Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Рачунарска техника и софтверско инжењерство

Ниво студија: Основне академске студије

Предмет: Оперативни системи

Број индекса: 564/2015

Ненад М. Пантелић

Имплементација гласовног асистента применом Гуглових сервиса говор-у-текст и текст-у-говор

дипломски рад

Комисија за преглед и одбрану:

1. **доц. др Владимир М. Миловановић - ментор**
2. _____ Датум одбране:_____
3. _____ Оцена:_____

	Универзитет у Крагујевцу **Факултет инжењерских наука**	

Основне студије: **Рачунарска техника и софтверско инжењерство**

Студијско подручје (усмерење): **електротехничко и рачунарско инжењерство**

Име и презиме: **Ненад Пантелић**

Број индекса: 564/2015

ПРИЈАВА ЗАВРШНОГ РАДА

Тема рада: Имплементација паметног асистента за Линукс платформу заснована на препознавању говора и компјутерском генерисању гласа

Задатак: Циљ овог завршног рада је имплементација паметног аудио асистента за Линукс платформу. Комуникација између корисника и асистента одвијаће се кроз гласовне команде - корисник говори асистенту шта жели од њега, а асистент на основу препознате наредбе извршава команду која је од њега тражена. Асистент ће одговарати кориснику такође гласовно, тј. генерисаће глас на основу текста. Команде су у одређеној мери предефинисане, односно асистент ће имати могућност контролисања оперативног система и програма у оквиру система и сам језик командовања ће имати своја правила, односно своју језичку семантику. Неке од додатних функционалност подразумевају претрагу информација на Интернету, и то пре свега информација на Википедија енциклопедији, као и диктирање и слање електронске поште. Такође, асистент ће комуницирати са другим web сервисима (Twitter, YouTube, Github…) тако да ће корисник моћи да користи њихове услуге, нпр. да тражи и пушта видео клипове са YouTube-а. Уз то, план је додати и неку специфичну функционалност попут алата за решавање једноставнијих математички израза диктирањем асистенту. Сам алат је могуће интегрисати у већи паметни систем, па ће бити представљен и концепт контроле кућних уређаја помоћу асистента. Асистент ће функционисати као апликација, односно алат уз оперативни систем и биће доступан на енглеском и српском језику.

Крагујевац, 31.05.2019.

Ментор:

Др Владимир М. Миловановић, доцент

Садржај

Резиме

Циљ овог дипломског рада је развој гласовног асистента (енг. *voice assistant*) названог **Линдо**. Гласовни асистент функционише као независна конзолна апликација (програм) и није платформски зависан. Основне функционалности на којима се заснива рад било ког гласовног асистента, подразумевају препознавање говора и извлачење текста изреченог у говору, као и генерисање говора којим асистент комуницира са корисником. Линдо гласовни асистент разуме и комуницира на српском и енглеском језику, али је сама апликација креирана да буде модуларна и уз врло мало рада могуће је додати и друге језике у систем. Линдо нуди разне услуге и функционалности које корисник може употребити за добијање података или управљање појединим процесима и активностима. Већина тих функционалности се заснива на примени библиотека и *апликационих програмских интерфејса* (АПИ).

Кључне речи: гласовни асистент, Линдо, говор-у-текст, текст-у-говор, АПИ.

Abstract

The purpose of this thesis is to develop a voice assistant named **Lindo**. Voice Assistant operates as a standalone console application (program) and it is platform-independent. The basic functionalities of any voice assistant include recognizing the speech and extracting the text spoken in the speech, as well as generating the speech by which the assistant communicates with the user. Lindo Voice Assistant understands and communicates in Serbian and English, but the application itself is designed to be modular and with very little work it is possible to add other languages to the system. Lindo offers a variety of services and functionalities that a user can use to obtain data or manage specific processes and activities. Most of these functionalities are based on the application of libraries and *application programming interfaces* (API).

Keywords: Voice Assistant, Lindo, speech-to-text, text-to-speech, API.

1 Увод

Гласовни асистент је врста личног виртуелног асистента који има способност разумевања и комуницирања са корисником путем гласа. Корисници могу да питају своје асистенте питања, контролишу кућне уређаје подложне аутоматизацији, управљају основим активностима које укључују управљање ресурсима попут електронске поште, планера и календара.

Први уређај који је могао да буде контролисан гласом је био Radio Rex, а први рачунарски уређај способан за дигитално препознавање гласа био је **IBM Shoebox**, развијен 1961. године. Другу половину 20.века обележили су **IBM** (енг. *International Business Machines*) и **америчка агенција за развој наоружања** (енг. *Defense Advanced Research Projects Agency (DARPA)*), како у домену рачунарске и технолошке индустрије, тако и у домену развоја гласовних асистената. Први конкретни гласовни асистент био је **IBM Simon**, представљен 1994. године, док је први модерни дигитални асистент, назван **Siri**, представљен код телефонског модела iPhone 4S, октобра 2011. године.

Од 2017. године употреба дигиталних гласовних асистената је доживела велику експанзију. Компаније Епл (*Apple*), употребом Сири асистента и Гугл (*Google*), употребом Google Assistant, предњаче у употреби гласовних асистената на паметним телефонима. Мајкрософт (*Microsoft*) пружа могућност употребе гласовног асистента на перзонализованим рачунарима, паметним телефоним и звучницима. Амазон (*Amazon*) је значајно повећао свој утицај у домену паметних звучника представљањем Алекса (*Alexa*) гласовног асистента. [1]

Линдо гласовни асистент је заправо пројекат са циљем примене гласовне интеракције у контроли рачунарског система, по угледу на комерцијалне гласовне асистенте. Апликација је превасходно развијена и намењена Линукс оперативном систему, али је у својој основи платформски независна. Цела апликација је развијена у Пајтон (енг. *Python*) програмском језику - конкретно верзија 3.6.8. За услуге препознавање говора и генерисање гласа се користе Гуглови сервиси о којима ће бити више речи у поглављима која следе.

1.1 Конверзија говора у текст

Препознавање говора је једна од комплекснијих тема у областима машинског учења и вештачке интелигенције. Да би се говор превео у текст, потребно је проћи кроз пар корака. Говор је заправо по својој природи звук, дакле талас. Када говоримо, стварамо вибрације у ваздуху, а то је заправо аналогни сигнал. С обзиром да су рачунари данас дигитални уређаји, потребно је дигитализовати тај сигнал. Први корак у обради говора је превођење звука из аналогног у дигитални облик употребом **PCM - Pulse Code Modulation**. Аналогни сигнал се тиме дискретизује, односно преводи у дигитални облик који се обрађује даље. Поред превођења, односно узорковања сигнала, потребно је извршити и неке додатне операције, као што су филтрирање шума, обрада јачине и темпа звука (људи не говоре увек истом динамиком и темпом).

Да би се поједноставио шаблон препознавања говора, PCM дигитални аудио сигнал се преводи у фреквентни домен употребом **брзе Фуријеове трансформације** (*Fast-Fourier Transform (FFT)*). У фреквентном домену је могуће идентификотвати фреквенцијске компоненте звука, па је тако могуће приближно одредити како људско ухо доживљава звук. Рецимо да када FFT анализира звук и преводи га у фреквентни домен, на сваку стотинку се генерише пар **својствених, кодних бројева** (*feature numbers*) за тај део сигнала. Ови бројеви заправо потичу из шифарника који сваки систем препознавања говора има. Шифарник представља попис звукова и њихових кодних бројева - база звукова на основу које систем одређује **фонеме** (база садржи много записа звукова и на основу тога систем учи да препознаје говор). Дакле, када кажемо анализира се свака стотинка, за тај део сигнала се додељују бројеви из шифарника према томе на шта највише личи тај звучни сигнал.

Након овога, циљ је из говора извући основну структурну јединицу - фонему. Фонему физички можемо замислити као јако кратак запис сигнала (као да је сигнал узоркован доста малом периодом одабирања). Формално, фонема је најмањи елемент језика, односно репрезентација звукова која правимо при изговору неке речи. Оквирно постоји 40-44 фонема у енглеском језику. Склапањем фонема добијамо континуалан говор.
Корак који следи се заснива на статистичким моделима. Према анализи која се обавља на сваки стоти део секунде, рецимо да је за фонему **“h”** вероватноћа 55% да се кодни број #52 појављује у тој стотинки, 30% шанса да се кодни број #189 појављује и 15% шанса да се кодни број #53 појављује. С друге стране, рецимо да свака стотинка секунде фонеме **“f”** има вероватноћу од 10% да се јавља кодни број #52, 10% вероватноће да се јавља кодни број #189 и преосталих 80% да се јавља кодни број #53.

Анализа вероватноће која се врши при тренингу се користи и при препознавању говора. Рецимо да је исечак говора који се анализира описан са 6 кодних бројева, и то: 52, 52, 189, 53, 52, 52

Систем препознавања рачуна вероватноћу да је звучни сегмент који је чуо заправо фонема "h" и са друге стране рачуна да је у питању фонема "f".

За "h", рачуница је следећа:

$$0.55 * 0.55 * 0.3 * 0.15 * 0.55 * 0.55 = 0.004117781$$

За "f", рачуница је следећа:

$$0.1 * 0.1 * 0.1 * 0.8 * 0.1 * 0.1 = 0.000008$$

Из ове рачунице, извлачимо закључак да је вероватније да је у питању фонема "h".

Још један детаљ око ког треба водити рачуна, је да систем препознавања треба да зна када се једна фонема прекида, а друга почиње. Алгоритми препознавања говора за ту намену користе математичку технику познату под називом **скривени Марковљеви модели** (енг.**Hidden Markov Models**). У овом раду неће бити детаљније описивана ова техника, већ само интуиција која стоји иза технике.

Рецимо да је систем препознавања чуо фонему "h" праћену фонемом "ее". Фонема "ее" има 75% вероватноћу да је описана кодним бројем #82, 15% да је описана кодним бројем #98 и 10% вероватноће да је садржан кодни број #52 (обратити пажњу да се овај кодни број јавља и код фонеме "h"). Рецимо да је дата секвенца бројева који описују изговорени звук: 52, 52, 189, 53, 52, 52, 82, 52, 82....
Питање је, где се догађа прелаз из једне фонеме у другу? Запажамо да је у првом делу секвенце већи утицај кодног броја #52, а у другом делу већи утицај броја #82. Људско око ово може да закључи. С друге стране рачунари користе скривени Марковљев модел за ту намену.

Треба додати да је алгоритам одређивања много комплекснији и сложенији за рачунање него што је то овде представљено. Постоји много фактора на које треба обратити пажњу и који утичу на рад алгоритма. Секвенца од три речи на систему који има речник од 60 000 речи (уобичајено за данашње системе), има $216*10^{12}$ могућности. Према томе, јасно је да овај проблем изискује велико искоришћење ресурса, па се ту онда јавља и фактор оптимизације, паралелизације итд. [2]

1.2 Конверзија текста у говор

Конверзија текста у говор, природно, иде обрнутим редоследом у односу на конверзију говора у текст. Фазе конверзије текст-у-говор су:

- текст у речи
- речи у фонеме
- фонеме у звук

Конверзија текста у речи - читање речи делује лако, али то баш и није тачно. Овај проблем се своде на принцип - колико је детету тешко да научи да изговара речи. Главни проблем код читања речи јесте двосмисленост. Многе речи, зависно од контекста, могу да имају различито значење (хомоними). Према томе, први корак код овог поступка је процес тзв. **нормализације**. Овај поступак се своди на процесирање текста тако да рачунар направи што мање грешака при читању текста. Елементи текста попут бројева, датума, скраћеница, акронима, специјалних карактера (ознаке валута нпр.) се преводе у речи. Рецимо, број 1843 може да алудира на количину или годину, а то се различито чита. Док људи то закључују према смислу реченице, рачунари немају способност да тоураде. Према томе, користе технике попут споменуте технике скривених Марковљевих модела или неуронских мрежа како би уочили шаблон. Интутитивно, ако се реч "година" нађе у истој реченици као и број 1843, разумно је закључити да је у питању број године, док у случају да постоји децимална тачка у броју 1843, вероватно се ради о мери.

Конверзија речи у фонеме - У теорији, ово је једноставан проблем. Све што рачунару треба је велика алфабеткса листа речи, дакле речник, са детаљима како се изговарају све речи - секвенца фонема које формирају ту реч. Граматички речници језика садрже податак о томе како се свака реч изговора, па отуда ова аналогија.

С друге стране, у пракси, то је ипак мало теже. Једна иста реченица може бити изговорена на много начина зависно од контекста реченице, особе која говори (пола или старости), емоција са којим говорник говори....

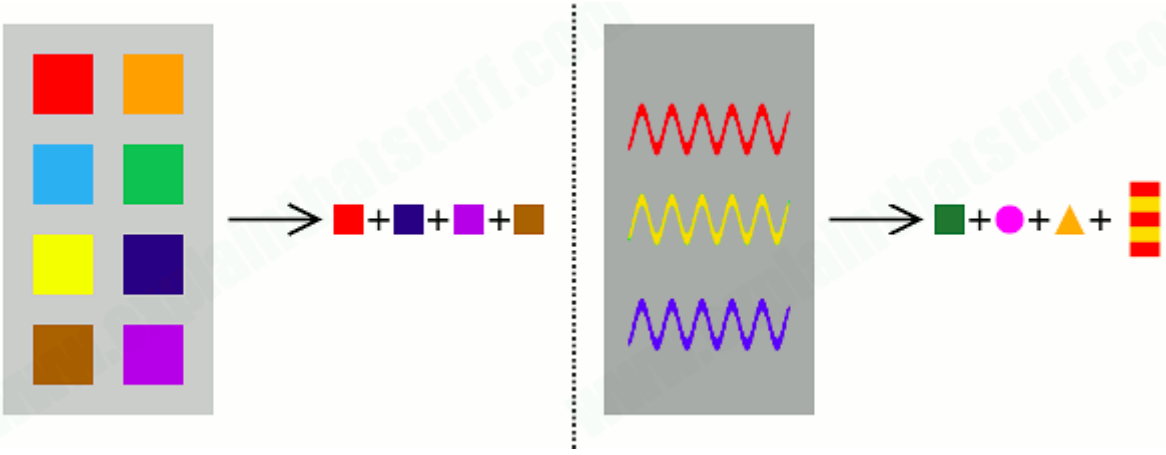
Алтернативни приступ код решавања овог проблема се заснива на разлагању речи на графеме (писане компоненте речи сачињене од слова или слогова) и генерисању фонема које одговарају тим графемама. Ово је заправо појава која се у српском језику догађа када неко "сриче". Мана овог приступа (која генерално и деци и људима са дислексијом прави проблем) је постојање изузетака у изговору.

Конверзија фонема у звук Када дођемо до овог корака, то значи да смо од текста дошли до листе фонема. Постоје три начина како рачунар изговара фонеме:

- **коришћење снимака како људи изговарају те фонеме** - код овог приступа, потребно је да постоји доста снимака људске особе како изговара различите реченице. Реченице се разбијају на речи, а речи на фонеме.

Овакав приступ даје говор који највише подсећа на људски. Често се користи код машина које немају много ствари да изговоре (на пример телефонске секретарице). Мана овог приступа је што је углавном ограничен на један глас (једног говорника по језику).

- **синтетизација фонема генерисањем основних звучних фреквенција**, попут музичког синтетизатора - код овог приступа синтетизатори могу да мењају редослед снимљених звукова и креирају разне комбинације, па и да говоре речи које нису познате ни у речницима. Пример где је ово врло корисно, је код рачунара који се користе у сателитским везама и навигацији. Ту често постоји потреба да се изговоре имена небеских тела или појава која су врло компликована за изговор и нису део стандардног вокабулара. Такође је могуће мењати и глас (мушко, женско, дете) мењањем фреквенције говора. Данас, ипак, први споменути метод има такође велике могућности, јер рачунари садржи велике библиотеке говора тако да и ти синтетизери гласа могу да кажу свашта, а предност у односу на други метод је што имају природну боју гласа за разлику од синтетизера фреквенције који звуче веома вештачки, роботски.
- **понашање механизма људског говора** - ово је практично најбољи метод изговарања, али је и најкомплекснији, па стога и најслабије истражен и примењен. Артикулисање гласова по узору на човека би подразумевало да аутоматски систем опонаша гласовни апарат човека и сва његова механичка, електрична и акустична својства. [3]



Слика 1: Методе генерисања гласа: лево, принцип методе засноване на примени звучних записа људског говора; десно, примена методе генерисања фреквенција [1]

[1] слика преузета са: `https://www.explainthatstuff.com/how-speech-synthesis-works.html`

1.3 Линдо гласовни асистент

Линдо нуди разне услуге и функционалности које корисник може употребити за добијање разних података или управљање појединим процесима. Управљање се може свести на управљање процесима у оквиру оперативног система или, посредством Ардуино платформе, на управљање другим уређајима. Већина тих функционалности се заснива на примени библиотека и апликационих програмских интерфејса (енг. *Application Programming Interface*; у даљем тексту АПИ). Предности употребе веб АПИ сервиса, је једноставност употребе и то што је програмеру остављено да развија услуге на основу већ постојећих функционалности, без потребе да пише функционалности од нуле. С друге стране, мана је неопходност постојања интернет конекције како би те функционалности могле бити коришћене.

Може се рећи да је Линдо асистент **отвореног кода** (енг. *open-source*) и целокупан изворни код је доступан за употребу и прилагођавање. Код се може преузети са *Github* репозиторије на следећој адреси:
`https://github.com/NenadPantelic/Voice-assistant`

Линдо користи латинично писмо као подразумевано, и на енглеском и на српском језику. Из тог разлога ће и у овом раду код сваког примера: уобичајених или кључних речи, структура команди, улазних инструкција које корисник задаје и излазних порука које Линдо изговара, текстови и поруке бити исписани на латиничном писму.

2 Архитектура апликације

Апликација је развијена по шаблону сличном Модел-Поглед-Контролер (енг. *Model-View-Controller*), ако изузмемо постојање Погледа као компоненте, с обзиром да се ради о конзолној апликацији и то апликацији чији интерфејс размене података користи глас као основни медијум уместо, најчешће коришћеног, визуелног. Такође треба напоменути постојање сервисног слоја на ком се заснива бизнис логика апликације. Према томе, овај шаблон би могли назвати Модел-Сервис-Контролер. Овај "хибридни модел" садржи следеће слојеве:

- **слој података, модел** - подаци са којима ради апликација се налазе у форми **JSON** (енг. *JavaScript Object Notation*) структуре података. Конкретно, у питању су уобичајене речи, кључне речи, команде, језици али и звучне датотеке у *mp3* формату. Подразумевано је да је овај део апликације статичан и не садржи никакву логику (код), већ се у њега искључиво складиште подаци. Предност овог приступа је једноставност употребе, као и брзина употребе, с обзиром да се подаци из JSON структуре преводе у хеш колекције - Python речнике. С друге стране, мана је скалабилност оваквог решења. За систем са великим бројем команди, односно великим бројем структура, јавља се проблем конзистентости података и праћења ажурираности података у датотекама. Измене у коду неке методе често изискују и мењање JSON структура или ажурирање података у њима, што може бити незгодно, нарочито ако од те методе зависе структуре у више различитих датотека.

- **сервисни слој** - ово је слој у коме се одиграва већи део бизнис логике апликације. Сервис је заправо класа која треба да поштује принцип јединствене одговорности (енг. *single responsibility*) и која је задужена за извршавање одређене функционалности виртуелног асистента. Као пример можемо навести препознавање говора - постоји један сервис чија је улога конверзија говора у текст и то је све што се од те класе очекује.

- **контролерски слој** - ово је слој који представља везу између корисника и бизнис логике и, како му и име каже, контролише извршење свих сервисних акција у апликацији.

Ако би ствари упростили, корисник комуницира са контролерским слојем, контролерски слој се ослања на сервисни, а сервисни на слој података, чиме се мање-више добија линеарни ток извршења.

Треба напоменути да у оквиру апликације постоје и неки помоћни слојеви и компоненте. У наставку следи детаљније појашњење сваког од ових слојева.

2.1 Модел - слој података

Као што је већ напоменуто, слој података је статични слој без икакве логике који искључиво представља слој у коме се подаци складиште. Подаци са којима апликација ради су језици, уобичајене (устаљене) речи, кључне речи, команде и аудио датотеке говора.

- **језици** - тренутно подржани језици су српски и енглески, који је уједно и подразумевани. Језици се дефинишу својим кодом и то према **ISO 639-1** номенклатури.
- **уобичајене речи** - речи које се углавном могу наћи у било којој команди гласовном асистенту. Примера ради то би биле речи попут: Линдо, молим, те, уради, помози, хоћеш, можеш, ли, бројни предлози, заменице и слично... За сваки језик треба да постоји по једна датотека са овим речима.

```
{
   "usual_words":["lindo", "linda", "please",
   "should", "could","tell", "say", "listen",
   "speak", "help", "hello", "hi","bye", "good bye",
   "yo", "the", "a", "an"],
   "conjunctions":["and", "or", "nor", "for", "but",
      "yet", "so" ],
   "pronouns":["i", "you", "he", "she", "it", "we",
   "they", "me", "them", "her", "his"],
   "prepositions":["at", "in", "to", "of", "by"]
}
```

Листинг 1: Уобичајене Линдо речи за енглески језик

```
{
   "usual_words":[
      "lindo", "linda", "molim", "treba", "можеш",
      "hoćeš", "reći", "reci", "kaži", "slušaj",
      "govori", "pomozi", "pomogneš", "kažeš",
      "zdravo", "ćao", "pozdrav", "ej", "hej"],
   "conjunctions":["i", "ili", "ni", "za", "ali", "ve
      ć", "pa", "niti" ],
   "pronouns":[
      "ja", "ti", "on", "ona", "ono", "mi", "oni", "
         mene", "tebe",
      "tobom", "njega", "njih", "nju",],
   "prepositions":["kod", "u", "na", "ka", "od", "sa"
      ]
}
```

Листинг 2: Уобичајене Линдо речи за српски језик

- **кључне речи** - речи које су карактеристичне за команду. Свака команда има речи по којима се одређује. Структура ове JSON датотеке изгледа тако да се ради о низу JSON података од којих сваки има следећу структуру:

```
{
"command_id":"identifikator komande",
      "words":{
         "word_1":vrednost,
         ....
         .....
         "word_n":vrednost
      }
}
```

Листинг 3: Структура JSON објекта за кључне речи команде

Поље **command_id** представља идентификатор команде чије кључне речи се наводе. По типу је универзални јединствени идентификатор (енг. *universal unique identifier*). Поље **words** је заправо JSON податак који садржи речи и њихове вредности. Свака реч уз себе има придружену вредност - реалан број у распону (0,1], који представља коефицијент тежине те речи за команду, односно меру колико је та реч битна при одређивању команде. Пример података за конкретну команду се може наћи у листингу 4. За сваки оперативни језик потребно је да постоји по једна оваква датотека где су за све извршне команде наведене речи које их одређују.

```
{
      "command_id":"f18e2c7d-5b28-4109-8bd2-7f25a233024f",
      "words":{
         "kakvo":0.25, "će":0.15, "biti":0.15,
         "je":0.15, "vreme":0.4, "vremenska":0.4,
         "prognoza":0.4, "temperatura":0.3,
         "vlažnost":0.3, "pritisak":0.3,
         "vetar":0.3, "oblačnost":0.3,
         "vazduh":0.15, "kiša":0.3,
         "mestu":0.15, "mesto":0.15,
         "lokaciji":0.15, "lokacija":0.1,
         "grad":0.15, "gradu":0.15
      }
   }
```

Листинг 4: Пример JSON објекта за кључне речи команде

- **команде** - кључни део ове апликације су команде. Команда је акција која се извршава услед захтева корисника. Свака команда има својства описана следећом листом:
  - **command_id** - идентификатор команде, по типу универзални јединствени идентификатор у облику ниске (енг. *string*).
  - **service** - назив надлежног сервиса који треба да изврши команду. Може да има и вредност **null** - у том случају за ову команду је надлежан сам контролер. По типу је ниска или **null** вредност.
  - **method** - назив надлежне методе у оквиру сервиса или контролера. Уколико је ова вредност **null**, то значи да је метода иницијализациона, односно циљ јој је да позове следећу команду. По типу је ниска или **null** вредност.
  - **has_args** - индикатор да ли надлежна метода прима аргументе. По типу је логичка вредност (енг. *boolean*).
  - **arg_name** - име аргумента који метода прима (један аргумент по команди, иако то не значи да надлежна метода прима један аргумент. Овај деталь ће бити објашњен у наставку рада). По типу је ниска или **null** вредност.
  - **arg_type** - тип аргумента (уколико га има), на пример *float* (у облику ниске). По типу је ниска или **null** вредност.
  - **need_input** - индикатор да ли се аргумент методе уноси са тастатуре или не. Ово је корисно за случајеве (функционалности) где је потребно да метода прими веома прецизан аргумент, најчешће онај који не садржи само алфанумерике, већ и симболе (нпр. интернет везу (енг. *link*) или *email* адресу). По типу је логичка вредност.
  - **process_input_text** - индикатор да ли се текст који се прослеђује као аргумент (препознати текст из говора) додатно обрађује, односно да ли се из њега уклањају кључне речи или се цео прослеђује даље. По типу је логичка вредност.
  - **input_processing_method** - име методе која треба да спреми аргумент за процесирање, односно да даље обради аргумент (рецимо да из њега извуче неку битну информацију, често подниску или да мапира вредност из улазног текста у неку другу ниску). По типу је ниска или **null** вредност.
  - **tag** - обележје команде, односно њен тип. По типу је ниска или **null** вредност. Уколико је ниска у питању, има неку од вредности из скупа **{initial, invalid, ambiguous, final}**. Овај податак се користи за тражење специјалних команди према тагу, без потребе да се зна идентификатор команде.

- **is_ready** - индикатор да ли је команда "извршна". Ово се директно надовезује на причу о постојању једног аргумента по команди, али не и једног аргумента по сервисној методи. Детаљније објашњење овога ће бити у даљем току рада. По типу је логичка вредност.
- **next_command_id** - идентификатор команде која следи после ове, јер се неке команде нужно надовезују. Према овоме се може закључити да ова структура команди подсећа на једноструко уланчане листе. По типу, универзални јединствени идентификатор у облику ниске (енг. *string*).
- **messages** - по типу је JSON са следећом структуром:
  - success - поруке успешног извршења команде.
  - fail - поруке неуспешног извршења команде.

  Оба својства као вредности садрже JSON податак код кога су својства сви оперативни језици, а вредности одговорајуће поруке. Пример једног таквог JSON-а изгледа овако:

```
1
2
3       "messages":{
4           "success":{
5              "en":"Language is set successfully.",
6              "sr":"Jezik je uspešno postavljen."
7           },
8
9           "fail":{
10             "en":"I have some problems with the
                  language setting. Be kind to try
                  again. Speak clear and loud!",
11             "sr":"Imam problema sa postavkom
                  jezika. Budi ljubazan da probaš
                  opet. Govori jasno i glasno!"
12          }
13      }
```

Листинг 5: Пример JSON објекта за поруке команде

- **descriptions** - JSON стурктура са описима команде на свим оперативним језицима. Представља мета-податак и није релевантан за развој апликације, већ више служи као кратак подсетник шта команда ради.

- **аудио датотеке** - датотеке који чувају говор гласовног асистента. Све датотеке се чувају до последње (финалне) команде или прекида рада апликације услед слања сигнала прекида (SIGINT - Ctrl + C). Тада се све аудио датотеке за ту сесију бришу.

2.2 Сервисни слој

Као што је већ наглашено, сервисни слој је задужен за бизнис логику апликације. Све функционалности гласовног асистента се заснивају на овом слоју. Сервиси су груписани према свом типу и намени:

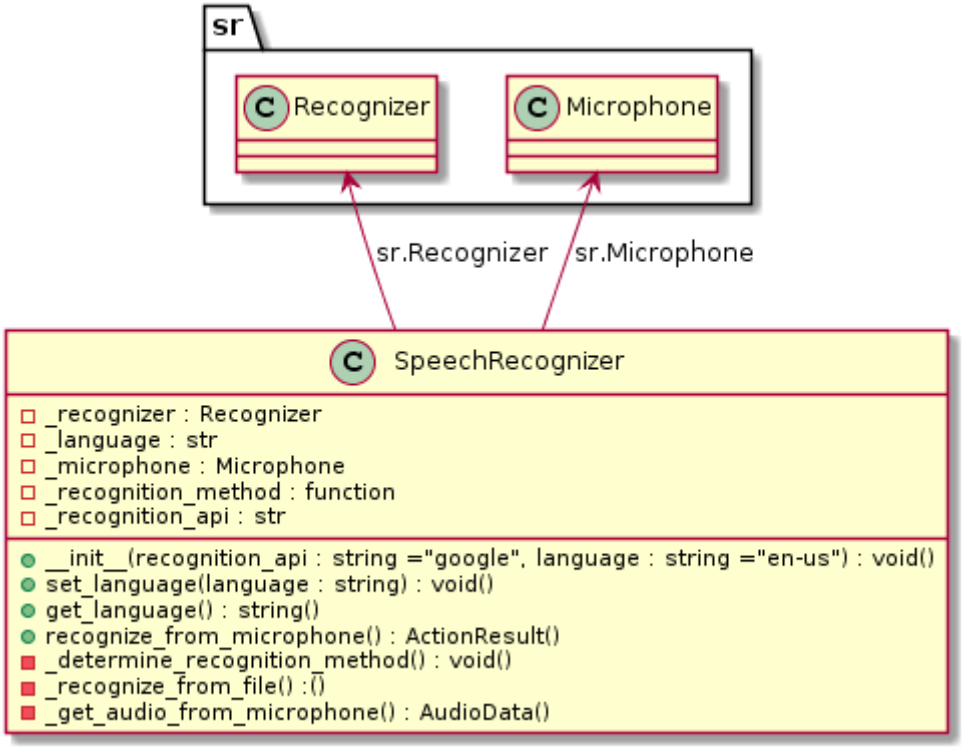
- **сервиси за обраду и генерисање говора**
- **сервис за обраду текста**
- **веб АПИ сервиси**
- **сервис за контролу оперативног система**
- **сервис за рад са претраживачем**
- **сервис за контролу Ардуино платформе**

2.2.1 Сервиси за обраду и генерисање говора

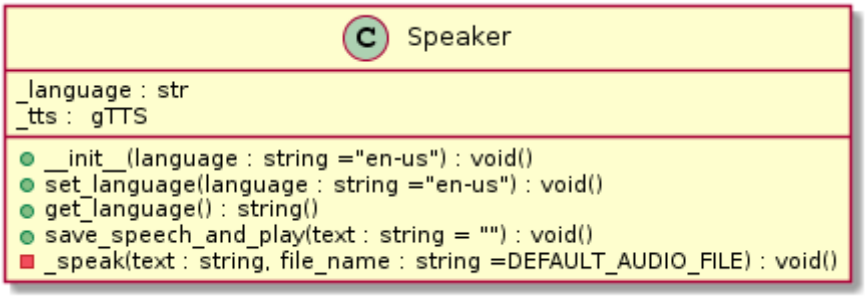
У сервисе за обраду и генерисање говора убрајамо: сервис који врши препознавање говора, као и сервис који генерише говор на основу текста.Код Линдо асистента, препознавање говора се врши применом Гугловог АПИ сервиса за препознавање говора. У основи, сервис за препознавање говора користи библиотеку **SpeechRecognition** која подржава употребу различитих АПИ сервиса са истом наменом: *Google*, *Google Cloud*, *Microsoft Azure*, *IBM Watson*, *Amazon Lex*, *CMU Sphinx*, *Wit.ai*, *Houndify*, *Snowboy Hotword Detection*. Избор је пао на Google сервис на основу три параметра: квалитет услуге, цена и подршка за српски језик.

Препознавање говора је могуће директном анализом говора са микрофона или учитавањем датотеке за анализу. Квалитет препознавања говора зависи од више параметара: квалитет звука (микрофона), близина говорника и гласноћа говора, да ли је говорнику језик којим говори матерњи, којим дијалектом говорник говори, да ли постоји позадински звук итд.

С друге стране, генерисање гласа се заснива на употреби АПИ сервиса текст-у-говор који иначе користи Гугл преводилац. Говор се чува у *mp3* формату и та датотека се репродукује, чиме се имитира говор асистента. Генерисани звук је могуће и директно репродуковати након завршетка процеса генерисања говора из текста, али је за потребе будућих оптимизација и евентуалне дораде пројекта, то ипак замењено репродукцијом говора из датотеке у коме је сачуван. Могуће оптимизације овог сервиса су кеширање података текста за изговор, јер се поједини говори понављају, нарочито они који представљају префиксе резултата команди (поруке успешног или неуспешног извршења команде) и поруке грешака услед изузетака. Такође, могуће је направити неку врсту обучавања система, тако да се врши самостално семантичко препознавање команди.



Слика 2: Дијаграм класе сервиса препознавања говора [1]



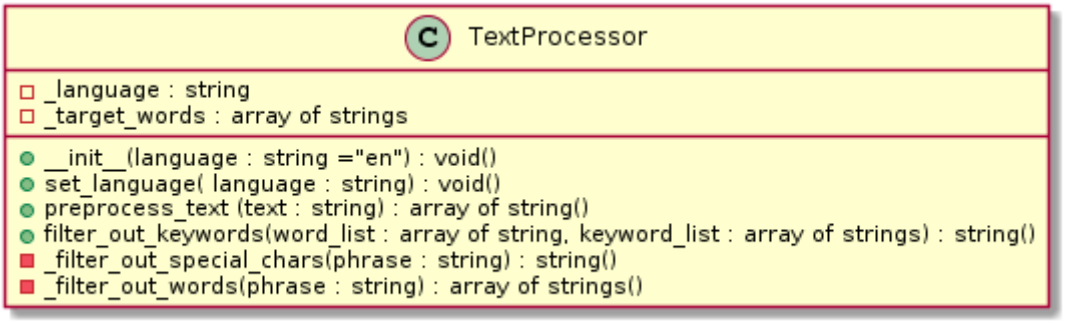
Слика 3: Дијаграм класе сервиса генерисања говора [2]

[1] слике класних дијаграма су креирана од стране аутора рада
[2] дијаграми су креирани употребом алата pynsource

2.2.2 Сервис за обраду текста

Сервис за обраду текста има за циљ обраду изговореног текста. Под обрадом подразумевамо следеће поступке:

- превођење текста у мала слова
- уклањање знакова интерпункције, конкретно било који од знакова из ове групе: `!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~`
- уклањање устаљених речи
- уклањање кључних речи, уколико својство **process_input_text** тражене команде има вредност *true*.



Слика 4: Дијаграм класе сервиса обраде текста

2.2.3 Веб АПИ сервиси

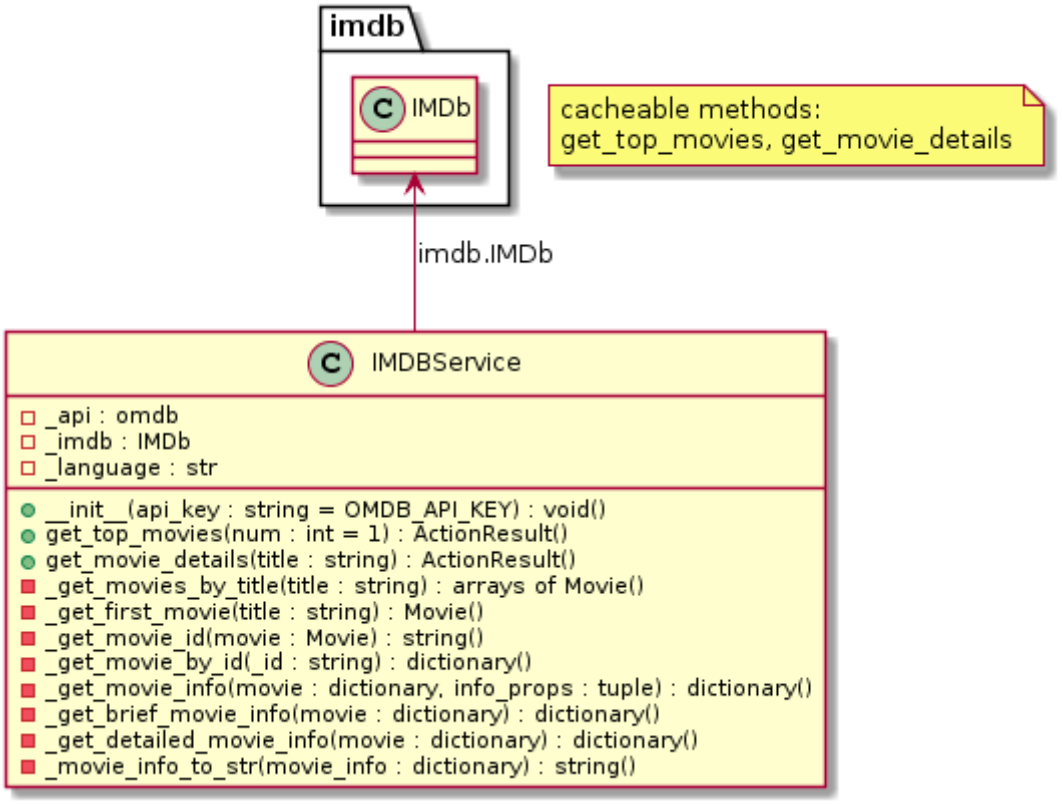
У ову групу сервиса спадају сви они сервиси који позивају неки услужни веб АПИ (енг. *Web API*) кроз одговарајућу библиотеку. С обзиром да се ради о веб АПИ позивима, методе које директно потражују податке од веб АПИ сервиса, су кеширајуће, односно резултати њихових позива се кеширају, како би се смањио број позива, чиме се процес убрзава, чувају ресурси и спречава евентуално пробијање АПИ лимита. За ове потребе се користи **LRU - Least Recently Used** кеш. У наставку следи опис свих сервиса из ове групе.

IMDBService је сервис који пружа могућности добављања детаља о филмовима са сајта **Интернет базе филмова** (енг. *Internet Movie Database - IMDB*). Функционалности које овај сервис пружа кориснику су могућност добијања података о неком филму, као и могућност добијања листе најбољих филмова према оценама са споменутог сајта.

Сервис комуницира са OMDB REST API сервисом кроз библиотеку **omdb**, али и директно потражује податке од IMDB сајта кроз библиотеку **IMDbPY**. Због предности и недостатака ових библиотека, користе се обе како би допуњавале једна другу. Библиотека IMDbPY је боља у претрази филмова по имену, и то методом претраге свих филмова са сличним именом - нека врста фази претраге. Тим поступком се извлачи идентификатор филма који IMDB додељује сваком

филму. Такође, ова библиотека се користи да би се добили подаци о најбоље рангираним филмовима (максимално 250 филмова).

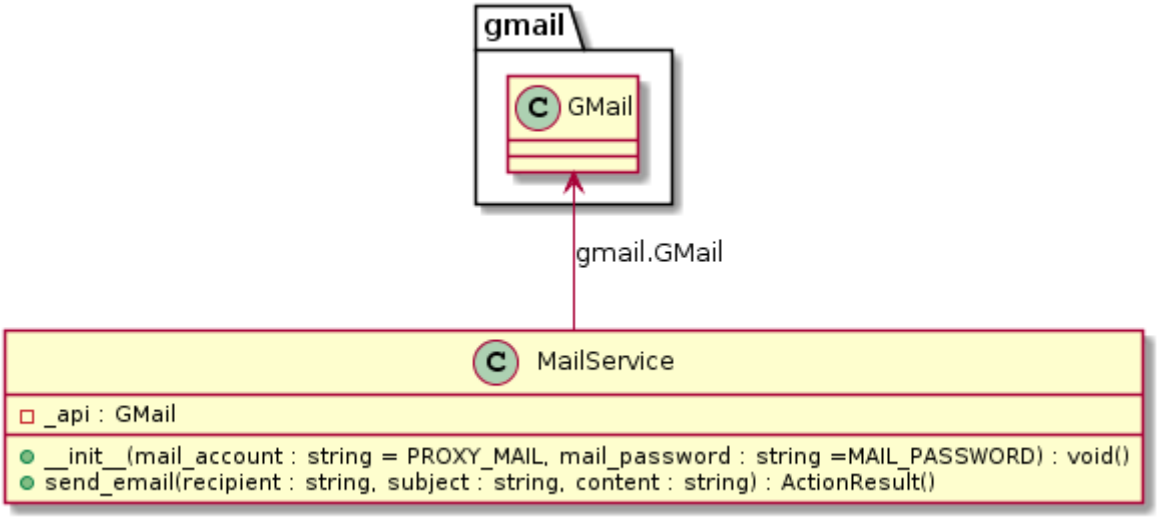
Када се извуче идентификатор, на основу њега се употребом omdb библиотеке претражује филм и добијају подаци. Предност ове библиотеке је детаљност информација које се добијају - много корисних података о филму се добија позивом методе класе из ове библиотеке.



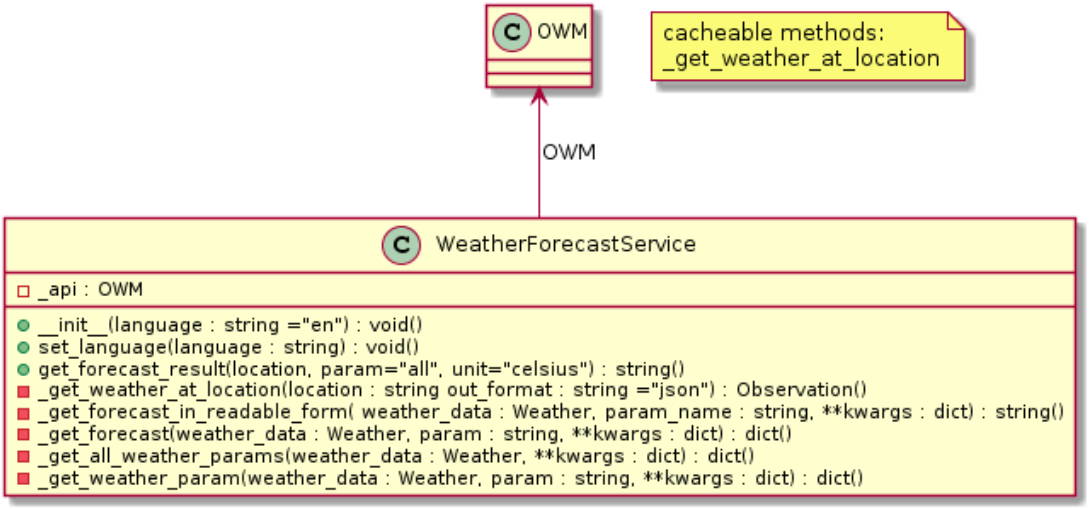
Слика 5: Дијаграм класе IMDB сервиса

MailService пружа услугу слања електронске поште. Овај сервис има само једну јавну методу која користи услугу **Gmail API** кроз **gmail** библиотеку. Основни параметри методе слања поште су прималац (адреса електронске поште), тема поруке и текст поруке. Такође је могуће додати датотеку у прилог или уградити HTML код. У конфигурационој датотеци су подешени параметри адресе електронске поште за слање порука.

WeatherForecastService је сервис намењен одређивању временске прогнозе за жељену локацију. Сервис се ослања на употребу библиотеке **pyowm** који користи **OpenWeatherMap API** за дохватање података о временској прогнози. Локација на којој се тражи временска прогноза се задаје у форми ниске (стринга), али може да прими и идентификатор локације или географску локацију у виду геограске ширине и дужине. За вредности се подразумевано користе метричке вредности (степени Целзијусове скале за температуру, m/s за брзину ветра, mbar за притисак, проценти за облачност и влажност ваздуха, текстуални опис за кишовитост...)

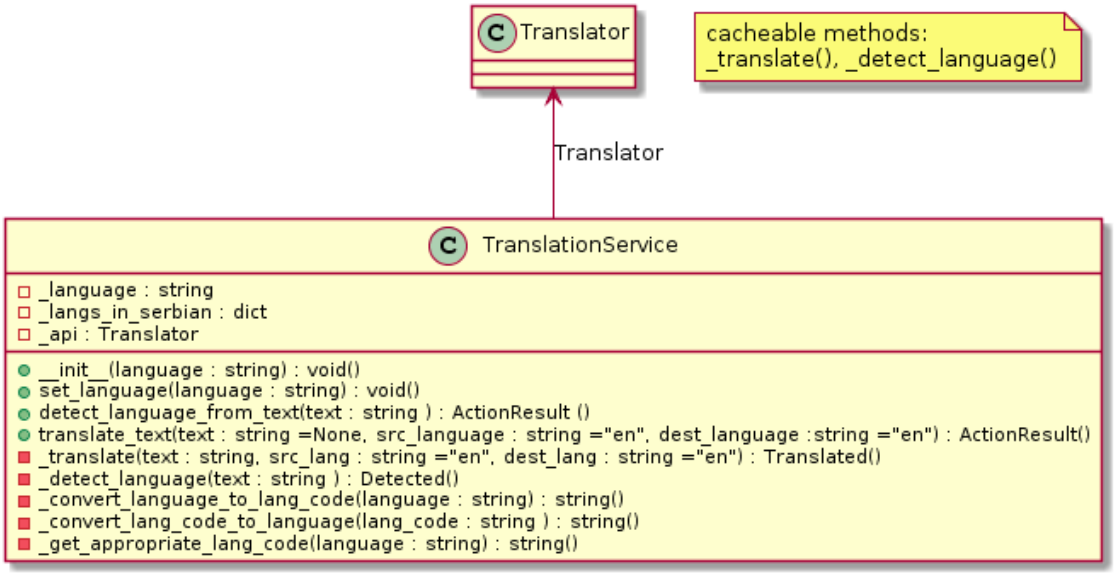


Слика 6: Дијаграм класе mail сервиса



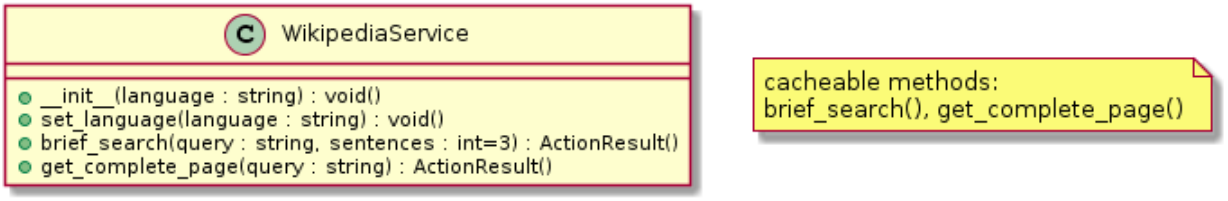
Слика 7: Дијаграм класе сервиса временске прогнозе

TranslationService је сервис који пружа могућност превода текста са било ког изворног језика, познатог Гугл преводиоцу, на било који језик. Овај сервис такође, пружа услугу детекције језика на ком је текст изговорен. Преводилачки сервис се ослања на услуге Гугл преводиоца, односно **Google Translate API** кроз **googletrans** библиотеку. Треба додати још, да Гугл преводилац за српски језик подразумева ћирилицу као службено писмо, па тако текст који долази из сервиса који препознаје текст из говора (који је подразумевано на латиници), види као хрватски. Из тог разлога је пожељно превести текст из латинице у ћирилицу уколико је српски језик тај са ког преводимо.



Слика 8: Дијаграм класе сервиса преводиоца

WikipediaService има за циљ дохватање података са Википедије. Подаци се добијају слањем захтева **Wikipedia API** посредством **wikipedia** библиотеке. Подаци се могу добити у виду резимеа који се може наћи на свакој Википедија (енг. *Wikipedia*) страни и то у највише 10 реченица. С друге стране могуће је добити и садржај целе странице са Википедије закључно са референцама и везама.

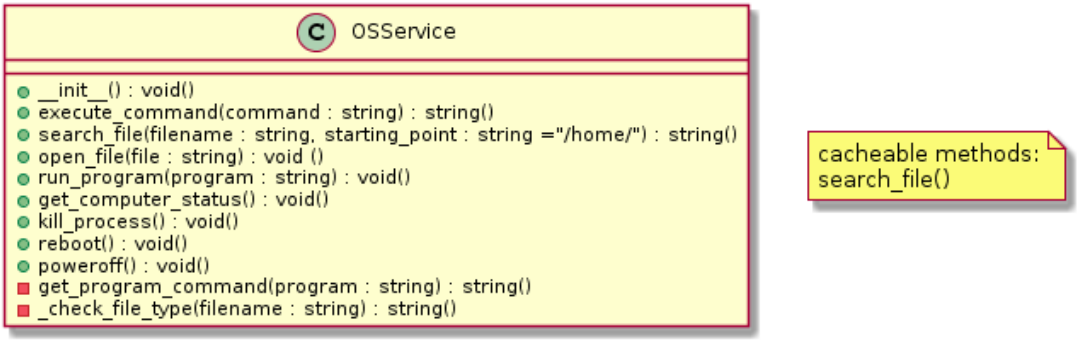


Слика 9: Дијаграм класе Википедија сервиса

2.2.4 Сервис за контролу оперативног система

Сервис за контролу система омогућава контролу оперативног система. Овај сервис није у потпуности функционално активан и развијен, већ је само испробан и то на примеру контроле Линукс оперативног система. Циљ даљег развоја овог сервиса је да се омогући контрола и *Windows* оперативног система, како би се постигла потпуна платформска независност.

Опције које су испробане су: покретање програма, извршење произвољне команде, претрага датотеке, претрага и покретање датотеке, добијање информација о статусу рачунара (активност процесора, искоришћење радне меморије и меморије на диску), рестарт и гашење рачунара.

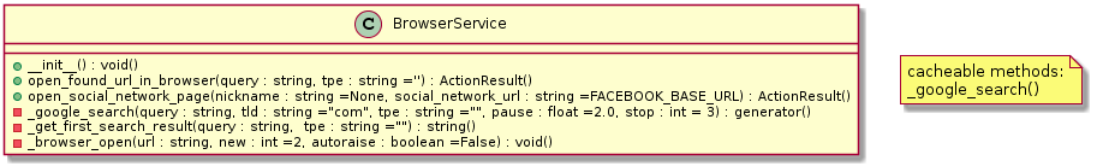


Слика 10: Дијаграм класе сервиса за контролу оперативног система

2.2.5 Сервис за рад са претраживачем

Линдо може да интерагује и са претраживачем. Од функционалности, пружа могућност претраживања појма на Гугл претраживачу и отварања прве стране коју претрага врати. Такође, пружа могућност отварања стране профила са друштвених мрежа - Фејсбук, Инстаграм, Твитер и Линкедин.

Претрага на Гугл претраживачу функционише захваљујући **googlesearch** библиотеци. При претрази, могуће је навести који тип претраге желимо, односно које резултате желимо да добијемо. Тако је могуће претражити слике, вести, апликације, видео клипове, шопинг, књиге или подразумевано. Након извршења претраге, уколико је нешто пронађено, отвара се први резултат претраге. За отварање картица и прозора у претраживачу користи се библиотека **webbrowser**.

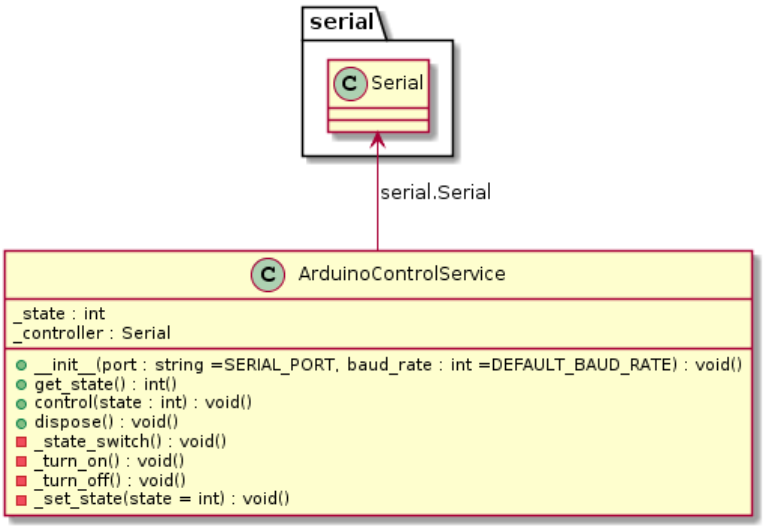


Слика 11: Дијаграм класе сервиса интернет претраживача

2.2.6 Сервис за контролу Ардуино платформе

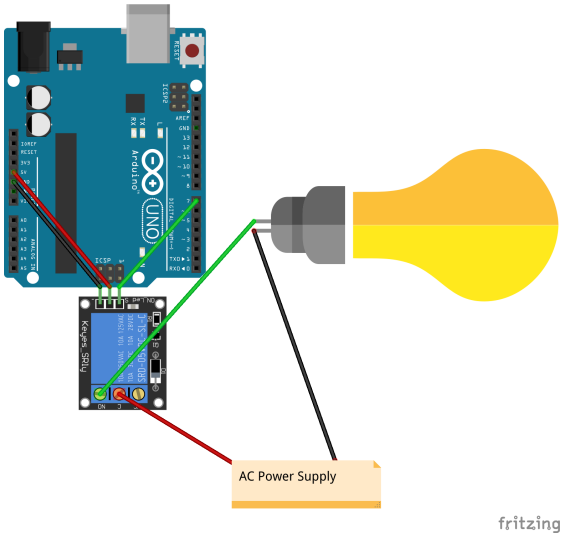
Још једна од могућности Линдо виртуелног асистента је контрола Ардуино платформе, са тенденцијом примене асистента у домену кућне аутоматизације. Асистент преко серијског порта шаље податке Ардуино картици која посредством релеја контролише кућне потрошаче, у овом случају сијалицу.

За комуникацију преко серијског порта користи се библиотека **pyserial**. Ардуино прима податке са серијског порта и на основу тога, сопственом логиком контролише сигнал који шаље ка релеју.



Слика 12: Дијаграм класе сервиса контроле Ардуино картице

Управљање релејем се одвија у два корака. Први корак представља комуникација између корисника и Линдо асистента, где Линдо када одреди шта корисник жели, шаље податке Ардуино картици преко серијског порта. С обзиром да се ради о контроли сијалице, постоје два дискретна стања - сијалица је укључена и сијалица је искључена. Линдо у сервису ово интерно представља преко поља **_state**. Вредност овог поља може бити 0 (сијалица искључена) и 1 (сијалица укључена). На основу команде корисника, Линдо шаље овај податак Ардуино картици која на основу вредности која је стигла, зна који сигнал треба да проследи на један од својих излазних пинова. Конкретно, за потребе контроле релеја, користи пин 7, као што се може и видети са слике 13.

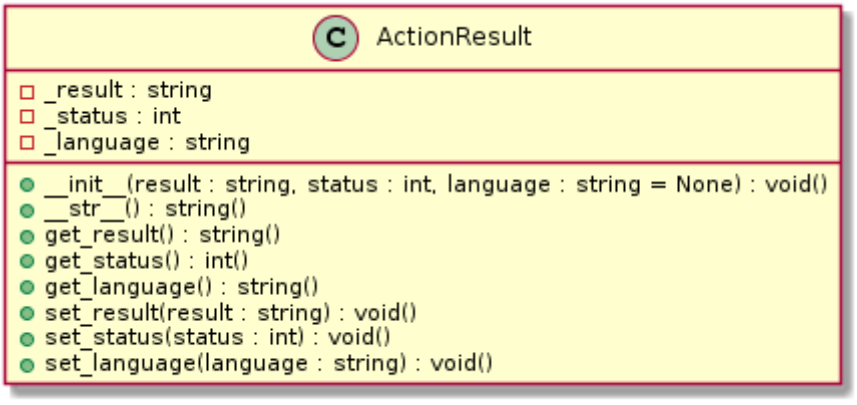


Слика 13: Шема повезивања Ардуино картице, релеја и сијалице (потрошача) [1]

2.2.7 Акциони резултат

Сервисне класе се понашају као АПИ слој и контролерском слоју су видљиве само методе са којима директно комуницирају (јавне методе). Из тих метода као повратни резултат се добија објекат типа акционог резултата - **ActionResult**. Акциони резултат има три поља у својој дефиницији:

- резултат - резултат извршења сервисне методе или порука грешке, уколико је дошло до неке грешке при извршењу методе. По типу је ниска.
- статус - статус извршења сервисне методе. Прима вредност из скупа **{SUCCESS, FAIL, FATAL}**. Ове вредности су по типу цели бројеви са следећим значењем: SUCCESS - извршење сервисне методе успешно извршено, FAIL - при извршењу сервисне методе дошло је до проблема (јавио се изузетак), FATAL - дошло је до неке критичне грешке и даље извршење би требало прекинути. Уколико све прође како треба (SUCCESS), контролерски слој узима из акционог резултата поруку коју носи као терет (енг. *payload*) и додаје поруку која се узима у случају успешног извршења команде (поље messages, подполе success). С друге стране ако дође до грешке (FAIL, FATAL), решавалац изузетака (ExceptionHandler) преводи грешку у поруку намењену кориснику (која је дефинисана за сваки изузетак у конфигурацији) и контролер додаје поруку за неуспешно извршење команде (поље messages, подполе fail).
- језик - језик на ком треба изговорити резултат који даје сервисна метода. Ово је неопходно за случајеве попут оног када се користи преводилачки сервис. Тада се задаје језик у који треба превести неки говор (текст). Према томе, преведени текст би требало изговорити у том циљном језику превода, без обзира на то који оперативни језик Линдо користи.



Слика 14: Дијаграм класе ActionResult

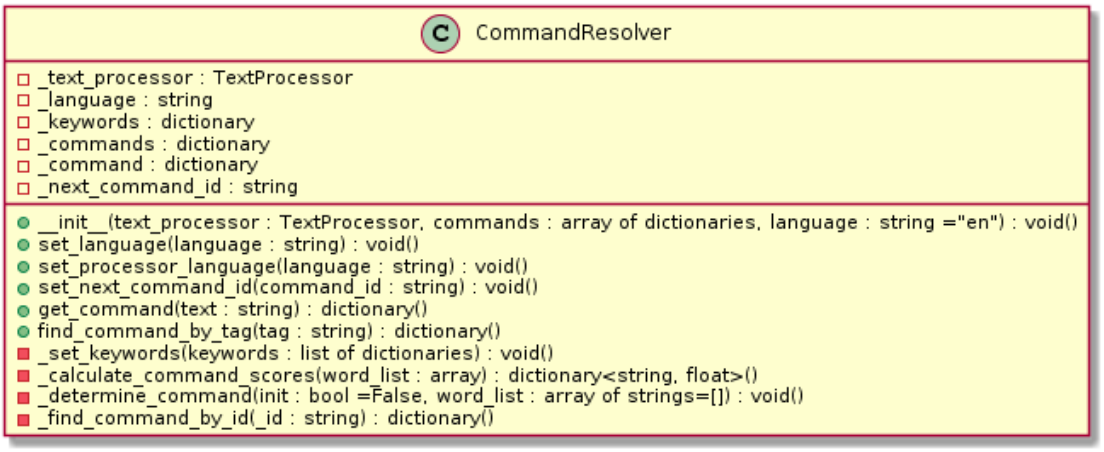
[1]Слика је дело аутора. Шема креирана употребом алата fritzing

2.3 Контролерски слој

Контролерски слој је задужен за контролу сервиса, одређивање жељене команде и њено извршење. У овај слој убрајамо следеће класе:

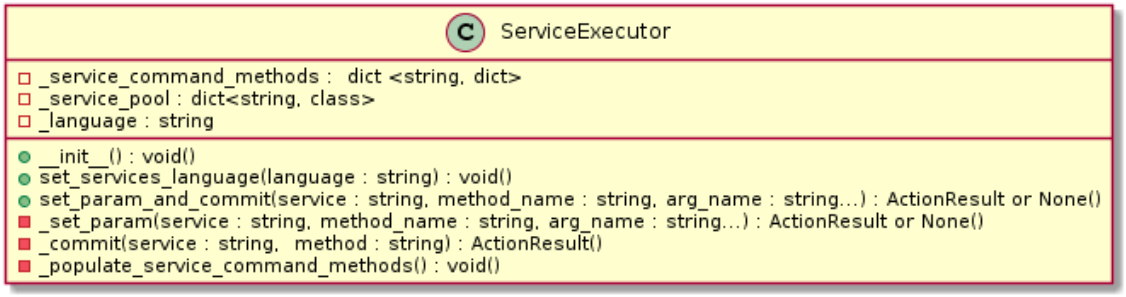
- **command resolver**
- **service executor**
- **controller**

CommandResolver је контролерска класа која одређује команду за извршење на основу обрађеног текста. Такође, ова класа може да претражује команде на основу идентификатора или тага и одређује следећу команду за извршење. Више детаља о логици ове класе ће бити у наредном поглављу.



Слика 15: Дијаграм класе CommandResolver

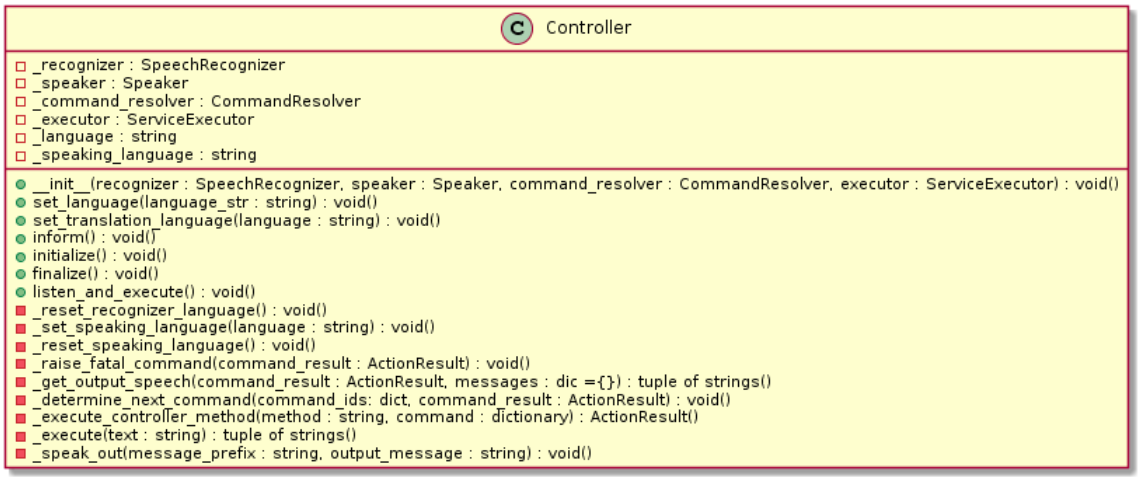
ServiceExecutor је, као што му и име каже, извршитељ команди. Када контролер зна коју команду треба извршити, уколико је у питању команда која се ослања на услугу коју неки сервис пружа, он позива ову класу која даље извршава команду, уколико је она извршна, или поставља параметре потребне за извршење извршне команде која следи. Сви сервиси се директно позивају из ове класе и то само оне методе које су наведене у структури команде (поље *method* у JSON-у). Сервиси се чувају у интерном кеш објекту (*service pool*). Логика ове класе ће бити појашњена у наредном поглављу.



Слика 16: Дијаграм класе ServiceExecutor

Controller је главна контролна класа ове апликације. Она од корисника прима инструкције и, ако је могуће, извршава команде. У себи има интегрисане четири инстанце:

- **препознаваоца говора - SpeechRecognizer**
- **инстанцу класе која одређује команду на основу текста - CommandResolver**
- **извршитеља команде - ServiceExecutor**
- **инстанцу која изговара текст - Speaker**



Слика 17: Дијаграм класе контолера

2.4 Помоћне компоненте

Поред ова три слоја на којима почива архитектура и логика апликације, треба споменути и пар помоћних модула који играју битну улогу у њеном фунцкионисању.

Конфигурација - конфигурација целе апликације се налази у једној датотеци. Та датотека садржи константе и податке о: логеру (сваки догађај и битан процес се уписује у лог датотеку ради лакшег праћења и отклањања евентуалних проблема), подржаним језицима, статусима команди, путањама до битних података (слој података), порукама у случају јављања изузетака и грешака, креденцијалима и АПИ кључевима, базним линковима друштвених мрежа, параметрима серијског порта итд. Предност овог приступа је адаптација на могуће измене. Нпр. у случају да желимо да додамо нови изузетак, односно поруке везане за њега, то радимо овде. Исто важи ако хоћемо да додамо подршку за нови језик или неки нови АПИ кључ. Све се налази на једном месту, чиме се олакшава процес измене кода и/или функционалности.

Изузеци - рад са изузецима је смештен у ову компоненту. Конкретно, ради се о дефинисању прилагођених (енг. *custom*) изузетака и руковању изузецима у случају да се негде јавио неки проблем. Класа **ExceptionHandler** садржи методу **get_exception_message(e, language)** која на основу типа изузетка зна коју поруку треба да врати контролерском слоју, који ће ту поруку репродуковати кориснику. Треба напоменути да се у лог датотеку уписује стек позива узрока грешке, док је то од корисника сакривено и њему се враћа корисниј порука (*user-friendly*). У прилагођене изузетке апликације спадају:

1. **GoogleSearchException** - диже се уколико Гугл претрага не врати никакав резултат за тражени термин претраге.
2. **CommandException** - диже се уколико се јавио проблем са командом (интерни проблем).
3. **VoiceAssistantException** - диже се уколико дође до неке нерешене интерне грешке.

Компонента услужних функција - садржи два типа функција:

1. услужне функције - помоћне функције које пружају неку услугу виталним слојевима апликације. Овде убрајамо функције попут функције која конвертује латиницу у ћирилицу, учитава JSON датотеку, прилагођава структуру података за неку методу, генерише универзални јединствени идентификатор, брише mp3 датотеке на крају сесије итд.
2. функције екстракције података - функције које из текста које добију, извлаче само онај део који ће бити коришћен у некој од сервисних метода.

3 Бизнис логика апликације

Бизнис логика апликације је смештена у сервисни и контролерски слој. Контролерски слој директно комуницира са корисником, одређује команде и извршава их. С друге стране, сервисни слој је тај који омогућава све функционалности асистента.

Линеарни ток извршења (*workflow*) и информација (*dataflow*) изгледа овако:

- контролер - сервис за препознавање говора
- контролер - одређивање команде
- контролер - извршилац команде
- контролер - сервис за гласовну репродукцију текста

Прва команда којом Линдо започиње рад је иницијализациона команда. Иницијализациона команда има за циљ да поздрави корисника и пита га за језик који жели да користи. Та порука гласи:
en: Hi, I'm Lindo voice assistant. Choose the operating language. The default option is English!
sr: Zdravo, ja sam Lindo glasovni asistent. Izaberite operativni jezik. Podrazumevani jezik je engleski!
Корисник бира неки од понуђених језика (енглески или српски), а уколико проба да одабере неки од непостојећих језика, добиће грешку:
en: I have some problems with the language setting. Be kind to try again. The only options are English and Serbian. Speak clear and loud!
sr: Imam problema sa postavkom jezika. Jedine opcije su engleski i srpski. Budite ljubazni da probate opet. Govorite jasno i glasno! Од корисника се затим очекује да понови команду, тј. изабере оперативни језик. При избору језика, било који облик команде који у себи садржи назив језика је валидан. Према томе, за избор енглеског језика је битно да команда садржи било коју од ових речи: **english , default, engleski, podrazumevan** (као корен речи подразумевана, подразумевано...). За избор српског језика, команда треба да садржи неку од речи: **serbian, srpski**. Примера ради то би могле бити ове команде:
en: Hi Lindo, I'm choosing English!
sr: Zdravo Lindo, moj izbor je srpski!
Оперативни језик је могуће променити у било ком тренутку. Кључне речи команде којом се врши промена оперативног језика су: **promeni, izbor, zameni, jezik, operativni, novi** и **switch, change, replace, language, operating, new**. На основу ових речи, команде које се могу користити су:

- **Promeni operativni jezik.**
- **Zameni operativni jezik.**
- **Switch the operating language.**
- **Change the operating language.**

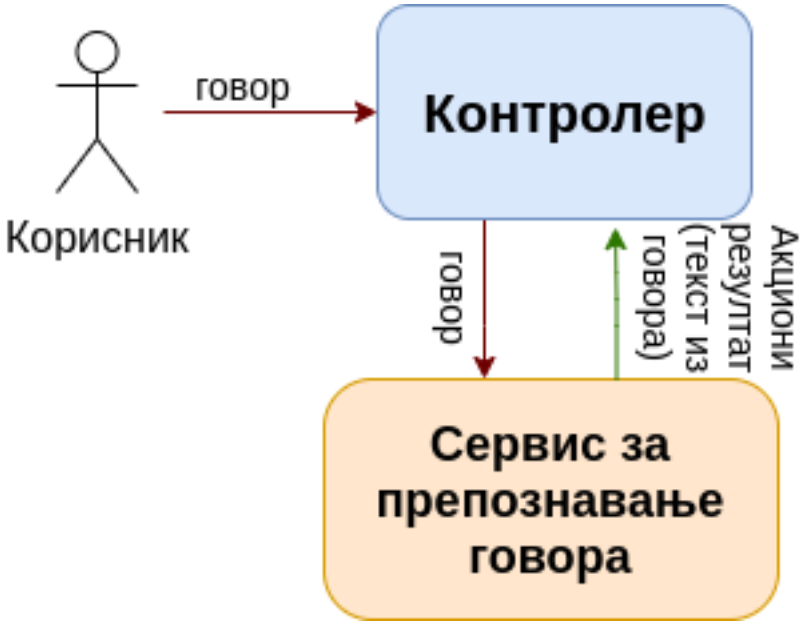
Након овога, од корисника се тражи да наведе нови оперативни језик.

3.1 Ток контролер - сервис за препознавање говора

Говор корисника обрађује сервис за препознавање говора. Уколико сервис препозна шта је корисник рекао, вратиће објекат акционог резултата са текстом енкапсулираним у објекат и статусом SUCCESS. С друге стране, уколико сервис не успе да препозна говор корисника или дође до грешке у позиву Гугл АПИ сервиса, сервис за препознавање говора диже изузетак који ExceptionHandler преводи у поруку уз статус FAIL. Порука која се пакује у ActionResult објекат онда изгледа овако:

- говор не може бити обрађен:
  en: Speech cannot be analyzed or recognized!,
  sr: Vaš govor ne može biti obrađen ili prepoznat!
- проблем са захтевом АПИ сервису:
  en: Request error problem. Check API limits and connectivity status!,
  sr: Problemi sa slanjem zahteva. Proverite API limit i status mreže!

Уколико је све прошло како треба, препознати текст се прослеђује даље како би се одредила команда.



Слика 18: Блок дијаграм тока контролер - сервис за препознавање говора

3.2 Ток контролер - одређивање команде

Када се одреди текст из говора корисника, следи одређивање команде. Алгоритам одређивања команде би изгледао овако:

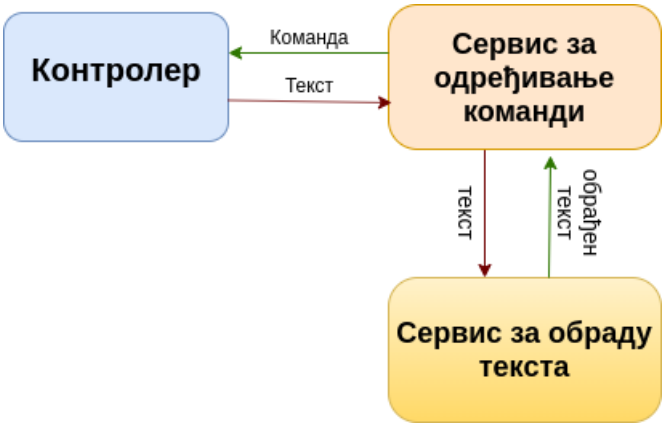
1. применом метода текст-процесора, обради улазни текст - преведи слова у мала, уклони знакове интерпункције и уобичајене речи
2. на основу кључних речи, одреди скор за сваку команду по принципу псеудокода датом у Алгоритму 1.
3. након примене алгоритма одређивања највероватније команде, могући су следећи случајеви:
   - метода је вратила само једну команду - алгоритам је сигуран која команда је у питању
   - метода је вратила више од једне команде - више команди има исти скор. У том случају се враћа команда са тагом **ambiguous**, јер алгоритам није сигуран која од команди је тражена. Излазна порука ове команде изгледа овако:
     en: I'm sorry, but I'm not sure what you want. Your command is ambiguous. There are a few options. Try again, please!,
     sr: Žao mi je, ali nisam siguran šta želite. Vaša komanda nije jednoznačna. Postoji više mogućih opcija. Probajte ponovo!
   - није могуће пронаћи команду на основу инструкције корисника. У овом случају враћа се команда са тагом **invalid**. Излазна поруке ове команде изгледа овако:
     en: I'm sorry, but I'm not sure what you want. I cannot precisely determine your command. Try again!,
     sr: Žao mi je, ali nisam siguran šta želite. Ne mogu da odredim precizno Vašu komandu. Probajte ponovo!
   - корисник је иазбрао финалну команду. Ово је команда којом се завршава рад Линдо асистента - Линдо гаси сесију и брише аудио датотеке и лог датотеку за ту сесију. Кључне речи команде којом се завршава рад апликације су: **slobodan, slobodna, idi, ugasi, prekini, povuci, završi** и **free, go, off, down, sleep, abort. exit**. Други случај у коме се враћа ова команда, је када корисник пошаље сигнал прекида апликацији (SIGINT - Ctrl + C). Излазна поруке ове команде изгледа овако:
     en: Ok, I'm going off now. I hope so you enjoyed. See you next time!,
     sr: U redu, ja se povlačim sada. Nadam se da ste uživali. Do narednog viđenja!

4. Уколико је команда успешно одређена, следећи корак је одређивање аргумента за надлежну сервисну методу, уколико својство команде **has_args** има вредност true. За аргумент се подразумевано поставља цео обрађени текст. Ипак, уколико је вредност својства **process_input_text** true, текст се додатно обрађује тако што се из њега уклањају кључне речи. Текст се даље додаје у структуру команде (додаје у Пајтон речник) и враћа контролеру.

Input: text
Output: commands with the highest score
scores = {}
foreach *command* ∈ *all_commands* **do**
  command_score = 0
  foreach *word* ∈ *command.words* **do**
    if *word* ∈ *text* **then**
      command_score =
      command_score + *text.count*(*word*) ∗ *word.value*;
    end
  end
  scores[command] = command_score
end
max_score = highest score of all scores in scores.values
return all commands with score = max_score

Алгоритам 1: алгоритам за одређивање највероватније команде

Класа **CommandResolver** садржи и поље **next_command_id** које представља идентификатор следеће команде за извршење. Ово поље преузима вредност истоименог поља из JSON-а команде (одељак 2.1, тачка о командама). Уколико је ова вредност различита од null, при следећем позиву методе за одређивање следеће команде, одмах ће бити враћена команда са тим идентификатором, без рачунања скорова за одређивање највероватније команде.



Слика 19: Блок дијаграм тока контролер - одређивање команде

3.3 Ток контролер - извршилац команде

Претпоставља се да, чим смо дошли до овог корака, имамо команду која треба да буде извршена. То је нека од команди из датотеке дефинисане у тачки о командама у одељку 2.1. Како је већ описано, својство **service** сваке команде може имати вредност алијаса неког од дефинисаних севиса или вредност null (None). Уколико је у питању назив алијаса неког од сервиса (**wikipedia, owm, translation, mail, imdb, browser, system, arduino)**, значи да је за ту команду надлежан наведени сервис. Ако је ипак вредност null, онда је за ту команду надлежан сам контролер.

За извршење команди за које је надлежан сервис, позива се извршилац команди. Извршење команди за које је надлежан контролер је, природно, препуштено контролеру.

Извршилац команди има две јавне методе. Једна је задужена за постављање оперативних језика сервисима (ова метода се позива при избору оперативног језика), док је друга задужена за постављање аргумената и извршење сервисних метода.

Метода задужена за постављање аргумената и извршавање сервисних метода је у директној спрези са причом о томе да свака команда има највише један аргумент, иако њена надлежна метода може имати и више аргумената. Следи објашњење ове појаве.

За неке услуге (функционалности), попут слања електронске поште, потребно је поставити више аргумената (у случају примера слања електронске поште то су прималац, тема и порука). Овај поступак се једноставности ради, али и сигурности тачног извршења, извршава из три команде:

- корисник наводи адресу примаоца
- корисник наводи тему поруке
- корисник наводи садржај поруке

Ово су три команде, али су све везане за исту сервисну методу - методу **send_email (recipient, subject,content)** из сервиса **MailService**. Према томе, свака од ових команди заправо узрокује постављање по једног аргумента циљне методе. Након што се сви аргументи поставе, следи извршење методе. Питање је, на који начин се чувају аргументи за извршење неке методе? То је решено постојањем приватног поља **_service_command_methods** у класи **ServiceExecutor**. Ово поље је по типу речник, и то речник код кога су кључеви називи сервиса, а вредности такође речници. Структура речника, који представљају вредности претходно споменутог речника, изгледа тако да су кључеви називи метода, а вредности су речници оформљени од уређених парова: назив аргумента : вредност аргумента. Структура овог поља је компликована, па је зато боље увидети изглед структуре на примеру из листинга 6.

```
{'wikipedia': {'brief_search': {'query': None}},
'owm': {'get_forecast_result': {'location': None}},
'translation': {'translate_text': {'text': None, 'dest_language':
    None}},
'mail': {'send_email': {'recipient': None, 'subject': None, '
    content': None}},
'imdb': {'get_top_movies': {'num': None}, 'get_movie_details': {'
    title': None}},
'browser': {'open_social_network_page': {'social_network_url': None
    , 'nickname': None}, 'open_found_url_in_browser': {'query': None
    , 'tpe': None}},
'system': {'run_program': {'program': None}},
'arduino': {'control': {'state': None}}}
```

Листинг 6: Структура поља _service_command_methods

У речнику се налазе алијаси свих коришћених сервиса. За сваки сервис, у речнику се налазе само оне методе које су изложене као јавни интерфејс како би биле коришћене од стране контролерског слоја. У листингу 6 је дат изглед речника пре извршења било које команде. Сада, ако би корисник желео да пошаље електронску поруку:

1. поставио би адресу примаоца (mail-send_email-recipient)
2. поставио би тему поруке (mail-send_email-subject)
3. поставио би садржај поруке (mail-send_email-content)

Након постављања свих параметара, позива се надлежна сервисна метода. Извршилац зна када су сви аргументи постављени гледајући вредност поља **is_ready** из JSON-а команде, без потребе да проверава да ли су сви постављени. Поље is_ready, у датом примеру, има вредност true за команду која поставља садржај поруке, тј. за последњу команду у низу.

Треба напоменути да пре постављања вредности параметра, вредност параметра се преводи у жељени тип података (на основу вредности поља **arg_type**). Такође, уколико је вредност поља **need_input** true, вредност аргумента се уноси са тастатуре. На крају, уколико је вредност поља **input_processing_method** различита од None, вредност аргумента се додатно обрађује користећи екстракциону методу која је наведена (ове методе потичу из компоненте услужних функција дефинисаних у одељку 2.4).

Резултат који извршилац враћа контролеру је свакако акциони резултат. Уколико је дошло до грешке у извршењу сервисне методе, као порука у оквиру објекта акционог резултата, враћа се порука грешке коју ExceptionHandler враћа на основу типа изузетка, уз вредност статуса FAIL. Уколико ипак извршење прође без грешке, излаз сервисне методе је акциони резултат, са вредношћу статуса SUCCESS, па се тај објекат и враћа контролерском слоју.

Када се акциони резултат врати контролерском слоју, на основу статуса акционог резултата, контролер зна да ли да поруци акционог резултата дода префикс

поруке успешног или неуспешног извршења. Такође зна да ли да прелази на следећу команду или не - нпр. ако постављање теме електронске поште није прошло успешно (сервис за препознавање говора није разумео шта корисник говори), нема потребе да се од корисника тражи да унесе садржај електронске поруке. Код ове ситуације су доступна два приступа решавања:

- очекује се да апликација остаје на истој команди и да ће корисник поновити инструкцију. Мана овог приступа је што се може десити да је дошло до неке функционалне грешке у апликацији, тако да би и поновљена инструкција проузроковала грешку.
- занемарује се све и асистент "заборавља" последњу команду, тако да цео поступак мора да почне од прве команде (у примеру би то значило да корисник треба да почне поново од команде захтева слања електронске поште, јер је асистент "заборавио" докле је овај стао). Мана овог приступа је та што би нека хипотетичка команда могла да има и 5-6 корака поставки аргумената, тако да ако би проблем избио у 5. или 6. инструкцији по реду, корисник би морао да почне све испочетка.

Ипак, ако је статус акционог резултата SUCCESS, контролер издаје наредбу класи која одређује коју команду је корисник изрекао (CommandResolver), да постави идентификатор следеће команде на вредност поља next_command_id из JSON-а текуће команде.

Слика 20: Блок дијаграм тока контролер - извршилац команде

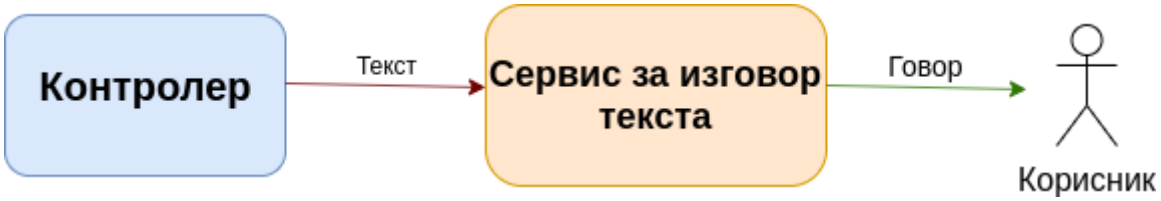
3.4 Ток контролер - сервис за гласовну репродукцију текста

Када се изврши жељена команда, неопходно је саопштити кориснику резултате извршења. За репродукцију резултата говором, контролер користи сервис за гласовну репродукцију текста.

Изговор резултата извршења се одвија у два позива сервису:

1. изговор поруке из поља **messages** - у зависности од тога да ли је команда успешно извршена, тј. да ли је статус акционог резултата SUCCESS или FAIL. Уколико је команда успешно извршена, узима се порука под кључем success, иначе се узима порука под кључем fail.
2. изговор поруке из акционог резултата - порука изузетка (ако је статус FAIL) или резултат извршења методе (ако је статус SUCCESS). Ова порука се изговора на језику који је означен вредношћу поља **language** акционог резултата.

Између ова два изговора, постоји мали временски интервал паузе (око 1 секунда) у коме се дешава чување датотеке, упис у лог датотеку и поставка језика за изговор (језик из објекта акционог резултата).



Слика 21: Блок дијаграм тока контролер - сервис за гласовну репродукцију текста

4 Примери употребе

Након што је разјашњено како Линдо функционише, следи демонстрација функционалности које Линдо може да изврши. Код сваког од примера фунцкионалности, биће дата листа кључних речи од којих је могуће саставити команду. На једном примеру сервисних функционалности ће бити детаљније објашњено како користити услуге Линдо асистента, односно како састављати команде. Остале сервисне функционалности ће бити само побројане уз примере команди на енглеском и српском језику.

4.1 Услуга читања временске прогнозе

Добијање временске прогнозе захтева од корисника да искључиво користи речи из скупа устаљених и кључних речи. Једина реч ван овог скупа коју треба користити је назив локације за коју тражимо временску прогнозу. Кључне речи на српском језику су **kakvo, će, biti, je, vreme, vremenska, prognoza, temperatura, vlažnost, pritisak, vetar, oblačnost, vazduh, kiša, mestu, mesto, lokaciji, lokacija, grad, gradu**, а на енглеском: **how, what, whats, is, will, be, weather, forecast, like, temperature, humidity, pressure, wind, clouds, snow, rain, location, place, city**.

Рецимо да желимо да сазнамо какво је време на локацији Денвер (САД). Команда: **Kakvo je vreme na lokaciji Denver?** ће успешно вратити прогнозу, док ће команда **Kakvo je vreme na lokaciji Denver i okolini?** вратити поруку да није могуће добити податке временске прогнозе за наведену локацију. Зашто је то тако?

Када се изговори реченица **Kakvo je vreme na lokaciji Denver?**, та реченица пролази кроз обраду текст-процесора. Прво се примењује обавезно процесирање, тако да се добија текст: **kakvo je vreme na lokaciji denver**, односно текст се преводи у мала слова, уклањају се знакови интерпункције и уклањају устаљене речи. С обзиром да за ову команду индикатор даљег процесирања текста има вредност true, текст-процесор из текста уклања и кључне речи. Текст након тог поступка има вредност **denver** и та вредност се шаље у надлежну сервисну методу која очекује локацију за коју се тражи временска прогноза.

Ако се исте методе обраде примене на текст **Kakvo je vreme na lokaciji Denver i okolini?**, добија се **denver okolini**, што не представља валидну географску локацију. Уколико би у кључним речима за ову команду постојала и реч **okolini**, та реч би била уклоњена у процесирању текста, па би команда резултовала враћањем временске прогнозе за град Денвер. АПИ који се користи у надлежном сервису нема могућност враћања података на српском језику. Најсроднији језик који је доступан је хрватски. Резултат изгледа овако:

```
oblačnost: 20,
pritisak: 1019,
brzina vetra: 1.5,
vlažnost vazduha: 80,
```

```
minimalna dnevna temperatura: 5.0,
maksimalna dnevna temperatura: 6.11,
prosečna dnevna temperatura: 5.3,
detaljniji opis: blaga naoblaka,
trenutak merenja: 04-10-2019 13:47:20
```

Друге команде које такође задовољавају форму су: **Kakvo je vreme u gradu Denver?**, **What's the weather like at location Denver?**, **What is the weather like in Denver?**.

4.2 Услуге претраге филмова

IMDB сервис пружа две услуге: добијање детаља о филму и добијање листе, до 250 најбољих филмова. Овде треба напоменути две ствари: 1) подразумевани језик на ком се изговарају детаљи о филму је енглески (јер су подаци са IMDB на енглеском), 2) могу да се десе проблеми са препознавањем бројева на српском језику, некада се деси да их преведе као текст, а некада у цифарном облику (три и 3), нарочито за једноцифрене бројеве.
Прва услуга подразумева тражење прегледа најбољих филмова према IMDB сајту. Кључне речи: **details, movie, info, information, informations, film, imdb** и **detalji, detalje, filmu, film, info, informacije, imbd, podatke, podaci**.

Команде на енглеском би могле изгледати овако:

- **I need some movie details.**
- **Tell me some film information.**
- **Can you provide some IMDB info?**

Ако је све прошло како треба, Линдо ће одговорити са: **That sounds fine. Tell me the title of the movie you want, as precise as you can.** Корисник онда треба да каже име филма који жели, нпр. **Green Book**.

Линдо ће изговорити следећи текст:

```
title: Green Book,
year: 2018,
released: 16 Nov 2018,
runtime: 130 min,
genre: Biography, Comedy, Drama, Music,
director: Peter Farrelly,
actors: Viggo Mortensen, Mahershala Ali, Linda Cardellini, Sebastian
Maniscalco,
plot: A working-class Italian-American bouncer becomes the driver of
an African-American classical pianist on a tour of venues through the
1960s American South.,
language: English, Italian, Russian, German,
imdb_rating: 8.2
```

Команде на српском би могле изгледати овако:

- **Potrebni su mi detalji o filmu.**
- **Želim informacije o filmu.**
- **Daj mi podatke o filmu sa IMDB.**

Након тога, корисник задаје име филма, нпр. **Tesna koža**.

```
title: A Tight Spot,
year: 1982,
released: 24 Sep 1982,
runtime: 92 min,
genre: Comedy,
director: Mica Milosevic,
actors: Nikola Simic, Milan 'Lane' Gutovic, Rahela Ferari, Ruzica Sokic,
plot: In this light, fluffy comedy, a low-level clerk cannot
make ends meet because his brood is in no way economically cooperative:
his daughter is a lawyer looking for work,
unsuccessfully; his...,
language: Serbo-Croatian,
imdb_rating: 8.2,
```

За услугу добијања детаља о најбољим филмовима, кључне речи су: **top, best, movies, what, imdb, rating** - енглески и **top, najbolji, najbolje, filmovi, filmove, koji, imdb, rejting** - српски.

Пример команди на енглеском изгледа овако:

- **What are the best movies by IMDB?**
- **Tell me top rated movies.**

Одоговор на ову команду је: **Sure, tell me what is the number of top movies you want? The maximum number is 250.** Линдо ће затим тражити од корисника да каже колико филмова са топ листе тражи. Ако затражи број који је мањи од 1 или већи од 250, добиће грешку: **Yikes. Some obstacles disabled this action. Maybe you asked for less than 1 or more than 250 movies. Try again!** за оперативни језик енглески, односно **Jao. Nešto je iskrslo pri ovoj akciji. Možda ste tražili manje od 1 ili više od 250 filmova. Probajte ponovo!** за српски. Ако ипак каже број који је у жељеном опсегу, рецимо два, Линдо ће изговорити овај резултат:

```
title: The Shawshank Redemption,
genre: Drama,
imdb_rating: 9.3,

title: The Godfather,
genre: Crime, Drama,
imdb_rating: 9.2,
```

Пример команди на српском изгледа овако:

- **Koji su najbolji filmovi po IMDB?**
- **Navedi mi filmove sa top rejtingom.**

4.3 Услуге слања електронске поште

Већ је разјашњено да слање поруке електронском поштом захтева три параметра, односно три команде. Уз то, треба додати прву команду која представља захтев слања поруке и ту се једино одређује команда на основу изреченог текста. Све команде везане за слање електронске поште које следе, су везане у ланац (уланчану листу), па заправо Линдо даље води комуникацију у жељеном смеру. При захтевању слања електронске поште кључну улогу играју речи: **send, mail, email, message, want**, односно на српском: **pošalji, pošaljem, mail, mejl, email, elektronsku, poruku, želim, hoću**.
Један случај комуникације без грешке, на енглеском, може да изгледа овако:

```
Корисник: Lindo, send an email!
(Lindo, compose an email!)
(Lindo, I want to send an email!)
Линдо: I'm ready for a new email. Who is the recipient?
It will be better if you enter an address from the keyboard.

Корисник: Ok, I'll do it (*затим уноси адресу са тастатуре*)
Линдо: What about the subject?

Корисник: Graduation thesis
Линдо: Finally, what is your message?

Корисник: This is my graduation thesis.
Линдо: Email was sent successfully!
```

На српском би то могло да изгледа овако:

```
Корисник: Lindo, želim da pošaljem mail.
(Lindo, sastavi email!)
(Lindo, pošalji poruku!)
Линдо: Spreman sam za novi email. Ko je primalac?
Sigurnije je da unesete adresu sa tastature.

Корисник: U redu. (*затим уноси адресу са тастатуре*)
Линдо: Koja je tema poruke?

Корисник: Diplomski rad
Линдо: Na kraju, izdiktirajte mi poruku.

Корисник: Ovo je diplomski rad.
Линдо: Email je uspešno poslat!
```

4.4 Услуга превода текста

Као и код слања електронске поште, и ова услуга захтева више команди. Прва у низу је команда којом се захтева превод текста, а након ње следи пар команди којима се постављају параметри превода - ове команде чине уланчану листу.

Кључне речи иницијализационе команде превода су: **translate, translation, language, translator, text, some**, односно на српском **prevedi, prevodilac, prevod, prevođenje, prevedeš, tekst, teksta, govor, govora, neki, nekog**.

Комуникација на енглеском би онда могла да изгледа овако:

```
Корисник: Lindo, translate some text.
(Lindo, I need a language translator.)
(Lindo, I need a text translation.)
Линдо: Ok, I can do it. What's the source language?.

Корисник: english (*или било који други језик*)
Линдо: Yes, I can handle that. Say text for translation.

Корисник: *корисник изговара текст за превод, нпр. Mary had
a little lamb.*
Линдо: What's the target language??

Корисник: *корисник изговара језик на који жели превод,
нпр. russian.*
Линдо: I think that translated speech sounds like this:
У Мари был  маленький ягненок
```

На српском, комуникација би могла да изгледа овако:

```
Корисник: Lindo, prevedi mi tekst.
(Lindo, treba mi prevod govora!)
(Lindo, treba mi prevođenje teksta!)
Линдо: Mislim da mogu da Vam pomognem. Koji je izvorni jezik
sa kog želite da prevedem?.

Корисник: srpski (*или било који други језик*)
Линдо: Da, snaći ću se sa tim. Koji tekst želite da prevedem?
Корисник: *корисник изговара текст за превод, нпр.
Ja danas branim diplomski rad.*
Линдо: Na koji jezik želite da prevedem ovo?

Корисник: *корисник изговара језик на који жели превод,
нпр. španski.*
Линдо: Mislim da prevod glasi ovako: Estoy defendiendo mi tesis hoy
```

4.5 Услуга претраге информација са Википедије

Ову услугу позивају све инструкције облика: **Ко је... Ко је био..., Ко су били, Шта је ..., Who is..., What is..., What was ...., What are... What were..., Tell me something about.....** Као пример можемо узети следеће команде:

- Ко је био Доситеј Обрадовић?
- Шта је интегрисано коло?
- Who was Thomas Edison?
- What is Ohm's Law?
- Tell me something about car.

Резултат ових инструкција је текст који почиње фразама: **Према Википедији** - српски или **By Wikipedia** - енглески, а затим следи текст од три реченице са Википедије. Примера ради, ако изговоримо инструкцију **Шта је интегрисано коло?**, добићемо следећи резултат (са српске Википедије, на ћирилици):

```
Према Википедији: Интегрисано коло, интегрисано електрично
коло (Интегрално коло) (скр. енг: IC) је сложено
електрично коло састављено из мноштва елемената (углавном транзистора)
обједињено на јединственој подлози и спремно за уградњу
у сложеније системе али као јединствена компонента.
Уобичајено је да је целокупна мрежа компоненти реализована
на јединственом  комаду полупроводника, који се налази у
средишту интегрисаног кола и који се обично зове чип (енг: Chip).
```

4.6 Услуге везане за интеракцију са претраживачем

Корисник може да претражи нешто на Гугл претраживачу и добије резултат претраге у виду интернет странице. Такође, може да затражи неки профил са друштвених мрежа и добије страницу тог профила.

Кључне речи команде која иницијализује Гугл претрагу су: **open, google, search, page, find, look, browser, term, query, internet, web**, а за српски: **otvori, google, gugl, guglu, guglaj, googlu, izguglaj, izgooglaj, pretraži, pretraga, pretragu, izvrši, strana, traži, pronađi, browser, upit, internet, internetu, web, webu, veb, vebu**.

Интеракција са Линдом, на енглеском би могла бити оваква:

```
Корисник: Lindo, search something on Google.
(Lindo, activate the Internet search.)
(Lindo, look for something on Google for me.)
Линдо: I'll do it with pleasure. What are you looking for?

Корисник: Novak Đoković (*или било који други упит*)
Линдо: Ok, and what about type?
```

```
Possible options are videos, news, images,
books, shopping, applications or all?

Корисник: *корисник изговара било коју инструкцију
која у себи садржи неку од речи:
videos, news, images, books, shopping, applications, all*
Линдо: I'm opening the first result that I found.
(*отвара се страна у претраживачу*)
```

Аналогно претходном, интеракција са Линдом на српском може изгледати овако:

```
Корисник: Lindo, pretraži nešto na Google-u.
(Lindo, izvrši pretragu na Google-u.)
(Lindo, treba mi internet pretraga.)
Линдо: Vrlo rado. Šta želite da potražim?

Корисник: Fakultet inženjerskih nauka u Kragujevcu
(*или било који други упит*)
Линдо: To je u redu. A šta ćemo sa tipom, ponuđene opcije
su: video, vesti,slike, knjige, šoping, aplikacije ili sve?

Корисник: *корисник изговара било коју инструкцију
која у себи садржи неку од речи: video,
vesti, slike, knjige, šoping, aplikacije ili sve*
Линдо: Otvaram prvi rezultat koji je pronađen.
(*отвара се страна у претраживачу*)
```

Још једна функционалност коју пружа сервис претраживача је могућност отварања странице профила са друштвених мрежа. Кључне речи ове команде на енглеском су: **open, social, network, page, facebook, twitter, instagram, linkedin, nickname, profile, account** и на српском: **otvori, prikaži, pokaži, socijalna, socijalnu, društvenu, društvene, mrežu, mreže, stranu, strana, stranica, facebook, fejsbuk, fejs, twitter, tviter, instagram, linkedin, profil, profila, nalog**.
Пример комуникације на енглеском:

```
Корисник: Lindo, open Linkedin page.
(Lindo, open Facebook/Instagram/Twitter account page.)
Линдо: Sure, I'm here to help. Tell me the nickname.
I'm suggesting to type it from the keyboard.

Корисник: Ok, I'm ready.
(*корисник уноси корисничко име профила - нпр. nenad-pantelić*)
Линдо: Here is the page.
(*отвара се страна у претраживачу*)
```

Следи један случај комуникације на српском језику:

```
Корисник: Lindo, otvori Linkedin stranu.
(Lindo, otvori stranu Facebook/Instagram/Twitter naloga.)
Линдо: Naravno, tu sam da pomognem.
Predlažem da unesete korisničko ime sa tastature.

Корисник: U redu, spreman sam za unos.
(*корисник уноси корисничко име профила - нпр. nenad-pantelić*)
Линдо: Evo strane koju sam našao.
(*отвара се страна у претраживачу*)
```

4.7 Услуге засноване на контроли Ардуино плоче

Линдо гласовни асистент има могућност контроле Ардуино плоче. Као демонстративни пример, испробана је контрола укључивања сијалице преко релеја. Кључне речи команде која извршава ову акцију су: **turn, on, off, switch, bulb, arduino, relay, power, light** и **укључи, upali, искључи, ugasi, promeni, stanje, sijalica, sijalice, sijalicu, relej, svetlo, treba, osvetljenje**.
Из овога можемо извести неке примере команди:

- Uključi sijalicu/Isključi sijalicu!
- Treba mi osvetljenje.
- Upali/Ugasi svetlo!
- Promeni stanje sijalice.
- Turn on/off the light
- Switch the state of the bulb

Линдо на ове команде, у случају да је све прошло како треба, одговара са: **Light state is changed.** и **Stanje sijalice je promenjeno.**

5 Закључак

Гласовни асистенти су за пар година доживели велики успон. Индустријска револуција 4.0 је са собом донела општу аутоматизацију, како у индустријском свету, тако и у свету забаве и услуга. Моћни технолошки гиганти, попут Гугла, Амазона и Мајкрософта су направили револуционарни помак у примени гласовних асистената кроз паметне телефоне и звучнике.

Продор концепата кућне аутоматизације повукао је и паметне асистенте у ту причу. Многи алати и библиотеке намењени кућној аутоматизацији пружају подршку у интеграцији гласовних асистената у веће системе контроле (контрола кућних уређаја гласом). Један такав пример је *Amazon Alexa*. Гласовни асистент **Jarvis** из Марвелових франшиза (*Iron Man* и *Avengers*), послужио је као инспирација власнику Фејсбука, Марку Цукербергу (енг. *Mark Zuckerberg*) који је креирао истоименог гласовног асистента за кућну контролу. Осим кућне аутоматизације, гласовни асистенти се спомињу у разним дисциплинама које засигурно, обележавају ову, а и деценије пред нама: аутономна вожња, допуњена стварност (енг. *augmented reality*), виртуелна стварност (енг. *virtual reality*), хуманоидна интелигенција...

Веома је битно нагласити да гласовни асистенти играју јако важну улогу у помоћи особама са инвалидитетом. Многи произвођачи телефона уграђују алате гласовне комуникације, пре свега због олакшаног коришћења слабовидим корисницима и корисницима са неком врстом парализе горњих екстремитета.

Линдо гласовни асистент пружа услуге из домена забаве и пројектован је на нивоу *proof-of-concept*. Циљ његове имплементације је био испитивање могућности примене готових веб АПИ решења у конструисању једне услужне гласовне апликације, као и да се покажу могућности које са собом носи препознавање говора и генерисање говора из текста.

Предности Линдо асистента су једноставност употребе, модуларност која подразумева да, осим постојећих језика, уз врло мало рада је могуће додати нове језике. Уз то, због слојевитости апликације и независности слојева, једноставно је додати и нове могућности, односно сервисе које те могућности пружају. С друге стране, мана је зависност апликације од многих веб АПИ сервиса, пре свега од сервиса за препознавање говора и генерисање говора из текста, па у случају да дође до пада интернет конекције или пробијања лимита, апликација бива делимично или потпуно онеспособљена.

Следећи корак у развоју овог пројекта би био развој апликације за мобилне уређаје, с обзиром да апликација овог карактера итекако налази своје место у свету мобилних телефона и таблета.

Литература

[1] *https://en.wikipedia.org/wiki/Virtual_assistant (приступљено у октобру 2019.године).*

[2] *https://www.codeguru.com/cpp/g-m/multimedia/audio/article.php/c12363/How-Speech-Recognition-Works.htm (приступљено у октобру 2019.године).*

[3] *https://www.explainthatstuff.com/how-speech-synthesis-works.html (приступљено у октобру 2019.године).*

[4] A. . S. R. V. . S. Zhang, *https://github.com/Uberi/speech_recognition (приступљено у октобру 2019.године).*

[5] *https://python-googlesearch.readthedocs.io/en/latest/ (приступљено у октобру 2019.године).*

[6] *https://docs.python.org/2/library/webbrowser.html (приступљено у октобру 2019.године).*

[7] *https://imdbpy.readthedocs.io/en/latest/ (приступљено у октобру 2019.године).*

[8] *https://omdbpy.readthedocs.io/en/latest/ (приступљено у октобру 2019.године).*

[9] *https://py-googletrans.readthedocs.io/en/latest/ (приступљено октобра 2019.године).*

[10] *https://pyowm.readthedocs.io/en/latest/ (приступљено у октобру 2019.године).*

[11] *https://github.com/paulc/gmail-sender (приступљено октобра 2019.године).*

[12] *https://wikipedia.readthedocs.io/en/latest/ (приступљено октобра 2019.године).*

[13] *https://pyserial.readthedocs.io/en/latest/shortintro.html (приступљено октобра 2019.године).*

(1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13)