UNIWERSYTET RZESZOWSKI

Kolegium Nauk Przyrodniczych

Paweł Durda

Nr albumu: 96449

Informatyka

Projekt i implementacja mobilnej aplikacji do informowania o zdarzeniach drogowych

Praca inżynierska

Praca wykonana pod kierunkiem

dra inż. Wiesława Paji

Rzeszów 2020

RZESZOW UNIVERSITY

College of Natural Sciences

Paweł Durda

96449

Computer Science

The Project and implementation of a mobile application for reporting road accidents

Type of the thesis: Engineer

The thesis written under the supervision of

dr inż. Wiesław Paja

Rzeszow 2020

Załącznik 1

Promotor

…….
(IMIĘ I NAZWISKO PROMOTORA)

Przyjmuję pracę pt.:
……..……
…….
…….

……..
(PODPIS PROMOTORA)

Załącznik nr 2 do Zarządzenia nr 61/2018
Rektora UR z dnia 19 grudnia 2018 r.

OŚWIADCZENIE

.....
Imię (imiona) i nazwisko studenta/doktoranta

.....
Nazwa jednostki

.....
Nazwa kierunku/ nazwa dyscypliny

.....
Numer albumu:

Oświadczam, że moja praca dyplomowa/rozprawa doktorska pt.: .....
.....
.....

1. została przygotowana przeze mnie samodzielnie*,
2. nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2018 r., poz. 1191 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
3. nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
4. nie była podstawą nadania dyplomu uczelni wyższej lub tytułu zawodowego ani mnie ani innej osobie.

Ponadto oświadczam, że treść pracy przedstawionej przeze mnie do obrony zawarta na przekazywanym nośniku elektronicznym jest identyczna z wersją drukowaną.

....
(miejscowość, data) *(czytelny podpis autora pracy)*

***** *uwzględniając merytoryczny wkład promotora pracy*

Załącznik 3

Oświadczenie

……., dnia …… roku

…….
(IMIĘ I NAZWISKO STUDENTA)

……..
(NR ALBUMU)

Kolegium Nauk Przyrodniczych

…….
(KIERUNEK STUDIÓW)

…….
(RODZAJ I FORMA STUDIÓW)

OŚWIADCZENIE

Oświadczam, że część badawcza niniejszej pracy została wykonana w pracowni …….. Laboratorium ……. w ramach Projektu „Uniwersyteckie Centrum Innowacji i Transferu Wiedzy Techniczno-Przyrodniczej", Nr UDA-RPPK.01.03.00-18-001/10-00, współfinansowanego przez Unię Europejską z Europejskiego Funduszu Rozwoju Regionalnego w ramach Regionalnego Programu Operacyjnego Województwa Podkarpackiego na lata 2007-2013.

…… ……

(PODPIS PROMOTORA) (PODPIS STUDENTA)

Składam serdeczne podziękowania
dr inż. Wiesławowi Paji
za zaangażowanie i merytoryczną pomoc
przy wykonaniu niniejszej pracy.

Spis treści

1. Wstęp

W mijającej dekadzie można było zaobserwować powstanie oraz szybki rozwój aplikacji mobilnych w kategorii/zakresie nawigacji samochodowych. Wraz z rozwojem tego sektora rozszerzano możliwości tych aplikacji o nowe funkcjonalności. Jedną z nich jest możliwość wcześniejszego informowania kierowcy o możliwym zdarzeniu drogowym tak, aby mógł przygotować się na ewentualny manewr drogowy. Na rynku można zaobserwować aplikacje, które oferują taką funkcjonalność, ale są one skierowane do określonych użytkowników, przez co baza danych o zgłoszeniach w trasie jest rozproszona.

Moim głównym zadaniem jest opracowanie uniwersalnej aplikacji tak, aby zwiększyć bazę użytkowników, co pozwoli na zwiększenie efektywności zgłoszeń drogowych. Kierowca sam będzie mógł zdecydować, czy chce używać zewnętrznej aplikacji do nawigacji oraz według własnych preferencji wybrać odpowiadające mu rozwiązanie.

2. Cel pracy

Celem mojej pracy jest zaprojektowanie oraz realizacja aplikacji mobilnej do informowania oraz zgłaszania zdarzeń drogowych na platformie Android. Scharakteryzuję również już istniejące rozwiązania na rynku, które posiadają podobne funkcjonalności, aby wskazać różnice oraz cechy wspólne tych rozwiązań.

Do zrealizowania w/w celu wykorzystałem usługę Firebase, która jest kompletnym narzędziem do zarządzania aplikacjami mobilnymi. Zapewnia ona szybką synchronizację danych pomiędzy użytkownikiem aplikacji i bazą danych w czasie rzeczywistym. Upraszczając proces logowania i rejestracji korzystam z rozwiązania Firebase Authentication, które pozwoli użytkownikowi na uwierzytelnienie się w aplikacji tylko za pomocą konta Google. Do implementacji aplikacji skorzystałem ze znajomości języków programowania Java i TypeScript. Kod źródłowy programu został napisany w środowisku programistycznym Android Studio oraz Visual Studio Code.

Projektowana aplikacja będzie realizować funkcjonalności tj. zgłoszenia nowego zdarzenia, wyświetlenie oraz powiadomienie użytkownika o określonym zdarzeniu. Kierowca będzie mógł ocenić zgłoszenie innego użytkownika ruchu drogowego oraz uzyskać informację o przybliżonej odległości do tego miejsca. Obsługa aplikacji dla użytkownika w trakcie prowadzenia pojazdu powinna być uproszczona, dlatego też aplikacja generuje komunikaty głosowe, a widok usługi obsługującej zgłoszenia działa w tle, nie zakłócając pracy innych aplikacji do nawigacji.

3. Opis zagadnienia

3.1. Zdarzenia drogowe

Zdarzeniem drogowym, można definiować wszystkie sytuacje na drodze, które potencjalnie mogą stanowić zagrożenie dla innych uczestników ruchu drogowego. Wcześniejsza informacja o możliwym niebezpieczeństwie ma na celu zwiększenie czujności u kierowcy, co wpływa pozytywnie na komfort jazdy oraz poprawia bezpieczeństwo w trasie. Użytkownik aplikacji poinformowany o określonym zdarzeniu, będzie mógł z wyprzedzeniem zareagować dostosowując prędkość do zmieniających się warunków, bądź wykonać odpowiedni manewr na jezdni.

Definicja: „*Zdarzenie drogowe to pojęcie nadrzędne wobec wszystkich innych. Zdarzeniem drogowym określamy wszystkie wydarzenia na drodze – niezależnie od tego, czy są jakieś ofiary lub poważne straty materialne.”* [WWW,1].

W celu przedstawienia skali problemu posłużę się raportem Policji [WWW,2] dotyczącego zwiększającego się ruchu drogowego, co automatycznie wpływa na wzrost liczby utrudnień i zagrożeń na drodze.

Liczba pojazdów silnikowych w latach 2009-2018*

Lata	Pojazdy silnikowe		w tym:					
			samochody osobowe		samochody ciężarowe		motocykle	
	Ogółem	2009=100%	Ogółem	2009=100%	Ogółem	2009=100%	Ogółem	2009=100%
2009	22 024 697	100,0	16 494 650	100,0	2 595 485	100,0	974 906	100,0
2010	23 037 149	104,6	17 239 800	104,5	2 767 035	106,6	1 013 014	103,9
2011	24 189 370	109,8	18 125 490	109,9	2 892 064	111,4	1 069 195	109,7
2012	24 875 717	112,9	18 744 412	113,6	2 920 779	112,5	1 107 260	113,6
2013	25 683 575	116,6	19 389 446	117,5	2 962 064	114,1	1 153 169	118,3
2014	26 472 274	120,2	20 003 863	121,3	3 037 427	117,0	1 189 527	122,0
2015	27 409 106	124,4	20 723 423	125,6	3 098 376	119,4	1 272 333	130,5
2016	28 601 037	129,9	21 675 388	131,4	3 179 655	122,5	1 355 625	139,1
2017	29 149 178	132,3	22 109 572	134,0	3 212 690	123,8	1 398 609	143,5
2018	29 656 238	134,6	22 514 047	136,5	3 249 961	125,2	1 428 299	146,5

* Dane GUS na dzień 30.06.2018 r.

Rysunek 1 Statystyki dotyczące liczby zarejestrowanych pojazdów

Źródło: [WWW,2]

Dysponując statystykami (Rysunek 1) z raportu rocznego Policji, można zauważyć wzrost liczby zarejestrowanych pojazdów rok do roku.

Rodzaje wypadków drogowych

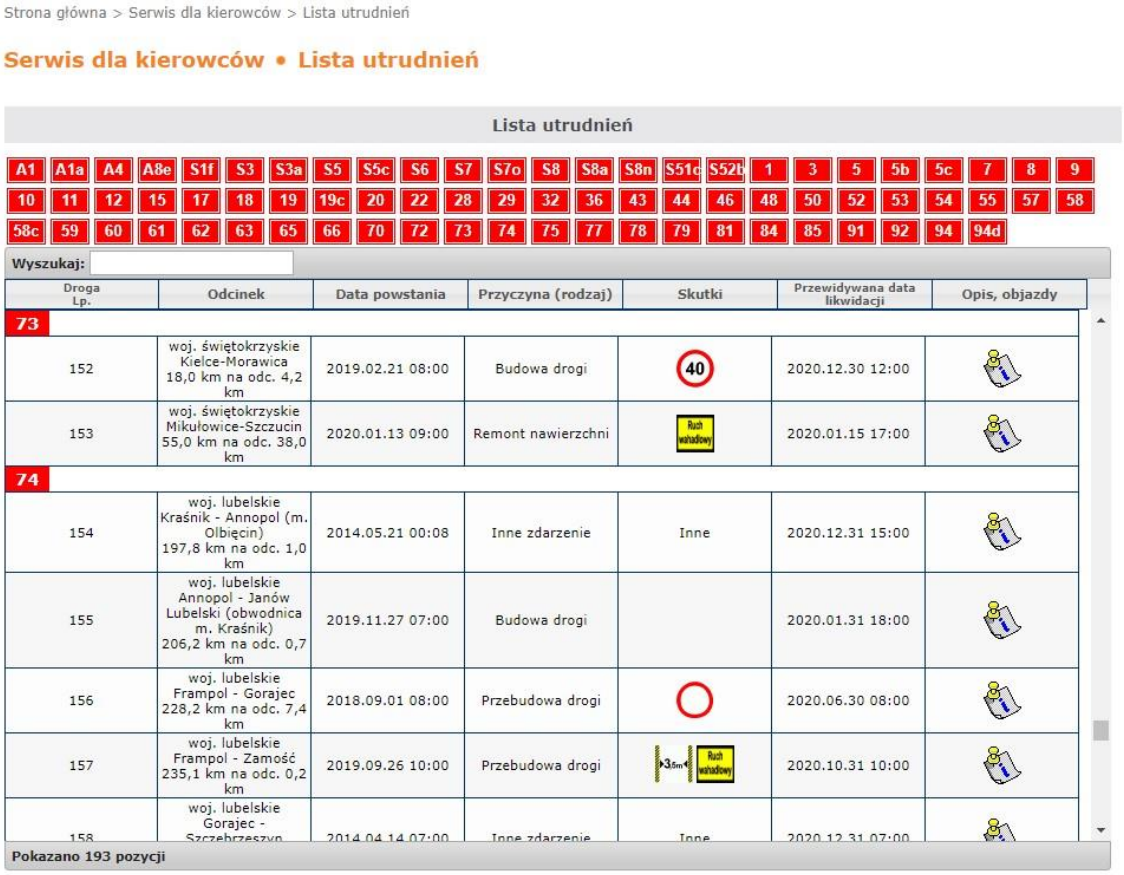
Rodzaj zdarzenia		Wypadki		Zabici		Ranni	
		Ogółem	%	Ogółem	%	Ogółem	%
Zderzenie się pojazdów w ruchu	boczne	9 949	31,4	552	19,3	12 024	32,2
	czołowe	3 104	9,8	510	17,8	4 676	12,5
	tylne	3 988	12,6	207	7,2	5 177	13,9
Najechanie na	pieszego	7 242	22,9	792	27,7	6 800	18,2
	drzewo	1 588	5,0	411	14,4	1 822	4,9
	na słup, znak	436	1,4	48	1,7	507	1,4
	unieruchomiony pojazd	330	1,0	24	0,8	383	1,0
	barierę ochronną	331	1,0	44	1,5	402	1,1
	zwierzę	182	0,6	13	0,5	214	0,6
	dziurę, wybój	42	0,1	-	-	43	0,1
	zaporę kolejową	3	-	-	-	4	-
Wywrócenie się pojazdu		2 396	7,6	183	6,4	2 874	7,7
Zdarzenie z pasażerem		627	2,0	7	0,2	737	2,0
Inne rodzaje		1 456	4,6	71	2,5	1 696	4,6
O g ó ł e m		31 674	100,0	2 862	100,0	37 359	100,0

Rysunek 2 Rodzaje wypadków drogowych

Źródło: [WWW,2]

Tabela przedstawiona na Rysunku 2 pokazuje podział wypadków, które zakwalifikowano według raportu do kategorii „zderzenie się pojazdów w ruchu”.

Kolejnym źródłem informacji na temat rodzajów zdarzeń drogowych jest lista aktualnych utrudnień na drogach ze strony Generalnej Dyrekcji Dróg Krajowych i Autostrad [WWW,3].



Rysunek 3 Lista utrudnień ze strony internetowej Generalnej Dyrekcji Dróg Krajowych i Autostrad

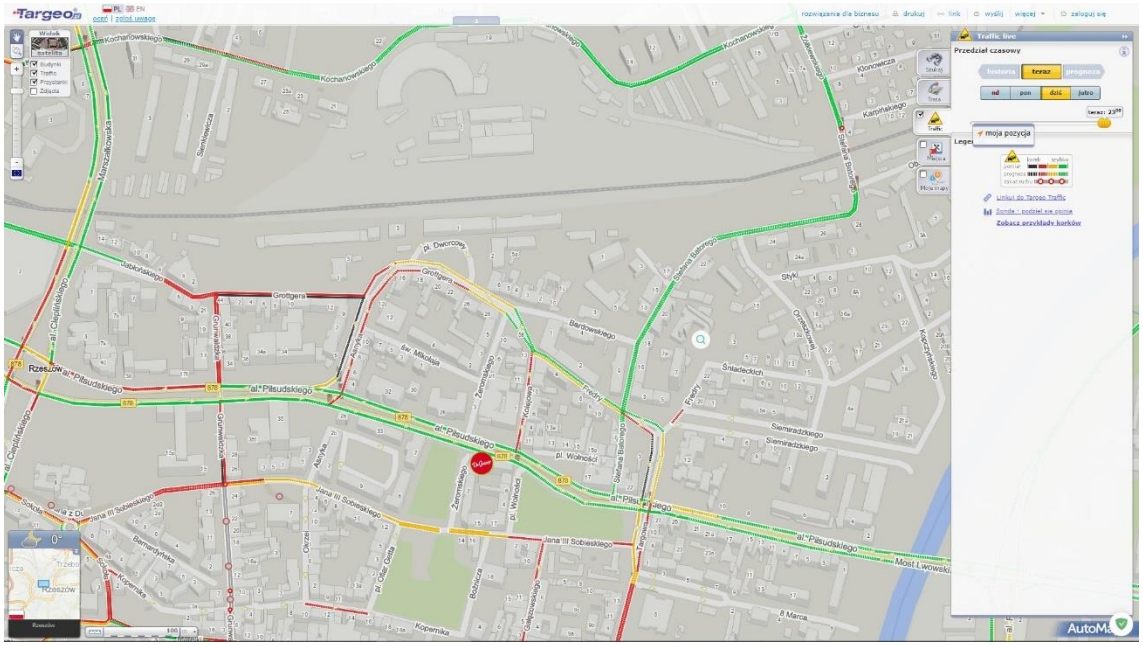
Źródło: [WWW,3]

Na Rysunku 3 narzędzie do wyszukiwania utrudnień na wyznaczonej drodze. Podane są szczegółowe dane o miejscu wystąpienia zdarzenia drogowego, okres obowiązywania danego ograniczenia i utrudnienia z tym związane np. ograniczenie prędkości.

Przykłady utrudnień wymienionych na stronie Generalnej Dyrekcji Dróg Krajowych i Autostrad:

- Przebudowa drogi/mostu
- Remont nawierzchni
- Czynne osuwisko
- Wycinka drzew
- Remont wiaduktu

Ostatnim przytoczonym przeze mnie źródłem dotyczącego informacji o aktualnych zdarzeniach drogowych jest serwis internetowy Targeo [WWW,4].



Rysunek 4 Serwis internetowy Targeo.pl

Źródło: [WWW,4]

Na Rysunku 4 przedstawiającym serwis internetowy Targeo znajduję się narzędzie do podglądu natężenia aktualnego ruchu drogowego, nieprzejezdnych odcinkach dróg w wybranej okolicy. Serwis Targeo oferuje również prognozę, jak i archiwalne dane na ten temat.

W swojej pracy wyróżniłem cztery typy zdarzeń drogowych, które użytkownik będzie mógł wybrać podczas tworzenia zgłoszenia:

- Wypadek, kolizja drogowa – zdarzenie drogowe, w którym brał udział przynajmniej jeden pojazd. Wydarzenie to powoduje utrudnienia w ruchu drogowym.
- Fotoradar – urządzenie stacjonarne rejestrujące pomiar prędkości pojazdu.
- Roboty drogowe – zatrzymanie lub spowolnienie ruchu drogowego przez przebudowa lub remont jezdni.
- Kontrola drogowa – patrol Policji lub Inspekcji Transportu Drogowego.

3.2. Istniejące dostępne rozwiązania na rynku

Przedstawię najpopularniejsze darmowe aplikacje, których główną funkcjonalnością jest nawigacja samochodowa rozszerzona o możliwość wcześniejszego informowania kierowcy o wystąpieniu danego zdarzenia drogowego. Korzystając z danej aplikacji w trasie, trafność zgłoszeń zależy od bazy użytkowników (ich równomiernego

rozmieszczenia), weryfikacji trafności zgłoszeń oraz preferencji kierowcy dotyczących wyboru nawigacji lub rezygnacji z jej korzystania.

3.2.1. Yanosik

Rysunek 5 Logo producenta aplikacji Yanosik

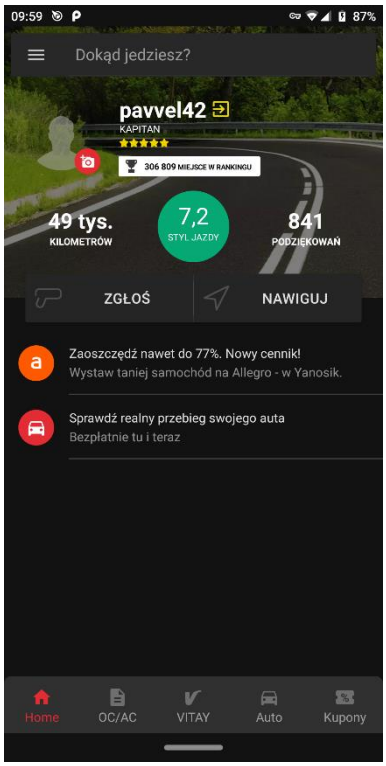
Źródło: https://supermechanik.pl/blog/wp-content/uploads/2014/06/Yanosik_Logo_Png.jpg

Nawigacja samochodowa, a dodatkowo aplikacja jest wyposażona w funkcję wideorejestratora oraz możliwość powiadomienia o aktualnych zdarzeniach np. kontrolach policji, fotoradarach, nieoznakowanych radiowozach, wypadkach, zagrożeniach i remontach drogowych, a także o kontrolach Inspekcji Transportu Drogowego. Rozwojem aplikacji zajmuje się Neptis S.A. Pierwsze wydanie stabilnej wersji nastąpiło w listopadzie 2010 roku.

Według informacji ze strony producenta:

- z aplikacji korzysta ponad 1,5 mln kierowców miesięcznie,
- średnio, co 2 sekundy pojawia się nowe zgłoszenie,
- 150 tysięcy użytkowników korzysta online w godzinach szczytu,
- zarejestrowano 12 milionów pobrań aplikacji.

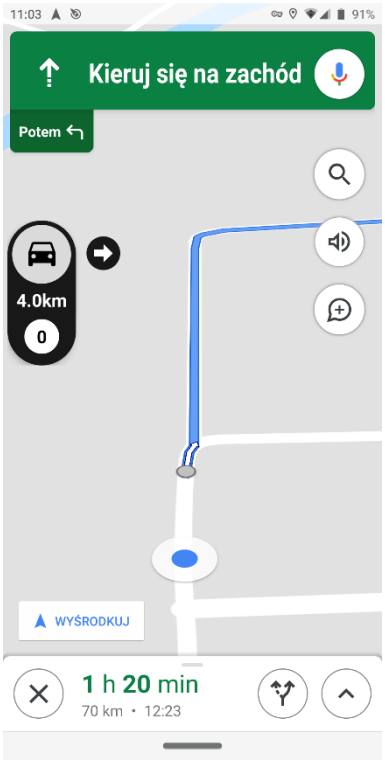
Jest ona rozwijana na systemie Android oraz iOS, oprócz tego producent wydał dedykowane urządzenia dla kierowców. Yanosik korzysta z map OpenStreetMap, nawigacja w tym momencie działa tylko w Polsce. Za granicą dostępna jest jedynie funkcja ostrzegania o zdarzeniu. Aplikacja dodatkowo posiada bazę fotoradarów stacjonarnych w Europie.



Rysunek 6 Widok główny aplikacji Yanosik

Źródło: Opracowanie własne

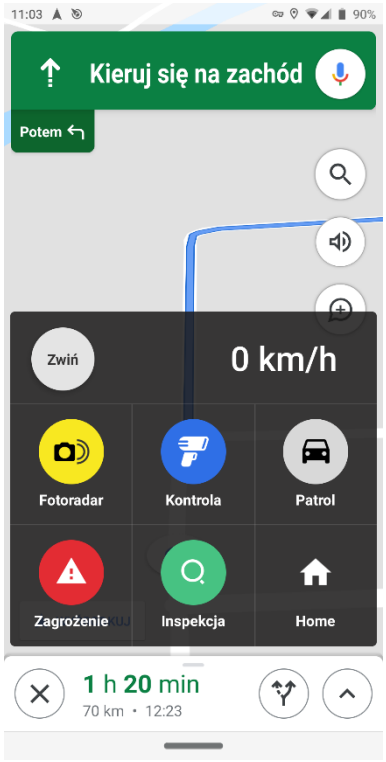
Zrzut ekranu na Rysunku 6 przedstawia ekran główny aplikacji, możliwość bezpośredniego zgłoszenia zdarzenia drogowego oraz możliwość wyboru nawigacji do celu. Widać również ocenę oraz statystyki kierowcy w aplikacji.



Rysunek 7 Zminimalizowany widok działającej w tle aplikacji Yanosik

Źródło: Opracowanie własne

Rysunek 7 pokazuje, że aplikacja ma możliwość działania w tle uzupełniając pracę innej uruchomionej w tym czasie nawigacji, wtedy działa tylko możliwość zgłoszenia i odbierania informacji z bazy danych Yanosik o aktualnych zdarzeniach drogowych.



Rysunek 8 Rozszerzony widok działającej aplikacji Yanosik w tle

Źródło: Opracowanie własne

Na zrzucie ekranu przedstawionym na Rysunku 6 widać rozsunięte okno aplikacji Yanosik działającej w tle. Wybór jednej z opcji powoduje zgłoszenie danego zdarzenia drogowego.

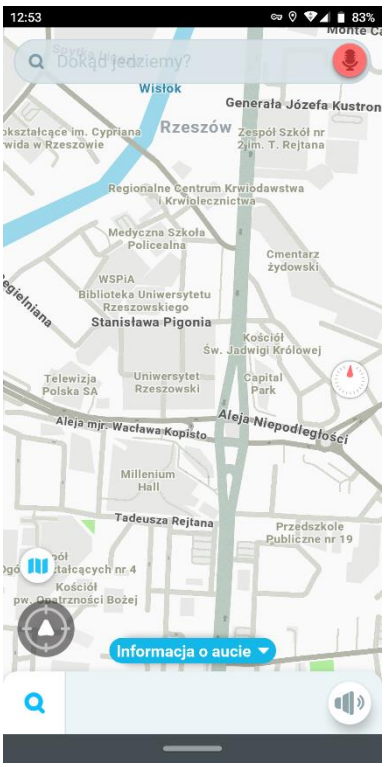
3.2.2. Waze

Rysunek 9 Logo aplikacji Waze

Źródło: https://logos-download.com/wp-content/uploads/2019/06/Waze_Logo_full-700x238.png

"Waze" to nawigacja samochodowa z naciskiem na informowanie i omijanie korków poprzez wybór alternatywnej trasy w trakcie nawigacji. W 2013 roku została kupiona przez Google za ok. 1,3 mld dolarów. Według informacji na stronie producenta jej globalną społeczność tworzy ponad 115 milionów użytkowników. Popularność i zarazem baza użytkowników, zależy od regionu, w którym używany jest Waze, co

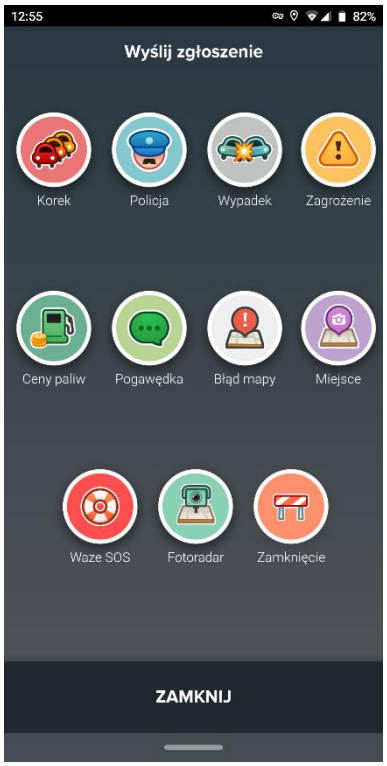
wpływa na ilość i wiarygodność zgłoszonych zdarzeń drogowych. Aplikacja jest dostępna na urządzeniach z systemem Android oraz iOS.



Rysunek 10 Widok główny aplikacji Waze

Źródło: Opracowanie własne

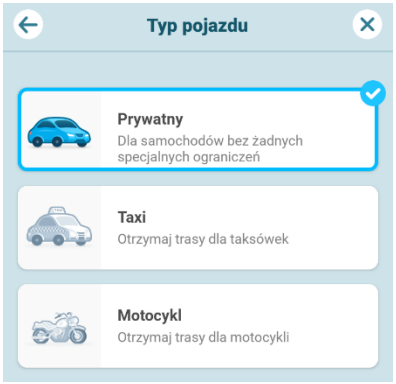
Zrzut ekranu (Rysunek 10) przedstawia widok główny aplikacji Waze, gdzie umożliwia się użytkownikowi na uruchamianie nawigacji, wybór typu pojazdu, konfiguracje widoku mapy.



Rysunek 11 Rodzaje zgłoszeń drogowych w aplikacji Waze

Źródło: Opracowanie własne

Rodzaje zgłoszeń drogowych (Rysunek 11) oferowanych przez aplikację Waze, zawierają opcje dotyczące zmian na mapie tj. np. zamknięcie danego odcinka trasy.



Rysunek 12 Zmiana typu pojazdu w aplikacji Waze

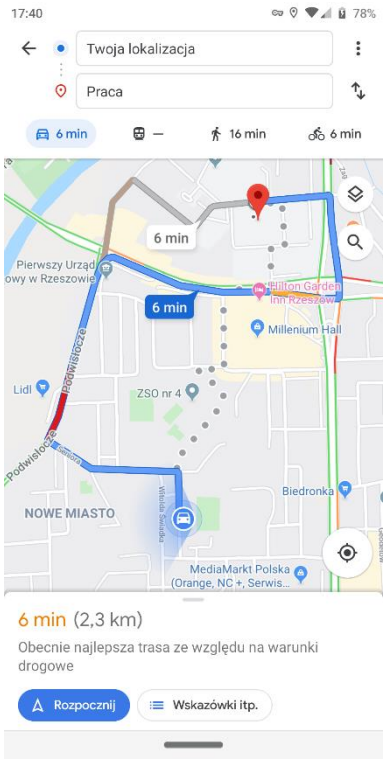
Źródło: Opracowanie własne

Na Rysunku 12 widać, jak aplikacja Waze pozwala na personalizację środka transportu, dzięki czemu potrafi lepiej dostosować trasę dla użytkownika.

3.2.3. Google Maps

Ta z kolei popularna aplikacja jest preinstalowana na większość smartfonów, razem z pozostałymi usługami Google. Dostarcza użytkownikom bieżące informacje o ruchu

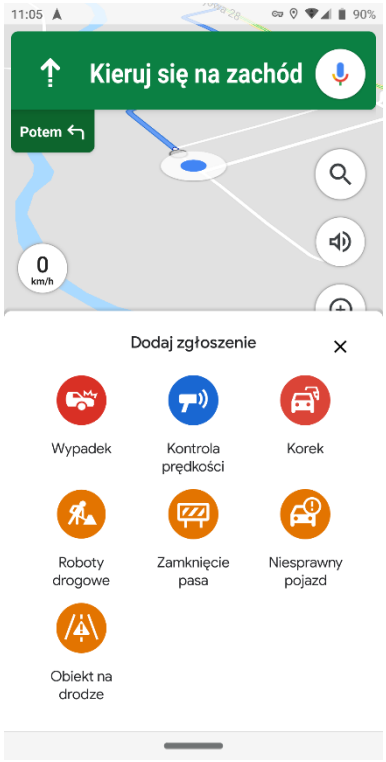
drogowym oraz potrafi dynamicznie zaproponować zmianę trasy podczas nawigacji. Jest to rozbudowany produkt oferujący nie tylko nawigację samochodową, ale także pieszą i rowerową.



Rysunek 13 Wyznaczanie trasy w aplikacji Google Maps

Źródło: Opracowanie własne

Według danych ze sklepu Google Play samych instalacji tej aplikacji jest ponad 5 miliardów. Dodatkowo jest oferowana na system iOS. Posiada dużą bazę użytkowników w porównaniu do konkurencyjnych rozwiązań, co pozytywnie wpływa na ilość oraz jakość zgłoszeń o zdarzeniach drogowych.



Rysunek 14 Rodzaje zgłoszeń drogowych w aplikacji Google Maps

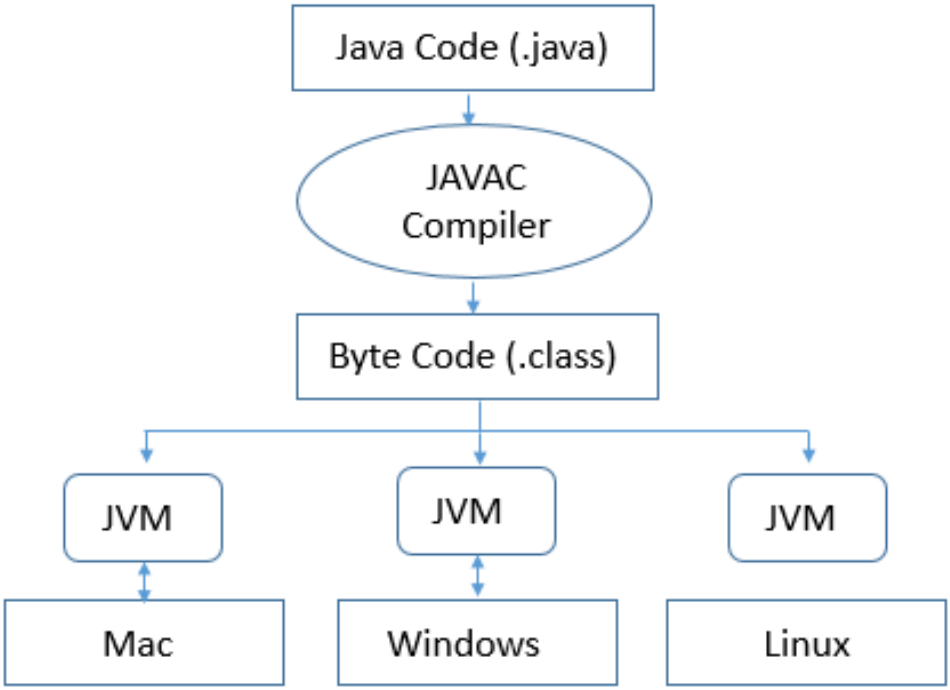
Źródło: Opracowanie własne

Zrzut ekranu na Rysunku 14 przedstawia rodzaje zgłoszeń oferowanych przez Google Maps. Możliwość ich zgłoszenia pojawia się w momencie, kiedy użytkownik wybierze cel podróży i uruchomi nawigację.

4. Narzędzia i technologie

4.1. Język programowania Java

Java - obiektowy język programowania z silną kontrolą typów, kompilowany do kodu bajtowego (Rysunek 15), który może być wykonywany przez maszynę wirtualną Java (JVM), co umożliwia uruchomienie programu na każdej platformie obsługującej Javę [WWW,5].



Rysunek 15 Schemat działania Java Virtual Machine

Źródło: http://net-informations.com/java/intro/img/java-virtual-machine.png

Język Java umożliwia pakowanie komponentów napisanych w języku Java w interfejsy API, z których inni mogą korzystać w swoich aplikacjach, umożliwiając dostęp do lokalnego systemu plików, sieci i wielu innych rzeczy [WWW,6]. Za zarządzanie pamięcią odpowiada mechanizm Garbage Collection [Eckel, 2006]. To program, który działa na JVM, pozbywając się niewykorzystanych obiektów, nie używanych przez program [WWW,7].

```
private void updateUI(FirebaseUser user) {
    if (user != null) {
        finish();
        startActivity(new Intent( packageContext: SignIn.this, MainActivity.class));
    } else {
        progressBar.setVisibility(View.INVISIBLE);
        Toast.makeText( context: SignIn.this,  text: "Authentication failed", Toast.LENGTH_SHORT).show();
    }
}
```

Rysunek 16 Fragment kodu języka programowania Java

Źródło: Opracowanie własne

Przedstawiony na Rysunku 16 fragment kodu pokazuje metodę *updateUI*, której zadaniem jest zamknięcie aktualnej oraz utworzenie nowej aktywności, gdy obiekt *user* nie jest typu *null*. W innym przypadku *progressBar* staje się „niewidoczny", a użytkownikowi wyświetlany jest stosowny komunikat.

4.2. Język programowania TypeScript

TypeScript został opracowany przez Microsoft i jest nadzbiorem JavaScript oraz kompiluje się do niego, konwertując kod TypeScript do czystego, czytelnego, opartego na standardach JavaScript kodu. Jest otwartym językiem programowania, który dodaje typy do JavaScript. TypeScript działa na dowolnej platformie, urządzeniu i przeglądarce. Kod źródłowy i projekt TypeScript są dostępne na platformie Github [WWW,8].

```
48  function distance(x1 : number,y1 : number, x2 : number, y2 : number) {
49      let dis;
50      dis = Math.sqrt(Math.pow((x2 - x1), 2) + Math.pow((Math.cos((x1 * Math.PI) / 180) * (y2 - y1)), 2)) * (40075.704 / 360);
51      console.log("x1 "+x1+" y1 "+y1+" x2 "+x2+" y2 "+y2+" odległość między pkt. wynosi = "+dis*1000+" m")
52      return dis*1000;
53  }
54
55
56  function deleteOldReport(timeRep:number){
57      const time = Date.now();
58      const diff = time - timeRep
59      console.log("Czas serwera: "+time+" Czas raportu: "+timeRep+ " Różnica: "+diff)
60      return diff
61  }
```

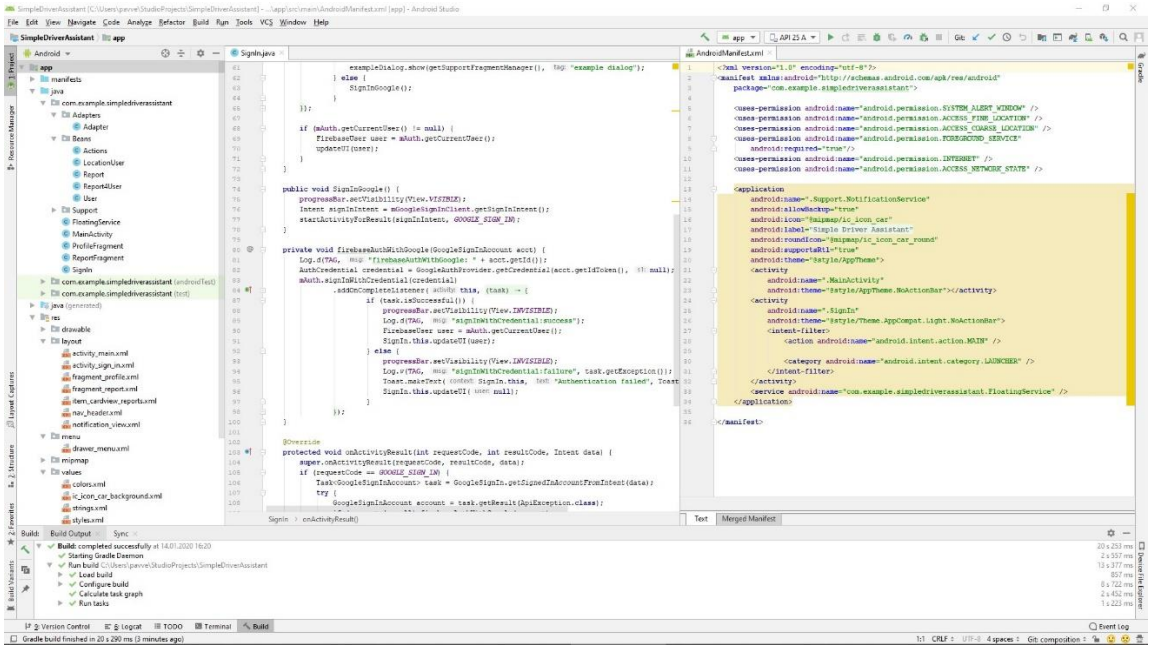
Rysunek 17 Fragment kodu języka programowania TypeScript

Źródło: Opracowanie własne

Przedstawiony na Rysunku 17 fragment kodu pokazuje metodę, którą na podstawie współrzędnych, zostanie zwrócona odległość w metrach pomiędzy dwoma punktami oraz metodę zwracającą różnicę czasu pomiędzy aktualnym czasem serwera, a czasem utworzenia raportu.

4.3. Środowisko programowania Android Studio

Android Studio [Stasiewicz, 2015] to oficjalne zintegrowane środowisko programistyczne (IDE) dla systemu operacyjnego Android, działa w oparciu o IntelliJ IDEA, zaprojektowany specjalnie dla rozwoju Android. Jest dostępny na systemach operacyjnych Windows, macOS i Linux. Posiada Android Virtual Device (Emulator) do uruchamiania i debugowania aplikacji w Android Studio oraz narzędzia do śledzenia wydajności, użyteczności i kompatybilności tworzonej aplikacji [WWW,9].



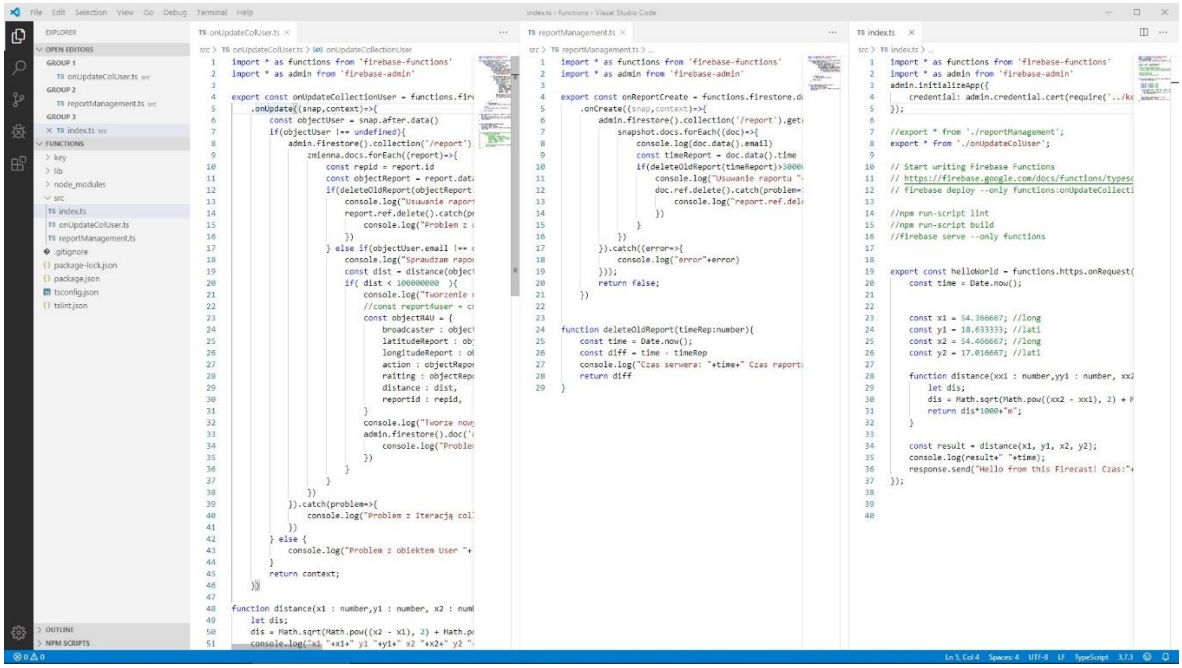
Rysunek 18 Widok środowiska programowania Android Studio

Źródło: Opracowanie własne

Rysunek 18 przedstawia konfigurowalny interfejs programu Android Studio. Po prawej stronie znajdują się pliki projektu, zaś w środkowej części, można dostosować według własnych preferencji rozmiar okna, w którym znajduję się kod danego pliku projektu.

4.4. Środowisko programowania Visual Studio Code

Visual Studio Code jest darmowym i open source'owym edytorem kodu źródłowego opracowanym przez Microsoft dla platform Windows, Linux i macOS. Obejmuje obsługę debugowania, wbudowaną kontrolę wersji Git, podświetlanie składni, inteligentne uzupełnianie i refaktoryzację kodu. Jest wysoce konfigurowalny, pozwalając użytkownikom instalować rozszerzenia, które dodają nowe funkcjonalności [WWW,10].



Rysunek 19 Widok środowiska programowania Visual Studio Code

Źródło: Opracowanie własne

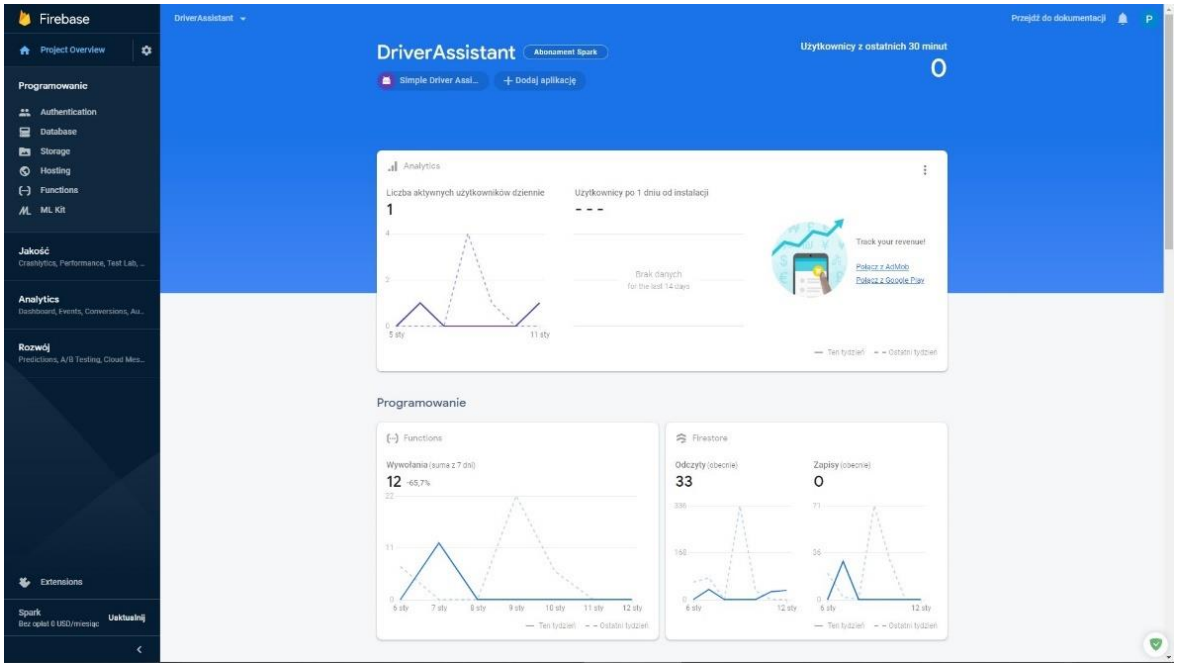
4.5. Usługa baz danych Firebase

Firebase [Smyth, 2017] jest oparty o model usługi przetwarzania w chmurze Backend as a service (BaaS). Zapewnia programistom sposoby łączenia aplikacji internetowych i mobilnych z usługami w chmurze za pośrednictwem interfejsów programowania aplikacji (API) i zestawów narzędzi programistycznych (SDK) [WWW,11]. Komunikacja pomiędzy aplikacją oraz Firebase odbywa się za pomocą WebSocket. Pozwala to na synchronizację danych w czasie rzeczywistym.

Firebase Authentication to system uwierzytelniania użytkownika. Obsługuje uwierzytelnianie przy użyciu numerów telefonów, kont Google, Facebooka i Twittera oraz innych dostawców, wykorzystując standardy pozwalające na budowanie bezpiecznych mechanizmów autoryzacyjnych, takich jak OAuth 2.0 i OpenID Connect [WWW,12].

Cloud Firestore to elastyczna, skalowalna baza danych NoSQL. Utrzymuje synchronizację danych między aplikacjami klienckimi za pośrednictwem nasłuchiwania w czasie rzeczywistym oraz oferuje wsparcie offline dla urządzeń mobilnych. Cloud Firestore oferuje również płynną integrację z innymi produktami Firebase i Google Cloud Platform, w tym funkcjami chmury [WWW,13].

Cloud Functions for Firebase umożliwiają automatyczne uruchamianie kodu w odpowiedzi na wywoływane zdarzenia w bazie danych. Funkcje w chmurze umożliwiają uruchamianie operacji na bazie danych [WWW,14].



Rysunek 20 Widok projektu w usłudze Firebase

Źródło: Opracowanie własne

Na Rysunku 20 widać interfejs usługi Firebase. W centralnej części wyświetlane są aktualne statystyki dotyczące projektu np. ilość wywołań funkcji, liczbę aktywnych użytkowników. Chcąc przełączać się pomiędzy usługami Firebase, wystarczy wybrać interesującą nas zakładkę po prawej stronie interfejsu. Niżej widoczny jest również aktualny plan abonamentu platformy Firebase.

5. Mobilna aplikacja do informowania oraz zgłaszania zdarzeń drogowych

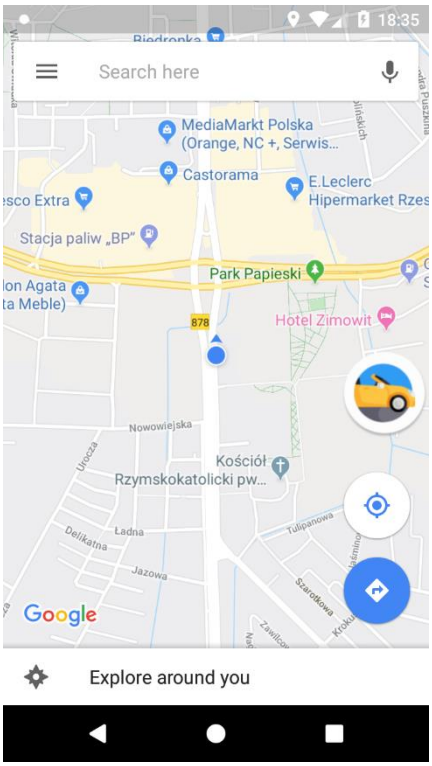
5.1. Charakterystyka aplikacji

Głównym zadaniem aplikacji jest wcześniejsze ostrzeżenie użytkownika w czasie podróży przed ewentualnym wybranym zdarzeniem drogowym. Dzięki temu, kierowca może zwiększyć czujność podczas jazdy, co również zwiększa jego bezpieczeństwo, ponieważ będzie mógł z wyprzedzeniem zareagować na wybrane zdarzenie drogowe.

Aplikacja jest przeznaczona na platformę Android API level 23 – 25 co odpowiada wersji numerycznej systemu 6.0 – 7.1, korzysta też z przyznanych przez użytkownika uprawnień do odczytywania jego aktualnej pozycji GPS. Użytkownik musi zapewnić połączenie z Internetem w trakcie korzystania z aplikacji do połączenia z bazą danych. Ostatnim wymaganym uprawnieniem jest zaznaczenie opcji: „Pozwól na wyświetlanie nad innymi aplikacjami". Umożliwi to utworzenie zminimalizowanego widoku aplikacji, która działa „w tle" (użytkownik dzięki temu, może uruchomić dowolną aplikację do nawigacji, obie działające jednocześnie). Widok ten zawiera tylko najpotrzebniejsze elementy aplikacji, czyli możliwość zgłoszenia zdarzenia drogowego oraz jego odbioru.

Ważnym elementem aplikacji jest usługa „Floating Bubble" w formie pływającego widoku, który można rozszerzyć, bądź też zwinąć. Jest on zawsze „na wierzchu" - wyświetlany nad innymi aplikacjami działającymi w tle. Dzięki temu, może uzupełniać pracę nawigacji (Rysunek 21).

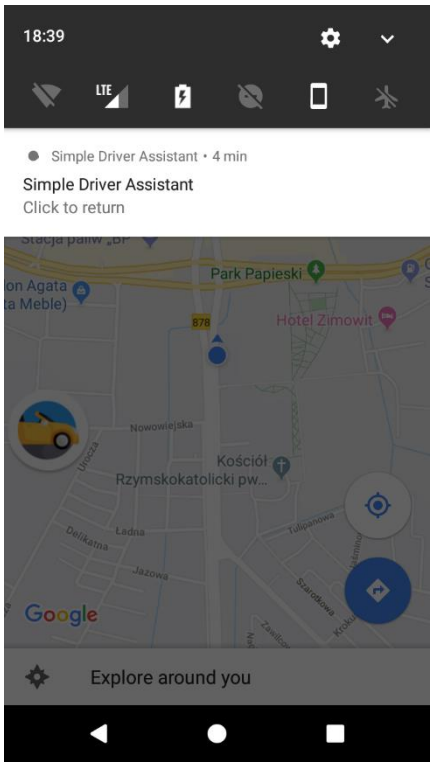
Napisana przeze mnie aplikacja otrzymała nazwę *Simple Driver Assistant* (z ang. prosty asystent kierowcy).



Rysunek 21 Zminimalizowany widok działającej w tle aplikacji

Źródło: Opracowanie własne

W każdej chwili działania usługi można powrócić do okna aplikacji (Rysunek 22), ponieważ na belce powiadomień znajduje się powiadomienie, a klikając w nie wyświetli się nam widok aktywności aplikacji.



Rysunek 22 Powiadomienie aplikacji Simple Driver Assistant

Źródło: Opracowanie własne

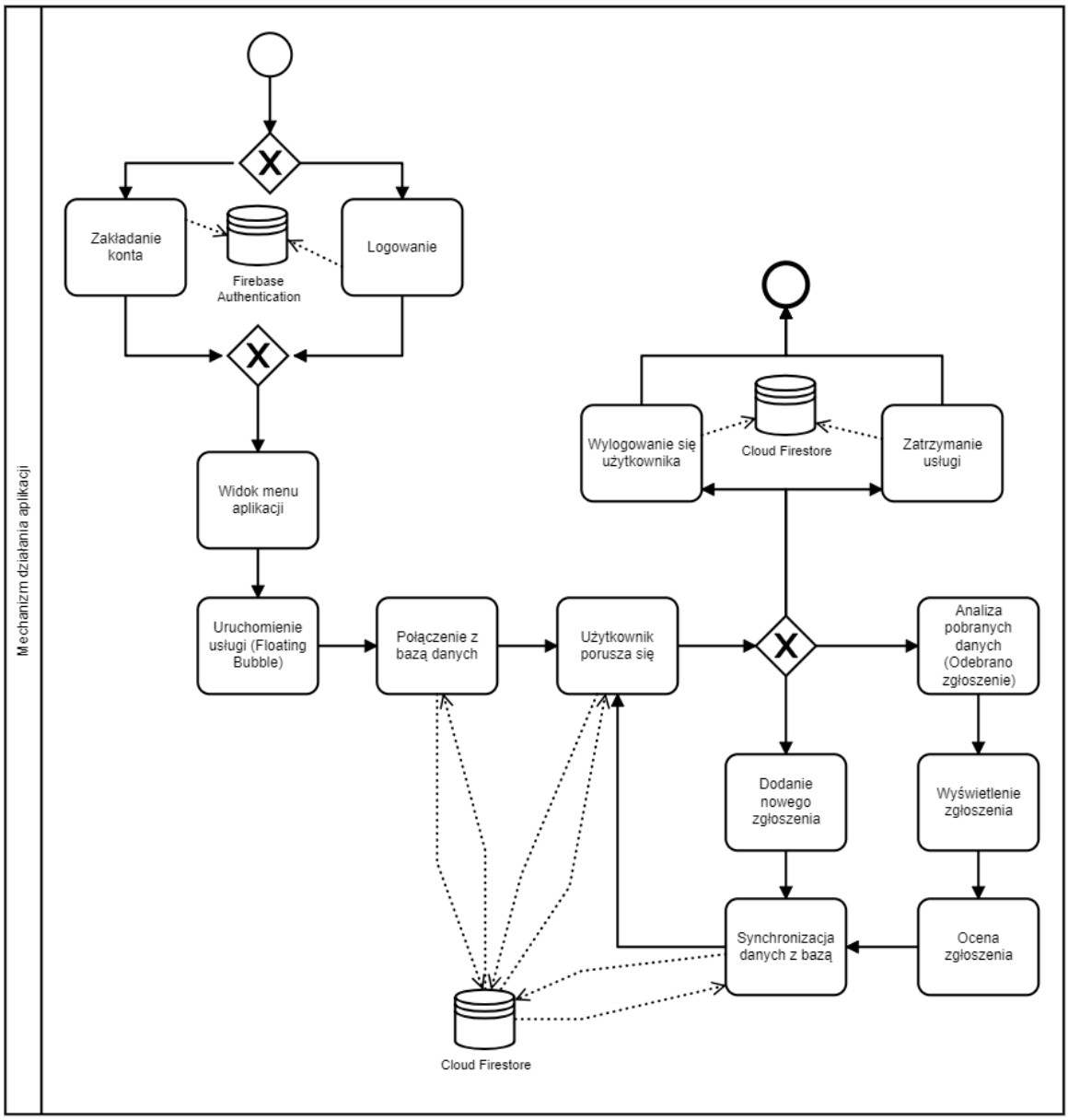
Klikając z kolei w „pływający bąbelek”, rozszerzy się nam widok, w którym znajdują się przyciski, aby można było zgłosić wybrane zdarzenie (Rysunek 23).

Rysunek 23 Rozszerzony widok działającej aplikacji

Źródło: Opracowanie własne

5.2. Mechanizm działania aplikacji

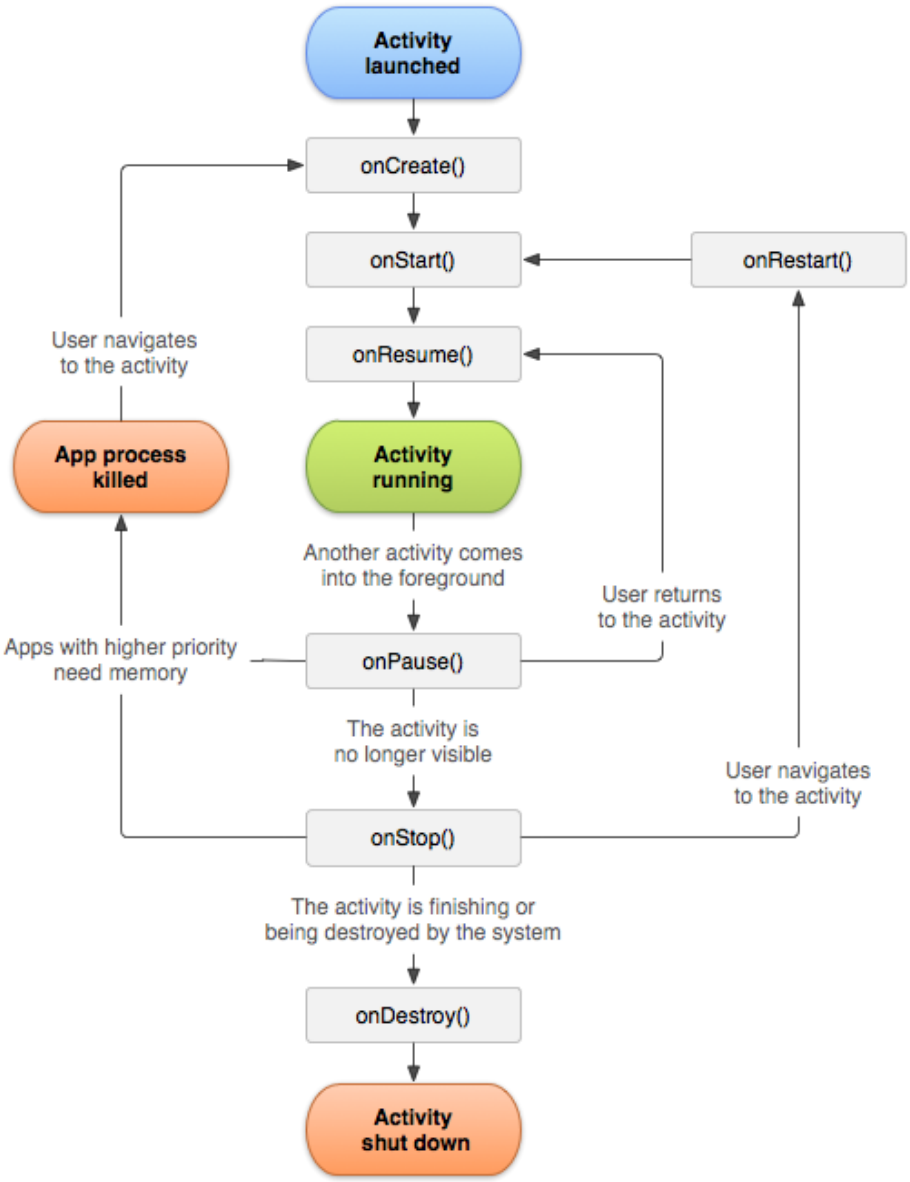
Na poniższym diagramie BPMN znajduje się schemat głównej funkcjonalności aplikacji.



Rysunek 24 Diagram przedstawiający mechanizm działania aplikacji

Źródło: Opracowanie własne

Po zalogowaniu się użytkownikowi pokaże się główny widok aplikacji (Rysunek 26). Do utworzenia interfejsu użytkownika korzystam z aktywności (ang. *Activity*). Komponent systemu Android wykorzystywany do interakcji z użytkownikiem. [WWW,15]



Rysunek 25 Cykl życia aktywności w systemie Android

Źródło: https://developer.android.com/images/activity_lifecycle.png

Główną aktywnością w aplikacji *Simple Driver Assistant* jest klasa *MainActivity*. Rozszerzając *MainActivity* o klasę *Activity*, można zaimplementować metodę *onCreate()*, która zostanie wywołana jako pierwsza, gdy aktywność jest tworzona (Rysunek 25). W ciele metody wywołane są potrzebne elementy do poprawnego działania aplikacji tj. *setContentView(R.layout.activity_main)* – odwołanie do pliku *XML* zawierającego układu widoku. Każda z metod obsługuje, konkretny stan, w których znajduje się aktywność. W klasie *MainActivity*, można przysłonić kolejne metody pokazane na Rysunku 25, rozszerzając możliwości aplikacji:

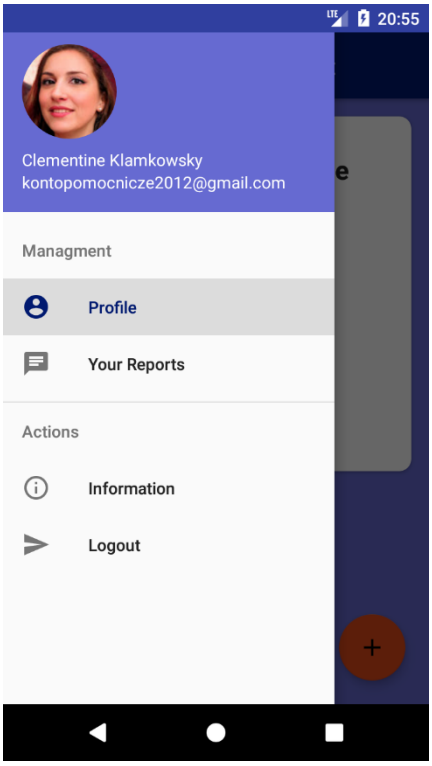
- onStart() – wywołanie następuje, gdy aktywność jest „zbudowana”, widoczna dla użytkownika,
- onResume() – wywołanie następuje, gdy aktywność jest już wyświetlona i może nastąpić interakcja użytkownika z tym komponentem,
- onPause() – wywołanie następuje, gdy użytkownik wchodzi w interakcje z inna aktywnością,
- onRestart() – wywołanie następuje, gdy następuje powrót do aktywności,
- onStop() – wywołanie następuje, gdy nie jest już widoczna dla użytkownika,
- onDestroy() – wywołanie następuje, przed tym jak aktywność zostanie zniszczona.

Rysunek 26 Widok główny aplikacji Simple Driver Assistant

Źródło: Opracowanie własne

W punkcie centralnym widoku z Rysunku 26, kierowca ma wgląd do swojego profilu, składającego się z pozytywnych (*Likes*) i negatywnych (*Dislikes*) ocen innych użytkowników, na temat jego wcześniejszych zgłoszeń drogowych. Ponad to, może obserwować swoją ocenę (*Rating*) dotyczącą oceny przydatności zgłoszeń.

Po lewej stronie zrzuty ekranu z Rysunku 26, znajduje się menu. Klikając w symbol „trzech poziomych pasków” w lewym górnym rogu, widok menu pokaże się użytkownikowi (Rysunek 27).



Rysunek 27 Widok menu aplikacji

Źródło: Opracowanie własne

Rysunek 27 przedstawia widok menu, w którym znajdują się informację o zalogowanym użytkowniku. Zdjęcie profilowe pobierane jest z profilu konta Google użytkownika, przez które się zalogował. Aktualnie wybraną zakładką w menu jest widok główny aplikacji, czyli profil użytkownika widoczny na Rysunku 26.

Wybór opcji „Information” w menu spowoduje wyświetlenie komunikatu o autorze aplikacji.

Wylogowanie się z aplikacji i konta Google nastąpi po wybraniu opcji „Logout” w menu aplikacji. Użytkownik powróci do widoku logowania (Rysunek 37).

Wybór zakładki „Your Reports”, został przedstawiony na Rysunku 28 oraz opisany poniżej.

Rysunek 28 Widok menu raportów

Źródło: Opracowanie własne

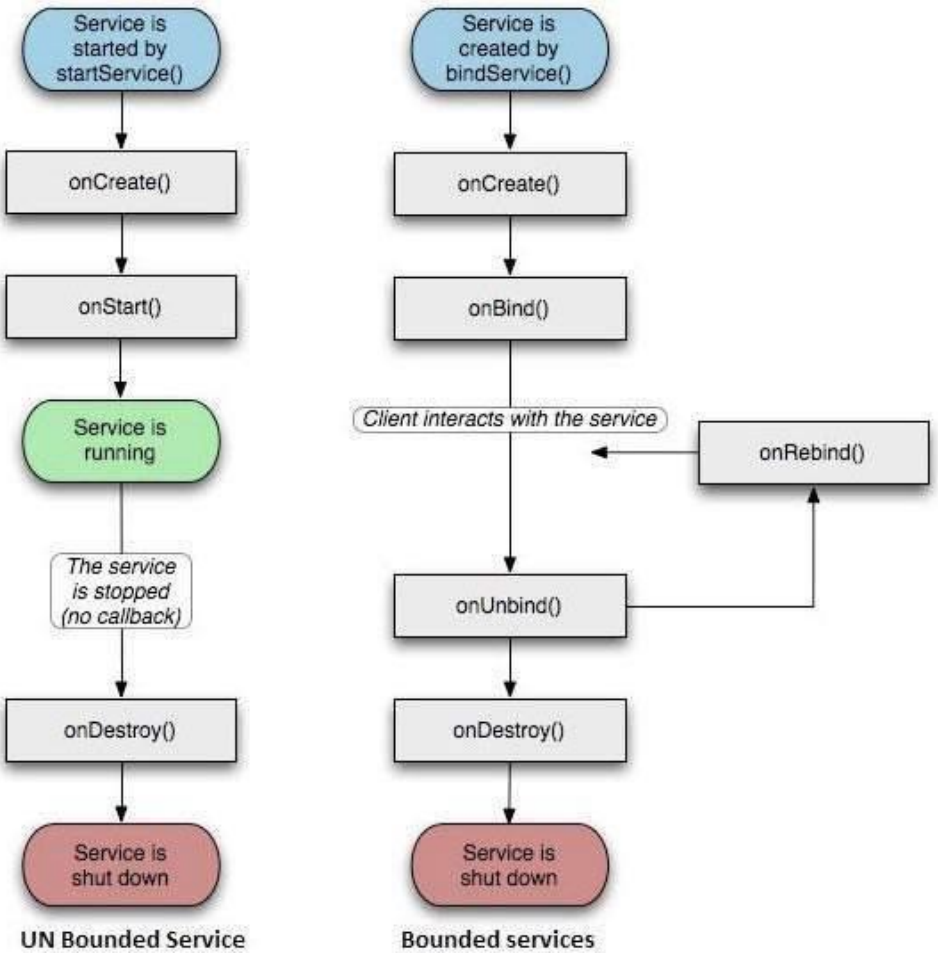
Rysunek 28 przedstawia zgłoszone raporty przez użytkownika. Kierowca ma wgląd do rodzaju i współrzędnych wysłanego zdarzenia drogowego oraz kiedy został zarejestrowany (ang. *Timestamp)*. Ważność zgłoszonego raportu ustawiono na 5 minut, ze względu na dynamicznie zmieniające się wydarzenia w trasie.

Po poprawnej konfiguracji aplikacji (udzielenie uprawnień), użytkownik uruchomi usługę klikając przycisk z symbolem „+”, znajdujący się w prawym dolnym rogu widoku aplikacji, przedstawionym na Rysunku 28.

W systemie Android, można wyróżnić dwa typy usług:

- Unbounded Service
- Bounded Service

Schemat tych usług przedstawiono na Rysunku 29.



Rysunek 29 Schemat przedstawiający dwa typy usług w systemie Android

Źródło: https://www.tutorialspoint.com/android/images/services.jpg

Na potrzeby aplikacji wykorzystano tworzenie usługi typu „Unbounded Service" za pomocą metody *startService()* (Rysunek 29), ponieważ uruchomiona usługa działa w tle. W przypadku, gdy komponent, który go uruchomił zostanie zniszczony, nie wpłynie to na pracę utworzonej usługi.

```
public void startService() {
    Intent serviceIntent = new Intent( packageContext: this, FloatingService.class);
    serviceIntent.putExtra( name: "inputExtra", getString(R.string.click_to_return));
    startService(serviceIntent);
    userOnline( state: true);
    floatingActionButton.setVisibility(View.INVISIBLE);
}
```

Rysunek 30 Fragment kodu pokazujący uruchomienie usługi

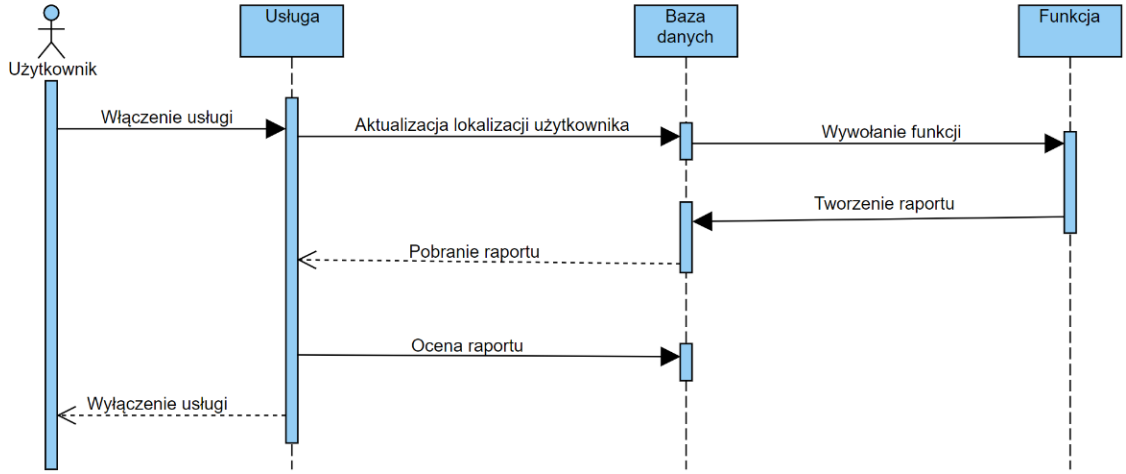
Źródło: Opracowanie własne

Po uruchomieniu użytkownik jest w stanie dodać nowe zgłoszenie – raport (Rysunek 23). Jest ono dodawane do kolekcji *Report* w bazie danych.

W skład tego zgłoszenia wchodzą następujące dane:

- Rodzaj zgłoszenia
- E-mail oraz ocena (*rating*) osoby tworzącej zgłoszenie
- Współrzędne pobrane w chwili, kiedy użytkownik zgłosił wybrane zdarzenie
- Czas utworzenia zdarzenia (ang. *Timestamp*)

Następnie wywoływana jest funkcja w usłudze Firebase, nasłuchująca kolekcje *Report* , gdzie znajdują się wszystkie zgłoszenia.



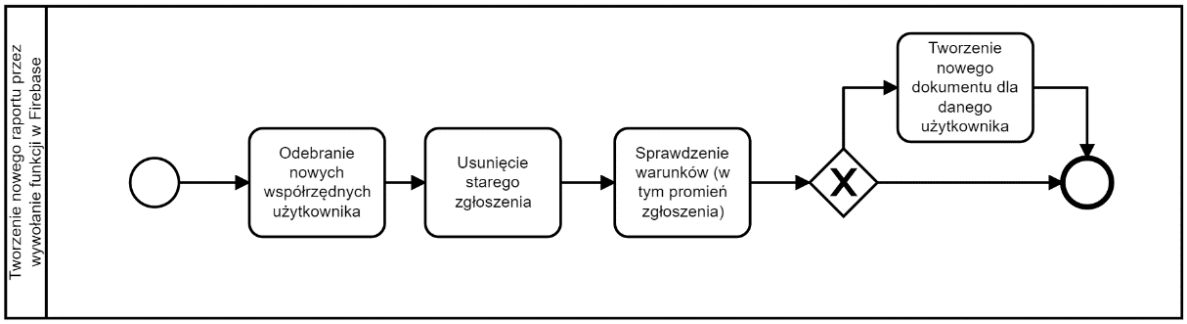
Rysunek 31 Diagram sekwencji tworzenia raportu dla użytkownika

Źródło: Opracowanie własne

Diagram sekwencji na Rysunku 31 pokazuje etapy powstawania raportu. Rozpoczyna się on w momencie uruchomienia usługi, następnie pobierana jest aktualna pozycja użytkownika i przesyłana do bazy danych. Po tym następuje wywołanie funkcji (Rysunek 32). Funkcja ta jest napisana w języku programowania TypeScript. W pierwszej kolejności po wywołaniu funkcji sprawdzany jest warunek, czy dla danego zgłoszenia nie upłynął termin ważności, a jest on ustawiony na 5 minut. Po tym czasie raport jest uważany za przedawniony i podlega usunięciu, z racji tego, że sytuacja na drodze jest dynamiczna, a kierowca musi posiadać aktualne dane. Kolejnym etapem jest sprawdzenie, czy raport danego kierowcy nie należy do niego samego. Jeśli nie należy, to pobierany jest punkt na podstawie współrzędnych, w którym nastąpiło zgłoszenie i liczony jest promień, inaczej obszar, w którym zgłoszenie obowiązuje. Następnie, jeżeli kierowca znajdzie się w promieniu 1000 m od punktu zgłoszenia raportu (tak, aby miał

czas na przygotowanie się do wybranego zgłoszenia), funkcja utworzy raport dla użytkownika, który znajdzie się w

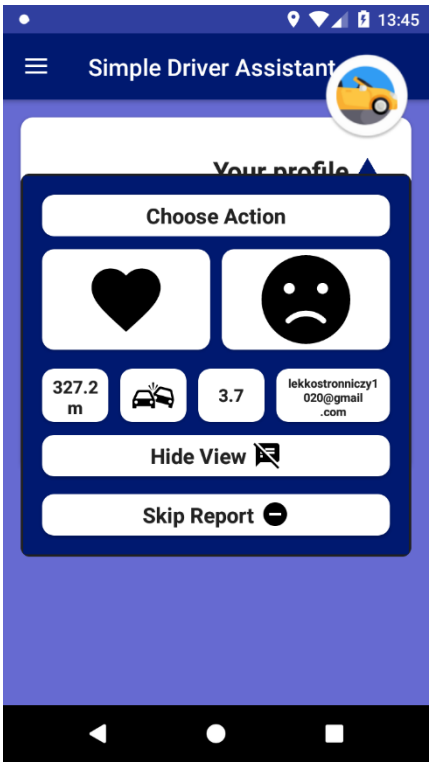
tym obszarze. Jest on zapisywany w kolekcji *report4user*.



Rysunek 32 Diagram ukazujący wywołanie funkcji Firebase tworzącej raport

Źródło: Opracowanie własne

Uruchomiona usługa nasłuchuje kolekcje *report4user* i w przypadku pojawienia się nowego raportu pobiera dane z bazy oraz wyświetla je użytkownikowi (Rysunek 33).



Rysunek 33 Widok przychodzącego zgłoszenia

Źródło: Opracowanie własne

W skład wyświetlonego zgłoszenia na Rysunku 33 wchodzi:

- Adres e-mail użytkownika zgłaszającego akcję

- Aktualna odległość od punktu zgłoszenia zdarzenia
- Rating użytkownika zgłaszającego akcję
- Rodzaj zdarzenia
- Możliwość oceny zgłoszenia lub też jego pominięcia

Uruchomiona usługa nasłuchuje kolekcje *report4user* i w przypadku pojawienia się nowego raportu pobiera dane z bazy oraz wyświetla je użytkownikowi.

Obliczanie odległości pomijając krzywiznę Ziemi

Pomiędzy miejscem położenia użytkownika, a miejscem zdarzenia należy wyliczyć odległość pomiędzy współrzędnymi. W tym celu zastosowano wzór:

[WWW,16]

$$\sqrt{(x_2 - x_1)^2 + \left(\cos(\frac{x_1 * \pi}{180}\right) * (y_2 - y_1))^2} * \frac{40075{,}704}{360} \quad (1)$$

gdzie:

x_1 – oznacza szerokość geograficzną punktu pierwszego,

y_1 – oznacza długość geograficzną punktu pierwszego,

x_2 – oznacza szerokość geograficzną punktu drugiego,

y_2 – oznacza długość geograficzną punktu drugiego.

Wzór (1) jest używany w momencie wywołania funkcji tworzącej zgłoszenie dla użytkownika. W następnym etapie, gdy użytkownik porusza się, dystans między punktami liczę lokalnie metodą *distanceBetween()* z klasy *Location*, aby ograniczyć liczbę zapytań do bazy danych.

Parametry pobierania pozycji użytkownika ustawione są (Rysunek 34), za pomocą metody *requestLocationUpdates()* z klasy *LocationManager* [Griffiths et al., 2016], podając odpowiednie argumenty w tym:

- minimalny odstęp czasu między aktualizacjami lokalizacji, w milisekundach (ustawiono 5000 milisekund)
- minimalna odległość między aktualizacjami lokalizacji, w metrach (ustawiono 20 metrów)

```
private void tracking() {
    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,  minTime: 5000,  minDistance: 20,  listener: this);
}
```

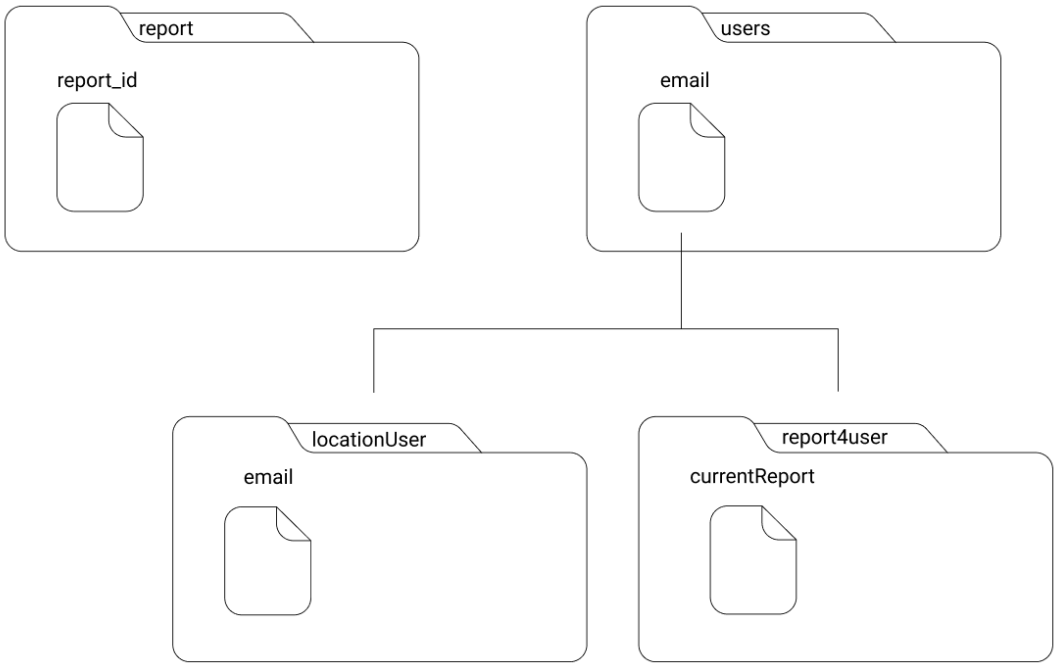
Rysunek 34 Fragment kodu pokazujący parametry śledzenia położenia użytkownika

Źródło: Opracowanie własne

W efekcie otrzymujemy aktualną pozycję użytkownika, jako szerokość i długość geograficzną.

5.3. Baza danych

Poprzez wykorzystanie usługi Cloud Firestore w projekcie aplikacji zastosowano bazę danych NoSQL. Dane przechowywane są w dokumentach znajdujących się w kolekcji. Struktura danych dokumentu zawiera parę klucz-wartość. Dokument może zawierać w sobie kolekcje tworząc podkolekcje [Guy, 2019]. Strukturę bazy danych tworząca aplikację przedstawiono na Rysunku 35.



Rysunek 35 Schemat bazy danych NoSQL

Źródło: Opracowanie własne

Pola dokumentów:

/report/report_id

- action: string
- email: string
- latitude: double
- longitude: double
- rating: double
- time: long

/users/email

- dislike: int
- like: int
- rating: double
- email: string
- name: string
- online: boolean
- uid: string

/users/email/locationUser/email

- email: string
- latitude: double
- longitude: double

/users/email/report4user/currentReport

- broadcaster: string
- latitudeReport: double
- longitudeReport: double
- action: string
- rating: double
- reportid: string
- distance: double

Kolekcja *report* gromadzi wszystkie raporty tworzone przez użytkowników. Dla każdego nowego dokumentu jest tworzony unikalny identyfikator, a ich generowaniem zajmuje się usługa Firebase.

Przykład: "VWlqQkQ8zGJtYNZC9eks”

Dokument w tej kolekcji przechowuje informacje takie jak: rodzaj zdarzenia drogowego(pole *action*), e-mail użytkownika zgłaszającego (pole *email*), współrzędne (pola *latitude* oraz *longitude*) ocenę jego profilu (pole *rating*). Przechowywany jest również czas utworzenia takiego raportu (pole *time*).

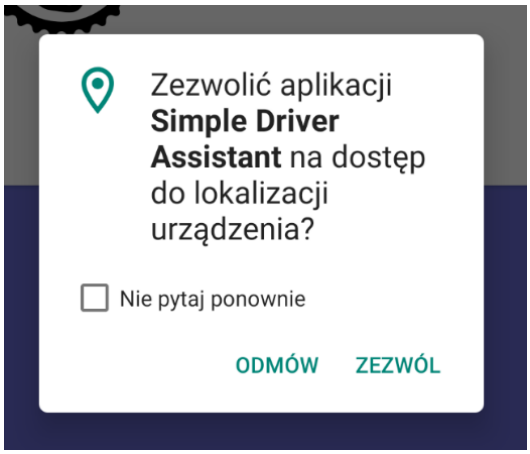
Kolekcja *users* gromadzi dokumenty wszystkich użytkowników oraz ich dane (suma pozytywnych (pole *like*) oraz negatywnych (pole *dislike*) ocen ich zgłoszeń, ocena profilu (pole *rating*), ich identyfikator (pole *uid*) oraz status online(pole *online*)). Ocena profilu, czyli pole dokumentu *rating* jest liczone przez iloraz pola *like* i *dislike*.

Kolekcja *locationUser* zawiera aktualizowany na bieżąco jeden dokument, który zapisuje współrzędne użytkownika (pola *latitude* oraz *longitude*). Jest on odseparowany od kolekcji *users*, aby nie wywoływać bez potrzeby funkcji *onUpdateCollectionLocationUser* w usłudze Firebase.

Kolekcja *report4user* zawiera jeden dokument dla danego użytkownika, który jest tworzony przy wywołaniu funkcji *onUpdateCollectionLocationUser*. Znajdują się w nim informacje o wybranym zdarzeniu (e-mail (pole *broadcaster*) i ocena użytkownika (pole *rating*), który stworzył raport, rodzaj (pole *action*) oraz współrzędne zdarzenia (pola *latitude* oraz *longitude*), identyfikator raportu (pole *reportid*), dystans pomiędzy użytkownikiem, a miejscem zgłoszenia raportu(pole *distance*)). Dokument jest cały czas nadpisywany poprzez wywoływaną funkcję. Aplikacja podczas swojego działania gromadzi każdy raport, przez co może zapisać wiele raportów dla użytkownika. Jest to wykonywane w celu ograniczenia przesyłanych danych przez internet oraz dla oszczędności czasu (śledzone są tylko zmiany w jednym dokumencie, kolekcja nie jest przeglądana).

5.4. Implementacje bibliotek w projekcie

EasyPermissions – upraszcza logikę uzyskiwania i weryfikowania uprawnień systemowych dla aplikacji.



Rysunek 36 Prośba o udzielenie zgody przez użytkownika

Źródło: Opracowanie własne

Aplikacja *Simple Driver Assistant* wymaga tzw. „*Dangerous Permission*" (Rysunek 36). Jest to uprawnienie, które obliguje do wyrażenia zgody przez użytkownika, ponieważ pozwala na dostęp do jego poufnych danych, w tym przypadku lokalizacji. Uprawnienie to musi być przyznane, inaczej usługa „*Floating Bubble*" nie uruchomi się.

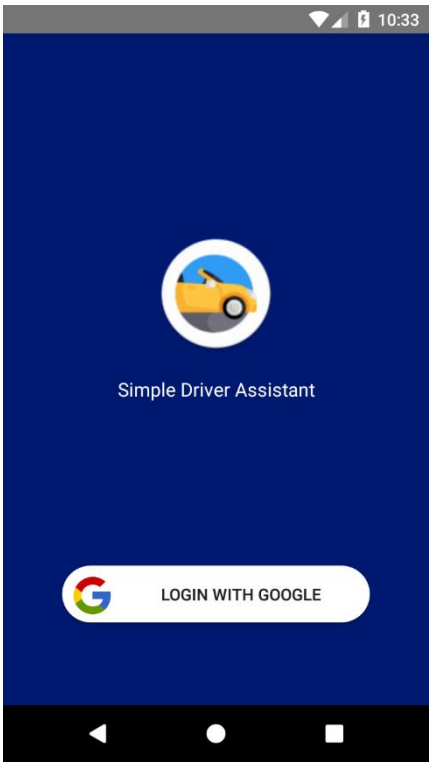
Floating Bubble Library – biblioteka, która odpowiada za tworzenie usługi „pływającego widoku" nad innymi aplikacjami. Pozwala ona na rozbudowanie i dostosowanie do własnych preferencji oraz zarządza mechaniką działania widoku. Dzięki temu, że biblioteka daje możliwość rozszerzenia albo minimalizacji widoku *Simple Driver Assistant*, może działać równolegle z innymi aplikacjami np. nawigacją.

CircleImageView – biblioteka pozwalająca na zaokrąglanie zdjęcia, używa jej się w projekcie w celach estetycznych (Rysunek 36).

Picasso – biblioteka umożliwiająca prostsze oraz szybsze pobieranie i buforowanie obrazów w aplikacji. W projekcie *Simple Driver Assistant* jest używana do pobrania i wyświetlenia zdjęcia zalogowanego użytkownika (Rysunek 36).

5.5. Uwierzytelnienie użytkownika

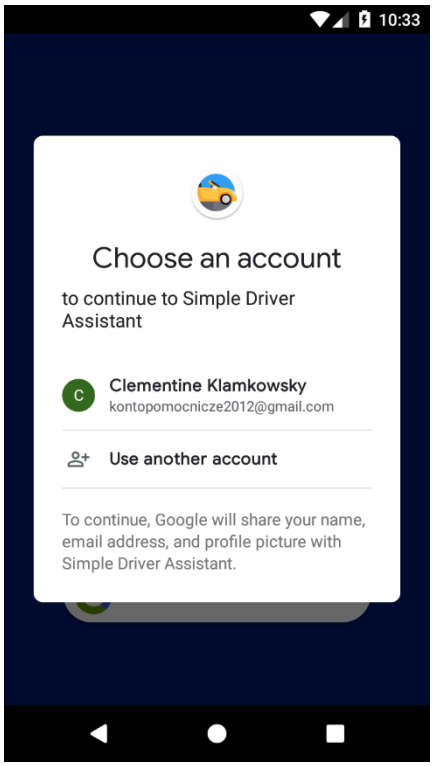
Proces logowania się do aplikacji zaczyna się w momencie wyboru przez użytkownika opcji „Login with Google" (Rysunek 37).



Rysunek 37 Widok logowania do aplikacji Simple Driver Assistant

Źródło: Opracowanie własne

Gdy aplikacja nawiąże połączenie z Firebase Authentication to zostanie wyświetlone okno logowania. Pozwoli ono na wybór konta Google, z którego chcemy być zalogowani do aplikacji (Rysunek 38).

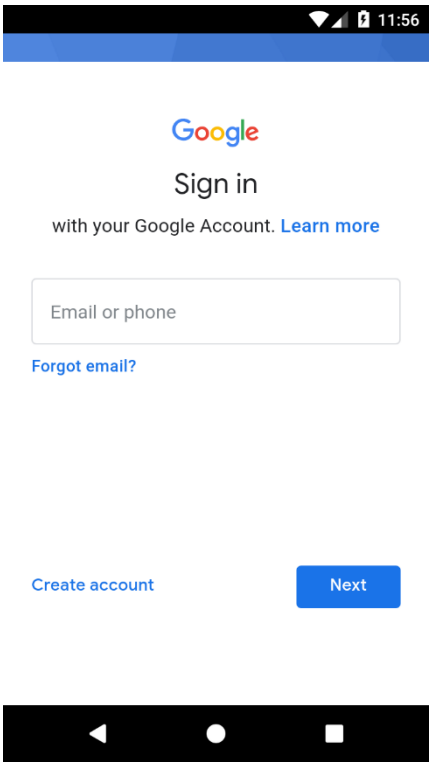


Rysunek 38 Okno logowania do aplikacji Simple Driver Assistant

Źródło: Opracowanie własne

Potwierdzając wybór konta (Rysunek 38) następuje wywołanie metody *getSignInIntent()*, rozpoczynającej proces autoryzacji użytkownika. Po zalogowaniu się do konta Google, następuje uwierzytelnienie przez metodę *firebaseAuthWithGoogle()*, która otrzymuje wynik logowania. Gdy wynik logowania jest pozytywny następuje utworzenie nowej aktywności, czyli przejście do widoku głównego aplikacji.

Proces rejestracji rozpoczyna się od wyboru przez użytkownika opcji „*Use another account*” (Rysunek 38).



Rysunek 39 Widok do logowania lub utworzenia konta Google

Źródło: Opracowanie własne

W celu kontynuowania procedury tworzenia nowego konta (Rysunek 39) wybieramy opcję „*Create account*”, następnie podajemy wymagane dane, aby pozytywnie zakończyć proces akceptujemy regulamin Google i zostaniemy przeniesieni do głównego widoku aplikacji.

6. Podsumowanie

Głównym celem niniejszej pracy dyplomowej była implementacja mobilnej aplikacji do informowania o zdarzeniach drogowych. Aplikacja ta pozwala kierowcy na zgłaszanie różnego rodzaju zdarzeń drogowych np. wypadek, partol policji, fotoradar itp. Dzięki temu inni użytkownicy mogą na bieżąco być informowani o zagrożeniach lub innych rodzajach zdarzeń drogowych. Prezentując istniejące już i dostępne na rynku rozwiązania, można zauważyć fragmentację, czyli rozproszenie użytkowników poprzez korzystanie z różnych dostępnych rozwiązań aplikacji tego typu. Bazy użytkowników są w ten sposób rozproszone, ze względu na osobiste preferencje każdego z kierowców. Taki podział niekorzystnie wpływa na funkcjonalność tych aplikacji w kontekście informowania o zdarzeniach drogowych. Zrozumiałe jest to, że od ilości użytkowników tworzących społeczność, zależy ilość oraz jakość zgłoszeń o wystąpieniu aktualnych zdarzeń drogowych.

Podsumowując, aplikacja *Simple Driver Assistant* swoją główną funkcjonalnością stara się rozwiązać problem fragmentacji bazy użytkowników, oferując uniwersalne rozwiązanie, które może uzupełnić pracę innych aplikacji do nawigowania w trasie. Proponuje również w miarę prostą obsługę poprzez logowanie się za pomocą konta Google, a także ocenę zgłoszenia drogowego, sprawdzenie wiarygodności osoby dodającej nowe zdarzenie drogowe oraz działanie aplikacji w tle w widoku zminimalizowanym i generowanie komunikatu głosowego w momencie tworzenia lub odbioru zgłoszenia.

Streszczenie

Zdarzeniem drogowym określa się wszystkie wydarzenia na drodze, które stanowią potencjalne zagrożenie w ruchu drogowym. Głównym celem tej pracy jest implementacja mobilnej aplikacji do informowania o zdarzeniach drogowych. Do osiągnięcia wyznaczonego celu projektu używam technologii tj. język programowania Java, TypeScript (funkcje w chmurze), wybranych usług Firebase. Aplikacja pozwala na tworzenie i odbieranie zgłoszeń drogowych, czyli dzielenie się aktualnymi informacji w trasie z innymi użytkownikami ruchu w okolicy.

Bibliografia

[Smyth, 2017] Smyth N., Firebase Essentials – Android Edition, Payload Media Inc, USA, 2017.

[Griffiths et al., 2016] Griffiths Dawn, Griffiths David, *Rusz głową! Android Programowanie aplikacji*, Rajca, Helion SA., PL, 2016.

[Guy, 2019] Guy H., NoSQL, NewSQL i BigData. Bazy danych następnej generacji, Pilch, Helion SA., PL, 2019.

[Stasiewicz, 2015] Stasiewicz A., *Android Studio Podstawy tworzenia aplikacji*, Helion SA., PL, 2015.

[Eckel, 2006] Eckel B., *Thinking in Java*, Szeremiota, wyd. 4, Helion SA., PL, 2006.

Zadłużenia (zadluzenia.com) – strona oficjalna

[WWW,1] https://www.zadluzenia.com/zdarzenie-drogowe/ (12.01.2020)

Policja – strona oficjalna

[WWW,2] http://statystyka.policja.pl/download/20/308515/Wypadki2018.pdf (13.01.2020)

Generalna Dyrekcja Dróg Krajowych i Autostrad – strona oficjalna

[WWW,3] https://www.gddkia.gov.pl/pl/19/lista-utrudnien (14.01.2020)

Targeo – strona oficjalna
[WWW,4] *https://mapa.targeo.pl/22.0089837,50.0416799,19* (14.01.2020)
Ważniak (wazniak.mimuw.edu.pl) – strona oficjalna

[WWW,5]
http://wazniak.mimuw.edu.pl/index.php?title=PO_Wst%C4%99p_do_Javy
(12.01.2020)

Tutorials Jenkov – strona oficjalna

[WWW,6] http://tutorials.jenkov.com/java/what-is-java.html (12.01.2020)

Net-Informations – strona oficjalna

[WWW,7] http://net-informations.com/java/cjava/garbage.htm (12.01.2020)

C-Sharpcorner – strona oficjalna

[WWW,8] https://www.c-sharpcorner.com/article/what-is-typescript/ (12.01.2020)

Wikipedia – strona oficjalna

[WWW,9] https://en.wikipedia.org/wiki/Android_Studio (12.01.2020)

[WWW,10] https://en.wikipedia.org/wiki/Visual_Studio_Code (12.01.2020)

Techopedia – strona oficjalna

[WWW,11] https://www.techopedia.com/definition/29428/backend-as-a-service-baas (12.01.2020)

Sekurak – strona oficjalna

[WWW,12] https://sekurak.pl/oauth-2-0-jak-dziala-jak-testowac-problemy-bezpieczenstwa/ (12.01.2020)

Firebase – oficjalna dokumentacja

[WWW,13] https://firebase.google.com/docs/firestore (12.01.2020)

[WWW,14] https://firebase.google.com/docs/functions (12.01.2020)

Strefakodera – strona oficjalna

[WWW,15] http://strefakodera.pl/programowanie/android-java/aktywnosci-w-androidzie (14.01.2020)

Wikibooks – strona oficjalna

[WWW,16] https://pl.wikibooks.org/wiki/Astronomiczne_podstawy_geografii/Odleg%C5%82o%C5%9Bci (12.01.2020)

Spis ilustracji

Załączniki

1) **Zawartość płyty CD**
   a) Wersja elektroniczna pracy dyplomowej
   b) Kod źródłowy *Aplikacji mobilnej do informowania o zdarzeniach drogowych*