



# DIPLOMSKI RAD

PRVOG CIKLUSA STUDIJA

## IOS APLIKACIJA ZA EVIDENTIRANJE I PRAĆENJE STANJA NA PUTEVIMA U BIH

STUDENT

HASELJIĆ ELDAR

# Sažetak

*U ovom diplomskom radu se govori o razvoju IOS aplikacije za evidentiranje i praćenje stanja na putevima u Bosni i Hercegovini korištenjem programskog jezika Swift i o načinu kako se ista aplikacija koristi. Unutar aplikacije za tip geografske karte na kojoj su prikazani radari i stanja na putevima korišten je MapKit. MapKit predstavlja Apple - ov Framework za predstavljanje geografskih karti za iOS, iPadOS, tvOS, watchOS i macOS uređajima. Korisnik pored vizualnog prikaza lokacija gore navedenih elemenata ima mogućnost dodavanja novih, ima opciju za filtriranja i brisanja lokacija radara ili stanja na putevima. Podaci su spremljeni u Cloud Firestore bazu podataka, koja je fleksibilna, skalabilna, NoSQL baza podataka koja se koristi prilikom razvoja mobilnih, web aplikacija. Aplikacija je dostupna na tri jezika Njemački, Engleski i Bosanski.*

**Ključne riječi:** Aplikacija, Swift, MapKit, Firestore, Radar, Stanje na putu

# Abstract

*This thesis discusses the development of IOS application for recording and monitoring the situation on the roads in Bosnia and Herzegovina using the Swift programming language and how to use that app. MapKit was used within the application for the type of geographic map showing radars and road conditions. MapKit is Apple 's Framework for presenting geographic maps for iOS, iPadOS, tvOS, watchOS and macOS devices. In addition to visually displaying the locations of the above elements, the user can add new ones, has the option to filter and delete radar locations or road conditions. The data is stored in the Cloud Firestore database, which is a flexible, scalable, NoSQL database used in the development of mobile, web applications. The application is available in three languages: German, English, and Bosnian.*

**Keywords:** Application, Swift, MapKit, Firestore, Radar, Road Condition

# Sadržaj

<b>1. Uvod.....</b>	<b>8</b>
<b>2. Razvoj mobilnih iOS aplikacija .....</b>	<b>9</b>
2.1. Reaktivno programiranje .....	9
2.1.1. RxSwift i RxCocoa Framework .....	10
2.2. Swift programski jezik .....	11
2.3. Xcode.....	11
2.3.1. Kreiranje Xcode projekta .....	12
2.4. iOS Simulator .....	12
2.5. CocoaPods .....	13
2.6. Načini pohrane podataka u iOS aplikacijama .....	15
2.6.1. UserDefaults .....	15
2.6.2. Keychain.....	16
2.6.3. File system.....	16
2.6.4. Sqlite.....	17
2.6.5. Core Data.....	17
2.6.6. Firebase / Firestore .....	17
2.7. Model – View – ViewModel (MVVM) .....	19
<b>3. Projektovanje iOS aplikacije za praćenje i evidentiranje stanja na putevima .....</b>	<b>20</b>
<b>4. Implementacija i način upotrebe iOS aplikacije za praćenje stanja na putevima.....</b>	<b>23</b>
4.1. Korisničko okruženje aplikacije .....	23
4.2. Opis rada glavnog zaslona.....	24
4.4. Opis rada zaslona za prikaz radara .....	26
4.5. Opis rada zaslona za prikaz stanja na putevima .....	28
4.6. Opis rada zaslona za filtriranje .....	30
4.7. Opis rada zaslona za prikaz detalja .....	31
4.8. Opis rada zaslona za prijavu novih radara ili stanja na putevima .....	34
4.9. Unos podataka u Firestore uz pomoć skripti .....	37
<b>5. Zaključak .....</b>	<b>38</b>
<b>6. Literatura.....</b>	<b>39</b>
<b>7. Dodatak.....</b>	<b>40</b>

# Popis slika

Slika 1. Prikaz iOS Simulator - a .....	13
Slika 2. MVVM (Model – View- ViewModel) arhitektura .....	19
Slika 3. Struktura aplikacije .....	21
Slika 4. Zaslona pokretanja .....	24
Slika 5. Glavni zaslon.....	24
Slika 6. Prikaz zaslona - ova vidljivih u aplikaciji .....	24
Slika 7. Prikaz otvorene stranice društvene mreže .....	25
Slika 8. Prikaz upozorenja prilikom klika na LinkedIn.....	25
Slika 9. Zaslona sa prikazanim radarima na originalnoj mapi .....	27
Slika 10. Zaslona sa prikazanim radarima na mapi sa terenom .....	27
Slika 11. Prikaz funkcionalnosti na zaslonu za prikaz radara na putevima .....	27
Slika 12. Prikaz zaslona za prikaz stanja na putevima .....	28
Slika 13. Prikaz funkcionalnosti na zaslonu za prikaz stanja na putevima .....	28
Slika 14. Prikaz svih znakova korištenih za prikaz stanja i radarana mapi.....	29
Slika 15. Zaslona za filtriranje stanja na putevima i zaslon za filtriranje radara .....	30
Slika 16. Prikaz malog opisa stanja radara ili stanja na putu .....	31
Slika 17. Prikaz zaslona za prikaz detalja radara, stanja na putevima i općih informacija .....	33
Slika 18. Uspješno prijavljen neaktivan radar ili stanje na putu .....	33
Slika 19. Pozicija oznake za prijavu.....	36
Slika 20. Prikaz report zaslona .....	36

# Popis skraćenica

iOS      iPhone Operating System

SDK      Software Development Kit

MacOS    Macintosh Operating System

LLVM    Low Level Virtual Machine

IDE      Integrated development Environment

MVVM    Model – View – ViewModel

NPM      Node Package Manager

ORM      Object Relational Mapping

# 1. Uvod

Zbog povećanja ljudskog stanovništva dolazi i do povećanja vozila na putevima. Prilikom povećanja broja vozila, javljaju se povećane gužve na svim putevima. Kako bi izbjegli te gužve potrebno je nešto što bi davalo informacije da li je određeni put prohodan, da li se na tom putu možda desila neka nesreća, da li se na tom putu nalazi neki radar itd. To nešto bi bila aplikacija i to mobilna aplikacija.

Odlučeno je da to bude mobilna aplikacija zato što osim povećanja gužvi na putevima, danas svaka osoba ima minimalno jedan mobilni uređaj pored sebe. Također u cilju jeste da korisniku bude omogućeno, pored praćenja, evidencije novih radara ili drugih stanja na putevima bude omogućeno i da ukloni neke radare ili stanja na putevima koji više nisu aktivni na istim lokacijama. U svijetu danas se nalazi 71.09% Android, 28.21% iOS i 0.7% ostalih korisnika. Aplikaciju je bilo moguće napraviti i u Kotlinu (programski jezik za programiranje Android aplikacija) međutim odlučeno je da to bude iOS aplikacija čisto iz razloga da se utvrdi i pokaže znanje budućeg inženjera iz oblasti Swift programskog jezika. Swift ujedno predstavlja osnovni programski jezik za programiranje iOS aplikacija.

Aplikacija bi bila dostupna na tri jezika tj. Bosanski, Engleski i Njemački, moguće bi bilo istu koristiti u vodoravnom (landscape) i vertikalnom (portrait) položaju mobitela, te kao i sve novije aplikacije posjedovala bi i tamni način rada tzv. „Dark mode“.

Kada bi aplikacija bila dostupna na App Store, korisnici bi nasumično mogli evidentirati gore navedene podatke, a ako bi bila otkupljena od strane neke firme ili ovlaštenog lica koji se bave time kao npr. BIHAMK ili firme koja želi da počne evidentirati stanja na putevima tada vjerojatno bi oni na dnevnoj bazi osvježavali bazu podataka sa novim podacima uz pomoć odgovarajućih skripti.

## 2. Razvoj mobilnih iOS aplikacija

Za razvoj iOS mobilnih aplikacija potrebni su minimalno sljedeći alati i tehnologije:

- MacOS računalo ili virtualna mašina koja pokreće MacOS
- Xcode i iOS SDK
- Poznavanje Swift programskog jezika

Također neki opcionalni alati i tehnologije koje se koriste prilikom izrade aplikacija su:

- Poznavanje načina rada sa Rx, Combine, SwiftUI i drugih Framework – a
- Realni uređaj za testiranje

U nastavku će biti predstavljeno reaktivno programiranje kao način razvoja iOS aplikacija.

### 2.1. Reaktivno programiranje

Reaktivno programiranje je način programiranja u kojem jedna komponenta odašilje niz podataka, dok se druga komponenta na njih pretplaćuje i sluša promjenu vrijednosti unutar toka, te ima odgovarajuću reakciju na njih. Reaktivno programiranje je uvedeno iz razloga što mobilni uređaji nisu dovoljno snažni za kompleksno računanje i teške poslove, pa dolazi do smrzavanja mobilnih aplikacija i slično. Protok podataka postoji sve dok se ne pošalje događaj greške ili događaj koji javlja da su poslani svi podaci.

Mobilno reaktivno programiranje se svodi na:

- Observable - predstavlja protok podataka. Podaci se mogu slati periodično ili samo jedanput, ovisno o postavkama protoka podataka. Pomoću njih se dohvataju i obrađuju podatci koji se prikazuju.



- Observer - predstavlja promatrača protoka podataka. Svaki promatrač se može pretplatiti na protok podataka. Svaki puta kada se odašilje podatak, promatrač u svojoj pretplati može izvršiti nekakvu akciju ili manipulaciju nad tim podacima ukoliko ima potrebe za tim.
- Scheduler – služi za definiranje koja akcija će se izvršavati na kojoj odnosno Thread - u. Tako da se sva promatranja protoka podataka izvršavaju na glavnoj niti odnosno Main thread -u, dok se sve pretplate izvršavaju na pozadinskoj niti odnosno Background thread - u.
- DisposeBag – predstavlja dodatni alat iz RxSwift – a koji pomaže da se ukine pretplata sa protoka podataka. On se izvršava kada se pozove 'deinit()' tzv. destruktora kontrolera. Da bi ukidanje bio uspješno potrebno je kreirati objekat 'DisposeBag' i priložiti ga objektu pretplate pomoću metode 'disposed(by:)', koja će automatski otkazati pretplatu kada se kontroler odbaci, odnosno ukloni. Bez ovoga moguće je da se desi curenje memorije (memory leak) što vodi do toga da bi aplikacija mogla pasti.

### 2.1.1. RxSwift i RxCocoa Framework

RxSwift je Framework za interakciju s programskim jezikom Swift, dok je RxCocoa Framework koji čini Cocoa API-je koji se koriste u iOS-u za korištenje s reaktivnim tehnikama, te predstavljaju dio paketa jezičnih alata ReactiveX (Rx). ReactiveX Framework - si pružaju zajednički rječnik za zadatke koji se više puta koriste u različitim programskim jezicima. Pristup RxCocoa ekstenzijama za odgovarajuću klasu, se vrši na način tako što nad objektom odgovarajuće klase se pozove metod „rx“. RxSwift ima mnoštvo operatora pomoću kojih se može odraditi pretvorba podataka. Neki od njih su map, filter, flatMap i slično. RxSwift i RxCocoa te druge biblioteke iz ReactiveX paketa u projekat se uključuju na način tako što u Podfile datoteku se dodaju sljedeće dvije linije koda:

```
pod 'RxSwift'
pod 'RxCocoa'
```

Dok u datotekama u kojim se žele koristiti ove biblioteke potrebno je da se pozovu sljedeće dvije linije koda:

```
import RxSwift
import RxCocoa
```

## 2.2. Swift programski jezik

Swift je objektno orijentiran programski jezik koji je kreiran od strane Apple - a za razvoj programa i aplikacija na prvenstveno na operativnim sistemima kreiranih od strane Apple - a. Prvi put je objavljen 2014. godine, te je razvijen kao zamjena za Apple - ov programski jezik Objective - C, koji je ranije korišten. Swift je brz i učinkovit jezik koji pruža povratne informacije u stvarnom vremenu i može se neprimjetno uklopiti u postojeći Objective - C kod. Swift radi s Apple Framework - sima Cocoa i Cocoa Touch, a ključni aspekt Swift - ovog dizajna bila je mogućnost interakcije s ogromnim brojem postojećeg Objective - C koda razvijenog za Apple proizvode u prethodnim desetljećima.

Swift koristi LLVM kompajler za kompajliranje koda koji dolazi sa Xcode 6 te koristi Objective - C runtime. Stoga, unutar jednog programa moguće je koristiti kod pisan u C, Objective - C, C++ i Swift programskom jeziku. Swift je također besplatan i otvorenog koda i dostupan je širokoj publici programera, edukatora i studenata pod licencom otvorenog koda Apache 2.0. Pruža binarne datoteke koje mogu kompilirati kod za iOS, macOS, watchOS, tvOS i Linux. Swift za Linux je prisutan od Swift 2.2 verzije (Mart 2016), dok je za Windows prisutan tek od Swift 5.3 verzije (Septembar 2020). Najnovija verzija u trenutku pisanja rada je bila Swift 5.5.

## 2.3. Xcode

Xcode je razvojna okolina koja se koristi za razvoj macOS, iOS, iPadOS, watchOS i tvOS aplikacija. Prvi put je objavljen 2003 godine, dok je najnovija stabilna verzija u trenutku pisanja ovog rada 13.1, objavljena 25. Oktobra 2021. godine, a dostupna je putem Mac App Store. Xcode sadrži editor, kompajler, emulator, razvojne Framework - e i još puno elemenata koji olakšavaju razvoj gore navedenih aplikacija. Moguće je koristiti neki drugi IDE za razvoj, no kako Apple ima striktna pravila kojima regulira strukturu aplikacije, koje mogu biti objavljene na App Store - u, preporučljivo je koristiti Xcode jer sam automatski otklanja potencijalne neregularnosti koje bi mogle prouzročiti probleme.

### **2.3.1. Kreiranje Xcode projekta**

Nakon što se preuzme Xcode sa Mac App Store - a potrebno je da se kreira projekat. Projekat se kreira na način da se odabere opcija „Create a new Xcode project“ na početnom ekranu kada se otvori Xcode. Prilikom kreiranja projekta moguće je izabrati opciju izrade različitih aplikacija (obična aplikacija, dokumentna aplikacija, igrice, sticker pack aplikacija i dr.) namijenjenih za više platformi (iOS, macOS, watchOS, tvOS, Driver Kit i dr.). Nakon odabira aplikacije potrebno je unijeti naziv iste unutar „Product Name“ polja, te je potrebno je odabrati tim unutar „Team“ uz pomoć padajućeg menija. Ukoliko je ranije izvršeno povezivanje Xcode - a sa vlastitim Apple ID – om u projektu će se pojaviti timovi za App Store Connect, dok u slučaju besplatnog računa programera također je moguće odabrati lični tim. Ukoliko je planirano da aplikacija bude rađena uz pomoć Storyboada (UIKit) ili uz pomoć SwiftUI potrebno je odabrati odgovarajući okruženje unutar „Interface“ sekcije. Definiciju koji će osnovni jezik biti korišten se nalazi unutar „Language“, sekcije gdje se nalaze opcije Objective - C i Swift za Storyboard okruženje dok za SwiftUI okruženje jedino Swift je dostupan. Gore navedena konfiguracija je dovoljna za kreiranje osnovnog projekta.

Također na raspolaganju su i dodatne opcije, koje dodaju određeni zadani kod aplikaciji, kao što je kod koji služi za spremanje trajnih podataka ili za keširanje privremenih podataka, kod koji omogućava pisanje test datoteka za testiranje i druge opcije

## **2.4. iOS Simulator**

iOS Simulator je alat koji služi za pokretanje i testiranje aplikacija na korisničkom računar. Simulator omogućuje simulaciju većeg broja Apple uređaja, to ima svoje prednosti i nedostatke. Koristan je za simuliranje i testiranje korisničkog okruženja koje se može obavljati interakcijom miša, međutim za simuliranje i testiranje složenijih programskih rješenja postaje nedovoljan. Cijela aplikacija ovog diplomskog rada je simulirana u iOS Simulatoru koji je mogao i uspio odraditi sve potrebne zahtjeve korisnika. Aplikacija je također testirana na fizičkom uređaju iPhone 8, OS verzija 15.1.



Slika 1. Prikaz iOS Simulator - a

## 2.5. CocoaPods

CocoaPods predstavlja menadžera koji upravlja bibliotekama koje se koriste u Xcode projektima. Biblioteke koje se koriste u projektima navedene su u jednoj tekstualnoj datoteci pod nazivom Podfile. CocoaPods riješava zavisnosti između biblioteka, dohvata izvorni kod, a zatim ga povezuje u Xcode radni prostor za izgradnju projekta. CocoaPods je izgrađen s Ruby-em i instaliran je sa zadanim Ruby - om dostupnim na korisničkom macOS-u. Da bi programer mogao koristiti CocoaPods u projektima, potrebno je da se izvrši sljedeća linija koda u terminalu.

```
sudo gem install cocoapods
```

Nakon što se instalira CocoaPods potrebno je napraviti Podfile unutar projekta. Podfile predstavlja datoteku koja sadrži dodatne biblioteke i specifikacije vezane za projekat. Kreiranje date datoteke se vrši uz pomoć ove linije koda

```
pod init
```

Prva linija koda u kreiranom fajlu je “platform :iOS, '14.0'” te ona predstavlja platformu i najnižu verziju iOS - a od koje je aplikacija podržana.

```
# Uncomment the next line to define a global platform for your project
platform :iOS, '14.0'

target 'BosniaRoadTraffic' do
  # Comment the next line if you don't want to use dynamic frameworks
  use_frameworks!

  # Pods for BosniaRoadTraffic
  pod 'RxSwift'
  pod 'RxCocoa'
  pod 'FirebaseFirestoreSwift'
  pod 'Firebase'
  pod 'Firebase/Analytics'
end

deployment_target = '14.0'
post_install do |installer|
  installer.generated_projects.each do |project|
    project.targets.each do |target|
      target.build_configurations.each do |config|
        config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] =
          deployment_target
        end
      end
    end
  end
  project.build_configurations.each do |bc|
    bc.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = deployment_target
    end
  end
end
```

Za korištenje CocoaPods biblioteke u određenom projektu, potrebno je definirati Xcode Target unutar Podfile datoteke. Na primjer, u gore navedenom fajlu to bi bio naziv aplikacije te to predstavlja liniju koda u kojoj piše „target 'BosniaRoadTraffic' do“. Nakon te linije koda slijede takozvani CocoaPod - ovi koji predstavljaju dodatne biblioteke koje se koriste u projektu koje se navode u formatu „pod 'PODNAME'“.

Nakon navedenih svih biblioteka slijedi `end` komanda na kraju. Unutar Podfile datoteke se mogu navesti i određene dodatne specifikacije kao što je urađeno u gore navedenom Podfile - u. Ovdje dodatni kod izvršava postavljanje „`deployment_target`” - a za sve biblioteke na 14.0 verziju.

Nakon kreiranja sadržaja Podfile datoteke potrebno je izvršiti određene linije koda koje će instalirati sve navedene biblioteke u kreirani projekat i izvršiti nadogradnju na njihove najnovije verzije. Također te komade kreiraju MyApp.xcworkspace datoteku koja se koristi dalje u razvoju aplikacije.

```
pod install  
pod update
```

## 2.6. Načini pohrane podataka u iOS aplikacijama

Neki od najčešće korištenih načina pohrane i dohvata podataka koji se koriste prilikom izrade iOS aplikacija su:

- UserDefaults
- KeyChain
- File System
- Sqlite
- Core Data
- Google Firebase

### 2.6.1. UserDefaults

**UserDefaults** je najčešći i najudobniji način pohrane i dohvaćanja informacija. Oni rade kao pohrana ključ/vrijednost, s nizovima kao ključevima. Kao što im naziv implicira, **UserDefaults** trebali bi koristiti za pohranu korisničkih postavki odnosno za pohranjivanje malih dijelova informacija (npr. da li korisnik preferira svijetli ili tamni način rada, u koje vrijeme želi primati dnevni podsjetnik, želi li zapravo primati obavijesti itd.).

## 2.6.2. Keychain

Neke aplikacije moraju pohraniti privatne podatke ili podatke koji su od velikog značaja kao npr. lozinke, certifikati i slične stavke, te kad god trebamo pohraniti male dijelove informacija koje moramo šifrirati i čuvati upotreba **Keychain** - a bi dobro došla. Ali potrebno je uzeti u obzir da se sve mora biti pohranjeno kao Data tip. Stoga je potrebno proslijediti samo objekte i vrijednosti koji se mogu dekodirati u taj tip.

## 2.6.3. File system

Kada ono što je potrebno pohraniti ne spada u gore navedene slučajeve, onda pohrana bi trebala ići u **File system**. **File system** u iOS svijetu radi točno kao **File system** na prijenosnom računalu. On koristi staze i URL - ove za identifikaciju resursa na disku. Međutim, u razvoju iOS - a postoje neke preferirane mape o kojima bi svi programeri trebali znati:

- Documents
  - Ovo je glavna mapa u koju se pohranjuju podaci koje generira korisnik.
- Documents/Inbox
  - Ovo je posebna mapa koju kreira sistem kad god druga aplikacija traži od naše aplikacije da otvori datoteku. Ova je mapa samo za čitanje iz perspektive aplikacije. Program Mail, na primjer, postavlja dodatke e - pošte povezane s vašom aplikacijom u ovaj direktorij.
- Library
  - Ovo je mapa u koju programeri, imaju mogućnost pohraniti datoteke koje aplikacija zahtijeva. Na primjer, moguće je pohraniti binarnu datoteku koja može pružiti neke dodatne podatke aplikaciji.
- Library/Caches
  - Ovo je posebna mapa gdje se mogu spremati datoteke koje bi mogle uskoro zatrebati, ali ne marimo za gubitak. Kada se iste datoteke ne koriste, sistem će ih izbrisati. Slike koje se preuzimaju radi bržeg učitavanja aplikacije trebale bi se nalaziti u ovoj mapi.

- tmp
  - Još jedna posebna mapa gdje pohranjene datoteke ne ostaju tijekom pokretanja aplikacije. Sistem bi mogao u bilo kojem trenutku očistiti ovu mapu te bi sadržaj iste bio obrisao.

**File system** dobar je kandidat za pohranu slika koje se preuzimaju s interneta i koje je poželjno držati pri ruci kako bi vrijeme učitavanja bilo kraće.

## 2.6.4. Sqlite

**SQLite** je definitivno standard za interne baze podataka. SQL baza podataka zahtijeva da se modeli organiziraju u tablice. Zatim će se pobrinuti za njihovo pohranjivanje i dohvaćanje na disk na najučinkovitiji mogući način. Osim što omogućava pohranu, posjedovanje SQL baze podataka omogućuje učinkovito pronalaženje i pruža mogućnost pisanja složenih upita za dohvaćanje zanimljivih uvida.

## 2.6.5. Core Data

**Core Data** je okvir za objektno - relaciono mapiranje (ORM) koji dolazi s iOS - om. **Objektno - relacijsko mapiranje** u računarstvu je tehnika programiranja za pretvaranje podataka između sistema nekompatibilnih tipova koristeći objektno orijentirane programske jezike. **Core Data**, kao i **SQLite**, nije mjesto za pohranjivanje podataka, već način da se njima manipulira. Na njega se gleda kao na evoluciju **SQLite** - a. **Core Data** se uključuje u projekat odabirom **Core Data** opcije prilikom kreiranja samog projekta. U ovom diplomskom radu **Core Data** je korišten za spremanje podataka na način da korisnik ima pristup zadnjim informacijama dohvaćenih sa interneta i u slučaju kada nema pristupa internetskoj mreži.

## 2.6.6. Firebase / Firestore

Firebase je Google - ova platforma za razvoj web i mobilnih aplikacija. Platforma Firebase sastoji se od mnogo servisa čija je uporaba besplatna, a neki od njih su npr. Cloud Firestore, ML Kit, Cloud Functions, Authentication, Hosting, Cloud Storage, Realtime Database i dr. Cloud Firestore služi za pohranu i sinkronizaciju podataka između korisnika i uređaja na globalnoj razini koristeći NoSQL bazu podataka. Cloud Firestore pruža trenutnu sinkronizaciju podataka i izvan mrežnu podršku uz učinkovite upite prema bazi.



Odlična integracija s drugim Firebase proizvodima omogućuje ubrzanu izgradnju aplikacija. U ovom radu korištena je samo Cloud Firestore funkcionalnost Firebase - a. Firestore je korišten za spremanje pozicija radara, stanja na putevima, te ostalih informacija i detalja koji oni nose sa sobom. Korišten je iz razloga jer je bilo potrebno da imao jednu bazu na nekom serveru kojoj bi svi korisnici pristupali za dohvaćanje, dodavanje ili brisanje određenih podataka.

Za korištenje Firebase - a, potreban je Google račun. Nakon prijave na Firebase s tim računom, potrebno je kreirati projekat na Firebase koji je također potrebno uvezati sa Xcode projektom. Nakon što se kreira projekat na Firebase.com, dobiva se datoteka pod nazivom „GoogleService-Info.plist“ koji je potrebno dodati u Xcode projekat. Također potrebno je dodati odgovarajuće biblioteke, koje su navedene u nastavku, u Podfile datoteku koja je ranije pomenuta i potrebno je pozvati „pod update“ liniju koda u terminalu.

Potrebno je pozvati i `Firebase.configure()` funkciju unutar class AppDelegate u dole navedenoj funkciji, te potrebno je dodati liniju `import Firebase` van iste klase. Na ovaj način je izvršeno podešavanje same Firestore baze podataka te je tim postupkom Firestore uspješno podešen za aplikaciju i rad.

```
pod 'Firebase/Firestore'
pod 'Firebase'
pod 'Firebase/Analytics'
```

```
import Firebase

@main
class AppDelegate: UIResponder, UIApplicationDelegate {

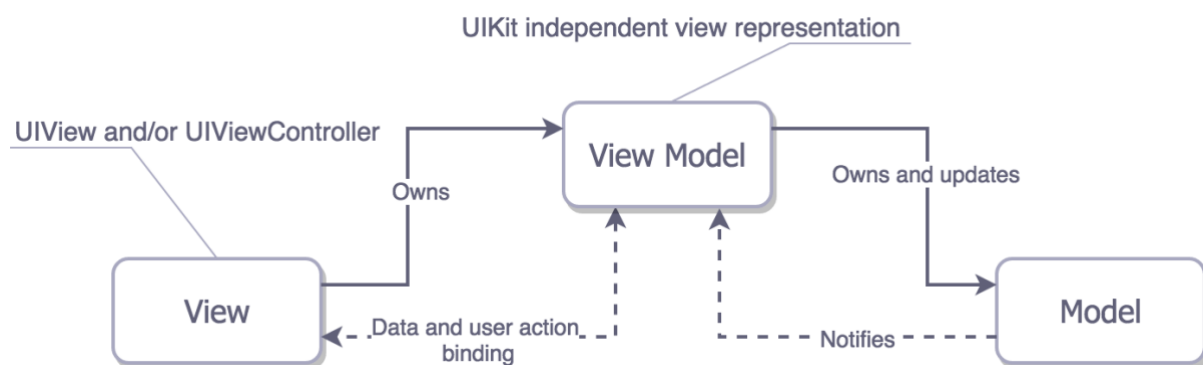
    func application(_ application: UIApplication,
                    didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        FirebaseApp.configure()
    }

    return true
}
```

## 2.7. Model – View – ViewModel (MVVM)

Model – View – ViewModel je način dizajna programske arhitekture. Programska arhitektura je plan koji opisuje skup aspekata i odluka koji su od iznimne važnosti za projekat. Arhitektura podrazumijeva uzimanje u obzir sve vrste zahtjeva, organizaciju sistema, međusobnu komunikaciju između dijelova sistema i drugo. Glavna karakteristika MVVM arhitekture je potpuna odvojenost poslovne logike od logike za postavljanje View - a. Unutar ViewModel - a se obavlja sva manipulacija nad podacima, tako da View ne zna ništa o podacima osim njihovog prikaza na korisničkom okruženju. Kao što se može primijetiti iz naziva, arhitektura se sastoji od tri dijela:

- Model
  - Sadrži podatke aplikacije. Najčešće se radi od klasama ili strukturama.
- View
  - Sadrži kod za prikaz vizualnih elemenata korisničkog okruženja i reakcije na korisničke akcije. Sastoji se od protokola koji sadrže metode korisnikovih akcija i ViewController - a koji je odgovoran za prikazivanje komponenti korisničkog okruženja i slanje informacija viewModel-u kada je korisnik izvršio neku akciju.
- ViewModel
  - Sadrži poslovnu logiku aplikacije i sastoji se od različitih programskih dijelova ovisno o složenosti aplikacije.



Slika 2. MVVM (Model – View- ViewModel) arhitektura

### 3. Projektovanje iOS aplikacije za praćenje i evidentiranje stanja na putevima

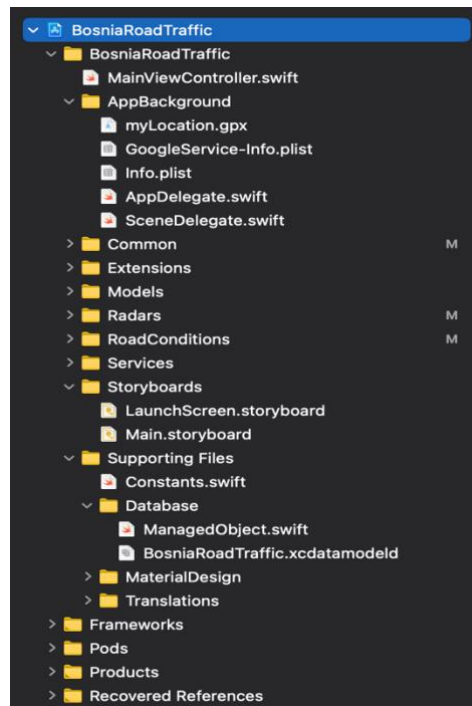
Zahtjevi aplikacije koju je bilo potrebno implementirati u ovom diplomskom radu su bili sljedeći:

- Omogućiti prikaz lokacija radara na putevima
- Omogućiti prikaz lokacija radova, saobraćajnih nesreća i drugih stanja na putevima
- Omogućiti dodavanje novih lokacija radara i stanja na putevima
- Omogućiti uklanjanje lokacija radara i stanja na putu ukoliko nisu više aktivne
- Omogućiti prikaz kratkog i detaljnog opisa radara i stanja na putu
- Omogućiti filtriranje lokacija radara i stanja na putu na osnovu određenog faktora
- Omogućiti prikaz detaljnog izvještaja stanja na putevima
- Omogućiti način da korisnik ima zadnje dohvaćene podatke u svom uređaju u slučaju kada nema pristupa internetskoj mreži.

Neke opcionalne funkcionalnosti koje nisu bile zahtijevane ali su pokrivene u ovom radu su:

- Mogućnost rada aplikacije u noćnom modu (Dark mode)
- Mogućnost rada aplikacije na tri različita jezika (Engleski, Bosanski, Njemački)
- Mogućnost rada aplikacije u horizontalnom i vertikalnom načinu rada

Aplikacija je pisana uz pomoć MVVM arhitekture, a na slici 3. se vidi struktura aplikacije. `Manager.swift` je glavna datoteka koji služi za uspostavljanje komunikacije između uređaja i Firestore baze podataka koja se nalazi na Google servisima, te veza i sa samim datotekama u kojim su implementirane klase i funkcionalnosti koje se koriste u cijeloj aplikaciji. *`AppDelegate.swift`* je glavna tačka ulaska u aplikaciju koji je kreiran prilikom kreiranja samog projekta.



Slika 3. Struktura aplikacije

Funkcije iz AppDelegate.swift pozivaju se za događaje u životnom ciklusu na razini aplikacije. U zadanom AppDelegate.swift postoje tri metode za koje Apple smatra da su važne koje moramo razmotriti

- **func application(\_: didFinishLaunchingWithOptions :) -> Bool**
  - Ova metoda se poziva kada se aplikacija pokrene i gdje se vrši postavljanje aplikacije. Tu je izvršena konfiguracija Firebase baze podataka.
- **func application(\_: configurationForConnecting: options :) -> UISceneConfiguration**
  - Ova metoda se poziva kada aplikaciji za prikaz bude potrebna nova scena ili prozor. Ova metoda se ne poziva pri pokretanju aplikacije, već samo kada je potrebno pribaviti novu scenu ili novi prozor.
- **func application (\_: didDiscardSceneSessions :)**
  - Ova metoda se poziva kad god korisnik odbaci scenu poput prevlačenja iz prozora za više zadataka ili ako se odbaci programski. Ova se metoda poziva za svaku odbačenu scenu nedugo nakon što se pozove metoda (\_: didFinishLaunchingWithOptions :) ako se aplikacija ne izvodi kada korisnik odbaci scenu.

Od verzije 13.0 pa nadalje, **SceneDelegate.swift** preuzima neke odgovornosti od AppDelegate.swift, tj. konkretno funkcije vezane za UIWindow iz AppDelegate.swift. Sada je UIScene u SceneDelegate tj, SceneDelegate.swift je odgovoran za ono što je prikazano na ekranu. Unutar aplikacije nije ništa mijenjano vezano za ovu datoteku, tako da je sadržaj ostao isti onakav kakav bude kada se kreira sam projekat. Kao što se može vidjeti na slici 3. pored navedene dvije glavne datoteke za pokretanje same aplikacije, koje se nalaze unutar AppBackground foldera, u strukturi aplikacije neke od važnijih datoteka također su i

- *BosniaRoadTraffic.xcdatamodeld*
  - *CoreData baza podataka na mobitelu*
- *GoogleService-Info.plist*
  - *Sadrži sve podatke potrebne Firebase - u za povezivanje s Firebase projektom*
- *Info.plist*
  - *Popis svih karakteristika čiji parovi ključ / vrijednost navode bitne informacije o konfiguraciji vezanih za aplikaciju.*
- *Localizable.strings*
  - *Datoteka koja sadrži tri različita datoteke, tj. svaka datoteka je za jedan jezik i sadrži rečenice koje je imaju odgovarajuće prevode za tri jezika koji su navedeni ranije*
- *Constants.swift*
  - *Datoteka koja sadrži određene konstantne vrijednosti kao url – ove za društvene mreže, storyboard identifikatore i druge vrste konstanti*
- *AppColor.swift*
  - *Datoteka koja sadrži vrijednosti određenih boja koje su korištene u aplikaciji*
- *Reachability.swift*
  - *Datoteka koja služi za detekciju da li korisnik ima pristup internetskoj mreži*
- *myLocation.gpx*
  - *Datoteka koja služi za postavljanje određene korisničke lokacije na mapi i koristi se samo u svrhe testiranja određene korisničke lokacije na simulatoru.*

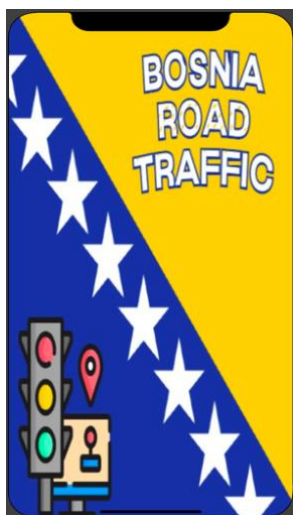
## 4. Implementacija i način upotrebe iOS aplikacije za praćenje stanja na putevima

### 4.1. Korisničko okruženje aplikacije

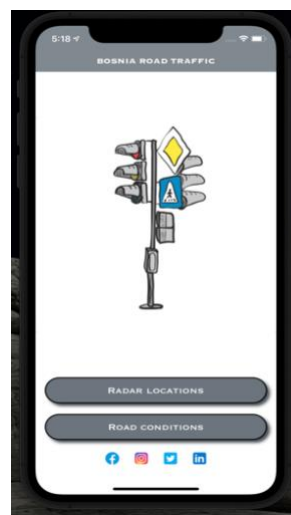
Aplikacija razvijena u diplomskom radu podržava sve vrste iPhone uređaja čija je verzija operativnog sistema veća ili jednaka 14.0. verziji. Također aplikacija ima podršku korištenja iste u vodoravnom (landscape) ili vertikalnom (portrait) načinu rada. Korisničko okruženje je razvijeno u Swift programskom jeziku i uz pomoć Storyboard - ova. Storyboard je vizualni prikaz korisničkog okruženja iOS aplikacije, koji prikazuje zaslone sadržaja i veze između tih ekrana. U nastavku rada su navedene i razrađene detaljno sve funkcionalnosti svakog zaslona.

Aplikacija ima nekoliko zaslona, a to su:

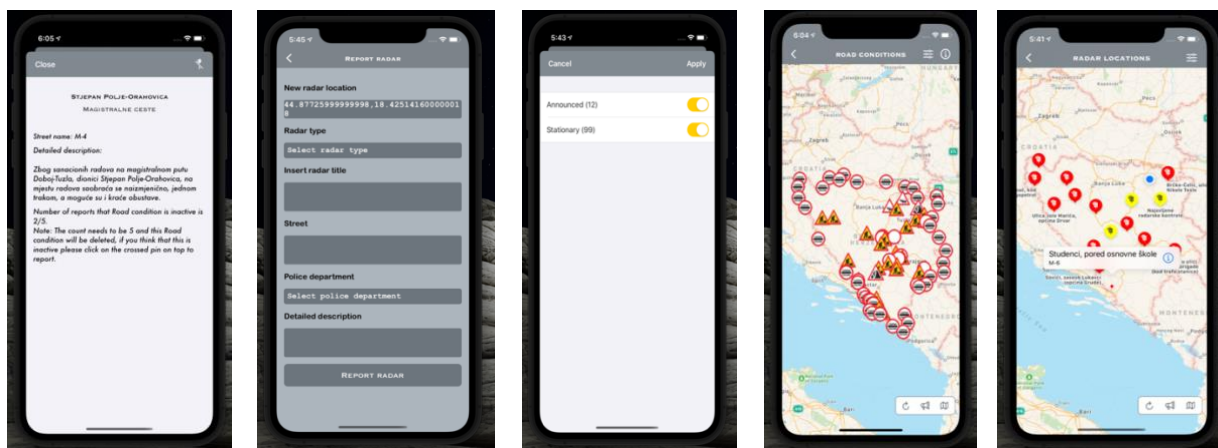
- Zaslون pokretanja
  - *LaunchScreen.storyboard*
- Glavni zaslon
  - *MainScreen.storyboard*
- Zaslon za prikaz radara
  - *RadarsMapStoryboard.storyboard*
- Zaslon za prikaz stanja na putevima
  - *RoadConditionsStoryboard.storyboard*
- Zaslon za prijavu novog radara / stanja na putu
  - *ReportStoryboard.storyboard*
- Zaslon za filtriranje radara / stanja na putevima
  - *FilterStoryboard.storyboard*
- Zaslon za prikaz detalja vezanih za radar / stanje na putu i prikaz općih informacija za puteve
  - *DetailsStoryboard.storyboard*



Slika 4. Zaslona pokretanja



Slika 5. Glavni zaslon



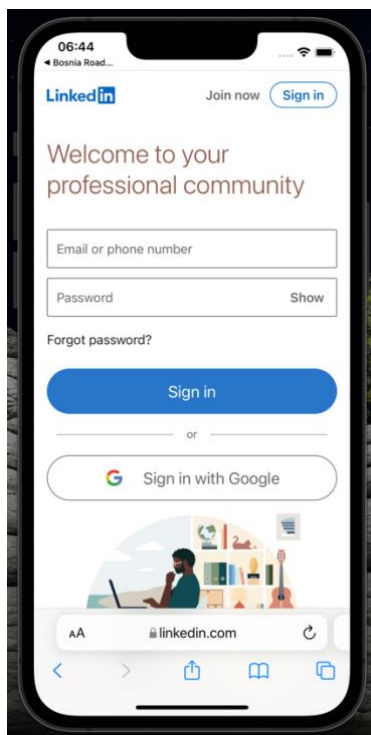
Slika 6. Prikaz zaslona - ova vidljivih u aplikaciji

## 4.2. Opis rada glavnog zaslona

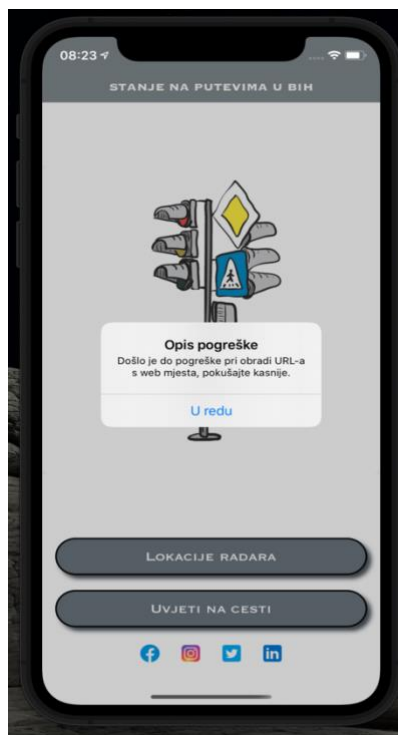
Kada korisnik pokrene aplikaciju na svom iPhone uređaju prva stvar koja se prikazuje jeste zaslon pokretanja koji je prikazan na Slici 4. U suštini to je zaslon koji dobijemo kada kreiramo sam projekat. Razlika između zadane vrijednosti koja se dobije prilikom kreiranja projekta i ovog zaslona (**LaunchScreen.storyboard**) jeste što se na ovome nalazi slika, dok na zadanoj vrijednosti se nalazi bijela ili crna pozadina koja zavisi da li je tamni način rada uključen. Nakon što se aplikacija učita prikazuje se glavni zaslon (**MainScreen.storyboard**) čiji je izgled prikazan na Slici 5. Na glavnom zaslonu se nalazi naziv na vrhu zaslona, dva siva dugmeta, te nekoliko sličica društvenih mreža koje također predstavljaju dugmad.

Pritiskom na neku od sličica društvenih mreža, otvoriti će se zadani vanjski pretraživač na mobitelu (Safari inače) sa zadanom društvenom mrežom. Sve sličice društvenih mreža otvaraju društvenu mrežu odgovarajuću društvenu mrežu, osim u slučaju ako je link društvene mreže pogrešno podešen, tada korisnik dobiva upozorenje (Alert) sa informacijom da je link krivo podešen.

Svaki klik na bilo koje navedeno dugme u aplikaciji je izvršen uz pomoć tzv. Reaktivnog wrappera za TouchUpInside event nad dugmetom (`rx.tap`). Na njega smo pretplaćeni uz pomoć `bind` funkcije, gdje unutar njenog tijela je navedeno šta je potrebno da se desi u slučaju klika na dugme. U konkretnom slučaju sličica društvenih mreža, otvara se odgovarajući URL u vanjskom pretraživaču.



**Slika 7. Prikaz otvorene stranice društvene mreže**



**Slika 8. Prikaz upozorenja prilikom klika na LinkedIn**



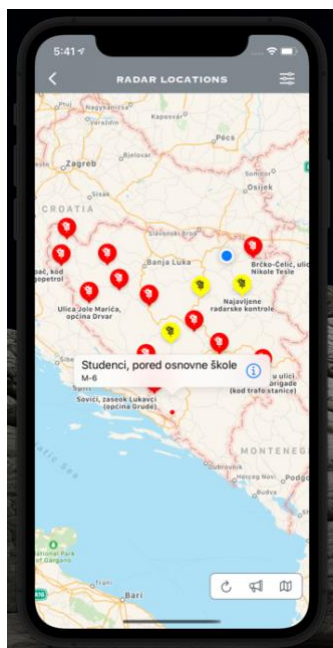
## 4.4. Opis rada zaslona za prikaz radara

Ukoliko se klikne na gornje sivo dugme na glavnom zaslonu na kojem piše „Lokacije radara“ na odgovarajućem jeziku, pokrenut će se spinner za učitavanje. Nakon toga korisniku će se prikazati upozorenje (Alert) koje od korisnika zahtijeva da dopusti prikaz svoje lokacije na mapi, gdje korisnik može odabrati da uvijek, samo jednom dopušta prikaz lokacije ili odbije prikaz lokacije.

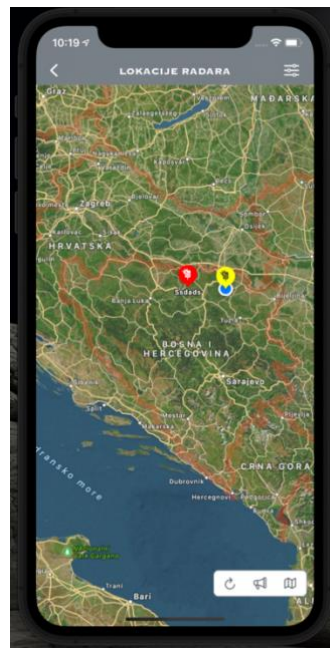
Ukoliko korisnik odbije prikaz lokacije biće vraćen automatski na glavni tj. početni zaslon, te naredni put kada bude otvara ovaj zaslon dobit će poruku da je potrebno da dopusti aplikaciji da pristupi njegovoj lokaciji da bi mogao nastaviti koristiti istu. Ukoliko je dopušten prikaz lokacije onda se poziva API poziv koji dohvata sve lokacije radara koje se nalaze trenutno u Firestore bazi, te se isti spremaju i u CoreData bazu na mobitelu. Nakon što su radari učitani oni se prikazuju na mapi kao na slici 9. U ovom slučaju kada je lokacija odobrena gore navedeno upozorenje se neće više javljati sve dok korisnik ne promijeni pristup lokaciji u postavkama. U gornjem desnom uglu, kada se radari učitaju korisnik ima opciju da filtriranje radara. Ukoliko je vidljiv samo jedan ili ni jedan tip radara na mapi onda je opcija za filtriranje isključena tj. nije vidljiva.

Pored opcije za filtriranje u desnom donjem uglu se nalaze tri dodatne opcije tj. dugmeta. Prvo dugme služi za ponovno učitavanje zaslona tj. služi da se osvježe lokacije radara, odnosno učitaju novi i izbrišu stari ukoliko nisu više u Firebase bazi. Pored te opcije nalazi se opcija za prijavu tj. ako uočimo novi radar na nekoj lokaciji možemo ga prijaviti. Zadnja opcija na ovom zaslonu služi za promjenu tipa mape tj. imamo mapu kao što je prikazana na slici 9. i mapu sa terenom koja je prikazana na slici 10. Ukoliko radara nema ili ako imaju neke informacije o novim radarima ili ako se desi neka pogreška uvijek će se pojaviti na zaslonu odgovarajuće upozorenje (Alert), tipa kao na slici 8.

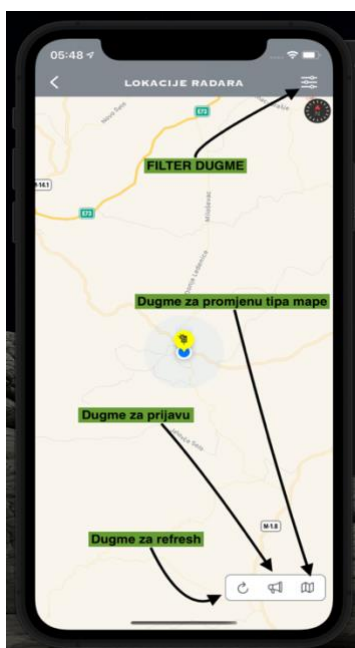
Radari na mapi su označeni sa crvenom ili žutom bojom, crveni su stacionarni, dok su žuti najavljeni radari. Prilikom klika na neki od njih dobivamo kratke informacije o istom kao što je učinjeno na slici 9, dok ukoliko ponovo se izvrši klik na taj mali skočni prozor otvorit će se zaslon sa detaljnijim informacijama vezanih za taj radar.



Slika 9. Zaslona sa prikazanim radarima na originalnoj mapi



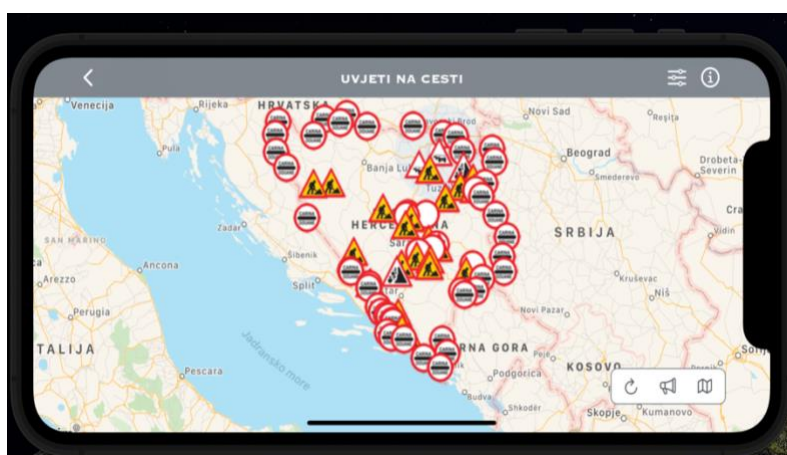
Slika 10. Zaslona sa prikazanim radarima na mapi sa terenom



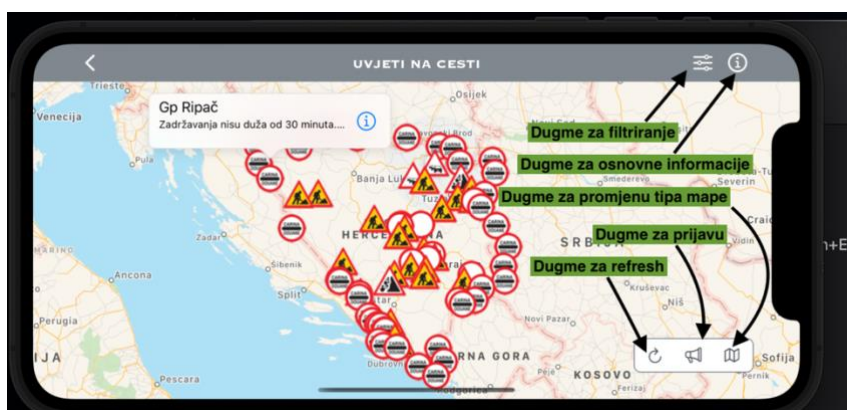
Slika 11. Prikaz funkcionalnosti na zaslonu za prikaz radara na putevima

## 4.5. Opis rada zaslona za prikaz stanja na putevima

Zaslon za prikaz stanja na putevima je po funkcionalnosti sličan zaslonu za prikaz radara. Zahtjev aplikacije za korisničkom lokacijom je isti kao i na zaslonu za prikaz radara te vrijede ista upozorenja. Opcije koje su vidljive na ovom zaslonu su kao i na zaslonu za prikaz radara, a to su opcije za refresh, prijavu novog stanja na putu, filtriranje stanja na putu, te promjena tipa mape. Ukoliko se izvrši klik na bilo koje stanje na putu prikazati će se prvo kratki opis, a ukoliko kliknemo na taj kratki opis otvoriti će se zaslon za detaljniji opis kao i na radarima. Razlike između ovog i prethodnog zaslona jesu u oznakama stanja na putevima jer ovdje koristimo odgovarajuće sličice za stanja umjesto crvene i žute boje koje smo koristili za radare. Također na ovom zaslonu se nalazi i info opcija koju vidimo na slici 12. i 13. u gornjem desnom uglu pored filter opcije.



Slika 12. Prikaz zaslona za prikaz stanja na putevima



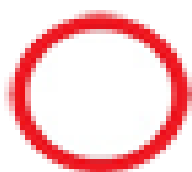
Slika 13. Prikaz funkcionalnosti na zaslonu za prikaz stanja na putevima

Svako stanje je predstavljeno odgovarajućom slikom. Trenutno od stanja na putevima aplikacija pokriva sljedeća stanja:

- Granični prijelaz
- Sanacija kolovoza
- Potpuna obustava saobraćaja
- Zagušenje
- Zabrana za teretna vozila
- Odron
- Saobraćajna nezgoda
- Poledica
- Opasnost



a) Odron



b) Potpuna obustava saobraćaja



c) Radar



d) Saobraćajna nezgoda



e) Granični prijelaz



f) Opasnost



g) Zabrana za teretna vozila



h) Zagušenje



i) Sanacija kolovoza




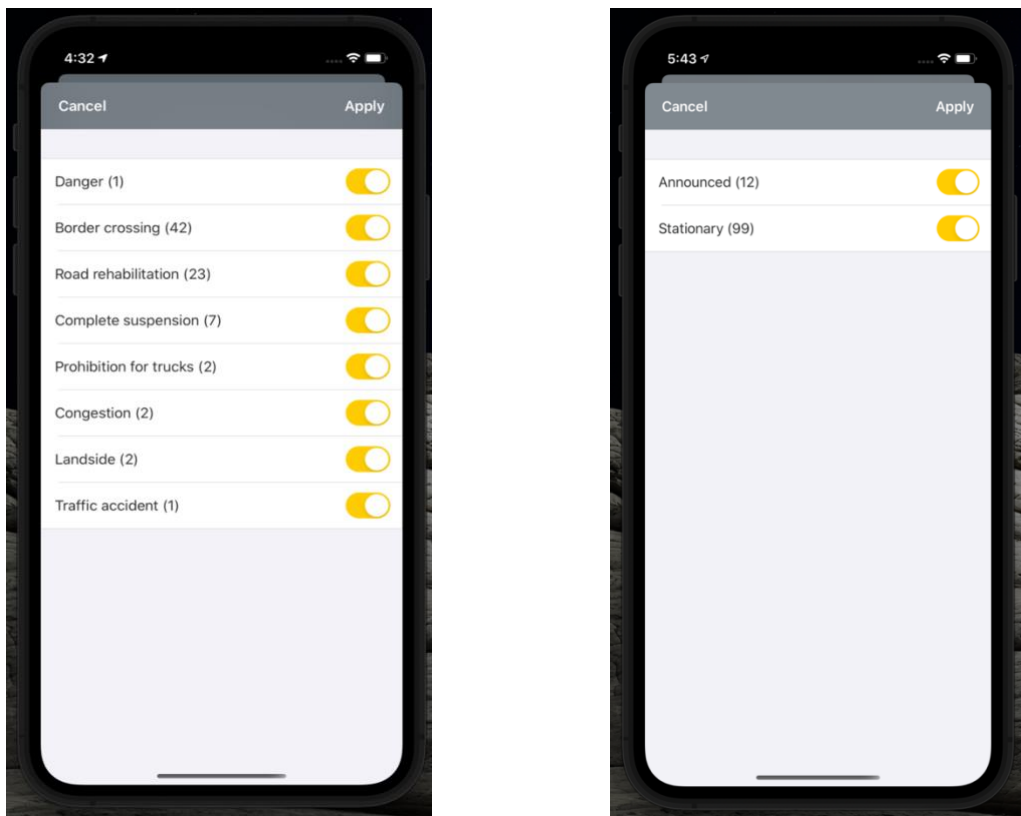
j) Poledica

**Slika 14. Prikaz svih znakova korištenih za prikaz stanja i radara na mapi**

Na slici 14. su prikazani svi znakovi koji su korišteni za prikaz odgovarajućih stanja na putu koje aplikacija pokriva i oznaka radara na mapi. U zavisnosti da li je radar stacionarni ili najavljeni vrši se promjena pozadine iza sličice za prikaz radara u žutu ili crvenu. Za stanje na putevima je drugačije, uzima se naslov slike koji dobijemo sa API - a te u zavisnosti od naziva koristimo odgovarajuću sličicu. Sve slike 14. su spremljene direktno u aplikaciji.

## 4.6. Opis rada zaslona za filtriranje



Ova dva zaslona su po izgledu isti, razlike su u sadržaju. Kada sa zaslona sa prikaz radara ili zaslona za prikaz stanja na putevima pritisne na dugme predstavljeno sa ikonicom  otvorit će se screen koji izgleda kao na Slici 15.

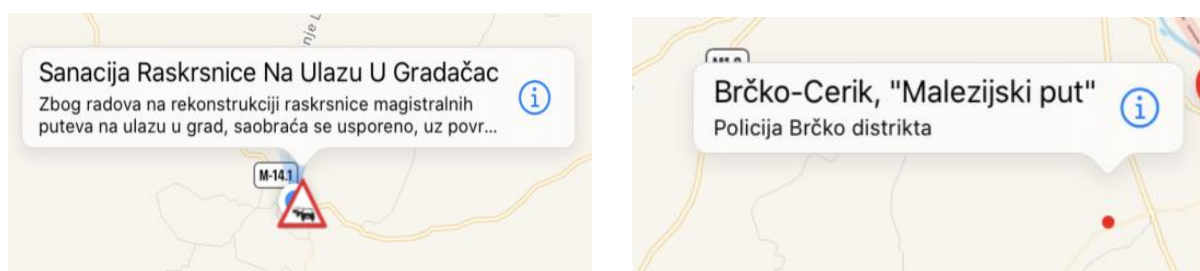


**Slika 15. Zaslon za filtriranje stanja na putevima i zaslon za filtriranje radara**

Navedeno dugme je vidljivo u slučaju da imamo dva različita radara ili dva različita stanja na putu u suprotnom dugme, neće biti vidljivo i korisnik nema mogućnost pristupiti ovom zaslonu. Filteri rade na način da ukoliko su uključeni te oznake koje su uključene biti će i prikazane na mapi. Ukoliko se isključe neke od njih i izvrši se klik na „Primijeni“ dugme na odgovarajućem jeziku na vrhu ovog zaslona, podaci će biti poslani na prethodni zaslon sa mapom te će na mapi biti prikazani samo navedene oznake koje smo ostavili uključene. Oznake se mogu sve isključiti ali implementirano je da uvijek jedna mora biti uključena. Također ovaj zaslon se razlikuje od drugih jer se on prezentira preko zaslona za prikaz radara ili stanja na putu. Dok se ostali zasloni pomjeraju u lijevu stranu i redaju u red. Ukoliko se klikne na „Otkazi“ dugme na odgovarajućem jeziku, zaslon će se zatvoriti i korisnik će biti vraćen na prethodni zaslon sa uključenim filterima koje je imao uključene kada je ušao na filter zaslon.

## 4.7. Opis rada zaslona za prikaz detalja

Do ovoga zaslona se dolazi na dva različita načina. Prvi način jeste klikom na dugme  na zaslonu za prikaz stanja na putevima. U tom slučaju će biti prikazane opšte informacije o stanjima na putevima. To dugme nije vidljivo ukoliko nismo dobili nikakve full report ili opšte informacije sa API strane. Drugi način jeste kada se klikne na  dugme unutar kvadratića koji se pojavi kada se klikne na bilo koju oznaku stanja na putu ili radara na mapi.



Slika 16. Prikaz malog opisa stanja radara ili stanja na putu

Na zaslonu općih informacija se nalazi naslov, podnaslov i detaljan opis, dok na zaslonu za prikaz detaljnih informacija određenog radara imamo sljedeće


- Naslov
- Podnaslov - Inače predstavlja tip policijske uprave
  - MUP Zeničko-dobojskog kantona
  - MUP Kantona Sarajevo
  - MUP Unsko-sanskog kantona
  - MUP Tuzlanskog kantona
  - MUP Srednjobosanskog kantona
  - MUP Srednjobosanskog kantona
  - Policija Brčko distrikta
  - MUP Posavski
  - MUP Hercegovačko-neretvanskog kantona
  - MUP Bosansko-podrinjskog kantona
  - MUP Kantona 10 (Livno)
  - MUP Republike Srpske
- Tip radara
  - Stacionarni
  - Najavljeni
- Detaljan opis
- Vrijeme trajanja radara
- Broj prijava da radar nije aktiva

Za radare je bitno da ima naslov i tip, sva ostala polja su opcionalna.

Na zaslonu za prikaz detaljnih informacija stanja na putu imamo sljedeće

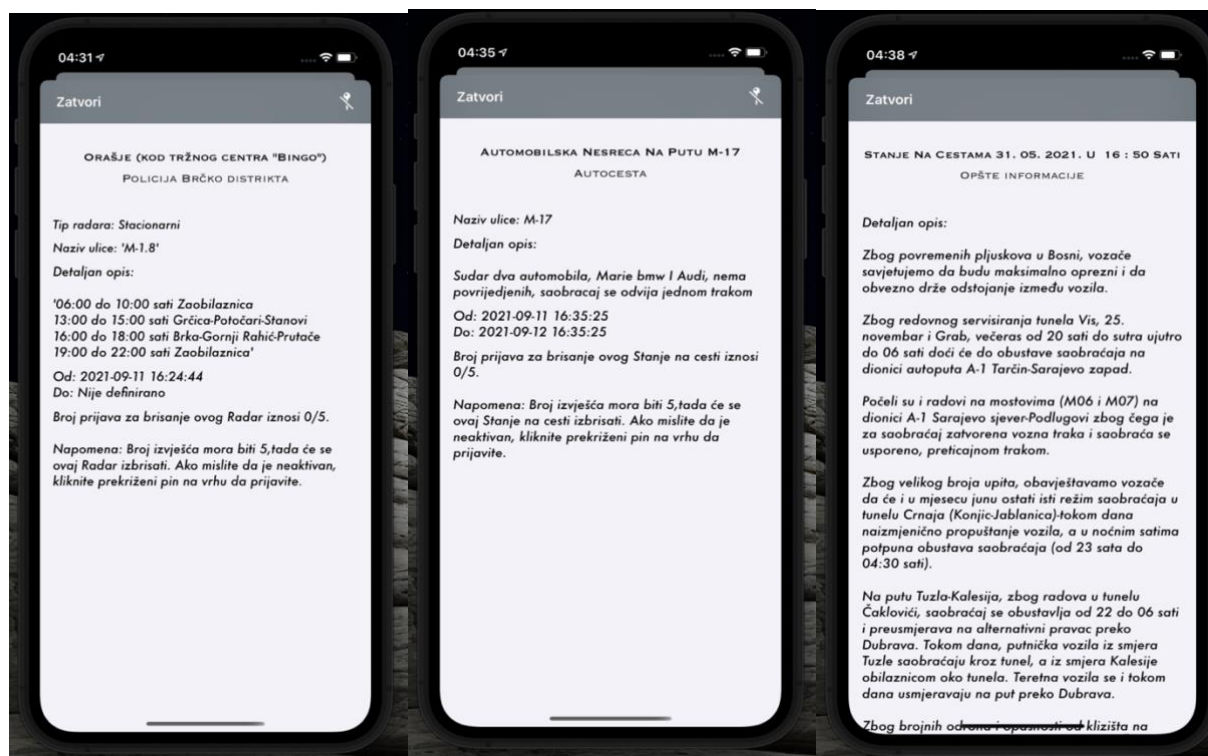
- Naslov
- Podnaslov - Inače predstavlja tip ulice na kojoj se nalazi stanje
  - Granični prijelaz
  - Autocesta
  - Magistralna cesta
  - Regionalna cesta
  - Gradska cesta
- Naziv ulice
- Detaljan opis
- Vrijeme trajanja stanja na putu
- Broj prijava da stanja na putu nije aktivan

Za stanja na putevima najvažnije je da objekt ima naslov, tip ulice i tip stanja koji se ne prikazuje na zaslonu sa detaljima ali definira koja će se slika prikazati na mapi.

Kao što se može primijetiti na zaslonima sa detaljima radara i stanja na putu se nalazi i napomena. Napomena govori da ukoliko korisnik smatra da radar nije aktivan ili stanje na putu nije prisutno da prijavi na način da se klikne na  dugme u desnom uglu novog zaslona. Klikom na ovo dugme aplikacija šalje odgovarajući endpoint prema Firestore bazi pri čemu se povećava broj prijava unutar tog objekta. Na ekranu će se pojaviti odgovarajuća poruka kao na Slici 18. koja nam govori da li je API poziv bio uspješan ili ne. Ukoliko API poziv jeste bio uspješan korisnik treba pritisnuti na OK dugme i može vidjeti da je broj prijava povećan za jedan.

Ukoliko broj prijava je veći od 5 nakon prijave, pojaviti će se odgovarajuća poruka na zaslonu da je radar ili stanje na putu obrisano, zaslon sa detaljima se zatvara i vrši osvježivanje mape, te će navedeni radar koji je prijavljen ili stanje na putu biti izbrisano.





a) Zaslona za prikaz detalja radar

b) Zaslona za prikaz detalja stanja na putu

a) Zaslona za prikaz detalja opšte informacije


**Slika 17. Prikaz zaslona za prikaz detalja radara, stanja na putevima i općih informacija**



**Slika 18. Uspješno prijavljen neaktivan radar ili stanje na putu**



## 4.8. Opis rada zaslona za prijavu novih radara ili stanja na putevima

Kao što je navedeno ranije korisnik u aplikaciji ima opciju i da prijavi nova stanja na putevima ili novi radar. Ukoliko je korisnik priključen na Internet imat će priliku da vidi  dugme koje je prisutno na zaslonima za prikaz radara i stanja na putevima te ima mogućnost da prijavi novi radar ili stanje na putu. Klikom na to dugme na mapi se pojavljuje jedna oznaka iznad naše trenutne lokacije. U trenutku prikaza navedene oznake na mapi što je prikazano na slici Slici 19. pojavljuju se i dva dugmeta tj. dugme „Prijavi“ i „Otkazi“, na odgovarajućem jeziku. Sada korisnik krećući se po mapi horizontalno ili vertikalno ima mogućnost da pozicionira navedenu oznaku na odgovarajuće mjesto gdje se nalazi nova lokacija radara ili stanja na putu. Ukoliko se klikne na „Otkazi“ dugme smatra se kao da je korisnik odustao od prijave radara ili stanja na putu, te se nakon toga navedena oznaka i dugmad uklone sa mape.

Ukoliko korisnik klikne na dugme „Prijavi“ sa mape na kojoj su prikazani radari ili sa mape sa stanjima na putu, korisniku se na zaslonu pojavljuje novi zaslon sa odgovarajućim poljima, a oni su prikazani na slici 20. Zajedničko na ova dva zaslona sa slike 20. jeste lokacija radara ili stanja na putu, naziv radara ili stanja na putu, ulica i detaljan opis. Lokacija predstavlja poziciju gdje je ostavljena oznaka sa prethodnog zaslona. Za naziv radara ili stanja na putu potrebno je unijeti neki kratki opis stanja ili naziv radara. Za polje Ulica se unosi naziv ulice, dok detaljni opis predstavlja neko detaljnije korisničko viđenje navedenog radara ili stanja na putu. Pored toga postoje polja „Tip radara“ ili „Tip ulice“ u zavisnosti da li se prijavljuje novi radar ili novo stanje na putu. „Tip radara“ može biti Stacionarni ili Najavljeni, dok „Tip ulice“ može biti

- Autocesta
- Granični prijelaz
- Gradska cesta
- Magistralna cesta
- Regionalna cesta

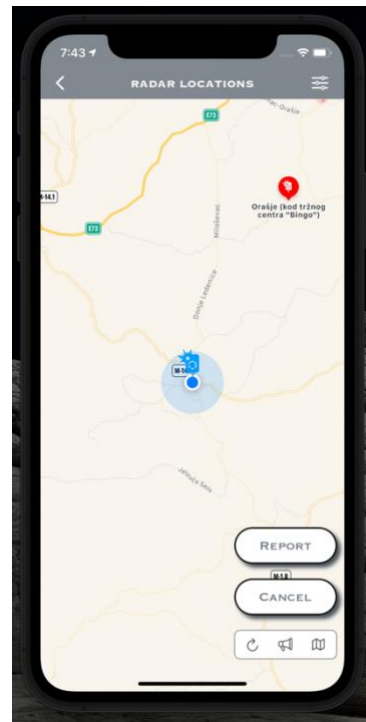
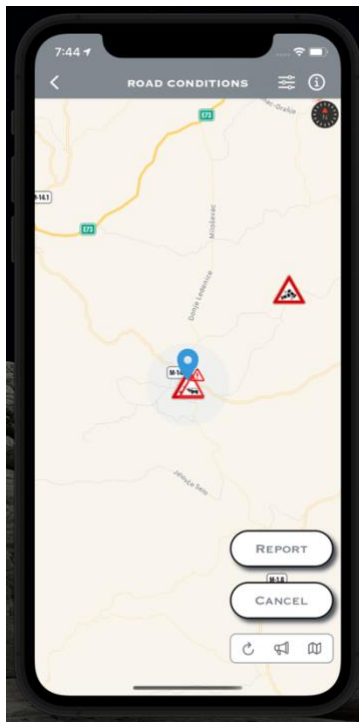
Pored toga na radarima se nalazi polje „Policijska uprava“ koje ima sljedeće vrijednosti:

- MUP Zeničko-dobojskog kantona
- MUP Kantona Sarajevo
- MUP Unsko-sanskog kantona
- MUP Tuzlanskog kantona
- MUP Srednjobosanskog kantona
- MUP Srednjobosanskog kantona
- Policija Brčko distrikta
- MUP Posavski
- MUP Hercegovačko-neretvanskog kantona
- MUP Bosansko-podrinjskog kantona
- MUP Kantona 10 (Livno)
- MUP Republike Srpske

Dok na stanjima na putevima se nalazi polje „Tip stanja na putu“ koje ima vrijednosti:

- Granični prijelaz
- Sanacija kolovoza
- Potpuna obustava prometa
- Zabrana za teretna vozila
- Zagušenje
- Odron
- Saobraćajna nezgoda
- Poledica
- Opasnost

„Tip stanja na putu“ ujedno definira ikonu znaka kojeg prikazujemo za to stanje na putu. „Policijska uprava“, „Tip stanja na putu“, „Tip radara“ i „Tip ulice“ predstavljaju komponente koje služe kao padajući meni. Korisnik da bi mogao prijaviti radar potrebno je da unese tip radara i naziv, dok da bi prijavio stanje na putu potrebno je da odabere tip ulice, naziv i tip stanja na putu koja predstavljaju obavezna polja za odgovarajuću prijavu. Ukoliko korisnik ne unese obavezna polja, a klikne na „Prijavi stanje na putu“ ili „Prijavi radar“ dugme koje se nalazi na dnu ekrana korisnik će dobiti odgovarajuću poruku u vidu upozorenja (Alert) da je potrebno da unese obavezna polja. Kada korisnik unese obavezna polja te klikne na „Prijavi stanje na putu“ ili „Prijavi radar“ dugme, pokreće se odgovarajući API poziv te korisnik dobiva poruku na ekranu da li je prijava bila uspješna ili ne. Ukoliko jeste bila uspješna zaslon za prijavu se zatvara, pokreće se spinner i osvježava se odgovarajuća mapa za prikaz tj. dohvataju se sa API - a stanja na putevima ili radari što je objašnjeno ranije.



Slika 19. Pozicija oznake za prijavu

03:56

REPORT RADAR

New radar location

44.87726000009701,18.425141600000018

Radar type

Select radar type

Insert radar title

Street

Police department

Select police department

Detailed description

REPORT RADAR

04:01

REPORT ROAD CONDITION

Location of the road conditions

44.877259999790226,18.425141600000018

Road type

Select road type

Insert title for the road condition

Street

Type of road condition

Select condition sign

Detailed description

REPORT ROAD CONDITION

Slika 20. Prikaz report zaslona

## 4.9. Unos podataka u Firestore uz pomoć skripti

Prilikom izrade ovog diplomskog rada za unos podataka u bazu napisane su odgovarajuće JavaScript skripte, koje priložene na kraju ovoga diplomskog rada, a također se nalaze u github firebase folderu od projekta. Za pokretanje istih potrebno je da imamo Node.js i NPM (Node Package Manager) instalirane na macOS uređaju. Node.js je okruženje koje uključuje sve što je potrebno za izvršavanje programa napisanog u JavaScript-u. NPM označava Node Package Manager, aplikaciju i spremište za razvoj i dijeljenje JavaScript koda. Da bi pokrenuli odgovarajuću skriptu potrebno je da pozovemo odgovarajuću komandu koje su navedene u nastavku. Skripte za dodavanje novih stvari u bazu kao parametar uzimaju json datoteku u kojoj se nalazi sadržaj koji dodajemo. Kada se proslijedi odgovarajuća json datoteka skripti za dodavanje, skripta prvo dohvatiti datoteku u vidu teksta te se vrši konverzija istog u niz odgovarajućih objekata bilo to da je niz radara ili niz stanja na putu ili samo objekat općih informacija. Nakon toga sve te objekte snimamo u odgovarajuće kolekcije unutar same Firestore baze. Također vršimo i brisanje objekata koji nisu više validni, te prisutne su i skripte za brisanje svih radara, stanja na putevima. Dok brisane općih informacija se vrši na način da u skriptu za dodavanje ne pošaljemo nikav parametar. Njima se ne šalju nikakvi parametri one unutar sebe imaju kodirano koje entitete treba isprazniti.

```
node addMainRoadConditionsInfo.js roadConditions16_30.json
node addRoadConditionsToFirestore.js roadConditionsAll.json
node addRadarsToFirestore.js radars.json
node deleteAllRadars.js
node deleteAllRoadConditions.js
node deleteExpiredRadars.js
```

## 5. Zaključak

Cilj ovog diplomskog rada jeste bila izrada IOS aplikacije za evidentiranje i praćenje radara i stanja na putevima korištenjem Swift programskog jezika. Iako Google Framework ima više opcija i dosta je zastupljeniji prilikom izrade aplikacije je odabran MapKit Framework za mapu zato što je Apple – ov Framework za prikaz mape te je i prirodnije ga koristiti. Ukoliko bi se u budućnosti odlučili da pravimo aplikaciju za Android, tada bi koristili Google Framework za mape. Aplikacija je veoma zanimljiva i ima puno opcija, moguće je koristiti na tri različita jezika Engleski, Njemački i Bosanski, različitim pozicijama telefona, te načinima rada.

Neki od planova u budućnosti za proširenje aplikacije jeste da se dodaju live kamere koje se nalaze na graničnim prelazima i većim petljama. Također bilo bi dobro uvesti koncept korisnika, npr. da se korisnici mogu prijaviti ukoliko žele da dodaju ili izbrišu neku lokaciju na mapi, jer trenutno svi korisnici mogu to raditi. Na taj način bi ograničili korisnicima koji nisu prijavljeni na našem servisu da brišu radare ili stanja na putevima, te bi oni korisnici koji nisu prijavljeni imali mogućost samo da pregledavaju radare, stanja na putevima. Također potrebno je prošiti klasu znakova, koje aplikacija podržava trenutno, na način da pokrivamo sve vrste prvenstveno znakova opasnosti i drugih znakova. Implementirati također način da se na dnevnoj bazi brišu stari radari ili stanja ukoliko je datum njihovog trajanja istekao.

Aplikaciju je moguće objaviti na Apple Store ali za to je potreban korisnički račun na Apple Developer Program - u koji košta 99.99\$ godišnje. Također ukoliko imamo korisnički račun, pošto Apple ima određena striktna pravila kakva aplikacija može biti objavljena postoji mogućnost da aplikacija bude odbijena prilikom objave, tako da u tom slučaju bi morali ispraviti određene pogreške koje je Apple našao tj. da ispunimo njihove zahtjeve.

## 6. Literatura

- [1] <https://github.com/eldarhaseljic/IOS-BosniaRoadTraffic>
- [2] <https://nodejs.org/en/>
- [3] <https://iOS.raywenderlich.com/7738344-mapkit-tutorial-getting-started>
- [4] <https://firebase.google.com/>
- [5] <https://developer.apple.com/xcode/>
- [6] <https://iOS.raywenderlich.com/1228891-getting-started-with-rxswift-and-rxcocoa>
- [7] <https://github.com/objcio/core-data/blob/master/SharedCode/NSManagedObjectContext%2BExtensions.swift>
- [8] <https://stackoverflow.com/questions/26056062/uiviewcontroller-extension-to-instantiate-from-storyboard>
- [9] <https://stackoverflow.com/questions/55964080/register-and-dequeue-uitableviewcell-of-specific-type>
- [10] <https://stackoverflow.com/questions/24034544/dispatch-after-gcd-in-swift>
- [11] <https://iOS.hackingwithswift.com/example-code/language/how-to-make-array-access-safer-using-a-custom-subscript>
- [12] <https://gist.github.com/cozzin/d6c45b51905d56f0bc8b097bb6aa4ce8>
- [13] <https://stackoverflow.com/questions/39348729/core-data-viewcontext-not-receiving-updates-from-newbackgroundcontext-with-nsf>
- [14] <https://stackoverflow.com/questions/30743408/check-for-internet-connection-with-swift/39782859#39782859>
- [15] <https://github.com/objcio/core-data/blob/master/SharedCode/Managed.swift>
- [16] <https://bihamk.ba/>
- [17] <https://betterprogramming.pub/5-ways-to-store-user-data-in-your-ios-app-595d61c89667>

## 7. Dodatak

### addRadarsToFirestore.js

```
1: // https://www.npmjs.com/package/date-and-time
2: const date = require('date-and-time');
3: // Required for side-effects
4: const firebase = require("firebase");
5: const fs = require('fs');
6: require("firebase/firestore");
7:
8: // Initialize Cloud Firestore through Firebase
9: firebase.initializeApp({
10:   apiKey: "AIzaSyCOQe97QzqHpQnkqvw5WGSC5SP0wcId0Gw",
11:   authDomain: "bosnia-road-traffic-fire-d6422.firebaseio.com",
12:   databaseURL: "https://bosnia-road-traffic-fire-d6422-default-rtdb.firebaseio.com",
13:   projectId: "bosnia-road-traffic-fire-d6422",
14:   storageBucket: "bosnia-road-traffic-fire-d6422.appspot.com",
15:   messagingSenderId: "882903880990",
16:   appId: "1:882903880990:web:024e2c0eddb40518d8250f",
17:   measurementId: "G-PQKF5RCX63"
18: });
19:
20: var db = firebase.firestore();
21: var currentDate = new Date()
22: const offset = currentDate.getTimezoneOffset()
23: currentDate = new Date(currentDate.getTime() - (offset*60*1000))
24: var validDateString = currentDate.toISOString().split('T')[0] + ' ' +
currentDate.toLocaleTimeString('en-GB')
25: var validToDate = date.parse(validDateString, 'YYYY-MM-DD hh:mm:ss', true);
26:
27: fs.readFile(process.argv[2], (err, data) => {
28:   if (err) throw err;
29:   console.log(process.argv[2])
30:   let radars = JSON.parse(data)
31:
32:   radars.forEach(function (obj) {
33:     db.collection("Radars").doc(obj.id).set({
34:       id: obj.id,
35:       title: obj.title,
36:       coordinates: obj.coordinates,
37:       type: obj.type,
38:       road: obj.road,
39:       valid_from: obj.valid_from,
40:       valid_to: obj.valid_to,
41:       text: obj.text,
42:       numberOfDeletions: obj.numberOfDeletions,
43:       category_id: obj.category_id,
44:       category_name: obj.category_name,
45:       updated_at: validToDate
46:     })
47:   });
48: });
49:
```

## addMainRoadConditionsInfo.js

```
1: // https://www.npmjs.com/package/date-and-time
2: const date = require('date-and-time');
3: // Required for side-effects
4: const firebase = require("firebase");
5: const fs = require('fs');
6: require("firebase/firestore");
7:
8: // Initialize Cloud Firestore through Firebase
9: firebase.initializeApp({
10:   apiKey: "AIzaSyCOQe97QzqHpQnkqvw5WGSC5SP0wcId0Gw",
11:   authDomain: "bosnia-road-traffic-fire-d6422.firebaseio.com",
12:   databaseURL: "https://bosnia-road-traffic-fire-d6422-default-rtdb.firebaseio.com",
13:   projectId: "bosnia-road-traffic-fire-d6422",
14:   storageBucket: "bosnia-road-traffic-fire-d6422.appspot.com",
15:   messagingSenderId: "882903880990",
16:   appId: "1:882903880990:web:024e2c0eddb40518d8250f",
17:   measurementId: "G-PQKF5RCX63"
18: });
19:
20: var db = firebase.firestore();
21: var currentDate = new Date()
22: const offset = currentDate.getTimezoneOffset()
23: currentDate = new Date(currentDate.getTime() - (offset*60*1000))
24: var validDateString = currentDate.toISOString().split('T')[0] + ' ' +
currentDate.toLocaleTimeString('en-GB')
25: var validToDate = date.parse(validDateString, 'YYYY-MM-DD hh:mm:ss', true);
26:
27: if (process.argv[2] !== null) {
28:   fs.readFile(process.argv[2], (err, data) => {
29:     if (err) throw err;
30:     console.log(process.argv[2])
31:     let roadConditions = JSON.parse(data)
32:
33:     roadConditions.forEach(function (obj) {
34:       db.collection("RoadConditionReport").doc('RoadConditionReport-
1111111').set({
35:         id: 'RoadConditionReport-1111111',
36:         title: "Stanje na cestama " + validDateString,
37:         startDate: validToDate,
38:         numberOfDeletions: null,
39:         endDate: null,
40:         text: obj.text,
41:         category_id: 1111111,
42:         category_name: "Op\u0161te informacije",
43:       })
44:       console.log(obj)
45:       console.log("Document with id: ", obj.id, ",successfully added!");
46:       console.log(validDateString)
47:     });
48:   });
49: } else {
50:   db.collection("RoadConditionReport").get().then((querySnapshot) => {
51:     querySnapshot.forEach((obj) => {
52:       db.collection("RoadConditionReport").doc(obj.id).delete().then(() => {
53:         console.log(obj.data())
54:         console.log("Document with id: ", obj.id, ",successfully deleted!");
55:       }).catch((error) => {
56:         console.error("Error removing document: ", error);
57:       });
58:     });
59:   });
60: }
61:
```



## addRoadConditionsToFirestore.js

```
1: // https://www.npmjs.com/package/date-and-time
2: const date = require('date-and-time');
3: // Required for side-effects
4: const firebase = require("firebase");
5: const fs = require('fs');
6: require("firebase/firestore");
7:
8: // Initialize Cloud Firestore through Firebase
9: firebase.initializeApp({
10:   apiKey: "AIzaSyCOQe97QzqHpQnkqv5WGSC5SP0wcId0Gw",
11:   authDomain: "bosnia-road-traffic-fire-d6422.firebaseio.com",
12:   databaseURL: "https://bosnia-road-traffic-fire-d6422-default-rtdb.firebaseio.com",
13:   projectId: "bosnia-road-traffic-fire-d6422",
14:   storageBucket: "bosnia-road-traffic-fire-d6422.appspot.com",
15:   messagingSenderId: "882903880990",
16:   appId: "1:882903880990:web:024e2c0eddb40518d8250f",
17:   measurementId: "G-PQKF5RCX63"
18: });
19:
20: var db = firebase.firestore();
21: var currentDate = new Date()
22: const offset = currentDate.getTimezoneOffset()
23: currentDate = new Date(currentDate.getTime() - (offset*60*1000))
24: var validDateString = currentDate.toISOString().split('T')[0] + ' ' +
currentDate.toLocaleTimeString('en-GB')
25: var validToDate = date.parse(validDateString, 'YYYY-MM-DD hh:mm:ss', true);
26:
27: fs.readFile(process.argv[2], (err, data) => {
28:   if (err) throw err;
29:   console.log(process.argv[2])
30:   let roadConditions = JSON.parse(data)
31:
32:   roadConditions.forEach(function (obj) {
33:     db.collection("RoadConditions").doc(obj.id).set({
34:       id: obj.id,
35:       icon: obj.icon,
36:       title: obj.title,
37:       coordinates: obj.coordinates,
38:       road: obj.road,
39:       valid_from: obj.valid_from,
40:       valid_to: obj.valid_to,
41:       numberOfDeletions: obj.numberOfDeletions,
42:       text: obj.text,
43:       category_id: obj.category_id,
44:       category_name: obj.category_name,
45:       updated_at: validToDate
46:     })
47:     console.log(obj)
48:     console.log("Document with id: ", obj.id, ",successfully added!");
49:   });
50: });
51:
```

deleteAllRadars.js

```
1: // Required for side-effects
2: const firebase = require("firebase");
3: require("firebase/firestore");
4:
5: // Initialize Cloud Firestore through Firebase
6: firebase.initializeApp({
7:   apiKey: "AIzaSyCOQe97QzqHpQnkqvw5WGSC5SP0wcId0Gw",
8:   authDomain: "bosnia-road-traffic-fire-
d6422.firebaseio.com",
9:   databaseURL: "https://bosnia-road-traffic-fire-d6422-
default-rtdb.firebaseio.com",
10:   projectId: "bosnia-road-traffic-fire-d6422",
11:   storageBucket: "bosnia-road-traffic-fire-
d6422.appspot.com",
12:   messagingSenderId: "882903880990",
13:   appId: "1:882903880990:web:024e2c0eddb40518d8250f",
14:   measurementId: "G-PQKF5RCX63"
15: });
16:
17: var db = firebase.firestore();
18:
19: db.collection("Radars").get().then((querySnapshot) => {
20:   querySnapshot.forEach((radar) => {
21:     db.collection("Radars").doc(radar.id).delete().then(()
=> {
22:       console.log(radar.data())
23:       console.log("Document with id: ", radar.id,
",successfully deleted!");
24:     }).catch((error) => {
25:       console.error("Error removing document: ", error);
26:     });
27:   });
28: });
29:
```

deleteAllRoadConditions.js

```
1: // Required for side-effects
2: const firebase = require("firebase");
3: require("firebase/firestore");
4:
5: // Initialize Cloud Firestore through Firebase
6: firebase.initializeApp({
7:   apiKey: "AIzaSyCOQe97QzqHpQnkqvw5WGSC5SP0wcId0Gw",
8:   authDomain: "bosnia-road-traffic-fire-
d6422.firebaseio.com",
9:   databaseURL: "https://bosnia-road-traffic-fire-d6422-
default-rtdb.firebaseio.com",
10:   projectId: "bosnia-road-traffic-fire-d6422",
11:   storageBucket: "bosnia-road-traffic-fire-
d6422.appspot.com",
12:   messagingSenderId: "882903880990",
13:   appId: "1:882903880990:web:024e2c0eddb40518d8250f",
14:   measurementId: "G-PQKF5RCX63"
15: });
16:
17: var db = firebase.firestore();
18:
19: db.collection("RoadConditions").get().then((querySnapshot) =>
{
20:   querySnapshot.forEach((roadConditions) => {
21:     db.collection("RoadConditions").doc(roadConditions.id).delete().th
en(() => {
22:       console.log(roadConditions.data())
23:       console.log("Document with id: ",
roadConditions.id, ", successfully deleted!");
24:     }).catch((error) => {
25:       console.error("Error removing document: ", error);
26:     });
27:   });
28: });
29:
```

## deleteExpiredRadars.js

```
1: // https://www.npmjs.com/package/date-and-time
2: const date = require('date-and-time');
3: // Required for side-effects
4: const firebase = require("firebase");
5: require("firebase/firestore");
6:
7: // Initialize Cloud Firestore through Firebase
8: firebase.initializeApp({
9:   apiKey: "AIzaSyCOQe97QzqHpQnkqvw5WGSC5SP0wcId0Gw",
10:   authDomain: "bosnia-road-traffic-fire-
d6422.firebaseio.com",
11:   databaseURL: "https://bosnia-road-traffic-fire-d6422-
default-rtdb.firebaseio.com",
12:   projectId: "bosnia-road-traffic-fire-d6422",
13:   storageBucket: "bosnia-road-traffic-fire-
d6422.appspot.com",
14:   messagingSenderId: "882903880990",
15:   appId: "1:882903880990:web:024e2c0eddb40518d8250f",
16:   measurementId: "G-PQKF5RCX63"
17: });
18:
19: var db = firebase.firestore();
20: var currentDate = new Date()
21: const offset = currentDate.getTimezoneOffset()
22: currentDate = new Date(currentDate.getTime() -
(offset*60*1000))
23: var validDateString = currentDate.toISOString().split('T')[0]
+ ' ' + currentDate.toLocaleTimeString('en-GB')
24: currentDate = date.parse(validDateString, 'YYYY-MM-DD
hh:mm:ss', true);
25:
26: db.collection("Radars").get().then((querySnapshot) => {
27:   querySnapshot.forEach((radar) => {
28:     if (radar.data().valid_to !== null) {
29:       var validToDate =
date.parse(radar.data().valid_to, 'YYYY-MM-DD hh:mm:ss', true);
30:       if (currentDate > validToDate ||
radar.data().numberOfDeletions >= 5) {
31:         db.collection("Radars").doc(radar.id).delete().then(() => {
32:           console.log("currentDate:", currentDate)
33:           console.log("validToDate:", validToDate)
34:           console.log(radar.data())
35:           console.log("Document with id: ",
radar.id, ",successfully deleted!");
36:         }).catch((error) => {
37:           console.error("Error removing document: ",
error);
38:         });
39:       }
40:     }
41:   });
42: });
43:
```