ŠOLSKI CENTER VELENJE

ELEKTRO IN RAČUNALNIŠKA ŠOLA VELENJE

Trg mladosti 3, 3320 Velenje

MLADI RAZISKOVALCI ZA RAZVOJ ŠALEŠKE DOLINE

RAZISKOVALNA NALOGA

PAMETNI AVTOMOBIL − MOBILNI ASISTENT

Tematsko področje: RAČUNALNIŠTVO

Avtorja:

Andrej Kronovšek, 4. letnik

Jan Liber, 4. letnik

Mentor:

Islam Mušić, prof.

Somentorja:

Uroš Remenih, inž. inf.

Simon Konečnik, univ. dipl. inž.

Velenje, 2017

Raziskovalna naloga je bila opravljena na Šolskem centru Velenje, Elektro in računalniški šoli.

Mentor: Islam Mušić, prof.
Somentorja: Uroš Remenih, inž. inf., Simon Konečnik, univ. dipl. inž.

Datum predstavitve:

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD ŠC Velenje, šolsko leto 2016/2017
KG zvočno upravljanje / spletni strežnik / mobilna aplikacija / prepoznavanje govora
AV KRONOVŠEK, Andrej / LIBER, Jan
SA MUŠIĆ, Islam / KONEČNIK, Simon / REMENIH, Uroš
KZ 3320 Velenje, SLO, Trg mladosti 3
ZA ŠC Velenje, Elektro in računalniška šola
LI 2017
IN MOBILNI ASISTENT
TD Raziskovalna naloga
OP VIII, 29 str., 2 graf., 19 sl., 27 vir.
IJ SL
JI sl / en
AI Kot voznika sva opazila, da med vožnjo potrebujeva določene informacije, ki niso takoj na voljo oz. je dostop do njih težaven in za voznika avtomobila dokaj nevaren. Primer teh informacij so trenutno stanje na cestah, zastoji, podatki o vremenu … Med vožnjo je gledanje na zaslon mobilne naprave nevarno, saj odvrača pozornost vozniku. Tega izziva sva se lotila tako, da sva se lotila razvoja mobilne aplikacije, ki omogoča glasovno upravljanje in lažji dostop do želenih informacij. Raziskovala sva najnovejše in najpogosteje uporabljene tehnologije za prepoznavo govora. V najino aplikacijo sva integrirala uporabo dveh različnih tehnologij. Najina aplikacija pridobiva podatke z različnih internetnih platform, ki podajajo informacije o trenutnem vremenu, stanju na cesti. Omogoča povezavo z Googlovim spletnim koledarjem in nam sporoča prihajajoče dogodke. Za lažje delovanje sva postavila lastno storitev za posredovanje koristnih informacij voznikom. Podatke črpava z različnih strežnikov in posredujeva le voznikom uporabne informacije. Ker je namen najine aplikacije asistenca vozniku, mu ta ponuja tudi podatke o stanju avtomobila, če ima le-ta priklopljen bralnik podatkov in dogodke, ki so vneseni v njegovem spletnem koledarju. Cilj najine raziskovalne naloge je izdelati mobilno aplikacijo, ki bi delovala, kot asistent vozniku, na takšen način, da uporaba le-te, ne predstavlja nevarnosti za udeležence v prometu.

KEY WORD DOCUMENTATION

ND Elektro in računalniška šola Velenje, 2016/2017
CX voice control / web server / mobile application / voice recognition
AU KRONOVŠEK, Andrej / LIBER, Jan
AA MUŠIĆ, Islam / KONEČNIK, Simon / REMENIH, Uroš
PP 3320 Velenje, SLO, Trg mladosti 3
PB ŠC Velenje, Elektro in računalniška šola
PY 2017
TI MOBILE ASSISTANT
DT RESEARCH WORK
NO VIII, 29 p., 2 graf, 19 sl., 27 ref.
LA SL
AL sl / en
AB As drivers we noticed that during driving we need specific information. Those are not available or they need driver's attention, which can be dangerous during driving. Example of that information is current traffic status, traffic jams, weather data. Looking on your mobile phone while driving is dangerous, because it can divert attention from the road. We approached this challenge, by developing mobile application, which uses voice control. This provides easier access to the information. We researched the newest and most used technologies for voice recognition and integrated two different technologies in the application. The application acquires data from different internet platforms, which provide information about current weather and traffic. It enables Google Calendar integration that informs the driver about incoming events. For better efficiency we developed our own service. Data is collected from different web servers. Only information that is useful is then forwarded to the driver. Because the application's purpose is to assist the driver, it provides cars current data, but only if it's connected with the OBD transmitter. Along with this data, there are also events entered in online calendar. The purpose of this research project is to create mobile application, which will work as an assistant to the driver. The application shouldn't obstruct or endanger the driver and other drivers.

KAZALO VSEBINE

KAZALO SLIK

KAZALO GRAFOV

UPORABLJENE OKRAJŠAVE

CMU - Carnegie Mellon University

IOS - iPhone OS

WP - Windows Phone

STT - Speech-to-text

TTS - Text-to-speech

API - Aplikacijski programski vmesnik (ang. application programming interface*)*

GPS - Globalni sistem pozicioniranja (ang. Global Positioning System)

OBD - ang. On-board diagnostics

MIT - Tehnološki inštitut Massachusettsa (ang. Massachusetts Institute of Technology)

CSS - kaskadne stilske podloge (Cascading Style Sheets)

HTML - jezik za označevanje nadbesedila (ang. HyperText Markup Language)

IDE - Integrirano razvojno okolje (ang. Integrated Development environment)

XML - razširljivi označevalni jezik (ang. Extensible Markup Language)

JDK - ang. Java Development Kit

JVM - ang. Java virtual machine

ECMAScript - Skupnost evropskih računalniških proizvajalcev (ang. European Computer Manufacturers Association)

JSON - ang. JavaScript Object Notation

SUPB - Sistem za upravljanje s podatkovnimi zbirkami

RDBMS - relacijski sistem za upravljanje baz (ang. relational database management system)

SQL - strukturirani povpraševalni jezik za delo s podatkovnimi bazami (ang. Structured Query Language)

NoSQL - "ne SQL" (ang. Non SQL)

SDK - ang. software development kit

HTTP - ang. HyperText Transfer Protocol

HTTPS - ang. HyperText Transfer Protocol Secure

URL - enolični krajevnik vira (ang. Uniform Resource Locator)

npm - node package manager

FRI - Fakulteta za računalništvo in informatiko

ARSO - Agencija Republike Slovenije za okolje

ang. - angleško

itd. - in tako dalje

npr. - na primer

1 UVOD

Oba avtorja sva v zadnjem času opravila vozniški izpit in se začela voziti. Kmalu sva opazila, da če se voziva sama, velikokrat potrebujeva kakšne podatke o cesti, zastojih, vremenu, kaj je naslednja stvar na mojem koledarju in še kaj. Na internetu lahko najdemo te podatke, prav tako tudi obstajajo storitve, ki jih ponujajo. Do teh podatkov med vožnjo ne moreva dostopati, saj je uporaba telefona med vožnjo prepovedana in tudi ni priporočena. Prav tako pa do vseh teh podatkov ne moremo dostopati z uporabo že delujočih asistentov, kot so Siri, Cortana itd. Možnost, da boš imel prometno nesrečo, če uporabljaš mobilni telefon je 4-krat večja. Poleg tega je bilo v Sloveniji leta 2015 kar 440 nesreč povezanih z uporabo mobilne naprave (www.policija.si). Tako sva se odločila izdelati aplikacijo, pri kateri voznik ne bi odvračal pozornosti od prometa in bi do potrebnih podatkov dostopal z glasovnim upravljanjem. Zato bi mogoče tej nalogi bolj ustrezal naslov Pametni voznik – mobilni asistent. Po tem ko sva si zadala to nalogo, sva si postavila štiri hipoteze.

1.1 Hipoteze

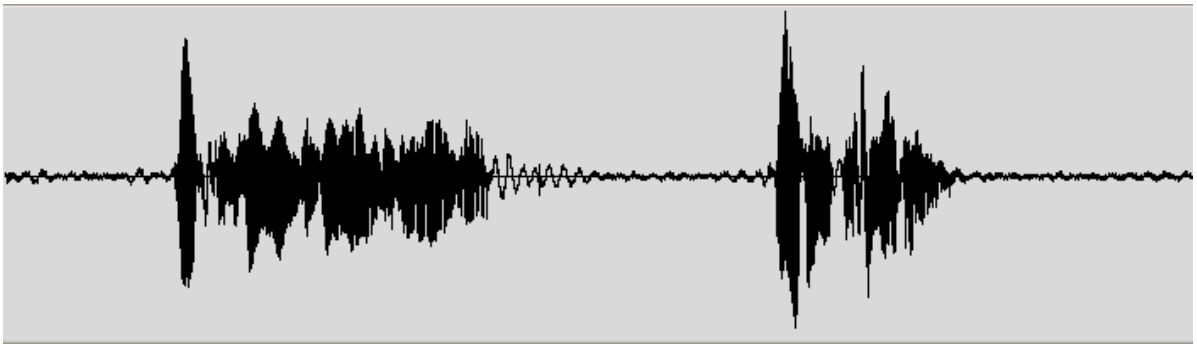
- Prepoznavanje govora bo uspešno kljub prisotnosti šumov, zvokov iz okolice.
- Uporaba slovenskega jezika ne bo ovirala uporabe aplikacije.
- Pridobivanje podatkov na mobilni telefon bo potekalo nemoteno.
- Aplikacijo je možno razviti z uporabo že obstoječih storitev.

2 PREGLED OBJAV

Najprej sva raziskala, kaj vse je že napisano o elementih, za katere sva mislila, da jih bova uporabila v najini raziskovalni nalogi. Našla sva veliko spletnih virov, saj je to področje, ki se ves čas nadgrajuje in spreminja. To pa je razlog, da je posledično manj pisnih virov, saj so ti bili velikokrat zastareli in nama niso bili v pomoč. Glede na to, da se najina aplikacija sproži ob besedah "Hey Alex!", je logično začeti pregled objav z govorom.

2.1 Govor

Govor je kompleksni fenomen. Ljudje poredko razumejo, kako je ustvarjen in zaznan. Naša percepcija velikokrat je, da je govor zgrajen iz besed in da je vsaka beseda sestavljena iz zlogov. Realnost je na žalost zelo drugačna. Govor je dinamični proces brez delov, ki so lahko razločeni od drugih.



Slika 1: Govor, vir: [1]

Vsi moderni opisi govora so do neke mere verjetnostni, kar pomeni, da ni nobene meje med enotami ali besedami, kar pomeni, da prevajanje iz govora v tekst in druge aplikacije z govorom niso nikoli 100 % pravilne. (Povzeto po Bučković, 2013)

2.1.1 Struktura govora

Do sedaj je razumevanje govora takšno: Govor je neprekinjen tok zvoka, kjer lahko prepoznamo več podobnih razredov zvoka ali glasov. Splošno znanje je, da so besede zgrajene iz glasov, ampak to sploh ni res. Akustične lastnosti valovne oblike se lahko zelo razlikujejo, ker so odvisne od mnogih dejavnikov, kot so telefonska zveza, zvočniki, slog govora in druge. Pogosto lahko poiščemo tri ali več delov glasa. Prvi del glasu je odvisen od prejšnjega glasa, srednji del glasa je stabilen, zadnji del glasu pa je odvisen od naslednjega glasa, ki mu sledi.
Včasih so glasovi mišljeni v kontekstu, takrat smo pozorni na to, da je izgovorjava enega glasu odvisna od tistih glasov, ki stojijo zraven njega. (Povzeto po Bučkovič, 2013)

2.2 Orodja za prepoznavanje govora in njihova zgodovina

Avtomatsko razpoznavanje govora se je začelo razvijati v petdesetih letih prejšnjega stoletja v Bellovih laboratorijih in tam so razvijali razpoznavanja izoliranih števk. To so izvajali s pomočjo takrat dosegljive tehnologije – elektronk, uporov, tuljav in kondenzatorjev. Šele trideset let kasneje pa je razvoj teh sistemov prišel tako daleč, da je dosegel praktično uporabo

sistemov. V osemdesetih letih je hiter razvoj signalnih procesorjev omogočil hitrejši razvoj sistemov avtomatskega razpoznavanja govora. (Povzeto po Kačič, 1995)

2.2.1 Razvrstitev sistemov glede na način prepoznavanja govora

Sistem za razpoznavanje govora lahko opredelimo kot sistem, ki prepozna in pretvori izgovorjene besede oz. akustični signal, ki ga oseba z govorjenjem v mikrofon proizvede v tekstovni zapis. Ta sisteme lahko prav tako uporablja za upravljanje računalnika ali drugih naprav s pomočjo govornih ukazov, s čimer se lahko znebimo uporabe tipkovnice in drugih vhodnih naprav. Takšen sistem poznamo tudi pod imenom "avtomatski sistem za razpoznavo govora" ali kot sistem "govor v besedilo" (ang. speech to text). (Povzeto po Bučković, 2013)

Glede na način razpoznave govora delimo sisteme v tri skupine. To so:

- sistemi za razpoznavanje izoliranih besed
- sistemi za razpoznavanje vezanega govora
- sistemi za razpoznavanje tekočega govora

Pri razpoznavanju izoliranih besed sistem zahteva premor med vsako besedo. Začetek in konec morata biti natančno določena, premori med besedami v stavku pa morajo biti dolgi vsaj 200 ms ali več. To je najpreprostejša oblika prepoznavanja, saj konec besede ne vpliva na začetek druge.

Pri razpoznavanju vezanega govora je govor sestavljen iz izoliranih besed, pri čemer je vhod tekoč govor. Ta način je zelo podoben prejšnjemu, vendar dovoljuje, da ločene izjave tečejo skupaj oz. jih izgovorimo z minimalnimi premori med njimi.

Pri razpoznavanju tekočega govora so besede med seboj povezane in niso ločene s premori oz. prekinitvami. Sistemi, ki uporabljajo ta sistem za razpoznavo govora, nimajo omejitev, a so zato veliko bolj kompleksni. Težava, s katero se ti soočajo je ta, da fonemi in besede vplivajo na besede in foneme za to besedo in tako otežujejo razpoznavanje. Težko je najti začetno in končno pozicijo vsake besede in tako se mora sistem prilagajati govorcu. (Povzeto po Bučković, 2013)

2.3 Sistemi za prepoznavanje govora

Na spletu je na voljo več sistemov za prepoznavanje govora, med katerimi ima vsak svoje prednosti in slabosti. Spodaj so našteti štirje, ki so bolj poznani, in med njimi sva izbirala.

- Pocketsphinx je odprtokodni program za prepoznavanje govora, razvit v okviru CMU Sphinx projekta. Je hiter in je osnovan tako, da deluje dobro na vgrajenih sistemih (kot RasperryPi). Njegovo slaba lastnost pa je počasno prepoznavanje govora. Je pa res, da je prepoznavanje narejeno brez internetne povezave, torej lahko deluje povsod, tudi tam kjer internetne povezave ni.
- Google STT je sistem za prepoznavanje govora, razvit s strani Googla. Uporabljajo ga telefoni z operacijskim sistemom Android, saj je to isti STT kot tisti, ki se zažene ko rečeš »OK, Google«. Za razliko od sistema PocketSphinx pa Google STT rabi internetno povezavo za delovanje.
- AT&T STT je sistem, ki ga uporablja telekomunikacijsko podjetje AT&T. Kot Google STT naredi vse dekodiranje na internetu in posledično rabi internetno povezavo.
- Julius je visokozmogljiv odprtokodni sistem za prepoznavanje govora. Ne rabi aktivne internetne povezave, vendar moraš ustvariti svoj akustični model, kar pa je precej kompleksna naloga.

(Povzeto po: https://jasperproject.github.io/documentation/configuration/, 26. 1. 2017)

2.4 Sintetizator govora

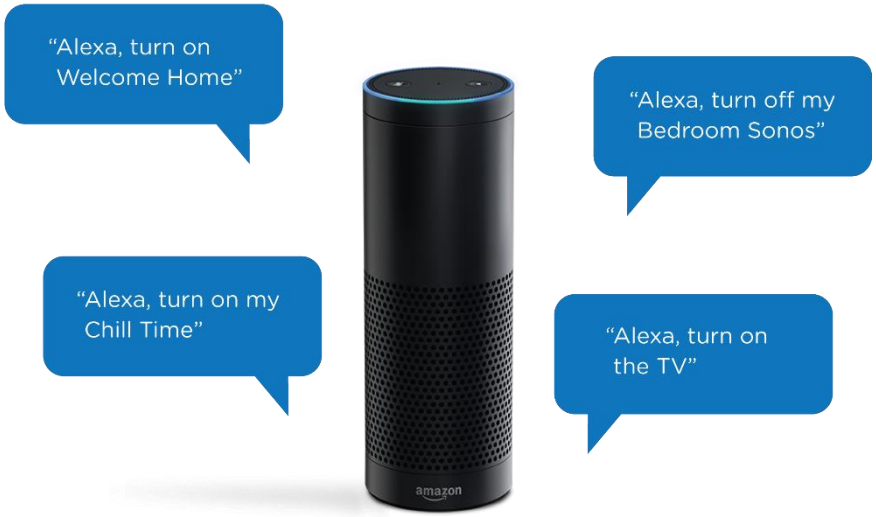
V preteklosti je veljalo, da pomeni avtomatska sinteza govora proces generiranja govornega signala na popolnoma umeten način s pomočjo sistemov, ki so do določene mere posnemali človekova govorila. Uporaba teh sistemov je bila omejena. Danes pa pojem pomeni vsako posedovanje informacij, ki jih s pomočjo govornega signala stroj posreduje uporabniku – človeku. Prednosti:

- Govor je najučinkovitejše sredstvo za sporazumevanje med ljudmi, tako lahko vsi, ki govorijo isti jezik razumejo govorjena poročila, ki jih posredujejo stroji. To znajo brez urjenja ali večje koncentracije.
- Uporabnik lahko sprejme informacije, četudi medtem opravlja kakšno drugo aktivnost.
- Uporabimo lahko telefonsko omrežje in informacijo sprejemamo na daljavo.
- Posredovano sporočilo ne potrebuje medija za zapis, kar pa je lahko slabost pri daljših besedilih. (Povzeto po Kačič, 1995)

2.5 Obstoječi pametni sistemi

Da sva vedela, kakšno aplikacijo izdelati, sva morala najprej pregledati, kaj ponujajo današnji asistenti. Osredotočila sva se na trenutno vodilne asistente, ki se lahko s tabo pogovarjajo, torej prepoznavajo cele strukture stavkov in oblikujejo ustrezen odgovor, po navadi dodajo informacije pridobljene iz interneta.

2.5.1 Alexa Amazon



Slika 2: Alexa Amazon, vir: [2]

Alexa je osebni asistent, ki omogoča nove zmožnosti za interakcijo med uporabniki in napravami. Poganja napravo Amazon Echo, skupaj z njo omogoča sposobnost predvajanja glasbe, nastavljanja alarma, štoparice in odgovarjanje na splošna vprašanja. Prilagaja se glasu in vzorcu govora, besedišča in osebnih nastavitev posameznika. Lahko upravlja s pametnimi napravami ter se tako uporablja kot centralna enota avtomatizacije doma. Aktivira se s pritiskom gumba ali z zaznavanjem določenih izgovorjenih besed uporabnika. Trenutno je na voljo le v angleškem in nemškem jeziku. (Povzeto po: https://developer.amazon.com/alexa, 28. 1. 2017)

2.5.2 Siri

Siri je pameten asistent, ki se uporablja na pametnih telefonih, tablicah in ostalih pametnih produktih podjetja Apple. Uporablja vsebinsko ozaveščenost, uporabniku omogoča upravljanje pametnih naprav, iskanje podatkov s spleta, prikazovanje priporočil …

2.5.3 Google asistent

Google asistent je virtualen asistent, ki je bil narejen kot nadgradnja dosedanjega "OK, Google" in je za razliko od tega narejen tako, da se lahko z njim pogovarjaš. Poleg tega pa lahko z njim brskaš po internetu, si dodajaš nove dogodke v urnik in opravljaš z aplikacijami, ki to dopuščajo. (Povzeto po: https://assistant.google.com, 25. 1. 2017)

2.5.4 Cortana

Cortana je pametna osebna pomočnica, ki pomaga iskati vsebine na računalniku, upravlja s koledarjem, upravlja s pošto, išče datoteke, se pogovarja z vami in tudi pove šale. Deluje z vnosom vprašanja v tekstovni obliki ali pa s pomočjo STT preko mikrofona. (Povzeto po: https://support.microsoft.com/sl-si/help/17214/windows-10-what-is, 25. 1. 2017)

2.6 Operacijski sistemi na mobilnih napravah

2.6.1 iOS

Je operacijski sistem, ki ga je razvil Apple le za svojo strojno opremo. Uporablja ga več naprav podjetja Apple − iPhone, iPad, iPod touch. Je drugi najbolj popularen operacijski sistem takoj za Androidom. Razvit je bil leta 2007 za iPhone, kasneje to leto so dodali še podporo za iPod Touch, tri leta kasneje pa še za iPad. (Povzeto po Zimic, 2012 in Mežnar, 2012)

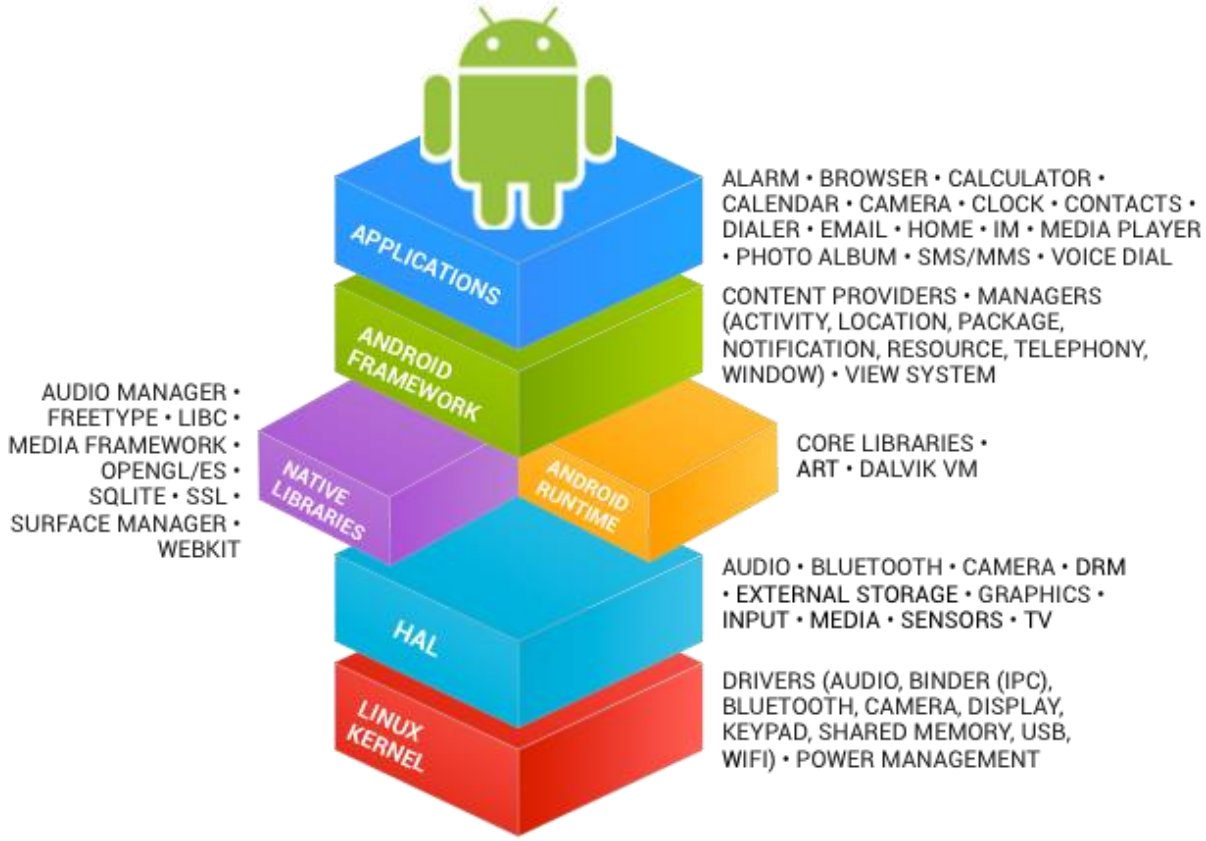
2.6.2 Windows Phone

Windows Phone je operacijski sistem razvit za Microsoftove mobilne naprave. Je manj popularen kot Android in iOS. Leta 2011 se je povezal s podjetjem NOKIA in od takrat naprej so imele mobilne naprave podjetja NOKIA operacijski sistem Windows Phone. Leta 2015 pa je prišel nov operacijski sistem za naprave, ki uporabljajo Windows Phone in sicer Windows 10 Mobile. (Povzeto po Zimic, 2012, Mežnar, 2012 in https://www.microsoft.com/en-us/mobile/windows10/, 5. 2. 2017)

2.6.3 Android

Android je odprtokodna programska oprema, ki jo je razvilo podjetje Google. Temelji na verziji sistema Linux, njegov glavni namen je uporaba na mobilnih napravah. Kadar govorimo o Androidu kot odprtokodni različici, je njegov pravilen naziv »Android Open Source Project«. Iz tega projekta izhaja veliko operacijskih sistemov, ki se med seboj razlikujejo in jih uporabljajo različna podjetja. Primer teh so različice podjetij Samsung, Huawei, LG ali katerokoli drugo podjetje, ki uporablja ta operacijski sistem. Lastnosti teh različic so dodatne

funkcije in prilagojen izgled. Prva različica Android 1.0, je izšla leta 2008. V času pisanja raziskovalne naloge, je najnovejša različica Android 7.1.1, imenovana Nougat. Na trgu je ta platforma najbolj popularna izmed vseh. Ker je glavni razvijalec te platforme Google, je razvijalcem omogočena lažja uporaba spletnih storitev podjetja Google. (Povzeto po Čulibrk, 2010)



Slika 3: Prikaz sistema Android in njegovih funkcij, vir: [3]

2.7 Orodja za razvoj mobilnih aplikacij

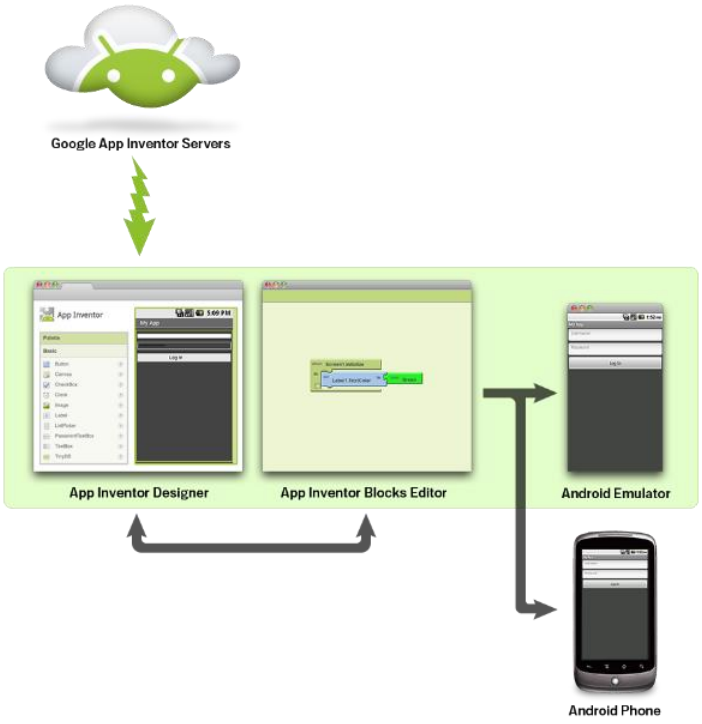
Za lažji razvoj mobilnih aplikacij, se uporabljajo razvojna okolja (IDE). To so programi, ki imajo dodatne funkcije in ponujajo lažji razvoj aplikacij za določeno platformo. Večinoma se razvojno okolje osredotoči na razvoj na eni platformi. Če želimo razvijati mobilne aplikacije, namenjene za več platform, t.i. hibridne aplikacije, lahko uporabljamo razvojna okolja ali pa uporabimo ogrodja (ang. frameworks). Z uporabo ogrodij, nismo vezani na en program, saj lahko kodo za aplikacijo pišemo v preprostem urejevalniku besedil. Za testiranje aplikacij zgrajenih s pomočjo ogrodij, potrebujemo dodatne razvojne module.

Preprostejše aplikacije lahko razvijemo s pomočjo enostavnih orodij. Omogočajo razvoj z

uporabo že izdelanih komponent, ki se jih nato na način »povleci in spusti« (ang. drag and drop) vključi v program. Za dodajanje funkcionalnosti, se na enak način dodajajo vnaprej določeni osnovni gradniki kode. (Povzeto po Zimic, 2012)

2.7.1 MIT App Inventor

App Inventor je orodje za začetnike, ki omogoča lažji uvod v programiranje in ustvarjanje aplikacij. Razvijanje v programu poteka z gradniki, ki jih uporabnik postavlja na zaslon, s tem pa ustvarja uporabniški vmesnik in hkrati delovanje aplikacije. V App Inventorju se aplikacija ustvarja v dveh oknih. Prvo okno je izbira in postavitev komponent oz. gradnikov na zaslon. Drugo oknovsebuje bloke kode. Z izbiro teh dodajamo funkcionalnost h gradnikom. Vse skupaj spominja na sestavljanje sestavljanke. Aplikacija se lahko nato zažene z emulatorjem naprave Android ali pa neposredno na napravo Android. (Povzeto po: http://appinventor.mit.edu/explore/about-us.html, 24. 1. 2017)



Slika 4: Delovanje MIT App Inventorja, vir: [4]

2.7.2 Ionic

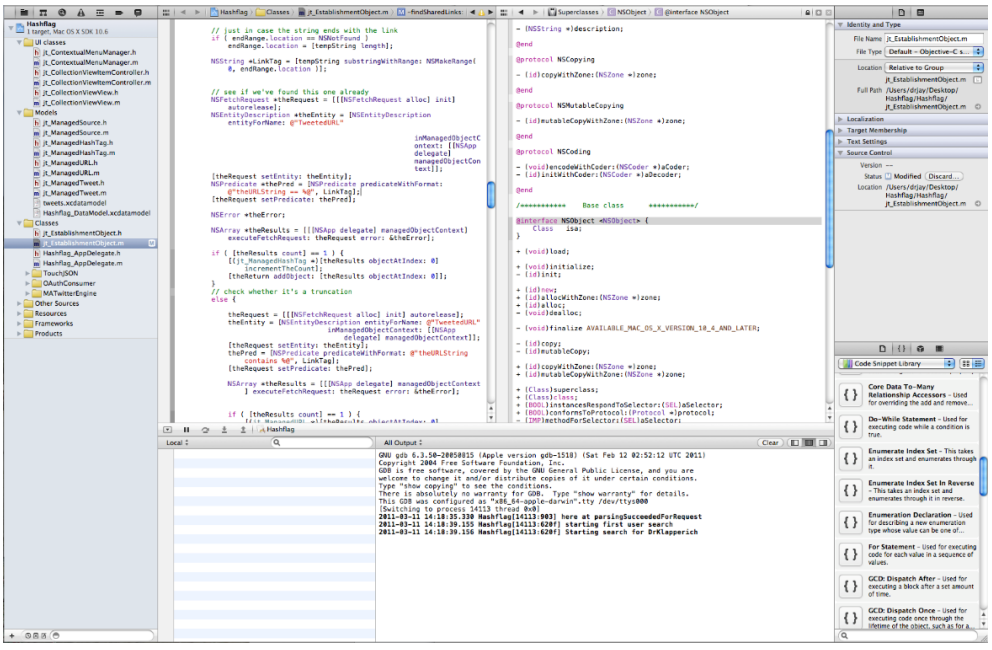
Slika 5: Ionic 2 logotip, vir: [5]

Ionic je hibridno orodje za izdelovanje mobilnih aplikacij, torej lahko z njim izdelamo aplikacije za Android, Apple iOS ter Windows Mobile. Narejeno je na ogrodjih AngularJS in Apache Cordove. Za prikaz vsebine uporablja internetne tehnologije, kot so CSS in HTML5. Ionic je npm modul in za svoje delovanje potrebuje Node.js. (Povzeto po Žnidar, 2016)

2.7.3 Xcode

Xcode je krajše ime za Xcode Developer Tools, ki ti omogoča, da razvijaš aplikacije osnovane s Cocoa, ki jih lahko oblikuješ tudi v navadnem urejevalniku kot Notepad++. Xcode pa je razvijalno okolje, ki ti omogoča lažje razvijanje aplikacij za Applove naprave.

Xcode je Applov IDE za razvijanje aplikacij. Podpira jezike kot so PHP, Swift, Objective-C, C++, C#, C, Java, Perl, JavaScript, Python in HTML. (Povzeto po Piper, 2009)

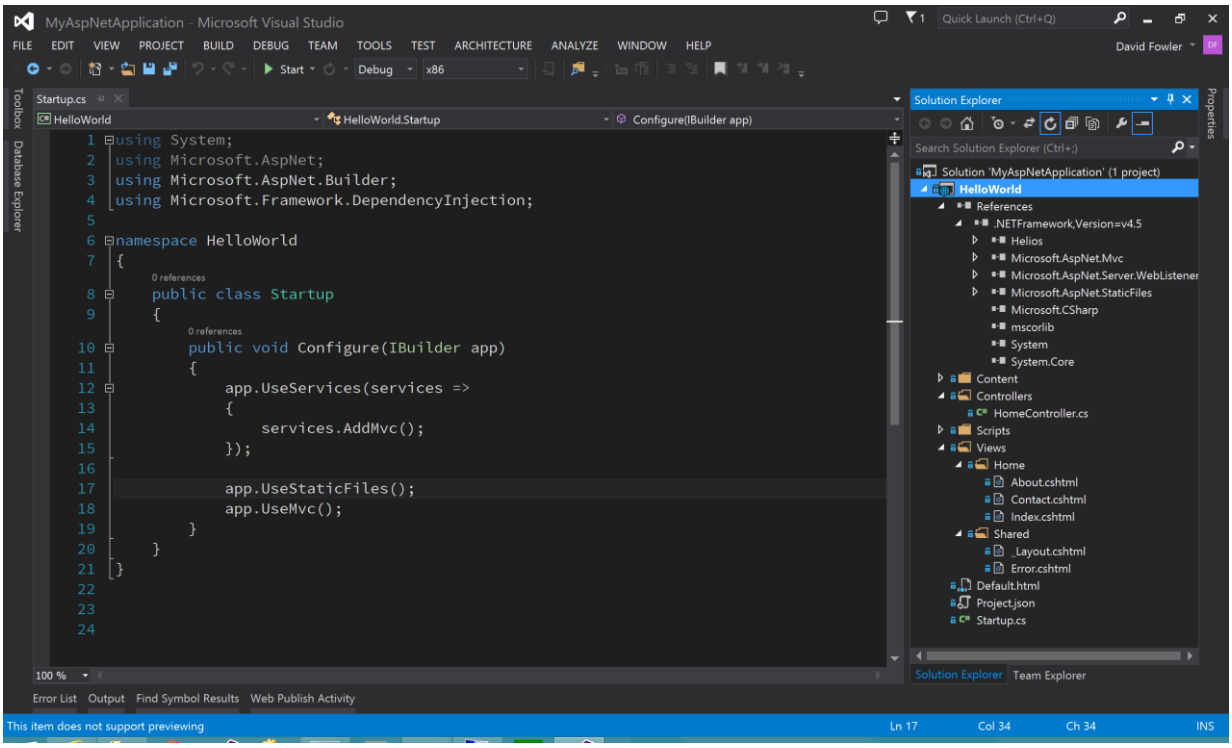


Slika 6: Xcode IDE, vir: [6]

2.7.4 Visual Studio

Visual Studio je set orodij za razvijanje ASP.NET Web aplikacij, XML Web servisov, računalniških in mobilnih aplikacij. Visual Studio združuje Visaul Basic, Visual C# in Visual C++, ki vsi uporabljajo isto okolje za razvijanje (IDE), ki omogoča deljenje posameznih orodij in mešanje prej naštetih jezikov, da pridemo do rešitve, ki nam ustreza. Ti jeziki vsi uporabljajo funkcionalnosti .NET Frameworka, ki omogoča dostop do tehnologij, ki olajšajo razvijanje aplikacij.

(Povzeto po: https://msdn.microsoft.com/en-us/library/fx6bk1f4%28v=vs.100%29.aspx, 29. 1. 2017)



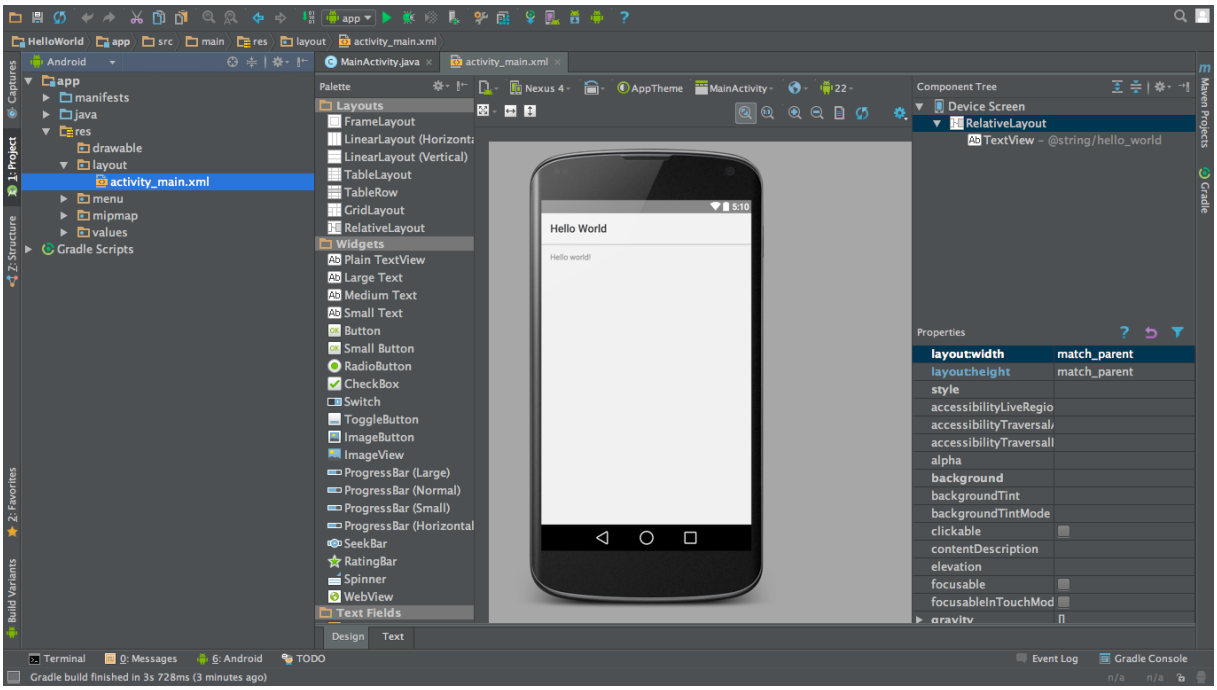
Slika 7: Visual studio IDE, vir: [7]

2.7.5 Android Studio

Android Studio je razvojno okolje, ki omogoča razvoj aplikacij za platforme Android. Temelji na razvojnem okolju IntelliJ IDEA, ki je eno izmed vodilnih na področju razvoja Java aplikacij. Ponuja orodja in pripomočke, ki omogočajo razvoj kvalitetnih aplikacij za vsako napravo z operacijskim sistemom Android. Vsebuje pregledovalnik kode, ki razvijalcu predlaga programsko kodo in napredno analizo le-te. Za preizkušanje aplikacije omogoča napreden zagon, ki zaznava spremembe in jih prikaže na napravi, brez ponovnega zagona aplikacije. Če nam naprava ni na voljo, lahko na Android Studio uporabimo vnaprej naložen emulator, ki

deluje kot samostojna naprava.

Android Studio je razvil Google, zato je tudi implementacija aplikacijskih programskih vmesnikov (API), ki jih ponuja Google, preprostejša. (Povzeto po Lukša, 2015)



Slika 8: Android Studio, vir: [8]

2.8 Programski in skriptni jeziki

2.8.1 Java

Java je objektno usmerjen programski jezik 3. generacije. Razvijalcem omogoča, da aplikacijo, ki jo napišejo v Javi, poganjajo na več platformah, ki podpirajo ta jezik. Uporablja se za vse vrste aplikacij, saj podpira veliko različnih platform, med drugim spletne aplikacije, programe namenjene za operacijske sisteme Linux, Windows in MacOS. Za razvijanje v jeziku Java potrebujemo razvojno orodje (JDK) in urejevalnik besedil oziroma kompleksno razvojno okolje. Programe, ki so prevedeni s prevajalnikom Java, je možno zagnati v virtualnem okolju (JVM). Zaradi tega Java ponuja platformno neodvisnost, vendar s počasnejšim časom delovanja. Trenutno je najnovejša različica programskega jezika Java 8, ki je izšla v letu 2014. (Programiranje v Javi, Viljan Mahnič, FRI Ljubljana, 2013)

2.8.2 Swift

Swift je programski jezik za razvoj iOS, macOS in ostalih Apple aplikacij. Temelji na osnovah programskih jezikov Objective-C in C, zato je v njem lažje pisati kodo, če poznamo katerega od teh jezikov. Ponuja lažji in bolj fleksibilen razvoj aplikacij kot njegov predhodnik (Objective-C). V osnovi si je zelo podoben s predhodnikom, vendar ponuja boljši pristop k postopkovnemu in objektno orientiranemu programiranju. V osnovi ima Swift varnostne funkcije, ki zmanjšajo možnost napak, ki jih lahko razvijalec napravi. Ponuja tudi opcijo »igrišč«, ki ima takojšen prikaz rezultatov ob pisanju kode, kar je mogoče z razvojnima okoljema Swift Playgrounds (iPad) ter Xcode (macOS). Najnovejša različica je Swift 3, ki razvijalcem omogoča enostaven razvoj aplikacij za platforme podjetja Apple. (Povzeto po: https://developer.apple.com/swift, 04. 2. 2017)

2.8.3 C#

C# je objektno orientiran programski jezik, ki ima veliko podobnosti z ostalimi jeziki 3. generacije (Java, C, C++). Namenjen je razvoju programov v okolju .NET, izdelavi namiznih, spletnih in mobilnih aplikacij. .NET ogrodje je knjižnica, ki ponuja veliko različnih rešitev pri razvoju programov. Različica .NET je Mono, ki omogoča izvajanje programov napisanih v C# na različnih platformah (Windows, Linux, Android), ker je napisan po specifikacijah, se lahko program neposredno prevede in izvaja na teh platformah, brez spreminjanja za specifično platformo. Najnovejša priporočena različica jezika C# je verzija 6.0, .NET ogrodja pa 4.6. (Programski jezik C#, Matija Lokar in Srečo Uranič, Kranj, oktober 2008)

2.8.4 JavaScript

JavaScript je dinamičen programski jezik, ki se uporablja večinoma za spletne aplikacije. Spada v skupino tolmačenih jezikov, kar pomeni, da se napisana koda izvršuje po vrsti in je ni potrebno prevesti v program pred zagonom. Standardiziran je po specifikaciji ECMAScript, ki z vsako verzijo doda funkcije, ki olajšajo razvoj aplikacij. Po sintaksi je podoben jeziku C#, Java. Za razliko od teh jezikov JavaScript nima strogo določnih tipov spremenljivk (int, string) in nima podpore za razrede (razen v novejši različici ECMAScript 6). S prihodom novejših verzij jezika pa je postalo možno razvijanje namiznih ter mobilnih aplikacij v jeziku JavaScript. (Povzeto po: https://developer.mozilla.org/en-US/docs/Web/JavaScript, 24. 1. 2017)

TypeScript je različica jezika JavaScript, ki omogoča lažji razvoj aplikacij. Vsa koda napisana v JavaScriptu, je izvedljiva v tej različici ter obratno, saj se TypeScript koda pretvori v preprosto JavaScript kodo, ki se nato poganja na vseh brskalnikih ter ostalih platformah, ki jo podpirajo. Prednost te različica je, da omogoča tipe spremenljivk (int, string …), kar zmanjša možnost napak pri razvoju in zagonu aplikacij. Vključuje tudi najnovejše standarde ECMAScript, ki ponujajo veliko izboljšav v primerjavi z osnovno različico JavaScripta. (Povzeto po: https://www.typescriptlang.org, 24. 1. 2017)

2.9 Aplikacijski programski vmesnik (API)

Aplikacijski programski vmesnik (Application programming interface) je skupek rutin, protokolov in orodij za izgradnjo programske opreme in aplikacij. API določi, kako naj programska oprema komunicira. Dodatno API olajšajo razvijanje programov s tem, ko priskrbijo vse osnovne dele. Programer potem vse dele le spravi skupaj. (Povzeto po: Hahn, 2015)

2.9.1 OpenData

Open data mednarodni projekt je izvedba ideje, da smo za določen sklop podatkov lastniki vsi prebivalci Slovenije in so posledično podatki brez licence. Iz teh podatkov je nastalo nekaj projektov, kot so tudi JSON API za nekatere prometne podatke, JSON API Statističnega urada Republike Slovenije, geolokacijsko podprti JSON API za vremenske podatke s strani ARSO itd. (Povzeto po: https://opendata.si, 8. 2. 2017)

2.9.2 OpenWeatherMap

Je spletna storitev, ki zagotavlja podatke o vremenu, vremenske napovedi. Podatke dobiva iz več različnih meteoroloških postaj, kot tudi iz letališč. Uporablja OpenStreetMap za prikaz vremenske slike. Ponuja tudi API, katerega rezultate lahko dobimo v JSON, XML ali HTML formatu. Zastonj lahko server kličemo do 60-krat na minuto, za več pa moramo plačati naročnino. (Povzeto po: https://openweathermap.org, 6. 2. 2017)

2.9.3 Google Calendar API

Slika 9: Google Calendar, vir: [9]

Je storitev, ki ti omogoča pokazati, ustvariti in spremeniti dogodke, kot tudi delati z drugimi objekti povezanimi s koledarji. Za povezavo do koledarja se je potrebno registrirati na Google Developer Console. S tem pridobimo ključ, ki omogoča uporabo API. (Povzeto po: https://developers.google.com/google-apps/calendar, 10. 2. 2017)

2.10 Node.js

Node.js je izvajalno okolje jezika JavaScript, ki temelji na pogonu V8, katerega je razvilo podjetje Google. To okolje deluje po asinhronem principu (več dogodkov naenkrat), zato je primerno za razvoj razširljivih spletnih aplikacij in strežniških rešitev. Ker te aplikacije uporabljajo JavaScript, jih je možno poganjati na operacijskih sistemih OS X, Windows ter Linux. Node.js vsebuje sistem modulov oz. paketov imenovan npm, ki je največji sistem odprtokodnih knjižnic na svetu. (Povzeto po Hahn, 2015)

2.11 JSON

JSON, je format za izmenjavo podatkov. Temelji na jeziku JavaScript, vendar je še vedno tekstovni format, ki je neodvisen od programskega jezika. Uporablja se lahko skupaj z jeziki Java, C, C++, C#, Python ter ostalimi programskimi jeziki. Strukturiran je na dva načina:

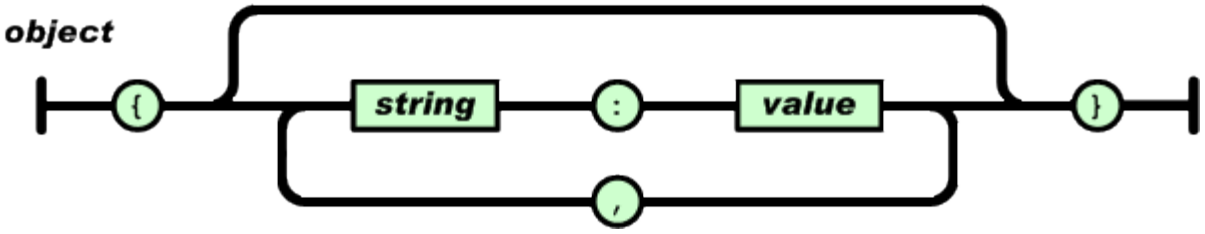
- Zbirka parov ime/vrednost – v programskih jezikih je to objekt ali asociativna tabela
- Urejen seznam vrednosti – v programskih jezikih je to polje, seznam

Ti strukturi sta univerzalni podatkovni strukturi. Podpirajo ju skoraj vsi moderni programski jeziki.

V JSON sta ti strukturi prikazani tako:

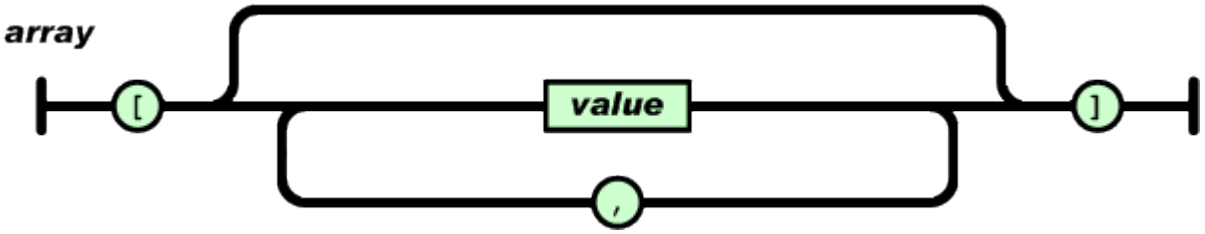
Objekt, ki vsebuje neurejen seznam parov ime/vrednost. Prične se z »{«, ter konča z »}«. Imenu

sledi »:«, vsakemu paru pa »,«.



Slika 10: JSON objekt, vir: [10]

Seznam je urejena zbirka vrednosti. Prične se z »[», ter konča z »]«. Vrednosti so med seboj ločene z »,«. Podobno strukturo imajo ostali programski jeziki.



Slika 11: JSON niz, vir: [10]

Vrednost v objektu ali seznamu je *besedilo* (string), *število* (number), *objekt*, *seznam*, vrednosti *true*, *false* ali *null*. Vsaka vrsta je lahko gnezdena. (Povzeto po: http://www.json.org, 12. 02. 2017)

2.12 Podatkovne baze

Podatkovne baze so zbirke med seboj pomensko povezanih podatkov. So oblika podatkovnega sistema, katerega namen je upravljanje in obdelava podatkov. Dostop do teh podatkov je omogočen s centraliziranim sistemom za upravljanje podatkovnih baz (SUPB). (Povzeto po: Mihelič, 2014)

2.12.1 MySQL

MySQL je odprtokoden in eden izmed bolj uporabljenih SUPB, ki ga je razvilo podjetje Oracle Corporation. Napisan je v jezikih C in C++ ter deluje na več operacijskih sistemih.

Podatki se shranjujejo v tabele, za dostop do baze in zapisovanje vanjo, se uporablja strukturiran jezik SQL. Pred uporabo podatkovne baze je potrebno ustvariti shemo baze, ki vsebuje podatke o tabelah in razmerja med njimi.

MySQL uporablja relacijski model podatkov, zato ga uvrščamo med RDBMS. To pomeni, da ima vsaka vrstica v tabeli svoj unikaten ključ. Tabeli se med seboj povežeta z uporabo

unikatnega ključa v eni tabeli in povezavo na ta ključ v drugi tabeli, imenovano tuji ključ. (Povzeto po: Mihelič, 2014)

MySQL

```
INSERT INTO users (user_id, age, status)
VALUES ('bcd001', 45, 'A')
```

```
SELECT * FROM users
```

```
UPDATE users SET status = 'C'
WHERE age > 25
```

Slika 12: MySQL stavki za poizvedbo, vir: [11]

2.12.2 MongoDB

MongoDB je odprtokodna podatkovna baza, razvilo jo je podjetje MongoDB, Inc. Za dostop do baze in iskanje po njej, se uporablja nestrukturiran jezik (NoSQL), ki ne omogoča relacijskih povezav med tabelami. Podatki so shranjeni v obliki JSON dokumentov. To omogoča dinamičen razvoj aplikacij, saj za uporabo baze ne potrebujemo določene sheme. Zaradi tega lahko podatke in njihovo strukturo ves čas spreminjamo. Podatki, ki se navezujejo drug na drugega, so shranjeni skupaj, kar omogoča hiter dostop do njih. Prednost te vrste podatkovnih baz je, da jih lahko hitro razširimo. (Povzeto po: Hahn, 2015)

MongoDB

```
db.users.insert({
  user_id: 'bcd001',
  age: 45,
  status: 'A'
})
```

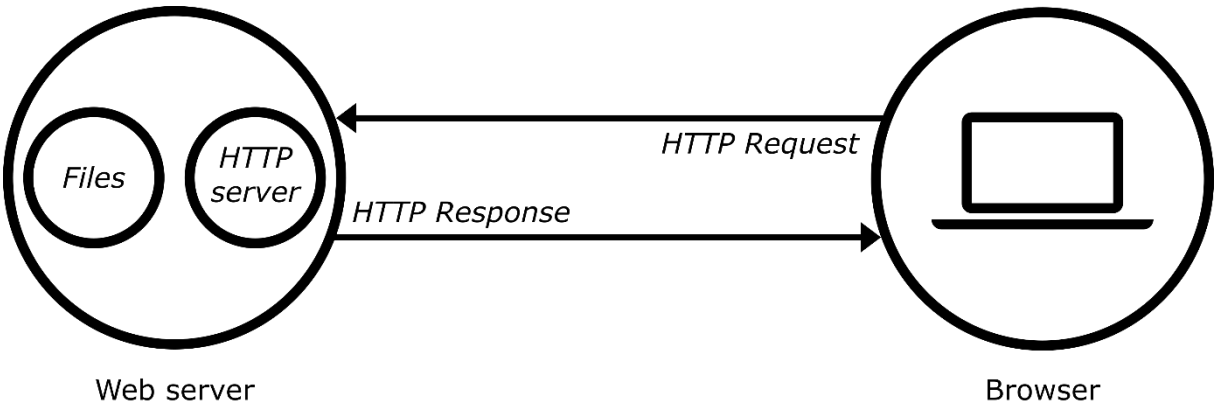
```
db.users.find()
```

```
db.users.update(
  { age: { $gt: 25 } },
  { $set: { status: 'C' } },
  { multi: true }
)
```

Slika 13: MongoDB stavki za poizvedbo, vir: [11]

2.13 Spletni strežniki

Spletni strežnik je programska oprema, ki sprejema HTTP zahteve, nato pa se glede na vrsto te zahteve odloči, kakšen odgovor bo posredoval nazaj. HTTP odgovori so večinoma HTML dokumenti, slike, lahko pa tudi ostale datoteke. (Povzeto po: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server, 12. 2. 2017)



Slika 14: Delovanje spletnega strežnika, vir: [12]

2.13.1 Apache

Apache je odprtokodni spletni strežnik, ki je bil razvit leta 1995. Ime je bilo izbrano kot izraz spoštovanja avtohtonemu ameriškemu plemenu Apači. Namenjen je uporabi na različnih operacijskih sistemih, najpogosteje je uporabljen na sistemih Unix. Podpira uporabo modulov, ki dodajo funkcionalnost in izboljšajo delovanje strežnika. Večinoma se uporabljajo za dodajanje HTTPS ali spreminjanje URL. Moduli so lahko razviti v različnih programskih jezikih. (Povzeto po: https://httpd.apache.org/ABOUT_APACHE.html, 4. 2. 2017)

Slika 15: Apache logotip, vir: [13]

2.13.2 NGINX

NGINX je odprtokodni projekt, ki se je začel razvijati leta 2002. Spletni strežnik NGINX je lahko uporabljen tudi kot spletni predpomnilnik za dokumente, kot so slike, HTML strani ipd. Omogoča tudi uravnavanje obremenitve strežnika, kar pomeni, da spletne zahteve razporedi med več strežnikov in s tem zmanjša zmožnost za preobremenitev. Lahko se ga uporablja na več operacijskih sistemih. Poleg zgoraj navedenih funkcij omogoča še nastavitev poštnega strežnika, spreminjanje URL in dodajanje možnosti HTTPS. (Povzeto po: https://nginx.org/en/, 12. 2. 2017)

Slika 16: Nginx logotip, vir: [14]

2.14 GPS

GPS je satelitski navigacijski sistem, ki je financiran s strani U. S. Ministrstva za obrambo, ki ga tudi upravlja. Sestavljen je iz najmanj 24. satelitov, ki potujejo okoli Zemlje vsak dvakrat dnevno. Na satelitih je nameščena atomska ura in satelit neprestano oddaja čas po svoji uri in podatke o svoji lokaciji.

(Povzeto po: http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html, 25. 1. 2017)

3 CILJI IN METODE RAZISKOVANJA

Glede na izbrane hipoteze, sva uporabila dve metodi raziskovanja in sicer kvantitativno raziskovanje za potrjevanje prve hipoteze ter kvalitativno raziskovanje za preostale tri. Prva hipoteza je zahtevala pridobivanje večjega števila podatkov za izračun statistike - kolikokrat je bilo prepoznavanje govora uspešno ali neuspešno. Ostale tri pa sva lahko potrdila oz. ovrgla že z opazovanjem med razvojem aplikacije.

3.1 Pregled že obstoječih pametnih asistentov in sistemov za prepoznavo govora

Na spletu obstaja veliko pametnih asistentov, ki se uporabljajo na različnih področjih. Osredotočila sva se predvsem na tiste, do katerih imava dostop kot dijaka ter uporabnika pametnih mobilnih naprav. Pametni asistenti, pregledani v tej raziskovalni nalogi, so Siri, Cortana, Alexa in Google Asistent. Namen raziskovalne naloge je bil poizvedeti, kako delujejo, spoznati tehnologije, ki jih uporabljajo in katere funkcije nam ponujajo kot uporabnikom.

Predvidevala sva, da vse zgoraj omenjene platforme delujejo na podoben princip ter ponujajo podobne storitve. Ker pa vsaka izmed platform pripada različnemu podjetju in se izvaja na operacijskem sistemu, ki ga je razvilo to podjetje, ima vsaka platforma svoje storitve in funkcije, ki delujejo na vseh pametnih napravah razvitih s strani tega podjetja.

Sistemi za prepoznavo govora pa za razliko od pametnih asistentov, nimajo možnosti odločanja glede na pridobljene podatke, temveč samo zapišejo te podatke, ki jih uporabnik izgovori. Zato tudi niso odvisni od izbire pametnega asistenta, v katerem so implementirani, vendar so večinoma uporabljeni v tistih, ki so razviti s strani istega podjetja. Potemtakem bi lahko asistentka kot je Cortana, uporabljala sistem PocketSphinx za prepoznavanje govora. Kljub temu bi delovala na enak način, saj sistem za prepoznavo govora ne vpliva na sistem za sprejemanje odločitev.

3.2 Uvod v izdelavo

Preden sva pričela z izdelavo, sva raziskala najbolj primerno platformo pametnih telefonov in njihove razvojne platforme. Izbirala sva med 3 glavnimi: Android, Apple iOS in Windows Mobile. Ker oba raziskovalca uporabljava pametne telefone s platformo Android, sva se odločala med razvojnim okoljem za Android (Android Studio) in razvojnim okoljem, ki podpira razvoj za več sistemov hkrati (Ionic, React Native).

3.3 Začetek izdelave

Zaradi že prej pridobljenega znanja razvoja aplikacij v okolju Ionic, sva se odločila za razvoj v tem okolju. Kmalu po začetku uporabe, sva opazila, da bo aplikacija potrebovala dodatne storitve, ki jih z uporabo tega razvojnega okolja nisva mogla pridobiti. Tako sva izključila možnost izdelave aplikacije, ki bi podpirala več mobilnih platform.

Izbrala sva razvojno okolje Android Studio. Zaradi pomanjkanja znanja o programskem jeziku Java, sva imeli nekaj težav pri razvoju aplikacije.

3.3.1 Spletna storitev za pridobivanje podatkov o dogodkih

Z uporabo spletnega ogrodja Node.js, sva izdelala spletno storitev, ki pridobi podatke s spletne strani OpenData. Pridobljeni podatki so v obliki JSON. Nekaterih podatkov ne potrebujeva, zato sva jih spremenila in jih takšne shranila v svojo podatkovno zbirko, ki uporablja tehnologijo MongoDB. Strukturo teh podatkov sva prilagodila najinim zahtevam in s tem omogočila lažje branje podatkov na aplikaciji.

3.3.2 Prepoznavanje govora

V tem koraku sva dodala knjižnico za prepoznavanje govora. Uporabila sva PocketSphinx, ki nam omogoča neprekinjeno poslušanje našega govora. Ko zazna ključno frazo, pokliče funkcijo, ki zažene Googlov sistem za prepoznavanje govora. Nato Googlov sistem prepozna naše izgovorjene besede, ki jih posreduje aplikaciji. Za uporabo dveh sistemov za prepoznavanje govora sva se odločila, saj nam Googlov sistem, ki je bolj izpopolnjen kot PocketSphinx, ne omogoča konstantnega poslušanja. Poleg tega pa PocketSphinx za boljše delovanje potrebuje svoj akustični in jezikovni model. To pri več uporabnikih aplikacije ni praktično.

3.3.3 Odločanje in koledar

V naslednjem koraku sva v aplikaciji naredila osnove odločanja, da program pozna razliko, če npr. rečemo »koledar« ali »vreme« in pokliče besedam primerno funkcijo, ki v naslednjih verzijah aplikacije tudi pove njej primeren odgovor. Za tem sva dodala še funkcionalnosti, ki jih ponuja Google Calendar API. Pridobimo lahko dogodke, vnesene v Googlov koledar, njihovo lokacijo − če je ta navedena in tudi vse sodelujoče v tem dogodku.

Odločanje poteka tako, da se najprej vse izgovorjene besede razdelijo v posamezno besedo, ki se doda v skupen niz. Nato z uporabo »if« stavkov preverjamo ali so v tem nizu podane ključne besede, kjer je prva beseda vrsta podatka, npr. dogodek, vreme, stanje avtomobila. Vse besede

za tem so parametri, ki se glede na vrsto podatka razlikujejo.

3.3.4 Dogodki in vreme

V tem koraku sva uporabila API, ki ni povezan z Google storitvami. Za vreme sva uporabila OpenWeatherMap API, ki za delovanje potrebuje ključ. Ta se pridobi z registracijo na spletni strani. Podatki o vremenu se pridobijo tako, da uporabnik pove ime mesta, o katerem želi izvedeti vremenske podatke. S strani se vrnejo podatki o temperaturi, zračnemu pritisku, vlažnosti in kratkemu opisu vremena. Zraven teh podatkov je podan še čas.

Za pridobivanje dogodkov, sva uporabila svoj API, ki s spletne strani OpenData pridobi podatke o stanju na slovenskih cestah. Nato se ti podatki na naši strani obdelajo, tiste podatke, ki jih ne potrebujemo pa odstranimo iz končnega izpisa.

3.3.5 Podatki o stanju avtomobila

Izdelala sva dodatno spletno storitev, ki omogoča beleženje trenutnega stanja avtomobila. To storitev poganja Node.js. Uporablja podatkovno bazo MongoDB, saj so prejeti podatki zapisani v obliki JSON in ni potrebe po relacijskem modelu baze. Ob klicu na API se pridobijo podatki o času zapisa, številu vrtljajev motorja na minuto, hitrosti avtomobila, obremenitvi motorja in njegovi temperaturi.

Podatki se na strežniku posodabljajo vsako sekundo, zato lahko uporabnik pridobi najnovejše podatke z zelo kratkim zamikom, kar mu omogoča preprost vpogled o stanju med vožnjo.

Za vpogled v te podatke potrebujemo dodatna pripomočka, bralnik podatkov o avtomobilu OBD in mikroračunalnik Raspberry Pi, ki te podatke pošlje na strežnik. Za pošiljanje potrebuje aktivno internetno povezavo, ki se doseže z uporabo dostopne točke na telefonu.

4 REZULTATI IN RAZPRAVA

4.1 Izbira orodij

Preden sva se lotila izdelave aplikacije, sva morala izbrati orodja, ki so najbolj primerna za to, kar najina aplikacija ponuja. Izbrati sva morala pravi spletni strežnik, podatkovno bazo in storitve, ki naredijo aplikacijo takšno kot je.

4.1.1 Spletni strežnik

Za izdelavo API sva izbrala Node.js, zaradi zanimanja o novih tehnologijah in uporabe programskega jezika JavaScript. Izbirala sva še med programskim jezikom PHP in ogrodjem Laravel, vendar sva se zaradi preprostejše izvedbe API, odločila za Node.js.

Na podlagi zgoraj omenjene odločitve, sva izbrala spletni strežnik NGINX, saj se v kombinaciji z Node.js odziva bolje. Kljub temu bi se lahko odločili za samostojno uporabo Node.js, saj je le-ta sam po sebi že spletni strežnik. Zaradi uporabe HTTPS, ki je v današnjih časih vedno bolj priporočljiv, sva dodala še NGINX.

Naš API torej teče v programu Node.js, hkrati pa za varen dostop do njenih storitev skrbi NGINX, ki preusmeri vse HTTP zahteve na varnejšo HTTPS stran.

4.1.2 Podatkovna baza

Pri izbiri podatkovne baze sva najprej gledala na naše prejšnje izkušnje, saj sva v preteklih letih uporabljali relacijske podatkovne baze, ki uporabljajo strukturiran jezik SQL (MySQL). Po pregledu spletne storitve OpenData in njenih podatkov, sva ugotovila, da uporaba relacijske baze v našem primeru ne bi bila učinkovita. Izvedba z njo bi bila možna, vendar bi morala podatke, ki so že bili združeni v smiselno obliko, razčleniti in razvrstiti v relacijsko shemo. Hkrati pa so podatki podani v obliki JSON, zato sva se odločila za uporabo podatkovne baze MongoDB. Vneseni podatki so zato brez relacij, vendar imajo vseeno smiselno strukturo, ki omogoča preprosto branje in urejanje.

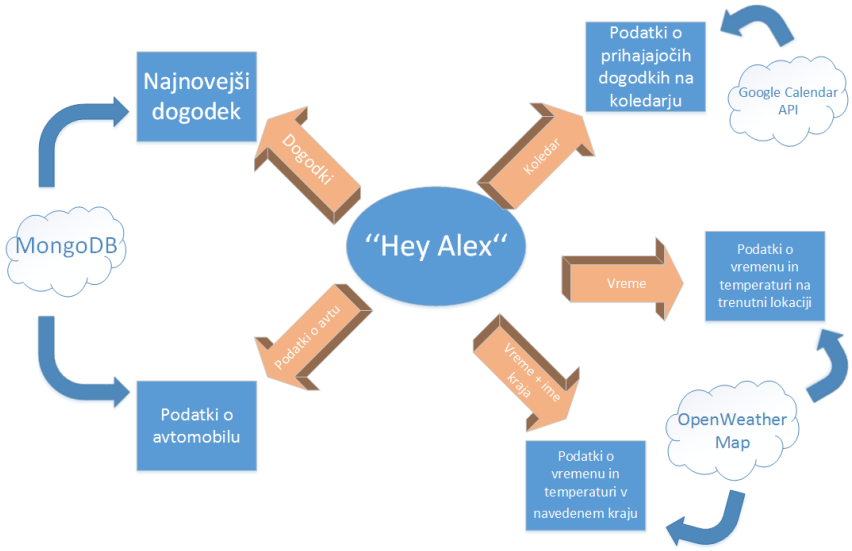
4.1.3 Aplikacija

Na začetku sva bila v dilemi med izdelavo aplikacije za eno platformo in med izdelavo hibridne aplikacije. Po izdelanem prvem prototipu sva ugotovila, da čeprav aplikacija razvita v ogrodju Ionic deluje na več platformah, to ne odtehta prednosti, ki jih ponuja razvoj aplikacije v razvojnem okolju Android Studio. Zaradi uporabe nekaterih storitev, ki so vezane na Android, je bila ta izbira primernejša. Na koncu sva razvila stabilno aplikacijo, ki združuje knjižnice in

storitve Google, API za vreme in dogodke ter stanje o avtomobilu, če je prisotna še dodatna strojna oprema.

Pri prepoznavanju govora sva imela na voljo več sistemov za prepoznavanje govora (STT). Izbrala sva dva najbolj dokumentirana, ki sta hkrati najbolj popularna sistema. To sta PocketSphinx in Google STT. Pri implementaciji obeh, sva morala predvsem paziti, da se nista prepoznavna sistema prekrivala. To pomeni, da sva ob zagonu Google STT ustavila že delujoč sistem prepoznavanja govora, saj Android ne dopušča sočasne uporabe mikrofona s strani dveh storitev.

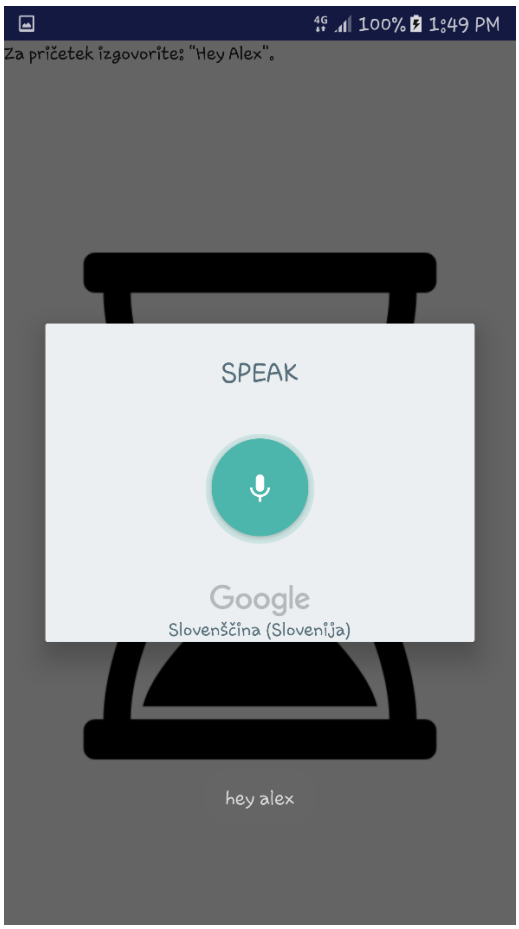
4.2 Delovanje



Slika 17: Osnovni prikaz delovanja aplikacije, vir: lasten

Aplikacija ob zagonu prične poslušati za ukaz »Hey Alex« (Slika 18). Ko uporabnik izreče ukaz in se ta pravilno zazna, se nato odpre okno, ki pričakuje enega izmed 5. ukazov (Slika 19), ki so navedeni v oranžnih puščicah na sliki (Slika 17). Ob ukazu »podatki o avtu«, dobimo nazaj podatke o avtomobilu, če so ti na voljo. Ko uporabnik izgovori »dogodki«, se pomočjo API podatki naložijo v najino podatkovno bazo MongoDB in iz nje v aplikacijo, od koder jih TTS prebere. Pri ukazu »vreme« dobimo podatke o vremenu preko API storitve s strani OpenWeatherMap za trenutno lokacijo, če pa temu ukazu dodamo še kraj, pa dobimo podatke o vremenu za ta kraj. Če je ukaz »koledar« pa dobimo podatke o našem naslednjem dogodku, s pomočjo Google Calendar API. Te nato TTS prebere, da lahko to seveda tudi slišimo.

Slika 18: Aplikacija čaka, da uporabnik reče "Hey Aley", vir: lasten

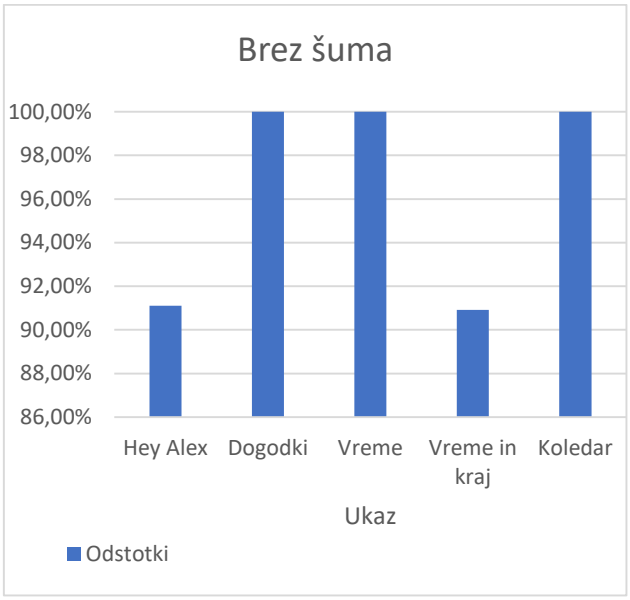


Slika 19: Uporabnik je rekel "Hey Alex" in sedaj posluša Google STT in čaka na ukaz, vir: lasten

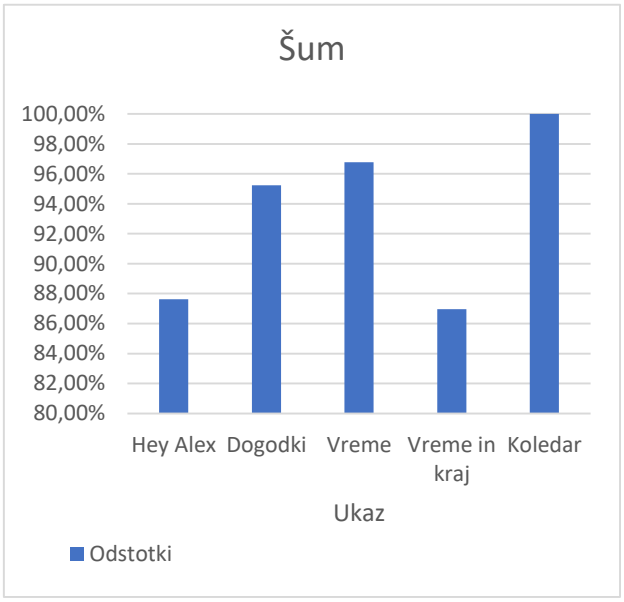
4.3 Ugotovitve

Ko sva izdelala aplikacijo ter jo testirala, sva prišla do odgovorov na postavljene hipoteze. Hipoteze, **prepoznavanje govora bo uspešno kljub prisotnosti šumov, zvokov iz okolice**, sva se lotila s testiranjem aplikacije v okolju brez hrupa in v okolju s šumom in sicer v obliki glasbe in občasnega zmerno tihega govorjenja. Tako sva opazila, da je bilo največ težav pri prepoznavanju ukazov, kjer uporabnik zahteva vreme za določen kraj, saj je pri kakšnem zahtevnejšem ali pa predvsem kraju, ki je manjši, prepoznalo napačen kraj. Opazila sva tudi, da orodje za prepoznavanje PocketSphinx (ukaz »hey alex«), prepoznava govor malo slabše kot Google STT (ukazi: »dogodki«, »vreme«, vreme in kraj, »koledar«), takrat ko je bil šum in takrat, ko ga ni bilo. Razlika med poskusoma je opazna, vendar ni zelo velika. Pri poskušanju s šumom (Graf 1), so bili ukazi dogodki, vreme in koledar vedno prepoznani, medtem ko začetni ukaz »hey alex« in ukaz za vreme za določen kraj v malo več kot 90 odstotkih. V okolju s šumom (Graf 2), je edini vedno prepoznan ukaz, ukaz za koledar. Ukaza za dogodke in vreme

sta prepoznana skoraj vedno, spet sta najmanjkrat prepoznana ukaza »hey alex« in vreme ter ime kraja. Aplikacija v okolju, kljub prisotnosti šumov iz okolice, zazna prvi ukaz v skoraj devetih primerih od desetih, za tem pa je povprečje prepoznanih ukazov 95 %. Zdelo se nama je, da je ta odstotek dovolj visok, da to hipotezo **potrdiva**.



Graf 1: Poskušanje brez šuma



Graf 2: Poskušanje s šumom

Aplikacijo lahko razdelimo na dva glavna dela.

V prvem delu se izvaja proces prepoznavanja govora, ko uporabnik govori ukaze napravi. Razvila sva del z uporabo angleške različice prepoznavanja govora. To pomeni, da je bilo možno uporabljati aplikacijo le v angleščini. Po preizkušanju sva ugotovila, da je na voljo tudi slovenska različica. Zaradi tega se je izboljšala natančnost ukazov, saj aplikacija z angleško različico prepoznavanja ni pravilno zaznala imen slovenskih krajev.

Drug del aplikacije se navezuje na sintezo govora. Pridobljeni podatki glede na uporabnikov ukaz se preko Google TTS predvajajo. Uporabila sva angleško različico TTS modula, kjer v začetku ni bilo problemov. Na tej stopnji so bili na voljo le ukazi za vreme in koledar, ki pa so v večini primerov v angleškem jeziku. Ko sva poskušala nadgraditi aplikacijo na slovensko različico, sva ugotovila, da modul za izgovorjavo besed, v našem maternem jeziku ne obstaja. Branje dogodkov, za katere pridobimo vse podatke v slovenskem jeziku, je torej nesmiselno. **Hipotezo, uporaba slovenskega jezika ne bo ovirala uporabe aplikacije, sva zaradi zadnje**

ugotovitve ovrgla, saj uporaba slovenskega jezika v aplikaciji in pridobljenih podatkih ovira pričakovano delovanje.

V končni različici aplikacija ponuja podatke o vremenu, dogodkih na cesti, prihajajočih dogodkih na koledarju in trenutno stanje avtomobila. Za pridobitev podatkov mora imeti uporabnik vključeno internetno povezavo. Ker so velikosti zgoraj navedenih podatkov majhne, se ti pridobijo hitro in nemoteno tudi na počasnejši povezavi. Ko sva preizkušala aplikacijo, nikoli nisva imela težav s pridobivanjem podatkov, zato sva hipotezo, pridobivanje podatkov na mobilni telefon bo potekalo nemoteno, potrdila.

Hipotezo, aplikacijo je možno razviti z uporabo že obstoječih storitev, sva potrdila. Storitve, ki omogočajo dostop do podatkov, že obstajajo in imajo urejeno dokumentacijo. Za prepoznavanje govora sva v aplikacijo vključila knjižnico PocketSphinx in razvojno ogrodje Google STT. Podatke sva pridobila iz različnih API, uporabila sva OpenWeatherMap za podatke o vremenu, Google Calendar API za prihajajoče dogodke v našem koledarju in OpenData API za trenutne dogodke na cesti. Storitev Google TTS, ki je prav tako dostopna razvijalcem aplikacij, nam omogoča sintezo govora.

Kljub temu da sva ustvarila svojo spletno storitev, ki uporablja že obstoječo storitev za pridobivanje dogodkov na cesti, sva hipotezo potrdila. Z uporabo lastne storitve, sva si zgolj olajšala delo, saj bi lahko aplikacijo izdelali s prej omenjeno storitvijo.

5 ZAKLJUČEK

V raziskovalni nalogi se je pojavljalo veliko različnih izzivov. Teh sva se lotila z zagnanostjo in željo po rešitvi problema. Večinoma so bili to izzivi organizacijske narave, saj velikokrat nisva vedela, kaj lahko vključimo v aplikacijo in čemu dati prioriteto. Pojavljali so se tudi programerski izzivi, saj včasih kljub dokumentaciji storitve, ta ni delovala po pričakovanjih. Nekaterih izzivov nama ni uspelo v celoti razrešiti, vendar sva vseeno razvila delujočo aplikacijo. Z njo lahko uspešno pridobivamo podatke o vremenu, trenutnih dogodkih na cesti, dogodkih na spletnem koledarju in trenutnem stanju avtomobila, vendar lahko slednje pridobivamo le z uporabo dodatnega modula Raspberry Pi. Največji dosežek te aplikacije je glasovno upravljanje, kar bi zmanjšalo stopnjo odvračanja voznika od vožnje.

Skozi celotno raziskovalno nalogo sva pridobila veliko znanja na področju postavitve in uporabe spletnih storitev, ki nam ponujajo podatke, uporabe razvojnih okolij ter področju strukture, prepoznavanja in sinteze govora. Izboljšala sva tudi najine organizacijske sposobnosti.

Aplikacijo je v tej fazi že možno uporabljati, saj uporabniku nudi koristne informacije. Vendar sva mnenja, da bi jo bilo potrebno za pogostejšo uporabo izboljšati in dodati še kakšno funkcijo, kot dodajanje dogodkov v spletni koledar z glasovnim upravljanjem, opozorilo na policijske kontrole, izračun trajanja poti potovanja glede na trenutne podatke o avtomobilu in razmere na cesti.

6 ZAHVALA

Rada bi se zahvalila mentorju Islamu Mušiću, za pomoč pri izdelavi raziskovalne naloge, ter vodenju ob trenutkih, ko nisva vedela, kako nadaljevati. Velika zahvala gre tudi somentorju Simonu Konečniku, za pomoč pri ustvarjanju dokumentacije in tudi Urošu Remenihu, za podajanje tehničnih informacij in svetovanju. Prav tako se zahvaljujeva dr. Nataši Meh Peer, za lektoriranje celotne raziskovalne naloge in najinim staršem, ki so naju podpirali in spodbujali. Zahvala gre tudi Elektro in računalniški šoli Velenje, ki nama je omogočila izdelavo raziskave.

7 VIRI IN LITERATURA

7.1 Knjižni viri

- Bučković, D. *Glasovno vodenje računalnika s pomočjo sistema za razpoznavanje govora Sphinx-4, diplomsko delo,* UNIVERZA V LJUBLJANI FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO, 2013
- Lukša, B. *Razvoj mobilne aplikacije za sporočanje, diplomsko delo,* UNIVERZA V MARIBORU, FAKULTETA ZA ELEKTROTEHNIKO, RAČUNALNIŠTVO IN INFORMATIKO, 2015
- Hahn, M. *Spletni poslovni sistem, diplomsko delo,* UNIVERZA V MARIBORU FAKULTETA ZA ELEKTROTEHNIKO, RAČUNALNIŠTVO IN INFORMATIKO, 2015
- Mihelič, M. *Migracija podatkov iz podatkovne zbirke MySQL na Percona Server in MariaDB, diplomsko delo,* UNIVERZA V LJUBLJANI, FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO, 2014
- Čulibrk, M. *Spletne storitve na Windows Mobile in Android, diplomsko delo,* UNIVERZA V MARIBORU FAKULTETA ZA ELEKTROTEHNIKO, RAČUNALNIŠTVO IN INFORMATIKO, 2010
- Kačič, Z. 1995 *Komunikacija človek-stroj,* Založniško tiskarska dejavnost Tehniških fakultet, Maribor
- Mežnar, S. *Razvoj mobilne aplikacije na platformi android, diplomsko delo,* UNIVERZA V LJUBLJANI FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO, 2012
- Zimic, M. *Medplatformski razvoj mobilnih aplikacij, diplomsko delo,* UNIVERZA V LJUBLJANI FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO, 2012
- Piper, I. *Learn Xcode Tools for Mac OS X and iPhone Development,* 1. izd, New York: Springer-Verlag, 2009
- Žnidar, K. *Razvoj mobilne hibridne aplikacije na platformi Apache Cordova/Ionic za prikaz vremena, magistrsko delo,* UNIVERZA V MARIBORU FAKULTETA ZA ELEKTROTEHNIKO, RAČUNALNIŠTVO IN INFORMATIKO, 2016

7.2 Spletni viri

www.policija.si, 12. 2. 2017

https://jasperproject.github.io/documentation/configuration/, 26. 1. 2017

https://developer.amazon.com/alexa, 28. 1. 2017

https://assistant.google.com, 25. 1. 2017

https://support.microsoft.com/sl-si/help/17214/windows-10-what-is, 25. 1. 2017

http://appinventor.mit.edu/explore/about-us.html, 24. 1. 2017

https://developer.apple.com/swift, 04. 2. 2017

https://developer.mozilla.org/en-US/docs/Web/JavaScript, 24. 1. 2017

https://www.typescriptlang.org, 24. 1. 2017

https://opendata.si, 08. 2. 2017

https://openweathermap.org, 06. 2. 2017

https://developers.google.com/google-apps/calendar, 10. 2. 2017

http://www.json.org, 12. 2. 2017

https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server, 12. 2. 2017

https://httpd.apache.org/ABOUT_APACHE.html, 4. 2. 2017

https://nginx.org/en/, 12. 2. 2017

http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html, 25. 1. 2017

7.3 Viri slik

[1], http://cmusphinx.sourceforge.net/wiki/tutorialconcepts, 24. 1. 2017

[2], https://www.yonomi.co/blog/hey-alexa-meet-yonomi, 24. 1. 2017

[3], https://source.android.com/source/index.html, 11. 1. 2017

[4], http://appinventor.mit.edu/explore/content/what-app-inventor.html, 24. 1. 2017

[5], https://www.linkedin.com/pulse/nouvelle-formation-ionic-2-le-nouveau-framework-mobile-j%C3%A9r%C3%A9mie-baldy, 11. 1. 2017

[6], http://cdn.arstechnica.net/2011/03/11/Interface_clutter_full.png, 29. 1. 2017

[7], https://ijlkgq.dm1.livefilestore.com/y2p3Y25S2SL-HhP1tKxyZ6xzfLzr8i7r6hj57hxvdxrjrK0rhd-_cPawxWMgBrBmuqAM6RPbeZuexuYTUKq5Q__ee0CDEkidqkixOUlWgeUUJs/VsAndKNet45.png?psid=1, 29. 1. 2017

[8], http://www.journaldev.com/8988/android-hello-world-application-tutorial-using-android-

studio, 29. 1. 2017

[9], https://www.cronofy.com/google-calendar-api/, 12. 2. 2017

[10], http://www.json.org/, 12. 2. 2017

[11], https://www.mongodb.com/compare/mongodb-mysql, 12. 2. 2017

[12], https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server, 12. 2. 2017

[13], https://en.wikipedia.org/wiki/Apache_HTTP_Server, 12. 2. 2017

[14], https://nginx.org/en/, 12. 2. 2017