Mogućnosti primjene geolokacijskih servisa pri razvoju mobilnih aplikacija

Dujaković, Stanko

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:499337

Rights / Prava: Attribution-NonCommercial 3.0 Unported/Imenovanje-Nekomercijalno 3.0

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Stanko Dujaković

MOGUĆNOSTI PRIMJENE GEOLOKACIJSKIH SERVISA PRI RAZVOJU MOBILNIH APLIKACIJA

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Stanko Dujaković

Matični broj: 46915/17–R

Studij: Informacijski sustavi

MOGUĆNOSTI PRIMJENE GEOLOKACIJSKIH SERVISA PRI RAZVOJU MOBILNIH APLIKACIJA

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Zlatko Stapić

Varaždin, rujan 2020.

Stanko Dujaković

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Ovim radom opisana je mogućnost primjene geolokacijskih servisa pri razvoju mobilnih aplikacija. Rad objedinjuje teorijsku mogućnost i svrhu primjene geolokacijskih servisa trećih strana, te opis praktične primjene jednog od opisanih servisa. U tu svrhu, nabrojane su i opisane tehnike pozicioniranja, te prikazana implementacija jednog geolokacijskog servisa. Mobilna aplikacija je razvijena u programskom jeziku Kotlin.

Ključne riječi: razvoj; razvoj mobilnih aplikacija; kotlin; geolokacijski servisi; mape;

Sadržaj

Sadržaj iii
1. Uvod 1
2. Metode i tehnike rada 2
3. Geolokacija 3
3.1. Tehnike pozicioniranja 4
3.1.1. Tehnike temeljene na mobilnim uređajima 4
3.1.2. Tehnike temeljene na mrežnim ćelijama 7
3.1.3. Tehnike hibridnog pristupa 9
4. Primjena geolokacijskih servisa 11
4.1. Kategorizacija primjene 11
4.1.1. Lokacijski temeljene društvene mreže 12
4.1.2. Lokacijski temeljene igre 14
4.1.3. Lokacijski temeljeno praćenje fitnesa i zdravlja 18
4.1.4. Transportni LBS 19
4.1.5. Lokacijski temeljene pomoćne tehnologije 21
4.2. Privatnost u lokacijskim uslugama 22
4.3. Budućnost lokacijskih usluga 23
5. Usporedba i tehnički opis geolokacijskih servisa 24
5.1. Usporedba servisa 24
5.2. Google Maps i OpenStreetMaps 26
5.3. Uporaba Google Maps servisa 27
5.4. Tehnički opis Google lokacijskog servisa 31
5.4.1. Sučelje LocationListener 32
5.4.2. Klase servisa 33
6. Implementacija aplikacije Moja Vozila 38
6.1. Zaslon učitavanja 41
6.2. Zaslon prijave 44
6.3. Zaslon karte 49
7. Zaključak 54
Popis literature 55
Popis slika 59
Popis tablica 60
Popis isječaka kôda 61
Prilozi 62

1. Uvod

Rasprostranjenost i razvoj mobilne tehnologije u posljednjem je desetljeću napredovao drastično. Mobilna tehnologija već godinama predstavlja sastavni dio svakodnevnog života čovjeka. Samim time možemo reći da je doživjela vrhunac rasprostranjenosti, a ono što nam slijedi u razvoju, predstavlja samo *profinjavanje* već viđene tehnologije. U grubo, tehnologija nam je omogućila dostupnost informacija i usluga neovisno o vremenu i u skladu s lokacijom.

Neovisnost o vremenu proizlazi iz same činjenice da mobilni uređaj možemo u bilo kojem trenutku imati uza se spreman na korištenje. Usklađenost informacija i usluga ovisno o lokaciji, temelji se na raznim eksternim tehnologijama koje nam omogućuju pozicioniranje uređaja u prostoru.

Upravo pozicioniranje uređaja i njegovo djelovanje u skladu s lokacijom predstavlja vodilju i crvenu nit ovog rada. Odrediti mogućnost i svrhu korištenja geolokacijskih servisa trećih strana pri razvoju mobilnih aplikacija inicijalni je cilj rada, te u tu svrhu objedinjuje teoretsku obradu tehnologija pozicioniranja, područja korištenja, ali i praktičnu implementaciju aplikacije za android platformu u programskom jeziku Kotlin. Sve navedeno podijeljeno je kroz četiri glavna poglavlja.

Drugo poglavlje posvećeno je tehnikama pozicioniranja mobilnih uređaja. Sastoji se od samog opisa pojma geolokacije, pobroja glavnih tehnika pozicioniranja i opisa pojedine tehnike. U tome poglavlju bit će dani najjednostavniji primjeri i pojmovi prilikom određivanja lokacije, ne samo mobilnih uređaja već i općenito. Treće poglavlje usmjereno je na samu svrhu primjene i područja korištenja geolokacijskih tehnologija. Tako su u tom poglavlju sistematizirana i opisana područja primjene. Kako se kod lokacije radi o osobnom podatku korisnika, neizbježno je opisat i očuvanje privatnost u lokacijskim uslugama. Četvrto poglavlje sadržajno je usmjereno na razvoj, te su u njemu dane usporedbe i tehnički opisi lokacijskih servisa. Zadnje poglavlje nastoji kroz implementaciju predočit praktičnu primjenu geolokacijskih servisa, te samim time sadrži implementaciju uz tekstualni opis rješenja.

2. Metode i tehnike rada

Korištene metode za ostvarenje rada su različite s obzirom na poglavlje. Za ostvarenje trećeg i četvrtog poglavlja, koji su po prirodi opisni, koristi se deskriptivna metoda. Peto poglavlje većinski je ostvareno komparativnom metodom, iako se i u njemu pronalaze opisni elementi. Praktični dio, prikazan u šestom poglavlju, uz priloženi programski kôd, izrađen je deskriptivnom metodom.

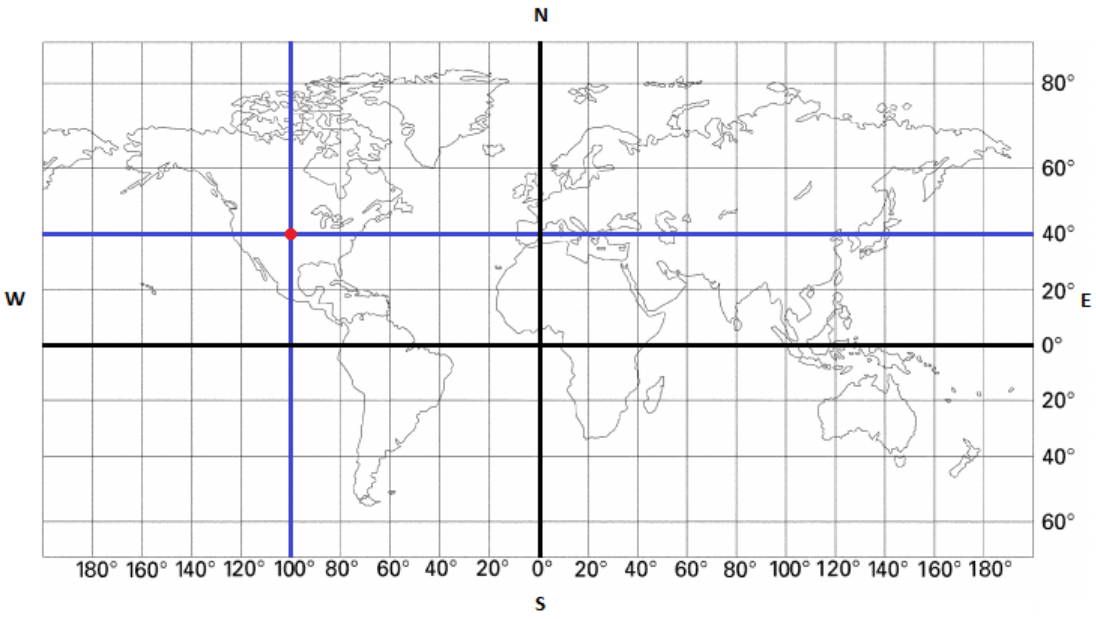
Kako je temelj rada geolokacija, početak izrade temeljen je na istraživanju i proučavanju navedenog pojma kroz različite članke i internetske izvore. Na navedeni način stvorena je općenita slika pojma geolokacije. Kroz sljedeća poglavlja su stečene općenite spoznaje usmjerene najprije teoretskoj obradi mogućnosti primjene, a zatim praktičnom načinu primjene. Pritom su mogućnosti primjene pronađene metodom istraživanja. Prijelaz između teoretskog i praktičnog dijela ostvaren je u petom poglavlju, gdje se najprije daje usporedba geolokacijskih servisa, a zatim opis najkorištenijeg.

S opisom najkorištenijeg geolokacijskog servisa, stečeno je dovoljno spoznaja za njegovu uporabu u razvojnom okruženju Android Studio. Konačna aplikacija realizirana je programskim jezikom Kotlin. Za samu realizaciju potrebna je bila i baza podataka, koja je stvorena uz pomoć Firebase alata.

Za realizaciju skica i dijagrama korišten je besplatni online alat Draw.io.

3. Geolokacija

Promatrajući pojam *geolokacija*, uočavamo da se sastoji od dva dijela, *geo*, što ukazuje da se nešto odnosi na zemlju i *lokacija*, što označava položaj čega. Spojimo li oba značenja možemo reći da geolokacija znači određivanje položaja na Zemlji. Još se u osnovnoškolskom obrazovanju, sve navedeno, učimo uz uporabu geografske mreže na geografskim kartama, gdje se nastoji utvrditi položaj putem geografske širine i dužine. Pri tom, računamo položaj na način da utvrdimo udaljenost promatranog predmeta od polutnika (geografska širina) i njegovu udaljenost od početnog podnevnika (geografska dužina). Udaljenosti se u ovome slučaju računaju u stupnjevima, minutama i sekundama.



Slika 1. Geografska mreža(Janko, bez datuma)

Tako ćemo za dani primjer geografske mreže sa slike, za crvenu točku reći da je 40° sjeverno od polutnika i 100° zapadno od početnog podnevnika. Sam zapis navedene lokacije bi tada izgledao na sljedeći način:

45° N 100° W.

U svrhu demonstracije složenije lokacije, navest ćemo lokaciju Fakulteta organizacije i informatike, preuzetu sa(*Fakultet Organizacije i Informatike*, 2020). Lokaciju zapisujemo na sljedeći način:

46° 18' 28.4" N 16° 20' 17.3" E.

Lako je za iščitat da se zgrada Fakulteta organizacije i informatike nalazi 46° 18' i 28.4“ sjeverno od polutnika i 16° 20' i 17.4“ istočno od početnog podnevnika.

Iako je navedena metoda određivanja lokacije u puno konkretnih slučajeva korisna, za samu mobilnu tehnologiju i lokacijske usluge (engl. *Location based services - LBS*), opisane u idućem poglavlju, je nedovoljna.

Kako bi LBS kvalitetno pružao svoju uslugu, brzo i pouzdano određivanje lokacija mobilnih uređaja je neizbježno, te zahtijeva složenije tehnologije i tehnike. Tako mnogi autori nabrajaju različite tehnike pozicioniranja mobilnih uređaja.

U idućim poglavljima ćemo nabrojati o kojim se tehnikama radi, te po kojem se principu pojedina tehnika ostvaruje.

3.1. Tehnike pozicioniranja

Različiti autori navode različite nazive koji se odnose na slične pojmove. Tako prema (Steinfield, 2004), ovisno o tome računa li sam mobilni uređaj svoju lokaciju ili mrežna ćelija na koju je spojen, izdvajamo tehnike:

- Temeljene na tehnologiji mobilnog uređaja (engl. *Handset Based*),
- Temeljene na mrežnim čelijama (engl. *Cellular Network-Based*),
- Hibridnog pristupa (engl. *Hybrid Approaches*).

Pritom, tehnike temeljene na tehnologiji mobilnog uređaja predstavljaju one tehnike gdje sam uređaj računa svoju lokaciju. Tehnike temeljene na mrežnim čelijama su one gdje same čelije računaju poziciju spojenog uređaja. Tehnika hibridnog pristupa predstavlja kombinaciju prvih dviju. Tehnika hibridnog pristupa je zbog svojih prednosti (veća preciznost, manja potrošnja baterije…) nad ostalima postala neizbježna pri izradi mobilnih aplikacija i sastavni su dio pri izradi aplikacija temeljenih na platformama poput Google Maps, Bing Maps, OpenStreetMap i dr. U nastavku će se svaka od tehnika opisati sa svojim predstavnicima, dok će pobrojanim platformama bit posvećeno jedno cijelo poglavlje.

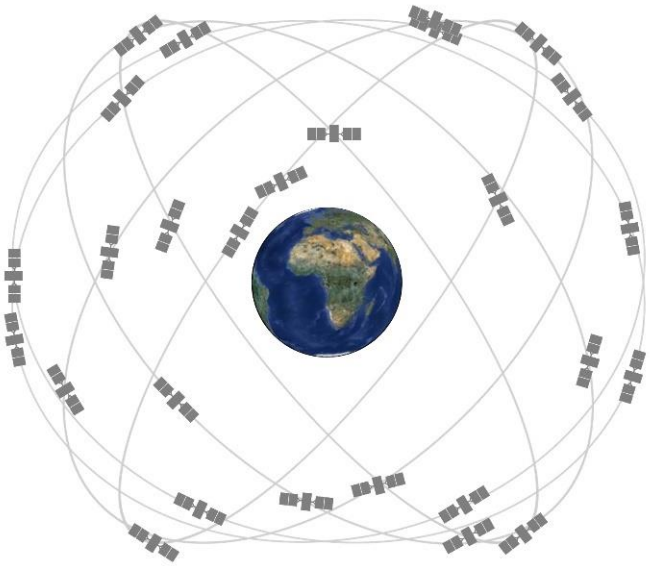
3.1.1. Tehnike temeljene na mobilnim uređajima

Već je napomenuto da kod tehnika temeljenih na mobilnim uređajima, sam mobilni uređaj nastoji izračunat svoju lokaciju. Način na koji to radi, opisat ćemo pomoću predstavnika ove tehnologije, a to je *Global Positioning System* (GPS).

GPS , kako navodi autor s (Mallick, 2003b), još je i prije 8 godina bila najrasprostranjenija tehnologija pozicioniranja, a danas ga pronalazimo u svakom *pametnom* telefonu. Velik broj izvora ukazuje na to da se ova tehnologija temelji na tri komponente, a to su:

- Svemirska komponenta,
- Kontrolna komponenta,
- Korisnička komponenta.

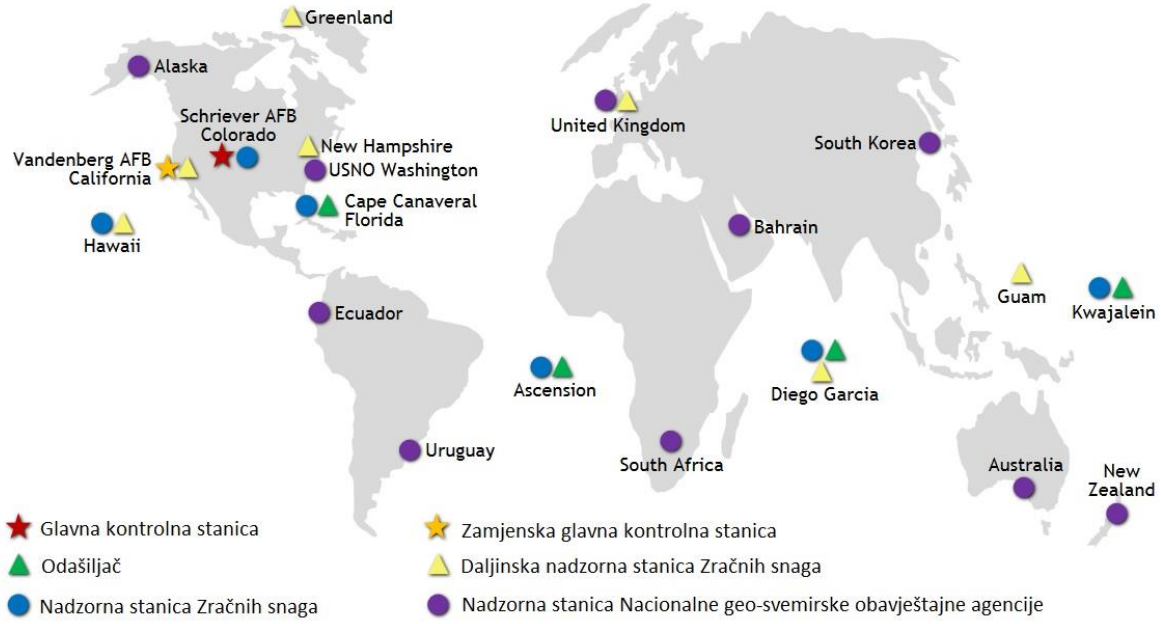
Pritom, svemirska komponenta, za kvalitetan rad zahtjeva minimalan broj od 24 satelita koji kruže na šest orbitalnih ravnina. Na svakoj orbitalnoj ravnini nalazi se 4 satelita. Sateliti se nalaze na visini od otprilike 20000 km, a svaki od njih okruži zemlju svakih 12 sati (*Space Segment*, 2020). Sateliti neprestano emitiraju radio valove niskih snaga (informacije o njihovom položaju), a GPS prijemnici (mobilni uređaji) ***slušaju*** signal. Saslušanim signalima se metodom triangulacije izračunava položaj uređaja. Triangulacija (lat. *triangulatio*), je određivanje položaja točaka u trigonometrijskoj mreži trokuta, koje se provodi mjerenjem kutova među stranicama trokuta, uz određivanje duljine najmanje jedne od stranica trokuta (*Triangulacija*, 2020). Pritom se duljina *stranice*, odnosno razmaka između satelita i uređaja računa mjerenjem vremena potrebnog signalu da dođe od satelita do uređaja.



Slika 2. Međusobni položaj satelita svemirske komponente (Space segment, 2020)

Kontrolna komponenta, po samome nazivu, kontrolira i nadzire rad cijelog sustava satelita. Prema (Bao-yen Tsui, 2000), kontrolnu komponentu je 2000. godine činio skup od 5

kontrolnih stanica koji uključuje jednu glavnu kontrolnu stanicu. Razvojem cjelokupnog sustava, danas dolazimo do nešto drugačije i složenije kontrolne komponente. GPS kontrolnu komponentu čini globalna mreža prizemnih objekata koji prate GPS satelite, nadziru njihove prijenose, obavljaju analize, te šalju naredbe i podatke konstelaciji satelita (*Control Segment*, 2018). Ukupno, prema slici 3., ovu komponentu danas čini jedna glavna kontrolna stanica, njezina zamjenska stanica, 11 kontrolnih odašiljača i 16 nadzornih stanica. Na istoj slici uočavamo i lokacije svih navedenih stanica, sa njihovim funkcijama unutar same komponente.



Slika 3. Položaji i funkcije stanica kontrolne komponente (Control segment, 2020)

Korisničku komponentu predstavlja GPS prijemnik. GPS prijemnik je uređaj (npr. mobilni telefon), koji ima mogućnost izračuna korisnikovog položaja na temelju radiovalova odašiljanih od satelita. Radio valovi prenose sljedeće podatke:

- Parametre položaja (engl. *Ephemeris parameters*),
- Parametre vremena (engl. *Time parameters*),
- Parametre usluge (engl. *Service parameters*),
- Ionosferske parametre (engl. *Ionospheric parameters*),
- Kalendarske parametre (engl. *Almanacs parameters*) [9].

Na temelju parametara položaja, mobilni uređaj računa koordinate satelita, parametrima vremena računa pomak vremena u slanju radiovalova. Parametri usluge nude

podatke o stanju satelita. Ionosferski parametri se koriste za prijemnike sa jednom frekvencijom, dok kalendarski parametri omogućuju izračun položaja svih satelita s nešto nižom preciznošću. Smjer slanja podataka je jednosmjeran, to omogućuje određivanje lokacije za bezbroj prijemnika. Za samo lociranje, uzimaju se parametri od minimalno tri različita satelita. Budući da je poznata brzina kojom signal putuje, uz vrijeme slanja, uređaji računaju udaljenost od satelita. Za pretpostavit je da za što veću preciznost lokacije, treba što veća preciznost u vremenu. U tu svrhu, sateliti koriste atomske satove (Mallick, 2003b). Po izračunu udaljenosti, uz već spomenutu triangulaciju, uređaji računaju svoju poziciju.

Karakteristike opisane tehnike pozicioniranja krase visoka preciznost, obično od 5 do 40 metara, od stvarne pozicije (Mallick, 2003b). Međutim, veliki nedostatak je činjenica, da uređaji koji primaju signale niske snage, moraju imat *čist* pogled na svoj izvor (satelite). Ukoliko se uređaj nalazi u zaklonjenom ili zatvorenom prostoru, uporaba isključivo ove tehnike za određivanje lokacije je ne pouzdana, u tu svrhu se u današnjoj uporabi kombinira sa tehnikama temeljenim na mrežnim čelijama.

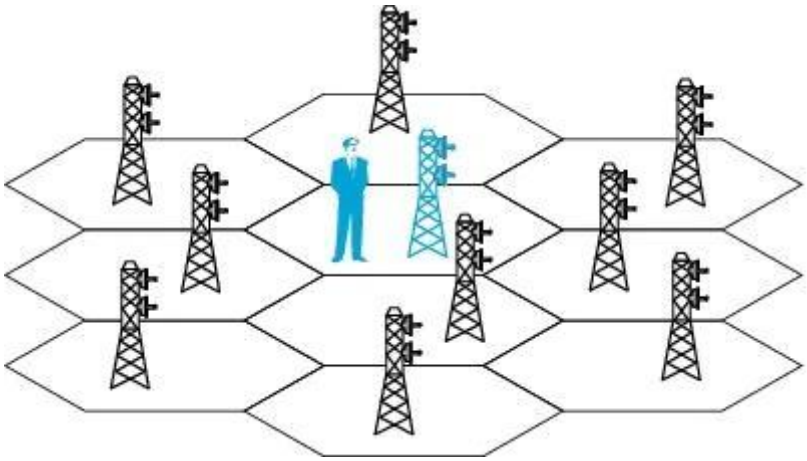
3.1.2. Tehnike temeljene na mrežnim čelijama

Ova tehnika im nešto više predstavnika od prethodne. A sve se temelje na signalima mobilne mreže koje emitiraju stanice pružatelja mobilne mreže. Točnije, prema (Steinfield, 2004), neke od tehnologija su:

- Stanica podrijetla *(engl. Cell of Origin - Cell ID*),
- Vrijeme dolaska signala (engl. *Time of Arrival – TOA*),
- Kut dolaska signala (engl. *Angle of Arrival – AOA*).

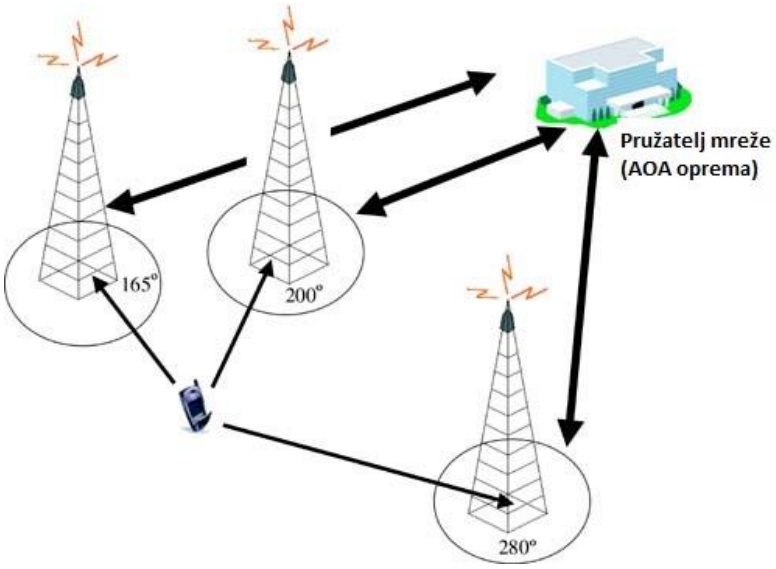
Cell ID je najjednostavniji i najjeftiniji način dolaska do informacija o poziciji (Steinfield, 2004). Razlog tomu leži u prirodi povezanosti mobilnog uređaja sa nekom od stanica pružatelja mobilne mreže. Naime, svaki mobilni uređaj spojen na mobilnu mrežu, zasigurno je povezan sa jednom stanicom koja mu pruža samu mrežu. Stanica koja mobilnom uređaju pruža mrežu, sa uređajem dijeli svoj Cell ID (identifikacijski parametar po kojem se svaka stanica razlikuje) i svoju lokaciju. Poznavajući identifikacijski parametar i lokaciju bazne stanice na koju je spojen, mobilni uređaj može ustanovit, sa ne toliko velikom preciznošću, gdje se nalazi. Naglašava se ne prevelika preciznost iz razloga što stanica može mobilnim uređajima pružat mrežu u radijusu od i do 1 kilometar. Samim time se ne može sa sigurnošću odredit gdje se mobilni uređaj točno nalazi u krugu bazne stanice. Postoje i razne tehnike dijeljenja područja

bazne stanice na više sektora, samim time se ograničava moguća pozicija uređaja, a preciznost se poveća do u 100 metara. Unatoč nedostatku preciznosti, ovakva tehnika zbog svoje niske cijene i jednostavnosti predstavlja idealnu metodu za širenje informacija gdje korisnik mora generalno znat svoju lokaciju (npr. vremenska prognoza nekog područja). Opisanu tehnologiju ćemo prikazat kroz 4. sliku, gdje uočavamo da je svaka stanica zadužena za svoj prostor. Unutar zadanog prostora lokacija mobilnog uređaja korisnika može bit proizvoljna, tj. Ne može se utvrditi sa sto postotnom sigurnošću (*Wi-Fi Location-Based Services*, n.d.).



Slika 4. Grafički prikaz tehnologije stanice podrijetla (Wi-Fi Location-Based Services, 2020)

Tehnologija AOA računa poziciju korisnika na temelju kuta dolaznog signala od dvije ili više baznih stanica (Steinfield, 2004). Slično kao i GPS, ova tehnologija koristi geometrijske odnose baznih stanica i mobilnog uređaja, te uz pomoć dostupnih parametara računa triangulacijom poziciju mobilnog uređaja. U teoriji, uz mobilni uređaj, dvije bazne stanice su dovoljne da se odredi lokacija mobilnog uređaja. Međutim u praksi, raznih smetnji i veće preciznosti radi, koristi se više baznih stanica. Pozicija uređaja utvrđuje se na način da bazne stanice uz pomoć dodatne opreme, prikupe informacije o smjeru dolaznog signala mobilnog uređaja. Prema (Mallick, 2003a), informacije se šalju pružatelju mreže na obradu, gdje se na temelju dobivenih informacija nastoji utvrdit pozicija uređaja. Primjer navedenog grafički je prikazan na slici 5.



Slika 5. Grafički prikaz rada AOA tehnologije (Angle Of Arrival, 2003)

TOA je tehnologija vrlo slična AOA tehnologiji. Bitna razlika kod ovog pristupa je da se za izračun ne koristi kut dospijevanja signala, nego vrijeme potrebno da signal mobilnog uređaja dospije do bazne stanice. Ova tehnologija se temelji na poznavanju brzine kojom signal putuje i sinkroniziranošću vremena baznih stanica. U svrhu izračuna lokacije, potrebno je poznavati i vrijeme početka prijenosa signala. Već opisani GPS sustav predstavlja primjer ovakve metode izračuna lokacije, bitna razlika je da sateliti za preciznu sinkroniziranost vremena rabe atomske satove.

Karakteristika ovakve tehnologije je uglavnom niska preciznost određivanja lokacije. Sam uređaj pritom ne mora bit u mogućnosti izvoditi bilo kakve izračune. Jedini preduvjet je da uređaj mora bit spojen na mobilnu mrežu. Veliki nedostatak je gubitak privatnosti.

3.1.3. Tehnike hibridnog pristupa

Budući da svaka od do sad nabrojanih tehnika donosi sa sobom neke nedostatke, razvijene su tehnike hibridnog pristupa. Pritom, kao i sve do sad navedene tehnologije i ova postoji u nekoliko izvedbi. Hibridni pristup pozicioniranja se ostvaruje kroz kombinaciju tehnika temeljenih na mobilnim uređajima i tehnika temeljenih na mrežnim čelijama.

U srži ova tehnika predstavlja odabir jedne od prethodno opisanih tehnika. Odabir se vrši na način da se uzmu u obzir zahtjevi lokacijskih usluga i aplikacija. Na taj način se nedostatci jedne tehnike utvrđivanja lokacije nadoknađuju drugom tehnikom. Tako u slučaju da je potrebna niža preciznost, aplikacija nastoji uštedit bateriju i prekinut korištenje GPS-a, te koristit tehniku određivanja lokacije temeljem ćelija.

Ovakve tehnike uglavnom su zadane u lokacijskim servisima, te programerima olakšavaju izradu samih usluga i aplikacija na način da ne moraju brinut o odabiru tehnike određivanja lokacije.

4. Primjena geolokacijskih servisa

Kroz ovo poglavlje prikazat ćemo mogućnost primjene geolokacijskih servisa na mobilnim uređajima. Još u prvom poglavlju spomenut je pojam *lokacijskih usluga* (engl. *Location based services - LBS*). Lokacijske usluge predstavljaju usluge, u našem slučaju aplikacije, koje se temelje na lokaciji korisnika. Točnije, prema (Barradas Pereira, 2011), LBS uključuje aplikacije koje ovisno o lokaciji korisnika pružaju usluge i informacije koje su relevantne za korisnika na toj lokaciji. Primjer takve usluge možemo dat intuitivno kroz današnje online tržnice, gdje se korisniku pružaju oni oglasi koji se nalaze do određene udaljenosti od samog korisnika. Navedeni slučaj je samo jedan između mnogih drugih korisnih slučajeva korištenja lokacije u uslugama. Potrebno je napomenut da se lokacijske usluge ne treba gledat samo kao korist za korisnika. Za same mobilne operatere, lokacijske usluge predstavljaju dodatno područje zarade, koje se u skladu s time razvija i poboljšava. Navedeni primjer online tržnice je suvremeni i razvijeni oblik lokacijske usluge. Među značajnijim oblicima bi se, prema mnogim autorima, trebalo spomenut i lokacijski marketing (engl. *Location-based advertising*). No korijeni lokacijskih usluga se svakako, kao i u većini područja znanosti, nalaze u vojnoj primjeni.

Kako bismo prikazali različite primjene geolokacijskih servisa, u idućim potpoglavljima prikazat ćemo kategorizaciju i sistematizaciju primjene geolokacijskih servisa. Potom ćemo svaku kategoriju detaljnije opisat i prikazat uz prikladni primjer.

4.1. Kategorizacija primjene

Kako se kod primjene geolokacijskih servisa radi o području znanosti koje se već duže vrijeme razvija, kroz vrijeme pronalazimo različite sistematizacije i vrste podjela. Tako na samom početku stoljeća, točnije 2003. godine, prema (Rao & Minakakis, 2003) pronalazimo podjelu i klasifikaciju lokacijskih usluga prema zahtjevima grupe korisnika. Tako imamo sljedeću podjelu:

- potrošači masovnog tržišta (engl. *Mass-market consumers*)
- specifični opći i poslovni potrošači (engl. *Niche Consumer and Business*)
- industrijski i poslovni korisnici (engl. *Business and industrial customers*)

Za detaljnu sistematizaciju je ovakva podjela preopćenita. To slijedi iz činjenice da na početku stoljeća tek počinje umrežavanje svijeta pomoću mobilnih uređaja, te da su se zahtjevi od tada do danas iz temelja promijenili ili bolje rečeno, uvelike proširili. Potražnja za lokacijskim

uslugama se proširila od navigacijskih sustava do različitih primjena (npr. zdravstvo, transport i igre) (Huang et al., 2018).

Za suvremeni prikaz primjene geolokacijskih servisa oslonit ćemo se na suvremeniji izvor, žurnal (Huang et al., 2018), gdje pronalazimo ključne domene lokacijskih usluga u zadnjih 10 godina. Ključne i glavne primjene geolokacijskih servisa su prema navedenom žurnalu:

- Lokacijski temeljene društvene mreže (engl. *Location based social networks – LBSN*)
- Lokacijski temeljene igre (engl. *Location based gaming – LBG*)
- Lokacijski temeljeno praćenje fitnesa i zdravstva (engl. *Location based fitness monitoring and healthcare*)
- Transportni LBS
- Lokacijski temeljene pomoćne tehnologije (engl. *Location based assistive technology*)

U nastavku ćemo svaki od nabrojanih primjena opisati detaljnije i prikazat kroz konkretne primjere.

4.1.1. Lokacijski temeljene društvene mreže

Prema web stranici hrvatskog strukovnog nazivlja (*Društvena Mreža*, n.d.), društvenu mrežu definiramo kao teorijski konstrukt društvene strukture određene odnosima i međudjelovanjima pojedinaca. U srži, navedena definicija, ukazuje na to da je društvena mreža društvena struktura u kojoj su pojedinci povezani sa određenim ovisnostima jedni o drugima. Ovisnosti najčešće predstavljaju prijateljstva i poznanstva u stvarnom životu ili zajednički interes za nekakvu aktivnost, mjesto, predmet ili područje znanosti. Pri tom se veze i ovisnosti ostvaruju kroz sadržaj koji korisnici generiraju i objavljuju na stranicama društvenih mreža. Generirani sadržaj najčešće je oblika fotografije, videa ili teksta.

Povećana dostupnost lokacijske tehnologije omogućuje dodavanje lokacijske dimenzije među navedeni sadržaj (Huang et al., 2018). U prenesenom značenju, korisnicima društvenih mreža se uz pomoć lokacijske tehnologije omogućuje geo označavanje sadržaja, dijeljenje trenutne lokacije i recenzija pojedinih lokacija (npr. restorana, plaža itd.). Nadalje možemo reći da se dimenzijom lokacije, korisnicima u postojećim društvenim mrežama, omogućuje proširenje društvene strukture uz pomoć ovisnosti lokacije korisnika (Zheng & Xiaofang, 2011).

Takvu *lokacijski* proširenu društvenu mrežu nazivamo *lokacijski temeljenom društvenom mrežom* (engl. *Location based social networks – LBSN*), a prema (Zheng & Xiaofang, 2011) definira se na sljedeći način:

> *Lokacijski temeljena društvena mreža (LBSN) ne znači samo dodavanje lokacije u postojeću društvenu mrežu tako da korisnici društvene strukture mogu dijelit lokacijski povezane informacije, već i stvaranje nove društvene strukture sastavljene od pojedinaca povezanih temeljem međuovisnosti njihovih lokacija u stvarnom svijetu i lokacijski označenom sadržaju koji generiraju.*

Uočavamo da je za stvaranje društvene mreže temeljene na lokaciji potrebno puno više od trenutne lokacije. Potrebno je pamćenje putanja i oznaka za određene lokacije kako bi se za druge korisnike moglo pružit korisno iskustvo pri uporabi lokacijske usluge. Konkretnije primjere aplikacija ćemo dati kroz detaljniju podjelu servisa za lokacijski temeljeno društveno umrežavanje. Prema (Zheng & Xiaofang, 2011), usluge za lokacijski temeljene društvene mreže možemo podijeliti u tri kategorije, temeljene na:

- Geooznačavanju sadržaja (engl. *Geo-tagged-media-based*)
- Točci položaja(engl. *Point-location-driven*)
- Lokacijskim putanjama (engl. *Trajectory-centric*)

Usluge temeljene na geooznačavanju omogućuju korisnicima bilježenje lokacije uz neki generirani sadržaj. Sama lokacija se pritom može samostalno dodat pri objavljivanju sadržaja ili se pak automatizirano pribilježi prilikom stvaranja sadržaja. Gotovo većina najraširenijih društvenih mreža je servis ovog tipa. Uzmemo li primjer jednu od najproširenijih aplikacija, Instagram, uočavamo oba tipa uporabe lokacije. Kod Instagrama se radi o servisu za dijeljenje fotografskog i video sadržaja. Snimimo li fotografiju kroz samu aplikaciju uz uključeno lociranje na mobilnom uređaju, aplikacija prilikom samog snimanja uz fotografiju veže i lokaciju mobilnog uređaja. Odlučimo li se odabrati već snimljenu fotografiju, prije same objave nudi se nadoknadna mogućnost odabira lokacije. Pri tom treba naglasit da i prilikom nadoknadne objave, aplikacija može koristit lokaciju snimanja fotografije u slučaju da se ona bilježi na samom uređaju i da ona predstavlja sastavni dio informacije o slici. Opisani servis i slični servisi u sebi objedinjuju dimenziju lokacije, no ona nikako ne čini glavni dio samog servisa, već samo jedan koristan dodatak. Drugačije to izgleda kod iduće opisanih servisa.

Usluge temeljene na točci položaja svoje postojanje temelje na uporabi tehnologije pozicioniranja. Kod ovakvih aplikacija dijeljenje lokacije drugim korisnicima je neizbježno za razliku od prvobitno opisanih usluga. Predstavnik ovih usluga, daleko iznad svih, je aplikacija Foursquare. Ova aplikacija pruža uslugu lokalnog vodiča na mobilnom uređaju. Pruža osobne

preporuke drugih korisnika za lokacije vrijedne posjeta u blizini korisnikove točke položaja. Preporuke se generiraju na temelju prijašnjih korisnikovih pretraživanja i posjeta drugim lokacijama (Wikipedia suradnici, 2020d). Osim navedene aplikacije, značajna je bila i Google-ova *Latitude* usluga koja je migrirala, te se potpunosti uklopila u Google Maps. Danas istu uslugu neizbježno uočavamo prilikom svakog otvaranje aplikacije Google Maps, gdje se uz korisnikovu lokaciju pokazuju različite korisne lokacije s njihovim recenzijama. U navedenim servisima se uočava puno veća korisnost od prethodno opisanih servisa. Stoga se lako zaključuje da se kod ovakvih usluga radi o onima koji se temelje na lokaciji i pozicioniranju.

Kod **usluga temeljenih na lokacijskim putanjama** se lokacija upotrebljava na način da se sa više lokacija stvori putanja. Putanja može biti čovjekova dnevna rutina, gdje bi se bilježili podaci o njegovom kretanju. U tu svrhu bi ovakav servis imao primjenu i u raznim istraživanjima. Česta primjena ovakvih servisa je i u bilježenju raznih ruta za obilazak. Rute mogu biti biciklističke, trkaće i povijesno edukativne. Tada ovakve usluge na putovanjima korisnicima omogućuju slijeđenje rute s preporukama i savjetima uz pomoć mobilnog uređaja.

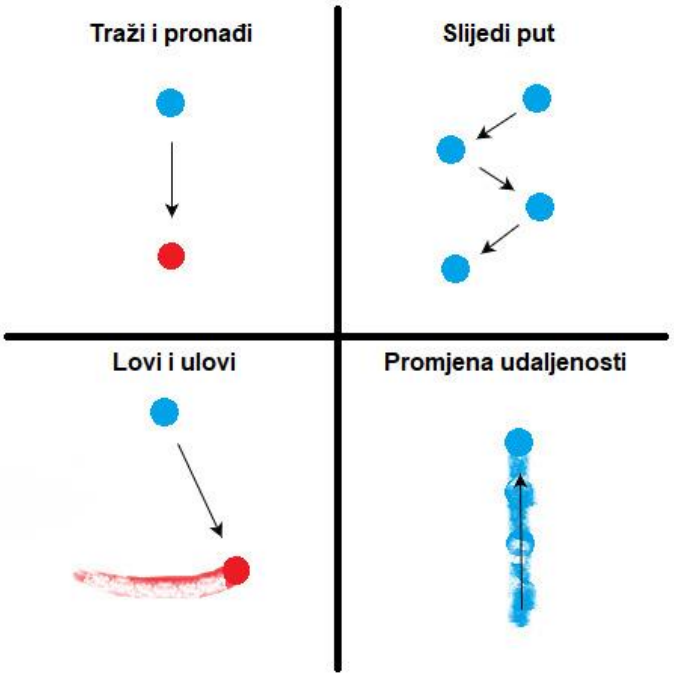
4.1.2. Lokacijski temeljene igre

Lokacijski temeljene igre su prema (Lehmann, 2012), oblik igre dizajnirane za mobilni uređaj u pokretu, gdje se korisnikovo iskustvo mijenja s obzirom na njegovu lokaciju. Drugim riječima, igra se razvija i napreduje s obzirom na promjenu lokacije. Intuitivno već znamo da je zadnjih godina upravo ovakvim aplikacijama i uslugama porasla popularnost. Jedan od najvećih uspona pronalazimo u lokacijski temeljenoj igri *Pokemon Go*[1], koja je 2016. godine otvorila i prikazala lokacijski temeljene igre široj javnosti. *Pokemon Go* je samo jedna od mnogih lokacijski temeljenih igara. Uz nju, tu su još i Geocaching, Tourality, Shadow Cities, The Journey, te mnogi drugi. Sve navedene igre se prema (Lehmann, 2012) mogu kategorizirati s obzirom na uzorak (engl. *pattern*) prema kojem se igra odvija. Tako imamo sljedeće uzorke igranja:

- Traži i pronađi (engl. *Search-and-Find*)
- Slijedi put (engl. *Follow-the-Path*)
- Lovi i ulovi (engl. *Chase-and-Catch*)
- Promjena udaljenosti (engl. *Change-of-Distance*)

[1] https://pokemongolive.com/en/

Na idućoj slici izrađenoj prema prethodno navedenoj literaturi, grafički je prikazan svaki pojedini uzorak odvijanja igre. Pri tom plavi krug predstavlja samog korisnika i njegovu lokaciju, a crveni krug lokaciju predmeta ili stvari koju treba pronaći ili ulovit.



Slika 6. Grafički prikaz uzoraka igara, po uzoru na (Lehmann, 2012)

Traži i pronađi je popularan tip igre koji kod korisnika izaziva najčešće fizičku aktivnost. U ovome tipu igre korisnik ima obavezu pronalaska određene lokacije (Lehmann, 2012). Sama igra je pri tom koncipirana na način da usmjerava korisnika prema traženoj lokaciji. Iz toga slijedi da se samog korisnika ne usmjerava direktno prema određenim koordinatama, nego mu se sugerira pronalazak određenog objekta u blizini tražene lokacije. Kao najzastupljeniji primjer ovakve vrste igre možemo uzet *Geocaching.* Radi se o aktivnosti proširenoj diljem svijeta. *Geocaching* je rekreacijska aktivnost na otvorenom, u kojoj sudionici koriste mobilne uređaje i drugu navigacijsku opremu kako bi sakrili i pronašli *geospremnike* (engl. *geocaches*) na specifičnim lokacijama označenim koordinatama diljem svijeta (Wikipedia suradnici, 2020b). *Geospremnik* predstavlja kutiju sa nekoliko stvari, a cilj je neku stvar uzet iz kutije i zamijenit ju drugom.

Slijedi put je uzorak igre vrlo sličan uzorku traži i pronađi. Razlika je u samom pronalasku i dolasku do lokacije. U prvom slučaju nije bitno kojim putem se dođe do

lokacije, bitno je da se dođe. U drugom slučaju se putanja zadana aplikacijom mora obavezno pratit. Praćenje putanje je sastavni dio igre. Odstupanje od putanje rezultira penalizacijom za korisnika. Često su to igre utrkivanja, gdje se korištenjem ne dopuštenih kratica gubi pravo na pobjedu. Primjer ovakve igre je *Tourality.* Radi se o aplikaciji dizajniranoj za utrke kroz grad. Igrači u ovoj igri imaju obavezu posjetit sva zadana mjesta na nekoj putanji, bilo kojim redoslijedom, u što kraćem vremenu.

Lovi i ulovi predstavlja daleko najpopularniji oblik lokacijski temeljenih igara. Prema samom nazivu, od igrača se u ovakvoj igri očekuje hvatanje nekog drugog igrača ili virtualnog objekta. Virtualni objekt je objekt koji postoji samo u svijetu igre, a sam svijet igre je virtualno izrađen po uzoru na stvarni svijet. Ovisno o učestalosti promjene lokacije objekta koji se lovi, ovakva igra može bit izuzetno izazovna. No unatoč tomu, kod ovakvih igara fokus leži u zabavi a ne u fizičkoj aktivnosti. Najpopularniji primjer ovakve igre već je spomenut, a to je igra *Pokemon Go.* Radi se o aplikaciji za mobilne uređaje putem koje se korisnika, uz pomoć tehnika lociranja, stavlja na Googleovu kartu svijeta. Uz njega, na samoj karti, aplikacija generira virtualne objekte naziva *Pokemon.* Kao i u izvornoj igri, igrači i u ovoj igri lovi virtualno stvorene objekte. Krajnja težnja je uloviti što više različitih *Pokemona*.

Rjeđe zastupljen oblik lokacijski temeljenih igara je uzorak tipa **promjena udaljenosti**. Prema samom nazivu, cilj u ovakvoj igri je približit se ili udaljit od neke zadane lokacija. U ovakvoj igri sama lokacija i smjer kretanja nije u fokusu, već udaljenost koju je potrebno prebrodit da bi se postigao uspjeh u igri. Neke od popularnih igara ovog tipa su *The Journey* i *Savannah*.

S obzirom na svrhu same lokacijski temeljene igre, prikazat ćemo nekoliko igara i njihove fokuse s obzirom na tip kojem pripadaju prema prethodnoj podjeli. Tablica je izrađena po uzoru na (Lehmann, 2012).

Igra	Uzorak igre				Mogućnosti korištenja				
	Traži i pronađi	Slijedi put	Lovi i ulovi	Promjena udaljenosti	Zabava	Obrazovanje	Fizička aktivnost	Reklamiranje	Prikupljanje podataka
Geocaching	+	-	-	-	+	-	+	-	-
Tourality	-	+	-	-	+	-	+	-	-
FoxHunt	-	-	+	-	+	-	-	-	-
Shadow Cities	-	-	+	-	+	-	-	-	-
Mobile Hunters	-	-	+	-	+	-	-	-	-
Botfighters	-	-	+	-	+	-	-	-	-
The Journal	-	-	-	+	+	-	-	-	-
Savannah	-	-	-	+	+	+	-	-	-
CityExplorer	+	-	-	-	+	-	-	-	+
FIASCO	+	-	-	-	+	-	-	-	+

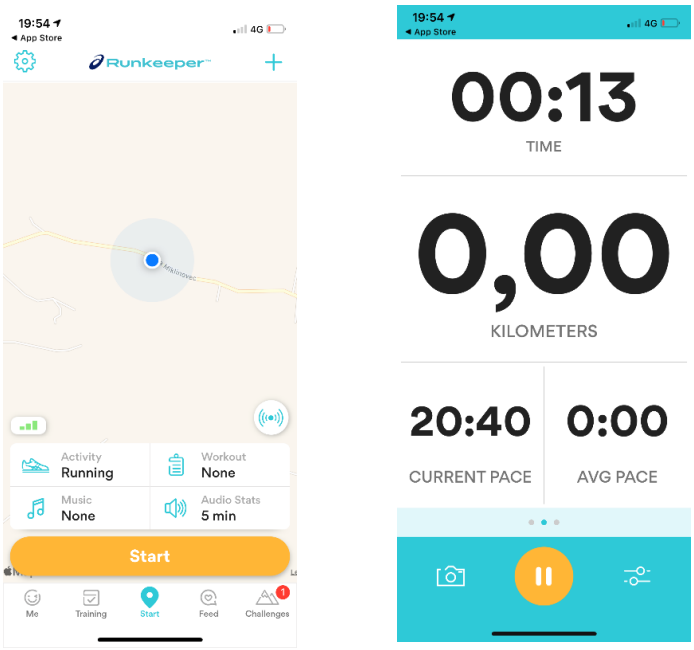
Tablica 1. Pregled nekoliko lokacijski temeljenih igrica, njihovih uzoraka i namjena (Lehmann, 2012)

Jednostavno je za uočit da je osnovna namjena lokacijski temeljenih igara zabava. Uz zabavu nastoje se u igru uklopit i druge namjene. Najčešće su to obrazovanje ili fizička aktivnost. Lako se zaključi da LBG predstavlja ostvarenje tradicionalnih igara na inovativan način koristeći lokacijske tehnologije.

4.1.3. Lokacijski temeljeno praćenje fitnesa i zdravlja

Razvojem usluga i aplikacija za praćenje zdravstva i fitnesa, neizbježna je postala implementacija lokacijskih tehnologija u iste. Takvom su implementacijom nastale usluge koje temeljem lokacije prate fitnes i zdravstvo korisnika. Ovakvi servisi uglavnom su usmjereni praćenju korisnikovog zdravlja, aktivnosti na otvorenom (npr. tračanje i kretnje na otvorenom), i pacijenata sa demencijom, te hitnim slučajevima (Huang et al., 2018).

Prema navedenom, u jednu ruku, ovakvi servisi korisniku mogu pomoć u fizičkom napredovanju. Primjer takvog slučaja je aplikacija *Runkeeper*[2]. *Runkeeper* je GPS aplikacija za praćenje fitnesa za mobilne uređaje. Ova aplikacija prati fitness aktivnosti poput hodanja, trčanja i bicikliranja koristeći tehnologije pozicioniranja. Kada pojedina aktivnost korisnika završi, aplikacija pruža korisniku njegove statističke podatke u praćenoj aktivnosti (Runkeeper, 2020). U svrhu pružanja što preciznijih i kvalitetnijih statističkih podataka, ovakve aplikacije uz tehnologije pozicioniranja koriste i druge senzore mobilnih uređaja.



Slika 7. Sučelje aplikacije Runkeeper

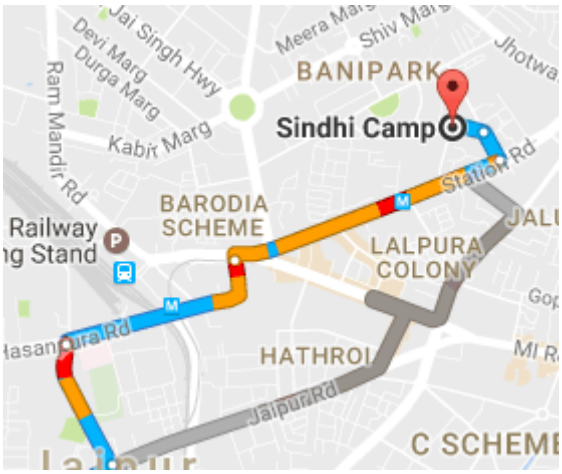
[2] https://runkeeper.com/cms/

S druge strane ovi servisi su korisni i u hitnim slučajevima. Radi se o situacijama kada hitne službe i službe pomoći uz lokacijske tehnologije mogu brzo i precizno utvrditi poziciju osoba kojima je potrebna medicinska pomoć, te osoba koje su izgubljene na nekom području. Takve aplikacije uz lokacijske tehnologije koriste, kao i u prethodnim slučajevima, razne druge senzore za mjerenje fizičkog stanja osobe.

4.1.4. Transportni LBS

Široj javnosti najpoznatiji lokacijski servisi su oni za usmjeravanje i pomoć u prometu, poznati pod nazivom *navigacija*. Takvi servisi spadaju u skupinu transportnih lokacijskih usluga. Osim poznatih navigacijskih usluga, u ovu skupinu spadaju i aplikacije koje pružaju informacije putnicima i upravljanje vozilima (Huang et al., 2018).

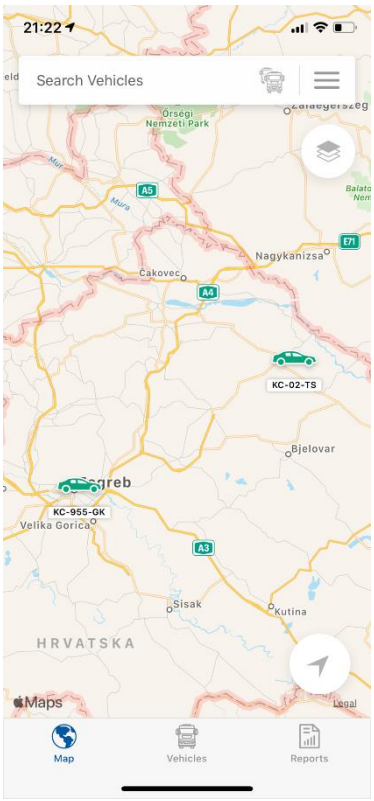
Navigacijski sustavi za vozila su daleko najprošireniji vrsta transportnih lokacijskih usluga. Sastavni dio svakog novo kupljenog mobilnog uređaja čini i navigacija. Radi se o usluzi koja je prvobitno stvorena kako bi korisnicima olakšala pronalazak puta do određene lokacije. Razvojem te usluge, nastale su mnogobrojne dodatne značajke koje uz samo usmjeravanje, korisnicima pružaju relevantne podatke vezane uz rutu usmjeravanja. Među najpopularnijim značajkama navigacijskih usluga je *stanje prometa uživo* (engl. *real-time traffic information*). Radi se o značajci koja unutar navigacijskog sustava, tijekom samog usmjeravanja, korisniku na neki način pruža korisne informacije vezane uz njegovu putanju. U poznatoj Googleovoj usluzi *Maps* možemo izdvojit primjer gdje se sama boja rute prikazane u aplikaciji ovisno o gustoći prometa promijeni. Prema slici 8., rijedak promet na ruti je označen plavom bojom, ovisno o većoj ili manjoj gustoći ruta je na mjestima označena crvenom ili narančastom bojom.



Slika 8. Primjer Googleove Maps usluge za informacije u prometu uživo

Još neke od popularnih dodatnih značajki raznih navigacijskih usluga su prikaz slobodnih parkinga ili *najzelenije rute.*

Osim navigacijskih usluga, velik dio transportnih lokacijskih usluga na tržištu zauzimaju mobilni sustavi za upravljanje vozilima logističkih i prijevoznih tvrtki. Među popularnijim uslugama izdvajamo aplikaciju *mobileMAP*[3] tvrtke *CVS Mobile*[4]. Radi se o aplikaciji koja služi praćenju većeg broja vozila voznog parka tvrtke. Aplikacija prvobitno služi pregledavanju lokacija pojedinih vozila. Dodatne funkcionalnosti koje pruža su različiti statistički podaci za pojedino vozilo. Aplikacija pohranjuje putanju kretanja pojedinog vozila, brzino kretanja, potrošnju goriva i vrijeme vožnje ili stajanja. Svi navedeni podaci se daju filtrirat i grafički prikazat pored karte na kojoj su prikazane lokacije vozila uživo. Primjer izgleda opisane mobilne aplikacije dan je na slici ispod.



Slika 9. Primjer izgleda aplikacije mobileMAP

[3] https://cvs-mobile.com/hr/rje%C5%A1enja/Mobilna-aplikacija
[4] https://cvs-mobile.com/hr

4.1.5. Lokacijski temeljene pomoćne tehnologije

Od svih navedenih i opisanih vrsta lokacijskih usluga, lokacijski temeljene pomoćne tehnologije, najrjeđe su zastupljene. Ovakve lokacijske usluge koriste se kao pomoćne tehnologije ljudima s oštećenjima vida i poteškoćama, te starijim ljudima kako bi obavljali svoje dnevne aktivnosti i poboljšali svoju kvalitetu života (Huang et al., 2018). Kako bi se navedenim korisnicima olakšale svakodnevice, razvijaju se usluge i aplikacije koje im pomažu pri orijentaciji u za život potrebnim aktivnostima (npr. odlazak u trgovinu i pronalazak puta). Upravo kod starije populacije sa oboljenjem od demencije, ovakve usluge su od velikog značaja. Naočigled se kod ovakvih aplikacija radi o navigacijskim sustavima, koji korisnicima pružaju dodatne informacije za opažanje i izbjegavanje prepreka ovisno o njihovim poteškoćama.

4.2. Privatnost u lokacijskim uslugama

Sve prisutnije lokacijske usluge osim velike koristi, za korisnike često predstavljaju rizik u okvirima privatnosti. Sam rizik nastaje pri slanju podataka poslužitelju o lokaciji korisnika. Samim slanjem podaci postaju ranjivi i podložni prisluškivanju. Sama lokacija korisnika ne predstavlja preveliku povredu privatnosti, problem se javlja kada lokacija prema (Parmar & Pratap Rao, 2020) postane kvazi-identifikator (engl. *quasi-identifier*). Tada se korisnika lokacijske usluge može identificirat na temelju lokacije. Prema mnogim autorima, očuvanje privatnosti se u lokacijskim uslugama ostvaruje uz pomoć k-anonimnosti (engl. *k-anonymity*) modela privatnosti. Točnije, uz pomoć tehnika očuvanja privatnosti temeljenim na lažnim lokacijama koje nazivamo *lutkama* (engl. *Dummy generation based privacy preservation techniques*) (Parmar & Pratap Rao, 2020).

> *Lokacijski lutkama temeljene tehnike očuvanja privatnosti izrađuju se po principu k-anonimsnosti, koncept koji je dobro istražen u području privatnosti baza podataka. U ovim tehnikama, k-anonimnost se ostvaruje kroz lokacijske lutke. Korisnik lokacijske usluge šalje svoju stvarnu lokaciju i k-1 lokacijskih lutki kako bi dobio k-anonimnost između ostalih lokacija* (Parmar & Pratap Rao, 2020).

Prema navedenom, korisnik uz svoju lokaciju šalje i k-1 broj lažnih lokacija kako bi se napadačima otežala identifikacija stvarne lokacije. Postoje razne tehnike (napadi) za otkrivanje stvarne lokacije, prema (Parmar & Pratap Rao, 2020), neke od njih su temeljene na podudaranju lokacija i homogenosti lokacija

Podudaranje lokacija (engl. *Map-matching attack*) proizlazi iz činjenice da se lokacijske lutke biraju nasumičnim načinom. Nasumični način ne garantira da će se uvijek radit o realističnoj lokaciji. Napadač ovakvim napadom eliminira nerealistične lokacije poput rijeka, planina i sl. Homogenost lokacija nastaje kada se velik broj nasumično generiranih lokacija generira u blizini stvarne lokacije, tada se stvarna lokacija zbog niske razine raznovrsnosti može približno utvrdit.

Rješavanje ovih problema može se ostvarit poboljšanjem algoritama generiranja lokacijskih lutaka, na način da se same lokacije i njihova homogenost, prije slanja poslužitelju, provjeri i filtrira iz popisa generiranih lokacija.

4.3. Budućnost lokacijskih usluga

Budućnost lokacijskih usluga se može samo teško pretpostaviti. Što se zasigurno može reći, prateći trendove zadnjih godina, je da se lokacijske usluge svakih nekoliko godina pronalazi u novim područjima. Bilo to zdravstvo, fitness, igranje ili logistika, u svim ovim područjima su lokacijske tehnologije poboljšale razinu kvalitete usluga.

Posebno bi trebalo istaknuti trenutnu korist lokacijskih usluga. Točnije u suzbijanju jedne od najvećih globalnih pandemija, COVID-19. Lokacijske tehnologije bi se mogle koristit na način da se posebno razvijenom uslugom, prate sve osobe koje su prema lokaciji bile s korisnikom u kontaktu. U slučaju da neka osoba dobije pozitivan laboratorijski nalaz na COVID-19, aplikacija korisnika obavještava o tome je li bio u kontaktu sa zaraženom osobom, navodi datum, te predlaže sljedeće mjere. Slična aplikacija već je i razvijena, naziva je *Stop COVID-19*. Ova aplikacija čini upravo navedeno, no bitna je razlika da se temelji na *Bluetooth* tehnologiji kratkog dometa, a ne na lokacijskim servisima.

Razlog primjene *Bluetooth* tehnologije u primjeru navedene aplikacije je zaštita osobnih podataka. Svaki put kada korisnik pristupi lokacijskoj usluzi, on otkriva svoje podatke o lokaciji i osobne informacije, poput onih gdje živi, o stilu života, posjetama restoranima, bolnici ili odmoru na neko vrijeme (*Location-Based Services (LBS) and Real-Time Location Systems (RTLS) Market*, 2020). U svrhu sprječavanja širenja zaraze virusom i istovremenu zaštitu privatnosti, mnogi autori spominju poseban model korištenja lokacijskih i osobnih podataka u aplikacijama za suzbijanje pandemije COVID-19. Radi se o modelu decentralizirane pohrane podataka koji se u skladu sa vremenom trajanja inkubacije virusa, brišu svakih četrnaest dana. Decentralizirana pohrana podataka znači da korisnikovi podaci ne napuštaju njegov mobilni uređaj. Komunikacija između drugih korisnika i spoznaje o njihovoj zarazi se razmjenjuju putem nasumično generiranih ključeva, koji ne sadrže nikakve osobne podatke. Navedeni model široko je prihvaćen u zapadnim zemljama, dok u istočnim zemljama fokus pri razvoj ovakvih aplikacija ne leži u zaštiti privatnosti.

Navedeno je samo primjer toga da lokacijska tehnologija i iznenadnim situacijama može ponudit korisno rješenje i pomoć. Unatoč iznenadnih situacija, tržište lokacijskih usluga bi prema predviđanjima do 2026. godine trebalo zabilježiti gotovo četverostruki rast. Tako prema (Arterburn, 2020), veličina tržišta lokacijskih usluga u 2018. iznosi 16.14 milijardi dolara, dok bi u 2026. godini trebalo iznosit čak 66.61 milijardi dolara. Isti autor navedeno pripisuje naglom razvoju *pametnih gradova*.

Iz opisanog u navedenom poglavlju uočavamo da su lokacijske usluge unatoč velikom prodiranju u privatnosti korisnika gotovo ne izbježen, te da se pri razvoju aplikacija velika težina daje zaštiti privatnosti. Način uporabe geolokacijskih servisa pri razvoju lokacijskih usluga, te općenita usporedba pojedinih servisa dana je u idućem poglavlju.

5. Usporedba i tehnički opis geolokacijskih servisa

Iako se prilikom spominjanja servisa za prikaz web karte na mobilnom uređaju najprije pomisli na Google Maps servis, činjenica je da postoje i mnogi drugi servisi koji u različitim slučajevima mogu predstavit kvalitetniju i povoljniju zamjenu za Google Maps. U tu svrhu ćemo u ovome poglavlju dat općenitu usporedbu značajnijih servisa i malo detaljniji opis Google Maps servisa. U usporedbi bit će prikazane općenitosti pojedinih servisa, dok će u detaljnom opisu bit prikazane mogućnosti i načini uporabe servisa.

5.1. Usporedba servisa

Uz već spomenuti servis Google Maps, u nešto rjeđem broju korisnika, prema (Comparison of web map services, 2020), pronalazimo i servise poput OpenStreetMap[5], Bing Maps[6], MapQuest[7], te Apple Maps[8]. Svrha ovakvih servisa je s jedne strane razvojnim programerima olakšati stvaranje lokacijskih aplikacija i usluga, a s druge strane korisniku pokazati gdje je, te prema potrebi, kamo se mora kretati. Budući da svi od navedenih servisa ispunjavaju svoju svrhu, teško je reći koji od navedenih je bolji. Prema tablici 2., sastavljenoj po uzoru na (Comparison of web map services, 2020), uočavamo nekoliko zanimljivosti. Svaki od navedenih servisa pruža dovoljan broj tipova karti za razne primjene. Svoje primjene pronalaze i u uređajima velikih kompanija poput onih iz automobilske industrije. Informacije uživo o prometu podržava svaki servis, no uočavamo jednu bitnu značajku. Radi se o tome da je OpenStreetMaps program *otvorenog kôda* (engl. *Open Source*), te da neki od pobrojanih servisa svoje temelje vuku iz OpenStreetMaps-a. Stoga informacije o prometu uživo predstavljaju u nekim od pobrojanih servisa značajku nadograđenu na OpenStreetMaps. Što se tiče samih lokacija u pojedinim servisima, svaki od njih prikazuje minimalni skup informacija o poštanskom broju, nazivu ulice i grada, te koordinatama. Prema razini približavanja Google Maps pokriva gotovo sav prostor sa svojom maksimalnom razinom, dok preostali servisi samo djelomično nude maksimalno približavanje.

[5] https://www.openstreetmap.org/#map=7/44.523/16.460
[6] https://www.bing.com/maps
[7] https://www.mapquest.com/a
[8] https://www.apple.com/maps/

Servis	Google Maps	Bing Maps	MapQuest	OpenStreetMap	Apple Maps
Tip karte	Satelitski, terenskim, cestovni, hibridni	Cestovni, satelitski, hibridni, 3D, prometni	Standardni, satelitski	Standardni, terenski, satelitski	Standardni, satelitski, terenski, hibridni, 3D
Primjena	Google Earth, BMW Assist, Tesla Navigation	Microsoft SharePoint, Bing Weather, Telescope	-	Apple Maps, MapQuest, Foursquare, Craigslist, Wikipedia	CarPlay
Informacije uživo	Da	Da	Da	Djelomično	Da
Lokacija	Poštanski broj, ulica, grad, naselje i koordinate	Poštanski broj, ulica, grad, naselje, orijentir, regija	Poštanski broj, ulica, grad, regija	Poštanski broj, ulica, grad, naselje, regija, koordinate	Poštanski broj, ulica, grad, naselje, regija, koordinate
Razina približavanja	22	19-22	17	19	vektorski

Tablica 2. Usporedba značajnijih značajki servisa

U nastavku ćemo detaljnije usporediti dva široj javnosti dobro poznata servisa, a to su Google Maps i OpenStreetMaps.

5.2. Google Maps i OpenStreetMaps

Već smo u prethodnom poglavlju, govoreći o raznim geolokacijskim servisima, dali nešto općenitiju usporedbu različitih servisa, no konkretnija usporedba dvaju popularnijih servisa tek slijedi. Naglašava se dvaju popularnijih servisa iz razloga što mnogi web portali ukazuju na udio tržišta od iznad 90% koji zauzima Google Maps, te prema tablici 3., manji od 5% koji zauzima OpenStreetMaps. Udjeli na tržištu za značajnije servise, prikazani su u tablici 3. izrađenoj prema (*Web Mapping Products*, 2020).

Servis	Broj kompanija koji koristi servis	Udio na tržištu(%)
Google Maps	1.075.449	92%
MapBox	42.707	<5%
Leaflet	25.442	<5%
MapQuest	3.288	<5%
Bing Maps	3.106	<5%
Yandex Maps	3.054	<5%
OpenStreetMap	3.027	<5%

Tablica 3. Udio servisa na tržištu

Prema (AgaRwal, 2018), OpenStreetMap predstavlja alternativu Google Maps-a, samim time bi se moglo lako pomislit da je inferioran, no pogledamo li neke od značajki poput cijene, performansi i ažurnosti, uvjeravamo se u suprotno.

Na području **cijene** OpenStreetMap samom činjenicom da je *open source*, ostaje nepobjediv. Svaki od servisa pruža dodatke poput funkcionalnosti za pretvaranje koordinata, usmjeravanje korisnika i drugih aplikacijskih programskih sučelja (engl. *application programming interface, API*), mnogi od takvih dodataka se od strane Google-a naplaćuje po broju korisnika na mjesečnoj bazi. U slučaju neizbježno potrebnog smanjenja troška razvoja, OpenStreetMap sa svojim besplatnim korištenjem, predstavlja idealnu zamjenu.

Samim time **performanse** pate, no ne toliko drastično. Google Maps pruža veliku brzinu učitavanja dijelova, bolje performanse mobitela, te ima veću pokrivenost područja, no

definitivno je i manje fleksibilan naspram OpenStreetMap-a, koji pruža mogućnost individualnog prilagođavanja servisa (AgaRwal, 2018).

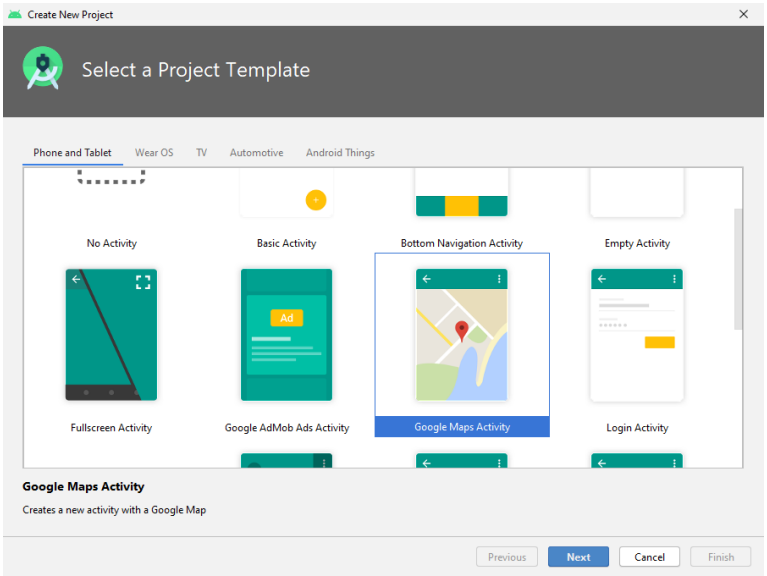
U individualno prilagođavanje servisa osim sa strane performansi možemo gledat i sa strane **stila** prikaza karte. Iako je Google-ova izvedba i odabir boja na karti prepoznatljiv i atraktivan, OpenStreetMap pruža potpunu kontrolu pri odabiru i oblikovanju stila karte.

Veliku razliku između ova dva servisa čine **podaci** koje pružaju. OpenStreetMap je zapravo velika otvorena baza podataka puna geoprostornih podataka (Tarasenko, 2019). Uz to ona ima najveću zajednicu svih geografskih informacijskih sustava (engl. *Geographic information system, GIS*). Navedena zajednica svakodnevno skuplja i strukturira geografske podatke te ih pohranjuje u bazu podataka. Samim time OpenStreetMap nudi ažurnost podataka visoke razine naspram Google Maps-a i drugih komercijalnih servisa, koji svoje podatke ažuriraju većinom nakon dužeg vremenskog razdoblja.

Iako sve navedeno ukazuje na to da OpenStreetMap dobro konkurira Google Maps servisu, činjenica ostaje da je Google Maps daleko najrasprostranjeniji geolokacijski servis, temeljem toga je u nastavku dan opis njegove uporabe.

5.3. Uporaba Google Maps servisa

Početak korištenja Google Maps servisa prikazat ćemo unutar razvojnog okruženja Android Studio. Navedeno razvojno okruženje je službeno razvojno okruženje za aplikacije operacijskog sustava Android. U skladu s time nudi lakše stvaranje aplikacija ovisno o svrsi primjene aplikacije. Tako u slučaju da želimo u aplikaciju implementirati Google Maps, prilikom stvaranja novog projekta (File -> New -> New Project), u prozoru za stvaranje projekta, odabiremo predložak *Google Maps Activity*, a potom kliknemo na *Next.*



Slika 10. Odabir predloška novog projekta

Po otvaranju novog prozora prema slici 10., unosimo podatke o nazivu projekta, putanji pohrane projekta, programskom jeziku korištenom i minimalnom SDK. Naziv projekta i putanju pohrane biramo prema želji. Što se tiče programskog jezika, bira se između Kotlin i Java programskog jezika. Za našu aplikaciju prikazanu u idućem poglavlju korišten je Kotlin programski jezik. Minimalni SDK omogućuje odabir minimalne android platforme koja će moć pokretati našu aplikaciju. Što je verzija Androida manja, to će bit veća pokrivenost uređaja koji će našu aplikaciju moć pokrenuti. Po kliku na gumba *Finish*, Android Studio generira gotov *kôd* za prikaz Google-ove karte svijeta. Generirani *kôd* uključuje 3 ključne datoteke: kotlin datoteku MapsActivity.kt i xml datoteke activity_maps.xml, te google_maps_api.xml.

Sadržaj Kotlin datoteke MapsActivity.kt je prikazan na idućoj slici, te sadrži nekoliko bitnih značajki i metoda.

```kotlin
package com.example.nazivaplikacije

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.MarkerOptions

class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var mMap: GoogleMap

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        val mapFragment = supportFragmentManager
                .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap

        // Add a marker in Sydney and move the camera
        val sydney = LatLng(-34.0, 151.0)
        mMap.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))
    }
```

Slika 11. Sadržaj MapsActivity.kt datoteke

Uočavamo da ova Kotlin datoteka sadrži implementaciju sučelja (engl. *interface*) *OnMapReadyCallback*. Radi se o sučelju koje se *okida* u trenutku kada je karta spremna za korištenje. Navedeno sučelje sadrži prijepis (engl. *override*) metoda *onCreate* i *onMapReady*. Unutar prve metode uočavamo poziv metode *setContentView*, koja služi za povezivanje Kotlin datoteke sa pripadajućom xml datotekom. Pripadajuća xml datoteka je pritom *activity_maps.xml,* te služi oblikovanju izgleda korisničkog sučelja aktivnosti. Unutar prijepisa OnMapReady metode uočavamo redom inicijalizaciju karte, definiranje koordinata za grad Sydney, te dodavanje oznake na karti za definirane koordinate pomoću metode *addMarker*.

Pripadajuća xml datoteka za opisanu Kotlin datoteku je *activity_maps.xml*. Sadržaj inicijalne datoteke je prikazan na idućoj slici.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <fragment xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:map="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:id="@+id/map"
6      android:name="com.google.android.gms.maps.SupportMapFragment"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      tools:context=".MapsActivity" />
```

Slika 12. Sadržaj activity_maps.xml datoteke

Kako bi se karta uspješno prikazala potrebna je dodatna komponenta koju nazivamo *fragment*. *SupportMapFragment* komponenta predstavlja modularan način dodavanje karte u aplikaciju.

Sve do sad navedeno predstavlja temelj prikaza Google-ove karte putom mobilne aplikacije. Budući da se kod Google Maps-a radi o komercijalnom servisu, potrebno je dodati i ključ (engl. *Key*) koji omogućuje korištenje servisa. Ključ se dodaje u generiranoj xml datoteci *google_maps_api.xml*. Na slici 13. prikazan je sadržaj navedene datoteke, a ispod nje dan je opis njenog sadržaja.

Slika 13. Sadržaj google_maps_api.xml datoteke

Na samom početku dokumenta uočava se obavijest o potrebnom Google Maps API ključu. Ispod same obavijesti dan je link koji neposredno vodi do stranice za nabavu ključa preko Google-ovog korisničkog računa. Po nabavku ključa, isti upisujemo u zato predviđeno mjesto unutar *string* elementa, umjesto poruke *YOUR_KEY_HERE.*

Nakon obavljanja navedenog, jedino što je potrebno obavit je dodati *Google Play Services ovisnost* (engl. *dependencies*) u *build.gradle*. Navedeni pojam predstavlja vanjski modul biblioteke (engl. *library*). U tu svrhu, otvori se build.gradle datoteka projekta, te se unutar uglatih zagrada *dependencies* dijela dodaje programski kôd prikazan na slici ispod. Navedeni kôd označava dodavanje Google-ovog *maps* API-a u projekt.

```
implementation 'com.google.android.gms:play-services-maps:17.0.0'
```

Slika 14. Ovisnost potrebna za prikaz Google-ove karte

Po obavljanju svega navedenoga, trebali bismo pokretanjem aplikacije na emulatoru ili uređaju vidjeti kartu sa označenim gradom Sydney, kao na slici 15.

Slika 15. Izgled sučelja pokrenute aplikacije

Sve navedeno predstavlja temelj pri izradi lokacijskih aplikacija i usluga. Osim navedenog potrebno je i puno drugih metoda i klasa koje se standardno upotrebljavaju za izradu lokacijskih usluga. Najčešće su to metode za pomicanje, dodavanje i uklanjanje markera, geokodiranje koordinata, te zahtijevanje lokacije korisnika. Za navedene aktivnosti, u kombinaciji sa Google Maps servisom, koristi se i Google lokacijski servis.

5.4. Tehnički opis Google lokacijskog servisa

Budući da je kod lokacijskih usluga i aplikacija osim prikaza karte i oznake lokacije potrebno koristiti i druge podatke, poput trenutne lokacije korisnika i sl., u izradi aplikacije osim *maps* API-a koristimo i *location* API. Ovaj API sadrži korisne klase i sučelja koja programerima omogućavaju olakšani pristup podacima o lokaciji mobilnog uređaja. Samu biblioteku za navedeni API, u projekt dodajemo u već spomenutu build.gradle datoteku na isti način kao što smo dodali Google-ov *maps* API. Sam kôd za dodavanje navedene biblioteke prikazan je

na slici 16. Budući da ova biblioteka sadrži velik broj sučelja i klasa u nastavku su nabrojane neke značajnije, te je dan njihov tehnički opis.

```
implementation 'com.google.android.gms:play-services-location:17.0.0'
```

Slika 16. Ovisnost potrebna za korištenje lokacije mobilnog uređaja

Značajnija sučelja iz navedene biblioteke, za izradu lokacijski svjesne aplikacije (engl. *Location aware application*) su *FusedLocationProviderApi* (ukinut i zamijenjen sa klasom FusedLocationProviderClient) i *LocationListener*, dok je klasa nešto više. Među klase ubrajamo već spomenuti *FusedLocationProviderClient*, *LocationCallback*, *LocationRequest*, *LocationServices* i *LocationSettingsRequest*. Podaci za opis navedenih klasa i njihovih metoda u nastavku, preuzeti su sa službene Google-ove stranice (*Com.Google.Android.Gms.Location*, 2019), koja sadrži dokumentaciju istih.

5.4.1. Sučelje LocationListener

LocationListener je javno sučelje (engl. *public interface LocationListener*) koje se koristi za dobivanje obavijesti o promijeni lokacije uređaja. Metoda ove klase poziva se svaki put kada aplikacija koristi metodu *requestLocationUpdates*. Opis metode dan je u idućoj tablici.

Tip	Metoda i parametri	Opis metode
abstract void	onLocationChanged(Location location)	Metoda se poziva svaki put kada se lokacija uređaja promijeni. Parametar *location*, označava promijenjenu lokaciju.

Tablica 4. Metoda sučelja LocationListener

5.4.2. Klase servisa

Biblioteka Google-ovog lokacijskog servisa sadrži ukupno 29 klasa. Budući da ih je samo nekoliko neizbježnih sa kreiranje lokacijske usluge, u nastavku ćemo dat njihov opis i popis njihovih metoda sa kratkim opisom.

5.4.2.1. FusedLocationProviderClient

Javna klasa *FusedLocationProviderClient* (engl. *public class FusedLocationProviderClient*) predstavlja glavnu pristupnu točku prilikom interakcije sa spojenim pružateljem lokacije (engl. *fused location provider*). Fused location provider lokacijski je API koji hibridnim načinom prikuplja podatke o lokaciji uređaja. Popis i opis značajnijih metoda ove klase dan je u idućoj tablici.

Tip	Metoda i parametri	Opis metode
public Task<Void>	flushLocations()	Dostavlja prikupljenu lokaciju svim dostupnim klasama *LocationListener*, *LocationCallback* i *PendingIntent.* Ova metoda je samo korisna kada se definira vrijeme čekanja prije slanja podatka o lokaciji iz razloga što se svaka promjena lokacije dostavi u istom trenu promijene.
public Task<Location>	getLastLocation()	Metoda predstavlja najlakši način pristupa lokaciji. Vraća zadnju dostupu lokaciju uređaja. U rijetkom slučaju da lokacija nije dostupna vraća *null.*
public Task<LocationAvailability>	getLocationAvailability()	Vraća dostupnost podataka lokacije. U slučaju da uređaj nije spojen na *Google Play services* i zahtjev istekne, vraća *null*.
public Task<Void>	removeLocationUpdates (LocationCallback callback)	Uklanja sve ažurirane lokacije za dani osluškivač lokacije (engl. *location result listener*). Parametar *callback*, označava povratni poziv po kojem se uklanja.

public Task<Void>	requestLocationUpdates (LocationRequest request, LocationCallback callback, Looper looper)	Metoda traži ažuriranje lokacije sa povratnim pozivom (engl. *callback*). Parametar *request* označava zahtjev lokacije za ažuriranje, parametar *callback* je povratni poziv za ažuriranje lokacije, dok *looper* označava poziv za specifičnu dretvu, najčešće je null, što označava pozivajući dretvu.

Tablica 5. Metode klase FusedLocationProviderClient

5.4.2.2. LocationCallback

Javna apstraktna klasa *LocationCallback* (engl. *public abstract class LocationCallback*) koristi se za pridobivanje obavijesti od FusedLocationProviderApi sučelja svaki put kada se lokacija uređaja promijeni ili se ne može ustanovit (*LocationCallback*, 2017). U tablici 6. dan je popis metoda klase i njihov opis.

Tip	Metoda i parametri	Opis metode
public void	onLocationAvailability(LocationAvailability locationAvailability)	Metoda se poziva kada dođe do promjene u dostupnosti podataka o lokaciji (*LocationCallback*, 2017). Parametar predstavlja status dostupnosti lokacije uređaja.
public void	onLocationResult(LocationResult result)	Poziva se svaki put kada su podaci o lokaciji uređaja dostupni (*LocationCallback*, 2017). Parametar result predstavlja zadnje dostupni rezultat lokacije.

Tablica 6. Metode klase LocationCallback

5.4.2.3. LocationRequest

Javna finalna klasa LocationRequest (engl. *public final class LocationRequest*) služi stvaranju objekta za podešavanje odnosa kvalitete i kvantitete podataka o lokaciji. Kvaliteta se podešava na način da se konstantama ove klase postavi razina preciznosti podataka o lokaciji, dok se kvantiteta podataka postavi zadavanjem intervala ažuriranja podataka. Prema navedenom ova metoda sadrži sljedeće konstante prikazane u tablici 7 (*LocationRequest*, 2019).

Tip	Naziv konstante	Opis
public int	PRIORITY_BALANCED_POWER_ACCURACY	Zahtijevanje lokacije uz uravnoteženu potrošnju.
public int	PRIORITY_HIGH_ACCURACY	Zahtijevanje najpreciznije lokacije bez obzira na potrošnju.
public int	PRIORITY_LOW_POWER	Zahtijevanje lokacije uz nisku potrošnju.
public int	PRIORITY_NO_POWER	Zahtijevanje lokacije uz najnižu moguću potrošnju.

Tablica 7. Konstante klase LocationRequest

Osim opisanih konstanti, ova klasa sadrži među ostalima i metode prikazane u sljedećoj tablici.

Tip	Metoda i parametri	Opis
LocationRequest	setExpirationDuration(long millis)	Metoda postavlja vrijeme trajanja zahtjeva. Po isteku vremena servis prekida ažuriranje lokacije. Parametar *millis* označava vrijeme trajanja u milisekundama.

LocationRequest	setFastestInterval(long millis)	Postavlja najbrži mogući interval ažuriranja lokacije. Prema parametru *millis*, postavlja se interval ažuriranja u milisekundama
LocationRequest	setInterval(long millis)	Metoda omogućuje postavljanje intervala ažuriranja lokacije kojeg će se aplikacija pokušat pridržavati. Postavimo li parametar *millis* visokim, aplikacija će rjeđe ažurirati lokaciju, te potrošnja biti u skladu s time niža. Postavimo li navedeni parametar niskim vrijedi obrnuto.
LocationRequest	setNumUpdates(int numUpdates)	Metoda omogućuje postavljanje broja ažuriranja lokacije prije isteka zahtjeva. Primjer korištenja je kada aplikacija treba samo jednom ažurnu lokaciju(*LocationRequest*, 2019). Parametar *numUpdates* označava broj ažuriranja.
LocationRequest	setPriority(int priority)	Metoda postavlja prioritet konstante zahtjeva. Prioritet predstavlja veliki *savjet* aplikaciji koju izvor koristiti. Kod velike preciznosti izvor će najvjerojatnije biti GPS tehnologija, dok će kod bilancirane preciznosti i potrošnje to biti Cell ID. Parametar priority predstavlja konstantu.

Tablica 8. Metode klase LocationRequest

Potrebno je napomenuti da sve ove metode vraćaju isti objekt kako bi se metode mogle ulančati. Uz navedene metode za postavljanje parametara zahtjeva (engl. *set methods*) postoje i metode za dobivanje trenutnih vrijednosti zahtjeva (engl. *get methods*).

5.4.2.4. LocationServices

Javna klasa *LocationServices* (engl. *public class LocationServices*) predstavlja glavnu točku integracije lokacijskih servisa (*LocationServices*, 2019). Neke od javnih metoda ove klase, dane su uz opis u tablici 9.

Tip	Metoda i parametri	Opis
public static FusedLocationProviderClient	getFusedLocationProviderClient (Activity activity)	Stvaranje nove instance klase FusedLocationProviderClient za korištenje u aktivnosti (*LocationServices*, 2019).
public static FusedLocationProviderClient	getFusedLocationProviderClient (Context context)	Stvaranje nove instance klase FusedLocationProviderClient za korištenje u *context*-u neaktivnosti (*LocationServices*, 2019).
public static SettingsClient	getSettingsClient (Context context)	Stvaranje nove instance klase SettingsClient za korištenje u context-u neaktivnosti (*LocationServices*, 2019).
public static SettingsClient	getSettingsClient (Activity activity)	Stvaranje nove instance klase SettingsClient za korištenje u aktivnosti (*LocationServices*, 2019).

Tablica 9. Metode klase LocationServices

6. Implementacija aplikacije Moja Vozila

Kako bismo predočili korist lokacijskih servisa i svega navedenog, u ovome ćemo poglavlju prikazati jednostavnu primjenu lokacijskih servisa pri izradi lokacijske usluge. Kako bismo to učinili, pomoću razvojnog okruženja Android Studio i programskog jezika Kotlin, stvorili smo aplikaciju za praćenje više uređaja istovremeno. Aplikacija je kreirana u svrhu praćenja vozača i njihovih vozila, stoga joj je naziv *Moja Vozila*. Funkcionalni zahtjevi opisane aplikacije dani su u tablici ispod.

Oznaka	Naziv	Kratak opis
F01	Učitavanje aplikacije	Prilikom pokretanja aplikacije korisniku se prikazuje zaslon učitavanja. Sam zaslon se samostalno nakon nekoliko trenutaka ukloni, te se prelazi na iduću funkcionalnost.
F02	Prijava u aplikaciju	Prijava u aplikaciju omogućuje korisniku unos vlastitog imena i prezimena, koje se prikazuje drugim korisnicima. Prilikom same prijave, uneseni podaci se pohranjuju u bazu podataka.
F03	Pregled lokacija	Pregled lokacija je funkcionalnost koja se otvara nakon prijave u aplikaciju. Ova funkcionalnost omogućuje prikaz lokacija svih prijavljenih korisnika na virtualnoj karti svijeta.
F04	Odjava iz aplikacije	Ova funkcionalnost omogućuje zatvaranje aplikacije i brisanje prijavljenog korisnika iz baze podataka.

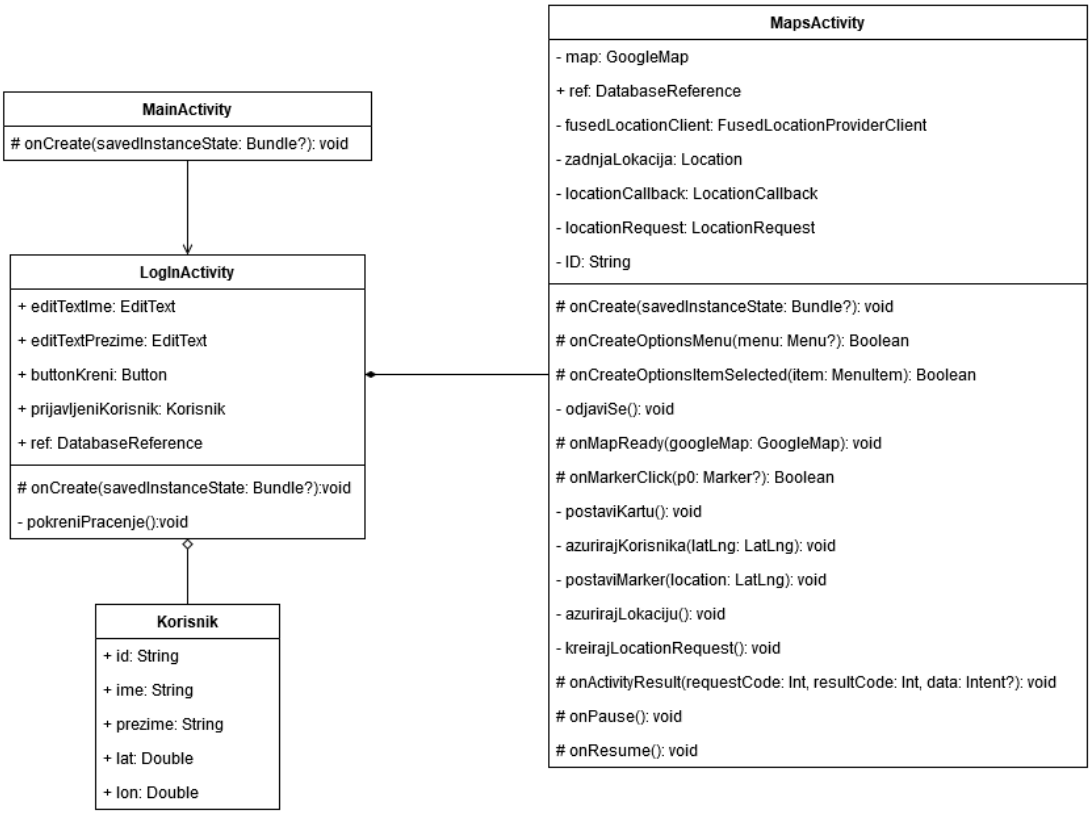
Tablica 10. Tablica funkcionalnih zahtjeva aplikacije

Na slici 17. uočavamo skice izgleda ekrana za svaku od navedenih funkcionalnosti i redoslijed njihovih prikazivanja.

Slika 17. Skice ekrana za pojedinu funkcionalnost

Aplikacija je jednostavnog izgleda, a prvo što se uočava prilikom pokretanja je zaslon učitavanja. Po završetku prikaza zaslona učitavanja, korisniku se omogućuje prijava bez registracije. Prijava služi isključivo svrsi prikupljanja podataka o imenu i prezimenu prijavljenog korisnika. Kada se korisnik prijavi, otvara mu se aktivnost koja prikazuje kartu svijeta, njegovu lokaciju, te lokaciju svih ostalih prijavljenih korisnika. Kako bi podatke o lokaciji uređaji mogli međusobno dijelit, koristi se Google platforma Firebase. Firebase omogućuje korištenje baze podataka u stvarnom vremenu (engl. real time database) uz pomoć tzv. *Realtime Database* modula.

Nabrojani dijelovi aplikacije poput zaslona učitavanja i zaslona prijave nazivamo aktivnostima. U užem smislu, aktivnost predstavlja korisničko sučelje koje korisnik vidi. Za oblikovanje aktivnosti aplikacije koriste se već spomenute xml datoteke, dok se funkcionalnost dodaje pomoću Kotlin datoteke. Dijagram klasa sa slike prikazuje sve Kotlin datoteke, njihove varijable, metode i odnos.



Slika 18. Dijagram klasa opisane aplikacije

Kako bismo prikazali sve navedene dijelove aplikacije, u nastavku ćemo prikazati pojedine aktivnosti, odnosno njihove implementacije izgleda i funkcionalnosti.

6.1. Zaslon učitavanja

Zaslon učitavanja, u stvarnom smislu ne predstavlja zaslon učitavanja, već kratki prikaz loga aplikacije. Iluzija učitavanja stvara se vremenskim ograničavanjem trajanja prikaza početnog zaslona. Izgled početnog zaslona prikazan je na slici 19.

Slika 19. Izgled zaslona učitavanja aplikacije

Sadržaj xml datoteke activity_main.xml omogućuje stvaranje izgleda sa slike 19. Sam sadržaj, odnosno programski kôd dan je ispod.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EDEDED"
    tools:context=".MainActivity">

    <ImageView
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_gravity="center"
        android:src="@drawable/logo"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.433" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Isječak kôda 1. Programski kod za izgled zaslona učitavanja

Programski kôd dan iznad služi postavljanju širine i visine prikazane aktivnosti, prema širini i visini zaslona uređaja. Navedeno se postiže pomoću atributa *layout_width* i *layout_height*, te vrijednosti *match_parent.* Sam logo dodaje se uz pomoć elementa *ImageView*, na način da mu se definira atribut *src*, koji predstavlja izvor slike. Ostali atributi služe pozicioniranju elementa te podešavanju njegove širine i visine.

Iluzija učitavanja postiže se u Kotlin datoteci *MainActivity.kt*. Sadržaj navedene datoteke prikazan je ispod.

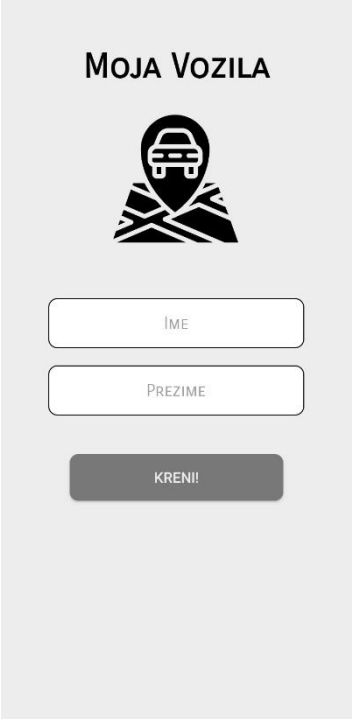
```
class MainActivity : Activity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        requestWindowFeature(Window.FEATURE_NO_TITLE)
        window.setFlags(FLAG_FULLSCREEN, FLAG_FULLSCREEN)
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val background = object : Thread(){
            override fun run() {
                try {
                    Thread.sleep(5000)
                    val intent = Intent(baseContext, LogInActivity::class.java)
                    startActivity(intent)
                }catch (e: Exception){
                    e.printStackTrace()
                }
            }
        }
        background.start()
    }
}
```

Isječak kôda 2. Programski kôd za prikaz ekrana učitavanja

Programski kôd prikazuje klasu *MainActivity* koja u sebi sadrži prijepis (engl. override) metode *onCreate*. Navedena metoda omogućuje definiranje što će se desit kada se klasa kreira. U našem slučaju će se pomoću metoda *requestWindowFeature* i *setFlags* početni zaslon postavit na puni prikaz, te će se sa početnog zaslona ukloniti akcijska traka (engl. *action bar*). Unutar *onCreate* metode pronalazimo još jedan prijepis metode. Radi se o prijepisu metode *run*. Navedena metoda nastoji unutar *try* bloka, uz pomoć funkcije *sleep* s parametrom od 5000 milisekundi, zaustavit izvođenje aplikacije na 5 sekundi. Po isteku vremena unutar istog *try* bloka kreira se novi *intent*. Intent predstavlja stvaranje nove aktivnosti, a sama aktivnost pokreće se uz pomoć funkcije *startActivity* uz navođenje parametra intenta stvorenog prije samog poziva. U našem slučaju intent predstavlja stvaranje nove aktivnosti za prijavu.

6.2. Zaslon prijave

Zaslon prijave aplikacije Moja Vozila ne predstavlja stvarnu prijavu u sustav sa autentifikacijom, nego stvaranje novog zapisa unutar baze podataka, koji sadrži podatke o imenu i prezimenu prijavljenog korisnika i njegove koordinate. Izgled zaslona prikazan je na slici 20.



Slika 20. Izgled zaslona prijave aplikacije

Izgled zaslona definiran je u xml datoteci *activity_log_in.xml*. Sadržaj datoteke prikazan je ispod, a u nastavku su opisani značajniji elementi, atributi i parametri.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EDEDED"
    tools:context=".LogInActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="25dp"
        android:orientation="vertical"
        android:layout_gravity="center">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="45dp"
            android:layout_gravity="center"
            android:fontFamily="sans-serif-smallcaps"
            android:text="Moja Vozila"
            android:textAlignment="center"
            android:textColor="#000000"
            android:textSize="40dp"
            android:textStyle="bold"
            android:layout_marginTop="30dp"/>

        <ImageView
            android:layout_width="150dp"
            android:layout_height="150dp"
            android:src="@drawable/logo"
            android:layout_gravity="center"
            android:layout_marginTop="35dp"/>

        <EditText
            android:id="@+id/ime_id"
            android:layout_width="300dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="65dp"
            android:background="@drawable/edit"
            android:fontFamily="sans-serif-smallcaps"
            android:hint="Ime"
            android:padding="17dp"
            android:textAlignment="center"
            android:maxLines="1"
            android:maxLength="20"/>
```

```
<EditText
            android:id="@+id/prezime_id"
            android:layout_width="300dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="20dp"
            android:background="@drawable/edit"
            android:fontFamily="sans-serif-smallcaps"
            android:hint="Prezime"
            android:padding="17dp"
            android:textAlignment="center"
            android:maxLines="1"
            android:maxLength="20"/>

        <Button
            android:id="@+id/buttonKreni"
            android:layout_width="250dp"
            android:layout_height="55dp"
            android:layout_gravity="center"
            android:layout_marginTop="45dp"
            android:background="@drawable/button"
            android:text="Kreni!"
            android:textColor="#ededed"
            android:textSize="17dp" />
    </LinearLayout>
</LinearLayout>
```

Isječak kôda 3. Programski kôd za ostvarenje izgleda zaslona prijave

Sam zaslon sastoji se imena aplikacije, slike loga, kućišta za unos imena i prezimena, te gumba za prijavu.

Ime aplikacije prikazano je uz pomoć elementa *TextView*. Sam element je vrlo jednostavan, bitna značajka je samo da je u svrhu dizajna, pomoću atributa *fontFamily* doda izgled slova (font).Logo je kao i u prethodnom slučaju prikazan i podešen uz pomoć elementa *ImageView.*

U izgled zaslona dodani su elementi za unos tekstualnog sadržaja. Izgled navedenih elemenata kreiran je uz pomoć nove xml datoteke. U novoizrađenoj datoteci definirani su rubovi i zaobljenost rubova. Kako bi se izmjene prikazale unutar elementa *EditText,* uz pomoć *background* atributa pozivamo novokreiranu xml datoteku. Broj redova i najveći dopušten broj slova definiran je atributima *maxLines* i *maxLength*.

Pomoću elementa *Button* dodan je gumb na početni zaslon. Na isti način kao i kod polja za unos podataka definiran je izgled gumba. Uz navedeno, također su postavljene dimenzije i boja gumba.

Klikom na gumb, pokreće se funkcionalnost prijave. Prijava se ostvaruje putem LogInActivity.kt datoteke čiji je sadržaj prikazan u nastavku.

```
class LogInActivity : Activity() {

    lateinit var editTextIme: EditText
    lateinit var editTextPrezime: EditText
    lateinit var  buttonKreni: Button
    lateinit var prijavljeniKorisnik: Korisnik
    lateinit var ref: DatabaseReference

    override fun onCreate(savedInstanceState: Bundle?) {
        requestWindowFeature(Window.FEATURE_NO_TITLE)
        window.setFlags(
            WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN
        )
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_log_in)

        ref = FirebaseDatabase.getInstance().getReference("korisnici")

        editTextIme = findViewById(R.id.ime_id)
        editTextPrezime = findViewById(R.id.prezime_id)
        buttonKreni = findViewById(R.id.buttonKreni)

            buttonKreni.setOnClickListener{
                pokreniPracenje()
                val intent = Intent(this, MapsActivity::class.java)
                intent.putExtra("id", prijavljeniKorisnik.id)
                editTextIme.text.clear()
                editTextPrezime.text.clear()
                startActivity(intent)

        }
    }

    private fun pokreniPracenje(){
        val ime = editTextIme.text.toString().capitalize()
        val prezime = editTextPrezime.text.toString().capitalize()

        if(ime.isEmpty()){
            editTextIme.error = "Unesi ime!"
            return
        }
        val korisnikId = ref.push().key!!

        prijavljeniKorisnik = Korisnik(korisnikId, ime, prezime, 0.0, 0.0)

ref.child(korisnikId).setValue(prijavljeniKorisnik).addOnCompleteListener{
        }
    }
}
```

Isječak kôda 4. Programski kôd za prijavu korisnika u aplikaciju

Na samom početku kôda pronalazimo varijable kasne inicijalizacije. Kasna inicijalizacija postiže se uz pomoć ključne riječi *lateinit*. Među navedenim varijablama pronalazimo elemente izgleda aplikacije *EditText* i *Button*, ali i objekt *prijavljeniKorisnik* klase *Korisnik*, te referencu baze podataka. Unutar prijepisa *onCreate* metode pronalazimo varijablu *ref* uz pomoć koje dohvaćamo putanju prema bazi podataka sa *tablicom* korisnici. Navedeni atribut nam omogućuje pisanje i dohvaćanje zapisa iz baze podataka.

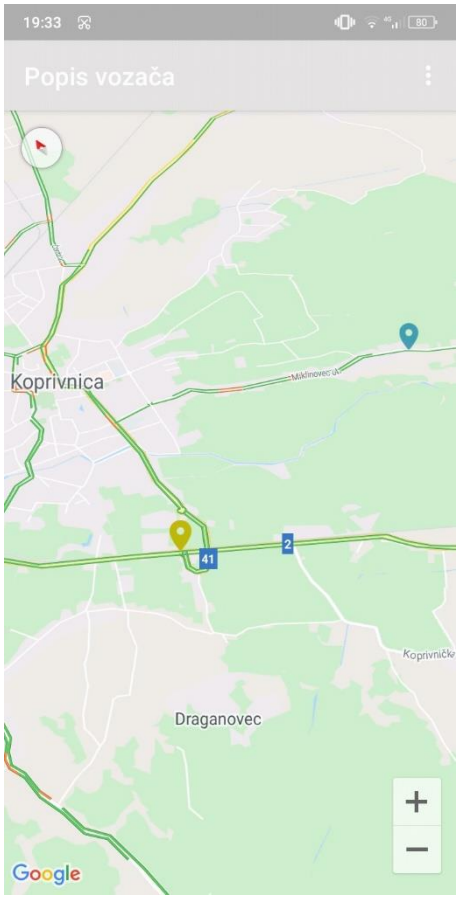
U nastavku su elementi za unos podataka i gumb *pronađeni* uz pomoć metode *findViewById*. Navedena metoda nam omogućuje da u nastavku dohvatimo podatke iz pojedinih elemenata i slušamo događaje koje se nad njima događaju. Tako se nadalje postavlja osluškivač događaja (engl. *listener*) *setOnClickListener* nad elementom gumba. Svaki put kada se klikne na gumb, navedena metoda se pokreće na način da izvrši privatnu funkciju *pokreniPracenje*.

Navedena funkcija ima zadatak dohvatiti ime i prezime iz elementa za unos i dodijeliti im veliko početno slovo. Potom se kreiranom *Korisniku* dodjeljuju generirani element id, ime, prezime i početne koordinate, te se korisnik zapisuje uz pomoć metode setValue i reference u bazu podataka.

Po izvršavanju funkcije, pokreće se nova aktivnost uz pomoć novog *intent* objekta. Nova aktivnost u ovom slučaju predstavlja kartu svijeta sa naznačenim lokacijama korisnika, odnosno *MapsActivity.kt*.

6.3. Zaslon karte

Zaslon karte predstavlja prikaz virtualne karte na mobilnom uređaju. Sama karta prikazuje markerima vlastitu lokaciju i lokacije drugih korisnika kao na slici 21. Vlastita pozicija prikazana je markerom žute boje, dok su lokacije drugih korisnika prikazane plavom bojom.



Slika 21. Prikaz zaslona karte

Datoteka *activity_maps.xml* zadužena je za generiranje elementa karte. Navedenu datoteku smo prikazali i objasnili u prethodnom poglavlju, stoga ćemo se u nastavku usmjerit na Kotlin datoteku ovog zaslona, *MapsActivity.kt*. Navedena datoteka je nešto složenija od prijašnjih Kotlin datoteka, stoga će u nastavku bit prikazani bitniji dijelovi kôda uz objašnjenje. Najprije ćemo pokazati zaglavlje klase *MapsActivity.*

```
private lateinit var map: GoogleMap
lateinit var ref: DatabaseReference
private lateinit var fusedLocationClient: FusedLocationProviderClient
private lateinit var zadnjaLokacija: Location
private lateinit var locationCallback: LocationCallback
private lateinit var locationRequest: LocationRequest
private lateinit var ID: String
lateinit var listaKorisnika: MutableList<Korisnik>

private var locationUpdateState = false

companion object {
    private const val LOCATION_PERMISSION_REQUEST_CODE = 1
    private  const val REQUEST_CHECK_SETTINGS = 2
}
```

Isječak kôda 5. Programski kôd zaglavlja aktivnosti MapsActivity.kt

Raspisani kôd predstavlja sve varijable sa već spomenutom kasnom inicijalizacijom. Većina klasa je objašnjena u prethodnim poglavljima. Novi pojmovi koji bi u ovom dijelu programa mogli bit nejasni su *MutableList* i *companion object*. Uz pomoć *MutableList* kreiramo generičku listu tipa *Korisnik*. Ovo je lista sa mogućnošću dodavanja novih elemenata. Kasnije u listu *listaKorisnika* pohranjujemo sve dohvaćene korisnike iz baze podataka kako bismo ih korisniku mogli pokazat. *Companion object* predstavlja objekt čijim se članovima može pristupiti direktno putem imena sa istom sintaksom kao i kod poziva statične metode. Nakon definiranih varijabli i konstanti slijedi prijepis *onCreate* metode.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_maps)

    val mapFragment = supportFragmentManager
        .findFragmentById(R.id.map) as SupportMapFragment
    mapFragment.getMapAsync(this)

    listaKorisnika = mutableListOf()
    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)

    val intent = intent
    val korisnikId = intent.getStringExtra("id")
    ID = korisnikId
    ref = FirebaseDatabase.getInstance().getReference("korisnici")

    ref.addValueEventListener(object :ValueEventListener {
        override fun onCancelled(error: DatabaseError) {
            TODO("Not yet implemented")
        }

        override fun onDataChange(snapshot: DataSnapshot) {
            if(snapshot!!.exists()){
                for(k in snapshot.children){
                    val korisnik = k.getValue(Korisnik::class.java)
                    if(korisnik!!.id != ID) {
                        val location = LatLng(korisnik.lat, korisnik.lon)
                        val markerOptions =
                            MarkerOptions().position(location)
                                .title(korisnik.ime.toString() + " " +
korisnik.prezime.toString())
                        markerOptions.icon(
                            BitmapDescriptorFactory.fromBitmap(
                                BitmapFactory.decodeResource(resources,
R.drawable.ic_lokacija)
                            )
                        )
                        map.addMarker(markerOptions)
                    }
                }
            }
        }
    })

    locationCallback = object : LocationCallback() {
        override fun onLocationResult(p0: LocationResult) {
            super.onLocationResult(p0)
            zadnjaLokacija = p0.lastLocation
            val azuriranaLokacija =
LatLng(zadnjaLokacija.latitude,zadnjaLokacija.longitude)
            azurirajKorisnika(azuriranaLokacija)
            postaviMarker(LatLng(zadnjaLokacija.latitude,
zadnjaLokacija.longitude))
        }
    }
    kreirajLocationRequest()
}
```

Isječak kôda 6. Programski kôd onCreate metode aktivnosti MapsActivity.kt

Na početku ove metode pronalazimo već spomenutu inicijalizaciju karte. Potom, slijedi inicijalizacija prazne liste korisnika i *fusedLocationCLient* objekta. Nakon navedene inicijalizacije iz *intent* objekta dobivamo proslijeđeni primarni ključ korisnika proslijeđenog u prijave korisnika, a zatim se stvara referenca na tablicu korisnika iz baze podataka. Za danu referencu primjenjuje se metoda *addValueEventListener* u kojoj se definira osluškivač događaja *onDataChanged*. Uz pomoć navedene metode definiramo što se treba desit kad se sadržaj baze podataka promijeni (ako se korisnik prijavi ili odjavi). U ovom slučaju za svakog korisnika zapisanog u bazi podataka kreiramo objekt *MarkerOptions*, koji na karti predstavlja marker. Navedenom markeru definiramo s podacima iz baze podataka lokaciju i naslov. Uz to metodom *icon*, dodajemo vlastiti izgled markera. Nakon kreiranja markera, uz pomoć metode *addMarker* kreirani marker dodajemo objektu *map*, koji označava kartu. Nakon dodavanja markera za druge korisnike, *locationCallback* objektom dodajemo vlastitu lokaciju. Najprije dohvaćamo zadnje poznatu lokaciju uređaja, a zatim ju prosljeđujemo privatnim funkcijama *azurirajKorisnika* i *postaviMarker*.

```
    private fun azurirajKorisnika(latLng: LatLng) {
        ref.child(ID).child("lat").setValue(latLng.latitude)
        ref.child(ID).child("lon").setValue(latLng.longitude)
    }
```

Isječak kôda 7. Programski kôd funkcije za ažuriranje korisnika

Funkcija azurirajKorisnika koristi se isključivo u svrhu ažuriranja lokacije korisnika u bazi podataka. Parametri ove funkcije su nove koordinate. Uočavamo da se referenci dodaje *child* sa primarnim ključem ID, koji je proslijeđen iz prijave. Za dobivenu referencu upisuju se latitude i longitude, odnosno koordinate.

```
private fun postaviMarker(location: LatLng){
    map.clear()
    val markerOptions = MarkerOptions().position(location)

    markerOptions.icon(
        BitmapDescriptorFactory.fromBitmap(
        BitmapFactory.decodeResource(resources, R.drawable.ic_lokacija_moja)))
    map.addMarker(markerOptions)
}
```

Isječak kôda 8. Programski kôd za postavljanje markera na kartu

Svaki put kada se pozove funkcija postaviMarker, čisti se sadržaj karte uz pomoć funkcije clear. Kreira se novi marker sa proslijeđenom lokacijom korisnika, te se dodaje kartu.

Opisano programsko rješenje predstavlja jednostavan način rješavanja zadanog problema. Dubljim proučavanjem tematike, dolazi se do većeg broja funkcionalnosti koji pruža Google Maps API. Radi se funkcionalnostima poput onih koji pružaju optimalniji način pomicanja markera na virtualnoj karti, te optimalniji odnos performansi aplikacije i utroška baterije uređaja. Stoga, u ovom poglavlju opisani i prikazani primjer ne predstavlja optimalnu aplikaciju nego samo temelj za razvoj složenijih aplikacija.

7. Zaključak

U izrađenom završnom radu analizirane su i sistematizirane mogućnosti primjene geolokacijskih servisa. Rad se temelji na opisu geolokacijskih tehnologija i načinu dobivanja informacija o korisnikovoj lokaciji. Iz navedenog temelja proizlaze mogućnosti njihove primjene. Navedenim i opisanim mogućnostima, uspostavlja se neizmjerna korisnost lokacijskih servisa u današnjem svijetu. Radi se o tehnologiji koja gotovo u doslovnom smislu riječi omogućuje držanje svijeta u jednoj ruci. Samu primjenu pritom ne pronalazimo samo kao pomoć korisnicima, već i kao zabavu, te edukaciju. Uz sami opis i usporedbu drugih, prikazan je i tehnički opis najkorištenijeg geolokacijskog servisa. Iz tehničkog opisa iščitava se način primjene geolokacijskih servisa za ostvarenje nabrojanih i opisanih mogućnosti primjena. Po obradi tehničkog opisa, izrađena je aplikacija za praćenje više prijavljenih korisnika. Implementacija aplikacije ostvarena je programskim jezikom Kotlin u razvojnom okruženju Android Studio. Izgled aplikacije definiran je xml jezikom. Testiranje aplikacije odvijalo se na stvarnom uređaju radi pouzdanije provjere pribavljanja podataka o lokaciji. Potrebno je napomenuti da izrađena aplikacije nikako ne predstavlja savršen opis uporabe kartografskih i lokacijskih servisa, te da postoji velik broj mogućnosti optimizacije i nadogradnje aplikacije.

Uspostavlja se zaključak da geolokacijske tehnologije i servisi, uz neizbježno kvalitetan model zaštite privatnosti, predstavljaju sastavnu gradivnu jedinicu trenutnih ali i budućih aplikacija dostupnih na tržištu mobilnih usluga.

.

Popis literature

AgaRwal, S. (2018). *Have you heard about Open Street Map?* https://medium.com/mindorks/have-you-heard-about-open-street-map-d6c51dc00bea

Arterburn, R. (2020, February 18). *Location-Based Services Market Insights with Latest Statistics and Growth Prediction to 2026*. https://fresnobserver.com/location-based-services-market-insights-with-latest-statistics-and-growth-prediction-to-2026/3771/

Bao-yen Tsui, J. (2000). *Fundamentals of Global Positioning System Receivers: A Software Approach*. JOHN WILEY & SONS, INC. http://twanclik.free.fr/electricity/electronic/pdfdone7/Fundamentals%20of%20Global%20Positioning%20System%20Receivers.pdf

Barradas Pereira, R. (2011). *Location Based Services*. Navipedia. https://gssc.esa.int/navipedia/index.php/Location_Based_Services

Com.google.android.gms.location. (2019, October 1). Google APIs for Android. https://developers.google.com/android/reference/com/google/android/gms/location/package-summary

Control segment. (2018, August 11). GPS.Org. https://www.gps.gov/systems/gps/control/

Društvena mreža. (n.d.). [Enciklopedija]. Struna. Retrieved September 8, 2020, from http://struna.ihjj.hr/naziv/drustvena-mreza/25409/

Fakultet organizacije i informatike. (2020). [Map]. https://www.google.com/maps/place/46%C2%B018'28.4%22N+16%C2%B020

'17.3%22E/@46.3077288,16.328747,15z/data=!4m5!3m4!1s0x0:0x0!8m2!3d46.3078889!4d16.3381389

Huang, H., Gartner, G., M. Krisp, J., Raubal, M., & Van de Weghe, N. (2018). *Jurnal of Location Based Services*. https://www.tandfonline.com/doi/pdf/10.1080/17489725.2018.1508763?needAccess=true

Lehmann, L. (2012). *Location-based Mobile Games*. https://pdfs.semanticscholar.org/aed4/28f1c3d70edcc9e990fde5af60f6178b9656.pdf?_ga=2.139741043.1704286338.1596975913-102208813.1596877198

Location-Based Services (LBS) and Real-Time Location Systems (RTLS) Market. (2020, June).

LocationCallback. (2017, August 14). Google APIs for Android. https://developers.google.com/android/reference/com/google/android/gms/location/LocationCallback

LocationRequest. (2019, January 10). Google APIs for Android. https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest#constants

LocationServices. (2019, January 10). Google APIs for Android. https://developers.google.com/android/reference/com/google/android/gms/location/LocationServices#public-methods

Mallick, M. (2003a). *Angle of Arrival*. ETutorials.Org. http://etutorials.org/Mobile+devices/mobile+location+services/Part+2+The+Mobile+Location+Server/Chapter+5.+Mobile+Positioning/Angle+of+Arrival/

Mallick, M. (2003b). *Mobile Positioning Techniques*. ETutorials.Org. http://etutorials.org/Mobile+devices/mobile+wireless+design/Part+Four+Beyon

d+Enterprise+Data/Chapter+17+Location-Based+Services/Mobile+Positioning+Techniques/

Parmar, D., & Pratap Rao, U. (2020). Towards Privacy-Preserving Dummy Generation in Location-Based Services. *Elsevier B.V.* https://www.sciencedirect.com/science/article/pii/S1877050920311200

Rao, B., & Minakakis, L. (2003). *Evolution of Mobile Location-based Services*. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=95E20E5B7B4B74E63E8C0286CF5192BF?doi=10.1.1.471.193&rep=rep1&type=pdf

Space segment. (2020, June 30). GPS.Org. https://www.gps.gov/systems/gps/space/

Steinfield, C. (2004). *The Development of Location Based Services in Mobile Commerce*. https://msu.edu/~steinfie/elifelbschap.pdf

Tarasenko, A. (2019, April 17). *OpenStreetMap is more than just a map*. Geoapify. https://www.geoapify.com/openstreetmap-is-more-than-just-a-map/

Triangulacija. (2020, June 9). Enciklopedija.Hr. https://www.enciklopedija.hr/natuknica.aspx?ID=62229

Web Mapping products. (2020). Enlygt.c. https://enlyft.com/tech/web-mapping

Wi-Fi Location-Based Services. (n.d.). Cisco.Com. Retrieved June 30, 2020, from https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Mobility/WiFiLBS-DG/wifich2.html

Wikipedia suradnici. (2020a). *Comparison of web map services* [Enciklopedija]. Wikipedia. https://en.wikipedia.org/wiki/Comparison_of_web_map_services

Wikipedia suradnici. (2020b). *Geocaching*. Wikipedia. https://en.wikipedia.org/wiki/Geocaching

Wikipedia suradnici. (2020c). *Runkeeper* [Enciklopedija]. Wikipedia. https://en.wikipedia.org/wiki/Runkeeper

Wikipedia suradnici. (2020d, April 21). *Foursquare City Guide* [Enciklopedija]. Wikipedia. https://en.wikipedia.org/wiki/Foursquare_City_Guide

Zheng, Y., & Xiaofang, Z. (2011). Location-Based Social Networks: Users. In *Computing with Spatial Trajectories*. Springer. https://www.tandfonline.com/doi/pdf/10.1080/17489725.2018.1508763?needAccess=true

Popis slika

Slika 1. Geografska mreža(Janko, bez datuma) 3
Slika 2. Međusobni položaj satelita svemirske komponente (Space segment, 2020)..... 5
Slika 3. Položaji i funkcije stanica kontrolne komponente (Control segment, 2020) 6
Slika 4. Grafički prikaz tehnologije stanice podrijetla (Wi-Fi Location-Based Services, 2020) 8
Slika 5. Grafički prikaz rada AOA tehnologije (Angle Of Arrival, 2003)..... 9
Slika 6. Grafički prikaz uzoraka igara, po uzoru na (Lehmann, 2012) 15
Slika 7. Sučelje aplikacije Runkeeper 18
Slika 8. Primjer Googleove Maps usluge za informacije u prometu uživo 19
Slika 9. Primjer izgleda aplikacije mobileMAP 20
Slika 10. Odabir predloška novog projekta 27
Slika 11. Sadržaj MapsActivity.kt datoteke 28
Slika 12. Sadržaj activity_maps.xml datoteke 29
Slika 13. Sadržaj google_maps_api.xml datoteke.....30
Slika 14. Ovisnost potrebna za prikaz Google-ove karte.....30
Slika 15. Izgled sučelja pokrenute aplikacije.....31
Slika 16. Ovisnost potrebna za korištenje lokacije mobilnog uređaja 32
Slika 17. Skice ekrana za pojedinu funkcionalnost 39
Slika 18. Dijagram klasa opisane aplikacije 40
Slika 19. Izgled zaslona učitavanja aplikacije 41
Slika 20. Izgled zaslona prijave aplikacije 44
Slika 21. Prikaz zaslona karte.....49

Popis tablica

Tablica 1. Pregled nekoliko lokacijski temeljenih igrica, njihovih uzoraka i namjena (Lehmann, 2012) 17
Tablica 2. Usporedba značajnijih značajki servisa 25
Tablica 3. Udio servisa na tržištu 26
Tablica 4. Metoda sučelja LocationListener 32
Tablica 5. Metode klase FusedLocationProviderClient 34
Tablica 6. Metode klase LocationCallback 34
Tablica 7. Konstante klase LocationRequest 35
Tablica 8. Metode klase LocationRequest 36
Tablica 9. Metode klase LocationServices 37
Tablica 10. Tablica funkcionalnih zahtjeva aplikacije 38

Popis isječaka kôda

Isječak kôda 1. Programski kod za izgled zaslona učitavanja 42
Isječak kôda 2. Programski kôd za prikaz ekrana učitavanja 43
Isječak kôda 3. Programski kôd za ostvarenje izgleda zaslona prijave 46
Isječak kôda 4. Programski kôd za prijavu korisnika u aplikaciju 47
Isječak kôda 5. Programski kôd zaglavlja aktivnosti MapsActivity.kt 50
Isječak kôda 6. Programski kôd onCreate metode aktivnosti MapsActivity.kt 51
Isječak kôda 7. Programski kôd funkcije za ažuriranje korisnika 52
Isječak kôda 8. Programski kôd za postavljanje markera na kartu 52

Prilozi

1. Izvorni kôd dostupan na:

   https://drive.google.com/drive/folders/15iKQ9PwpsEyRxqE2scJyk1ZHdA1zDuwb?usp=sharing