Inteligentni transportni sustav za dijeljenje prometnih informacija u stvarnom vremenu pomoću pametnih mobilnih uređaja

Videković, Jakov

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:771477

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij računarstva

INTELIGENTNI TRANSPORTNI SUSTAV ZA DIJELJENJE PROMETNIH INFORMACIJA U STVARNOM VREMENU POMOĆU PAMETNIH MOBILNIH UREĐAJA

Diplomski rad

Jakov Videković

Osijek, 2017.

SADRŽAJ

1. UVOD 1
2. INTELIGENTNI TRANSPORNI SUSTAVI I BEŽIČNA AD-HOC MREŽA 3
2.1. INTELIGENTNI TRANSPORNI SUSTAVI 3
2.2. AD-HOC MREŽE VOZILA - VANET 6
2.3. BEŽIČNE AD-HOC MREŽE 8
2.4. TIPOVI BEŽIČNIH AD-HOC MREŽA 9
2.4.1. MOBILNA AD-HOC MREŽA (MANET) 9
2.4.2. MOBILNA AD-HOC MREŽA SENZORA 10
3. PAMETNI MOBILNI UREĐAJI I ANDROID PLATFORMA 11
3.1. PAMETNI MOBILNI UREĐAJI 11
3.1.1. DETEKCIJA PROMETNE NESREĆE POMOĆU PAMETNOG MOBILNOG UREĐAJA 14
3.2. ANDROID PLATFORMA 15
3.2.1. ARHITEKTURA ANDROID PLATFORME 16
3.2.2. SENZORI 17
3.2.3. ANDROID RAZVOJNO OKRUŽENJE 19
4. INTELIGENTNI TRANSPORTNI SUSTAV ZA DIJELJENJE PROMETNIH INFORMACIJA U STVARNOM VREMENU POMOĆU PAMETNIH MOBILNIH UREĐAJA 22
4.1. ANDROID MOBILNA APLIKACIJA 22
4.1.1. ARHITEKTURA APLIKACIJE 25
4.1.2. UPRAVLJANJE KORISNICIMA 31
4.1.3. KORISNIČKI NAČIN RADA 33
4.1.4. PROAKTIVNI NAČIN RADA 34
4.1.5. REAKTIVNI NAČIN RADA 36
4.1.6. SINKRONIZACIJA PODATAKA 38
4.2. POSLUŽITELJSKA APLIKACIJA 41
4.2.1. POSLUŽITELJSKO SUČELJE 43
4.2.2. KLIJENTSKA APLIKACIJA 49

5. ZAKLJUČAK 53
LITERATURA 54
SAŽETAK 56
ABSTRACT 57
ŽIVOTOPIS 58

1. UVOD

Napretkom automobila i cestovnog prometa pojavljuje se potreba za povećanjem sigurnosti u prometu. Nova vozila su sve češće opremljena dodatnim uređajima za mjerenje i dobivanje informacija iz njihove okoline čime je povećana sigurnost prometa. Nadogradnjom postojećih mogućnosti na način da vozila mogu razmjenjivati prometne informacije s ostalim vozilima u svojoj blizini znatno bi povećala sigurnost prometa. Velik broj funkcionalnosti današnjih pametnih mobilnih uređaja omogućile bi dijeljenje i prikupljanje prometnih informacija. Svako vozilo bi bilo pravovremeno obaviješteno o stanju u prometu te bi se na taj način spriječile brojne nepovoljne prometne situacije. Vozila bi dijelila informacije o potencijalnim zastojima, prometnim nesrećama, naglom kočenju, nepovoljnim vremenskim uvjetima. Informacije o stanju u prometu bile bi dostupne svim sudionicima prometa te bi oni mogli reagirati sukladno njima. Zamislite da prilikom kretanja na posao ili putovanje možete provjeriti trenutno stanje u prometu prikupljeno od strane drugih sudionika u prometu koji se kreću tom rutom. Znatno bi se smanjio broj zastoja jer bi vozači bili unaprijed obaviješteni da na njihovoj ruti postoji zastoj te bi mogli izbjeći zastoj. Kvaliteta i sigurnost prometovanja bi bila znatno povećana te time i zadovoljstvo korisnika prometnica.

Većina današnjih vozila ne dolazi s ugrađenom opremom koja bi omogućila snimanje prometnice tijekom vožnje, detekciju naglog kočenja ili zastoja u prometu, ali većina vozača posjeduje pametne mobilne uređaje koje je moguće iskoristiti u tu svrhu. Pametni mobilni uređaji opremljeni mnoštvom senzora te stalno dostupnom i sve više raširenom mobilnom podatkovnom vezom pokazuju se kao podobni uređaji za obavještavanje vozača o stanju u prometu. Kao dio ovog rada razvijena je Android mobilna aplikacija koja snima prometnicu u 3 načina rada. Zapisi snimljeni mobilnom aplikacijom šalju se udaljenom poslužitelju. Razvijena je web aplikacija koja prikazuje na mapi zapise poslane od strane korisnika mobilne aplikacije. Web aplikacija ima mogućnost pretrage ruta i filtriranje zapisa.

U drugom poglavlju ovog rada opisani su inteligentni transportni sustavi i bežična ad-hoc mreža vozila. Opisane su glavne značajke inteligentnih transportnih sustava te mogućnosti primjene u današnje vrijeme. Opisan je način komunikacije u bežičnim ad-hoc mrežama te su objašnjeni glavni tipovi ad-hoc mreža. Treće poglavlje ovog rada sadržava opis pametnih mobilnih uređaja i Android platforme. Navedene su glavne značajke pametnih mobilnih uređaja i mogućnost detekcije prometne nesreće korištenjem mobilnih uređaja. Unutar poglavlja o Android platformi navedena je arhitektura platforme, dostupni senzori i razvojno okruženje za izradu mobilnih aplikacija za Android platformu. Četvrto poglavlje rada sadržava opis

razvijenog inteligentnog transportnog sustava za dijeljenje prometnih informacija u stvarnom vremenu pomoću pametnih mobilnih uređaja. Opisani sustav je podijeljen na Android mobilnu aplikaciju koja služi za prikupljanje podataka tokom vožnje i poslužiteljsku aplikaciju koja prikazuje prikupljene podatke na mapi.

2. INTELIGENTNI TRANSPORNI SUSTAVI I BEŽIČNA AD-HOC MREŽA

Krajem 20. stoljeća sve veća zagušenost svih prometnica i potreba za transportom potaknulo je razvoj načina za rješavanja problema organiziranosti prometa. Povećanjem broja vozila smanjuje se sigurnost zbog više sudionika u prometu. Javlja se potreba za nadogradnjom prometnica i vozila.

Inteligentni transportni sustavi (ITS) ostvareni bežičnom ad-hoc mrežom pridonose razvoju i efikasnosti postojećih transportnih sustava. Do nedavno prikupljanje prometnih informacija u svrhu poboljšanja i unaprjeđenja prometnih informacija temeljeno je na podacima prikupljenim putem nepomičnih senzora. Visoka cijena opreme i održavanja sprječavali su uvođenje praćenje i kontrolu prometa u stvarnom vremenu. Problem visoke cijene infrastrukture riješen je pojavom malih senzora s integriranim modulima za obradu podataka opremljenih bežičnom tehnologijom.

Bežična mreža senzora pruža optimalna rješenja po prihvatljivoj cijeni i jednostavnoj implementaciji transportnih sustava. Jedno od takvih rješenja je inteligentna parking aplikacija ostvarena pomoću bežične mreže senzora [1].

2.1. INTELIGENTNI TRANSPORNI SUSTAVI

Glavne značajke ITS-a (eng. *intelligent transport systems*) su bolje upravljanje i poboljšan odziv prometnog sustava čime on postaje inteligentan. Da bi sustav bio „inteligentan“ mora se prilagođavati okolini i imati mogućnost prikupiti i obraditi dovoljno podataka u realnom vremenu. ITS uveden je u stručni i znanstveni rječnik 1994. godine u Parizu [2].

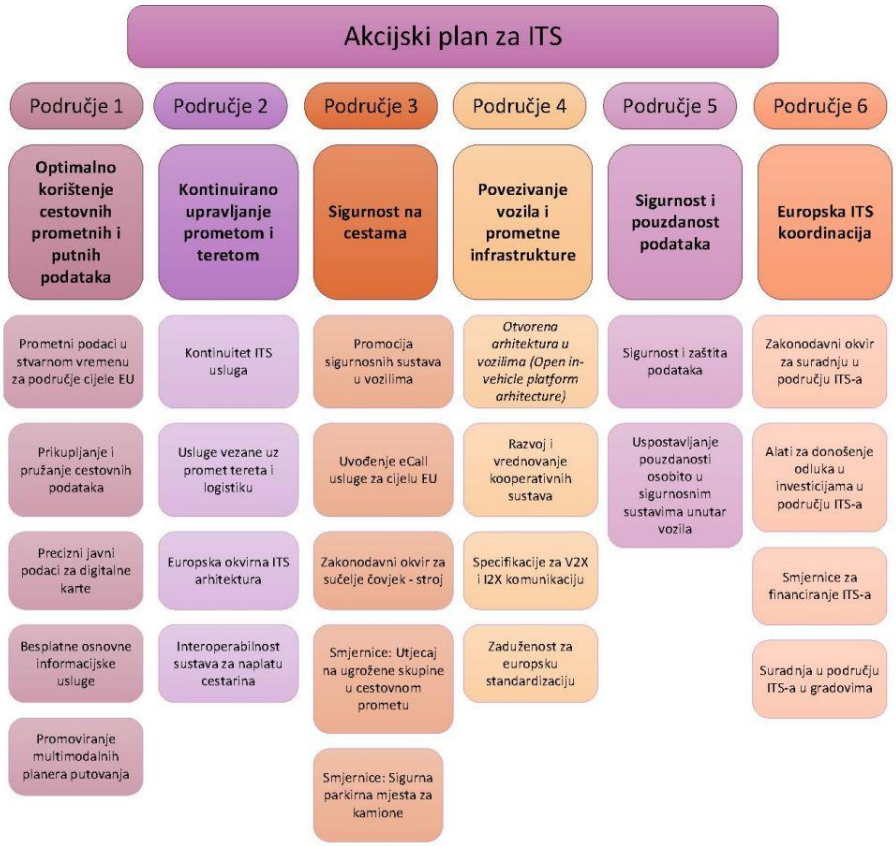
ITS mogu se definirati kao holistička, upravljačka i informacijsko komunikacijska nadogradnja klasičnog sustava prometa i transporta kojim se postiže znatno poboljšanje performansi odvijanja prometa kroz učinkovitiji prijevoz putnika i robe, poboljšanje sigurnosti u prometu, udobnost i zaštita putnika, smanjenje onečišćenja okoliša [3]. ITS donosi rješenje trenutnih prometnih problema u vidu inteligentnih cestovnih prometnica koje se predstavljaju kao nadogradnja trenutne infrastrukture. Nadogradnja osim fizičkih funkcija nadogradnje prometnica podrazumijeva i veću informativnost vozača.

Konkretna korist ITS-a promatraju se kroz različite skupine pokazatelja, odnosno kategorije ITS učinaka [3]:

1. sigurnost
2. učinkovitost protoka

3. proizvodnost i smanjenje troškova
4. korist za okoliš.

Unutar akcijskog plana za uvođenje ITS-a u Europi definirano je 6 područja aktivnosti te 24 prioritetne aktivnosti prikazane slikom 2.1. [3]



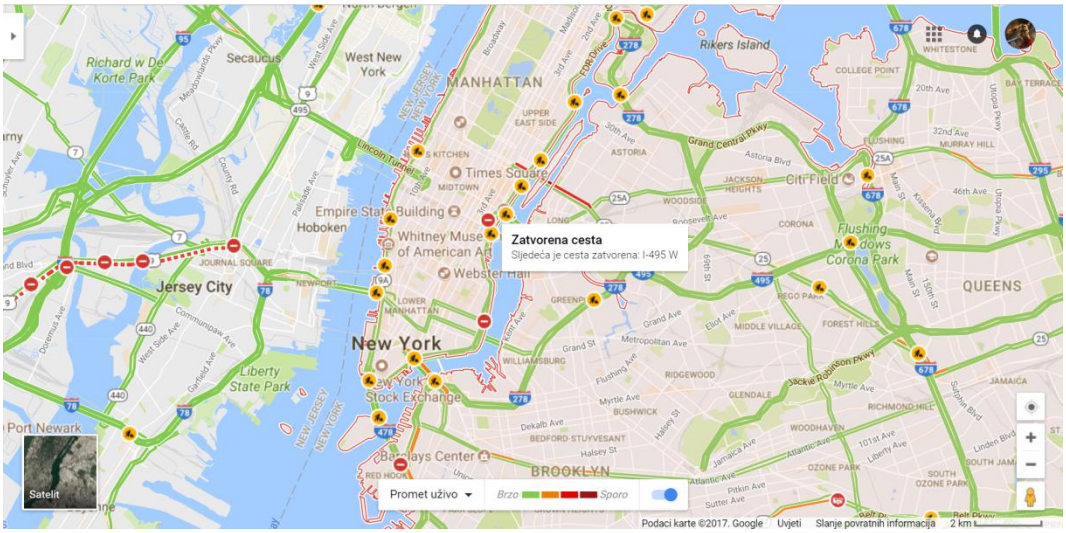
Slika 2.1. Prioritetna područja i aktivnosti [3]

Glavni ciljevi ITS-a su sigurnost, efikasnost, zaštita okoliša i poboljšanje prometnica primjenom tehnologije. Napredak tehnologije prijenosa informacija, komunikacije, određivanja lokacije i senzora ujedno donosi i napredak ITS-a. Prvotni ciljevi ITS-a su veća informativnost prometnice koja bi se postigla integracijom raznih sustava. Neki od njih su sustavi za sigurnu vožnju koji obavještavaju vozača o potencijalnim opasnostima izvan njegovog vidnog kruga, trenutne informacije o prometu u širem području koje nisu direktno dostupne vozaču, informacije o potencijalnim zastojima u prometu. Vozač bi korištenjem tih informacijama mogao izbjeći potencijalne poteškoće u prometu. Također, obavještavaju vozača o trenutnim uvjetima na cesti što smanjuje mogućnosti prometnih nesreća. Tehnologije kao što su ABS (engl. *anti-lock breaking system*) i sustav za automatsko izbjegavanje sudara koje bi pomažu vozaču u

upravljanju vozila. Generalno sve ove promjene predstavljaju prelazak s pasivnog načina sigurnosti na aktivan način te veliki iskorak prema automatskom upravljanju na više razina.

VICS [4] (engl. *Vehicle Information and Communication System*) je sustav odašiljanja i prikupljanja potrebnih informacija pomoću senzora koji se nalazile uz prometnice. Podaci prikupljeni ovim načinom ograničeni su na područje u kojem se nalaze senzori. Razvojem tehnologije unutar zadnjih 10 godina uvode se sustavi temeljeni na vozilima opremljenim senzorima kako bi se riješila ograničenja statičkih senzora. Prikupljeni podaci se šalju udaljenim poslužiteljima koji obavještavaju ostale sudionike prometa o trenutnoj situaciji u prometu. Ovakvi sustavi su ovisni o lokaciji senzora i broju automobila s ugrađenim senzorima stoga nisu podobni za područja bez senzora i malom količinom prometa.

U današnje vrijeme postoje mnogi poslužitelji koji pružaju informacije o trenutnom stanju na prometnicama. Prometne informacije su većinom dostupne za veće gradove s većom količinom prometa. Neki od najpoznatijih pružatelja informacija o trenutnom stanju u prometu su Google TrafficMap, MapQuest, Yahoo Map, NAVTEQ, SigAlert, BeatTheTraffic. Primjer jednog poslužitelja informacija o prometnu možemo vidjeti na slici 2.2 na kojoj je prikazana Google mapa New Yorka s dostupnim prometnim informacijama za to područje. Brzina prometa označena je zelenom bojom za brzi promet preko nekoliko razina do crvene boje za stop promet. Mapa također prikazuje zatvorene ceste i građevinske radove za odabrano područje.



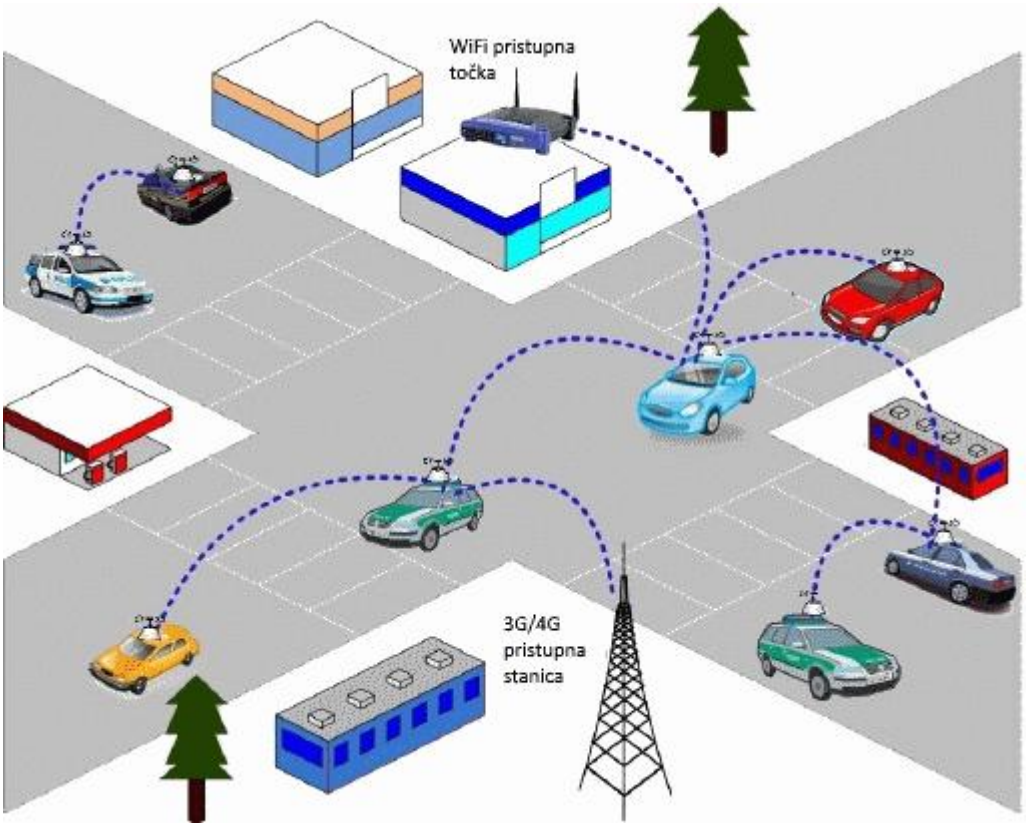
Slika 2.1. Google Maps, promet za New York

2.2. AD-HOC MREŽE VOZILA - VANET

Povećanjem populacije i industrijskim razvojem dolazi do povećanog prometa diljem svijeta što povećava vjerojatnost prometnih nesreća. Zbog toga automobilska industrija daje posebnu pažnju razvoju ITS-a i opremanju automobila potrebnom opremom kako bi se postigla zadovoljavajuća razina sigurnosti. U znanstvenim i tehnološkim područjima pridaje se sve više pažnje mobilnim ad-hoc mrežama.

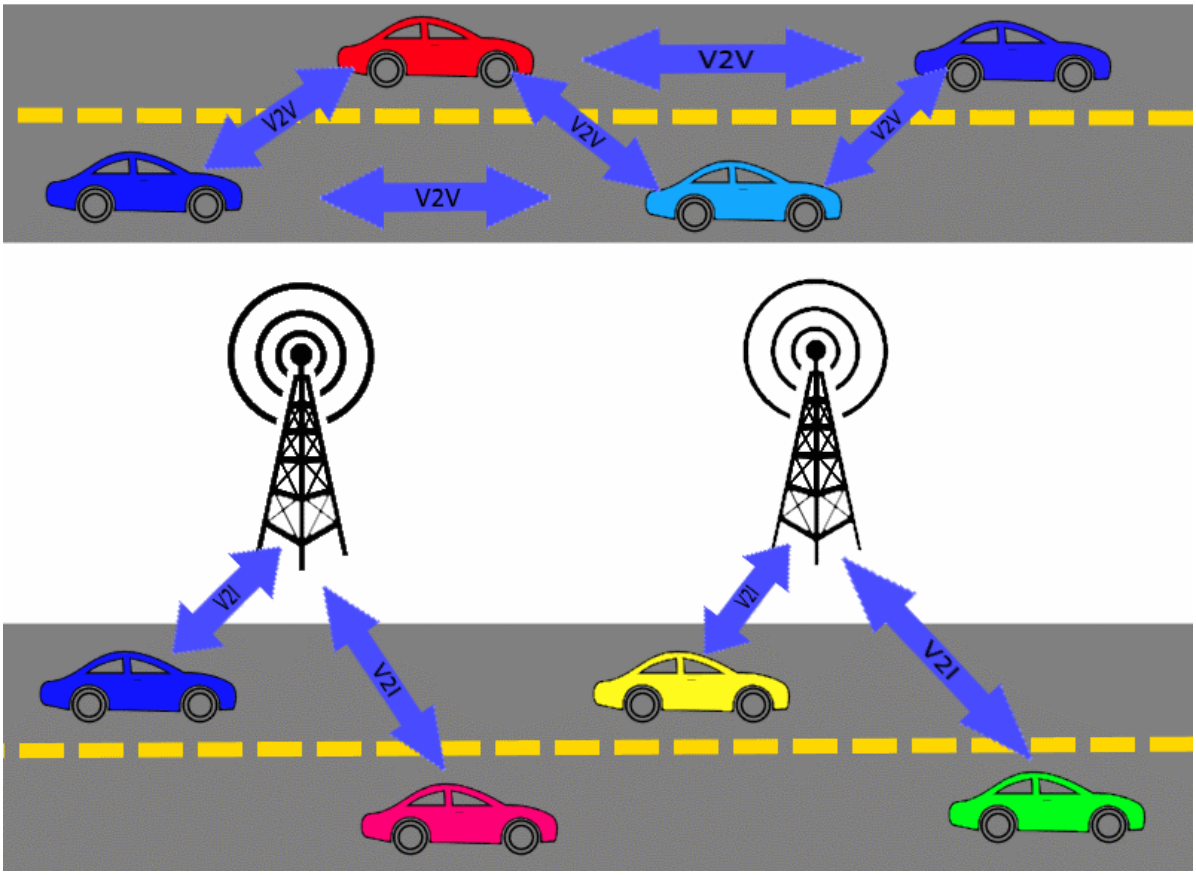
Razvojem IEEE 802.11p protokola [5] omogućena je razmjena podataka među vozilima koja se kreću prometnicama što je vidljivo na slici 2.2. Bežični DSRC (engl. *Device Short-Range Communications*) vozilima je omogućio međusobnu komunikaciju preko VANET-a. Vozila imaju ugrađene senzore za određivanje uvjeta na prometnici.

VANET (engl. *vehicular ad hoc networks*) je podgrupa mobilnih ad-hoc mreža i prepoznata je kao važan element za razvoj ITS-a [6]. Podržava širok spektar aplikacija i poslužitelja. Jedan od glavnih fokusa ovog tipa mreže je do sada bio razvoj sigurnosnih aplikacija u obliku nadzora ceste, upozorenja na nesreću, nadzor prometa. VANET ima veliki utjecaj na smanjenje prometnih nesreća i povećanje sigurnosti u prometu.



Slika 2.2. Prikaz VANET infrastrukture

VANET je definiran kao tip bežične mreže u kojoj su vozila čvorovi mreže. Vozila (čvorovi) međusobno komuniciraju kako bi promet bio sigurniji i bolji. Čvor u mreži ponaša se kao odašiljač koji obavještava ostale čvorove o trenutnom stanju na prometnici. Postoje dvije vrste mrežne komunikacije, V2V (engl. *vehicle-to-vehicle*) i V2I (engl. *vehicle-to-infrastructure*) [7]. V2V komunikacijom se ostvaruje veza između vozila, a V2I služi za komunikaciju senzora i ostale infrastrukture s vozilima što se može vidjeti na slici 2.3. Glavna zadaća je pravovremeni prijenos podataka u stvarnom vremenu s ciljem smanjenja prometnih nesreća i zastoja u prometu.

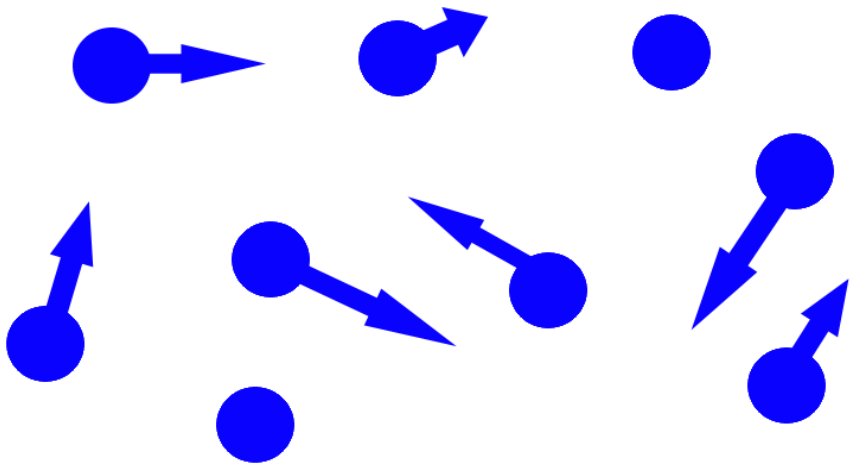


Slika 2.3. V2V i V2I mrežna komunikacija

Postoje brojne poteškoće vezane uz sigurnost i zaštitu dijeljenih podataka. Za nesmetani rad bežičnih mreža kao što je VANET potrebna je dobra komunikacija između čvorova mreže. Postavljaju se visoki zahtjevi prilikom kreiranja potrebnih servisa i uređaja koji se lako mogu prilagoditi takvoj dinamičnoj okolini. U svakom trenutku uređaji i servisi moraju međusobno biti povezani sigurnom i brzom vezom. Podaci koji se razmjenjuju sastoje se od osnovnih informacija poput očitanja senzora do kompleksnih poput slika i video zapisa.

2.3. BEŽIČNE AD-HOC MREŽE

Bežična ad-hoc mreža je skupina od dva ili više uređaja ili čvora s mogućnostima bežičnog povezivanja koji međusobno komuniciraju bez potrebe za centralnim uređajima. Bežični čvorovi su povezani na način da formiraju mrežu bez fiksne mrežne infrastrukture. To je brzo promjenjiva otvorena mreža u kojoj su uređaji povezani bežičnom vezom. Svi čvorovi unutar mreže se ponašaju kao primatelji, ali ujedno i kao poslužitelji informacija.U potpunosti je dinamična jer bilo koji novi čvor se u bilo kojem trenutku može uključiti, a bilo koji od postojećih čvorova može i izaći iz dometa mreže. Velik broj funkcionalnosti bežičnih ad-hoc mreža otvara nove mogućnosti za razvoj, ali uz mnogo izazovnih problema. Čvorovi unutar mreže su u potpunosti odgovorni za međusobnu komunikaciju pri čemu moraju tvoriti zadovoljavajuću mrežnu arhitekturu te održavati funkcionalnost mreže bez centralne jedinice za komunikaciju i upravljanje. Svaki čvor u mreži brine se o mrežnoj komunikaciji, održavanju mreže i prosljeđivanju poruka. Takav način rada nazivamo višestruko skokovitim (engl. *Multi Hopping*) [8] i glavni je građevinski blok ad-hoc mreža. Bežične ad-hoc mreže su kompleksnije od ostalih bežičnih mreža zbog nedostatka centralne mrežne strukture, ali su puno fleksibilnije i dinamičnije. Ad-hoc mreže formiraju tip nakupine (engl. *cluster*) kako bi efektivno i pouzdano podržavale ovakav način rada [9]. Na slici 2.4 prikazana je nakupina koji se kreću u različitim smjerovima različitim brzinama.



Slika 2.4. Nakupina (engl. *Cluster*) čvorova u Ad-hoc mreži

2.4. TIPOVI BEŽIČNIH AD-HOC MREŽA

Bežične ad-hoc mreže mogu se podijeliti u nekoliko glavnih tipova. Potrebno je istaknuti polu statične ad-hoc mreže i mobile ad-hoc mreže (MANET). U polu statičnim ad-hoc mrežama čvorovi mogu biti dinamički (mobilni) ili statični. Ovakva struktura je neizostavna zbog potrebe za napajanjem i međusobnim povezivanjem čvorova mreže. Mreža senzora je jedan od tipova ovakve mreže [10]. Unutar mobilne ad-hoc mreže čvorovi se slobodno kreću različitim brzinama u odnosu jedan na drugog. Napravit ćemo kratak pregled oba tipa mreže.

2.4.1. MOBILNA AD-HOC MREŽA (MANET)

Mobilna ad-hoc mreža je skupina neovisnih mrežnih mobilnih uređaja koji su povezani nekom od bežičnih tehnologija. Mrežna arhitektura je u potpunosti dinamična i može se mijenjati iz trenutka u trenutak. Svaki uređaj u mreži se ponaša kao odašiljač informacija koje se razmjenjuju unutar mreže. Ovakav tip mreže može raditi bez dodatne mrežne infrastrukture, ali može biti i povezan u veće mreže kao što je LAN (engl. *large area network*).

Postoje nekoliko osnovnih vrsta MANET mreža [11], a to su:

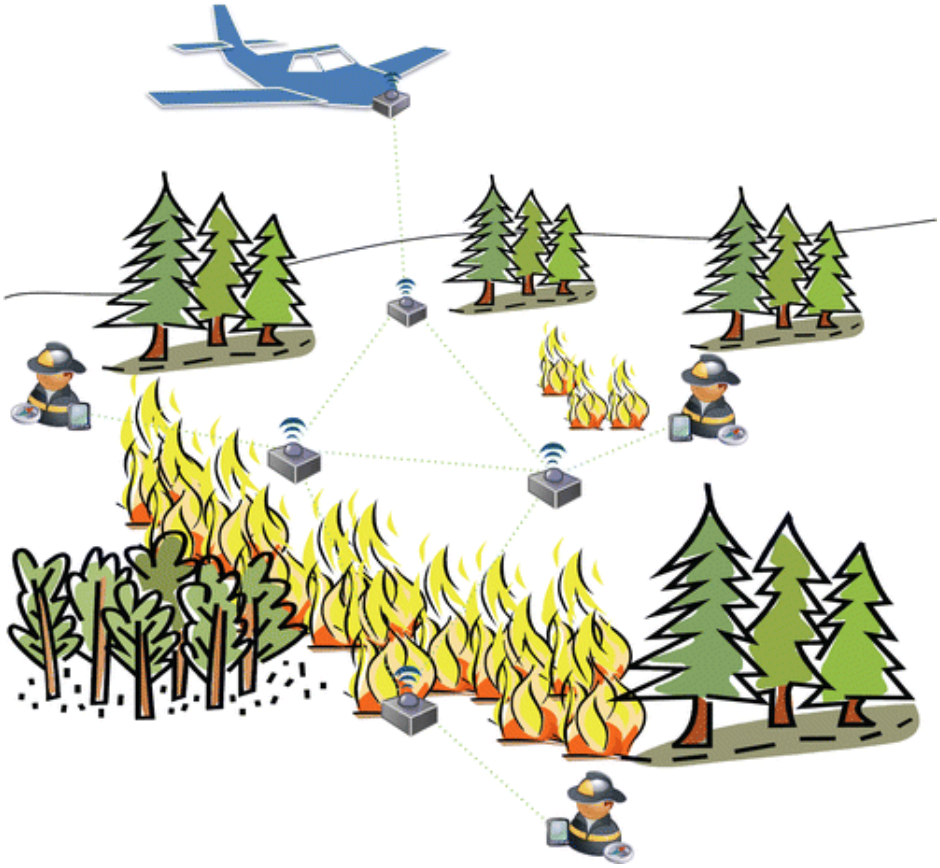
- VANETs (engl. *Vehicular Ad-Hoc Networks*) – Ad-hoc mreže vozila
- InVANETs (engl. *Inteligent Vehicular Ad-Hoc Networks*) – Inteligentne ad-hoc mreže vozila
- iMANET (engl. *Internet Based Mobile Ad-Hoc Networks*) – mobilne ad-hoc mreže bazirane na prijenosu podataka putem interneta.

Tipovi i vrste aplikacija koje se koriste u MANET mrežama variraju od malih s manjim područjem rada vezanom uz neki od tipova statičke infrastrukture do velikih mobilnih mreža s dinamičkom infrastrukturom. Povrh toga za nesmetani i pouzdan rad potreban je dobar mrežni protokol koji podržava takvu dinamičku strukturu mreže. Osim aplikacijskog dijela MANET zahtjeva i kompliciranе algoritme mrežne organizacije, propusnosti i ostvarivanja veza među povezanim uređajima. Usmjeravanje se u potpunosti razlikuje od uobičajenog jer se struktura mreže može promijeniti u svakom trenutku rada. Kod statičnih mreža najkraći put od pošiljatelja do primatelja je ujedno i najprihvatljiviji, ali se taj način povezivanja nije primjenjiv u MANET mrežama. Osim poteškoća prilikom usmjeravanja informacija također se javljaju problemi zaštite informacija, osiguranja od različitih zagušenja i namjernog narušavanja mreže te oporavka nakon pada.

2.4.2. MOBILNA AD-HOC MREŽA SENZORA

Mobilna mreža senzora za razliku od klasične mreže senzora ne zahtjeva kompleksnu infrastrukturu i skupu infrastrukturu s centraliziranim upravljačem. Mobilna mreža senzora ili hibridna ad-hoc mreža (engl. *Hybrid Ad-Hoc Network*) [12] sastoji se varijabilnog broja senzora raspoređenih na određenom geografskom području. Pojedini senzor ima mogućnosti mobilne komunikacije te razinu inteligencije tako da je u mogućnosti obrađivati signale i prenositi podatke. Kako bi se podržalo mrežno usmjeravanje i prijenos podatka između dva mobilna uređaja putem mrežnog protokola određen je način povezivanja i raspoređivanja paketa. Navedena svojstva mobilne mreže senzora omogućuju veliku fleksibilnost i široku dostupnost u različitim okolina.

Sve veća dostupnost različitih vrsta senzora i sve manja cijena izrade dovela je do znatnog napretka u području bežičnih mreža senzora. Jedan od dobrih primjera mreže senzora je detekcija šumskih požara korištenjem senzora [13] koji su postavljeni na određenim lokacijama u šumi što je prikazano slikom 2.5.



Slika 2.5. Detekcija požara korištenjem senzora

3. PAMETNI MOBILNI UREĐAJI I ANDROID PLATFORMA

Korištenje pametnih mobilnih uređaja (engl. *Smartphones*) postala je svakodnevica svakog čovjeka. Sve veće funkcionalnosti i svakodnevna upotreba otvara nove mogućnosti za korištenje. Velik i nagli razvoj i napredak pametnih mobilnih uređaja omogućuje svestranu primjenu.

Upotreba mobilnih uređaja tokom vožnje može biti od velike koristi i pomoći pri puno boljoj informiranosti o prometnici i nadolazećim situacijama s kojima će se vozač susresti. Pametni mobilni uređaji imaju mogućnost snimanja prometnice te detekcije raznih događaja u svojoj okolini. Osim toga te sadržaje na jednostavan način mogu dijeliti s drugim korisnicima mobilnih uređaja. Dijeljenje i prikupljanje takvih sadržaja o prometnici i važnim događajima na prometnici može uvelike pomoći ostalim sudionicima prometa u izbjegavanju nepoželjnih situacija kao što su zastoji u prometu ili čak prometne nesreće. Sudionici prometa bili bi pravovremeno obaviješteni te bi mogli sukladno tome prilagoditi svoj način sudjelovanja u prometu na njihovo zadovoljstvo i zadovoljstvo svih ostalih sudionika u prometu.

Današnji pametni mobilni uređaji imaju ugrađen velik broj senzora koji mogu detektirati razne događaje. Ugradnjom modula za određivanje lokacije zamijenili su tradicionalne uređaje za navigaciju. Prilikom navigacije mobilni uređaj se postavlja pomoću posebnog držača na prednje staklo vozila. Takvo pozicioniranje uređaja omogućuje stražnjoj kameri uređaja dobar pregled prometnice te je na taj način moguće snimanje slike i video zapisa. Korištenjem ostalih senzora kao što je senzor ubrzanja moguće je detektirati naglo kočenje. Senzor za određivanje lokacije također omogućuje praćenje brzine vozila te na temelju tih podataka može se detektirati zastoj u prometu. Proširenjem funkcionalnosti uređaja te ugradnjom novih senzora i kombiniranim informacija s postojećih senzora moguće je doći do vrlo korisnih informacija prilikom kretanja prometnicama te na jednostavan način podijeliti prikupljene informacije s ostalim sudionicima prometa.

3.1. PAMETNI MOBILNI UREĐAJI

Pametni mobilni uređaji mogu se opisati kao mobilni uređaji sa zaslonom osjetljivim na dodir ili LCD (engl. *liquid cristal display*) zaslonom te ugrađenim programskim alatima (kao što su kalendar, imenik, elektronička pošta) uobičajenim za PDA (engl. *personal digital assistant*). Dolaze s instaliranim OS-om (engl. *operating system*) koji pruža podršku za instalaciju dodatnih korisničkih aplikacija raznih namjena. Pametni mobilni uređaj može se opisati kao mobilni

uređaj s ugrađenim računalom i OS-om (operacijskim sustavom). Na slici 3.1 prikazani su senzori pametnih mobilnih uređaja.

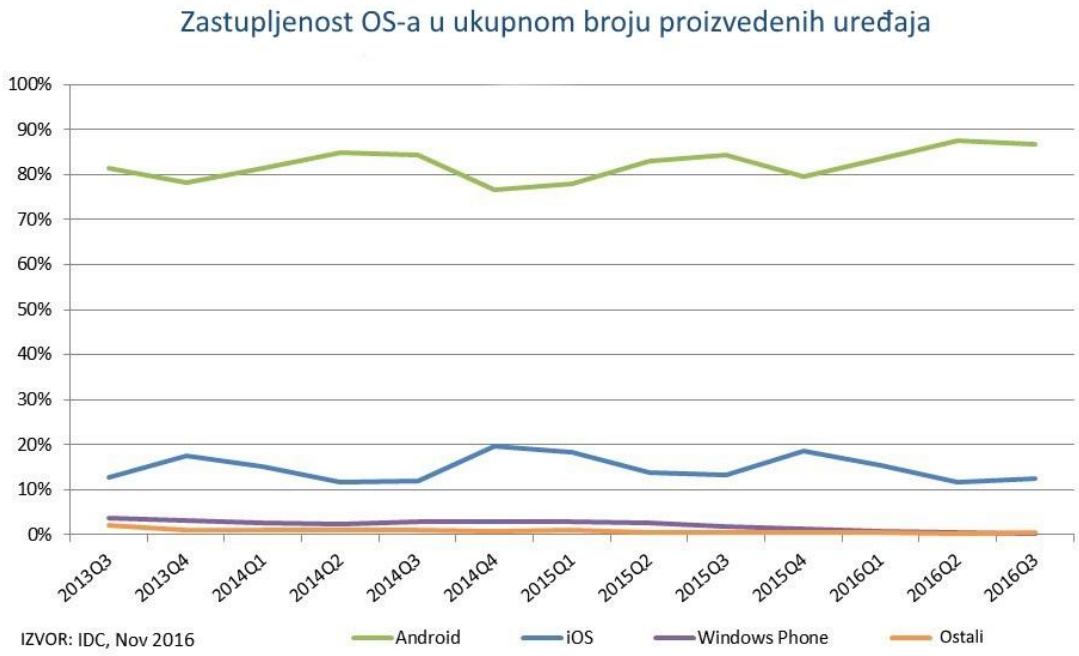
Slika 3.1. Senzori pametnih mobilnih uređaja

Prvi mobilni uređaj osmišljen od strane kompanije IBM (engl. *International Business Machines Corporation*) te pojavio na tržištu 1993. godine [14]. Uključivao je zaslon te je imao aplikacije za kalendar, imenik, kalkulator. Sazrijevanjem tržišta i padom cijene komponenti pametnih mobilnih uređaja dolazi do velikog napretka uređaja. Pristup internetu putem pametnih mobilnih uređaja je postao svakodnevica pogotovo nakon pojave 3G (engl. *third generation*) mobilnih mreža 2001. godine. 3G mobilna mreža omogućila je razmjenjivanje sadržaja kao što su fotografije, video zapisi, audio zapisi, elektronička pošta itd.

Uvođenjem 3G mobilne mreže povećana je brzina prijenosa podataka na oko 14 Mbps što je znatan napredak u odnosu na prethodne 2G mreže. Omogućen je prijenos svih tipova podataka s razmjerno stabilnom i pouzdanom vezom. Uvedeni su standardi UMTS i CDMA2000. Dolaskom 4G (engl. *forth generation*) mreža korisnicima su dostupne još veće brzine prijenosa podataka i stabilnija veza. Svakodnevno se radi na povećanju dostupnosti 4G mreža.

Pametni mobilni uređaji dolaze s virtualnom „QWERTY“ tipkovnicom za unos teksta koja se koristi za pisanje poruka, elektroničke pošte ili korištenje preglednika. Većina uređaja posjeduje ugrađenu kameru za snimanje slika i videa. Pristup internetu omogućen je putem mobilne mreže te bežičnih modula za spajanje na bežične mreže koje su u današnje vrijeme sveprisutne. Također, dolaze s ugrađenim senzorima kao što su brzinomjer (engl. *Accelerometer*), žiroskop (engl. *Gyroscope*), senzor za određivanje lokacije (engl. *GPS sensor*), kompas itd. Detalji o senzorima korištenim u ovom radu bit će opisani u jednom od narednih poglavlja.

Proizvođači isporučuju pametne mobilne uređaje s pripremljenim OS-om spremnim za instalaciju novih korisničkih aplikacija. Većina uređaja u današnje vrijeme dolazi s Android ili iOS operacijskim sustavom. Slikom 3.2. prikazana je zastupljenost pojedinih OS-a u odnosu na ukupan broj proizvedenih uređaja [15]. Velika većina uređaja dolazi s instaliranim Android OS-om te nakon toga iOS. Ostali OS-ovi imaju nisku zastupljenost.



Slika 3.2. Zastupljenost operacijskih sustava u ukupnom broju proizvedenih uređaja [15]

Velik broj uređaja pogonjen je Android OS-om zbog mogućnosti instalacije na razne tipove uređaja. iOS je privatni OS tvrtke Apple te je instaliran samo na njihove uređaje.

3.1.1. DETEKCIJA PROMETNE NESREĆE POMOĆU PAMETNOG MOBILNOG UREĐAJA

Velik broj smrtnih slučajeva u prometu indikacija je sve većeg broja prometnih nesreća. Detekcija prometne nesreće pomaže pri spašavanju ljudskih života na način da pravovremeno može obavijestiti nadležne službe o prometnoj nesreći i lokaciji gdje se dogodila.

U posljednjih deset godina na hrvatskim cestama dogodilo se oko 445 tisuća prometnih nesreća. U tim nesrećama nastradalo je 194 tisuće osoba: poginulo je 4706 osoba, teško je ozlijeđeno 34574 osoba, a 154720 osoba je lakše ozlijeđeno prema podacima MUP-a za 2015. godinu [16]. To je veoma velik broj poginulih te je potrebno poduzeti korake kako bi se taj broj što je moguće više smanjio.

Tradicionalni uređaji za detekciju prometne nesreće automatski obavještavaju nadležne službe. Koriste senzore za detekciju izbacivanja zračnih jastuka, okretanja automobila, udaraca automobila te pomoću njih otkrivaju prometnu nesreću. U današnje vrijeme mnoga vozila i dalje nisu opremljena takvom opremom. Opremanje vozila takvom opremom nije jeftino te mobilni uređaj opremljen mnoštvom senzora može poslužiti u tu svrhu. Slikom 3.3. prikazan je način otkrivanja prometne nesreće korištenjem pametnog mobilnog uređaja.

Slika 3.3. Detekcija prometne nesreće pomoću mobilnog uređaja

Osim detekcije prometne nesreće mobilni uređaj je u mogućnosti obavijestiti nadležne služe o prometnoj nesreći te na taj način osigurati brže pružanje pomoći prilikom nesreće [17]. Brži dolazak prve pomoći na mjesto nesreće pokazao se kao ključni faktor u spašavanju unesrećenih.

3.2. ANDROID PLATFORMA

Android je mobilni OS razvijan od strane Googlea, baziran je na Linux jezgri i dizajniran primarno za pametne mobilne uređaje i tablete. Android korisničko sučelje bazirano je na direktnoj manipulaciji korištenjem ekrana osjetljivih na dodir. Razvojem omogućena je upotreba na širokom broju uređaja kao što su televizija, automobili, pametni satovi [18].

Značenje riječi android dolazi iz grčkog, a sastoji se od riječi *andr* koja se prevodi kao čovjek ili muškarac te riječi *oid* koja ima se opisuje kao „sličnost“ ili „nalik na“ [19]. Koristi se opis čovjekolikog robota. Logo OS-a prikazan slikom 3.4. ukazuje na tu sličnost.

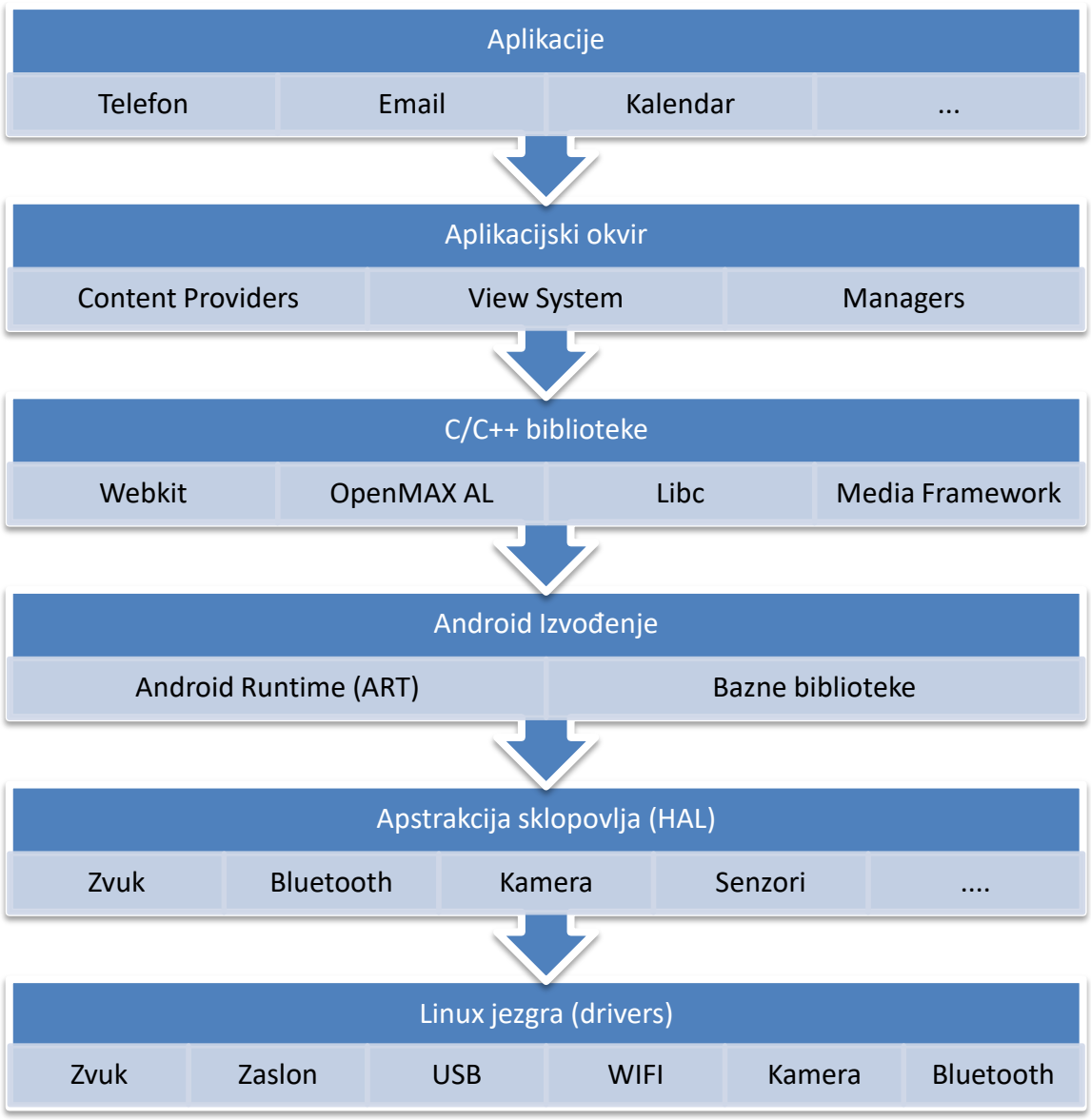
Slika 3.4. Logo Android operacijskog sustava [20]

Razvijen je od strane Android Inc. koju je 2005. godine kupio Google. Na tržištu se pojavljuje 2007. godine. Prvi uređaj s Android OS-om pušten je prodaju u studenom 2008. godine [18]. Od tada do danas objavljeno je veliki broj nadogradnji i novih verzija OS-a, a zadnja verzija je 7.0 Nougat [21]. Proširenjem i novim verzijama povećane su i mogućnosti OS-a te ugrađene nove funkcionalnosti i proširene postojeće. Android aplikacije instaliraju se na uređaj putem Google Play trgovine. U veljači 2017. broj dostupnih aplikacija je prešao 2.7 milijuna. Od 2013. godine postaje najzastupljeniji OS na tabletima. Postaje najzastupljeniji OS u svibnju 2017. s 2 bilijuna aktivnih korisnika [18].

Android OS pušten je u upotrebu pod licencom otvorenog koda (engl. *open source licence*). Na većinu uređaja ugrađena je modificirana Android verzija koja se sastoji od službene verzije te dodatnih komponenti ugrađenih od strane proizvođača mobilnih uređaja. Službena verzija Android OS-a instalirana je Google Nexus uređaje te novije Google Pixel uređaje.

3.2.1. ARHITEKTURA ANDROID PLATFORME

Arhitektura Android aplikacijskog sustava prikazana je slikom 3.5.



Slika 3.5. Arhitektura Android operacijskog sustava

Linux jezgra je osnova Android platforme. ART (engl. *Android Runtime*) oslanja se na Linux jezgru za funkcije kao što su višenitnost i upravljanje memorijom. Korištenje Linux jezgre donosi sigurnosne funkcionalnosti, a ujedno proizvođačima uređaja omogućuje razvoj programske podrške za dodatne komponente [22].

Apstrakcija sklopovlja ili HAL (engl. *Hardware apstraction layer*) pruža standardna sučelja za upravljanje sklopovljem putem Java aplikacijskog okvira. HAL se sastoji od nekoliko biblioteka koje pružaju sučelje za upravljanje određenom komponentom kao što je kamera ili

bluetooth. Kada aplikacijski okvir napravi zahtjev za korištenjem određene komponente Android učitava skupinu biblioteka za tu komponentu [22].

Uređaji na Android verziji 5.0 i nakon toga koriste ART (engl. *Android Runtime*) kako bi se aplikacije pokretale u odvojenim procesima sa zasebnim instancama. ART pokreće virtualne strojeve na uređajima s malom memorijom tako da izvršava DEX datoteke. DEX je *bytecode* dizajniran specifično za Android platformu i optimiziran za sustave s malom memorijom [22]. Prethodnik ART u verzijama manjim od 5.0 je Dalvik. Aplikacije pokretane ART-om moguće je pokrenuti i Dalvikom.

Biblioteke (engl. *libraries*) su skup C/C++ biblioteka koje omogućuju pristup komponentama Android sustava. Njihove mogućnosti su dostupne programerima putem aplikacijskog okvira. Primjer korištenja je *OpenGL ES* putem Android aplikacijskog okvira *Java OpenGL API* za podršku crtanja 2D i 3D grafika u aplikaciji [22].

Java aplikacijski okvir koriste programeri aplikacija. Kako je Android OS otvorenog tipa omogućuje potpunu kontrolu i modifikaciju postojećih funkcionalnosti. Dostupne su mogućnosti za određivanje lokacije, pokretanje servisa u pozadini, postavljanje alarma, prikazivanje notifikacija i još puno toga. Programeri imaju puni pristup skupu aplikacijskih funkcija korištenim za razvoj osnovnih aplikacija. Arhitektura aplikacija implementirana da otvara mogućnosti ponovnog korištenja. Mogućnosti pojedine aplikacije moguće je otvoriti za korištenje u ostalim aplikacijama te na taj način iskoristiti postojeće aplikacije za određene funkcije. Primjer takve mogućnosti je aplikacija za kameru. Dovoljno je pozvati kroz nekoliko linija koda aplikaciju za kameru te će se ona pobrinuti za snimanje slike ili videa.

Aplikacijski sloj je vidljiv krajnjem korisniku OS-a. To su aplikacije poput imenika, telefona, poruka, kalkulatora, karte, internet preglednik. Android OS dolazi s ugrađenim osnovnim aplikacijama. Dodatne aplikacije moguće je instalirati putem servisa Google Play.

3.2.2. **SENZORI**

Većina uređaja s Android OS-om posjeduju ugrađene senzore za pokret, orijentaciju i određivanje raznih događaja u okolini. Senzori pružaju pristup neobrađenim podacima prikupljenim iz neposredne blizine uređaja. Podaci kao što su trodimenzionalni pokret i pozicija u prostoru te podaci o okolini uređaja. Kao primjer može poslužiti igra koja može detektirati rotaciju mobitela te na taj način omogućiti korisniku upravljanje virtualnim automobilom. Aplikacija za putovanja može koristiti senzor za određivanje geomagnetskog polja i senzor ubrzanja za određivanje azimuta.

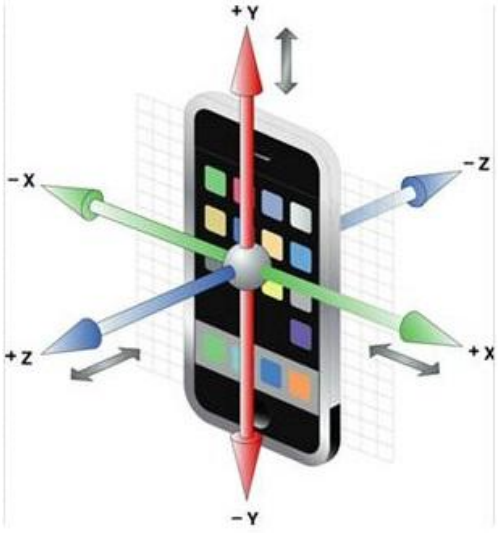
Android platforma podržava tri kategorije senzora [23]:

- Senzori pokreta – mjere ubrzanje i rotaciju uređaja. Ova kategorija uključuje senzore ubrzanja, gravitacijske senzore, žiroskop, senzore za rotaciju.
- Senzori okoline – mjerenje parametara okoline kao što su temperatura zraka i pritisak, osvjetljenje, vlaga. U ovu kategoriju spadaju barometar, senzor za određivanje količine svijetlosti i termometar.
- Senzori lokacije – određivanje fizičke lokacije uređaja. U ovu kategoriju ulaze senzori orijentacije i magnetometri [24].

Pristup senzorima dostupnim na mobilnom uređaju i podacima sa senzora omogućen je pomoću Android senzor okvira. Senzor okvir (engl. *Sensor Framework*) pruža klase i sučelja za upravljanje senzorima.

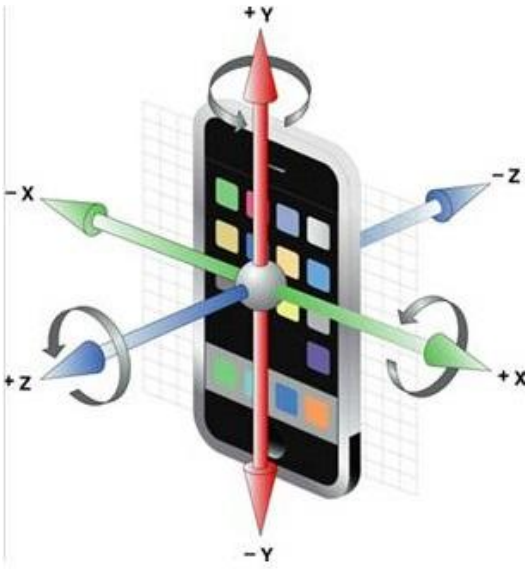
Android senzor okvir pruža pristup velikom broju senzora. Neki od tih senzora su direktno vezani uz komponente uređaja dok su neki omogućeni kroz programsku podršku. Senzori omogućeni kroz programsku podršku nisu direktno vezani uz komponentu uređaja, nego koriste više senzora za prikupljanje podataka te kombinacijom komponenti pružaju informacije.

Na slici 3.6. prikazan je senzor ubrzanja. Takav tip senzora mjeri ubrzanje uređaja prema tri osi (x, y, z). Senzor ugrađen u mobitel mjeri silu ubrzanja uzrokovanu gravitacijom, pomicanjem ili okretom koristeći seizmičku masu. Kako je senzor priključen unutar mobitela mjerenja koja očitava mogu se primijeniti na uređaj.



Slika 3.6. Prikaz rada senzora ubrzanja

Žiroskop je senzor kojim se mjeri rotacija uređaja po tri osi (x, y, z). Način rada žiroskopa prikazan je slikom 3.7. Senzor ugrađen u uređaj baziran je na Coriolis efektu, koji je uzrokovan rotacijom zemlje. Mjereći promjene u kretanju vibrirajućih uređaja uzrokovane rotacijom uređaja i Coriolis efektom prikupljaju se mjerenja žiroskopa.



Slika 3.7. Prikaz rada žiroskopa

3.2.3. **ANDROID RAZVOJNO OKRUŽENJE**

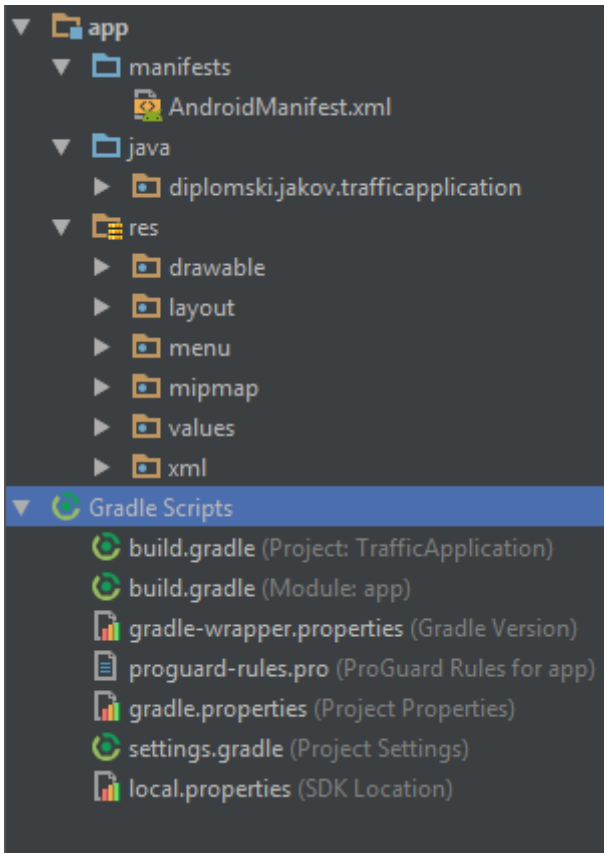
Android Studio je službeni IDE (engl. *Integrated Development Environment*) za razvoj Androd aplikacija, baziran na IntelliJ IDEA [25]. Povrh IntelliJ moćnog editora i razvojnih alata, Android Studio nudi još dodatnih mogućnosti koje pomažu u povećanju produktivnosti pri izgradnji Android aplikacija [26].

Neke od dodatnih mogućnosti su [25]:

- fleksibilni sustav za izgradnju aplikacija baziran na Gradle [27]
- brz i multifunkcionalan emulator
- univerzalno okruženje za razvoj svih vrsta Android aplikacija
- *Instant Run* za primjenu promjena nad aplikacijom tokom rada
- predlošci koda i integracija s GitHubom za pomoć pri izgradnji uobičajenih komponenti
- alati i okviri za testiranje
- Lint alati za praćenje performansa aplikacije
- C++ i NDK podrška.

Struktura Android projekta sastoji se od modula sa izvornim kodom i resurs datoteka. Android projekt sastoji se od Gradle skripti (engl. *GradleScripts*) i modula sa aplikacijom što je vidljivo na slici 3.8. Svaki od modula aplikacije sadrži slijedeće direktorije [25]:

- *manifests* koji sadržava datoteku s postavkama i sadržajima aplikacije
- *java* direktorij unutar kojeg se nalaze datoteke s izvornim *Java* kodom i testovima
- *res* direktorij unutar kojeg se nalaze resursi aplikacije kao što su *XML layouts*, tekstovi aplikacije, slike…



Slika 3.8. Prikaz projekta u Android Studio razvojnom okruženju

Android Studio koristi Gradle sustav za izgradnju aplikacija [27]. Sustav za izgradnju moguće je koristiti direktno iz Android Studio, ali i neovisno o njemu kao *Comman line* alat. Mogućnosti Gradle sustava za izgradnju su slijedeće [26]:

- uređivanje, konfiguracija i proširenje sustava za izgradnju aplikacije
- izrada više verzija aplikacije iz istih izvornih datoteka
- iskorištavanje istih resursa i koda na više mjesta.

Gradle mogućnosti izvršavaju se neovisno o izvornom kodu aplikacije. Unutar Android Studio datoteke sustava za izgradnju aplikacije imaju naziv *build.gradle*. One su obične tekstualne datoteke koje sadrže *Groovy* sintaksu za konfiguraciju sustav za izgradnju [28]. Svaki projekt sadržava jednu datoteku za izgradnju na razini projekta i pojedine datoteke za izgradnju svakog modula.

4. INTELIGENTNI TRANSPORTNI SUSTAV ZA DIJELJENJE PROMETNIH INFORMACIJA U STVARNOM VREMENU POMOĆU PAMETNIH MOBILNIH UREĐAJA

Sustav izgrađen u ovom radu definiran je kao iMANET (engl. *Internet-based mobile ad-hoc network*) i Manet je tip bežične ad-hoc mreže u kojoj su mobilni čvorovi međusobno povezani koristeći internet protokole. Konkretna implementacija se temelji na skupini mobilnih uređaja povezanih u ad-hoc mrežu koristeći BaaS ili MBaaS [29](engl. *backend as a service* ili *Mobile backend as a service*).

BaaS su sustavi bazirani na računalstvu u oblaku gdje se korisnicima daje pristup gotovim rješenjima putem API-ja (engl. *application programming interfaces*). Upravljanje korisnicima sustava, registracija i aktivacija novih korisnika, prijava postojećih korisnika te upravljanje svim potrebnim resursima i podacima u sustavu omogućeno je koristeći BaaS.

Uređaji sustava povezani su putem servisa u oblaku. Na taj način uređaji međusobno nisu povezani direktnom vezom. Razmjena podataka i informacija unutar sustava obavlja se preko centralnog računala u oblaku koje služi kao posrednik kod međusobne komunikacije mobilnih uređaja.

Konkretna implementacija ovog sustava sastoji se od računala u oblaku koje u mogućnosti identificirati korisnika koristeći korisničko ime i lozinku, preuzeti slike i video zapise poslane od strane korisnika te primiti dodatne podatke vezane uz poslani resurs. Omogućen je javni pristup resursima poslanim od strane korisnika te na taj način iskoristivost na bilo kojoj platformi ili sustavu.

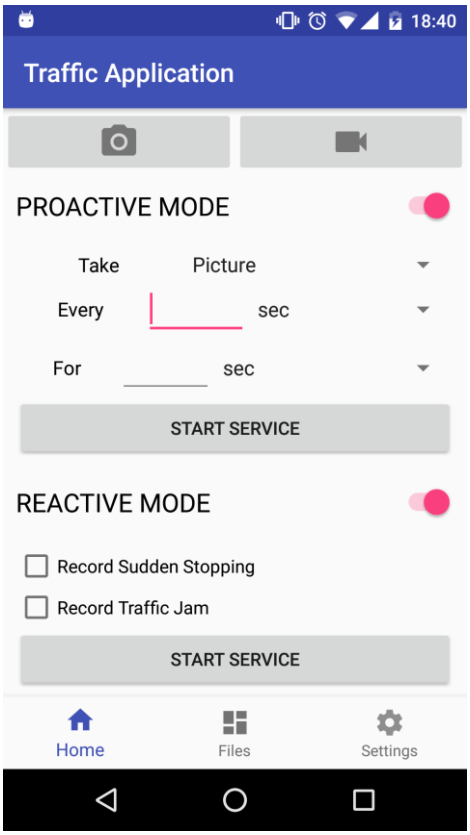
Izrađena je mobilna aplikacija za Android platformu koja prikuplja podatke te je putem internet protokola povezana s računalom u oblaku. Aplikacija nakon registracija i prijave korisnika u sustav ima mogućnost snimanja video zapisa i slika. Pruža podršku za snimanje u nekoliko načina rada. Svaki od načina moguće je dodatno prilagoditi uređivanjem postavki aplikacije. Snimljeni zapisi se automatski ili ovisno o postavkama šalju udaljenom računalu.

4.1. ANDROID MOBILNA APLIKACIJA

Mobilna aplikacija je napravljena za Android OS koji je trenutno najzastupljenija platforma u svijetu za mobilne uređaje. Za vrijeme rada aplikacije uređaj je potrebno postaviti na prednje vjetrobransko staklo vozila. Uređaj se pričvršćuje posebnim držačem koji se postavlja na staklo,

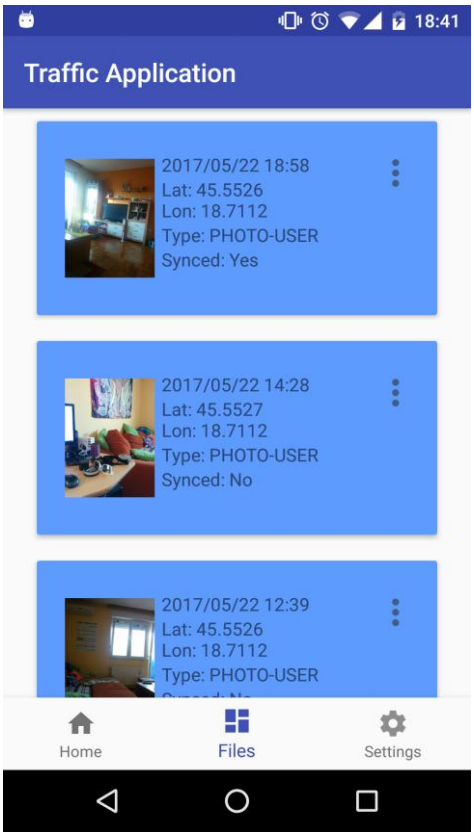
a nakon toga uređaj u držač. Stražnja kamera uređaja služi za snimanje prometnice i mora biti okrenuta prema prometnici. Uređaj je potrebno postaviti u vodoravni ili okomiti položaj.

Rad aplikacije podijeljen je u tri načina rada. Prvi način rada je korisnički u kojem je korisniku aplikacije omogućeno samostalno slikanje slika i video zapisa. Drugi način rada je proaktivni prilikom kojeg aplikacija samostalno snima zapise. Snimanje se događa u određenom vremenskom intervalu određenom prilikom pokretanja servisa. Treći način rada je reaktivni u kojem aplikacija također samostalno uzima zapise. Snimanje se događa ovisno o nekom vanjskom događaju kao što je naglo kočenje ili zastoj u prometu. Grafičko sučelje za pokretanje snimanja prikazano je slikom 4.1.



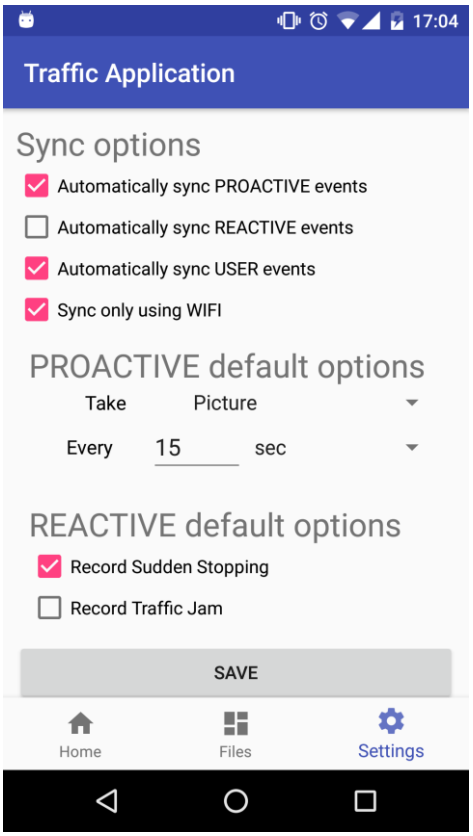
Slika 4.1. Grafičko sučelje pokretanje snimanja

Unutar aplikacije je omogućen pregled snimljenih zapisa. Slike i video zapisi spremaju se na memoriju uređaja, a prikazani su unutar aplikacije u obliku liste. Grafičko sučelje za pregled snimljenih zapisa prikazano je na slici 4.2. Uz zapis spremaju se i dodatni podaci o lokaciji na kojoj je snimljen, vremenu kada je snimljen, načinu rada korištenom za snimanje te dodatni podaci potrebni za sinkronizaciju. Detaljniji opis spremanja i prikaza bit će opisan u jednom od nadolazećih poglavlja.



Slika 4.2. Grafičko sučelje za pregled snimljenih zapisa

Korisnik je u mogućnosti odabrati početne postavke za svaki od načina rada. Grafičko sučelje postavki aplikacije prikazano je slikom 4.3. Osim postavki za način rada dostupne su i postavke za sinkronizaciju koje služe za reguliranje potrošnje mobilnog internet prometa.

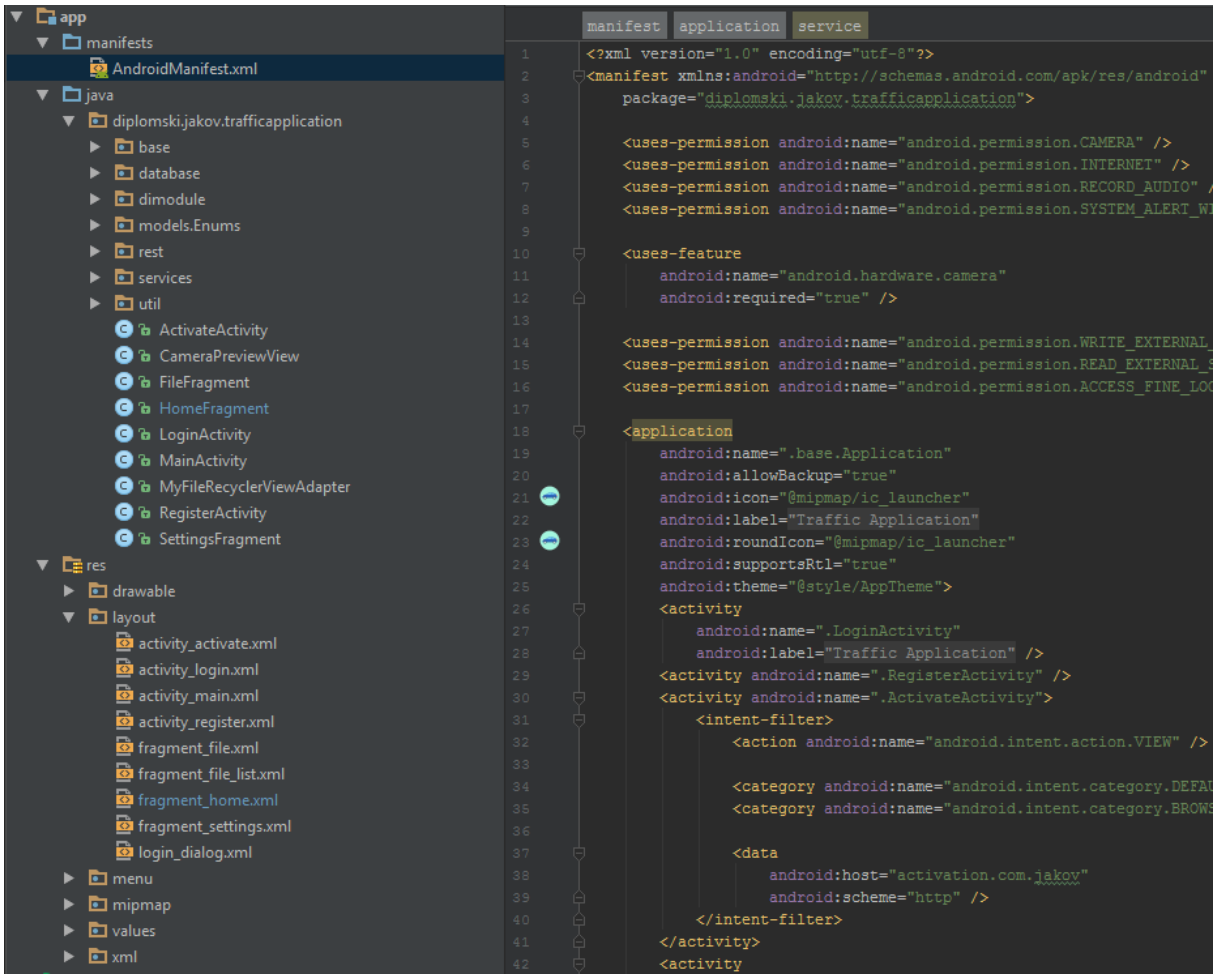


Slika 4.3. Grafičko sučelje postavki aplikacije

Grafičko sučelje aplikacije izvedeno je na način da se koristi jedan glavni *Activity* za prikaz 3 *Fragmenta* s navigacijom na dnu ekrana što je vidljivo na slikama 4.1., 4.2., 4.3. U ovisnosti o kliku na određenu ikonu izmjenjuju se tri navedena *Fragmenta*. Unutar aplikacije postoji grafički dio za registraciju, aktivaciju i prijavu korisnika koji će biti detaljnije opisan i popraćen slikama u jednom od nadolazećih poglavlja.

4.1.1. ARHITEKTURA APLIKACIJE

Arhitektura svake Android aplikacije sastoji se tri glavne komponente. Manifest datotekom su definirane mogućnosti aplikacije te dodatne postavke poput dopuštenja i minimalne verzije OS-a. Direktoriji unutar kojih se nalaze datoteke s Java izvornim kodom aplikacije. Java kodom definiramo željena svojstva i mogućnosti aplikacije. Direktoriji s resursima aplikacije kao što su grafički opisi zaslona aplikacije te dodatni resursi kao što su slike, tekstovi i prijevodi tekstova, ikone, definicije boja i stilova korištenih u aplikaciji. Prikaz arhitekture projekta unutar razvojnog okruženja te Manifest datoteke vidljiv je na slici 4.4.



Slika 4.4. Prikaz arhitekture unutar razvojnog okruženja

Manifest datoteka sadržava nekoliko glavnih karakteristika aplikacije te je zapisana u *xml* [30] opisnom jeziku. Na početku su navedena potrebna dopuštenja za rad aplikacije. Korisnik prilikom prvog pokretanja aplikacije mora odobriti korištenje funkcija sustava. Za korištenje aplikacije potrebna sljedeća dopuštenja:

- pristup kameri (*android.permission.CAMERA*) da bi aplikacija mogla samostalno slikati slike i video
- pristup snimanju zvuka (*android.permission.RECORD_AUDIO*) kako bi se uz video mogao snimiti i zvuk
- pristup zapisivanju i čitanju dodatne (većinom micro SD kartica) memorije uređaja (*android.permission.WRITE_EXTERNAL_STORAGE*, *android.permission.READ_EXTERNAL_STORAGE*) na koju se spremaju slike i video zapisi

- pristup preciznoj lokaciji uređaja (*android.permission.ACCESS_FINE_LOCATION*) radi određivanja točne lokacije koje se povezuje sa zapisom
- pristup prikazu sistemskih prozora (*android.permission.SYSTEM_ALERT_WINDOW*) jer prilikom rada aplikacije u pozadini potrebno je korisniku barem na kratko prikazati prozor sa zabilježenim sadržajem.

Osim navedenih dopuštenja na verzijama Androida novijim od 6.0 korisnik mora ručno omogućiti iscrtavanje prozora preko aktivnih aplikacija. Nakon pokretanja aplikacije korisnik je preusmjeren na postavke sustava (*Settings.ACTION_MANAGE_OVERLAY_PERMISSION*) kako bi omogućio iscrtavanje prozora.

Nakon dopuštenja unutar manifest datoteke navedene su osnovne postavke i sadržaj aplikacije. Definira se objekt *application* s postavkama aplikacije poput naziva, ikone, teme, a unutar njega pojedine komponente aplikacije. Navode se komponente poput *activitya* i *service* za postavke pozadinskih servisa. Unutar *activity* objekta nalaze se postavke za naziv, način izvođenja i pozivanja te poveznica na pripadajuću Java datoteku s izvornim kodom.

Unutar java direktorija nalaze se Java datoteke sa izvornim kodom. Struktura direktorija definirana je paketima (engl. *packages*). Svaka datoteka s izvornim kodom ima definiran paket u kojem se nalazi. Svaki paket može sadržavati druge pakete, a nazivi paketa odvojeni su točkom. Hijerarhija paketa predstavlja strukturu direktorija na disku unutar kojih se nalaze Java datoteke. Glavni paket aplikacije naziva *diplomski.jakov.trafficapplication*sadržava Java klase koje se koriste kao glavni zasloni aplikacije poput *MainActivity* koji je glavni *activity* za prikaz opisanih *fragmenta* za postavke, pokretanje servisa i pregled snimki. Osim klasa za prikaz aplikacije glavni paket sadržava i sljedeće pakete:

- *base* paket koji sadržava java klasu aplikacije i baznu klasu za ostale *activitye*
- *database* paket unutar kojeg su klase za upravljanje lokalnom bazom podatka
- *dimodule* paket koji sadrži klase koje definiraju DI (engl. *dependency injection*)
- *models.Enums* paket unutar kojeg su enumi korišteni u aplikaciji
- *rest* paket koji je dodatno podijeljen u dva paketa
    - *models* paket unutar kojeg se nalaze modeli podataka za prilikom komunikacije putem interneta
    - *services* paket sa servisima koji se koriste za prilikom mrežne komunikacije i prijenosa podataka s uređaja na udaljeno računalo

- *services* paket sa servisima koji se koriste unutar aplikacije kao što je *ReactiveService* za snimanje u reaktivnom načinu rada
- *util* paket s klasama opće upotrebe

Direktorij s resursima aplikacije sadržava opise grafičkog sučelja aplikacije pisane u *xml*-u, slike, ikone i tekstove aplikacije. Sadržaj direktorija podijeljen je na nekoliko poddirektorija, a neki od njih su:

- *drawable* direktorij unutar kojeg se nalaze slike i ikone aplikacije u različitim veličinama kako bi se podržala raznolikost veličina ekrana Android uređaja
- *layout* direktorij koji sadržava opise grafičkog sučelja pojedinih elemenata aplikacije pisanih u *xml*-u
- *values* direktorij unutar kojeg se nalaze tekstovi, boje i stilovi korišteni u aplikaciji.

Dodatne biblioteke aplikacije navedene su u datoteci *build.gradle*. Gradle datoteka koristi se za definiranje procesa izgradnje aplikacije te određivanje dodatnih biblioteka aplikacije. Programski kod 4.1. prikazuje *gradle* datoteku aplikacije.

```
android {
compileSdkVersion 25
buildToolsVersion "25.0.3"
defaultConfig {
applicationId "diplomski.jakov.trafficapplication"
minSdkVersion 15
targetSdkVersion 25
versionCode 1
versionName "1.0"
    }
buildTypes {
release {
minifyEnabledfalse
proguardFilesgetDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
compilefileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
excludegroup: 'com.android.support', module: 'support-annotations'
    })
compile 'com.android.support:appcompat-v7:25.3.1'
compile 'com.android.support:design:25.3.1'
...
annotationProcessor 'android.arch.persistence.room:compiler:1.0.0-alpha1'
provided 'javax.annotation:jsr250-api:1.0'
}
```

Programski kod 4.1. Gradle datoteka

Unutar datoteke navedena je verzija Android SDK alata te zadana konfiguracija za izgradnju aplikacije. Vidljivo je da je minimalna verzija potrebna za pokretanje aplikacije 15, a zadana

verzija 25. Od dodatnih biblioteka bitno je istaknuti neke koje su naviše korištene. Pomoćne biblioteke izdane službeno od Google-a, a koriste se za podršku pri razvoju aplikacija za starije verzije Android platforme.

Retrofit biblioteka koristi se za komunikaciju s udaljenim računalom putem API-a [31]. Prilikom implementacije dovoljno je definirati *interface* koji opisuje strukturu API *endpointa*, model koji odgovara podacima koje vraća te parametre koje prima. Programskim kodom 4.2. prikazana je implementacija *endpointa* za prijavu korisnika.

```
@FormUrlEncoded
@POST("login")
Call<LoginModel> login(@Field("username") Stringusername, @Field("password") String
password, @Field("grant_type") Stringtype);
```

Programski kod 4.2. Primjer *endpointa* za prijavu korisnika

Potrebno je definirati metodu koja se koristi za izvršavanje *requesta* u ovom je slučaju to POST metoda. Definiran je model podatka koji se dobije u *response* te parametri *requesta* koji su zadani kao parametri funkcije. Na ovaj način omogućeno je jednostavno korištenje API-a te razmjena podataka. Potrebno je postaviti i zadane postavke prilikom izvršavanja *requesta* te je moguće automatsko dodavanje *heardera* za autorizaciju.

Dagger biblioteka se koristi za DI (engl. *dependency injection*) unutar aplikacije. Omogućuje ubacivanje potrebnih servisa i svih njihovih ovisnosti na jednostavan način. U posebnoj klasi definiraju se servisi i datoteke o kojima ovise te se u takvom obliku mogu direktno ubaciti u klase gdje su potrebni. Kreiranjem takve strukture moguće je jednostavan način iskoristiti široke mogućnosti servisa u svim dijelovima aplikacije.

Play Services bibliotekom iskorištene su mogućnosti Google servisa kako bi se brže i točnije odredila lokacija uređaja. Google servisi bilježe podatke o lokaciji prilikom korištenja iz stranih aplikacija te na taj način omogućuju brže i točnije određivanje lokacije uređaja.

ButterKnife biblioteka služi za jednostavnije povezivanje grafičkog izgleda s Java programskim kodom. Koriste se anotacije za povezivanje izgleda i konkretnog objekta.

Room biblioteka nedavno izdana od Googlea iskorištena je za kreiranje i upravljanje lokalnom bazom podataka na jednostavan način. Potrebno je kreirati klasu koja nasljeđuje *RoomDatabase* i predstavlja baznu klasu za upravljanje bazom podataka. Osim bazne klase za bazu potrebne su DAO klase (engl. *dana base access object*) za pristup i upravljanje pojedinim tablicama unutar baze. DAO klase podatke iz baze prebacuju u modele (entitete) te na taj način omogućuju jednostavnije upravljanje informacijama iz Java koda. Programski kod 4.3. prikazuje osnovni entitet lokalne snimke unutar aplikacije.

```
@Entity
publicclassLocalFile {
    @PrimaryKey(autoGenerate = true)
  publiclongid;

  publicFileTypefileType;

  publicStringfileName;

  publicStringfileExtension;

  publicStringlocalURI;

  publicdoublelatitude;

  publicdoublelongitude;

  publicfloataccuracy;

  publicStringlinkToFile;

  public Date dateCreated;

  publicbooleansync;

  publicRecordTyperecordType;

  publicLocalFile() {
    }
}
```

Programski kod 4.3. Entitet lokalne snimke

Definirana su svojstva (engl. *property*) odnosno kolone u tablici s podacima o zapisu:

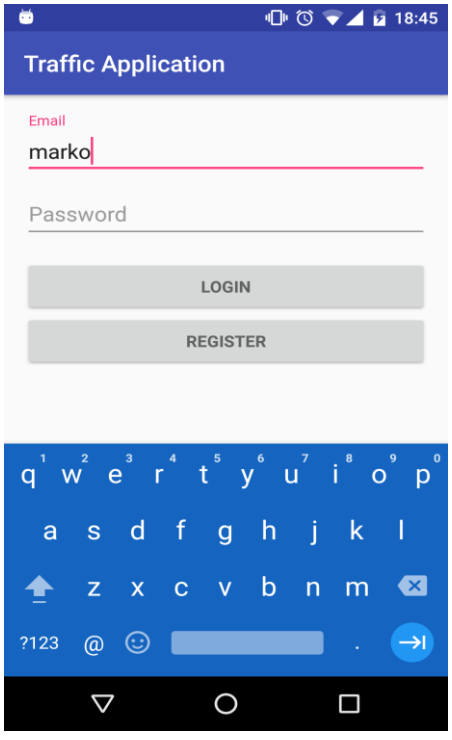
- *id* koji predstavlja univerzalni broj svakog zapisa u tablici, automatski generiran
- *fileType* enum koji definira vrstu spremljene datoteke, moguće vrste su VIDEO i PHOTO
- *fileName* sadrži naziv spremljene datoteke
- *fileExtension* ekstenziju spremljene datoteke, u slučaju slika će to biti .jpg, a za video zapise .mp4
- *localURI* je potpuna putanja spremljenog zapisa na uređaju
- *latitude* je geografska širina na kojoj je snimljen zapis
- *longitude* je geografska dužina na kojoj je snimljen zapis
- *accuracy* je broj u metrima koji određuje preciznost GPS u trenutku uzimanja lokacije
- *linkToFile* sadrži URL (engl. *Uniform Resource Locator*) preko kojeg je moguće pristupiti zapisu na udaljenom računalu
- *dateCreated* predstavlja datum i vrijeme kada je zapis snimljen
- *sync* služi vođenje evidencije o tome dali je lokalni zapis poslan na udaljeno računalo

- *recordType* enum određuje način rada u kojem je snimljen zapis, moguće vrijednosti su: USER, PROACTIVE, REACTIVE.

Za spremanje postavki aplikacije korišten je *PreferenceService*. Taj service koristi *SharedPreferences* [32] klasu Android sustava kako bi spremio postavke aplikacije. Postavke se spremaju u obliku jednostavnih tipova podataka. Svaka pojedina postavka sastoji se od univerzalnog ključa i vrijednosti koja može biti samo jednostavni tip podatka. Postavke se unutar sustava spremaju u *xml* zapisu te nisu omogućeni kompleksni tipovi podataka. Ako se javi potreba za spremanjem objekata kao postavki aplikacije moguće je vrijednost objekta serializirati. Na taj način moguće je spremiti objekt u obliku teksta te prilikom dohvaćanja ponovno deserializirati tekst natrag u objekt.

4.1.2. UPRAVLJANJE KORISNICIMA

Prilikom prvog pokretanja aplikacije korisnik je usmjeren na zaslon za prijavu vidljiv na slici 4.5. Korisniku je ponuđena opcija za unos korisničkog ime i lozinke te nakon toga prijava ili registracija.

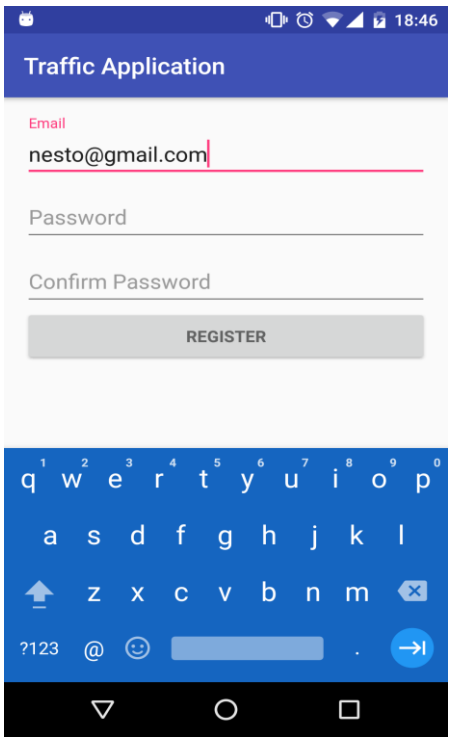


Slika 4.5. Prijava korisnika

Prijava korisnika odrađuje se putem *endpointa* za prijavu. Metoda *requesta* je POST, a potrebno je dodati anotaciju *FormUrlEncoded* koja će postaviti MIME tip *requesta* na

application/x-www-from-urlencoded. Potrebno je poslati i parametre koji su: *username, password, grand_type*. Parametar *grand_type* potrebno je postaviti na *password*. Prilikom uspješne prijave korisnika poslužitelj nam vraća *response* sa HTTP status kodom 200, a unutar *body*-a nalazi se token za pristup (*access_token*) i dodatni podaci o isteku tokena. Ukoliko su korisnički podaci nevažeći poslužitelj će vratiti *response* s HTTP status kodom 400 i opisom pogreške. Token za pristup sprema se u internu memoriju aplikacije te će se koristiti prilikom izvršavanja ostalih *requesta* prema poslužitelju. Putem tokena poslužitelj identificira korisnika, a dodaje se na svaki *request* unutar zaglavlja (engl. *header*) s nazivom *Authorization*. Ovakav tip prijave definiran je OAuth 2.0 standardom[33].

Ukoliko korisnik nema kreiran korisnički račun odabire *REGISTER* te je preusmjeren na zaslon na kojem je potrebno popuniti korisničke podatke. Zaslon s podacima za registraciju prikazan je slikom 4.6.



Slika 4.6. Registracija korisnika

Prilikom registracije korisnik mora unijeti adresu elektroničke pošte, lozinku te ponovno upisati lozinku kao potvrdu. Naknadno će se upisana adresa elektroničke pošte koristiti kao korisničko ime za prijavu u sustav. Nakon što je korisnik unio podatke i stisnuo *REGISTER* izvršava se *request* prema poslužitelju. Metoda *requesta* je POST. Poslužitelj nam vraća *resposnse* s odgovarajućim HTTP status kodom. Ukoliko je registracija uspješno određena

korisnik će na svoju adresu elektroničke pošte primiti poruku s uputama i linkom za aktivaciju. Link u poruci potrebno je otvoriti putem mobilne aplikacije. Nakon što korisnik klikne na link za aktivaciju preusmjeren na aplikaciju gdje završava kreiranje korisničkog računa dodirom na *ACTIVATE*. Unutar linka za aktivaciju nalazi se aktivacijski token koji se koristi za izvršavanje PUT *requesta* prema poslužitelju. Na taj način korisnik je potvrdio svoju adresu elektroničke pošte i završio registraciju u sustavu. Nakon uspješne aktivacije korisnički račun je kreiran i aktiviran te je moguća prijava.

4.1.3. KORISNIČKI NAČIN RADA

Mobilna aplikacije daje mogućnost korisniku samostalno uzimanje zapisa. Na početnom zaslonu aplikacije prikazanom slikom 4.1. može se odabrati jedan od dva gumba na vrhu ekrana. Odabir gumba sa slikom fotoaparata pokreće snimanje slike u korisničkom načinu rada. Odabirom gumba sa slikom kamere pokreće se snimanje video zapisa u korisničkom načinu rada. Snimanje u ovom načinu rada omogućeno je putem sistemske podrške kamere. Dovoljno je kreirati *Intent* s odgovarajućom akcijom koja poziva zadanu aplikaciju za kameru. Primjer koda za poziv kamere prikazana je programskim kodom 4.4.

```
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
if (takePictureIntent.resolveActivity(getActivity().getPackageManager()) != null) {
     LocalFileService.FileModelphotoFile =
localFileService.createImageFile(RecordType.USER);
     if (photoFile != null&&photoFile.file != null) {
           Uri photoURI = FileProvider.getUriForFile(getActivity(),
                        "diplomski.jakov.trafficapplication.fileprovider",
                        photoFile.file);
           takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
           localFile = photoFile.localFile;
           startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO);
     }
}
```

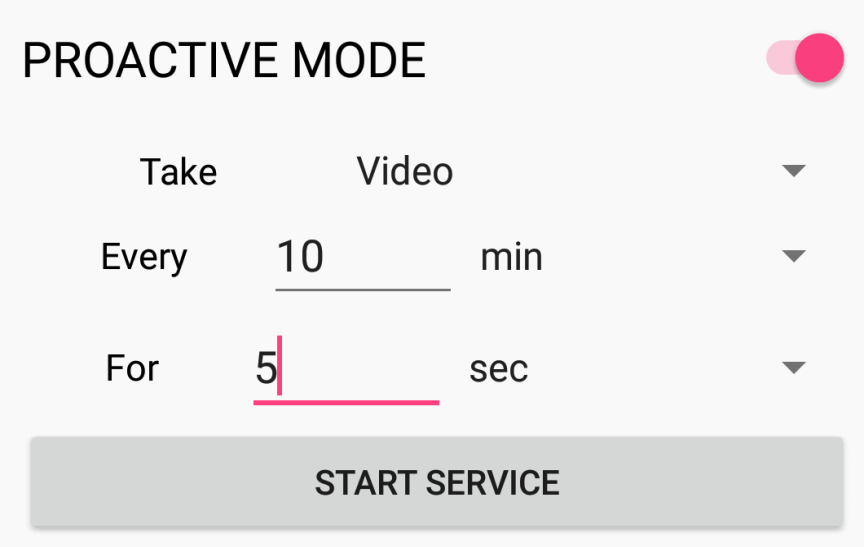
Programski kod 4.4. Poziv zadane aplikacije za kameru

Potrebno je proslijediti lokaciju na uređajugdje želimo da se spremi slika te kreirati *Intent* s akcijom *MediaStore.ACTION_IMAGE_CAPTURE*. Pozivom *Intenta* korisnik napušta trenutnu aplikaciju i preusmjeren je na aplikaciju kamere. Nakon što je zapis uspješno snimljen i potvrđen korisnik je preusmjeren natrag u aplikaciju za praćenje prometa. Kreiranom zapisu određuje se lokacija koristeći *Google location API*. Podaci o lokaciji zapisa u sustavu, mjestu zapisa i vremenu spremaju se u lokalnu bazu. Zapis je moguće vidjeti na zaslonu sa svim zapisima, a ako je dopuštena automatska sinkronizacija pokreće se slanje poslužitelju.

Zapisi kreirani u korisničkom načinu rada spremaju se na zadanu lokaciju gdje i ostale korisnikove slike i video zapisi. Ovo se događa samo u slučaju korisničkog načina rada. Na ovaj način korisniku su dostupni snimljeni sadržaji i iz ostalih aplikacija.

4.1.4. PROAKTIVNI NAČIN RADA

Proaktivni način rada snima zapise u jedinici vremena. Korisnik odabire vrstu zapisa (slika, video), interval snimanja te dužinu trajanja ako se radi o video zapisu. Potrebno je unijeti interval (broj) i odabrati mjernu jedinicu (sekunda, minuta, sat). Na isti način odabire se i družina video zapisa. Slikom 4.7. prikazane su postavke proaktivnog načina rada. Za zadane postavke aplikacija će snimati video zapis svakih 10 minuta. Dužina video zapisa bit će 5 sekundi.



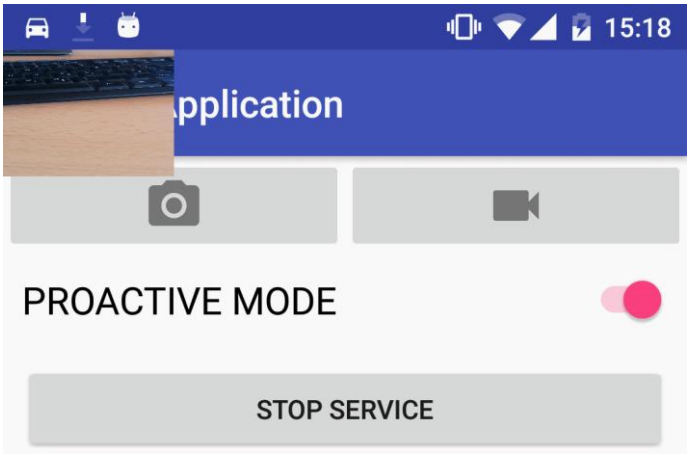
Slika 4.7. Postavke proaktivnog načina rada

Nakon odabira željenih postavki rada potrebno je dodirnuti gumb *START SERVICE* kako bi se pokrenuo servis. Pokreće se servis koji je neovisan o grafičkom sučelju i ostatku aplikacije. Korisnik je obaviješten o radu servisa putem stalne notifikacije koja mu je vidljiva na uređaju za vrijeme rada servisa. U slučaju gašenja aplikacije od strane Android OS-a servis će neovisno nastaviti raditi. Implementirani servis nasljeđuje klasu *android.app.Service* [34] te su mi prilikom pokretanja proslijeđeni odabrani parametri. Određivanje intervala i određivanje vremena snimanja implementirano je putem *Handlera* te je vidljivo na programskom kodu 4.5. Za postavljanje početne vrijednosti koristi se *switch* grananje kako bi se odredio točan interval za određenu mjernu jedinicu. Zadani interval sprema se u *long* varijablu zbog veličine podatka izraženog u milisekundama.

```
longintervalInMills = interval;
switch (timeUnits) {
      case SEC: intervalInMills *= 1000; break;
      case MIN: intervalInMills *= 1000 * 60; break;
      case HOUR: intervalInMills *= 1000 * 60 * 60; break;
}
handler = newHandler();
final long finalIntervalInMills = intervalInMills;
handlerRunnable = newRunnable() {
      @Override
      publicvoidrun() {
            cameraPreviewView.takeRecord(RecordType.PROACTIVE, fileType,
videoDurationUnits, forInterval);
            handler.postDelayed(this, finalIntervalInMills);
      }
};
handler.post(handlerRunnable);
```

Programski kod 4.5. Određivanje vremena snimanja u proaktivnom modu

Početni interval izražen u milisekundama množi se sa odabranom mjernom jedinicom za vrijeme kako bi se dobio puni interval u milisekundama. Nakon određivanja vremena u milisekundama pokreće se *Handler* s odgodom pokretanja. Prilikom svakog pokretanja snima se zapis te se ponovno pokreće odgoda za zadani interval. Tijekom rada prikazana je stalna notifikacija koja nudi opciju zaustavljanja rada servisa. Uzimanje zapisa putem kamere moguće je samo ako je korisniku prikazan pregled snimljenog sadržaja. U tu svrhu kreiran je sistemski prozor koji služi kao pregled snimljenog sadržaja. Prozor s pregledom postavljen je u gornji desni kut ekrana uređaja, a vidljiv je na slici 4.8.



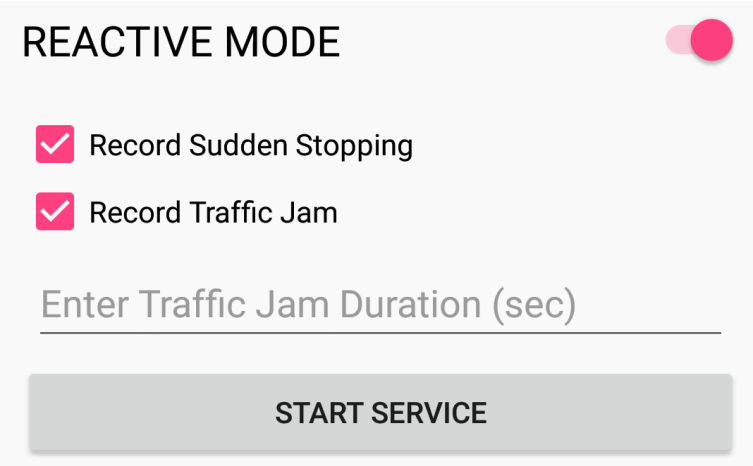
Slika 4.8. Pregled snimljenog sadržaja

Sistemski prozor za pregled vidljiv je i tokom rada ostalih aplikacija. Pretpostavljeno je da će korisnik tokom rada servisa koristiti mape ili neku drugu aplikaciju za navigaciju te je na ovaj način implementirano rješenje koje ne ometa rad ostalih aplikacija.

Zaustavljanje servisa moguće je preko notifikacije ili direktno iz aplikacije. Prilikom zaustavljanja isključuje se *Handler* i miče se notifikacija.

4.1.5. REAKTIVNI NAČIN RADA

Reaktivni način rada snima zapise uvjetovane nekim događajem. Mobilna aplikacija u ovom načinu rada može detektirati naglo kočenje ili potencijalni sudar te zastoj u prometu. Prilikom pokretanja servisa korisniku su ponuđene dvije opcije prikazane na slici 4.9.



Slika 4.9. Reaktivni način rada

Izvedba servisa slična je proaktivnom servisu dok je sam rad servisa definiran drugim tipom događaja. Na uređaju je vidljiva stalna notifikacija kao i kod proaktivnog servisa preko koje je moguće u svakom trenutku zaustaviti servis. Korisniku su ponuđene dvije opcije za snimanje tokom rada servisa:

- *RecordSuddenStopping* – snimanje u slučaju naglog kočenja ili sudara vozila
- *RecordTrafficJam* – snimanje zastoja u prometu.

Detekcija naglog kočenja implementirana je koristeći senzor ubrzanja mobilnog uređaja. Reaktivni servis pokreće servis za praćenje rada senzora ubrzanja koji prima podatke o promjenama na senzoru. Podaci sa senzora primaju se unutar metode *onSensorChanged* te je implementacija prikazana programskim kodom 4.6.

```
SensormySensor = sensorEvent.sensor;

if (mySensor.getType() == Sensor.TYPE_ACCELEROMETER) {
     float x = sensorEvent.values[0];
     float y = sensorEvent.values[1];
     float z = sensorEvent.values[2];
     doublegForce = Math.sqrt(x * x + y * y + z * z);
     if (gForce> MIN_FORCE && !takingPicture) {
           takingPicture = true;
           takePicture();
     }
}
```

Programski kod 4.6. Detekcija naglog kočenja

Senzor ubrzanja mjeri tri podatka o načinu kretanja uređaja. Mjeri se ubrzanje uređaja po sve tri osi (x,y,z) koji su predstavljeni u obliku realnog broja čija je mjerna jedinica metar po sekundi na kvadrat. Iz dostupnih podataka može se dobiti sila ubrzanja računanjem drugog korijena zbroja kvadrata vrijednosti svake osi. Postupak računanja sile ubrzanja vidljiv je na programskom kodu 4.6. Izračunata vrijednost sile ubrzanja uspoređuje se zadanom minimalnom silom potrebnom za sudar ili naglo kočenje i ako je veća kamera mobilnog uređaja snima trenutno stanje. Minimalna vrijednost sile ubrzanja postavljena je na 25 što je prosječna minimalna vrijednost sile ubrzanja tijekom sudara. Snimanje zapisa i pregled omogućeno je na identičan način kao i kod proaktivnog načina rada.

Detekcija zastoja u prometu implementirana je mjerenjem brzine dostupne putem lokacijskih usluga uređaja. Korisnik prilikom pokretanja servisa unosi dužinu zastoja u sekundama. Unošenjem podatka o dužini zastoja sprječava se detekcija lažnih zastoja kao što je čekanje na semaforu. Nakon pokretanja servisa uređaj će putem lokacijskih usluga određivati brzinu vozila te spremati vrijednosti za zadanu dužinu zastoja. Programski kod 4.7. prikazuje detekciju zastoja.

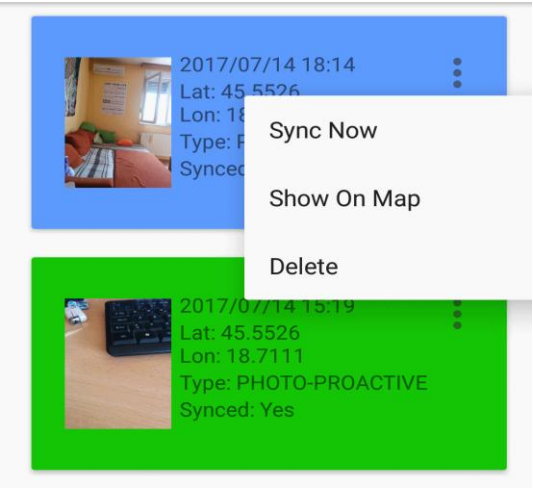
```
longnow = new Date().getTime();
floatapproxSpeed = 0;
Iterator<Location>iterator = listOfLocations.iterator();
while (iterator.hasNext()) {
      Locationloc = iterator.next();
      if (now - loc.getTime() >trafficJamDuration) {
            iterator.remove();continue;
      }
      approxSpeed+=loc.getSpeed();
}approxSpeed/= (float)listOfLocations.size();
if(approxSpeed< 5.556 &&startingTime - now< 0){
      reportTrafficJam();
}
```

Programski kod 4.7. Detekcija zastoja

Iz liste sa svim lokacijama i brzinama izbacuju se stari podaci. Stari podaci su sve vrijednosti čije je vrijeme uzimanjastarije od dužine zastoja. Nakon filtriranja računa se prosječna brzina u metrima po sekundi. Ukoliko je prosječna brzina manja od 5.556 metara po sekundi (20 km/h) snima se zastoj u prometu. Kako bi se izbjeglo inicijalno ubrzavanje vozila i uključivanje u promet servis ne uzima u obzir podatke prikupljene u prvih 40 sekundi od pokretanja. Podaci o brzini prikupljaju se svakih 5-7 sekundi.

4.1.6. SINKRONIZACIJA PODATAKA

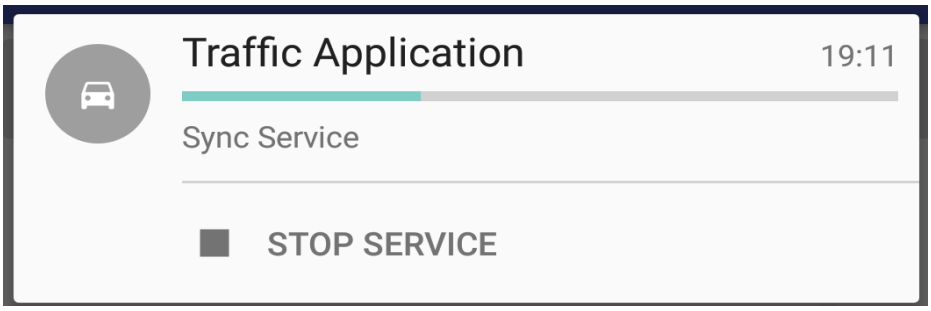
Sinkronizacija podataka unutar mobilne aplikacije ima dva načina rada. Prvi način rada je na zahtjev korisnika prikazan na slici 4.10. U ovom načinu rada korisnik samostalno odabire zapise koje želi podijeliti s ostalim sudionicima prometa.



Slika 4.10. Sinkronizacija podataka

Na zaslonu sa svim zapisima pored željenog zapisa odabire dodatno (tri točkice) te dodirom na *SyncNow* započinje slanje zapisa udaljenom poslužitelju. Drugi način rada je automatski u

kojem se zapisi šalju udaljenom poslužitelju bez dodatnih aktivnosti korisnika aplikacije. Ovaj način rada omogućen je od strane korisnika odabirom željenih postavki. Svi zapisi vidljivi su na zaslonu s listom zapisa, a broja pozadine je indikator uspješno sinkroniziranih zapisa. Zelena boja označava da je zapis uspješno sinkroniziran te dostupan putem interneta ostalim korisnicima. Plava boja indikator je da zapis još uvijek nije sinkroniziran. Korisniku je omogućeno brisanje zapisa s mobilnog uređaja odabirom *Delete* u padajućem izborniku. Odabirom *Show On Map* pokreće se zadana aplikacija za karte te je prikazana lokacija zapisa na mapi. Tijekom rada servisa za sinkronizaciju korisniku je prikazana stalna notifikacija s indikacijom progresa što je prikazano na slici 4.11.



Slika 4.11. Notifikacija servisa za sinkronizaciju

Servis za sinkronizaciju istog je tipa kao proaktivni i reaktivni servisi te je povezan s notifikacijom na isti način tokom rada kao i prethodno opisani servisi. Automatska sinkronizacija bit će pokrenuta ako zadane postavke to omogućuju i ako je aktivan proaktivni ili reaktivni servis. Provjera postoje li novi zapisi za sinkronizaciju obavljat će se svakih 30 sekundi od pokretanja servisa, prilikom podizanja aplikacije iz pozadine i od stane korisnika preko zaslona sa svim zapisima. Programski kod 4.8. prikazuje dohvaćanje zapisa koji još uvijek nisu sinkronizirani.

```
List<LocalFile>allLocalFiles = localFileDao.getAllUnsyncedLocalFiles();
if (!preferenceService.getSyncProactive()) {
     removeTypeFromList(allLocalFiles, RecordType.PROACTIVE);}
if (!preferenceService.getSyncReactive()) {
     removeTypeFromList(allLocalFiles, RecordType.REACTIVE);}
if (!preferenceService.getSyncUser()) {
     removeTypeFromList(allLocalFiles, RecordType.USER);}
allFiles = allLocalFiles.size(); filesForSync = newStack<>();
for (LocalFile file : allLocalFiles) {
     filesForSync.push(file);
}
```

Programski kod 4.8. Dohvaćanje nesinkroniziranih zapisa

Dohvaćaju se svi zapisi koji nisu sinkronizirani. Pomoću servisa za postavke dohvaćamo zadane postavke te ako nije uključena automatska sinkronizacija za određeni tip svi zapisi tog

tipa bit će obrisani iz liste zapisa za sinkronizaciju. Listu svi zapisa pretvaramo u stog kako bismo mogli koristiti funkcije za dodavanje novog elementa i uzimanje prvog elementa stoga. Stog je korišten zbog praktičnosti i FIFO svojstva. U svakom je trenutku moguće dodatni novi element koji će biti postavljen kao idući za sinkronizaciju dok funkcijom *pop()* uzimamo prvi element i brišemo ga sa stoga.

Implementacije sinkronizacije na zahtjev korisnika prikazana je programskim kodom 4.9. Nakon što je korisnik pokrenuo sinkronizaciju pojedinog zapisa šalje se zahtjev za pokretanjem servisa za sinkronizaciju. Uz zahtjev za pokretanje šalje se i univerzalni indikator zapisa (id). Ukoliko je servis pokrenut sustav ga neće ponovno pokretati, nego će se samo odraditi programski kod 4.9.

```
long localFileId = intent.getLongExtra(ARG_FILE_ID, -1);
if (localFileId != -1) {
     LocalFile localFile = localFileDao.getLocalFile(localFileId);
     if (!filesForSync.contains(localFile)) {
            filesForSync.push(localFile);
            allFiles++;
     }
}
```

Programski kod 4.9. Sinkronizacija pojedinog zapisa

Ukoliko uz zahtjev za pokretanje servisa postoji univerzalni indikator zapisa dohvaća se zapis te se dodaje na prvo mjesto prilikom sinkronizacije. Pokretanjem sinkronizacije pojedinog zapisa ujedno se pokreće i automatska sinkronizacija ako je ona omogućena. Potrebno je provjeriti nalazi li se zapis na stogu te ako se ne nalazi bit će dodan u protivnom neće jer ne može biti dva puta sinkroniziran.

```
if (filesForSync.size() > 0 && !autosync) {
     startSync();
} elseif (filesForSync.size() == 0 &&autosync) {
     startAutosync();
} elseif (filesForSync.size() > 0 &&autosync) {
     startSync();
     startAutosync();
} else {
     stopSelf();
     return START_NOT_STICKY;
}
```

Programski kod 4.10. Pokretanje sinkronizacije

Ovisno o postavkama i trenutnom stanju zapisa servis za sinkronizaciju će nastaviti s radom ili se zaustaviti što je vidljivo programskim kodom 4.10. Ukoliko postoje zapisi za sinkronizaciju i nisu ispunjeni zahtjevi za automatsku sinkronizaciju pokreće samo sinkronizacija, u slučaju kada ne postoje zapisi za sinkronizaciju, ali ispunjeni su uvjeti za automatskom pokreće se samo

automatska sinkronizacija. Ako postoje zapisi za sinkronizaciju i ispunjeni su uvjeti automatske pokreće se sinkronizacija i automatska sinkronizacija.

Sinkronizacija pojedinog zapisa odvija se u dva koraka. Kako bi zapis bio dostupan potrebno je obaviti oba koraka uspješno. Prvi korak je slanje datoteke zapisa udaljenom poslužitelju. Potrebno je odraditi *multipart POST request* koji sadržava zaglavlje s autorizacijom te unutar tijela *requesta* nalazi se putanja datoteke na poslužitelju, naziv datoteke i sadržaj datoteke. Taj *request* je najkritičniji za stabilan rad aplikacije jer datoteka koja se šalje poslužitelju prelazi nekoliko MB (megabajta) te je potrebna stabilna veza kako bi se uspješno odradio. Unutar *response* poslužitelja nalazi se URL do novo kreirane datoteke i dodatne informacije o datoteci. Nakon što je uspješno odrađen korak sa slanjem datoteke potrebno je poslati podatke o zapisu. U drugom koraku izvršit će se *POST request* za kreiranje novog dinamičkog resursa koji će sadržavati detalje o poslanoj datoteci. Udaljenom poslužitelju šalju se podaci o lokaciji, vremenu i tipu poslane datoteke.

4.2. POSLUŽITELJSKA APLIKACIJA

BaaS (engl. *backend as service*) je poslužiteljska aplikacija korištena u ovom radu. Pristup i upravljanje svim on-line resursima omogućen je putem API-a. Glavne značajke BaaS su:

- upravljanje korisnicima– registracija, prijava, kreiranje rola, postavljanje rola, brisanje korisnika
- integracija s društvenim mrežama – povezivanje korisnika s profilom za društvene mreže, dijeljenje sadržaja na društvenim mrežama
- upravljanje resursima – slanje, uređivanje i pristup datotekamana udaljenom poslužitelju
- notifikacije – sustav obavještavanja korisnika
- upravljanje podacima – CRUD funkcije nad različitim entitetima kreiranim od strane korisnika
- dopuštenja – upravljanje dopuštenjima za pristup različitih resursa API-a.

Korištenjem BaaS omogućena je podrška za razvoj neovisan o platformi. Pristup resursima je globalan te je moguć sa svake platforme na jednostavan način napraviti implementaciju i pregled zabilježenih zapisa.

Za BaaS korišten je Baasic, proizvod dostupan on-line http://www.baasic.com/ . Proizvod je besplatan uz limite, a plaćanjem je omogućena veća propusnost, više poziva, veći prostor za

pohranu. Konzumiranje REST API-a moguće je putem nekog od dostupnih SDK (engl. *software develpment kit*) ili direktno koristeći API. Dostupni SDK za Baasic su: JavaScript SDK, Angular SDK, Polymer SDK, .Net Framework. SDK omogućuje jednostavno korištenje API-a pomoću gotovih servisa. Baasic je podijeljen u module. Svaki modul je zasebna cjelina koja predstavlja nadogradnju postojećeg modula ili dodatnu funkcionalnost.

Moduli Baasic-a su:

- *Authentication & Authorization* – modul za autorizaciju i autentikaciju korisnika, prijava, registracija, integracija sa društvenim mrežama
- *User Management* – omogućuje upravljanje korisnicima od strane administrator korisnika, brisanje, dodavanje, postavljanje rola, dopuštenja
- *UserProfiles* – koristi se kao proširenje osnovnih podataka korisnika, mogućnost dodavanja profila s detaljnim informacijama o korisniku
- *Dynamic Resources* – dinamički resursi, omogućuje kreiranje proizvoljnih shema podataka te upravljanje tim podacima
- *Articles* – kreiranje članaka iz predložaka ili direktno putem HTML-a, dodavanja resursa člancima, korisno kod kreiranja blogova
- *Media Valut & Files* – upravljanje, razmjena i pristup datotekama, moguće se u i razne modifikacije na određenim tipovima datoteka
- *Notifications* – slanje notifikacija korisnicima putem jednostavnog sučelja
- *Templates* – predlošci raznih namjena kao što su aktivacijska poruka ili predložak koji će biti korišten za kreiranje članka
- *Metering* – modul za praćenje korištenja modula, aplikacije, dostupnog prostora, razna mjerenja
- *Keyvalues & Valuesets* – jednostavno sučelje za spremanje podataka u obliku ključ-vrijednost, veoma korisno za spremanje postavki i jednostavnih podataka općenito potrebnih u aplikaciji
- *Permissions* – upravljanje svim dopuštenjima za pristup na razini aplikacije, modula ili čak pojedinog resursa.

Važno je znati da Baasic podržava i koristi HAL+JSON format. Svi Baasic SDK-ovi će imati postavljen HAL+JSON HTTP format prilikom komunikacije s poslužiteljem. Time je omogućeno da svaki resurs vezan uz druge resurse može na jednostavan način dohvatiti vezane resurse bez potrebe za više upita prema poslužitelju.

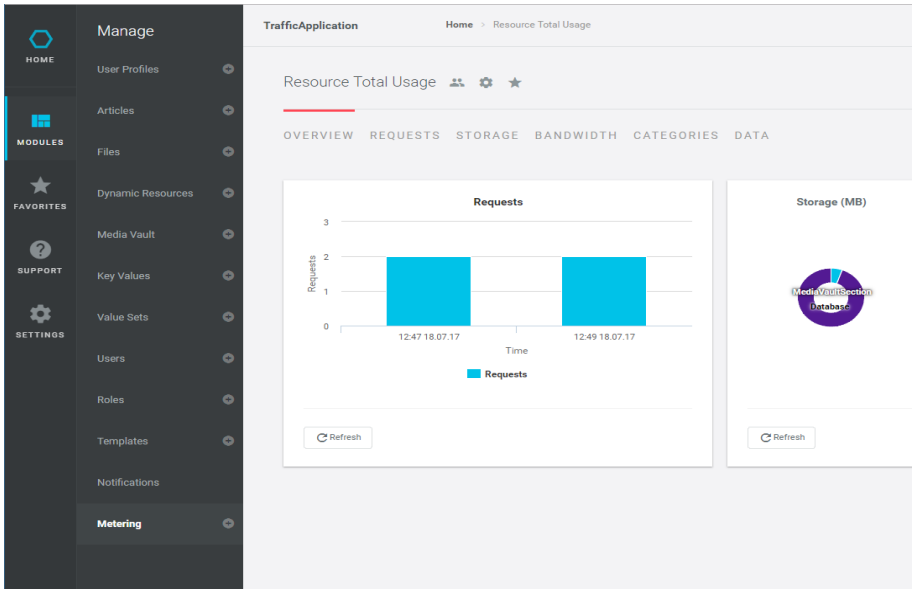
SQL upiti su mogući putem BQL (engl. *Baasic Query Language*) te se koristi putem *searchQuery* parametra prilikom upita prema poslužitelju kako bi se rezultati mogli dodatno filtrirati. Primjer BQL-a prikazan je programskim kodom 4.11.

```
WHERE Field = 'Value'
WHERE Field> 1
WHERE Field1 > 1 AND Field2 = 'Value'

SELECT Field, MAX(Field2) WHERE Field> 1

SELECT Field WHERE Field> 1 ORDER BY Field
```

Programski kod 4.11. Primjer BQL-a

4.2.1. POSLUŽITELJSKO SUČELJE

Poslužiteljsko sučelje Baasic-a za upravljanje i pregled resursa aplikacije omogućeno je putem nadzorne ploče (engl. *dashboard*). Nadzorna ploča je AngularJS aplikacija koja pruža administratoru sustava grafički pristup resursima. Na ovaj način omogućen je jednostavniji i korisnicima bliži prikaz resursa. Nadzorna ploča nije nužan dio sustava, nego samo pomoć za pristup i pregled. Sve mogućnosti nadzorne ploče implementirane su korištenjem API-a dostupnog administratoru sustava. Moguće je implementirati vlastitu nadzornu ploču kombiniranje postojećih modula i API-a. Grafičko sučelje nadzorne ploče prikazano je na slici 4.12.

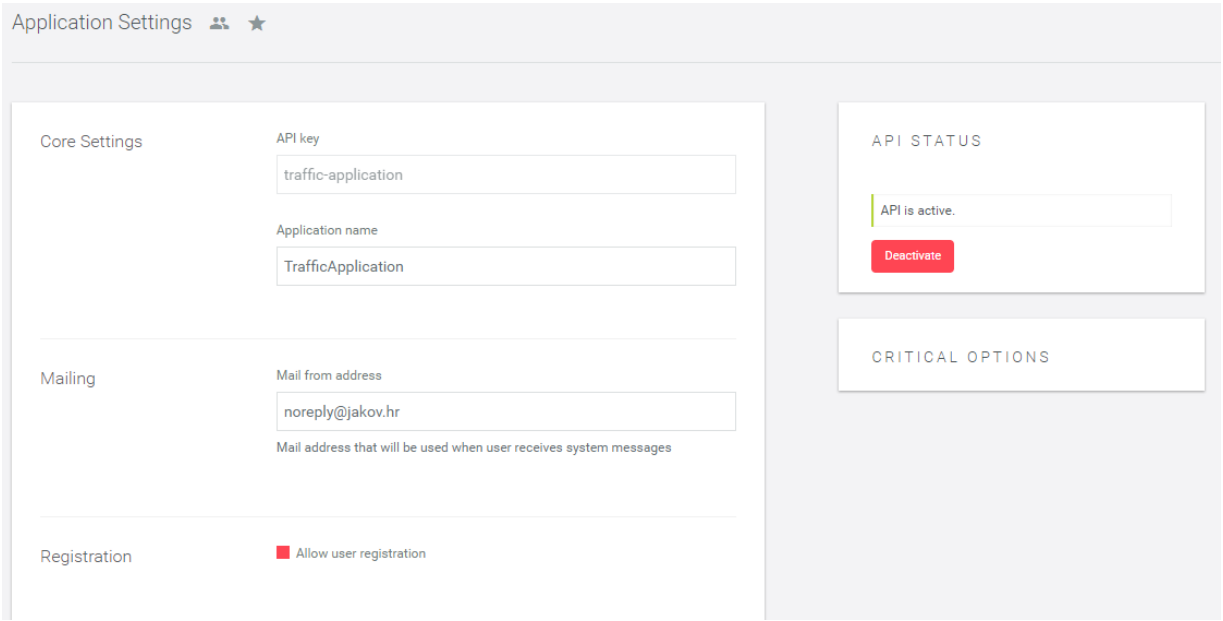


Slika 4.12. Baasic nadzorna ploča

Početna stranici nadzorne ploče prikazuje osnovnu statistiku korištenja API-a. Dodatne statistike moguće je vidjeti unutar modula *Metering*. Cijelo sučelje podijeljeno je na cjeline po

modulima. Za svaki modul prikazani su njegovi resursi u obliku različitih lista te dodatne postavke za pristup modulu ili pojedinom podatku. Pristup nadzornoj ploči omogućen je glavnom korisniku koji je kreirao aplikaciju. Glavnom korisniku aplikacije omogućeno je davanje pristupa nadzornoj ploči ostalim korisnicima.

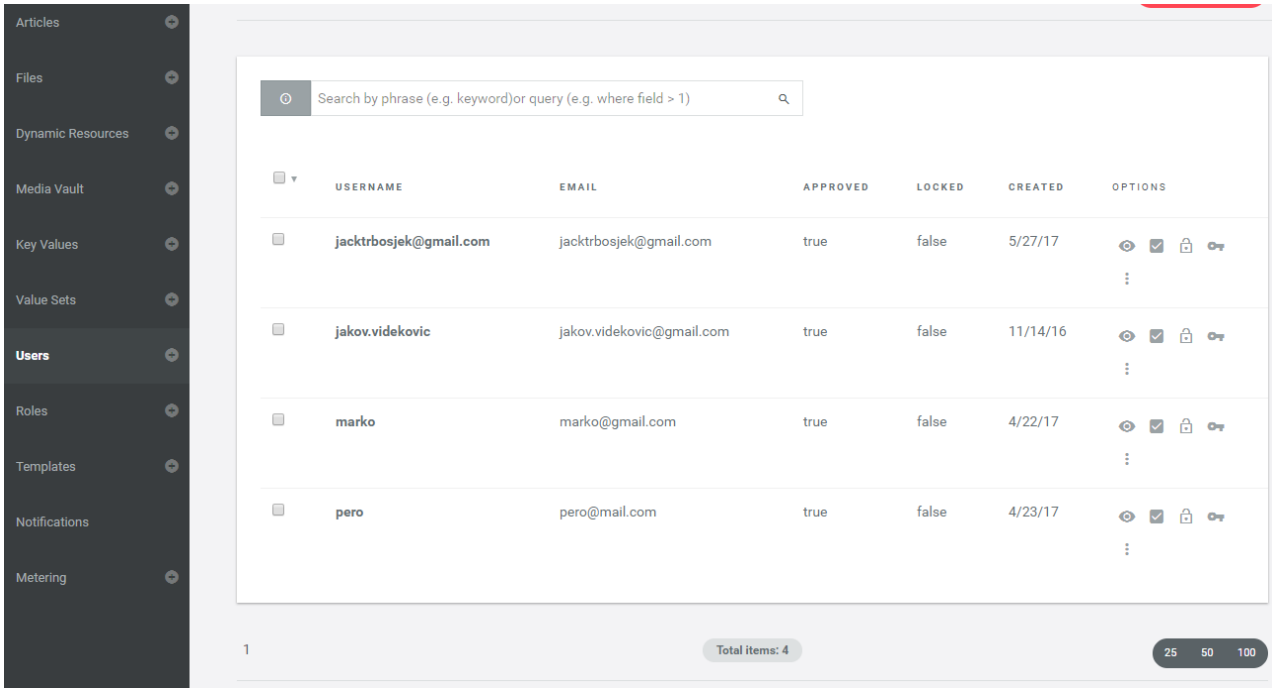
Postavkama aplikacije pristupa se odabirom *Settings* – *Application Settings* te su prikazane na slici 4.13.



Slika 4.13. Postavke aplikacije

API key aplikacije kreira se prilikom kreiranje nove aplikacije putem nadzorne ploče. Veoma je važan podatak jer pristup cijelom API-u aplikacije omogućen je preko ovog ključa. Svaki upit prema poslužitelju mora sadržavati ovaj ključ kako bi poslužitelj mogao povezati upit s određenom aplikacijom. Osim ključa za aplikaciju unutar postavki moguće je trenutno deaktivirati cijeli API, isključiti mogućnost registracije, urediti naziv aplikacije, dopustiti pristup samo određenoj domeni i ostale postavke vezane uz aplikaciju.

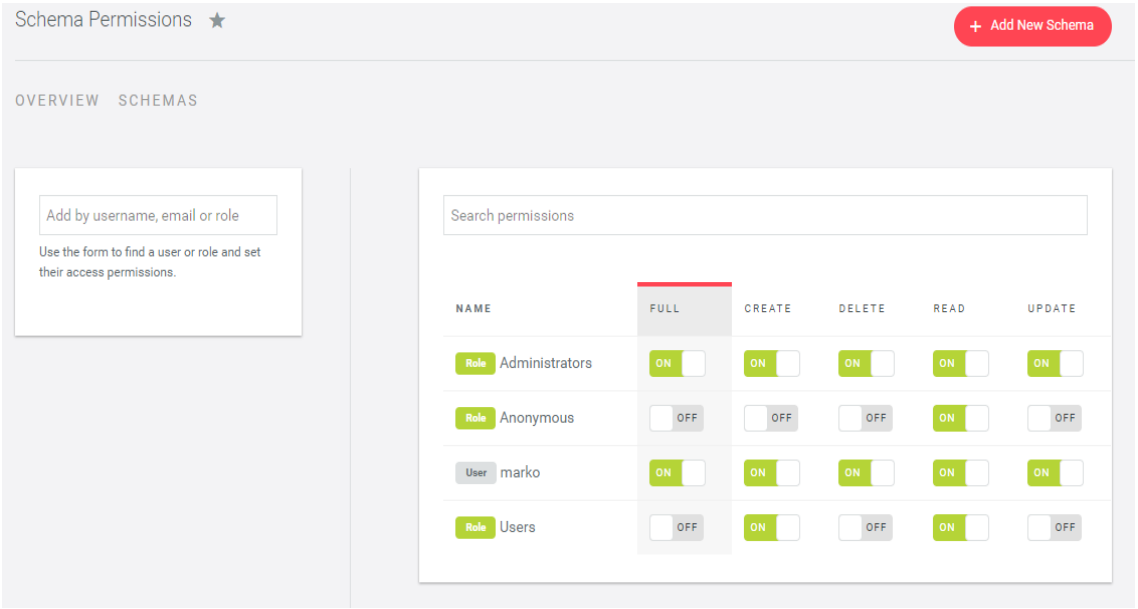
Korisnik se dodaje u sustav putem nadzorne ploče od strane drugih korisnika ili registracijom. Prilikom registracije da bi korisnički račun bio potpun potrebno je napraviti aktivaciju. Aktivacija korisnika moguća je putem linka koji je poslan na elektroničku poštu korisnika ili od strane korisnika s pristupom nadzornoj ploči. Svakom korisniku moguće je automatski dodijeliti rolu prilikom registracije, a u nadzornom ploči moguće ih je mijenjati. Slika 4.14. prikazuje sučelje nadzorne ploče za upravljanje korisnicima.



Slika 4.14. Upravljanje korisnicima putem nadzorne ploče

Omogućeno je dodavanje novih rola u sustav, a tri osnovne role su: *Administrator*, *User*, *Anonymous*. Svakom korisniku može se dodijeliti jedna ili više rola. U slučaju kada se koristi modul s profilom za svakog je korisnika moguće unijeti dodatne osobne informacije korisnika.

Pristup je implementiran pomoću sekcija i akcija. Svaki modul se nalazi u zasebnoj sekciji je na taj način pristup pojedinom modulu omogućen preko njegove sekcije. Unutar svake sekcije može se nalaziti pojedini korisnik ili rola. Kada se radi o korisniku onda je njemu dopušten pristup određenoj sekciji, ako se radi o roli onda je svim korisnicima s dodanom rolom dopušten pristup sekciji.Dodavanjem role ili korisnika u sekciju dopušten je pristup modulu dok je razina pristupa ograničena akcijama. Postoji pet akcija za pristupa i one su: *FULL*, *CREATE*, *READ*, *DELETE*, *UPDATE*. *Full* razina pristupa omogućuje potpuni pristup modulu s mogućnosti upravljanja pristupa te uređivanje postavki modula. Korisnici s *full* pristupom mogu uređivati pristup za druge korisnike. *Create* pristup dopušta dodavanje novih resursa. *Read* pristup dopušta samo pregled resursa. *Delete* pristup dopušta brisanje resursa. *Update* pristup dopušta uređivanje postojećih resursa. Na slici 4.15. prikazane su postavke za pristup dinamičkom resursu korištenom u ovom radu.



Slika 4.15. Pristup dinamičkom resursu *File*

Kao što je vidljivo iz slike 4.15. *full* pristup dinamičkom resursu *File* omogućen je svim korisnicima u roli administrator te posebno za korisnika marko. Bitno je za uočiti da registrirani korisnici sustava s rolom *Users* imaju dopuštenja za kreiranje novih i čitanje resursa. Na ovaj način registrirani korisnici mogu dodavati nove zapise. *Anonymous* imaju dopuštenja za čitanje. Ovom postavkom omogućeno je javno čitanje svih resursa ovog tipa što znači da nije potreban korisnik niti autorizacija da bi se dohvatio resurs. Na sličan način su postavljena dopuštenja za *Files* modul jer je potrebna ista razina autorizacije.

Dinamički resursi (engl. *dynamic resources*) su podaci proizvoljnog oblika i sadržaja koje se može spremiti unutar baze poslužitelja. Upravljanje resursima kao što su filtriranje, dohvaćanje, uređivanje, brisanje izvršava se preko API-a ili nadzorne ploče. Za potrebe rada kreiran je resurs *File*, a strukturu jednog resursa vidljiva je na programskom kodu 4.12.

```
{
  "id": "e0LDyxaPEQV3zkdbW4mzrP",
  "user_id": "ofpU609QUqU5ElKfoOpOn2",
  "file_url": "https://api.baasic.com/v1/traffic-application/file-
streams/hMyV2dZWk2LDMydbW4myL8",
  "is_image": true,
  "latitude": 45.5505727,
  "longitude": 18.7089022,
  "date_created": 1497796499546
}
```

Programski kod 4.12. Struktura *File* dinamičkog resursa

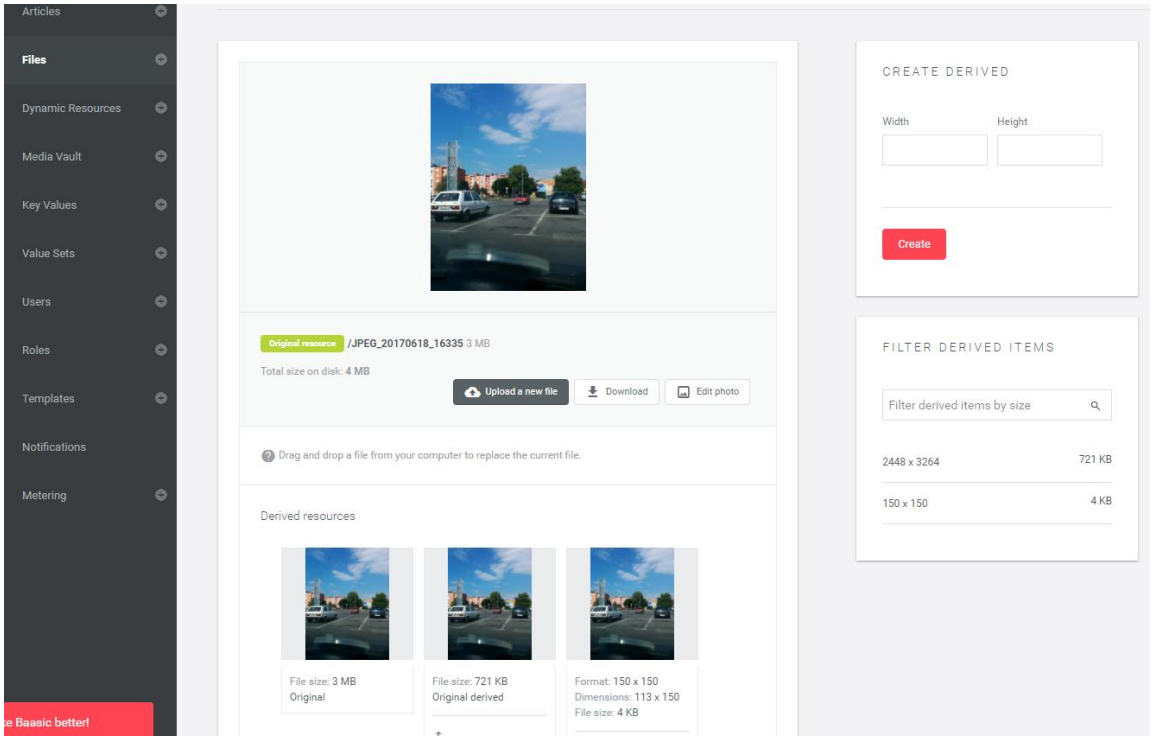
Resursi su spremljeni i prikazani u JSON obliku. Prilikom kreiranja novog resursa sustav mu dodjeljuje univerzalni indikator (id). Za kreiranje univerzalnog indikatora koristi se *Base64*

encode GUID vrijednosti univerzalnog identifikatora. Osim univerzalnog identifikatora dinamički resurs sadržava sljedeće podatke:

- *user_id* – univerzalni identifikator korisnika sustava koji je postavio zapis
- *file_url* – putanja za pristup zapisu na poslužitelju
- *is_image* – parametar koji služi kao indikator za određivanje radi li se o slici ili video zapisu
- *latitude* – vrijednost zemljopisne širine na kojoj je snimljen zapis
- *longitude* – vrijednost zemljopisne dužen na kojoj je snimljen zapis
- *date_created* – vrijeme kada je snimljen zapis u obliku broja koji predstavlja UNIX vrijeme u milisekundama.

API *endpoint* za prisut *File* dinamičkom resursu je ovog oblika: *https://api.baasic.com/v1/traffic-application/resources/File* . POST metodom na taj *endpoint* dodaje se novi resurs gdje se u tijelu upita prosljeđuju vrijednosti. Upit GET metodom na ovaj *endpoint* uz potrebne parametre vraća listu svih resursa za definirani filter. Dodavanjem univerzalnog indikatora na kraj *endpoint* odvojenog sa / omogućen je pristup pojedinom resursu. Metodom GET dohvaćamo resurs, a metodom PUT uređujemo postojeći resurs. Na ovaj način pristupalo bi se bilo kojem dinamičkom resursu.

Osim dinamičkih resursa za izradu ovog rada korišten je *Files* modul. Baasic pruža pristup svim datotekama potrebnim za rad preko *Files* modula. Nadzorna ploča omogućuje pregled dostupnih datoteka u sustavu, a grafičko sučelje za pregled pojedine datoteke vidljiv je na slici 4.16.



Slika 4.16. Pregled slike u nadzornoj ploči

Određene tipove datoteka kao što su slike moguće je uređivati na način da se iz originala slike stvori verzija potrebne širine i visine. Primjer je vidljiv na slici 4.14. gdje je uz originalnu verziju slike veličine 3MB stvorena manja verzija od svega 4KB za prikaz na mapi. Na ovaj način je ubrzan rad aplikacije jer prilikom dohvaćanja slika nije potrebno učitavati originalnu verziju, nego izvedenu verziju manje veličine. Iste modifikacije nisu moguće nad video zapisima pa se oni uvijek učitavaju u originalnoj veličini.

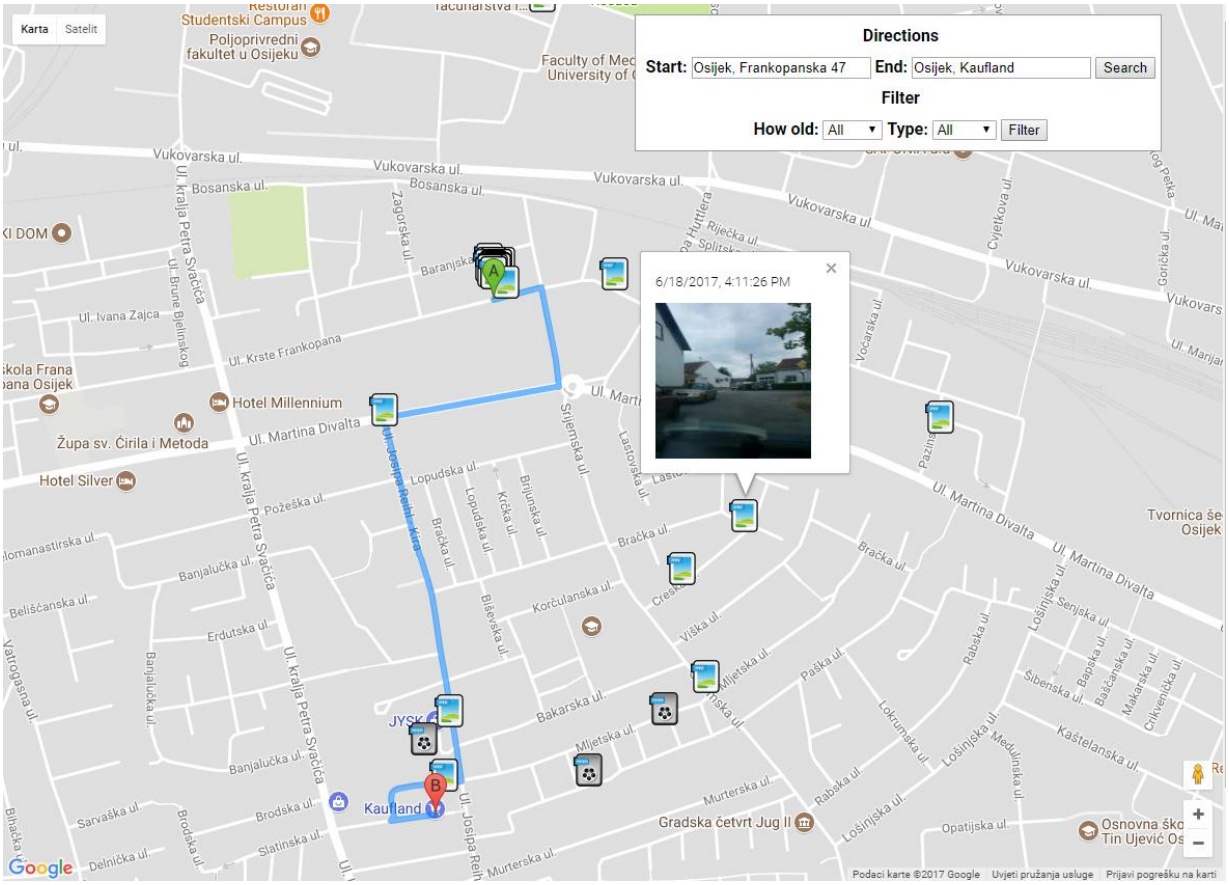
Pristup datotekama izvršava se preko API *endpointa*: *https://api.baasic.com/v1/traffic-application/files/* . GET metoda vraća filtriranu listu svih datoteka korisnika. Filtriranje se vrši pomoću parametara upita. U ovom radu nije korišten pristup listi svih datoteka, ali je pojedinoj datoteci. Pristup pojedinoj datoteci omogućen je putem *https://api.baasic.com/v1/traffic-application/file-streams/* . Nakon slanja datoteke korisnikov mobilni uređaj kreira dinamički resurs s direktnom lokacijom datoteke na poslužitelju.

Postavke za pristup posebno su uređene za potrebe rada. Korisnici u roli *Administrator* imaju puni pristup svim datotekama. Korisnici u roli *Users* imaju dopuštenja za čitanje svih datoteka i dodavanje novih. Prilikom registracije korisniku se postavlja rola *Users*. Neregistrirani korisnici imaju dopuštenja za čitanje, ali ne i za dodavanje novih datoteka. Na ovaj način javno je moguć pristup svim datotekama, ali samo za čitanje dok registrirani korisnici mogu dodavati nove datoteke.

4.2.2. **KLIJENTSKA APLIKACIJA**

Klijentska aplikacija je kombinacija HTMLa, CSSa, Javascripta. Za pokretanje aplikacije nije potreban poslužitelj jer se radi o jednostavnoj aplikaciji koju pokreće korisnikov preglednik. Kao pozadinski poslužitelj koristi se Baasic opisan u prethodnom poglavlju.

Sučelje aplikacije sastoji se od mape na kojoj su prikazani markeri s ikonama koje označavaju sliku ili video zapis. Klikom na marker otvara se prozor s pregledom odabranog zapisa i vremenom kada je zapis kreiran. Osim mape i markera u gornjem desnom kutu nalazi se prozor unutar kojeg se nalazi filtriranje i upute. Slika 4.17. prikazuje sučelje aplikacije.



Slika 4.17. Klijentska aplikacija

Unutar prozora s uputama i filtriranjem moguće je unijeti polazište i odredište te klikom na *Search* plavom crtom bit će iscrtane upute za vožnju. Na ovaj način korisnik može lakše odrediti njemu bitne zapise. Filtriranje zapisa je moguće po starosti zapisa i po tipu zapisa. Prilikom filtriranja po starosti zapisa može se odabrati: 2 sata, 12 sati, 24 sata i 48 sati. Kod filtriranja po tipu moguće je odabrati slike ili video zapise. Da bi se primijenili odabrani parametri potrebno je kliknuti na *Filter* nakon čega se brišu postojeći markeri te se kreiraju novi za odabrani filter.

Za izradu aplikacije korišteni su dodatne Javascript biblioteke, a njihovo uključivanje u rad aplikacije prikazano je programskim kodom 4.13.

```
<scriptsrc='https://code.jquery.com/jquery-3.1.1.min.js'></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/underscore.js/1.8.3/underscore-
min.js"></script>
<script src='https://cdnjs.cloudflare.com/ajax/libs/reflect-
metadata/0.1.10/Reflect.min.js'></script>
<scriptsrc='scripts/baasic-sdk-javascript.js'></script>
<scriptsrc='scripts/app.js'></script>
<scriptasyncdefer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyB0hoL7184TMovuIeIYy8ujjVA4a1
WSqzA&callback=initMap">
```

Programski kod 4.13. Javascript biblioteke

Od dodatnih biblioteka korištene su:

- *JQuery* – potrebna za rad ostalih biblioteka, dohvaćanje elemenata HTML-a i njihovih vrijednosti
- *underscoreJS* – za jednostavno upravljanje listama i ostalim osnovnim funkcijama
- *ReflectJS* – potrebna za BaasicJavascript SDK-a
- *BaasicJavascript SDK* – za komunikaciju s udaljenim poslužiteljem putem gotovih servisa
- *app*.js – interna skripta aplikacije, u njoj se nalazi većina Javascript koda vezanog za aplikaciju
- *Google Map JS* – skripta za prikaz mape i uputa, za kreiranje mape potreban je ključ i unutar *Google Console* uključiti *Map API* i *Map Directions API.*

Prilikom pokretanja aplikacije potrebno je inicijalizirati postavke za mapu i upute te postavke za Baasic Javascript SDK što je prikazano programskim kodom 4.14.

```
functioninitMap() {
     var markerArray = [];
     map = newgoogle.maps.Map(document.getElementById('map'), {
           zoom: 15
     });
     var directionsService = newgoogle.maps.DirectionsService;
}
var options = {
apiRootUrl: 'api.baasic.com',
apiVersion: 'v1',
enableHALJSON: false
};
var application = newbaasicSdkJavaScript.BaasicApp('traffic-application',options);
```

Programski kod 4.14. Inicijalizacija mapa i SDK-a

Nakon učitavanja *Google Map* skripte poziva se funkcija *initMap* koja kreira objekt mape i postavlja uvećanje mape, nakon čega se kreira objekt *DirectionsService* korišten za dohvaćanje i

iscrtavanje uputa. Kod postavljanja Baasic Javascript SDK-a potrebno je kreirati *options* objekt sa osnovnim postavkama za rad te potom *application* objekt s proslijeđenim *API key* i *options*. Ovim postupkom aplikacija je postavljena za rad te je omogućeno korištenje mape i Baasic Javascript SDK-a.

Programskim kodom 4.15. prikazano je dohvaćanje dinamičkih resursa korištenjem BaasicJavascript SDK-a.

```
functionloadResources() {
	var searchObject = {
		pageNumber: 1,
		pageSize: 100,
		orderBy: 'dateCreated',
		orderDirection: 'asc'
	}
	if(searchQuery){
		searchObject.searchQuery= searchQuery;
	}
	application.dynamicResourceModule.find('File', searchObject)
		.then(function (collection) {
			console.log(collection);
			setupMarkers(collection.data);
		},
		function (response, status, headers, config) {
			console.log(response);
		});
}
```

Programski kod 4.15. Dohvaćanje dinamičkih resursa

Kreira se objekt s parametrima za pretraživanje te ako postoje parametri filtera dodaju se na objekt za pretragu. Pomoću *application* objekta pristupa se servisu za pretragu dinamičkih resursa. Poziva se funkcija *find* kojoj je proslijeđen naziv dinamičkog resursa i objekt s parametrima za pretragu. Funkcija vraća *promise* koji prilikom uspješnog *resolvea* prosljeđuje kolekciju za odabrani filter.

Kreiranje BQL upita pretragu dinamičkih resursa prikazan je programskim kodom 4.16.

```
searchQuery = "";
var range = document.getElementById('range').value;
if(range){
	var now = new Date().getTime();
	searchQuery = "WHERE date_created>'"+(now - range*1000*60*60)+"'";
}var type = document.getElementById('type').value;
if(type){
	if(searchQuery){
		searchQuery+=" AND ";
	}else{
		searchQuery = "WHERE ";
	}
	searchQuery+="is_image='"+type+"'";}
```

Programski kod 4.16. Kreiranje BQL upita

Postavlja se početni prazni tekst za upit zatim se dohvaća vrijednost odabrana za filter. Ukoliko vrijednost postoji na tekst za BQL upit postavlja se *WHERE* upit za parametar

date_created koji mora biti veći od trenutnog vremena umanjenog za odabrani interval u satima. Nakon provjere postoji li odabrani parametar za tip zapisa postavlja se upit za *is_image* parametar. Ovako će izgledati BQL upit za filtriranje svih slika unutar zadnjih 12 sati: *WHERE date_created>'1500430461320' AND is_image='true'* . Parametar korišten za *date_created* ovisan je o trenutnom vremenu.

Dohvaćanje i prikaz odabrane rute koristeći polaznu i odredišnu točku prikazan je programskim kodom 4.17.

```
for (var i = 0; i <markerArray.length; i++) {
     markerArray[i].setMap(null);
}
directionsService.route({
     origin: document.getElementById('start').value,
     destination: document.getElementById('end').value,
     travelMode: 'DRIVING'
}, function (response, status) {
     if (status === 'OK') {
           document.getElementById('warnings-panel').innerHTML =
                 '<b>' + response.routes[0].warnings + '</b>';
           directionsDisplay.setDirections(response);
     } else {
           window.alert('Directionsrequestfaileddue to ' + status);
     }
});
```

Programski kod 4.17. Dohvaćanje i prikaz rute

Ukoliko postoje markeri za rutu postavljaju se na *null* vrijednost. Korištenjem *directionService* funkcije *route* s proslijeđenim parametrima za početnu i odredišnu točku te način putovanja dohvaćaju se upute za vožnju. Nakon uspješnog dohvaćanja preko *directionDisplay* objekta upute se prikazuju na mapi. Ukoliko početna ili odredišna točka ne postoje ili se dogodio neki drugi tip greške poruka se prikazuje korisniku unutar *alert* prozora.

5. ZAKLJUČAK

Uvođenje i primjena ITS-a donosi unaprjeđenje prometnica i prometa općenito. Razvojem transportnih sustava povećava se sigurnost prometnica, učinkovitost prometovanja i zadovoljstvo korisnika prometnica. Osim poboljšanja fizičkih funkcija prometnica veliki naglasak je i na većoj informiranosti vozača. Napredak ITS-a direktno je vezan uz razvoj tehnologije. Niska cijena, veća dostupnost i poboljšana kvaliteta uređaja poput senzora i raznih vrsta kamera otvara nove mogućnosti za razvoj i inovacije. Veliki korak u napretku ITS-a čine bežične ad-hoc mreže koje omogućuju povezivanje vozila u VANET. Vozila u ad-hoc mrežama međusobno komuniciraju i razmjenjuju korisne prometne informacije. Svako vozilo predstavlja čvor u mreži putem kojeg se razmjenjuju podaci. Postoje dvije vrste komunikacije u ovakvim mrežama. V2V komunikacija temelji se na razmjeni informacija među vozilima dok V2I komunikacija ostvaruje razmjenu informacija s mrežnom infrastrukturom koja se nalazi uz prometnicu. Ovakva struktura mreže neovisna je o infrastrukturi i lokaciji jer mrežu čine vozila. iMANET je tip mobilne bežične ad-hoc mreže unutar koje se prijenos podataka vrši putem interneta.

ITS koji se sastoji od mobilne aplikacije i web sučelja za pregled zapisa daje dobar pregled trenutnih mogućnosti korištenja tehnologije u cestovnom prometu. Mobilna aplikacija koja ima mogućnost snimanja prometnice u tri načina rada prikuplja zapise koji su u stvarnom vremenu dostupni ostalim sudionicima prometa. Korištenje aplikacije je jednostavno i nakon početnog postavljanja te pokretanja ne zahtjeva dodatnu interakciju s korisnikom. Vozaču je tijekom vožnje osigurano neometano upravljanje vozilom. Rad aplikacije odvija se u pozadini tako da je moguće koristiti ostale aplikacije za navigaciju. Otvorene su mogućnosti nadogradnje aplikacije uvođenjem novih načina rada te napredne mogućnosti prijenosa i sinkronizacije podataka. Web aplikacija nudi jednostavan i brz pregled dostupnih zapisa na mapi. Omogućeno je filtriranje zapisa po vremenu i tipu te odabir rute unošenjem početne i odredišne točke putovanja. Korisnici na jednostavan i brz način mogu pregledati informacije bitne za njihovu rutu putovanja. Moguća je nadogradnja proširenjem funkcionalnosti prikaza rute i odabira filtera te uvođenjem dodatnih informacija o prometnici koje bi bile prikazane na mapi.

LITERATURA

[1] Sangwon Lee, DukheeYoon, AmitabhaGhosh, Intelligent Parking Lot Application Using Wireless Sensor Networks, članak

[2] World Congress on Intelligent Transport Systems, članak, dostupno na: [24.06.2017.]

[3] Nacionalni program za razvoj i uvođenje inteligentnih transportnih sustava u cestovnom prometu za razdoblje od 2014. do 2018. godine , članak, dostupno na: http://narodne-novine.nn.hr/clanci/sluzbeni/2014_07_82_1580.html [24.06.2017.]

[4] Vehicle Information and Communication System, dostupno na: https://en.wikipedia.org/wiki/Vehicle_Information_and_Communication_System

[5] YiWang, IEEE 802.11p Performance Evaluationand Protocol Enhancement, članak

[6] Lubomir Dobos,Tibor Kolos, Intelligent Transport Systems and Vehicular Ad hoc Networks, dostupno na: http://sociallab.fer.hr/innosoc/case-studies/zagreb-2016/intelligent-transport-systems-and-vehicular-ad-hoc-networks/ [24.06.2017.]

[7] Congestion based mechanism for route discovery in a V2I-V2V system applying smart devices and IoT, https://openi.nlm.nih.gov/detailedresult.php?img=PMC4431278_sensors-15-07768-g002&req=4 [21.06.2017.]

[8] JoshBroch David A. Maltz David B. Johnson Yih-Chun Hu Jorjeta Jetcheva, A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, članak, dostupno na: https://cseweb.ucsd.edu/classes/wi01/cse222/papers/broch-adhoc-mobicom98.pdf [24.06.2017.]

[9] Saleh Ali Alomariand Putra Sumari, Multimedia Applications for MANETs over Homogeneous and Heterogeneous Mobile Devices, članak

[10] D. Estrinetal, “New CenturyChallenges: Scalable Coordinationin Sensor Networks”, ACM Mobicom, 1999.

[11] Jesna Jamal, MANET (Mobile ad hoc network)- Characteristics and Features, članak, http://www.eexploria.com/manet-mobile-ad-hoc-network-characteristics-and-features/

[12] Routing and Broadcastingin Hybrid Ad Hoc Networks, članak, dostupno na: https://hal.inria.fr/inria-00069889

[13] M. Ángeles Serna, Rafael Casado, Aurelio Bermúdez, Nuno Pereira, Stefano Tennina, Distributed Fores tFire Monitoring Using Wireless Sensor Networks, članak, dostupno na: http://journals.sagepub.com/doi/full/10.1155/2015/964564

[14] Wikipedia, Smartphone, izvor: https://en.wikipedia.org/wiki/Smartphone#History [22.06.2017.]

[15] Smartphone OS Market Share, 2016 Q3, članak, http://www.idc.com/promo/smartphone-market-share/os [24.06.2017.]

[16] Bilten o sigurnosti cestovnog prometa 2015., https://www.mup.hr/UserDocsImages/Publikacije/2016/bilten_promet_2015_2.pdf [24.06.2017.]

[17] Hamid M. Ali, Zainab S. Alwan, Car Accident Detection and Notification System Using Smartphone, http://ijcsmc.com/docs/papers/April2015/V4I4201599a40.pdf [24.06.2017.]

[18] Android (operating system), wikipedia, članak https://en.wikipedia.org/wiki/Android_(operating_system) [24.06.2017.]

[19] Android (robot), wikipedia, https://en.wikipedia.org/wiki/Android_(robot) [24.06.2017.]

[20] Android logo, slika, http://www.freepik.com/free-vector/android-boot-logo_683826.htm [24.06.2017.]

[21] Android Nougat 7.0, https://www.android.com/versions/nougat-7-0/

[22] Platform architecture, https://developer.android.com/guide/platform/index.html [24.06.2017.]

[23] SensorsOverview, https://developer.android.com/guide/topics/sensors/sensors_overview.html [lipanj 2017.]

[24] Magnetometar, wikipedia, https://hr.wikipedia.org/wiki/Magnetometar [24.06.2017.]

[25] IntelliJ IDEA, https://www.jetbrains.com/idea/ [lipanj 2017.]

[26] Android Studio, https://developer.android.com/studio/intro/index.html [lipanj 2017.]

[27] Android Studio Gradle build system, https://developer.android.com/studio/releases/gradle-plugin.html [lipanj 2017.]

[28] Groovy syntax, http://groovy-lang.org/syntax.html

[29] Mobile backend as a service, https://en.wikipedia.org/wiki/Mobile_backend_as_a_service [srpanj, 2017.]

[30] XML , https://en.wikipedia.org/wiki/XML [srpanj, 2017.]

[31] Retrofit, http://square.github.io/retrofit/[srpanj, 2017.]

[32] SharedPreferences, https://developer.android.com/reference/android/content/SharedPreferences.html [srpanj, 2017.]

[33] OAuth 2.0 Autentication, https://tools.ietf.org/html/rfc6749#section-4.3 [srpanj, 2017.]

[34] Android Service, https://developer.android.com/reference/android/app/Service.html [srpanj, 2017.]

SAŽETAK

Diplomski rad istražuje razvoj ITS-a te mogućnosti bežičnih ad-hoc mreža. Opisane su glavne karakteristike i smjernice razvoja cestovnog prometa. Naglasak je na sigurnosti i boljoj informiranosti vozača korištenjem trenutno dostupnih tehnologija. Svakodnevna upotreba i velik broj pametnih mobilnih uređaja opremljenih mnoštvom senzora mogu se iskoristiti za praćenje i prenošenje prometnih informacija. Velika pokrivenost i dostupnost mobile podatkovne mreže omogućuje dijeljenje informacija s ostalim sudionicima prometa. U diplomskom radu razvijena je Android mobilna aplikacija za praćenje prometa i web aplikacija za pregled dostupnih informacija. Mobilna aplikacija snima prometnicu u tri načina rada: proaktivni, reaktivni i korisnički. Snimaju se slike i video zapisi s lokacijom i vremenom. Zapisi se dijele s ostalim korisnicima putem interneta. Web aplikacija omogućuje pregled svih zapisa na karti. Omogućeno je filtriranje zapisa i pregled rute putovanja. Razvijeni sustav nudi korisnicima otvoreni pristup svim zapisima te omogućuje dijeljenje podataka s ostalim korisnicima.

Ključne riječi: pametni mobilni uređaji, bežična ad-hoc mreža, inteligentni transportni sustavi, mobilna aplikacija, web aplikacija

ABSTRACT

Intelligent transportation system for sharing useful real-time traffic information through smartphones

Master thesis explores growth of ITS and achievements of wireless ad-hoc networks. Main characteristics and guidelines for development of road trafficking are described. Emphasis is on security and better driver informing using available technologies. Everyday use and great number of smartphones equipped with numerous sensors can be used for tracking and sharing traffic information. Good coverage and availability of mobile data network enables sharing information among traffickers. Master thesis includes development of Android mobile application for traffic recording and web application for record display. Mobile application takes records of road ahead in three modes: proactive, reactive and user. Picture and video files along with location and current time are recorded. Records are shared with other users via Internet. Web application enables overview of all records on map. Advanced filtering and user route are implemented. Developed system offers open access to all records and enables sharing information with other users.

Keywords: Smartphones, wireless ad-hoc network, intelligent transportation systems, mobile application, web application

ŽIVOTOPIS

Jakov Videković, rođen 08.03.1989. u Vukovaru. Do završetka prvog razreda osnovne škole živi u Rijeci te se potom seli u Osijek. Osnovnoškolsko obrazovanje završava u Osnovnoj školi Ljudevit Gaj Osijek. Nakon osnovne škole, godine 2004. upisuje 3. gimnaziju u Osijeku. Godine 2008. upisuje Elektrotehnički fakultet u Osijeku smjer Sveučilišni preddiplomski studij računarstava. Godine 2013. stječe status prvostupnog inženjera Računarstva te iste godine nastavlja studij upisom Sveučilišnog diplomskog studija Računarstva, smjer Procesno Računarstvo.

Jakov Videković