APLIKACIJA ZA DIJELJENJE OSOBNOG VOZILA

Prügelhof, Ivan

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

APLIKACIJA ZA DIJELJENJE OSOBNOG VOZILA

Ivan Prügelhof

Zagreb, veljača 2020.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada".

U Zagrebu, 18.02.2020.

Ivan Prügelhof

Predgovor

Nakon završenog srednjoškolskog obrazovanja, godina traženja pravog smjera i odabira Algebre kao ustanove za visoko obrazovanje mogu reći da sam ispunio svoja očekivanja u stečenom znanju. Prihvativši savjet jednog od profesora konstantnim radom sam uspio prošetati kroz studij bez zaostataka. Ovim putem želim se zahvaliti mentoru Aleksanderu Radovanu na ukazanoj pomoći i usmjeravanju u izradi završnog rada. Posebne zahvale majci i bratu na potpori i razumijevanju koje su mi pružili tokom studija, te ostatku života. Ovaj rad, odnosno stečenu diplomu posvećujem svojim pokojnima ocu, baki i djedu koji su me cijelo vrijeme poticali da upišem i završim neki od studija po želji.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Profesionalna motivacija predlaganja teme je nedostatak takve mobilne aplikacije na Hrvatskom tržištu, njome bi se mogla ponuditi alternativa postojećim aplikacijama koja nedostaje, te platforma koja iz provizije pruža dobar izvor prihoda, jer vozila najčešće većinu vremena provode parkirana na mjestu. Samim time bi se troškovi održavanja vozila (registracija, redoviti servisi) smanjili na prihvatljiviju razinu zbog zarade iz cijene najma. Ova tema je odabrana jer postoji potreba za najmom vozila po cijeni boljoj od trenutno dostupnih putem rent-a car servisa, trenutno ekonomsko stanje ovom aplikacijom omogućuje brz pristup vozilu osobama koje ga nisu u mogućnosti imati u vlasništvu. Ovakvom aplikacijom bi po prihvatljivoj cijeni mogao imati pristup vozilu u vrijeme kad je to potrebno bez brige o troškovima posjedovanja i održavanja vozila. Aplikacija za razliku od postojećih ima prednosti kao što su najmanja provizija i anonimni način koji nudi pregled bez registracije.

Ključne riječi: Android aplikacija, dijeljenje osobnog vozila, najam.

Abstract

The professional motivation for proposing the theme is the lack of such a mobile application on the Croatian market, which could offer an alternative to the existing missing applications, and a platform that provides a good source of income from commission, since vehicles spend most of their time parked on site. This would reduce the cost of vehicle maintenance (registration, regular services) to a more acceptable level due to the rental price. This topic was chosen because I personally needed to rent a car at a price better than currently available through car rental service, as a student I was unable to afford a car. Such an application would be able to access the vehicle at an affordable price at a reasonable time without having to worry about the cost of owning and maintaining the vehicle. The application, unlike the existing ones, has advantages such as the lowest commission and an anonymous mode that offers a non-registration view.

Keywords: Android application, personal car sharing, rent.

Sadržaj

1. Uvod

Ideja same aplikacije je brz i jednostavan najam vozila.

Aplikacija se sastoji od trenutnog i kalendarskog pregleda dostupnih vozila, njihovih lokacija na zadanom području (korištenjem Google Maps i GPS) i korisničke stranice. Iznajmljivači mogu definirati vrijeme dostupnosti vozila kalendarski na određeni datum, za idućih mjesec dana ili odabrati trenutno dostupno za taj dan. Naplata za korištenje vozila vrši se korištenjem postojeće Google Pay platforme za naplatu i globalnog naplatnog *gateway*-a BlueSnap. Ovime bi se povećala upotreba samih vozila uz pristupačnije cijene od rent-a car kompanija i sličnih aplikacija za dijeljenje vozila (“Turo“ ili “Drivy“).

Za rad aplikacije potreban je Android uređaj s pristupom internetu i GPS-om.

Platforma na kojoj sve radi osim jednokratne uplate na Google Play ne stvara nikakve troškove, pa je samim time održavanje jednostavno. Firebase baza podataka je besplatna za potrebe normalnog rada ovakve aplikacije.

Kroz završni rad moći će se bolje upoznati funkcionalnosti i osnovni elementi aplikacije te korišteni *software* Android Studio programski jezik Java, Firebase baza podataka, Android platforma za izradu aplikacije, te testna implementacija samog sustava na Google Play.

2. Android aplikacija

2.1. Opis aplikacije

Aplikacija nazvana GoShare je podijeljena na tri dijela, a to su anonimni, najmoprimac i najmodavac.

Anonimni dio služi za pregled dostupnih vozila iz liste ili po krati u odnosu na korisnikovu lokaciju i zamišljen je kako bi privukao korisnike ponudom sadržaja bez potrebe za prethodnom registracijom za razliku od postojećih rješenja koja traže registraciju, te samim time odbijaju potencijalne korisnike.

Najmoprimac nakon registracije iz postojećih pregleda vozila za anonimne ima mogućnost odabirom željenog vozila vidjeti detalje o vozilu i prema zadanim datumima najma unajmiti vozila za određeni dan ili ako je vozilo dostupno taj dan ponudi se opcija “rent now“. Nakon željenog odabira podaci se prosljeđuju na Google Pay platformu koja ima postojeće predloške za naplatu. Po uspješnoj naplati automatski se upućuje poziv najmodavcu kako bi se što jednostavnije isti dogovorili oko primopredaje vozila. U ovom načinu rada korisniku je za vrijeme trajanja najma dostupan pregled unajmljenog vozila i pozivi najmodavcu.

Treći dio funkcionalnosti je u ulozi najmodavca koji nakon registracije, unosa osobnih podataka i ima mogućnost pregleda konkurentskih vozila bez opcije najma istih kao i anonimni korisnik.

2.1.1. Usporedba i prednosti u odnosu na postojeća rješenja

Prednosti GoShare aplikacije u odnosu na postojeća rješenja[7] su najmanja provizija na tržištu koja iznosi 5% u odnosu na postojeću aplikaciju Turo[11] kojoj je najmanja provizija 10%, posjeduje ostale značajke ove aplikacije, nudi i opciju osiguranja. U Zagrebu postoji mogućnost najma svega dva vozila koja su od rent-a car tvrtke i nalaze se na istoj lokaciji, te time nije dobra pokrivenost. Iako Turo ima odlično napravljenu aplikaciju prednost je moguće ostvariti nižim cijenama i jednostavnosti aplikacije koja ne traži od korisnika previše detalja radi što bržeg ostvarenja najma.

Getaround Europe[10] koji je prije funkcionirao pod nazivom Drivy je aplikacija koja u startu traži registraciju i može odbiti korisnika koji je ovdje da bi istražio osnovne funkcionalnosti i izgled. U slučaju registracije preko Facebook-a traži pristup popisu prijatelja bez opravdanog razloga, a samim time narušava GDPR. Forma za registraciju traži poštanski broj mjesta u kojem se planira najam vozila koja ne funkcionira dobro jer niti ne prepoznaje hrvatske poštanske brojeve, pa se samim time onemogućava korištenje na ovom području. Jedina prednost koja je uočena prema reklami je u mogućnosti otključavanja vozila preko aplikacije bez fizičke interakcije s vlasnikom.

2.2. Korištene tehnologije, programski jezici, alati

Pri izradi aplikacije korišten je objektno orijentirani programski jezik Java. Ona je specifična po tome što se izvodi unutar Java virtualne mašine i može se izvoditi na svim platformama i nije joj potrebna prilagodba za rad, jer se o tome brine virtualna mašina (JVM).

Objektno orijentirano programiranje (OOP) je način pristupa programiranju koji teži korištenju objekata kao strukturiranih podataka iz kojih se korištenjem manjih objekata stvara struktura i način razmjene informacija između istih.

Google Pay[5] API kao alat ima već gotove primjere implementacije kojem se na ključna mjesta unesu podaci za naplatu.

Google Maps[3] API korištenjem dobivenog ključa za aplikaciju omogućuje korištenje platforme unutar podatkovnih limita koji su definirani planom naplate u većim slučajevima.

GPS iz satelitskih podataka ostvaruje preciznu geolokaciju.

Android[1] funkcionira na prilagođenoj verziji operacijskog sustava Linux. U početku je bila bazirana na Javi, a u nekoliko prethodnih godina počinje se koristiti programski jezik Kotlin.

Firebase[2] kao baza podataka napravljena za potrebe mrežnih baza za Android i Internetske preglednike.

3. Struktura aplikacije

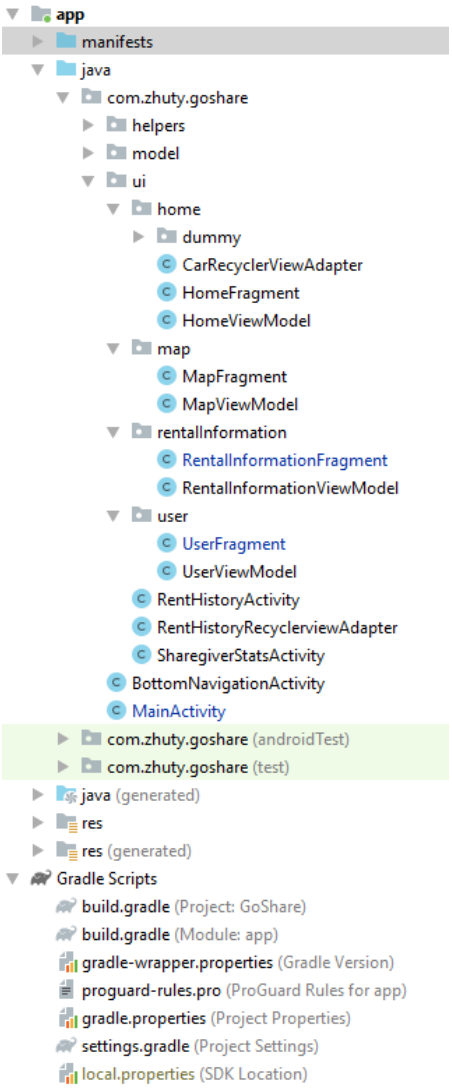
Korištenje aplikacije je isključivo putem Android uređaja minimalne API verzije 24 koja pokriva više od 60% uređaja s ovim operacijskim sustavom. Pokretanjem aplikacije otvara se lista vozila s pripadajućim opisima, cijenom i zračnom udaljenosti od pojedinog vozila. Kroz navigaciju dostupan je pregled vozila na karti, prijava i registracija u sustav, te fragment koji se mijenja ovisno o korisniku. Ako je korisnik najmodavac, moguće je unijeti podatke o vozilu, a u slučaju da je korisnik najmoprimac, ovaj fragment se pretvara u podatke o najmu vozila ukoliko postoji plaćeni najam vozila s opcijom poziva najmodavca.

3.1. Organizacija projekta

U Android studiju je projekt organiziran na dva dijela:

- prvi je sama aplikacija koja se sastoji od Java koda koji sadrži manifest, Java klase i resursi
    - manifest je definicija aplikacije u kojoj su navedene dozvole koje aplikacija traži poput GPS-a, upućivanja poziva i interneta, sadrži API ključ za Google karte, omogućeno korištenje Google Pay API-ja
    - Java se sastoji od klasa koje sadrže sam kod i definiciju metoda koje rade pojedine funkcionalnosti
    - resursi u podmapama imaju sortirane uglavnom XML datoteke koji sadrže definicije slika, izgleda sučelja, tekstove, jezične definicije kao i lokalizaciju ukoliko se napravi
- drugi dio su Gradle[8] skripte koje navode verzije androida kojima se aplikacija koristi, pravila za aplikaciju. Gradle je alat za automatizaciju izgradnje aplikacije kojem se umeću pojedine implementacije aplikacija, te time ubrzavaju izgradnju aplikacije jer se funkcionalnosti iz postojeće implementacije ne moraju ponovno raditi za aplikaciju odnosno netko ih je već napravio i time olakšao programiranje.

Organizacija projekta je vidljiva na slici (Slika 3.1.).



Slika 3.1. Organizacija projekta

3.2. Osnovne mape projekta

3.2.1. Model

Sadrži osnovne klase objekata. Klasa `Car` definira vozilo i parametre koji opisuju samo vozilo i vežu ga ključem uz vlasnika, klasa `User` definira korisnikove podatke, `RentDetails` detalje najma vozila, enumeraciju `SharingTypeEnum` enumeraciju korisnika kao najmodavca ili najmoprimca.

3.2.2. Helpers

Klase koje se često koriste i pomažu u programiranju, te su ondje smještene. Neke od bitnijih su `PrepareMapDataAsyncTask` koji u pozadini priprema geolokacijske podatke jer bi aplikacija u suprotnom prilikom korištenja korisniku davala dojam smrzavanja sučelja, kod (Kôd 3.1. Asinkrona priprema geolokacijskih oznaka**Error! Reference source not found.**).

```
public class PrepareMapDataAsyncTask extends AsyncTask<Void,
Integer, Boolean> {
    private final Context context;
    public PrepareMapDataAsyncTask(Context context) {
        this.context=context;}
    @Override
    protected Boolean doInBackground(Void... voids) {
        if (dataFromDbReady){
            for (Car car : carList) {
                try {
                    Geocoder geocoder=new Geocoder(context);
                    car.setLatLng((new LatLng(
                        geocoder.getFromLocationName(
                              car.getAddress(),1)
                              .get(0)
                              .getLatitude(),
                        geocoder.getFromLocationName(
                              car.getAddress(),1)
                              .get(0)
                              .getLongitude())));
                } catch (IOException e) {e.printStackTrace();
                }
            }
            mapDataAsyncTaskCompleted=true;
        }
        return null;
    }
    public Context getContext() {return context;
    }
    @Override
    protected void onPostExecute(Boolean aBoolean) {
        super.onPostExecute(aBoolean);
        if (mapDataAsyncTaskCompleted)
```

```
                Toast.makeText(getContext(),"Map data
                ready",Toast.LENGTH_SHORT).show();
        }}
```

Kôd 3.1. Asinkrona priprema geolokacijskih oznaka

Klasa `Utils` sadrži metode za pripremu najma. `PaymentsUtil` kreira token za naplatu koji sadrži *gateway* i identifikacijski broj trgovca kod (Kôd 3.2. Specifikacija *gateway* tokena za naplatu**Error! Reference source not found.**). *Gateway* omogućuje posredovanje u naplati transakcije između kupca, trgovca i izdavača kartice.

```
private static JSONObject
getGatewayTokenizationSpecification() throws JSONException {
        return new JSONObject(){{
            put("type", "PAYMENT_GATEWAY");
            put("parameters", new JSONObject(){{
                put("gateway", "bluesnap");
                put("gatewayMerchantId", "781463");
            }
            });
        }};
    }
```

Kôd 3.2. Specifikacija *gateway* tokena za naplatu

3.2.3. `MainActivity`

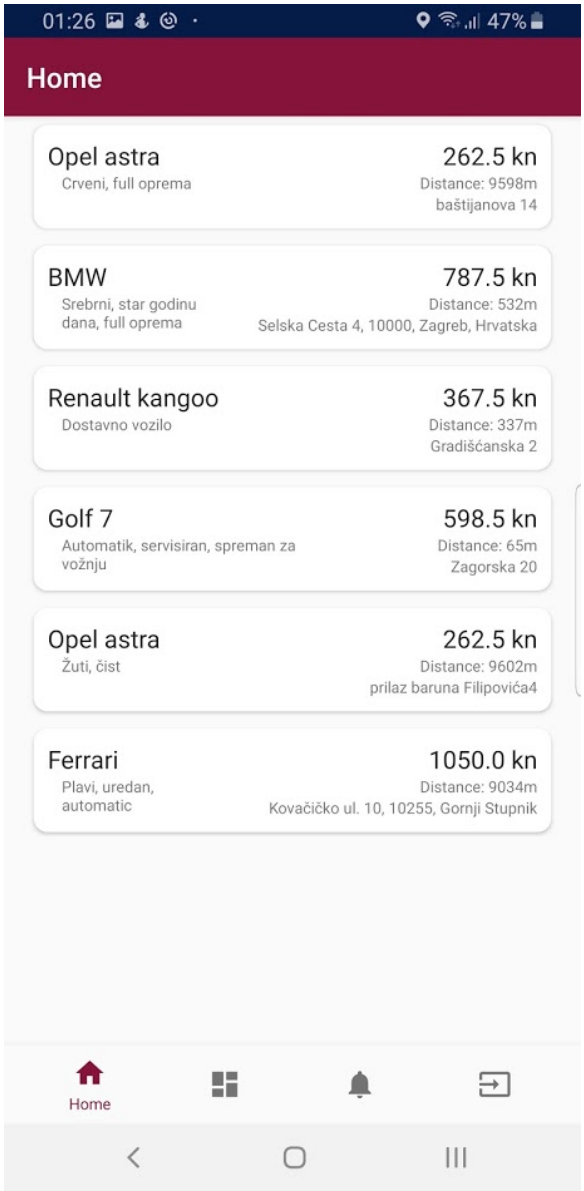
Početna aktivnost koja se prva izvršava prilikom pokretanja aplikacije, tu se prvo provjeravaju dozvole aplikaciji za korištenje GPS-a, poziva i interneta, pripremaju podaci iz baze podataka za nesmetan rad, pokreće asinkrono zadatke pripreme karte koji se paralelno izvode. Po završetku pokreće se aktivnost `BottomNavigationActivity`.

3.2.4. `BottomNavigationActivity`

Aktivnost koju većina modernih Android aplikacija koristi jer omogućava pregledan rad i prelazak između kartica korisničkog sučelja.

3.2.5. `UI`

Skraćenica za korisničko sučelje (engl. *user interface*) je podijeljeno na sljedeće. `HomeActivity`, slika (Slika 3.2. Home aktivnost).



Slika 3.2. Home aktivnost

`HomeActivity` je aktivnost koja služi za prikaz liste vozila iz pogleda `CarRecyclerViewAdapter` kod, (Kôd 3.3.).

```
public void onBindViewHolder(@NonNull
      CarRecyclerViewAdapter.ViewHolder holder
      int position) {
            final Car car = carList.get(position);
            Location l=new Location(car.getId());
            l.setLongitude(car.getLongitude());
            l.setLatitude(car.getLatitude());
      holder.tvTitle.setText(car.getName());

      holder.tvCarDescription.setText(car.getDescription());
      holder.tvPrice.setText(car.getRentalPriceWithFee()+" "
            +holder.itemView.getContext()
            .getResources().getString(R.string.currency));
      holder.tvAddress.setText(car.getAddress());
      if (location!=null)
            holder.tvDistance.setText("Distance: "
                  +(int) location.distanceTo(l)+"m");
      }
```

Kôd 3.3. Adapter reciklirajuće liste vozila

Pogled se reciklira kako aplikacija ne bi trošila previše resursa, ispisuje osnovne podatke o vozilu, cijeni i zračnu udaljenost od trenutne lokacije korisnika aplikacije. Odabirom vozila kreira se okvir za dijalog `AlertDialog`, kod (Kôd 3.4.).

```
new AlertDialog.Builder(view.getContext())
      .setTitle("Rent car")
      .setCancelable(true)
      .setNegativeButton("Pick date to rent", (dialog, which)
->          {
            if (finalDates.getCarId()!=null) {
            Calendar calendar = Calendar.getInstance();
            DatePickerDialog datePickerDialog =
                  new DatePickerDialog(view.getContext(),
                  (view13, year, month, dayOfMonth) -> {
                        month++;//month indexes start from 0
                  if (!checkIfRentedOnSelectedDate(finalC
                        , view13.getContext()
```

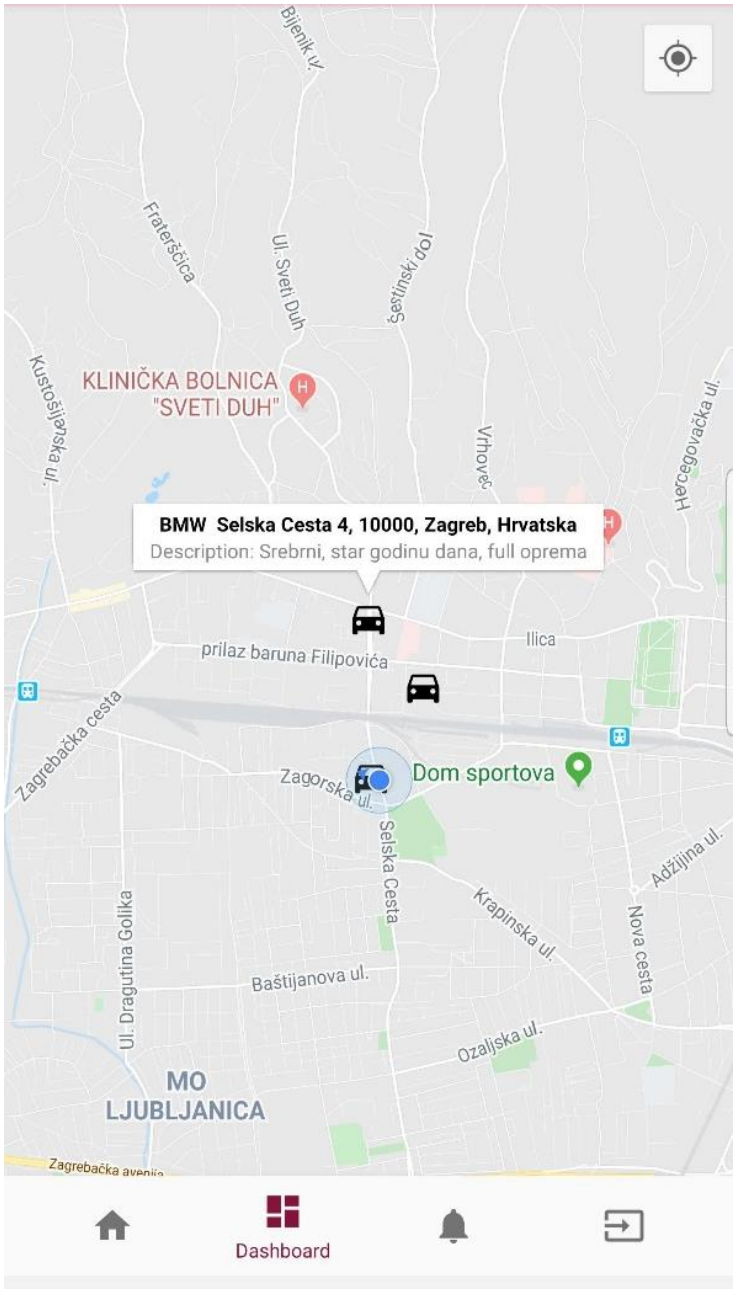
```
                ,getDateString(dayOfMonth,month,year)
        )){
                Toast.makeText(view13.getContext()
                ,"Car already rented on selected
                date"
                ,Toast.LENGTH_SHORT).show();
                dialog.cancel();
        }
        else {
        datePicked = true;
        prepareDataForTransaction(finalC
                , view13.getContext()
                , getDateString(dayOfMonth, month,
                year));
                }
}
, calendar.get(Calendar.DAY_OF_MONTH)
, calendar.get(Calendar.DAY_OF_YEAR)
, calendar.get(Calendar.YEAR));
SimpleDateFormat date=new
SimpleDateFormat("dd/MM/yyyy");
Date resultDate= new
Date(finalC.getRentalDate());
try {
        resultDate = date
                        .parse(finalC.getRentalDate());
} catch (ParseException e) {e.printStackTrace();}
datePickerDialog.getDatePicker()
.setMinDate(System.currentTimeMillis());
calendar.add(Calendar.MONTH, 1);
datePickerDialog.getDatePicker()
.setMaxDate(resultDate.getTime());
datePickerDialog.show();
}
else {
        Calendar calendar = Calendar.getInstance();
        DatePickerDialog datePickerDialog =
        new DatePickerDialog(view.getContext()
        , (view14, year, month, dayOfMonth) -> {
        month++;//month indexes start from 0
        datePicked=true;
```

```
                        prepareDataForTransaction(finalC
                              , view14.getContext()
                              , getDateString(dayOfMonth
                                    ,month,year));
                        }
                        , calendar.get(Calendar.DAY_OF_MONTH)
                        , calendar.get(Calendar.DAY_OF_YEAR)
                        , calendar.get(Calendar.YEAR));
                        SimpleDateFormat date=new
                        SimpleDateFormat("dd/MM/yyyy");
                        Date resultDate= new
                        Date(finalC.getRentalDate());
                        try {
                        resultDate = date
                              .parse(finalC.getRentalDate());
                        } catch (ParseException e) {
                        e.printStackTrace();
                        }
                  datePickerDialog.getDatePicker()
                  .setMinDate(System.currentTimeMillis());
                  datePickerDialog.getDatePicker()
                  .setMaxDate(resultDate.getTime());
                  datePickerDialog.show();
                  }
            })
            .setMessage("About car:\n" + c.getName() + "\n "
                  + c.getDescription() + "\n " + c.getAddress()
                  + "\n " + c.getRentalPrice() + "kn")
            .setNeutralButton("Cancel", null)
            .setIcon(R.drawable.ic_directions_car_black_24dp)
            .show();
      }
```

Kôd 3.4. Primjer dijaloškog okvira za najam vozila

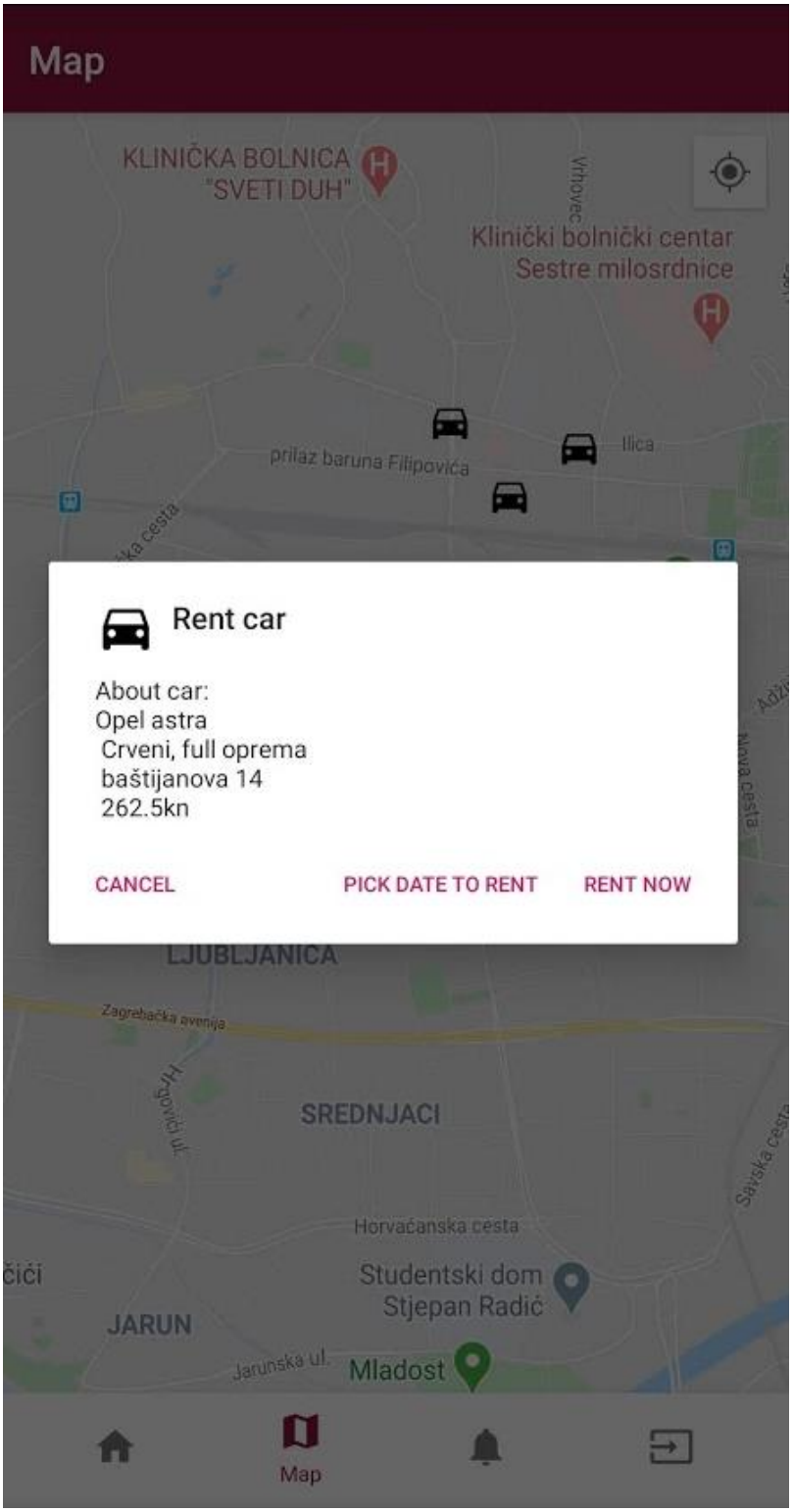
Okvir za dijalog nudi opcije najma “rent now“, “pick date to rent“ i poništavanje odabira “cancel“, slika (Slika 3.2.). Odabirom na “pick date to rent“ stvara se kalendarski prikaz pozivom metode `DatePickerDialog` navedenog unutar koda (Kôd 3.4.).

`MapActivity` radi grafički prikaz lokacija vozila na karti, kao i `HomeActivity` aktivnost nudi opciju najma vozila klikom na vozilo, slika (Slika 3.3.).



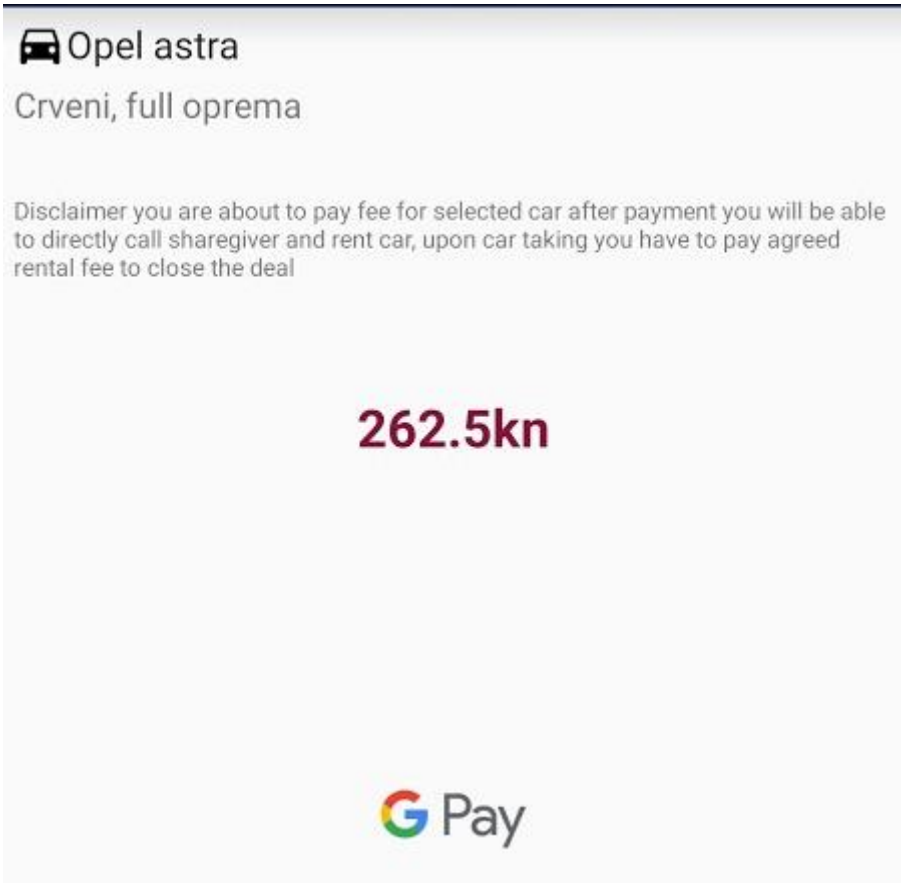
Slika 3.3. Aktivnost mapa

Odabirom željenog vozila otvara se okvir s dijalogom u kojem su ispisani detalji i mogućnosti najma kalendarski ili trenutno, slika (Slika 3.4.).



Slika 3.4. Dijalog najma

Odabirom najma vozila slijedi fragment koji klikom na gumb “G Pay“ pokreće naplatu preko Google Pay sustava, slika (Slika 3.5.).



Slika 3.5. Google Pay naplata

Aplikacija vrši pripremu podataka za naplatu u metodi `prepareDataForTransaction` postavlja odabrano vozilo koje sadrži sve podatke o vozilu i cijeni najma, kontakt telefon najmodavca, datum. Podaci se šalju na `CheckoutActivity` aktivnost, kod (Kôd 3.5. Priprema podataka za naplatu).

```
public static void prepareDataForTransaction(Car finalC,
Context context, String dateString) {
      mCar=finalC;
      rentDetails.setCar(finalC);
      rentDetails.setPhone(userPhoneMap
      .get(finalC.getOwnerId()));
      rentDetails.setRentalDate(dateString);
      context.startActivity(
            new Intent(context, CheckoutActivity.class));
      }
```

Kôd 3.5. Priprema podataka za naplatu

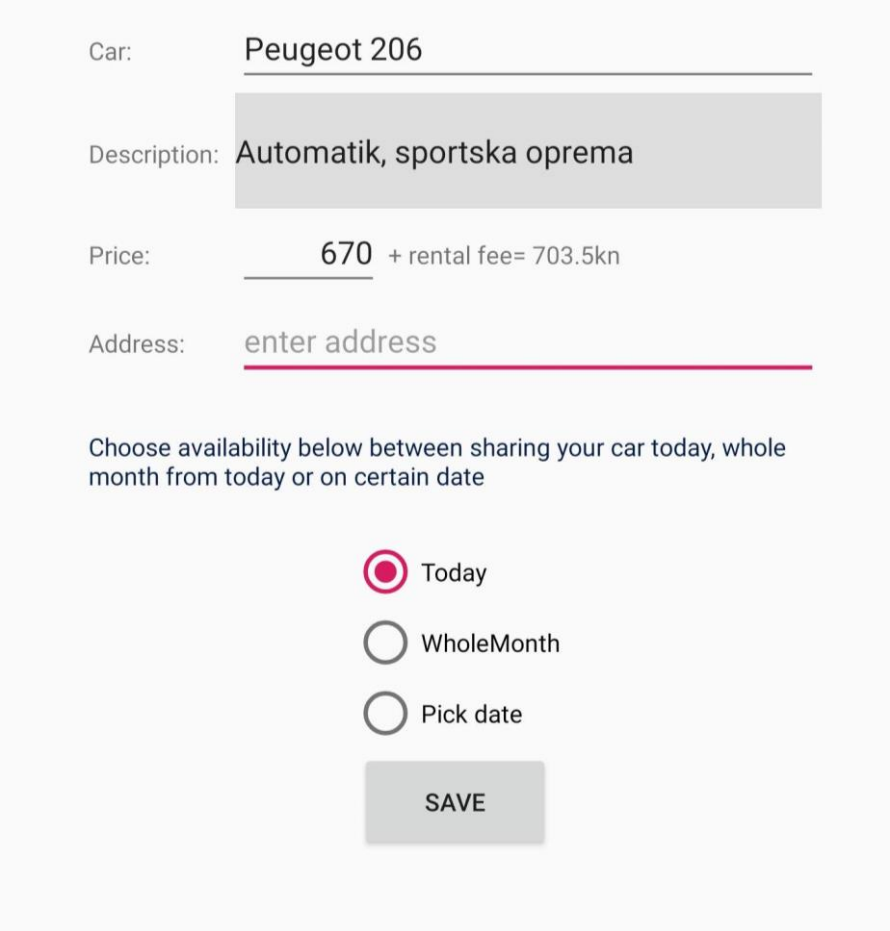
Aktivnost `CheckoutActivity` ima implementiranu metodu `onActivityResult` obrade rezultata naplate koja u slučaju greške obrađuje grešku zbog koje nije izvršena naplata, te o tome obavještava korisnika, a u pozitivnom slučaju naplate prosljeđuje metodi `handlePaymentSuccess` zajedno sa sadržajem naplate i vrši ažuriranje slobodnih datuma najma vozila pozivom `removeDateFromCar`, zatim sprema promjene u bazi i umeće nove podatke o najmu metodom `saveRentData`, kod (Kôd 3.6. Spremanje podataka o najmuKôd 3.6).

```
private void saveRentData(RentalDates dates, boolean dlChg) {
        DatabaseReference databaseReference=
        FirebaseDatabase.getInstance().getReference();
        databaseReference.child("cars").child(mCar.getId())
            .setValue(mCar)
            .addOnSuccessListener(aVoid ->
            Toast.makeText(getApplicationContext()
                  ,"Data edit saved"
                  ,Toast.LENGTH_SHORT).show());
      if (dlChg)databaseReference.child("rentalDates")
            .child(mCar.getId()).setValue(dates);
                  saveToHistory(rentDetails
                        ,user.getUid(), databaseReference);
      }
```

Kôd 3.6. Spremanje podataka o najmu

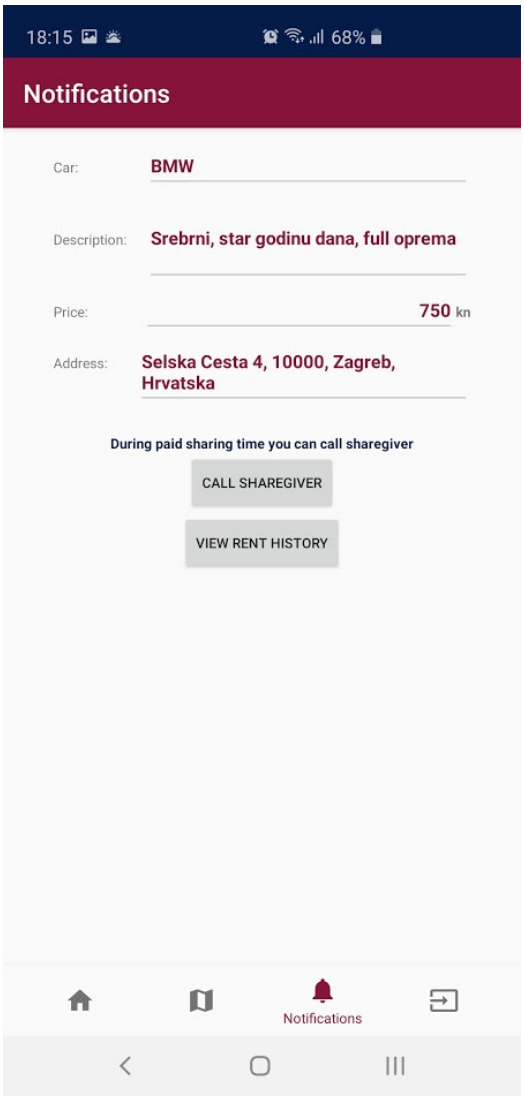
Po uspješnoj naplati i spremanju podataka upućuje se poziv najmodavcu kako bi se dogovorila primopredaja vozila na odabrani dan.

`RentalInformationActivity` najma ovisno o vrsti korisnika ispisuje podatke o najmu vozila. Najmodavac unosi podatke vozila i jednu od opcija najma i cijenu uz koju je prikazana cijena s provizijom koju će korisnik platiti kako bi ostvario najam. Klikom na oznaku adrese popunjava se adresa iz zadnje zabilježene GPS pozicije, slika (Slika 3.6.).



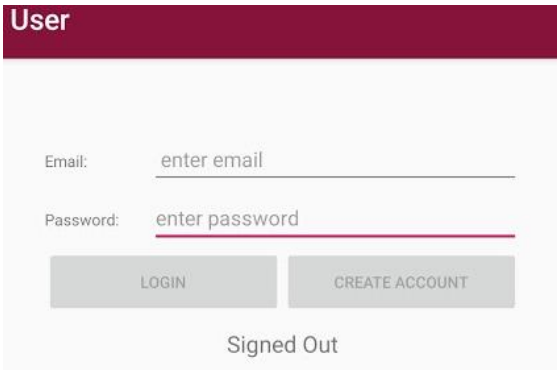
Slika 3.6. Aktivnost s informacijama za najmodavca

Najmoprimac unutar aktivnosti ima drugačiji izgled sučelja prilagođen njemu. Sadrži podatke o vozilu cijenu najma s provizijom koja je naplaćena, adresu na kojoj se vozilo nalazi, opcije pregleda povijesti najma i mogućnost poziva najmodavca, slika (Slika 3.7.).

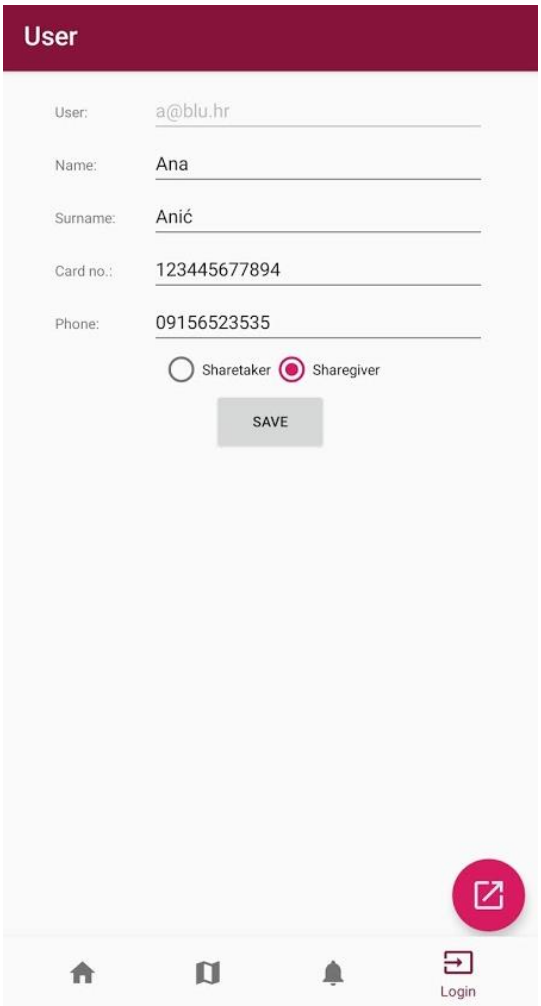


Slika 3.7. Izgled sučelja pregleda unajmljenog vozila

`UserActivity` aktivnost iz koje je moguća brza registracija, slika (Slika 3.8), prijava u aplikaciju i odjava pritiskom na plutajući crveni gumb *floating button*, slika (Slika 3.9.Slika 3.9).

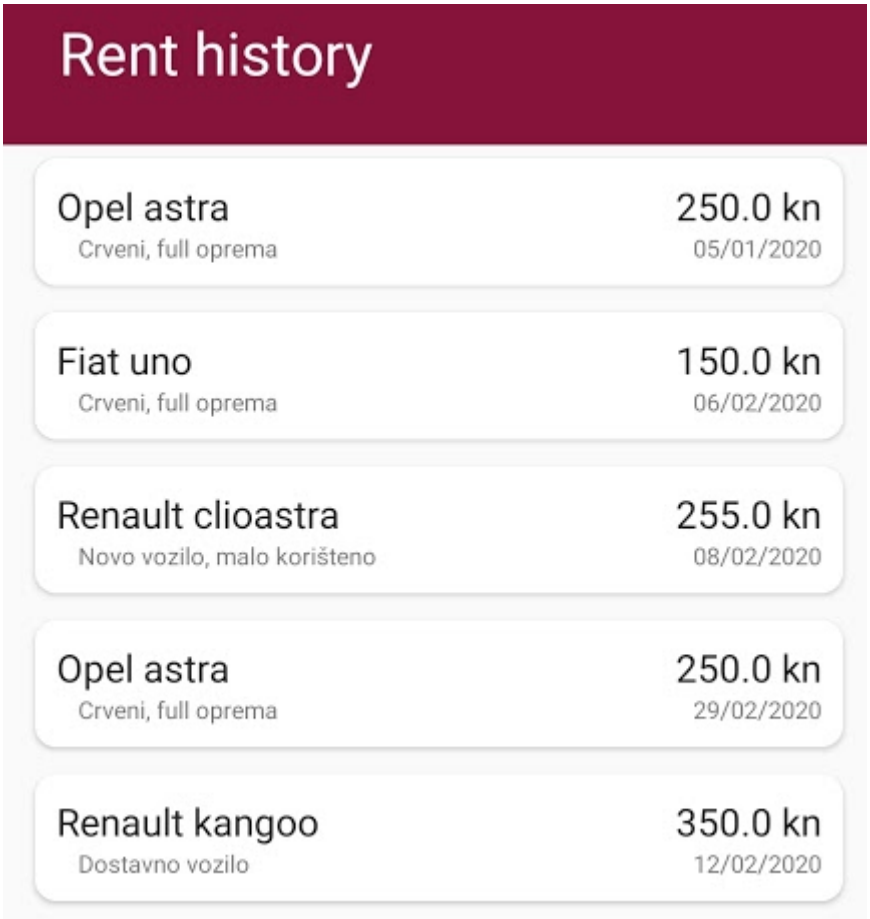


Slika 3.8. Registracija korisnika ili izrada novog korisnika



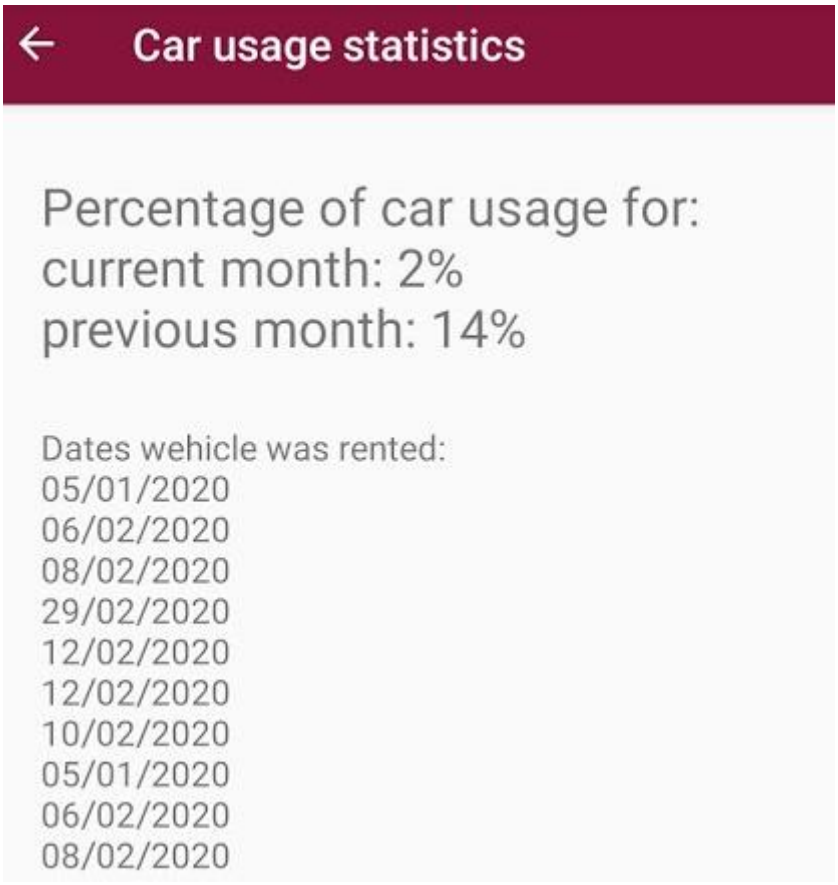
Slika 3.9. Korisnička aktivnost

`RentHistoryActivity` je aktivnost unutar koje je najmoprimcu moguće pregledati povijest plaćenih najmova vozila. Povijest sadrži listu vozila s pripadajućim opisom, datumom najma i plaćenom cijenom najma, slika (Slika 3.10.).



Slika 3.10. Povijest najma

`SharegiverStatsActivity` se odnosi na pregled informacija vezanih za najmodavca, unutar te aktivnosti pokazuje se statistika iskorištenosti vozila za tekući mjesec i prošli kako bi najmodavac imao uvid u uspješnost najma, odnosno postotak iskorištenog kapaciteta i datume koje je vozilo bilo u najmu, slika (Slika 3.11.`Slika 3.11`).



Slika 3.11. Statistika najma

3.2.6. Res

Skraćenica za resurse definira mape s vrijednostima (boje, dimenzije, stilovi, resursi tekstova), slike, te vizualne predloške za vanjski izgled i gradivni fragmenti aplikacije prikazuju izgled sučelja.

Kroz resurse se na jednom mjestu vrše promjene koje su referencirane unutar aplikacije. Ovdje se mogu mijenjati tekstovi, dodavati jezici, slika (Kôd 3.7. Primjer dijela tekstova resursa**Error! Reference source not found.**).

```
<resources>
    <string name="app_name">GoShare</string>
    <string name="enableGPS">Enable GPS</string>
    <string name="enableINTERNET">enable internet</string>
    <string name="title_home">Home</string>
    <string name="title_dashboard">Dashboard</string>
    <string name="title_notifications">Notifications</string>
    <string name="title_activity_maps">Map</string>
```

Kôd 3.7. Primjer dijela tekstova resursa

4. Baza podataka

Firebase baza podataka je odabrana jer dobro funkcionira unutar Android sustava, stvorena je od strane Google-a i samim time ima dobru potporu za potrebne funkcionalnosti. Za ovu aplikaciju odabrana je besplatna verzija jer pokriva zahtjeve aplikacije u radu s podacima.

Podaci se spremaju kao jedan veliki JSON dokument kao NoSQL baza, takve baze podataka ne iziskuju znanje SQL strukturiranja relacijskih tablica. Ovakav pristup je fleksibilniji od relacijskih baza podataka, jednostavniji je dizajn baze. Veza između podataka je organizirana po ključevima i pripadajućim vrijednostima za svaki ključ u bazi, primjer slika (Slika 4.1.).

Firebase baza je isplativa kod manjih prometa i manjih baza podataka od nekoliko desetaka gigabajta, veće baze ne funkcioniraju tako brzo niti ih je lako održavati kada je NoSQL i Firebase u pitanju, čime se narušava korisničko iskustvo i isplativost za programere zbog manjeg prostora za zaradu i duljim vremenom razvoja.

Baza nudi modul za registraciju korisnika engl. *authentication* koja brine o novim i postojećim korisnicima aplikacije, programeri ovdje mogu brisati, deaktivirati korisnike, pregledavati njihovo korisničko ime i identifikacijsku oznaku bez uvida u lozinke, te tako osigurava sigurnost korisnika.

Firebase Crashlytics opcija je također implementirana tokom razvoja kao koristan dodatak aplikaciji jer bilježi u konzoli *web* pretraživača projekta sva rušenja aplikacije s pratećim informacijama u kojoj liniji koda je došlo do rušenja i razlog rušenja. Implementacija se vrši u Gradle mapi projekta unosom putanje do dodatka prema dokumentaciji Firebase-a.

Slika 4.1. Organizacija baze podataka

Čitanje vozila iz baze vrši se referencom na naziv unutar baze podataka. Referenci se dodaje `addValueEventListener` koji osluškuje izmjene unutar baze, te nakon izmjene osvježava podatke. Primjer čitanja vozila iz baze podataka, kod (Kôd 4.1. Čitanje vozila iz baze podataka).

```
DatabaseReference
myRef=FirebaseDatabase.getInstance().getReference("cars");
myRef.addValueEventListener(new ValueEventListener() {
      @Override
      public void onDataChange(@NonNull DataSnapshot snap){
            carList=new ArrayList<>();
            for (DataSnapshot ds : snap.getChildren()) {
                  Car car=ds.getValue(Car.class);
```

```
                    assert car != null;
                    car.setId(ds.getKey());
                    carList.add(car);
                }
            }
        });
```

Kôd 4.1. Čitanje vozila iz baze podataka

Firebase unutar korisničkog sučelja za programere nudi opciju *authentication* koja je implementirana u aplikaciju kao gotovo rješenje za sigurnu registraciju novih metodom `register`, izvodi se provjera unosa kako ne bi došlo do neočekivanog rušenja aplikacije, kod (Kôd 4.2.), i registracija postojećih korisnika `login`, kod (Kôd 4.3).

```
void register(){
      String email=etEmail.getText().toString().trim();
      String password=etPassword.getText().toString().trim();
      if (password.length()<8){
            Toast.makeText(getContext()
            ,"Required minimum length 8 characters"
            , Toast.LENGTH_SHORT).show();
            return;
      }
      prepareUserFields();
      mAuth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(Objects.requireNonNull(
                  this.getActivity()), task -> {
                        if (task.isSuccessful()) {
                        // Sign in success, update UI with
                        the signed-in user's information
                        Log.d(TAG
                        , "createUserWithEmail:success");
                        user = mAuth.getCurrentUser();
                        Toast.makeText(getContext()
                        , "User sucessfully created!"
                        ,Toast.LENGTH_SHORT).show();
                        updateUI(user);
                  } else {// If sign in fails, display a
                  message to the user.
                        Log.w(TAG
                        , "createUserWithEmail:failure"
                        , task.getException());
```

```
                        Toast.makeText(getContext()
                        , "Authentication failed."
                        +task.getException()
                        .getLocalizedMessage()
                        ,Toast.LENGTH_SHORT).show();
                        updateUI(null);
                        }
                  });
            }
```

Kôd 4.2. Registracija korisnika

```
void login(){
      String email=etEmail.getText().toString().trim();
      String password=etPassword.getText().toString().trim();
      if (password.length()<8){
            Toast.makeText(getContext()
            ,"Required minimum length 8 characters"
            , Toast.LENGTH_SHORT).show();
            return;
      }
      prepareUserFields();
      mAuth.signInWithEmailAndPassword(email, password)
      .addOnCompleteListener(Objects.requireNonNull(this
            .getActivity()), task -> {
                  if (task.isSuccessful()) {
                  // Sign in success, update UI with
                   the signed-in user's information
            Log.d(TAG, "signInWithEmail:success");
            user = mAuth.getCurrentUser();
            getUserCarDetails();
            Toast.makeText(getContext(), "Logged in as: "
            +user.getEmail(),Toast.LENGTH_SHORT).show();
            tvUser.setText("Logged in as: "+user.getEmail());
            getRentHistory();
            showUserData();
            setUserData();
            updateUI(user);
            } else {// If sign in fails, display a message
                  to the user.
            Log.w(TAG, "signInWithEmail:failure"
                  , task.getException());
```

```
                Toast.makeText(getContext()
                      , "Authentication failed." +
                      task.getException()
                      .getLocalizedMessage(),Toast.LENGTH_SHORT)
                      .show();
                updateUI(null);
                }
          });
    if (mUser.getSharingTypeEnum()==SharingTypeEnum.sharetaker)
          getRentedCarDetails();
    }
```

Kôd 4.3. Registracija postojećih korisnika

Korisnici nakon izmjene ili unosa svojih podataka spremaju podatke metodom `save`, postavljaju se podaci o korisniku i spaja se na bazu pozivom metode `DatabaseReference`, prikazano u kodu (Kôd 4.4.).

```
    void save(){
          mUser.setId(user.getUid());
          mUser.setName(etName.getText().toString());
          mUser.setSurname(etSurname.getText().toString());
          mUser.setCardNumber(etCardNumber.getText().toString());
          mUser.setPhone(etPhone.getText().toString());
          if (rbSharegiver.isChecked())
          mUser.setSharingTypeEnum(SharingTypeEnum.sharegiver);
          else mUser.setSharingTypeEnum(SharingTypeEnum
                                        .sharetaker);
          DatabaseReference databaseReference=
          FirebaseDatabase.getInstance().getReference();
          databaseReference.child("users").child(mUser.getId())
          .setValue(mUser).addOnSuccessListener(aVoid ->
                Toast.makeText(getContext()
                ,"Data saved",Toast.LENGTH_SHORT).show());
          }
```

Kôd 4.4. Spremanje korisničkih podataka

5. Omogućavanje dostupnosti aplikacije krajnjim korisnicima

Kada je aplikacija spremna za produkciju, potrebno ju je unutar Android studija osvježiti naredbom čišćenja koda engl. *clean project*, generirati ključ za potpis i potpisati, nakon čega se radi sama aplikacija s ekstenzijom .apk. Takvu aplikaciju je moguće prenijeti na bilo koji uređaj i ondje instalirati, kako bi napravili aplikaciju dostupnom na Google Play[4] Store-u potrebno je imati Google Play Developer račun za kojeg se plaća jednokratna naknada od 25 američkih dolara. Na korisničkim stranicama se odabire lista podržanih zemalja, prihvaćaju uvjeti korištenja, učita se prethodno generirana i potpisana aplikacija. Kada je predan zahtjev za produkciju, aplikacije Google vrši provjeru koda i ona je najčešće kroz 2 sata vidljiva na Google Play Store-u.

6. Zaključak

Ovoj aplikaciji je cilj bio kroz nekoliko kratkih koraka iznajmiti vozilo od privatne osobe i po želji imati pregled iz liste ili karte. Jednostavnom formom za registraciju korisnika omogućiti brz unos najosnovnijih podataka za ostvarenje najma, te po plaćanju upućivanje poziva najmodavcu oko dogovora preuzimanja vozila.

Android platforma se pokazala kao pouzdan temelj za izgradnju mobilne aplikacije pružajući fleksibilnost pri izradi programskog rješenja koje nudi razvoj aplikacije u svim do sada aktualnim SDK verzijama. Razvojna okolina Android studio[6] na jednom mjestu objedinjuje module za izradu, spajanje i osnovne upute za korištenje Firebase baze. Gradle odmah otkriva nove engl. *library*-je i sinkronizira ih s projektom. Implementacija Google API-ja za Maps i Pay je jednostavna, brza i dobro dokumentirana, podržana primjerima.

Usporedbom postojećih aplikacija u budućnosti je moguće proširiti dodatnim funkcionalnostima koje imaju konkurentske aplikacije, ponuditi osiguranje vozila u suradnji s osiguravajućim kućama.

Daljnji razvoj aplikacije ovisi o zaradi provizije, odzivu i ocjeni korisnika zbog jake konkurencije koja velikim resursima može napraviti brži razvoj, te implementaciju novih funkcionalnosti.

Popis kratica

API	*Application Programming Interface*	sučelje za programiranje aplikacija
GDPR	*General Data Protection Regulation*	opća uredba o zaštiti podataka
GPS	*Global Positioning System*	globalni položajni sustav
JSON	*JavaScript Object Notation*	format za razmjenu podataka
JVM	*Java virtual machine*	Java virtualna mašina
NoSQL	*nonSQL*	vrsta baze podataka
SDK	*software development kit*	skup za izradu softvera
SQL	*Structured Query Language*	jezik za relacijski model baze podataka
UI	*user interface*	korisničko sučelje
WEB	*World Wide Web*	skraćenica za internet
XML	*EXtensible Markup Language*	jezik za označavanje podataka

Popis slika

Popis kôdova

Literatura

[1] ANDROID, https://developer.android.com/guide/platform/, preuzeto 19.01. 2020.

[2] FIREBASE, https://firebase.google.com/docs/guides/, preuzeto 19.01. 2020.

[3] GOOGLE MAPS, https://cloud.google.com/maps-platform/, preuzeto 20.01. 2020.

[4] GOOGLE PLAY, https://developer.android.com/distribute/, preuzeto 20.01. 2020.

[5] GOOGLE PAY, https://developers.google.com/pay/api, preuzeto 20.01. 2020.

[6] ANDROID STUDIO, https://developer.android.com/studio/, preuzeto 20.01. 2020.

[7] POSTOJEĆA RJEŠENJA, https://www.mobindustry.net/top-8-carsharing-mobile-apps/, preuzeto 09.10.2019.

[8] GRADLE, https://docs.gradle.org/current/userguide/what_is_gradle.html, preuzeto 20.01. 2020.

[9] BLUESNAP, https://developers.bluesnap.com/v8976-Basics/docs/google-pay#section-implementing-google-pay-in-your-android-app, preuzeto 20.01. 2020.

[10] GETAROUND, https://www.getaround.com/, preuzeto 19.01. 2020.

[11] TURO, https://turo.com/, preuzeto 19.01. 2020.

Prilog

CD sa sljedećim sadržajima:

- Završni rad
- Arhivu aplikacije
- Aplikacija

ALGEBRA VISOKO UČILIŠTE

NASLOV ZAVRŠNOG RADA

Pristupnik: Ivan Prügelhof,

JMBAG:0321006639

Mentor: prof. Aleksander Radovan