Upotreba MVVM arhitekture za razvoj IOS aplikacija

Mihalj, Matko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Upotreba MVVM arhitekture za razvoj IOS aplikacija

Diplomski rad

Matko Mihalj

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Matko Mihalj
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D-1229R, 07.10.2021.
JMBAG:	0165077844
Mentor:	prof. dr. sc. Damir Blažević
Sumentor:	prof. dr. sc. Tomislav Keser
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 1:	prof. dr. sc. Damir Blažević
Član Povjerenstva 2:	Ivana Kovačević, univ. mag. ing. comp.
Naslov diplomskog rada:	Upotreba MVVM arhitekture za razvoj IOS aplikacija
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Opisati i objasniti MVVM arhitekturu za razvoj aplikacija. Usporediti s drugim suvremenim arhitekturama razvoja te na primjeru izrade IOS aplikacije pokazati prednosti i izazove korištenju MVVM arhitekture. Izraditi aplikaciju.

Datum ocjene pismenog dijela diplomskog rada od strane mentora:	23.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	27.9.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	30.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 30.09.2024.

Ime i prezime Pristupnika:	Matko Mihalj
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1229R, 07.10.2021.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Upotreba MVVM arhitekture za razvoj IOS aplikacija**

izrađen pod vodstvom mentora prof. dr. sc. Damir Blažević

i sumentora prof. dr. sc. Tomislav Keser

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1.UVOD

MVVM (Model-View-ViewModel) arhitektura predstavlja izuzetno koristan obrazac za razvoj iOS aplikacija, zahvaljujući preciznom odvajanju poslovne logike smještene u ViewModel sloju od korisničkog sučelja (View). Ova podjela omogućava poboljšanu testabilnost, održivost i prilagodljivost aplikacija, istovremeno pojednostavljujući integraciju novih funkcionalnosti i upravljanje složenijim sustavima. ViewModel služi kao posrednik između Modela i View-a, omogućavajući da promjene u podacima budu automatski prikazane u korisničkom sučelju, dok View ostaje pasivan i zadužen samo za prikaz podataka.

odabir MVVM arhitekture omogućuje modularan razvoj, što rezultira aplikacijama koje su jednostavne za održavanje, skalabilne i lako prilagodljive budućim promjenama. Jedna od ključnih prednosti ove arhitekture je jasno razdvajanje odgovornosti, gdje poslovna logika ostaje izolirana od sučelja, smanjujući složenost koda te olakšavajući ponovnu upotrebu komponenti. Osim toga, MVVM omogućuje jednostavniju implementaciju naprednih funkcionalnosti poput navigacije, integracije QR kodova i text-to-speech opcija, bez ugrožavanja stabilnosti aplikacije.

kao praktični primjer primjene ove tehnologije odabrana je izrada turističke aplikacije za grad Đakovo, koja je svojim zahtjevima prikladna za demonstraciju mogućnosti MVVM arhitekture. Ovaj projekt uključuje prikaz informacija o znamenitostima, upravljanje velikim količinama podataka te omogućuje korisnicima interakciju s aplikacijom, što predstavlja kompleksan izazov. Aplikacija zahtijeva visoku razinu skalabilnosti kako bi se mogla proširivati novim funkcionalnostima, poput dodavanja novih turističkih atrakcija, naprednih mogućnosti filtriranja ili interaktivnih modula za navigaciju. Zbog složenosti i potrebe za efikasnim upravljanjem podacima, MVVM arhitektura omogućava jasno odvajanje poslovne logike od sučelja, olakšavajući upravljanje takvim procesima.
Primjenom MVVM arhitekture, aplikacija će biti modularna, skalabilna i jednostavna za proširenje, što je ključno za dugoročnu održivost i uspješno prilagođavanje novim funkcionalnostima ili zahtjevima korisnika. Na praktičnom primjeru razvoja turističke aplikacije, detaljno će se prikazati svi izazovi i prednosti korištenja ove arhitekture, omogućujući stvaranje visokokvalitetne aplikacije koja pruža bogato i intuitivno korisničko iskustvo. U Poglavlju 1. objašnjava se primjena MVVM arhitekture u razvoju iOS aplikacija, s

fokusom na turističku aplikaciju za grad Đakovo. Poglavlje 2. detaljno analizira različite arhitekturne obrasce, kao što su MVVM, MVC, MVP i VIPER, te njihove prednosti i primjenu. Poglavlje 3. pokriva tehnički aspekt razvoja iOS aplikacija, uključujući Swift, Xcode i automatsko upravljanje memorijom. U Poglavlju 4. razmatraju se slične aplikacije poput TripAdvisora i Google Mapsa te se uspoređuju s rješenjem razvijenim u ovom radu. Poglavlje 5. opisuje arhitekturu aplikacije, funkcionalne zahtjeve i uporabu Firebase baze podataka. Poglavlje 6. fokusira se na implementaciju MVVM obrasca u aplikaciji, s naglaskom na korisničko sučelje i funkcionalnosti poput filtriranja i prikaza detalja znamenitosti. Poglavlje 7. zaključuje rad, naglašavajući prednosti MVVM arhitekture i njezinu uspješnu primjenu u projektu.

1.1 Pregled područja

Ovo poglavlje pruža pregled tehnologija korištenih u razvoju iOS aplikacija, s posebnim naglaskom na MVVM (Model-View-ViewModel) arhitekturu i srodne obrasce. MVVM arhitektura omogućuje jasno odvajanje poslovne logike (ViewModel) od korisničkog sučelja (View), čime se olakšava testiranje, održavanje i proširivanje aplikacija. Osim MVVM-a, u razvoju iOS aplikacija često se primjenjuju i arhitekturni obrasci poput MVC (Model-View-Controller), MVP (Model-View-Presenter) i VIPER. Svaki od ovih obrazaca nudi specifične prednosti, koje su detaljnije obrađene u Poglavlju 2.

Što se tiče postojećih aplikacija u području turizma, one obično uključuju napredne funkcionalnosti kao što su interaktivne karte, personalizirane preporuke atrakcija i navigacija. Primjeri takvih aplikacija su TripAdvisor, Google Maps i Airbnb, koje pružaju širok raspon usluga za putnike, uključujući prikaz turističkih atrakcija, recenzije i mogućnost rezervacija. Te aplikacije implementiraju različite tehnološke obrasce i funkcionalnosti, a detaljnija analiza tih rješenja može se pronaći u Poglavlju 4, gdje se uspoređuju s aplikacijom razvijenom u ovom radu.

2. ARHITEKTURNI OBRASCI ZA RAZVOJ IOS MOBILNIH APLIKACIJA

Arhitekturni obrazac je skup pravila, tehnika i praksi koji definiraju kako bi razvoj mobilnih aplikacija trebao teći. Ova pravila omogućuju stvaranje logičnog i koherentnog proizvoda koji ispunjava zahtjeve klijenata i zadovoljava industrijske standarde. Arhitekturni obrasci su ključni za razvoj iOS aplikacija jer pružaju strukturu za organizaciju koda, poboljšavaju održivost i skalabilnost aplikacija. iOS razvojni inženjeri koriste nekoliko arhitekturnih obrazaca, a odabir pravog obrasca ovisi o različitim faktorima kao što su ciljana publika, platforma za razvoj, potrebne značajke i funkcionalnosti, te raspoloživo vrijeme i budžet, uzimajući u obzir i vještine razvojnog tima. Apple pruža smjernice za razvoj iOS mobilnih aplikacija koristeći MVVM model, no razvojni inženjeri nisu ograničeni samo na ovaj obrazac.

Mnoge aplikacije su razvijene bez ikakve arhitekturne strukture ili bez poštivanja industrijskih standarda, što može dovesti do duljeg i skupljeg razvojnog procesa, teške održivosti (posebno u slučaju promjene tima), otežane nadogradnje i skalabilnosti, poteškoća u testiranju i povećane sklonosti greškama. Dobar arhitekturni obrazac za mobilne aplikacije treba slijediti principe razvoja softverskih rješenja*: „Kiss“(eng. Keep it Simple Stupid), „DRY“(eng. Do not repeat yourself) i „SOLID“(eng. single responsibility,open closed principle, Liskov substitution principle, interface segregation principle and dependency inversion principle)* . Jasno definiran arhitekturni obrazac podržava fleksibilnost i agilne metodologije razvoja, što olakšava testiranje i buduće održavanje, čineći ih jednostavnijim i manje sklonom greškama.

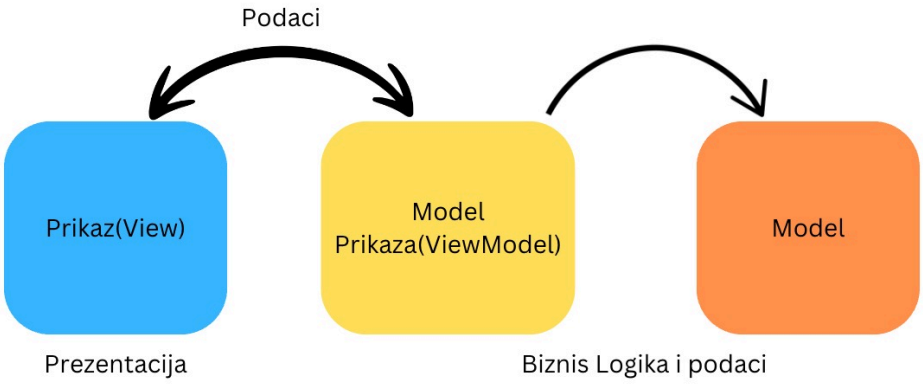
Dobar arhitekturni obrazac nije specifičan za jednu platformu, već bi trebao biti primjenjiv i na nativna i na hibridna rješenja. Iako ne postoji savršen univerzalni arhitekturni obrazac, odabirom odgovarajućeg obrasca može se povećati učinkovitost razvojnog tima, poboljšati kvaliteta koda te ubrzati buduće nadogradnje i promjene. No, sama odluka o arhitekturnom obrascu neće automatski rezultirati dobrom arhitekturom koda. Važno je organizirati kod u manje, što je moguće neovisne dijelove i podijeliti aplikaciju na više modula. Također, preporučuje se korištenje ubrizgavanja ovisnosti (*eng. dependency injection*) [1].

Pojedini arhitekturni obrasci pružaju različite prednosti i izazove, stoga je važno pažljivo odabrati najprikladniji obrazac za specifične potrebe projekta kako bi se osigurala visoka kvaliteta i dugoročna održivost aplikacije.

Arhitekturni obrasci ne moraju biti striktno određeni tako da se u jednoj aplikaciji koristi isključivo jedan obrazac. Mogu se primjenjivati na različitim razinama, bilo na cijeloj aplikaciji, pojedinim modulima ili čak unutar dijelova modula. Različiti moduli mogu koristiti različite arhitekturne obrasce, omogućujući fleksibilnost i prilagodljivost u dizajnu i razvoju aplikacije.

2.1. ARHITEKTURNI OBRAZAC MVVM

Arhitekturni obrazac Model-View-ViewModel (MVVM) [2] je popularan među razvojnim inženjerima zbog svojih brojnih prednosti u organizaciji i održavanju koda, te poboljšanja mogućnosti testiranja aplikacija. MVVM arhitektura dijeli aplikaciju na tri glavne komponente: Model, View i ViewModel. Model predstavlja podatke i poslovnu logiku aplikacije, View je odgovoran za prikaz korisničkog sučelja, dok ViewModel služi kao posrednik između Modela i View-a. ViewModel sadrži logiku i stanje aplikacije te transformira podatke iz Modela u oblik pogodan za prikaz u View-u. Ovaj arhitekturni obrazac omogućava jasnu separaciju odgovornosti, što olakšava razvoj, održavanje i testiranje aplikacija, te potiče modularnost koda što je posebno korisno u većim i kompleksnijim projektima.



Slika 2.1. MVVM ARHITEKTURA

2.1.1. Komponente MVVM Obrasca

Model

- **Opis:** Model predstavlja poslovnu logiku i podatke aplikacije. Ova komponenta enkapsulira podatkovne strukture, algoritme i interakcije s vanjskim uslugama ili bazama podataka.
- **Odgovornosti:** Upravljanje integritetom podataka, izvođenje izračuna i implementacija poslovnih pravila.

View

- **Opis:** View upravlja korisničkim sučeljem i njegovim prikazivanjem. Fokusira se na prikaz informacija korisniku i primanje korisničkog unosa.
- **Odgovornosti:** Prikaz podataka i ažuriranje prikaza u skladu s promjenama u ViewModelu, te prijenos radnji korisnika na ViewModel za obradu.

ViewModel

- **Opis:** ViewModel djeluje kao posrednik između Viewa i Modela. Otkriva podatke i naredbe koje View može promatrati.
- **Odgovornosti:** Dohvaća podatke iz Modela, oblikuje ih i omogućava Viewu da reagira na promjene, prilagođavajući prikaz u skladu s tim.

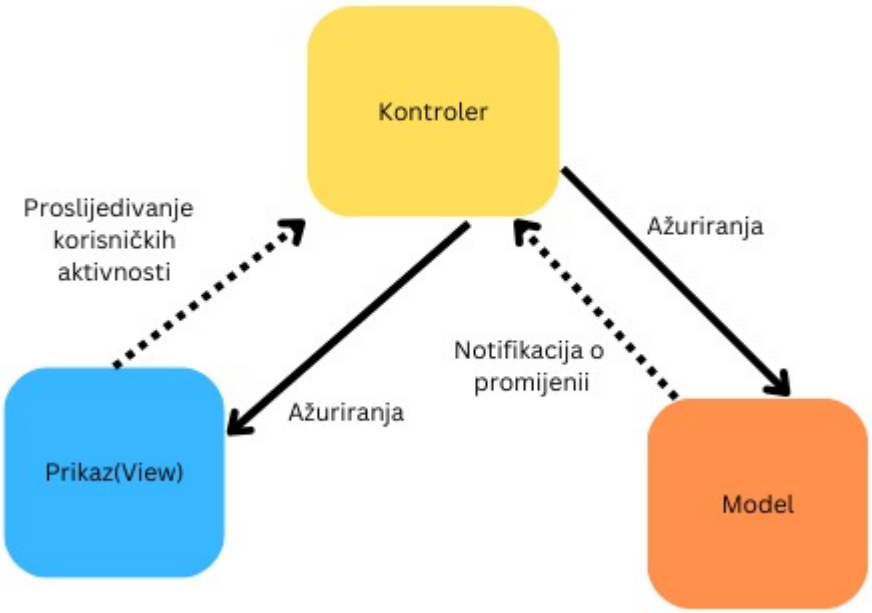
2.1.2. Prednosti MVVM Arhitekture

1. **Modularnost koda:** MVVM arhitektura vodi do jasne separacije aplikacije na slojeve, gdje svaki sloj ima vlastitu odgovornost. Ovo omogućava jasnu granicu između korisničkog sučelja, poslovne logike i dohvaćanja podataka.
2. **Vezanje podataka (Data Binding):** MVVM često koristi radne okvire za vezanje podataka kao što su RxSwift ili Combine. Ovo omogućava reaktivni protok podataka između Viewa i ViewModela, što pojednostavljuje ažuriranje elemenata korisničkog sučelja i smanjuje potrebu za ručnom sinkronizacijom podataka.

3. **Mogućnost testiranja:** MVVM poboljšava mogućnost testiranja jer je poslovna logika odvojena od korisničkog sučelja. ViewModel se može testirati neovisno o Viewu, što omogućuje sveobuhvatno jedinično testiranje.
4. **Recikliranje koda:** Odvajanjem Viewa od logike u ViewModelu, obje se komponente mogu ponovno koristiti u više slučajeva ili čak u različitim projektima. Ovo smanjuje razvojni napor i povećava efikasnost.

2.2. ARHITEKTURNI OBRAZAC MVC

Model-View-Controller (MVC) jedan je od najpopularnijih arhitekturnih obrazaca u razvoju iOS aplikacija. Ovaj obrazac nudi metodičan način raspodjele koda, podjele problema i poboljšanja skalabilnosti te mogućnosti održavanja iOS aplikacija .



Slika 2.2 MVC ARHITEKTURA

2.2.1. Komponente MVC Obrasca

Model

- **Opis:** Model predstavlja podatke i poslovnu logiku aplikacije. Uključuje interakcije s vanjskim servisima ili bazama podataka, strukturama podataka i algoritmima.
- **Odgovornosti:** Upravljanje podacima, izračuni, implementacija poslovnih pravila i održavanje integriteta podataka.

View

- **Opis:** View upravlja korisničkim sučeljem i njegovim prikazivanjem. Fokusira se na davanje informacija korisniku i primanje korisničkog unosa.
- **Odgovornosti:** Prikaz podataka iz Modela korisniku i slanje korisničkih unosa Controlleru. View prati izmjene u Modelu i po potrebi prilagođava svoj prikaz.

Controller

- **Opis:** Controller djeluje kao posrednik između Modela i Viewa. Obraduje korisničke unose i ažurira Model ili View prema potrebi.
- **Odgovornosti:** Obrada korisničkih akcija, ažuriranje Modela i osvježavanje Viewa.

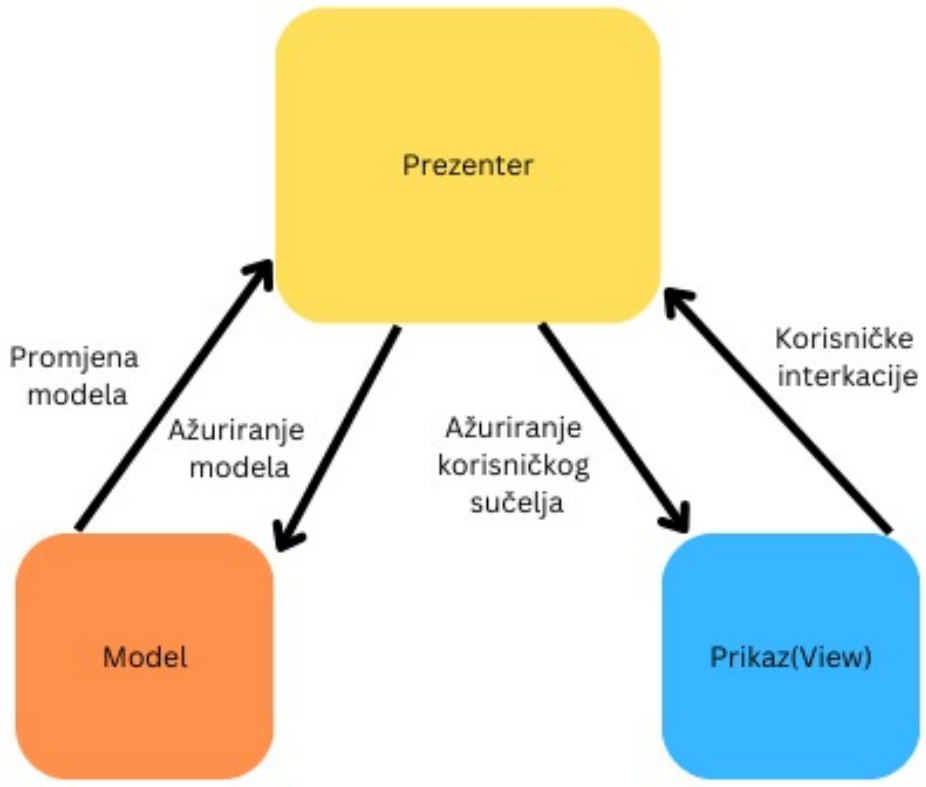
2.2.2. Prednosti MVC Arhitekture

Razdvajanje odgovornosti: MVC omogućuje jasnu podjelu između slojeva podataka, poslovne logike i korisničkog sučelja. Ova podjela poboljšava organizaciju koda, olakšava njegovo održavanje i poboljšava mogućnost testiranja.

Ponovno korištenje koda: Odvajanjem Modela, Viewa i Controller-a, razvojni inženjeri mogu ponovno koristiti dijelove koda u nekoliko značajki ili čak u različitim projektima.

2.3. ARHITEKTURNI OBRAZAC MVP

Ovaj arhitekturni obrazac je posebno koristan za kompleksne iOS aplikacije jer pruža strukturu koja potiče modularnost, skalabilnost i lakše upravljanje kodom. Integriranjem MVP-a, timovi mogu brže reagirati na promjene i iterativno poboljšavati svoje proizvode na temelju povratnih informacija korisnika. U MVP arhitekturi, ključna promjena je ta što komponenta View ne posjeduje direktnu instancu komponente Model. Umjesto toga, instanca komponente Model je smještena unutar komponente Presenter. Kada korisnik interagira s komponentom View, Presenter preuzima kontrolu nad tom interakcijom. Presenter zatim manipulira Modelom, a nakon što se Model promijeni, Presenter prima obavijest i ažurira komponentu View prema potrebi.



Slika 2.3 MVP ARHITEKTURA

2.3.1. Komponente MVP arhitekture

 Model: Komponenta koja predstavlja podatke i poslovnu logiku aplikacije. Model je odgovoran za obradu podataka, komunikaciju s bazom podataka ili API-ima te za obavljanje poslovnih operacija.

 View: Komponenta koja je odgovorna za prikaz korisničkog sučelja (UI). View ne sadrži poslovnu logiku; umjesto toga, prikazuje podatke koje dobiva od Presentera i reagira na korisničke akcije (npr. klikove na gumb, unos teksta).

 Presenter: Posrednik između Modela i View-a. Presenter upravlja logikom aplikacije i poslovnom logikom. On dobiva informacije od Modela, obrađuje ih i priprema za prikaz u View-u. Presenter također prima događaje od View-a (kao što su korisnički inputi) i provodi odgovarajuće akcije (npr. ažuriranje Modela).

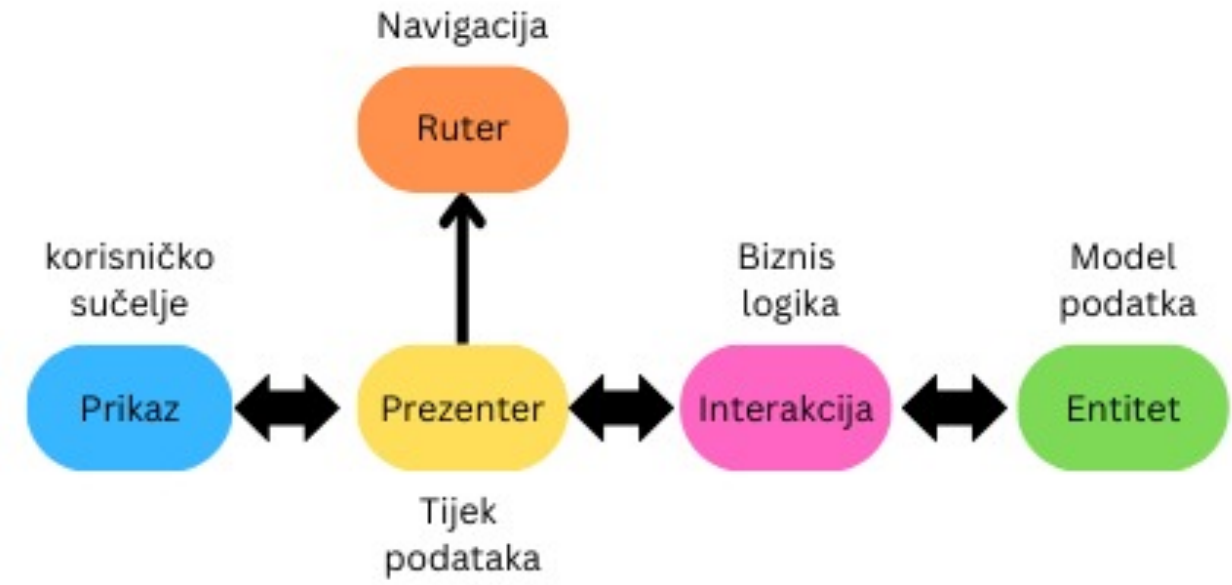
2.3.2. Prednosti MVP arhitekture

Prednosti MVP arhitekture su:

- **Razdvajanje odgovornosti**: MVP jasno razdvaja korisničko sučelje od komponente koja upravlja podacima. Ovo odvajanje olakšava održavanje aplikacije, testiranje i čitljivost koda jer svaka komponenta ima jasno definiranu ulogu.
- **Jednostavnije testiranje**: MVP arhitektura olakšava jedinično testiranje jer poslovna logika (koja se nalazi u Presenteru) je odvojena od sloja korisničkog sučelja (View). Ovo omogućuje razvojnicima da izoliraju i testiraju logiku aplikacije neovisno o korisničkom sučelju.

2.4. ARHITEKTURNI OBRAZAC VIPER

VIPER je napredni arhitekturni obrazac koji se često koristi u razvoju iOS aplikacija radi jasnog organiziranja i razdvajanja odgovornosti među komponentama aplikacije. Svaka komponenta u VIPER arhitekturi ima precizno definiranu ulogu: View je odgovoran za prikazivanje korisničkog sučelja i reagiranje na korisničke interakcije; Interactor sadrži poslovnu logiku i obrađuje podatke između različitih izvora; Presenter upravlja komunikacijom između View-a i Interactora te ažurira sučelje aplikacije s podacima; Entity predstavlja modele podataka bez poslovne logike, dok Router upravlja navigacijom između različitih dijelova aplikacije. VIPER arhitektura pruža visoku razinu skalabilnosti, modularnosti i olakšava jedinično testiranje zbog jasnog razdvajanja odgovornosti između komponenata.



Slika 2.4. VIPER ARHITEKTURA

2.4.1. Komponente VIPER obrasca

 View: Komponenta koja je odgovorna za prikazivanje korisničkog sučelja. U VIPER arhitekturi, View je pasivan i ne sadrži poslovnu logiku. Umjesto toga, komunicira s Presenterom putem protokola kako bi obavila prikazivanje podataka korisniku ili reagirala na korisničke interakcije.

 Interactor: Sadrži poslovnu logiku aplikacije. Interactor je odgovoran za obradu poslovnih zahtjeva, kao što su dohvaćanje podataka iz baze podataka ili mrežnih zahtjeva. Interactor također obrađuje podatke i priprema ih za prikazivanje kroz Presenter.

 Presenter: Djeluje kao posrednik između View i Interactora. Presenter prima ulazne podatke od View-a, obrađuje ih prema potrebama poslovne logike (Interactora) te zatim ažurira View s rezultatima. Presenter je također odgovoran za formatiranje podataka prije nego što ih proslijedi View-u.

 Entity: Odnosi se na modele podataka ili entitete koji se koriste unutar aplikacije. Ovi entiteti su obično jednostavni objekti podataka bez poslovne logike, često korišteni kao DTO (Data Transfer Objects).

 Router: Router je odgovoran za navigaciju između različitih modula ili zaslona u aplikaciji. U VIPER arhitekturi, Router zna o svim komponentama unutar modula (View, Interactor, Presenter) i odgovoran je za usmjeravanje korisničkih interakcija prema odgovarajućem zaslonu.

2.4.2. Prednosti VIPER arhitekture

 Jasno razdvajanje odgovornosti: Svaka komponenta (View, Interactor, Presenter) ima jasno definiranu ulogu, što olakšava održavanje i testiranje aplikacije.

 Skalabilnost: VIPER omogućuje lakše dodavanje novih funkcionalnosti ili modifikacija postojećih bez utjecaja na ostatak aplikacije.

 Testiranje: Zbog jasnog razdvajanja poslovne logike od korisničkog sučelja, jedinično testiranje (unit testing) postaje lakše i učinkovitije.

 Poboljšana modularnost: Moduli u VIPER arhitekturi su dobro definirani, što olakšava timsko suradnju i razvoj paralelno više funkcionalnosti.

Uvođenje i implementacija VIPER arhitekture može zahtijevati znatno više vremena i truda u usporedbi s jednostavnijim arhitekturnim obrascima kao što su MVC ili MVP. To je zbog složenije strukture i potrebe za detaljnom organizacijom svake komponente (View, Interactor, Presenter, Entity, Router). Unatoč tomu, za veće i kompleksnije iOS aplikacije, VIPER se često smatra optimalnim izborom zbog svoje sposobnosti da jasno razdvaja odgovornosti između komponenata, što olakšava održavanje, skaliranje i testiranje aplikacije na duži rok. VIPER omogućuje bolju modularnost i fleksibilnost u razvoju, što doprinosi dugoročnoj stabilnosti i poboljšava timsku suradnju u razvojnom procesu.

3. RAZVOJ IOS MOBILNIH APLIKACIJA

Ovo poglavlje fokusira se na tehnički aspekt razvoja iOS mobilnih aplikacija. Razmatra se korištenje programskog jezika Swift, integriranog razvojnog okruženja Xcode i iOS operativnog sustava. Posebna pozornost posvećena je automatskom upravljanju memorijom, smjernicama za korisničko sučelje, te procesu testiranja aplikacija. Također je obrađena distribucijska platforma App Store, koja omogućava razvijateljima distribuciju aplikacija

3.1. Programski jezik Swift

Programski jezik Swift, razvijen od strane Apple-a i predstavljen 2014. godine kao nasljednik Objective-C, ističe se svojom kombinacijom sigurnosti, brzine izvršavanja i modernog sintaksnog pristupa koji olakšava razvoj aplikacija za iOS, macOS, watchOS i tvOS platforme. Swift koristi inferenciju tipova kako bi smanjio potrebu za eksplicitnim navođenjem tipova, što doprinosi čitljivosti i produktivnosti kodiranja. Ovaj programski jezik nudi podršku za više različitih programskih paradigmi, kao što su objektno-orijentirana, funkcionalna i protokolno-orijentirana, omogućujući developerima širok spektar pristupa u pronalaženju optimalnih rješenja za različite izazove. Jedna od ključnih karakteristika Swifta je automatsko upravljanje memorijom , što olakšava upravljanje resursima i smanjuje rizik od curenja memorije. Apple redovito ažurira Swift s novim verzijama koje donose poboljšanja i novitete, potičući kontinuirani razvoj jezika i integraciju novih tehnologija. Swift je postao popularan među razvojnim zajednicama zbog svoje intuitivne sintakse, podrške za moderne tehnologije poput machine learninga i podrške za razvoj aplikacija na više platformi unutar Apple ekosustava. Swift je također otvorenog koda, što potiče zajednicu programera da doprinosi njegovom razvoju i poboljšanjima. Ova otvorenost omogućava programerima da prilagode jezik svojim potrebama i integriraju dodatne biblioteke i alate. Swift podržava razvoj server-side aplikacija putem različitih frameworka kao što su Vapor, Kitura i Perfect, što proširuje njegovu primjenu izvan tradicionalnog mobilnog i desktop razvoja. Jezik je poznat po svojoj visokoj performansama, što ga čini idealnim izborom za razvoj aplikacija koje zahtijevaju brzo izvršavanje i efikasno upravljanje resursima. Swift[3] također podržava moderni pristup asinkronog programiranja putem svojih API-ja i konstrukata jezika kao što su async/await, što olakšava razvoj aplikacija koje se integriraju s mrežnim servisima i bazama podataka.

Slika 3. SWIFT-LOGO

3.2. Integrirano razvojno okruženje Xcode

Xcode je integrirano razvojno okruženje koje je Apple razvio kao ključni alat za razvoj aplikacija na svojim platformama, uključujući iOS, macOS, watchOS i tvOS. Ovo sveobuhvatno okruženje podržava sve faze razvoja softvera, počevši od dizajniranja i programiranja, pa sve do testiranja i distribucije aplikacija. Xcode je specijaliziran za rad s jezicima kao što su Swift, Objective-C, C++ i drugi jezici podržani na Apple platformama, pružajući napredni editor koda s funkcionalnostima kao što su sintaksno bojanje, automatsko dovršavanje koda i refaktoriranje. Također sadrži vizualne alate poput Storyboarda za dizajniranje korisničkog sučelja putem jednostavnog drag-and-drop pristupa.[4]

Integracija s Simulatorom omogućava programerima simuliranje različitih Apple uređaja radi testiranja aplikacija u kontroliranom okruženju. Xcode uključuje i alate za profiliranje i analizu performansi aplikacija, koji pomažu u optimizaciji koda i otkrivanju problema s performansama. Za razvoj server-side aplikacija, Xcode podržava različite alate i frameworkove kao što su Vapor, Kitura i Perfect.

Osim toga, Xcode omogućuje jednostavnu integraciju s Apple-ovim servisima poput iCloud-a, In-App Purchases, Game Center-a i drugih. Podržava različite vrste projekata i aplikacija, uključujući iOS aplikacije, macOS aplikacije, aplikacije za Apple Watch i Apple TV, kao i kombinirane projekte. Xcode također nudi mogućnosti za automatizaciju procesa kroz skriptiranje i integraciju s alatima za kontinuiranu integraciju kao što su Jenkins ili Xcode Server.

Redovito ažuriranje Xcode s novim verzijama donosi poboljšanja u performansama, nove funkcionalnosti i podršku za najnovije verzije operativnih sustava Apple platformi. S ovim besplatno dostupnim alatom putem Mac App Store-a, programeri mogu upravljati resursima

aplikacija kao što su lokalizacija, upravljanje ikonama i launch screen-ovima. Xcode kao centralni alat za razvoj na Apple platformama ključan je za stvaranje visokokvalitetnih aplikacija prilagođenih Apple ekosustavu, od jednostavnih mobilnih igara do složenih poslovnih aplikacija.

3.3. Operacijski sustav – IOS

iOS je operacijski sustav razvijen od strane tvrtke Apple za svoje mobilne uređaje, uključujući iPhone, iPad i iPod Touch. Ovaj operacijski sustav je poznat po svojoj stabilnosti, sigurnosti i integraciji s Apple-ovim ekosustavom aplikacija i usluga. iOS omogućuje korisnicima pristup širokom spektru aplikacija putem App Store-a, koji nudi milijune aplikacija za različite svrhe, uključujući igre, produktivnost, zabavu i druge. Sučelje iOS-a karakterizira jednostavnost i intuitivnost, što olakšava navigaciju i korištenje aplikacija na mobilnim uređajima. iOS redovito dobiva nadogradnje s novim značajkama, poboljšanjima performansi i sigurnosnim ažuriranjima koje Apple distribuira svim podržanim uređajima. Osim aplikacija, iOS podržava integraciju s Apple-ovim uslugama kao što su iCloud za pohranu i sinkronizaciju podataka te Apple Pay za sigurno plaćanje putem mobilnih uređaja. Operacijski sustav iOS također je poznat po svojoj podršci za prilagodbu i pristupačnost, pružajući korisnicima mogućnost personalizacije sučelja i pristup alatima za olakšavanje upotrebe osobama s posebnim potrebama. Sustav također nudi napredne funkcionalnosti poput Siri, Apple-ove digitalne asistentice, koja omogućuje glasovnu kontrolu i interakciju s uređajem. iOS je također poznat po svojoj visokoj razini privatnosti i sigurnosti podataka korisnika, što je ključni faktor u popularnosti platforme. Apple redovito radi na poboljšanju sigurnosti operativnog sustava kroz enkripciju podataka i stroge smjernice za aplikacije u App Store-u. Korisnici iOS-a imaju pristup ekskluzivnim značajkama kao što su AirDrop za jednostavan prijenos datoteka između uređaja te Handoff za kontinuirano radno iskustvo između iOS uređaja i Mac računala. Sustav podržava najnovije tehnologije poput proširena stvarnost *(eng. Augmented Reality)* i mašinsko učenje *(eng. Machine Learning)*, što omogućuje razvoj inovativnih aplikacija i iskustava. iOS je postao sinonim za mobilnu produktivnost, zabavu i sigurnost, te nastavlja postavljati standarde u industriji mobilnih operativnih sustava[5].

Slika 3.1. APPLE-LOGO [13]

3.4. Testiranje IOS aplikacija

Testiranje iOS aplikacija ključno je za osiguravanje njihove funkcionalnosti, performansi i stabilnosti prije nego što budu puštene u produkciju. Razvojni inženjeri koriste različite metode i alate kako bi osigurali da aplikacije rade kako treba na različitim uređajima i iOS verzijama. Apple-ovo integrirano razvojno okruženje (integrated development environment-IDE), Xcode, pruža snažan radni okvir za testiranje poznat kao XCTest, koji podržava unit testove i testove korisničkog sučelja (UI testove). XCTest omogućuje automatsko izvršavanje testova unutar Xcode-a, što značajno olakšava provođenje testova tijekom razvojnog procesa.

Za automatizaciju testiranja iOS aplikacija, popularan alat je Appium, koji je open-source okvir za testiranje na više platformi. Appium podržava različite programerske jezike i omogućuje razvojnicima da pišu testove koji simuliraju korisničke interakcije s aplikacijom, kao što su klikovi, geste ili unos teksta. Ovaj okvir koristi WebDriver protokol za komunikaciju s aplikacijama putem UI, čime omogućuje detaljno testiranje funkcionalnosti.

Pri testiranju iOS aplikacija, važno je također provjeriti njihovu kompatibilnost s različitim verzijama iOS-a i različitim uređajima. Testiranje u stvarnom okruženju (real device testing) može pružiti preciznije rezultate u usporedbi s simuliranim okruženjem, posebno kad je potrebno provjeriti performanse i responzivnost aplikacije.

Korisničko sučelje aplikacije treba temeljito testirati kako bi se osiguralo da je intuitivno za krajnje korisnike i da pravilno reagira na različite ulaze i scenarije upotrebe. Testiranje sigurnosti aplikacije također je ključno kako bi se spriječile ranjivosti i zaštitile korisničke podatke[9].

Konačno, kontinuirano integracijsko testiranje igra važnu ulogu u osiguravanju da svaka promjena u kodu ne utječe negativno na postojeću funkcionalnost aplikacije te da novi feature-i ne uzrokuju neočekivane probleme. Integracija s alatima poput Jenkinsa ili GitLab omogućuje automatizirano izvršavanje testova pri svakoj promjeni u kodu, čime se osigurava stabilnost i kvaliteta aplikacije prije njezinog puštanja u produkciju.

3.5. Smjernice za korisničko sučelje

Smjernice za korisničko sučelje su ključni elementi u razvoju aplikacija koje imaju za cilj poboljšati korisničko iskustvo i funkcionalnost softvera. One obuhvaćaju skup pravila, načela i preporuka koje definiraju kako bi trebalo izgledati i ponašati se korisničko sučelje kako bi bilo intuitivno za korištenje. Ove smjernice pomažu razvojnim timovima da stvore sučelja koja su dosljedna, funkcionalna i laka za navigaciju, bez obzira na platformu ili uređaj na kojem se koriste.

Jedan od ključnih aspekata smjernica za korisničko sučelje je vizualni dizajn, koji uključuje upotrebu boja, tipografije, ikona i ostalih vizualnih elemenata kako bi se postigao jasan i atraktivan izgled. Hijerarhijska struktura navigacije je također važan element, jer pomaže korisnicima da lako pronalaze i koriste funkcionalnosti aplikacije.

Osim estetskog aspekta, smjernice također naglašavaju funkcionalnost sučelja, uključujući brzinu odziva, intuitivnost gesti i upravljanje elementima korisničkog sučelja kao što su gumbi, izbornici i dijaloški okviri. Ova funkcionalnost je ključna za stvaranje ugodnog korisničkog iskustva bez frustracija ili poteškoća.

Važan dio smjernica za korisničko sučelje je također prilagodba aplikacije specifičnim zahtjevima platforme na kojoj se koristi, kao što su iOS, Android, Windows ili web platforme. Svaka platforma ima svoje karakteristike i pravila koja treba poštovati kako bi se osiguralo dosljedno iskustvo za korisnike.

Pristupačnost je još jedan ključan element smjernica za korisničko sučelje, jer se naglašava potreba da su aplikacije dostupne svim korisnicima, uključujući one s invaliditetom. To

uključuje podršku za alate kao što su screen readeri, povećanje fontova i podršku za prilagođavanje boja.

Konačno, smjernice za korisničko sučelje su dinamičan dokument koji se često ažurira s novim tehnologijama, trendovima i feedbackom korisnika kako bi se osiguralo da aplikacije i dalje pružaju optimalno korisničko iskustvo. Integracija smjernica u razvojni proces ključna je za stvaranje uspješnih aplikacija koje korisnici vole i redovito koriste.

3.6. App Store – distribucijska platforma

App Store je globalna distribucijska platforma koju je razvila tvrtka Apple za distribuciju mobilnih aplikacija namijenjenih iOS uređajima poput iPhonea i iPada. Ova platforma omogućuje developerima da svoje aplikacije ponude milijunima korisnika širom svijeta, što čini App Store ključnim kanalom za distribuciju mobilnih softverskih rješenja. Apple strogo provjerava aplikacije prije nego što ih dopusti u App Store, osiguravajući tako da su aplikacije sigurne, stabilne i u skladu s Apple-ovim smjernicama.[6]

App Store pruža korisnicima širok izbor aplikacija u različitim kategorijama kao što su igre, produktivnost, zdravlje, obrazovanje i mnoge druge, olakšavajući pronalaženje novih i korisnih aplikacija. Razvojni timovi mogu zaraditi putem App Store-a prodajom aplikacija, pretplatama, in-app kupnjama ili oglašavanjem, što čini platformu atraktivnom za monetizaciju mobilnih aplikacija.

Korisnici App Store-a mogu ocjenjivati i recenzirati aplikacije, pružajući povratne informacije koje pomažu drugim korisnicima pri odabiru aplikacija. Apple redovito ažurira App Store s novim značajkama i poboljšanjima kako bi poboljšao korisničko iskustvo i olakšao otkrivanje novih aplikacija.

Jedna od ključnih funkcionalnosti App Store-a je mogućnost automatskog ažuriranja aplikacija, što korisnicima omogućuje da uvijek imaju najnoviju verziju svojih aplikacija bez potrebe za ručnim ažuriranjem. Osim toga, App Store pruža alate za razvojnim timovima kao što su

analitički alati za praćenje performansi aplikacija i marketinške kampanje za promociju novih aplikacija.

Apple nastoji održavati visoke standarde sigurnosti i privatnosti na App Store-u, što uključuje pravila o zaštiti podataka korisnika i sprečavanju zloupotrebe osobnih informacija. Platforma također podržava lokalizaciju aplikacija, omogućujući developerima da svoje aplikacije prilagode različitim jezicima i regionalnim preferencijama korisnika.Uz rastući broj korisnika iOS uređaja i stalno širenje globalne dostupnosti, App Store ostaje ključni dio ekosustava Apple-ovih proizvoda i usluga. Integracija s drugim Apple-ovim servisima kao što su iCloud, Apple Pay i Apple Music dodatno obogaćuje korisničko iskustvo i olakšava interakciju između aplikacija i uređaja [7].

4. POSTOJEĆA SLIČNA PROGRAMSKA RJEŠENJA

Poglavlje analizira trenutno stanje u području programskih rješenja, posebno mobilnih aplikacija u turističkoj industriji. Spomenute su popularne aplikacije poput TripAdvisora, Google Mapsa i Airbnb-a, koje nude funkcionalnosti poput interaktivnih karata, preporuka i mogućnosti rezervacije. Rad uspoređuje te aplikacije s razvijenom iOS aplikacijom koja integrira napredne funkcionalnosti kao što su Text-to-Speech, skeniranje QR kodova i filtriranje sadržaja

4.1. Stanje u području programskih rješenja

Razvoj mobilnih aplikacija za turizam postao je sve popularniji zbog porasta upotrebe pametnih telefona i povećane potražnje za digitalnim alatima koji unapređuju turističko iskustvo. U ovom kontekstu, mnoge postojeće aplikacije nude korisnicima mogućnosti poput interaktivnih karata, vodiča kroz turističke destinacije, personaliziranih preporuka, kao i mogućnost rezerviranja smještaja i aktivnosti.

Jedan od ključnih trendova u ovom području je integracija naprednih tehnologija kao što su proširena stvarnost (AR) i umjetna inteligencija (AI), koje omogućuju stvaranje još interaktivnijih i prilagođenijih korisničkih iskustava. AR tehnologija, na primjer, omogućuje korisnicima da putem kamere svojih uređaja dobiju dodatne informacije o turističkim atrakcijama u stvarnom vremenu, dok AI pruža personalizirane preporuke na temelju korisničkih preferencija i ponašanja.

Većina postojećih aplikacija u ovom području nudi osnovne funkcionalnosti poput prikaza turističkih atrakcija na karti, navigacije do tih atrakcija, kao i pristup informacijama o povijesti, kulturi i drugim zanimljivostima vezanim uz destinacije. Međutim, često se primjećuje da ove aplikacije pate od problema kao što su neintuitivno korisničko sučelje, manjak personalizacije, te ograničene mogućnosti filtriranja informacija prema korisničkim interesima.

Uz sve to, rastuća upotreba glasovnih asistenata i funkcionalnosti poput Text-to-Speech (TTS)[8] dodatno poboljšava pristupačnost aplikacija, omogućujući korisnicima da dobiju informacije bez potrebe za gledanjem u ekran, što je posebno korisno tijekom vožnje ili šetnje.

U kontekstu ovih postojećih rješenja, naša turistička iOS aplikacija donosi značajna unapređenja u vidu intuitivnog korisničkog sučelja, naprednog filtriranja atrakcija prema interesima korisnika, te integracije TTS funkcionalnosti koja omogućuje hands-free iskustvo. Dodatno, naša aplikacija uključuje mogućnost skeniranja QR kodova, što omogućuje brzo i

jednostavno preuzimanje informacija o znamenitostima ili pristupanje specifičnim sadržajima unutar aplikacije. Ova funkcionalnost doprinosi većem stupnju interakcije i personalizacije korisničkog iskustva, pružajući korisnicima priliku da dobiju dodatne informacije skeniranjem kodova postavljenih na stvarnim lokacijama.
Nadalje, korištenje MVVM arhitekture osigurava lakše održavanje i skaliranje aplikacije, što doprinosi njezinoj dugoročnoj održivosti i mogućnosti proširenja funkcionalnosti u budućnosti.

4.2. Postojeća programska rješenja

U današnjem digitalnom svijetu, mobilne aplikacije za turizam igraju ključnu ulogu u oblikovanju i poboljšanju iskustava putnika. Postojeća programska rješenja u ovoj domeni već nude širok spektar funkcionalnosti koje omogućuju korisnicima da istražuju destinacije, planiraju putovanja i dobiju personalizirane preporuke. Ove aplikacije često integriraju interaktivne karte, vodiče kroz turističke destinacije, te opcije za rezervaciju smještaja i aktivnosti.
U tom kontekstu, naša turistička iOS aplikacija nastoji unaprijediti postojeće pristupe integriranjem naprednih funkcionalnosti kao što su Text-to-Speech, QR kod skeniranje i dinamičko filtriranje sadržaja, dok se pri tome oslanja na MVVM arhitekturu za bolje održavanje i skalabilnost. Korištenje modernih tehnologija i naglasak na personalizaciju omogućava nam da pružimo superiorno korisničko iskustvo u odnosu na postojeća rješenja na tržištu.

4.2.1. TripAdvisor

TripAdvisor je vodeća globalna platforma za planiranje i rezervaciju putovanja, poznata po svojim sveobuhvatnim recenzijama, ocjenama i preporukama za hotele, restorane i turističke atrakcije diljem svijeta. Aplikacija korisnicima pruža bogatstvo informacija, uključujući milijune autentičnih recenzija i fotografija koje omogućuju donošenje informiranih odluka.

Uz integrirane funkcionalnosti interaktivnih karata i personaliziranih itinerera, TripAdvisor omogućuje jednostavno planiranje putovanja prema individualnim preferencijama korisnika.
Aplikacija također nudi mogućnost direktne rezervacije smještaja, letova i aktivnosti, čime pojednostavljuje cijeli proces putovanja.

S naprednim algoritmima za personalizaciju, TripAdvisor nudi preporuke temeljene na povijesti pretraživanja i preferencijama korisnika. Iako nudi bogatstvo informacija, korisničko sučelje aplikacije je intuitivno i jednostavno za navigaciju. Kao jedan od najpouzdanijih izvora za planiranje putovanja, TripAdvisor surađuje s brojnim partnerima kako bi korisnicima osigurao najbolje ponude i iskustva, pružajući sveobuhvatan alat za planiranje putovanja na jednom mjestu.

Slika 4. TripAdvisor logo [14]

4.2.2. Google Maps

Google Maps je jedna od najpopularnijih i najčešće korištenih navigacijskih aplikacija na svijetu, pružajući korisnicima precizne karte, upute za vožnju, hodanje, javni prijevoz, kao i informacije o prometu u stvarnom vremenu. Osim osnovnih navigacijskih funkcionalnosti, Google Maps omogućuje korisnicima istraživanje okolnih zanimljivosti, poput restorana, hotela, trgovina i turističkih atrakcija, uz korisničke recenzije i ocjene koje pomažu pri donošenju odluka.

Aplikacija također nudi integraciju s raznim uslugama poput Google Street View, koja omogućuje pregledavanje ulica u 360 stupnjeva, kao i mogućnost preuzimanja offline karata za korištenje bez internetske veze. Google Maps koristi napredne algoritme za optimizaciju ruta, uzimajući u obzir prometne uvjete, radove na cestama i druge prepreke, kako bi korisnicima pružila najbrže i najefikasnije rute do njihovih odredišta.

Pomoću funkcionalnosti poput "Explore" korisnici mogu otkriti nove destinacije, a opcija "Timeline" omogućuje pregled prethodnih putovanja i posjećenih mjesta. Aplikacija je postala neizostavan alat za svakodnevno kretanje, planiranje putovanja i istraživanje novih područja, zbog čega je široko prihvaćena kao pouzdan vodič u digitalnom svijetu.

Slika 4.1. Google Maps logo [15]

4.3.3. Airbnb

Airbnb je globalna online platforma koja omogućuje korisnicima da iznajmljuju smještaj, poput stanova, kuća, vila ili jedinstvenih lokacija, poput kućica na drvetu ili dvoraca, direktno od vlasnika. Aplikacija povezuje putnike s domaćinima, pružajući im mogućnost da biraju između različitih vrsta smještaja diljem svijeta, često po povoljnijim cijenama u usporedbi s tradicionalnim hotelima. Airbnb također nudi dodatne usluge, poput "Airbnb Experiences," gdje korisnici mogu rezervirati aktivnosti, obilaske ili jedinstvena iskustva koja nude lokalni stanovnici.

Korisničke recenzije i ocjene pomažu u osiguravanju povjerenja i sigurnosti, dok napredni sustavi filtriranja omogućuju pronalaženje smještaja prema specifičnim kriterijima, poput cijene, lokacije, dostupnosti pogodnosti i drugih preferencija. Platforma se ističe i po fleksibilnosti, omogućujući korisnicima da prilagode svoja putovanja prema individualnim potrebama, dok domaćini imaju mogućnost dodatne zarade iznajmljivanjem svojih nekretnina.

Airbnb je prepoznatljiv po svojoj inovativnosti u turističkoj industriji, značajno mijenjajući način na koji ljudi putuju i doživljavaju nova mjesta, te je postao sinonim za fleksibilno i autentično putovanje.

Slika 4.2. AirBnB logo [16]

5. ARHITEKTURA PROGRAMSKOG RJEŠENJA

Ovo poglavlje opisuje funkcionalne zahtjeve i arhitekturu aplikacije. Prikazani su dijagrami tijeka rada aplikacije, opisane su glavne funkcionalnosti kao što su istraživanje znamenitosti i filtriranje prema kategorijama. Također se razmatra korištenje Firebase baze podataka za pohranu podataka u stvarnom vremenu te implementacija MVVM arhitekture koja osigurava modularnost i lakše održavanje aplikacije

5.1. Funkcionalni zahtjevi na aplikaciju

U praktičnom dijelu ovog projekta bit će razvijena mobilna iOS aplikacija koja će korisnicima pružiti sveobuhvatan vodič kroz grad Đakovo. Aplikacija će omogućiti korisnicima istraživanje raznih znamenitosti, skeniranje QR kodova za trenutačno dobivanje informacija o određenim lokacijama te filtriranje znamenitosti prema kategorijama. Osnovni funkcionalni zahtjevi pokrivaju sve slučajeve koje aplikacija mora zadovoljiti te olakšavaju upravljanje samom aplikacijom. Za ovu aplikaciju, funkcionalni zahtjevi su:

- **Istraživanje znamenistosti:**
    - Popuniti bazu podataka informacijama o raznim znamenitostima u Đakovu, uključujući slike, opise i geografske podatke.
    - Prikazati znamenitosti na interaktivnoj karti koristeći različite markere za različite kategorije znamenitosti (npr. povijesne, kulturne, prirodne).
    - Omogućiti korisnicima da dodirnu marker na karti kako bi vidjeli detaljne informacije o specifičnoj znamenitosti, uključujući slike, opise i druge relevantne podatke

- **Integracija QR koda:**

    - Generirati i obraditi QR kodove koji izravno povezuju sa specifičnim znamenitostima unutar aplikacije koristeći prilagođenu URL shemu.
    - Omogućiti korisnicima skeniranje QR kodova za brzo pristupanje informacijama o određenoj znamenitosti.
    - Osigurati da skenirani QR kod otvori ispravan prikaz detalja o znamenitosti, čak i ako aplikacija nije bila već otvorena.

- **Filtriranje prema kategorijama:**

    - Pružiti izbornik korisnicima za filtriranje znamenitosti prema kategorijama (npr. Povijesne, Glazbene, Sportske, Osnovne).
    - Omogućiti korisnicima da odaberu kategoriju i pregledaju samo znamenitosti koje pripadaju toj kategoriji na karti.
    - Implementirati opciju "Sve" za uklanjanje aktivnih filtera i prikaz svih znamenitosti.

- **Korisnička interakcija:**

    - Omogućiti korisnicima dobivanje uputa za dolazak do odabrane znamenitosti putem integriranih usluga karte.
    - Pružiti funkcionalnost pretvorbe teksta u govor koja će čitati opis odabrane znamenitosti.

- **Korisničko sučelje:**

    - Dizajnirati korisničko sučelje koje je jednostavno za korištenje s jasnim rasporedom i lakom navigacijom.
    - Osigurati da aplikacija bude responzivna i vizualno privlačna na različitim iOS uređajima i veličinama zaslona.

- **Upravljanje bazom podataka:**

    - Spremiti sve korisničke interakcije, kao što su odabrane kategorije i posjećene znamenitosti, u bazu podataka.
    - Osigurati da se sve informacije o znamenitostima pohranjuju i preuzimaju iz pouzdane i sigurne baze podataka u oblaku.

- **Performanse i popuzdanost**

    - Osigurati da aplikacija učinkovito učitava podatke, posebno kada prikazuje više znamenitosti na karti.
    - Rukovati slučajevima gubitka mrežne povezanosti predmemoriranjem podataka lokalno za nastavak korištenja.

Neophodno je osigurati da svaki specificirani zahtjev bude pravilno realiziran kako bi korisnici imali potpuni doživljaj prilikom korištenja aplikacije. Nakon što razvoj aplikacije bude završen, ti funkcionalni zahtjevi bit će ključni za provjeru i potvrdu ispravnog rada svih funkcionalnosti. U narednim poglavljima detaljno će biti opisani koraci implementacije te objašnjeni ključni dijelovi koda.

5.2. Dijagram tijeka aplikacije

Dijagram tijeka prikazuje osnovne korake i tok interakcija unutar turističke iOS aplikacije. Dijagram počinje s početkom rada aplikacije, nakon čega se provjerava dostupnost lokacije korisnika i dohvaćanje podataka iz Firebase baze podataka. Ako je lokacija uspješno pronađena, korisnik se preusmjerava na početni zaslon, gdje može vidjeti popis znamenitosti prikazanih na karti.

Na početnom zaslonu, korisnik ima nekoliko opcija interakcije:

1. **Klik na filter menu button:** Ako korisnik klikne na gumb za filter, otvara se zaslon s filtrima, gdje korisnik može odabrati kategorije znamenitosti koje želi vidjeti. Nakon odabira filtera, aplikacija se vraća na početni zaslon s ažuriranim prikazom znamenitosti.

2. **Klik na ping znamenitosti:** Kada korisnik odabere neku od označenih znamenitosti na karti, otvara se zaslon s detaljima te znamenitosti. Na ovom zaslonu korisnik može vidjeti informacije kao što su opis, lokacija i druge relevantne detalje.

3. **Klik na gumb QR code:** Ako korisnik odluči skenirati QR kod, otvara se zaslon za skeniranje. Nakon što aplikacija prepozna QR kod, dohvaća se odgovarajući sadržaj iz baze podataka i korisniku se prikazuje zaslon s detaljima.

Na zaslonu s detaljima znamenitosti, korisnik ima dvije glavne opcije:

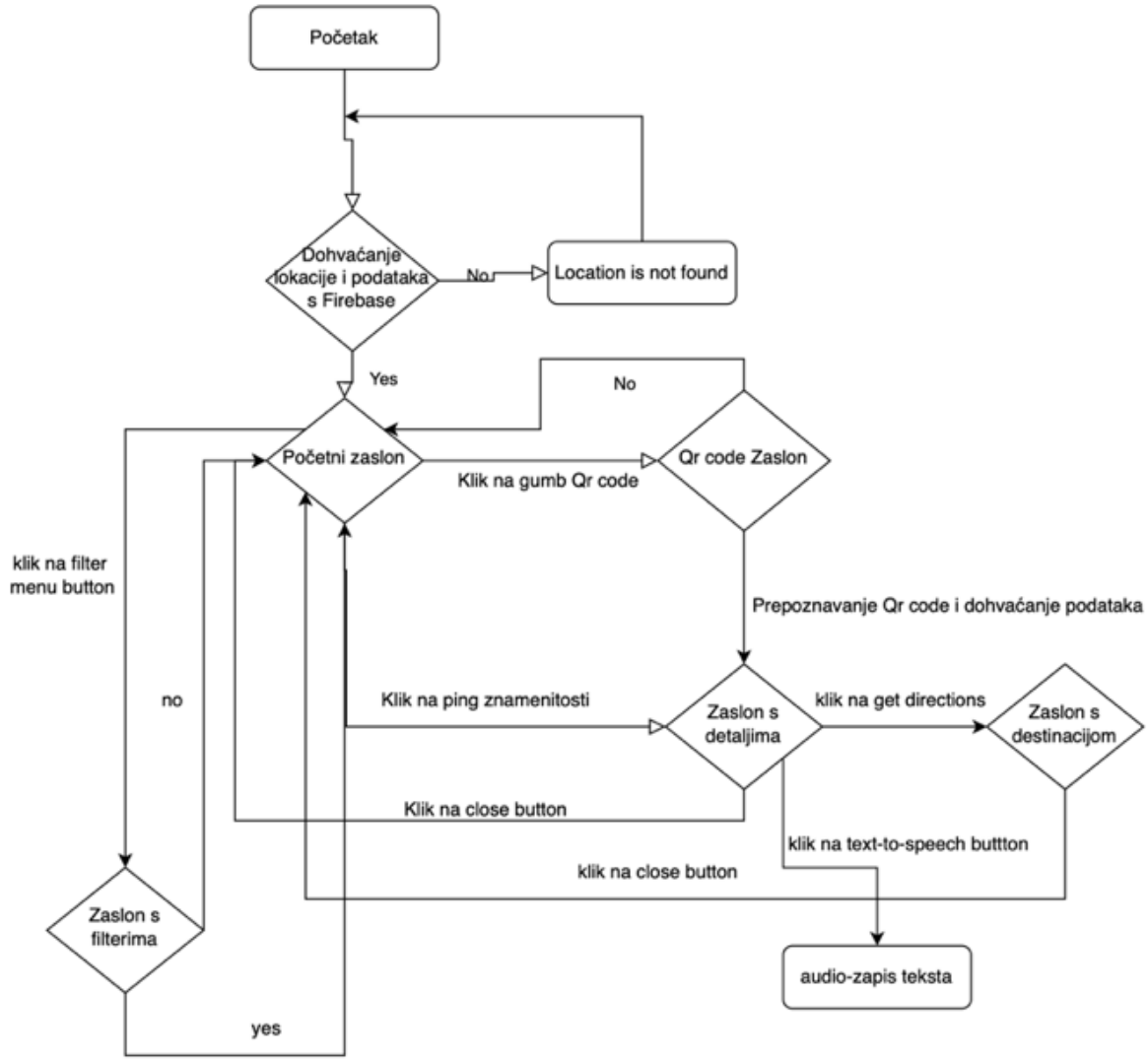
- **Klik na gumb get directions:** Ova opcija vodi korisnika na zaslon s destinacijom, gdje može pregledati put do odabrane znamenitosti s opcijama različitih ruta i načina prijevoza.

- **Klik na gumb text-to-speech:** Ova opcija omogućuje korisniku slušanje audio-zapisa opisa znamenitosti, što je korisno u situacijama kada korisnik ne može gledati u zaslon.

Nakon što korisnik završi s pregledom ili interakcijom unutar aplikacije, može jednostavno kliknuti na gumb za zatvaranje kako bi se vratio na početni zaslon ili drugi odgovarajući zaslon, ovisno o svojoj aktivnosti. Ova funkcionalnost omogućava glatku i nesmetanu navigaciju kroz aplikaciju, bez potrebe za dodatnim koracima ili složenim izbornicima. Dijagram tijeka aplikacije jasno prikazuje sve ključne funkcionalnosti, pomažući korisniku da razumije glavne korake koje može poduzeti prilikom korištenja aplikacije. Svaka odluka koju korisnik donese bilo da je riječ o pregledavanju detalja znamenitosti, filtriranju sadržaja ili korištenju QR koda vodi do konkretne promjene u prikazu aplikacije.

Promjene u prikazu su intuitivne i prilagođene korisnikovim akcijama, čime se osigurava lako razumljiv i logičan tijek korištenja. Bilo da se korisnik vraća na početni zaslon ili na određeni zaslon unutar aplikacije, svaki korak je osmišljen da smanji mogućnost zbunjenosti i omogući jednostavan prijelaz između različitih dijelova aplikacije. Ova pažljivo strukturirana navigacija osigurava da korisnik bez poteškoća može pristupiti svim funkcionalnostima, te da svaka

njegova akcija donosi jasno vidljive rezultate na sučelju. Intuitivan tijek korisničkog iskustva ključan je za ugodno i efikasno korištenje aplikacije, pružajući korisnicima osjećaj kontrole i jednostavnosti.



Slika 5. .Dijagram tijeka aplikacije

5.3. Firebase baza podataka

Firebase, platforma [10] koju je razvio Google 2012. godine, osmišljena je kako bi pojednostavila razvoj, održavanje i skaliranje mobilnih aplikacija, uključujući i turističke aplikacije poput ove koja pokriva znamenitosti grada Đakova. Kao skup alata baziranih u oblaku, Firebase predstavlja izvrstan primjer BaaS (engl. *Backend as a Service*) rješenja,

omogućavajući brži razvoj aplikacija uz korištenje unaprijed definiranih alata i API-ja. U kontekstu ove turističke aplikacije, Firebase pruža ključne funkcionalnosti koje su bitne za njezino nesmetano funkcioniranje:

- **Autentifikacija** - Firebase osigurava sigurno i jednostavno upravljanje korisničkim računima. Korisnici aplikacije mogu brzo kreirati račune i prijaviti se putem svojih vjerodajnica, što je ključno za personalizirano iskustvo unutar aplikacije.
- **Crashlytics** - Ovaj alat omogućuje praćenje stabilnosti aplikacije, nudeći detaljne izvještaje o bilo kakvim padovima aplikacije. Programeri mogu brzo identificirati i otkloniti probleme, osiguravajući korisnicima neprekidno i stabilno korištenje aplikacije.
- **Praćenje performansi** - Firebase pruža alat za praćenje performansi aplikacije, omogućujući programerima uvid u to kako aplikacija koristi resurse kao što su procesor, memorija i mrežni promet. To pomaže u optimizaciji aplikacije kako bi bila brza i responzivna, čak i kada korisnici pregledavaju više znamenitosti na karti.
- **Firebase Cloud Messaging** - Korištenjem FCM-a, vlasnici aplikacije mogu slati obavijesti korisnicima o novim znamenitostima, posebnim događajima ili promocijama, čime se poboljšava korisničko iskustvo i angažman.
- **Baza podataka u stvarnom vremenu** - Ova značajka omogućava aplikaciji trenutnu pohranu i sinkronizaciju podataka o znamenitostima u Đakovu. Kada se dodaju novi podaci ili se ažuriraju postojeći, korisnici aplikacije odmah dobivaju pristup najnovijim informacijama, što je posebno važno za točne i ažurirane podatke o turističkim lokacijama.

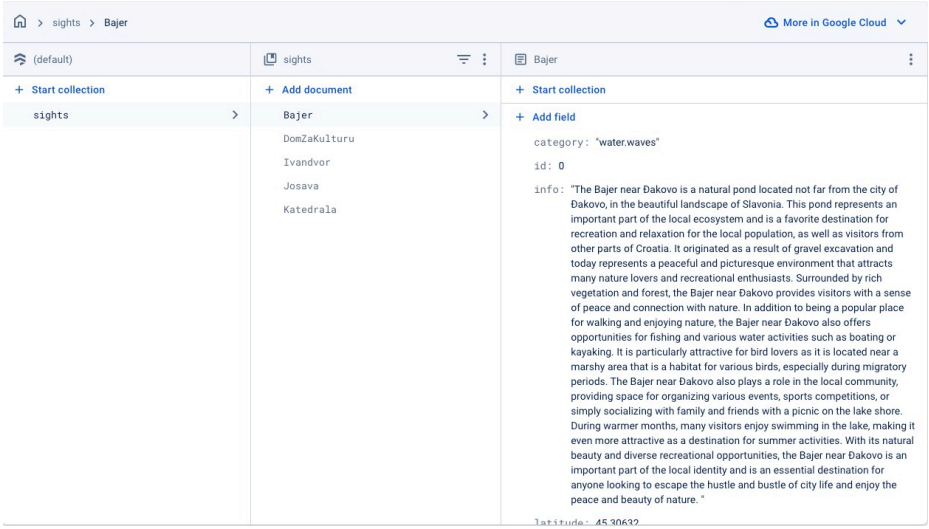
5.3.1. Firebase usluga baze podataka

Firebase usluga baze podataka u oblaku (engl. Firestore) pruža fleksibilno i skalabilno rješenje za pohranu različitih podataka koje pripadaju u NoSQL grupu baza podataka. Za potrebe ovog rada, ova usluga će se koristiti kako bi se pohranili podaci o znamenitostima grada Đakova, uključujući informacije kao što su naziv, opis, kategorija, geografske koordinate, slike i ostali relevantni podaci.

Firestore omogućava jednostavno rukovanje i upravljanje podacima te omogućava aplikaciji da u stvarnom vremenu dohvaća i prikazuje ažurirane informacije o znamenitostima. To

omogućava korisnicima da uvijek imaju pristup najnovijim podacima bez obzira na to kada su zadnji put otvorili aplikaciju. Također, Firestore omogućava filtriranje i pretraživanje znamenitosti prema kategorijama, što olakšava korisnicima navigaciju kroz aplikaciju.
Jedna od glavnih prednosti ove usluge je sinkronizacija podataka u stvarnom vremenu na svim uređajima koji koriste aplikaciju. Firestore također pruža podršku za rad izvan mreže, što znači da aplikacija ostaje funkcionalna čak i kada korisnik nema pristup internetu. Podaci se lokalno spremaju i sinkroniziraju kada se ponovno uspostavi mrežna veza, što značajno povećava responzivnost aplikacije.
Glavne značajke i prednosti Firestore usluge u kontekstu ove aplikacije su:

- **Prilagodljivost** - Firestore omogućuje organizaciju podataka kroz hijerarhijske strukture, pohranjujući ih unutar dokumenata koji mogu imati različite objekte i formate. Ova značajka posebno je korisna za kategorizaciju podataka, primjerice o znamenitostima.
- **Napredni upiti** - Firestore također nudi mogućnost preciznog pretraživanja dokumenata u bazi podataka koristeći sofisticirane upite. Parametri upita mogu se prilagoditi kako bi se dobili ciljani rezultati.
- **Ažurnost** - Aplikacija putem Firestore-a može automatski osvježavati podatke na svim povezanim uređajima, pružajući korisnicima trenutni prikaz novih ili izmijenjenih informacija o znamenitostima bez potrebe za ručnim osvježavanjem.
- **Offline funkcionalnost** - Aktivno korišteni podaci spremaju se lokalno na uređaj, omogućujući korisnicima rad čak i bez internetske veze. Time se osigurava neprekinuto korisničko iskustvo, jer aplikacija ostaje operativna i u offline načinu rada.



Slika 5.1 Sučelje FireBase usluge

5.4. MVVM Arhitektura mobilne aplikacije

Prilikom izrade ove turističke iOS aplikacije primijenjena je MVVM (Model-View-ViewModel) arhitektura, koja je danas općeprihvaćena u razvoju mobilnih aplikacija zahvaljujući svojoj sposobnosti da učinkovito razdvoji poslovnu logiku od korisničkog sučelja. Ovaj arhitektonski obrazac pruža visoku razinu modularnosti, značajno olakšava održavanje koda, poboljšava testabilnost te osigurava dugoročno skaliranje aplikacije, što je od ključne važnosti za složene projekte poput ove turističke aplikacije.

- **Model**

Model komponenta MVVM arhitekture sadrži sve podatkovne klase koje predstavljaju osnovne podatke unutar aplikacije, kao i poslovnu logiku vezanu uz te podatke. U slučaju ove turističke aplikacije, Model je odgovoran za upravljanje podacima o znamenitostima grada Đakova, informacijama o korisnicima, podatke o GPS lokaciji, kao i interakcijama s Firebase bazom podataka. Unutar Model komponente, podaci se mogu pohranjivati, dohvaćati iz baze podataka, ili komunicirati s vanjskim API servisima. Ova komponenta je ključna jer osigurava da se svi podaci u aplikaciji pravilno obrađuju i upravljaju, bez izravne interakcije s korisničkim sučeljem.
Model također upravlja funkcijama kao što su serijalizacija i deserijalizacija podataka, validacija unosa korisnika i povezivanje s vanjskim resursima. U našoj aplikaciji, Model komponente obuhvaćaju strukture za serijalizaciju JSON podataka, podatkovne entitete za spremanje u lokalnu bazu podataka te repozitorije koji sadrže poslovnu logiku potrebnu za pripremu podataka prije nego što ih ViewModel koristi.

- **ViewModel**

ViewModel je središnja komponenta koja služi kao most između Modela i Viewa. Njegova glavna uloga je obrada podataka koje prima od Modela i njihova priprema za prikaz u korisničkom sučelju. Ovaj sloj omogućuje aplikaciji da bude reaktivna i da se dinamički prilagođava potrebama korisnika. ViewModel, u kontekstu ove aplikacije, upravlja složenim funkcijama kao što su filtriranje znamenitosti prema kategorijama, obrada podataka nakon skeniranja QR kodova te ažuriranje karte s relevantnim informacijama. Budući da ViewModel nije svjestan Viewa (korisničkog sučelja), postiže se veća fleksibilnost i mogućnost ponovne

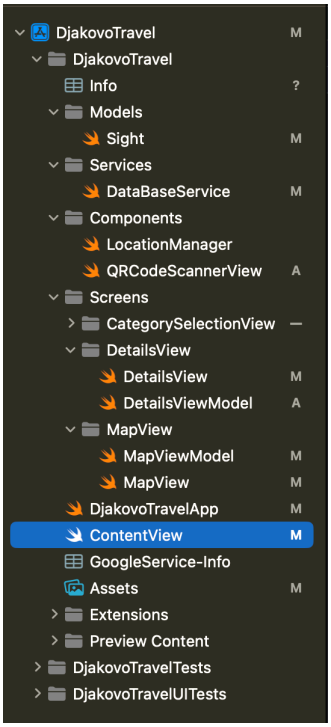
uporabe koda. Osim toga, ViewModel je dizajniran tako da može preživjeti konfiguracijske promjene, poput rotacije zaslona, što omogućuje zadržavanje podataka i nakon što korisničko sučelje prođe kroz promjene. Ova osobina je iznimno korisna u aplikacijama koje se koriste na mobilnim uređajima gdje su promjene orijentacije česte. Također, ViewModel može upravljati više izvora podataka istovremeno, kao što su podaci iz baze podataka i podaci iz vanjskih API-a. U ovom slučaju, ViewModel preuzima podatke iz Modela, prilagođava ih potrebama aplikacije i prosljeđuje ih Viewu za prikaz.

- **View**

View je komponenta MVVM arhitekture koja se bavi isključivo korisničkim sučeljem. Njen zadatak je prikazati podatke na zaslonu korisnika te reagirati na korisničke interakcije, kao što su dodiri zaslona, unosi putem tipkovnice i slične radnje. U ovoj aplikaciji, View komponenta uključuje prikaz interaktivne karte na kojoj su označene znamenitosti grada Đakova, sučelje za skeniranje QR kodova, te razne kontrole poput gumba za filtriranje znamenitosti prema kategorijama. Važno je napomenuti da View ne sadrži nikakvu poslovnu logiku; svi podaci potrebni za prikaz dolaze iz ViewModela. Ovaj pristup omogućuje jasnu odvojenost odgovornosti unutar aplikacije. View komunicira s ViewModelom putem data bindinga, koji omogućuje automatsko ažuriranje prikaza kada se podaci u ViewModelu promijene. Ovo je ključno za aplikacije koje trebaju biti reaktivne i brzo reagirati na promjene podataka. U ovoj aplikaciji, View će prikazivati sve relevantne informacije o odabranoj znamenitosti, uključujući tekstualne opise, slike, kao i mogućnosti za interakciju kao što su prikaz rute do lokacije ili slušanje audio opisa znamenitosti.

Primjenom MVVM arhitekture u razvoju ove turističke iOS aplikacije postignuta je modularnost i održivost koda, što je od ključne važnosti za aplikacije koje trebaju biti skalabilne i lako proširive. MVVM omogućuje da se poslovna logika, prezentacija podataka i korisničko sučelje razvijaju nezavisno jedno od drugoga, što olakšava testiranje, održavanje i buduće nadogradnje aplikacije. Ovakva arhitektura omogućuje timskom radu da bude učinkovitiji, jer svaki član tima može raditi na svom dijelu aplikacije bez utjecaja na druge dijelove, čime se osigurava konzistentnost i kvaliteta konačnog proizvoda.

Integracijom MVVM principa, turistička aplikacija postaje ne samo funkcionalna, već i dugoročno održiva, osiguravajući da korisnici dobiju najbolje moguće iskustvo pri korištenju aplikacije za istraživanje znamenitosti grada Đakova.



Slika 5.2. MVVM Arhitektura

6. IMPLEMENTACIJA MVVM ARHITEKTURNOG OBRAZCA NA PRIMJERU TURISTIČKE APLIKACIJE ZA GRAD ĐAKOVO

6.1 Programsko rješenje na strani korisnika

6.1.1. Početni zaslon s popisom znamenitosti

Početni zaslon ove turističke iOS aplikacije nudi korisnicima interaktivan prikaz znamenitosti na karti, omogućujući im intuitivno i efikasno istraživanje kulturnih, povijesnih i prirodnih točaka interesa unutar odabranog područja. Središnji dio zaslona zauzima karta s označenim znamenitostima, a svaki element dizajna i funkcionalnosti na zaslonu pažljivo je osmišljen kako bi podržao glavni cilj aplikacije—poboljšanje turističkog iskustva putem jednostavne navigacije i brzog pristupa ključnim informacijama.

Dizajn i funkcionalnost

Dizajn i funkcionalnost početnog zaslona turističke aplikacije osmišljeni su kako bi korisniku omogućili intuitivno istraživanje znamenitosti grada. Po pokretanju aplikacije, korisnik se odmah preusmjerava na interaktivnu kartu koja prikazuje označene znamenitosti. Svaka znamenitost ima specifičan marker, prilagođen kategoriji atrakcije, poput povijesnih, kulturnih ili prirodnih znamenitosti. Korištenje različitih boja i ikona za svaku kategoriju olakšava korisniku razlikovanje vrsta atrakcija na prvi pogled.

Navigacija kartom omogućava jednostavno povećavanje i smanjivanje prikaza, kao i pomicanje karte povlačenjem prsta, pružajući pregled šireg područja ili detaljan uvid u specifičnu lokaciju. Klikom na bilo koji marker, korisniku se prikazuju osnovni podaci o znamenitosti, poput naziva i kratkog opisa. Osim toga, korisnik može pristupiti dodatnim funkcijama kao što su prikaz detaljnijih informacija ili navigacija do odabrane lokacije. Ovaj dizajn omogućava brz i jednostavan pregled svih znamenitosti bez potrebe za dodatnim uputama. Interaktivna karta i markeri ključni su elementi koji korisniku omogućuju pregled i istraživanje grada na vizualno privlačan i funkcionalan način.

```swift
struct MapView: View {

    @ObservedObject var viewModel: MapViewModel
    @Binding var mapSelection: Sight?
    @Binding var showDetails: Bool
    var onMapLoaded: (() -> Void)?

    @StateObject var locationManager: LocationManager = .init()
    @State var mapRegion: MapCameraPosition = .region(.myRegion)
    @State var showMap: Bool = false

    var body: some View {
        VStack {
            if showMap {
                Map(position: $mapRegion, selection: $mapSelection) {
                    Marker("Matko", systemImage: "person.fill", coordinate: CLLocationCoordinate2D(latitude: locationManager.userLocation.latitude, longitude:
                        locationManager.userLocation.longitude))

                    ForEach(viewModel.sightsInDjakovo, id: \.self) { item in
                        Marker("\(item.name)", systemImage: "\(item.category)", coordinate: CLLocationCoordinate2D(latitude: item.latitude, longitude:
                            item.longitude))
                    }
                }
                .onAppear {
                    mapRegion = .region(MKCoordinateRegion(center: locationManager.userLocation, latitudinalMeters: 10000, longitudinalMeters: 10000))
                    onMapLoaded?()
                }
            } else {
                ProgressView()
                    .progressViewStyle(.circular)
                Text("Fetching your location")
            }
        }
        .onChange(of: mapSelection) { oldValue, newValue in
            if let newSelection = newValue {
                print("New map selection: \(newSelection.name)")
                showDetails = true
            } else {
                showDetails = false
            }
        }
         sheet(isPresented: $showDetails) {
```

Slika 6.1 MapView aplikacije

Na početnom zaslonu aplikacije nalaze se dva ključna interaktivna gumba koja dodatno unapređuju korisničko iskustvo. Prvi gumb, smješten u gornjem lijevom kutu, otvara izbornik s različitim kategorijama znamenitosti, poput povijesnih, kulturnih ili prirodnih atrakcija. Ovaj izbornik omogućava korisniku da jednostavno filtrira prikaz na karti prema odabranoj kategoriji, čime se pregled znamenitosti prilagođava osobnim interesima korisnika. Nakon odabira kategorije, karta prikazuje samo one atrakcije koje pripadaju odabranoj vrsti, što olakšava navigaciju i pregled sadržaja.

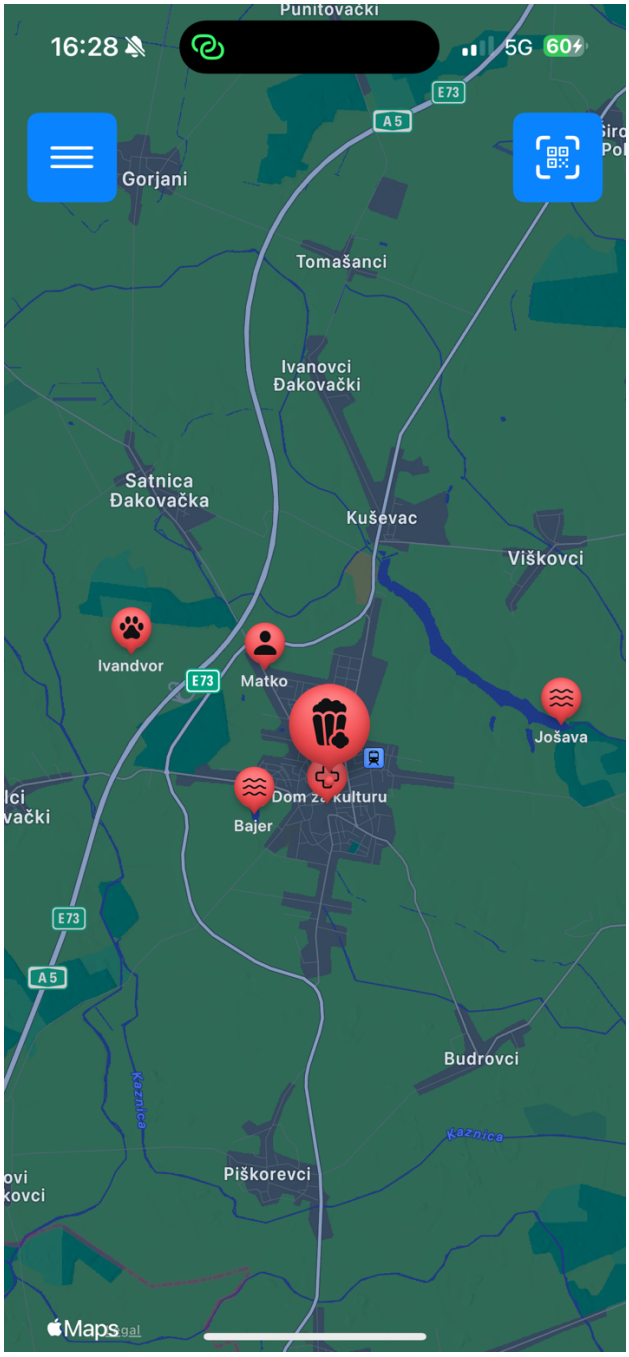
Drugi gumb, koji se nalazi u gornjem desnom kutu, omogućava korisniku skeniranje quick-response (QR) koda povezanog s određenom znamenitosti. Skeniranjem QR koda korisnik brzo dobiva pristup dodatnim informacijama o toj atrakciji, poput povijesnih podataka, fotografija ili uputa za dolazak. Ova funkcionalnost omogućuje jednostavnu interakciju s fizičkim oznakama na terenu, čime aplikacija postaje dinamičan alat za istraživanje grada. Kombinacija filtriranja i QR koda pruža korisnicima prilagođeno i informativno iskustvo korištenja aplikacije.

```swift
class MapViewModel: ObservableObject {

    @Published var sightsInDjakovo: [Sight] = []
    private var dataBaseService: DataBaseService
    private var allSights: [Sight] = []

    init(DataBaseService: DataBaseService) {
        self.dataBaseService = DataBaseService
    }

    func fetchSightsDataFromDatabase(completion: @escaping () -> Void) {
        dataBaseService.fetchSightsData { sights, error in
            if let sights = sights {
                DispatchQueue.main.async {
                    self.allSights = sights
                    self.sightsInDjakovo = sights
                    completion()
                }
            } else {
                print("Error fetching sights: \(error?.localizedDescription ?? "Unknown error")")
            }
        }
    }

    func applyCategoryFilter(type: String?) {
        if let type = type {
            sightsInDjakovo = allSights.filter { $0.type == type }
        } else {
            sightsInDjakovo = allSights
        }
    }
}
```

Slika 6.2. MapViewModel aplikacije

Navigacija i interakcija

Korisnik može povećavati i smanjivati prikaz karte, kao i pomicati se po karti kako bi istražio različite dijelove područja. Klikom na bilo koji marker na karti, korisnik dobiva detaljne informacije o odabranoj znamenitosti, uključujući naziv, opis, povijesnu pozadinu, slike, te opcije za navigaciju do te lokacije ili korištenje dodatnih funkcionalnosti poput "Tekst u govor". Ovaj početni zaslon osmišljen je kako bi bio intuitivan i jednostavan za korištenje, čime se osigurava da korisnici, bez obzira na tehničko predznanje, mogu lako istražiti i pronaći zanimljive lokacije tijekom svog putovanja. Korištenje interaktivne karte u središtu zaslona omogućuje korisnicima prirodnu i vizualno privlačnu navigaciju kroz turističku ponudu, čime se postiže glavna svrha aplikacije - poboljšanje turističkog iskustva kroz tehnologiju.



Slika 6.3. Početni zaslon aplikacije

6.1.2. Zaslon za filtriranje znamenitosti

Dizajn i funkcionalnost

Zaslon za filtriranje znamenitosti u turističkoj iOS aplikaciji predstavlja ključnu komponentu koja korisnicima omogućava personalizaciju prikaza sadržaja prema vlastitim interesima. Dizajn ovog zaslona temelji se na principima jasnoće i pristupačnosti, pri čemu su sve kategorije znamenitosti, poput povijesnih, glazbenih, sportskih i osnovnih lokacija, prikazane putem velikih, lako prepoznatljivih gumbova. Svaka kategorija vizualno je diferencirana pomoću prepoznatljivih boja, čime se osigurava brzo i intuitivno prepoznavanje opcija.

Funkcionalnost ovog zaslona omogućuje korisnicima intuitivno odabiranje jedne ili više kategorija znamenitosti koje žele istražiti, s neposrednim prilagođavanjem prikaza na glavnoj karti aplikacije. Zaslon podržava dinamičko filtriranje podataka u stvarnom vremenu, osiguravajući da korisnici odmah vide rezultate svojih izbora, bez potrebe za dodatnim koracima ili potvrđivanjem radnji. Ova značajka omogućuje besprijekorno korisničko iskustvo, pružajući brz i efikasan pristup relevantnim informacijama.

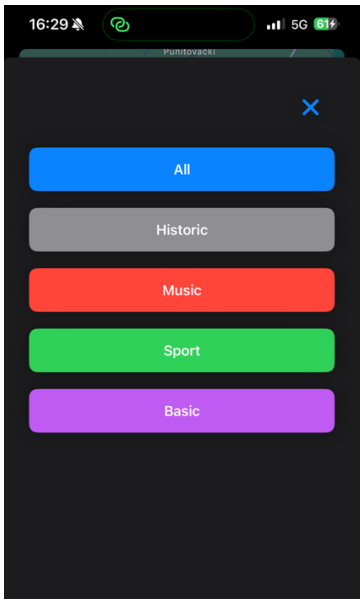
```
struct CategorySelectionView: View {

    @Binding var selectedCategory: String?
    @Binding var isPresented: Bool
    var applyCategoryFilter: () -> Void

    let categories = [("All", Color.blue),("Historic", Color.gray),("Music", Color.red),("Sport", Color.green),("Basic", Color.purple)]

    var body: some View {
        ZStack(alignment: .topLeading) {
            VStack(spacing: 20) {
                HStack {
                    Spacer()
                    Button(action: {
                        isPresented = false}) {
                        Image(systemName: "xmark")
                            .font(.system(size: 24, weight: .bold))
                            .padding()}}

                ForEach(categories, id: \.0) { category, color in
                    Button(action: {
                        selectedCategory = category == "All" ? nil : category
                        applyCategoryFilter()
                        isPresented = false}) {
                        Text(category)
                            .font(.headline)
                            .foregroundColor(.white)
                            .frame(maxWidth: .infinity)
                            .padding()
                            .background(color)
                            .cornerRadius(10)}}
                Spacer()}
            .padding()}
        .cornerRadius(15)
        .shadow(radius: 10)
        .padding()}}

struct CategorySelectionView_Previews: PreviewProvider {
    static var previews: some View {
        CategorySelectionView(selectedCategory: .constant(nil), isPresented: .constant(true)) {}}}
```

Slika 6.4. CategorySelectionView aplikacije

Navigacija i interakcija

Navigacija unutar zaslona za filtriranje znamenitosti osmišljena je s naglaskom na jednostavnost i fluidnost korisničkog iskustva. Korisnici mogu lako pristupiti ovom zaslonu putem jasno istaknutog izbornika ili gumba na početnom zaslonu, što omogućuje brzi i intuitivni ulazak u proces filtriranja. Nakon što korisnik odabere željenu kategoriju, aplikacija odmah ažurira prikaz znamenitosti na glavnoj karti, omogućujući korisnicima da istražuju samo one točke interesa koje odgovaraju njihovim odabranim kriterijima.

Interakcija s ovim zaslonom dodatno je poboljšana implementacijom intuitivnih gesti i glatkih animacija koje vode korisnika kroz proces filtriranja. Na primjer, korisnici mogu jednostavno zatvoriti zaslon za filtriranje povlačenjem prema dolje ili klikom na gumb za zatvaranje, čime se omogućuje brz povratak na glavnu kartu s ažuriranim rezultatima. Ovakav dizajnerski pristup osigurava da korisnici mogu lako navigirati i interagirati s aplikacijom, zadržavajući pritom visoku razinu kontrole nad prikazanim sadržajem.
Ovaj zaslon za filtriranje nije samo funkcionalni alat, već ključni element korisničkog iskustva koji omogućuje personaliziranu navigaciju kroz turističku ponudu. Prilagođavajući se individualnim interesima svakog korisnika, zaslon za filtriranje doprinosi boljem i učinkovitijem istraživanju znamenitosti, čineći aplikaciju prilagodljivom i korisnički orijentiranom.

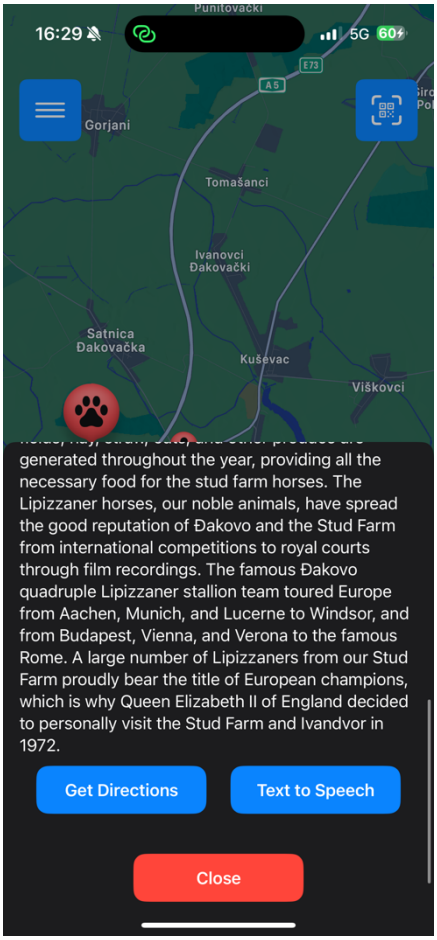


Slika 6.4. Odabir filtera zaslon

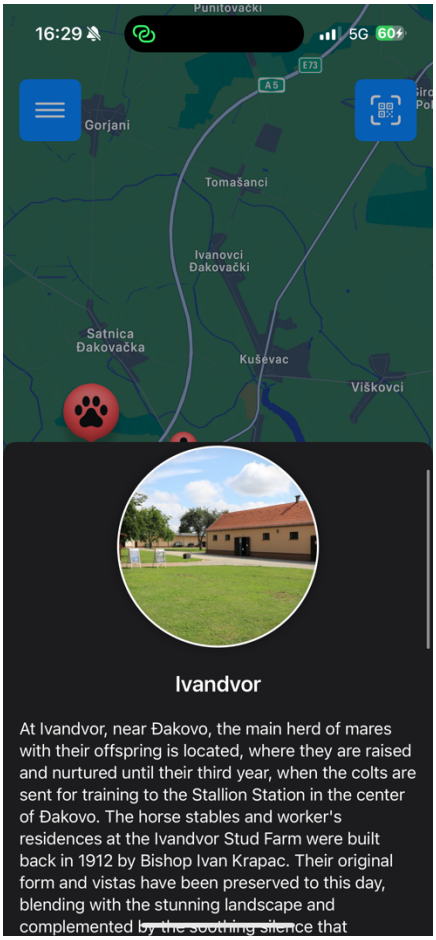
6.1.3. Zaslon o detaljima odabrane znamenitosti

Dizajn i funkcionalnost

Zaslon s detaljima odabrane znamenitosti je ključna komponenta turističke iOS aplikacije, dizajniran kako bi korisnicima omogućio sveobuhvatan pregled informacija o odabranom mjestu. Na ovom zaslonu korisnici mogu pronaći detaljne podatke o znamenitosti, uključujući povijesne, kulturne, i geografske informacije, sve u vizualno privlačnom i intuitivnom sučelju. Dizajn je pažljivo osmišljen kako bi korisnicima omogućio lako razumijevanje i navigaciju kroz pružene informacije, s jasnim naglaskom na vizualne elemente poput slika i ikona koji dodatno obogaćuju korisničko iskustvo.



Slika 6.5. DetailsView aplikacije



Slika 6.6. DetailsView aplikacije

Navigacija i interakcija

Navigacija na ovom zaslonu je fluidna i usmjerena na jednostavnost, omogućujući korisnicima da se bez poteškoća kreću između različitih sekcija sadržaja. Korisnici mogu jednostavno zatvoriti zaslon s detaljima i vratiti se na glavnu kartu ili nastaviti istraživanje drugih znamenitosti putem jasno istaknutih navigacijskih elemenata. Dodatno, implementirane su intuitivne geste za interakciju, poput klizanja prstom za zatvaranje zaslona, čime se korisnicima omogućuje prirodan i ugodan način korištenja aplikacije. Funkcionalnosti poput "Get Directions" i "Text to Speech" dodatno unapređuju interakciju, omogućujući korisnicima ne samo da pregledavaju informacije, već i da ih koriste u stvarnom vremenu, čineći njihovo turističko iskustvo bogatijim i interaktivnijim.

Ovaj zaslon ne služi samo kao informacijski centar, već i kao interaktivni alat koji korisnicima pruža sve potrebne informacije na dohvat ruke, omogućujući im da u potpunosti iskoriste prednosti turističke aplikacije. Ovaj pristup osigurava da svaki korisnik dobije prilagođeno i bogato iskustvo istraživanja znamenitosti, s mogućnošću brzog povratka na glavnu kartu ili daljnje istraživanje drugih interesnih točaka.

```
struct DetailsView: View {

    @Binding var mapSelection: Sight?
    @Binding var show: Bool
    @ObservedObject var viewModel: DetailsViewModel

    var body: some View {
        ScrollView {
            HStack {
                Image("\(mapSelection?.name ?? "")")
                    .resizable()
                    .aspectRatio(contentMode: .fill)
                    .frame(width: 200, height: 200)
                    .clipShape(Circle())
                    .overlay(Circle().stroke(Color.white, lineWidth: 2))
                    .shadow(radius: 5)}
            VStack {
                HStack {
                    VStack(alignment: .leading) {
                        Text(mapSelection?.name ?? "")
                            .font(.title2)
                            .fontWeight(.semibold)}}
                Text(mapSelection?.info ?? "")
                    .padding(.top, 8)
                HStack(spacing: 24) {
                    Button {
                    } label: {
                        Text("Get Directions")
                            .font(.headline)
                            .foregroundColor(.white)
                            .frame(width: 170, height: 48)
                            .background(.blue)
                            .cornerRadius(12)}
                    Button {
                        viewModel.speak(text: mapSelection?.info ?? "no text")
                    } label: {
                        Text("Text to Speech")
                            .font(.headline)
                            .foregroundColor(.white)
                            .frame(width: 170, height: 48)
```

Slika 6.7. DetailsView-code aplikacije

6.1.4. Zaslon s mapom i navigacijom do znamenitosti

Zaslon s mapom i navigacijom unutar ove turističke iOS aplikacije predstavlja ključni element korisničkog iskustva, osmišljen kako bi turistima pružio sve potrebne alate za bezbrižno i učinkovito kretanje prema odabranim znamenitostima. Ovaj zaslon ne samo da prikazuje trenutnu lokaciju korisnika u odnosu na znamenitosti koje žele posjetiti, već također nudi niz funkcionalnosti koje omogućuju precizno planiranje puta i prilagodbu navigacije prema osobnim preferencijama.

Dizajn i funkcionalnost

Dizajn zaslona je minimalistički, ali izuzetno funkcionalan, s naglaskom na preglednost i jednostavnost korištenja. Na gornjem dijelu zaslona nalazi se interaktivna karta koja omogućuje korisnicima da jasno vide svoju trenutnu poziciju, kao i odabranu znamenitost. Karta je opremljena intuitivnim ikonama i oznakama koje korisnicima pružaju brze vizualne smjernice, čineći cijeli proces navigacije jednostavnim i razumljivim.

Korisnici mogu birati između različitih ruta prema svojim potrebama i preferencijama. Aplikacija nudi mogućnost izbora najbrže rute, rute s najmanje skretanja ili rute prilagođene određenom načinu prijevoza, kao što su vožnja automobilom, pješačenje ili korištenje javnog prijevoza. Sve ove opcije su jasno istaknute na zaslonu, s preciznim vremenskim procjenama i udaljenostima kako bi korisnici mogli donijeti informirane odluke.

Dodatne funkcionalnosti zaslona uključuju mogućnost prilagođavanja postavki rute, kao što su izbjegavanje autocesta, cestarina ili područja s gustom prometom. Korisnici mogu također dodati međustanice na svojoj ruti, što im omogućuje da isplaniraju višestruke posjete znamenitostima tijekom jednog izleta.

Navigacija i interakcija

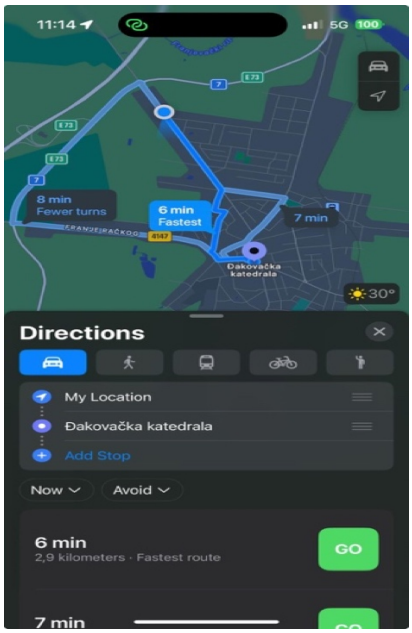
Navigacija na ovom zaslonu je osmišljena s naglaskom na fluidnost i intuitivnost. Korisnici mogu jednostavno prstom pomicati kartu, povećavati ili smanjivati prikaz kako bi dobili bolji

uvid u svoju okolinu ili odabranu rutu. Interaktivne smjernice se ažuriraju u stvarnom vremenu, prilagođavajući se trenutnoj lokaciji korisnika i uvjetima na putu, kao što su prometne gužve ili radovi na cesti.

Kada korisnik unese svoje odredište, zaslon automatski prikazuje najoptimalniju rutu s detaljnim smjernicama za svaki korak puta. Smjernice su jasno prikazane na zaslonu, a uz to, aplikacija omogućava korisnicima da se prebace između različitih prikaza karte, kao što su satelitski prikaz, 3D prikaz ili standardni cestovni prikaz, čime se dodatno poboljšava vizualno iskustvo navigacije.

Interakcija je također olakšana implementacijom glasovnih smjernica koje korisnicima omogućuju da zadrže pogled na cesti dok aplikacija pruža upute putem zvuka. Ova značajka je posebno korisna tijekom vožnje, gdje je pažnja na putu ključna za sigurnost.

Sve ove funkcionalnosti kombinirane su s modernim dizajnom kako bi se osiguralo da korisnici imaju potpuni kontrolu nad svojim turističkim iskustvom. Zaslon za mapu i navigaciju nije samo alat za kretanje od točke A do točke B, već predstavlja personalizirani vodič koji korisnicima omogućuje da istražuju i otkrivaju znamenitosti na način koji im najviše odgovara. Takav pristup ne samo da poboljšava funkcionalnost aplikacije, već i značajno doprinosi cjelokupnom zadovoljstvu korisnika.



Slika 6.8. NavigationView

6.1.5. Zaslon za skeniranje QR koda

Zaslon za skeniranje QR koda predstavlja ključnu funkcionalnost unutar turističke iOS aplikacije, omogućujući korisnicima brzu i jednostavnu interakciju s fizičkim točkama interesa putem digitalne platforme. Ovaj zaslon pruža korisnicima mogućnost da putem integrirane kamere na svom uređaju skeniraju QR kodove postavljene na različitim znamenitostima, što rezultira trenutnim prikazom relevantnih informacija ili navigacijskih uputa na njihovom uređaju.

Dizajn i funkcionalnost

Dizajn zaslona za skeniranje QR koda je minimalistički, ali usmjeren na maksimalnu funkcionalnost. Središnji dio zaslona rezerviran je za prikaz kamere, koja zauzima većinu prostora, omogućujući korisnicima jasan i nesmetan pregled okoline prilikom skeniranja QR koda. Oko prikaza kamere nalaze se suptilne grafike ili okvir koji vizualno vodi korisnika gdje da postavi QR kod za optimalno skeniranje.

U gornjem dijelu zaslona može biti prikazana ikona za povratak na prethodni zaslon, dok su u donjem dijelu smještene informacije ili upute vezane uz skeniranje, kao i tipka za pristup dodatnim opcijama, poput omogućavanja ili onemogućavanja bljeskalice. Bljeskalica se automatski može aktivirati u uvjetima slabog osvjetljenja, čime se osigurava precizno skeniranje bez obzira na vanjske uvjete.

Jednom kada QR kod bude prepoznat, aplikacija odmah procesira sadržaj koda i korisnika vodi na relevantnu stranicu unutar aplikacije, bilo da se radi o detaljnim informacijama o znamenitosti, virtualnom vodiču, ili prikazu rute na karti prema odabranom odredištu.

Navigacija i interakcija

Navigacija na zaslonu za skeniranje QR koda je intuitivna i jednostavna, s fokusom na brzoj i bezproblematničkoj interakciji. Korisnici mogu pristupiti ovom zaslonu iz glavnog izbornika aplikacije ili putem specifične ikone za skeniranje koja se nalazi na ključnim dijelovima sučelja, poput kartografskog prikaza ili početnog zaslona aplikacije.

Jednom kada se korisnik nađe na zaslonu za skeniranje, interakcija je izuzetno jednostavna – korisnik samo treba usmjeriti kameru prema QR kodu. Aplikacija automatski detektira kod i obrađuje podatke bez potrebe za dodatnim koracima s korisnikove strane. Također, korisnici mogu lako izaći iz zaslona za skeniranje jednostavnim klikom na ikonu za povratak ili povlačenjem prema dolje, omogućujući brz povratak na prethodno korišteni dio aplikacije.

Slika 6.9. Jošava poster- QR code

Ovaj zaslon je posebno dizajniran kako bi olakšao korisnicima prijelaz između fizičkog i digitalnog svijeta, nudeći im pristup dodatnim informacijama i uslugama na temelju njihovih stvarnih lokacija i interesa. Zahvaljujući integraciji sa sustavom za prepoznavanje QR kodova, aplikacija može pružiti personalizirano iskustvo, vodeći korisnike kroz njihove turističke avanture na način koji je prilagođen njihovim specifičnim potrebama i željama.

Kombinacija jednostavnog dizajna, intuitivne navigacije i funkcionalnosti koja omogućuje trenutan pristup relevantnim informacijama čini zaslon za skeniranje QR koda jednim od najvažnijih alata u ovoj turističkoj aplikaciji, osiguravajući da korisnici mogu lako i efikasno koristiti sve prednosti koje aplikacija nudi.

6.1.6. Text-to-spech funkcionalnost aplikacije

Funkcionalnost pretvaranja teksta u govor (Text-to-Speech, TTS) unutar ove turističke iOS aplikacije pruža korisnicima mogućnost praktičnog slušanja informacija o znamenitostima, umjesto njihovog čitanja na zaslonu. Ova funkcionalnost postaje izuzetno korisna u situacijama gdje je potrebna hands-free interakcija, ili kada je vizualno praćenje sadržaja nepraktično ili ometajuće, kao što je to slučaj tijekom vožnje, hodanja ili u situacijama smanjene vidljivosti. TTS tehnologija značajno doprinosi pristupačnosti aplikacije, omogućujući korisnicima svih profila da se lako informiraju o kulturnim i povijesnim znamenitostima, bez potrebe za kontinuiranim pogledom na ekran. Ova funkcionalnost ne samo da poboljšava korisničko iskustvo, već i omogućuje aplikaciji da bolje zadovolji potrebe različitih korisničkih skupina, uključujući osobe s oštećenjem vida, vozače i druge korisnike koji preferiraju audio sadržaje.

Dizajn i funkcionalnost

Funkcionalnost Text-to-Speech integrirana je na svim relevantnim zaslonima aplikacije koji prikazuju tekstualne informacije o znamenitostima. Jednostavnim dodirom na ikonu za zvučnik, korisnici mogu pokrenuti funkciju čitanja teksta. Dizajn ove ikone je suptilan, ali prepoznatljiv, smješten u donjem dijelu zaslona ili unutar dodatnih opcija, kako bi bio lako dostupan, ali ne i nametljiv.

Nakon što korisnik pokrene Text-to-Speech, aplikacija automatski pretvara pisani tekst u govor, koristeći zadani glas i jezik koji je konfiguriran na korisnikovom uređaju. U aplikaciji su također dostupne opcije za pauziranje, nastavak ili zaustavljanje govora, što omogućuje korisniku potpunu kontrolu nad ovom funkcionalnošću.

Dodatno, aplikacija nudi prilagodbe brzine govora i tona, što korisnicima omogućuje personalizaciju iskustva slušanja u skladu s njihovim preferencijama.

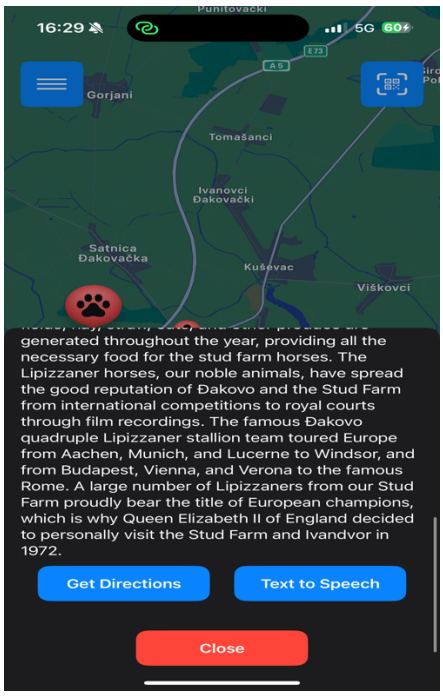
Navigacija i interakcija

Navigacija unutar Text-to-Speech funkcionalnosti je dizajnirana tako da bude intuitivna i jednostavna. Korisnici mogu lako pristupiti opciji za slušanje teksta na bilo kojem zaslonu s

detaljnim informacijama o znamenitostima. Nakon aktiviranja, aplikacija automatski prelazi u način rada u kojem prioritet ima audio sadržaj, dok vizualne informacije ostaju dostupne na zaslonu.

Interakcija s Text-to-Speech funkcijom dodatno je poboljšana putem jednostavnih kontrola koje omogućuju brz pristup osnovnim funkcijama, poput pauze ili ponovnog pokretanja. Također, korisnici mogu jednostavno isključiti govor i vratiti se na standardni način prikaza informacija jednim dodirom na ikonu.

Ova funkcionalnost nije samo alat za pristup informacijama, već ključni dio pristupačnosti aplikacije, osiguravajući da svi korisnici, bez obzira na njihove specifične potrebe ili situaciju u kojoj se nalaze, mogu potpuno iskoristiti prednosti koje aplikacija nudi. Text-to-Speech omogućuje dublje angažiranje korisnika, čineći informacije dostupnijima i prilagodljivijima, te doprinosi cjelokupnom iskustvu korištenja aplikacije.



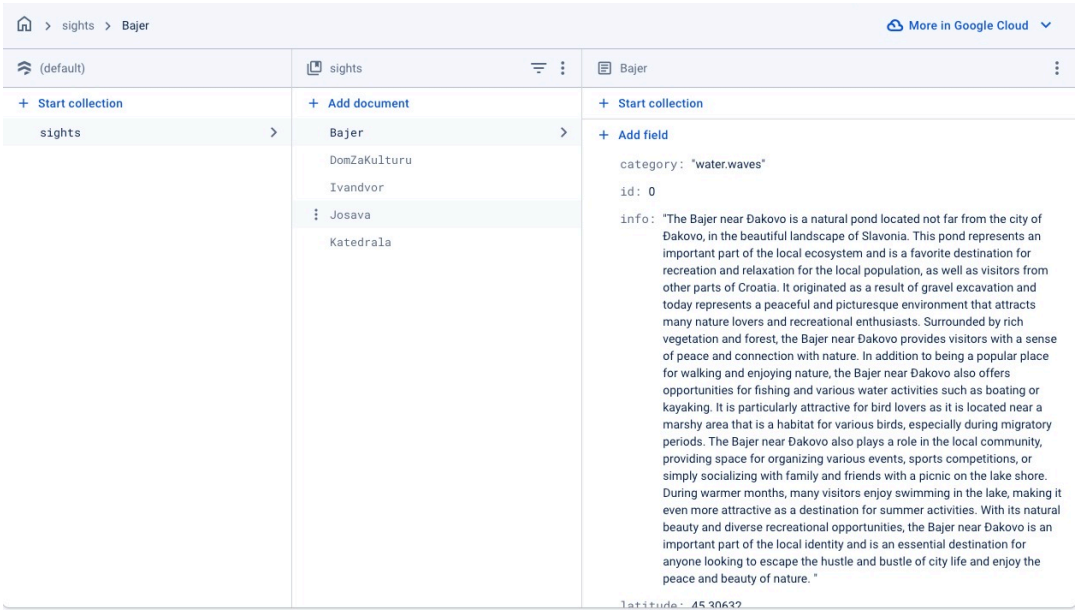
Slika 6.10. Text-to-spech button

6.2 Programsko rješenje na strani poslužitelja

6.2.1 Postavljanje podataka o znamenitostima na Firebase

Ovo poglavlje detaljno opisuje proces postavljanja i ažuriranja podataka o znamenitostima na Firebase, što uključuje pohranu ključnih informacija poput lokacije, kategorije, opisa, te dodatnih atributa važnih za korisničko iskustvo. Korištenjem Firebase baze podataka, aplikacija omogućuje učinkovitu pohranu i organizaciju podataka koji su temelj za prikaz i interakciju sa sadržajem unutar aplikacije.Prilikom postavljanja podataka, programeri definiraju strukturu dokumenata u Firebaseu, gdje svaki dokument predstavlja pojedinačnu znamenitost. Unutar svakog dokumenta, polja kao što su *latitude* i *longitude* definiraju geografsku lokaciju znamenitosti, dok polje *category* označava vrstu znamenitosti (npr. povijesna, kulturna, prirodna). Opis (*description*) pruža korisnicima detaljan tekstualni pregled znamenitosti, omogućujući im da se informiraju o njenoj povijesti, značaju i drugim relevantnim činjenicama.

Ažuriranje podataka na Firebaseu također je važno kako bi se osiguralo da korisnici uvijek imaju pristup najnovijim informacijama. Svaka promjena u podacima o znamenitostima automatski se reflektira na korisničkim uređajima putem Firebaseove funkcionalnosti ažuriranja u stvarnom vremenu. Ovaj pristup osigurava dosljednost i točnost podataka koji se prikazuju unutar aplikacije, pružajući korisnicima pouzdano i aktualno iskustvo prilikom istraživanja znamenitosti.



Slika 6.11.Firebase baza podataka

6.2.2. Programska implementacija dohvaćanja podataka iz Firebase baze podataka

U ovom poglavlju detaljno ćemo opisati način na koji se implementira dohvaćanje podataka iz Firebase Firestore baze podataka unutar turističke iOS aplikacije. Ova funkcionalnost je ključna za prikaz ažuriranih informacija o znamenitostima unutar aplikacije, omogućavajući korisnicima pristup najnovijim podacima bez potrebe za ručnim ažuriranjem. Korištenjem Firebase Firestore baze podataka, aplikacija omogućava pohranu i organizaciju podataka na način koji je skalabilan i lako dostupan. U nastavku je prikazan kod koji implementira dohvaćanje podataka iz Firebasea pomoću Firestore klase unutar aplikacije.

```
class DataBaseService {

    let db = Firestore.firestore()

    func fetchSightsData(completion: @escaping ([Sight]?, Error?) -> Void) {
        db.collection("sights").getDocuments { (querySnapshot, error) in
            if let error = error {
                completion(nil, error)
            } else {
                var sights = [Sight]()
                for document in querySnapshot!.documents {
                    if let latitude = document.data()["latitude"] as? Double,
                       let longitude = document.data()["longitude"] as? Double,
                       let name = document.data()["name"] as? String,
                       let id = document.data()["id"] as? Int,
                       let category = document.data()["category"] as? String,
                       let info = document.data()["info"] as? String,
                       let type = document.data()["type"] as? String{
                        let sight = Sight(id: id, name: name, latitude: latitude, longitude: longitude, category:category, info: info, type: type)
                        sights.append(sight)
                    }
                }
                completion(sights, nil)
            }
        }
    }
}
```

Slika 6.12. DataBaseService kod

Funkcionalnost:

Ova metoda koristi `Firestore` za dohvaćanje svih dokumenata iz kolekcije *"sights"* pohranjene u *Firebase* bazi podataka. Metoda `fetchSightsData` koristi `getDocuments` metodu za dobivanje svih dokumenata u kolekciji i zatim iterira kroz svaki dokument kako bi izdvojila relevantne podatke kao što su `latitude`, `longitude`, `name`, `id`, `category`, `info`, *i* `type`. Nakon toga, podaci se mapiraju u `Sight` objekte koji se dodaju u niz znamenitosti *(*`sights`).

Navigacija i interakcija:

Implementacija je dizajnirana tako da podržava asinkroni obrazac pomoću *completion handlera.* Ovo omogućava da se dohvaćanje podataka iz baze podataka obavlja u pozadini, dok korisničko sučelje ostaje responzivno. U slučaju uspješnog dohvaćanja, niz sights se vraća natrag kroz completion handler, omogućavajući ViewModelu ili kontroleru koji koristi ovu metodu da ažurira sučelje aplikacije s novim podacima. U slučaju greške, completion handler vraća odgovarajuću grešku, omogućavajući aplikaciji da upravlja iznimkama na odgovarajući način.

Prednosti:

Korištenje Firebase Firestore-a u ovakvoj implementaciji nudi nekoliko ključnih prednosti:

1. **Skalabilnost** - Firestore je NoSQL baza podataka koja se skalira s rastom broja korisnika i podataka.

2. **Ažurnost** - Firebase podržava real-time ažuriranja, što znači da korisnici mogu odmah vidjeti promjene podataka bez potrebe za ručnim osvježavanjem aplikacije.

3. **Asinkronost** - Metoda ***fetchSightsData*** koristi asinkrono dohvaćanje podataka, čime osigurava da aplikacija ostane responzivna tijekom interakcije s bazom podataka.

Ova implementacija osigurava da korisnici aplikacije uvijek imaju pristup najnovijim i najtočnijim informacijama o znamenitostima, što poboljšava cjelokupno iskustvo korištenja aplikacije.

6.2.3. Implementacija upravljanja korisiničkom lokacijom

Ovo poglavlje detaljno opisuje implementaciju upravljanja korisničkom lokacijom unutar turističke iOS aplikacije, s posebnim naglaskom na korištenje Apple-ovih lokacijskih usluga. Kroz integraciju CLLocationManager klase, aplikacija precizno prati trenutnu geografsku poziciju korisnika, pružajući mu personalizirane informacije i smjernice koje su relevantne za njegovu trenutnu lokaciju.Primarni cilj ove funkcionalnosti je omogućiti korisnicima da lako identificiraju i lociraju znamenitosti u svojoj neposrednoj blizini, poboljšavajući time njihovo

cjelokupno iskustvo korištenja aplikacije. Aplikacija koristi različite dozvole za pristup lokacijskim uslugama, osiguravajući korisnicima optimalan balans između privatnosti i funkcionalnosti.

Lokacijski podaci prikupljeni putem CLLocationManager klase se također koriste za dinamičko ažuriranje prikaza na karti, omogućujući korisnicima da u stvarnom vremenu vide svoje kretanje i prilagode svoju rutu prema željenoj destinaciji. Osim toga, aplikacija obrađuje i situacije u kojima korisnici mogu odbiti dozvolu za korištenje lokacijskih usluga, pružajući im alternativne načine za ručno pretraživanje i istraživanje znamenitosti.

Ova implementacija značajno doprinosi personalizaciji korisničkog iskustva unutar aplikacije, čineći je ne samo alatkom za informiranje, već i dinamičnim vodičem koji se prilagođava specifičnim potrebama i preferencijama svakog korisnika. S obzirom na ključnu ulogu lokacije u kontekstu turističkih aplikacija, ovo poglavlje osigurava detaljan pregled metodologije i tehnika koje su korištene kako bi se ostvarila ova funkcionalnost na najučinkovitiji način.

```
class LocationManager: NSObject, ObservableObject, MKMapViewDelegate, CLLocationManagerDelegate {

    @Published var manager: CLLocationManager = .init()
    @Published var userLocation: CLLocationCoordinate2D = .myLocation
    @Published var isLocationAuthorized:Bool = false
    override init() {
        super.init()
        manager.delegate = self
        manager.requestWhenInUseAuthorization()
        manager.desiredAccuracy = kCLLocationAccuracyNearestTenMeters}

    func locationManagerDidChangeAuthorization(_ manager: CLLocationManager) {
        checkAuthorization()}
    func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
        guard let currentLocation = locations.last else{return}

        print(currentLocation)

        userLocation = .init(latitude: currentLocation.coordinate.latitude, longitude: currentLocation.coordinate.longitude)
        isLocationAuthorized = true}

    func locationManager(_ manager: CLLocationManager, didFailWithError error: Error) {

    }
    func checkAuthorization(){
        switch manager.authorizationStatus{
        case .notDetermined:
            print("Location is not determined")
            manager.requestWhenInUseAuthorization()
        case .denied:
            print("Location is denied")
        case .authorizedAlways,.authorizedWhenInUse:
            print("Location permision done")
            manager.requestLocation()
        default:
            break;}}}
```

Slika 6.13. LocationManagerClass

MVVM arhitektura donosi brojne prednosti u razvoju turističke aplikacije za grad Đakovo, posebno zahvaljujući jasnom razdvajanju poslovne logike (*ViewModel*) od korisničkog sučelja (*View*). Ova podjela olakšava održavanje koda, testiranje i skaliranje aplikacije. Za razliku od MVC-a, gdje *Controller* može postati preopterećen, MVVM osigurava čišću separaciju odgovornosti, čime se smanjuje složenost i poboljšava korisničko iskustvo. Iako je VIPER arhitektura prikladna za velike projekte, zbog veće složenosti uvodi više komponenti, dok je MVVM jednostavniji i primjereniji za aplikacije srednje veličine.

MVVM podržava napredne funkcionalnosti kao što su dinamičko filtriranje znamenitosti i integracija QR kodova, gdje ViewModel obrađuje podatke, a View prikazuje rezultate. Ovakva struktura omogućava brže nadogradnje i jednostavnije promjene bez destabilizacije aplikacije. ViewModel se lako testira neovisno o View-u, što poboljšava kvalitetu koda. S boljom organizacijom i modularnošću, MVVM arhitektura čini aplikaciju spremnom za buduće tehnološke inovacije i prilagodbu korisničkim potrebama. Zbog ovih karakteristika, MVVM je optimalan izbor za ovu aplikaciju, omogućujući korisnicima bogato i intuitivno iskustvo.

7. ZAKLJUČAK

U izradi ove turističke IOS aplikacije za grad Đakovo, primjena MVVM (Model-View-ViewModel) arhitekture pokazala se kao ključni faktor u osiguravanju modularnosti, skalabilnosti i održivosti koda. Ovaj arhitektonski pristup omogućio je jasno odvajanje poslovne logike od korisničkog sučelja, što ne samo da je pojednostavilo proces razvoja, već je također osiguralo lakšu prilagodbu i proširenje aplikacije s novim funkcionalnostima u budućnosti. Aplikacija već sada pruža bogato korisničko iskustvo kroz interaktivne mape, QR kod skeniranje, te Text-to-Speech funkcionalnost, što je čini sveobuhvatnim alatom za istraživanje kulturnih, povijesnih i prirodnih znamenitosti grada Đakova.

Korištenje Firebase baze podataka omogućilo je sigurno i učinkovito upravljanje podacima, s real-time sinkronizacijom koja korisnicima osigurava pristup najnovijim informacijama. Također, integracija naprednih tehnologija dodatno je povećala pristupačnost i funkcionalnost aplikacije, omogućujući korisnicima personalizirano i hands-free iskustvo prilikom istraživanja znamenitosti. Unatoč trenutnom opsegu funkcionalnosti, ova aplikacija ostaje otvorena za daljnji razvoj i unaprjeđenje.

MVVM arhitektura pruža fleksibilnu osnovu koja omogućava jednostavno dodavanje novih značajki, kao što su proširena stvarnost (AR) za još interaktivnije doživljaje, integracija s umjetnom inteligencijom (AI) za personalizirane preporuke, te proširenje postojećih mogućnosti filtriranja i navigacije.

Buduće verzije aplikacije mogle bi uključivati dodatne opcije za društvenu interakciju, omogućavajući korisnicima da dijele svoja iskustva i recenzije unutar aplikacije, čime bi se dodatno povećala angažiranost korisnika.

Nadalje, aplikacija može biti proširena i u pogledu pokrivenosti, dodavanjem novih destinacija izvan grada Đakova, čime bi se otvorile nove mogućnosti za istraživanje i privukli korisnici iz šire regije ili čak međunarodni turisti. Također, moguća je integracija s vanjskim API-jevima za lokalne događaje, vremensku prognozu ili javni prijevoz, što bi dodatno obogatilo korisničko iskustvo i učinilo aplikaciju još korisnijom.

Sve ove mogućnosti pokazuju da aplikacija, zahvaljujući svojoj temeljitoj arhitektonskoj osnovi i fleksibilnosti, ima ogroman potencijal za daljnji rast i razvoj. Njena sposobnost da se prilagodi i evoluira u skladu s potrebama korisnika i tehnološkim napretkom osigurava dugoročnu relevantnost i uspjeh, čineći je ne samo vodičem za sadašnjost, već i platformom koja će i dalje rasti i napredovati u budućnosti.

LITERATURA

[1] D. Yuristiawan, Reactive way of Mobile App Development using iOS Combine Framework [online], Halodoc, 2022, dostupno na: : https://blogs.halodoc.io/reactiveway-of-mobile-app-development-using-ios-combineframework/#:~:text=Combine%20is%20a%20powerful%20framework,easy%20to%2 0read%20and%20maintain [09.Rujna 2024].

[2] F. Laso-Marsetti, Model-View-Controller (MVC) in iOS – A Modern Approach [online], dostupno na: https://www.kodeco.com/1000705-model-view-controller-mvc-inios-a-modern-approach [09.Rujna 2024].

[3] The Ultimate Guide to Mobile Application Architecture [online], Scand, 2022, dostupno na: https://www.netsolutions.com/insights/mobile-app-architecture-guide/ [09.Rujna 2024].

[4] J. Varma, u SwiftUI for Absolute Beginners: Program Controls and Views for iPhone, iPad, and Mac Apps, Apress, 2019 [09.Rujna 2024].

[5] TIOBE Index for June 2023 [online], TIOBE, dostuno na: https://www.tiobe.com/tiobeindex/ [09.Rujna 2024].

[6] F. Nayebi, J.-M. Desharnais i A. Abran, An Expert-based Framework for Evaluating iOS Application Usability, u Joint Conference of the 23rd International Workshop on Software Measurement, Ankara, 2013 [09.Rujna 2024].

[7] W. Kelton, Apple App Store [online], Investopedia, 2023, dostupno na: https://www.investopedia.com/terms/a/apple-app-store.asp [09.Rujna 2024].

[8] K. W. Tracy, Mobile Application Development Experiences on Apple's iOS and Android OS, IEEE Potentials [09.Rujna 2024].

[9]. C. Tozzi, A Beginner's Guide to iOS App Testing [online], SweetCode, dostupno na: : https://sweetcode.io/a-beginners-guide-to-ios-app-testing/ [09.Rujna 2024].

[10] Firebase documentation, Cloud Storage for Firebase, dostupno na: https://firebase.google.com/docs/storage [09.Rujna 2024].

[11] R. Ranjan, The Mobile App Architecture Guide for 2023 [online], Net Solutions, 2022, Dostupno na: https://www.netsolutions.com/insights/mobile-app-architecture-guide/. [09.Rujna 2024].

[12] D. Indrawan, D. S. Kusumo i S. Y. Puspitasari, Analysis of the implementation of MVVM architecture pattern on performance of iOS mobilebased applications, JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika) [09.Rujna 2024].

[13] https://www.logohistories.com/p/apple-computer-logo-design-history-rob-janoff

[14] https://logos-world.net/tripadvisor-logo/

[15] https://www.fontinlogo.com/logo/google-maps

[16] https://www.digital.ink/blog/airbnb-logo/

SAŽETAK

Ovaj diplomski rad fokusira se na razvoj i implementaciju napredne mobilne iOS aplikacije, koja je koncipirana kao interaktivni turistički vodič za grad Đakovo. Aplikacija je osmišljena s ciljem da korisnicima pruži cjelovito i intuitivno iskustvo istraživanja gradskih znamenitosti, uključujući kulturne, povijesne i prirodne atrakcije.

Glavni korisnički interface temelji se na interaktivnoj karti koja omogućuje jednostavno pretraživanje i filtriranje sadržaja prema individualnim interesima korisnika, čime se značajno olakšava pronalaženje relevantnih točaka interesa. Aplikacija koristi Model-View-ViewModel (MVVM) arhitekturu, koja omogućuje jasnu separaciju poslovne logike od korisničkog sučelja, čime se postiže bolje održavanje koda, veća skalabilnost, te lakša nadogradnja funkcionalnosti u budućnosti. Za razvoj korisničkog sučelja upotrijebljen je SwiftUI, moderna i moćna tehnologija za razvoj sučelja na iOS platformi, koja dodatno pridonosi responzivnosti i fluidnosti korisničkog iskustva.

Podaci o znamenitostima pohranjuju se u Firebase bazu podataka, što omogućuje kontinuiranu ažuriranost i dinamičnost sadržaja unutar aplikacije. Također, aplikacija koristi lokacijske usluge kako bi pratila trenutnu poziciju korisnika, omogućujući precizno navođenje do odabrane destinacije putem različitih načina prijevoza, poput pješačenja, vožnje automobilom ili korištenja javnog prijevoza. Jedna od ključnih funkcionalnosti aplikacije je integracija Text-to-Speech (TTS) tehnologije, koja korisnicima omogućuje da slušaju informacije o znamenitostima umjesto da ih čitaju, čime se poboljšava pristupačnost aplikacije, posebice u situacijama gdje je vizualno praćenje zaslona otežano.

U budućnosti, aplikacija nudi značajan prostor za daljnji razvoj i nadogradnju. Potencijalne nadogradnje uključuju integraciju proširene stvarnosti (AR), koja bi omogućila korisnicima interakciju sa stvarnim okolišem kroz digitalne dodatke, kao i uvođenje umjetne inteligencije (AI) za pružanje još personaliziranijih preporuka na temelju korisničkih preferencija i ponašanja.

Ova prilagodljivost i mogućnost daljnjeg razvoja čine aplikaciju dugoročno održivim rješenjem koje može nastaviti rasti i evoluirati u skladu s tehnološkim napretkom i promjenama korisničkih potreba.

Ključne riječi: iOS aplikacija, MVVM arhitektura, turizam, Đakovo, SwiftUI, Firebase, Text-to-Speech, lokacijske usluge.

ABSTRACT

This thesis focuses on the development and implementation of an advanced mobile iOS application, designed as an interactive tourist guide for the city of Đakovo. The application is intended to provide users with a comprehensive and intuitive experience in exploring the city's landmarks, including cultural, historical, and natural attractions. The main user interface is based on an interactive map that allows easy searching and filtering of content according to the individual interests of users, significantly facilitating the discovery of relevant points of interest. The application employs the Model-View-ViewModel (MVVM) architecture, which ensures a clear separation of business logic from the user interface, resulting in better code maintenance, greater scalability, and easier functionality upgrades in the future. SwiftUI, a modern and powerful technology for interface development on the iOS platform, was used for developing the user interface, further contributing to the responsiveness and fluidity of the user experience. Data on landmarks are stored in a Firebase database, allowing continuous updating and dynamic content within the application. Additionally, the application uses location services to track the user's current position, enabling precise navigation to the selected destination via different modes of transportation, such as walking, driving, or using public transport. One of the key features of the application is the integration of Text-to-Speech (TTS) technology, which allows users to listen to information about landmarks instead of reading it, thereby enhancing the application's accessibility, especially in situations where visual attention to the screen is challenging. In the future, the application offers significant potential for further development and enhancement. Potential upgrades include the integration of augmented reality (AR), which would allow users to interact with the real environment through digital overlays, as well as the introduction of artificial intelligence (AI) to provide even more personalized recommendations based on user preferences and behavior. This adaptability and potential for further development make the application a sustainable long-term solution that can continue to grow and evolve in line with technological advancements and changing user needs.

Keywords: iOS application, MVVM architecture, tourism, Đakovo, SwiftUI, Firebase, Text-to-Speech, location services.

ŽIVOTOPIS

Matko Mihalj rođen je 21. veljače 2000. godine u Đakovu, Hrvatska, gdje i trenutno živi. Nakon uspješnog završetka Osnovne škole Josipa Antuna Ćolnića u Đakovu, upisuje matematički smjer Gimnazije Antuna Gustava Matoša, koju završava 2018. godine. Iste godine započinje svoje akademsko obrazovanje na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, upisom sveučilišnog preddiplomskog studija računarstva. Po završetku preddiplomskog studija, nastavlja svoje obrazovanje na istom fakultetu, upisom diplomskog studija programskog inženjerstva. Tijekom svog akademskog puta, razvio je napredno znanje i vještine u različitim programskim jezicima i tehnologijama, uključujući iOS razvoj, Swift, Python, C, .NET, React i PostgreSQL. Osim toga, posjeduje kompetencije u radu s Javom, C++, C#, te s alatima iz Microsoft Office paketa. Njegovo tehničko znanje omogućilo mu je aktivno sudjelovanje u različitim projektima, kako tijekom studija tako i u samostalnim inicijativama. ističe se svojom sposobnošću rada pod pritiskom, visokim stupnjem upornosti i odgovornosti te izvrsnim znanjem engleskog jezika, koje je ključan alat u njegovoj svakodnevnoj profesionalnoj komunikaciji i u radu s međunarodnim tehnologijama i platformama. Osim tehničkih vještina, Matko je strastveni ljubitelj sporta, osobito nogometa, što mu je pomoglo razviti timski duh, disciplinu i natjecateljski pristup, vrijednosti koje primjenjuje i u profesionalnom životu. Sport mu pruža ravnotežu između mentalnog i fizičkog zdravlja, doprinoseći njegovom fokusiranom i učinkovito organiziranom pristupu u radu.