# SUSTAV ZA UPRAVLJANJE VOZNIM PARKOM

**Banay, Mario**

**VISOKO UČILIŠTE ALGEBRA**

DIPLOMSKI RAD

# SUSTAV ZA UPRAVLJANJE VOZNIM PARKOM

Mario Banay

Zagreb, rujan 2019.

# Predgovor

Zahvaljujem se svojoj supruzi Tanji na povjerenju, ljubavi, strpljenju i energiji koje je prenijela na mene kada mi je bilo najteže. Puno hvala mojim roditeljima i braći, puncu i punici, šogoru i šogorici, kumovima i prijateljima koji su i u teškim vremenima vjerovali u mene i bili mi podrška. Veliko hvala mentoru Juričić Vedranu koji me je potakao na sagledavanje problema iz različitih perspektiva, strpljivo odgovarao na sva pitanja i pomogao mi da dođem do odgovarajućih rješenja pri izradi rada. Hvala svim nastavnicima koji su s puno entuzijazma prenosili znanje na mene, ponekad vikendom od jutra do večeri. Hvala osnivačima Algebre koji su imali hrabrosti i energije da pokrenu izgradnju ovako kvalitetne ustanove te svim djelatnicima koji su dali svoj doprinos.

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

# Sažetak

Tema diplomskog rada je izrada programskog rješenja za upravljanje voznim parkom. Motivacija za odabir teme je primjena znanja stečenog tijekom studija važnog za daljnji profesionalni razvoj. Rezultat rada je sustav koji se sastoji od mobilne aplikacije za Android platformu i internet sučelja za administraciju korisnika. Mobilna aplikacija je predviđena za upotrebu od strane korisnika vozila, dok je internet sučelje namijenjeno za korištenje od strane administratora voznog parka. Mobilna aplikacija prikuplja podatke o kretanju vozila putem GPS modula iz mobilnog telefona. Administrator putem internet sučelja ima mogućnost unošenja svih troškova, pregled povijesti prijeđenih ruta i druge funkcionalnosti opisane u radu. Programsko rješenje će omogućiti tvrtkama koje se bave upravljanjem voznim parkom učinkovito upravljanje troškovima eksploatacije vozila koje iznajmljuju svojim korisnicima.

**Ključne riječi:** vozni park, upravljanje troškovima, mobilna aplikacija, web aplikacija

# Abstract

The subject of this thesis is a development of fleet management software. The Motivation for selecting this topic of this thesis is the application of knowledge acquired during studying important for further professional development. Result of the Master's thesis is a system consisting of a mobile application and a web interface. The Mobile application is developed for users driving a car, while the web interface is developed for the fleet management administrator. The mobile application collects data from a GPS module. Fleet management administration provides expense management, route history checking and other functionalities, also described in this thesis. This software will provide fleet management companies an efficient fleet management and help with overall expense reduction.

**Keywords:** vehicle fleet, expense management, mobile application, web application

# Sadržaj

# 1. Uvod

Upravljanje voznim parkom temeljni je dio poslovanja svake tvrtke koja se bavi iznajmljivanjem vozila ili prijevozom dobara, a veliki utjecaj ima na troškove poslovanja tvrtki koje posjeduju veći broj vozila. Kod dugoročnog najma vozila, klijent najčešće plaća mjesečnu cijenu najma u koju su uključeni svi ili gotovo svi troškovi vozila. S obzirom na veliku količinu vozila koje tvrtka iznajmljuje i činjenicu da je trošak goriva minimalno 50% ukupnih troškova eksploatacije vozila, optimizacija troškova donosi vrijednost za tvrtku, klijenta, te ima pozitivan utjecaj na očuvanje okoliša. U diplomskom radu bit će obrađen primjer sustava za tvrtke koje se počinju baviti pružanjem usluge dugoročnog najma vozila. Takve tvrtke imaju potrebu za jednostavnim i ekonomski prihvatljivim sustavom, koji može pratiti rast tvrtke. Cilj rada je izraditi prikladan sustav koje bi mogle koristiti navedene tvrtke. Rad će čitatelje voditi kroz postupak izrade sustava počevši s analizom dostupnih rješenja na tržištu i određivanja zahtjeva do završetka izrade aplikacije, te analize mogućnosti i nedostataka gotovog proizvoda.

# 2. Upravljanje voznim parkom

Vozni park je skup prijevoznih sredstava tvrtke. Prijevoznim sredstvima koja se smatraju primjerena za primjenu radom obrađenim rješenjem se smatraju cestovna vozila, s pojednostavljenom zakonskom podjelom [10] na kategorije:

- A – motocikli s ili bez bočne prikolice i motorna vozila s tri kotača čija je snaga veća od 15 kW
- B – vozila s najvećom dopuštenom masom 3500 kg konstruirana za prijevoz do osam putnika
- C – motorna vozila s najvećom dopuštenom masom većom od 7500 kg
- D – motorna vozila za prijevoz više od osam putnika uz vozača

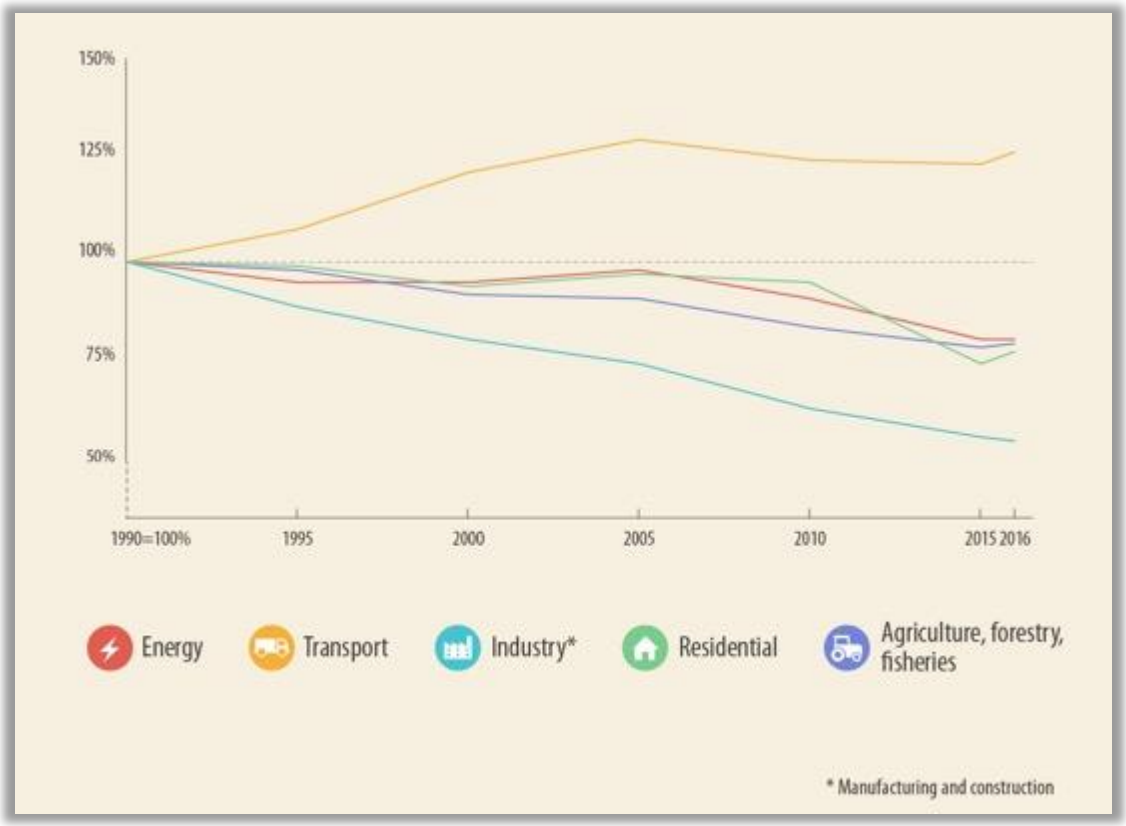
Vozni parkovi prema veličini dijele se na sljedeće kategorije [15]:

- Mali vozni park s manje od 20 vozila
- Srednji vozni park s manje od 100 vozila
- Veliki vozni park s manje od 500 vozila
- Vrlo veliki vozni park s 500 vozila ili više

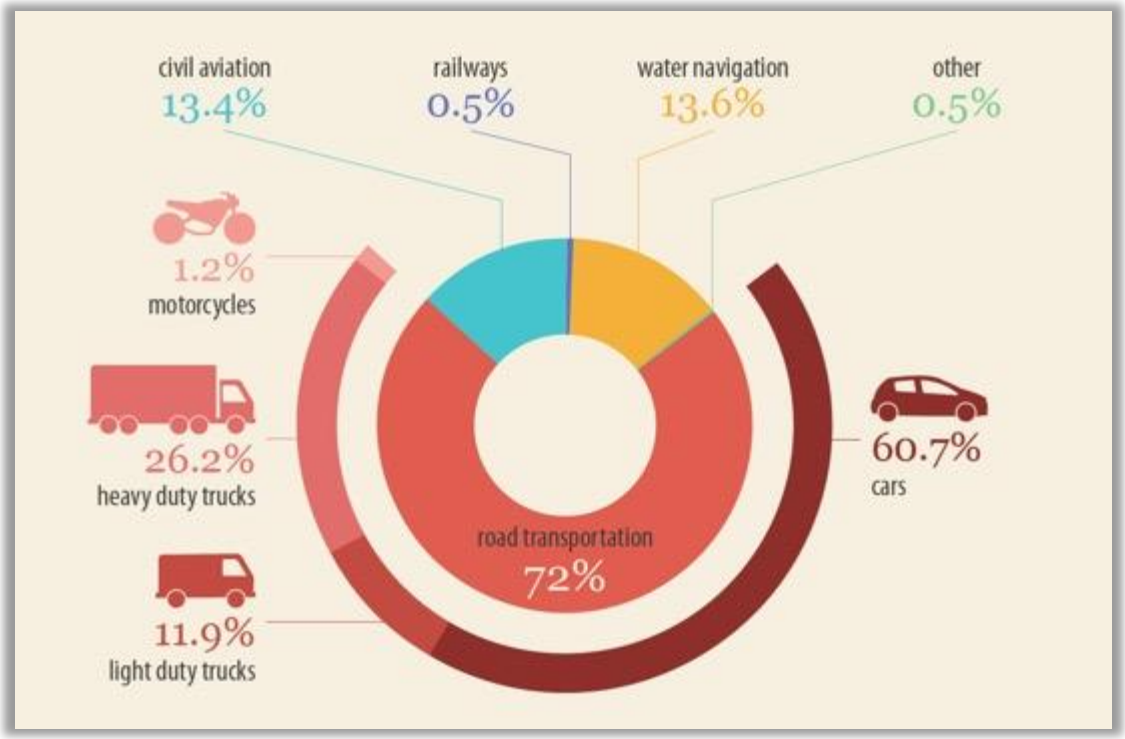
Svrha sustava je optimizirati troškove eksploatacije vozila kako bi prijevoz bio učinkovitiji. Temelj učinkovitog upravljanja voznim parkom je donošenje ispravnih odluka vezanih za troškove koji nastaju tijekom eksploatacije vozila. Osoba ovlaštena za upravljanje voznim parkom pristupa podacima putem aplikacije koja je instalirana na računalu i/ili mobitelu lokalno, a može se nalaziti i na poslužitelju.

## 2.1. Značaj upravljanja voznim parkom u modernom okruženju

Količina emisije CO2 (*engl. Carbon Dioxide*) u atmosferu povezana je s globalnim zatopljenjem. Zbog navedenog, globalni cilj je smanjiti emisije koliko god je moguće. Proteklih 20-ak godina ukupna količina emisija CO2 u atmosferu ima trend smanjenja po svim kategorijama osim u prijevozu, slika 2.1. Prijevoz sudjeluje s približno 30% ukupnih emisija CO2 u Europskoj uniji s tendencijom rasta. Zbog navedenog Europska unija je postavila strateški cilj smanjiti emisije CO2 u prijevozu za 60% do 2050 godine.




Slika 2.1 Emisije CO2 u Europskoj uniji




Slika 2.2 Raspodjela emisija CO2 prema vrsti prijevoza

Može se zaključiti da je utjecaj prijevoza značajan za globalno zatopljenje, te je nužno iskoristiti svaku mogućnost za smanjenje emisija CO2. Na temelju raspodjele emisija CO2 koju prikazuje slika 2.2, vidi se da prijevoz osobnim automobilima emitira više od 60% količine emisija koju se emitira za prijevoz. Dakle, postoji veliki potencijal za smanjenje emisije CO2 optimizirajući eksploataciju osobnih automobila, na čemu se i temelji ovaj rad.

Upravljanje voznim parkom podrazumijeva uvid u podatke poput prijeđenih udaljenosti, količine potrošenog goriva na temelju kojih se mogu izračunati emisije CO2 po prijeđenom kilometru. Uvidom u navedene podatke, moguće je utjecati na odluku klijenata i motivirati ih za nabavku vozila s manjom emisijom CO2. S vremenom i napretkom tehnologije, nastupit će vrijeme kada će autonomna vozila postati dio svakodnevice, do tada će upravljanje voznim parkom postati sastavni dio svakog vozila. Aplikacija koja je sada odvojena od vozila, postat će sastavni dio, dostupna i isplativa svakom korisniku, a ne samo tvrtkama s voznim parkom. Tako će svaka osoba koja koristi vozilo moći na jednostavan način doprinijeti smanjenju CO2.

## 2.2. Dostupna softverska rješenja na tržištu

Na tržištu je dostupan čitav niz softverskih rješenja pogodnih za upravljanje voznim parkom. Neka od rješenja nude se u obliku samostalne web aplikacije koje je potrebno instalirati na uređaj, a neka se smještena na web poslužitelj (*engl. server*) te dostupna na internetu. Pojedine tvrtke nude i uslugu obuke za rad sa sustavom i 24/7 podršku putem korisničke službe. Plaćanje se obavlja putem mjesečne pretplate po vozilu voznog parka. Rješenja su razvijena modularno, tako da kupac ima mogućnost odabrati pretplatu s opcijama koje najbolje odgovaraju njegovim potrebama. U daljnjem tekstu bit će opisana 3 rješenja najsličnija po namjeni rješenju izrađenom za potrebe ovog rada.

Webfleet aplikacija ima jednostavno sučelje, optimizirano za maksimalno korisničko iskustvo. Korisnik može izabrati između TomTom i Google maps karte. Aplikacija može osim lokacije vozila prikazati i različite razine detalja poput; stanja u prometu, nazive kvartova, adrese i nazive županija. Webfleet nadzorna ploča (*eng. Dashboard*) omogućuje korisnicima praćenje do 27 KPI parametara (eng. *Key Performance Indicator*, skraćeno KPI). Postoji i mogućnost izrade izvještaja kako bi korisnik mogao vidjeti trendove važnih parametara za poslovanje. Dodatno postoji i mogućnost optimiranja ruta za tvrtke koje se bave prijevozom. Korisnik može odabrati između 5 modela GPS prijamnika.

Najjednostavniji model prijemnika bilježi lokaciju i prijeđenu udaljenost, a najnapredniji ima mogućnost bilježenja naprednih parametara poput spajanja s tahografom, bilježenja stila vožnje i očitavanje trenutnog stanja vozila. Ekran veličine 5 ili 7 inča koji se ugrađuje u vozilo se isporučuje kao dio paketa. Najjednostavniji model radi kao konvencionalni navigacijski sustav gdje korisnik zadaje rute, a uređaj prikazuje putanju s razlikom da se podaci šalju na poslužitelj. Najnapredniji uređaj ima ugrađenu kameru koja može snimati put, prikazuje ograničenja brzine i druge podatke.

Tvrtka TomTom Telematics je razvila sustav za upravljanje voznim parkom temeljen na velikom iskustvu u izradi navigacija za vozila [18]. Sustav je razvijen u suradnji s američkim ministarstvom obrane, što samo po sebi govori važnosti pouzdanosti sustava. U ponudi su 3 varijante sustava gdje svako od njih obuhvaća sljedeće komponente :

- Webfleet aplikaciju i GPS prijamnik s osnovnim funkcijama
- Webfleet aplikaciju i GPS prijamnik s naprednim funkcijama
- Webfleet aplikaciju, GPS prijamnik s naprednim funkcijama i profesionalni mobilni uređaj s namijenjen za smještaj u vozilo

Varijante sustava s komponentama koje su uključene u paket su prikazane na slici 2.3.


Slika 2.3 TomTom Telematics ponuda sustava za upravljanje voznim parkom

Gps Trackit je još jedno rješenje za upravljanje voznim parkom [8]. Funkcionira na istom principu kao i TomTom sustav. Sastoji se od web i mobilnih aplikacija, GPS uređaja koji se priključuje na vozilo i opcionalnog ekrana za prikaz ruta. Za razliku od prethodnog rješenja, Gps Trackit ima funkciju prepoznavanja neovlaštenog korištenja vozila, što može pomoći i u slučaju krađe vozila kao što je prikazano na slici 2.4. Funkcija radi na principu ograničenja. Korisnik može postaviti ograničenja jedan ili više parametara kretanja vozila (područje, vrijeme, brzinu i sl.), a sustav će poslati obavijest ako je neki od zadanih uvjeta za upozorenje ispunjen.

Gps Insight je još jedno zanimljivo rješenje s korisnim mogućnostima [9]. Princip rada je jednak kao u prethodno opisana dva rješenja. Poseban je po tome što ugrađena kamera omogućava prepoznavanje mogućih događaja:

- pogreške vozača u vožnji
- nagla ubrzanja i kočenja
- prepoznavanje nepoštivanja prometne signalizacije (npr. Prolazak kroz crveno svjetlo)
- nekorištenje sigurnosnog pojasa


Slika 2.4 Prikaz ograničenja kretanja vozila

Osim toga kupac ima na raspolaganju usluge pomoći na cesti i plaćanja goriva. Sva tri opisana rješenja nude korisne mogućnosti pomoću kojih je, osim optimizacije troškova i očuvanja okoliša, moguće povećati sigurnost u prometu.

# 3. Sustav za upravljanje voznim parkom

Sustav za upravljanje voznim parkom izrađen za potrebe ovog diplomskog rada predviđen je za upravljanje malim ili srednje velikim voznim parkom. Sustav se sastoji od:

- mobilne aplikacije za Android uređaje
- baze podataka koja se nalazi u oblaku na internetu
- korisničkog sučelja u obliku web aplikacije

Mobilnu aplikaciju je potrebno instalirati na mobilni uređaj s Android operacijskim sustavom i GPS prijemnikom. Baza podataka je smještena u oblaku na internetu, a dostupna je kao besplatna usluga s mogućnošću proširenja. Korisničko sučelje u obliku web aplikacije instalirano je na poslužitelj, kao i baza podataka.

Prilikom dizajna aplikacije uzeto je u obzir da je sustav namijenjen za klijente koji se tek počinju baviti upravljanjem voznim parkom, stoga im je važno da je sustav jednostavan i da su ulazni troškovi što je moguće niži. Zbog toga je odabrana za razvoj mobilne aplikacije Android platforma koja je najraširenija na tržištu, a ujedno postoji mnogo uređaja vrlo niske cijene koji su kompatibilni sa sustavom. S obzirom na to da postoji samo jedno korisničko sučelje optimizirano za prikaz na svim vrstama uređaja, troškovi izrade sustava su minimizirani. Baza podataka je odabrana tako da početni troškovi budu minimalni, a performanse baze dovoljne za ispravno funkcioniranje sustava, s mogućnošću jednostavne nadogradnje.

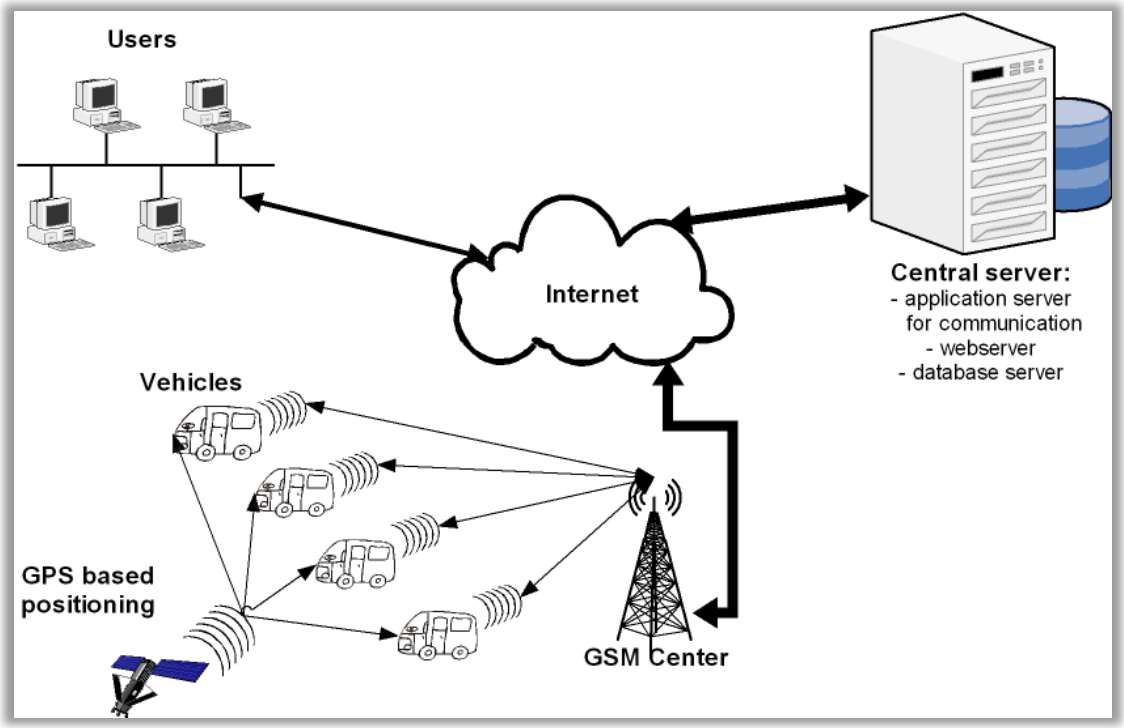
## 3.1. Arhitektura

Arhitekturu sustava izrađenog za potrebe ovog rada prikazuje slika 3.1. Središnji dio arhitekture sustava je centralni poslužitelj (*engl. Central server*) na koji su instalirani web aplikacija i DBMS (*engl. Database Management System*, skraćeno DBMS) koji upravlja bazom podataka. U svakom trenutku je moguće pristupiti podacima iz baze, a na kontrolnoj ploči moguće je pratiti performanse korištenja baze.. Komponente koje sudjeluju u prijenosu informacija su:

- središnji poslužitelj (*engl. Central server*)
- korisnici (*engl. Users*)
- GPS (*engl. Global Positioning System*, skraćeno GPS) odašiljač

- GPS prijemnici koji se nalaze u vozilima
- GSM (*engl. Global System for Mobile Communications*, skraćeno GSM) središte mobilnog operatera (*engl. GSM Center*)

Središnji dio arhitekture sustava je centralni poslužitelj (*engl. Central server*) na koji su instalirani web aplikacija i DBMS (*engl. Database Management System*, skraćeno DBMS) koji upravlja bazom podataka. U svakom trenutku je moguće pristupiti podacima iz baze, a na kontrolnoj ploči moguće je pratiti performanse korištenja baze.




Slika 3.1 Arhitektura sustava

Korisnici tj. osobe zadužene za upravljanje voznim parkom mogu pristupiti korisničkom sučelju u obliku web aplikacije, u bilo koje vrijeme i s bilo kojeg uređaja priključenog na internet. Putem korisničkog sučelja moguće je administrirati vozila, rute, klijente, dobavljače, te ulazne i izlazne račune. Sljedeća vrlo važna komponenta sustava je mobilni uređaj koji ima ulogu GPS prijemnika. Na mobilni je uređaj instalirana mobilna aplikacija. Mobilni uređaj je potrebno odložiti u vozilo koje je sastavni dio voznog parka. Mobilna aplikacija prepoznaje kretanje vozila i u realnom vremenu šalje podatke o lokaciji na poslužitelj koji podatke sprema u bazu podataka. Pojedini podatak o lokaciji sadrži geografsku širinu, geografsku duljinu, trenutnu brzinu i trenutno vrijeme. GPS odašiljač je dio satelitskog radionavigacijskog sustava za određivanje položaja. Sustav čine sateliti koji

kruže u orbitama oko zemlje i kontinuirano šalju radiosignale o trenutnom položaju i vremenu odašiljanja.

Pomoću sustava moguće je odrediti 3 koordinate: geografsku duljinu, geografsku širinu i visinu. GPS prijamnik prima koordinate od GPS odašiljača te ih obrađuje i kreira podatkovnu strukturu u koju sprema primljene koordinate i vrijednosti sa senzora koje mobilni uređaj ima. Za potrebe ovog rada osim koordinata, bilježi se trenutna brzina i trenutno vrijeme, a ne bilježi se geografska visina. GSM središte mobilnog operatera služi za prijenos informacija s mobilnog uređaja internetom na središnji poslužitelj.

## 3.2. Potrebne funkcionalnosti sustava

Svrha ovog rada je bila proizvesti sustav namijenjen tvrtkama koje tek počinju nuditi svojim klijentima uslugu upravljanja voznim parkom. Glavni zahtjevi za sustav su bili ekonomičnost, kratko vrijeme učenja i skalabilnost sustava. Kako bi se zadovoljili glavni zahtjevi, potrebne funkcionalnosti sustava su sljedeće:

- Bilježenje lokacije, brzine i vremena pojedinog vozila
- Prikaz, dodavanje, ažuriranje i brisanje vozila
- Kategorizacija vozila
- Prikaz, trajanje i duljina pojedine rute
- Prikaz, dodavanje, ažuriranje i brisanje korisnika vozila
- Prikaz ukupno prijeđenog puta i ukupnih troškova vozila
- Prikaz, dodavanje, ažuriranje i brisanje klijenata vozila
- Prikaz prihoda klijenata
- Prikaz, dodavanje, ažuriranje i brisanje troškova vozila
- Prikaz, dodavanje, ažuriranje i brisanje dobavljača
- Prikaz rashoda prema dobavljačima
- Prikaz, dodavanje, ažuriranje i naplate usluge

Kako bi sustav počeo raditi, potrebno je na početku unijeti podatke o klijentima i vozilima. U svako vozilo koje je dio voznog parka potrebno je staviti mobilni uređaj s instaliranom mobilnom aplikacijom. Tijekom korištenja vozila, korisnik vozila treba na poslati informaciju za svaki nastali trošak osobi zaduženoj za njegovo vozilo (npr. točenje goriva). Preporuka je poslati sliku računa svakog troška. Odgovorna osoba zatim unosi podatke u sustav. Na temelju unesenih podataka, sustav može prikazati korisne informacije:

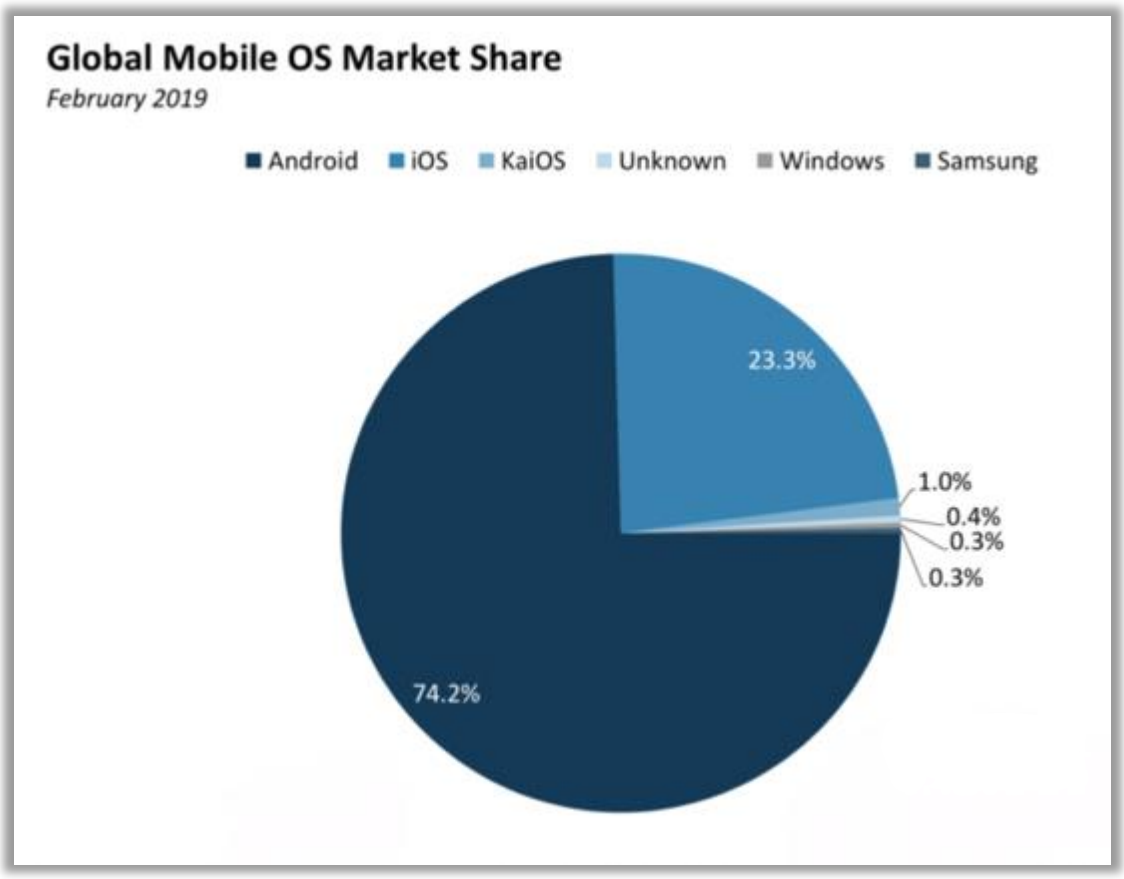
- Tko su najvažniji klijenti tvrtke
- Prema kojim dobavljačima nastaju najveći troškovi
- Koja vozila klijenti najviše koriste
- Koja vozila su ekonomski najisplativija
- Kolika je zarada tvrtke

Navedene stavke su dobar temelj za ulazak u područje djelatnosti upravljanja voznim parkom. Poznato je da u pojednostavljenom slučaju za uspješnost tvrtke presudna razlika između prihoda i rashoda. Ako se uzme u obzir da prihode donose klijenti, a rashodi odlaze dobavljačima, može se zaključiti da na poslovanje tvrtke veliki utjecaj ima odnos prema najvažnijim klijentima i dobavljačima. Na temelju podataka čija će se količina s vremenom povećavati, odgovorna osoba za pojedini vozni park će moći donositi odluke koji će pozitivno utjecati na poslovanje tvrtke. Primjerice, u kojem trenutku je najisplativije kupiti, a u kojem prodati vozilo.

## 3.3. Mobilna aplikacija

Mobilna aplikacija je pisana za Android operacijski sustav. Android operacijski sustav je operacijski sustav otvorenog kôda (*engl. Open source*) namijenjen upotrebi na mobilnim uređajima. Android koji je danas poznat je Google Inc. 2005. godine kupio od tvrtke Android Inc.. Prvi put je predstavljen javnosti 2007. godine, a prvi uređaji koji su koristili Android operacijski sustav bili su dostupni na tržištu 2008. godine. Raspodjela tržišnog udjela operacijskih sustava na mobilnim uređajima prikazana je na slici 3.2 prema kojoj se vidi da je Android najrasprostranjeniji operativni sustav korišten na mobilnim uređajima. Od preuzimanja tvrtke od strane Google-a, razvijeno je 29 nadogradnji sustava, a danas je aktualna 29. verzija pod nazivom Android Pie. Android je odabran za razvojnu platformu zbog mnogobrojnih prednosti:

- Raširenost
- Jednostavna distribucija
- Nije potrebno certificiranje
- Nije potreban postupak odobravanja softvera
- Komunikacija s ugrađenim aplikacijama
- Izjednačavanje u radu s ugrađenim aplikacijama




Slika 3.2 Raspodjela tržišnog udjela operacijskih sustava na mobilnim uređajima 2019.

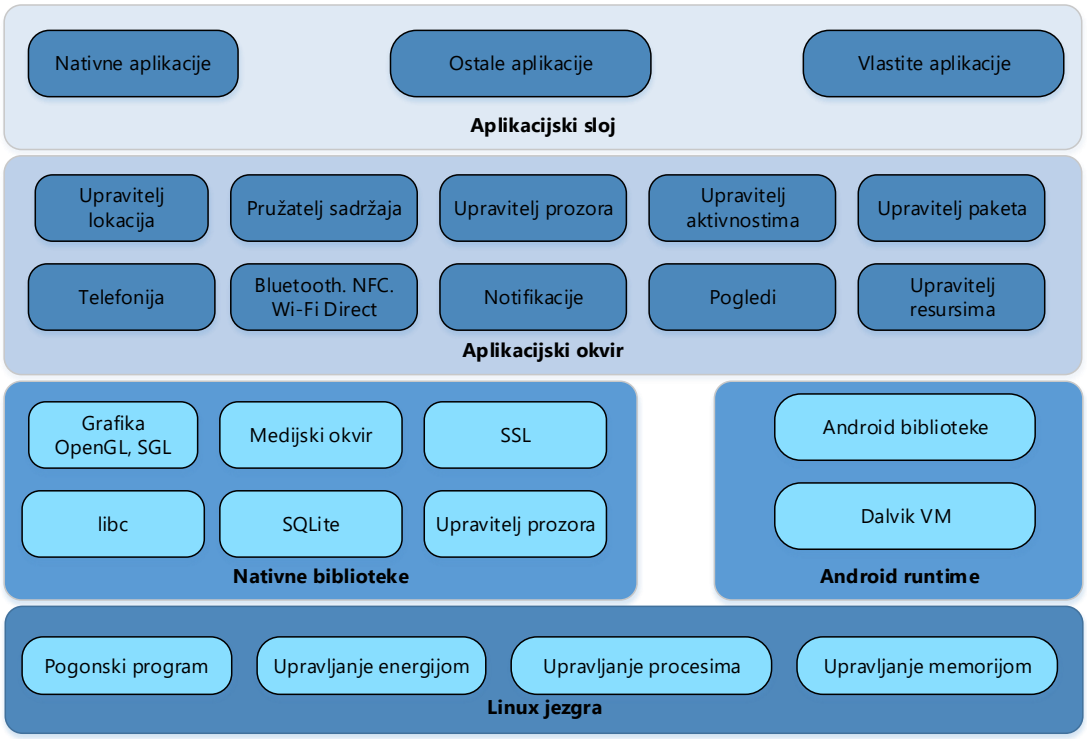
Glavne komponente Android operacijskog sustava prikazane su na slici 3.3:

- Linux jezgra
- Nativne biblioteke
- Android runtime
- Aplikacijski okvir

Android operacijski sustav temelji se na Linux 2.6 jezgri, a napisan je u programskom jeziku C/C++. Linux jezgra sadrži drivere od kojih su najvažniji driver za među procesnu komunikaciju koji služi za razmjenu podataka između različitih procesa ili između niti unutar istog procesa,te driver za upravljanje napajanjem (*engl. Power Management*). Na jezgru su nadograđene nativne biblioteke pisane u C/C++ programskom jeziku:

- Grafika, OpenGL, SGL – koriste se za optimalan prikaz grafičkog sučelja
- Medijski okvir – koristi se za snimanje i reproduciranje audio/video sadržaja
- SSL – koristi se za sigurnu komunikaciju putem interneta

- libc – sistemska C biblioteka prilagođena za ugradbene sustave zasnovane na Linux operacijskom sustavu
- SQLite – koristi se za upravljanje bazama podataka u aplikacijama
- Upravitelj prozora – koristi se upravljanjem prozora unutar sučelja



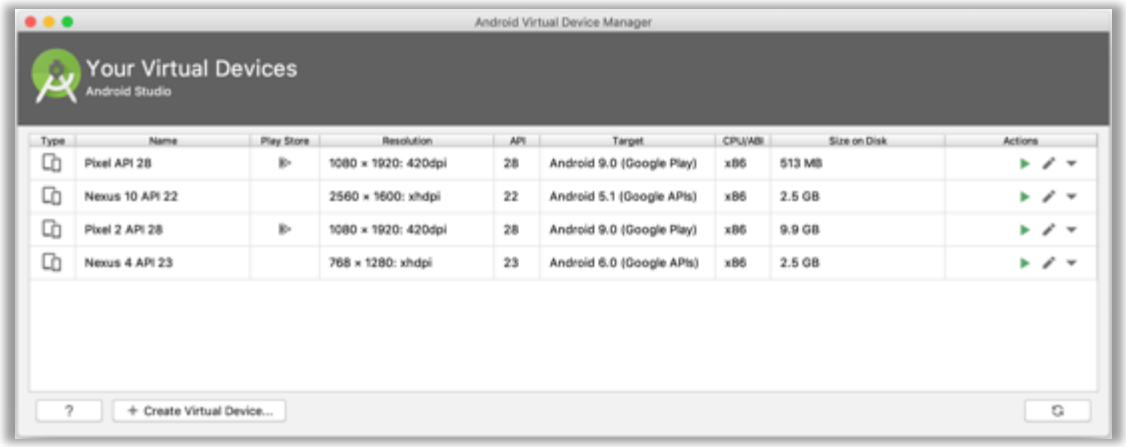

Slika 3.3 Arhitektura Android operacijskog sustava

Android runtime sloj se koristi za pokretanje aplikacija, te se sastoji od jezgrenih biblioteka pisanih u Java programskom jeziku i DVM (*engl. Dalvik Virtual Machine*, skraćeno DVM). Jezgrene biblioteke osiguravaju rad sa stringovima, datotečni sustav i pristup mreži. DVM optimizira trajanje baterije, regulira procesorsku moć i licenciranje. DVM pokreće aplikacije kao zasebne procese, instance virtualnog stroja. DVM pretvara Java datoteke u vlastiti format (.dex) koji je optimiziran za izvođenje uz minimalni utrošak energije.

Aplikacijski okvir (*engl. Application Framework*) je sljedeća nadogradnja prema shemi koju prikazuje slika 3.3. Aplikacijski okvir putem API-a (*engl. Application Programming Interface*, skraćeno API) omogućava upravljanje programskim paketima. Tako sustav upravlja aktivnostima aplikacije, prozorima, resursima, podacima lokacije korisnika, prikazom obavijesti, razmjenjuje podatke između aplikacija i dr. Na samom vrhu arhitekture nalazi se aplikacijski sloj vidljiv korisniku. U aplikacijskom sloju nalaze se ugrađene aplikacije u operacijski sustav poput e-mail klijenta, sata, SMS programa,

kalendara i sl. Osim ugrađenih aplikacija, u aplikacijskom sloju nalaze se i aplikacije koje sam korisnik može instalirati. Korisnik aplikacije može instalirati putem direktnog spremanja datoteke na uređaj ili ih može preuzeti u Google trgovini, ugrađenoj aplikaciji koja je instalirana na svaki Android uređaj.

Klijentska aplikacija za upravljanje voznim parkom je razvijena u Android Studio programskom okruženju (*engl. Integrated Development Environment*, skraćeno IDE). Android studio je odabran zbog toga što je jedini službeno odobren softver za razvoj aplikacija za android operacijski sustav od strane Google-a. Android studio ima velike prednosti: opširnu dokumentaciju, ugrađen emulator za mobilne uređaje (*engl. Android Virtual Devices,* skraćeno AVD) te velik broj dostupnih dodataka (*engl. plugin*) koji pomažu u razvoju aplikacije. Da bi se aplikacija mogla pokrenuti, u Android studio je ugrađen Gradle alat koji optimizira napisani kôd u .apk datoteku prikladnu za izvođenje na mobilnom uređaju. Slika 3.4 prikazuje AVD Manager koji služi za kreiranje uređaja na kojemu će se simulirati aplikacija. U izborniku može izabrati neke od predefiniranih uređaja koji postoje na tržištu, a može i sam kreirati uređaj sa određenom količinom memorije, veličinom i rezolucijom ekrana te verzijom operacijskog sustava. Na taj način se može približiti stvarnim uvjetima na pojedinom uređaju, a može i testirati aplikaciju na više uređaja. AVD manager nudi i mogućnost simulacije aplikacija razvijenih ta televizore i pametne satove.




Slika 3.4 AVD Manager

Gradle alat za kreiranje izvršne datoteke je još jedna važna komponenta koju sadrži Android studio. Programer u build.gradle datoteci konfigurira minimalnu, najveću i predviđenu SDK (*engl. Software Development Kit*, skraćeno SDK) verziju na kojoj će se aplikacija izvoditi. Gradle na temelju zavisnosti (*engl. dependencies*) dohvaća udaljene

module s dodatnim funkcionalnostima za programiranje. Kod izrade aplikacije korišten je najrasprostranjeniji Maven Repsitory. Primjer kôda iz gradle datoteke prikazuje kôd 3.1.

```
dependencies {
    api 'com.google.android.material:material:1.1.0-alpha09'
}
```

Kôd 3.1 Definiranje zavisnosti

Material design modul dohvaćen je kôd 3.1, a koristi se za dohvaćanje stiliziranih elemenata korisničkog sučelja poput kontrole gumba, polja za unos teksta i sl.

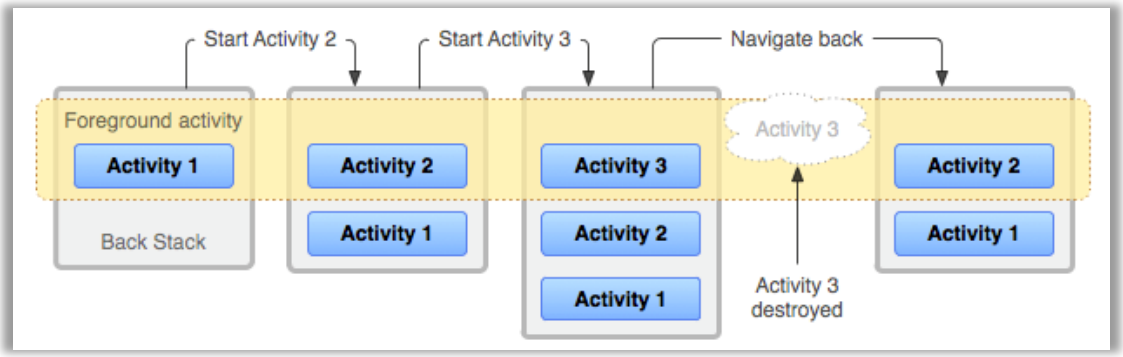
### 3.3.1. Arhitektura

Arhitekturu aplikacije za upravljanje voznim parkom prikazuje slika 3.5. Za izradu aplikacije korištene su sljedeće komponente: aktivnosti (*engl. Activities*), prijemnici emitiranja (*engl. Broadcast receivers*), intenti (*engl. Intents*), obavijesti (*engl. Notifications*) i servisi (*engl. Services*).


Slika 3.5 Arhitektura aplikacije

Aktivnost je klasa koja predstavlja ulaz u aplikaciju. Aplikacija može imati više aktivnosti. Svaka aktivnost je neovisna o drugoj i ima svoj životni ciklus. Svaka kreirana aktivnost dodaje se na stog [4] (*engl. back stack*) kao što je prikazano na slici 3.6. Nakon što

aktivnost završi, briše se sa stoga i program nastavlja s prethodnom aktivnosti. Aplikacija za upravljanje voznim parkom ima 7 aktivnosti koje će biti opisane u nastavku. MainActivity je aktivnost koja se aktivira pri pokretanja aplikacije. Koristi se za tijek izvođenja aplikacije. Pokretanjem aplikacije prvo se provjeravaju dopuštenja (*engl. permissions*). Poziva se GpsPermissionActivity aktivnost koja provjerava dali je omogućeno dopuštenje za pristup lokaciji uređaja. Navedeno dopuštenje programer mora sam omogućiti prilikom programiranja aplikacije. Ako dopuštenje nije omogućeno, aplikacija prestaje s radom. Nakon provjere dopuštenja za pristup lokaciji, MainActivity poziva GpsPermissionActivity koji služi za provjeru dali je omogućen GPS prijemnik. Ako nije, aplikacija će putem skočnog prozora (*engl. Popup*) zatražiti da korisnik omogući GPS prijemnik. Ako korisnik ne omogući Gps prijemnik aplikacija završava, u suprotnom se poziva aktivnost LoginActivity.



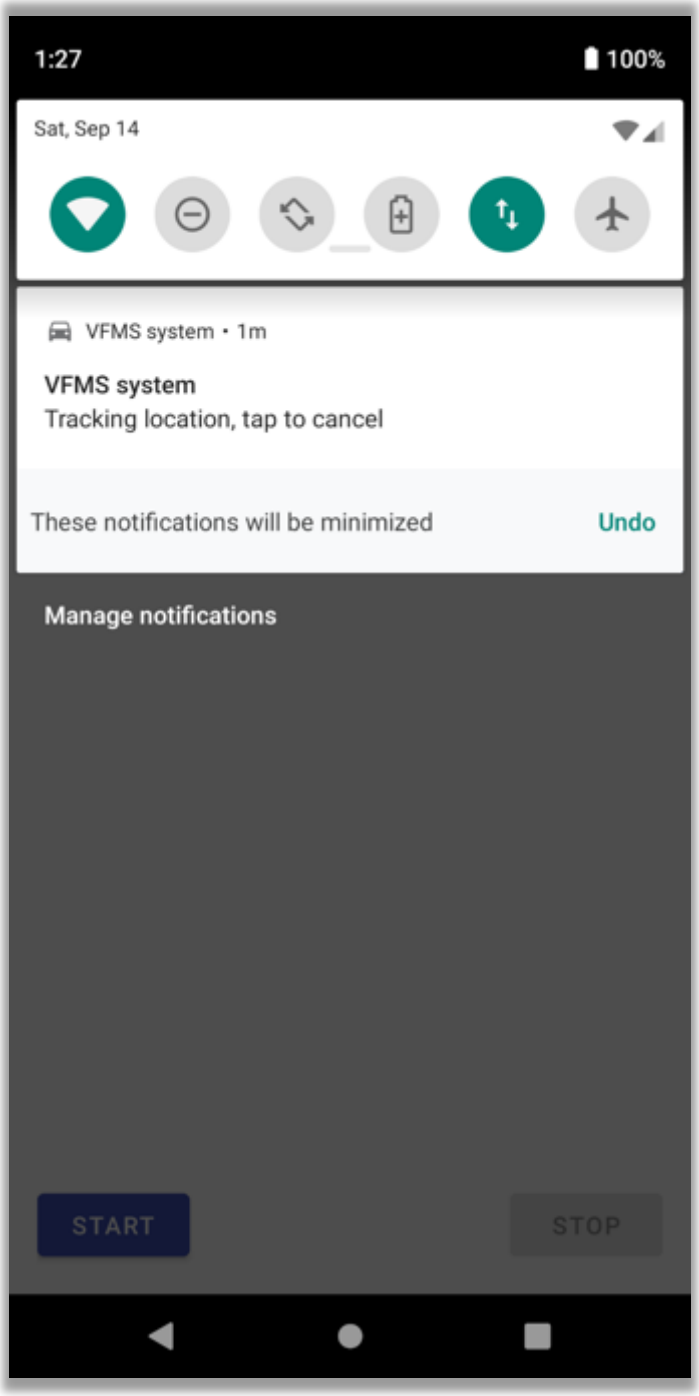

Slika 3.6 Aktivnosti i stog

LoginActivity se koristi za prijavu na Firebase bazu podataka koja se nalazi na poslužitelju. Podaci se iz baze podataka dohvaćaju u JSON (*engl. JavaSctipt Object Notation*, skraćeno JSON) obliku. Metoda *loginUserAccount* unutar aktivnosti šalje podatke za prijavu, korisničko ime i lozinku na Firebase poslužitelj, te odgovor poslužitelja vraća u MainActivity. Ako je prijava na bazu uspješna, pokreće se aktivnost ActivityRecognitionActivity koja pomoću servisa dohvaća trenutnu aktivnost. Servisi (*engl. services*) su komponente koje se koriste za obavljanje nekog trajnog zadatka u pozadini aplikacije.

Kada je prepoznata aktivnost „In Vehicle“ ActivityRecognitionActivity aktivnost poziva servis BackgroundDetectedActivitiesService koji radi u pozadini i upravlja sa servisom DetectedActivitiesIntentService. Zatim DetectedActivitiesIntentService servis prepoznaje i emitira (*engl. broadcast*) tip aktivnosti koji se trenutno događa. Prijemnici

emitiranja (*engl. Broadcast receivers*) primaju emitirane poruke od strane operacijskog sustava ili neke druge aplikacije.

BackgroundDetectedActivitiesService putem komponente prijemnik emitiranja prima trenutnu vrstu aktivnosti. Ako je primljena aktivnost naziva „In Vehicle“ pokreće servis TrackerService koji šalje podatke o trenutnoj lokaciji u bazu podataka. Kada prestane aktivnost „In Vehicle“, prestaje slanje lokacijskih podataka u bazu.




Slika 3.7 Prikaz obavijesti na zaslonu

Obavijesti (engl. notifications) komponente se koriste za obavještavanje korisnika o nekom događaju. Dok je TrackerService servis aktivan, obaveza je programera obavijestiti korisnika aplikacije da je servis aktivan. To se u aplikaciji koja je predmet ovog rada postiže prikazom ikone automobila na vrhu zaslona kao što je prikazano na slici 3.7.

Za pozivanje aktivnosti i servisa, te njihovu međusobnu komunikaciju koriste se intenti. Intent je još jedna neophodna komponenta svake Android aplikacije. Pomoću intenta možemo proslijediti vrijednost u servis ili aktivnost koju pozivamo, ili primiti vrijednost od pozvanog servisa ili aktivnosti. Upravo na taj način i funkcionira mehanizam koji kontrolira tok izvođenja ove aplikacije.

Osim navedenih osnovnih komponenti, za rad aplikacije neophodne su još pogledi (*engl. views*), raspored (*engl. layout*) koji su dio resursa (*engl. resources*) i .xml (*engl. Extensible Markup Language*, skraćeno xml) datoteka za konfiguraciju Manifest. Pogledi su neophodan dio svake Android aplikacije koja ima grafičko sučelje. Korišteni su TextView za prikaz teksta, EditText za uređivanje teksta, ProgressBar za animiranje dok se čeka odgovor aplikacije, ImageView za prikaz slika i Button za prikaz gumba. Raspored se koristi za smještaj pogleda na željenu poziciju na ekranu.

U aplikaciji je korišten LinearLayout raspored koji omogućava odgovarajući raspored pogleda na ekranu. Android Manifest datoteka sadrži konfiguraciju aplikacije. U ovom slučaju sadrži dozvole za korištenje servisa za prepoznavanje aktivnosti, pristup lokaciji, pristup internetu i aktiviranje servisa u prvom planu (*engl. Foreground service*). Datoteka sadrži podešenja za prikaz naziva aplikacije, ikone za pokretanje aplikacije (engl. Launcher), teme koje određuju boje i vrstu slova i definiciju svih aktivnosti koje se koriste u aplikaciji.

Za izgradnju aplikacije korišten je sustav za izgradnju aplikacije (*engl. Automated build toolkit*) Gradle. Gradle omogućuje postavljanje zavisnosti (*engl. dependencies*), brzu konfiguraciju, testiranje, potpisivanje i generiranje više APK datoteka (*engl. Android Package*, skraćeno APK). Gradle konfiguracijska datoteka sadrži podatke o minimalnoj (postavljeno na 26) i ciljanoj razini API-a (postavljeno na 29) na kojoj se može izvršavati aplikacija. Moguće je još podesiti i maksimalnu razinu API-a na kojoj se može izvršavati aplikacija. Datoteka također sadrži razinu API-a na kojoj se razvija aplikacija. Osim navedenih podešenja, važan dio Gradle datoteke su zavisnosti koje omogućuju korištenje vanjskih API-a. Za razvoj aplikacije korištene su zavisnosti za firebase prijavu, bazu podataka i zavisnost za Google play servis.

### 3.3.2. Funkcionalnost

Svrha mobilne aplikacije sustava za upravljanje voznim parkom je dinamičko spremanje podataka o trenutnoj lokaciji i brzini vozila. Osim podataka lokacije i brzine, trenutno vrijeme se također sprema u bazu podataka. Nakon što korisnik pokrene aplikaciju, napredovanjem kroz aplikaciju, pojavljivat će se prikazi na ekranu kao što prikazuje slika 3.8, redom s lijeva na desno. Na početku se otvara skočni prozor koji korisnika traži dopuštenje pristupa lokaciji mobilnog uređaja. Ako korisnik na ekranu pritisne „DENY“ pojavljuje se sljedeći ekran na kojemu je isključivo moguće izaći iz aplikacije. Ako korisnik pritisne na ekranu „ALLOW“ aplikacija napreduje dalje, te se pojavljuje ekran za prijavu. Na ekranu za prijavu nalazi se logo tvrtke, naziv aplikacije, polja za unos korisničkog imena i lozinke te tipke „LOGIN“ za ulaz u aplikaciju i „EXIT“ izlaz iz aplikacije. Ako korisnik pogrešno unese korisničko ime ili lozinku, pojavljuje se odgovarajuća poruka. Nakon 3 pogrešna unosa pojavljuje se ekran na kojemu je isključivo moguće izaći iz aplikacije. Dok traje provjera korisničkog imena i lozinke, na ekranu se prikazuje kružna animacija (*engl. spinner*) koja označava da je proces provjere u tijeku.

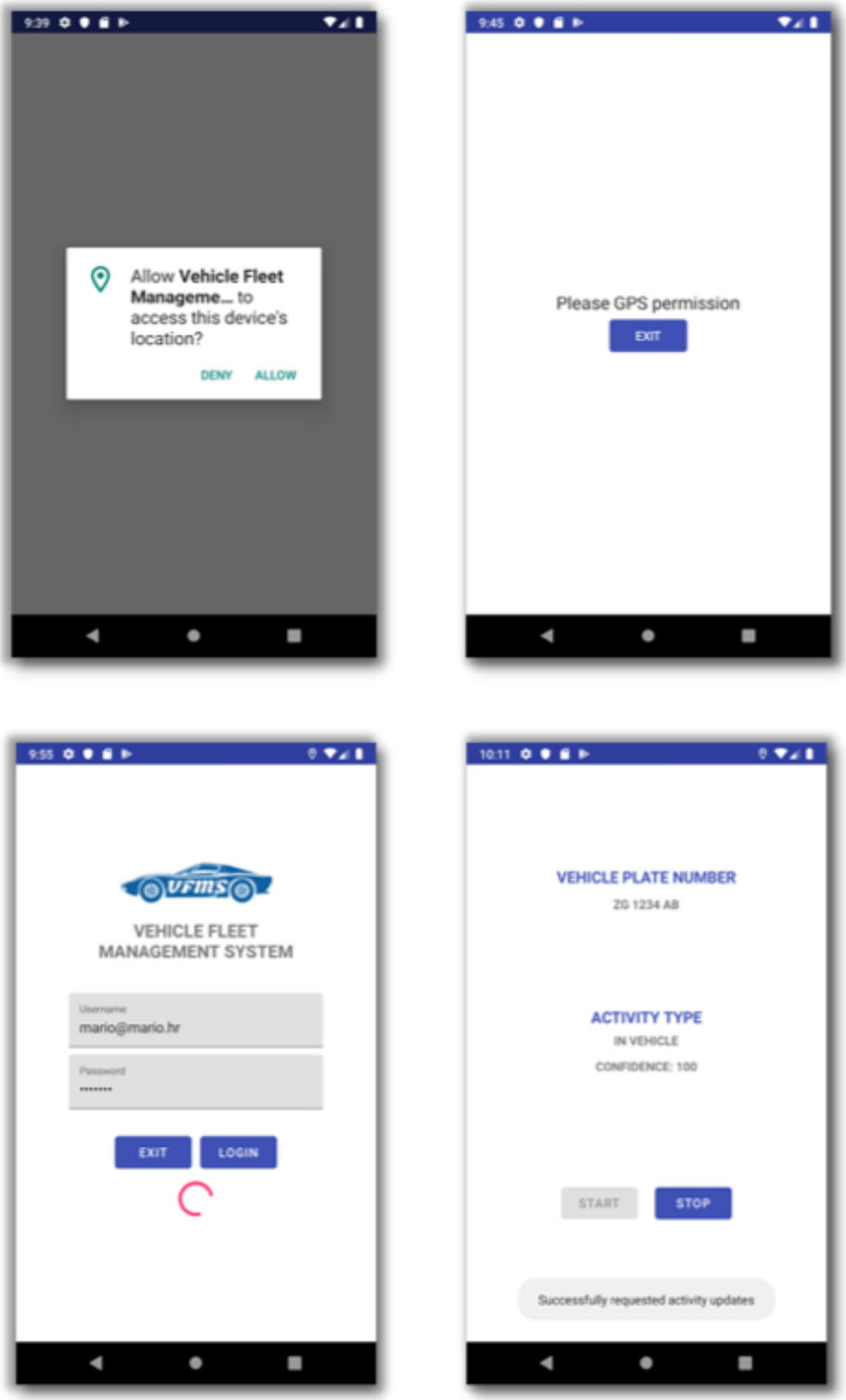
Ako su korisničko ime i lozinka točni, pojavljuje se sljedeći ekran na kojemu su prikazani:

- registarska oznaka vozila
- vrsta prepoznate aktivnosti
- pouzdanost prepoznate aktivnosti
- kontrole za početak i prestanak spremanja podataka u bazu

Registracijska oznaka je jedinstvena identifikacijska oznaka za vozilo koju postavlja administrator u web sučelju. Za jedinstveni identifikator se može upotrijebiti i broj šasije vozila. Vrsta prepoznate aktivnosti se pojavljuje promjenom stanja vozila. Aplikacija može prepoznati kad je motor vozila upaljen, tada aktivira bilježenje podataka u bazu, pod uvjetom da je korisnik na početku stisnuo start. Kada je vozilo upaljeno, na ekranu se ispisuje vrsta aktivnosti „IN VEHICLE“. Kada je vozilo ugašeno, vrsta aktivnosti je „STILL“, a spremanje podataka u bazu je prekinuto. Pouzdanost prepoznate aktivnosti je vjerojatnost da je prikazana aktivnost točna.

Aplikacija je podešena tako da prepoznaje aktivnosti s vjerojatnošću većom od 85%. Kontrolama „START“ i „STOP“ na dnu ekrana upravljanja se pokretanjem i zaustavljanjem servisa za prepoznavanje aktivnosti. Ako je servis već pokrenut, kontrola „START“ je onemogućena, a samo kontrola „STOP“ omogućena i obrnuto. Na početku je nužno da

korisnik pokrene praćenje aktivnosti. Nakon pokretanja, korisnik može izaći iz aplikacije i koristiti druge aplikacije. Dok je servis za spremanje podataka aktivan, na vrhu ekrana će biti oznaka prikazana na Slika 3.7. Ako korisnik želi prekinuti izvođenje aplikacije, potrebno je ući u obavijesti i kliknuti na oznaku sa Slika 3.7.




Slika 3.8 Navigacija kroz aplikaciju

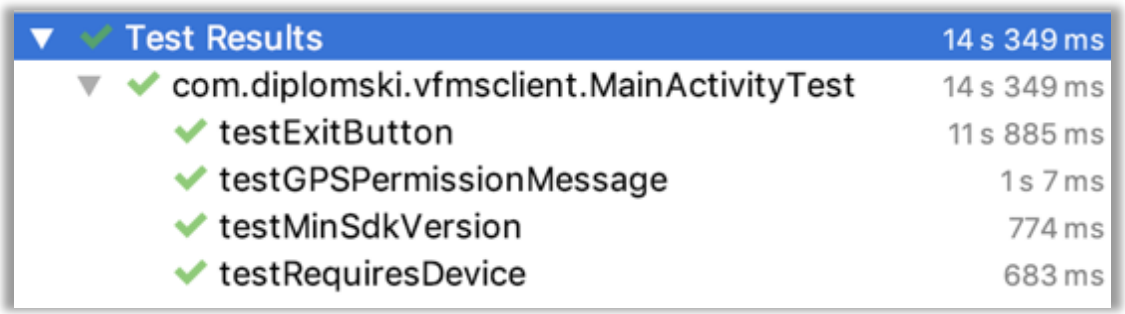
### 3.3.3. Testiranje

Aplikacija je testirana pomoću JUnit4 okvira (*engl. framework*). JUnit4 je programski okvir otvorenog kôda (*engl. open source*) koji se koristi za testiranje aplikacija. Glavna prednost aplikacija koje imaju napisane testove je da se mogu raditi unapređenja kôda bez mogućnosti da će aplikacija prestati raditi ako je napravljen propust kod dodavanja nove funkcionalnosti ili neke druge izmjene. Preduvjet za to je da testovi budu kvalitetno napisani. Kvalitetno napisani testovi obuhvaćaju sve moguće slučajeve koji se mogu dogoditi pri izvođenju aplikacije. S obzirom na to da je broj mogućnosti beskonačan, za kvalitetno pisanje testova je potrebno mnogo znanja i iskustva. Na aplikaciji u ovom radu je provedeno testiranje kontrola korisničkog sučelja kako bi se potvrdilo aplikacija radi kako je predviđeno.

Budući da se kontrole sučelja nalaze u aktivnostima, bit će testirane aktivnosti koje sadrže pojedine kontrole:

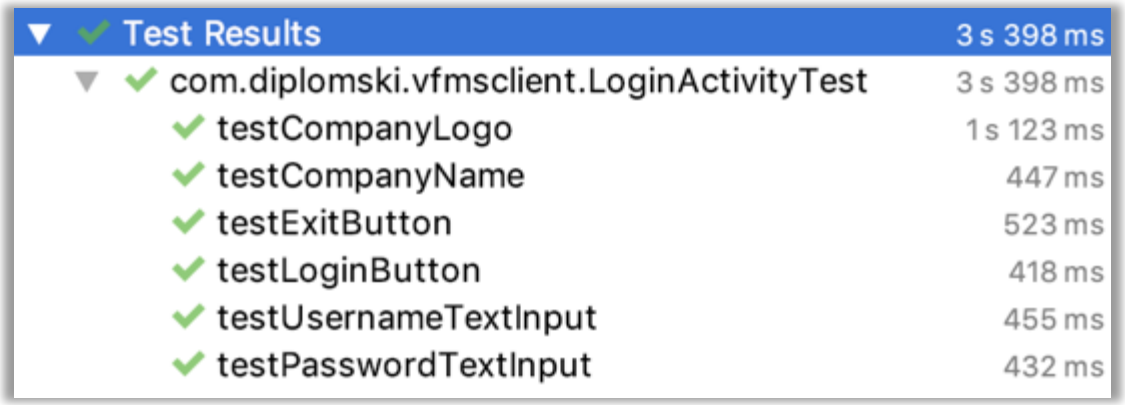
- MainActivity
- LoginActivity
- ActivityRecognitionActivity

Rezultate testova klase MainActivity prikazuje slika 3.9. Provjerava se da li su prisutne kontrola za izlaz iz aplikacije (testExitButton) i da li se pojavila poruka za izlaz iz aplikacije (testGPSPermissionMessage). Osim navedenog, provjerava se ispravnost minimalne SDK (*engl. software development kit*, skraćeno SDK) verzije.




Slika 3.9 Rezultati testova klase MainActivity

Slika 3.10 prikazuje rezultate testova klase LoginActivity. Testira se da li su prikazane kontrole za: prikaz loga tvrtke (testCompanyLogo), naziva tvrtke (testCompanyName), za unos korisničkog imena (testUsernameTextInput) i lozinke (testPasswordTextInput) te gumba za izlaz iz aplikacije (testExitButton) i nastavak (testLoginButton).




Slika 3.10 Rezultati testova klase LoginActivity

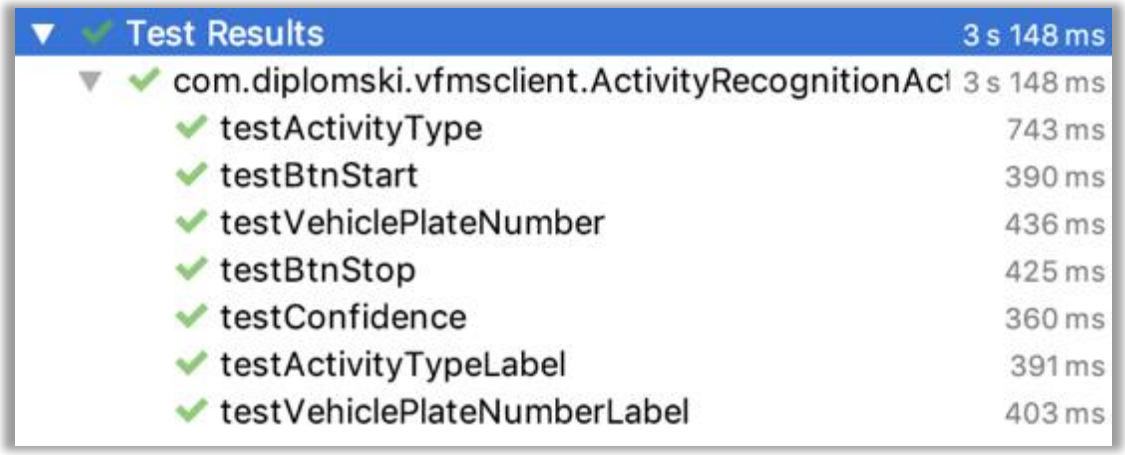
Kôd 3.2 prikazuje primjer testa iz klase „LoginActivityTest“:

```
@Test
public void testUsernameTextInput() {
    Activity activity = activityTestRule.getActivity();

assertNotNull(activity.findViewById(R.id.username_text_input)
);
    TextView username =
activity.findViewById(R.id.username_text_input);
    assertTrue(username.isShown());
}
```

Kôd 3.2 Test kontrole unosa korisničkog imena

Pokretanjem testa dohvaća se aktivnost koja sadrži testiranu kontrolu. Zatim se dohvaća referenca kontrole, te se na kraju provjerava da li je navedena kontrola prikazana na ekranu korisnika. ActivityRecognitionActivity klasa sadrži kontrole za pokretanje (testBtnStart) i zaustavljanje aktivnosti (testBtnStop), prikaz naziva i vrijednost registarske oznake (testVehiclePlateNumberLabel, testVehiclePlateNumber), prikaz pouzdanosti i tipa aktivnosti (testConfidence, testActivityType). Rezultati testova prikazani su na slici 3.11. Osim navedenih testova, aplikacija je testirana i u praksi. Instalirana je na mobilni uređaj koji je korišten u vozilu 300 kilometara, kako bi se potvrdilo da radi kako je predviđeno. Koliko je bilo moguće, simulirane su vožnja na autoputu, otvorenoj cesti i vožnja u gradu. Posebno je obraćena pažnja na simuliranje uvjeta na cesti u gradskoj gužvi.




Slika 3.11 Rezultati testova klase ActivityRecognitionActivity

## 3.4. Internet aplikacija

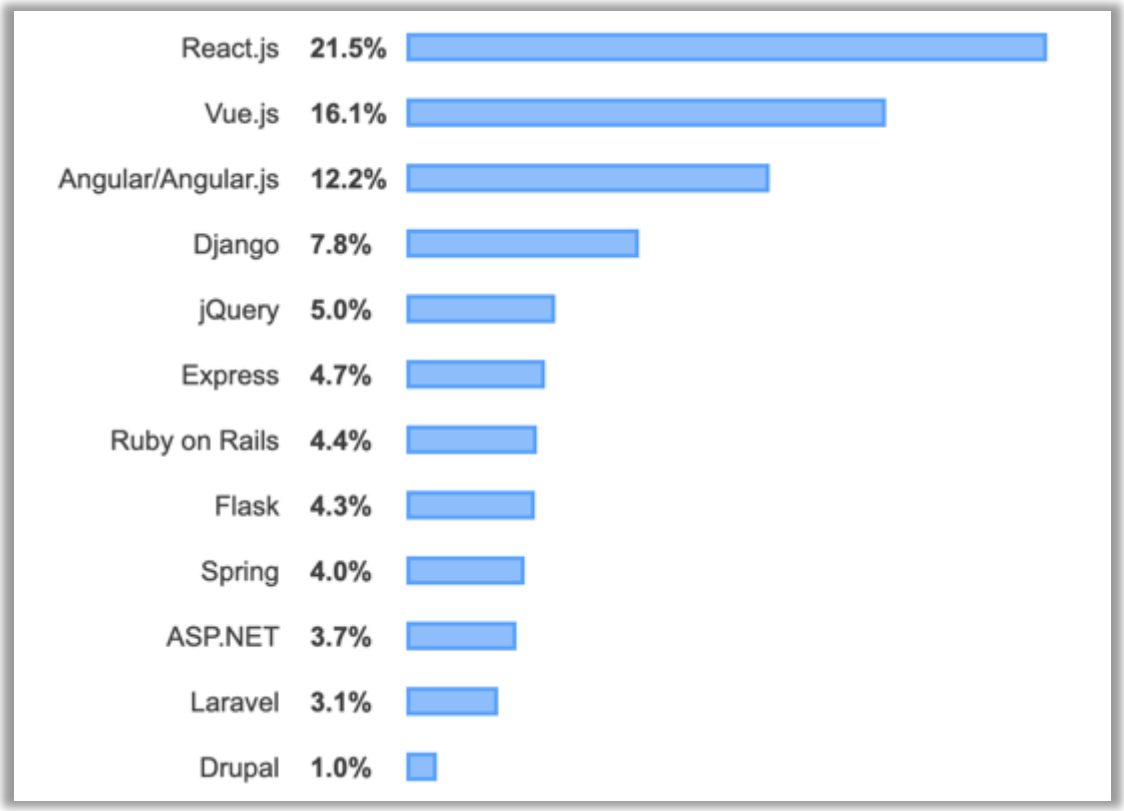
Internet aplikacija pisana je JavaScript programskim jezikom. JavaScript je objektno orijentirani skriptni jezik. Koristi se u kombinaciji s HTML-om. JavaScript kôd komunicira s HTML DOM elementima s pomoću kojih dinamički upravlja Internet stranicom. Može se upravljati svim elementima, poput ponašanja kontole gumba, validacije teksta unesenog u polje i sl. Osim navedenog s JavaScript kôdom se mogu asinkrono učitavati podaci s poslužitelja bez osvježavanja stranice. Glavne prednosti JavaScript jezika su:

- Brzina – kôd se izvršava odmah bez čekanja odgovora s poslužitelja
- Jednostavnost – kôd je jednostavan za pisanje, a sam jezik se brzo uči
- Opterećenje poslužitelja – budući da je dio logike na klijentskoj strani, rasterećuje poslužiteljsku stranu

JavaScript skriptni jezik se može koristiti za programiranje na strani poslužitelja (*engl. server-side scripting*) ili kao u ovom slučaju za programiranje na strani klijenta (*engl. client side scripting*) u kojemu se kôd izvršava u Internet pregledniku. JavaScript kôd se može pisati u bilo kojem tekst editoru poput Notepad-a. Postoje i alati poput Visual Studio Code koji nude dodatne funkcionalnosti koje ubrzavaju pisanje kôda, rad na kôdu u timovima i verzioniranje na servisima poput GitHub-a. Osim navedenog postoje i biblioteke koje dodatno ubrzavaju razvoj aplikacija.

U ovom radu korištena je biblioteka ReactJS za razvoj korisničkog sučelja. ReactJS je nastao suradnjom Facebook-a i programerske zajednice. Prema rezultatima ankete Stackoverflow zajednicekoju prikazuje slika 3.12, ReactJS je najtraženija tehnologija za

razvoj web aplikacija. Razvoj aplikacija pomoću ReactJS-a zadržava sve karakteristike i mogućnosti koje pruža programski jezik JavaScript.




Slika 3.12 Anketa najtraženiji programski okvir za razvoj web aplikacija

Prednosti razvoja aplikacija pomoću ReactJS-a:

- razvoj aplikacija se temelji na komponentama
- jednosmjeran tok podataka
- omogućava korištenje dodataka (*engl. plugins*)
- kompatibilan je s mobilnim uređajima
- prikladan je za korištenje u timovima
- jednostavan je za testiranje

Osim navedenih prednosti, ReactJS karakterizira i nekoliko nedostataka:

- kompleksna dokumentacija
- nije podržan od strane svih web preglednika
- nedostatak proširenja

Komponente su osnova izrade aplikacije pomoću ReactJS-a. Programer može razviti komponentu sa željenim funkcionalnostima koju kasnije može upotrijebiti na više mjesta.

Primjerice vrlo jednostavno se može programirati animirani gumb (*engl. Button*) s određenom veličinom. Tok podataka je uvijek od komponenti roditelja (*engl. parent components*) prema djeci (*engl. child components*). Komponente roditelji mogu prema potrebi proslijediti potrebne parametre komponentama djeci. Parametri mogu biti standardni JavaScript tipovi podataka ili funkcije.

ReactJS korištenjem npm biblioteke ima mogućnost nadogradnje širokim rasponom dodataka. Npm je najveća biblioteka na svijetu za nadogradnju. Tako se ReactJS može proširiti novim komponentama (npr. za izradu grafova), mogu se dodati i nove funkcionalnosti tipa web routing. Prednost je što programeri mogu izraditi dodatne funkcionalnosti ili komponente, te ih snimiti na npm poslužitelj. Za razvoj aplikacija za mobilne uređaje postoji inačica React Native koja se koristi na isti način kao ReactJS. To je velika je prednost jer je potrebno da programer poznaje samo jednu tehnologiju, što znatno smanjuje troškove razvoja aplikacije. Timovi mogu uštedjeti mnogo vremena korištenjem ReactJS-a. S obzirom na to da se razvoj temelji na komponentama, razvoj pojedinih komponenata može se podijeliti na članove tima, a komponente koje su članovi tima razvili mogu lako međusobno dijeliti i povezati u aplikaciju. Za testiranje komponenti preporučuje se JEST programski okvir za testiranje. Kreiran je kao i ReactJS od strane programera Facebook-a. Da bi se koristio JEST nije potrebna dodatna konfiguracija, ima mogućnost trenutnog prikaza stanja (*engl. snapshot*) , a testovi se zbog povećanja performansi odvijaju u odvojenoj niti (*engl. thread*). Za JEST je dostupna kvalitetna dokumentacija što programerima olakšava rad i povećava kvalitetu testova.

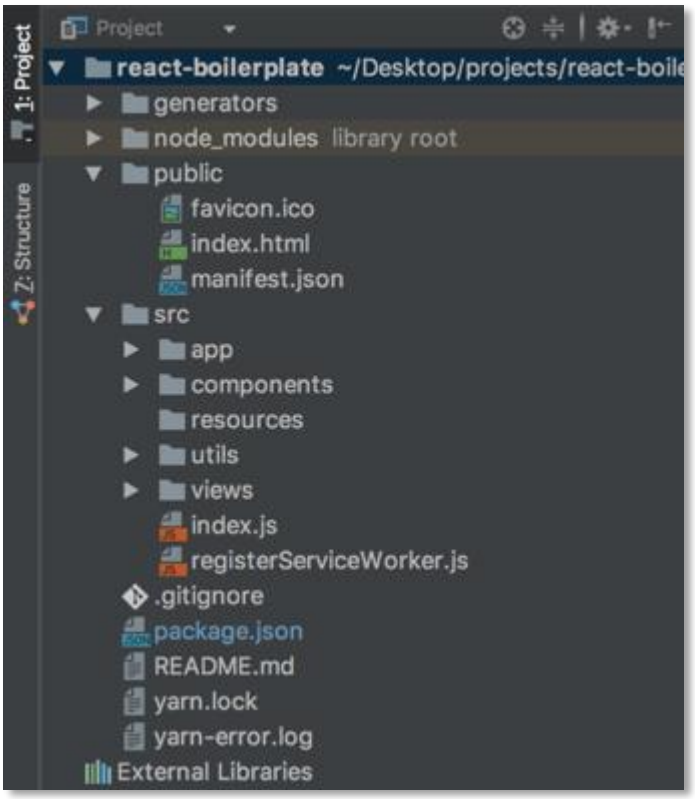
Unatoč prednostima i mogućnostima ReactJS biblioteke, dokumentacija koja se može pronaći na službenim stranicama na slovi kao kompleksna. Na internetu je dostupno mnogo neslužbene dokumentacije, ali je potrebno uložiti dodatno vrijeme za traženje i strukturiranje znanja koje programer traži. To je posljedica činjenice da je ReactJS relativno mlada biblioteka, stoga će taj problem u budućnosti vjerojatno nestati. Osim kompleksne dokumentacije, ReactJS se ne izvršava na svim preglednicima bez dodatnih plugin-ova. Količina dodatnih proširenja za ReactJS je također ograničena, zbog činjenice da je ReactJS relativno nedavno nastao.

### 3.4.1. Arhitektura

Internet aplikacija je kreirana pomoću alata baziranom na NodeJS „create-react-app“ koji je razvio Facebook. Tako se automatski konfigurira ReactJS projekt koji se pokreće na

lokalnoj adresi: `http://localhost:3000`. Strukturu kreiranog projekta prikazuje slika 3.13. Postoje 3 ključna direktorija za aplikaciju:

- node_modules
- public
- src




Slika 3.13 Struktura projekta kreiranog alatom create-react-app

Node modules direktorij sadrži biblioteke preuzete s npm poslužitelja koje nadograđuju aplikaciju. Prije korištenja pojedine biblioteke programer treba na početku JavaScript datoteke pojedine komponente napraviti import kako bi mogao koristiti komponente biblioteke. Public direktorij sadrži temeljni HTML (*engl. HyperText Markup Language*, skraćeno HTML) dokument gdje će se ReactJS aplikacija renderirati. U ovom direktoriju se nalaze CDN (*engl. content delivery network*, skraćeno CDN) poveznice za uvoz fontova, ikona i dr. U public direktoriju nalazi se i datoteka manifest.json koja služi za konfiguraciju poput naziva aplikacije, ikone, teme za prikaz boja i dr. Src direktorij sadrži elemente koji su temeljni dio aplikacije. Tu se nalaze direktoriji: app, components, resources, utils i views koji će biti opisani u nastavku. Direktorij app sadrži datoteke koje zajedno čine konfiguraciju aplikacije:

- index.js se izvršava ulaskom u aplikaciju. Najčešće se koristi za konfiguraciju sustava za upravljanje stanjem aplikacije (u ovom slučaju redux koji će biti opisan u daljem tekstu) i primjenu komponente za usmjeravanje (router-a).
- reducers.js sadrži tzv. reducere (*engl. reducers*) za svaku vizualnu komponentu sa svrhom da stanje pojedine komponente mapira na globalno stanje aplikacije
- store.js sadrži osnovnu konfiguraciju za redux komponentu
- routes.js sadrži rute na koje će vizualne komponente biti mapirane tj. na adrese na kojima će biti dostupne. Npr. komponenta početna stranica „Home“ će biti dostupna na adresi `domena/home`.
- app.css sadrži globalnu css (engl. Cascade Style Sheet, skraćeno CSS) konfiguraciju za vrstu slova, veličinu slova, za zaglavlje, podnožje stranice i dr.
- config direktorij sadrži konfiguracije poput adrese na kojoj dostupan poslužitelj i brzine osvježavanja podataka.

Components direktorij sadrži .jsx datoteke koje predstavljaju komponente koje se prikazuju korisniku. Primjer komponente za prikaz teksta prikazuje kôd 3.3 :

```
import React from 'react';
import PropTypes from 'prop-types';
import Typography from '@material-ui/core/Typography';

export default function Title(props) {
  return (
    <Typography align="center" component="h2" variant="h6"
color="primary" gutterBottom>
      {props.children}
    </Typography>
  );
}
Title.propTypes = {
  children: PropTypes.node,
};
```

Kôd 3.3 Komponenta .jsx za prikaz teksta

Na vrhu datoteke uvezene su vanjske biblioteke (koje su instalirane putem npm biblioteke) 'react' koja se koristi za prepoznavanje .jsx kôda, 'prop-types' za kontrolu tipa podataka i 'Typography' komponenta koja prikazuje tekst. Na samoj komponenti 'Typography' korištenjem atributa, analogno HTML kôdu postavljaju se predefinirana

podešenja za smještaj komponente (align=“center“) i druga podešenja prema potrebi. Navedena komponenta je dio MaterialUI biblioteke u čijoj dokumentaciji se nalaze sva moguća podešenja za pojedinu komponentu. Primjer poziva komponente prikazuje kôd 3.4:

```
<Typography
      className={classes.heading}>
      Expansion Panel 1
</Typography>
```

Kôd 3.4 Korištenje komponente Typography

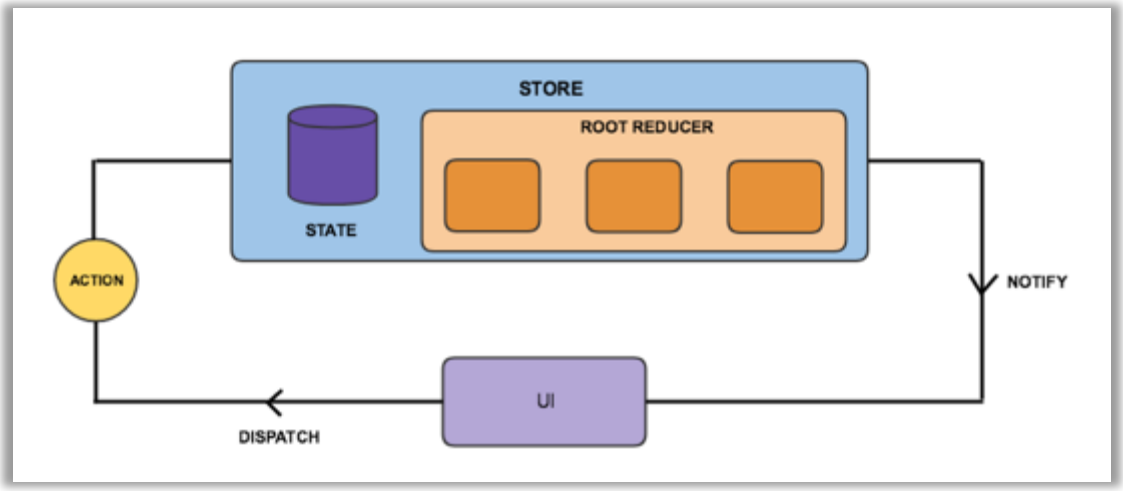
Komponenta će ispisati text „Expansion Panel 1“ na način podešen u .jsx datoteci komponente. U Resources direktorij smještene su ikone, fotografije i drugi najčešće vizualni elementi koji nisu dio npm biblioteke. Utils direktorij sadrži pomoćne datoteke u kojima se nalaze funkcije koje izvršavaju logiku koja je potrebna za rad aplikacije. Primjer iz aplikacije za upravljanje voznim parkom je funkcija za ažuriranje atributa pojedinog objekta koju prikazuje kôd 3.5:

```
export const updateObject = (oldObject, updatedProperties) =>
{
    return {
        ...oldObject,
        ...updatedProperties
    };
};
```

Kôd 3.5 Funkcija za ažuriranje atributa objekta

Funkcija kao parametre prima objekt, čije atribute je potrebno ažurirati, zatim nove atribute, a vraća novi ažurirani objekt. Views direktorij sadrži gotove prikaze ekrana (*engl. containers*) koje su sačinjene od komponenti. Primjer container-a je početna (*engl. home*) stranica na kojoj se u našem slučaju nalaze grafovi i navigacijski izbornik.

Prethodno spomenuta React-redux biblioteka se koristi za upravljanje stanjem aplikacije. Arhitekturu ažuriranja stanja aplikacije korištenjem React-redux biblioteke prikazuje slika 3.14.




Slika 3.14 Arhitektura redux komponente za upravljanje stanjem aplikacije

Mehanizam ažuriranja stanja izvodi se na principu objavljivač-pretplatnik (*engl. publisher-subscriber*). Stanje aplikacije predstavlja objekt „Store“. Element korisničkog sučelja (*engl. user interface*, skraćeno UI) distribuira (*engl. dispatch*) akciju koju preuzima odgovarajući reducer čiji je zadatak ažurirati stanje aplikacije.

Nakon ažuriranja stanja, obavještavanju se sve komponente koje su pretplaćene na tu akciju. Osim managementa stanja, vrlo važnu ulogu ima i axios biblioteka također preuzeta preko npm baze. Axios biblioteka se koristi unutar reducera za asinkrono dohvaćanje podataka iz Firebase baze podataka. Firebase baza podataka je dostupna Google-ova usluga. Glavni razlozi za odabir Firebase baze podataka su:

- nije potrebno održavanje baze od strane korisnika
- dohvaćanje podataka se odvija velikom brzinom
- moguće je selektivno i sortirano dohvaćanje podataka
- besplatna je za potrebe ovog rada

### 3.4.2. Funkcionalnost

Internet aplikacija je korisničko sučelje koje služi za upravljanje poslovnim procesima upravljanja voznim parkom. Nakon pokretanja aplikacije pojavljuje se početni ekran kao na slici 3.15. Zatim je potrebno unesti podatke za prijavu korisnika. Ako su podaci za prijavu podaci pogrešni, pojavljuje se odgovarajuća poruka, u protivnom se otvara nadzorna ploča (*engl. dashboard*). Nadzorna ploča je prikazana na slici 3.16, a sadrži zbirni prikaz najvažnijih podataka koji daju uvid u poslovanje. S lijeve strane se nalazi meni za navigaciju kroz aplikaciju. Meni je otvoren na ekranima koji se koriste na računalima, a

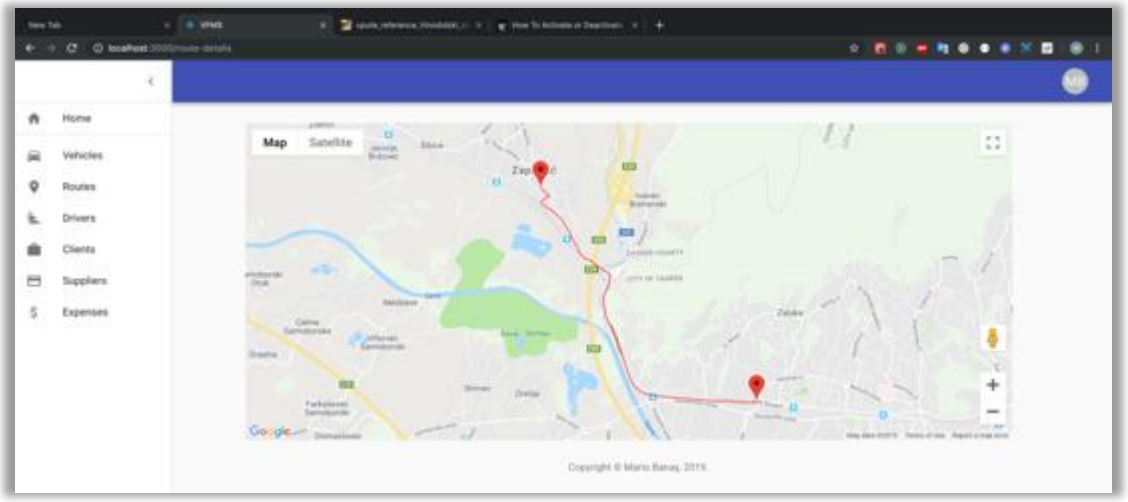
zatvoren kod korištenja mobilnih telefona ili tableta. Pomoću menija se može direktno opcijama: Vehicles, Routes, Drivers, Clients, Suppliers i Expenses.


Slika 3.15 Prijava korisnika


Slika 3.16 Nadzorna ploča

Vehicles opcija prikazuje popis vozila koji sadrži najvažnije podatke za pojedino vozilo: reg. oznaku, marku i model i korisnika vozila. Na vrhu ekrana nalazi se kontrola za dodavanje novog vozila. Za svako vozilo na popisu postoje kontrole za uređivanje, brisanje i prikaz detalja za pojedino vozilo. Pritiskom na kontrolu details, prikazuju se dodatne informacije o pojedinom vozilu.

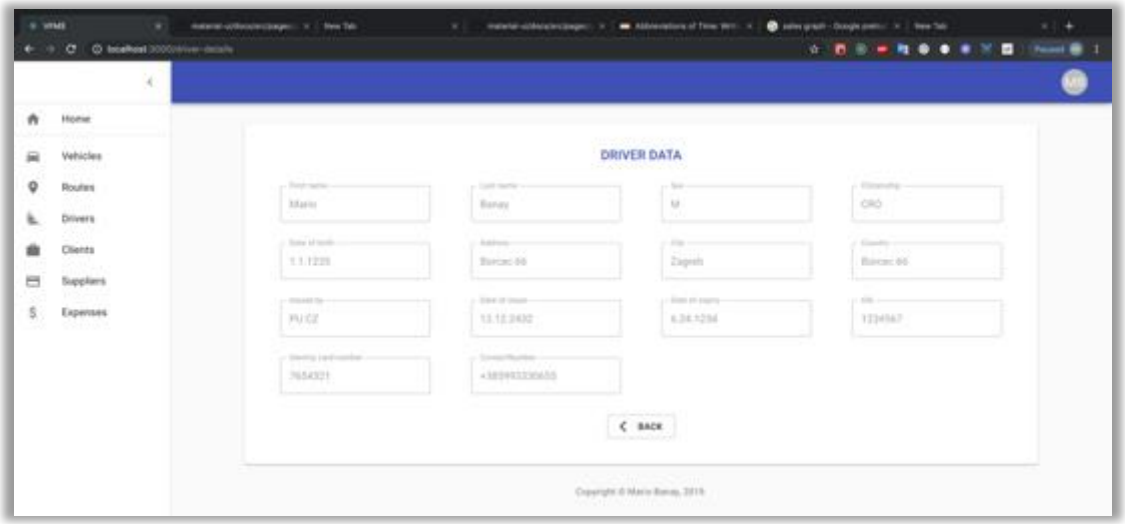
Routes opcija prikazuje popis prijeđenih ruta koji sadrži najvažnije podatke za pojedinu rutu: reg. oznaku vozila koje je prešlo rutu, vrijeme početka rute, vrijeme završetka rute, ukupno trajanje rute i duljina rute. Za svaku rutu postoji kontrola za pregledavanje rute na karti i brisanje rute. Prikaz mape prijeđene rute korisnika prikazuje slika 3.17. Drivers opcija prikazuje popis korisnika pojedinog vozila koji sadrži najvažnije podatke o pojedinom korisniku: ime i prezime, grad i državu, te registracijsku oznaku vozila koje korisnik koristi.



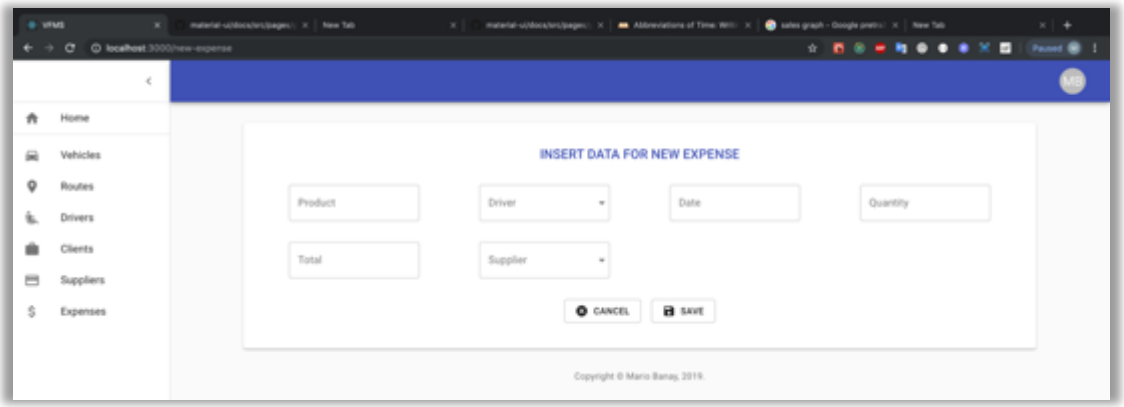

Slika 3.17 Prikaz prijeđene rute

Na vrhu prikaza nalazi se kontrola za dodavanje novog korisnika. Za svakog korisnika postoje kontrole za uređivanje, brisanje i prikaz detalja o korisniku. Sučelje za prikaz detalja korisnika prikazan je na slici 3.18.

Clients opcija prikazuje popis klijenata tvrtke s prikazom najvažnijih informacija pojedinog klijenta: naziv, grad i državu. Na vrhu prikaza nalazi se kontrola za dodavanje novog klijenta. Za svakog klijenta postoje kontrole za uređivanje, brisanje i prikaz detalja o klijentu. Suppliers opcija prikazuje popis dobavljača tvrtke s prikazom najvažnijih informacija pojedinog dobavljača: naziv, grad i državu. Na vrhu prikaza nalazi se kontrola za dodavanje novog dobavljača. Za svakog dobavljača postoje kontrole za uređivanje, brisanje i prikaz detalja o dobavljaču.




Slika 3.18 Prikaz detalja korisnika vozila




Slika 3.19 Unos novog troška

Expenses opcija prikazuje popis nastalih troškova s prikazom najvažnijih informacija za pojedini trošak: vrstu troška, datum i vrijeme nastajanja troška, količinu i iznos troška. Na vrhu prikaza nalazi se kontrola za dodavanje novog troška. Za svaki trošak postoje kontrole za uređivanje, brisanje i prikaz detalja o pojedinom trošku. Sučelje za unos novog troška prikazan je na slici 3.19.

### 3.4.3. Testiranje

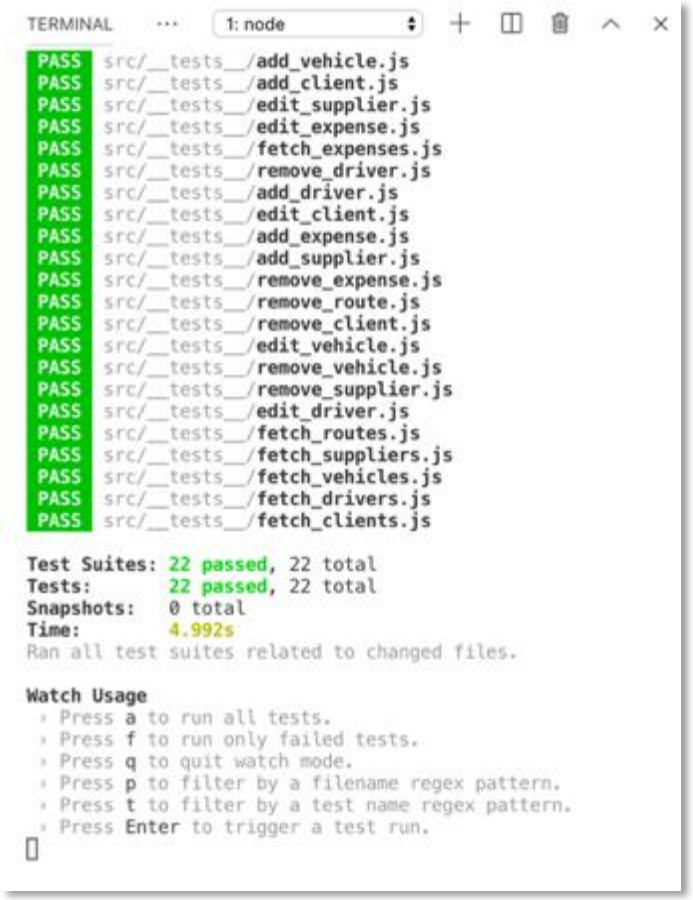
Postoji nekoliko načina za testiranje aplikacija izrađenih pomoću ReactJS biblioteke. Postoji načelna podjela na dvije kategorije:

- testiranje renderiranja komponenti gdje se u pojednostavljenom testnom okruženju provjeravaju izlazne vrijednosti pojedine komponente
- testiranje čitave aplikacije simulacijom preglednika u stvarnim uvjetima, tzv. end-to-end testiranje

Kod planiranja testiranja, potrebno je imati na umu moguće kompromise koji se mogu pojaviti s obzirom na izbor alata za testiranje. Neki alati izvršavanju testove mnogo brže nego što je to slučaju u realnosti, u stvarnom Internet pregledniku pa se može činiti da su performanse aplikacije bolje nego što jesu. S druge strane, neki alati za testiranje koriste internet preglednik kao realno okruženje, ali sporije izvršavaju testove što opet dovodi do mogućih zabluda pri tumačenju rezultata. Jest je preporučeni alat za izvršavanje testova komponenti izraženih za ReactJS. Jest ima odlične performanse, mocking mogućnosti i timere što daje korisniku relativno visok stupanj kontrole nad izvođenjem aplikacije.

U ovom radu provedeni su testovi za asinkrono dohvaćanje podataka iz baze podataka, te operacije za dohvaćanje, uređivanje, brisanje i kreiranje novih entiteta po

kategorijama iz aplikacije: Vehicles, Routes, Drivers, Clients, Suppliers i Expenses. Rezultati testova prikazani su na slici 3.20.




Slika 3.20 Rezultati testova internet aplikacije

## 3.5. Mogućnosti

Mogućnosti sustava za upravljanje voznim parkom su prilagođene tvrtkama koje se počinju baviti upravljanjem voznim parkom. Pretpostavljeno je da je u tom slučaju najvažnije imati osnovne podatke o klijentima i njihovim korisnicima vozila, dobavljačima, troškovima, naplati usluge, podatke o vozilima i prijeđenim rutama. Podaci o ulaznim i izlaznim računima su bitni za prikaz trenda poslovanja. Ako se iznosi ulaznih i izlaznih računa povežu s klijentima i dobavljačima, može se vidjeti koji su najveći i najvažniji klijenti i dobavljači tvrtke. Tako se najvažnijim klijentima može ponuditi popust na usluge, a od najvećih dobavljača tražiti dodatni popust. Tehničke karakteristike vozila poput potrošnje goriva se mogu povezati s duljinom prijeđenih ruta te dobiti uvid o trendu troškova ovisno o vozaču. Tako se može odrediti koji vozači imaju primjerice najveću potrošnju goriva. Na

temelju podataka iz aplikacije mogu dobiti informacije o dinamici eksploatacije vozila i predvidjeti troškovi koji će nastati u budućnosti. Tako se može sugerirati klijentima promjena vozila ako je isplativa. Primjerice, ako određeni klijent prelazi godišnje vrlo malo kilometara, a koristi vozilo s dizelskim motorom, može mu se sugerirati nabavka slijedećeg vozila s benzinskim motorom. Na temelju osnovnih informacija o klijentima, npr. lokacije dodatne uštede se mogu napraviti planskim usmjeravanjem u najbliže servise za vozila. Navedene informacije su dobar temelj za izgradnju dobre poslovne suradnje s klijentima i dobavljačima što je osnovni preduvjet za uspješan razvoj novog biznisa.

Sustav je dizajniran tako da bude pogodan za nadogradnju. Ako bi se poslovanje znatno povećalo, sustav bi se mogao nadograditi tako da podržava proizvoljan broj korisnika. Tada bi bilo potrebno testirati bazu podataka na veći broj korisnika kako performanse sustava ne bi postale znatno narušene. Ako bi se to dogodili, preporučljivo bi bilo zamijeniti Firebase realtime bazu podataka koja sprema tekstualne zapise, za relacijsku bazu podataka koja bi imala znatno bolje performanse ako postoji velik broj korisnika. Sustav bi se jednostavno mogao nadograditi i s novim funkcionalnostima poput:

- nadzorna ploča bi mogla pokazivati predviđanje budućeg poslovanja
- rute bi mogle biti kategorizirane po regijama, mogle bi se uvesti geografske granice na temelju kojih bi administrator mogao dobivati obavijesti
- vozila bi se mogla razdijeliti na kategorije po emisiji CO2 s obzirom na troškove, čime bi se moglo odrediti najisplativija vozila koja najmanje zagađuju
- klijenti bi se mogli kategorizirati prema visini prometa, prema broju korisnika vozila i prema broju vozila. Iz tih podataka moglo bi bolje savjetovati klijente o nabavci vozila.
- dobavljače bi se moglo kategorizirati prema tipu robe koju dobavljaju, te kumulativnom iznosu računa. Na temelji tih informacija moglo bi se pregovarati s dobavljačima o boljim uvjetima
- troškove bi se moglo kategorizirati na troškove dobavljača, korisnika vozila i troškove pojedine rute

Navedenim kategorizacijama bi se otvorile mogućnosti za izradu različitih grafova i drugih grafičkih prikaza, određivanje KPI parametara, na temelju kojih bi se omogućilo i predviđanje poslovanja.

## 3.6. Nedostaci

Sustav za upravljanje voznim parkom je sustav koji se sastoji od dvije jednostavne aplikacije s osnovnim funkcijama. Jednostavnost sustava sa sobom donosi neka ograničenja. Činjenica je da je sustav najbolje iskoristiv za manje tvrtke koje imaju potrebu za jednostavnim i ekonomski prihvatljivim rješenjem za koje nije potrebna posebna edukacija. Male tvrtke u početku poslovanja mogu potpuno iskoristiti mogućnosti aplikacije i s dostupnim mogućnostima sustava proširiti poslovanje. Sustav za upravljanje voznim parkom ima nedostatke u vidu funkcionalnosti i neke tehničke nedostatke.

Funkcionalni nedostaci sustava leže u nedostatku funkcionalnosti koje komercijalne aplikacije imaju. Funkcionalnosti koje nedostaju, a dodatno bi poboljšale ovaj sustav su:

- predviđanje događaja
- mogućnost upozorenja administratora kod važnih događaja
- mogućnost komunikacije između korisnika vozila i administratora
- mogućnost očitanja stanja vozila

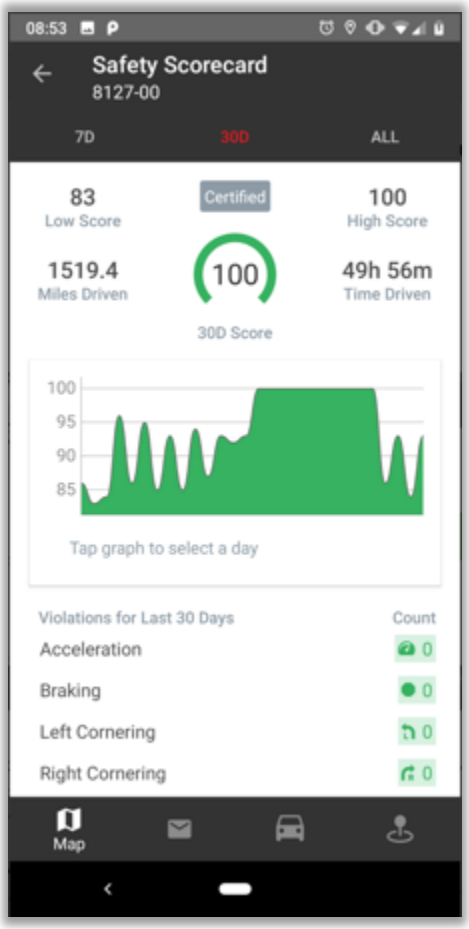
Veliko unaprjeđenje bi bilo kada bi postojala mogućnost na temelju prošlih događaja predvidjeti buduće. Primjerice, kada je planirani datum slijedećeg servisa vozila, za koliko vremena će vozilo prijeći određen broj kilometara. Tvrtke najčešće ugovaraju korištenje vozila na 5 godina ili 150 000 prijeđenih kilometara tj. što prije nastupi. Kada bi aplikacija predvidjela da će sadašnjom dinamikom pojedini korisnik dostići određeni broj kilometara za 6 mjeseci, navedenom klijentu bi se već moglo pristupati sa ponudama za novo vozilo.

Sustav upozorenja bi dao znatan doprinos aplikaciji. Kada bi postojao, administrator bi mogao putem sučelja imati prikaz važnih događaja poput isteka registracije pojedinog vozila, potrebnog odlaska na redovni servis ili isteka ugovora klijenta. Osim putem internet sučelja, administrator bi mogao primati obavijesti putem elektroničke pošte ili sms poruke. Putem kanala komunikacije bi administrator mogao slati važne obavijesti korisniku vozila. Primjerice datum servisa, a korisnik vozila bi mogao nakon obavljanja servisa jednim klikom potvrditi da je servis obavljen.

Putem kanala komunikacije korisnik bi mogao od administratora tražiti odobrenje za izvanredne troškove, poput zamjene metlica brisača na vozilu. U izrađenom sustavu klijentska mobilna aplikacija koja šalje samo podatke o lokaciji u bazu podataka prilagođena je samo za Android mobilne uređaje. Prednost bi bila kad bi sustav mogao prihvaćati podatke

s GPS uređaja koji se priključuju na vozilo putem OBD (*engl. On Board Diagnostic*, skraćeno OBD) priključka. Prednost korištenja navedenih uređaja je da imaju mogućnost čitanja podataka iz kontrolne jedinice vozila. Korištenjem takvih uređaja mogli bi se spremati u bazu i podaci o stanju vozila, kvarovima, servisnim zahtjevima i sl. Na temelju podataka iz vozila mogla bi se utvrditi nagla ubrzanja i nagla kočenja te na temelju tih podataka odrediti sigurnost vožnje kao na što prikazuje slika 3.21.

Osim funkcionalnih nedostataka, usko grlo sustava predstavlja baza podataka. Ograničenje sustava je u bazi podataka koja nije relacijskog tipa, zbog čega postoje ograničenja u performansama. Relacijska baza podataka mogla bi postići puno bolje




Slika 3.21 Kršenje postavljenih ograničenja

performanse za veći broj korisnika sustava. U sadašnjem slučaju poteškoće mogu nastati zbog velikog broja klijenata mobilne aplikacije koji šalju podatke u bazu Android broj ruta premaši određen iznos. Navedeni nedostaci pokazuju da bez obzira na dizajn korisničkog sučelja veliki utjecaj na opću kvalitetu ima baza podataka. Baza podataka je radi ciljane

primjene odabrana tako da ispunjava minimalne zahtjeve. Ipak, za dizajn sustava koji bi imao širu primjenu potrebno je dodijeliti više resursa za izbor i optimizaciju baze podataka.

# 4. Usporedba s ostalim sustavima

Sličnosti sustava za upravljanje voznim parkom izrađenog za potrebe ovog rada i komercijalnih sustava opisanih u poglavlju 2.2 očituju se u arhitekturi sustava, osnovnim funkcionalnostima, osnovnim sistemskim zahtjevima i načinu korištenja. Različitosti proizlaze iz broja funkcionalnosti koje uvjetuju i dodatne sistemske zahtjeve i hardveru koje će biti navedeni u nastavku. Osnovne komponente svih opisanih sustava su:

- GPS prijemnik
- korisničko sučelje
- baza podataka

Svi opisani sustavi imaju GPS prijemnik, međutim razlika je u načinu korištenja i mogućnostima. Sustav izrađen za potrebe ovog rada koristi GPS prijemnik ugrađen u mobitel, a ostali navedeni sustavi imaju posebno namijenjen uređaj koji osim GPS prijemnika ima sučelje za spajanje na vozilo kako bi prikupljalo podatke o trenutnom stanju vozila. Tako je moguće u svakom trenutku imati informacije o trenutnom stanju vozila. Uređaj je spojen na OBD priključak vozila ima funkciju dijagnostičkog uređaja koji prikuplja sve dostupne informacije o vozilu poput količine goriva, stanje istrošenosti kočnica, napumpanosti guma i dr..


Slika 4.1 Uređaj za ugradnju u vozilo korisnika

Kod TomTom Telematics, Webfleet i Gps Insight sustava postoji i dodatan mobilni uređaj poput konvencionalne navigacije s displayem koji ima instalirano sučelje za vozača koji šalje podatke u bazu podataka, razlikuje se od konvencionalne navigacije po tome što je mnogo robusnije izrade. Uređaj je prikazan na slici 4.1. Tako vozač ima mogućnost

korištenja navigacije te dodatnih mogućnosti ovisno o tome što je omogućio administrator sustava. Primjerice, uz dozvolu administratora korisnik vozila može vidjeti kvalitetu svoje vožnje.

Svi opisani sustavi imaju korisničko sučelje, međutim sustav za upravljanje voznim parkom izrađen za potrebe ovog rada ima samo sučelje koje je smješteno na poslužitelj, a pristup je omogućen putem internet preglednika za sve uređaje. S obzirom na to da postoji samo jedno korisničko sučelje za sve uređaje, sučelje je optimizirano i za prikaz na mobilnim uređajima. Ostali sustavi imaju kao i opisani sustav internet sučelje, ali dodatno imaju mogućnost instaliranja mobilne aplikacije za Android ili Apple uređaje. Osnovne funkcionalnosti korisničkog sučelja su iste: administracija vozača, troškova i prijeđenih ruta. Prednost odvojene aplikacije je jednostavan pristup i čuvanje postavki, zatim korisnik može biti odmah prijavljen u sustav. U slučaju izrađene aplikacije korisnik se svaki puta mora prijaviti putem internet preglednika što ipak smanjuje kvalitetu korisničkog iskustva. S druge strane pristup je jednostavniji jer nije potrebna instalacija aplikacije.

Baza podataka je neophodna komponenta koju imaju svi navedeni sustavi. Sustav izrađen za potrebe ovog rada ima jednostavnu tekstualnu bazu podataka koju omogućava Google. Baza je primjerena za korištenje u manjim sustavima. S obzirom na to da su drugi sustavi komercijalni i izrađeni za veliki broj korisnika, koriste napredniju relacijsku bazu podataka. Najveća prednost relacijske baze je brzina dohvaćanja podataka, ali zahtjeva mnogo više vremena za implementaciju.

Različitosti opisanih sustava proizlaze iz osnovne namjene. TomTom Telematics, Webfleet i Gps Insight sustavi su namijenjeni za tvrtke koje se bave transportom dobara, a ne samo upravljanjem voznim parkom. To su tvrtke koje mogu imati vrlo velike vozne parkove [15]. Zbog toga imaju osim funkcionalnosti za upravljanje troškovima eksploatacije vozila i dodatne mogućnosti praćenja robe koja se dostavlja. U najnaprednijim paketima usluga nude svojim korisnicima i praćenje vozača. Moguće je pratiti satnicu vozača, pomoću tehnologije za prepoznavanje lica čak i raspoloženje vozača. Sustav može prepoznati ako je vozač umoran ili je napravio prometni prekršaj. Dodatno, sustav može mjeriti i kvalitetu vožnje na temelju zadanih parametara. Za navedene dodatne mogućnosti potrebno je da korisnik instalira kameru u vozilo namijenjenu za tu svrhu dostupnu kod pružatelja usluge. Unatoč nabrojenim naprednim mogućnostima, ipak korištenje kamere znatno zadire u privatnost korisnika te je zbog toga primjena u praksi upitna.

Sustav koji je izrađen za potrebe rada ima samo dio funkcionalnosti drugih opisanih sustava, ali je za početak bavljenja upravljanjem voznim parkom dovoljan. Ideja je bila razviti POC (engl. Proof of concept, skraćeno POC) koji će biti spreman za korištenje. Velika je prednost što su funkcionalnosti intuitivne pa nije potrebna dodatna edukacija. Osim toga, troškovi korištenja sustava su zanemarivi. Ako bi se pojavila potreba, sustav bi se mogao nadograditi novim funkcionalnostima koje bi podržale razvoj poslovanja tvrtke.

# Zaključak

Upravljanje voznim parkom je složen poslovni proces koji omogućuje optimizaciju troškova eksploatacije vozila. Zbog velike količine ulaznih parametara koje je potrebno pratiti, korištenje softvera se nameće kao nužan odabir za tvrtke koje se bave iznajmljivanjem vozila. Budući da je sama izrada kvalitetnog sustava kompleksan posao koji zahtjeva mnogo vremena za razvoj i testiranje, troškovi nabavke ili korištenja softvera su relativno visoki za tvrtke koje se tek počinju baviti tom uslugom. Istraživanje tržišta pokazalo da ima dovoljno kompleksnih sustava s najnaprednijim mogućnostima poput snimanja i prepoznavanja prometnih prekršaja, ali ima vrlo malo jednostavnih i besplatnih sustava s kojima je moguće početi raditi. Osim toga za sve sustave je potreban i dodatni hardver u vidu GPS prijemnika, a funkcionalnosti ima toliko mnogo da tvrtke nude i usluge edukacije korisnika. Zbog toga je opravdana izrada jednostavnog rješenja poput ovog, izrađenog za potrebe diplomskog rada. Kao GPS prijemnik se može koristiti bilo koji mobilni uređaj s instaliranim Android operacijskim sustavom i GPS senzorom. S vremenom i porastom baze klijenata, sustav se može nadograditi novim funkcionalnostima. Budući da veliki dio ukupnih troškova eksploatacije vozila otpada na pogonske troškove (potrošnja goriva) koji su u direktnoj vezi s emisijama CO2 štetnog za okoliš, optimizacija pozitivno utječe na okoliš.

# Popis kratica

| | |
|---|---|
| *GPS* | *Global Positioning Sistem* |
| CO2 | *Carbon Dioxide* |
| KPI | *Key Performance Indicator* |
| DBMS | *Database Management System* |
| DVM | *Dalvik Virtual Machine* |
| API | *Application Programming Interface* |
| JSON | *JavaScript Object Notation* |
| XML | *Extensible Markup Language* |
| APK | *Android Package* |
| SDK | *Software Development Kit* |
| HTML | *HyperText Markup Language* |
| CDN | *Content delivery network* |
| CSS | *Cascade Style Sheet* |
| UI | *User Interface* |
| OBD | *On Board Diagnostic* |
| SaaS | *Software as a service* |
| POCS | *Proof of concept* |
| IDE | *Integrated Development Environment* |
| AVD | *Android Virtual Devices* |
| SDK | *Software Development Kit* |

# Popis slika

# Popis kôdova

# Literatura


[1] Android Testing Samples
https://github.com/googlesamples/android-testing (03.09.2019.)

[2] Bert Bat1es, Kathy Sierra, Eric Freeman, Elisabeth Robson; Head First Design Patterns; O'Reilly Media; (2009); ISBN-13: 978-0-596-00712-6

[3] CO2 Emissions From Cars: Facts And Figures (Infographics)
http://www.europarl.europa.eu/news/en/headlines/society/20190313STO31218/co2-emissions-from-cars-facts-and-figures-infographics (31.8.2019.)

[4] Dataflair Team: Major Android Application Components – You Must Know
https://data-flair.training/blogs/android-application-components (02.09.2019.)

[5] Dawn Griffiths, David Griffiths; Head First Android Development: A Brain-Friendly Guide; O'Reilly Media; (2017); ISBN-13: 978-1-491-97405-6

[6] Developer Survey Results
https://insights.stackoverflow.com/survey/2019 (03.09.2019.)

[7] Gáspár, P.; Szalay, Z.; Aradi, S.: Highly Automated Vehicle Systems
http://www.mogi.bme.hu/TAMOP/jarmurendszerek_iranyitasa_angol/ch14.html (31.08.2019.)

[8] Gps Trackit
https://gpstrackit.com/pricing/#1554749397604-627dbf59-6413 (31.08.2019.)

[9] Gps Insight
https://www.gpsinsight.com/solutions (31.08.2019.)

[10] HAK (Hrvatski Autoklub)
https://www.hak.hr/vozacki-ispiti/kategorije-vozila (30.08.2019.)

[11] Lilys, M. Final data structures. Doktorski rad. Sveučilište u Zagrebu, 2010.

[12] Paul, G.: Why Huawei's Android backup would flop, Business Insider
https://www.businessinsider.com/huawei-android-backup-operating-system-would-fail-2019-3 (31.08.2019.)

[13] React Reference Storefront Overview: Platform Architecture https://developers.elasticpath.com/referencestore/2.0/referencestore-architecture (04.09.2019.)

[14] ReactJS Documentation: Testing Overview https://reactjs.org/docs/testing.html (05.09.2019.)

[15] Rogić, K.; Šutić, B.;Methodology Of Introducing Fleet Management System, https://www.researchgate.net/publication/298951839_Methodology_of_Introducing_Fleet_Management_System (30.08.2019.)

[16] Sean Lockhart; Fleet: A Guide to Simplifying Vehicle Fleet Management for Small Business; (2016); ISBN-13: 978-0692726853

[17] Sharma, A.: Building React App Part 2 - Architecture Boilerplate https://itnext.io/building-react-app-part-2-architecture-boilerplate-683b992089a6 (04.09.2019.)

[18] TomTom Telematics: Fleet Management https://telematics.tomtom.com/en_gb/webfleet/fleet-management/vehicle-tracking (31.08.2019.)

Student vlastoručno potpisuje diplomski rad iza zaključka s datumom i oznakom mjesta završetka rada te naznakom:

*„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, datum.*

*Ime Prezime*

# Prilog

Diplomski rad može imati priloge, ali se oni ne prilažu uz pisanu verziju diplomskog rada već se mogu priložiti na diplomskom ispitu ukoliko povjerenstvo na diplomskom ispitu tako odluči. Važno je čuvati svu poratnu dokumentaciju koja je nastala pri izradi diplomskog rada.

S unutarnje strane na zadnjim koricama originala, kao i svake kopije diplomskog rada, pričvršćuje se CD s kompletnim diplomskim radom u izvornom formatu (npr. .docx) i .pdf formatu sa svom popratnom dokumentacijom i programima. Pri čemu je obvezno da na tom CD- u postoji i dokument koji opisuje kako se rezultat njegova diplomskog rada (softver ili hardver) koristi (ili kako se npr. izvode mjerenja koja je opisao u radu). Ako se radi o softveru nužno je opisati i kako se programska podrška instalira.