

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**  
**FACULTATEA DE INFORMATICĂ**



**LUCRARE DE LICENȚĂ**

# **Ajutor vocal de navigare online**

**Rareș Arvinte**

**Sesiunea:** iulie 2018

**Coordonator științific**  
**Profesor Dr. Dan Cristea**

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**  
**FACULTATEA DE INFORMATICĂ**

# **Ajutor vocal de navigare online**

**Rareș Arvinte**

**Sesiunea: iulie 2018**

**Coordonator științific**  
**Profesor Dr. Dan Cristea**

Avizat,  
Îndrumător Lucrare de Licență  
Titlul, Numele și prenumele

Data \_\_\_\_\_ Semnătura  
\_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemnatul(a) .....  
domiciliul în .....  
născut(ă) la data de ....., identificat prin CNP .....,  
absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de  
..... specializarea ....., promoția  
....., declar pe propria răspundere, cunoscând consecințele falsului în  
declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale  
nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul:

\_\_\_\_\_ elaborată sub îndrumarea dl. / d-na  
\_\_\_\_\_, pe care urmează să o susțină în fața  
comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată  
prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la  
introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări  
științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei  
lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie  
răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am  
întreprins-o.

Data azi, .....

Semnătură student .....

## DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul *Ajutor vocal de navigare online*, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

*Absolvent Rareș Arvinte*

---

(semnătura în original)

## Cuprins

1. Introducere .....	pag. 5
1.1.Motivația alegerii temei .....	pag. 5
1.2.Obiectivele generale ale lucrării .....	pag. 6
1.3.Propunere de metodologie .....	pag. 6
1.4.Structura lucrării .....	pag. 7
2. Descrierea aplicației .....	pag. 7
2.1.Descriere funcțională .....	pag. 7
2.2.Configurarea aplicației .....	pag. 8
2.3. Caracteristica de rulare în <i>background</i> .....	pag. 8
2.4. Căutarea în online .....	pag. 9
2.5. Căutarea cuvintelor în dicționar .....	pag. 9
2.6. Acces Youtube .....	pag. 10
2.7. Acces E-mail .....	pag. 10
3. Metode utilizate .....	pag. 11
3.1.Biblioteca Pyttsx3 .....	pag. 11
3.2.Biblioteca de recunoaștere vocală .....	pag. 12
3.3.Biblioteca WebBrowser .....	pag. 12
3.4.Biblioteca PyDictionary .....	pag. 13
3.5.Biblioteca urllib .....	pag. 13
3.5.1. urllib request .....	pag. 13
3.5.2. urllib error .....	pag. 14
3.5.3. urllib parse .....	pag. 14
3.6.Biblioteca imaplib .....	pag. 15
4. Concluzii .....	pag. 16
4.1.Contribuții originale .....	pag. 16
4.2.Dezvoltări ulterioare .....	pag. 17
5. Bibliografie .....	pag. 18
6. Anexe .....	pag. 19

# Introducere

## 1.1 Motivația alegerii temei

În lume există persoane cu deficiențe de vedere, acestea fiind cauzate de diferiți factori neurologici sau fiziologici. În secolul 21, odată cu dezvoltarea rapidă a internetului, a apărut nevoia creării condițiilor necesare pentru ca aceste persoane să poată folosi noile tehnologii care chiar pot să îi ajute să se integreze mai ușor în societate și de asemenea să le faciliteze viața.

Intenționăm ca prin aplicația pe care o descriem în această lucrare de licență să oferim persoanelor cu probleme de vedere o șansă de a avea acces la noile tehnologii, mai exact cele din mediul online fără a avea nevoie de o tastatură specială în alfabetul Braille. Utilizatorii vor avea prin intermediul vocii acces atât la anumite facilități oferite de *Google* cât și la fișiere din calculatorul personal. Utilizatorul va putea accesa *e-mailul*, asculta muzică, va putea primi definiții pentru cuvinte din dicționar, va avea acces la *Google Drive* și multe altele. Astfel, un utilizator va transmite o comandă vocală către aplicație iar aplicația va accesa diferite module *Google* integrate sistemului de operare în vederea obținerii informațiilor dorite.

Aplicația poate fi dezvoltată și în alte domenii care necesita control vocal: aviație, medicină, transport rutier, etc

Ceea ce m-a motivat să aleg această temă este în primul rând pasiunea mea pentru inteligența artificială, mereu mi-au plăcut roboții și ideea de a-ți putea dezvolta propria mașină care să te ajute sau să îți fie ca un prieten. În esență omul este o ființă socială care are nevoie în permanență de cineva cu care să comunice sau să practice anumite activități în prezența ori folosindu-se de cineva sau ceva.

## 1.2 Obiectivele generale ale lucrării

În această lucrare vom prezenta aplicația și vom explica aplicabilitatea acesteia în societate și în viitoare proiecte. De asemenea vom urmări expunerea tehnologiilor folosite în crearea acesteia și posibile idei de dezvoltare în viitor.

Prin descrierea aplicației vom urmări detalierea funcționalităților acesteia, tehnologiile folosite din mediul *open-source* și limbajul folosit, acesta fiind *Python*. Se dorește de asemenea să se aducă la cunoștință cititorilor acestei lucrări existența unor tehnologii utilizate pentru prelucrare vocală care vor putea fi folosite în viitor în dezvoltarea unor aplicații cu aceeași tematică.

## 1.3 Propunere de metodologie

În crearea acestei aplicații s-a folosit limbajul de programare *Python* și câteva dintre bibliotecile aferente acestuia precum: *Pytsx3*, *SpeechRecognition*, *WebBrowser*, *PyDictionary*, *UrlLib*, *ImapLib* etc. De asemenea, în urma documentației făcute pentru selectarea librăriilor necesare, le-am ales pe acelea compatibile cu ultima versiune de *Python* la momentul respectiv (*Python 3.6.4*).

Funcționalitatea de bază a proiectului este *Speech-to-Speech*, aceasta fiind oferită de cele două biblioteci: *SpeechRecognition* și *Pytsx3*, ce fac conversia voce-text respectiv text-voce. Alte funcționalități precum *web-surfing*, dicționar și acces la *e-mail* vor fi implementate cu ajutorul bibliotecilor *WebBrowser*, *PyDictionary*, *UrlLib*, *ImapLib* și *email* care împreună oferă și accesul la conținutul de pe *Youtube*.

## 1.4 Structura lucrării

Lucrarea este structurată în trei părți. Prima parte este descrierea aplicației din punct de vedere tehnic. Aici se regăsesc descrierile metodelor implementate. În următoarea parte sunt enumerate tehnologiile folosite și scopul lor în proiect. A treia parte conține concluzii cu privire la contribuția mea la acest proiect.

Lucrarea se finalizează cu bibliografia studiată și folosită pentru implementarea programului precum și anexele aferente proiectului.

## Descrierea aplicației

### 2.1 Descriere funcțională

Aplicația dezvoltată de mine reprezintă o modalitate prin care persoanele cu dizabilități de vedere pot interacționa în mediul online. Aplicația rulează în background neavând o interfață grafică, aceasta funcționând ca un majordom. La prima utilizare, *user*-ul este rugat să urmeze anumiți pași de configurare a aplicației. După aceste configurări aplicația este pregătită pentru a fi utilizată, *user*-ul având posibilitatea de a efectua următoarele acțiuni: acces la pagini web, dicționar, *e-mail*-uri sau *Youtube*.

Aplicația începe printr-o scurtă prezentare a posibilităților de utilizare și modul de folosire în diferite situații. Dacă utilizatorul dorește să cunoască opțiunile implementate în aplicație și modul de accesare al acestora, i se va comunica la deschiderea programului că prin rostirea cuvântului *meniu* i se va oferi un mini meniu de utilizare. (Pentru un exemplu de cod, a se consulta Anexa 1)

Fiecare serviciu poate fi apelat printr-un cuvânt cheie rostit de utilizator. În funcție de serviciul ales aplicația îl va îndruma pe utilizator către următorul pas.

În cazul în care utilizatorul dorește oprirea aplicației, acesta va folosi un cuvânt cheie (spre exemplu *quit*) în momentul în care aplicația este în *stand-by*.



## 2.2 Configurarea aplicației

În urma primei execuții a aplicației, utilizatorului i se solicită configurarea acesteia cu anumite date cu caracter personal. În primă instanță i se va solicita introducerea unui nume de utilizator pentru ca aplicația să îl poată recunoaște. Ulterior i se vor cere informații referitoare la accesarea contului *Google*, spre exemplu o adresă de *e-mail* și parolă, doar în cazul în care utilizatorul va dori să acceseze prin intermediul aplicației date cu caracter personal.

## 2.3 Caracteristica de rulare în *background*

Aplicația rulează în *background*, independent de alte aplicații simultan active pe desktop/laptop, ea îndeplinind acțiuni prin intermediul unor comenzi vocale date de utilizator. În cazul în care utilizatorul nu mai dorește să folosească aplicația, acesta o va putea opri prin comanda vocală *quit*. Am ales această abordare pentru a nu permite aplicației să interfereze cu alte conversații ale utilizatorului.

## 2.4 Căutarea în online

Aplicația are opțiunea de interogare a unui motor de căutare, opțiune ce poate fi selectată doar în cazul în care aplicația este într-o stare activă. Această opțiune poate fi selectată cu ajutorul cuvântului cheie *search*, utilizatorul fiind nevoit apoi să introducă datele căutării. Aplicația va accesa primul rezultat oferit de motorul de căutare și va oferi de pe respectiva pagină conținutul acesteia dacă este posibil. În eventualitatea în care conținutul nu a putut fi efectuat, vor fi accesate alte pagini până la preluarea acestuia sau până la o limită predefinită de căutare. (Pentru un exemplu de cod, a se consulta Anexa 2)

## 2.5 Căutarea cuvintelor în dicționar

Aplicația oferă posibilitatea de a căuta definiții într-un dicționar online, citind utilizatorului definiția oferită. Această opțiune poate fi selectată cu ajutorul cuvântului cheie *dex*, utilizatorul trebuind apoi să pronunțe cuvântul căutat. În cazul în care utilizatorul rostește un singur cuvânt, aplicația se va folosi de *PyDictionary* pentru a oferi utilizatorului definiția pentru cuvântul cerut. În cazul în care utilizatorul introduce mai mult de un cuvânt, aplicația va apela funcția de căutare în online pentru o definiție cât mai precisă. (Pentru un exemplu de cod, a se consulta Anexa 3)

## 2.6 Acces Youtube

Dat fiind faptul că oamenii cu dezabilități vizuale folosesc ca principal organ de simț auzul, vor fi încântați să utilizeze *YouTube*-ul direct din aplicație. În vederea soluționării acestei nevoi aplicația caută și redă videoclipuri de pe *YouTube* în concordanță cu comanda primită. În cazul în care videoclipul nu este cel căutat, utilizatorul are posibilitatea de a selecta următorul videoclip din rezultatul căutării.

Aplicația oferă posibilitatea de a reda un *video* de pe *Youtube* în *browser*-ul prestabilit. Această opțiune poate fi selectată cu ajutorul cuvântului cheie *Youtube*, utilizatorul trebuind apoi să pronunțe datele căutării. În urma introducerii parametrilor de căutare, aplicația reține rezultatele căutării într-un vector, redând primul rezultat utilizatorului. Utilizatorul se poate folosi de cuvinte cheie precum *next* sau *back* pentru a se întoarce în lista de rezultate din vector și a selecta pentru redare următorul video. (Pentru un exemplu de cod, a se consulta Anexa 4)

## 2.7 Acces E-mail

În urma configurărilor efectuate de utilizator, dacă acesta a optat pentru accesul la *e-mail*, aplicația îi poate oferi această facilități. În urma conectării la *gmail* aplicația îi va putea citi utilizatorului mesajele necitite. Dezvoltarea ulterioară a programului poate crea posibilitatea accesării pentru audiere, doar a mesajului dorit. După accesarea inițială a tuturor mesajelor necitite, utilizatorul va putea printr-o comandă vocală (ex: mesajul 1, mesajul 3) audierea mesajului dorit.

Aplicația oferă posibilitatea de a accesa *e-mail*-uri prin intermediul contului de *gmail*. Această opțiune poate fi selectată cu ajutorul cuvântului cheie *email*, aplicația oferind o listă de *e-mail*-uri necitite ce va fi citită utilizatorului. (Pentru un exemplu de cod, a se consulta Anexa 5)

## Metode Utilizate

În crearea acestui proiect s-a folosit limbajul *Python*. Am optat pentru acest limbaj deoarece este nou, codul este simplificat și este propice construirii de aplicații de dimensiuni reduse.

De asemenea, *Python* suportă programarea orientată obiect, imperativă, funcțională sau procedurală. Sistemul de tipizare este dinamic iar administrarea memoriei decurge automat prin intermediul unui serviciu „gunoier” (*garbage collector*). Un alt avantaj al limbajului este existența unei ample biblioteci standard de metode (Downey, Allen B., 2012).

### 3.1 Biblioteca Pyttsx3

Pentru vocea care răspunde cererilor utilizatorului s-a folosit biblioteca *Pyttsx3*. Această bibliotecă din *python* preia cuvintele ca *string*-uri și le pasează la *engine*-ul *pyttsx* care face conversie *text-to-speech*. Prin comanda *say* se trimite la ieșire *string*-ul, acesta fiind comunicat vocal. Când *Pyttsx3* primește o cerere se invocă funcția *init()* care primește o referință la o instanță a *engine*-ului. *Engine*-ul creează un obiect *DriverProxy* responsabil pentru conversia *text-to-speech*. Se poate specifica ce *driver* audio să se folosească în funcție de sistemul de operare. Dacă nu este specificat, se va încerca prin *default* să se aleagă cel mai potrivit *driver*. Momentan există trei *drivere*. Unul dintre acestea este *SAPI5* pentru *Windows*, altul este *NSSpeechSynthesizer* pentru *Mac OS* și unul compatibil pe orice platformă numit *eSpeak*. Biblioteca vine și cu posibilitatea de a customiza vocea de răspuns sau rata de răspuns (cuvinte pe minut). (Natesh, 2017)

### 3.2 Biblioteca de recunoaștere vocală

Biblioteca *Speech Recognition* a fost folosită pentru recunoaștere vocală. În momentul interacțiunii utilizatorului cu programul, acesta preia conținutul vocal și îl convertește într-un *string*. Pentru a folosi biblioteca este recomandat să avem acces la un microfon, *bind*-ul microfonului făcându-se cu ajutorul bibliotecii *PyAudio* (Pham, 2006). Astfel, se instanțiază un nou microfon cu comanda *Microphone()*, iar în cazul în care nu se știe ce fel de microfon se folosește sau avem mai multe microfoane, se invocă funcția *list\_microphone\_names()* (ce returnează o listă cu toate microfoanele conectate la dispozitiv) și programul va alege următorul microfon din listă până va găsi unul compatibil. Pentru a funcționa, aplicația are nevoie de un *recognizer*, ce se instanțiază prin funcția *Recognizer()*, acesta având rolul de a recunoaște sunetele. La această funcție se pot face o multitudine de setări pentru a captura cât mai bine și mai clar sunetul. În momentul rulării aplicației se instanțiază microfonul. Cât timp acesta captează *input* audio, *recognizer*-ul capturează sunetele folosind funcția *listen()*. Datele audio sunt pasate unei funcții *recognize\_google()* care convertește aceste date într-un format potrivit pentru a face request-uri la *google*. Biblioteca poate fi folosită și în scopul de a crea fișiere audio, care pot fi prelucrate mai târziu sau pentru înregistrare vocală. De asemenea, biblioteca, pe lângă faptul că face request-uri la *google*, oferă posibilitatea de a lucra cu *Microsoft Bing Speech API*, *Wit.ai API*, *Google cloud Speech API*, *Houndify API* sau *IBM Speech to Text API*. (Zhang, 2017)

### 3.3 Biblioteca WebBrowser

*WebBrowser* este o bibliotecă *python* folosită pentru deschiderea de pagini web. Funcția *open()* primește un *URL* ca date de intrare și, pe baza acestuia, accesează pagina respectivă. Această bibliotecă este utilă în momentul în care dorim să accesăm rapid o pagină web. Înainte de a folosi biblioteca este necesar să specificăm ce fel de *webbrowser* folosim, acest lucru realizându-se prin funcția *get()* căreia i se atribuie un *string* cu *PATH*-ul unde se află instalat *webbrowserul*. (Python Software Foundation, 2018)

### 3.4 Biblioteca PyDictionary

*PyDictionary* este o bibliotecă ce oferă acces la un dicționar. Aceasta conține funcții pentru a căuta și oferi definiții, sinonime, antonime sau traduce cuvinte în alte limbi. Avem funcția *meaning()* care returnează definiția unui cuvânt, funcția *synonym()* care returnează sinonimele unui cuvânt dat ca *input*, funcția *antonym()* care returnează antonimele unui cuvânt și funcția *translate()* care primește doi parametri, primul fiind cuvântul de tradus iar al doilea este limba în care să se traducă. Rezultatele sunt returnate în format *JSON* din bazele de date ce rulează pe servere de la *Red Hat*. Datele de *input* sunt transmise vocal de către utilizator și sunt apoi convertite în text cu ajutorul bibliotecii *SpeechRecognition*. Datele de *output* sunt transmise vocal utilizatorului prin intermediul bibliotecii *Pytsx3*. (Bora, 2014)

### 3.5 Biblioteca Urllib

*Urllib* este un pachet ce conține mai multe module ce lucrează cu *URL*-uri. (Python Software Foundation, 2018)

#### 3.5.1 Urllib request

Modulul de *request* pune la dispoziție o colecție de funcții și clase în scopul de a deschide *URL*-uri. Funcția *urlopen()* primește ca parametru un *string* și deschide resursa de la adresa specificată. De asemenea se pun la dispoziție funcții pentru gestionarea resursei în cauză. (Python Software Foundation, 2018)

### 3.5.2 Urllib error

Acest pachet este folosit pentru a detecta eventualele erori apărute în urma execuției unei funcții din *Urllib request*. Conține funcția *URLError()* care returnează motivul erorii sub forma unei instanțe din clasa *URLError*. (Python Software Foundation, 2018)

### 3.5.3 Urllib parse

Această bibliotecă se ocupă cu manipularea *string*-urilor date ca *input* sau a conținutului propriu-zis al unui *URL*. De asemenea, ajută la identificarea conținutului unui *URL*. Funcția *urlparse()* sparge *URL*-ul în fragmente și comunică detalii despre acestea cum ar fi schema adresei, *username* și parolă, dacă acestea au fost specificate în *URL* și multe altele. Această bibliotecă este utilă deoarece putem codifica și decodifica *URL*-uri ceea ce este necesar când se fac spre exemplu *request*-uri la *YouTube* sau când accesăm un *URL* și vrem să vedem ce semnifică codificarea sa. Acestea se realizează cu funcțiile *urlencode()* și *urldecode()*. (Python Software Foundation, 2018)

### 3.6 Biblioteca ImapLib

Scopul acestei biblioteci este de a realiza comunicarea cu adresa de *e-mail* a utilizatorului. Biblioteca *imaplib* vine cu trei clase: *IMAP4*, *IMAP4\_SSL* și *IMAP4\_stream*. Clasa *IMAP4* implementează protocolul *imap4* care creează o conexiune având ca date de intrare *host*-ul și *port*-ul (*port*-ul este opțional).

*IMAP4\_SSL* este asemănătoare cu *IMAP4*, doar că acesta realizează conexiunea peste un *socket* criptat *SSL*. La fel se furnizează numele *host*-ului și *port*-ul (dacă *port*-ul este omis se va conecta la *port*-ul standard *IMAP-over-SSL* 993), iar opțional se poate specifica o cheie privată sau un certificat pentru conexiune *SSL*, acestea asigurând securitatea transmiterii datelor.

*IMAP4\_stream*, la fel ca *IMAP4\_SSL*, este o subclasă a clasei *IMAP4* al cărui scop este de a crea o conexiune *stdin/stdout*.

După realizarea unei conexiuni funcția *login()*, primește ca *input* un nume de utilizator și o parolă și încearcă să ofere accesul clientului la *e-mail*-urile sale. Se recomandă folosirea funcției *login\_cram\_md5()* pentru a asigura securitatea datelor cu caracter personal. Biblioteca conține de asemenea o multitudine de funcții pentru selectarea, citirea, scrierea și trimiterea mesajelor. Mesajele pot fi selectate prin mai multe moduri, de la selectarea întregului *inbox* până la căutarea unui anumit cuvânt cheie. Dacă utilizatorul dorește să citească un mesaj, acesta va fi selectat cu un procedeu de selecție, apoi va fi redat vocal de către program. (Python Software Foundation, 2018)



## Concluzii

### 4.1. Contribuții originale

În urma cercetărilor efectuate în domeniul *speech recognition* am ajuns la concluzia că acestui tip de proiect i se poate acorda o atenție deosebită datorită utilității acestuia atât în industria IT cât și în toate celelalte domenii ale societății. În ciuda acestor aspecte pozitive, putem observa că nu există suficiente produse *open-source* legate de acest domeniu. Acest lucru m-a determinat să dezvolt un punct de plecare pentru viitoare proiecte *open-source*.

În mediul *online* se găsesc diferite biblioteci ce se ocupă spre exemplu de recunoașterea *input*-ului vocal, convertirea acestuia în *string*-uri și căutarea *web*. Contribuția mea constă în îmbinarea acestor elemente având ca scop efectuarea de *task*-uri în mediul *online*, ce fac redundantă folosirea tastaturii.

Ca prim pas, am început prin a implementa o metodă de căutare pe *google*, având ca *input* un *string* primit vocal. Ulterior am adăugat o funcție care accesează primul *link* și returnează conținutul de pe acea pagină. Am propus o abordare diferită și simplistă, demnă de luat în vedere în momentul realizării unui proiect cu aceeași tematică.

Evaluare funcționalității a relevat o rată de eroare datorită intonației și puterii cu care se transmit comenzile vocale dar și a zgomotului de fond care perturbă percepția corectă a sunetelor. Cred că pe viitor implementarea unor soluții tehnice referitoare la adaptarea programului la vocea utilizatorului (setări specifice pentru fiecare utilizator în funcție de tonalitatea vocii) va reduce rata de eroare și va face utilizarea mai prietenoasă.

## 4.2 Dezvoltări ulterioare

Pe viitor îmi propun să dezvolt noi facilități ce vor reduce interacțiunea fizică a utilizatorului cu mașina.

O idee ar fi implementarea unei opțiuni pentru adăugarea de evenimente sau notificarea utilizatorului despre viitoarele evenimente din calendarul *google*.

O altă funcționalitate pe care aş dori să o adaug este interacțiunea cu mediul *cloud*. La recunoaşterea anumitor cuvinte cheie, aplicația va încărca sau descărca fişiere din *google drive*.

Cât despre partea de *message sharing* îmi propun să implementez o funcție ce va permite scrierea şi trimiterea de mesaje.

Ca să trag o concluzie la ceea ce am explicat mai sus, aş spune că în viitor de pe urma dezvoltării acestei aplicații vor beneficia spre exemplu persoanele cu dizabilități de ordin fizic. Totodată, aplicația eficientizează interacțiunea om-calculator prin faptul că înlocuieşte timpul necesar tastării unei comenzi cu echivalentul exprimării sale orale.

Pe viitor ar fi interesant ca o mare majoritate a populației să folosească dispozitive cu interacțiune vocală, astfel reducând folosirea textului şi permițând utilizatorului să acorde o atenție sporită mediului înconjurător. (Voxtab, 2015)

În preocupările viitoare am în vedere studierea de biblioteci ale mediului *Python*, mai ales cele din mediul *open-source*, precum şi studierea posibilității de a oferi suport pentru limba română.

## Bibliografie

- Downey, Allen B. (May 2012). Think Python: How to Think Like a Computer Scientist (Version 1.6.6 ed.). ISBN 978-0-521-72596-5. Disponibil la <http://www.greenteapress.com/thinkpython/html/>
- Natesh M Bhat. (2017). Pyttsx3 - Text-to-speech x-platform. Disponibil la <https://pyttsx3.readthedocs.io/en/latest/>
- Anthony Zhang, Arvind Chembarpu, Kevin Smith, Kamus Hadenes, Sarah Braden, Bohdan Turkynewych, Steve Dougherty, Broderick Carlin (Dec 5, 2017). Speech recognition module for Python, supporting several engines and APIs, online and offline. Disponibil la [https://github.com/Uberi/speech\\_recognition#readme](https://github.com/Uberi/speech_recognition#readme)
- Hubert Pham (2006). PyAudio Documentation. Disponibil la <http://people.csail.mit.edu/hubert/pyaudio/docs/>
- Python Software Foundation (Ultimul update Apr 29, 2018). Convenient Web-browser controller <https://docs.python.org/2/library/webbrowser.html>
- Pradipta Bora (2014), (ultima versiune Mar 10, 2015). PyDictionary. Disponibil la <https://pypi.org/project/PyDictionary/>
- Python Software Foundation (Ultimul update Jun 17, 2018). URL handling modules. Disponibil la <https://docs.python.org/3/library/urllib.html>
- Python Software Foundation (Ultimul update Jun 17, 2018). Extensible library for opening URLs. Disponibil la <https://docs.python.org/3/library/urllib.request.html#module-urllib.request>
- Python Software Foundation (Ultimul update Jun 17, 2018). Exception classes raised by urllib.request. Disponibil la <https://docs.python.org/3/library/urllib.error.html#module-urllib.error>
- Python Software Foundation (Ultimul update Jun 17, 2018). Parse URLs into components. Disponibil la <https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse>
- Python Software Foundation (Ultimul update Apr 29, 2018). IMAP4 protocol client. Disponibil la <https://docs.python.org/2/library/imaplib.html>
- Voxtab (Octombrie 15, 2015). Voice Recognition Software: Pros and Cons. Disponibil la <https://www.voxtab.com/transcription-blog/voice-recognition-software-pros-and-cons/>

# Anexe

## Anexa 1

```
def menu_menu():
    engine = pyttsx3.init()
    mesaj_open1 = "Hello, welcome to the menu! "
    engine.say(mesaj_open1)
    print(mesaj_open1)
    mesaj_open1 = "Now i will read you all the available commands! "
    engine.say(mesaj_open1)
    print(mesaj_open1)
    mesaj_open1 = "If you want to access Youtube, say youtube ! "
    engine.say(mesaj_open1)
    print(mesaj_open1)
    mesaj_open1 = "If you want to search for a web-page, say search ! "
    engine.say(mesaj_open1)
    print(mesaj_open1)
    mesaj_open1 = "If you want to search for a word in dictionary, say dex ! "
    engine.say(mesaj_open1)
    print(mesaj_open1)
    mesaj_open1 = "If you want me to read your emails, say email ! "
    engine.say(mesaj_open1)
    print(mesaj_open1)
    mesaj_open1 = "If you want to close me, say quit ! "
    engine.say(mesaj_open1)
    print(mesaj_open1)
    engine.runAndWait()
```

## Anexa2

```
def search_google():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        engine = pyttsx3.init()
        mesaj_question = "What do you wish to search?"
        engine.say(mesaj_question)
        print(mesaj_question)
        engine.runAndWait()
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source, phrase_time_limit=5)
    try:
        webbrowser.get(google_api).open_new(url + r.recognize_google(audio))
    except:
        pass
```

## Anexa 3

```
def cauta_dex():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        engine = pyttsx3.init()
        mesaj_question = "What do you wish to search?"
        engine.say(mesaj_question)
        print(mesaj_question)
        engine.runAndWait()
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source, phrase_time_limit=5)
        message = r.recognize_google(audio)
    if lungime_text(message) == 1:
        try:
            engine = pyttsx3.init()
            bot_response = dictionary.meaning(message)
            print(bot_response)
            if not bot_response:
                webbrowser.get(google_api).open_new(url + message)
            else:
                engine.say(bot_response)
                print(bot_response)
                engine.runAndWait()
        except (IndexError, ValueError):
            webbrowser.get(google_api).open_new(url + message)
    elif lungime_text(message) > 1:
        try:
            webbrowser.get(google_api).open_new(url + r.recognize_google(audio))
        except:
            pass
```

## Anexa 4

```
def youtube_helper():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Ce doriti sa cautati?")
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source, phrase_time_limit=5)

        query_string = urllib.parse.urlencode({"search_query": audio})
        html_content = urllib.request.urlopen(url2 + r.recognize_google(audio))
        search_results = re.findall(r'href="\\"/watch?v=(.{11})"', html_content.read().decode())
        print(search_results)
    try:
        webbrowser.get(google_api).open_new(url3 + search_results[0])
    except:
        pass
```

## Anexa 5

```
result, data = mail.uid('search', None, "(UNSEEN)")
i = len(data[0].split())
for x in range(i):
    latest_email_uid = data[0].split()[x]

    result, email_data = mail.uid('fetch', latest_email_uid, '(RFC822)')

    raw_email = email_data[0][1]

    raw_email_string = raw_email.decode('utf-8')

    email_message = email.message_from_string(raw_email_string)

    i = 0
    for part in email_message.walk():
        if part.get_content_type() == "text/plain": # ignore attachments/html
            body = part.get_payload(decode=True)
            i = i + 1
            engine = pyttsx3.init()
            print("Unseen email %s" % i)
            engine.say("Unseen email %s" % i)
            print("from " + email_message['From'])
            engine.say("from " + email_message['From'])
            if email_message['Subject'] == '':
                print("with subject blank")
                engine.say("with subject blank")
```