



# DIPLOMSKI RAD

PRVOG CIKLUSA STUDIJA

## RAZVOJ IOS APLIKACIJE ZA PRIKAZ RADARA I STANJA NA PUTEVIMA

STUDENT

HASELJIĆ ELDAR

TUZLA, SEPTEMBAR 2021. GODINE

UNIVERZITET U TUZLI  
FAKULTET ELEKTROTEHNIKE

Broj: \_\_\_\_\_  
Tuzla, \_\_\_\_\_ godine

**Haseljić Eldar**

Na osnovu Vašeg zahtjeva i izdate teme na predmetu " \_\_\_\_\_"  
\_\_\_\_\_ " izdaje Vam se slijedeći pismeni završni zadatak:

**“Razvoj IOS aplikacije za prikaz radara i stanja na putevima”**

Na osnovu člana 192. Statuta Univerziteta u Tuzli, završni rad možete braniti kada ispunite sve uslove predviđene Nastavnim planom i programom prvog ciklusa studija, Statutom i drugim opštim aktima Univerziteta.

Dekan

.....

Dr. Sc. Dekan fakulteta, vanr. Prof.

## **Sažetak**

*U ovom diplomskom radu ćemo govoriti o razvoju IOS aplikacije za prikaz radara i stanja na putevima korištenjem programskog jezika Swift i njenoj upotrebi. 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, također ima i opciju za filtriranje 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 prevedena na tri jezika Njemački, Engleski i Bosanski*

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

# Abstract

*In this graduate work, we will talk about the development of IOS application for displaying radar and road conditions using the programming language Swift and its use. MapKit was used within the application for the type of 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 has the option of adding new ones and has the option to filter and delete radar locations or road conditions. The data is stored in a Cloud Firestore database, which is a flexible, scalable, NoSQL database used in the development of mobile, web applications. The application has been translated into three languages: German, English and Bosnian*

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

## **Sadržaj**

<b>1. Uvod.....</b>	<b>6</b>
<b>2. Razvoj IOS mobilne aplikacije .....</b>	<b>7</b>
2.1. Reaktivno programiranje .....	7
2.1.1. RxSwift i RxCocoa Framework .....	8
2.2. Swift programski jezik .....	9
2.3. Xcode.....	9
2.3.1. Kreiranje Xcode projekta .....	10
2.4. iOS Simulator .....	10
2.5. CocoaPods .....	11
2.6. Upotreba baza podataka .....	13
2.6.1. UserDefaults .....	14
2.6.2. Keychain.....	14
2.6.3. File system.....	14
2.6.4. Sqlite.....	15
2.6.5. Core Data.....	15
2.6.6. Firebase / Firestore .....	16
2.7. Model – View – ViewModel (MVVM) .....	17
<b>3. Struktura aplikacije .....</b>	<b>18</b>
<b>4. Korisničko okruženje aplikacije .....</b>	<b>21</b>
<b>5. Implementacija i način upotrebe iOS aplikacije za praćenje stanja na putevima.....</b>	<b>22</b>
5.1. Opis rada glavnog zaslona.....	22
5.2. Opis rada zaslona za prikaz radara .....	24
5.3. Opis rada zaslona za prikaz stanja na putevima .....	26
5.4. Opis rada zaslona za filtriranje .....	28
5.5. Opis rada zaslona za prikaz detalja .....	29
5.6. Opis rada zaslona za prijavu novih radara ili stanja na putevima .....	32
5.7. Unos podataka u firebase uz pomoć skripti.....	35
<b>6. Zaključak .....</b>	<b>36</b>
<b>7. Literatura.....</b>	<b>37</b>

## **Popis slika**

Slika 1. Prikaz iOS Simulator - a .....	11
Slika 9. MVVM (Model – View- ViewModel) arhitektura .....	17
Slika 3. Struktura aplikacije .....	19
Slika 4. Zaslون pokretanja .....	21
Slika 5. Glavni zaslon.....	21
Slika 6. Prikaz ostalih screenova koji su vidljivi u aplikaciji.....	21
Slika 7. Prikaz upozorenja prilikom klika na LinkedIn.....	23
Slika 8. Zaslون sa prikazanim radarima .....	25
Slika 9. Zaslون sa prikazanim radarima, mapa teren.....	25
Slika 10. Prikaz funkcionalnosti na zaslonu za prikaz radara na putevima .....	25
Slika 11. Prikaz zaslona za prikaz stanja na putevima .....	26
Slika 12. Prikaz funkcionalnosti na zaslonu za prikaz stanja na putevima .....	26
Slika 13. Prikaz svih znakova korištenih za prikaz stanja i radarana mapi.....	27
Slika 14. Zaslون za filtriranje stanja na putevima i zaslon za fitriranje radara .....	28
Slika 15. Prikaz malog opsia stanja radara ili stanja na putu .....	29
Slika 16. Prikaz zaslona za prikaz detalja radara, stanja na putevima i općih informacija .....	31
Slika 17. Uspješno prijavljen neaktivan radar ili stanje na putu .....	32
Slika 18. Pozicija oznake za prijavu.....	33
Slika 19. Prikaz report zaslona .....	33

# 1. Uvod

Zbog povećanja ljudskog stanovništva dolazi i do povećanja vozila na putevima. Prilikom povećanja broja vozila, javljaju se i povećane gužve na svim putevima. Kako bi izbjegli te gužve potrebno nam je nešto što bi nam govorilo 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., dok to nešto bi nam bila aplikacija i to mobilna aplikacija.

Odlučio sam se 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 mi je bilo da korisnik ima mogućnost, pored pregleda, prijavljivanja novih radova, radara ili drugih dešavanja na putevima i da ima mogućnost da uklanja neke radove koji više nisu aktivni na istim lokacijama. U svijetu danas imamo 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 ja sam se odlučio da to bude IOS aplikacija čisto iz razloga da utvrdim i pokažem svoje znanje iz oblasti Swift programskog jezika (programski jezik za programiranje IOS aplikacija).

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

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

## **2. Razvoj IOS mobilne aplikacije**

Za razvoj konkretno ove aplikacije koja se obrađuje u ovom radu su korišteni razni alati i tehnologije.

Za izradu navedene aplikacije potrebno je bilo sljedeće:

- MacOS računalo ili virtualna mašina koja pokreće MacOS
- Xcode
- iOS SDK (eng. software development kit)
- Poznavanje Swift programskog jezika
- Poznavanje načina upotrebe RxSwift i MapKit Framework - a
- Realni uređaj za testiranje – opcionalno

Za početak ću Vam opisati 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.

Cijelo 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 dohvaćamo i obrađujemo podatke koje prikazujemo.
- 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 – pomoću njega postavljamo koja akcija će se izvršavati na kojem Threadu. Tako da se sva promatranja protoka podataka izvršavaju na Main threadu, dok se sve pretplate izvršavaju na Background threadu.
- DisposeBag – predstavlja dodatni alat iz RxSwift – a koji nam pomaže da ukinemo pretplatu sa protoka podataka. Mi to želimo otkazati kada se pozove 'deinit()' tzv. destruktorkontrolera. To radimo na način da stvorimo objekt 'DisposeBag' i priložimo ga objektu pretplate pomoću metode 'disposed(by:)', koja će automatski otkazati pretplatu umjesto nas kada se kontroler odbaci odnosno ukloni. Bez ovoga moguće je da dobijemo 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, vršimo na način tako što nad objektom odgovarajuće klase pozovemo 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 dodamo sljedeće dvije linije koda:

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

**Kod 1. Komande za importiranje Rx biblioteka unutar Podfile datoteke**

Dok u fajlovima u kojim želimo da koristimo nešto iz tih biblioteka potrebno je da pozovemo sljedeće dvije linije koda:

```
import RxSwift  
import RxCocoa
```

**Kod 2. Komande za importiranje Rx biblioteka unutar određenog fajla**

## **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 iOS i OS X. Prvi put 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-ovim 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-ove proizvode u prethodnim desetljećima.

Swift koristi LLVM (Low Level Virtual Machine) kompajler za kompajliranje koda koji dolazi sa Xcode 6 te koristi Objective-C runtime. Stoga, unutar jednog programa možemo 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. Pružamo binarne datoteke za macOS i Linux koje mogu kompilirati kod za iOS, macOS, watchOS, tvOS i Linux. Swift za Linux je prisutan od Swift 2.2 (Mart 2016) verzije, dok je za Windows prisutan tek od Swift 5.3 (Septembar 2020) verzije. Najnovija verzija ovog programskog jezika je Swift 5.5.

## **2.3. Xcode**

Xcode je razvojna okolina (eng. integrated development environment - IDE) koju koristimo za razvoj macOS, iOS, iPadOS, watchOS i tvOS aplikacija. Prvi put je objavljen 2003 godine, dok je najnovije stabilno verzija 13.1, objavljena 25. Oktobra 2021. godine, a dostupno je putem Mac App Store. Xcode sadrži editor, kompajler, emulator, razvojne Framework-e i još puno elemenata koja će nam olakšati razvoj gore navedenih aplikacija. Možemo 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 smo preuzeli Xcode sa Mac App Store - a potrebno je da kreiramo projekat. Projekat kreiramo na način da odaberemo opciju „Create a new Xcode project“ na početnom ekranu kada otvorimo XCode. Stim da u ovom diplomskom radu mi želimo iOS aplikaciju potrebno je da odaberemo Multiplatform -> App. Nakon toga unosimo „Product Name“ što predstavlja naziv naše aplikacije. Nakon toga potrebno je da odaberemo „Team“, naši timovi za App Store Connect bi se trebali pojaviti ako smo povezali Xcode sa svojim Apple ID - om. Dok u slučaju ako imamo besplatni račun programera, tu ćemo odabrati Lični tim, što je učinjeno u slučaju naše aplikacije. Potrebno je odabrati „Interface“, tj. biramo između Storyboard (UIKit) ili SwiftUI. Potrebno je odabrati „Life Cycle“, tj. biramo između UIKit App Delegate i SwiftUI App- Potrebno je odabrati „Language“, tj. biramo između Objective-C ili Swift. Ovo se odnosi koji ćemo jezik koristiti unutar naše aplikacije. Za novije verzije Xcode ova opcija je postavljena automatski na Swift. Također imamo i dodatne opcije, koje nam dodaju određeni zadani kod našoj 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. Kada unesemo sve potrebne podatke i dodatne opcije uspješno smo kreirali projekat i možemo početi sa radom.

## **2.4. iOS Simulator**

iOS simulator je alat koji nam služi kako bi pokrenuli i testirali aplikaciju na našem računaru. Simulator omogućuje simulaciju za većinu Apple uređaja te podržava opcije koje ti uređaji imaju. Simulator 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 i simulator je mogao i uspio odraditi sve potrebne zahtjeve korisnika. Aplikacija je također testirana na fizičkom uređaju Iphone 8, OS verzija 13.7.



Slika 1. Prikaz iOS Simulator - a

## 2.5. CocoaPods

CocoaPods predstavlja menadžera koji upravlja bibliotekama koje koristimo u našim Xcode projektima. Biblioteke koje koristimo u našim projektima navedene su u jednoj tekstualnoj datoteci pod nazivom Podfile. CocoaPods će riješiti zavisnosti između biblioteka, dohvatiti izvorni kod, a zatim ga povezati u Xcode radni prostor za izgradnju našeg projekta. CocoaPods je izgrađen s Ruby-em i bit će instaliran sa zadanim Ruby-om dostupnim na macOS-u. Da bi smo mogli koristiti CocoaPods u našem projektu potrebno je da pokrenemo sljedeću komadnu u terminalu

```
sudo gem install cocoapods
```

Kod 3. Komanda za instalaciju cocoaPods

# UNIVERZITET U TUZLI

## FAKULTET ELEKTROTEHNIKE

```
pod init
```

### Kod 4. Komanda za inicijalizaciju Podfile datoteke

Nakon što smo instalirali CocoaPods potrebno je da napravimo Podfile unutar našeg projekta. On predstavlja datoteku koja sadrži dodatne biblioteke i specifikacije vezane za projekta. To možemo izvršiti uz pomoć komande navedene u Kod 4. sekciji. Prva linija koda u kreiranom fajlu je “platform :ios, '12.0'” te ona predstavlja platformu i najnižu verziju koju podržavamo.

```
# Uncomment the next line to define a global platform for your project
platform :ios, '12.0'

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 = '12.0'
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
  project.build_configurations.each do |bc|
    bc.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = deployment_target
  end
end
```

### Kod 5. Sadržaj Podfile datoteke BosniaRoadTraffic aplikacije

Da bismo koristili CocoaPods biblioteke u određenom projektu, moramo definirati Xcode target s kojim ćemo ih povezati unutar Podfile datoteke. Na primjer, u gore navedenom fajlu za našu iOS aplikaciju, to bi bio naziv naše aplikacije.

U našoj Podfile datoteci imamo target sekciju u kojoj piše „target 'BosniaRoadTraffic' do“ nakon te linije koda slijede takozvani CocoaPod - ovi koji predstavljaju dodatne biblioteke koje ćemo koristiti u našem projektu. One se navode u formatu „pod 'PODNAME'“.

Nakon navedenih svih biblioteka slijedi end komanda na kraju. Unutar Podfile datoteke možemo navesti i određene dodatne specifikacije kao što smo mi uradili gore u našem Podfile-u. Dodatni kod izvršava postavljanje „deployment\_target” - a za sve biblioteke na 12.0.

Nakon što smo kreirali sadržaj našeg Podfile datoteke potrebno je da izvršimo komade koje će instalirati sve navedene biblioteke u naš projekat i izvršiti nadogradnju na njihove najnovije verzije. Također te komade kreiraju MyApp.xcworkspace fajl koji koristimo dalje u razvoju naše aplikacije.

```
pod install  
pod update
```

**Kod 6. Komade za instaliranje i nadogradnju biblioteka koje su navedene u Podfile**

## 2.6. Upotreba baza podataka

Prvo objasnimo neke od poznatih načina pohranjivanja i dohvaćanja podataka, te ćemo dati njihov kratki opis. Najpoznatiji načini su:

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

### 2.6.1. UserDefaults

Počnimo s najjednostavnijim među pohranama podataka, **UserDefaults**. Ovo je najčešći i najudobniji način pohrane i dohvaćanja informacija. **UserDefaults** rade kao pohrana ključ/vrijednost, s nizovima kao ključevima. Kao što im naziv implicira, **UserDefaults** trebali bismo 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 primiti dnevni podsjetnik, želi li zapravo primiti obavijesti itd.).

### 2.6.2. Keychain

Neke aplikacije moraju pohraniti privatne podatke ili podatke koje želimo zaštititi: lozinke, certifikate i slične stavke, te kad god trebamo pohraniti male dijelove informacija koje moramo šifrirati i čuvati upotreba **Keychain** - a bi dobro došla. Uzmite u obzir da se sve mora pohraniti kao Data. Stoga moramo proslijediti samo objekte i vrijednosti koji se mogu serijalizirati u taj tip.

### 2.6.3. File system

Kada ono što trebamo 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: 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 možemo pohraniti podatke koje generira korisnik.
- Documents/Inbox
  - Ovo je posebna mapa koju kreira sistem kad god druga aplikacija zatraži od naše aplikacije da otvori datoteku. Ova je mapa samo za čitanje iz perspektive aplikacije. Program Mail, na primjer, postavlja privitke e-pošte povezane s vašom aplikacijom u ovaj direktorij.

- Library
  - Ovo je mapa u koju mi, kao programeri, možemo pohraniti datoteke koje aplikacija zahtijeva. Na primjer, možemo pohraniti binarnu datoteku koja može pružiti podatke aplikaciji.
- Library/Caches
  - Ovo je posebna mapa gdje možemo spremiti datoteke koje bi nam mogle uskoro zatrebati, ali ne marimo za gubitak. Kada prestanemo koristiti te datoteke, sistem će ih izbrisati. Slike koje preuzimamo radi bržeg učitavanja aplikacije trebale bi završiti u ovoj mapi.
- tmp
  - Još jedna posebna mapa gdje pohranjene datoteke ne ostaju tijekom pokretanja aplikacije. Sistem bi mogao očistiti obrisati ovu mapu.

**File system** dobar je kandidat za pohranu slika koje preuzimamo s weba i koje želimo držati pri ruci kako bismo skratili vrijeme učitavanja.

## 2.6.4. Sqlite

Više od mjesta za pohranu naših podataka, **SQLite** je de facto standard za interne baze podataka. SQL baza podataka zahtijeva od nas da svoje modele organiziramo u tablice. Zatim će se pobrinuti za njihovo pohranjivanje i dohvaćanje na disk na najučinkovitiji mogući način. Osim mogućnosti pohrane, posjedovanje SQL baze podataka omogućuje učinkovito pronalaženje i mogućnost pisanja složenih upita za izvlačenje zanimljivih uvida.

## 2.6.5. Core Data

**Core Data** je okvir za objektno-relacijsko mapiranje (ORM) koji dolazi s iOS-om. Kao i SQLite, **Core Data** nije mjesto za pohranjivanje podataka, već način da se njima manipulira. Možemo na njega gledati kao na evoluciju SQLitea. **Core Data** možemo uključiti u naš projekat obavirom **Core Data** opcije prilikom kreiranja samog projekta. **Core Data** smo koristili u našem projektu da bi smo spremili podatke na mobitel tako da bi korisnik imao pristup zadnjim informacijama dohvaćenih sa interneta i u slučaju kada nije konektovan na internetsku mrežu.



## **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: Cloud Firestore, ML Kit, Cloud Functions, Authentication, Hosting, Cloud Storage, Realtime Database.

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 bez poslužitelja.

U ovome radu korištena je samo Cloud Firestore funkcionalnost Firebase-a. Firestore u aplikaciji je korišten za spremanje pozicija radara, stanja na putevima, te ostalih informacija i detalja koji oni nose sa sobom. Korišten je iz razloga jer sam htio da imam jednu bazu na serveru kojoj bi svi korisnici pristupali za dohvaćanje, dodavanje ili brisanje određenih podataka.

Da bismo mogli koristiti Firebase, treba nam samo Google račun. Nakon što se prijavimo na Firebase s našim računom, potrebno je da kreiramo projekat na Firebase koji također moramo uvezati sa našim Xcode projektom.

Nakon što kreiramo projekat na Firebase.com, dobivamo sljedeći fajl pod nazivom „GoogleService-Info.plist“ koji je potrebno da dodamo u naš Xcode projekat. Također je potrebno da se dodaju odgovarajuće biblioteke u Podfile koji smo ranije naveli i pozove „pod update“ komanda. Pozovemo `Firebase.configure()` unutar `class AppDelegate` u dole navedenoj funkciji i dodamo liniju `import Firebase` van klase. Tim postupkom Firestore je uspješno podešen.

```
pod 'FirebaseFirestoreSwift'  
pod 'Firebase'  
pod 'Firebase/Analytics'
```

**Kod 7. Komande za importovanje Firebase biblioteka unutar Podfile datoteke**

```
import Firebase

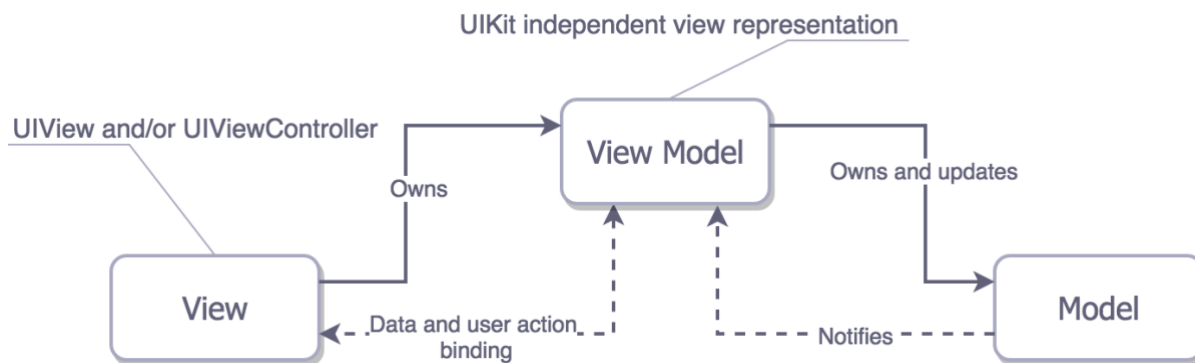
@main
class AppDelegate: UIResponder, UIApplicationDelegate {

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

**Kod 8. Konfiguracija Firebase baze unutar projekta**

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

Model-View-ViewModel je način dizajna programske arhitekture. Arhitektura programske podrške je plan koji opisuje skup aspekata i odluka koji su od iznimne važnosti za projekat. 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 vidimo iz naziva, arhitektura se sastoji od tri dijela:



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

- ViewModel
  - Sadrži poslovnu logiku aplikacije i sastoji se od različitih programskih dijelova ovisno o složenosti aplikacije.
- View
  - sadrži kod za prikaz vizualnih elemenata korisničkog okruženja i reakcije na korisničke akcije. Sastoji se od protokola ili evenata koji sadrže metode korisnikovih akcija i ViewController koji je odgovoran za materijalizaciju i prikazivanje komponenti korisničkog okruženja te otkrivanje događaja koje prosljeđuje viewModel-u.
- Model
  - Sadrži podatke aplikacije. Najčešće se radi od klasama ili strukturama.

### 3. Struktura aplikacije

Sama aplikacija je pisana uz pomoć MVVM arhitekturom, a na slici 3. možemo vidjeti strukturu same aplikacije. Svaka funkcionalnost je odvojena u zasebnu datoteku te svaka funkcionalnost ima svoj View, Model i ViewModel. Osim funkcionalnosti postoji i Manager koji nam služi za postavljanje 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. Metode AppDelegate 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 i razmotrimo ih:

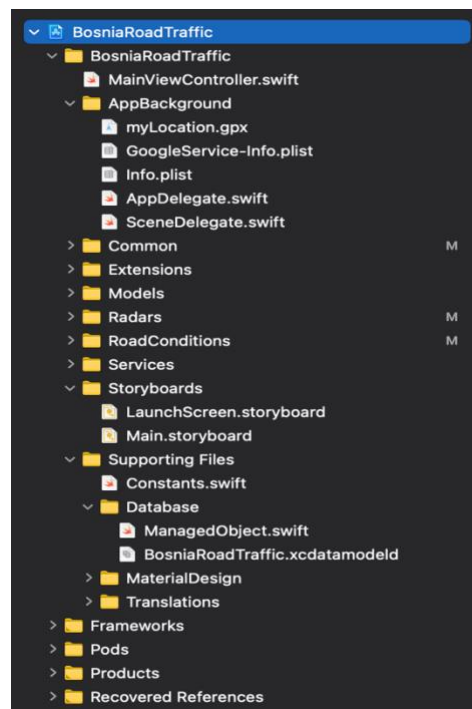
- **func application(\_: didFinishLaunchingWithOptions :) -> Bool**
  - Ova metoda se poziva kada se aplikacija pokrene i gdje se vrši postavljanje aplikacije. Tu smo izvršili konfiguraciju Firebase baze podataka.

# UNIVERZITET U TUZLI

## FAKULTET ELEKTROTEHNIKE

---

- `func application(_: configurationForConnecting: options :)`  
`-> UISceneConfiguration`
  - Ova metoda se poziva kada aplikaciji za prikaz bude potrebna nova scena ili prozor. Ova se metoda 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.



Slika 3. Struktura aplikacije

Od verzije 13 pa nadalje, **SceneDelegate.swift** preuzima neke odgovornosti od **AppDelegate.swift**. Konkretno vezano za **UIWindow** iz **AppDelegate.swift** sada je **UIScene** u **SceneDelegate**. Aplikacija može imati više scena koje uglavnom obrađuju sučelje aplikacije i sadržaj aplikacije. Dakle, **SceneDelegate.swift** je odgovoran za ono što je prikazano na ekranu u terminima korisničkog sučelja i podataka.

# UNIVERZITET U TUZLI

## FAKULTET ELEKTROTEHNIKE

---

Unutar naše aplikacije nismo ništa mijenjali vezano za ovaj fajl, tako da je njegov sadržaj ostao isti onakav kakav bude kada se kreira sam porjekat. Samo ćemo nabrojati njegove zadane metode, te objasniti ih u par recenica, a te metode su:

- **scene(\_:willConnectTo:options:)**
  - Ovo je prva metoda koja se poziva u životnom ciklusu UIWindowSession. Ova metoda će stvoriti novi UIWindow, postaviti glavni kontroler pogleda i učiniti ovaj prozor ključnim prozorom za prikaz.
- **sceneWillEnterForeground(\_:)**
  - Ova metoda se poziva kada se scena treba pokrenuti, primjerice kada aplikacija postane aktivna po prvi put ili pri prijelazu iz pozadine u prednji plan.
- **sceneDidBecomeActive(\_:)**
  - Ova metoda se poziva odmah nakon metode WillEnterForeground i ovdje je scena postavljena i vidljiva i spremna za upotrebu.
- **sceneWillResignActive(\_:)**
- **sceneDidEnterBackground(\_:)**
  - Ove se metode pozivaju pri stupnju aplikacije u pozadinu.
- **sceneDidDisconnect(\_:)**
  - Kad god se scena pošalje u pozadinu, iOS bi mogao odlučiti potpuno odbaciti scenu kako bi oslobodio resurse. To znači da je scena isključena iz sesije i nije aktivna. iOS može odlučiti ponovno povezati ovu scenu sa scenom kad korisnik tu scenu ponovno dovede u prvi plan. Ova se metoda može koristiti za odbacivanje svih resursa koji se više ne koriste.

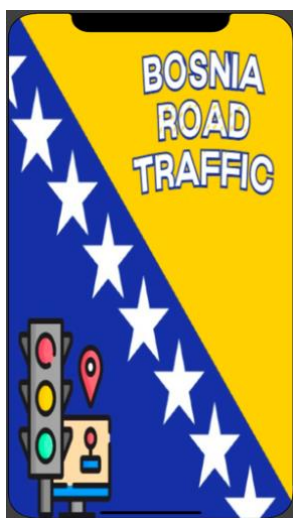
Pored navedena dva glavna fajla za pokretanje aplikacije koji se nalaze unutar AppBackground foldera kao što možemo vidjeti na slici 3. U strukturi su također važni :

- *BosniaRoadTraffic.xcdatamodeld*
  - *CoreData baza podataka na mobitelu*
- *GoogleService-Info.plist*
  - *Sadrži sve podatke potrebne Firebase iOS SDK -u za povezivanje s vašim Firebase projektom*
- *Info.plist*
  - *To je popis karakteristika čiji parovi ključ / vrijednost navode bitne informacije o konfiguraciji vremena izvođenja za aplikaciju.*

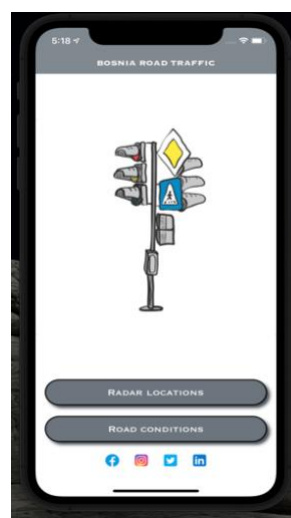
*Ostale fajlove i datoteke ćemo proći u narednim stranicama*

## 4. Korisničko okruženje aplikacije

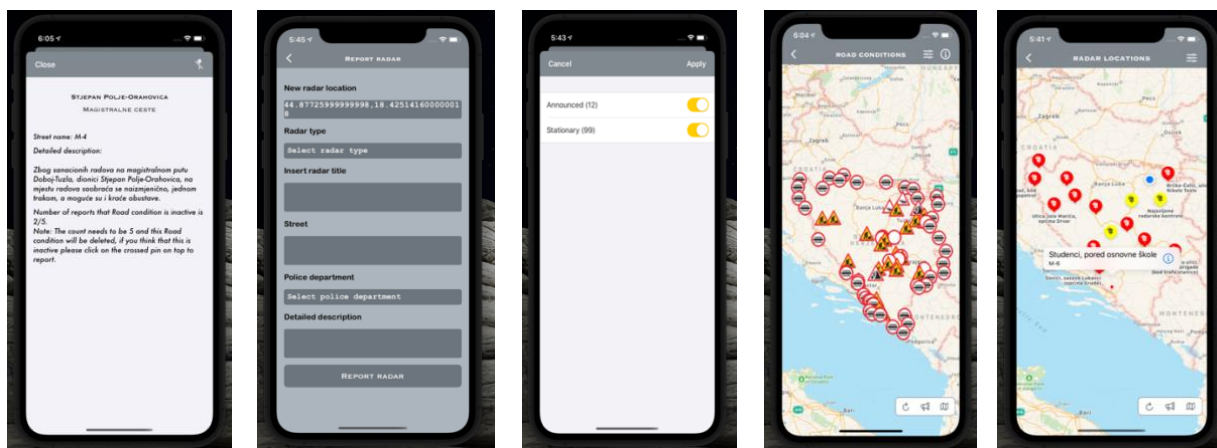
Aplikacija razvijena u diplomskom radu podržava sve dimenzije iPhone uređaja. Također ima podršku korištenja aplikacije u vodoravnom (landscape) ili vertikalnom (portrait) načinu rada. Korisničko okruženje je razvijeno pisanjem programskog koda u Swiftu i uz pomoć Storyboard-ova. Storyboard je vizuelni prikaz korisničkog okruženja iOS aplikacije, koji prikazuje zaslone sadržaja i veze između tih ekrana.



Slika 4. Zaslون pokretanja



Slika 5. Glavni zaslon



Slika 6. Prikaz ostalih screenova koji su vidljivi u aplikaciji

Aplikacija ima nekoliko zaslona, a to su:

- Zaslon 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*

U nastavku rada ćemo razraditi samo detaljno sve funkcionalnosti svakog zaslona.

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

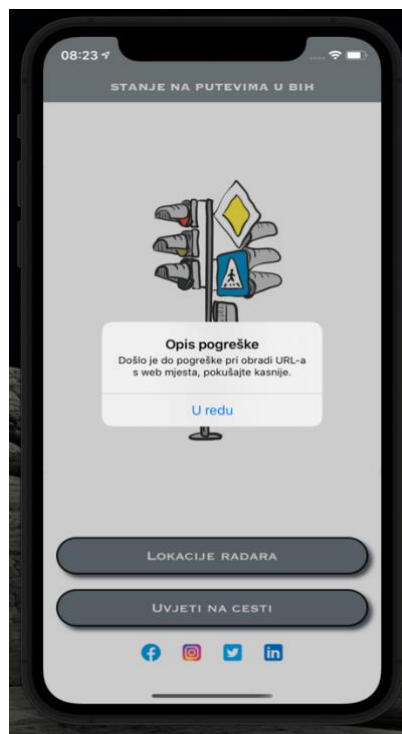
### 5.1. Opis rada glavnog zaslona

Kada korisnik pokrene aplikaciju na svom iPhone uređaju prva stvar na koju nailazi jeste zaslon pokretanja koji je prikazan na Slici 4. U suštini to je zaslon koji dobijemo kada kreiramo sam projekat. Razlika između zadane vrijednosti koju dobijemo prilikom kreiranja projekta i našeg zaslona pokretanja (*LaunchScreen.storyboard*) jeste što se na našem nalazi slika, dok na zadanoj vrijednosti imamo samo bijelu pozadinu ili crnu.

Da li vidimo crnu ili bijelu pozadinu zavisi da li je tamni način rada uključen. Nakon što se aplikacija učita dolazimo na glavni zaslon (**MainScreen.storyboard**) koji je na prikazan na Slici 5. Na glavnom zaslonu možemo primijetiti na vrhu zaslona da ima naziv, dva siva dugmeta, te nekoliko sličica društvenih mreža koje također predstavljaju dugmad.

Ukoliko stisnemo na neku od sličica društvenih mreža, otvoriti će nam se zadani vanjski pretraživač na mobitelu (Safari inače) sa zadanom društvenom mrežom. Sve sličice društvenih mreža otvaraju društvenu mrežu osim LinkedIn sličice koja namjerno ima pogrešno podešen link, koji otvara upozorenje (Alert) gdje imamo informaciju da je link krivo podešen. To je urađeno samo za prikaz šta bi se desilo da je Link krivo podešen.

Svaki klik na bilo koje navedeno dugme je izvršen uz pomoć tzv. Reaktivnog wrappera za TouchUpInside event nad dugmetom (rx.tap). Na njega pretplaćujemo uz pomoć bind funkcije, gdje unutar nje pišemo šta želimo da se desi u slučaju klika na dugme. U slučaju da dođe do greške, bit će pokrenuta fatalError funkcija koja ruši aplikaciju, tako da je potrebno je koristiti sa oprezom.



**Slika 7. Prikaz upozorenja prilikom klika na LinkedIn**



## **5.2. Opis rada zaslona za prikaz radara**

Ukoliko stisnemo na gornje sivo dugme na glavnom zaslonu na kojem piše „Lokacije radara“ na odgovarajućem jeziku, pokreće se spinner za učitavanje te prvo na što ćemo naići jeste Alert koji od nas traži da dopustimo prikaz naše lokacije na mapi, gdje korisnik može odabrati da uvijek ili samo jednom dopušta prikaz lokacije ili odbije prikaz lokacije.

Ukoliko korisnik odbije prikaz lokacije biće vraćen automatski na glavni tj. početni zaslon. Ukoliko je dopustio 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 bazu na mobitel. Nakon što su radari učitani oni se prikazuju na mapi kao na slici 8. u gornjem desnom uglu korisnik ima opciju da filtriranje radara. Radari se mogu filtrirati samo na dva načina tj. stacionarni ili najavljeni radari. Ukoliko su vidljivi na mapi samo jedni ili nijedni onda je opcija za filtriranje isključena tj. nije vidljiva. O filtriranju radara ćemo malo detaljnije u nastavku rada.

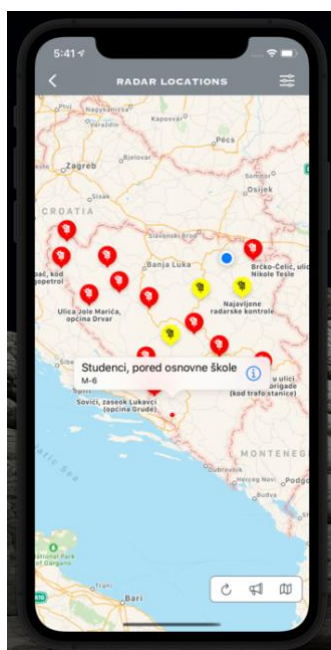
Pored opcije za filtriranje u desnom donjem uglu imamo tri dodatne opcije tj. dugmeta. Prvo dugme jeste za ponovno učitavanje zaslona, pri čemu osvježimo radare tj. učitamo nove ili se izbrišu određeni ako nisu više u Firebase bazi. Pored nje imamo opciju za prijavu tj. ako smo uočili novi radar možemo ga prijaviti. O tome također detaljnije u nastavku. Zadnja opcija jeste promjena tipa mape tj. imamo mapu kao što je prikazana na slici 8 i mapu sa terenom koja je prikazana na slici 9. Ukoliko radara nemamo ili ako smo dobili informacije o novim radarima ili ako se desi neka pogreška uvijek ćemo dobiti odgovarajući Alert, tipa kao na slici 7.

Radari na mapi su označeni sa crvenom ili žutom bojom, crveni su stacionarni, dok su žuti najavljeni radari. Ukoliko pritisnemo na neki od njih dobit ćemo kratke informacije o istom kao što je učinjeno na slici 8, dok ukoliko ponovo pritisnemo na taj mali skočni prozor otvorit će nam se zaslon sa malo detaljnijim informacijama vezanih za taj radar.

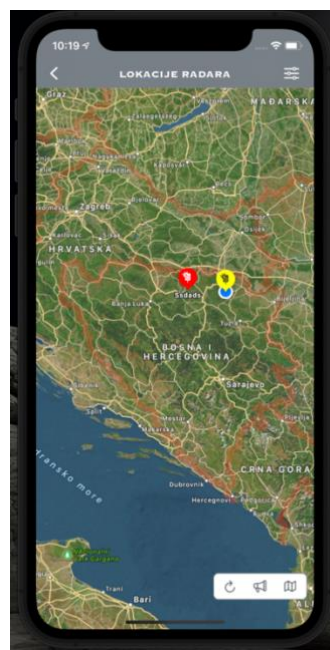
# UNIVERZITET U TUZLI

## FAKULTET ELEKTROTEHNIKE

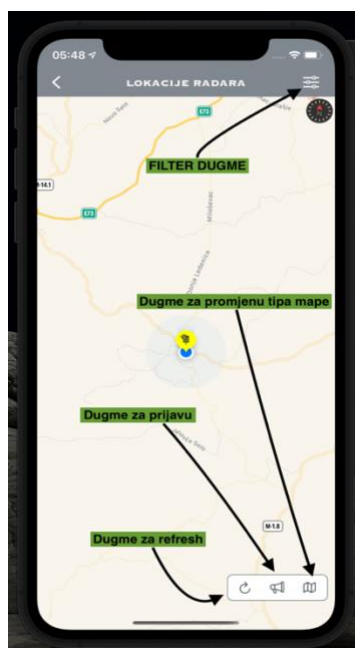
---



Slika 8. Zaslona sa prikazanim radarima



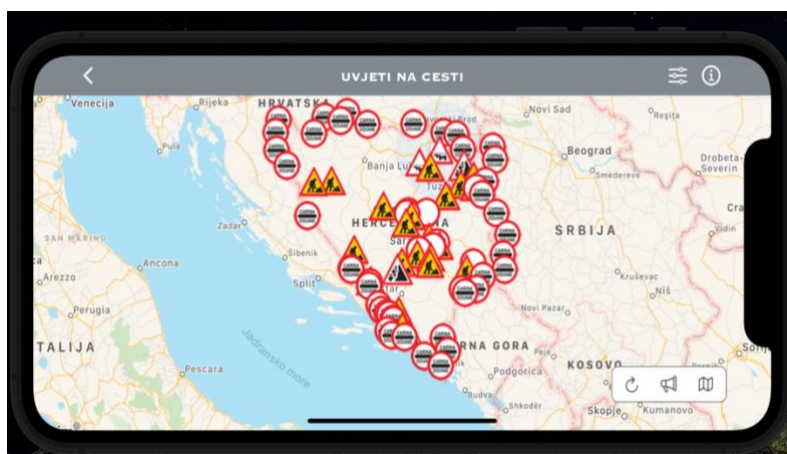
Slika 9. Zaslona sa prikazanim radarima, mapa teren



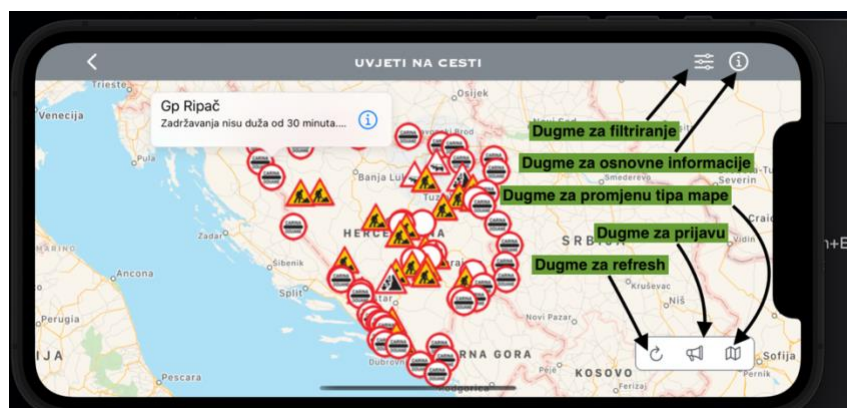
Slika 10. Prikaz funkcionalnosti na zaslonu za prikaz radara na putevima

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

Zaslon za prikaz stanja na putevima je po funkcionalnosti sličan zaslonu za prikaz radara. Opcije koje su vidljive na tom zaslonu su kao i na zaslonu za radare a to su opcije za refresh, prijavu novog stanja na putu, filtriranje stanja na putu, te tip mape. Ukoliko stitnemo na bilo koje stanje na put će nam otvoriti prvo kratki opis, a ukoliko stisnemo na taj kratki opis otvoriti će nam se zaslon za detaljniji opis. Razlike jesu u prikazu stanja na putevima jer ovdje koristimo odgovarajuće sličice za stanja umjesto crvene i žute boje koje smo koristili za radare. Također na ovom zaslonu imamo info opcija koju vidimo na slici 11 u desnom ćošku pored filter opcije.



Slika 11. Prikaz zaslona za prikaz stanja na putevima



Slika 12. Prikaz funkcionalnosti na zaslonu za prikaz stanja na putevima

# UNIVERZITET U TUZLI

## FAKULTET ELEKTROTEHNIKE

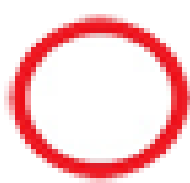
---

Svako stanje je predstavljeno odgovarajućom slikom. Trenutno od stanja na putevima imamo mogućnosti prikazati 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



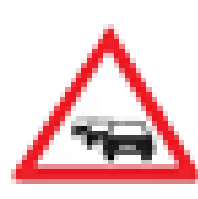
e) Granični prijelaz



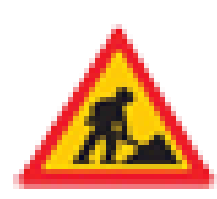
f) Opasnost



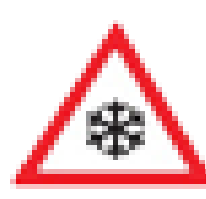
g) Zabrana za teretna vozila



h) Zagušenje



i) Sanacija kolovoza




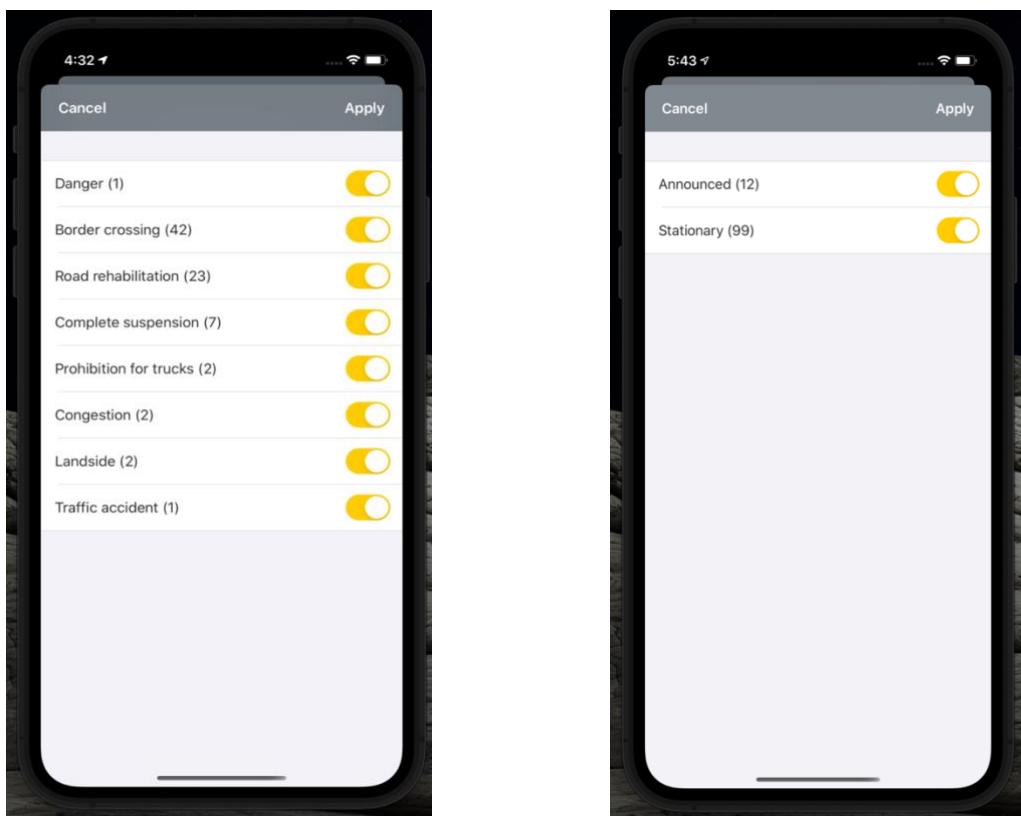
j) Poledica

**Slika 13. Prikaz svih znakova korištenih za prikaz stanja i radarana mapi**

Na slici 13 su prikazani svi znakovi i oznaka radara, koji su korišteni za prikaz stanja na mapi. Oznaka radara je, te u zavisnosti da li je Radar stacionarni ili najavljeni mi samo mijenjamo boju pozadine. Za stanje na putevima je drugačije, mi uzimamo naslov slike koji dobijemo sa API-a i samo u zavisnosti od naziva koristimo odgovarajuću sličicu. Sve slike 13. su spremljene direktno u aplikaciji. U nastavku ćemo prvo objasniti zaslon za filtriranje, te zaslon za prikaz detaljnih informacija o radaru, stanju na putevima i općih informacija, te zaslon za prijavu novih radara i stanja na putevima.

## 5.4. 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 pritisnemo na dugme  otvorit će nam se screen koji izgleda kao na Slici 14.





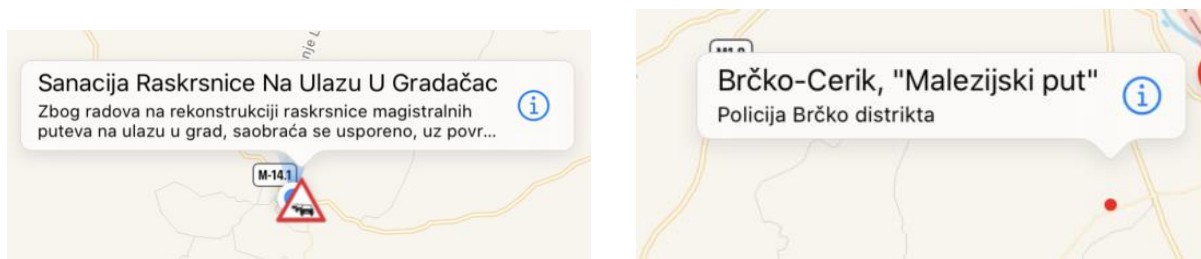
**Slika 14. Zaslon za filtriranje stanja na putevima i zaslon za filtriranje radara**

Do ovih zaslona možemo doći kada stisnemo na filter opciju na zaslonu za prikaz radara ili za prikaz stanja na putevima jedino u slučaju da imamo dva različita tipa oznake prikazana na mapi tj. ako je opcija za filtriranje vidljiva. Filteri rade na način da ukoliko su uključeni te oznake koje su uključene biti će i prikazane na mapi. Ukoliko isključimo neke od njih i stisnemo „Primijeni“ dugme na odgovarajućem jeziku kojeg aplikacija podržava na vrhu ti podaci će se poslati našoj mapi te će na mapi biti prikazani samo određene oznake.

Oznake možemo isključiti sve ali implementirano je da uvijek jedna mora biti uključena, to sam dodao iz razloga da ne bi korisnik se vratio na mapu i vidio praznu. Također ovaj zaslon se razlikuje od drugih jer se on prezentuje preko mape, a ne pusha predhodni zaslon u lijevu stranu. Ukoliko odaberemo opciju „Otkazi“ na odgovarajućem jeziku, vraćamo se na mapu sa uključenim filterima koje smo imali uključene kada smo ušli na filter zaslon.

## 5.5. Opis rada zaslona za prikaz detalja

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



**Slika 15. Prikaz malog opsia stanja radara ili stanja na putu**

# UNIVERZITET U TUZLI

## FAKULTET ELEKTROTEHNIKE

---

Na zaslonu općih informacija imamo samo nazlov, podnaslov i detaljan opis. 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 aktivan

Za radare najvažnije je da objekat 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 radar nije aktivan

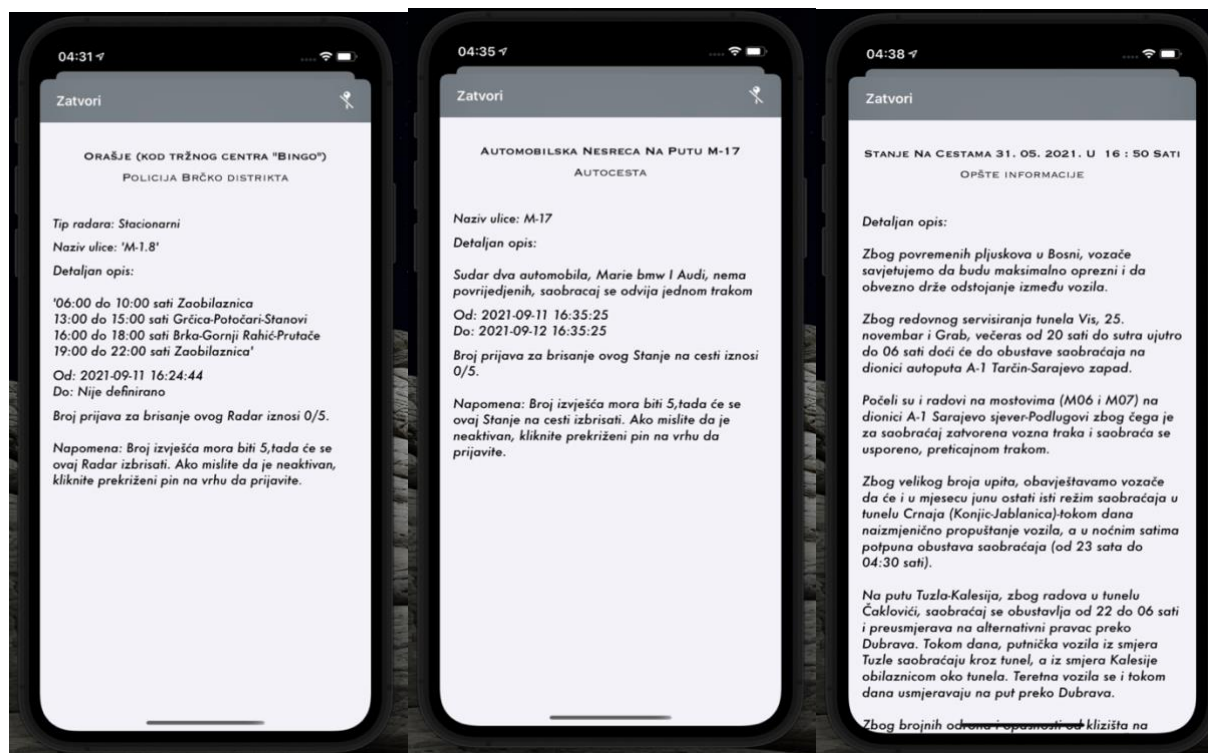


# UNIVERZITET U TUZLI

## FAKULTET ELEKTROTEHNIKE

Za stanja na putevima najvažnije je da objekat ima naslov, tip ulice i tip nesreće koji se ne prikazuje na zaslonu sa detaljima ali nam definiše koju ćemo sliku prikazati na mapi.

Kao što možemo primijetiti na zaslonima sa detaljima radara i stanja na putu imamo i napomenu jednu. Napomena nam kaže da ukoliko smatramo da radar nije aktivan ili stanje na putu nije prisutno da možemo prijaviti tako što ćemo pritisnuti na  dugme u desnom ćošku novog zaslona. Klikom na ovo dugme mi šaljemo odgovarajući endpoint pri čemu povećavamo broj prijava unutar objekta. Na ekranu će se pojaviti odgovarajuća poruka da li je api poziv bio uspješan ili ne kao na Slici 17. Ukoliko API poziv jeste bio uspješan pritisnut ćemo na OK dugme i možemo vidjeti da je broj prijava povećan za jedan. Ukoliko broj prijava je veći od 5 nakon prijave zaslon sa detaljima se zatvara i vršimo osvježivanje mape, te će navedeni radar koji smo prijavili 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 16. Prikaz zaslona za prikaz detalja radara, stanja na putevima i općih informacija

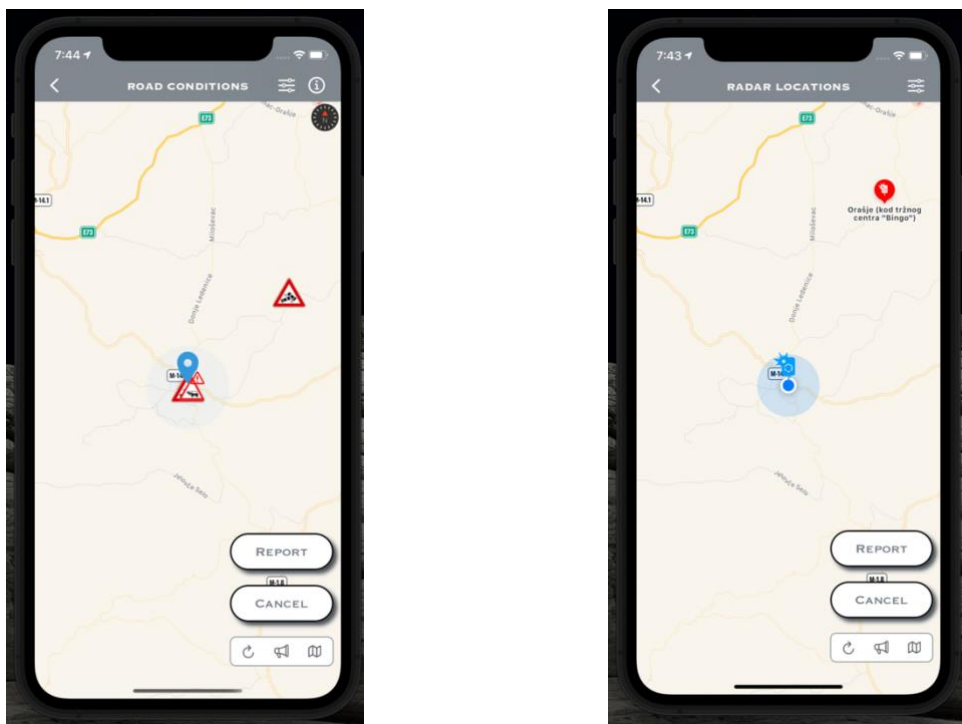




Slika 17. Uspješno prijavljen neaktivan radar ili stanje na putu

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

Kao što sam naveo ranije korisnik u našoj aplikaciji ima i mogućnost da prijavi nova stanja na putevima ili novi radar. Klikom na  dugme koje je prisutno na zaslonima za prikaz radara i stanja na putevima ukoliko je korisnik priključen na Internet, na mapi se pojavljuje jedna oznaka na mapi iznad naše trenutne lokacije. Krećući se po mapi horizontalno ili vertikalno mi pozicioniramo navedenu oznaku na odgovarajuće mjesto. Pored oznake na mapi se pojavljuju dva dugmeta tj. dugme Report i dugme Cancel, takav prikaz imamo na Slici 18. Klikom na Cancel dugme smatra se kao da smo odustali od prijave radara ili stanja na putu, te se nakon toga navedena oznaka i dugmad uklone sa mape. Ukoliko korisnik pritisne na dugme Report, korisniku se na zaslonu pojavljuje novi zaslon sa odgovarajućim poljima.



Slika 18. Pozicija oznake za prijavu



Slika 19. Prikaz report zaslona

## UNIVERZITET U TUZLI

### FAKULTET ELEKTROTEHNIKE

---

Zajedničko na ova dva zaslona koji su prikazani na Slici 19. jeste lokacija radara ili stanja na putu, naziv radara ili stanja na putu, ulica i detaljan opis. Lokacija predstavlja poziciju gdje smo ostavili našu oznaku sa prethodnog zaslona. Za naziv radara ili stanja na putu potrebno je unijeti neki kratki opis stanja. Za polje Ulica unosimo naziv ulice, dok detaljni opis sama riječ nam govori na šta se odnosi.

Pored toga imamo polje Radar type ili Road Type. Pri čemu Radar Type može biti Stacionarni ili Najavljeni, dok Road Type može biti

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

Pored toga na radarima imamo polje Police department koje ima opcije:

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

Dok na stanjima na putevima imamo polje Type of road condition koje ima opcije:

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

koji ujedno predstavljaju ikone znakova koje možemo prikazati za stanje na putu.

Da bi korisnik 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. Ukoliko korisnik ne unese obavezna polja a pritisne na Report dugme koje se nalazi na dnu ekrana korisnik će dobiti odgovarajuću poruku u vidu popUp-a da je potrebno da unese obavezna polja. Kada korisnik unese obavezna polja onda kada pritisne na report 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 osvježavamo odgovarajuću mapu za prikaz tj dohvatamo sa API-a stanja na putevima ili radare što smo objasnili na samom početku ovog rada.

## **5.7. Unos podataka u firebase uz pomoć skripti**

Prilikom izrade ovog diplomskog rada za unos podataka u bazu napisane su odgovarajuće JavaScript skripte, koje nisam dodavao u ovaj diplomski rad ali se nalaze u github firebase folderu. Za pokretanje isti potrebno je da imamo Node.js i NPM instaliran na našem uređaji. Node.js je okruženje koje uključuje sve što vam 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

```
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
```

**Kod 9. Načini pokretanja skripti za unos i brisanje podataka**

## **6. Zaključak**

Cilj ovog diplomskog rada je bila izrada IOS aplikacije za prikaz radara i stanja na putevima korištenjem Swift programskog jezika. Prilikom izrade aplikacije je odabran MapKit Framework za mapu zato što je striktno Apple – ov Framework za prikaz mape, iako Google Framework ima više opcija i dosta je zastupljeniji. Aplikacija je veoma zanimljiva i ima puno opcija, moguće je koristiti na tri različita jezika Engleski, Njemački i Bosanski, različitim pozicijama telefona te modovima rada. Najvjerojatnije ista neće biti na Apple Store cisto iz razloga jer ovakvih aplikacija imamo već na Apple Store. Primjer ovakve aplikacije jeste Bihamk aplikacija te je i moja aplikacija rađena po uzoru na nju.

## 7. Literatura

1. <https://github.com/eldarhaseljic/IOS-BosniaRoadTraffic>
2. <https://nodejs.org/en/>
3. <https://www.raywenderlich.com/7738344-mapkit-tutorial-getting-started>
4. <https://firebase.google.com/>
5. <https://developer.apple.com/xcode/>
6. <https://www.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://www.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>