Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Koželj

Aplikacija za protinaletni sistem v avtomobilu

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Mojca Ciglarič

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Izdelajte protinaletni sistem, ki bo voznika opozarjal pred morebitnimi ovirami ali zgostitvami na cesti. Najprej preučite protokole, ki jih za komunikacijo znotraj vozila uporabljajo današnji avtomobili. Raziščite, kakšne varnostne sisteme proizvajalci že vgrajujejo v vozila in identificirajte njihove pomanjkljivosti. Predlagajte lasten sistem, ki bo rešil nekatere izmed identificiranih težav. Izberite ustrezne tehnologije in orodja, zlasti pa metode in postopke za določanje hitrosti, lokacije, orientacije vozila in podobno. Sistem izdelajte, preizkusite in kritično ovrednotite.

Zahvaljujem se mentorici doc. dr. Mojci Ciglarič za vso pomoč pri izdelavi diplomskega dela. Zahvaljujem se tudi staršema in dedku Stanku za vso pomoč tekom študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
XML	Extensible Markup Language	razširljiv označevalni jezik
API	Application Programming Interface	aplikacijski programski vmesnik
FCM	Firebase Cloud Messaging	oblačna sporočila Firebase
GCM	Google Cloud Messaging	oblačna sporočila Googlew
GPS	Global Positioning System	sistem globalnega pozicioniranja
UID	User identifier	identifikator uporabnika

Povzetek

Naslov: Aplikacija za protinaletni sistem v avtomobilu

Avtor: Jernej Koželj

Cilj diplomske naloge je bil razvoj mobilne aplikacije, ki bi uporabnika opozorila na nevarnost naleta na avtocesti. S tem bi lahko preprečili nesreče in povečali varnost na cestah. Aplikacijo bi uporabnik pred vožnjo prižgal, ta pa bi tekla v ozadju in oddajala oziroma prejemala opozorila o nevarnosti naleta ter jih posredovala uporabniku(vozniku). Razvoj je potekal v orodju Android Studio. V prvem delu diplome so predstavljene uporabljene tehnologije ter teoretična plat te diplomske naloge, v drugem delu pa bolj podrobni razvoj same aplikacije, njeno delovanje in povzetek celotnega dela.

Ključne besede: aplikacija, avto, računalnik.

Abstract

Title: Collision Avoidance System Car Application

Author: Jernej Koželj

The aim of the thesis was to develop a mobile application that alerts the user of the risk of collision on the highway. This could prevent accidents and increase safety on the roads. User would run the application before driving and it will run in the background. It will emit or receive warnings about the risk of collision and transmit them to the user (driver). The development was done in Android Studio. The first part of thesis presents the technology used and is more theoretical. The second part is detailed development of the application itself, its operation and summary of the whole work.

Keywords: application, car, computer.

Poglavje 1

Uvod

Ljudje vedno več časa preživimo na cestah, zato je neizbežno, da se dogajajo nesreče. V diplomskem delu smo se odločili razviti sistem, ki bi voznike opozoril na nevarnost naleta. Tako bi lahko pravočasno ukrepali in se izognili morebitni nezgodi. Obstaja veliko podobnih rešitev znanih proizvajalcev avtomobilov vendar te delujejo bodisi preko vizualnega zaznavanja ali pa s pomočjo radarja. Kar je drugače pri našem sistemu je to, da na nevarnost opozori veliko prej in lahko uporabnik bolje odreagira. Sistem bo zaenkrat deloval preko pametnih mobilih telefonov, kasneje pa se namerava razviti samostojen modul, ki bo vgrajen direktno v avtomobil.

Uporabnik mora pred vožnjo na avtocesti vklopiti aplikacijo, ta pa v ozadju teče in spremlja dogajanje. V primeru, da prejme signal, zvočno in vizualno opozori voznika brez da bi ga s tem ovirala pri vožnji. Če vozilo v katerem je telefon z aplikacijo, izpolnjuje vse pogoje, ki so potrebni, da generirajo obvestilo, se obvestilo pošlje. To obvestilo se pošlje vsem vozilom ki se vozijo v isto smer in so oddaljeni določeno število metrov za vozilom, ki je poslalo obvestilo. Da bi sistem deloval tako kot mora, je potrebno, da aplikacijo uporablja čim večje število ljudi.

V prvem delu diplomske naloge je predstavljena teoretična plat, vse uporabljene tehnologije za razvoj aplikacije in vsi podobni sistemi, ki so nas pripeljali do te ideje. V drugem delu pa je podrobno predstavljen razvoj aplikacije, njeno delovanje ter povzetek celotnega dela.

1.1 Cilji

V tem diplomskem delu bi radi predstavili razvoj protinaletnega sistema. Gre za sistem v avtomobilu, ki bi voznika pravočasno obvestil o zgostitvi prometa pred njim. To je problem, s katerim se največkrat srečujemo na avtocestah, predvsem v slabših vremenskih razmerah. Sistem bi deloval tako, da bi se pošiljali podatki o močnem zaviranju voznikov. Ta signal bi se poslal v okolico in s tem opozoril, da je pred voznikom nekaj kar bi lahko ogrozilo vožnjo.

1.2 Metodološki pristop

V diplomskem delu so najprej predstavljeni protokoli v sodobnih avtomobilih, nato smo predstavili varnostne sisteme najbolj naprednih vozil. Kasneje smo predstavili uporabljene tehnologije in ogrodja. Na koncu pa so predstavljeni uporabljeni pristopi in metode pri izdelavi aplikacije, podana je ocena ter kratek povzetek našega dela. Temu sledi zaključek in nadaljnje delo.

Poglavje 2

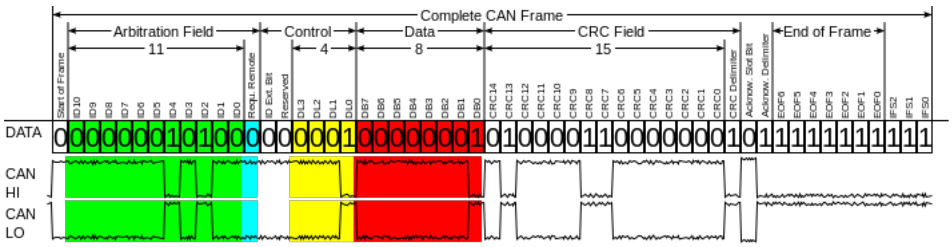
Teoretična predstavitev

2.1 Protokoli v avtomobilih

Današnji avtomobili so več kot le prevozno sredstvo, v njih so vgrajeni različni napredni sistemi in druge naprave, kar jih naredi velike vozeče se računalnike. Da bi vsi ti sistemi delovali pravilno, pa skrbijo spodaj našteti protokoli in vodila, ki so na kratko opisani. Avtomobil je postal kompleksen sistem naprav, ki sodelujejo in komunicirajo med sabo. To sodelovanje vseh naprav, pa je mogoče uporabiti za veliko uporabnih rešitev na področju varnosti v vozilih.

2.1.1 CAN

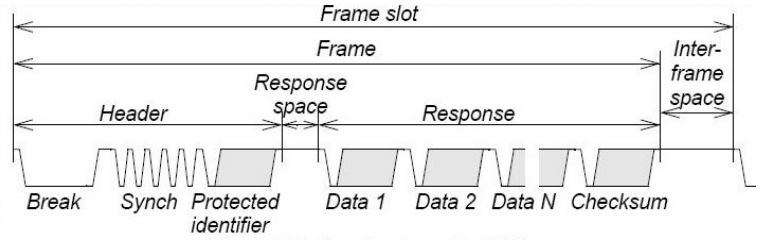
Protokol CAN je najbolj pogosto uporabljen v avtomobilskih komunikacijskih omrežjih in ima veliko prednosti pred ostalimi protokoli. Zagotavlja fleksibilno in robustno komunikacijo z omejeno zamudo, prav tako pa je cenovno zelo ugoden ter enostaven. Ponuja različne stopnje pasovne širine. Ta protokol se je začel uporabljati leta 1983 v podjetju Robert Bosch. Leta 1991 so predstavili dvodelno specifikacijo protokola CAN 2.0, ki se je delila na dve različici. Ti dve sta CAN 2.0A ter CAN 2.0B, ki ju je leta 1993 standardizirala organizacija ISO [22].



Slika 2.1: Podatkovni okvir protokola CAN [6].

2.1.2 LIN

Protokol LIN je nizkocenovni serijski protokol razvit za komunikacijo med sistemi v avtomobilih. Nudi zelo nizke hitrosti 20kbit/s, uporablja pa se predvsem za komfortne funkcije. Deluje po principu gospodar/suženj in spada v skupino časovno proženih protokolov. Ta protokol je danes množično uporabljen na področju potniške kabine zaradi njegove preprostosti ter relativno nizke cene [22].



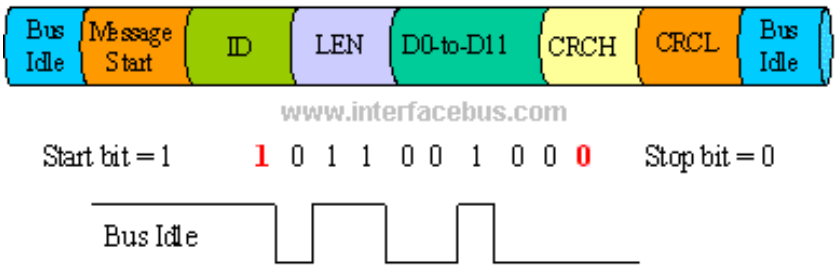
Slika 2.2: Podatkovni okvir protokola LIN [33].

2.1.3 MOST

Protokol MOST je bil razvit za podporo multimedije in infotainment sistemov s komunikacijo med prenosom avdio, video podatkov in nadzorom informacij. Ta protokol je nastal leta 1998 s sodelovanjem konzorcija proizvajalcev avtomobilov, sistemskih arhitektov ter dobaviteljev sestavnih delov ključnih komponent. Sedaj se uporablja kot privzeti protokol za te sisteme v avtomobilih, saj nudi poceni izdelavo in implementacijo ter učinkovit prenos podatkov. MOST je sinhrono omrežje, ki uporablja point-to-point prenos podatkov. Doseže lahko hitrosti do 150Mbit/s [22, 15].

2.1.4 Byteflight

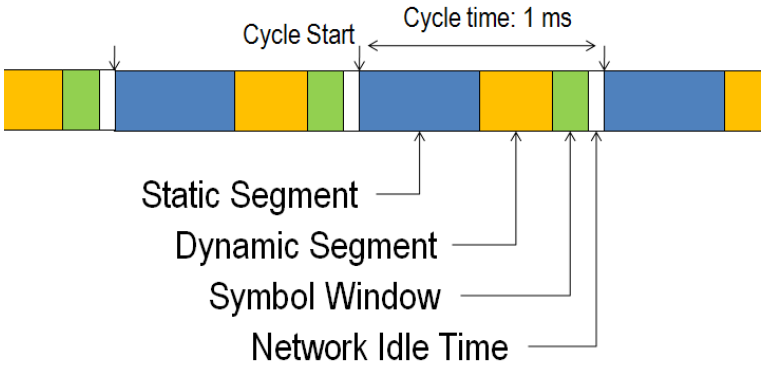
Protokol Byteflight je razvilo podjetje BMW. V glavnem se uporablja v sistemih, ki zahtevajo visoko stopnjo varnosti, zato je nepogrešljiv v avtomobilizmu in letalski elektroniki. Byteflight temelji na FTDMA mehanizmu in tipično uporablja zvezdno topologijo omrežja. Nastal je leta 1996, prvič pa je bil uporabljen leta 2001 v vozilih BMW serije 7 [22].



Slika 2.3: Podatkovni okvir protokola Byteflight [8].

2.1.5 FlexRay

Protokol FlexRay je razvilo združenje velikih avtomobilskih podjetij s ciljem, da bi dosegli visoko hitrost prenosa podatkov, hkrati pa bi bil zelo zanesljiv in prilagodljiv. Prva specifikacija protokola je bila objavljena leta 2004. Protokol je zasnovan na TDMA in FTDMA mehanizmih, uporablja pa zvezdno in večzvezdno topologijo omrežja [22, 18].



Slika 2.4: Komunikacijski cikel pri protokolu FlexRay [9].

2.2 Obstoječi varnostni sistemi v avtomobilih

V tem poglavju so predstavljene že obstoječe rešitve znanih proizvajalcev vozil. Bile so pomemben del pri razvoju moje ideje. V določenih segmentih so precej podobne, imajo pa tudi nekaj manjših pomanjkljivosti, ki jih želim z realizacijo lastnega sistema odpraviti.

2.2.1 Audi

Braking Guard

Leta 2006 je Audi z modelom Q7 predstavil sistem Braking guard, ki je aktiven nad 30km/h in meri razdaljo do ovire, ki je pred vozilom [16]. Radarska enota je nameščena na prednjem delu avtomobila, pri merjenje razdalje pa potrebuje odbojne površine za pravilno delovanje.

Pre Sense

Leta 2010 so predstavili sistem imenovan Pre sense, ki uporablja dvojni radarski sistem z dodatnimi kamerami. Deluje v štirih fazah, od opozorila voznika do dejanskega avtomatskega zaviranja [32]. Na nek način se avto pripravi na trk in zavaruje vse potnike. Izboljšana različica sistema je bila leta 2012 nagrajena s strani Euro NCAP [25], kar potrjuje kvaliteto sistema.

Avoidance Assistant

Leta 2015 je izšel sistem Avoidance Assistant, ki posreduje pri krmiljenju in zaviranju ob trku. Spremlja vozila na nasprotnem pasu in se pomaga izogniti le tem, ter ostalim oviram pred vozilom [3]. Tudi ta sistem je bil prvič prisotnem pri modelu Audi Q7, ki so ga izdali leta 2015.

2.2.2 BMW

Active Protection

BMW je boljše sisteme za preprečevanje nesreč predstavil leta 2012 v modelih razreda 7. Gre za dobro organiziran podsistem nadzora vzmetenja, motorja, zavor in pnevmatik, ki skupaj z različnimi elektronskimi senzorji preventivno posredujejo ob nesreči [29].

Driving Assistant Plus

Naslednja leta je BMW razvijal sistem Driving Assistant Plus pri večini njihovih avtomobilov. Sestoji iz kamere na prednjem delu in več senzorjev razporejenih po vozilu. Vključuje opozorila pri menjavi cestnega pasu, zaznavo ovir pred avtomobilom, ki delujejo tudi v težkih vremenskih razmerah in še več zelo pomembnih in praktičnih stvari, ki zelo vplivajo na varnost potnikov [28].

2.2.3 Volkswagen

Front Assist

Predstavljeno v modelu Touareg leta 2010, sistem detektira če se preveč približamo vozilu in v najslabšem primeru tudi avtomatsko zavira [2]. Proizvajalec to in več podobnih funkcij vsako leto posodablja in izboljšuje.

2.2.4 Mercedes-Benz

Pre-Safe

Mercedes-Benz je pričel z varnostnimi sistemi leta 2003 v razredu S. Bil je eden izmed prvih proizvajalcev, ki je v svoja vozila vgradil tovrstne varnostne zaščite [30]. Pre-Safe se je skozi leta izpopolnjeval in tako so v sezoni 2010 dobili nagrado s strani Euro NCAP [25]. Naslednje leto (2011), je proizvajalec v serijo B vgradil radarsko zaznavo ovir tudi v osnovnem modelu. Skupek funkcij pod imenom Pre-Safe vsebuje avtomatsko zaviranje pred trkom, zaznavo ovir ali pešcev pred vozilom in mnogo podobnih rešitev, ki Mercedez-Benz uvršča v sam vrh najbolje izpopolnjenih vozil na področju zaznavanja in preprečitve nesreč.

Pre-Safe Brake

Leta 2006 se je kot Brake Assist BAS Plus prvič predstavil sistem za zaznavo nesreč pred vozilom, ta pa se je kasneje razvil v Pre-Safe Brake (2009). Prenovljena različica z novim imenom s pomočjo radarja zazna da bo vozilo trčilo in avtomatsko zavira. Tik pred trkom oz 0.6 sekunde pred, pa vozilo zavira z maksimalno močjo, da ublaži udarec. Poleg prednjega radarja, ima isto tehnologijo tudi zadaj in ustrezno spremlja promet ter dogajanje za vozilom, da ne bi prišlo do naleta [4].

Distronic

Distronic je sistem za avtomatsko prilagajanje razdalje do naslednjega vozila spredaj. Aktiven je le pri tempomatu, njegov domet pa je 150-200m. Sestavlja ga radarska komponenta za merjenje razdalje, v boljši različici Distronic PLUS, pa sta radarja dva [10].

2.2.5 Volvo

Collision Warning

Volvo je proizvajalec, ki je precej napreden na področju zaznave nesreč in že več let vodilna sila. Začeli so leta 2006, ko so predstavili Collision Warning, ki opozori voznika na oviro pred njim [35].

City Safety

City Safety se od leta 2008 vgrajuje v vsa vozila Volva, od najbolj osnovnega modela do najdražjih modelov. Gre za skupek funkcij, ki preprečijo neposreden trk spredaj in zadaj, s tem pa omilijo poškodbe voznika in sopotnikov. Ker se je sistem izkazal za tako dobrega, so nekatere zavarovalnice znižale cene zavarovanj za vozila Volvo [31]. Ta sistem je požel več nagrad s strani Euro NCAP [25]. Pri prvi različici je sistem deloval do hitrosti 30km/h, leta 2013 so to hitrost povišali na 50km/h. V najnovejšem modelu XC90, predstavljenem leta 2015, pa sistem City Safety deluje pri vseh hitrostih [31].

IntelliSafe

To je najnovejša tehnologija s strani Volva. Njihova vizija je, da bodo sistem do leta 2020 razvili tako dobro, da nihče v Volvu ne bo umrl ali utrpel težkih poškodb [20]. Sistem seveda ni v vseh modelih enako prisoten, v modelu XC90, pa je v celoti zastopan. Sedaj IntelliSafe ponuja raznovrsten nabor funkcij:

- zaznavanje in preprečitev nesreč,
- zaščita pešcev in kolesarjev okoli vozila,
- spremljanje mrtvih kotov,
- 360 stopinjski pogled na vozilo,
- opozarjanje na izčrpanost voznika,
- varno menjavo voznih pasov,...

2.2.6 Analiza pomanjkljivosti obstoječih sistemov

Iz vseh teh tehnologij je preprosto razbrati, da so vse v prid vozniku in čim bolj varni vožnji na cestah. Nekatere delujejo celo v prid peščem in kolesarjem okoli vozila. Nekatere so zelo dovršene, druge malo manj. Precej nas je zmotil doseg vseh teh naprav. Zakaj ne bi voznika opozorili na dogajanje na cesti, ki je od njega tako oddaljeno, da se je resnično mogoče izogniti nesreči? Večina sistemov proizvajalcev avtomobilov je preventivna ampak omejena na približno 200m. Na avtocesti je to zelo majhna razdalja in je potrebno hitro ukrepati. Sami bi radi to razdaljo raztegnili na vsaj 1000m, tako se povsem izognemo morebitnemu trčenju. Gre za kompleksen problem, ki ga bomo poskušali rešiti z aplikacijo.

Slika 2.5: Volvo XC90 je trenutno najbolj varen serijski avtomobil [34].

2.3 Uporabljene tehnologije

2.3.1 Java

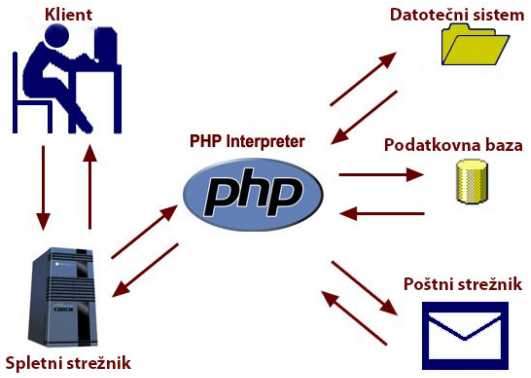
Java je objektno usmerjeni, prenosljivi programski jezik, ki ga je razvil James Gosling s sodelavci v podjetju Sun Microsystems. Projekt, ki se je v začetku (leta 1991) imenoval Oak(hrast), je bil razvit kot zamenjava za C++. Jave ne smemo zamenjevati z jezikom JavaScript, ki ima podobno ime, ter podobno, C-jevsko skladnjo. Različica Java 1.0 je bila objavljena leta 1996, zadnja različica je 8.0(marec 2014). Javo vzdržuje in posodablja Oracle - Sun Microsystems. Tolmač za Javo je vgrajen v večino spletnih brskalnikov, s tem se javanski programi(appleti) lahko izvajajo kot del HTML dokumenta. Java je en izmed bolj uporabljenih programskih jezikov, v letu 2016 ga uporablja več kot 9 milijonov razvijalcev po vsem svetu[21].

2.3.2 XML

Extensible Markup Language ali XML je razširljiv označevalni jezik[37], ki omogoča format za zapisovanje podatkov in njihovo izmenjavo med več omrežji. Berljiv je tako človeku kot napravam, stremi pa k simplističnosti, generalizaciji in uporabnosti po celotnem spletu. Prvič se je pojavil leta 1998, kot prenovljena različica jezika SGML, ko so Jon Bosak, Tim Bray, C. M. Sperberg-McQueen, James Clark in še mnogo drugih želeli strniti vse dobre lastnosti SGML v en jezik brez funkcionalnosti, ki so se izkazale za redundantne. XML se uporablja tudi pri Android platformi. Ker je tako široko uporabljen, je možnost, da podatke s spleta dobimo v formatu XML. Prav tako se bodo podatki verjetno pošiljali v obliki XML. Ker ima Android omogočenih veliko funkcij, je tesno povezan z XML, saj se tako podatki najlažje izmenjujejo.

2.3.3 PHP

PHP je skriptni jezik za uporabo na strani strežnika namenjen za razvoj spletnih aplikacij, prav tako pa je uporabljen kot splošno namenski programski jezik. Prvotno ga je razvil Rasmus Lerdorf leta 1994, sedaj pa razvoj nadaljuje tako imenovana The PHP Group. Kratica imena PHP je bila naprej okrajšava za Personal Home Page, zdaj pa se je to spremenilo v Hypertext Preprocessor. PHP koda je lahko implementirana v HTML kodi, lahko pa je tudi uporabljena v kombinaciji z različnimi spletnimi tehnologijami in ogrodji[27]. Lahko ga primerjamo z Microsoftovim sistemom ASP, VBScript in drugimi. Podoben je običajno strukturiranim programskim jezikom, najbolj jezikoma C in Perl. Gre za strežniški programski jezik, kar pomeni, da teče na strežniku, za njegovo delovanje pa potrebujemo spletni strežnik. PHP spada v skupino interpreterskih programskih jezikov, kar pomeni, da se na začetku ne prevede celotna izvorna koda, ampak se interpretira sproti. Spletni strežni ima to funkcijo, da interpretirano izvorno kodo pošlje brskalniku v obliki HTML kode, tako uporabnik ne more videti originalne izvorne kode.



Slika 2.6: Potek obdelave PHP na spletnem strežniku.

2.3.4 MySQL

MySQL je odprtokodna implementacija ali sistem relacijske podatkovne baze, ki za delo s podatki uporablja jezik SQL. Ime izvira iz kombinacije "My", ki je okrajšava za hčer soustanovitelja Michaela Wideniusa in ŠQL", kar pa je kratica za Structured Query Language. MySQL je bil razvit izpod rok švedskega podjetja MySQL AB, glavni ljudje za vsem tem pa so David Axmark, Allan Larsson in Michael Widenius. Prva vezija je izšla leta 1995 in je nemudoma postala it[23]. Leta 2008 jih je kupilo podjetje Sun Microsystems, nato pa je Sun Microsystems leta 2010 kupilo podjetje Oracle Corporation, ki je sedaj gigant na področju računalništva[24]. MySQL je napisan v C in C++, deluje pa na veliko operacijskih sistemih kot so AIX, BSDi, FreeBSD, HP-UX, eComStation, Linux, OS X, Microsoft Windows in drugih. Deluje na principu odjemalec-strežnik, pri čemer lahko strežnik namestimo kot sistem, porazdeljen na več strežnikih. Obstaja veliko število odjemalcev, zbirk ukazov in programskih vmsenikov za dostop do podatkovne baze MySQL. Gre za zelo uporabljen model podatkovnih baz, ki si je tekom let pridobil veliko privržencev med razvijalci.

2.3.5 Git

Git je brezplačen, odprtokodni sistem za verzioniranje kode[13]. To je poseben sistem za varnostno shranjevanje kode, v zadnjih letih je postal nepogrešljivi, celo obvezni del znanja razvijalca programske opreme. Nastal je leta 2005 izpod rok Linusa Torvaldsa, ki ga je razvil v namene podpore razvoja jedra operacijskega sistema Linux[14]. Njegova preprostost za uporabo ter podpora porazdeljenemu programiranju skoraj neomejenemu številu razvijalcev na enem ali večih projektih, je ena izmed večjih prednosti pred ostalimi podobnimi sistemi za verzioniranje kode.

2.4 Uporabljena ogrodja

2.4.1 Android Studio

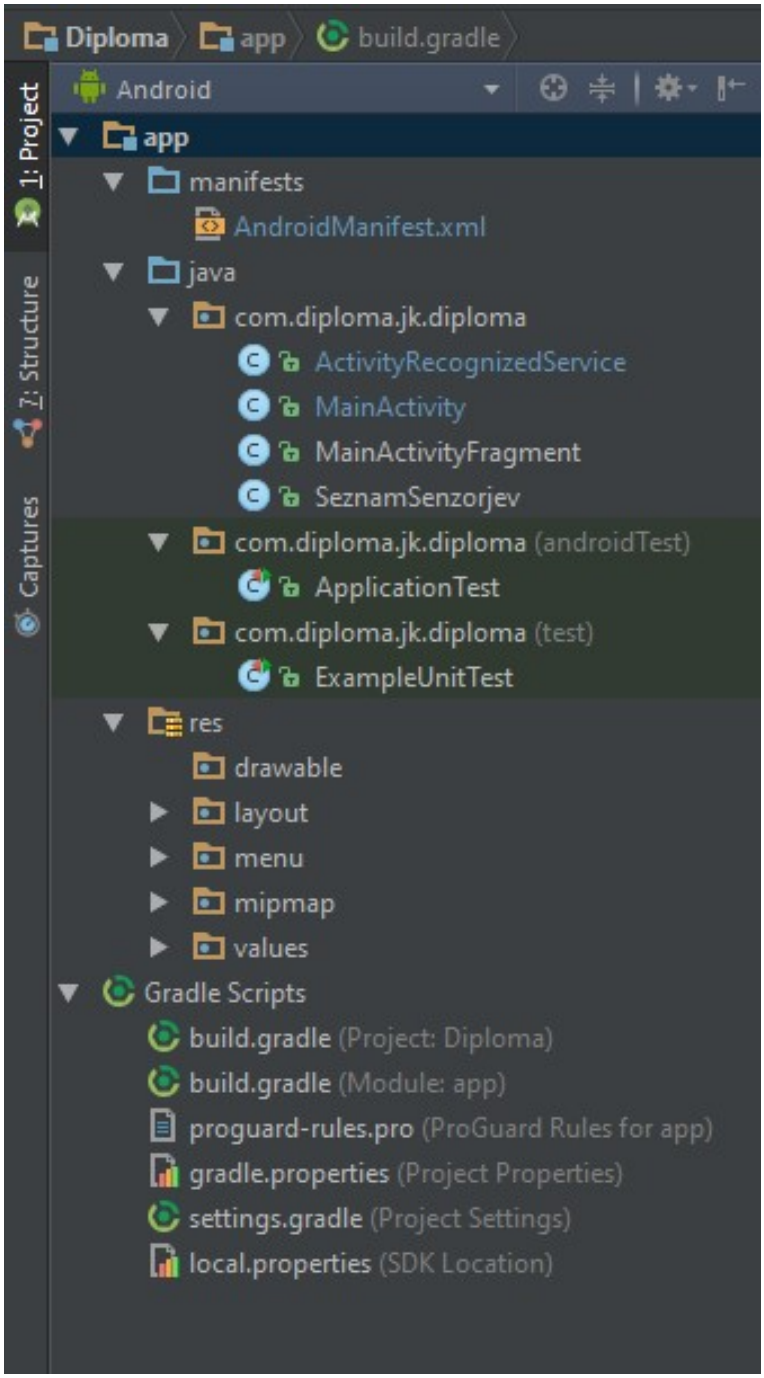
Android Studio je uradno razvojno okolje(IDE) za razvoj Android aplikacij[1]. Temelji na IntelliJ IDEA[19], ki je eno izmed najboljših razvojnih okolij za razvijalce programske opreme. Poleg IntelliJ-evega močnega urejevalnika in razvijalskih orodij, Android Studio ponuja veliko množico uporabnih stvari, ki olajšajo razvoj Android aplikacij.

Struktura projekta

Vsak projekt v Android Studiu ima enega ali več modulov z datotekami, ki vsebujejo programsko kodo ter datoteke z različnimi viri. Tipi modulov so:

- Android app modules,
- Library modules,
- Google App Engine modules.

Privzeto, Android Studio prikaže projekt v Android Project prikazu, kot je prikazano na sliki 2.7.



Slika 2.7: Začetna struktura map pri projektu, generirana z Android Studio.

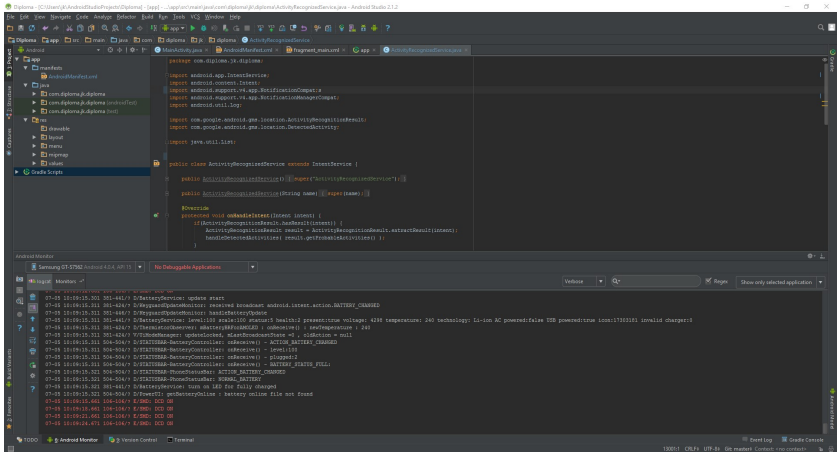
Vse pomembne datoteke za zagon aplikacije so na vrhnjem nivoju pod Gradle Scripts, vsak modul pa vsebuje naslednje mape:

- manifests,
- java,
- res.

Projekt, ki ga ustvari Android Studio je na disku nekoliko drugačen, kot je prikazan v tem pogledu. Med različnimi pogledi se lahko enostavno preklaplja vendar menim, da je privzeti dejansko najboljši.

Uporabniški vmesnik

Uporabniški vmesnik v Android Studiu je sestavljen iz več logičnih oken, ki so prikazane na sliki 2.8.



Slika 2.8: Osnovno okno Android Studia.

1. Orodna vrstica omogoča veliko opravil, med drugim tudi zagon aplikacije in zagon Android orodij,

2. Krmilna vrstica vodi čez projekt in odpira datoteke za urejanje. Omogoča strnjen pogled na strukturo projekta,

3. V Urejevalnem oknu se ureja koda. Okno se prilagaja glede na tip trenutno odprte datoteke, to pride do izraza pri urejanju XML datotek,

4. Okno za orodja omogoča dostop do posebnih opravil kot so urejanje projekta, iskanje, verzioniranje kode in več. Okno se lahko skrči ali razširi.

5. Statusna vrstica prikazuje status projekta in samega IDE-ja poleg vseh opozoril in sporočil.

Android Studio ima možnost, da razvijalec aplikacijo med samim delom tudi testira. To je možno izvesti na dva načina. Prvi je ta, da se na računalnik fizično poveže Android napravo in se na njo naloži aplikacija, ki se razvija. Drugi način pa je, da uporabimo vgrajen emulator, ki simulira Android napravo. Ta emulator lahko simulira veliko različnih Android naprav in je zaradi tega zelo uporaben pripomoček.

2.4.2 Firebase

Firebase je ponudnik storitev v oblaku ter backend storitev. Gre za podjetje, ki ima sedež v Kaliforniji, natančneje v San Franciscu. Ponuja široko paleto storitev za razvijalce spletnih kot tudi mobilnih aplikacij. Podjetje sta leta 2011 ustanovila Andrew Lee in James Tampin[11]. Primarni produkt Firebase je tako imenovana realtime database storitev, ki razvijalcem ponuja API s katerim lahko shranjujejo in sihronizirajo podatke preko več različnih platform ali naprav. Oktobra leta 2014 jih je kupil in pod svoje okrilje vzel gigant Google. Firebase je resnično močno orodje pri razvoju spletnih in mobilnih aplikacij, saj ponuja velik nabor storitev:

- **Analytics**
  - **Firebase Analytics** je brezplačna storitev, ki ponuja vpogled v uporabo aplikacije in dejavnost uporabnikov.
- **Develop**
  - **Firebase Cloud Messaging** je naslednjik GCM in ponuja rešitev za sporočila in obvestila na platformah kot so Android, iOS ter spletnih aplikacijah
  - **Firebase Auth** je storitev za avtentikacijo uporabnikov le z client-side izvorno kodo. Omogoča vpis z znanimi socialnimi omrežji kot so Facebook, Twitter, GitHub in Googlom. Omogoča vpis tudi samo z emailom in geslom.
  - **Realtime Database** je en izmed paradnih konjev Firebase, saj omogoča tako imenovano podatkovno bazo v realnem času ter zaledno, backend storitev. Gre za API, ki omogoča, da so podatki aplikacije shranjeni in uporabljeni v oblaku pripravljeni na uporabo preko vseh uporabnikov.

– **Firebase Storage** omogoča varovano shranjevanje in prenašanje datotek aplikacije, ne glede na kvaliteto povezave. Razvijalci jo lahko uporabljajo za shrambo slik, audio in video datotek. Ta storitev je podprta s strani Google Cloud Storage.

– **Firebase Hosting** storitev je namenjena statičnim spletnim stranem, ki se ne spreminjajo dinamično.

– **Firebase Test Lab for Android** ponuja testno okolje za aplikacije. To orodje je zelo dober pripomoček razvijalcem, saj lahko aplikacijo testirajo tudi če sami niso napisali testne kode. Storitev sama preizkusi in testira aplikacijo, vnre pa nam razne opise, videe in slike napak.

Slika 2.9: Vse storitve, ki jih ponuja Firebase [12].

Poglavje 3

Uporabljene metode in postopki

Tukaj je opisana večina postopkov in metod, ki so bile potrebne za delovanje aplikacije. Nekatere so bile bolj kompleksne kot druge, v veliko pomoč pa so bile razne storitve, ki jih ponuja Google ali kakšen drugi ponudnik. Aplikacija se imenuje Diploma in ima logo kot je na spodnji sliki:

Slika 3.1: Logotip aplikacije pod imenom Diploma [26].

3.1 Preverjanje položaja naprave

Za pravilno uporabo aplikacije, se mora naprava nahajati v avtomobilu. Najprej smo želeli sami preverjati kdaj je naprava v vozilu, vendar smo se zaradi lažje uporabe zatekli h Googlovem API-ju DetectedActivity, ki izhaja iz ActivityRecognitionApi in prepozna aktivnost naprave. Možna je zaznava med osmimi različnimi aktivnostmi, a nas je zanimala samo "IN_VEHICLE", se

pravi ko je naprava v avtomobilu. Elegantna rešitev, ki nam je prihranila ogromno časa. Uporabnik aplikacije ne bo mogel uporabljati drugje kot samo v vozilu, saj je samo tam uporabna. Tako smo se lahko lotili nadaljnega dela pri aplikaciji. Na sliki 3.2 je odsek kode, ki prikazuje zaznavo vseh aktivnosti na voljo v tem API-ju.

3.2 Orientacija naprave in smer vožnje

Da bi lahko omejili pošiljanje obvestil tistim, ki se jih ne tiče, je potrebno poznati orientacijo naprave v določenem trenutku. To je nujno, da se ne pošilja obvestil uporabnikom, ki se peljejo v nasprotno smer na avtocesti. Ker smo v prvem delu pošiljali obvestila kar vsem uporabnikom, smo se poskušali s pridobljeno orientacijo temu izogniti. Smer vožnje v našem primeru se pridobi s pomočjo vgrajenih senzorjev v telefonu in sicer s pospeškomerom in senzorjem za geomagnetno polje. Seveda pa ima Android to dobro pripravljeno na uporabo in dobimo orientacijo naše naprave le z nekaj klici funkcij, ki so vnaprej pripravljene za to.

3.3 Merjenje lokacije in hitrosti

V grobem se za merjenje hitrosti uporabljata dva načina. Prvi je z uporabo GPS tehnologije, drugi pa s pomočjo pospeškomera(Accelerometer), ki je v telefonu. Pri prvi je potrebna podatkovna povezava kot tudi GPS povezava, kar se pozna pri malenkost večji porabi baterije. Hitrost bomo v tem primeru merili s pomočjo GPS in sicer tako, da spremljamo frekvenco spreminjanja geo-lokacijskih točk med vožnjo v določenem času. Ker je hitrost enaka prevoženi razdalji v nekem času, potrebujemo samo še razdaljo, ki smo jo prevozili v eni periodi časa. Razdalje med posameznimi geo-točkami so drugačne zaradi različnih faktorjev, tudi zaradi ukrivljenosti Zemlje. Za to

```java
public class ActivityRecognizedService extends IntentService {

    public ActivityRecognizedService() {
        super("ActivityRecognizedService");
    }

    public ActivityRecognizedService(String name) {
        super(name);
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        if(ActivityRecognitionResult.hasResult(intent)) {
            ActivityRecognitionResult result = ActivityRecognitionResult.extractResult(intent);
            handleDetectedActivities( result.getProbableActivities() );
        }
    }

    private void handleDetectedActivities(List<DetectedActivity> probableActivities) {
        for( DetectedActivity activity : probableActivities ) {
            switch( activity.getType() ) {
                case DetectedActivity.IN_VEHICLE: {
                    Log.e( "ActivityRecogition", "In Vehicle: " + activity.getConfidence() );
                    break;
                }
                case DetectedActivity.ON_BICYCLE: {
                    Log.e( "ActivityRecogition", "On Bicycle: " + activity.getConfidence() );
                    break;
                }
                case DetectedActivity.ON_FOOT: {
                    Log.e( "ActivityRecogition", "On Foot: " + activity.getConfidence() );
                    break;
                }
                case DetectedActivity.RUNNING: {
                    Log.e( "ActivityRecogition", "Running: " + activity.getConfidence() );
                    break;
                }
                case DetectedActivity.STILL: {
                    Log.e( "ActivityRecogition", "Still: " + activity.getConfidence() );
                    break;
                }
                case DetectedActivity.TILTING: {
                    Log.e( "ActivityRecogition", "Tilting: " + activity.getConfidence() );
                    break;
                }
                case DetectedActivity.WALKING: {
                    Log.e( "ActivityRecogition", "Walking: " + activity.getConfidence() );
                    if( activity.getConfidence() >= 75 ) {
                        NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
                        builder.setContentText( "Are you walking?" );
                        builder.setSmallIcon( R.mipmap.ic_launcher );
                        builder.setContentTitle( getString( R.string.app_name ) );
                        NotificationManagerCompat.from(this).notify(0, builder.build());
                    }
                    break;
                }
                case DetectedActivity.UNKNOWN: {
                    Log.e( "ActivityRecogition", "Unknown: " + activity.getConfidence() );
                    break;
                }
            }
        }
    }
}
```

Slika 3.2: Vse aktivnosti, ki jih beleži ActivityRecognitionApi.

se uporablja Haversinov algoritem, ki ga bomo opisali spodaj.

3.4 Haversinov algoritem

Haversinova formula je zelo pomembna pri navigaciji. Z njo se računajo razdalje med dvema točkama na sferičnem objektu glede na zemljepisno širino in dolžino.

$$hav\left(\frac{d}{r}\right) = hav(\varphi_2 - \varphi_1) + cos(\varphi_1)cos(\varphi_2)hav(\lambda_2 - \lambda_1) \tag{1}$$

Formula Haversine.

- hav() je funkcija haversine:

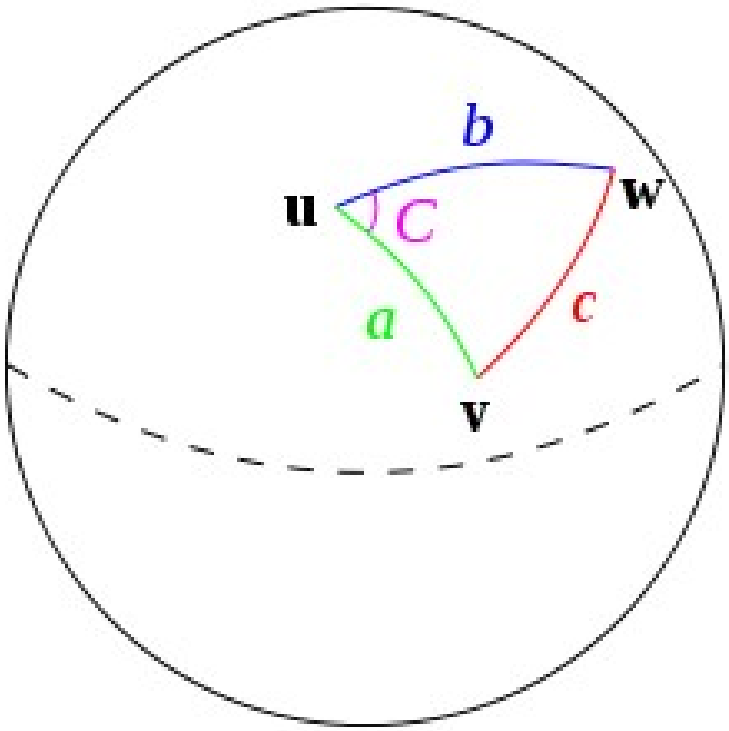
$$hav(\theta) = sin^2\left(\frac{\theta}{2}\right) = \frac{1 - cos(\theta)}{2} \tag{2}$$

- d je razdalja med dvema točkama
- r je radij krogle(sfere)
- φ_1, φ_2: zemljepisna širina točke 1 in 2 v radianih
- λ_1, λ_2: zemljepisna dolžina točke 1 in 2 v radianih

Gre za poseben primer bolj splošne formule v sferični trigonometriji, ki se imenuje Zakon Haversin in opisuje stranice ter kote v sferičnih trikotnikih. Prva tabela haversin v angleškem jeziku je bila predstavljena leta 1805 s strani Jamesa Andrewa [17].

Za podano kroglo je trikotnik na površini, ki ga opisujejo krožnice in povezuje točke $\boldsymbol{u}$, $\boldsymbol{v}$ in $\boldsymbol{w}$. Če so dolžine teh stranic $\boldsymbol{a}$(od $\boldsymbol{u}$ do $\boldsymbol{v}$), $\boldsymbol{b}$(od $\boldsymbol{u}$ do $\boldsymbol{w}$) in $\boldsymbol{c}$(od $\boldsymbol{v}$ do $\boldsymbol{w}$), in je nasproti stranice $\boldsymbol{c}$ kot $\boldsymbol{C}$, potem pravi Zakon Haversin:

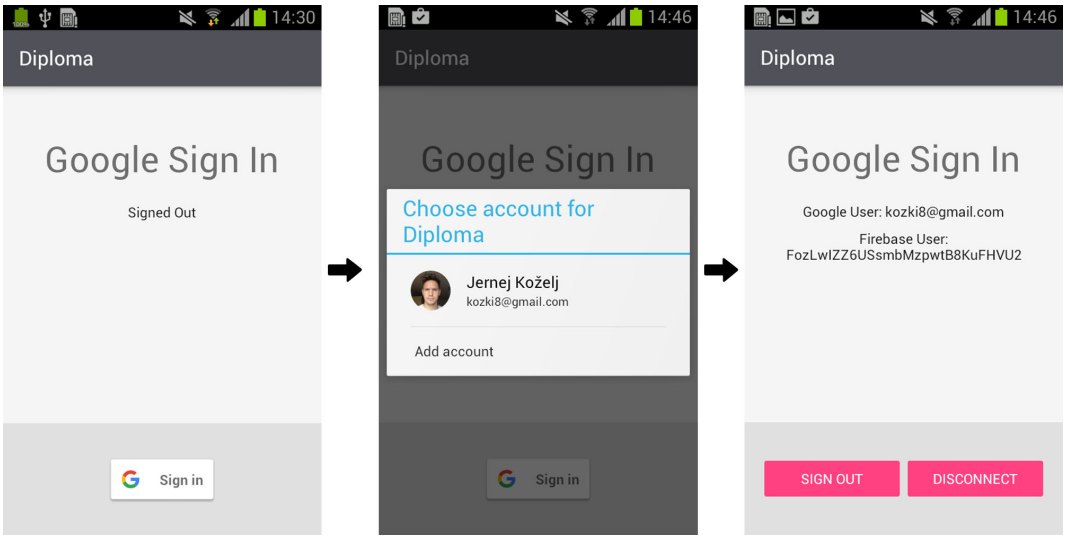
$$hav(c) = hav(a - b) + sin(a)sin(b)hav(C) \tag{3}$$



Slika 3.3: Sferični trikotnik rešen s pomočjo haversin [7].

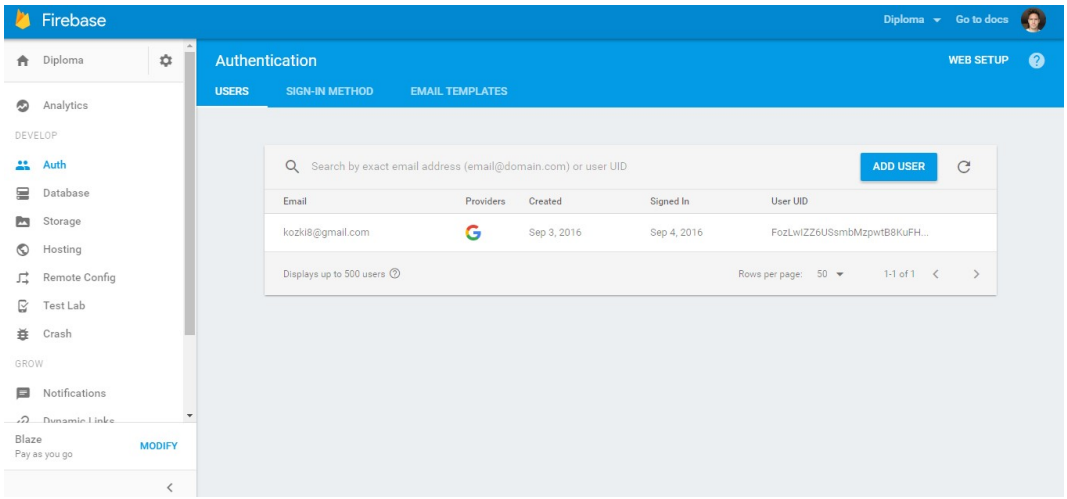
3.5 Prijava in registracija uporabnika

Da bi lahko pošiljali obvestila o nevarnostih na cesti, moramo imeti neko bazo uporabnikov, kateri pošiljamo ta obvestila. Tega smo se lotili na dva načina. Prvi je bil preko Firebase in sicer z registracijo z Googlovim računom, saj vemo da večina ljudi v tem času uporablja vsaj eno njihovo storitev in imajo zato zagotovo Google račun. Pri tem se uporabnik brez težav vpiše v aplikacijo z lastnim Google računom, mi pa pridobimo novega uporabnika v bazi na Firebase oblaku. To je prikazano s slikami spodaj.



Slika 3.4: Prijava v aplikacijo z Google računom.

Na spodnji sliki je prikazana Firebase konzola, s katero lahko uporavljamo našo aplikacijo, dodajamo API-je, preglejujemo aktivnost uporabnikov in njihovo število in še mnogo drugih uporabnih stvari. Na tej sliki je prikaz dodanega uporabnika, v tem primeru smo to mi. Generira se tudi User UID, s katerim lahko targetiramo posameznega uporabnika.



Slika 3.5: Prikaz uporabnika v Firebase konzoli.

Za drugi način smo naredili prijavo samo z email-om in geslom. Za to smo na brezplačnem strežniku "000webhost"[36] postavili MySQL bazo s pomočjo PHP. Gre za preprosto bazo z malo tabelami, saj potrebujemo le seznam uporabnikov in obvestila za pošiljanje.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	ID	int(16)			No			
☐	Email	varchar(30)	latin1_general_ci		No			
☐	Password	varchar(16)	latin1_general_ci		No			
☐	Latitude	varchar(10)	latin1_general_ci		No			
☐	Longitude	varchar(10)	latin1_general_ci		No			
☐	Direction	varchar(5)	latin1_general_ci		No			

Check All / Uncheck All *With selected:*

Slika 3.6: Tabela User v podatkovni bazi.

Oba načina smo uporabili samo za preizkus. Firebase je dosti bolj sodoben kot klasični način s postavitvijo in nudi boljše rezultate, prav tako pa porabi manj baterije na napravi. Ker imamo pri Firebase brezplačno licenco, smo nekoliko omejeni, a že ta ponuja precej dobrih storitev.

3.6 Pogoj za pošiljanje obvestila

Ta del naloge je najpomembnejši, zato smo mu namenili tudi največ časa. Zelo je pomembno, kdaj se pošlje obvestilo ostalim uporabnikom aplikacije, saj se v tem primeru lahko tudi "rešujejo življenja". Izhajali smo iz Googlove tehnologije Google Traffic, ki v realnem času beleži gostoto prometa na cestah in jih v Google Maps temu primerno obarva:

- zelena barva - cesta je normalno prevozna
- oranžna barva - cesta je srednje obremenjena
- rdeča barva - na cesti je precej vozil in prihaja do zastojev

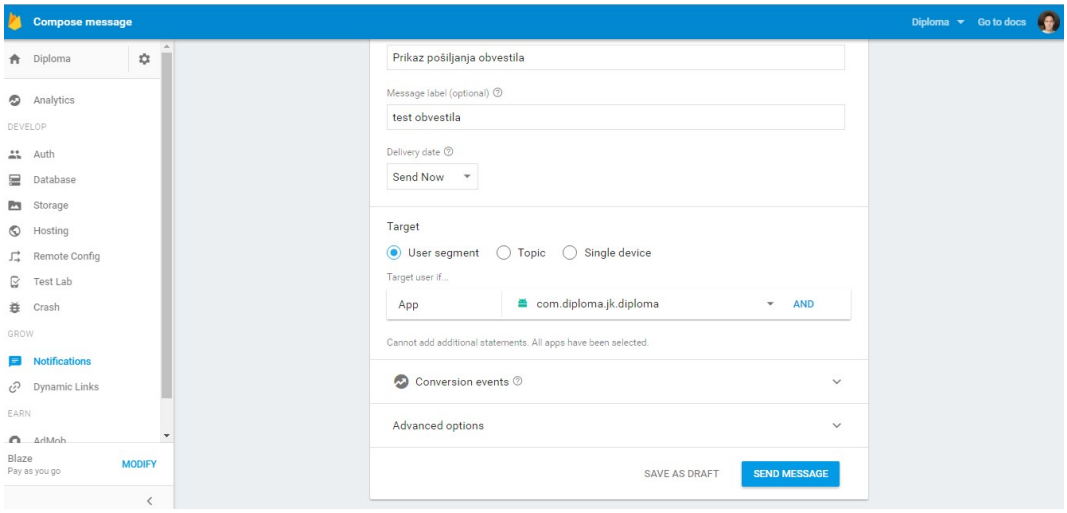
Preko Googlovega APIja Google Maps JavaScript lahko dodamo prikaz gostote prometa v realnem času v katerokoli aplikacijo, ki uporablja njihove zemljevide. Glede zbiranja informacij o gostoti prometa so mnenja deljena. Po zbranih informacijah s spleta niti ni povsem jasno in transparentno s strani Googla, kako se ti podatki pridobivajo. Najboljša in tudi najverjetnejša ocena je, da Google pobira podatke s telefonov, se pravi lokacijo kjer se telefoni nahajajo. Izbira ali želimo, da Google vedno spremlja našo lokacijo je odvisna od nas samih, to možnost se da v nastavitvah Android telefona tudi izključiti. Sami smo mnenja, da je v tem primeru več pozitivnih kot negativnih strani beleženja informacij o lokaciji, saj pripomorejo pri obveščanju o zastojih na cestah in posledičnem izogibanju le teh. Stvar je v bistvu zelo preprosta, Google pobira lokacije in hitrost s telefonov, ki imajo vključeno njihovo aplikacijo Maps ter možnost za pošiljanje lokacije. Te podatke prenese na njihove mape in tako imamo gostoto prometa v realnem času prikazano na mapi na telefonu. Kot v primeru naše aplikacije za protinaletni sistem, je tudi tukaj veliko odvisno od števila uporabnikov. Več kot je trenutnih uporabnikov, lepše se bo aplikacija obnesla in boljši bodo rezultati. Kar se tiče varnosti podatkov o lokaciji posameznega uporabnika, Google zagotavlja, da se podatki kriptirajo, ter da se začetna in končna postaja vedno pobrišeta, da ne bi prišlo do kakšnih zlorab. Ogromna baza

podatkov o prometnih informacijah Googlu omogoča, da lahko s pomočjo statistike predvideva kako se bo promet na določen dan obnašal. Zanimiva je izjava ene izmed vodilnih ljudi pri razvoju njihove aplikacije Maps, ki pravi, da lahko ugotovijo ali se v določenem mestu odvija maraton. Veliko več ljudi se premika hitreje kot ponavadi, na cestah pa je manj vozil. Prav tako pravi, da je potrebno aplikaciji zaupati in v veliki meri, nas bo pripeljala na cilj hitreje kot bi se sami oziroma brez pomoči aplikacije. Torej s pomočjo Googlovega APIja spremljamo gostoto prometa na cesti, obenem pa se lahko zanašamo še na protinaletni sistem, ki bo v primeru nesreče pred nami takoj opozoril vse uporabnike. Zelo varno!

Pogoj za pošiljanje obvestila pa je precej kompleksen. Če se uporabnikova hitrost znatno zmanjša v zelo kratkem času in na kratki razdalji, potem je pogoj skoraj izpolnjen. Potrebno pa je upoštevati vse mogoče robne pogoje. Uporabnik se lahko ustavi na bencinski postaji ob avtocesti. Lahko se vozi po slovenskih avtocestah, kjer imamo cestninske postaje, ki ne služijo več nobenemu namenu, le temu, da je potrebno hitrost znižati na okoli 70km/h. V prvi iteraciji aplikacije se bodo obvestila poslala vsem uporabnikom ob vsakem znatnem zaviranju na zelo kratki razdalji, to pošiljanje se bo postopoma izboljševalo glede na rezultate, ki jih bomo pridobili s testiranjem na cestah. Torej obvestilo se pošlje, če uporabnik zmanjša hitrost za vsaj 45km/h na razdalji, ki je krajša od 80m. To je razdalja na kateri se padec hitrosti za 45km/h zelo pozna, torej lahko sklepamo, da je zaviranje zaradi nekega nepredvidenega dogodka spredaj.

3.7 Pošiljanje obvestil

Pri aplikaciji je najpomembnejša stvar to, da uporabnik prejme obvestilo o nekem dogodku, ki se dogaja pred njim na cesti. Za pošiljanje obvestil sem uporabil Firebase Cloud Messaging ali FCM, ki olajša proceduro ter pripomore k manjši porabi baterije in mobilnih podatkov. Gre za prenovljeno verzijo Google Cloud Messaging, ki je podedovala vse dobre lastnostni GCM, dobila pa še veliko novih možnosti in funkcionalnosti. Osnovna ideja GCM je ta, da lahko razvijalec preko serverja pošlje tako imenovano vsiljeno obvestilo uporabniku, ob kakšni posodobitvi ali pa zgolj za opomnik k večji uporabi aplikacije. To stvar smo obrnili sebi v prid in izkoristili možnost pošiljanja teh obvestil, ko aplikacija teče v ozadju ali pa sploh ne teče. Prva stopnja pošiljanja opozoril je ta, da bo nek dogodek sprožil pošiljanje obvestila vsem uporabnikom aplikacije. Kasneje se bo to pošiljanje omejilo na uporabnike, ki se nahajajo na določenem območju in se premikajo v isto smer kot pošiljatelj opozorila; torej se vozijo za njim in je to opozorilo najbolj relevantno samo zanje.Preko Firebase lahko pošljemo obvestila uporabnikom, kot je prikazano na sliki spodaj:



Slika 3.7: Pošiljanje obvestila preko Firebase.

Sedaj smo poslali obvestilo. Na napravi se pojavi najprej v zgornji vrstici, nato pa lahko obvestilo pregledamo. Prikaz tega je na spodnji sliki. Da so obvestila boljša je potrebno dodati še zvok, ki uporabnika dodatno opozori na nevarnost. Prejeta obvestila:

Slika 3.8: Prejeta obvestila na napravi.

Ker je test sistema precej težko izedljiv z našimi resursi, poleg tega pa bi bil zelo nevaren, smo aplikacijo testirali samo na računalniku, v testnem okolju. Kot je prikazano zgoraj, se pošiljajo obvestila o nevarnosti vsem uporabnikom naše aplikacije. Želja je, da bi lahko aplikacijo testirali v varovanem okolju v dejanskih avtomobilih in po nekem vnaprej pripravljenem scenariju. Tako bi dobili dejanske rezultate in uspešnost aplikacije. Več o tem je napisano v sklepnih ugotovitvah.

Poglavje 4

Evalvacija

Celotno delo bi ocenil kot uspešno, razen dejstva, da aplikacija ni bila uporabljena v realnem svetu ter določenih pomanjkljivosti pri izdelavi. Problem varnosti na cestah je nekaj kar me zanima in temu bom namenil preostanek študija na fakulteti. To je le prvi korak v smer, ki me interesira, zato sem zadovoljen z rezultatom. Veliko bi se lahko popravilo pri izgledu aplikacije, ter seveda pri samih performansah. Pri nekaterih uporabljenih metodah sem pazil na porabo baterije in prenos podatkov, pri drugih ne, tako da je tukaj še prostor za izboljšave. S testiranjem bi verjetno prišli do novih ugotovitev in boljših rešitev, zato je ideja, da bi naprave s sistemom Android namestili na daljinsko vodene avtomobilčke verjetno korak v pravo smer. Z mentorico, doc. dr. Mojca Ciglarič so pogovori tekli v tej smeri, tako da upam da se bodo uresničili v prihodnje. Največja izboljšava pa je omejitev pošiljanja obvestila vsem uporabnikom aplikacije. Prvotna ideja je bila, da bi ta obvestila dobili le tisti, ki se nahajajo v radiju 1000m od lokacije poslanega obvestila. Poleg omejitve z oddaljenostjo pa je tu pomembna tudi smer vožnje, saj nas ponavadi na avtocesti ne zanima, kaj se dogaja na nasproti vozečem pasu. To mi ni uspelo, obvestilo se pošlje vsem uporabnikom aplikacije, tako da ta del bi moral v prihodnje izboljšati. Gre za precej kompleksno kombinacijo rešitev, ki mi je v danem času ni uspelo dokončati. Kot sem že omenil, zanima me avtomobilska industrija, saj je tesno prepletena z računalništvom.

Sodobni avtomobili so skoraj bolj računalniki, kot prevozno sredstvo, zato vem, da je to smer katero želim raziskati.

Poglavje 5

Sklepne ugotovitve

V okviru diplomske naloge sem razvil aplikacijo protinaletnega sistema, ki je uporabna v avtomobilu. Aplikacija je razvita za mobilno platformo Android in deluje na veliki večini naprav s tem operacijskim sistemom. Delo je razdeljeno na dva dela, v prvem je teoretična predstavitev vseh protokolov v sodobnih avtomobilih, varnostnih sistemov, uporabljene tehnologije ter uporabljena ogrodja. V drugem delu, pa so predstavljene glavne funkcionalnosti aplikacije ali pa rešitev nekega problema, na katerega sem naletel med samim razvojem. Tako si bralec lažje predstavlja potek samega dela ter izzive, ki so prisotni pri razvoju. V drugem delu je nekaj več slik in posnetkov zaslona, za lažjo predstavo.

5.1 Zaključek

Ideja za temo diplomske naloge se nam je porodila slučajno in še vedno smo mnenja, da je zelo dobra ter da ima potencial. Smo pa tekom izdelave ugotovili, da smo si zastavili precej zahtevno nalogo, ki je v tako kratkem času izjemno težko izvedljiva oziroma je težko izvedljiva v takem obsegu, kot je bila prvotno mišljena. Zadeva je na prvi pogled skoraj trivialna, ko pa se

začnemo poglabljati v detajle, pa ugotovimo, da temu ni tako. Ampak tako je pri vsaki stvari. Med delom oziroma pisanjem naloge, smo se ogromno naučili. Spoznali smo Android Studio, prebrali zelo veliko različnih člankov o tehnologijah v sodobnih avtomobilih, njihovih varnostnih sistemih in dejanski uporabi le teh. Prišli smo do veliko zaključkov, kako bi posamezne lahko izboljšali in kaj bi dodali pri protinaletnem sistemu. Izdelali smo aplikacijo ki deluje, testirana pa je bila večinoma v testnem okolju, se pravi simulirana na računalniku, zato je težko reči, kako bi se dejansko obnesla z več uporabniki na realnem problemu. Vemo, da bomo sistem še večkrat popravljali in ga spravili do te stopnje, kot je bil v naših glavah.

5.2 Nadaljnji razvoj

Aplikacija je dobro zasnovana, ampak kot vsaka stvar, bi lahko bilo precej stvari boljših. Obvestila bi lahko pošiljal le tistim, ki se jih dejansko obvestilo tiče, tako pa je ostalo pri tem, da se pošlje kar vsem uporabnikom. Pri samem razvoju nisem bil dosti pozoren na izgled aplikacije. V teh časih je izgled aplikacije zelo pomemben, prav tako pa sama uporabniška izkušnja. Tako, da je v tem segmentu še veliko prostora za izboljšave. Končni izdelek je zaenkrat na voljo le tistim, ki dobijo datoteko aplikacije od mene, razvijalca. Zaradi pomanjkanja testiranja, sem se odločil, da aplikacije ne objavim v Google Play Store, saj ne služi namenu. Verjetno pa kdaj v prihodnosti bo, saj bi rad, da se preizkusi na realnem problemu in da bi se aplikacija dejansko uporabljala. Kot je napisano na začetku dela, bi bilo zanimivo videti ta sistem v nekem posebnem modulu, ki bi se vgradil v avto in bi bil prisoten prav v vsakem vozilu. Namen je, da se ljudje zavedajo nevarnosti naleta na avtocestah in nekaj naredijo v tej smeri.

Literatura

[1] AndroidStudio. https://developer.android.com/studio/index.html, 2016. [Online; accessed 28-June-2016].

[2] Volkswagen Front Assist. http://www.volkswagen.co.uk/technology/proximity-sensing/front-assist, 2016. [Online; accessed 28-June-2016].

[3] Audi Avoidance Assistant. https://www.audi-mediacenter.com/en/lighter-more-efficient-and-full-of-high-tech-the-new-audi-q7-2486/driver-assistance-systems-2551, 2016. [Online; accessed 28-June-2016].

[4] Mercedes-Benz Pre Safe Brake. https://techcenter.mercedes-benz.com/en/pre_safe_brake/detail.html, 2016. [Online; accessed 28-June-2016].

[5] Gianluca Cena and Adriano Valenzano. An improved can fieldbus for industrial applications. *Industrial Electronics, IEEE Transactions on*, 44(4):553–564, 1997.

[6] Wikimedia Commons. Can-frame in base format with electrical levels without stuffbits. https://en.wikipedia.org/wiki/CAN_bus#/media/File:CAN-Bus-frame_in_base_format_without_stuffbits.svg, 2016.

[7] Wikimedia Commons. Kupljen .png logo aplikacije. https://en.wikipedia.org/wiki/File:Law-of-haversines.svg, 2016.

[8] Larry Davis Copyright. Structure of byteflight frame. http://www.interfacebus.com/byteflight-Protocol-Format-frame.png, 2016.

[9] National Instruments Corporation. Communication cycle flexray. http://www.ni.com/cms/images/devzone/tut/FlexRay_Cycle_Overview.png, 2016.

[10] Mercedes-Benz Distronic. https://techcenter.mercedes-benz.com/en/distronic_plus/detail.html, 2016. [Online; accessed 28-June-2016].

[11] Firebase. https://firebase.google.com/, 2016. [Online; accessed 22-August-2016].

[12] Firebase. Firebase features. https://lh3.googleusercontent.com/pmFdSCiNJf4foF41QJvWGKhkB_sn3Lneql4Vk5kos_nP7n3ieddBGnCKsxQxGjl2tl2A-OEd3_az1Yo8kU0tPnDLe2N2uQ=s888, 2016.

[13] Git. https://git-scm.com/, 2016. [Online; accessed 28-June-2016].

[14] Git2. https://en.wikipedia.org/wiki/Git_(software), 2016. [Online; accessed 28-June-2016].

[15] Andreas Grzemba. *MOST®: The Automotive Multimedia Network from MOST25 to MOST150*. Franzis, 2011.

[16] Audi Braking Guard. http://www.euroncap.com/en/ratings-rewards/euro-ncap-advanced-rewards/2012-audi-pre-sense-basic/, 2016. [Online; accessed 28-June-2016].

[17] Haversine. https://en.wikipedia.org/wiki/Haversine_formula, 2016. [Online; accessed 24-July-2016].

[18] Xuewen He, Qiang Wang, and Zhenli Zhang. A survey of study of flexray systems for automotive net. In *Electronic and Mechanical Engineering*

and Information Technology (EMEIT), 2011 International Conference on, volume 3, pages 1197–1204. IEEE, 2011.

[19] IntelliJ. https://www.jetbrains.com/idea/, 2016. [Online; accessed 28-June-2016].

[20] Volvo IntelliSafe. http://www.volvocars.com/us/about/our-innovations/intellisafe, 2016. [Online; accessed 28-June-20016].

[21] Java. https://en.wikipedia.org/wiki/Java_(programming_language), 2016. [Online; accessed 28-June-2016].

[22] Ugur Keskin. In-vehicle communication networks: a literature survey. *Computer Science Report*, 10, 2009.

[23] MySQL. https://en.wikipedia.org/wiki/MySQL, 2016. [Online; accessed 26-August-2016].

[24] MySQLAB. https://en.wikipedia.org/wiki/MySQL_AB, 2016. [Online; accessed 26-August-2016].

[25] Euro NCAP. http://www.euroncap.com/en/about-euro-ncap/, 2016. [Online; accessed 28-June-2016].

[26] Pixel perfect. Kupljen .png logo aplikacije. http://image.flaticon.com/icons/svg/196/196156.svg, 2016.

[27] PHP. https://en.wikipedia.org/wiki/PHP, 2016. [Online; accessed 24-August-2016].

[28] BMW Driving Assistant Plus. http://www.bmw.com/com/en/newvehicles/x/x6/2014/showroom/driver_assistance/driving_assistant_plus.html#t=l, 2016. [Online; accessed 28-June-2016].

[29] BMW Active Protection. http://www.bmw.com/com/en/newvehicles/7series/sedan/2012/showroom/driver_assistance/active-protection.html#t=l, 2016. [Online; accessed 28-June-2016].

[30] Mercedes-Benz Pre Safe. https://techcenter.mercedes-benz.com/en/pre_safe_system/detail.html, 2016. [Online; accessed 28-June-2016].

[31] Volvo City Safety. http://www.volvocars.com/uk/about/our-innovations/city-safety, 2016. [Online; accessed 28-June-2016].

[32] Audi Pre Sense. http://www.auditech.org/acont-917.html, 2016. [Online; accessed 28-June-2016].

[33] HW server. Structure of lin frame. http://www.hw-server.com/obrazek/lin_frame_structure, 2016.

[34] Swedespeed. Volvo xc90 sets new standards. http://www.swedespeed.com/wp-content/uploads/2015/09/16SEPT15_XC90_cover.jpg, 2016.

[35] Volvo Collision Warning. http://support.volvocars.com/uk/cars/Pages/owners-manual.aspx?mc=Y555&my=2015&sw=14w20&article=c2aa4a930c8b6746c0a801e801ce49be, 2016. [Online; accessed 28-June-2016].

[36] webhost. https://www.000webhost.com/, 2016. [Online; accessed 24-August-2016].

[37] XML. https://en.wikipedia.org/wiki/XML, 2016. [Online; accessed 28-June-2016].