Razvoj aplikacija za Android operacijski sustav

Šalgaj, Dario

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Virovitica University of Applied Sciences / Veleučilište u Virovitici**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:165:824280

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-01-02**

Repository / Repozitorij:

Virovitica University of Applied Sciences Repository - Virovitica University of Applied Sciences Academic Repository

VELEUČILIŠTE U VIROVITICI

Preddiplomski stručni studij Programsko inženjerstvo

DARIO ŠALGAJ

RAZVOJ APLIKACIJA ZA ANDROID OPERACIJSKI SUSTAV

ZAVRŠNI RAD

VIROVITICA, 2022

VELEUČILIŠTE U VIROVITICI

Preddiplomski stručni studij Programsko inženjerstvo

RAZVOJ APLIKACIJA ZA ANDROID OPERACIJSKI SUSTAV

ZAVRŠNI RAD

Predmet: Programiranje mobilnih aplikacija

Mentor:
Ivan Heđi, dipl.ing., v.pred.

Student:
Dario Šalgaj

VIROVITICA, 2022

Veleučilište u Virovitici

Preddiplomski stručni studij Računarstva - Smjer Programsko inženjerstvo

OBRAZAC 1b

ZADATAK ZAVRŠNOG RADA

Student/ica: **DARIO ŠALGAJ** **JMBAG:** **0307016317**

Imenovani mentor: Ivan Heđi, dipl. ing., v. pred.

Imenovani komentor: -

Naslov rada:

Razvoj aplikacija za Android operacijski sustav

Puni tekst zadatka završnog rada:

U vašem radu opišite Android operacijski sustav i programski jezik Java za izradu aplikacija. Kao praktični primjer implementacije teorijskog dijele osmislite i izradite aplikaciju pomoću koje se korisnici mogu naručiti te pratiti status narudžbe servisa automobila. Za slanje obavijesti o statusu narudžbe korisnika koristite sustav Firebase Cloud Messaging. Osim programskog rješenja, u pisanom dijelu završnog rada opišite ukratko korištene tehnologije, opišite detaljno arhitekturu sustava te opišite detaljno odabrane procese i/ili funkcije unutar same aplikacije. Prilikom opisivanja, osim neformalnih koristite i neke od formalnih metoda koje poznajete.

Datum uručenja zadatka studentu/ici: 28.07.2022..
Rok za predaju gotovog rada: 09.09.2022.

Mentor:

Ivan Heđi, dipl. ing., v. pred.

Dostaviti:
1. Studentu/ici
2. Povjerenstvu za završni rad - tajniku

RAZVOJ APLIKACIJA ZA ANDROID OPERACIJSKI SUSTAV
DEVELOPMENT OF APPLICATIONS FOR THE ANDROID OPERATING SYSTEM

Sažetak

Cilj ovoga rada bila je izrada mobilne aplikacije koja služi za rezerviranje servisa na vlastitom automobilu. Aplikacija je razvijena u svrhu olakšanja komunikacije između auto servisera i korisnika kako bi mogli održavati vlastiti automobil. Korisnik rezervira tip servisa, datum i vrijeme koje mu odgovara pa čeka odobrenje rezervacije od strane auto servisera. Korisnik može odabrati više mogućih servisa i vidjeti lokaciju svakoga. Auto serviser može prihvatiti rezervacije koje dobiva od korisnika i prati njihov tijek događaja. Također može pregledati sve, do sada, obavljene rezervacije.

Ključne riječi: Mobilna aplikacija, Android aplikacija, automobil, servisi

Summary

The goal of this work was to create a mobile application that books services on your own car. The application was developed for the purpose of facilitating communication between car repairers and users to be able to maintain their car. The user books the service type, date and time that suits him, then waits for the approval of the reservation by the car repairer. The user can choose from several possible service stations and see the location of each one. The car repairer can accept reservations received from users and he is able to monitor their progress. He can also review all reservations he has done so far.

Keywords: Mobile application, Android application, car, services

SADRŽAJ

1. UVOD

U ovome završnom radu tema je izrada mobilne aplikacije koja može olakšati i ubrzati komunikaciju između korisnika i auto servisera. U aplikaciji postoji mogućnost rezervirati servis svog automobila u par klikova. Kod izrade aplikacije korištena su znanja stečena na veleučilištu iz kolegija „Programiranje mobilnih aplikacija“ i „Objektno-orijentirano programiranje“. Korišteno je razvojno okruženje Android studio koji se temelji na objektno-orijentiranom programskom jeziku Java. Završni rad se sastoji od teorijskog osvrta na tehnologije i programske jezike, objašnjeni su dijelovi koda te je vidljiv vizualni opis nastanka aplikacije.

2. Opis korištenih alata/tehnologija

Aplikacija je razvijena u razvojnom okruženju Android Studio koji je namijenjen za kreiranje Android aplikacija.

2.1. Android Studio

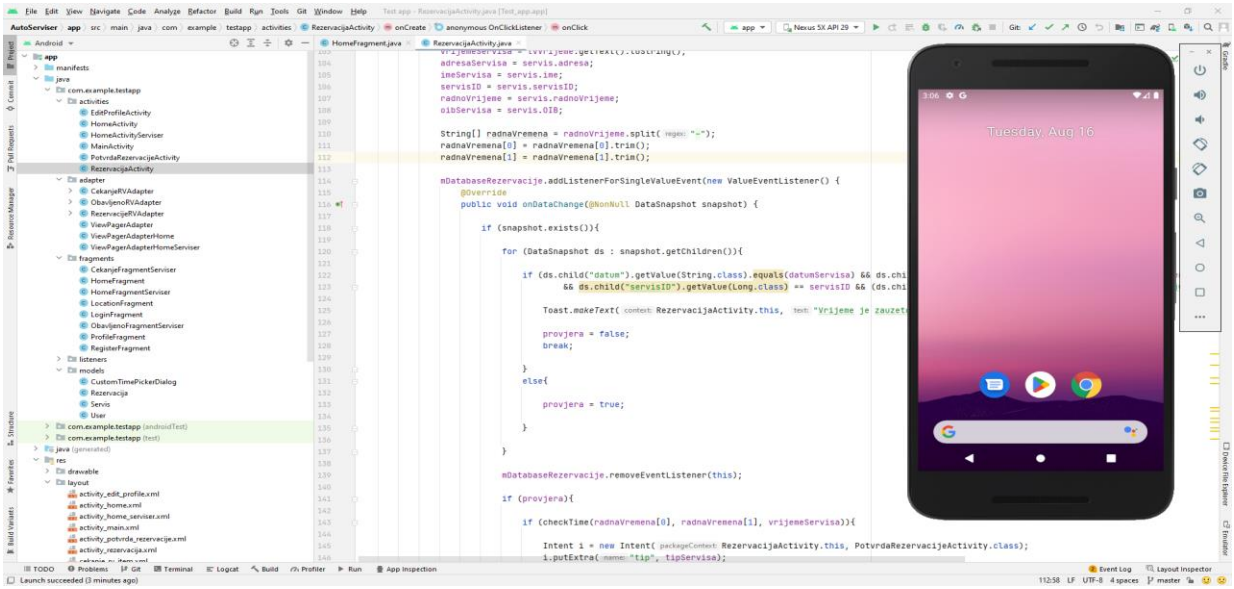
Android Studio je službeno integrirano razvojno okruženje (IDE) za razvoj Android aplikacija, temeljeno na IntelliJ IDEA. Na slici 1. prikazan je primjer izrade aplikacija u Android Studio-u. Osim IntelliJ moćnog uređivača koda i razvojnih alata, Android Studio nudi još više značajki koje povećavaju produktivnost pri izradi Android aplikacija kao što su:

- Fleksibilan sustav izrade temeljen na Gradle-u
- Brz emulator bogat značajkama
- Jedinstveno okruženje u kojem možete razvijati za sve Android uređaje
- Predlošci koda i GitHub integracija koja će vam pomoći da izgradite zajedničke značajke aplikacije
- Opsežni alati i okviri za testiranje
- C++ i NDK podrška
- Ugrađena podrška za Google Cloud Platform, što olakšava integraciju Google Cloud Messaging i App Engine

Svaki projekt u Android Studio-u sadrži jedan ili više modula s datotekama izvornog koda i datotekama resursa. Vrste modula uključuju:

- Moduli Android aplikacije
- Library moduli
- Google App Engine moduli [1]

[1] Meet Android studio, https://developer.android.com/studio/intro



Slika 1. Primjer izrade aplikacije u Android Studio-u

2.1.1. View

Sve aplikacije se sastoje od layout-a koji označava grupu view-ova i ima ulogu pozicionirati pojedine view-ove na zaslonu uređaja. „*View*“ je nešto što možemo vidjeti na korisničkom sučelju, odnosno na zaslonu uređaja. Neke od korištenih view-ova za aplikaciju su:

- **TextView** - element korisničkog sučelja koji korisniku prikazuje tekst
- **EditText** - element korisničkog sučelja koji služi za unos i izmjenu teksta
- **ImageView** - prikazuje resurse slika, na primjer resurse bitmape ili crteže, a također se koristi za primjenu nijansi na slikama i upravljanje skaliranjem slike
- **Button** - element korisničkog sučelja koji korisnik može dodirnuti ili kliknuti da izvrši radnju
- **RecyclerView** - olakšava učinkovito prikazivanje velikih skupova podataka. Omogućuje dostavljanje podataka i definiranje kako će pojedina stavka izgledati. Kao što naziv govori *RecyclerView* reciklira pojedinačne elemente. Kada se stavka pomakne sa zaslona *RecyclerView* ne uništava njezin prikaz, već ponovno koristi taj prikaz za nove stavke koje su se pomicale na zaslonu.

2.2. Java programski jezik

Programski jezik Java razvila je Sun Microsystems tvrtka početkom 1990-ih. Iako se prvenstveno koristi za internetske aplikacije Java je jednostavan i učinkovit jezik opće namjene. Java je izvorno dizajnirana za ugrađene mrežne aplikacije koje rade na više platformi. To je izuzetno prijenosni, objektno-orijentirani i interpretirani jezik.

Java aplikacija radit će identično na bilo kojem računalu, bez obzira na hardverske značajke ili operativni sustav, sve dok ima Java interpreter koda. Osim prenosivosti, još jedna od ključnih prednosti Jave je skup sigurnosnih značajki koje štite računalo na kojem se izvodi Java program, ne samo od problema uzrokovanih pogrešnim kodom, već i od zlonamjernih programa (Austerlitz, 2003).

2.3 IntelliJ IDEA

IntelliJ IDEA je inteligentan IDE(Integrated Development Environment), svjestan konteksta za rad s Javom i drugim JVM jezicima kao što su Kotlin, Scala i Groovy na svim vrstama aplikacija. Uz to, može pomoći u razvoju web aplikacija zahvaljujući moćnim integriranim alatima, podršci za JavaScript i povezanim tehnologijama te naprednoj podršci za popularne okvire kao što su Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus, Helidon. IntelliJ IDEA može se proširiti besplatnim dodacima koje je razvio JetBrains, omogućujući rad s drugim programskim jezicima uključujući Go, Python, SQL, Ruby i PHP. [2]

2.4. Operacijski sustav Android

Operativni sustav Android prvi je razvio Android Inc., softverska tvrtka sa sjedištem u Silicijskoj Dolini prije nego što ga je Google kupio 2005. Ulagači i analitičari elektroničke industrije doveli su u pitanje Googleove prave namjere za ulazak na mobilno tržište od te akvizicije. Ubrzo nakon toga Google je 2007. godine najavio predstavljanje svog prvog komercijalno dostupnog uređaja s Androidom, iako je taj proizvod stigao na tržište godinu kasnije.

[2] IntelliJ IDEA, https://www.jetbrains.com/idea/features/

Od tada, programeri softvera i aplikacija mogu koristiti Android tehnologiju za razvoj mobilnih aplikacija, koje se prodaju putem trgovina aplikacijama kao što je Google Play. Budući da je razvijen kao Googleov proizvod korisnici Androida imaju priliku povezati svoje mobilne uređaje s drugim Googleovim proizvodima, kao što su pohrana u oblaku, platforme za e-poštu i video usluge.

Izvorni kod za Android objavljen je u otvorenom formatu kako bi se unaprijedili standardi na mobilnim uređajima. Unatoč tome, Android je još uvijek zapakiran s vlasničkim softverom kada se prodaje na mobilnim uređajima. [3]

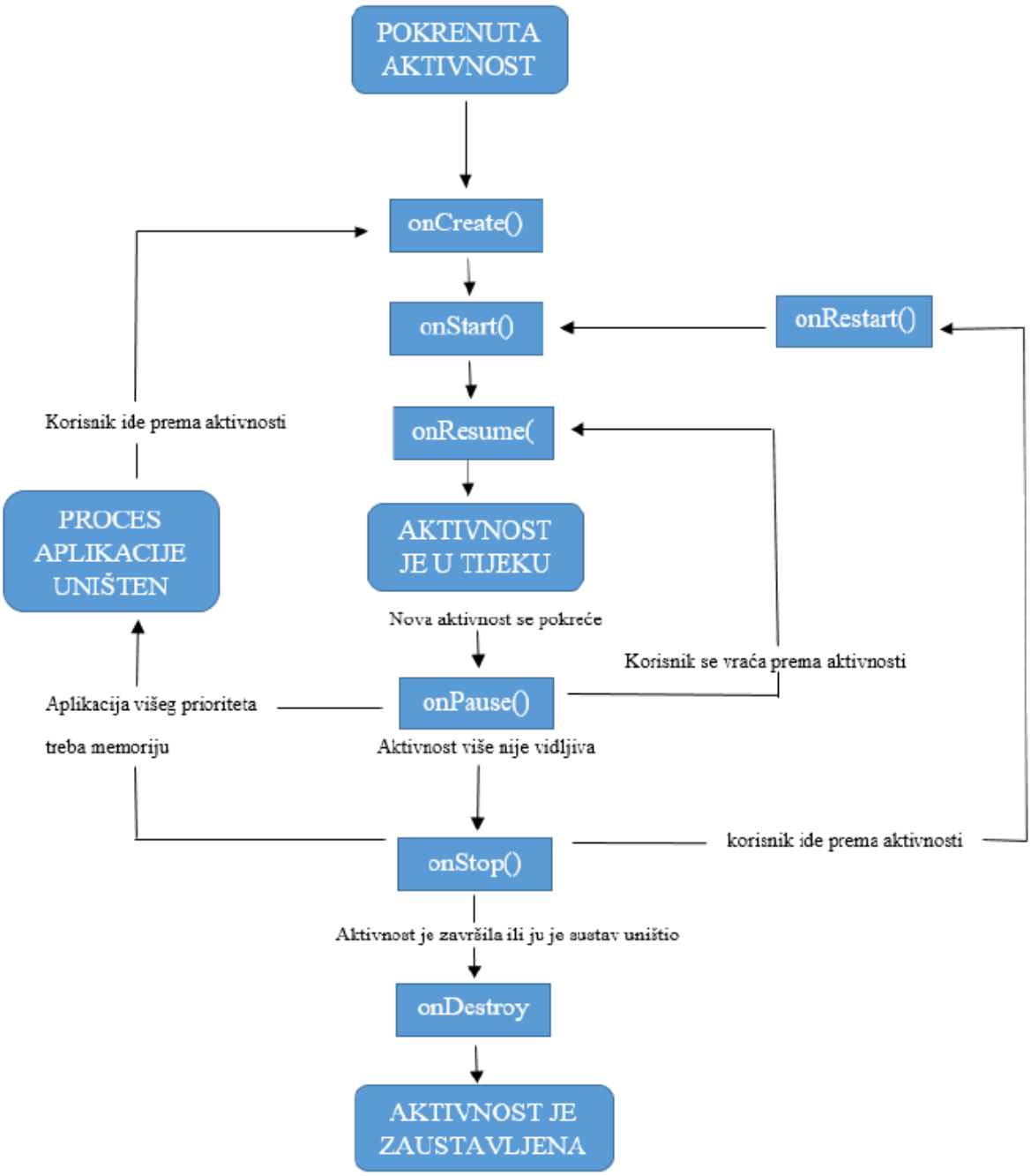
Svaka aplikacija ima životni ciklus koji čine metode [4]:

- OnStart()
- OnCreate()
- OnStop()
- OnDestroy()
- OnRestart()
- OnResume()
- OnPause()

Ove metode se pozivaju ovisno o stanju aplikacije. Na slici 2. je prikazan životni ciklus aplikacije.

[3]Android Operating System, Understanding the Android Operating System, https://www.investopedia.com/terms/a/android-operating-system.asp

[4] The Activity Lifecycle, Activity-lifecycle components https://developer.android.com/guide/components/activities/activity-lifecycle



Slika 2. Prikaz životnog ciklusa aplikacije

2.5 Firebase

Firebase je platforma za razvoj aplikacija koja pomaže u izradi i razvoju aplikacija.[5] Pruža razvojnim programerima razne alate i usluge koji im pomažu u razvoju kvalitetnih aplikacija, povećanju baze korisnika i ostvarivanju profita. Izgrađen je na Google-ovoj infrastrukturi, a kategoriziran kao NoSQL program baze podataka, koji pohranjuje podatke u dokumente slične JSON-u [6].

Neke od ključnih značajki Firebase-a [6]:

- **Autentičnost** - podržava autentifikaciju pomoću lozinki, telefonskih brojeva, Googlea, Facebooka, Twittera i više. Firebase Authentication (SDK) može se koristiti za ručnu integraciju jedne ili više metoda prijave u aplikaciju.
- **Baza podataka u stvarnom vremenu** - podaci se sinkroniziraju na svim klijentima u stvarnom vremenu i ostaju dostupni čak i kada je aplikacija izvan mreže.
- **Usluge poslužitelja** - Firebase usluge poslužitelja pružaju brze usluge za web aplikaciju. Sadržaj se sprema u pred-memoriju u mrežama za isporuku sadržaja širom svijeta.
- **Testni laboratorij** - aplikacija je testirana na virtualnim i fizičkim uređajima koji se nalaze u Googleovim podatkovnim centrima.
- **Notifikacije** - notifikacije se mogu slati s Firebase-om bez dodatnog kodiranja.[6]

[5] Firebase, https://firebase.google.com

[6] What is Firebase?, https://www.educative.io/answers/what-is-firebase

3. Programsko rješenje aplikacije za rezervacije auto servisa

3.1 Model aplikacije

Ova aplikacija traži korisnika da se registrira, odnosno unese osobne podatke kao što su ime, prezime, email, broj mobitela i lozinka. Kada unese potrebne podatke oni se spremaju u bazu podataka Firebase i račun je kreiran. Zatim se korisnik može prijaviti u aplikaciju te dobiva pristup mogućnostima aplikacije.

3.1.1 Registracija i kreiranje računa

Kada korisnik instalira aplikaciju prvo mora obaviti registraciju kako bi mogao nastaviti s daljnjim korištenjem. Kod registracije se traži od korisnika njegovo ime, prezime, email, broj mobitela i lozinka. Ako korisnik sve pravilno unese podaci se spremaju u Firebase bazu podataka te može nastaviti s prijavom u aplikaciju.

3.1.2. Prijava u aplikaciju

Svaki put kada korisnik pokrene aplikaciju mora se prijaviti pomoću email-a i lozinke koju je kreirao prilikom registracije u aplikaciju. Tokom prijave provjerava se je li ispravno upisao podatke (email i lozinku) i ako je sve točno upisano prosljeđuje ga se na sljedeći ekran aplikacije.

3.1.3. Sučelje korisnika

Kada se korisnik prijavi u svoj račun dobiva tri ekrana različitih funkcionalnosti. Na prvom ekranu se nalaze svi auto servisi koji su trenutno mogući za rezervacije te njihovi podaci u obliku kartica. Drugi ekran prikazuje lokacije dostupnih auto servisa unutar grada pomoću Google mapa. Treći ekran je korisnikov profil u kojemu može izmijeniti svoje korisničke podatke i napraviti proces odjave.

3.1.4. Sučelje auto servisera

Auto serviseri se ne moraju registrirati u sustav, oni su ručno uneseni u bazi podataka da se lakše raspoznaju od korisnika aplikacije. Nakon što auto serviser izvrši prijavu u aplikaciju dobiva različito sučelje od korisnika koje sadrži četiri ekrana različitih funkcionalnosti. U početku prva tri ekrana su prazna i koriste se za praćenje određenih servisa koje rezerviraju korisnici. Na prvom ekranu se nalaze moguće rezervacije od korisnika koje može prihvatiti. Na drugom ekranu se nalaze rezervacije koje je prihvatio te

koje su u tijeku. Na trećem ekranu se nalazi lista svih rezervacija koje je auto serviser obavio. Četvrti ekran je profil auto servisera koji također može mijenjati svoje podatke računa kao i kod korisnika.

3.2. Programsko rješenje po komponentama

Ovdje će se prikazati programsko rješenje redom po komponentama kako je nastala sama aplikacija. Prikazati će se i kodovi pojedinačno za svaku komponentu ove aplikacije.

3.2.1. Registracija i prijava korisnika

Prikazano je što će se dogoditi prvi puta kada korisnik otvori aplikaciju, što sve mora napraviti, odnosno zadovoljiti da bi mogao nastaviti koristiti aplikaciju. Korisnik prvo mora obaviti registraciju novog računa pomoću osobnih podataka kao što je ime, prezime, email, broj mobitela te sigurna lozinka. Pomoću funkcije *registerUser()* aplikacija provjerava jesu li su sva polja popunjena, tj. ne smiju biti prazna. Ako su sva polja popunjena i uvjeti ispunjeni, pritisak na tipku „Registracija“ će spremiti sve unesene informacije u bazu podataka. Kod za registraciju korisnika je prikazan slikom 3. i 4.

```
1.  private void registerUser(){
2.
3.          String ime = mIme.getText().toString();
4.          String prezime = mPrezime.getText().toString();
5.          String email = mEmail.getText().toString();
6.          String lozinka = mLozinka.getText().toString();
7.          String mobitel = mMobitel.getText().toString();
8.
9.          if (ime.isEmpty()){
10.             mIme.setError("Ime je obavezno!");
11.             return;
12.         }
13.
14.         if (prezime.isEmpty()){
15.             mPrezime.setError("Prezime je obavezno!");
16.             return;
17.         }
18.
19.         if (email.isEmpty()){
20.             mEmail.setError("Email je obavezan!");
21.             return;
22.         }
23.
24.         if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
25.             mEmail.setError("Unesite valjani email");
26.             return;
27.         }
28.
29.         if (mobitel.isEmpty()){
30.             mMobitel.setError("Mobitel je obavezan!");
31.             return;
32.         }
33.
34.         if (lozinka.isEmpty()){
35.             mLozinka.setError("Lozinka je obavezna!");
```

```
36.             return;
37.         }
38.
39.         if (lozinka.length() < 6){
40.             mLozinka.setError("Minimalno 6 znakova za lozinku");
41.             return;
42.         }
43.
44.         mRef.addValueEventListener(new ValueEventListener() {
45.             @Override
46.             public void onDataChange(@NonNull DataSnapshot snapshot) {
47.                 if (snapshot.exists()){
48.                     count = (snapshot.getChildrenCount());
49.                 }
50.             }
51.
52.             @Override
53.             public void onCancelled(@NonNull DatabaseError error) {
54.
55.             }
56.         });
```

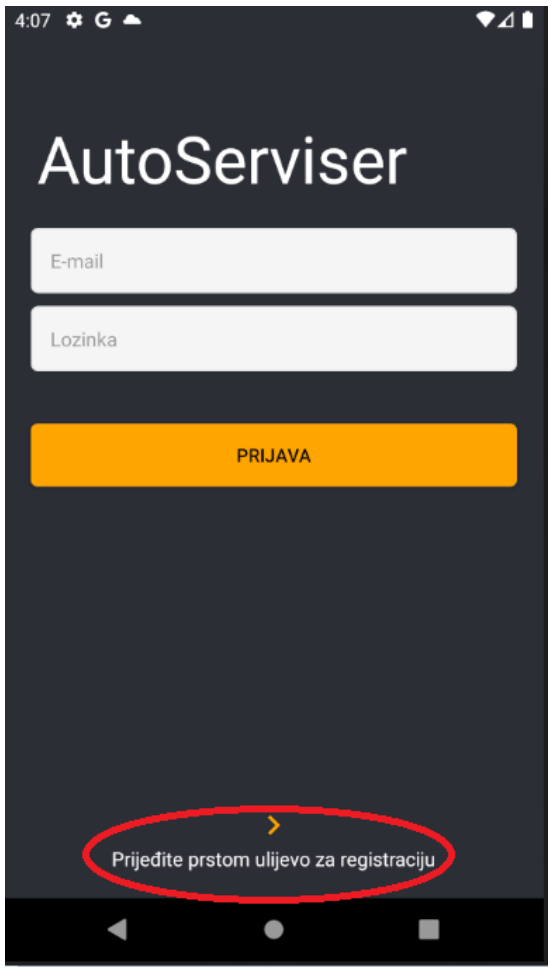
Slika 3. Kod funkcije registerUser()

Na slici 3. je prikazano kako aplikacija provjerava jesu li sva polja ispravno popunjena. Prvo se gleda da korisnik mora popuniti sve podatke tako da ne ostane ni jedno polje prazno. Sljedeće se provjerava da email adresa mora biti ispravna te lozinka mora sadržavati minimalno 6 znakova. Ukoliko korisnik unese sve ispravno ima mogućnost daljnjeg korištenja aplikacije, a u slučaju da krivo unese neku od informacija dobiva određenu poruku ovisno o grešci. Funkcija *registerUser()* će vratiti vrijednost *true* ako unese ispravno, odnosno ispisat će poruku u slučaju da unos nije ispravan. Kod registracije računa Firebase ima ugrađenu funkciju koja provjerava ako već postoji korisnik s unesenim email-om, tada korisnik dobiva poruku „*The email address is already in use by another account*“. Ukoliko ne postoji i svi podaci su ispravni račun korisnika se sprema u Firebase bazu podataka i dobiva poruku „*Registracija uspješna*“.

```
1.  auth.createUserWithEmailAndPassword(email, lozinka)
2.                  .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
3.                      @Override
4.                      public void onComplete(@NonNull Task<AuthResult> task) {
5.
6.                          if (task.isSuccessful()){
7.                              User oUser = new User(0, count + 1, "korisnik", ime,
    prezime, email, mobitel, lozinka);
8.
9.                              FirebaseDatabase.getInstance().getReference("Users")
10.
    .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
11.                                 .setValue(oUser).addOnCompleteListener(new
    OnCompleteListener<Void>() {
12.                                 @Override
13.                                 public void onComplete(@NonNull Task<Void> task) {
14.
15.                                     if (task.isSuccessful()){
16.                                         Toast.makeText(getContext(), "Registracija
    uspjesna!", Toast.LENGTH_SHORT).show();
17.                                         mIme.getText().clear();
18.                                         mPrezime.getText().clear();
19.                                         mEmail.getText().clear();
20.                                         mMobitel.getText().clear();
21.                                         mLozinka.getText().clear();
22.                                     }
23.                                     else{
24.                                         Toast.makeText(getContext(), "Registracija
    nije uspjela, pokusaj ponovno", Toast.LENGTH_SHORT).show();
25.                                     }
26.
27.                                 }
28.                             });
29.                         }
30.                         else{
31.                             Toast.makeText(getContext(),
    task.getException().getMessage(), Toast.LENGTH_SHORT).show();
32.                         }
33.
34.                     }
35.                 });
36.
37.     }
```

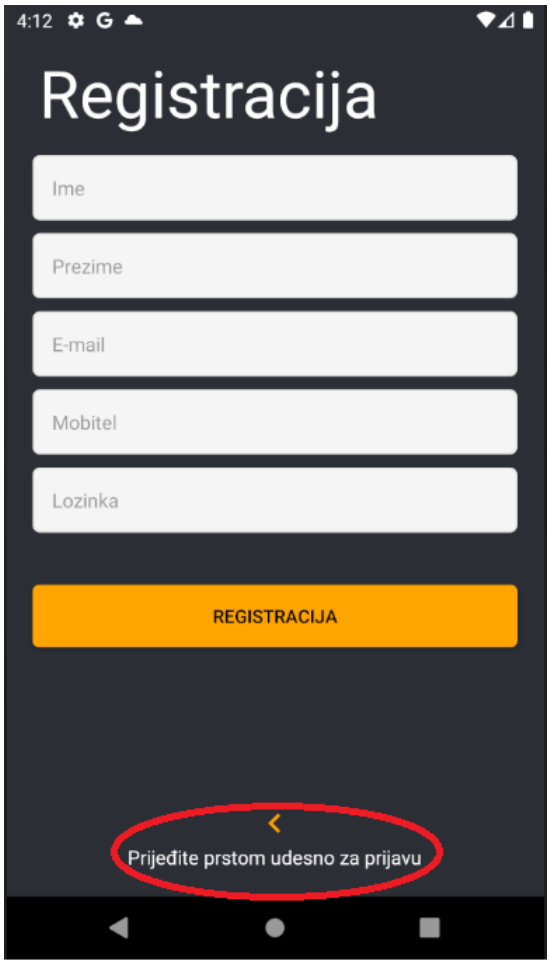
Slika 4. Kod za registraciju računa

Kada se prvi put pokrene aplikacija korisnik se mora registrirati da bi se njegov korisnički račun spremio u bazu podataka. Korisnik se može prelaskom prsta u lijevu stranu prebaciti na sučelje registracije kao što je prikazano na slici 5. i 6.



Slika 5. Proces registracije

Prelaskom prsta ulijevo označenog crvenim krugom na slici 5, aplikacija prikazuje sučelje za registraciju korisnika kao što se vidi na slici 6.



Slika 6. Registracija korisnika

Kod registracije moraju se unijeti svi podaci kao što je ime, prezime, email, broj mobitela te lozinka. Nakon pritiska tipke „Registracija“ podaci o korisničkom računu se spreme u bazi podataka ako je sve ispravno te se korisniku ispiše poruka da je prijava uspjela. Ako korisnik već ima postojeći korisnički račun može se vratiti na ekran prijave prelaskom prsta u desno kao što je označeno crvenim krugom na slici 6.

Kada korisnik uspije registrirati svoj račun može se prijaviti te nastaviti dalje s radom u aplikaciji. Svaki puta kada se otvori aplikacija korisnik se mora prijaviti ako već nije ostao prijavljen. Omogućena je i opcija za registraciju ako nema kreiran račun. Kod prijave u aplikaciju provjerava se jesu li popunjena polja za email i lozinku te odgovaraju li podacima koje je korisnik koristio tokom registracije računa. Ako je ispravno upisao podatke za prijavu, klikom na tipku „Prijava“ uspješno može nastaviti dalje koristiti aplikaciju. Kod za prijavu u aplikaciju prikazan je na slici 7. i 8.

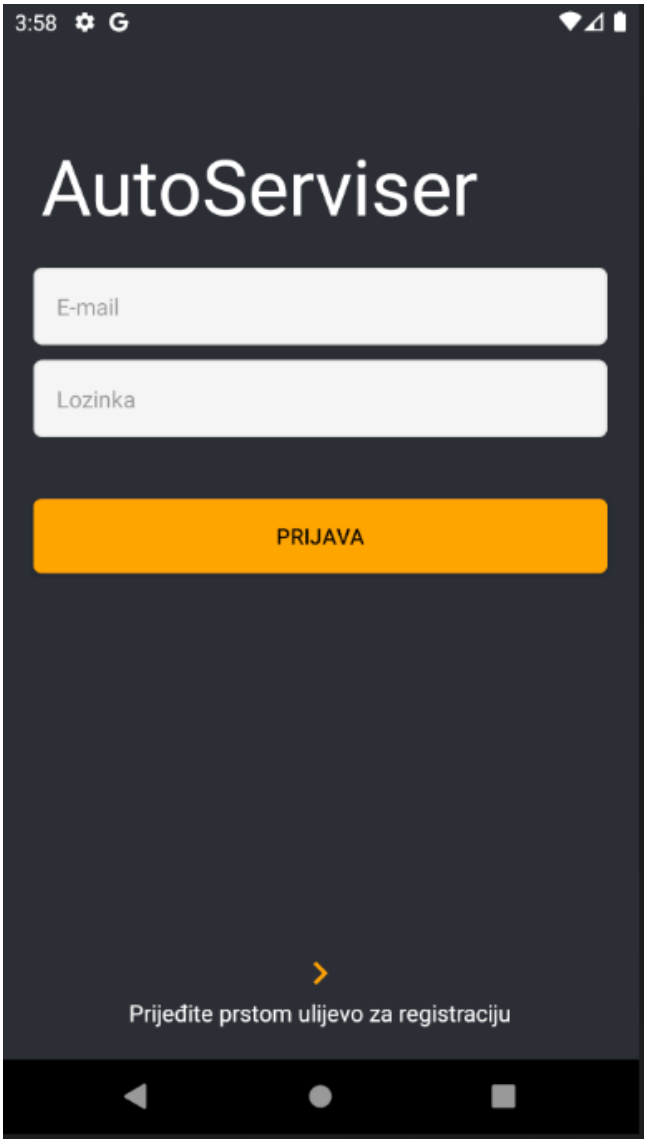
```
1.  private void userLogin(){
2.          String email = mEmail.getText().toString();
3.          String lozinka = mLozinka.getText().toString();
4.
5.          if (email.isEmpty()){
6.              mEmail.setError("Email popunite!");
7.              return;
8.          }
9.
10.         if (lozinka.isEmpty()){
11.             mLozinka.setError("Lozinku popunite");
12.             return;
13.         }
14.
15.
16.         auth.signInWithEmailAndPassword(email, lozinka)
17.                 .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
18.                     @Override
19.                     public void onComplete(@NonNull Task<AuthResult> task) {
20.
21.                         if (task.isSuccessful()){
22.
23.                             verifyUserType();
24.
25.                         }
26.                         else{
27.                             Toast.makeText(getContext(), "Neispravan email ili
    lozinka", Toast.LENGTH_SHORT).show();
28.                         }
29.
30.                     }
31.                 });
32.
33.     }
```

Slika 7. Kod funkcije userLogin()

```
1.  private void verifyUserType(){
2.
3.
    mDatabase.child(FirebaseAuth.getInstance().getUid()).addListenerForSingleValueEvent(ne
    w ValueEventListener() {
4.              @Override
5.              public void onDataChange(@NonNull DataSnapshot snapshot) {
6.                  String userType = snapshot.child("type").getValue(String.class);
7.
8.                  if (userType != null && userType.equals("korisnik")){
9.                      startActivity(new Intent(getContext(), HomeActivity.class));
10.                 }
11.                 else{
12.                     startActivity(new Intent(getContext(),
    HomeActivityServiser.class));
13.                 }
14.             }
15.
16.             @Override
17.             public void onCancelled(@NonNull DatabaseError error) {
18.
19.             }
20.         });
21.
22.     }
```

Slika 8. Kod funkcije verifyUserType() za provjeru računa

Nakon registracije korisnik se mora prijaviti. To može napraviti tako što će upisati svoj email i lozinku koje je odredio prilikom registracije. Zatim aplikacija će provjeriti ispravnost polja email-a i lozinke te ako je točno upisano korisnik ima mogućnost nastaviti koristiti aplikaciju. Prijava korisnika u aplikaciju prikazana je slikom 9.



Slika 9. Prijava korisnika

3.2.2. Rezerviranje auto servisa

Nakon što korisnik odabere rezervaciju za jedan od ponuđenih auto servisa aplikacija ga šalje na novi ekran gdje odabire tip servisa, datum koji mu odgovara te vrijeme servisa. Ako je netko drugi rezervirao isti dan u isto vrijeme, dobiva poruku sustava da je vrijeme zauzeto te mora odabrati drugi termin. Ako odabere izvan radnog vremena auto servisa također dobiva poruku da mora odabrati unutar radnog vremena. Kada je sve u redu upisano korisnik je poslan na sljedeći ekran potvrde rezervacije. Potvrda rezervacije sadrži unesene podatke iz prošlog ekrana te lokaciju na mapi kako bi lakše pronašli adresu gdje se auto servis nalazi. Kada korisnik potvrdi rezervaciju njene informacije se spreme u Firebase. Nakon toga mora čekati na odgovor ovlaštenog auto servisera. Kod za rezervaciju opisan je na slikama 10. i 11.

```
1.  public void rezervacijaOpel(){
2.          Intent intentRezervacija = new Intent(getContext(),
    RezervacijaActivity.class);
3.          Servis servis = new Servis();
4.
5.          for (int i = 0; i < servisi.size(); i++){
6.              if (servisi.get(i).ime.equals("Opel")){
7.
8.                  servis.OIB = servisi.get(i).OIB;
9.                  servis.adresa = servisi.get(i).adresa;
10.                 servis.email = servisi.get(i).email;
11.                 servis.ime = servisi.get(i).ime;
12.                 servis.servisID = servisi.get(i).servisID;
13.                 servis.telefon = servisi.get(i).telefon;
14.                 servis.radnoVrijeme = servisi.get(i).radnoVrijeme;
15.
16.                 intentRezervacija.putExtra("servis", servis);
17.
18.                 startActivity(intentRezervacija);
19.             }
20.         }
21.
22.     }
```

Slika 10. Primjer funkcije za rezervaciju auto servisa

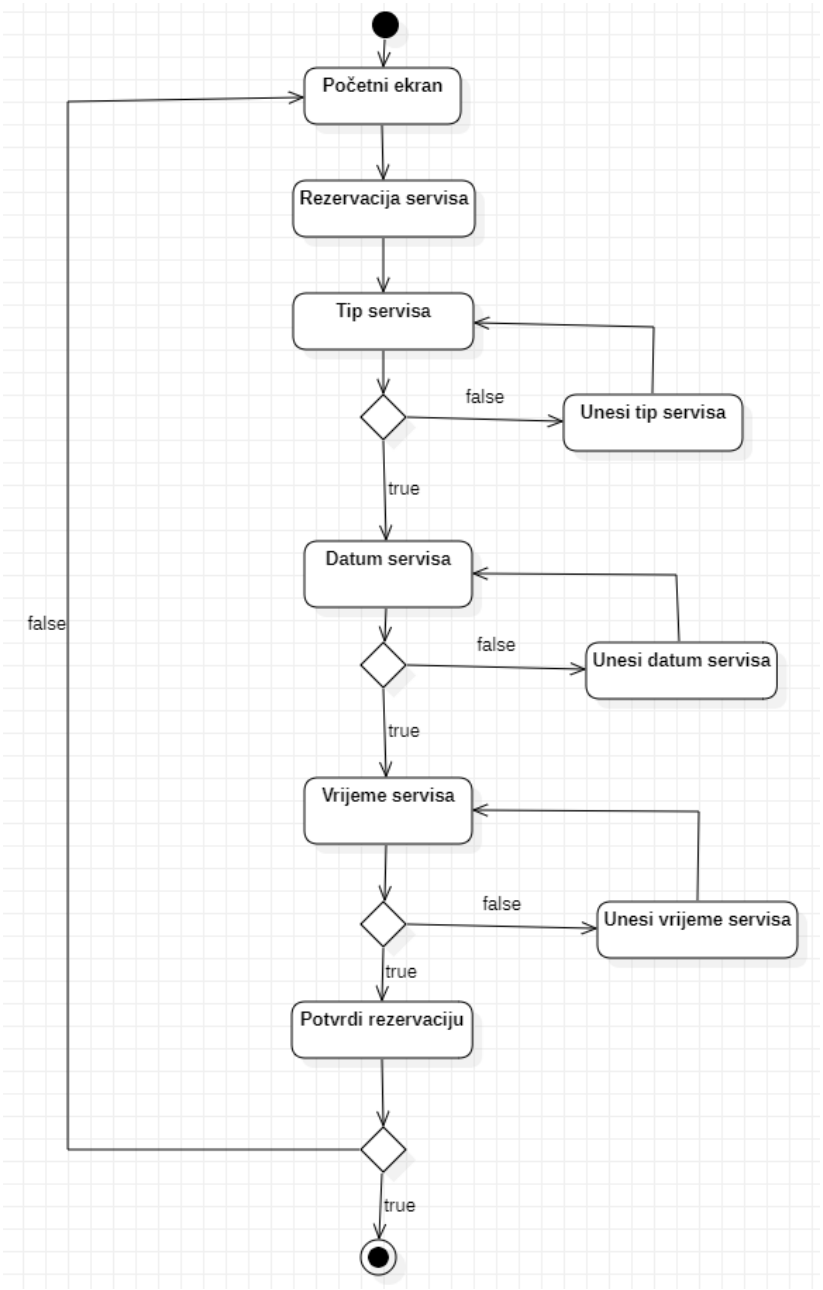
```
1.  mDatabaseUsers.child(userID).addListenerForSingleValueEvent(new ValueEventListener() {
2.                                      @Override
3.                                      public void onDataChange(@NonNull DataSnapshot
    snapshot) {
4.
5.                                          User userProfile =
    snapshot.getValue(User.class);
6.
7.                                          if (userProfile != null){
8.                                              rezervacijaID = userProfile.rezervacijaID;
9.
10.                                             Rezervacija rezervacija = new
    Rezervacija(oibServisa, userProfile.ime, userProfile.prezime, userProfile.mobitel,
    tip, datum, vrijeme, adresa, "Na cekanju", servisID, rezervacijaID);
11.
    mDatabaseRezervacije.push().setValue(rezervacija);
12.
    Toast.makeText(PotvrdaRezervacijeActivity.this, "Uspješno rezervirano",
    Toast.LENGTH_SHORT).show();
13.
14.                                             mDatabaseUsers.removeEventListener(this);
15.
16.                                             Intent i = new
    Intent(PotvrdaRezervacijeActivity.this, HomeActivity.class);
17.
    i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
18.                                             startActivity(i);
19.                                         }
20.                                     }
21.
22.                                     @Override
23.                                     public void onCancelled(@NonNull DatabaseError
    error) {
24.
25.                                     }
26.                                 });
```

Slika 11. Kod za potvrdu rezervacije auto servisa

```
1.  MultiFormatWriter writer = new MultiFormatWriter();
2.  BitMatrix matrix = writer.encode(
3.     "Podaci servisa: " +  "\n" +
4.
5.     "OIB servisa: " + dsRezervacije.child("oibServisa").getValue(Long.class) + "\n" +
6.
7.     "Adresa servisa: " + dsRezervacije.child("adresa").getValue(String.class) + "\n" +
8.
9.     "Tip servisa: " + dsRezervacije.child("tip").getValue(String.class) + "\n" +
10.
11.    "Datum servisa: " + dsRezervacije.child("datum").getValue(String.class) + "\n" +
12.
13.    "Vrijeme servisa: " + dsRezervacije.child("vrijeme").getValue(String.class) + "\n"
    +
14.
15.    "\nPodaci korisnika: \n" +
16.
17.    "Ime: " + dsRezervacije.child("ime").getValue(String.class) + "\n" +
18.
19.    "Prezime: " + dsRezervacije.child("prezime").getValue(String.class) + "\n" +
20.
21.    "Mobitel: " + dsRezervacije.child("mobitel").getValue(String.class)
    , BarcodeFormat.QR_CODE, 350, 350);
22.
23.   BarcodeEncoder encoder = new BarcodeEncoder();
24.
25.   Bitmap bitmap = encoder.createBitmap(matrix);
```

Slika 12. Kod za generiranje QR koda

Da bi mogli generirati QR kod kao što je prikazano na slici 9. moramo koristiti biblioteku „*Zxing*“ s kojom možemo koristiti razne funkcije. Kako bi mogli pohraniti sve podatke koje želimo da se nalaze unutar QR koda moramo ih zapisati unutar varijable „*matrix*“ te ju kodirati pomoću *.encode()* funkcije. Nakon toga možemo varijablu spremiti unutar „*Bitmap*“ klase kako bi prikazali kreirani QR kod.

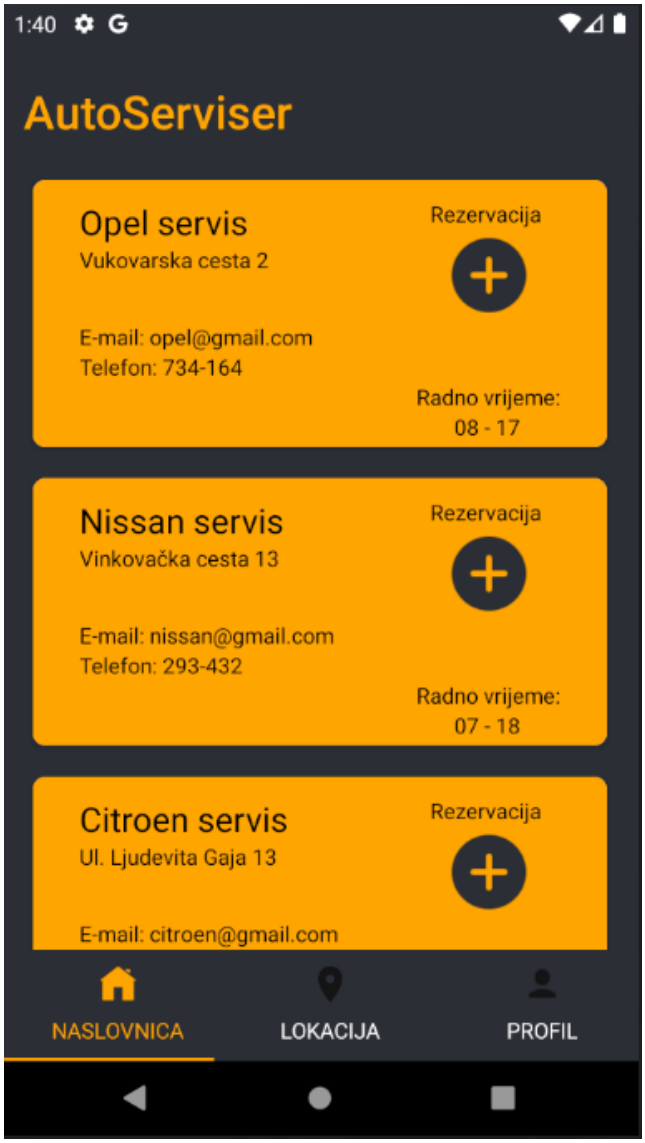


Slika 13. UML dijagram aktivnosti rezervacije servisa

Na slici 13. se nalazi dijagram aktivnosti kreiranja jedne rezervacije. Da bi rezervirali servis prvo se mora kliknuti na tipku koja to omogućava nakon čega se unosi koji tip servisa za automobil je potreban. Ako se tip servisa ne unese korisnika se vraća na odabir zbog toga što je to polje obavezno. Kada unese ispravan podatak dalje može izabrati datum servisa te vrijeme koji mu odgovaraju. Također mora odabrati ispravan datum i vrijeme jer su to obavezna polja da bi mogao nastaviti s rezervacijom. Ako je sve ispravno šalje se korisnika na potvrdu rezervacije. Ako korisnik potvrdi rezervaciju tada je obavio cijeli proces

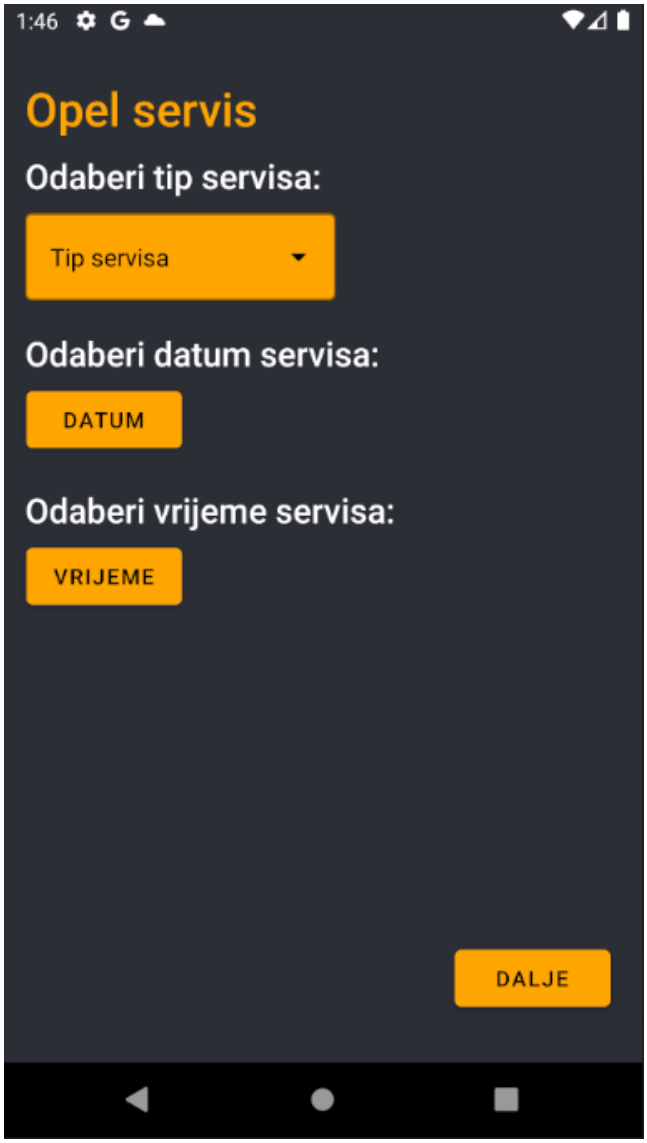
rezerviranja servisa, a u slučaju da odustane ili promjeni mišljenje vraća ga se nazad na početni ekran.

Kada se korisnik prijavi u aplikaciju na početnom ekranu ga dočeka sučelje s karticama koje sadržavaju podatke o pojedinom auto servisu. Na slici 14. je prikazano sučelje početnog ekrana korisnika.



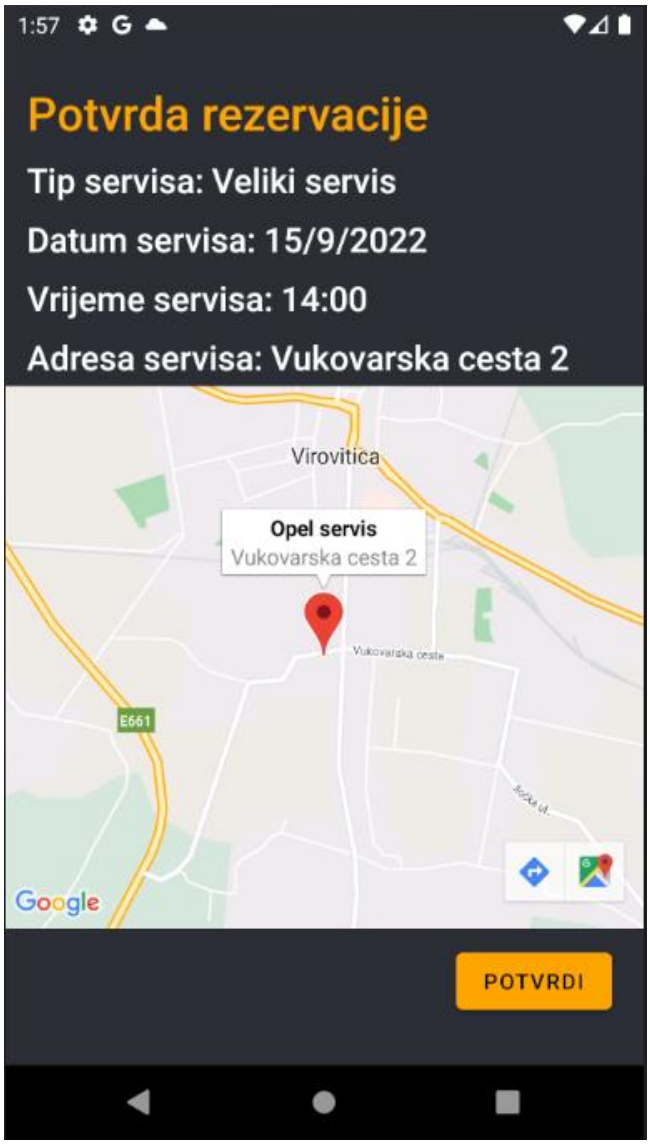
Slika 14. Početni ekran korisnika

Kada korisnik hoće napraviti servis na svome autu, mora kliknuti na tipku za rezervaciju nakon čega se otvara sljedeći ekran na kojemu odabire tip servisa, datum i vrijeme servisa koje mu odgovara. Na slici 15. je prikazano sučelje kreiranja rezervacije.



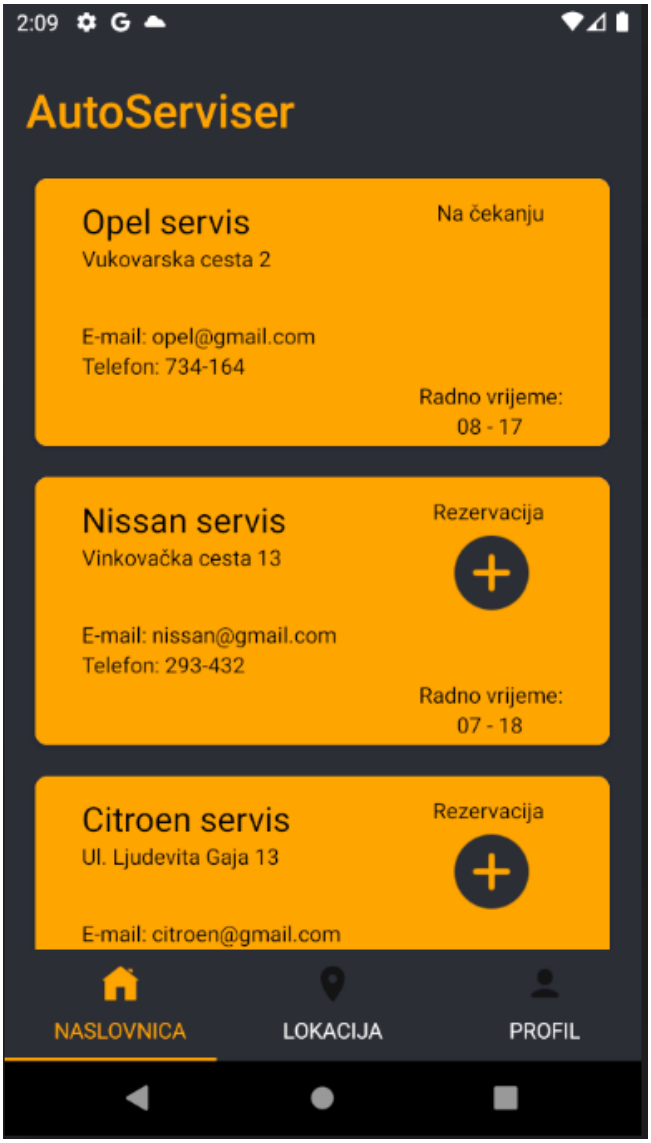
Slika 15. Kreiranje rezervacije

Nakon što korisnik odabere i popuni sve podatke može kliknuti na tipku „Dalje“ te se na taj način prosljeđuje na ekran za potvrdu rezervacije. Ovdje se nalaze svi uneseni podaci iz prethodnog ekrana i karta koja sadrži lokaciju gdje se nalazi auto servis. Na slici 16. je prikazano sučelje za potvrdu rezervacije.



Slika 16. Potvrda rezervacije

Nakon što korisnik potvrdi rezervaciju vraća ga se na početni ekran gdje može vidjeti da mu je rezervacija na čekanju. Na slici 17. je prikazano sučelje nakon potvrde rezervacije.



Slika 17. Početni ekran nakon kreiranja rezervacije

Kada auto serviser prihvati rezervaciju na svome ekranu korisniku se pojavi QR kod koji se može skenirati da bi se vidjeli podaci vezani za servis koji je rezervirao. Početni ekran nakon prihvaćene rezervacije te ekran QR koda prikazan je na slici 18. i 19.

Slika 18. Početni ekran nakon što se rezervacija prihvati

Nakon što se klikne na tipku QR koda otvara se novi prozor koji se može skenirati da bi se prikazali podaci o servisu kao što su OIB servisa, adresa, tip servisa, datum i vrijeme te podaci korisnika koji je rezervirao servis. QR kod služi kao potvrda koju mora pokazati auto serviseru da bi mogao obaviti servis.

Slika 19. QR kod s podacima rezervacije

Nakon što korisnik obavi servis, auto serviser stavlja rezervaciju u listu obavljenih servisa i korisniku se prikaže notifikacija da je servis na autu završen. Korisniku se vrati početni ekran kao što je bilo prikazano slikom 14.

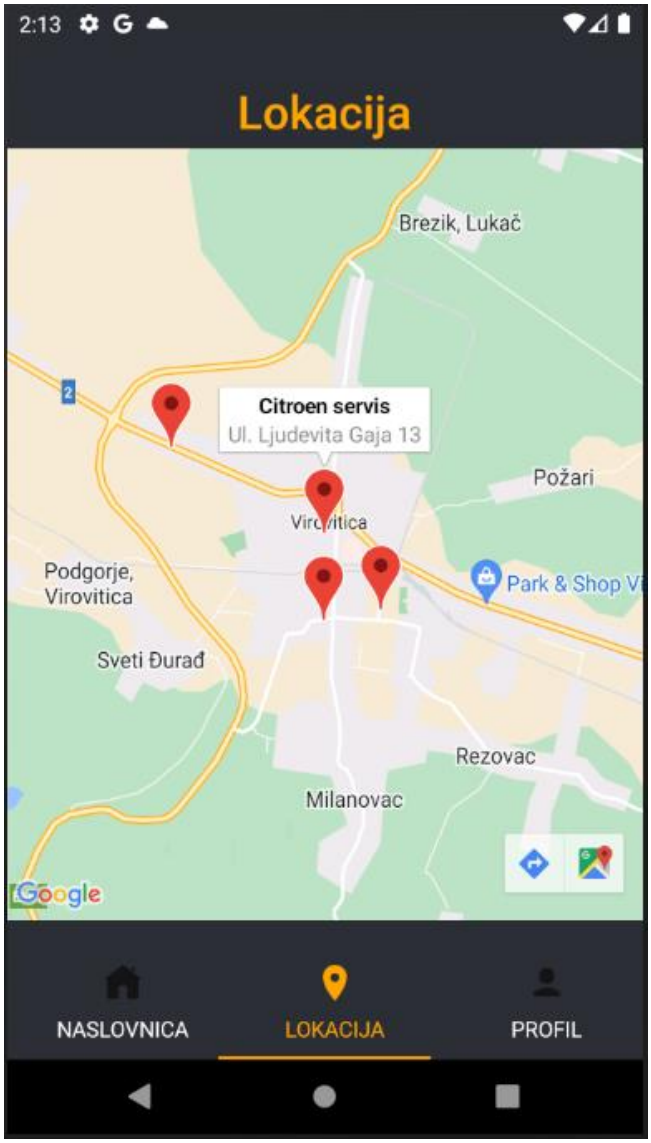
3.2.3. Lokacije mogućih auto servisa

Na ovom ekranu se nalaze svi mogući auto servisi u kojima korisnik može rezervirati servis za vlastiti automobil. Lokacije su označene na mapi pomoću markera radi lakšeg snalaženja. Kod na slici 20. prikazuje implementaciju Google Maps API-ja.

```
1.  @Override
2.          public void onMapReady(GoogleMap googleMap) {
3.              LatLng opel = new LatLng(45.8206169, 17.3845614);
4.              googleMap.addMarker(new MarkerOptions().position(opel).title("Opel
    servis").snippet("Vukovarska cesta 2"));
5.
6.              LatLng nissan = new LatLng(45.8218038, 17.3934923);
7.              googleMap.addMarker(new MarkerOptions().position(nissan).title("Nissan
    servis").snippet("Vinkovačka cesta 13"));
8.
9.              LatLng citroen = new LatLng(45.830199, 17.3846591);
10.             googleMap.addMarker(new MarkerOptions().position(citroen).title("Citroen
    servis").snippet("Ul. Ljudevita Gaja 13"));
11.
12.             LatLng renault = new LatLng(45.8395842, 17.3606061);
13.             googleMap.addMarker(new MarkerOptions().position(renault).title("Renault
    servis").snippet("Ul. Josipa Jurja Strossmayera 188"));
14.
15.             googleMap.getUiSettings().setScrollGesturesEnabled(false);
16.             googleMap.moveCamera(CameraUpdateFactory.newLatLng(citroen));
17.
18.             googleMap.setMinZoomPreference(12.5f);
19.             googleMap.setMaxZoomPreference(14.0f);
20.         }
21.     };
22.
23.     @Nullable
24.     @Override
25.     public View onCreateView(@NonNull LayoutInflater inflater,
26.                              @Nullable ViewGroup container,
27.                              @Nullable Bundle savedInstanceState) {
28.         return inflater.inflate(R.layout.fragment_location, container, false);
29.     }
30.
31.     @Override
32.     public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState)
    {
33.         super.onViewCreated(view, savedInstanceState);
34.         SupportMapFragment mapFragment =
35.                 (SupportMapFragment)
    getChildFragmentManager().findFragmentById(R.id.map);
36.         if (mapFragment != null) {
37.             mapFragment.getMapAsync(callback);
38.         }
39.     }
```

Slika 20. Kod za prikaz mogućih lokacija servisa

Na ovom sučelju se nalazi mapa sa svim auto servisima u kojima se može rezervirati servis. Lokacije su prikazane markerima da se lakše pronađe gdje se koji auto servis nalazi. Kada se klikne na pojedini marker pojavi se ime servisa te njegova adresa. Na slici 21. prikazano je sučelje za prikaz lokacija mogućih auto servisera.



Slika 21. Lokacije auto servisa

3.2.4. Pregled primljenih rezervacija

Kada se auto serviser prijavi u aplikaciju prvo ga dočeka ekran na kojemu vidi sve primljene rezervacije od korisnika. Rezervacije se prikazuju u obliku kartica i sadrže sve podatke kao što su korisnički podaci te podaci o rezervaciji. Može potvrditi rezervaciju te ju na taj način poslati na sljedeći ekran gdje se nalaze sve rezervacije koje su u tijeku. Nakon što potvrdi rezervaciju korisnik na svome ekranu vidi na kartici auto servisa da je rezervacija prihvaćena tj. u tijeku te mu stigne notifikacija kako bi lakše pratio status servisa. Kod za pregled rezervacija je prikazan na slici 22. Klasa koja služi za slanje notifikacija i koja koristi Firebase Cloud Messaging je opisana na slici 23.

```
1.  for (DataSnapshot dsRezervacije : snapshotRezervacije.getChildren()){
2.
3.                                          if (userProfile.servisID ==
    dsRezervacije.child("servisID").getValue(Long.class) &&
    dsRezervacije.child("status").getValue(String.class).equals("Na cekanju")){
4.
5.                                              Rezervacija rezervacija =
    dsRezervacije.getValue(Rezervacija.class);
6.
7.                                              rezervacije.add(rezervacija);
8.
9.                                              //RV Rezervacije
10.                                             rezervacijeAdapter = new
    RezervacijeRVAdapter(rezervacije, tvEmpty, getContext());
11.
12.                                             mRecyclerView.setLayoutManager(new
    LinearLayoutManager(getContext()));
13.
    mRecyclerView.setAdapter(rezervacijeAdapter);
14.
15.                                         }
16.
17.                                     }
18.
19.                                     if (rezervacijeAdapter.getItemCount() == 0){
20.                                         tvEmpty.setText("Trenutno nema rezervacija
    na čekanju");
21.                                     }
```

Slika 22. Kod za pregled i prihvaćanje rezervacija

```
1.  public class FCMSend {
2.      private static String BASE_URL = "https://fcm.googleapis.com/fcm/send";
3.      private static String SERVER_KEY = "key=AAAAIr-
    KTfk:APA91bEsyAs4uJNu7g6ziAXwoZd9TeiFmTh7H-
    Kew5QhvWN9XSXZobaCS_52fyWUtlJpB_pclGzJtFeGYDl6YU2UMUIlXftgMBwdKcgIz4-
    UvUyF9vejiOs4G2Is2ZdZj2k6YN4Sjhzp";
4.
5.      public static void pushNotification(Context context, String token, String title,
    String message) {
6.          StrictMode.ThreadPolicy policy = new
    StrictMode.ThreadPolicy.Builder().permitAll().build();
7.          StrictMode.setThreadPolicy(policy);
8.
9.          RequestQueue queue = Volley.newRequestQueue(context);
10.
11.         try {
12.             JSONObject json = new JSONObject();
13.             json.put("to", token);
14.             JSONObject notification = new JSONObject();
15.             notification.put("title", title);
16.             notification.put("body", message);
17.             json.put("notification", notification);
18.
19.             JsonObjectRequest jsonObjectRequest = new
    JsonObjectRequest(Request.Method.POST, BASE_URL, json, new
    Response.Listener<JSONObject>() {
20.                 @Override                 public void onResponse(JSONObject response)
    {
21.                     System.out.println("FCM " + response);
22.                 }
23.             }, new Response.ErrorListener() {
24.                 @Override                 public void onErrorResponse(VolleyError
    error) {
25.
26.                 }
27.             }){
28.                 @Override                 public Map<String, String> getHeaders()
    throws AuthFailureError {
29.                     Map<String, String> params = new HashMap<>();
30.                     params.put("Content-Type", "application/json");
31.                     params.put ("Authorization", SERVER_KEY);
32.                     return params;
33.                 }
34.             };
35.
36.             queue.add(jsonObjectRequest);
37.         } catch (JSONException e) {
38.             e.printStackTrace();
39.         }
40.     }
41. }
```

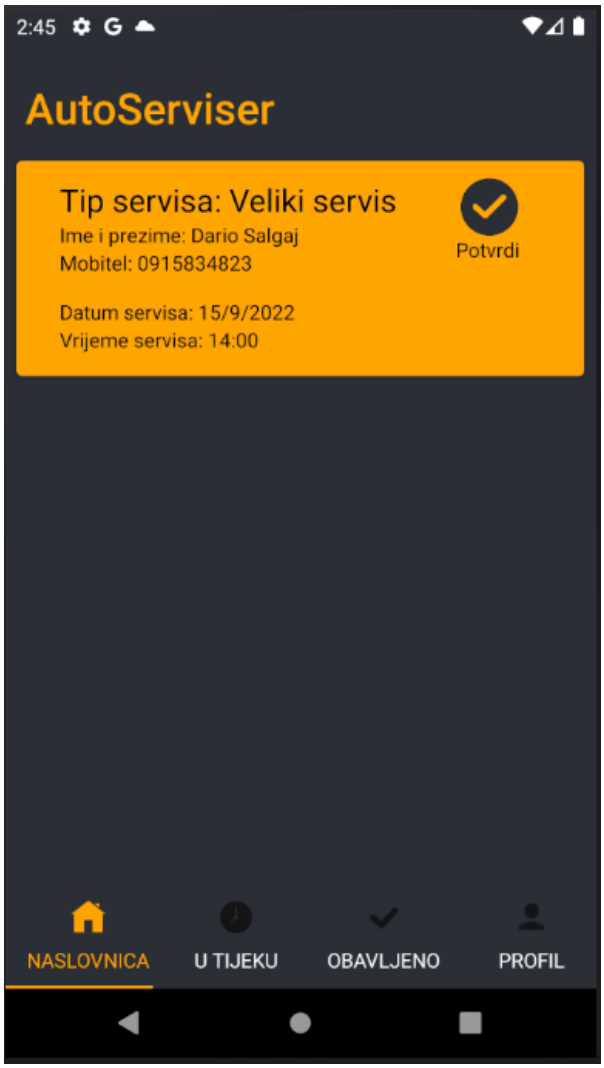
Slika 23. Klasa za slanje notifikacija putem FCM

Kako bi se notifikacija poslala korisniku koristi se klasa „*FCMSend*“ te funkcija *.pushNotification().* Da bi slanje notifikacije radilo potreban je ključ uređaja kojemu se šalje notifikacija te se može definirati naslov notifikacije i poruka. Na slici 24. prikazan je kod slanja notifikacije korisniku.

```
1.  String title = "Auto Servis";
2.  String message = "Vaš servis je prihvaćen";
3.  FCMSend.pushNotification(
4.          context,
5.          "eNEwvCzyQ2a1Fh1b1SiS-k:APA91bGPgl9wufItlnTux7b59acKFN31ZbyoEL7x_ujYyVstIW-
    _ehVWz_r1xelzo3v6psOdlZWV8xuguzsFQMwdgwcmppvjbIW-
    3HV_e6feYh_XRJOlDIOU86YGUyqHi5rxTfwd5BYu",
6.          title,
7.          message
8.  );
```

Slika 24. Kod za slanje notifikacije kada je servis prihvaćen

Auto serviser kada se prvi put prijavi u aplikaciju dočeka ga prazan početni ekran gdje vidi sve servise koje korisnici rezerviraju. Kada korisnik rezervira servis na ovom ekranu se pojavi kartica s podacima korisnika i servisa. Auto serviser može prihvatiti rezervaciju te na taj način ju pošalje na sljedeći ekran. Sučelje početnog ekrana auto servisera kada se prvi put prijavi te nakon što zaprimi rezervaciju prikazano je na slici 25.



Slika 25. Početni ekran nakon što zaprimi rezervaciju

Nakon što auto serviser potvrdi rezervaciju ona se šalje na ekran „U tijeku“ te na taj način se obavještava korisnika da je njegova rezervacija prihvaćena i da može doći na servis u terminu koji je zadao.

3.2.5. Pregled rezervacija koje su u tijeku

Ovdje auto serviser ima pregled svih rezervacija koje je prihvatio s prethodnog ekrana. Ima mogućnost da označi rezervaciju kao obavljenu te ju se šalje na zadnji ekran koji sadrži listu svih obavljenih servisa. Nakon što se rezervacija pojavi na ovom ekranu korisnik na svojoj kartici auto servisa dobiva QR kod koji sadrži podatke te rezervacije kako bi mogao pokazati auto serviseru kao potvrdu. Također, korisnik prima notifikaciju kako bi znao da je servis obavljen. Kod za pregled rezervacija u tijeku je prikazan na slici 26.

```
1.  for (DataSnapshot dsRezervacije : snapshotRezervacije.getChildren()){
2.
3.                                          if (userProfile.servisID ==
    dsRezervacije.child("servisID").getValue(Long.class) &&
    dsRezervacije.child("status").getValue(String.class).equals("U tijeku")){
4.
5.                                              Rezervacija rezervacija =
    dsRezervacije.getValue(Rezervacija.class);
6.
7.                                              rezervacije.add(rezervacija);
8.
9.                                              //RV RezervacijeCekanje
10.                                             rezervacijeAdapter = new
    CekanjeRVAdapter(rezervacije, tvEmptyCekanje, getContext());
11.
12.                                             mRecyclerView.setLayoutManager(new
    LinearLayoutManager(getContext()));
13.
    mRecyclerView.setAdapter(rezervacijeAdapter);
14.
15.                                         }
16.
17.                                     }
18.
19.                                     if (rezervacijeAdapter.getItemCount() == 0){
20.                                         tvEmptyCekanje.setText("Trenutno nema
    rezervacija u tijeku");
21.                                     }
22.                                     else{
23.
    tvEmptyCekanje.setVisibility(View.INVISIBLE);
24.                                     }
```

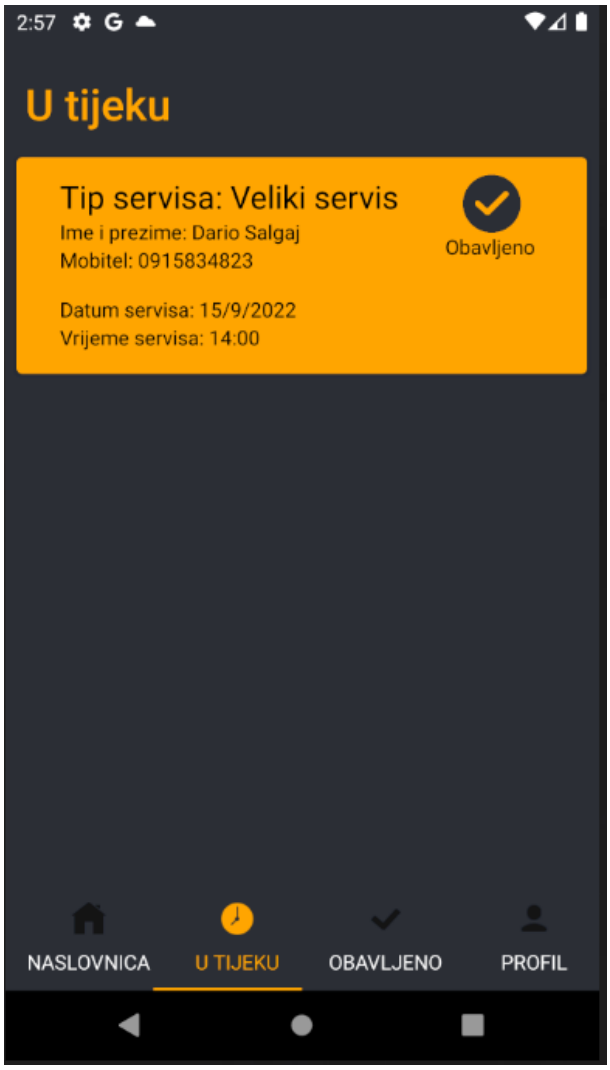
Slika 26. Kod za pregled rezervacija u tijeku

Na slici 27. prikazano je kako se šalje notifikacija korisniku kada je servis obavljen. Kod je identičan kao kod slanja notifikacije u prethodnom primjeru.

```
1.  String title = "Auto Servis";
2.  String message = "Vaš servis je obavljen";
3.  FCMSend.pushNotification(
4.          context,
5.          "eNEwvCzyQ2a1Fh1b1SiS-k:APA91bGPgl9wufItlnTux7b59acKFN31ZbyoEL7x_ujYyVstIW-
    _ehVWz_r1xelzo3v6psOdlZWV8xuguzsFQMwdgwcmppvjbIW-
    3HV_e6feYh_XRJOlDIOU86YGUyqHi5rxTfwd5BYu",
6.          title,
7.          message);
```

Slika 27. Kod za slanje notifikacije kada je servis obavljen

Kada auto serviser prihvati rezervaciju na ovom ekranu se ta rezervacija pojavi kako bi auto serviser znao koje su rezervacije u tijeku. Kao i na prethodnom ekranu u slučaju da je prazno prikazuje se poruka da trenutno nema rezervacija u tijeku. Nakon što auto serviser potvrdi rezervaciju ona se također pojavi u obliku kartice s podacima korisnika i servisa, ali ovaj puta se nalazi tipka koja šalje rezervaciju na ekran „Obavljeno“. Na slici 28. prikazano je sučelje rezervacija koje su u tijeku.



Slika 28. Rezervacije koje su u tijeku

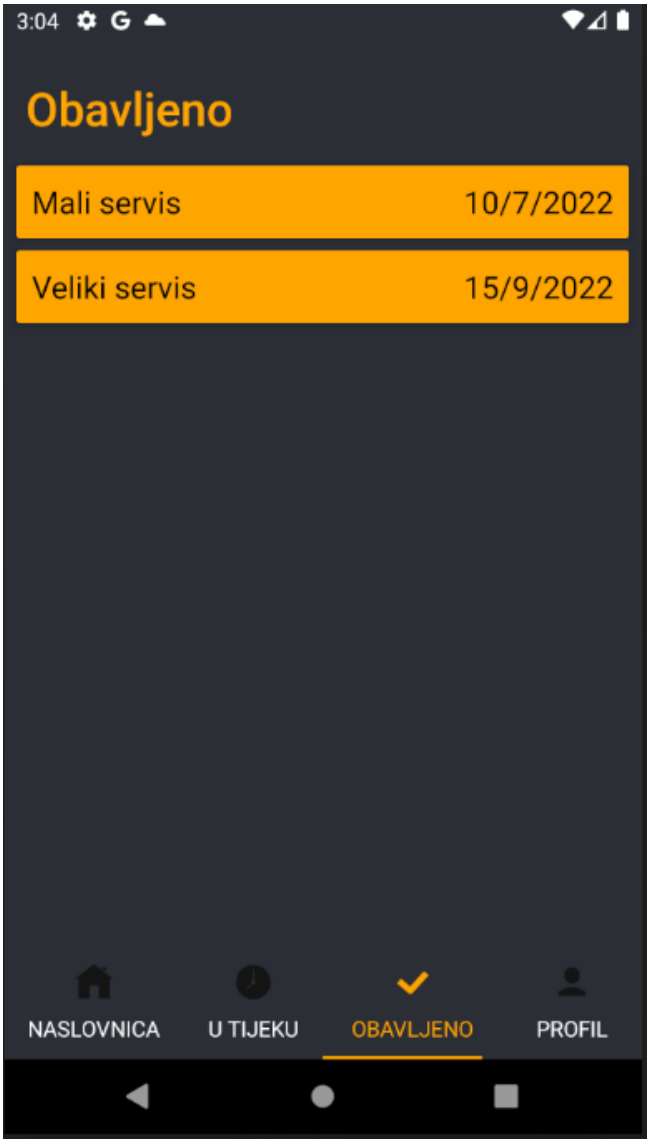
3.2.6. Pregled svih obavljenih servisa

Treći ekran prikazuje sve obavljene servise koje auto serviser pošalje iz prethodnog ekrana u obliku liste koja se može povećati za pojedini element da bi se prikazali detaljnije podaci o pojedinom servisu. Kod za prikaz obavljenih servisa je prikazan na slici 29.

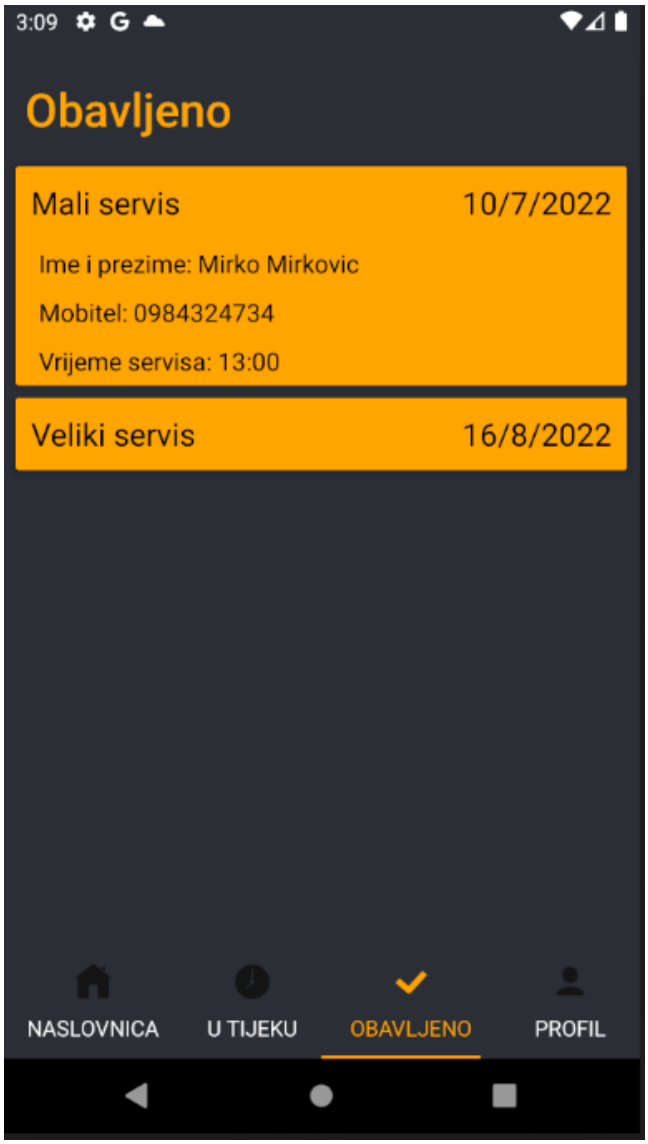
```
1.  for (DataSnapshot dsRezervacije : snapshotRezervacije.getChildren()){
2.
3.                                          if (userProfile.servisID ==
    dsRezervacije.child("servisID").getValue(Long.class) &&
    (dsRezervacije.child("status").getValue(String.class).equals("ObavljenoN") ||
    dsRezervacije.child("status").getValue(String.class).equals("Obavljeno"))){
4.
5.                                              Rezervacija rezervacija =
    dsRezervacije.getValue(Rezervacija.class);
6.
7.                                              rezervacije.add(rezervacija);
8.
9.                                              //RV RezervacijeCekanje
10.                                             rezervacijeAdapter = new
    ObavljenoRVAdapter(rezervacije, getContext());
11.
12.                                             mRecyclerView.setLayoutManager(new
    LinearLayoutManager(getContext()));
13.
    mRecyclerView.setAdapter(rezervacijeAdapter);
14.
15.                                         }
16.
17.                                     }
18.
19.                                     if (rezervacijeAdapter.getItemCount() == 0){
20.                                         tvEmptyObavljeno.setText("Trenutno nema
    obavljenih servisa");
21.                                     }
22.                                     else{
23.
    tvEmptyObavljeno.setVisibility(View.INVISIBLE);
24.                                     }
```

Slika 29. Kod za prikaz obavljenih servisa u obliku liste

Kada auto serviser obavi određene servise ima mogućnost poslati rezervacije na ekran „Obavljeno“ u kojemu se nalaze svi servisi koje je auto serviser do sada obavio. Prikazani su u obliku liste kako bi ih se moglo brže pretražiti. Liste prikazuju tip servisa te datum kada je on obavljen, ali ako se klikne na njih mogu se povećati za sve bitne podatke servisa i korisnika. Sučelje služi kao arhiva za sve obavljene servise te je prikazano slikama 30. i 31.



Slika 30. Lista obavljenih rezervacija auto servisa



Slika 31. Proširenі element liste obavljenih servisa

4. Zaključak

Svakodnevni život ljudi postao je nezamisliv bez pametnih uređaja. U svim djelatnostima je njihova primjena olakšala i poboljšala kvalitetu života. Kako bi pojednostavili način komunikacije između korisnika i auto servisera, napravljena je aplikacija koja će to omogućiti. Budući da veliki broj korisnika u svijetu i kod nas koristi mobilne uređaje s Android operacijskim sustavom ideja je bila napraviti aplikaciju baš za taj operacijski sustav. Aplikacija je jednostavna za korištenje, izrađena je u Android Studio-u i koristi Firebase bazu podataka za pohranu korisnika, servisa te rezervacija. Za razvoj ove aplikacije bilo je nužno znanje osnovnih koncepata objektno-orijentiranog programiranja te programskog jezika Jave. Aplikacija se može pokrenuti samo na uređajima koji podržavaju Android operativni sustav. U budućnosti je potrebno pratiti potrebe korisnika i unaprijediti rad aplikacije. Moguća su proširenja baze podataka i poboljšanje vizualnog izgleda korisničkog sučelja.

Literatura

Knjige:

1. H. Austerlitz, *Data Acquisition Techniques Using PCs (Second Edition),* Sjedinjene Američke Države: Elsevier Science, 2003.

Internetski izvori:

1. Meet Android Studio, https://developer.android.com/studio/intro (22.08.2022)
2. IntelliJ IDEA, https://www.jetbrains.com/idea/features/ (23.08.2022)
3. Android Operating System, https://www.investopedia.com/terms/a/android-operating-system.asp (24.08.2022)
4. The Activity Lifecycle, https://developer.android.com/guide/components/activities/activity-lifecycle (24.08.2022)
5. What is Firebase?, https://www.educative.io/answers/what-is-firebase (25.08.2022)
6. Firebase, https://firebase.google.com (25.08.2022)

Materijali s predavanja:

Materijali s predavanja predmeta Programiranje mobilnih aplikacija, Veleučilište u Virovitici

Materijali s predavanja predmeta Objektno orijentirano programiranje, Veleučilište u Virovitici

POPIS ILUSTRACIJA

Veleučilište u Virovitici

OBRAZAC 5

IZJAVA O AUTORSTVU

Ja, Dario Šalgaj

izjavljujem da sam autor/ica završnog/diplomskog rada pod nazivom

Razvoj aplikacija za Android operacijski sustav

Svojim vlastoručnim potpisom jamčim sljedeće:

- da je predani završni/diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija,
- da su radovi i mišljenja drugih autora/ica, koje sam u svom radu koristio/la, jasno navedeni i označeni u tekstu te u popisu literature,
- da sam u radu poštivao/la pravila znanstvenog i akademskog rada.

Potpis studenta/ice

Dario Šalgaj

Veleučilište u Virovitici

OBRAZAC 6

ODOBRENJE ZA POHRANU I OBJAVU ZAVRŠNOG/DIPLOMSKOG RADA

Ja Dario Šalgaj

dajem odobrenje za objavljivanje mog autorskog završnog/diplomskog rada u javno dostupnom digitalnom repozitoriju Veleučilišta u Virovitici te u javnoj internetskoj bazi završnih radova Nacionalne i sveučilišne knjižnice bez vremenskog ograničenja i novčane nadoknade, a u skladu s odredbama članka 83. stavka 11. Zakona o znanstvenoj djelatnosti i visokom obrazovanju (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15, 131/17).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog završnog/diplomskog rada. Ovom izjavom, kao autor navedenog rada dajem odobrenje i da se moj rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

(a) široj javnosti
b) studentima i djelatnicima ustanove
c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

Potpis studenta/ice

Dario Šalgaj

U Virovitici, 31.08.2022.

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem završnom/diplomskom radu, podnosi se pisani obrazloženi zahtjev.*