



# УНИВЕРЗИТЕТ У НОВОМ САДУ

## ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

НОВИ САД

Департман за рачунарство и аутоматику

Одсек за рачунарску технику и рачунарске комуникације

## ЗАВРШНИ (BACHELOR) РАД

Кандидат: Мирољуб Стефановић

Број индекса: Е12369

Тема рада: Имплементација интерактивне апликације са подршком за препознавање и могућност управљања гласом на Андроид платформи

Ментор рада: проф. др Јелена Ковачевић

Нови Сад, 22.06.2012. година





## КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:		
Идентификациони број, ИБР:		
Тип документације, ТД:	Монографска документација	
Тип записа, ТЗ:	Текстуални штампани материјал	
Врста рада, ВР:	Завршни (Bachelor) рад	
Аутор, АУ:	Мирослав Стефановић	
Ментор, МН:	др Јелена Ковачевић, ред. проф.	
Наслов рада, НР:	Имплементација интерактивне апликације са подршком за препознавање и могућност управљања гласом на Андроид платформи	
Језик публикације, ЈП:	Српски / латиница	
Језик извода, ЈИ:	Српски	
Земља публиковања, ЗП:	Република Србија	
Уже географско подручје, УГП:	Војводина	
Година, ГО:	2012	
Издавач, ИЗ:	Ауторски репринт	
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/ слика/графика/прилога)	7/33/0/0/15/0/0	
Научна област, НО:	Електротехника и рачунарство	
Научна дисциплина, НД:	Рачунарска техника	
Предметна одредница/Кључне речи, ПО:	Андроид, препознавање говора, интерактивна апликација	
УДК		
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, ВН:		
Извод, ИЗ:	У овом раду је приказано једно решење имплементације интерактивне апликације са подршком за препознавање и могућност управљања гласом на Андроид платформи	
Датум прихватања теме, ДП:		
Датум одбране, ДО:		
Чланови комисије, КО:	Председник:	
	Члан:	
	Члан, ментор:	др Јелена Ковачевић
		Потпис ментора



## KEY WORDS DOCUMENTATION

Accession number, <b>ANO:</b>		
Identification number, <b>INO:</b>		
Document type, <b>DT:</b>	Monographic publication	
Type of record, <b>TR:</b>	Textual printed material	
Contents code, <b>CC:</b>	Bachelor Thesis	
Author, <b>AU:</b>	Miroslav Stefanović	
Mentor, <b>MN:</b>	Jelena Kovačević, PhD	
Title, <b>TI:</b>	Interactive application implementation with voice recognition and voice control support servis on Android platform	
Language of text, <b>LT:</b>	Serbian	
Language of abstract, <b>LA:</b>	Serbian	
Country of publication, <b>CP:</b>	Republic of Serbia	
Locality of publication, <b>LP:</b>	Vojvodina	
Publication year, <b>PY:</b>	2012	
Publisher, <b>PB:</b>	Author's reprint	
Publication place, <b>PP:</b>	Novi Sad, Dositeja Obradovića sq. 6	
Physical description, <b>PD:</b> (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/33/0/0/15/0/0	
Scientific field, <b>SF:</b>	Electrical Engineering	
Scientific discipline, <b>SD:</b>	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, <b>S/KW:</b>	Voice recognition, Android, Interactive application	
<b>UC</b>		
Holding data, <b>HD:</b>	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, <b>N:</b>		
Abstract, <b>AB:</b>	Interactive application implementation with voice recognition and voice control support servis on Android platform	
Accepted by the Scientific Board on, <b>ASB:</b>		
Defended on, <b>DE:</b>		
Defended Board, <b>DB:</b>	President:	
Member:		Menthor's sign
Member, Mentor:	Jelena Kovačević, PhD	

## **Zahvalnost**

Zahvaljujem se svom mentoru prof. dr Jeleni Kovačević na stručnoj pomoći tokom izrade završnog (*bachelor*) rada.

Srdačno se zahvaljujem asistentu Nenadu Četiću i Ivanu Letvenčuku na podsticanju za nastanak završnog rada, savete, strpljenje i podršku tokom izrade.

Na kraju se zahvaljujem svima onima koji su na bilo koji način doprineli izradi ovog završnog rada.

## SADRŽAJ

1.	Uvod.....	1
2.	Teorijske osnove .....	2
2.1	Android OS.....	2
2.2	STT tehnologija.....	3
2.3	HTTP .....	4
3.	Koncept rešenja.....	6
3.1	Modul STT .....	6
3.2	Modul za iscrtavanje bitmapa na ekranu i njihovo snimanje .....	10
3.3	Modul veštačke inteligencije (AI) .....	12
4.	Programko rešenje .....	14
4.1	Paket Main.....	15
4.2	Paket Game.....	16
4.3	Paket Levels.....	19
4.4	Paket Shaker .....	20
4.5	Paket Artifical Intelligence (AI) .....	20
5.	Rezultati .....	22
6.	Zaključak .....	25
7.	Literatura.....	26

## SPISAK SLIKA

<i>Slika 1 – Dijagram veze između korisnika, servisa i uređaja .....</i>	1
<i>Slika 2 - Grafički prikaz dugmadi za komunikaciju .....</i>	7
<i>Slika 3 - Grafički prikaz dugmadi za komunikaciju kada ne postoji internet konekcija .....</i>	8
<i>Slika 4 - MSC dijagram pozadinskih aktivnosti .....</i>	9
<i>Slika 5 - Način dodele bitmapa na osnovu karaktera .....</i>	10
<i>Slika 6 - Izgled iscrtane mape i dugmadi za komunikaciju .....</i>	11
<i>Slika 7 - Ilustrativni prikaz brisanja sadržaja vektora .....</i>	12
<i>Slika 8 – Sadržaj osnovnih paketa programskog rešenja .....</i>	14
<i>Slika 9 - Klasni dijagram paketa Main .....</i>	15
<i>Slika 10 - Klasni dijagram paketa Game .....</i>	18
<i>Slika 11 - Klasni dijagram paketa levels.....</i>	19
<i>Slika 12 - Klasni dijagram paketa Shaker .....</i>	20
<i>Slika 13 - Klasni dijagram paketa Artifical Intelligence .....</i>	21
<i>Slika 14 - MSC dijagram FlacChek projekta .....</i>	23
<i>Slika 15 - Rezultati prepznavanja glasovnih komandi .....</i>	23

## SKRAĆENICE

<b>AI</b>	- <i>Artifical Intelligence</i> , veštačka inteligencija
<b>STT</b>	- <i>Speech To Text</i> , tehnologija koja pretvara govor u tekst
<b>MSC</b>	- <i>Message Sequence Chart Diagram</i> , grafik sekvenci poruka
<b>CSV</b>	- <i>Comma-separated Values</i> , zarezima razdvojene vrednosti
<b>HTTP</b>	- <i>Hypertext Transfer Protocol</i> , protokol za prenos hiper teksta
<b>OS</b>	- <i>Operating System</i> , operativni sistem
<b>API</b>	- <i>Application programming interface</i> , programska sprega
<b>URL</b>	- <i>Uniform Resource Locator</i> , jedinstvena adresa resursa
<b>FLAC</b>	- <i>Free Lossless Audio Codec</i> , audio kodek bez gubitaka
<b>ADT Plugin</b>	- <i>Android Development Tools plugin</i> , Android razvojni alat
<b>SDK Tool</b>	- <i>Software Development Kit Tool</i> , skup softverskih alata za kreiranje aplikacija
<b>LCS</b>	- <i>Longest common subsequence algorithm</i> , algoritam za pronalaženje najdužeg zajedničkog podniza
<b>OSI</b>	- <i>Open Systems Interconnection</i> , referentni model za otvoreno povezivanje sistema

## 1. Uvod

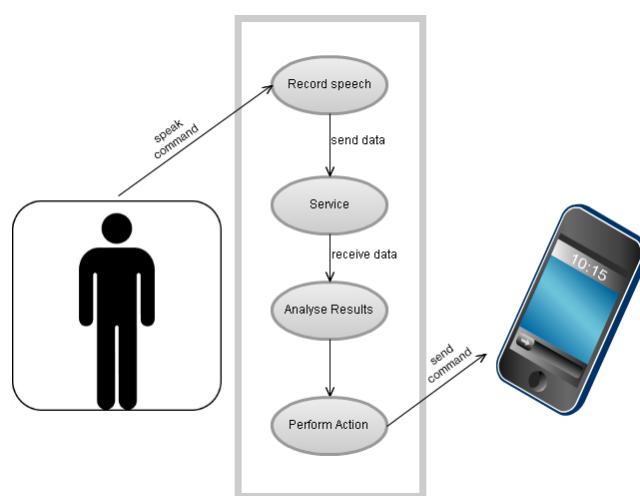
U ovom radu biće predstavljen jedan od načina implementacije interaktivne aplikacije sa podrškom za prepoznavanje i mogućnost upravljanja glasom (zadavanjem komandi), namenjene isključivo korisnicima uređaja sa Android operativnim sistemom.

U poglavlju teorijske osnove biće objašnjen nastanak i razvoj Android operativnog sistema, njegov koncept i primena, osnove tehnologije koja pretvara govor u tekst (u daljem tekstu biće korićena skraćenica STT), podela signala i protokol za prenos hiper teksta (skraćenica HTTP će biti korišćena).

Konceptom rešenja obuhvaćena je izrada najbitnijih modula aplikacije, njihova implementacija i način korišćenja.

Kroz programsko rešenje biće objašnjena međusobna povezanost svih modula kao i metode najznačajnijih klasa i pojedinih interfejsa.

U poglavlju rezultati nalazi se detaljan opis postupka testiranja STT modula.



Slika 1 – Dijagram veze između korisnika, servisa i uređaja

## 2. Teorijske osnove

### 2.1 Android OS

Android™ operativni sistem je trenutno najrasprostranjenij operativni sistem za mobilne telefone, zasnovan je na Linux kernelu i prilagođen je tako da se može koristiti na većini mobilnih uređaja, uključujući tablet, laptop, netbook, smartbook računare, čitače elektronskih knjiga, pa čak i ručne satove.

Pokretanje samih aplikacija se ne vrši direktno, već se aplikacije pokreću u okruženju odvojenom od ostatka sistema gde dobijaju samo određeni deo sistemskih resursa, pa tako nemaju pristup delovima sistema koji su im nepotrebni, što donekle poboljšava sigurnost i stabilnost sistema, takođe pri instalaciji aplikacija korisnik dobija listu svih dozvola koje jedna aplikacija zahteva da bi se instalirala, što korisniku daje mogućnost da uoči potencijalno štetne aplikacije i obustavi njihovu instalaciju pre nego što dođe do oštećenja.

Sa tehničke strane Android predstavlja Linux operativni sistem razvijen za ARM i x86 arhitekturu i sastoji se od modifikovanog monolitnog Linux kernela zaduženog za podršku hardvera i funkcija niskog nivoa, skupa biblioteka zaduženih za dodatne podrške kao što su iscrtavanje grafike, podrška za dekodovanje video snimaka, podrška za SSL enkripciju itd., u sklopu biblioteka se nalazi i odvojeni Android Runtime koji sadrži osnovne, bazne, biblioteke i Dalvik virtualna mašina zadužena za pokretanje aplikacija višeg nivoa napisanih u Java programskom jeziku. Na višem nivou od biblioteka su sistemske aplikacije, tu se nalaze, *window manager*, menadžer resursa, menadžer instalacionih paketa, kao i aplikacije zadužene za obavljanje osnovnih funkcija vezenih za mobilne telefone ili uređaj na kom je instaliran Android, na najvišem nivou se nalaze krajnje korisničke aplikacije, odnosno aplikacije koje direktno koristi korisnik.

Ovakva arhitektura sistema nije iznenađujuća jer predstavlja standardnu arhitekturu Linux sistema gde su segmenti sistema razdvojeni po nivoima na kojim rade. Za crtanje 3D grafike Android koristi biblioteku zasnovanu na OpenGL ES 2.0 specifikaciji, što ovom sistemu daje mnoge napredne grafičke sposobnosti. Android poseduje i ugrađenu podršku za multitasking.

Kroz svoju istoriju Android je imao nekoliko verzija od kojih je svaka donosila neku novinu i poboljšanje, tako je npr. verzija 1.0 bila prva zvanično dostupna verzija Android operativnog sistema, V1.5 Cupcake je bila njena nadogradnja zasnovana na Linux kernelu 2.6.27, V1.6 Donut je koristio 2.6.29 Linux kernel i imao još više dodatnih mogućnosti u odnosu na prvu 1.0 verziju. Sa pojavom verzija 2.0 i 2.1 pod nazivom Eclair ispravljene su mnoge postojeće greške u samom sistemu i dodate dodatne podrške za rad sa kamerom, kao i poboljšana virtualna tastatura. Verzija 2.2 Froyo je prešla na novi kernel 2.6.32, ubrzala je rad sa memorijom i poboljšala performanse samog sistema, V2.3 Gingerbread takođe prelazi na novi kernel 2.6.35 i dodatno poboljšava korisnički interfejs, takođe donosi sa sobom i podršku za veće displeje kao i za neke dodatne senzore. Verzija 3.0 poznata i kao Honeycomb bila je zasnovana na kernelu 2.6.36 i bila je prilagođena tablet računarima, dodati su joj interfejs elementi kao što su system bar i action bar koji su prilagođeni za tablet računare, takođe pojednostavljena je i upotreba multitaskinga, redizajnirana je i virtualna tastatura tako da omogući lakše i brže kucanje a uklonjeni su i neki sigurnosni propusti.

Verzija 4.0 koja nosi naziv Ice Cream Sandwich je trenutno najnovija verzija Android operativnog sistema.

## 2.2 STT tehnologija

Uopšteno govoreći realni svet proizvodi merljive izlaze koje mi nazivamo signalima. Signali po svojoj prirodi mogu biti diskretni ili kontinualni. Izvor ovog signala može biti stacionaran na način da se njegove statistike tokom vremena ne menjaju, ili nestacionaran. Signali mogu biti čisti (direktno mereni sa izvora signala) ili zašumljeni uticajem drugih izvora signala (šumova).

Problem od neverovatnog značaja jeste okarakterisati takav realni svet signala pomoću odgovarajućeg modela signala. Mnogo je razloga zbog kojih je ovakav model signala od značaja. Najvažniji od njih je da model signala omogućava odgovarajući teorijski opis signala pomoću koga se može predložiti odgovarajuća obrada signala i na taj način dobiti željeni izlaz. Na primer, ako je u pitanju obrada govornog signala koji je zagađen šumom ili smetnjama prilikom prenosa signala, mi možemo iskoristiti model signala kako bismo optimalno otklonili šum i potisnuli poremećaj koji se pojavio prilikom prenosa. Drugi značajan razlog zbog koga je važno razvijati modele procesa jeste da nam ti modeli često govore nešto o izvoru signala čak i onda

kada nam ti izvori nisu na raspolaganju. U takvim slučajevima mi možemo, ako su nam na raspolaganju dobri modeli signala, generisati, odnosno simulirati realne signale. Na kraju, možda i najvažniji razlog razvoja modela signala jeste razvoj efikasnih uređaja sistema za predikciju, prepoznavanje, identifikaciju i t.d.

Postoji nekoliko različitih tipova modela signala. Oni se, na grubi način, mogu podeliti u klasu determinističkih i u klasu statističkih modela. Deterministički modeli se baziraju na konkretnom znanju specifičnih osobina signala. Kod ovih modela se uglavnom, na osnovu ovog znanja, usvajaju karakteristični oblici a zatim se uglavnom estimiraju parametri kao što su amplituda, učestanost, fazni stav i tome slično. Druga široka klasa modela signala su statistički modeli u kojima je cilj da se izvrši karakterizacija statističkih osobina procesa. Primeri takvih statističkih modela uključuju Gausovske procese, Poasonove procese, Markovljeve procese, pa i skrivene Markovljeve procese. Zajednička nit ovih modela jeste da se signali mogu okarakterisati kao slučajni procesi pri čemu je važno odrediti stohastičke parametre ovih procesa.

Niti je teorija o skrivenim Markovljevim modelima nova, niti je njena primena u prepoznavanju govora nova. Osnovni teorijski rezultati su objavljeni kasnih šesdesetih i ranih sedamdesetih (Baum i dr.) dok su prve implementacije ovih rezultata u oblasti prepoznavanja govora izvršene sedamdesetih (Baker na CMU i Jelinek na IBM). Međutim, šira primena i razumevanja ovakve metodologije je naišla tek nekoliko godina kasnije.

Prepoznavanje govora se danas vrši na mnogo načina, sa dobro razvijenim i dobro uhodanim algoritmima, no ni jedna metoda još nije dovedena do vrhunca, sa 100%-tним prepoznavanjem. Kompanija Google je razvila *SpeechToText* API koji omogućava korisnicima uređaja sa Android operativnim sistemom, mogućnost korišćenja ovog servisa. Postupak implementacije i obrade obuhvaćen je realizacijom projektnog zadatka i biće detaljno opisan u konceptu rešenja i programskom rešenju.

## 2.3 HTTP

HTTP (engl. HyperText Transfer Protocol) je mrežni protokol koji pripada sloju aplikacije OSI referentnog modela, predstavlja glavni i najčešći metod prenosa informacija na vebu. Osnovna namena ovog protokola je isporučivanje HTML dokumenata, tj. veb stranica. HTTP je samo jedan od internet protokola. Razvoj i standardizaciju HTTP protokola nadgledaju W3C i Internet Engineering Task Force.

HTTP je protokol za komunikaciju između servera i klijenta, koji funkcioniše po principu zahtev/odgovor. HTTP klijent, koji je najčešće veb pregledač, inicira prenos podataka nakon što uspostavi TCP/IP vezu s udaljenim veb serverom na određenom portu.

---

Server konstantno osluškuje zahteve na određenom mrežnom komunikacijskom portu (tipično port 80), čekajući da se klijent poveže i pošalje svoj zahtev. Zahtev se sastoji od osnovne HTTP komande (čija je sintaksa propisana standardom i koja se sastoji od naziva komande, imena traženog dokumenta i verzije podržanog HTTP-a) i zaglavlja koje se sastoji od određenog broja redova teksta koji pobliže određuju aspekte zahteva.

Zahtev klijenta se obrađuje na serveru i, u zavisnosti od ispravnosti zahteva i mogućnosti zadovoljavanja istog, klijentu se šalje odgovarajući odgovor. Odgovor se sastoji od izveštaja o statusu zahteva (koji se sastoji od trocifrenog koda i kratkog deskriptivnog teksta statusa, npr. 200 OK) i od konkretnog odgovora, ukoliko je zahtev moguće zadovoljiti. Odgovor se sastoji od zaglavlja, koje je iste sintakse kao i zaglavlj zahteva i daje osnovne podatke o prirodi odgovora, i od eventualnog konkretnog sadržaja koji se tražio u zahtevu. U zavisnosti od verzije HTTP protokola kao i od zaglavlja zahteva, veza se može nakon toga prekinuti, a može se ista veza iskoristiti za slanje novog zahteva, radi uštede vremena.

## 3. Koncept rešenja

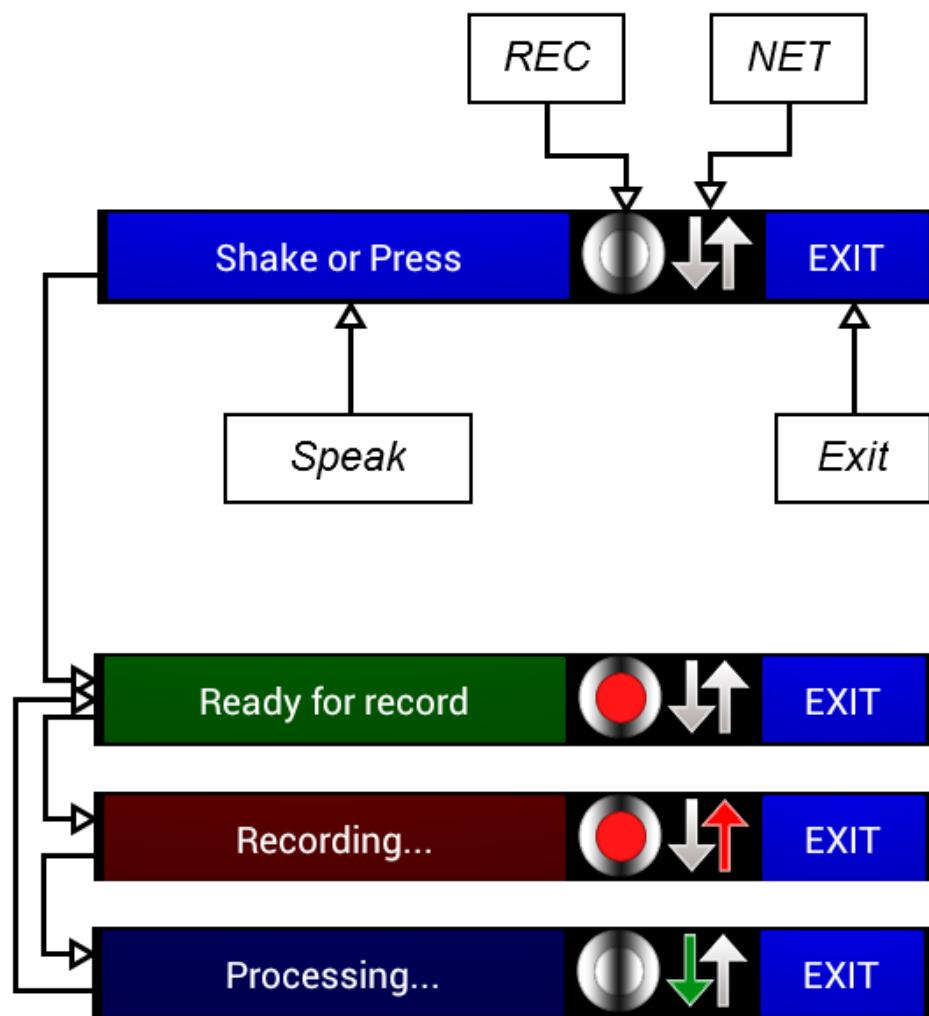
Realizacija Android aplikacije prvenstveno se oslanja na korišćenje Android sistemskih klasa, više-nitno programiranje i projektovanje nekoliko manjih modula sa jasno definisanim ciljem. U konceptu rešenja biće posvećena pažnja najviše tim modulima koju su između ostalog i najbitniji za pokretanje i normalan tok aplikacije. U tu grupu spadaju:

- 1) Modul STT
- 2) Modul za iscrtavanje bitmapa na ekranu i njihovo snimanje
- 3) Modul veštačke inteligencije ( AI )

### 3.1 Modul STT

STT modul je napravljen kao veza između korisnika, aplikacije i Google servisa za prepoznavanje i obradu govora tj., pretvaranja u tekst. Sastoji se od prilično jednostavnog grafičkog interfejsa i pozadinskog “sevisa“, dela koda koje se izvršava paralelno dok korisnika komunicira sa aplikacijom putem govora. Glasovne komande koje je moguće koristiti su:

- 1) *LEFT* – za pomeraj u levo
- 2) *RIGHT* – za pomeraj u desno
- 3) *UP* – za pomeraj na gore
- 4) *DOWN* – za pomeraj na dole
- 5) *STOP* – za prekid korišćenja glasovnih komandi
- 6) *RESET* – za resetovanje tekuće mape
- 7) *UNDO* – za vraćanje poteza unazad
- 8) *EXIT* – za završetak aplikacije



Slika 2 - Grafički prikaz dugmadi za komunikaciju

Na slici je prikazan grafički interfejs koji povezuje korisnika sa aplikacijom i pruža mogućnost uvida u sam tok izvršenja obrade govora signalizirajući korisniku u kom trenutku se snima, u kom trenutku se prosleđuju informacije servisu koji radi u pozadini, kada se sve to vraća i kada je moguće započeti novu sesiju. Na njemu se nalaze četiri dugmeta:

- 1) Speak – predstavlja glavno dugme i služi kao indikator korisniku, na njemu se ispisuju poruke o trenutnom stanju. Može se nalaziti u pet različitih stanja. Prvo i osnovno je stanje koje se instancira prilikom pokretanja aplikacije – *Shake or Press*. U njemu se čeka da korisnik pritiskom na dugme ili potresom telefona otpočne sesiju snimanja. Nakon čega se prelazi u *Ready for record* stanje. Ovde upaljen mikrofon čeka promenu amplitude zvuka, koja će preći granicu minimalne jačine koju smatramo govorom, i signalizirati prelazak u sledeće stanje. Iz *Ready for record* prelazimo na snimanje tačnije u stanje *Recording* gde se vrši automatsko čuvanje i slanje audio sadržaja pozadinskom servisu. Zatim prelazimo u obradu - *Processing* gde dobijamo rezultate od servisa u vidu Liste String-ova. Iz ovog stanja rezultati se šalju direktno Artifical Intelligence modulu (AI) na parsiranje i

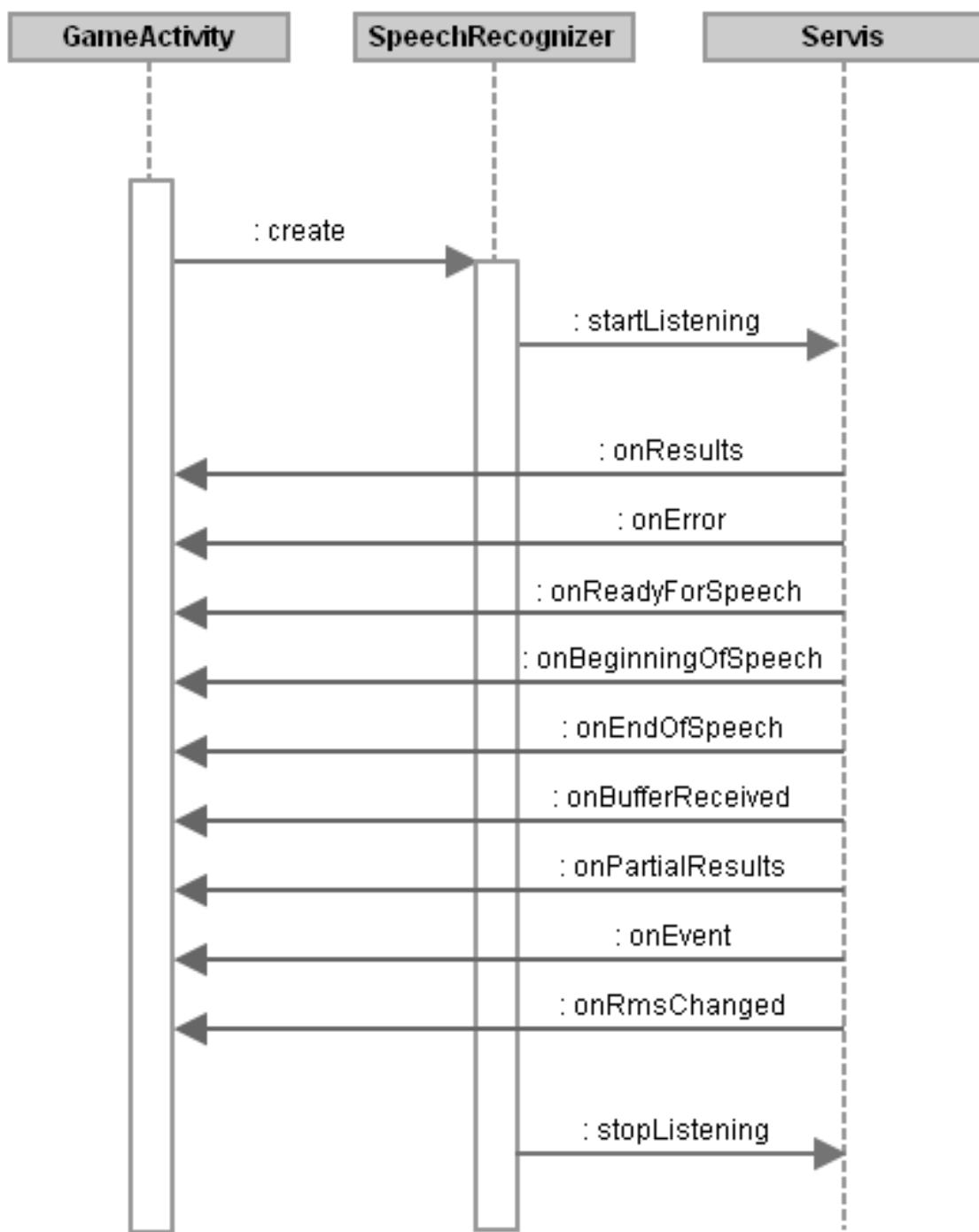
obradu o čemu će biti više reči u opisu modula AI. Takođe postoji stanje *No Internet Connection* u koje se prelazi u slučaju da ne postoji ni jedna aktivna internet konekcija, na bilo kom android uređaju u koji je instalirana aplikacija ili u slučaju da u toku igranja dođe do nepredviđenog gubljenja konekcije.



Slika 3 - Grafički prikaz dugmadi za komunikaciju kada ne postoji internet konekcija

- 2) REC – je dugme koje predstavlja indikator snimanja. Može imati dva stanja *ON* i *OFF*. U skladu sa željom korisnika za igranjem pomoću glasovnih komandi, ovo dugme se aktivira.
- 3) NET – je dugme koje predstavlja indikator postojanja internet konekcije kao i indikator slanja i primanja podataka sa interneta. Može imati četiri stanja. Prvo kada postoji internet konekcija i nema nikakve aktivnosti na mreži u smislu slanja - primanja sadržaja. Drugo kada se šalje snimljeni audio fajl. Zatim treće stanje u kome se primaju rezultati od servisa kada je audio fajl poslat. Četvrto stanje predstavlja nepostojanje internet konekcije.
- 4) Exit – je dugme koje zatvara postojeću internet konekciju i prekida obradu koja je u toku ili bilo koje druge aktivnoisti, u slučaju da su pokrenute, i izlazi u prethodni meni - meni izbora nivoa.

Pozadinski “servis“ je deo koda koje se odnosi na izvršenje snimanja i slanja audio sadržaja i dobijanje rezultata koji se dalje prosleđuju modulu AI. Detaljniji prikaz se nalazi na slici 3.

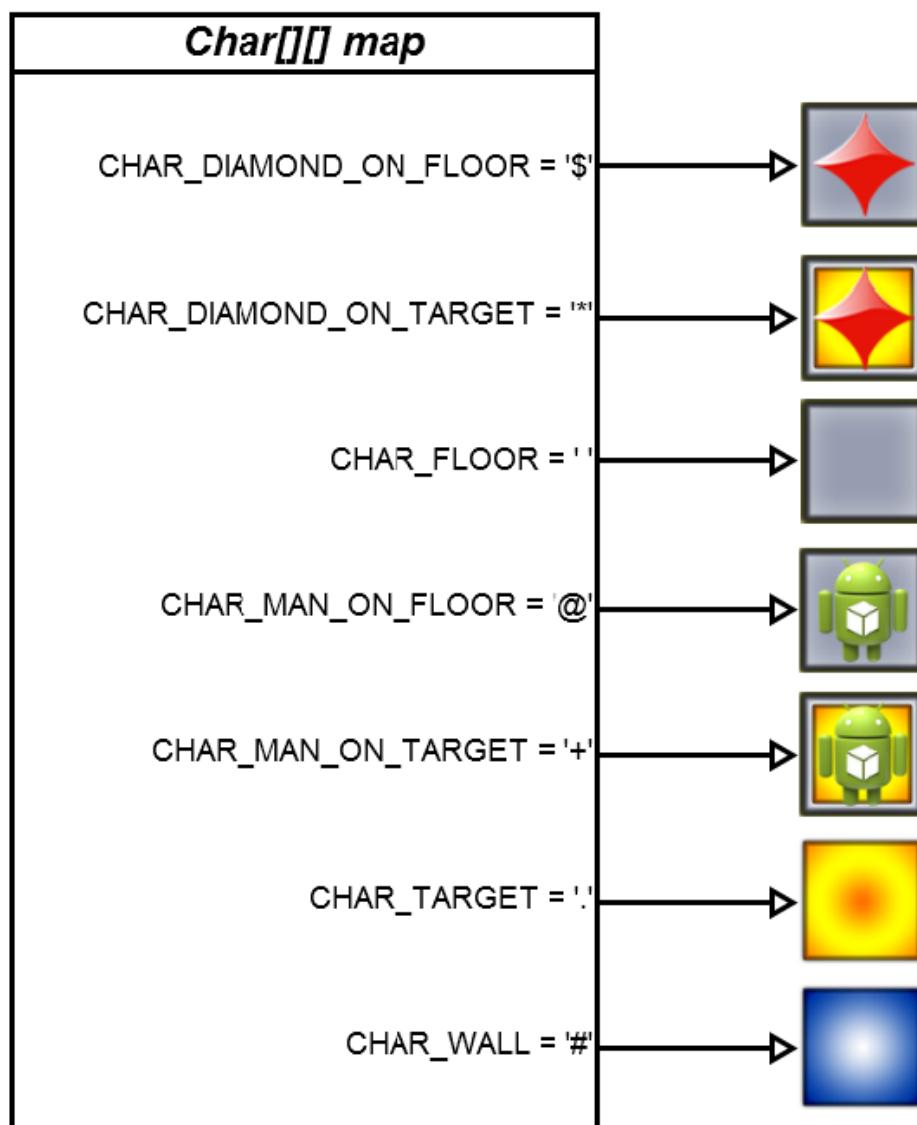


Slika 4 - MSC dijagram pozadinskih aktivnosti

Ovde vidimo da se u klasi *GameActivity* instancira objekat klase *SpeechRecognizer* koji implementira *SpeechRecognizerListener* i pozivom metode *startListening* aktivira se snimanje korisnika i slanje audio zapisa servisu. Po završetku obrade, rezultate ukoliko ih ima, dobijamo u metodi *onResults*, a ukoliko nije uspela obrada dobijamo kod greške u metodi *onError*. Ostale metode nam služe za indikaciju menjanja stanja na dugmićima Speak, REC i NET. Metoda *StopListening* označava završetak sesije.

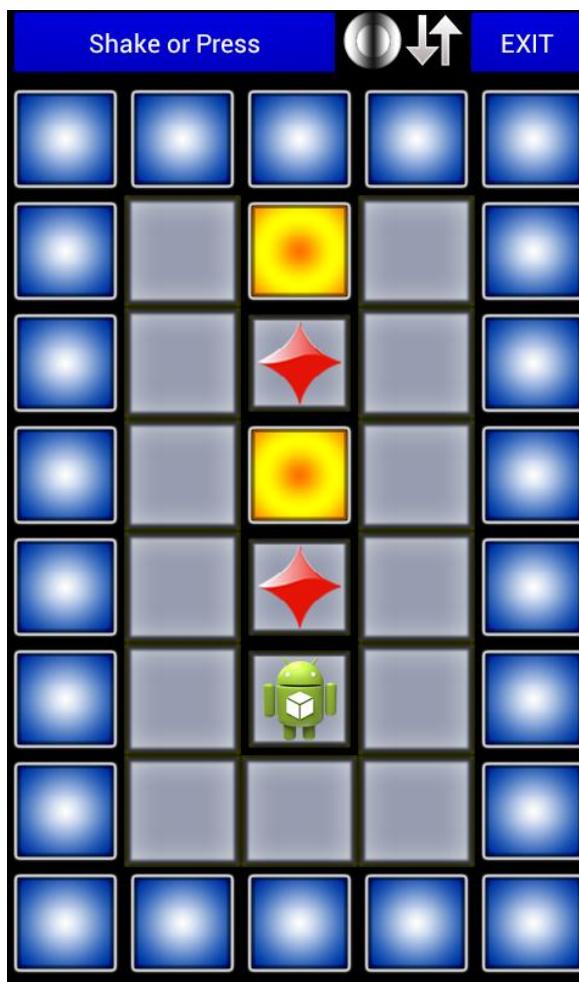
### 3.2 Modul za iscrtavanje bitmapa na ekranu i njihovo snimanje

Da bi se omogućilo korisniku da nesmetano koristi, potrebno je pre svega iscrtati aplikaciju na ekran što ujedno predstavlja zadatak ovog modula. Ideja je da se učitavaljem različitih karaktera iz dvostuke matrice (char[][] ) dodeli svakom istom elementu određena bitmapa koja će biti dalje prosleđena funkciji zaduženoj za iscrtavanje na ekran. U dvostrukoj matrici karaktera mogu se naći samo znakovi kao što su “\$“, “\*“, ““ , “@“, “+“, “.“, “#“. Tako npr., ako se prilikom prolaska petljom kroz dvostruku matricu nađe na karakter “#“ svaki put biće učitana slika određenih dimenzija koja predstavlja zid, i ta slika će biti konvertovana u bitmapu koja će se iscrtati na određenom prostoru ekrana. Na ovaj način korisnik će imati utisak da su na ekranu samo slike što nam je i bio cilj.



Slika 5 - Način dodelje bitmapa na osnovu karaktera

Kada se završi dodata bitmapa i njihovo iscrtavanje na ekranu, aplikacija ima ovakav izgled:

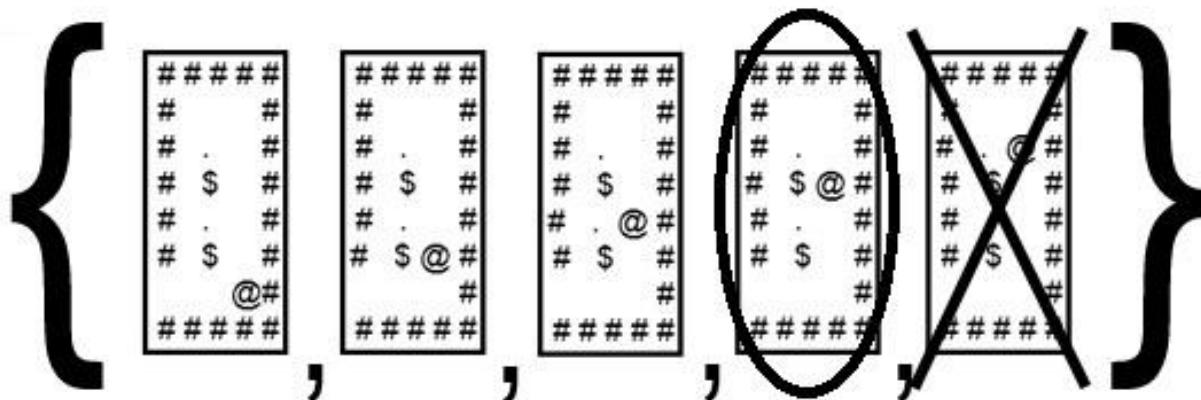


Slika 6 - Izgled iscrtane mape i dugmadi za komunikaciju

Implementacijom Android-ovog sistemskog *onTouchListener* dobijamo mogućnosti kretanja po ekranu na dodir. Potrebno je čikicom pomeriti crvene dijamante na žute kvadratiće da bi se prešlo na sledeći nivo. Ono što predstavlja problem jeste:

- 1) Zamena elemenata u dvostrukoj matrici karaktera. Da bi se izvršila kretnja na ekranu, dok korisnik dodirom vuče poteze ili ih zadaje korišćenjem glasovnih komandi, u pozadini se pronalazi korisnikovo željeno mesto u matrici karaktera i proverava se da li je moguće i u skladu sa pravilima igre odigrati taj potez. Ukoliko su svi uslovi ispunjeni odredišni karakter će biti zamenjen novim a na mesto izvornog će biti stavljen odgovarajući karakter. Nakon ovog se poziva metoda iscrtavalja koja će ponovo projuritu kroz matricu, dodeliti bitmape i izvršiti njihovo iscrtavanje na ekranu. To će se desiti dovoljno brzo da će korisnik imati utisak da se samo čikica pomera tj., odigrava zadati potez.
- 2) Realizacija snimanja svakog odigranog poteza ukoliko korisnik napravi grešku ili iz bilo kog razloga želi da se vrati unazad proizvoljan broj poteza. Rešava se smeštanjem celokupne dvostrukе matrice karaktera u Vector nakon svakog uspesno

odigranog poteza. Ukoliko korisnik zahteva izvršenje metode *UNDO*, poslednji element Vector-a sa mapama se briše i učitava predposlednji, koji se dalje prosleđuje na iscrtavanje. Na taj način vrši vraćanje proizvoljnog broja poteza unazad kako bi korisnik mogao ispraviti svoju grešku nastalu prilikom loše izrečene glasovne komande ili drugih razloga.



Slika 7 - Ilustrativni prikaz brisanja sadržaja vektora

### 3.3 Modul veštačke inteligencije (AI)

Modul veštačke inteligencije koji se koristi u ovom projektu je napravljen za svrhu poboljšanja igrivosti i odnosi se samo na glasovne komande. Naime kad korisnik pokuša da zada neku od komandi glasom, jedan od mogućih scenarija je 100%-tno prepoznavanje nakon čega sledi odigravanje ili prepoznavanje sa manje tačnosti. Međutim u većini slučajeva dolazi do mnogo manjeg procenta prepoznavanja (čak i do 50% manjeg) usled nepravilnog izgovaranja reči, različitog akcenta, pozadinskog šuma, buke, razgovora više lica itd... Da bi se povećao procenat uspešno pogodenih poteza oslonićemo se na modul veštačke inteligencije.

Nakon što dobijemo rezultate od pozadinskog "servisa" za snimljeni fajl koji smo poslali, prosleđujemo ih u AI. Zatim imamo nekoliko koraka u obradi a to su:

- 1) Parsiranje i sortiranje – postupak u kome se dobijena lista String-ova sređuje tako što se za svaki element proverava koliko sadrži komandi.
- 2) *LongestCommonSubsequence* je algoritam kroz koji se, nakon parsiranja i sortiranja, svaka reč svakog elementa liste propušta na obradu. Ovaj algoritam proverava sličnost između učitane reči i komandnih reči vraćajući kao rezultat broj. Ukoliko je taj broj veći do broja proglašenim graničnom vrednošću sličnosti, tekuća reč će biti zamenjena sa najsličnijom

komandom. Nakon ovoga se uzima sledeća i tako sve do kraja dobijenih rezultata. Primeri:  
leaf -> left, wrihgt -> right, dawn -> down itd...

- 3) Po završetku dodele vrši se sortiranje novodobijenih rezultata od kojih se odabira onaj sa najviše pogodaka. Zatim se rezultat prosleđuje funkciji za odigravanje poteza koja će pokušati da izmeni sadržaj dvostuke matrice karaktera novopristiglem potezima i pozvati funkciju za iscrtavanje koja će prikazati odigrane poteze.

## 4. Programko rešenje

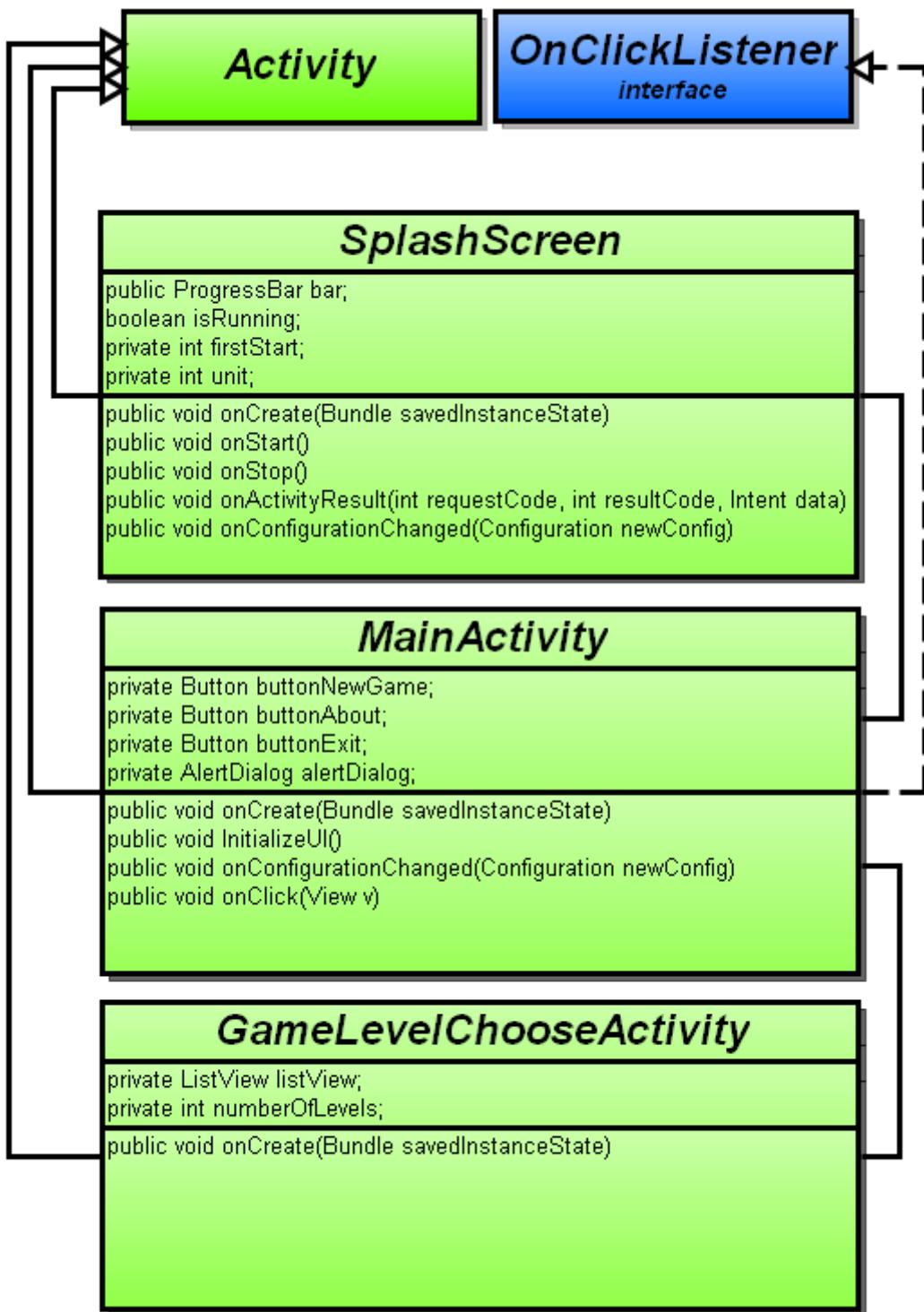
Programska realizacija projektnog zadatka napisana je u Java programskom jeziku i primjenjen je objektno orijentisan pristup programiranja. Korišćeno je *Eclipse Helios* razvojno okruženje sa *ADT Plugin-om* i *Android SDK Tools*. Realizovano rešenje zasnovano je na korišćenju Android sistemskih klasa, nasleđivanjem, menjanjem postojećih i dodavanjem novih. Osnovni paketi koji čine programsko rešenje zadatka prikazani su na slici 7.

Naziv paketa	Opis Paketa	Sadrzaj paketa - Klase
pak.main	Sadrži skup klasa koje služe za pokretanje Splash, Main i Level aktivnosti	SplashScreen.java MainActivity.java GameLevelChooseActivity.java
pak.game	Sadrži skup klasa koje su zadužene za prikaz grafike na ekranu, izvršenje algoritma igre, korišćenje glasovnih komandi i senzora	GameActivity.java GameView.java GameEngine.java
pak.levels	Sadrži skup nivoa igre	Levels.java
pak.shaker	Sadrži klasu za kontrolu senzora	Shaker.java
artificial.intelligence	Sadrži skup klasa i interfejs koji omogućuju korišćenje veštačke inteligencije u cilju poboljšanja STT rezultata i same igrivosti	LongestCommonSubsequence.java LcsString.java ArtificialIntelligenceEngine.java AlgorithmInterface ( interfejs ) Alg1.java

Slika 8 – Sadržaj osnovnih paketa programskog rešenja

## 4.1 Paket Main

Na slici možemo videti međusobnu zavisnost klase i interfejsa, kao i njihove metode i atribute koji su sadržani u paketu Main.



Slika 9 - Klasni dijagram paketa Main

Main paket sadrži skup klasa koje služe za pokretanje Splash, Main i Level aktivnosti. Sve tri klase nasleđuju Android sistemsku klasu Activity pa možemo reći da one same postaju aktivnosti svaka za sebe.

SplashScreen klasa se prva startuje nakon pokretanja aplikacije. Ona prikazuje njen zaštitni znak i progres dijalog koji, kada dostigne maksimalnu vrednost, automatski pokreće MainActivity klasu.

MainActivity klasa preventstveno instancira glavni meni aplikacije. Tačnije tri dugmeta ( nova igra, o igri i za izlazak iz igre ) kao i pozadinsku sliku, kako za horizontalnu poziciju ekrana tako i za vertikalnu. Ova klasa implementira *OnClickListener* Androidov sistemski interfejs, na osnovu čega je moguće kliknuti na neko od dugmadi iz menija. U zavisnosti od izbora u meniju pokreće se GameLevelChooseActivity klasa koja nam omogućava izbor nivoa aplikacije, AlertDialog za informacije o igri ili zatvaranje svih aktivnosti i izlazak.

GameLevelChooseActivity klasa sadrži listu svih nivoa aplikacije, oni su prikazani na ekranu i korisniku je omogućen izbor. Nakon odabira nivoa startuje se GameActivity klasa.

## 4.2 Paket Game

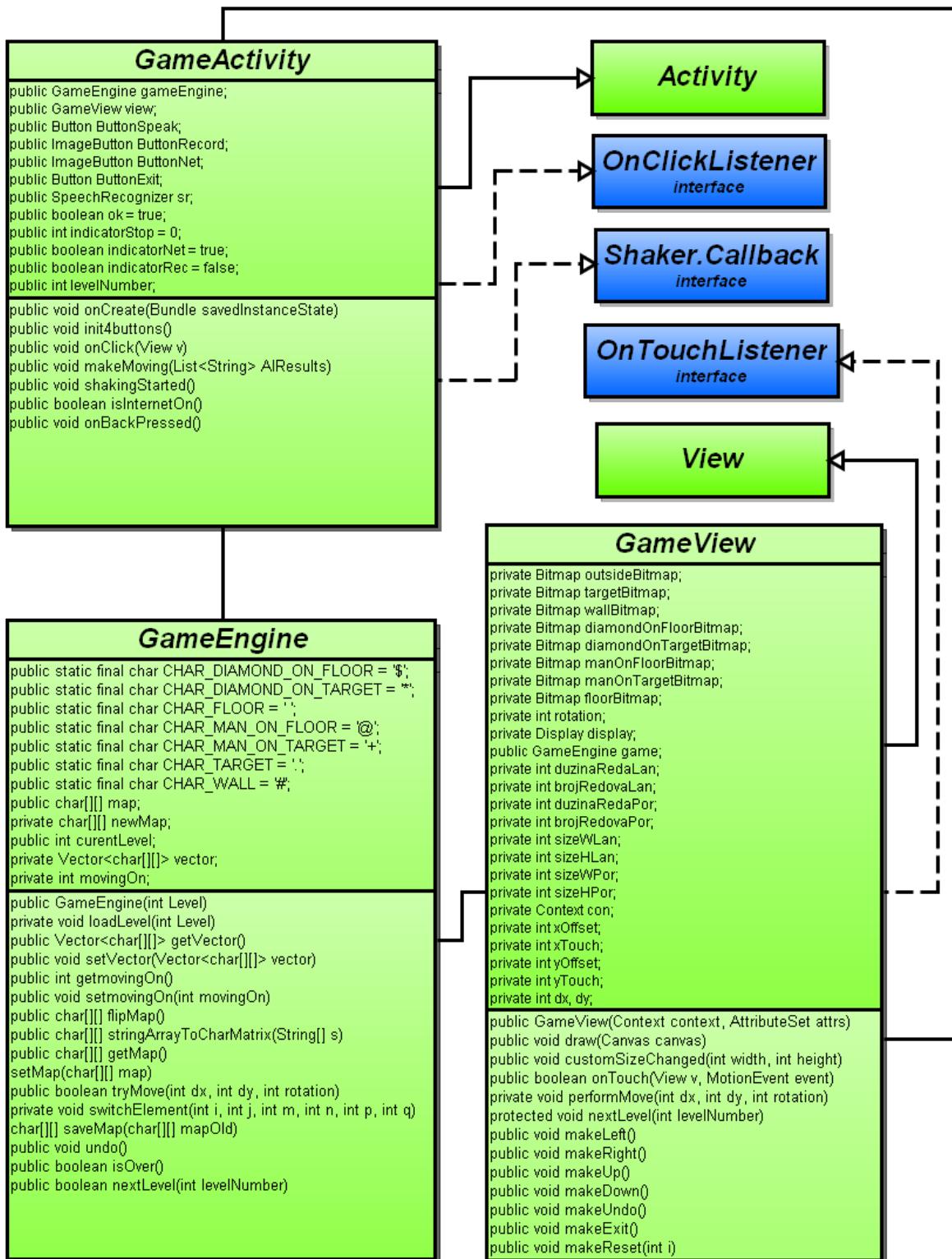
Paket Game je zadužen za iscrtavanje aplikacije na ekranu, samo igranje tj.kretnje kao i korišćenje glasovnih komandi. Glavna klasa ovog paketa je GameActivity.

GameActivity klasa nasleđuje Androidovu sistemsku klasu Activity. Ona instancira četiri dugmeta na severnoj strani ekrana ( dugme indikator govora, indikator snimanja, indikator internet konekcije i dugme za izlazak ) a ostatak rezerviše za iscrtavanje bitmapa, kako za horizontalnu poziciju ekrana tako i za vertikalnu. Ova klasa implementira *OnClickListener* Androidov sistemski interfejs, na osnovu čega je moguće kliknuti na neko od prethodnih dugmadi. Takođe implementira i Shaker.Callback interfejs koji omogućava praćenje potresa telefona na osnovu senzora čime se startuje snimanje kao preduslov za korišćenje glasovnih komandi. Algoritam glasovnih komandi kao i sama funkcionalnost su detaljnije objašnjeni u poglavljju koncept rešenja. Da bi se mogla iscrtati grafika na rezervisanom delu ekrana neophodno je instancirati objekat klase GameView.

GameView je klasa koje nasleđuje Androidovu sistemsku klasu View i njen zadatak je da prikaže grafički interfejs na mestu koje je klasa GameActivity predvidela. Implementira Androidov sistemski *OnTouchListener* interfejs kojim je omogućena funkcionalnost na dodir ekrana. Takođe sadrži metodu *public void draw()* koja iscrtava aplikaciju, metodu *public void*

*performMove()* zaduženu za pravljenje kretanja na ekranu, kao i metode koje se pozivaju prilikom obrade glasovnih komandi a to su : *public void makeLeft()*, *public void makeRight()*, *public void makeUp()*, *public void makeDown()*, *public void makeUndo()*, *public void makeReset()*, *public void makeExit()*. Metoda *isertavanja* detaljnije je objašnjena u poglavljiju koncept rešenja.

Na slici možemo videti međusobnu zavisnost klasa i interfejsa, kao i njihove metode i atribute koji su sadržani u paketu Game.



Slika 10 - Klasni dijagram paketa Game

*GameEngine* klasa predstavlja srce aplikacije, tačnije ona sadrži sve najbitnije atribute i metode koje klasa *GameView* koristi kako bi uspela da prikaže grafiku na ekranu u klasi *GameActivity*. Konstruktor ove klase kao parametar prima broj nivoa koji korisnik želi da proba i iz *Levels* klase učitava u *char[][]* matricu - nivo. Na osnovu karaktera iz matrice

određuje se koja će bitmapa biti iscrtana na ekranu. Funkcija *public boolean tryMove()* na osnovu parametara *dx*, *dy* i *rotation* pokušava da odigra potez ako je to moguće. Nakon odigranog poteza pamti se sadržaj novonastale mapu kao i prethodne u *private Vector<char[][]>* jer nam to omogućava da se vraćamo u nazad sa potezima ukoliko je korisnik napravio grešku, sve do početnog stanja. Metode *private void switchElement()* i sam postupak snimanja nivoa je detaljnije objašnjen u konceptu rešenja.

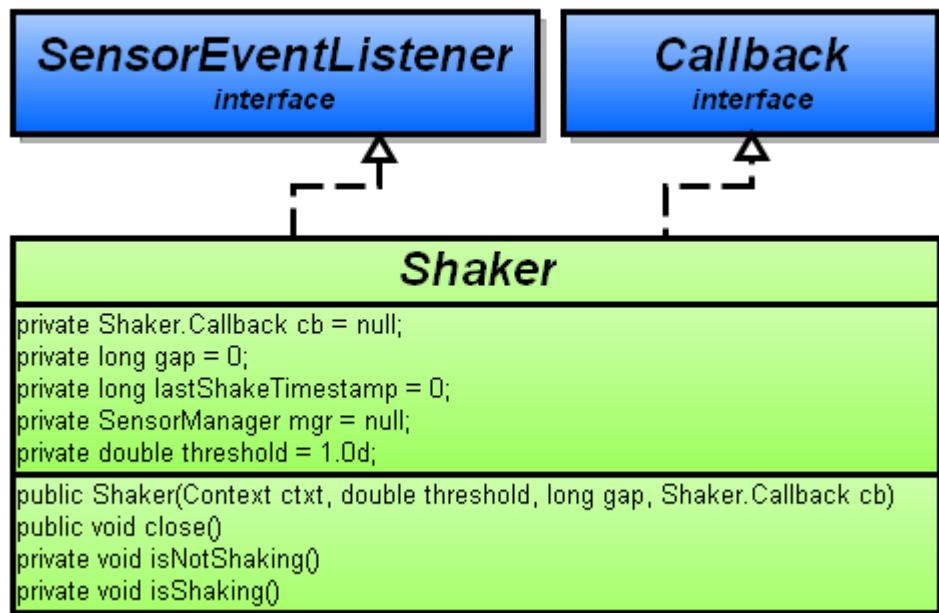
### 4.3 Paket Levels

Paket levels sadrži klasu *Levels* koja kao atribut ima dvostruki niz stringova, tačnije, jedan nivo se sastoji od određenog broja vrsta u kojima se nalazi po osam karaktera (onoliko koliko ima kolona). Vrsta sa karakterima pretstavlja jedan string a niz tih stringova čine nivo aplikacije. Niz od n nivoa pretstavljen je dvostrukim nizom stringova kao atribut klase *Levels* i pretstavlja skup svih nivoa.



Slika 11 - Klasni dijagram paketa levels

## 4.4 Paket Shaker



Slika 12 - Klasni dijagram paketa Shaker

Na slici možemo videti međusobnu zavisnost klase i interfejsa, kao i njihove metode i atribute koji su sadržani u paketu Shaker.

Klasa Shaker implementira SensorEventListener i Callback interfejs koji omogućuju da se na osnovu promene parametara dobijenih očitavanjem senzora akcelerometra, može znati kada je došlo do potresa telefona, na osnovu čega će biti pokrenuta aktivnost snimanja govora.

## 4.5 Paket Artifical Intelligence (AI)

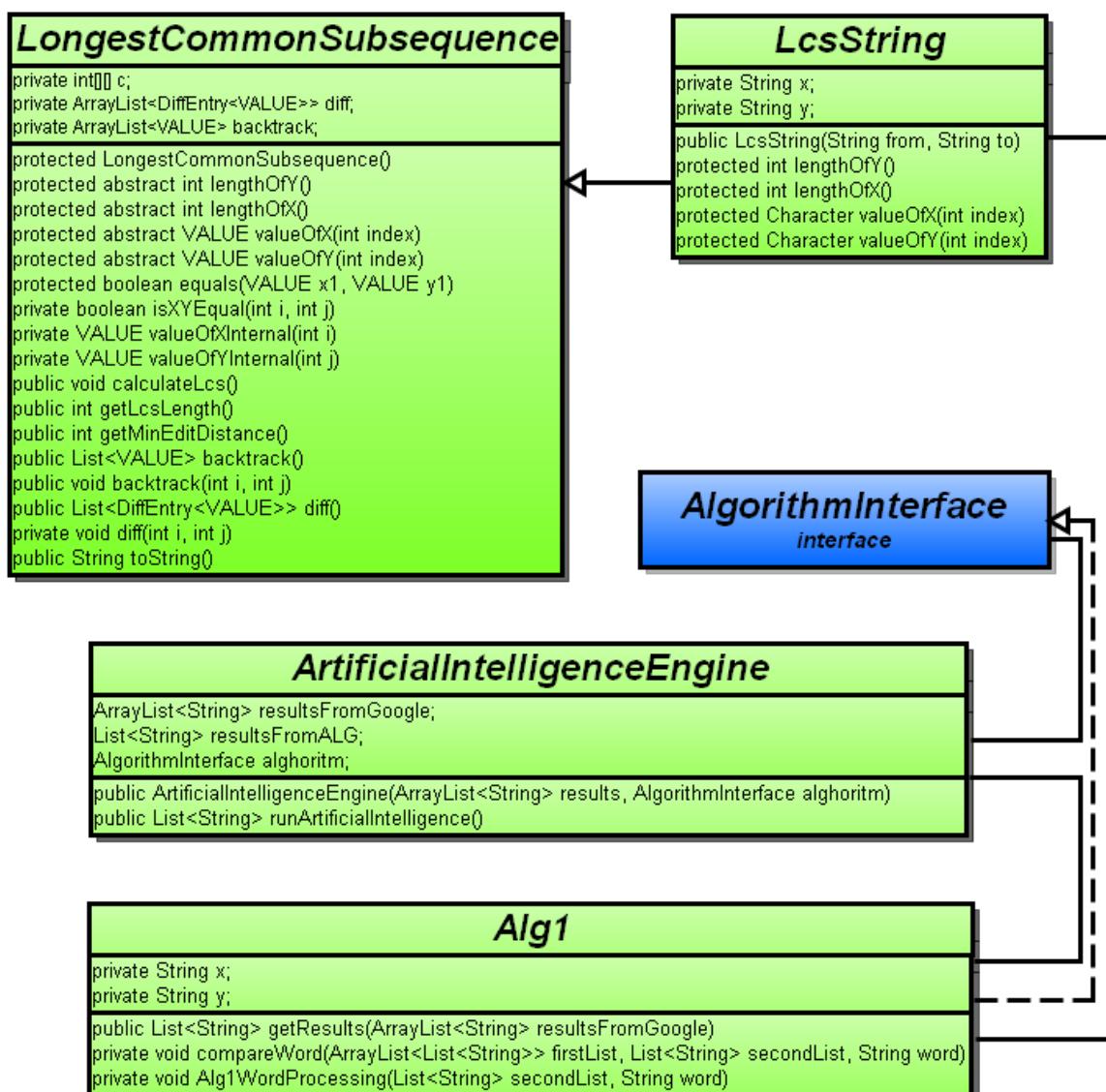
Paket Artifical Intelligence je namenjen za obradu rezultata dobijenih od Google-a prilikom slanja audio sadržaja u kome se nalaze potencijalni potezi ili komande aplikacije koje je korisnik izgovorio. Cilj je da se poveća procenat prepoznatih poteza (ponekad se izgovorene komande razlikuju u jednom ili više karaktera pa nije moguće uvek dobiti 100% poklapanje) kako aplikacija ne bi gubila na funkcionalnosti. Za to je zadužena prvenstveno *LongestCommonSubsequence* klasa.

*LongestCommonSubsequence* klasa je apstraktna klasa koja implementira algoritam. Dizajnirana je tako da može da poredi karaktere u strungu, linije u datotekama, blokove koda, čvorišta u XML dokumentu i slično. Klasa LcsString nasleđuje *LongestCommonSubsequence* klasu ali ograničava i uprošćava njen korišćenje samo na komparaciju stringova. U

zavisnosti od dužine dva stringa koja se porede algoritam će vratiti broj koliko se istih karaktera nalazi i u jednom i u drugom stringu, takođe će vratiti i parametar *distance* koji predstavlja koliko su ta dva stringa različita, odnosno broj karaktera koji se ne poklapa.

Pored pomenute dve klase postoje *ArtificalIntelligenceEngine* i *Alg1* klase. Dizajnirane su da se na najjednostavniji način implementiraju i pokrenu u aplikaciji onog trenutka kada se dobiju rezultati obrade audio datoteka poslatih servisu. One će te rezultate propustiti kroz *LongestCommonSubsequence* algoritam, izvršiti obradu i parsiranje nakon čega će zadržati samo one sa najvećim brojem pogodaka i proslediti ih na izvršenja, a ostatak odstraniti. Detaljniji opis ovog postupka je oписан u poglavljju koncept rešenja.

Na slici možemo videti međusobnu zavisnost klasa i interfejsa, kao i njihove metode i atribute koji su sadržani u paketu Artifical Intelligence.



Slika 13 - Klasni dijagram paketa Artifical Intelligence

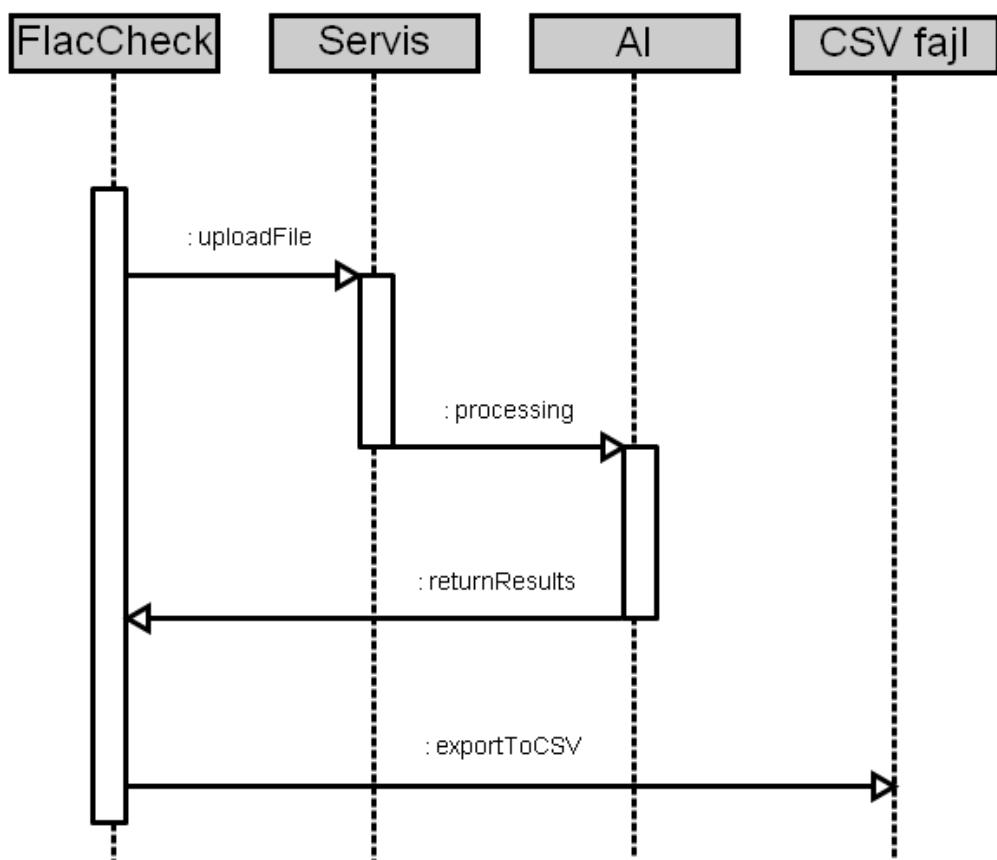
## 5. Rezultati

Ispitivanje rada AI realizovanog projektnim zadatkom vršeno je pomoću *FlacCheck* i *Test* projekata, koje poziva *test.bat* datoteka.

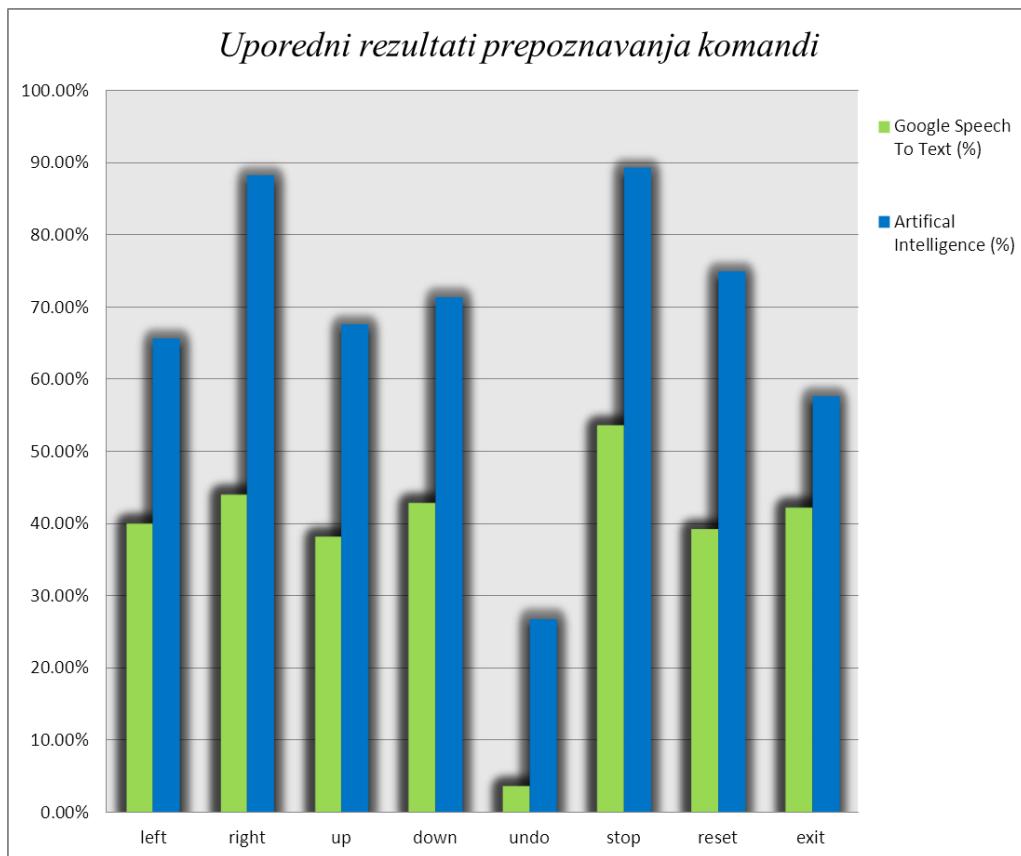
*FlacCheck* kao argumente komandne linije prima lokaciju i naziv Flac audio datoteka, sample rate tj., frekvenciju odabiranja, broj rezultata i String originalno izgovorenih komandi. Nakon slanja audio datoteka setvisu putem HTTP konekcije, kao odgovor će nam stići rezultati. Parser obrađuje rezultate tj., svaku reč posebno, propustajući kroz LCS algoritam. LCS će na osnovu sličnosti dva String-a donositi zaključak da li je reč koja se obrađuje potencijalna komanda ili ne. U slučaju da jeste ona će biti preimenovana u komandu. Na osnovu ovih parametara i obrade u izlaznu CSV datoteku će biti snimljeni novodobijeni rezultati.

*Test* kao argument komandne linije prima ime izlazne CSV datoteke. Učitavaju se svi rezultati prethodno kreirani pomoću *FlacCheck* i vrši se njihovo sortiranje. Prilikom sortiranja sabira se broj ponavljanja komandnih reči za svaku grupu. AI,Servis i Original predstavljaju grupe. Nakon ovoga će se u izlaznu datoteku upisati rezultati koje dalje korištimo za dobijanje grafika prikazanog na slici 13.

*Test.bat* je izvršna datoteka koja učitava sve audio zapise i sve originalno izgovorene komande u vidu tekstualnih datoteka, iz zadatog foldera, prosleđuje ih kao argumente komandne linije prilikom poziva *FlacCheck* zajedno sa parametrima frekvencije odabiranja i brojem rezultata. Nakon obrade svih audio zapisa poziva se *Test* i kao finalni rezultat dobijamo graf prepoznavanja komandi.



Slika 14 - MSC dijagram FlacChek projekta



Slika 15 - Rezultati prepznavanja glasovnih komandi

Ovaj test je rađen na osnovu snimanja govora 10 lica, koji su izgovorili po 5 rečenica sa minimalno 5 - 7 komandi. Mozemo zaključiti da je prilično mali broj tačno pogodjenih komandi od strane servisa ( manji od 50%) iz više razloga - pozadinski šum, govor više lica, drugačijeg akcenta, pogrešnog izgovaranja i td... Propuštanjem kroz algoritam veštačke inteligencije procenat uspešnosti se povećava za 35% od ukupnog broja izgovorenih komandi. Samim tim imamo oko 70% odigranih poteza u slučaju kada se zadaje između 5-7 komandi glasom, dok se za manji broj zadatih komandi preciznost povećava.

## 6. Zaključak

Ovaj rad predstavlja jedno rešenje realizacije interaktivne aplikacije sa podrškom za prepoznavanje i mogućnost upravljanja glasom (zadavanjem glasovnih komandi), namenjene isključivo korisnicima uređaja sa Android operativnim sistemom. Aplikacija je napravljena u programskom jeziku Java, primenom objektno orijentisanog pristupa programiranja zasnovanog na korišćenju Android sistemskih klasa, menjanjem ili dodavanjem novih. Korišćeno je *Eclipse Helios* razvojno okruženje sa *ADT Plugin-om* i *Android SDK Tools*.

Na osnovu dobijenih rezultata i iskustva stečenog kroz realizaciju diplomskog rada, može se zaključiti da se postupak prepoznavanja i obrade glasa (zvuka), odnosno STT tehnologija, na Android operativnom sistemu može realizovati vrlo jednostavno.

Implementacija se svodi na korišćenje Google API-ja (java programsko rešenje koje se odnosi na aplikacije) ili slanjem zahteva na određen URL uz pomoć HTTP protokola (rešenje koje zaobilazi pravljenje aplikativnog dela i može biti veoma korisno za implementaciju na svim platformama koje podržavaju HTTP). Zahvaljujući velikom broju raznovrsnih metoda i interfejsa koje nam API pruža, moguće je kontrolisati gotovo svaki segment toka obrade. Počevši od snimanja zvuka, preko indikatora promene njegove jačine, podešavanja pauze u izgovoru između reči, signalizacije završetka govora pa do preuzimanja rezultata nakon obrade.

Verodostojnost dobijenih rezultata je relativna, ali kada se uzme u obzir da ni jedan STT algoritam nije doveden do savršenstva, nakon izvršenog testiranja, stiče se utisak da su na zavidnom nivou, tačnije procenat uspešnosti direkto zavisi od količine pozadinske buke prilikom snimanja. Takođe datoteke, u kojima je sniman govor lica sa drugačijim akcentom ili nepravilnim izgovorom dovodi do povećanja nepreciznosti. Ovaj vid problema se veoma uspešno zaobilazi implementacijom AI algoritma veštačke inteligencije, koji povećava procenat uspešnosti do 70% a nekada i više.

## 7. Literatura

- [1] Vladimir Kovačević, Miroslav Popović: Sistemska programska podrška u realnom vremenu, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka, 2002
- [2] Reto Meier: *Professional Android 4 Application Development (Wrox Professional Guides)*, USA, May 1, 2012
- [3] Reto Meier: *Professional Android Application Development*, USA, 2008
- [4] Paul Michael Kilgo: *Android OS: A robust, free, open-source operating system for mobile devices*. USA, 2012
- [5] *Android Summer School 2011* RT-RK Novi Sad
- [6] Sajt Wikipedia, *The Free Encyclopedia*,  
[http://en.wikipedia.org/wiki/Longest\\_common\\_subsequence\\_problem](http://en.wikipedia.org/wiki/Longest_common_subsequence_problem),  
učitano 21.06.2012
- [7] Sajt Wikipedia, *The Free Encyclopedia*, <http://sr.wikipedia.org/sr-el/HTTP>,  
učitano 21.06.2012
- [8] Sajt Stackoverflow programerski forum, *free to ask questions, free to answer questions, free to read, free to index...*, <http://stackoverflow.com/>,  
korišćen od 1.04.2012 do 21.06.2012
- [9] Sajt Android podrške za razvoj, *Android Developers*,  
<http://developer.android.com/resources/articles/speech-input.html>,  
korišćen od 1.04.2012 do 21.06.2012