycznym oraz poprawnościowym.

- Ostani etap testu polegał na sprawdzeniu poprawności funkcjonowania mapy Google zaimplementowanej w projekcie, poprawnym wyświetlaniu markera oznacza-

jącego pozycję użytkownika oraz odpowiednim rysowaniu przebiegu szlaku.

4.1.1 Miniatury zdjęć, na których testowana była funkcja rozpoznająca kolor

4.2 Raport testów

Przeprowadzany test	**Wynik testu**
Przyznanie uprawnienia do korzystania z modułu GPS	Pozytywny
Brak uprawnienia do korzystania z modułu GPS	Pozytywny
Przyznanie uprawnienia do korzystania z kamery	Pozytywny
Brak uprawnienia do korzystania z kamery	Pozytywny
Poprawne wyświetlanie graficznego interfejsu użytkownika	Pozytywny
Poprawne wyświetlanie ramki aparatu	Pozytywny
Odnajdywanie szlaków w odpowiednich miejscach	Pozytywny
Nie odnajdywanie szlaków w miejscach, w których ich nie ma	Pozytywny
Nie odnajdywanie szlaków po wybraniu z listy złego koloru	Pozytywny
Podgląd obrazu z kamery	Pozytywny
Wykonywanie zdjęcia	Pozytywny
Wyświetlanie pustej mapy	Pozytywny
Wykrywanie kolorów obsługiwanych przez aplikację: czarnego, czerwonego, zielonego, niebieskiego lub żółtego	Częściowo pozytywny[1]
Wykrywanie kolorów nieobsługiwanych przez aplikację	Pozytywny
Rysowanie profilu wysokościowego	Pozytywny
Przybliżanie profilu wysokościowego	Pozytywny
Przesuwanie obrazu profilu wysokościowego	Pozytywny
Rysowanie linii na mapie	Pozytywny
Ustawianie markera na mapie w lokalizacji użytkownika	Pozytywny
Obsługa mapy Google	Pozytywny
Powrót do aktywności startowej	Pozytywny

[1] Testy kończyły się niepowodzeniem jedynie w przypadkach złego oświetlenia - światła odbijającego się od zdjęcia lub niedostatecznego doświetlenia

Bibliografia

[1] Wikipedia, historia wersji systemu Android,

`https://en.wikipedia.org/wiki/Android_version_history`,

Treść strony dostępna w dniu 25.05.2021.

[2] Cykl życia aktywności (ang. Activity),

`https://developer.android.com/guide/components/activities/activity-lifecycle`,

Treść strony dostępna w dniu 05.06.2021.

[3] Toast w systemie Android,

`https://developer.android.com/guide/topics/ui/notifiers/toasts`,

Treść strony dostępna w dniu 05.06.2021.

[4] Dokumentacja dotycząca klasy Intent,

`https://developer.android.com/reference/android/content/Intent`,

Treść strony dostępna w dniu 05.06.2021.

[5] Dokumentacja dotycząca klasy Parcel,

`https://developer.android.com/reference/android/os/Parcel`,

Treść strony dostępna w dniu 05.06.2021.

[6] Dokumentacja dotycząca klasy String (łańcuch char'ów),

`https://docs.oracle.com/javase/7/docs/api/java/lang/String.html`,

Treść strony dostępna w dniu 05.06.2021.

[7] Dokumentacja dotycząca modułu Palette,

`https://developer.android.com/reference/androidx/palette/graphics/Palette`,

Treść strony dostępna w dniu 05.06.2021.

[8] Wikipedia, opis modelu przestrzeni barw HSV,

https://pl.wikipedia.org/wiki/HSV_(grafika)

Treść strony dostępna w dniu 05.06.2021.

[9] Dokumentacja klasy Fragment,

https://developer.android.com/guide/fragments,

Treść strony dostępna w dniu 09.06.2021.

[10] Przeliczenie współrzędnych geograficznych na system metryczny, strona United States Naval Academy

https://www.usna.edu/Users/oceano/pguth/md_help/html/approx_equivalents.htm,

Treść strony dostępna w dniu 09.06.2021.

[11] CameraKit GitHub,

https://github.com/CameraKit/camerakit-android,

Treść strony dostępna w dniu 05.06.2021.

[12] Interoperacyjność języków Java i Kotlin,

https://kotlinlang.org/docs/java-interop.html,

Treść strony dostępna w dniu 05.06.2021.

[13] MPAndroidChart GitHub,

https://github.com/PhilJay/MPAndroidChart,

Treść strony dostępna w dniu 05.06.2021.

[14] Licencja - OpenCV,

https://opencv.org/license/

Treść strony dostępna w dniu 05.06.2021.

[15] Strona internetowa PTTK,

http://pomorskieszlakipttk.pl/szlaki-piesze/trojmiejski/,

Treść strony dostępna w dniu 10.06.2021.

[16] Zagnieżdżenie pikseli na ekranach urządzeń z systemem Android,

https://developer.android.com/training/multiscreen/screendensities,

Treść strony dostępna w dniu 05.06.2021

[17] Odczyt specyfikacji z ustawień telefonu Huawei Mate 10 Lite w wersji fizycznej.

[18] Specyfikacja urządzenia Samsung Galaxy A51,

https://phonesdata.com/pl/smartphones/samsung/galaxy-a51-5458463/,

Treść strony dostępna w dniu 05.06.2021