

Diplomski rad

Prvog ciklusa studija

Razvoj IOS aplikacije za prikaz radara i stanja na putevima

Student

Haseljić Eldar

Mentor:

Dr.sc. Mešković Emir

red.prof.

Tuzla, 2021.

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 putevima

**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

[Uvod 7](#_Toc80611000)

[Korištene tehnologije i alati 8](#_Toc80611001)

[Swift programski jezik 8](#_Toc80611002)

[Xcode 9](#_Toc80611003)

[Kreiranje Xcode projekta 9](#_Toc80611004)

[iOS Simulator 12](#_Toc80611005)

[CocoaPods 12](#_Toc80611006)

[Povezivanje CocoaPods sa projektom 12](#_Toc80611007)

[Reaktivno programiranje 14](#_Toc80611008)

[RxSwift i RxCocoa Framework 14](#_Toc80611009)

[Firebase - Cloud Firestore baza podataka 15](#_Toc80611010)

[Model – View – ViewModel (MVVM) 17](#_Toc80611011)

[Struktura aplikacije 18](#_Toc80611012)

**Popis slika**

[Slika 1. Prikaz početnog zaslona aplikacije u landscape i portrait modu 7](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611013)

[Slika 2. Swift programski jezik 8](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611014)

[Slika 3. Xcode razvojna okolina 9](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611015)

[Slika 4. Welcome to Xcode 9](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611016)

[Slika 5. Xcode interface 10](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611017)

[Slika 6. Opcije novog Xcode projekta 10](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611018)

[Slika 7. iOS Simulator 12](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611019)

[Slika 8. Cloud Firestore 15](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611020)

[Slika 9. MVVM (Model – View- ViewModel) arhitektura 17](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611021)

**Kod Reference**

[Kod 1. Komanda za instalaciju cocoaPods 12](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611028)

[Kod 2. Komanda za inicijalizaciju Podfile datoteke 13](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611029)

[Kod 3. Sadržaj Podfile datoteke BosniaRoadTraffic aplikacije 13](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611030)

[Kod 4. Komade za instaliranje i nadogradnju biblioteka koje su navedene u Podfile 14](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611031)

[Kod 5. Komande za importiranje Rx biblioteka unutar Podfile datoteke 15](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611032)

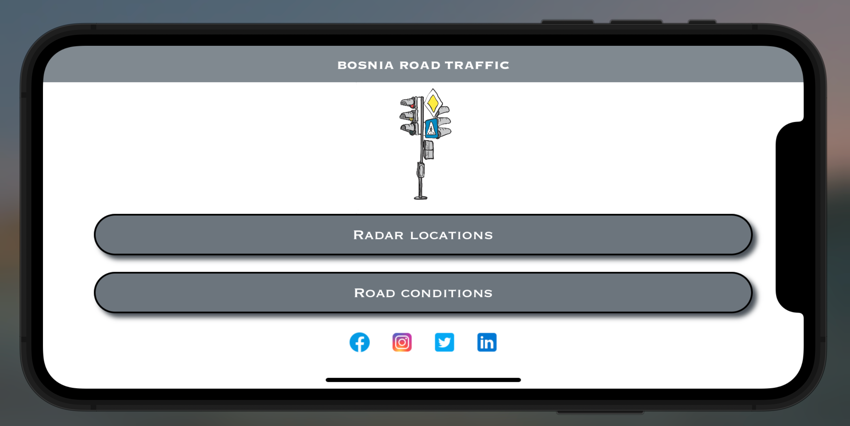
[Kod 6. Komande za importovanje Firebase biblioteka unutar Podfile datoteke 16](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611033)

[Kod 7. Konfiguracija Firebase baze unutar projekta 16](file:////Users/eldar.haseljic/Desktop/ios/zavrsni_rad_Haseljic_Eldar.docx#_Toc80611034)

# Uvod

Aplikacija o kojoj ćemo pričati u ovom diplomskom radu je namijenjena svim korisnicima koji se nalaze u saobraćaju. Ona sadrži prikaz svih radara na putevima, te prikaz samog stanja na putevima. Najveću korist ove aplikacije bi imali vozači motornih vozila, jer bi u bilo kojem trenutku mogli saznati gdje se nalazi radar ili bi mogli provjeriti stanje na putu.

Naravno aplikacija bi bila korisna i za pješake jer bi npr. mogli vidjeti ako je se negdje desila prometna nesreća. Također unutar aplikacije imamo dijelove gdje korisnik može sam da unosi nove pozicije radara ili pozicije gdje se desila neka nesreća, zastoj, moguća poledica ili drugi oblici stanja na putevima, ukoliko ta pozicija nije već navedena.



Slika 1. Prikaz početnog zaslona aplikacije u landscape i portrait modu

Ukoliko bi aplikacija bila u vlasništvu neke firme ili ovlaštenog lica te ako isti iz nekog izvora posjeduju veću količinu pozicija gdje se nalaze radari ili neka stanja na putevima ona se mogu unijeti u bazu uz pomoć odgovarajućih skripti. Unutar same aplikacije je ugrađen sistem brisanja pozicija radara ili stanja na putevima nakon određenog broja prijava korisnika da se na tim pozicijama više ne nalazi radar ili neki drugi tip stanja na putu, ali također postoje i odgovarajuće skripte za brisanje istih.

Sama aplikacija je pravljena samo za iOS platformu ali je moguće da se ista napravi i za druge mobilne platforme. Aplikacija je prevedena na tri jezika Njemački, Engleski i Bosanski. Na mapu unutar aplikacije Dark mode je primjenjiv tako da ukoliko je ta opcija uključena na mobitelu mapa će biti crne boje i okolne ikonice će se također prilagoditi istom. Aplikaciju je moguće koristit u vodoravnom (landscape) ili vertikalnom (portrait) modu

# Korištene tehnologije i alati

Krenut ćemo od osnovnih pojmova s kojima ćemo se susretati u ovom diplomskom radu i koji su bili neophodni za razvoj ove iOS aplikacije. Objasnit ćemo za početak pojmove poput Xcode, Swift, Rx biblioteke i druge.

Za izradu ove aplikacije potrebne su nam tri glavne stavke:

* MacOS računalo ili virtuelna mašina koja pokreće MacOS
* Xcode
* iOS SDK (eng. software development kit)

## Swift programski jezik

Swift je objektno orijentiran programski jezik koji je kreiran od strane Apple-a za razvoj programa i aplikacija na operativnim sistemima iOS i OS X. Prvi put objavljen 2014. godine, te je razvijen kao zamjena za Apple-ov programski jezik, koji je ranije korišten, Objective-C. 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.A picture containing text, clipart

Description automatically generated

Slika 2. Swift programski jezik

## Xcode

Text

Description automatically generated Xcode je razvojna okolina (eng. integrated development environment - IDE) koju koristimo za razvoj iPhone aplikacija. Xcode sadrži source editor, kompajler, emulator, razvojne Framework-e i još puno elemenata koja će nam olakšati razvoj iPhone aplikacija. U principu, od alata za razvoj potreban nam je samo Xcode. 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. Xcode je besplatan te se može skinuti sa App Store-a.

Slika 3. Xcode razvojna okolina

### Kreiranje Xcode projekta

Nakon što smo preuzeli Xcode sa App Store-a potrebno je da kreiramo projekat. Projekat kreiramo na sljedeći način:

* Na slici 4. odaberemo opciju Create a new Xcode project

Graphical user interface, application

Description automatically generated

Slika 4. Welcome to Xcode

* A screenshot of a computer

  Description automatically generated with medium confidenceNa Slici 5.

Slika 5. Xcode interface

* + Odaberemo ios za Multiplarform
  + Odaberemo App u Application sekciji
* A screen shot of a computer

  Description automatically generated with medium confidencePritisnemo Next

Slika 6. Opcije novog Xcode projekta

* Na slici 6.
  + Unesemo Product Name
    - Naziv naše aplikacije
  + Odaberemo Team
    - Naši timovi za App Store Connect bi se trebali pojaviti ovdje ako smo povezali Xcode sa svojim Apple ID -om.
    - Ako imate besplatni račun programera, ovdje odaberite Lični tim. Što je učinjemo u slučaju naše aplikacije
  + Editujemo Organization Identifier (opcionalno)
    - Organization Identifier je obrnuti naziv domene koji jedinstveno identificira vašu aplikaciju (u App Storeu).
    - Uobičajeno je izabrati nešto poput naziva com.organization name.app
  + Odaberemo Interface
    - Biramo UI framewor koji želimo koristiti za izradu svoje aplikacije.
    - Možete birati između Storyboard (UIKit) ili SwiftUI.
    - Za našu aplikaciju je korišten Storyboard interface
  + Odaberemo Life Cycle
    - Ovo određuje pristup koji vaš Xcode projekt koristi za upravljanje životnim ciklusom vaše aplikacije, uključujući način na koji se početni User Interface vaše aplikacije pokreće.
    - Možete birati između UIKit App Delegate ili SwiftUI App.
    - Za našu aplikaciju je korišten UIKit App Delegate
  + Odaberemo Language
    - Ovdje se misli u kojem jeziku želimo da pišemo našu aplikaciju
    - Možemo birati između izrade aplikacije s Objective-C ili sa Swiftom.
    - Međutim za moju verziju XCoda je bilo ponuđeno samo Swift te smo i odabrali Swift opciju
  + Dodatne checkmakr opcije
    - Ovo dodaje neki zadani kod vašoj aplikaciji.
    - Use Core Data služi za spremanje trajnih podataka vaše aplikacije za izvan mrežnu upotrebu te za keširanje privremenih podataka.
    - Include Tests se koristi u slučaju da želimo napisati određene test datoteke za testiranje naše aplikacije
    - Host in CloudKit se koristi ukoliko želimo da povežemo našu aplikaciju sa Apple CloudKit-om
    - Za našu aplikaciju smo odabrali samo Use Core Data opciju.
  + Pritisnemo Next i Create

Uspješno smo kreirali projekat i možemo početi sa radom

## A screenshot of a cell phone Description automatically generated with medium confidenceiOS Simulator

iOS simulator je alat koji nam služi kako bi pokrenuli i testirali aplikaciju na našem računalu. 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.

## 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 rezultirajući izvorni kod, a zatim ga povezati u Xcode radni prostor za izgradnju našeg projekta.

Slika 7. iOS Simulator

### Povezivanje CocoaPods sa projektom

CocoaPods je izgrađen s Ruby-em i bit će instaliran sa zadanim Ruby-om dostupnim na macOS-u. Da bi smo instalirali CocoaPods potrebno je da pokrenemo sljedeći komadnu u terminal.

Kod 1. Komanda za instalaciju cocoaPods

sudo gem install cocoapods

Nakon što smo instalirali CocoaPods potrebno je da napravimo Podfile unutar našeg projekta je datoteka koja sadrži dodatne biblioteke i specifikacije vezane za projekta. Datoteka bi se jednostavno trebala nazvati Podfile.

To možemo izvršiti uz pomoć komande

pod init

Kod 2. Komanda za inicijalizaciju Podfile datoteke

Prva linija koda u kreiranom fajlu potrebno je da bude “platform :ios, '12.0'” te ona predstavlja platformu i najnižu verziju koju podržavamo.

Kod 3. Sadržaj Podfile datoteke BosniaRoadTraffic aplikacije

# 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

project.build\_configurations.each do |bc|

bc.build\_settings['IPHONEOS\_DEPLOYMENT\_TARGET'] = deployment\_target

end

end

end

Da bismo koristili CocoaPods, moramo definirati Xcode target s kojim ćemo ih povezati. Na primjer, u gore navedenom fajlu za našu iOS aplikaciju, to bi bio naziv naše aplikacije. U fajlu 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 sto 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 komadu

Kod 4. Komade za instaliranje i nadogradnju biblioteka koje su navedene u Podfile

pod install

pod update

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.

## 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.

### RxSwift i RxCocoa Framework

RxSwift i RxCocoa dio su paketa jezičnih alata ReactiveX (Rx) koji obuhvaćaju više programskih jezika i platformi. RxSwift je Framework za interakciju sa programskim jezikom Swift, dok je RxCocoa framework koji čini API-je za Cocoa koji se koriste u iOS -u i OS X lakšim za upotrebu s reaktivnim tehnikama. Cocoa je razvojni Framework za izradu korisničkih okruženja za aplikacije i programe na operativnom sistemu OS X, dok je Cocoa Touch UI razvojni Framework za razvoj aplikacija na iOS-u (za iPhone, iPod Touch i iPad).

Neke od funkcionalnosti koje se koriste u aplikaciji su:

* Observable – pomoću ove funkcionalnosti odašilju se podaci, tj. stvaraju se asinhroni tokovi podataka.
* Observer – pomoću ove funkcionalnosti se pretplaćuje na asinhrone protoke podataka. Ukoliko se odašilje nekakva vrijednost, u pretplati se dobije njena vrijednost te se s njom može dalje manipulirati ili pozvati akciju kojoj je ta vrijednost potrebna.
* DisposeBag – Kada se deinit() pozove na objekt koji drži DisposeBag, svaki pretplatnik automatski se odjavljuje sa onoga na što je pretplaćen, bez ovoga moguće da dobijemo memory leak što vodi da bi aplikacija mogla pasti ili da aplikacija automatski padne.

RxSwift ima mnoštvo operatora pomoću kojih se može odraditi pretvorba podataka. Neki od njih su map, filter, flatMap i slično.Da bi smo dodali Rx biblioteke u naš projekat potrebno je u Podfile datoteku dodati sljedeće dvije linije koda

pod 'RxSwift'

pod 'RxCocoa'

Kod 5. Komande za importiranje Rx biblioteka unutar Podfile datoteke

## Firebase - Cloud Firestore baza podataka

Shape

Description automatically generated with low confidence 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. U ovome radu korištena je samo Cloud Firestore funkcionalnost Firebase-a.

Slika 8. Cloud Firestore

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 izvanmrežnu podršku uz učinkovite upite prema bazi. Odlična integracija s drugim Firebase proizvodima omogućuje ubrzanu izgradnju aplikacija bez poslužitelja. Firestore u aplikaciji je korišten za spremanje pozicija radara, stanja na putevima, te ostalih informacija i detalja koji oni nose sa sobom.

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

Postupak kreiranja projekta na Firebase je sljedeći:

1. Odaberemo opciju “Add Project”
2. Unesemo naziv projekta te pritisnemo “Continue”
3. Uključiti Google Analytics (opcionalno)
4. Pritisnemo opciju “Create Project”

Nakon što je projekat kreiran potrebno je isti uvezati sa našim Xcode projektom. Postupak povezivanje Firebase projekta sa Xcode projektom je sljedeći:

1. Otvorimo Firebase project Overview
2. Odaberemo opciju iOS
3. Unesemo iOS bundle ID iz Xcode projekta, te pritisnemo “Register App”
4. Potrebno je da preuzmemo GoogleService-Info.plist
5. Dodamo GoogleService-Info.plist u naš Xcode projekat
6. Kreiramo Podfile (ukoliko nismo) te dodamo

Kod 6. Komande za importovanje Firebase biblioteka unutar Podfile datoteke

pod 'FirebaseFirestoreSwift'

pod 'Firebase'

pod 'Firebase/Analytics'

1. U terminal pozovemo pod update
2. Pozovemo Firebase.configure() unutar class AppDelegate u dole navedenoj funkciji i dodamo liniju import Firebase van klase

Kod 7. Konfiguracija Firebase baze unutar projekta

**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**

}

}

1. Firebase uspješno podešen

## 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:

* 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.
* Diagram

  Description automatically generatedModel – sadrži podatke aplikacije. Najčešće se radi od klasama ili strukturama.

Slika 9. MVVM (Model – View- ViewModel) arhitektura

# Struktura aplikacije

Sama aplikacija je pisana uz pomoć MVVM arhitekturom, a na slici 10. 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.

Slika 10. Struktura aplikacije

***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.
* **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.

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

1: //

2: // AppDelegate.swift

3: // Bosnia Road Traffic

4: //

5: // Created by Eldar Haseljic on 1/3/21.

6: // Copyright ¬© 2021 Eldar Haseljic. All rights reserved.

7: //

8:

9: import UIKit

10: import CoreData

11: import Firebase

12:

13: @main

14: class AppDelegate: UIResponder, UIApplicationDelegate {

15:

16: func application(\_ application: UIApplication,

17: didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

18: // Override point for customization after application launch.

19: FirebaseApp.configure()

20: return true

21: }

22:

23: // MARK: UISceneSession Lifecycle

24:

25: @available(iOS 13.0, \*)

26: func application(\_ application: UIApplication,

27: configurationForConnecting connectingSceneSession: UISceneSession,

28: options: UIScene.ConnectionOptions) -> UISceneConfiguration {

29: // Called when a new scene session is being created.

30: // Use this method to select a configuration to create the new scene with.

31: return UISceneConfiguration(name: "Default Configuration", sessionRole: connectingSceneSession.role)

32: }

33:

34: @available(iOS 13.0, \*)

35: func application(\_ application: UIApplication,

36: didDiscardSceneSessions sceneSessions: Set<UISceneSession>) {

37: // Called when the user discards a scene session.

38: // If any sessions were discarded while the application was not running, this will be called shortly after application:didFinishLaunchingWithOptions.

39: // Use this method to release any resources that were specific to the discarded scenes, as they will not return.

40: }

41: }

42:

Kod 8. Sadržaj AppDelegate.swift fajla

* **scene(\_:willConnectTo:options:)**
  + Ovo je prva metoda koja se poziva u životnom ciklusu UISceneSession. 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 datoteke kao što možemo vidjeti na slici 10. 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*

# Korisničko okruženje aplikacije

A picture containing text, outdoor, sign, painted

Description automatically generated

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.

Aplikacija ima nekoliko zaslona, a to su:

* Zaslon pokretanja

Slika 11. Zaslon pokretanja

* + ***LauchScreen.storyboard***
* Graphical user interface

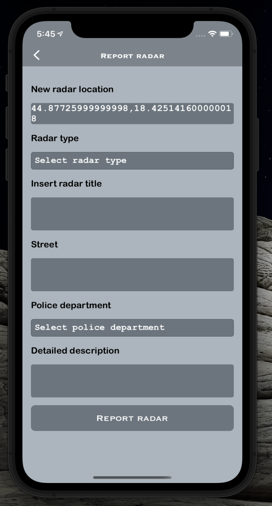
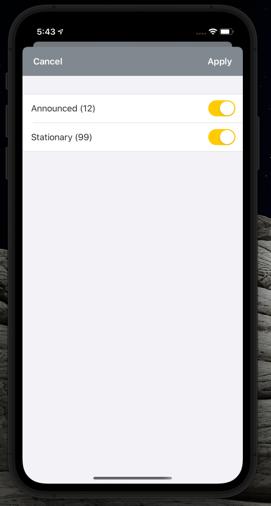
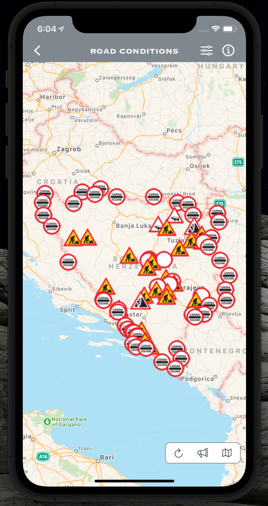
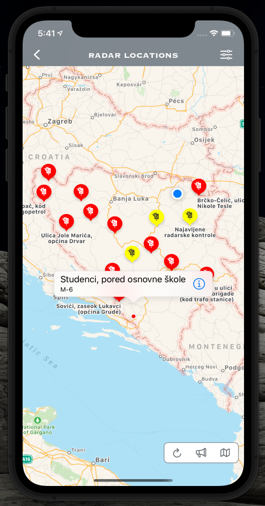
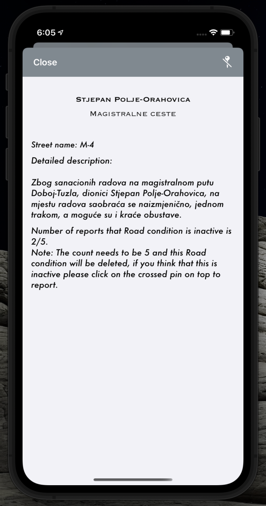
  Description automatically generatedGlavni zaslon
  + ***MainScreen.storyboard***
* Zaslon za prikaz radara
  + ***RadarsMapStoryboard.storyboard***
* Zaslon za prikaz stanja na putevima
  + ***RoadConditionsStoryboard.storyboard***
* Zaslon za prijavu novog radara/ stanja na putu
  + ***ReportStoryboard.storyboard***
* 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

Slika 12. Glavni zaslon

* + ***DetailsStoryboard.storyboard***

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

Slika 13. Prikaz ostalih screenova koji su vidljivi u aplikaciji



# Opis rada i implementacija aplikacije

## Opis rada glavnog zaslona

A picture containing text, electronics, iPod, screenshot

Description automatically generated Kada korisnik pokrene aplikaciju na svom iPhone uređaju prva stvar na koju nailazi jeste zaslon pokretanja koji je prikazan na Slici 11. U suštini to je zaslon koji dobijemo kada kreiramo sam projekat. Razlika izmedju zadane vrijednosti koju dobijemo prilikom kreiranja projekta i našeg zaslona pokretanja (***LauchScreen.storyboard***)jeste što se na našem nalazi slika, dok na zadanoj vrijednosti imamo samo bijelu pozadinu ili crnu u zavisnosti da li je tamni način rada uključen. Nakon što se aplikacija učita dolazimo na glavni zaslon (***MainScreen.storyboard***)koji je na prikazan na Slici 12.

Na glavnom zaslonu možemo primijetiti na vrhu zaslona da ima naziv, dva siva dugmeta, te nekoliko sličica društvenih mreža koje također predstavljaju dugmad. Ukoliko pritsnesnemo 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 otvraju 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.

Slika 14. Prikaz upozorenja prilikom klika na LinkedIn

### Implementacija glavnog zaslona

Svaki klik na bilo koje navedeno dugme je izvršen uz pomoć tzv. Reakaktivnog wrappera za TouchUpInside event nad dugmetom (rx.tap). Na njega predplaćujemo 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 koristit sa oprezom. Svaki klik na većinu dugmadi u aplikaciji je izvršen na način koji je naveden u Kod 9. sekciji.

Unutar prvog bind vidimo funkciju pushView(viewController: RoadConditionsViewController.showRoadConditions()) koja služi za prelazak sa jednog zaslona na drugi pri čenu se oni pomijeraju u lijevu stranu tj vrši se push nad zaslonima kao da se nalaze u jednoj stack strukturi.

Kod 9. Prikaz sadržaja funkcije pushView(viewController: \_)

14: public func pushView(viewController: UIViewController) {

15: navigationController?.pushViewController(viewController,

16: animated: true)

17: }

Kod 10. Reaktivni wrapper za TouchUpInside event nad dugmetom

52: roadConditionsButton.rx.tap.bind { [unowned self] in

53: pushView(viewController: RoadConditionsViewController.showRoadConditions())

54: }.disposed(by: disposeBag)

55:

56: facebookButton.rx.tap.bind { [weak self] in

57: guard let self = self else { return }

58: self.openExternalUrl(urlString: Constants.URLPaths.facebookURL)

59: }.disposed(by: disposeBag)

Funkcija se nalazi unutar fajla UIViewController+Extension.swift, gdje su ujedno smještene sve tzv. extenzije za UIViewController. Taj fajl se nalazi u folderu na putanji BosniaRoadTraffic/Extensions/UIExtensions gdje se nalaze i razne druge extenzije za druge UI komponente, dok unutar folderu BosniaRoadTraffic/Extensions pored UI extenzija imamo i neke male extenzije za osnovne tipove kao sto su Array, String, Date i Optional. U dugom bind vidimo funkciju openExternalUrl(urlString:\_) gdje nam sami naziv funkcije govori šta radi tj otvara odgovarajuću url u vanjskom pretraživaču.

Kod 11. Sadržaj funckije openExternalUrl(urlString:\_)

67: public func openExternalUrl(urlString: String) {

68: guard let webURL = URL(string: urlString) else {

69: self.presentAlert(title: ERROR\_DESCRIPTION,

70: message: ERROR\_URL\_MESSAGE,

71: buttonTitle: OK,

72: handler: nil)

73: return

74: }

75: UIApplication.shared.open(webURL)

76: }

Također i ova funkcija se nalazi unutar fajla UIViewController+Extension.swift

Funkcionalnost glavnog zaslona je definisana u fajlu MainViewController.swift i to je jedini zaslon koji nema viewModel jer mu nije bio potreban.

## Opis rada zaslona za prikaz radara

A picture containing text, electronics, iPod, cellphone

Description automatically generated Ukoliko pritisnemo na gornje sivo dugme dugme na glavnom zaslonu, pokreće se spiner 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 bice vraćen automatski na glavni tj početni zaslon. Ukoliko je dopustio prikaz lokacije onda se poziva API poziv koji dovata 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 kaon a slici 15. U gornjem desnom uglu korisnik ima opciju da filtriranje radara.

Slika 15. Zaslon sa prikazanim radarima

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. 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, pričemu osvježimo radare tj učitamo nove ili se izbrišu odredjeni 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 15 i mapu sa terenom koja je prikazana na slici 16.

A screen shot of a map

Description automatically generated with low confidence

Slika 16. Zaslon sa prikazanim radarima, mapa teren

Ukoliko radara nemamo ili ako smo dobili informacije o novim radarima ilia ko se desi neka pogreška uvijek ćemo dobiti odgovarajuci Alert, tipa kao na slici 14.

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 15, dok ukoliko ponovo pritisnemo na taj mali skočni prozor otvorit će nam se zaslon sa malo detaljnijim informacijama vezanih za taj radar.

### Implementacija zaslona za prikaz radara

Sa glavnog zaslona na zaslon sa radarima dolazimo nakon klika na gornje sivo dugme na glavnom zaslonu, tu se poziva funkcija pushView(viewController: RadarsMapViewController.showRadars()).

pushView(viewController:\_) funkciju smo objasnili ranije, te znamo da služi sa prikaz novg zaslona sada je potrebno da pogledamo sadržaj funkcije showRadars().

Kod 12. Prikaz funkcije showRadars()

U showRadars() funkciji možemo primijetiti da se prvo dohvata viewCotroller koji je RadarsMapViewController. Nakon toga postavljamo viewModel za taj kontroler i vratimo kotroler nazad. U to vrijeme unutar postavljenog viewModela, koji u ovom slučaju predstavlja RadarsMapViewModel, se obavlja funkcionalnost koja na zaslon za korisnika izbacuje poruku da korisnik dopusti aplikaciji pristup njegovoj lokaciji. To se izvršava unutar funkcije checkLocationServices().

57: func checkLocationServices() {

58: if CLLocationManager.locationServicesEnabled() {

59: setupLocationManager()

60: checkLocationAuthorization()

61: } else {

62: messageTransmitter.onNext(Adviser(title: ERROR\_DESCRIPTION,

63: message: LOCATION\_SERVICE\_DISABLED,

64: isError: true))

65: }

66: }

67:

68: private func setupLocationManager() {

69: locationManager.delegate = self

70: locationManager.desiredAccuracy = kCLLocationAccuracyBest

71: }

Kod 13. Sadrzaj checkLocationServices() funkcije

233: extension RadarsMapViewController {

234: static func getViewController() -> RadarsMapViewController {

235: return UIStoryboard(name:Constants.StoryboardIdentifiers.RadarsMapStoryboard,

236: bundle: nil)

237:.instantiateViewControllerWithIdentifier(RadarsMapViewController.self)!

238: }

239:

240: static func showRadars() -> RadarsMapViewController {

241: let radarsViewController = RadarsMapViewController.getViewController()

242: radarsViewController.setViewModel()

243: return radarsViewController

244: }

245: }

Prvo što primijetimo unutar ove funkcije jeste da provjeravamo da li je uopšte uključena opcija lokacije globalno na mobitelu ukoliko nije korisnik će dobiti poruku da bi trebao uključiti Lokaciju na svom mobitelu. Poruku će dobiti preko messageTransmitter koji je PublishSubject<Adviser>() objekat i on se može koristiti i kao odašiljač ili prijemnik informacija između ViewCotrollera i viewModela. Pošto je PublishSubject generičkog tipa vidimo da u našem slučaju uzma objekat Adviser.Model te strukture je prikazan u Kod 14.

9: import Foundation

10: import UIKit

11:

12: class Adviser {

13: var title: String

14: var message: String

15: var isError: Bool

16:

17: init(title: String,

message: String,

isError: Bool = false) {

18: self.title = title.localizedUppercase

19: self.message = message

20: self.isError = isError

21: }

22: }

Unutar navedene strukture imamo tri prarametra to su:

* title
  + predstavlja naziv našeg Alerta kojeg prikazujemo useru
* message
  + predsavlja poruku koju želimo proslijediti našem korisniku
* isError
  + predstavlja parametar koji je inače false jer smatramo da je Alert samo poruka za korisnika, međutim ovu strukturu koristimo I kada želimo da prikažemo korisniku da se desila određena pogreška, tako da se taj parametar koristi u takvim situacijama

Ukoliko lokacija kod korisnika na mobitelu je uključena izvrše se određene postavke za LocationManagera, te se pozove funkcija checkLocationAuthorization().

Koja provjerava da li je korisnik odbio prikazati lokaciju ili je označio da je lokacija vidljiva samo jednom ili neke druge opcije lokacije te u zavisnosti od opcije šalje se odgovarajuća informacija viewCotrolerru koji je dalje koristi. Ako je lokacija odobrena jednom ili uvijek za aplikaciju poziva se funkcija fetchData(), koja unutar sebe koristi managera koji je u ovom slučaju MainManager te se poziva fukcija getRadars()koja dalje dohvata podatke o radarima iz FireStore.

Ako korisnik nije konektovan na internet funkcija će mu vratiti radare iz baze koje trenutno ima u mobitelu i dobit će odgovarajuću Alert poruku. U supronom dohvaćeni radari se dekodiraju u Radar strukturu te se spremaju u bazu u mobitel. Ukoliko se neki od radara nalazi u bazi na mobitelu a nismo ga ponovo dobili u response od firestore onda se takvi radari brišu iz baze u mobitelu. Također ukoliko u response dobijemo radar koji nema kordinate lokacije ili ima 5 prijava da je neaktivan i taj se ne unosi u bazu u mobitel, te se šalje I zahtjev za brisanje istog iz baze na Firstore. Nakon što su radari spremljeni u bazu šalju se na prikaz na mapu.