MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

VOICE CONTROL - UN ASISTENT VIRTUAL

LUCRARE DE LICENŢĂ

Absolvent: **Vlad-Şerban SELEGEAN**
Conducător ştiinţific: **As. ing. Cosmina IVAN**

2018

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Vlad-Şerban SELEGEAN**

VOICE CONTROL - UN ASISTENT VIRTUAL

1. **Enunţul temei:** Sistemul reprezintă un asistent virtual, ce foloseşte recunoaşterea vocală pentru a executa anumite comenzi venite de la utilizator.
2. **Conţinutul lucrării:** Cuprins, Introducere, Obiectivele proiectului, Studiu bibliografic, Analiză şi fundamentare teoretică, Proiectare de detaliu şi implementare, Testare şi validare, Manual de instalare şi utilizare, Concluzii, Bibliografie, Anexe.
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanţi:**
5. **Data emiterii temei:** 1 Noiembrie 2017
6. **Data predării:** 9 Iulie 2018

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnatul(a) _____, legitimat(ă) cu _____ seria _____ nr. _____
CNP _____, autorul lucrării _____
_____
_____
elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la Facultatea de Automatică şi Calculatoare, Specializarea _____ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea _____ a anului universitar _____, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

În cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data _____

Nume, Prenume _____

Semnătura

Cuprins

Capitolul 1

Introducere

Scopul acestui proiect este de a construi un sistem software capabil să înţeleagă anumite comenzi vocale şi să realizeze anumite task-uri în funcţie de aceste comenzi.

1.1 Contextul proiectului

În aceste timpuri tehnologia este peste tot în jurul nostru şi este în continuă dezvoltare. Conform [1], viaţa digitală a oamenilor este determinată de către inovaţii. Mai ales în ultimii ani tot mai multe tehnologii au fost dezvoltate pentru a ne îmbunătăţii viaţa personală şi profesională.

Astfel agenţii personali inteligenţi sunt o realizare importantă. Aceştia au devenit omniprezenţi în procesul de digitizare. Astăzi asistenţii se regăsesc în toate dispozitivele, cum ar fi smartphone-uri, tablete chiar şi în smartwatch-uri. Competiţia ce se află în continuă creştere în această zonă a dus la multe îmbunătăţiri.

Un asistent virtual se bazează pe inteligienţa artificială, pentru a putea recunoaşte cuvintele rostite cu acurateţe. Software-ul foloseşte microfonul dispozitivului pentru a capta comenzile utilizatorului în timp ce acesta oferă un raspuns înapoi prin intermediul difuzoarelor. Între aceste doua acţiuni este prezentă o acţiune ce combină mai multe tehnologii şi anume: recunoaşterea vocală, analiza vocală şi procesarea limabajului. Când un utilizator pune o întrebare sau oferă o comandă, semnalul audio este convertit în informaţie digitală ce poate fi analizată de software.

Companiile mari precum Amazon, Google, Microsoft şi Apple oferă soluţii software ce pot fi controlate folosind un asistent virtual bazat pe voce. Printre aceste soluţii se numară Alexa de la Amazon fiind un dispozitiv separat construit doar în acest scop. Google a dezvoltat un dispozitiv similar cu Alexa, denumit Google Home. Apple a introdus un asistent virtual numit Siri, ce este prezent în toate dizpozitivle lor de dimensiune medie şi mare. În cele din urmă şi Microsoft a dezvoltat propriul lor asistent, numit Cortana, ce este prezent în Windows 10.

Cu toţii ştim că timpul este preţios şi dorim să rezolvam cât mai multe probleme în

cel mai scurt timp şi deci să fim cât mai productivi. Pentru a atinge acest deziderat vin în ajutorul nostru asistenţii personali virtuali. De la simplul fapt că putem interacţiona mai repede sau uneori mai eficient cu device-ul folosit aceştia mai sunt folositori pentru persoanele cu anumite deficienţe de vedere, motorii, etc.

Asistenţii virtuali au aparut de mult timp dar iniţial nu erau atat de eficienţi şi nu puteau fi folosiţi la potenţialul lor maxim. Iniţial, aceştia nu oferau un avantaj considerabil utilizatorilor de rand. În schimb , astazi aceştia pot oferi anumite servicii mult mai rapid decât dacă un utilizator ar executa acceiaşi acţiune manual, i.e setarea unui reminder, cautarea unei aplicaţii, etc.

Conform autorilor din [2] şi [3], asistenţii virtuali reprezintă viitorul, aceştia putând fi folosiţi în companii, pentru a creşte profitul acestora şi pentru a ajuta anumiţi angajaţi în a-şi executa mai bine şi mai repede task-urile pe care trebuie sa le îndeplinească. Aceştia vor fi prezenţi în maşinile inteligente ce sunt şi urmează să fie dezvoltate [4].

1.2 Motivaţia

După cum a fost menţionat in secţiunea anterioară motivul principal pentru dezvoltarea acestei aplicaţii este pentru a putea fi folosită de persoane ce prezintă anumite deficienţe motorii, de vedere, etc.

Un al doilea motiv este acela că utilizatorii actuali doresc sa obţină cea mai mare productivitate si să folosească uşor calculatorul. Astfel se doreşte implementarea unui asistent virtual ce ajută utilizatorul să realizeze anumite task-uri. Acestea pot fi simple de la deschiderea unei aplicaţii până la logarea pe o Facebook sau trimiterea unui email.

Pe langa ideea de creştere a productivităţii, s-a luat în calcul pentru dezvoltare şi faptul că utilizarea calculatorului, pentru unii, nu este simplă şi astfel, s-a dorit o solutie ce va oferi un ajutor în a utiliza calculatorul mai uşor.

1.3 Contribuţii personale

Asistentul virtual oferă assistenţă utilizatorului în executarea anumitor task-uri. Contribuţiile personale asupra implementării asistentului personal virtual sunt următoarele:

- Analiza sistemelor similare: Presupune studierea sistemelor similare şi trecerea în revistă a conceptelor şi practicilor folosite pentru a dezvolta o soluţie cât mai performantă.
- Proiectare: presupune proiectarea soluţiei respectând anumite cerinţe, astfel încat să se obţină un sistem performant, mentenabil şi uşor de folosit.
- Implementare: în implementare s-a urmărit realizarea unei aplicaţii ce poate să fie folosită cu uşurinţa de oricine. În plus s-a dorit folosirea de componente ce sunt open-source şi deci gratuite.

- Testare: presupune verificarea componetelor sistemului.

1.4 Structura lucrării

În continuare se va prezenta structura lucrării pe capitole,acestea fiind însoţite de o scurtă descriere.

- Capitolul 1 – Introducere – Capitol introductiv în care se va descrie contextul problemei.
- Capitolul 2 – Obiectivele Proiectului – Capitol în care sunt descrise obiectivele propuse pentru implementare.
- Capitolul 3 – Studiu Bibliografic – Capitol în care sunt prezentate tehnologiile şi conceptele folosite şi un studiu a sistemelor similare deja existente pe piaţă.
- Capitolul 4 – Analiză şi Fundamentare Teoretică – Capitol în care sunt descrise tehnologiile folosite în dezvoltarea sistemului precum şi motivul alegerii fiecăreia şi cazurile de utilizare ale sistemului.
- Capitolul 5 – Proiectare de Detaliu şi Implementare – Capitol în care este prezentat modul în care sistemul a fost proiectat.
- Capitolul 6 – Testare şi Validare – Capitol în care se vor prezenta testele realizate asupra sistemului precum şi rezultatele acestora.
- Capitolul 7 – Manual de Instalare şi Utilizare – Capitol în care sunt descrişi toţi paşii necesari pentru instalarea cu succes a componentelor sisistemului precum şi manualul de utilizare a aplicaţiei.
- Capitolul 8 – Concluzii – Capitol în care se vor prezenta concluziile legate de implementarea sistemului precum şi prezentarea tuturor realizărilor şi obiectivelor care au fost duse la bun sfârşit în cadrul acestui proiect urmată de o descriere a posibilităţilor de dezvoltare ulterioară.

Capitolul 2

Obiectivele Proiectului

Obiectivul principal al acestui proiect constă în implementarea unui sistem ce permite utilizarea calculatorului prin comenzi vocale. Prin această aplicaţie se urmăreşte eficientizarea folosirii calculatorului, oferirea unei metode de utilizare pentru persoanele cu dizabilităţi şi să ofere o metodă mai eleganta şi user friendly utilizatorilor fară experienţă.

Utilizatorii au la dispoziţie o multitudine de operaţii pe care le pot regăsi în cadrul aplicaţiei.

2.1 Cerinţe Funcţionale

Conform ingineriei software, o cerinţă funcţională defineşte o funcţionalitate a unei componente sau a întregului sistem. Aceasta precizează serviciile pe care sistemul trebuie să le ofere, cum trebuie să reacţioneze sistemul la intrari particulare şi cum trebuie să se comporte sistemul în anumite situaţii particulare [5].

- Deschiderea aplicaţiilor native Windows: Calculator, Notepad, Paint, etc.
- Deschiderea aplicaţiilor des folosite în Windows: Word,Excel, etc.
- Setarea unui reminder folosind Sticky Notes
- Căutarea unui fişier în sistemul de fisiere
- Utilizarea unui mini–calculator vocal pentru operaţii simple
- Navigarea pe internet
  - Accesarea de pagini web: Facebook, Google, UTCN, etc.
  - Capacitatea de a efectua operaţia de Login pe Facebook şi Gmail
  - Capacitatea de a efectua operaţia de Logout pe Facebook şi Gmail
  - Verificarea vremii

  - Utilizare Google Maps
  - Căutare subiecte pe Wikipedia
  - Capacitatea de a scrie si trimite un email folosind Gmail.

- Utilizarea unui transcriber(se ofera un fisier audio .wav si se identifica cuvinte)
- Adăugarea sau ştergerea unor produse într-o listă de cumpărături şi enumerarea celor existente în listă.

2.2 Cerinţe Non-funcţionale

Conform ingineriei software, o cerinţă non-funcţională este o cerinţă prin care putem judeca funcţionarea unui sistem. Aceasta precizează constrângeri asupra serviciilor sau funcţiilor oferite de sistem, cum ar fi constrângeri de timp, constrângeri asupra procesului de dezvoltare, standarde etc. Aceste cerinţe se mai numesc atribute de calitate pentru un sistem.

- Performanţa: Se referă la faptul că o aplicaţie sa fie performantă trebuie să prezinte timpii de răspuns şi procesare mici. Astfel în această aplicaţie se folosesc librării(i.e Sphinx, MaryTTS) ce satisfac aceste cerinţe.
- Eficienţă(Efficiency): Face referire la faptul că un sistem trebuie să ofere un raspuns într-un timp scurt dar folosind un numar mic de resurse. Astfel procesul de recunoaştere vocală este rapid folosind minimul de resurse.
- Mentenabilitate(Maintainability): În inginerie, mentenabilitatea se referă la cât de uşor de menţinut este un sistem. Astfel pentru ca sistemul să fie usor de întreţinut acesta a fost construit respectând anumite Design Pattern-uri(i.e Chain of Responsibility).
- Rezilienţă(Resilience): Este abilitatea unui sistem de a oferi un nivel acceptabil de servicii chiar şi în prezenţa anumitor erori. Astfel sistemul este capabil să ofere răspunsuri oricarei comenzi şi să revină din potenţialele erori.
- Reutilizabilitate(Reusability): Se referă la utilizarea unor elemente existente în sistemul proiectat şi la realizarea proiectului în acest fel încat sa conţină elemente ce pot fi la rândul lor reutilizate. Astfel sistemul poate fi refolosit deorece exista o separare între contexte şi deci componentele pot fi uşor reutilizate. În plus sistemul utilizează elemente deja existente.
- Testabilitatea(Testability): Se referă la uşurinţa cu care poate fi testat un sistem. Astfel sistemul poate fi uşor testat urmărind lista comenzilor posibile si verificând rezultatul obţinut.

Capitolul 3

Studiu Bibliografic

Documentarea bibliografică are ca obiectiv prezentarea stadiului actual al domeniului sau sub-domeniului în care se situează tema.

Într-o primă fază se vor identifica şi prezenta aspecte importante legate de caracteristicile si evolutia asistenţilor virtuali.

În continuare se vor prezenta informaţii legate de recunoaşterea vocală. Se prezintă date legate de istoria tehnologiei, tipuri şi procesul de recunoaştere şi librării/API–uri folosite pentru această tehnologie.

În final sunt prezentate sistemele similare ce folosesc recunoaşterea vocală. Dintre acestea sunt detaliate Siri, Alexa, Google Home şi Cortana.

3.1 Asistenţi Personali Virtuali

Conform [6], un Asistent Personal Virtual este o aplicaţie ce înţelege comenzile vocale şi executa anumite sarcini pentru utilizator. Printre aceste sarcini se numară: scrierea unui text prin dictare, citirea unui text cu voce tare, citirea şi scrierea de email-uri,etc. Aceste sarcini erau, deobicei, realizate de catre un asistent sau o secretară.

3.1.1 Tipuri de asistenţi virtuali

Avem doua tipuri de asistenţi virtuali:

- Smart advisers (Consultant inteligent) subject-oriented (orientat pe subiect). Autorii [7] definesc un smart adviser ca reprezintând conceptul de utilizare a automatizării şi a tehnicilor digitale pentru îmbunatăţirea semnificativă a relaţiei consultant-client.

- Virtual assistants (Asistent virtual) task-oriented (orientat pe sarcini). Un asistent virtual este un agent software ce este folosit pentru a executa task-uri sau servicii pentru un individ.

3.1.2 Forme de Interacţiune

Analog [8] asistenţii virtuali prezintă mai multe forme de interacţiune cu utilizatorul. Acestea sunt:

- Text(chat online) prin intermediul unei aplicaţii
- Voce folosind tehnologii pentru recunoaşterea vocală
- Imagini prin upload-are de imagini

3.1.3 Servicii

Un asistent virtual poate oferi o gamă largă de servicii, cum ar fi:

- Oferă informaţii legate de vreme, fapte, etc.
- Operaţii precum setarea unei alarme, to-do list, etc.
- Redarea de muzică prin aplicaţii precum Spotify, Pandora sau Radio
- Redarea de videoclipuri, seriale TV, filme, streaming prin Netflix
- Comerţ conversaţional. Conform [9], comerţul conversaţional se referă la intersecţia aplicaţiilor de mesagerie şi a cumpărăturilor. Este trendul de a interacţiona cu afacerile prin intermediul aplicaţiilor de mesagerie precum WhatsApp, Wechat sau prin tehnologia vocală cum ar fi produsul Echo al companiei Amazon.
- Oferă suport pentru servicii de tip relaţii cu clienţii

3.1.4 Evoluţia asistenţilor virtuali

Evoluţia asistenţilor virtuali începe din 1961 când IBM introduce IBM Shoebox, fiind primul tool ce foloseşte recunoaşterea vocală. Acesta era folosit pentru a recunoaşte 16 cuvinte şi numere.

În 1972 Carnegie Mellon termină Harpy Program. Acesta este capabil să înţeleagă în jur de 1000 de cuvinte.

În 1990 este lansat produsul Dragon Dictate de către firma Dragon. Acesta este primul produs ce foloseşte recunoaşterea vocală, fiind disponibil în comerţ.

Până in 2011 tehnologia mai evoluează fiind introdus Clippy în Microsoft şi posibilitatea de a utiliza recunoaşterea vocală în Microsoft Office XP. Dar un pas mare îl reprezinta apariţia asistentului virtual Siri de către compania Apple.

În continuare firmele mari precum Google,Microsoft şi Amazon lansează produse precum Google Now, Cortana BETA, Alexa.

În prezent se incearcă introducerea asistenţilor în mai multe dispozitive precum boxe,căsti,etc. În plus se realizează demersuri în scopul creşterii numarului de cuvinte ce pot fi înţelese şi a limbilor folosite.

Evoluţia asistenţilor virtuali,mai în detaliu, este ilustrată în figura 3.1, conform datelor din [10].

3.1.5 Prezenţa Asistenţilor virtuali

Asistenţii virtuali personali pot fi gasiţi într-o grămadă de dispozitive, printre care se numară:

- Boxele inteligente, ex: Amazon Echo, Google Home
- Aplicaţiile de mesagerie instantă, ex: Facebook Messenger
- Integraţi în sistemele de operare, ex: Apple-Siri, Cortana-Microsoft
- Aplicaţiile mobile, ex: Google Allo, Dom-Pizza Domino
- Ceasurile inteligente

În viitor asistenţii personali vor fi mai des folosiţi atât în viaţa personală cât şi în activitatea comercială deoarece aceştia evoluează reuşind să ofere mai multe servicii utilizatorilor. Evoluţia acestora depinde de evoluţia mecanismelor folosite pentru recunoaşterea vocală şi procesarea limbajului natural.

3.2 Recunoasterea Vocală

Conform [11] recunoaşterea vocală este o tehnologie ce îi permite unui dispozitiv să inregistreze cuvintele rostite de o persoană prin intermediul unui microfon. Aceste cuvinte sunt apoi, recunoscute de către un identificator de vorbire iar in final se pot executa anumite comenzi.

Autorii [12] descriu o situaţie ideală pentru procesul de recunoaştere vocală ca fiind aceea când mecanismul idetifică toate cuvintele rostite. Performanţa unui astfel de sistem depinde de un număr mare de factori cum ar fi: vocabularul folosit, limba folosită, zgomotul de fundal, etc.

3.2.1 Istoric

Analog [13] conceptul de recunoaştere vocală a inceput in jurul anilor 1940, astfel primul program de recunoaştere a vorbirii a aparut in 1952 la Bell Labs şi a fost folosit pentru recunoaşterea unei cifre intr-un mediu fară zgomot.

Conform [14] avem următoarele perioade importante:

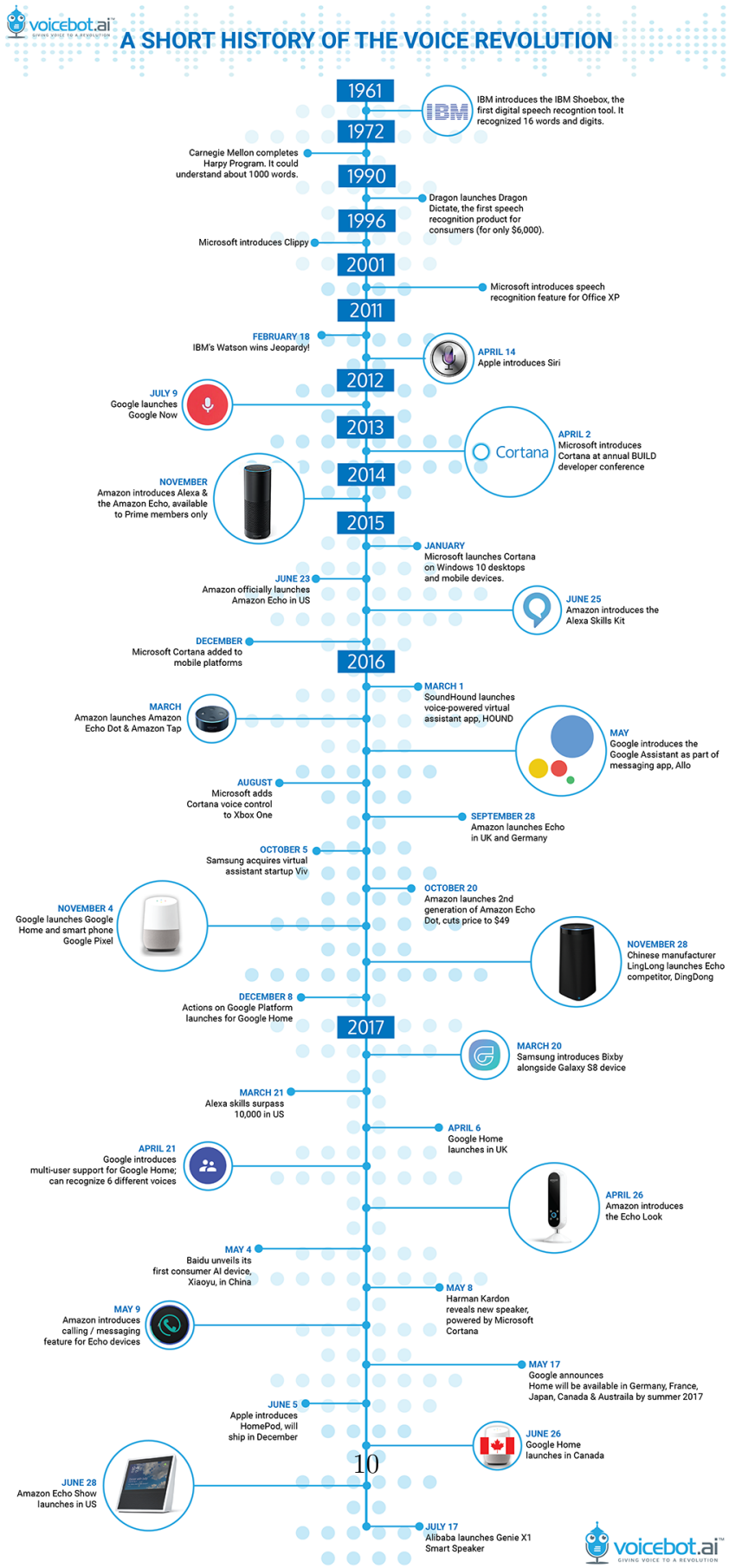

Figura 3.1: Evolutia Asistentilor Virtuali

www.voicebot.ai

- Anii 1940 şi 1950 sunt consideraţi ca fiind anii în care s-au pus bazele tehnicii pentru recunoaşterea vocală. In această perioadă s-au realizat lucrări cu privire la paradigmele recunoaşterii vocale.
- În anii 1960 s-a reuşit recunoaşterea unui numar redus de cuvinte, intre 10 si 100, pe baza proprietaţilor acustice a sunetelor.
- În anii 1970 s-a reuşit recunoaşterea unui numar mai mare de cuvinte, intre 100-1000, folosind tehnici de recunoaştere a modelelor.
- În anii 1980 s-au folosit vocabulare de dimensiuni mari, 1000+ cuvinte. Cea mai importanta inventţie din acestă perioadă a fost modelul HMM (Hidden Markov Model) şi modelul limbajului stocastic. Acestea impreuna au făcut posibilă descoperirea de noi metode mai puternice de a gestiona recunoaşterea continuă a vorbirii intr-un mod eficient si cu performanţe ridicate.
- În 1990-2000 s-au dezvoltat metode de înţelegere a limbajului stocastic, învăţarea statistică a modelelor acustice şi a modelelor de limbaj.
- In 2011 Apple introduce Siri, noul asistent personal virtual prezent in dispozitivele Apple.
- In 2012 Google lansează Google Now, un asistent ce afiseaza anumite informaţii sub forma de cărţi(cards) în funcţie de anumite acţiuni repetate şi date din jurul utilizatorului.
- In 2013 Microsoft introduce Cortana Beta.
- In 2014 Amazon lansează Alexa&Amazon Echo pentru membrii premium.
- In 2015 Cortana este acum prezentă pe toate calculatoarele ce folosesc Windows 10.
- In 2016 Google lansează Google Home, un device asemanător cu Alexa.
- Din 2017 până în prezent are loc creşterea performanţei acestor dispozitive şi mărirea zonelor de distribuţie a acestora.

3.2.2 Tipuri de Recunoaştere Vocală

Sistemele ce folosesc recunoaşterea vocală pot fi categorisite în funcţie de abilitatea lor în a recunoaşte cuvinte şi liste de cuvinte.Conform [11] avem urmatoarele categorii:

- Isolated Speech(Vorbire Izolată): Cuvintele izolate prezintă obligatoriu o pauză între doua cuvinte.
- Connected Speech(Vorbire Conectată): Cuvintele conectate sunt similare cu cele izolate dar acceptă o pauză minimala între ele.
- Continuous Speech(Vorbire Continuă): Îi permite utilizatorului să vorbească aproape natural. Acest tip mai este denumit computer dictation(dictarea către calculator).
- Spontaneous Speech(Vorbire Spontană): Se referă la recunoaşterea unui discurs natural, ce nu a fost repetat înainte, adică poate prezenta onomatopee precum ”umm”, ”aaa”.

3.2.3 Procesul de Recunoaştere Vocală

Proocesul de recunoaştere vocală este format din 6 componente, acestea sunt urmatoarele:

- Audio Input(Intrare Audio): Sunetul este captat de către sistem prin intermediul unui microfon iar placa de sunet a calculatorului produce reprezentarea digitală a sunetului.
- Digitization(Digitizarea): Este procesul prin care un semnal analog este convertit într-un semnal digital. Prezintă atat un proces de eşantionare cât şi unul de cuantificare.
- Acoustic Model(Modelul Acustic): Un model acustic este creat prin înregistrarea audio a discursului şi prin folosirea unui software pentru a crea reprezentări statistice ale sunetelor ce alcătuiesc fiecare cuvânt. Acesta este folosit de un motor de recunoaştere a vorbirii. Modelul împarte cuvintele in foneme.
- Language Model(Modelul Lingvistic): Modelarea limbajului este folosită în multe aplicaţii de prelucrare a limbajului natural, cum ar fi recunoaşterea vocală. Modelul este folosit pentru a compara fenomenele cu cuvinte din propriul dicţionar.
- Speech Engine(Motor de Vorbire): Funcţia motorului este de a converti intrarea audio în text. Pentru a realiza acest deziderat se utilizează mai multe tipuri de date, algoritmi şi statistici.

Componentele procesului de recunoaştere vocală sunt ilustrate in figura 3.2

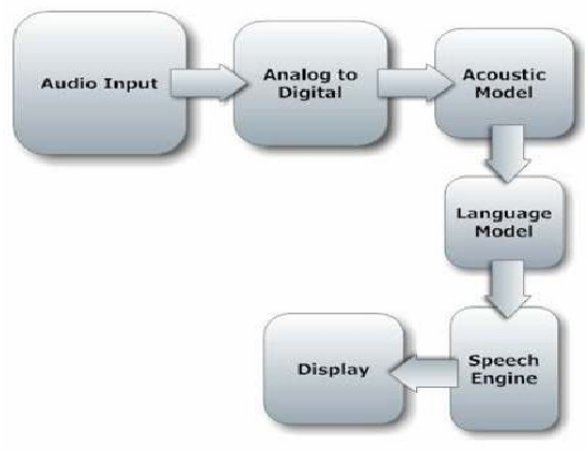

Figura 3.2: Componentele procesului de recunoaştere vocală
www.slideshare.net/sarangafle/speech-recognition-project-report

3.2.4 APIs pentru Recunoaştere Vocală

Conform [15] putem avea două categorii:

Open Source

- CMU Sphinx: Foloseste HMM(Hidden Markov Model). Este cross-platform iar deoarece nu foloseşte multe resurse poate fi folosit pe mobil.
- Kaldi: Este un toolkit scris in C++. Foloseşte reţele neuronale şi există mai multe variante atât în cloud cât şi pentru utilizare offline.

Contra cost

- Google Speech API: Prezinta cea mai performantă tehnologie pentru recunoaşterea vocală, folosind reţele neuronale. Recunoaşte până la 120 de limbi. Poate fi folosit free pentru recunoaşterea vocală pana la 60 de minute.
- Microsoft Cognitive Services - Bing Speech API: Prezintă o performanţă ridicată şi multe addon-uri ce pot fi folosite, de ex: autentificare vocala.
- Speechmatics: Prezintă un vocabular mare, situat în cloud. Poate identifica doar limba engleză cu o acurateţe mare.

Având în vedere faptul că dorim implementarea unui sistem bazat pe componente open-source, am ales CMU Sphinx. În plus această librarie permite uşor integrarea in Java.

Această librarie prezintă un numar relativ mare de exemple de cod şi ne ofera un model acustic şi un dictionar al limbii engleze. Libraria poate fi folosita atât în aplicaţii Desktop cât şi pe Mobil.

Sistemul de recunoaştere este bazat Modele Markov Ascunse (Hidden Markov Models-HMM). Modelarea cuvintelor se efectuează pe baza unităţilor de sub-cuvinte. Recent au fost introduse filtre pentru excluderea zgomotului înconjurator.

Pe lângă recunoasterea vocală, ce ne permite să oferim comenzi, această librarie mai este capabila sa identifice persoanele care vorbesc şi transcrierea înregistrarilor video şi audio.

3.3 Sisteme Similare

3.3.1 Siri

Este asistentul inteligent prezent acum în produsele companiei Apple. A apărut în 4 Octombrie, 2011 odată cu lansarea sistemului de operare, IOS5 pentru Iphone4S.

Conform [16] Siri a fost un proiect iniţiat de SRI-ul Internaţional şi finanţat de DARPA, ea fiind clasată pe locul unu în topul agenţilor personali inteligenţi folosiţi în 2017.

Aceasta a fost construită pentru a oferi o metoda unitară de a interacţiona cu device-ul prin anumite comenzi vocale, oferind înapoi anumite răspunsuri, atât vizuale cât şi vocale.

Siri prezintă urmatoarele tehnologii:

- Recunoaşterea vocală automată (Automatic Speech Recognition-ASR-) pentru a transcrie cuvintele rostite în text.
- Procesare de limbaj natural (Natural Language Processing-NLP-)
- Analizator de text pentru a afla dacă textul este o întrebare sau o comandă
- Tehnologii de mashup pentru a comunica cu anumite servicii web
- Deep Learning, metoda de machine learning bazată pe modele de învătare reprezentative
- Deep Neural Network (DNN) pentru a converti modelul acustic al vocii, în fiecare moment, intr-o distribuţie de probabilitate peste suntele de vorbire

În figura 3.3 putem observa arhitectura agentului.

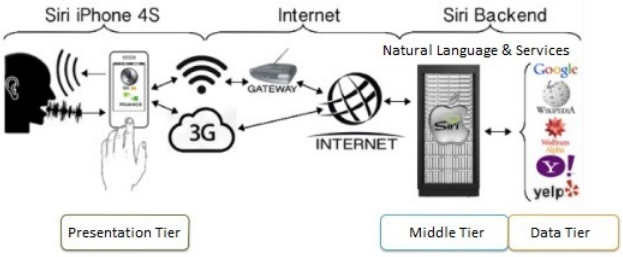

Figura 3.3: Arhitectura Siri
https://www.slideshare.net/mulesoft/application-architecture-the-next-wave

3.3.2 Alexa

Alexa, este un agent personal inteligent creat de către Amazon ce prezintă o multitudine de funcţii. Spre deosebire de Siri sau Cortana, Alexa este un dispozitiv separat. Agentul poate fi activat folosind un cuvânt de deşteptare, i.e Echo.

Produsul a apărut în Noiembrie, 2014 impreună cu Echo. Alexa a fost inspirată din sistemul de conversaţie şi calculator de bord din serialul Starship Enterpise. Acesta prezintă un serviciu numit Alexa Voice Services(AVS), ce procesează comenzile primite. In [17] autorii prezintă faptul că, Cortana foloseste un sistem de procesare a limbajului natural(NLP) şi deep neural networks (DNN).

Alexa prezintă doua tipuri de functionalităţi: comenzi şi third-party skills(competenţe). Comenzile şi skill-urile sunt dezvoltate de către developerii de la Amazon dar skill-urile pot fi implementate şi de useri. Pe lângă acestea Alexa prezintă o problemă majoră şi anume faptul că necesită o conexiune la internet pentru a funcţiona.

În figura 3.4 putem observa arhitectura agentului.

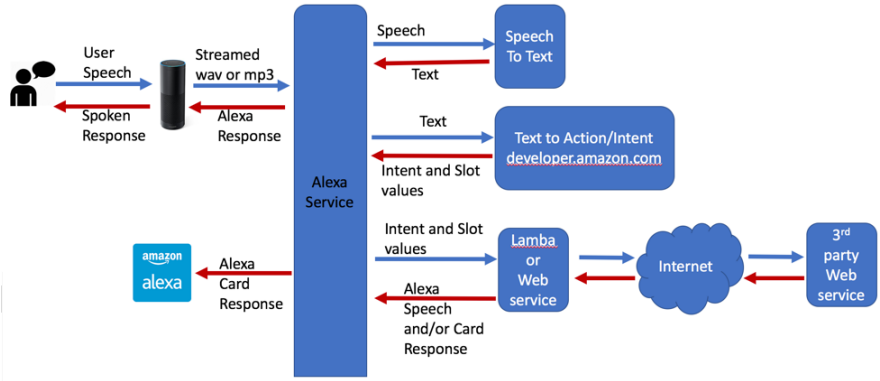

Figura 3.4: Fluxul execuţiei unei comenzi pentru Alexa
www.medium.com/@abraham.kang/understanding-the-differences-between-alexa-api-ai-wit-ai-and-luis-cortana-2404ece0977c

3.3.3 Cortana

Cortana, este un agent personal inteligent realizat de către Microsoft şi a fost introdus in Windows 10 şi telefoanele cu sistem de operare Windows. Numele provine din jocul Halo, unde Cortana este un caracter inteligent sintetic.

A fost realizată pentru a rezolva task-uri de la trimiterea de email-uri până la rostirea glumelor. În prezent poate vorbi şi înţelege opt limbi. Aceasta foloseşte un API pentru a traduce cuvintele, acesta fiind localizat pe cloud.

Prezintă o multitudine de funcţionalităţi ce duc la cresterea productivităţii utilizatorului. Printre aceastea se numară: setarea de remindere, GPS, vreme, conversii, căutare loacală, căutare pe internet, etc.

Conform [18] capacităţile de procesare a limbajului natural au derivat din Tellme Networks (ce a fost cumparată de Microsoft 2007) şi au fost cuplate cu o bază semantică de date numită Satori.

Cortana foloseşte în plus faţă de alţi agenţi, anumiţi algoritmi de prelucrare precum Dynamic Time Warping (deformarea dinamică a timpului) pentru a recunoaşte şi diferenţia mai bine cuvintele rostite.

În figura 3.5 se poate observa "inteligenţa" agentului şi anume multitudinea de servicii pe care le foloseşte.

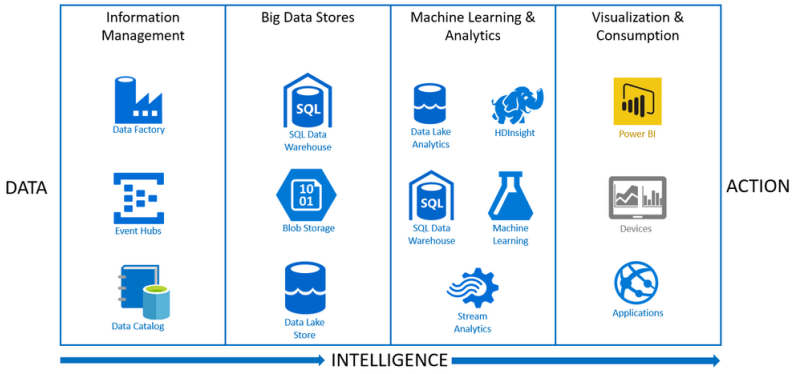

Figura 3.5: Cortana Inteligence
http://www.datamic.net/blog/cortana-intelligence-basics-storage-and-compute

3.3.4 Google Home

Conform [1] Google Home este un brand de difuzoare inteligente(smart speakers) dezvoltate de către Google. Primul device a fost anunţat in Mai 2016 şi lansat in US în Noiembrie 2016.

Acesta primeşte de la utilizatori comenzi pentru a interacţiona cu anumite servicii folosind asistentul personal denumit Google Assitant. Dispozitivul prezintă un număr mare

[1] https://en.wikipedia.org/wiki/Google_Home

de servicii atât built-in cât şi third-party. Prezintă o gamă largă de funcţionalităţi dar cea mai importantă este utilizarea ca un controller pentru sistemul Chromecast.

Un avantaj pe care îl prezinta Google Home este faptul ca Google Search-ul şi voice control-ul sunt foarte performante. Prezintă abilitatea de a memora mai multe conturi Google şi este capabil sa diferenţieze vocile utilizatorilor diferiţi cât şi datele specifice fiecăruia. Din noiembrie 2017 este capabil să proceseze două comenzi deodată.

În figura 3.6 putem observa flow-ul execuţiei unei comenzi pentru Google Home.

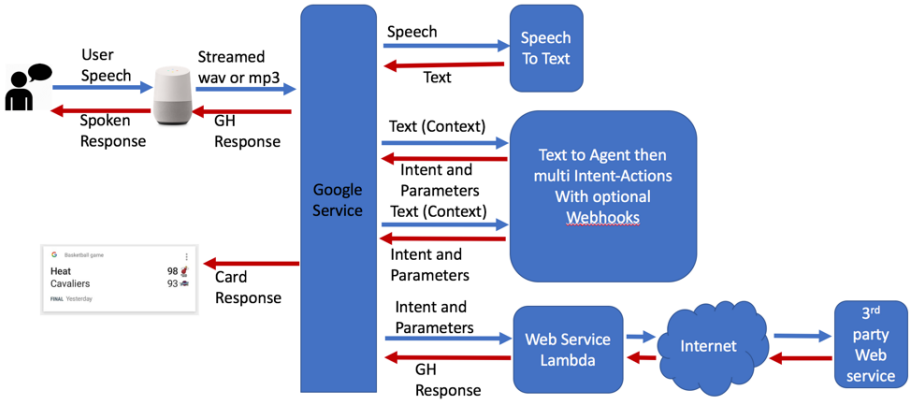

Figura 3.6: Fluxul execuţiei unei comenzi pentru Google Home
https://medium.com/@abraham.kang/understanding-the-differences-between-alexa-api-ai-wit-ai-and-luis-cortana-2404ece0977c

3.3.5 Analiză Comparativă

În continuare se va face o comparaţie între sistemele menţionate anterior cu sistemul ce va fi implementat. Comparaţia se va face în funcţie de mai multe caracteristici care se doresc sa fie implementate.

Tabelul 3.1: Analiză comparativă a sistemelor similare caracteristici generale

Asistent Personal	Developer	Free software	IoT	Smart Phone App	Always On
Siri	Apple	Nu	Da	Da	Da
Alexa	Amazon	Nu	Da	Da	Da
Cortana	Microsoft	Nu	Da	Da	Da
Google Home	Google	Nu	Da	Da	Da
Proiectul propus	-	Da	Nu	Nu	Da

Toate sistemele prezentate anterior nu sunt free deoarece acestea sunt prezente doar în produse ce trebuiesc cumpărate. Sistemul implementat va fi free şi deci va putea fi instalat gratuit pe orice calculator. Din punctul de vedere al IoT sistemul implementat

nu va interschimba date cu alte dispozitive faţă de sistemele similare. Sistemele similare sunt Always On adică mereu ascultă pentru a detecta daca un utilizator a folosit cuvântul de ”activare”(de exemplu: Hey Siri). Sistemul implementat este Always On prin faptul că odata rulat programul asistentul ascultă mereu şi aşteaptă sa primească comenzi(desigur recunoaşterea poate fi oprită din interfaţă).

Tabelul 3.2: Analiză comparativă a sistemelor similare caracteristici funcţionale

Cerinţe funcţionale	Alexa	Cortana	Siri	Google Home	Proiectul Propus
Deschidere aplicaţii	✓	✓	✓	✓	✓
Web search	✓	✓	✓	✓	✓
Verificarea vremii	✓	✓	✓	✓	✓
Redare muzică	✓	✓	✓	✓	✓
Setare de alarme	✓	✓	✓	✓	✓
Setare de reimendere	✓	✓	✓	✓	✓
Calendar	✓	✓	✓	✓	✗
Realizearea unei liste de shopping	✓	✓	✓	✓	✓
Realizarea de comenzi online	✓	✓	✓	✓	✗
Redare cărţi audio	✓	✓	✓	✓	✗
Trimitere email	✓	✓	✓	✗	✓
Realizarea operaţiei de login pe anumite site-uri	✗	✗	✗	✗	✓
Realizarea operaţiei de logout pe anumite site-uri	✗	✗	✗	✗	✓

În urma comparaţiei se observă faptul că implementarea proprie oferă în plus anumite funcţionalitaţi faţă de sistemele similare.

Capitolul 4

Analiză şi Fundamentare Teoretică

În acest capitol se vor prezenta cazurile de utilizare a sistemului. În plus vor fi detaliate tehnologiile alese în dezvoltarea aplicaţiei.

4.1 Arhitectura Conceptuală

O privire de ansamblu a arhitecturii conceptuale se poate deduce din Figura 4.1 unde sunt reprezentate componentele proiectului. Se observă faptul că proiectul contine 4 componente principale.

- **Utilizatorul**. Acesta rosteşte o comandă şi aşteaptă înapoi un raspuns. Comanda rostită este captată de către microfonul calculatorului şi trimisă mai departe sistemului.
- **Microfonul**. Rolul acestuia este să capteaze sunetele rostite de către utilizator. Aceste sunete sunt apoi transmise sistemului unde sunt recunoscute de către algoritmul folosit de librăria Sphinx 4.
- **Recunoaşterea Vocală**. În acest pas are loc identificarea cuvintelor rostite şi transmiterea acestora mai departe pentru a putea fi interpretate.
- **Interpretarea Raspunsului**. Dupa primirea cuvintelor identificate, acestea sunt interpretate de către sistem pentru a pune în execuţie acţiunile cerute. Această componentă oferă înapoi un raspuns vocal folosind libraria MaryTTS şi execută comanda oferind înapoi raspunsul cerut.

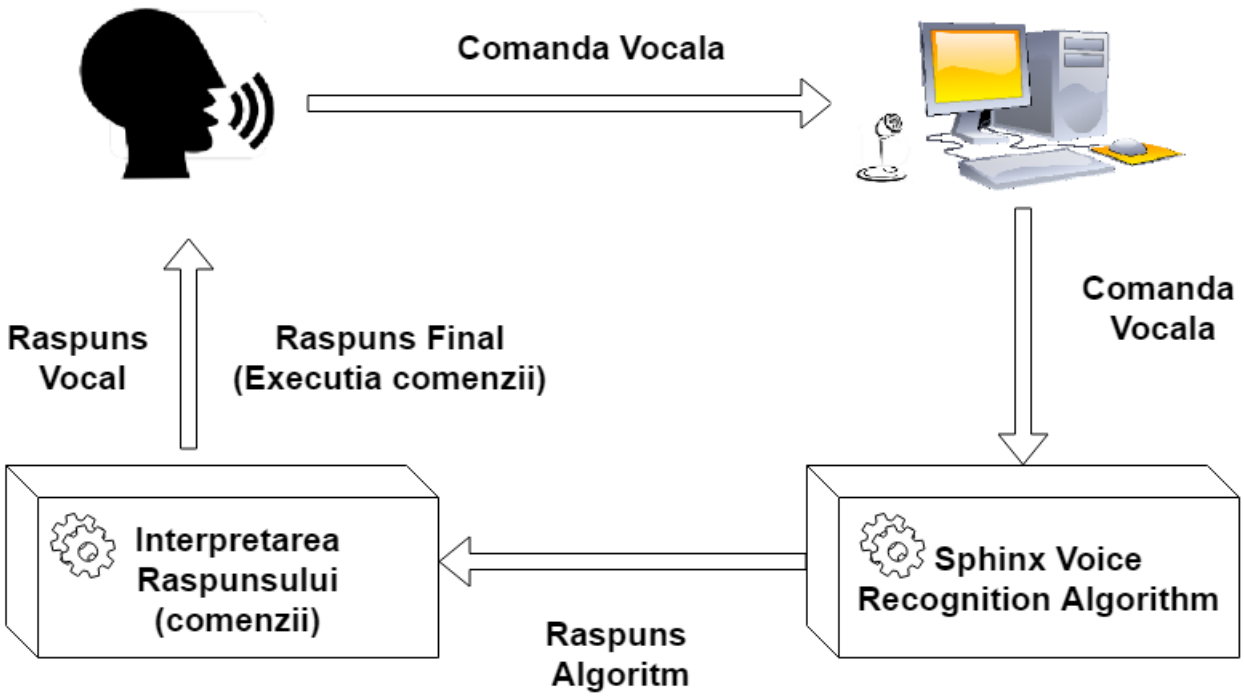

Figura 4.1: Arhitectura Conceptuală a Sistemului

4.2 Cazuri de utilizare

În acestă secţiune sunt descrise principalele cazuri de utilizare a sistemului. Având în vedere domeniul proiectului, avem un singur actor şi acesta este utilizatorul. În figura 4.2 putem observa diagrama de utilizare generală.

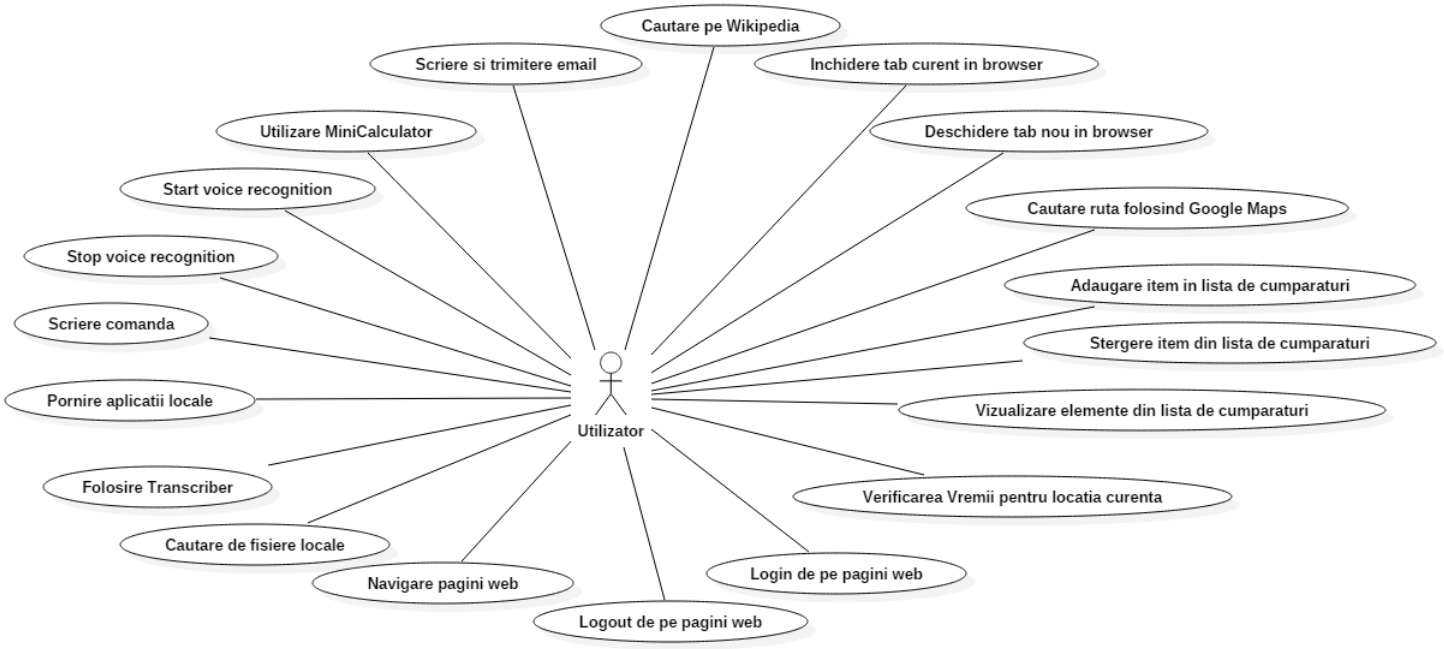

Figura 4.2: Diagramă Use Case

În continuare sunt descrise cazurile de utilizare a sistemului, specificând pentru fiecare caz, paşii necesari, actorii, precodiţiile şi postcondiţile daca există precum şi scenariile de succes sau eroare.

4.2.1 Start voice recognition

Actor principal: Utilizatorul

Precondiţii

1. Aplicaţia trebuie să fie pornită
2. Click pe butonul de start

Postcondiţii

Scenariu succes

1. Recunoaşterea vocală este pornită

2. Se aşteaptă introducerea unei comenzi

Scenariu eroare

4.2.2 Stop voice recognition

Actor principal: Utilizatorul
Precondiţii

1. Recunoaşterea vocală trebuie să fie pornită

2. Click pe butonul de stop

Postcondiţii
Scenariu succes

1. Recunoaşterea vocală se opreşte

Scenariu eroare

4.2.3 Scriere comandă

Actor principal: Utilizatorul
Precondiţii

1. Aplicaţia trebuie să fie pornită

2. Introducere comandă în text field

Postcondiţii
Scenariu succes

1. Se oferă un raspuns ce ne spune că comanda se execută

2. Comanda este executată

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.4 Deschiderea aplicaţiilor native Windows

Exemple de aplicaţii, native, ce pot fi deschise: Calculator, Notepad, Paint, etc.
Actor principal: Utilizatorul
Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie să fie pornită

Postcondiţii
Scenariu succes

1. Se oferă un raspuns ce ne spune că aplicaţia se deschide
2. Aplicaţia dorită se deschide

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.5 Deschiderea aplicaţiilor des folosite in Windows

Exemple de aplicaţii: Word,Excel, etc.
Actor principal: Utilizatorul
Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie să fie pornită
3. Aplicaţia dorită trebuie să fie instalată şi accesibilă
4. Trebuie setată calea către executabilul aplicaţiei

Postcondiţii
Scenariu succes

1. Se oferă un raspuns ce ne spune că aplicatia se deschide
2. Aplicaţia dorită se deschide

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.6 Setarea unui reminder folosind Sticky Notes

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie să fie pornită

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că aplicatia Sticky Notes se deschide
2. Se aşteaptă introducerea datelor

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.7 Cautarea unui fişier in sistemul de fisiere

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită
3. Existenţa unei partiţii cu numele E, C, etc.

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că fişierul se caută
2. Utilizatorul este anunţat dacă fişierul a fost găsit sau nu

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii
2. Dacă fişierul nu este găsit atunci se afisează un mesaj

4.2.8 Utilizarea unui mini–calculator vocal pentru operaţii simple

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie să fie pornită

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se calculează rezultatul
2. Se afişează şi comunică rezultatul operaţiei

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii
2. Dacă se folosesc numerere mai mari de 1-9 atunci rezultatul nu va fi valid

4.2.9 Accesarea de pagini web

Exemple de pagini ce pot fi accesate: Facebook, Google, UTCN, etc.

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie să fie pornită
3. Existenţa browser-ului Google Chrome

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se accesează pagina
2. Se accesează pagina

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.10 Închidere tab curent în browser

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită
3. Existenţa browser-ului Google Chrome
4. Browser-ul trebuie sa fie deschis

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se închide tab-ul curent
2. Se închide tab-ul curent

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.11 Deschidere tab nou în browser

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită
3. Existenţa browser-ului Google Chrome
4. Browser-ul trebuie sa fie deschis

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se deschide tab-ul nou
2. Se deschide şi se selectează tab-ul nou

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.12 Capacitatea de a efectua operaţia de Login

Operaţia poate fi efectuată pentru paginile: Facebook şi Gmail
Actor principal: Utilizatorul
Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită
3. Existenţa browser-ului Google Chrome
4. Pagina deschisă să fie Facebook, Gmail
5. Să nu fim deja logaţi
6. Sunt necesare datele de logare

Postcondiţii
Scenariu succes

1. Se oferă un raspuns ce ne spune că se execută operaţia de Login
2. Se execută operaţia de Login

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii
2. Dacă datele sunt greşite operaţia va eşua şi se cere repetarea comenzii

4.2.13 Capacitatea de a efectua operaţia de Logout

pe Facebook şi Gmail
Actor principal: Utilizatorul
Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită
3. Existenţa browser-ului Google Chrome
4. Pagina deschisă să fie Facebook, Gmail
5. Să fim logaţi pe aceasta pagină

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se execută operaţia de Logout
2. Se execută operaţia de Logout

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.14 Verificarea vremii

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită
3. Existenţa browser-ului Google Chrome

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se execută operaţia
2. Se deschide browser-ul Google Chrome
3. Se afişează vremea din locaţia curentă

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.15 Căutare rută folosind Google Maps

Momentan se pot folosi ca puncte de start/stop doar oraşe sau ţari.

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită
3. Existenţa browser-ului Google Chrome
4. Este necesară sursa şi destinaţia rutei

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se execută operaţia
2. Se deschide brower-ul Google Chrome
3. Se accesează pagina Google Maps
4. Se caută ruta cerută

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.16 Capacitatea de a scrie si trimite un email folosind Gmail

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită
3. Existenţa browser-ului Google Chrome
4. Existenţa unei adrese de mail de tip Gmail
5. Pagina curentă sa fie Gmail
6. Sunt necesare datele precum: adresă destinatar, subiect email şi mesaj

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se execută operaţia
2. Are loc compunerea email-ului

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.17 Utilizarea unui transcriber

Se ofera un fisier audio .wav şi se identifica cuvinte sau propoziţii.

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unei înregistrări audio de tip .wav

Postcondiţii

Scenariu succes

1. Sunt afişate pe rând 3 cele mai bune ipoteze din ce sa auzit

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.18 Adăugare item în lista de cumpărături

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită
3. Item-ul să existe în dicţionar

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se execută operaţia
2. Item-ul este adăugat la lista de cumpărături

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.19 Ştergere item din lista de cumpărături

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional
2. Recunoaşterea vocală trebuie sa fie pornită

3. Item-ul să existe în dicţionar

4. Existenţa a cel puţin unui item în lista de cumpărături

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se execută operaţia

2. Item-ul este şters din lista de cumpărături

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.20 Vizualizare iteme din lista de cumpărături

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional

2. Recunoaşterea vocală trebuie sa fie pornită

3. Existenţa a cel puţin unui item în lista de cumpărături

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se execută operaţia

2. Lista este afişată şi comunicată vocal utilizatorului

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

4.2.21 Căutare pe Wikipedia

Căutarea se face prin încercări, se oferă un subiect şi se încearcă accesarea acelui subiect prin adăugarea subiectului la adresa site-ului. Din aceste cauze, funcţionează pentru cuvinte fară spaţii.

Actor principal: Utilizatorul

Precondiţii

1. Existenţa unui microfon funcţional

2. Recunoaşterea vocală trebuie sa fie pornită

Postcondiţii

Scenariu succes

1. Se oferă un raspuns ce ne spune că se execută operaţia

2. Se încearcă accesearea paginii

Scenariu eroare

1. Dacă comanda nu este înţeleasă atunci se va cere repetarea comenzii

2. E posibil ca pagina sa nu fie accesată corect şi sa afişeze 404. Astfel se va cere repetarea comenzii

4.3 Diagrame de Flux

Scopul diagramelor de flux este acela de a ilustra dinamica anumitor funcţionalităţi a sistemului. În continuare vor fi prezentate diagramele de flux pentru unele funcţionalităţile mai simple şi mai complexe.

4.3.1 Diagrama de Flux pentru Login

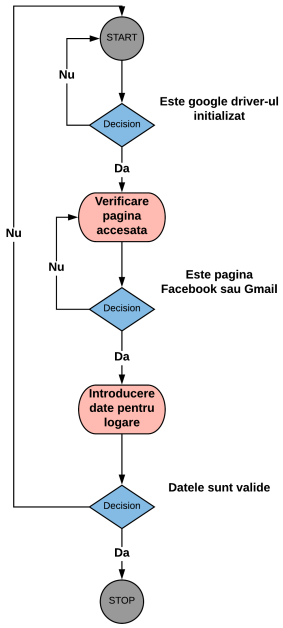

Figura 4.3: Diagramă de Flux pentru Login

4.3.2 Diagrama de Flux pentru Trimitere Email

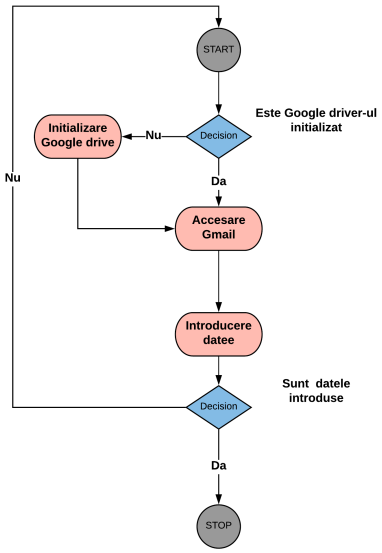

Figura 4.4: Diagramă de Flux pentru Trimitere Email

4.3.3 Diagrama de Flux pentru Deschidere Tab Nou

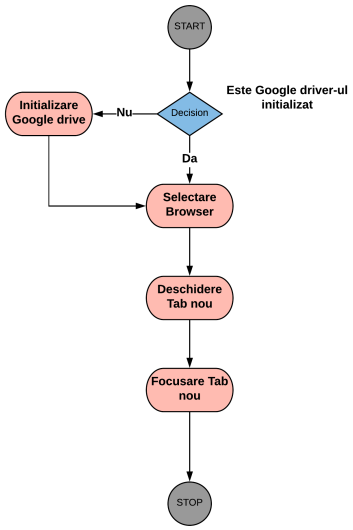

Figura 4.5: Diagramă de Flux pentru eschidere Tab Nou

4.4 Perspectiva tehnologică

4.4.1 CMU Sphinx

Conform [19], Sphinx este un framework modular, open-source folosit pentru recunoaşterea vocală, dezvoltat la Carnegie Mellon University. Acesta este modular deoarece este format din componente ce sunt dedicate unor task-uri specifice. Sistemul este realizat in Java, fiind foarte portabil, flexibil şi usor de folosit în sistemele bazate pe multithreading.

Sphinx-4 este proiectat diferit faţă de versiunile anterioare în termeni legaţi de modularitate, flexibilitatate şi din punct de vedere al algoritmilor folosiţi pentru recunoaşterea vocală. Acesta foloseşte strategii de căutare(search strategies) mai noi şi mai evoluate, prezintă un numar variat de gramatici şi modele de limbă, modele acustice şi modele pentru streaming(feature streams). Inovaţiile algoritmice incluse în sistem permit încorporarea mai multor surse de informare într-un mod elegant.

Arhitectura Sistemului

În figura 4.6 este ilustrată arhitectura sistemului. Fiecare element reprezinta un modul ce poate fi uşor înlocuit oferind astfel multiple posibilitaţi de experimentare fără a fi nevoie de modificari în tot sistemul. Sphinx-4 este format din 3 module principale şi anume [20]:

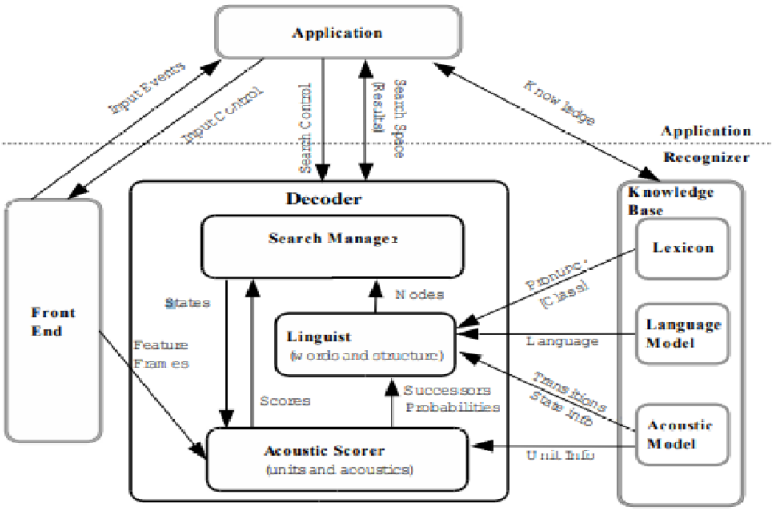

Figura 4.6: Sphinx-4 Arhitectura Sistemului[19]

- **FrontEnd**. Acesta preia unul sau mai multe semnale de intrare şi le parametrizează într-o secvenţa de feature-uri(Features).

  Figura 4.7 ilustrează o reprezentare detaliată a modulului Front-End. Modulul este format din câteva blocuri ce comunica (communicating blocks), fiecare având o intrare(input) şi o ieşire(output), iar fiecare ieşire este intrarea urmatorului bloc. Când un bloc este pregatit pentru a primi mai multe date, acesta citeşte date de la predecesorul său.
  Acesta interpretează datele pentru a afla dacă informaţia primită reprezintă date de vorbire (speech data) sau un semnal de control. Un semnal de control poate indica începutul sau sfîarşitul unui discurs sau poate indica renunţarea la anumite date sau o problemă. Dacă datele ce le primeşte reprezintă un discurs, acesta este procesat şi ieşirea este buffer-ată aşteptând ca uramtorul bloc sa o ceară.

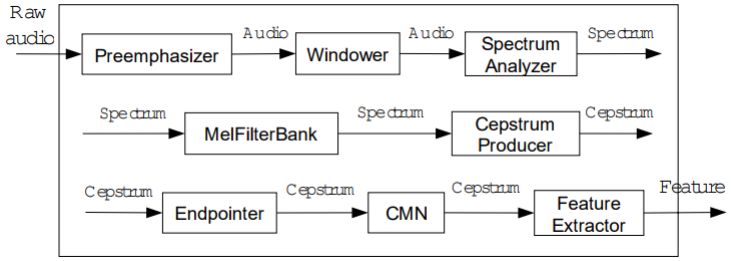

Figura 4.7: Sphinx-4 Front End[19]

- **Decoder**. Este folosit pentru a translata orice model standard de limbă, împreuna cu informaţiile legate de pronunţare din Dicţionar(Dictionary) şi informaţia structurată de la un set sau mai multe seturi de modele acustice(Acoustic Models) într-un Grafic de Căutare(Search Graph). Acesta este format din 3 module:

  – **Search Manager**. Funcţia principală a manager-ului este să construiască un arbore de posibilităţi şi să caute în el cea mea bună ipoteză. Construcţia arborelui se face bazându-se pe informaţiile obţinute din linguist. În plus manager-ul comunică cu acoustic scorer-ul pentru a obţine scoruri acustice (acoustic scores) pentru datele primite.

    Search Manager-ul se foloseşte de un arbore de token-uri(token tree). Token tree-ul consistă dintr-un set de token-uri ce conţine informaţii legate de căutare şi oferă un istoric complet al tuturor căilor active din search(cautare). Fiecare token conţine scoruri legate de limba (language) şi acustica (acoustic) la un punct dat, o referinţă SentenceHMM. Această referinţă îi permite managerului să poată categorisii token-urile în senone, context-dependent phonetic unit, pronunciation, word şi grammar state.

    Căutarea prin token tree şi sentence HMM se poate face în doua moduri: depth-first(DFS) sau breadth-first(BFS). DFS este similar cu stack decoding iar BFS

se face folosind algoritmul Viterbi şi Bush-derby.

- **Linguist**. Acesta translatează constrângerile lingvistice oferite sistemului într-o structura de date numită gramatică (grammar) ce este foosită de search manager. Constrângerile lingvistice sunt în general oferite sub formă de gramatici, modele de limbă N-gram, etc.

  Gramatica este un grafic orientat, unde fiecare nod reprezintă un set de cuvinte ce pot fi rostite la un moment dat. Nodurile sunt conectate prin arce ce au asociate probabilităţi acustice şi lingvistice ce sunt folosite pentru a prezice tranziţia de la un nod la altul.

  Sphinx-4 oferă mai multe loader-e de gramatică ce sunt folosite pentru a încărca formate de gramatici externe şi pentru a genera structura gramaticii interne. Gramatica este mai departe compilată într-o propoziţie HMM (Sentence HMM) ce este un graf de stări orientat unde fiecare stare reprezintă o unitate de vorbire. Nodurile din gramatică sunt descompuse într-o serie de stări de cuvinte. Stările cuvintelor sunt mai departe descompuse în stări de pronunţare, bazate pe datele din dicţionar conţinut de linguist. Fiecare stare de pronunţare este descompusă într-o serie de stări de unitate unde fiecare unitate reprezintă foneme,diphones, etc. Fiecare unitate este apoi descompusă într-o secvenţă de stări HMM. Astfel Sentence HMM compune toate aceste stări.

- **Acoustic Scorer**. Obiectivul acoustic scorer-ului este de a calcula probabilitatea stării de ieşire sau valorea densităţii pentru diferite stari pentru orice vector de intrare dat. Acesta oferă calificative (scores) la cererea modulului de căutare. Pentru a calcula aceste calificative, acoustic scorer-ul trebuie să comunice cu modulul Front-End pentru a obţine feature-uri pentru care se vor calcula aceste calificative. Acoustic scorer-ul pastrează toate informaţiile legate de stările densităţiilor de ieşire. Astfel modulul de căutare ignoră daca punctarea (scoring-ul) se face cu HMM continue, semi-continue sau discrete.

- **Knowledge Base**. Acesta conţine datele necesare pentru procesul de recunoaştere vocală. Modulul conţine Lexiconul (lexicon) ce conţine pronunţia cuvintelor, modelul lingvistic (language model) ce conţine limba folosita şi modelul acustic (acoustic model) ce conţine tranziţiile dintre cuvinte.

4.4.2 MarryTTS

MaryTTS este o platformă open-source cu o arhitectură modulară folosită pentru sistemele text-to-speech. Conform [21], primele versiuni ale acestei platforme au apărut în jurul aniilor 2000, fiind un proiect realizat de Marc Schroder în colaborare cu DFKI (Deutsche Forschungszentrum fur Kunstliche Intelligenz) şi Institute of Phonetics al Saarland University.

Mary permite procesarea pas cu pas permiţând accesul la rezultatele parţiale din timpul procesului. Sistemul este compus din mai multe module distincte ce le permite utilizatorilor sa îşi dezvolte propriile module şi să le introducă uşor în sistem.

Conform [22], arhitectura sistemului este ilustrata in figura 4.8, aceasta fiind similara cu arhitectura tipica a unui sistem TTS descris de Dutoit [23].

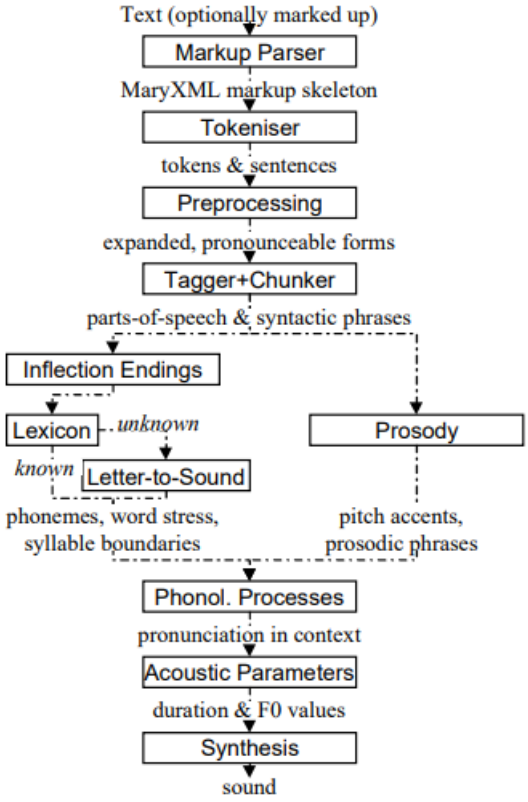

Figura 4.8: Arhitectura Sistemului MaryTTS [22]

- **Optional Markup Parser**. Sistemul acceptă atat text simplu (plain text) ca intrare (input) sau input marcat pentru sinteza de vorbire (speech synthesis) cu un limbaj de sinteza cum ar fi SABLE. Astfel limbajul de marcare (markup language) oferit ca input este tranlatat într-un limbaj de marcare intern, mai low-level denumit MaryXML. Acesta se bazeaza pe XML.

- **Tokeniser**. Acesta împarte textul in token-uri(de exemplu: cuvintele şi semnele de punctuaţie). Foloseşte un set de reguli determinate din analiza corpus-ului. Fiecare token se află intre tag-uri de tip <t>...</t>. În plus semnele de punctuaţie sunt folosite pentru a determina începutul sau sfârşitul unei propozitţii, iar acestea sunt marcate folosind tag-urile ... .

- **Preprocessing**. În acest modul token-ii a căror formă de vorbire nu corespunde exact cu forma scrisă sunt înlocuiţii cu o forma mai pronunţabilă.

- **Part-of-speech tagger / chunk parser**. Part-of-speech tagging este realizată folosind un etichetator statistic (statistical tagger) TnT, folosind Stuttgart-TÃijbingen Tagset (STTS). Un parser de bucăţi (chunk parser) este folosit pentru a determina limitele frazelor esenţiale, a frazelor prepoziţionale şi a frazelor adjective.

- **Phonemisation**. Foloseşte un alfabet fonetic (phonetic alphabet) pentru traducerile fonetice. Pentru limba germana se foloseşte SAMPA. În plus se foloseşte un lexicon extins, ce se ocupa de cuvintele cunoscute şi un algoritm ce converteşte litere la sunete (letter-to-sound algorithm) este folosit pentru cuvintele necunoscute.

- **Prosody rules**. Prozodia este modelată folosind GToBI, o adaptare a ToBI(Tones and Break Indices) pentru limba germana. ToBI este folosit pentru a face distincţia între cuvinte pronunţate cu accente diferite. Regulile de prozodie asignează etichete (labels) simbolice GToBI iar în urmatoarele etape acestea sunt translatate în ţinte concrete F0(concrete F0 targets) şi durate de pauza(pause durations).

- **Postlexical phonological processes**. Odată ce cuvintele sunt transcrise într-un şir fonetic standard incluzând limetele silabelor şi stresul lexical şi etichetele prozodice pentru accentele înalte şi limitele enunţurilor prozodice. Rezultatul reprezentării fonetice poate fi astfel restructurat de către o serie de reguli fonetice. Aceste reguli operează asupra contextului fonetic cum ar fi accentul, stresul cuvintelor, articularea, etc.

  Ieşirea (output-ul) acestui modul ofera structura bogată MaryXML, ce conţine toate informaţiile adaugate de modulele anterioare.

- **Calculation of acoustic parameters**. Acest modul realizează translatarea din domeniul simbolic in cel fizic. Structura MaryXML este interpretată de către regulile de durată (duration rules) şi regulile de realizare GToBI.

  Ieşirea acestui modul nu mai este o structura MaryXML ce o listă ce conţine toate segmentele individuale cu duraţiile respective şi ţintele F0 (F0 targets).

- **Synthesis**. În prezent, MBROLA este utilizat pentru sintetizarea ieşirii modulului anterior. Datorită ahitecturii modulare a sistemului orice modul de sinteză ce prezintă o interfaţă similara poate fi uşor folosit.

4.4.3 Selenium

Selenium [1] a apărut in 2004 când Jason Huggins testa o aplicaţie internă la ThoughtWorks. El şi-a dat dat seama că poate face altceva cu timpul său decât sa tot repete manual aceleaşi teste. Astfel el a dezvoltat o bibliotecă JavaScript care ar putea conduce interacţiuni cu pagina, permiţându-i să redea automat testele în mai multe browser-e. Această biblioteca a devenit în cele din urma Selenium Core ce stă la baza Selenium Remote Control (RC) şi Selenium IDE.

Selenium IDE este un mediu de dezvoltare integrat pentru scripturile de seleniu. Acesta este implementat ca extensie pentru Chrome şi Firefox şi ne permite să înregistrăm, să modificăm şi să depanăm testele.

Caracteristici

- Permite înregistrarea şi redarea cu uşurinţă
- Selectarea inteligentă a câmpurilor folosind ID-uri, nume sau XPath dupa necesitate
- Completarea automată a tuturor comenzilor comune ale selenium-ului
- Depanare şi setarea de breakpoint-uri
- Totate testele situate într-un singur fişier

4.4.4 JavaFX

Conform [24], JavaFX este un set de pachete grafice şi media ce le permite dezvoltatorilor să proiecteze, testeze şi să implementeze aplicaţii complexe ce funcţionează în mod consistent pe mai multe platforme.

JavaFX a apărut cu scopul de a înlocui librăia cu acelaşi scop dar mai veche şi anume Swing. În prezent librăria este la versiunea 2.2, acesta poate fi integrata în totalitate în aplicaţii ce folosesc Java Runtime Environment(JRE) şi Java Development Kit(JDK) versiunea 7 sau mai mare.

Principalele avantaje aduse de către JavaFX faţă de Swing sunt următoarele[2]:

- FXML. JavaFX foloseşte un fişier FXML, ce le permite dezvoltatorilor să creeze interfeţe utilizator separate de logica aplicaţiei. FXML este un limbaj de tip markup.
- JavaFX Scene Builder. Este un tool ce vine în ajutorul dezvoltatorilor, acesta oferind funcţionalitatea de drag-and-drop a componentelor de UI. În final tool-ul generază cod FXML ce poate fi folosit în aplicaţie.

[1] https://www.seleniumhq.org/
[2] https://docs.oracle.com/javafx/2/swing/overview.htm

- CSS Support. JavaFX permite folosirea de elemente de ”înfrumuseţare”, exact ca şi CSS-ul în paginile HTML. Utilizarea este simplistă şi similara celei din programarea web.

- JavaFX Media Support. Platforma oferă funcţionalităţi de a reda fişiere audio şi video. Fişierele media sunt valabile în toate platformele unde JavaFX este suportat.

- Animaţii. JavaFX a introdus utilizarea animaţiilor pentru a oferi o înfăţişare mai moderna şi mai dinamica aplicaţiei.

- Conţinut HTML. Oferă posibilitatea de render a codului HTML în aplicaţiile Java.

4.4.5 Gradle

Gradle [3] este un sistem de automatizare bazat pe open-source care se bazează pe conceptele Apache Ant şi Apache Maven şi introduce un limbaj specific domeniului Groovy (DSL) în locul formularului XML folosit de Apache Maven pentru declararea configuraţiei proiectului.Gradle utilizează un grafic aciclic direcţionat (DAG) pentru a determina ordinea în care pot fi executate sarcini.

Gradle a fost proiectat pentru modele multi-proiecte, care pot deveni destul de mari. Acesta susţine construirea incrementală a aplicaţiilor prin determinarea inteligentă a părţilor, din copacul de construcţie (build tree), ce sunt actuale (up to date). Astfel nici o sarcină care depinde numai de acele părţi nu trebuie să fie re-executată.

În figura 4.9 ilustrează faptul că Gradle se potriveşte generaţiei de tool-uri de construire (build tools) şi satisface multe cerinţe ale instrumentelor de construcţie modernă.[25]

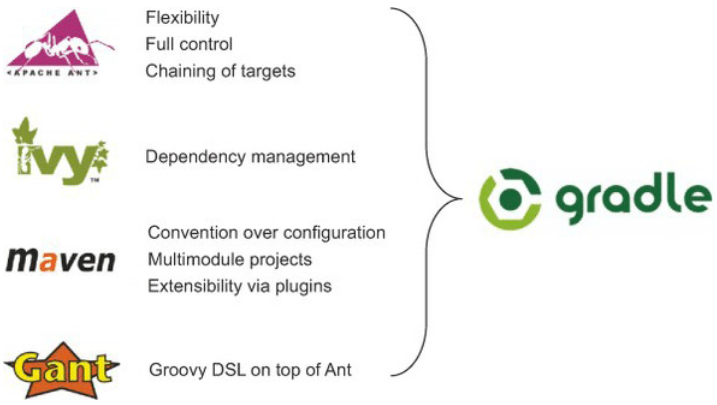

Figura 4.9: Gradle
www.drdobbs.com/jvm/why-build-your-java-projects-with-gradle/240168608

[3] https://gradle.org/

4.4.6 Git

Git [4] reprezintă un sistem de control al versiunilor pentru urmărirea modificărilor şi coordonarea activităţii fişierelor în cadrul unui proiect mai complex. Sistemul de control al versiunilor ne permite să avem "versiuni" ale unui proiect, care arată schimbările ce au fost făcute şi permite revenirea la o versiune anterioară dacă este necesar. Git permite structurarea mai eficientă a codului şi o gestiune mai uşoară a acestuia, iar în proiecte mai complexe uşurează şi permite munca în echipă.

În contextul Git, un commit reprezintă o înregistrare a modificărilor din repository (proiect). Figura 4.10 ilustreaza commuit-urile realizate în productia acestui proiect,

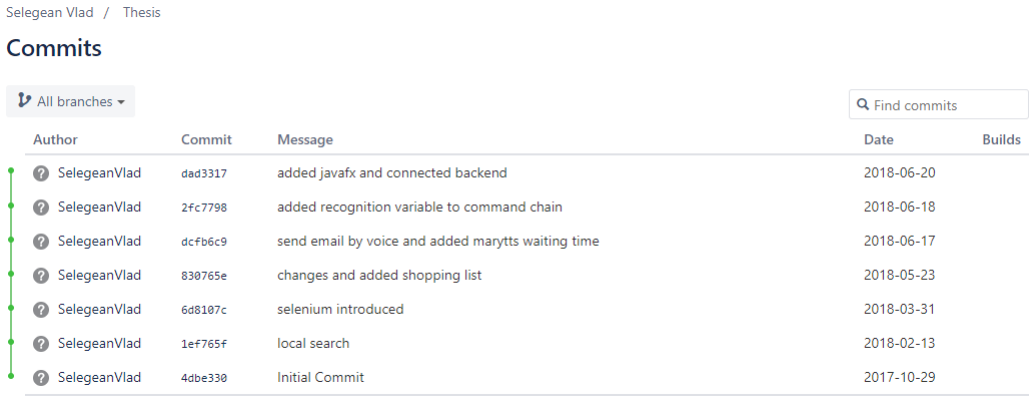

Figura 4.10: Git Commits

Există şi alte alternative faţă de Git şi anume Mercurial SCM şi TFS. Diferenţele dintre acestea nu sunt mari dar am ales Git deoarece este cel mai popular sistem de control al versiunilor la ora actuală, este gratuit, open source şi gestionează totul de la proiecte mici la proiecte complexe cu rapiditate şi eficientă.

4.4.7 GDPR

Actul denumit GDPR(General Data Protection Regulation), adoptat de Parlamentul European în aprilie 2016, a intrat în vigoare pe 25 mai 2018. Actul prevede ca toate datele cu caracter personal trebuiesc protejate. Conform [5] aceste date sunt *înseamnă orice informaţii privind o persoană fizică identificată sau identificabilă (persoana vizată); o persoană fizică identificabilă este o persoană care poate fi identificată, direct sau indirect, în special prin referire la un element de identificare, cum ar fi un nume, un număr de identificare, date de localizare, un identificator online, sau la unul sau mai multe elemente specifice, proprii identităţii sale fizice, fiziologice, genetice, psihice, economice, culturale sau sociale.*

[4] https://github.com/

[5] https://www.avocatnet.ro/articol_46806/GDPR-Regulamentul-european-de-protec%C8%9Bie-a-datelor-se-aplic%C4%83-%C8%99i-Romaniei-in-mod-direct-incepand-de-azi.html#

Având în vedere ca acest sistem este folosit pentru a accesa anumite site-uri web cu caracter personal (ex: Facebook) şi că este nevoie de username şi parolă pentru operaţia de logare automată folosind comenzile vocale. Se poate spune că informaţiile oferite nu sunt publice publicului sau folosite în alt scop. Totate datele introduse în aplicaţie sunt la latitudinea utilizatorului şi doar el le poate face publice.

Capitolul 5

Proiectare de Detaliu şi Implementare

Obiectivul principal al acestei aplicaţii este acela de a veni în ajutorul utilizatorului oferindu-i anumite funcţionalităţii ce duc la creşterea eficienţei şi a productivităţii. Aplicaţia conţine doua nivele, nivelul de front-end(interfaţa utilizator) şi nivelul de back-end(funcţionalitatea propriu-zisă).

Astfel acest captiol prezintă nivelele din cadrul aplicaţiei, modul în care acestea au fost implementate şi interacţiunea dintre acestea.

5.1 Arhitectura Sistemului

Arhitectura sistemului face referire la modul de organizare şi modul de funcţionare al componentelor în cadrul aplicaţiei. Sistemul este construit sub forma unei aplicaţii de tip client-server, având 2 nievele cel de front-end şi cel de back-end. Aceasta este ilustrată în Figura 5.1

Clientul (utilizatorul) este obiectivul principal, acţiunea fiind concentrată pe el. Acesta rosteste o comandă şi aşteaptă înapoi un raspuns. Comanda rostită este captată de către microfonul calculatorului şi trimisă mai departe sistemului.

Astfel, în Figura 5.2 este ilustrată diagrama de secvenţă a unei acţiuni generale între utilizator şi sistem, mai exact face referire la interacţiunea utilizator-sistem şi modul în care se ajunge la un răspuns.

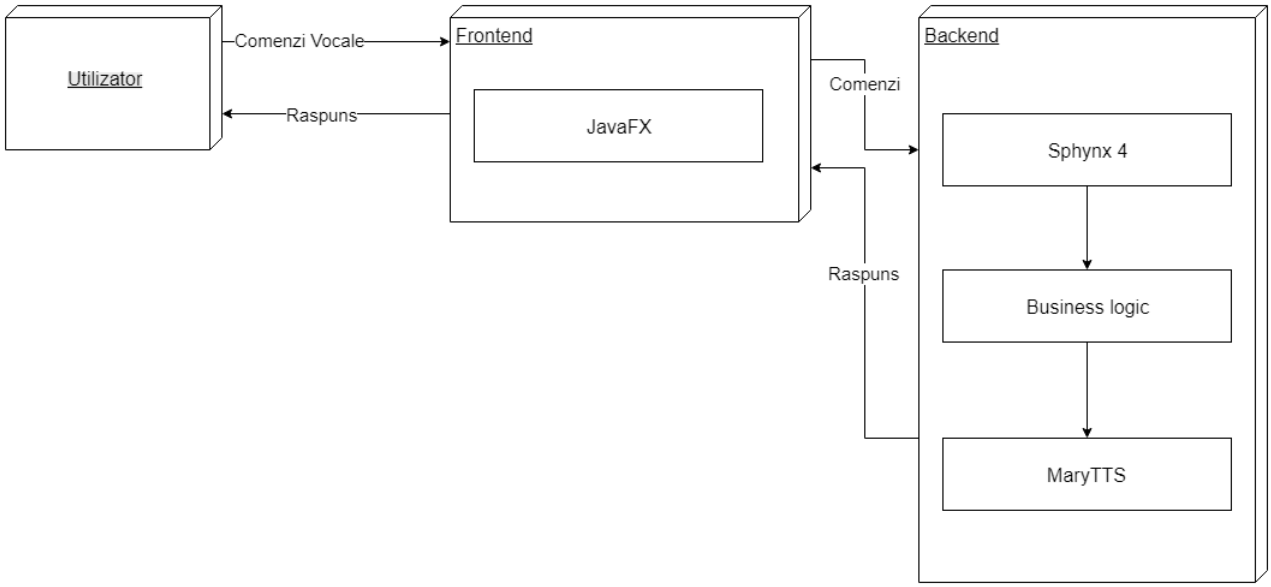

Figura 5.1: Arhitectura Sistemului

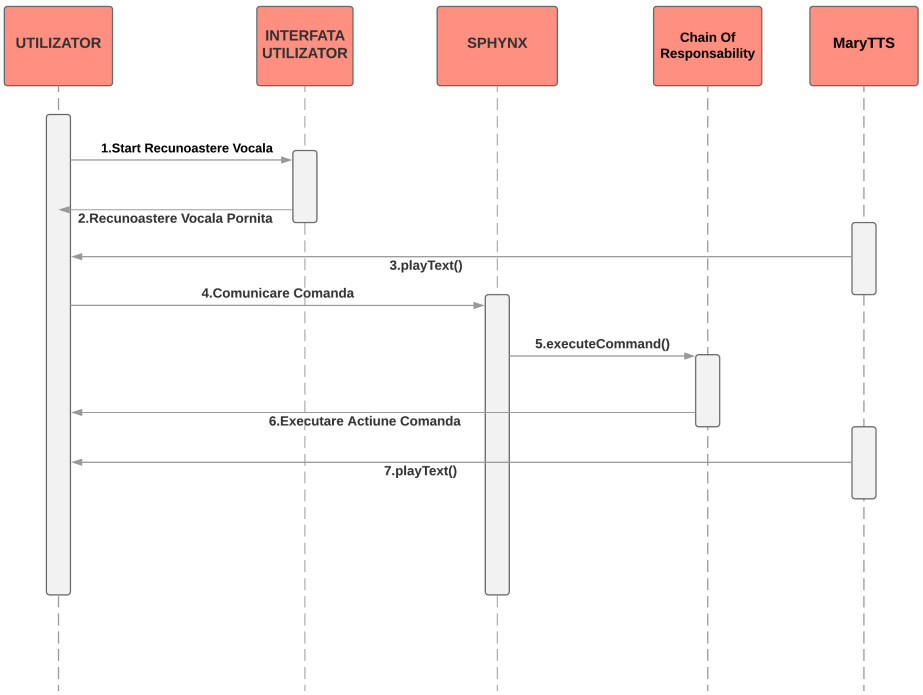

Figura 5.2: Diagrama de Secvenţă

5.2 Nivelul Front-end

Nivelul front-end este prezent pentru a face aplicaţia user friendly şi pentru a ajuta la anuminte funcţionalitaţi(i.e logarea automată). Astfel componenta principală a acestui nivel o constituie componenta JavaFX. Sa folosit această librărie deoarece ofera un look mai modern şi mai dinamic interfeţei. JavaFX prezintă mai multe avantaje faţă de vechea librarie Swing, printre aceste avantaje se numară:

- FXML. JavaFX foloseşte un fişier FXML, ce le permite dezvoltatorilor să creeze interfeţe utilizator separate de logica aplicaţiei. FXML este un limbaj de tip markup.
- CSS Support. JavaFX permite folosirea de elemente de "înfrumuseţare", exact ca şi CSS-ul în paginile HTML. Utilizarea este simplistă şi similara celei din programarea web. Exemplu de cod CSS folosit în aplicatîe:

```
.root {
    -fx-background-color: LIGHTSTEELBLUE;
}

#startButton{
    -fx-background-image: url("mic.png");
    -fx-background-position: center;
    -fx-background-size: 55 55;
}
```

Aplicaţia prezintă o interfaţă utilizator ce este împarţită în două tab-uri, unul principal denumit *Main* şi unul secundar denumit *Accounts*. Tab-ul *Main*, ilustrat în Figura 5.3, este tab-ul principal de unde se porneşte respectiv opreşte recunoaşterea vocală. În plus mai există un text field în care se pot scrie comenzi în cazul în care utilizatorul nu poate vorbi sau comanda nu a fost înţeleasă de mai multe ori.

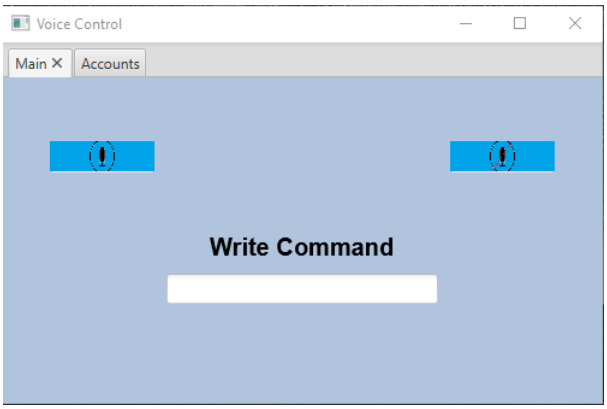

Figura 5.3: Interfaţa Principală

Al doilea tab este *Accounts*, ilustrat în Figura 5.4, este tab-ul secundar în care sunt trecute conturile şi parolele pentru logarea automată. Momentam aplicaţia suporta logarea automata pe Facebook şi Gmail. Acestea nu sunt cerute vocal din cauze de siguranţă. Aplicaţia introduce automat date de test, acestea trebuiesc schimbate înainte de a rula recunoaşterea vocală pentru a se putea executa logarea.

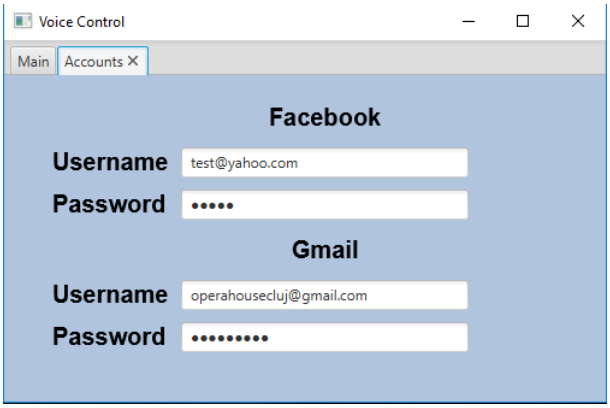

Figura 5.4: Interfaţa Accounts

Astfel clasa principală folosită pentru interfaţa utilizator este *MainJavaFX*. Aceasta prezinta doua funcţii importante, acestea fiind *start* şi *launchJavaFX*. În *start* are loc construirea interfeţei iar în *launchJavaFX* are loc lansarea ei. Principalele variabile de clasă sunt accesate din clasa principală pentru a prelua date din interfaţă şi a porni respectiv operi recunoaşterea vocală. Aceste variabile sunt:

```
public Button startButton;
public Button stopButton;
public TextField commandField;
public TextField usernameFacebookField;
public PasswordField passwordFacebookField;
public TextField usernameGmailField;
public PasswordField  passwordGmailField;
```

5.3 Nivelul Back-end

Nivelul back-end primeşte comenzile vocale de la utilizator şi oferă înapoi un raspuns, acesta fiind execuţia comenzii. Acest nivel prezintă trei componente mari, acestea fiind:

- Componenta Sphynx 4
- Componenta Business Logic
- Componenta MaryTTS

Toate aceste trei componentele comunica între ele folosindu-se de datele oferite de componenta inferioară. Sphynx 4 se ocupă de recunoaşterea vocală a celor spuse de utilizator, a comenzii. Business logic interpretează această comandă şi executa o acţiune şi oferă un raspuns. Raspunsul este preluat de MaryTTS şi este rostit utilizatorului.

5.3.1 Componenta Sphynx 4

Această componentă este responsabilă de recunoaşterea vocală având la bază librăria Sphinx4. Aceasta primeşte sunetele captate de microfon şi încearcă sa le recunoască conform unor gramatici sau dicţionare, modele de limbă şi modele acustice. Astfel în aplicaţie a fost construită o clasă ce se ocupa de configurarea librăiei. Clasa se numeşte *ResourceConfiguration*, fiind responasabilă de setarea modelului acustic, gramatica/dicţionarul şi modelul de limbă.

Modelul acustic este cel oferit de cei de la Spynx şi anume este modelul acustic pentru limba engleza(Acesta se găseşte aici). Pentru *gramatică/dicţionar* şi pentru *modelul de limbă* ne-am folosit de un tool ce ne genereaza aceste doua modele în funcţie de corpus-ul creat. Corpus-ul folosit este unul minimal conţinând comenzile folosite şi anumite cuvinte des folosite. Cu cât corpus-ul este mai vast cu atât recunoaşterea vocală devine mai inexactă deoarece există prea multe cuvinte asemanătoare. Pentru corpus-uri vaste e nevoie de o îmbunătăţire a precesului de recunoaştere vocală. Un exemplu de model de dicţionar în Figura 5.5a şi modelul de limba în Figura 5.5b

```
ADD AE D
ALARM   AH L AA R M
ARAD    AE R AH D
BREAD   B R EH D
CALCULATOR  K AE L K Y AH L EY T ER
CANDY   K AE N D IY
CLOSE   K L OW S
CLOSE(2)    K L OW Z
CLUJ    K L UW JH
CLUJ(2) S IY EH L Y UW JH EY
COLA    K OW L AH
DISPLAY D IH S P L EY
DIVIDE  D IH V AY D
EIGHT   EY T
EMAIL   IY M EY L
EXCEL   IH K S EH L
FACEBOOK    F EY S B UH K
FIVE    F AY V
FOUR    F AO R
GMAIL   G M EY L
GOOGLE  G UW G AH L
IN  IH N
```

(a) Exemplu de Model de dicţionar

```
This model based on a corpus of 56 sentences and 58 words

\data\
ngram 1=58
ngram 2=112
ngram 3=56

\1-grams:
-0.7782 </s> -0.3010
-0.7782 <s> -0.2218
-2.5263 ADD -0.2218
-2.5263 ALARM -0.2218
-2.5263 ARAD -0.2218
-2.5263 BREAD -0.2218
-2.5263 CALCULATOR -0.2218
-2.5263 CANDY -0.2218
-2.5263 CLOSE -0.2218
-2.5263 CLUJ -0.2218
-2.5263 COLA -0.2218
-2.5263 DISPLAY -0.2218
-2.5263 DIVIDE -0.2218
-2.5263 EIGHT -0.2218
```

(b) Exemplu de Model de limbă

Setarea acestor modele se face în cod folosindu-ne de metodele ajutătoare oferite de librarie.

```
configuration = new Configuration();
configuration.setAcousticModelPath("/edu/cmu/sphinx/models/en-us/en-us");
configuration.setDictionaryPath("..\\resources\\1400.dic");
configuration.setLanguageModelPath("..\\resources\\1400.lm");
```

5.3.2 Componenta Business Logic

În acestă componentă are loc interpretarea comenzii rostite şi execuţia acţiunii cerute sau în cazul în care comanda nu este validă se oferă ca răspuns ”comandă invalidă” şi se cere rostirea din nou a comenzii. Logica din acestă componentă se bazează pe design pattern-ul *Chain of Responsibility*. Astfel comanda trece prin mai multe clase ce sunt grupate logic, în funcţie de acţiunile executate. În Figura 5.6 se poate implementarea acestui lanţ(chain).

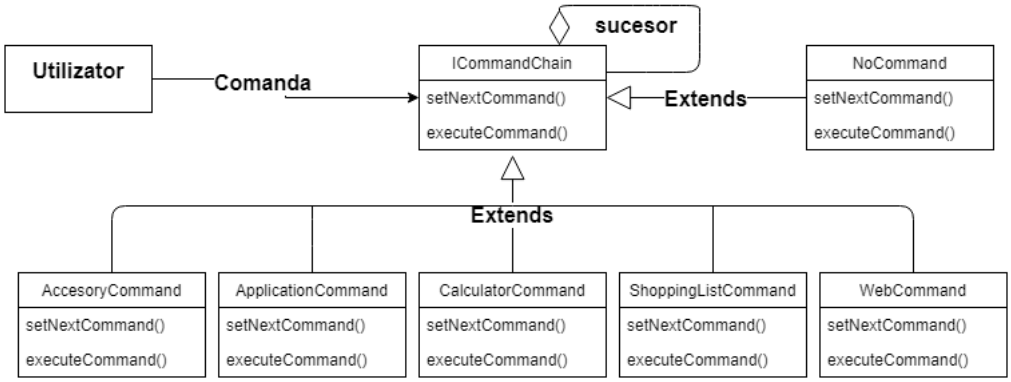

Figura 5.6: Chain Of Responsibility Diagram

În continuare, în Figura 5.7 este ilustrat flow-ul prin care trece o comandă în lanţ(chain). Căutarea se face pornind căutarea comenzii din clasa *WebCommand*, fiind cea mai mare clasă şi fiind una cu comenzile cele mai folosite, se ocupa de toate comenzile ce au legatură cu navigarea pe internet. Daca comanda nu este găsită in acesta clasă se trece în *ApplicationCommand*, aici fiind executate comenzi de deschidere de aplicaţii locale. Urmează *CalculatorCommand*, clasă folosită pentru a calcula operaţii minimale pe numere mici(i.e adunare, scadere, înmultire, împartire).După acesta vine *AccessoryCommand*, ce conţine comenzi pentru căutarea locală şi folosirea unui transcriber pentru o înregistrare audio. Înainte de final este clasa *ShoppingListCommand*, fiind folosită pentru funcţionalitatea de adăugare, ştergere de elemente pentru o lista de cumpărături şi rostirea acesteia folosind MaryTTS.În final este finalul lanţului în care se află clasă *NoCommand* în care se oferă ca răspuns faptul că comanda nu este valida şi se cere reintroducerea acesteia. În continuare se poate observa implementarea prin cod.

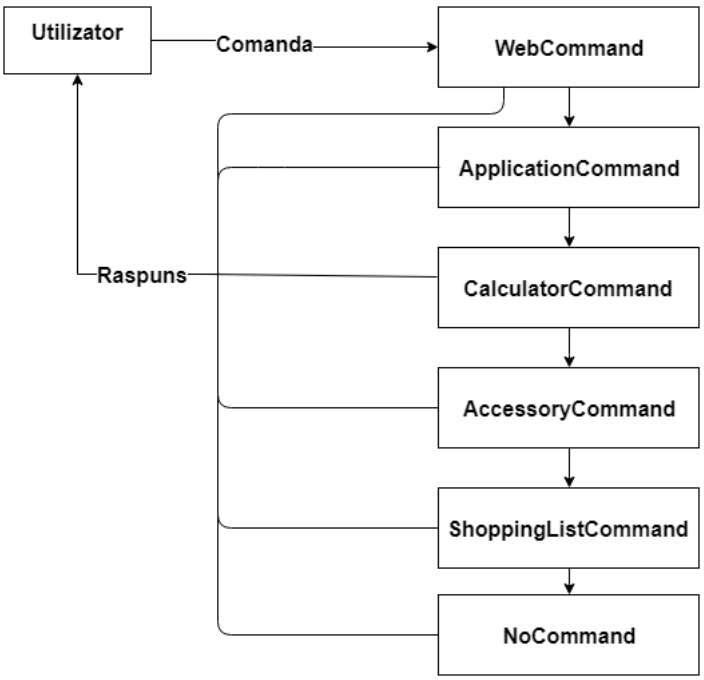

Figura 5.7: Chain Of Responsibility Flow Diagram

```
//Initialize Chain
 WebCommand webCommand = new WebCommand();
CalculatorCommand calculatorCommand = new CalculatorCommand();
ApplicationCommand applicationCommand = new ApplicationCommand();
AccessoryCommand accessoryCommand = new AccessoryCommand();
ShoppingListCommand shoppingListCommand = new ShoppingListCommand();
NoCommand noCommand = new NoCommand();

//Set chain
webCommand.setNextCommand(applicationCommand);
applicationCommand.setNextCommand(calculatorCommand);
calculatorCommand.setNextCommand(accessoryCommand);
accessoryCommand.setNextCommand(shoppingListCommand);
shoppingListCommand.setNextCommand(noCommand);
```

În continuare sunt prezentate pe rând fiecare componenta din acest Chain of Responsability.

WebCommand

Acestă componentă este cea mai mare şi mai complexă clasă. Acesta se ocupa de toate comenzile ce au legatură cu navigarea pe internet. Pentru a putea deschide browser-ul şi pentru a putea accesa anumite pagini şi pentru a putea executa anumite funcţii pe aceste pagini se foloseşte Selenium IDE pentru browser-ul Google Chrome. Astfel pentru acesta este nevoie de un tool numit *ChromeDriver* a cărui versiune trebuie sa coincidă cu versiunea browser-ului.

Pentru a identifica comanda şi a executa acţiunea corectă se foloseşte un HashMap şi un Switch. În HashMap se pun numele comenzilor implementate, acestea fiind filtrate în funcţie de cuvintele cheie din comandă rostită folosindu-se Java 8 streams. Exemplu în cod de aşezare, filtrare şi selecţie.

```
HashMap<String, String> commandsHashMap = new HashMap<>();
        commandsHashMap.put("GOOGLE", "GOOGLE");
        commandsHashMap.put("GMAIL", "GMAIL");
        ...

String keyResult = commandsHashMap.entrySet()
        .stream()
        .filter(x -> command.toUpperCase().contains(x.getKey()))
        .map(Map.Entry::getValue)
        .findFirst()
        .orElse("");
switch (keyResult) {
 case "GOOGLE":
      maryTTL.playText("Opening google...");
      waitMaryToFinish("Opening google...");
      enterSite("https://www.google.com/");
      break;
 ...
 default:
      commandChain.executeCommand(command, recognizer, parameters);
      break;
}
```

După cum se poate observa pentru accesarea unui site se foloseşte o funcţie denumită *enterSite* ce primeşte ca parametru un String. Aceasta are scopul de a accesa un site, adresa acestuia fiind trimisă ca şi parametru. Se fac verificări daca driver-ul de selenium este deja iniţializat se acceseaza site-ul, dacă nu se iniţializează driver-ul şi are loc accesul.

```
private void enterSite(String url) {
        if (driver == null) {
```

```
            driver = new ChromeDriver(chromeOptions);
            driver.get(url);
        } else {
            boolean error = false;
            try {
                driver.get(url);
            } catch (Exception e) {
                error = true;
                System.out.println("Selenium Exception");
            }

            if (error) {
                driver = new ChromeDriver(chromeOptions);
                driver.get(url);
            }

        }
    }
```

Acestă componentă prezintă cele mai multe comenzi, acestea realizând acţiuni atât simple cât şi complexe. Lista de funcţionalităţi este următoarea:

- Google -> Accesare Google
- Gmail -> Accesare Gmail
- Maps -> Accesare Google Maps
- Facebook -> Accesare Facebook
- Youtube -> Accesare Youtube
- University -> Accesare site UTCN
- Weather -> Accesare site ce prezintă vremea în locaţia din care se face requestul
- Radio -> Accesare site de posturi radio
- New Tab -> Deschide un tab nou în browser
- Close Tab -> Închide tabul curent din browser
- Sign in -> Realizează operaţia de login pe Facebook sau Gmail
- Sign out -> Realizează operaţia de logout de pe Facebook sau Gmail

- Send email -> Acceseaza Gmail şi începe scrierea unui email
- Route -> Accesează Google Maps şi caută ruta între punctul susrsa şi destinaţie, acestea fiind rostite în prealabil
- Wikipedia -> Acceseaza Wikipedia
- Search Wikipedia -> Încearca căutarea cuvintelor rostite pe Wikipedia. Daca exista date despre acestea se va afişa pagina corespunzatoare.(i.e. Romania, First World War)

ApplicationCommand

Componenta aceasta este a doua cea mai mare şi este responsabilă de deschidera aplicaţiilor locale. Se foloseşte obiectul de tip *Runtime* pentru execuţia comenzilor de pornire a aplicaţiilor, funcţia folosită este *exec()*. Exemplu de execuţie pentru deschiderea aplicaţiei Paint:

```
case "PAINT":
 maryTTL.playText("Opening paint...");
 waitMaryToFinish("Opening paint...");
 runtime.exec("mspaint.exe");
 break;
```

Lista de funcţionalităţi este următoarea:

- Calculator -> Deschidere Calculator
- Notepad -> Deschidere Notepad
- Paint -> Deschidere Paint
- Notes -> Deschidere Notes
- Word -> Deschidere Word
- Excel -> Deschidere Excel
- Player -> Deschidere Media Player
- Alarm -> Deschidere setare alarma

CalculatorCommand

Rolul acestei componente este acela de a simula un mini-calculator, fiind posibila execuţia operaţiilor matematice de bază(adunare, scadere, împartire, înmultire) pe numere mici(de la 1 la 9). Răspunsul este calculat în funcţia *computeAnswer()* ce primeşte ca String ecuţia. Se mai foloseşte o funcţie numita *convertStringToNumber()* pentru a converti numărul citit din String în integer.

```
private String computeAnswer(String ecuation) {
 String[] words = ecuation.split(" ");
 int firstNumber = covertStringToNumber(words[0]);
 int secondNumber = covertStringToNumber(words[2]);
 int result = 0;
 switch (words[1]) {
       case "PLUS":
             result = firstNumber + secondNumber;
             break;
       case "MINUS":
             result = firstNumber - secondNumber;
             break;
       case "MULTIPLY":
             result = firstNumber * secondNumber;
             break;
       case "DIVIDE":
             result = firstNumber / secondNumber;
             break;
       default:
             System.out.println("Can not Calculate.......");
             break;
   }
  return Integer.toString(result);
 }
```

AccessoryCommand

Această componentă se ocupă cu funcţionalitatea de căutare în sistemul de fişiere local şi pentru funcţia de transcriber. În sistemul de fişiere local căutarea se face din aproape în aproape. E nevoie de numele fişierului căutat şi disk-ul în care să se facă căutarea. Funcţia *findFile()* se ocupă de această căutare iar ca şi parametrii primeşte numele fişierului şi disk-ul în care se face căutarea.

```
private void findFile(String name, File file) {
        File[] list = file.listFiles();
```

```
        if (list != null) {
            for (File fil : list) {
                if (fil.isDirectory()) {
                    findFile(name, fil);
                } else if (name.equalsIgnoreCase(fil.getName())) {
                    System.out.println(fil.getParentFile());
                }
            }
        }
    }
```

Operaţia de transcriber se referă la faptul că se introduce un fişier audio şi aceasta oferă ca raspuns, trei cele mai bune ipoteze din textul auzit. Acestă funcţie este oferită de libraria Sphynx4, demo aici. Partea importantă de cod este iniţializarea obiectului de recunoaştere vocală ca fiind **StreamSpeechRecognizer** pentru fişiere audio şi nu **LiveSpeechRecognizer** pentru recunoaşterea vocală în momentul vorbirii.

```
recognizer = new StreamSpeechRecognizer(configuration);
stream = new FileInputStream(new File(filePath));
while ((result = recognizer.getResult()) != null) {
 System.out.format("Hypothesis: %s\n", result.getHypothesis());
 System.out.println("List of recognized words and their times:");
 for (WordResult r : result.getWords()) {
    system.out.println(r);
 }
 System.out.println("Best 3 hypothesis:");
 for (String s : result.getNbest(3))
   System.out.println(s);
}
```

Lista de funcţionalităţi este următoarea:

- Search Local -> Căutarea unui fişier în sistemul de fişiere local
- Transcriber -> Introducerea unui fişier audio şi returnarea a trei ipoteze din înregistrarea introdusă.

ShoppingListCommand

Rolul acestei componente este acela de gestiona o listă de cumpărături. Prin gestionare se înţelege adăugarea de elemente şi ştergerea de elemente din listă. În plus se foloseşte MaryTTS pentru ai comunica utilizatorului conţinutul listei finale. Funcţiile importante sunt *addToShoppingList()* pentru adăugare, *removeFromShoppingList* pentru ştergere şi *displayShoppingList* pentru comunicarea listei finale.

NoCommand

Această componentă reprezintă finalul lanţului, dacă sa ajuns aici înseamnă că nu există comanda cerută s-au nu a fost înteleasă corect. Având în vedere că am ajuns la finalul lanţului, componenta oferă ca răspuns prin MaryTTS că acestă comandă nu este recunoscută şi se cere încercarea din nou a unei comenzi valide. Exemplu în cod:

```
String randomPhrase = generateHelloPhrase();
maryTTL.playText(randomPhrase);
maryTTL.waitMaryToFinish(randomPhrase);
Thread.sleep(3500);
```

Se observă faptul că se foloseşte funcţia *generateHelloPhrase()*, ceea ce ne returnează o replica random din cele scrie. Astfel am introdus mai multe răspunsuri pentru a face aplicaţia mai puţin predictibilă.

Main Class

Clasa Main, este clasă principală a proiectului. În aceasta au loc iniţializările necesare precum *LiveSpeechRecognizer*, iniţializarea lanţului(Chain of Responsability) şi a interfeţei utilizator. Aceste iniţializări au fost exemplificate în secţiunile anterioare.

În plus părţile importante sunt setarea evenimentelor pe butoanele principale şi preluarea datelor din text field-uri. Acestea au loc în acestă clasă şi astfel toate modificările şi toate datele legate de interfaţa se fac printr-un thread de JavaFX, folosind funcţia *Platform.runLater()*. Butoanele de start şi stop rulează în thread-uri separate pentru a nu bloca interfaţa şi pentru a ne permite să oprim recunoaşterea vocală când se doreşte.

Exemplificare pentru setarea evenimentelor pe butoane, controlul recunoaşterii vocale şi a procesului de selecţie şi execuţie a comenzilor.

Setarea butonului de start:

```
Platform.runLater(
() -> {
mainJavaFX.startButton.setOnAction((event) -> {
Runnable myRunnable = () -> {
keepRecognizing = true;
while (keepRecognizing) {
try {
String randomHelloPhrase = generateHelloPhrase();
maryTTL.playText(randomHelloPhrase);
maryTTL.waitMaryToFinish(randomHelloPhrase);
} catch (SynthesisException e) {
e.printStackTrace();}
```

```
...
}

Thread thread = new Thread(myRunnable);
thread.start();
};
       });
});
```

Setarea butonului de stop:

```
 Platform.runLater(
() -> {
mainJavaFX.stopButton.setOnAction((event) -> {
Runnable myRunnable = () -> {
System.out.println("Runnable running is stopping");
keepRecognizing = false;.
};
Thread thread = new Thread(myRunnable);
thread.start();
});
});
```

Preluarea de date din interfaţă, datele din tabul de Accounts sunt salvate într-o listă pentru a putea fi transmise mai departe:

```
commandText = mainJavaFX.commandField.getText();

parameters.add(mainJavaFX.usernameFacebookField.getText());
parameters.add(mainJavaFX.passwordFacebookField.getText());
parameters.add(mainJavaFX.usernameGmailField.getText());
parameters.add(mainJavaFX.passwordGmailField.getText());
```

5.3.3 Componenta MaryTTS

Această componentă este folosită pentru a putea comunica cu utilizatorul, oferindu-i un raspuns verbal în momentul în care se execută o acţiune sau răspunsul unei cereri(de exemplu: elementele din shopping list). Acesta are la bază libria MaryTTS.

Este necesara folosirea unui audio player pentru a putea genera text audio. Astfel în dezvoltarea clasei *MaryTTL* se foloseşte o clasă modificată a unui audio player ce aparţine de Pioneer Corporation, aceasta se numeşte *AudioPlayer*. Metoda folosită pentru a genera text audio se numeşte *playText*:

```
public void playText(String textToSpeak) throws SynthesisException {
        AudioInputStream audio = _maryInterface.generateAudio(textToSpeak);
        _audioPlayer.startPlayback(audio);
    }
```

Sa observat o problema legată de generarea text-ului audio, în momentul în care se un răspuns este în curs de rostire şi se generează înca unul, cel din urmă nu aşteaptă implicit finalizarea rostirii primului text şi astfel opreşte brusc rostirea primului text şi începe rostirea celui de-al doilea. Astfel pentru a evita astfel de probleme am creat o funcţie ajutătoare numită *waitMaryToFinish*, ce blochează thread-ul pentru un timp proporţional cu numarul de cuvinte ce v-or fi rostite de maryTTS. Funcţia este ilustrată mai jos:

```
private static void waitMaryToFinish(String text) {
        int waitTime = text.length() * 15;

        try {
            Thread.sleep(waitTime);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

5.4 Diagrama de clase

O diagramă de clase descrie structural sistemul. Acesta evidenţiză toate clasele folosite in implementare, atributele şi metodele acestora. În plus mai ilustrează relaţiile dintre aceste clase. Avem trei componente importante într-o diagramă de clase:

- **Clasa**. Este reprezentată sub forma unui dreptunghi, ce conţine în interior numele clasei şi adiţional pot fi prezentate atributele şi metodele clasei.
- **Interfaţa**. Este reprezentată sub forma unui cerc şi numele acesteia deasupra cercului.
- **Relaţia**. Este reprezentată sub forma unei drepte orientate, ce leagă doua elemente. Există mai multe tipuri de relaţii, acestea sunt amintite mai jos.

Tipuri de relaţii prezente între clasele unei diagrame:

- **Asocierea(Association)** Este reprezentată printr-o linie şi reprezintă relaţia dintre clase. Este de mai multe tipuri, unidirecţională şi bidirecţională. Cea bidirecţională este relaţia de bază(default) dintre doua clase. Cea unidirecţionaă este mai puţin întâlnită şi face referire la faptul că o clasă interacţionează cu cea de la capătul sageţii.

- **Agregarea(Aggregation)** Este reprezentată printr-o linie cu un romb gol la un capăt. Este mult mai specifică decât asocierea şi face referire la faptul că o clasă face parte din alta.(o clasă conţine o alta). Ciclul de viaţă a sub-componentelor nu este legat de cel al componentei pricipale.
- **Compoziţia(Composition)** ă cu cea de la capătul sageţii.
- **Agregarea(Aggregation)** Este reprezentată printr-o linie cu un romb plin la un capăt. Este tot o formă de agregare dar ciclul de viaţă al sub-componentelor este strâns legat de cel al componentei principale.
- **Moştenirea(Inheritence)** Este reprezentată printr-o linie cu un triunghi gol la un capăt. Reprezintă relatia de copil-parintre unde o clasă moşteneşte atributele şi metodele părintelui.

În continuare se va prezenta diagrama de clase aferentă sistemului implementat, Figura 5.8

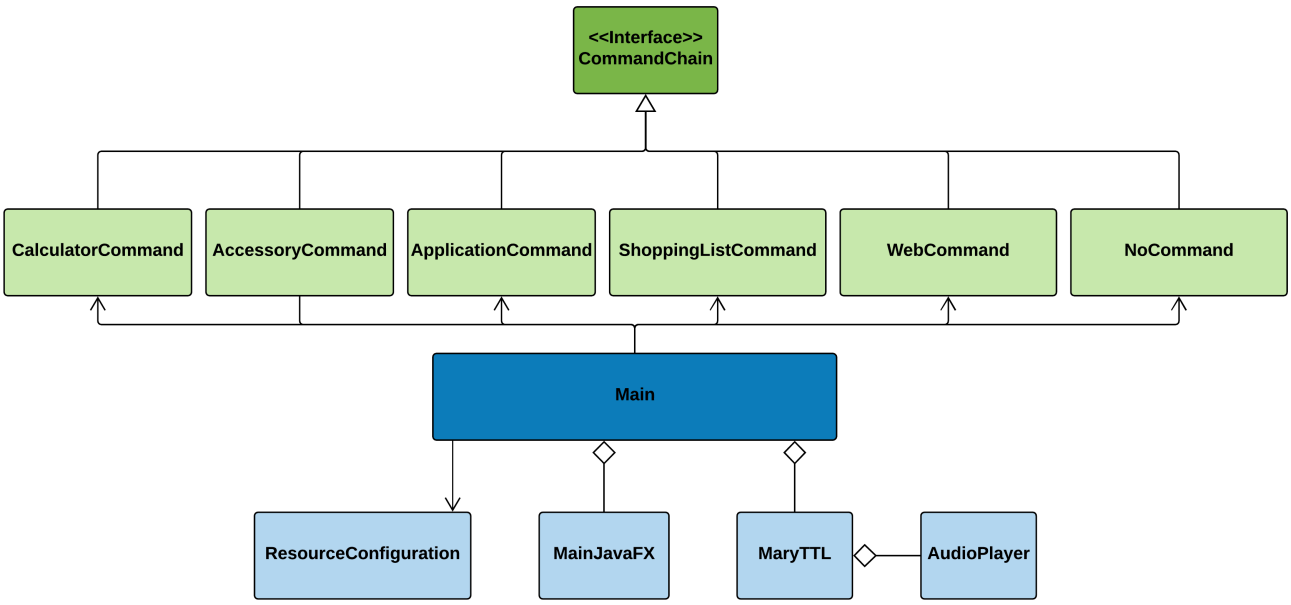

Figura 5.8: Diagrama de Clase

Capitolul 6

Testare şi Validare

Conform [26], testarea software reprezintă o investigaţie empirică realizată cu scopul de a oferi părţilor interesate informaţii referitoare la calitatea produsului sau serviciului supus testării, luând în consideraţie contextul operaţional în care acesta din urma va fi folosit.

Astfel "Testarea e procesul prin care se execută un program cu intenţia de a găsi erori"(Myers). "Testarea este utilizată pentru a semnala prezenţa defectelor unui program, fară a garanta absenţa lor"(Dijkstra)

Având în vedere faptul că aplicaţia construită este un asistent virtual, acţiunile sale duc la anumite răspunsuri vizuale astfel testarea se va face manual pentru a verifica că dupa fiecare acţiune se oferă răspunsul aşteptat. Pentru a testa aplicaţia complet, tester-ul poate construi un checklist bazând-use pe lista de funcţionalităţi a aplicaţiei şi să verifice toate acţiunile. Un exemplu de checklist este ilustrat în figura 6.1

Pe lângă verificarea acţiunilor şi anume funcţionarea acestora, s-a verificat şi timpul necesar acestor acţiuni astfel încât utilizatorul sa nu aştepte prea mult. Pentru cronometrare se poate folosi un cronometru in cod (un stopwatch) sau se poate folosi un cronometru fizic.

În continuare au fost verificate raspunsurile oferite la cererile utilizatorilor. Mai exact câte cereri au fost întelese si executate corect şi numărul cererilor care nu au fost înţelese sau întelese şi executate greşit. Avem trei cazuri ce pot sa apară:

- Comanda recunoscuta şi executată corect (**CREC**). Prin aceasta se întelege faptul ca cererea utilizatorului a fost recunoscută ca fiind o comandă existentă şi că sa executată exact ce dorea utilizatorul.
- Comanda recunoscuta şi executată gresit (**CREG**). Prin aceasta se întelege faptul ca cererea utilizatorului a fost recunoscută ca fiind o comandă existentă şi că sa executat o comandă dar nu a fost cea dorita de utilizator.
- Comanda nu este recunoscuta(**CNR**). Prin aceasta se întelege faptul ca cererea utilizatorului nu a fost recunoscută ca fiind o comandă existentă şi astfel utilizatorul trebuie sa repete cererea.

#	Test type	Checked Item	★ Desktop
1	FUNCT	Checked Item	
2	FUNCT	Deschidere Notepad	
3	FUNCT	Deschidere Google	
5	FUNCT	Verificare Vreme	
43	FUNCT	Trimitere Email	

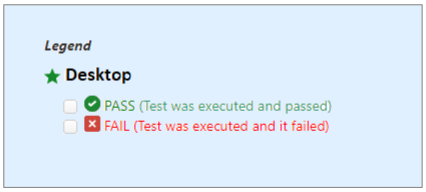

Figura 6.1: Exemplu CheckList

În urma testării se doreşte maximizarea variabilei CREC şi minimizarea variabilelor CREG şi CNR. Astfel în tabelul 6.1 se poate observa performanţa obţinută de fiecare comandă în parte pentru 10 incercări. În funcţie de acestea au avut loc modificari ale comenzilor mai exact a modului de execuţie şi recunoaştere.

6.1 Experimentare

Pe parcursul dezvoltării acestui sistem au fost încercate diferite metode de realizare a anumitor comenzi şi de prezentare a interfeţei, până când sa ajuns la o metă mai optimă.

Anumite metode de executîe şi recunoaştere a comenzilor au fost încercate pentru a putea oferi experienţă mai plăcută utilizatorului. De exemplu comenzile pentru shopping list, deschidere/închidere tab-uri, căutare, au fost schimbate într-o execuţie în 2 paşi pentru a creşte utilitatea şi perfomantă comenzii.

Schimbările legate de interfaţă sunt majore prin faptul că iniţial sa folosit Java Swing, iar interfaţa nu era atât de user friendly. Astfel folosind Java FX interfaţa este mult mai plăcută şi este o tehnologie actuală fiind uşor de schimbat.

Tabelul 6.1: Analiza performanţei pentru recunoaşterea comenzilor

Comanda	Incercări	CREC	CREG	CNR	Procent
Transcriber	10	10	0	0	100%
Calculator	10	5	1	4	50%
Notepad	10	9	0	1	90%
Paint	10	8	1	1	80%
Notes	10	10	0	0	100%
Word	10	7	0	3	70%
Excel	10	8	0	2	80%
Player	10	9	0	1	90%
Alarm	10	9	0	1	90%
Plus	10	6	2	2	60%
Minus	10	6	3	1	60 %
Multiply	10	4	4	2	40%
Divide	10	4	3	3	40%
Add *item* to shopping list	10	0	2	8	0%
Remove *item* from shopping list	10	0	2	8	0%
Display shopping list	10	0	2	8	0%
Shopping(List, 3 steps)	10	8	1	1	80%
Google	10	9	1	0	90%
Gmail	10	9	1	0	90%
Maps	10	9	0	1	90%
Facebook	10	7	0	3	70%
Youtube	10	10	0	0	100%
University	10	7	0	3	70%
Weather	10	9	0	1	90%
Radio	10	10	0	0	100%
Wikipedia	10	7	2	1	70%
New Tab	10	5	1	4	50%
New(2 steps -> Tab)	10	9	0	1	90%
Close Tab	10	3	2	5	30%
Close(2 steps -> Tab)	10	9	0	1	90%
Sign In	10	3	2	5	30%
Sign Out	10	3	1	6	30%
Sign(2 steps)	10	9	0	1	90%
Send(Email)	10	8	0	2	80%
Route	10	6	2	2	60%
Search Local	10	5	1	4	50%
Search Wikipedia	10	3	2	5	30%
Search(2 steps)	10	9	0	1	90%

Capitolul 7

Manual de Instalare şi Utilizare

În cadrul acestui capitol sunt detaliate resursele software şi hardware necesare pentru instalarea şi rularea aplicaţiei, precum şi o descriere pas cu pas a procesului de instalare.

7.1 Manual de Instalare

În continuare sunt descrise resursele principale de care are nevoie sistemul.

Creinţe Hardware

Pentru a utiliza aplicaţia cu uşurinţa, în condiţii optime şi pentru a oferi o experienţă placută utilizatorului, sistemul trebuie sa conţina:

- Minim 4GB memorie RAM
- Minim 1GB spatiu pe disk pentru aplicaţie
- Un microfon funcţional
- Placă de sunet

Creinţe Software

Sistemul de Operare

Având în vedere ca aplicaţia a fost construita pentru sistemul de operare Windows, e nevoie ca sistemul pe care va fi instalat sa fie tot Windows de la versiunea 7 în sus. Acesta poate fi achiziţionat de pe pagina oficială a celor de la Microsoft: Windows 10 Home

JAVA

Sistemul trebuie să conţina JAVA SE Development Kit 8(sau mai mare):

- JDK 1.8.0

- JRE 1.8.0

Pentru a instala acest kit se accesează pagina oficială Oracle de unde se poate descarca varianta dorita, pentru Windows(64/32 bits): Oracle Page to Download Java

IntelliJ IDEA
Pentru rularea aplicaţiei se poate folosi IDE de la JetBrains, IntelliJ IDEA, acesta fiind unul dintre cele mai bune IDE-uri pentru programarea in JAVA. Se descarcă mediul de programare de pe site-ul oficial: Download IntelliJ IDEA

GIT
Integrarea GIT in sistem. Acesta poate fi download-at de pe site-ul oficial unde sunt explicaţi şi paşii necesari pentru instalare: Download GIT.
Pentru a descărca aplicaţia pe computer se urmează paşii:

- Crearea unui folder gol în spaţiul de lucru
- Click dreapta în folderul gol şi se selectează Git Bash
- Se ruleaza comenzile:
  - git init
  - git clone https://Sele7@bitbucket.org/Sele7/thesis.git

Gradle
Aplicaţia foloseşte Gradle pentru anumite dependinţe astfel e nevoie de acesta. Se poate descarca de pe site-ul oficial: Download Gradle

7.2 Manual de Utilizare

Daca au fost respectaţii paşii din manualul de instalare atunci utilizarea aplicaţiei este foarte simplă şi intuitivă. În momentul accesării aplicaţiei ne este oferită o interfaţa utilizator, ilustrată in figura 7.1. Aceasta prezintă două butoane şi un câmp. După cum se poate observa, butonul din stânga este pentru a porni recunoaşterea vocală iar cel din dreapta pentru a o opri. Recunoaşterea poate fi oprită şi pornită de câte ori doreşte utilizatorul(conceptul de Toggle-to-Talk). Câmpul("Write Command") este pentru acel moment când utilizatorul nu poate vorbi sau comanda nu este înţeleasă şi astfel se pot scrie comenzi şi acestea vor fi executate la fel ca şi cele vocale.

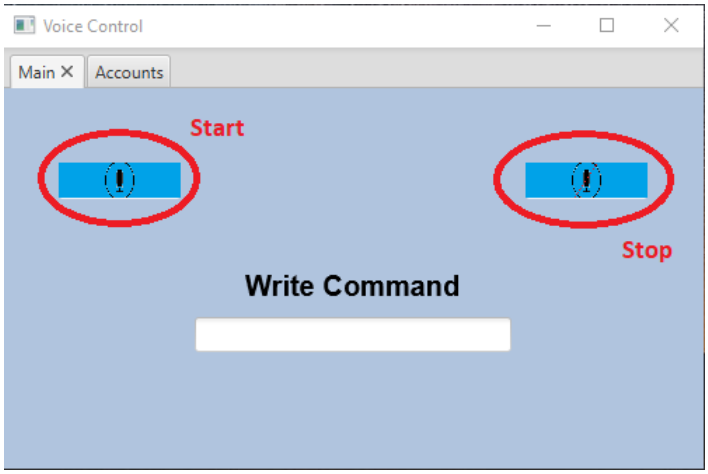

Figura 7.1: Interfaţa Principală

Astfel dupa rularea aplicaţiei din IDE, primul pas este acela de a porni recunoaşterea vocală prin apăsarea butonului de Start din Figura 7.1. Dupa acest pas, aplicaţia ne va comunica un mesaj pentru a ne atenţiona că recunoaşterea vocală este pronită şi că aşteaptă o comandă(Toate comenzile posibile sunt în Tabelul 6.1). În urma rostirii unei comenzi, asistentul încearcă identificarea comenzii, daca acesta este recunoscută atunci este comunicat un mesaj de executare a comenzii şi acţiunea este efectuată. Dacă comanda nu este recunoscută atunci asistentul comunică un mesaj şi este se cere reintroducerea unei comenzi. În Figurile următoare sunt ilustraţi ca şi exemplu paşii necesari pentru deschiderea aplicaţiei Notepad.

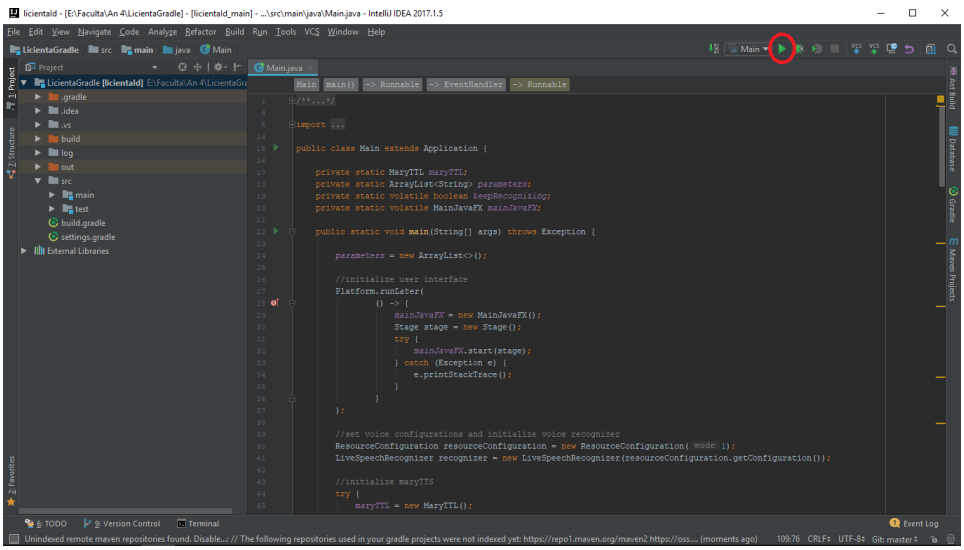

Figura 7.2: Exemplu Rulare Notepad Pas 1

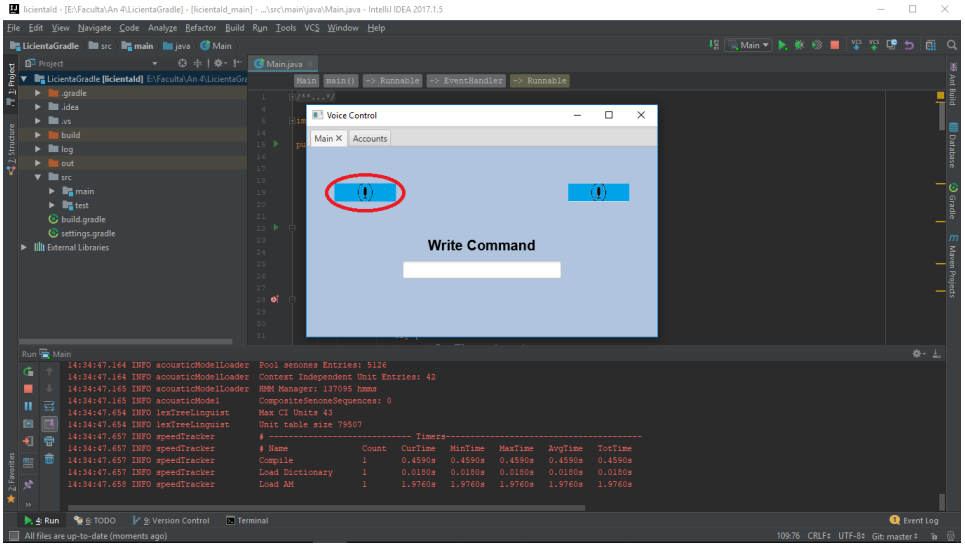

Figura 7.3: Exemplu Rulare Notepad Pas 2

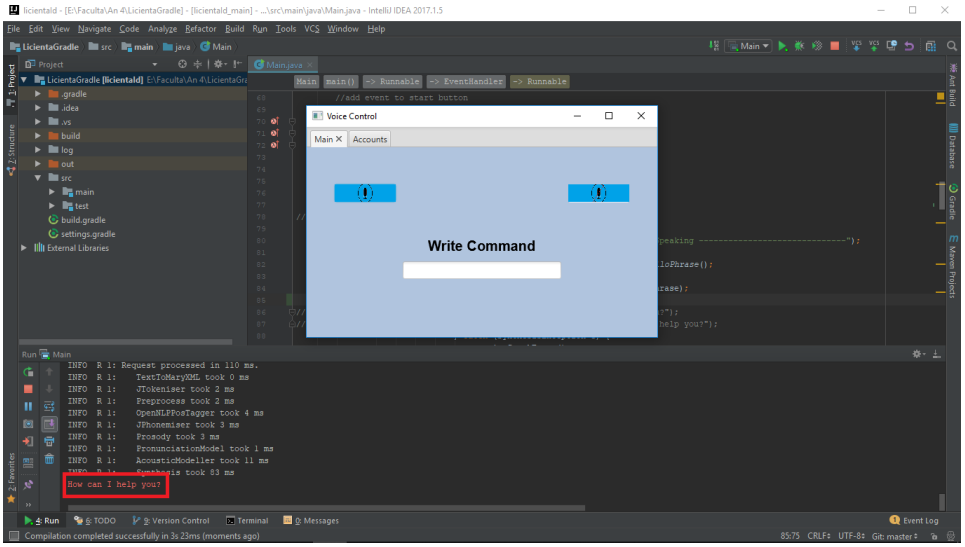

Figura 7.4: Exemplu Rulare Notepad Pas 3

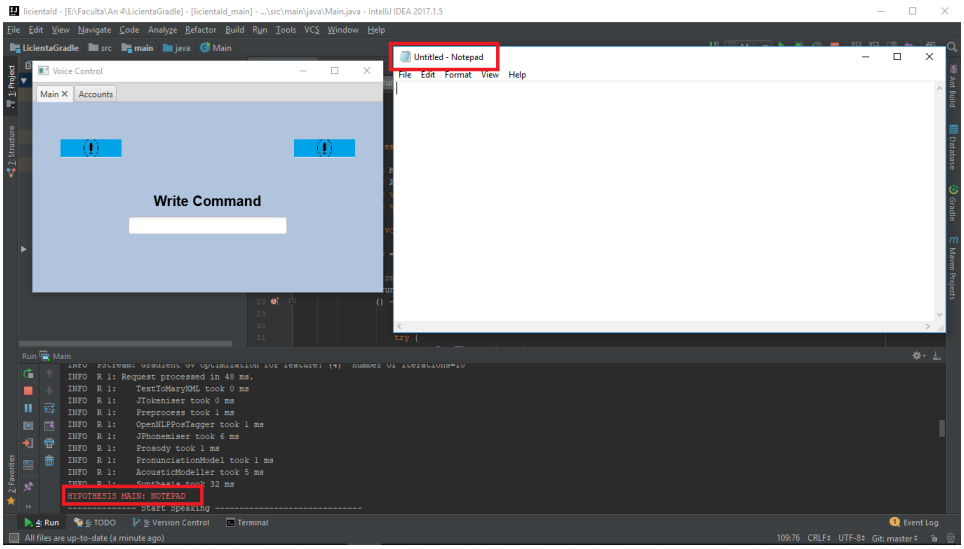

Figura 7.5: Exemplu Rulare Notepad Pas 4

Pe lângă interfaţa principală mai exista una, acesta fiind numită Accounts ilustrata in figura 7.6. Aici sunt trecute conturile şi parolele pentru logarea automată. Momentam aplicaţia suporta logarea automata pe Facebook şi Gmail. Acestea nu sunt cerute vocal din cauze de siguranţă. Aplicaţia introduce automat date de test, acestea trebuiesc schimbate înainte de a rula recunoaşterea vocală pentru a se putea executa logarea.

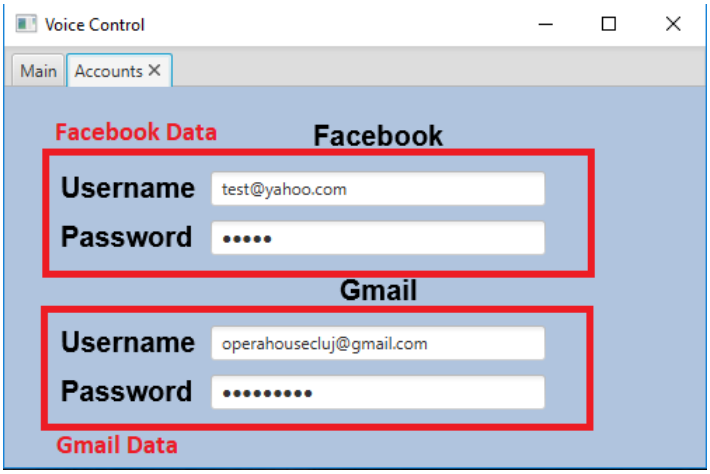

Figura 7.6: Interfaţa Accounts

Manualul de utilizare este complet iar utilizarea aplicaţiei fiind foarte uşoară, acesta fiind unul dintre obiectivele impuse în dezvoltarea acestei aplicaţii.

Capitolul 8

Concluzii

În acest capitol sunt menţionate concluziile trase în urma dezvoltarii acestui asistent personal virtual. Sunt menţionate contribuţiile aduse sitemului şi realizările, o analiză a sistemului, precum şi posibilitaţiile de dezvoltare ulterioră.

8.1 Contribuţii şi Realizări

Pentru realizarea acestui sistem, iniţial a avut loc o perioada de documentare şi explorare. Au fost analizare diferite sisteme similare şi s-au extras funcţionalităţille importante din acestea. Pe lânga acestea au mai fost adăugate anuminte funcţionalităţi care au părut folositoare. Dupa culegerea de informaţii s-a realizat planul de lucru în care task-urile au fost grupate in funcţie de funcţionalităţile asemanatoare şi realizate împreună.

Principala realizare este dezvoltarea acestui sistem, fiind un sistem uşor de folosit şi realizat cu librării şi tehnici în trend. Acesta vine în ajutorul utilizatorilor oferindule suport in anumite task-uri ce duc la salvarea timpului şi creşterea productivitaţii(i.e deschiderea aplicaţiilor).

Contribuţiile aduse sistemului pe lânga adăugarea de funcţionalităţi noi sunt realizarea unui sistem folosind doar componente open-source astfel aplicaţia fiind free şi deschisa publicului pentru dezvoltare.

În final sistemul implementat respectă atât cerinţele funcţionale cât şi cele nontionale, descrise în capitolele anterioare. Principalele caracteristici ale sistemului sunt:

- Deschidera anumitor aplicaţii
- Navigarea pe internet
- Realizarea operaţiunilor de Login/Logout pe anumite pagini
- Trimiterea unui email folosind Gmail
- Adăugarea sau ştergerea unor produse într-o listă de cumpărături şi enumerarea celor existente în listă.

În urma analizei sistemului cu sistemele similare se poate spune ca sistemul dezvoltat este unul relativ modern, ce se ridica la standardele actuale open-source. Acesta a fost dezvoltat folosind ultimele tehnologii , open-source, iar structura este bine gândita folosind design patterns.

Având în vedere cele menţionate, sistemul aduce următoarele beneficii utilizatorilor:

- Un sistem open-source disponibil free of charge
- Folosirea minima de resurse
- Economisirea timpului
- Creşterea productivităţii
- Interfaţa user–friendly

Aplicaţia a fost dezvoltată cu gândul de a oferi o experienţa cât mai bună utilizatorilor, fiind uşor de folosit şi de înteles. Iar pentru cei ce doresc sa ajute la dezovltarea acesteia uşor de modificat şi îmbunătăţit.

8.2 Dezvoltări ulterioare

Având în vedere constrângerile de timp acest sitem nu a putut fi implementat la capacitatea maxima astfel pot fi adăugate imbunătăţiri. Aceste îmbunătăţiri pot fi:

- Portarea aplicaţiei pe dispozitivele mobile. Având în vedere faptul că majoritatea smartphone-urilor prezintă un mecanism de recunoaştere vocală implementat, este nevoie doar de folosirea acestuia pentru recunoaşterea şi execuţia comenzilor.
- Recunoaştere vocală pentru diferite limbi. Momentan librăria Sphynx4 suportă mai multe limbi printre care Germană, Spaniolă, Franceză, etc. Pentru a schimba limba folosită trebuie schimbate modelul acustic, dicţionarul şi modelul de limbă. Pentru a recunoaşte o limbă pentru care nu există aceste modele încă, se poate folosi SphinxTrain pentru a creea modelul acustic iar pentru restul modelelor se foloseşte un corpus ce va fi transformat în modelele necesare.
- Îmbunătaţirea procesului de recunoaştere vocală. Pentru a creşte acurateţea cei de la Sphynx au creat un mini-tutorial[1]. Practic este nevoie de o bază de date de teste pentru a măsura acurateţea curentă. Mai departe se foloseşte SphinxTrain pentru a îmbunătaţi acurateţea. O alta modalitate poate fi schimbarea librăriei folosite pentru recunoaşterea vocală, folosind una mai performantă(probabil una ce trebuie cumpărată).

[1] https://cmusphinx.github.io/wiki/tutorialtuning/

- Adăugarea de funcţionalităţi noi (i.e. commerce shopping). Pot fi adăugate o multitudine de funcţionalităţi noi. Cea mai simplă metoda ar fi realizarea unui poll unde utilizatorii pot vota ce funcţionalităţi doresc să fie prezente în viitorul apropiat.

- Adăugarea protocoalelor de comunicare cu alte dispozitive (i.e. aspirator tip Romba)

- Adăugarea unui soft support ce le permite utilizatorilor sa îşi creeze propriile comenzi. În prezent, aplicaţia se rulează dintr-un IDE şi nu dintr-un .exe, deci putem spune ca utilizatorii au acces la cod şi il pot modifica cum doresc. Dar în viitor se doreşte pornirea aplicaţiei doar dintr-un .exe şi astfel este nevoie existenţa unui mediu în care se pot adăuga funcţionalităţi noi. De exemplu un mediu de test în care se download-ează toată aplicaţia(codul sursă), se fac modificările dorite şi utilizatorul poate să genereze .exe-ul dorit.

- Introducerea unui Sistem Expert pentru a o comunicare mai realista intre utilizator si sistem(a.k.a Chat Bot). Pentru acest deziderat este nevoie de dezvoltarea unui sistem expert, învăţarea acestuia şi introducerea în aplicaţia curentă. Se poate folosi Jess [2] pentru programarea sistemului expert, având în vedere faptul că Jess poate fi uşor introdus în aplicaţiile Java, folosind algoritmul Rete[3].

[2] este un tool folosit pentru dezvoltarea de sisteme expert,scris in Java. Este bazat pe fapte şi reguli.
[3]http://www.jessrules.com/docs/71/rete.html

Bibliografie

[1] "An introduction to intelligent personal assistants.' [Online]. Available: https://chatbotsmagazine.com/an-introduction-to-intelligent-personal-assistants-37c07aa7c5ab

[2] "The future of ai: The voice of the enterprise.' [Online]. Available: https://tech.economictimes.indiatimes.com/news/corporate/the-future-of-ai-the-voice-of-the-enterprise/63315418

[3] "How voice-based ai improves the digital workplace.' [Online]. Available: https://www.cmswire.com/digital-workplace/how-voice-based-ai-improves-the-digital-workplace/

[4] "8 ways artificial intelligence is going to change the way you live, work and play in 2018.' [Online]. Available: https://www.cnbc.com/2018/01/05/how-artificial-intelligence-will-affect-your-life-and-work-in-2018.html

[5] "Inginerie software.' [Online]. Available: http://web.info.uvt.ro/~cristiana.dragoescu/ANUl%20II/ISw/

[6] "Virtual personal assistant.' [Online]. Available: https://searchcrm.techtarget.com/definition/virtual-assistant

[7] "Virtual personal assistants (vpa) and smart advisors: Autonomous agent and smart machine technology and the market for ambient user experience.' [Online]. Available: www.prnewswire.com/news-releases/virtual-personal-assistants-vpa-and-smart-advisors-autonomous-agent-and-smart-machine-technology-and-the-market-for-ambient-user-experience-300245994.html

[8] "Virtual assistant.' [Online]. Available: https://en.wikipedia.org/wiki/Virtual_assistant

[9] "Conversational commerce.' [Online]. Available: https://www.shopify.com/encyclopedia/conversational-commerce

[10] "Voice assistant timeline.' [Online]. Available: https://www.voicebot.ai/2017/07/14/timeline-voice-assistants-short-history-voice-revolution/

[11] J. Abbasi, M. Hussain, and S. Ahmed, "An implementation of speech recognition for desktop application,' Masterş thesis, Mehran University of Engineering & Technology, Jamshoro, 2010.

[12] "Speech recognition.' [Online]. Available: http://www.abilityhub.com/speech/speech-description.htm

[13] L. R. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. PTR Prentice Hall Englewood Cliffs, 1993, vol. 14.

[14] B.-H. Juang and L. R. Rabiner, "Automatic speech recognition–a brief history of the technology development,' *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, vol. 1, p. 67, 2005.

[15] "Top ten speech recognition apis.' [Online]. Available: https://www.quora.com/What-are-the-top-ten-speech-recognition-APIs

[16] "Siri rising: The inside story of siris origins and why she could overshadow the iphone.' [Online]. Available: https://www.huffingtonpost.com/2013/01/22/siri-do-engine-apple-iphone_n_2499165.html

[17] L. Carvajal, L. Quesada, G. López, and J. A. Brenes, "Developing a proxy service to bring naturality to amazons personal assistant alexa,' in *International Conference on Applied Human Factors and Ergonomics*. Springer, 2017, pp. 260–270.

[18] P. Zubairm, H. Bhat, and T. Lone, "Cortana-intelligent personal digital assistant: A review,' *International Journal of Advanced Research in Computer Science*, vol. 8, no. 7, 2017.

[19] P. Lamere, P. Kwok, E. Gouvea, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf, "The cmu sphinx-4 speech recognition system,' in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong*, vol. 1, 2003, pp. 2–5.

[20] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel, "Sphinx-4: A flexible open source framework for speech recognition,' 2004.

[21] "Mary text to speech.' [Online]. Available: http://mary.dfki.de/documentation/history.html

[22] M. Schröder and J. Trouvain, "The german text-to-speech synthesis system mary: A tool for research, development and teaching,' *4th ISCA Workshop on Speech Synthesis*, 2001.

[23] T. Dutoit, *An introduction to text-to-speech synthesis*. Springer Science & Business Media, 1997, vol. 3.

[24] "Javafx.' [Online]. Available: https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

[25] "Why build your java projects with gradle rather than ant or maven?' [Online]. Available: http://www.drdobbs.com/jvm/why-build-your-java-projects-with-gradle/240168608

[26] C. Kaner, "Quality assurance institute worldwide annual software testing conference,' *Exploratory Testing*, 2006.

Anexa A

Figuri

Capitol	Figuri
Capitolul 1 Introducere	–
Capitolul 2 Obiectivele Proiectului	–
Capitolul 3 Studiu Bibliograc	Figura 3.1: Evolutia Asistentilor Virtuali Figura 3.2: Componentele procesului de recunoaştere vocală Figura 3.3: Arhitectura Siri Figura 3.4: Fluxul execuţiei unei comenzi pentru Alexa Figura 3.5: Cortana Intelligence Figura 3.6: Fluxul execuţiei unei comenzi pentruGoogle Home
Capitolul 4 Analiză şi Fundamentare Teoretică	Figura 4.1: Arhitectura Conceptuală a Sistemului Figura 4.2: Diagramă Use Case Figura 4.3: Diagramă de Flux pentru Login Figura 4.4: Diagramă de Flux pentru Trimitere Email Figura 4.5: Diagramă de Flux pentru eschidere Tab Nou Figura 4.6: Sphinx-4 Arhitectura Sistemului Figura 4.7: Sphinx-4 Front End Figura 4.8: Arhitectura Sistemului MaryTTS Figura 4.9: Gradle Figura 4.10: Git Commits
Capitolul 5 Proiectare de Detaliu şi Implementare	Figura 5.1: Arhitectura Sistemului Figura 5.2: Diagrama de Secvenţă Figura 5.3: Interfaţa Principală Figura 5.4: Interfaţa Accounts Figura 5.5a: Exemplu de Model de dicţionar Figura 5.5b: Exemplu de Model de limbă Figura 5.6: Chain Of Responsibility Diagram Figura 5.7: Chain Of Responsibility Flow Diagram Figura 5.8: Diagrama de Clase
Capitolul 6 Testare şi Validare	Figura 6.1: Exemplu CheckList
Capitolul 7 Manual de Instalare şi Utilizare	Figura 7.1: Interfaţa Principală Figura 7.2: Exemplu Rulare Notepad Pas 1 Figura 7.3: Exemplu Rulare Notepad Pas 2 Figura 7.4: Exemplu Rulare Notepad Pas 3 Figura 7.5: Exemplu Rulare Notepad Pas 4 Figura 7.6: Interfaţa Accounts
Capitolul 8 Concluzii	–

Anexa B

Tabele

Capitol	Tabele
Capitolul 1 Introducere	–
Capitolul 2 Obiectivele Proiectului	–
Capitolul 3 Studiu Bibliograc	Tabelul 3.1: Analiză comparativă a sistemelor similare caracteristici generale Tabelul 3.2: Analiză comparativă a sistemelor similare caracteristici funcţionale
Capitolul 4 Analiză şi Fundamentare Teoretică	–
Capitolul 5 Proiectare de Detaliu şi Implementare	–
Capitolul 6 Testare şi Validare	Tabelul 6.1: Analiza performanţei pentru recunoaşterea comenzilor
Capitolul 7 Manual de Instalare şi Utilizare	–
Capitolul 8 Concluzii	–

Anexa C

Glosar de Termeni

Framework = un software care furnizează funcţionalităţi generice ce pot fi schimbate selectiv prin cod suplimentar scris de către utilizatori.
Recunoaşterea vocală = procesul de convertire a cuvintelor rostite în format digital. Acestea pot fi folosite ulterior pentru diferite acţiuni.[1]
Asistent Virtual = un software ce foloseşte inteligienţa artificială şi recunoaştea vocală pentru a veni în ajutorul utilizatorului în anumite task-uri.
Comerţ conversaţional = se referă la intersecţia aplicaţiilor de mesagerie şi a cumpărăturilor. Este trendul de a interacţiona cu afacerile prin intermediul aplicaţiilor de mesagerie precum WhatsApp, Wechat sau prin tehnologia vocală cum ar fi produsul Echo al companiei Amazon.
Application Programming Interface = un set de sub–programe,protocoale şi unelte pentru programarea de aplicaţii
open-source = o practică ce le permite utilizatorilor accesul liber asupra procesului de dezvoltare.

[1] https://ro.wikipedia.org

Anexa D

Glosar de Acronime

ASR = Automatic Speech Recognition
NLP = Natural Language Processing
DNN = Deep Neural Network
JDK = Java SE Development Kit
JRE = Java Runtime Environment
IDE = Integrated development environment
API = Application Programming Interface