Mobilna aplikacija za traženje suputnika

Popić, Ivan

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

Stručni studij Računarstva

MOBILNA APLIKACIJA ZA TRAŽENJE SUPUTNIKA

Završni rad

Ivan Popić

Osijek, 2023 godina.

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

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Osijek, 20.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Ime i prezime Pristupnika:	Ivan Popić
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	AI 4631, 27.07.2017.
OIB Pristupnika:	05020048192
Mentor:	Marina Peko, dipl. ing.
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Robert Šojo, mag. ing. comp.
Član Povjerenstva 1:	Marina Peko, dipl. ing.
Član Povjerenstva 2:	mr. sc. Željko Štanfel
Naslov završnog rada:	Mobilna aplikacija za traženje suputnika
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Izraditi mobilnu aplikaciju koja omogućava traženje suputnika. Aplikacija ima 2 role: administrator i putnik. Administrator ima pravo blokirati i obrisati neprimjerene zahtjeve, dok putnik moze postaviti post u kojem traži suputnika za predloženo odredište. Također, na aplikaciji se mogu prijaviti i agencije koje nude organizaciju putovanja za manje grupe ljudi. Aplikacija koristi i lokaciju korisnika, te mu izdvaja postove koji su mu "u blizini" kako bih mogao odluciti i prema tom kriterju s kim ce putovati (osim ostalilh kriterija pretraživanja: destinacija, datum putovanja, način putovanja i slično). U aplikaciji je moguće ocijeniti putnike (obavezna prethodna registracija) prema vise kriterija. Tema rezervirana za: Ivan Popić
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	20.09.2023.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 04.10.2023.

Ime i prezime studenta:	Ivan Popić
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. studenta, godina upisa:	AI 4631, 27.07.2017.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom: **Mobilna aplikacija za traženje suputnika**

izrađen pod vodstvom mentora Marina Peko, dipl. ing.

i sumentora ,

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 studenta:

Ivan Popić

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. ISTRAŽIVANJE MOBILNIH APLIKACIJA 2

2.1. Internet oglasnik Putoholičari – Tražim suputnika 2

2.2. Mobilna aplikacija BlaBlaCar 3

3. KORIŠTENE TEHNOLOGIJE 4

3.1. Kotlin 4

3.2. XML 5

3.3. Firebase 6

3.4. Android Studio 6

4. FUNKCIONALNOST APLIKACIJE 8

5. IMPLEMENTACIJA ANDROID APLIKACIJE 13

6. ZAKLJUČAK 23

LITERATURA 24

SAŽETAK 25

ABSTRACT 26

ŽIVOTOPIS 27

1. UVOD

Jedan od razloga zašto ljudi ne putuju je sigurno nedostatak društva za putovanje, pa s inspiracijom iz oglasnika „Tražim suputnika“, stvorila se ideja koja bi se mogla pretvoriti u mobilnu aplikaciju. Želeći pomoći ljudima da se još više povežu i da ostvare svoje želje za putovanjem. Učinkovit tijek procesa razvoja aplikacije obuhvaća šest ključnih faza. U ovom završnom radu kao cilj realizirana je mobilna aplikacija koja je namijenjena za operacijski sustav Android. Za izradu mobilne aplikacije koristiti se program „Android Studio“ te programski jezik Kotlin te XML odnosno programski jezik za označavanje podataka. XML (engl. *Extensible Markup Language*) služi za izgled aplikacije, drugim riječima pomoću XML opisnog jezika stvara se sučelje gdje korisnik može vidjeti kod na njemu prihvatljiv način, dok je Kotlin usmjeren na interakciju s aplikacijom.

Ovaj rad podijeljen je u četiri poglavlja. U prvom poglavlju, prikazan je razvoj mobilnih aplikacija gdje se objašnjava koje su to tehnologije ključne za izradu aplikacije. Drugo poglavlje pruža informacije o radu aplikacije. Zatim u trećem poglavlju kreće upoznavanje sa primjerom razvoja android aplikacije. U četvrtom poglavlju prolazimo kroz kod te samu aplikaciju. Na kraju se dolazi do zaključka te osvrta na rad gdje su istaknuti izazovi prilikom razvijanja aplikacije.

1.1. Zadatak završnog rada

Zadatak je napraviti aplikaciju koja omogućava prijavu korisnicima te da ti korisnici imaju mogućnost objavljivanja svojih putovanja te pregled drugih putovanja od drugih registriranih korisnika. Korisnik se može drugom korisniku javiti putem mobilnog broja ili putem e-pošte. Kada se korisnik drugom korisniku javi za određeno putovanje na koje želi ići, korisnik koji je napravio objavu odlučuje želi li tog korisnika dodati na svoje putovanje.

2. ISTRAŽIVANJE MOBILNIH APLIKACIJA

Mobilne usluge velik su dio svakodnevice. Tijekom posljednjeg desetljeća dogodio se ogroman napredak u mobilnoj tehnologiji i pristupačnosti mreži. Napredak u mobilnoj industriji čine mobilne uređaje preferiranom platformom za pristup i razmjenu podataka.

2.1. Internet oglasnik Putoholičari – Tražim suputnika

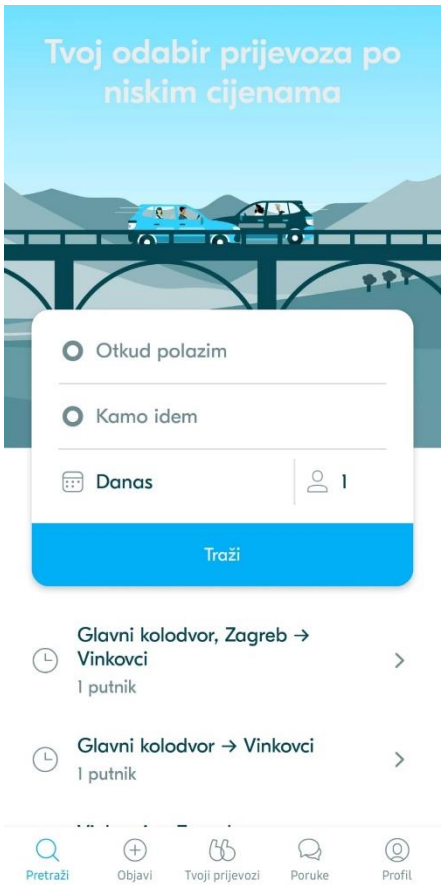
Za aplikaciju koja je cilj završnog rada, značajnu ulogu odigrao je internet oglasnik Putoholičari [1] na čijim *web*-stranicama postoji opcija Tražim suputnika. Na tom internet oglasniku korisnik može objaviti oglas za putovanje. Oglas za putovanje sadrži mjesto na koje korisnik želi putovati i opis bitnih stvari poput: mjesto polaska, datum i vrijeme, godine željenih suputnika te karakterne osobine istih.

Sl. 2.1. *Početni zaslon Internet stranice Putoholičari – Tražim suputnika.*

Ovaj internet oglasnik odigrao je značajnu ulogu u kreiranju ideje za mobilnu aplikaciju za završni rad. Oglasnik, kao i sama aplikacija završnog rada, spaja osobe sličnih želja i interesa te olakšava organizaciju putovanja na različite destinacije.

2.2. Mobilna aplikacija BlaBlaCar

BlaBlaCar [2] je mobilna aplikacija i platforma za zajedničko dijeljenje vožnje koja omogućuje korisnicima da pronađu suvozače s kojima mogu podijeliti putovanje. Ova aplikacija povezuje vozače koji idu na isti put s putnicima koji traže prijevoz do određene lokacije, čime se smanjuju troškovi prijevoza i ekološki otisak. BlaBlaCar aplikacija popularan je način putovanja diljem svijeta, posebno za dulja putovanja između gradova ili zemalja. Kao i primjer iznad, ovaj primjer omogućava korisnicima da uštede novac na prijevozu, upoznaju nove ljude i doprinesu očuvanju okoliša.



Slika 2.2 *Početni izgled mobilne aplikacije BlaBlaCar.*

3. KORIŠTENE TEHNOLOGIJE

Mobilne aplikacije su programske aplikacije dizajnirane za korištenje na mobilnim uređajima kao što su pametni telefoni i tableti. One su postale neizostavan dio suvremenog digitalnog života i igraju ključnu ulogu u načinu komuniciranja, rada, zabave i obavljanje svakodnevnih zadataka. Zbog složenosti mobilne aplikacije koristi se više programskih jezika i alata. Korištenje ovih tehnologija omogućuje programerima da razvijaju mobilne aplikacije koje su sigurne, brze skalabilne i prilagodljive, što je ključno za uspjeh i konkurentske prednosti na tržištu mobilnih aplikacija. Korišteni jezici i alati su:

1. Kotlin
2. XML
3. Firebase
4. Android Studio

Sve nabrojane tehnologije detaljno su pojašnjene u sljedećim potpoglavljima.

3.1. Kotlin

Kotlin je moderni programski jezik koji je razvila tvrtka JetBrains [3]. Predstavljen je 2011. godine, ali je postao posebno popularan u kontekstu razvoja Android aplikacija jer je 2017. godine postao službeni jezik za Android aplikacije. Kotlin se često opisuje kao „statično tipiziran, izražajan i interoperabilan“ jezik, što znači da kombinira najbolje osobine različitih jezika kako bi omogućio produktivnost i jednostavnost razvoja. Neke bitne karakteristike Kotlin programskog jezika:

1. Izražajan i čitljiv – Kotlin je poznat po svojoj čitljivoj sintaksi koja omogućava programerima izražavanje ideja na jasan način. To pomaže u smanjenu koda i povećava čitljivost.
2. Interoperabilnost – Kotlin je potpuno interoperabilan s programskim jezikom Java. To znači da možete koristiti postojeće Java biblioteke i komponente u Kotlin projektima i obrnuto.
3. Nulabilnost – Kotlin ima koncept nulabilnosti ugrađen u jezik, što znači da se varijable podrazumijevano ne mogu postaviti na null. To pomaže u sprječavanju mnogih uobičajenih grešaka povezanih s null referencama u kodu.

4. Razvoj mobilnih aplikacija - Kotlin je postao popularan izbor za razvoj Android aplikacija zbog svoje jednostavne sintakse i poboljšane sigurnosti tipova. Google ga je službeno podržao kao jezik za Android, a mnogi programeri smatraju da je nadmašio Java u kontekstu Android razvoja.
5. Aktivna zajednica - Kotlin ima aktivnu zajednicu programera i razvijača alata koji doprinose njegovom ekosustavu. To uključuje integraciju u popularne razvojne okoline kao što su IntelliJ IDEA i Android Studio [3].

3.2. XML

XML je standardizirani jezik označavanja koji se koristi za označavanje strukturiranih podataka u obliku teksta. XML je dizajniran da bude lagan, čitljiv za ljude i jednostavan za strojeve za analizu. [4] XML najbolje opisuje:

1. Označavanje i hijerarhija - XML koristi oznake (tagove) za označavanje podataka. Oznake su obično napisane u obliku `<ime>`. Svaka oznaka ima početni tag `<ime>` i završni tag `</ime>`, gdje je "ime" naziv elementa ili čvora. Ovo stvara hijerarhijsku strukturu podataka, gdje se elementi mogu gnijezditi unutar drugih elemenata.
2. Tekstualna osnova - XML je opisni jezik, što znači da podaci zapisani u XML formatu mogu biti lako čitani ljudima i uređivani u jednostavnim tekstualnim uređivačima. To ga čini pogodnim za pohranu i razmjenu strukturiranih podataka.
3. Raspon primjene - XML se može koristiti za označavanje različitih vrsta podataka, uključujući konfiguracijske datoteke, podatkovne datoteke, poruke za razmjenu podataka između aplikacija i još mnogo toga. On nije povezan s konkretnom domenom i stoga je "proširiv" (engl. *extensible*) kako bi se prilagodio različitim potrebama.
4. Atributi - Elementi u XML-u mogu imati atribute koji pružaju dodatne informacije o elementima. Atributi se obično nalaze unutar početnog taga elementa i napisani su u obliku `ime="vrijednost"`.

3.3. Firebase

Firebase je razvojna platforma koju je izvorno razvila tvrtka Firebase, Inc., a kasnije je kupila Google. Ova platforma pruža niz alata i usluga koji olakšavaju izradu mobilnih i web aplikacija. Firebase je posebno popularan među programerima zbog svoje jednostavnosti korištenja, skalabilnosti i integracije s drugim Google uslugama. Neke od funkcija koje Firebase pruža:

1. *Realtime Database*, koja je *cloud* bazirana NoSQL baza podataka. Ova baza omogućava trenutni prijenos podataka između klijenata i servera u stvarnom vremenu. To je posebno korisno za izradu aplikacija koje zahtijevaju sinkronizaciju podataka između korisnika, kao što su chat aplikacije ili kolaborativni alati.
2. Firestore je druga Firebaseova baza podataka koja nudi skalabilnu, fleksibilnu i brzu NoSQL bazu podataka s bogatim upitnim mogućnostima. Firestore je posebno prikladan za razvoj web i mobilnih aplikacija i omogućava korisnicima da lako pristupaju i uređuju svoje podatke.
3. Sustav za autentifikaciju koji omogućava jednostavnu integraciju prijave putem e-pošte i lozinke, društvenih mreža (kao što su Google, Facebook i Twitter), sustava za jednokratnu prijavu (OTP) i drugih metoda autentifikacije.
4. Omogućava konfiguriranje pristupa podacima putem pravila i pruža napredne sigurnosne opcije zaštite podataka. [5]

Firebase je široko korišten u razvoju mobilnih i web aplikacija jer pojednostavljuje mnoge aspekte razvoja, uključujući pohranu podataka, autentifikaciju i upravljanje infrastrukturom, što omogućava programerima da se fokusiraju na stvaranje kvalitetnih aplikacija.

3.4. Android Studio

Android Studio je integrirano razvojno okruženje (IDE) koje je specifično dizajnirano za razvoj Android aplikacija. Ovo razvojno okruženje razvijeno je od strane Google-a i pruža alate i resurse koji olakšavaju proces izrade, testiranja i puštanja Android aplikacija. [6] Sadrži moćan grafički alat za dizajniranje korisničkog sučelja aplikacije. Programeri mogu povlačiti i ispuštati

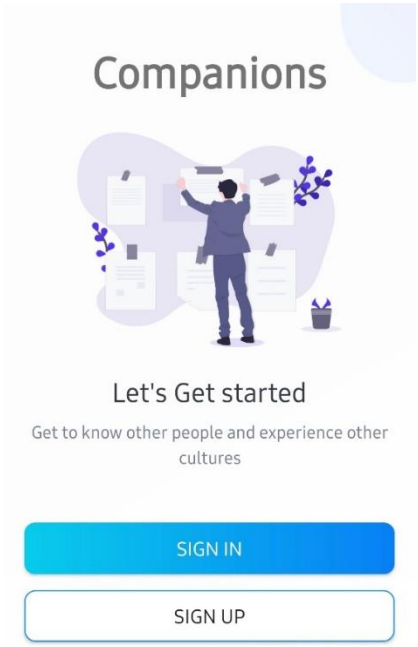
grafičke elemente kako bi stvorili korisničko sučelje bez pisanja koda. Ovo je korisno za brzo prototipiranje i uređivanje izgleda aplikacije. Dolazi s bogatim i inteligentnim uređivačem koda koji podržava programski jezik Kotlin, Java i C++. Android Studio se lako integrira s Firebase-om, što omogućava programerima pristup Firebase uslugama kao što su Firebase Analytics, Firestore i autentifikacija, kako bi unaprijedili svoje aplikacije. Android Studio je osnovni alat za razvoj Android aplikacija i često se koristi od strane programera kako bi stvorili visoko kvalitetne mobilne aplikacije za Android operacijski sustav. Kasnije u radu se pokazuje kod u programu Android Studio.

4. FUNKCIONALNOST APLIKACIJE

Mobilne aplikacije imaju mnoge prednosti, kako za korisnike tako i za razvojne timove i poslovne subjekte. Mobilne aplikacije pružaju korisnicima brz i jednostavan pristup informacijama, uslugama i zabavi putem njihovih pametnih telefona i tableta. To čini aplikacije izuzetno praktičnim i dostupnim. To se želi ostvariti i ovom aplikacijom, odnosno pokazat ljudima jednostavan i brz način kako upoznati nove ljude i nove kulture putovanjem.

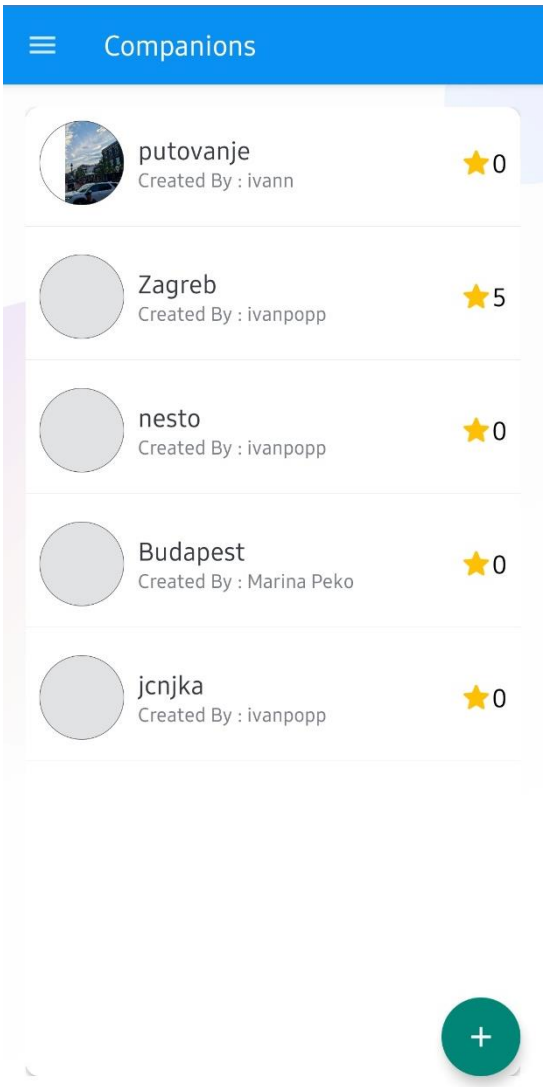
Ideja ove aplikacije je da ljudi koji ne žele provesti putovanje sami, ili se pak ne snalaze u organiziranju vlastitih putovanja to naprave s drugim osobama te da lakše otkriju svijet te samim time otkriju sebe. Aplikacija omogućuje lako povezivanje s drugim ljudima koji imaju isto ili slično zanimanje za određena mjesta.

Da bi se sve to ostvarilo korisnik se prvo mora registrirati. Kada korisnik upali aplikaciju pojavljuje mu se početni zaslon gdje odabire želi li se prijaviti ili registrirati. Početni zaslon prikazan je na slici 4.1. Klikom na opciju Sign up, korisniku se otvara zaslon gdje se može registrirati. Korisnik unosi svoje ime, svoju e-poštu te zatim i svoju zaporku.



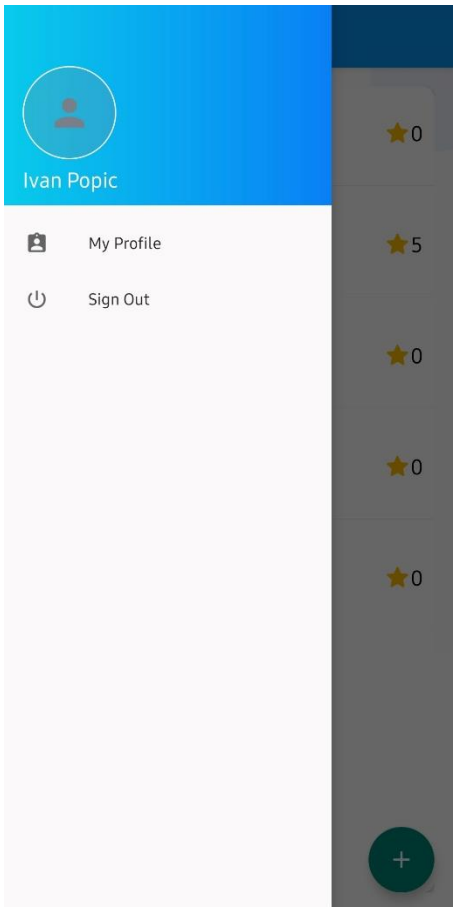
Sl. 4.1. Početno sučelje prije korisničke prijave.

Kada se korisnik uspješno registrira, tada ide na opciju Sign in, odnosno klikom na to dugme korisniku se otvara novi zaslon gdje se može uspješno prijaviti. Kada se korisnik prijavi na svoj račun otvara mu se glavni zaslon gdje može vidjeti sve objave drugih korisnika koji su se već registrirali, te samim time ocjenu putovanja (Sl. 4.2.).



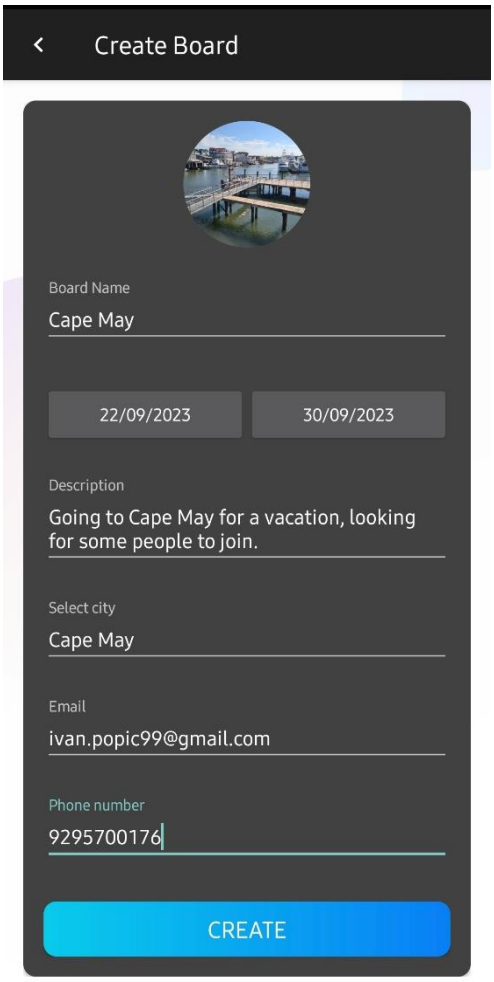
Sl. 4.2. *Izgled sučelja nakon prijave korisnika.*

Na početnom zaslonu također se mogu vidjeti i opcija za dodavanje objave te izbornik gdje korisnik može otići na svoj profil ili se odjaviti (Sl. 4.3.). Klikom na svoj profil korisnik može postaviti sliku profila te dodati svoj telefonski broj.



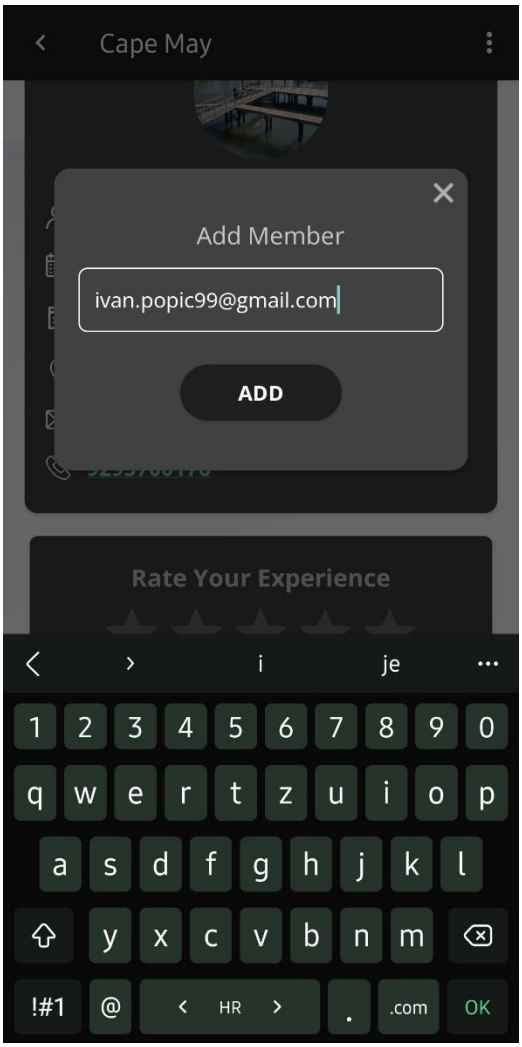
Sl. 4.3. *Izgled bočnog izbornika nakon pritiska na hamburger gumb.*

Ako se korisnik odluči za dodavanje objave, klikom na dugme „+“otvara mu se zaslon gdje korisnik može stvoriti svoju objavu, odnosno može sam postaviti gdje želi putovati, te tamo postavlja sliku, datum, opis putovanja, gdje korisnik opisuje gdje želi ići i također može navesti njegovu osobnost da bi privukao druge korisnike sličnih osobnosti. Zatim odabire grad te upisuje svoju e-poštu te broj na koji ga drugi korisnici mogu kontaktirati. Na slici se može vidjeti sučelje gdje korisnik unosi podatke, te na slici (Sl. 4.4.) može se vidjeti primjer ispunjene objave.



Sl. 4.4. *Primjer popunjene objave.*

Kada korisnik objavi svoju objavu gdje želi putovati, onda i drugi korisnici mogu vidjeti tu objavu te ući u tu objavu. Kada korisnik uđe u objavu može klikom na grad vidjeti gdje se nalazi taj grad, te također ako želi može korisnika kontaktirati putem e-pošte ili broja telefona. Ako korisnik koji je objavio objavu želi povesti drugog korisnika koji ga je kontaktirao na svoje putovanje može ga dodati na listu pomoću njegove e-pošte (Sl. 4.5.), na kraju se stvara lista ljudi koji idu na putovanje te svi korisnici mogu vidjeti tu listu, te kontaktirati druge osobe također putem e-pošte.



Sl. 4.5. *Dodavanje članova putovanja.*

5. IMPLEMENTACIJA ANDROID APLIKACIJE

Kada se prođe kroz sve osnove programa Android studio te jezik Kotlin u kojem je napisana ova aplikacija, upušta se u rad na projektu „Tražim suputnika“.

Na početku se odmah stvara takozvani *Splashscreen* radi vizualnog ugođaja korisnika pri otvaranju aplikacije. Napravljen je Zaslon gdje je *Splashscreen* postavljen na 3 sekunde te se u ovom kodu prikazanom na slici (Sl. 5.1.) navodi da se on prikaže prvi po redu pri otvaranju aplikacije

```
ivanp
class SplashActivity : AppCompatActivity() {
    private var binding:ActivitySplashBinding?= null

    ivanp
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivitySplashBinding.inflate(layoutInflater)
        setContentView(binding?.root)


        window.setFlags(
            WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN
        )

        Handler(Looper.getMainLooper()).postDelayed({
            var currentUserID = FirestoreClass().getCurrentUserId()
            if (currentUserID.isNotEmpty()){
                val intent = Intent( packageContext: this, MainActivity::class.java)
                startActivity(intent)
            }else{
                val intent = Intent( packageContext: this, IntroActivity::class.java)
                startActivity(intent)
            }
            finish()
        }, delayMillis: 3000)


        binding?.tvAppName?.typeface = Typeface.createFromAsset(this.assets, path: "titlovi/carbon bl.ttf")

    }
```

Sl. 5.1. *Prikaz Splashscreena u kodu.*

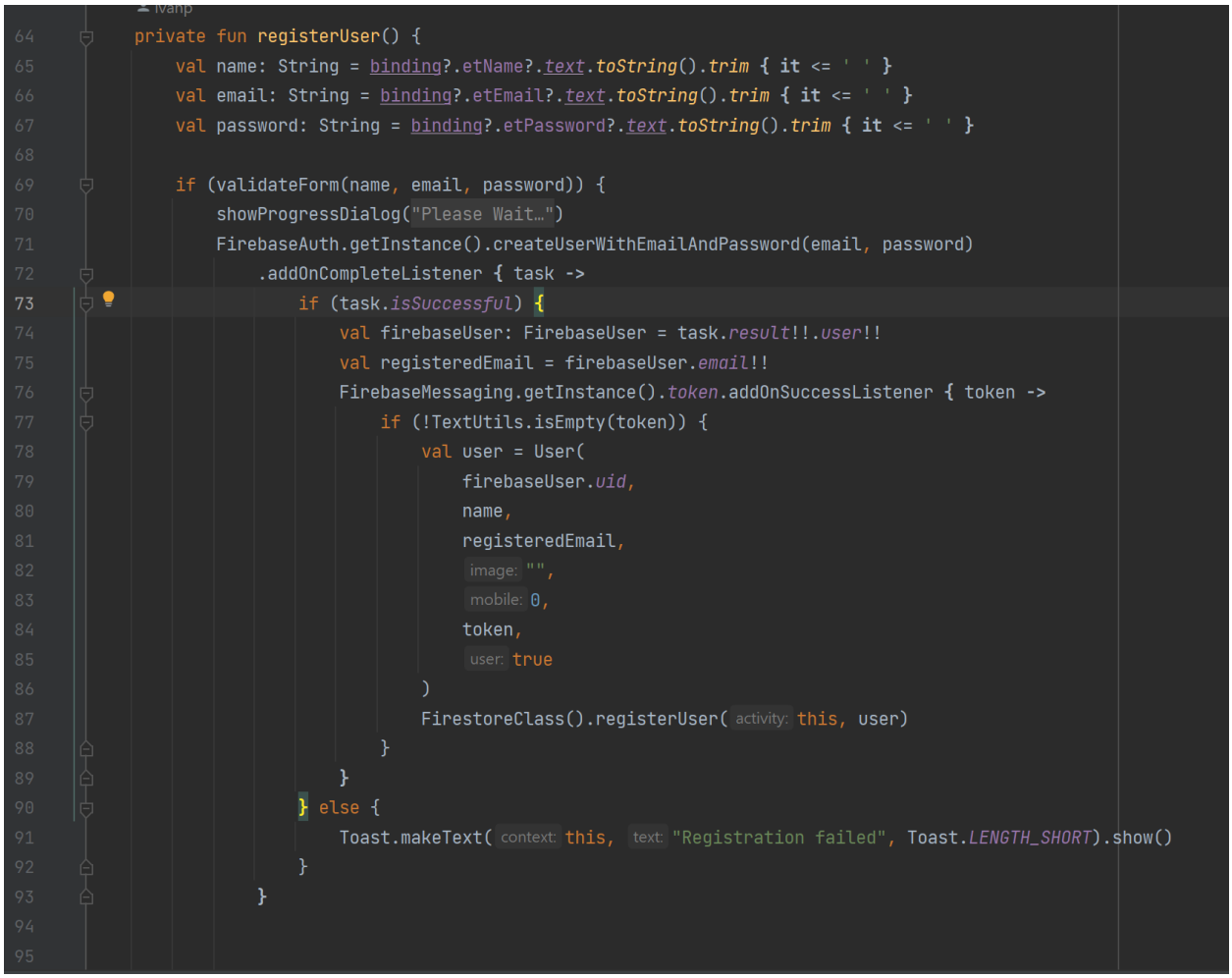
Zatim nakon toga pravi se tri nova zaslona, prvi koji dolazi nakon *Splashscreena* je zaslon gdje je ponuđena Sign In i Sign Up opcija te svaka otvara svoju aktivnost, gdje se dolazi do sve tri aktivnosti odnosno zaslona (Sl. 5.2.).

Dizajnirao se izgled u XML kodu, zatim kada se dobio konačan i željeni izgled, napisan je kod kako bi to sve funkcioniralo.

```
11  class IntroActivity : BaseActivity() {
12      private var binding:ActivityIntroBinding? = null
13
            ivanp
14      override fun onCreate(savedInstanceState: Bundle?) {
15          super.onCreate(savedInstanceState)
16          binding = ActivityIntroBinding.inflate(layoutInflater)
17          setContentView(binding?.root)
18
19          window.setFlags(
20              WindowManager.LayoutParams.FLAG_FULLSCREEN,
21              WindowManager.LayoutParams.FLAG_FULLSCREEN
22          )
23
24          binding?.btnSignUpIntro?.setOnClickListener{ it: View!
25                  startActivity(Intent( packageContext: this, SignUpActivity::class.java))
26
27          }
28
29          binding?.btnSignInIntro?.setOnClickListener{ it: View!
30              startActivity(Intent( packageContext: this, LogInActivity::class.java))
31
32          }
33
34      }
35
```

Sl. 5.2. *Prikaz dijela koda početnog sučelja prije prijave.*

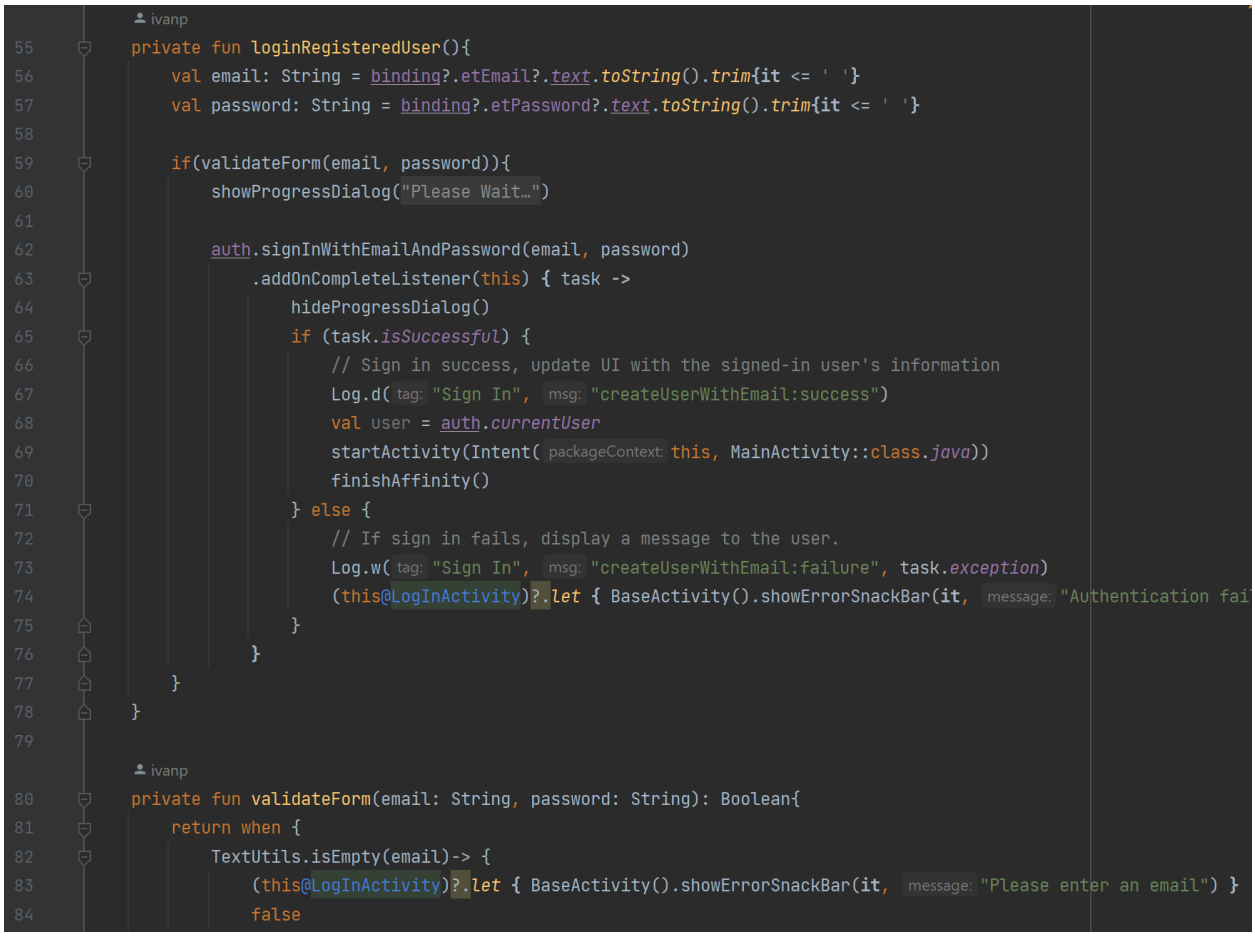
Kako bi se korisnik uspješno mogao registrirati i prijaviti, treba se stvoriti baza podataka. Za ovaj projekt se koristi Firebase i njegov *Cloud Firestore Database*, što možemo vidjeti i u kodu na slici (Sl. 5.3.).



```
64  private fun registerUser() {
65      val name: String = binding?.etName?.text.toString().trim { it <= ' ' }
66      val email: String = binding?.etEmail?.text.toString().trim { it <= ' ' }
67      val password: String = binding?.etPassword?.text.toString().trim { it <= ' ' }
68
69      if (validateForm(name, email, password)) {
70          showProgressDialog("Please Wait…")
71          FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password)
72              .addOnCompleteListener { task ->
73                  if (task.isSuccessful) {
74                      val firebaseUser: FirebaseUser = task.result!!.user!!
75                      val registeredEmail = firebaseUser.email!!
76                      FirebaseMessaging.getInstance().token.addOnSuccessListener { token ->
77                          if (!TextUtils.isEmpty(token)) {
78                              val user = User(
79                                  firebaseUser.uid,
80                                  name,
81                                  registeredEmail,
82                                  image: "",
83                                  mobile: 0,
84                                  token,
85                                  user: true
86                              )
87                              FirestoreClass().registerUser( activity: this, user)
88                          }
89                      }
90                  } else {
91                      Toast.makeText( context: this,  text: "Registration failed", Toast.LENGTH_SHORT).show()
92                  }
93              }
94
95
```

Sl. 5.3. *Firebase autentifikacija za registraciju korisnika.*

Kada se korisnik registrira onda mora proći kroz proces prijave u aplikaciju što se također dobiva Firebaseom (Sl. 5.4.).



```
ivanp
private fun loginRegisteredUser(){
    val email: String = binding?.etEmail?.text.toString().trim{it <= ' '}
    val password: String = binding?.etPassword?.text.toString().trim{it <= ' '}

    if(validateForm(email, password)){
        showProgressDialog("Please Wait…")

        auth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                hideProgressDialog()
                if (task.isSuccessful) {
                    // Sign in success, update UI with the signed-in user's information
                    Log.d( tag: "Sign In",  msg: "createUserWithEmail:success")
                    val user = auth.currentUser
                    startActivity(Intent( packageContext: this, MainActivity::class.java))
                    finishAffinity()
                } else {
                    // If sign in fails, display a message to the user.
                    Log.w( tag: "Sign In",  msg: "createUserWithEmail:failure", task.exception)
                    (this@LogInActivity)?.let { BaseActivity().showErrorSnackBar(it,  message: "Authentication fai
                }
            }
    }
}

ivanp
private fun validateForm(email: String, password: String): Boolean{
    return when {
        TextUtils.isEmpty(email)-> {
            (this@LogInActivity)?.let { BaseActivity().showErrorSnackBar(it,  message: "Please enter an email") }
            false
```

Sl. 5.4. *Prikaz koda prilikom korisničke prijave.*

Kada se korisnik uspješno prijavi dolazi se na glavnu stranicu gdje su sve objave drugih registriranih korisnika. Prvo što je stvoreno je glavni izbornik gdje korisnik može otići i urediti svoj profil, te se na tom izborniku i odjaviti. Prvo je dizajniran izbornik i izgled ladice kad se izbornik otvori, zatim se stvara kod gdje se obavlja funkcionalnost tog izbornika (Sl. 5.5.).

```
ivanp
override fun onNavigationItemSelected(item: MenuItem): Boolean {

    when (item.itemId) {
        R.id.nav_my_profile -> {
            startActivityForResult(
                Intent( packageContext: this, MyProfileActivity::class.java),
                MY_PROFILE_REQUEST_CODE
            )
        }

        R.id.nav_sign_out -> {
            FirebaseAuth.getInstance().signOut()

            val intent = Intent( packageContext: this, IntroActivity::class.java)
            intent.addFlags( flags: Intent.FLAG_ACTIVITY_CLEAR_TOP or Intent.FLAG_ACTIVITY_NEW_TASK)
            startActivity(intent)
            finish()
        }

    }
    binding?.drawerLayout?.closeDrawer(GravityCompat.START)

    return true
}
```

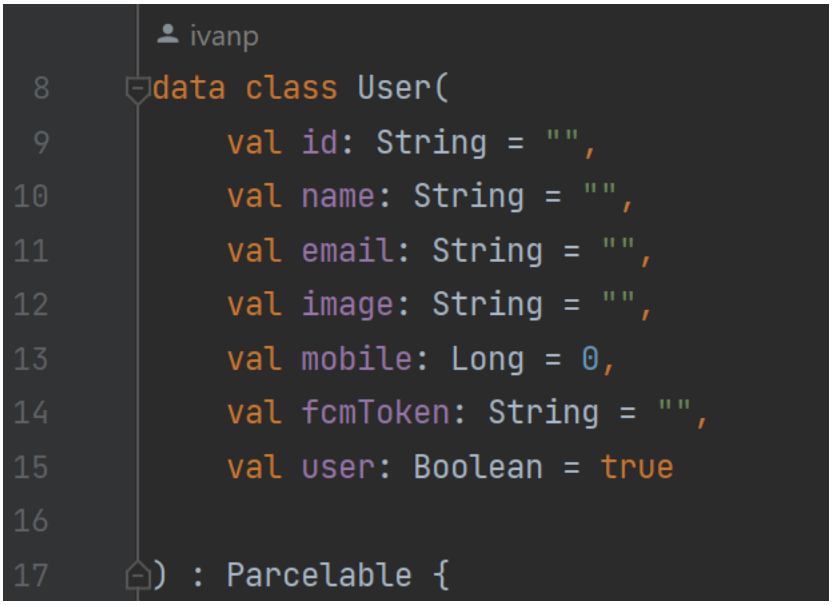
Sl. 5.5. *Prikaz dijela koda hamburger gumba.*

Zatim se pravi objekt koji se zove *Constants*, gdje se sadrže konstantne vrijednosti poput fotografije, imena, e-pošte i drugo što je korišteno prilikom ažuriranja profila (Sl. 5.6.).

```
ivanp
private fun updateUserProfileData(){
    val userHashMap = HashMap<String, Any>()

    if (mProfileImageURL.isNotEmpty() && mProfileImageURL != mUserDetails.image){
        userHashMap[Constants.IMAGE] = mProfileImageURL
    }

    if(binding?.etName?.text.toString() != mUserDetails.name){
        userHashMap[Constants.NAME] = binding?.etName?.text.toString()
    }
    if (binding?.etMobile?.text.toString()!=mUserDetails.mobile.toString()){
        userHashMap[Constants.MOBILE] = binding?.etMobile?.text.toString().toLong()
    }

    FirestoreClass().updateUserProfileData( activity: this, userHashMap)

}
```

Sl. 5.6. *Prikaz funkcije za ažuriranje korisničkog profila.*

Napravljen je *data class* Board (*data class* za objave) i User (*data class* za korisnike) gdje se su se koristile vrijednosti poput imena, e-pošte, članovi i drugo (Sl. 5.7.). Dizajnirani su kako bi se pojednostavio i ubrzao proces pisanja koda za definiranje i upravljanje podacima, te kako bi se smanjile greške u pisanju koda. Dosta je popularan u kotlinu, te je i u ovom projektu pokazao svoju vrijednost.

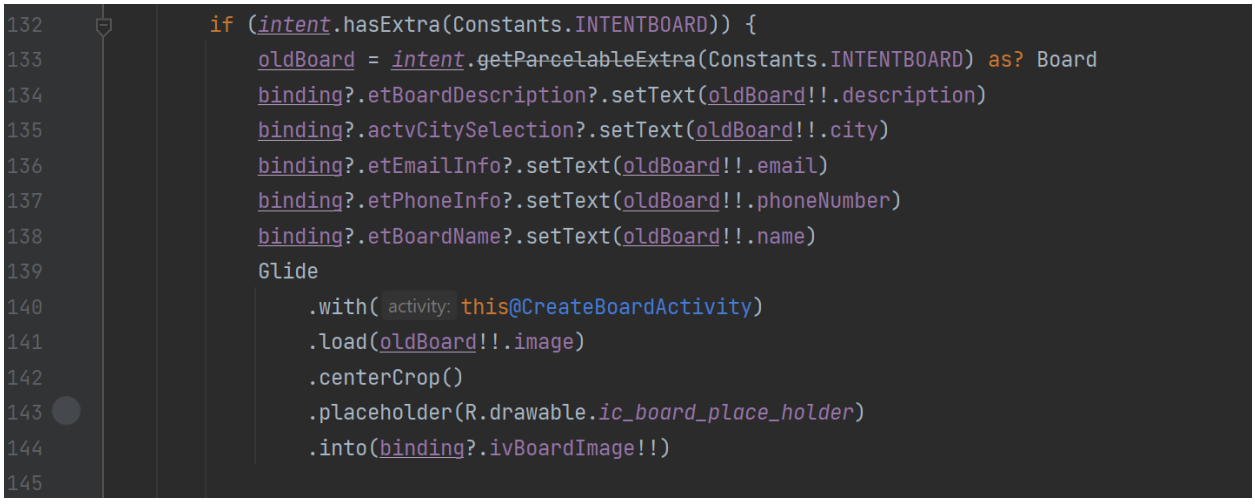


Sl. 5.7. *Prikaz data class za korisnika.*

Data class Board i User su se koristili tijekom pravljenja objave. Klikom na dugme plus, koji je napravljen u glavnom zaslonu, korisnika se odvodi do aktivnosti gdje može kreirati svoju objavu. Korisnik tu može staviti fotografiju koju želi, te navodi ime, prezime, datum polaska i odlaska, opis putovanja, te najvažnije upisuje grad u koji želi ići (Sl. 5.8.). Kada korisnik upisuje grad, preko Google API mu se nude gradovi na temelju Google prijedloga. Preko Google API drugi korisnici također mogu kliknuti na grad koji je korisnik objavio i vidjeti gdje se nalazi to mjesto (Sl. 5.9.).

```
ivanp
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityCreateBoardBinding.inflate(layoutInflater)
    setContentView(binding?.root)

    if (!Places.isInitialized()) {
        Places.initialize(
            applicationContext,
            apiKey: API_KEY_PART_1 + API_KEY_PART_2 + API_KEY_PART_3 + API_KEY_PART_4,
            Locale.US
        )
    }

    autocompleteFragment =
        supportFragmentManager.findFragmentById(R.id.autocomplete_fragment) as AutocompleteSupportFragment?

    autocompleteFragment!!.setHint("Enter a city name")
    autocompleteFragment!!.setPlaceFields(
        listOf(
            Place.Field.ID,
            Place.Field.NAME,
            Place.Field.LAT_LNG
        )
    )
    autocompleteFragment!!.setOnPlaceSelectedListener(object : PlaceSelectionListener {
        override fun onPlaceSelected(place: Place) {
```

Sl. 5.8 *Prikaz dijela Google API koji je korišten za lokaciju.*



```
    if (intent.hasExtra(Constants.INTENTBOARD)) {
        oldBoard = intent.getParcelableExtra(Constants.INTENTBOARD) as? Board
        binding?.etBoardDescription?.setText(oldBoard!!.description)
        binding?.actvCitySelection?.setText(oldBoard!!.city)
        binding?.etEmailInfo?.setText(oldBoard!!.email)
        binding?.etPhoneInfo?.setText(oldBoard!!.phoneNumber)
        binding?.etBoardName?.setText(oldBoard!!.name)
        Glide
            .with( activity: this@CreateBoardActivity)
            .load(oldBoard!!.image)
            .centerCrop()
            .placeholder(R.drawable.ic_board_place_holder)
            .into(binding?.ivBoardImage!!)

```

Sl. 5.9. *Prikaz dijela koda gdje korisnik unosi podatke prilikom izrade objave putovanja.*

Kada se uspješno implementiralo postavljanje objave, kreće se na izradu aktivnosti odnosno zaslona gdje korisnik klikom na objavu ulazi u objavu i vidi sve što je korisnik objavio.

Pravi se adapter da korisnici mogu klikom ući u objavu, te pomoću tog adaptera drugi korisnici mogu ukratko vidjeti tko je objavio objavu te na koje mjesto se planira ići i ocjena (Sl. 5.10.).

```
ivanp *
override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
    val model = list[position]
    if (holder is MyViewHolder) {
        Glide
            .with(context)
            .load(model.image)
            .centerCrop()
            .placeholder(R.drawable.ic_board_place_holder)
            .into(holder.itemView.findViewById(R.id.iv_board_image))

        holder.itemView.findViewById<TextView>(R.id.tv_name).text = model.name
        holder.itemView.findViewById<TextView>(R.id.tv_created_by).text =
            "Created By : ${model.createdBy}"

        holder.itemView.setOnClickListener { it: View!
            if (itemClickListener != null) {
                itemClickListener!!.onClick(position, model)
            }
        }

        var total = 0.0
        var count = 0.0
        var average = 0.0
        for ((key, value) in model.feedback) {
            println("$key = $value")
            val feedback = model.feedback.getValue(key)
            total += feedback.rating.toFloat();
            count += 1;
            average = total / count;
        }
```

Sl. 5.10. *Prikaz adaptera objave.*

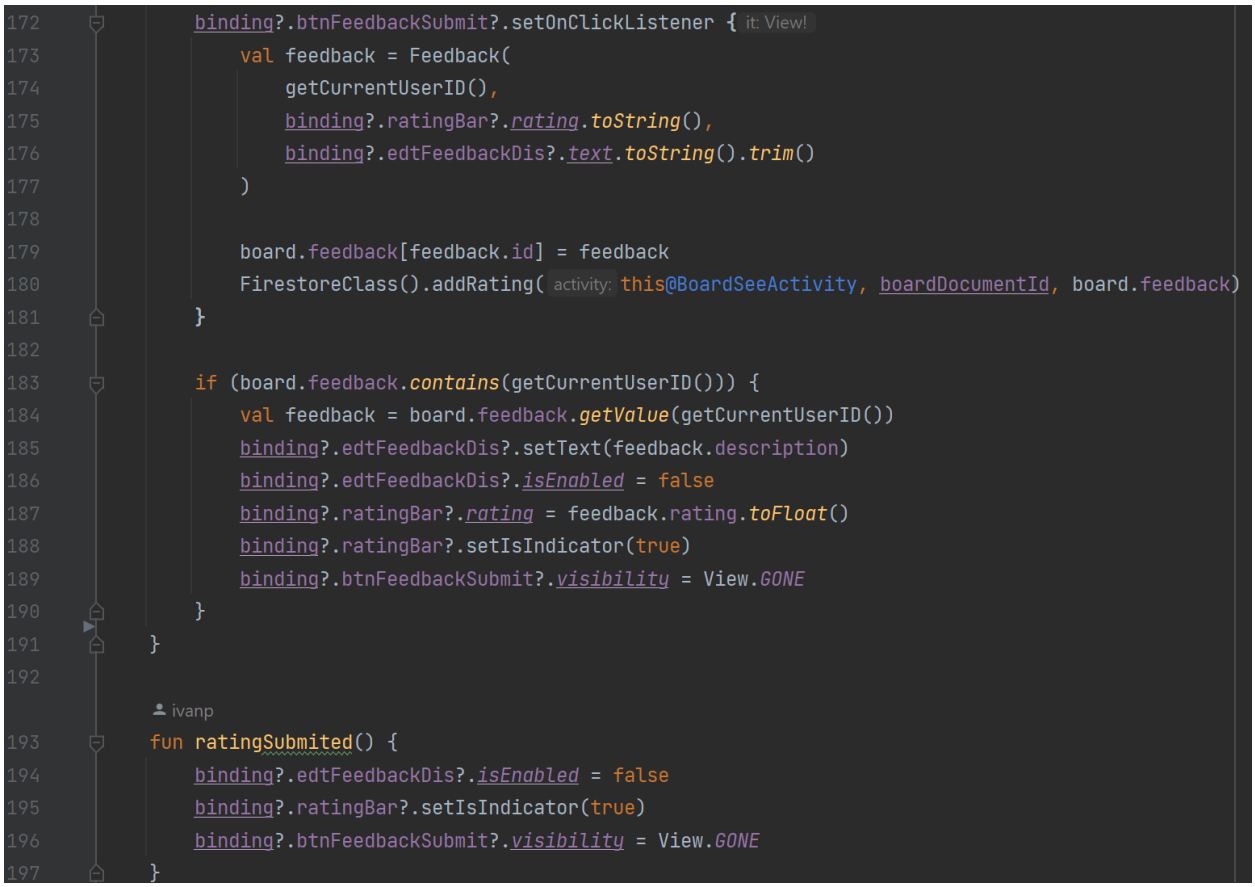
Nakon što je uspješno napravljen adapter, korisnik dolazi do konačnog koraka gdje može pregledati objave od drugih korisnika te vidjeti druge suputnike koji prisustvuju istom putovanju.

Korisnik se može drugom korisniku javiti putem e-pošte ili putem broja telefona. Ako se korisnik koji je objavio objavu tijekom kontakta s drugim korisnikom povezao on ga može dodati na svoju listu članova koju smo nazvali *members* (Sl. 5.11.).

```
205     fun createMemberDialog() {
206         dialog = Dialog( context: this@BoardSeeActivity)
207         dialog!!.window!!.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))
208         dialog!!.setContentView(R.layout.dialog_add_member)
209         val edtEmail = dialog!!.findViewById<View>(R.id.edt_email) as EditText
210         val btnAdd = dialog!!.findViewById<View>(R.id.btn_add) as CardView
211         val btnClose = dialog!!.findViewById<View>(R.id.btn_close) as ImageView
212         btnClose.setOnClickListener { it: View!
213             dialog!!.dismiss()
214         }
215         btnAdd.setOnClickListener { it: View!
216             if (edtEmail.text.toString().isNotEmpty()) {
217                 FirestoreClass().addMembers(
218                     activity: this@BoardSeeActivity,
219                     edtEmail.text.toString().trim(),
220                     mTravelDetail.documentId,
221                     mTravelDetail.members
222                 )
223             }
224         }
225         dialog?.setOnDismissListener { it: DialogInterface!
226             edtEmail.setText("")
227         }
228     }
229
```

Sl. 5.11. *Prikaz dijela koda za dodavanje članova.*

Konačno drugi korisnik može objaviti svoju ocjenu kako on vidi to putovanje, ocjena je u rasponu od jedan do pet (Sl. 5.12.).



Sl. 5.12. *Dio koda koji prikazuje ocjenjivanje putovanja.*

6. ZAKLJUČAK

U ovom se radu razmatra razvoj mobilne aplikacije koja omogućuje korisnicima dijeljenje svojih putničkih želja i omogućuje drugim korisnica da se javljaju kako bi se pridružili tim putovanjima. Mobilne aplikacije su postale neizostavan dio života ljudi, a ova aplikacija pruža korisnicima jednostavan i učinkovit način za povezivanje i dijeljenje iskustava putovanja. Kroz razvoj ove aplikacije koriste se suvremene tehnologije kao što su Kotlin programski jezik za razvoj mobilnih aplikacija, XML za oblikovanje korisničkog sučelja, Firebase kao *backend* platformu za pohranu podataka i autentifikaciju korisnika te Android Studio kao razvojno okruženje. Sve ove tehnologije zajedno čine ovu aplikaciju stabilnom, skalabilnom i sigurnom za korisnike.

Ova aplikacija olakšava korisnicima da pronađu putničke partnere i dijele troškove putovanja, što može učiniti njihova putovanja pristupačnijim i zabavnijim.

Razvoj mobilnih aplikacija je dinamičan proces koji zahtijeva pažljivo planiranje, dizajniranje i implementaciju, ali rezultat je korisna aplikacija koja može poboljšati iskustvo putovanja za mnoge korisnike.

U budućnosti se mogu razmotriti dodavanje još opcija poput razgovora unutar same aplikacije putem poruka, bolji dizajn, napraviti ograničenja za korisnike i drugo.

Mobilne aplikacije imaju ogroman potencijal za olakšavanje života ljudi i povezivanje zajednica, te ovaj projekt predstavlja samo mali korak prema ostvarenju tog potencijala.

LITERATURA

[1] Putoholičari, "Putoholičari", 2023. [Na internetu]. Dostupno: https://www.putoholicari.rtl.hr/. [pristupano 13.09.2023.].

[2] BlaBlaCar, "BlaBlaCar", 2023. [Na internetu]. Dostupno: https://www.blablacar.hr/apps-mobile. [pristupano 13.09.2023.].

[3] Kotlin, "Kotlin", 2023. [Na internetu]. Dostupno: https://kotlinlang.org/. [pristupano 27.08.2023.].

[4] W3Schools, "XML Tutorial", 2023. [Na internetu]. Dostupno: https://www.w3schools.com/xml/. [pristupano 27.08.2023.].

[5] Google for Developers, "Firebase", 2023. [Na internetu]. Dostupno: https://firebase.google.com/. [pristupano 11.08.2023.].

[6] Google for Developers, "Get started with Android", 2023. [Na internetu]. Dostupno: https://developer.android.com/get-started/overview. [pristupano 27.08.2023.].

SAŽETAK

Ova mobilna aplikacija omogućuje korisnicima da dijele svoje putničke želje i povezuju se s drugim putnicima radi zajedničkog planiranja putovanja. Korisnici mogu objavljivati svoje putničke želje, navodeći destinacije i datume putovanja. Drugi korisnici mogu pregledavati ove objave i javiti se korisnicima kako bi se pridružili njihovim putovanjima.

Kroz korištenje suvremenih tehnologija kao što su Kotlin, XML, Firebase i Android Studio, aplikacija nudi sigurno, skalabilno i jednostavno korisničko iskustvo. Cilj aplikacije je olakšati putovanja, potaknuti povezivanje putnika i doprinijeti boljem iskustvu putovanja za sve korisnike.

Aplikacija pruža mogućnost komunikacije između korisnika putem mobilnog telefona ili e-pošte kako bi olakšala dogovaranje detalja putovanja. Ovo pomaže korisnicima da se upoznaju i osjećaju sigurnijima prije nego što krenu na putovanje.

Ključne riječi: Android Studio, Firebase, Kotlin, Mobilna aplikacija, XML.

ABSTRACT

This mobile application enables users to share their travel desires and connect with other travelers for collaborative trip planning. Users can post their travel wishes, specifying destinations and travel dates. Other users can browse these posts and reach out to fellow travelers to join their journeys.

The application also facilitates communication between users through mobile phones or email to streamline the arrangement of travel details. This helps users get acquainted and feel more secure before embarking on their trips.

By utilizing modern technologies such as Kotlin, XML, Firebase, and Android Studio, application offers a secure, scalable, and user-friendly experience. The goal of the application is to simplify travel, promote traveler connections, and enhance the overall travel experience for all users.

Keywords: Android Studio, Firebase, Kotlin, Mobile Application, XML.

ŽIVOTOPIS

Ivan Popić, rođen 19. siječnja 1999. godine u Vinkovcima. Pohađao osnovnu školu Vladimira Nazora u Vinkovcima. U istom gradu upisao je srednju Tehničku školu Ruđera Boškovića gdje je završio smjer Elektrotehniku. Godine 2017. postao je student na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku na preddiplomskom stručnom studiju Računarstvo.

Na fakultetu se upoznao s osnovama programiranja u programskom jeziku C, tehnologijama izrade web aplikacija, mobilnih aplikacija i radom s jezicima HTML, CSS, Kotlin, PHP, SQL i PYTHON.

Pohađao je stručnu praksu u tvrtki Informatika Fortuno gdje je iskusio rad u timovima na projektima i također stekao osnovno poznavanje područja znanosti o podacima.

_____

Potpis autora