Universitatea Națională de Știință și Tehnologie POLITEHNICA București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Sinteză vocală pentru limba română: implementare open-source și aplicații în chatbot-uri vocale

Lucrare de dizertație

prezentată ca cerință parțială pentru obținerea titlului de *Master* în domeniul *Inginerie electronică, telecomunicații și tehnologii informaționale*
programul de studii de masterat *Tehnologii multimedia în aplicații de biometrie și securitatea informației (BIOSINF)*

Conducător(i) științific(i):
Prof. dr. ing. Horia CUCU,
Drd. ing. Alexandru-Lucian GEORGESCU

Absolvent:
Ana-Maria Beatrice RIZAN

2026

Universitatea Națională de Știință și Tehnologie POLITEHNICA București
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Programul de masterat **BIOSINF**

Anexa 2

TEMA LUCRĂRII DE DISERTAȚIE

a masterandului **RIZAN G.L. Ana-Maria Beatrice, 421-BIOSINF**

1. Titlul temei: Sinteză vocală pentru limba română: implementare open-source și aplicații în chatbot-uri vocale

2. Descrierea temei:
Context
Sinteza vocală (Text-to-Speech, TTS) a evoluat semnificativ în ultimii ani datorită progresului rețelelor neuronale și modelelor de învățare profundă. Framework-uri open-source precum Coqui TTS sau Piper TTS oferă posibilitatea de a antrena modele personalizate pentru limbi și voci noi, utilizând seturi de date ce conțin asocieri de tip voce-text.
În prezent, pentru limba română există foarte puține modele TTS de calitate open-source. Crearea unei voci românești naturale, adaptate contextelor conversaționale, este esențială pentru dezvoltarea de aplicații vocale, cum ar fi asistenți vocali, sisteme IVR inteligente sau chatboturi pentru diverse domenii (ex. sănătate, educație, servicii publice).

Descriere proiect
Scopul proiectului este crearea unei noi voci TTS românești, folosind un framework open-source (Coqui TTS sau alternativă modernă compatibilă), urmată de integrarea acesteia într-un prototip de chatbot vocal.

3. Contribuția originală:
Activități principale
alegerea framework-ului TTS și analiză comparativă (Coqui, Piper, VoxCPM)
obținerea și pregătirea seturilor de date pe limba română SWARA dataset Zevo
antrenarea/adaptarea modelului TTS pentru limba română
expunerea modelului ca serviciu HTTP/WebSocket
integrarea noului TTS într-o arhitectură de IVR formată din Twilio Zevo STT LLM
testare, evaluare MOS și comparare cu Zevo TTS
documentare și împachetare finală
Cerinte minime necesare
competențe solide de programare Python (procesare audio, asincron, REST)
cunoștințe de machine learning / deep learning
experiență cu sisteme Linux și containere Docker
familiaritate cu API-uri web și comunicații pe socket-uri
noțiuni de procesare a limbajului natural și TTS/STT

4. Resurse și bibliografie:
1. Coqui.ai – Coqui TTS: A deep learning toolkit for Text-to-Speech synthesis: https://github.com/idiap/coqui-ai-TTS
2. ICVSI, Laborator 6, Biosinf, UNSTPB.
3. Twilio – Programmable Voice API Documentation, https://www.twilio.com/docs/voice
4. Stan, Adriana, et al. "The SWARA speech corpus: A large parallel Romanian read speech dataset." 2017 International Conference on Speech Technology and Human-Computer Dialogue (SpeD). IEEE, 2017.

5. Data înregistrării temei: 2026-01-15 18:54:02

Conducător(i) lucrare,
Conf. dr. ing. Horia CUCU

Dr. Lucian Georgescu

Student,
RIZAN G.L. Ana-Maria Beatrice

Responsabil program master,
Prof. dr. ing. Dragoş BURILEANU

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: **40cdcf2178**

Declarație de onestitate academică

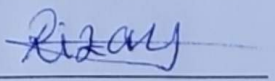
Prin prezenta declar că lucrarea cu titlul *Sinteză vocală pentru limba română: implementare open-source și aplicații în chatbot-uri vocale*, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității Naționale de Știință și Tehnologie POLITEHNICA București drept cerință parțială pentru obținerea titlului de *Master* în domeniul *Inginerie electronică, telecomunicații și tehnologii informaționale*, programul de studii de masterat *Tehnologii multimedia în aplicații de biometrie și securitatea informației (BIOSINF)*, este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referire la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referire la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 19.06.2026

Absolvent:
Ana-Maria Beatrice RIZAN



(semnătura în original)

Cuprins

LISTA ABREVIERILOR **xi**

1 INTRODUCERE **1**

2 CERCETAREA ȘI IMPLEMENTAREA COMPONENTEI DE SINTEZĂ VOCALĂ **5**
2.1 Fundamente teoretice în sinteza vocală **5**
2.2 Analiza soluțiilor existente (State of the Art) **6**
2.3 Arhitectura sistemului propus **7**
2.3.1 Componentele arhitecturii VITS **9**
2.3.2 Metodologia de antrenare și selecția scenariului optim **9**
2.4 Infrastructura experimentală și mediul de lucru **10**
2.5 Setul de date utilizat — Corpusul SWARA **12**
2.5.1 Organizarea, preprocesarea și adaptarea datelor **13**
2.6 Experimentele de antrenare — Hyperparameter Search **14**
2.6.1 Strategia experimentală și parametrii ficși **14**
2.6.2 Configurațiile experimentale și preprocesarea avansată **15**
2.6.3 Analiza rezultatelor și monitorizarea antrenării **17**
2.6.4 Fine-tuning pe vorbitori specifici **18**
2.7 Evaluarea sistemului de sinteză vocală **20**
2.7.1 Metodologia de evaluare și metrici utilizate **20**
2.7.2 Rezultatele evaluării **21**
2.7.3 Corecția pronunției termenilor din vocabularul de specialitate italian **25**
2.7.4 Evaluarea perceptuală a calității sintezei **30**
2.7.5 Comparație cu sistemul comercial Zevo TTS **33**
2.8 Expunerea modelului ca serviciu web HTTP **34**
2.9 Sinteza capitolului **35**

3 PROIECTAREA ȘI IMPLEMENTAREA COMPONENTEI DE ÎNȚELEGERE A LIMBAJULUI NATURAL **37**
3.1 Fundamente teoretice și selecția modelelor LLM **37**
3.2 Arhitectura componentei NLU și *Prompt Engineering* **38**
3.3 Setul de date pentru testare (NLU Dataset) **41**
3.4 Metodologia de evaluare a componentei NLU **42**
3.5 Rezultate experimentale și analiză comparativă **43**
3.6 Evaluarea unui model LLM local (CPU-only) **45**

3.7 Sinteză comparativă și direcții de optimizare **47**

4 PROIECTAREA ȘI IMPLEMENTAREA SERVERULUI IVR **49**
4.1 Arhitectura sistemului: componente și schimb de mesaje **49**
4.2 Modulul de gestiune a dialogului **50**
4.3 Integrarea componentelor în fluxul de procesare **51**
4.4 Interfața de monitorizare și demonstrație **52**
4.5 Limitări și concluzii **53**

5 CONCLUZII ȘI DIRECȚII VIITOARE DE DEZVOLTARE **55**
5.1 Concluzii pe componente implementate **55**
5.1.1 Sinteza vocală (TTS) **55**
5.1.2 Înțelegerea limbajului natural (NLU) **55**
5.1.3 Integrarea serverului IVR **56**
5.2 Contribuții originale **56**
5.3 Limitări și direcții viitoare de dezvoltare **57**

BIBLIOGRAFIE **59**

Listă de tabele

2.1 Analiza comparativă a framework-urilor TTS 6
2.2 Specificațiile infrastructurii de antrenare 10
2.3 Sinteza caracteristicilor corpusului SWARA 12
2.4 Distribuția corpusului SWARA utilizat în experimente 13
2.5 Parametrii ficși ai experimentelor 15
2.6 Configurațiile hyperparameter search 16
2.7 Rezultatele finale ale antrenării generale 17
2.8 Rezultatele experimentelor de fine-tuning 19
2.9 Distribuția setului de evaluare (100 enunțuri) 22
2.10 Rezultatele globale ale evaluării — modele originale vs. augmentate. Valorile WER/CER pentru modelele augmentate sunt calculate cu normalizarea inițială; valorile corectate sunt prezentate în Tabelul 2.15. 22
2.11 WER și CER normalizat per categorie — toate modelele. Valorile coloanei FT-C-aug utilizează normalizarea inițială; valoarea corectată WER pizza este 29,7% (Tabelul 2.15). 23
2.12 Dicționar fonetic pentru modelul grafemic FT-C-aug. Termenii *Diavola* și *Carnivora* nu necesită substituție, evaluarea perceptuală indicând pronunție acceptabilă la nivel fonetic. 27
2.13 Comparație între strategiile de corecție a pronunției OOV și baseline-ul fără dicționar, investigate pe modelul grafemic FT-C-aug. Simbolul "–" indică valori neraportate separat, deteriorate global față de baseline. 27
2.14 Fonetizarea automată a termenilor italieni de eSpeak-NG pentru limba română. Patru din cinci termeni sunt fonetizați rezonabil, beneficiind de particularitățile comune ale fonetismului românesc și italian. 28
2.15 Rezultate WER normalizat cu și fără dicționar fonetic pentru modelele evaluate. Coloana Δ pizza indică variația față de baseline-ul fără dicționar al fiecărui model. 30
2.16 Sistemele TTS evaluate perceptual 31
2.17 Scoruri perceptuale globale (17 evaluatori, $N = 97$ eșantioane per sistem) . 31
2.18 Scoruri perceptuale per categorie de enunțuri 32
2.19 Comparație WER între modelul antrenat și Zevo TTS pe setul de 100 de enunțuri (Whisper medium) 33
3.1 Modelele LLM utilizate în evaluarea componentei NLU 38
3.2 Rezultate agregate (100 exemple) pentru fiecare model și prompt 44

3.3 Modele LLM testate în configurație CPU-only 46
3.4 Rezultate modele CPU-only (P3, 100 exemple) 46
3.5 Comparație modele comerciale (P3) vs. modele locale CPU-only 47

Listă de figuri

2.1 **Schema generală a arhitecturii framework-ului Coqui TTS [24].** **8**
2.2 **Interfețele utilizate pentru accesul la distanță: WireGuard VPN (stânga) și PuTTY SSH (dreapta).** **11**
2.3 **Exemplu al procesului de adnotare la nivel de enunț; enunțurile corecte sunt marcate cu „a” (audio), iar restul cu „j” (junk) [23].** **13**
2.4 **Evoluția funcției de pierdere avg_loss_mel pentru configurațiile A, B, C și D pe durata antrenării.** **18**
2.5 **Proiecție t-SNE a embedding-urilor de vorbitor (ECAPA-TDNN): distincția între vorbitorii SWARA, vorbitorul extern real și eșantioanele sintetizate prin modelele augmentate.** **25**
2.6 **Interfața web de testare a serviciului TTS HTTP.** **35**
3.1 **Evoluția Strict Record Accuracy pentru variantele de prompt P1–P3** **44**
4.1 **Arhitectura sistemului IVR conversational integrat. Săgețile continue indică fluxul de date dus, iar cele punctate răspunsul întors. Componentele locale rulează pe serverul de calcul; comunicarea cu Twilio se realizează prin tunelul ngrok, iar procesarea vocii implică serviciile externe Zevo STT și Azure OpenAI, respectiv modelul TTS antrenat local.** **50**
4.2 **Interfața web de monitorizare a conversației IVR în timp real. Mesajele sintetizate de robot apar în stânga (albastru deschis), iar transcrierile vocii apelantului în dreapta (albastru închis).** **53**

LISTA ABREVIERILOR

API	*Application Programming Interface* (lb. en.; interfață de programare a aplicațiilor)
ASR	*Automatic Speech Recognition* (lb. en.; recunoaștere automată a vorbirii)
CER	*Character Error Rate* (lb. en.; rată de eroare la nivel de caracter)
CNN	*Convolutional Neural Network* (lb. en.; rețea neurală convoluțională)
GPU	*Graphics Processing Unit* (lb. en.; unitate de procesare grafică)
HMM	*Hidden Markov Model* (lb. en.; model Markov ascuns)
HoReCa	Hotels, Restaurants, Catering
HTTP	*Hypertext Transfer Protocol*
IA	Inteligență artificială
IPA	*International Phonetic Alphabet* (lb. en.; alfabet fonetic internațional)
IVR	*Interactive Voice Response* (lb. en.; răspuns vocal interactiv)
JSON	*JavaScript Object Notation*
LLM	*Large Language Model* (lb. en.; model de limbaj de mari dimensiuni)
MFCC	*Mel-frequency Cepstral Coefficient* (lb. en.; coeficient Mel-cepstral)
NLU	*Natural Language Understanding* (lb. en.; înțelegerea limbajului natural)
OOV	*Out-of-Vocabulary* (lb. en.; în afara vocabularului)
REST	*Representational State Transfer*
RTF	*Real-Time Factor* (lb. en.; factor de timp real)
STT	*Speech-to-Text* (lb. en.; conversie vorbire în text)
TTS	*Text-to-Speech* (lb. en.; sinteză vocală)
URL	*Uniform Resource Locator*
VAD	*Voice Activity Detection* (lb. en.; detecție a activității vocale)
VITS	*Variational Inference with adversarial learning for end-to-end Text-to-Speech*
VRAM	*Video Random Access Memory*
WER	*Word Error Rate* (lb. en.; rată de eroare la nivel de cuvânt)

1. INTRODUCERE

Sistemele conversaționale vocale au devenit, în ultimul deceniu, una dintre cele mai active direcții de cercetare la intersecția dintre procesarea limbajului natural, sinteza vocală și recunoașterea automată a vorbirii. Adopția pe scară largă a asistenților virtuali în limbi de circulație internațională (engleză, spaniolă, mandarină) a fost facilitată de disponibilitatea unor corpusuri vocale de mari dimensiuni și a unor modele lingvistice preantrenate robuste. Pentru limbile cu resurse reduse, printre care se numără și limba română, dezvoltarea unor astfel de sisteme rămâne însă o provocare deschisă, atât din perspectiva resurselor de date, cât și a adaptării arhitecturilor moderne la particularitățile fonetice și morfologice ale limbii.

Domeniul HoReCa (Hotels, Restaurants, Catering) reprezintă un scenariu de aplicare foarte relevant pentru sistemele conversaționale vocale, datorită volumului ridicat de interacțiuni repetitive (rezervări, comenzi, solicitări de informații) care pot fi automatizate cu un grad ridicat de fiabilitate. Implementările comerciale existente se bazează aproape exclusiv pe servicii cloud proprietare, care ridică probleme de cost operațional, dependență de furnizor și conformitate cu reglementările privind protecția datelor cu caracter personal. Absența unor alternative open-source mature, executabile pe infrastructură locală și adaptate particularităților fonetice ale limbii române, reprezintă deopotrivă un gol în literatura de specialitate și o barieră practică pentru operatorii din această industrie care doresc automatizarea interacțiunilor vocale fără dependențe față de servicii terțe.

Prezenta lucrare propune proiectarea și implementarea unui sistem vocal conversațional în limba română, specializat pentru domeniul HoReCa, construit integral pe componente open-source. Sistemul integrează trei module principale: o componentă proprie de sinteză vocală (Text-to-Speech, TTS) bazată pe arhitectura VITS [13] și antrenată pe corpusul SWARA [23], o componentă de înțelegere a limbajului natural (Natural Language Understanding, NLU) bazată pe modele de limbaj de mari dimensiuni (Large Language Models, LLM) pentru extragerea simultană a intenției și a entităților din enunțuri în limba română, și o componentă de telefonie interactivă (Interactive Voice Response, IVR) care permite integrarea într-un flux real de procesare a apelurilor prin platforma Twilio.

Obiectivele principale ale lucrării sunt:

(i) dezvoltarea unui model TTS multi-speaker pentru limba română antrenat de la zero pe corpusul SWARA consolidat, cu un Real-Time Factor compatibil cu sinteza în timp real și o calitate perceptuală suficientă pentru deployment conversațional;

(ii) specializarea modelului general pe vorbitori individuali prin fine-tuning, inclusiv pe un vorbitor extern corpusului de antrenare, cu adresarea problemei vocabularului lipsă

(Out-of-Vocabulary, OOV) pentru terminologia italiană specifică domeniului restaurant;

(iii) evaluarea comparativă a patru modele LLM comerciale și a trei modele locale rulate exclusiv pe procesor, pentru sarcina de extragere combinată a intenției și a sloturilor semantice din enunțuri în limba română, sub constrângere de format JSON validabil;

(iv) integrarea componentelor TTS, STT și NLU într-un server IVR funcțional, capabil să susțină o conversație vocală completă printr-un apel telefonic real.

În cadrul componentei TTS, a fost realizată o căutare sistematică a hiperparametrilor pe patru configurații experimentale care izolează efectul dimensiunii batch-ului, al preprocesării acustice și al reprezentării lingvistice (grafeme versus foneme IPA), antrenând modelul VITS multi-speaker pe corpusul SWARA consolidat (versiunile 1.0 și 2.0, aproximativ 60 de ore, 41 de vorbitori). Configurația optimă a fost supusă fine-tuning-ului pe un vorbitor extern, pentru care s-a identificat fenomenul de word-sticking ca limitare arhitecturală intrinsecă a modelelor VITS end-to-end. Pentru adresarea vocabularului lipsă, a fost elaborată o strategie de augmentare prin distilarea cunoștințelor vocale dintr-un model comercial (ElevenLabs), reducând WER-ul pe categoria comenzilor de pizza de la 74,6% la 29,7% (configurația augmentată fără dicționar fonetic), respectiv la 26,5% cu aplicarea dicționarului fonetic. Calitatea sintezei a fost validată atât prin metrici obiective — Word Error Rate (WER), Character Error Rate (CER), Real-Time Factor (RTF) și similaritate a vorbitorului prin ECAPA-TDNN — cât și printr-o evaluare perceptuală de tip MUSHRA cu 17 evaluatori, modelul augmentat selectat obținând un scor global de 78,3/100. Suplimentar, a fost realizată o validare ASR bilingvă (română și italiană) pentru a disocia erorile de calitate ale sintezei de penalizările introduse de modelul de limbă al sistemului de recunoaștere utilizat în evaluare.

În cadrul componentei NLU, a fost realizată o evaluare comparativă între soluții bazate pe modele LLM comerciale accesibile prin API și modele de dimensiuni reduse disponibile public, rulate local pe CPU, în vederea identificării unei arhitecturi capabile să proceseze robust solicitări complexe și colocviale în limba română. Prin aplicarea unor strategii de inginerie iterativă a prompturilor, de la constrângeri zero-shot la rubrici procedurale explicite, s-a demonstrat viabilitatea net superioară a modelelor comerciale pentru sarcina de extragere strictă a intențiilor și a sloturilor semantice, în contrast cu limitările severe ale modelelor locale în ceea ce privește respectarea constrângerilor de format și canonicalizarea entităților. Componenta NLU îndeplinește în arhitectura sistemului rolul unui modul de suport esențial pentru funcționarea end-to-end a fluxului IVR, completând și valorificând contribuțiile aduse în direcția sintezei vocale.

La nivelul arhitecturii integrate, componenta TTS a fost expusă ca serviciu web HTTP prin Flask, cu un RTF mediu de 0,022, facilitând integrarea modulară în fluxul IVR independent de mediul de antrenare. Serverul IVR a fost implementat prin tehnica de monkey-patching (suprascriere dinamică a funcțiilor la runtime) a interfețelor de intrare/ieșire ale modulului de dialog, permițând reutilizarea neschimbată a logicii conversaționale în contextul canalului telefonic Twilio.

Lucrarea este structurată în patru capitole de conținut. Capitolul 2 prezintă fundamentele teoretice ale sintezei vocale, analiza comparativă a framework-urilor open-source disponibile, metodologia experimentală de antrenare a modelului VITS pe corpusul SWARA, etapa de

fine-tuning pe vorbitori specifici, inclusiv strategia de augmentare cu date sintetice și corecția pronunției termenilor OOV, și evaluarea obiectivă și perceptuală a sistemului rezultat. Capitolul 3 detaliază proiectarea componentei NLU, incluzând selecția și evaluarea comparativă a modelelor LLM, ingineria iterativă a prompturilor și analiza taxonomică a erorilor. Capitolul 4 descrie arhitectura serverului IVR și integrarea componentelor TTS și STT într-un flux funcțional de procesare a apelurilor telefonice reale, incluzând interfața de monitorizare în timp real și limitările inerente ale canalului telefonic.

2. CERCETAREA ȘI IMPLEMENTAREA COMPONENTEI DE SINTEZĂ VOCALĂ

În acest capitol sunt prezentate fundamentele teoretice ale sintezei vocale, urmate de o analiză a principalelor soluții disponibile la nivelul actual al tehnologiei (*State of the Art*). Totodată, este descrisă metodologia experimentală adoptată pentru antrenarea unui model de sinteză vocală *multi-speaker* pentru limba română, incluzând infrastructura utilizată, corpusul de date, configurarea framework-ului și strategia de evaluare. Prin adaptarea arhitecturii VITS la particularitățile fonetice ale limbii române, obiectivul urmărit este obținerea unei voci naturale și cu o prozodie adecvată contextelor conversaționale, integrabilă în soluții complexe de tip chatbot.

2.1. Fundamente teoretice în sinteza vocală

Sinteza vocală, cunoscută sub acronimul TTS (*text-to-speech*), reprezintă procesul de transformare a textului scris în vorbire articulată, prin mijloace artificiale. Pe parcurs, această tehnologie a trecut prin mai multe etape, fiecare fiind definită de schimbări semnificative în modul de procesare a semnalului audio.

a) **Sinteza concatenativă:** Reprezentând paradigma dominantă a deceniilor trecute, metoda se bazează pe procesul de selecție și îmbinare a unor unități audio pre-înregistrate dintr-un corpus vocal uriaș. Deși segmentele individuale au fidelitate acustică ridicată, rezultatul final suferă de prozodie rigidă și artefacte sonore la punctele de joncțiune, ceea ce limitează naturalețea vorbirii sintetizate [4].

b) **Sinteza parametrică statistică (HMM):** Marcând o etapă importantă în evoluția sintezei vocale a fost introducerea Modelelor Markov Ascunse (*Hidden Markov Models*), care permit generarea vorbirii prin modelarea probabilistică a parametrilor acustici [28], această abordare a condus la o reducere semnificativă a cerințelor computaționale. Totuși, semnalul generat era perceput ca artificial, cu expresivitate redusă.

c) **Sinteza bazată pe învățare profundă (Neural TTS):** Constituind paradigma dominantă în prezent, această abordare se bazează pe utilizarea rețelelor neuronale profunde (DNN) pentru estimarea caracteristicilor acustice ale vorbirii. Aceasta permite modelarea nuanțată a variațiilor subtile de intonație și ritm, oferind un nivel ridicat de flexibilitate și naturalețe prozodică, apropiindu-se semnificativ de vocea umană din punct de vedere perceptual în multe aplicații.[4, 25].

   În arhitecturile neuronale moderne de tip „Two-stage”, procesul de sinteză este divizat în

două componente fundamentale:

- **Modelul acustic (Front-end):** Realizează transpunerea secvenței lingvistice într-o reprezentare intermediară de înaltă rezoluție, numită mel-spectrogramă. Din punct de vedere tehnic, aceasta reprezintă o transformată Fourier pe durată scurtă (STFT), unde frecvențele sunt proiectate pe scala Mel [17]. Conform studiilor recente privind paradigma de îmbunătățire a spectrogramei, această reprezentare este esențială deoarece realizează o compresie eficientă a datelor, păstrând în același timp trăsăturile acustice critice pentru percepția umană [25].Modelul acustic aliniază durata fonemelor cu variațiile spectrale, producând o reprezentare compactă care păstrează trăsăturile acustice relevante perceptual.
- **Vocoderul (Back-end):** Reprezintă sistemul de reconstrucție neuronală care are sarcina complexă de a recupera informația de fază pierdută în timpul generării spectrogramei [25]. Spre deosebire de metodele tradiționale de estimare a fazei care folosesc algoritmi deterministici de estimare, vocoderele de tip neural (precum HiFi-GAN [14]) utilizează rețele generative de tip adversarial pentru a sintetiza forma de undă [25]. Această abordare permite obținerea unei clarități sonore ridicată și a unei texturi naturale a discursului, eliminând artefactele de aliasing și zgomotul de cuantizare care, în tehnologiile anterioare, confereau vocii un caracter sintetic, lipsit de fluiditate.

În cercetarea actuală se adoptă modelele de tip „totul-într-unul” (*end-to-end*), precum arhitectura VITS, care se bazează pe o fuziune a modelului acustic cu vocoderul într-un flux de lucru integrat. Această abordare unifică întregul proces de transformare a textului în undă sonoră sub forma unei singure rețele neuronale complexe, eliminând barierele dintre etapele clasice de procesare. Prin această optimizare globală, se evită propagarea erorilor între module, rezultatul fiind un semnal vocal mult mai autentic, caracterizat printr-o coerență și o naturalețe superioară a discursului [25].

2.2. Analiza soluțiilor existente (State of the Art)

Pentru selectarea cadrului de lucru (framework), au fost analizate mai multe soluții open-source relevante acestui proiect, luând în calcul capacitatea de gestionare a întregului pipeline de sinteză și pe suportul pentru *fine-tuning* pe date în limba română. O sinteză a principalelor caracteristici tehnice este prezentată în Tabelul 2.1.

Tabelul 2.1. Analiza comparativă a framework-urilor TTS

Criteriu	**Piper TTS**	**Coqui TTS**	**VoxCPM / Clip-TTS**
Paradigmă	VITS (Non-AR)	Modular (AR/VITS)	Semantic-Aware
Modularitate	Scăzută	Ridicată	Medie
Hardware	CPU (Optimized)	GPU / CPU	GPU High-end
Fine-tuning	Restrictiv	Avansat (Trainer API)	Experimental

- **Piper TTS:** O soluție axată pe eficiență, care utilizează modele VITS exportate în format ONNX. Piper elimină complexitatea prin utilizarea unor modele pre-antrenate rigide, fiind optimizat pentru o latență minimă de sub 15 ms [8]. Deși ideal pentru sisteme embedded, acesta oferă puține opțiuni de intervenție asupra arhitecturii interne.
- **Coqui TTS:** Se definește ca un ecosistem modular complet, oferind o separare clară între componentele de procesare: convertorul text-to-phoneme, modelul acustic și vocoderul. Această arhitectură permite utilizatorului să experimenteze cu diverse configurații (de exemplu, combinarea modelului *Glow-TTS* cu vocoderul *HiFi-GAN* [14] sau utilizarea arhitecturii *VITS* pentru sinteză end-to-end). Un avantaj critic, evidențiat în analizele tehnice recente [24], este interfața de antrenare (Trainer API), care automatizează procesul de *transfer learning* și gestionarea seturilor de date, facilitând adaptarea rapidă pe corpusuri precum SWARA.
- **VoxCPM / Clip-TTS:** Reprezintă cele mai avansate soluții pentru procesarea semantică a contextului. Modelele folosesc învățarea contrastivă pentru corelarea textului cu reprezentările spectrale, oferind o naturalețe semantică optimizată prin eliminarea limitărilor tehnice ale modelelor vechi [16]. Complexitatea arhitecturală le face însă dificil de integrat în aplicații cu cerințe stricte de latență pe hardware standard..

În urma analizei, s-a optat pentru framework-ul **Coqui TTS** (versiunea 0.27.2, fork Idiap), utilizând modelul **VITS**. Acesta reprezintă soluția optimă pentru un chatbot vocal în limba română, deoarece oferă echilibrul necesar între viteza de răspuns a arhitecturii VITS, modularitatea ridicată a toolkit-ului și suportul nativ pentru scenarii *multi-speaker* [3]. O alternativă evaluată este **YourTTS** [3], variantă a VITS adaptată explicit pentru scenarii *multi-speaker* cu resurse lingvistice limitate, disponibilă de asemenea în Coqui TTS și utilă pentru transfer-learning. Totuși, datorită volumului suficient de date oferit de corpusul SWARA (1.0+2.0) și a stabilității dovedite a configurației de antrenare, a fost folosit VITS standard.

O evaluare sistematică a framework-urilor TTS open-source pentru limba română, incluzând VITS, FastPitch, Grad-TTS și Matcha-TTS, este realizată de Răgman et al. [22], care constituie principala lucrare înrudită a prezentei cercetări.

2.3. Arhitectura sistemului propus

Sistemul de sinteză vocală propus utilizează arhitectura **VITS** (*Variational Inference with adversarial learning for end-to-end Text-to-Speech*), publicată de Kim et al. în 2021 [13], în cadrul framework-ului selectat în Secțiunea 2.2. Spre deosebire de lucrările anterioare în domeniu care utilizează Tacotron2 [18], VITS este un model *end-to-end* care generează audio direct din text, fără a necesita un vocoder separat, obținând o calitate superioară a vorbirii sintetizate.

Din punct de vedere conceptual, arhitectura Coqui TTS este organizată sub forma unui pipeline de procesare care transformă textul de intrare într-un semnal audio natural, printr-o succesiune de etape bine definite [25]. Arhitectura generală a sistemului include următoarele componente funcționale principale:

- **Front-end textual:** componentă responsabilă pentru preprocesarea și normalizarea textului

de intrare, incluzând expandarea numerelor, a abrevierilor și eliminarea simbolurilor non-standard [25].

– **Reprezentarea lingvistică:** textul normalizat este transformat într-o secvență de unități lingvistice (caractere sau foneme), care constituie intrarea pentru modelul neuronal de sinteză [25].
– **Modelul de sinteză neuronală:** componenta centrală, care realizează maparea dintre reprezentarea lingvistică și caracteristicile acustice ale vorbirii, generând o reprezentare latentă intermediară [17, 25].
– **Generatorul audio (vocoder):** în arhitectura VITS, vocoderul (HiFi-GAN [14]) este integrat direct în model și generează forma de undă audio final din reprezentarea latentă, eliminând etapa separată de reconstrucție a semnalului [25].

Arhitecturile neuronale suportate de Coqui TTS pot fi grupate în trei categorii, relevante pentru înțelegerea alegerii făcute în această lucrare:

– **Arhitecturi autoregresive**, care generează vorbirea secvențial, cadru cu cadru. Aceste modele oferă stabilitate ridicată a alinierii text–audio, însă prezintă o latență crescută, fiind mai puțin potrivite pentru aplicații conversaționale în timp real [4].
– **Arhitecturi non-autoregresive**, care permit generarea paralelă a vorbirii, reducând semnificativ timpul de inferență, însă necesită de regulă un vocoder separat pentru reconstrucția semnalului audio [25].
– **Arhitecturi end-to-end**, care unifică procesul de generare acustică și vocoding într-o singură rețea neuronală, eliminând etapele intermediare explicite și permițând optimizarea globală a procesului de sinteză [16, 25]. VITS aparține acestei categorii, ceea ce justifică alegerea sa pentru un sistem conversațional cu cerințe de latență redusă.

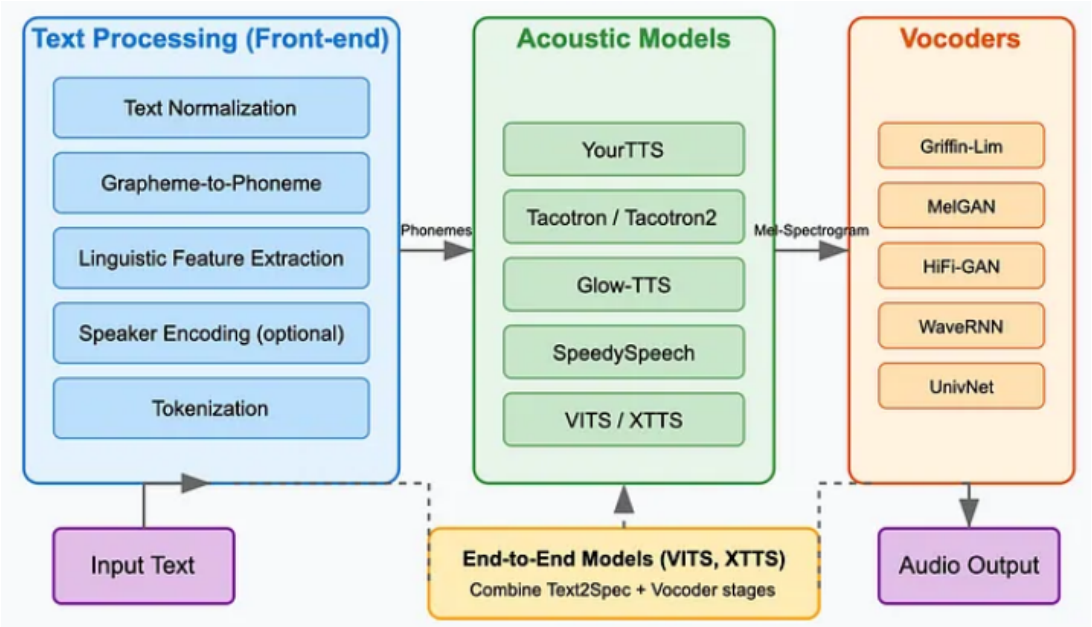


Figura 2.1. Schema generală a arhitecturii framework-ului Coqui TTS [24].

Figura 2.1 evidențiază structura modulară a framework-ului Coqui TTS, precum și fluxul de procesare de la textul de intrare până la generarea semnalului audio.

2.3.1. Componentele arhitecturii VITS

Modelul VITS utilizat în această lucrare este configurat în varianta *multi-speaker* și cuprinde următoarele componente funcționale principale, care conlucrează pentru a asigura naturalețea și fidelitatea semnalului audio generat:

- **Encoder de text:** Bazat pe o arhitectură Transformer cu mecanisme de atenție multi-cap, acesta transformă secvența de intrare într-o reprezentare latentă bogată în informații lingvistice. În funcție de configurația experimentală, intrarea constă fie în caractere grafematice (vocabular de 36 de simboluri), fie în foneme IPA (vocabular de 132 de simboluri) generate prin fonetizatorul eSpeak-NG [6].
- **Model de durată stochastic (Stochastic Duration Predictor):** Estimează durata fiecărui fonem sau caracter. Fiind un model stochastic, acesta permite generarea unor variații naturale de ritm pentru același text, imitând variabilitatea vorbirii umane [25].
- **Flux normalizant (Normalizing Flow):** Realizează maparea complexă dintre spațiul latent lingvistic și distribuția caracteristicilor acustice (mel-spectrogramă), asigurând consistența spectrală a sunetelor.
- **Decoder integrat (HiFi-GAN):** Reconstruiește forma de undă audio direct din reprezentarea latentă. Integrarea acestuia sub formă de generator adversarial elimină necesitatea unui vocoder separat și asigură o claritate ridicată a vocii [25].
- **Embedding de vorbitor (Speaker Embedding):** Constă într-un vector de 256 de dimensiuni care codifică identitatea vocală unică a fiecărui vorbitor. Acest modul permite modelului să partajeze cunoștințele lingvistice între toți cei 41 de vorbitori din corpusul SWARA, izolând în același timp trăsăturile de timbru specifice fiecăruia.

Dimensiunea totală a modelului implementat este de **86.402.668 de parametri antrenabili**, suficienți pentru a acoperi diversitatea fonetică a celor 41 de vorbitori din corpusul SWARA.

2.3.2. Metodologia de antrenare și selecția scenariului optim

Dezvoltarea unui sistem TTS performant pentru limba română presupune luarea unor decizii metodologice critice, influențate de disponibilitatea resurselor lingvistice și de capacitatea computațională. Au fost evaluate două scenarii principale de abordare:

a) **Antrenarea de la zero (*from scratch*):** Presupune antrenarea integrală a modelului VITS utilizând exclusiv corpusul SWARA 1.0 + 2.0, fără utilizarea unor ponderi pre-antrenate. Această metodă oferă un control total asupra adaptării vocabularului la diacriticele românești și permite modelului să învețe prosodia limbii fără influențe din alte limbi.

b) **Învățare prin transfer (*transfer learning*):** Utilizează un model pre-antrenat pe un corpus extins într-o limbă cu resurse bogate, urmat de o etapă de adaptare pe datele în limba română. Deși converge mai rapid, acest scenariu poate introduce artefacte prozodice străine sau limitări în reprezentarea diacriticelor specifice.

A fost ales **Scenariul 1 — antrenarea de la zero**, motivat de absența unui model VITS pre-antrenat robust disponibil public pentru limba română la momentul lansării experimentelor și de volumul considerabil de date oferit de consolidarea versiunilor SWARA (~60 ore, 41 vorbitori). Această decizie a permis un control complet asupra procesului de antrenare în condițiile specifice ale limbii române, asigurând o bază solidă pentru etapa ulterioară de specializare prin fine-tuning.

Procesul de antrenare urmează un etape bine definite, pornind de la configurarea vocabularului grafematic, trecând prin preprocesarea audio cu eliminarea zgomotului și a enunțurilor scurte, până la monitorizarea convergenței prin metrici de tip *loss* și evaluarea prin vizualizări t-SNE ale spațiului embedding-urilor.

2.4. Infrastructura experimentală și mediul de lucru

Experimentele de antrenare a modelului VITS au fost realizate pe serverul de calcul al facultății, accesat de la distanță prin intermediul unei conexiuni securizate. Serverul, identificat ca `rtx-4090-01`, dispune de o unitate de procesare grafică **NVIDIA GeForce RTX 4090** cu 24 GB VRAM, esențială pentru încărcarea tensorilor în regim *batch* și pentru accelerarea algoritmilor de *backpropagation*, 64 GB RAM și un procesor cu 32 de nuclee. Stiva software este construită pe Python 3.10 într-un mediu virtual Conda (`miniforge3`), utilizând PyTorch 2.8.0 cu suport CUDA 12.8 pentru interfațarea cu GPU-ul și framework-ul Coqui-TTS 0.27.2 (fork Idiap). Specificațiile complete sunt sintetizate în Tabelul 2.2.

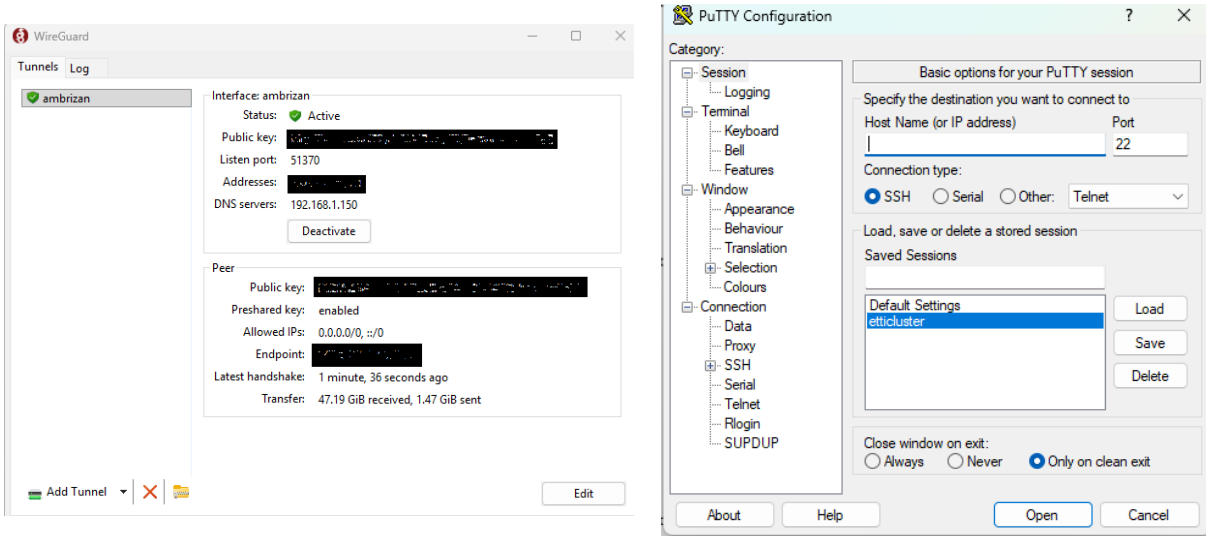
Tabelul 2.2. Specificațiile infrastructurii de antrenare

Componentă	**Specificații**
GPU	NVIDIA GeForce RTX 4090, 24 GB VRAM
CPU	32 nuclee
RAM	64 GB
CUDA	Versiunea 12.8, Driver 580.126.09
Sistem de operare	Ubuntu 22.04.5 LTS (Linux 5.15.0)
Python	3.10, mediu virtual Conda (miniforge3)
PyTorch	2.8.0 cu suport CUDA
Framework TTS	Coqui-TTS 0.27.2 (fork Idiap)

Accesul la server din afara rețelei facultății se realizează în două etape securizate. Inițial, se stabilește un tunel **VPN** (*Virtual Private Network*) utilizând protocolul **WireGuard**, urmat de o conexiune de tip terminal prin protocolul **SSH** cu ajutorul clientului **PuTTY**. Această configurație permite gestionarea experimentelor independent de locația fizică a autorului sau de resursele hardware locale.

Odată stabilită conexiunea, mediul de lucru este activat prin scriptul de inițializare Conda:

```
source ~/miniforge3/bin/activate
conda activate tts_env
```



(a) Interfața WireGuard **(b) Configurația PuTTY**

Figura 2.2. Interfețele utilizate pentru accesul la distanță: WireGuard VPN (stânga) și PuTTY SSH (dreapta).

Deoarece antrenarea modelelor VITS este un proces consumator de timp, putând dura zeci de ore, s-a utilizat utilitarul **tmux** (*terminal multiplexer*). Acesta permite detașarea sesiunii active (prin comanda `Ctrl+B, D`) fără a întrerupe procesul de antrenare, permițând reatașarea ulterioară (`tmux attach -t configA`) chiar și după deconectarea SSH. Această abordare a facilitat lansarea simultană a mai multor configurații experimentale (`configA`, `configB`, `configC`) și monitorizarea lor independentă.

Pe parcursul configurării, a fost identificată o incompatibilitate între PyTorch 2.8 și CUDA 12.8 la nivelul operațiilor de convoluție în modul `fp16`, soluționată prin activarea formatului **TF32** (*TensorFloat-32*). Acest format asigură un compromis optim între viteza de calcul și precizia numerică pe arhitectura RTX 4090:

```
torch.backends.cudnn.enabled = True
torch.backends.cuda.matmul.allow_tf32 = True
torch.backends.cudnn.allow_tf32 = True
mixed_precision = True
```

Această optimizare a permis utilizarea preciziei mixte `fp16`, reducând timpul per epocă cu aproximativ **20%** (de la ~15 la ~12 minute), un câștig de eficiență critic pentru convergența modelului în termenele stabilite.

2.5. Setul de date utilizat — Corpusul SWARA

Pentru dezvoltarea sistemului de sinteză vocală în limba română, s-a utilizat corpusul **SWARA** (*Speech Corpus for Romanian*), o resursă concepută special pentru aplicații de tip *Text-to-Speech* [23]. Corpusul este structurat în două versiuni distincte, SWARA 1.0 și SWARA 2.0, care diferă atât prin condițiile de înregistrare, cât și prin setul de propoziții utilizat.

Versiunea **SWARA 1.0** cuprinde aproximativ 21 de ore de înregistrări de înaltă fidelitate de la 18 vorbitori nativi [23]. Acestea au fost realizate într-un studio profesional complet izolat fonic, utilizând un microfon AKG C214 și o placă de sunet MOTU UltraLite MK3 [23]. Conversia analog-digitală a fost efectuată la o frecvență de eșantionare de 48 kHz, cu o adâncime de 16 biți [23]. Setul de propoziții utilizat conține **1.792 de enunțuri unice**.

Extensia **SWARA 2.0** a intenționat să adauge 30 de vorbitori noi, însă corpusul disponibil pe serverul de antrenare conține 23 de vorbitori din acest set, ale căror înregistrări au fost efectuate în mediul casnic, utilizând utilitarul *RECOApy* [18]. Această abordare introduce diversitate acustică prin variația condițiilor de înregistrare — zgomot ambiental, reverberație, echipamente diferite — cu beneficii pentru robustețea modelului în condiții reale. Setul de propoziții al SWARA 2.0 conține **1.724 de enunțuri unice**, parțial diferit de cel al SWARA 1.0.

Tabelul 2.3. Sinteza caracteristicilor corpusului SWARA

Caracteristică	**SWARA 1.0**	**SWARA 2.0**	**Total**
Vorbitori	18	23	41
Propoziții unice	1.792	1.724	2.792*
Fișiere audio	22.916	32.241	55.157
Durată audio	~21h	~39h	~60h
Mediu înregistrare	Studio profesional	Mediu casnic	Mixt
Eșantionare	48 kHz / 16 bit	48 kHz / 16 bit	22.050 Hz (server)

* *Numărul de propoziții unice la nivel de corpus consolidat, după deduplicare.*

O analiză a suprapunerii de text între cele două versiuni relevă că corpusul SWARA este **parțial paralel**: din totalul propozițiilor, **1.397 sunt comune** ambelor versiuni, în timp ce 395 apar exclusiv în SWARA 1.0 și 327 exclusiv în SWARA 2.0. Această proprietate este avantajoasă pentru antrenarea modelelor *multi-speaker*: pentru cele 1.397 de propoziții comune, modelul aude același enunț rostit de toți cei 41 de vorbitori, ceea ce facilitează separarea identității vocale de conținutul lingvistic. Propozițiile exclusive fiecărei versiuni extind în schimb acoperirea fonetică globală a corpusului. O direcție de explorat în cercetările ulterioare ar fi antrenarea exclusiv pe subsetul paralel strict de 1.397 de propoziții comune, pentru a evalua impactul paralelismului complet asupra calității embedding-urilor de vorbitor.

2.5.1. Organizarea, preprocesarea și adaptarea datelor

Corpusul disponibil pe serverul facultății conține date preprocesate din ambele versiuni SWARA, organizate în fișiere text cu format pipe-delimited, compatibil cu framework-ul Coqui TTS [10]:

```
cale_fisier_audio|text_transcris|id_vorbitor
SWARA1.0_22k_noSil/sam_rnd2_216.wav|buna ziua|0
```

Sufixul `noSil` din denumirea folderelor indică eliminarea segmentelor de silență realizată în cadrul preprocesării originale. Frecvența de eșantionare utilizată în experimente este de 22.050 Hz, compatibilă cu cerințele arhitecturii VITS, un standard curent în sinteza neurală a vorbirii [25]. Distribuția corpusului în partiții de antrenare, validare și testare este prezentată în Tabelul 2.4.

Tabelul 2.4. Distribuția corpusului SWARA utilizat în experimente

Partiție	Număr exemple	Utilizare
Train	55.157	Antrenarea modelului
Validation	3.063	Monitorizarea antrenării
Test	3.064	Evaluarea finală
TOTAL	**61.284**	**41 vorbitori unici**

Preprocesarea originală a corpusului a inclus mai multe etape esențiale pentru asigurarea calității datelor de antrenare [23]. Segmentarea a fost realizată manual la nivel de enunț (*utterance-level*), asigurând o corespondență precisă între semnalul audio și transcrierea textului. Normalizarea textului a acoperit expandarea abrevierilor (ex. „dl.” devine „domnul”) și conversia numerelor în cuvinte (ex. „15” devine „cincisprezece”), asigurând o reprezentare fonetică unitară [23]. Curățarea semnalului audio a presupus eliminarea segmentelor de silență excesivă, menținând aproximativ 200 ms la începutul și sfârșitul fiecărui enunț pentru a optimiza convergența modelului [23]. Datele au fost structurate conform standardului LJSpeech, asigurând compatibilitatea cu framework-ul Coqui TTS.

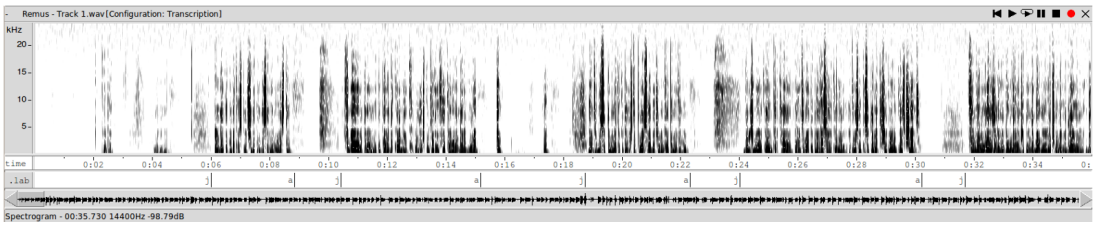


Figura 2.3. Exemplu al procesului de adnotare la nivel de enunț; enunțurile corecte sunt marcate cu „a” (audio), iar restul cu „j” (junk) [23].

O provocare tehnică a constituit-o adaptarea vocabularului de caractere al framework-ului Coqui TTS [10] pentru limba română, o problematică identificată și în studiile recente de evaluare a uneltelor TTS open-source pentru această limbă [22]. Vocabularul implicit este configurat pentru limba engleză și nu include diacriticele specifice limbii române. Soluția implementată a constat în definirea explicită a unui obiect `Graphemes` personalizat:

```
ro_characters = Graphemes(
    characters="aăâbcdefghiîjklmnopqrsștțuvwxyz",
    punctuations=" ",
    pad="_", eos="~", bos="^"
)
config.characters = ro_characters.to_config()
```

Această abordare a rezolvat problema caracterelor ignorate în timpul antrenării, verificată prin absența erorilor de tip `Character not found in the vocabulary` pentru diacriticele *ă, â, î, ș, ț*. Vocabularul rezultat conține **36 de caractere unice**, acoperind complet fonemele grafematice ale limbii române standard.

O alternativă la reprezentarea grafematică constă în utilizarea reprezentării fonetice, prin intermediul fonetizatorului **eSpeak-NG** (versiunea 1.52.0.1) [6] și al bibliotecii `phonemizer` (versiunea 3.3.0) [1]. În acest caz, vocabularul este generat automat din fonemele IPA produse pentru limba română (`ro`), rezultând **132 de simboluri fonetice** față de 36 în varianta grafematică. Fonemele sunt precalculate și stocate în cache înaintea antrenării, eliminând overhead-ul de fonetizare din bucla de antrenare.

Spre deosebire de reprezentarea grafematică, unde fiecare caracter corespunde unui simbol din vocabular, reprezentarea fonetică captează direct sunetele limbii, independent de ortografie. De exemplu, secvența grafematică *„bună ziua”* este convertită în transcrierea IPA /bˈunə ˈziwa/, unde ə reprezintă o vocală redusă absentă din alfabetul grafematic standard.

2.6. Experimentele de antrenare — Hyperparameter Search

2.6.1. Strategia experimentală și parametrii ficși

Pentru identificarea configurației optime de antrenare, a fost adoptată o strategie de **hyperparameter search** sistematic. Această abordare a permis compararea mai multor configurații pe un număr de pași globali suficient pentru a observa tendința de convergență, urmată de continuarea antrenării și specializarea prin fine-tuning pe configurația selectată.

Anumiți parametri au fost menținuți ficși pe durata tuturor experimentelor, fie din constrângeri tehnice (limita VRAM a plăcii RTX 4090), fie din cerințele corpusului SWARA, aceștia fiind prezentați în Tabelul 2.5. Reprezentarea lingvistică a textului variază între configurații: configurațiile A, B și C utilizează caractere grafematice (vocabular de 36 de simboluri), în timp ce Configurația D testează reprezentarea fonetică IPA prin fonetizatorul eSpeak-NG (vocabular de 132 de simboluri).

Tabelul 2.5. Parametrii ficși ai experimentelor

Parametru	Valoare	Motivație
mixed_precision	True (fp16)	Optimizat cu TF32; ∼20% mai rapid decât fp32.
sample_rate	22.050 Hz	Dictat de preprocesarea originală a corpusului SWARA.
use_speaker_embedding	True	Necesar pentru modelarea celor 41 de vorbitori unici.
hidden_channels	192	Valoare implicită VITS; reducerea la 128 constituie o direcție de cercetare ulterioară.
save_step	3.447	Corespunde unei salvări per epocă.

O modificare arhitecturală potențial benefică, neinclusă în experimente pentru a menține comparabilitatea configurațiilor, constă în reducerea dimensiunii hidden_channels de la valoarea implicită de 192 la 128. Această modificare ar reduce capacitatea modelului și ar putea acționa ca o formă de regularizare structurală, limitând tendința de suprainvățare și favorizând reprezentări latente mai compacte. În special în situațiile în care dimensiunea corpusului nu justifică pe deplin capacitatea modelului de 86M parametri, o astfel de reducere ar putea îmbunătăți generalizarea. Această modificare constituie o direcție de cercetare ulterioară.

2.6.2. Configurațiile experimentale și preprocesarea avansată

Experimentele au vizat trei direcții de optimizare: dimensiunea batch-ului, calitatea acustică a datelor și reprezentarea lingvistică. **Dimensiunea batch-ului** (batch_size) influențează atât stabilitatea antrenării, cât și cerințele de memorie VRAM. Valorile candidate sunt 16 (conservativ, ∼10 GB VRAM) și 32 (convergență potențial mai rapidă, ∼14–16 GB VRAM). O valoare de 64 ar depăși capacitatea GPU-ului disponibil (24 GB VRAM), generând erori de tip *Out of Memory*, motiv pentru care a fost exclusă. Cele patru configurații principale sunt detaliate în Tabelul 2.6, fiecare izolând o singură variabilă față de baseline-ul reprezentat de Configurația A.

Tabelul 2.6. Configurațiile hyperparameter search

Config	Batch	Group	Date	Variabila testată
A	16	5	Orig.	Baseline (grafeme)
B	32	5	Orig.	Impactul creșterii dimensiunii batch-ului
C	32	5	Filtr.	Impactul curățării audio și filtrării textului
D	32	5	Filtr.	Impactul fonetizării IPA (eSpeak-NG)

Configurația A (`batch_size`=16, date originale) reprezintă baseline-ul experimentelor, utilizând datele SWARA exact cum au fost puse la dispoziție pe server.

Configurația B (`batch_size`=32, date originale) izolează efectul dimensiunii batch-ului față de Config A, păstrând același corpus de intrare.

Configurația C (`batch_size`=32, date preprocesate) introduce două intervenții suplimentare asupra corpusului față de Config B. Prima constă în **filtrarea zgomotului din SWARA 2.0**: înregistrările casnice prezintă un nivel variabil de zgomot de fond — staticit, reverberație, zgomot ambiental — care poate degrada calitatea modelului antrenat. A fost aplicat un algoritm de reducere a zgomotului bazat pe subtracție spectrală, implementat în Python cu biblioteca `librosa`. Algoritmul estimează profilul de zgomot din primele și ultimele 500 ms ale fiecărui fișier audio și aplică un câștig spectral adaptiv cu un *gain floor* de 5% pentru a evita efectul de „muzical noise”. Un filtru trece-jos la 9.500 Hz elimină zgomotul de înaltă frecvență fără a afecta vocea umană (situată sub 8.000 Hz). Scriptul a fost aplicat recursiv pe folderul `SWARA2.0_22k_noSil`, generând `SWARA2.0_22k_noSil_denoised` fără modificarea datelor originale:

```
python3 denoise_advanced_final.py ~/swara-dataset/SWARA2.0_22k_noSil \
    ~/swara-dataset/SWARA2.0_22k_noSil_denoised
```

A doua intervenție constă în **excluderea enunțurilor scurte**: corpusul conține un număr mic de înregistrări cu text de 1–2 cuvinte (ex. „da”, „nu știu”), care contribuie marginal la învățarea alinierii text-audio și pot introduce instabilitate în predictorul de durată. A fost implementat un filtru în formatter-ul de date care exclude enunțurile cu mai puțin de 3 cuvinte:

```
if len(cols) < 3 or len(cols[1].split()) < 3:
    continue
```

Au fost excluse **403 enunțuri** (0,73% din date) — o pierdere neglijabilă în raport cu beneficiul obținut. Datele SWARA 1.0, înregistrate în studio profesional, nu au necesitat niciuna dintre cele două intervenții descrise mai sus.

Configurația D (`batch_size`=32, date preprocesate, foneme) extinde Configurația C prin înlocuirea reprezentării grafematice cu cea fonetică IPA, utilizând fonetizatorul eSpeak-NG

descris în Secțiunea 2.5. Această configurație testează dacă reprezentarea fonetică facilitează o convergență mai bună prin captarea mai precisă a fenomenelor fonetice ale limbii române față de cele 36 de caractere grafematice din configurațiile anterioare.

2.6.3. Analiza rezultatelor și monitorizarea antrenării

Antrenarea s-a desfășurat în sesiuni `tmux` pe serverul `rtx-4090-01`. Progresul a fost monitorizat prin analiza metricilor afișate la fiecare 25 de pași: **loss_mel** — eroarea de reconstrucție a mel-spectrogramei; **loss_duration** — eroarea de aliniere text-audio; **loss_kl** — divergența KL a spațiului latent. Modelele VITS *multi-speaker* ating maturitatea acustică în intervalul 500.000–800.000 de pași globali [18], toate configurațiile aflându-se în această zonă la finalul antrenării.

Rezultatele după convergență sunt prezentate în Tabelul 2.7, iar evoluția funcției de pierdere `avg_loss_mel` este ilustrată în Figura 2.4.

Tabelul 2.7. Rezultatele finale ale antrenării generale

Metrică	**Config A**	**Config B**	**Config C**	**Config D**
Epoci antrenate	209	435	400	400
Pași globali	~720k	~755k	~686k	~683k
avg_loss_mel final	20,36	19,45	**19,10**	20,24
step_time (medie)	~0,21 s	~0,24 s	~0,24 s	~0,25 s

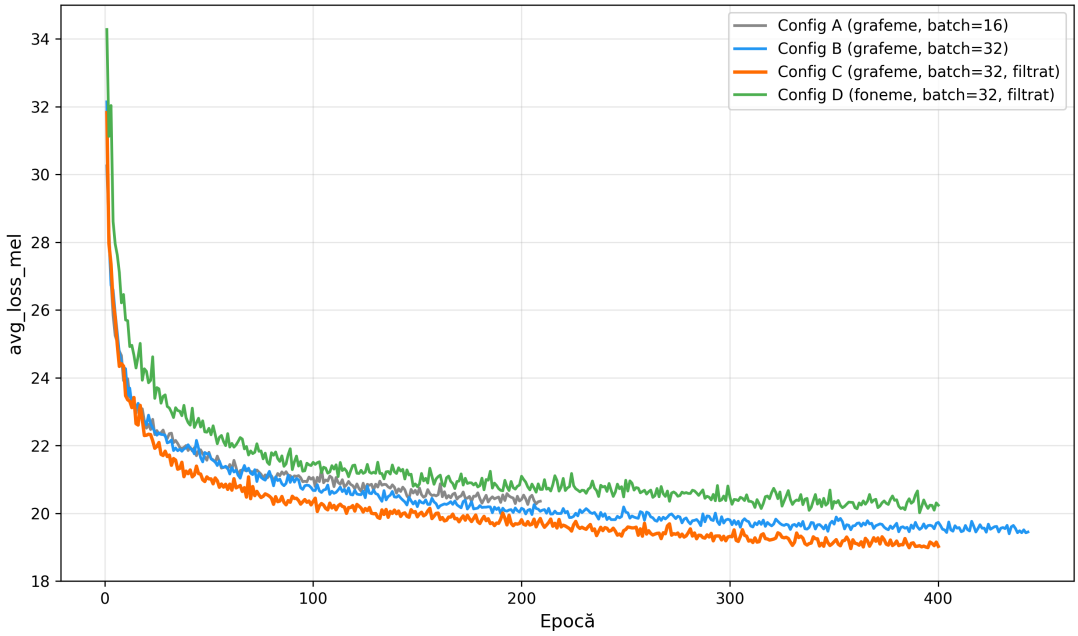


Figura 2.4. Evoluția funcției de pierdere avg_loss_mel pentru configurațiile A, B, C și D pe durata antrenării.

Configurația C a obținut cea mai mică valoare a funcției de pierdere (**19,10**), confirmând că preprocesarea riguroasă a datelor zgomotoase din SWARA 2.0 are un impact superior față de simpla schimbare a reprezentării lingvistice. Configurația D, deși utilizează reprezentarea fonetică IPA, nu depășește Configurația C (20,24 față de 19,10), sugerând că un vocabular fonetic extins necesită un număr mai mare de pași de antrenare pentru convergență completă.

2.6.4. Fine-tuning pe vorbitori specifici

Pornind de la modelele generale, s-a realizat etapa de fine-tuning pentru specializarea pe vorbitori individuali. S-a utilizat un learning rate redus ($lr = 5 \times 10^{-5}$, față de 2×10^{-4} în antrenarea generală) și medierea ultimelor 5 checkpointuri (*checkpoint averaging*) pentru stabilitate, conform metodologiei descrise în Secțiunea 2.7.

Au fost realizate trei experimente inițiale de fine-tuning, prezentate în Tabelul 2.8. Primul vizează specializarea pe **speaker_14** din corpusul SWARA 1.0, un vorbitor de studio cu înregistrări de înaltă fidelitate. Celelalte două experimente testează adaptarea la un **vorbitor extern** corpusului SWARA, cu aproximativ 97 de minute de înregistrări de înaltă calitate, pornind din două modele generale diferite (ConfigC și ConfigD).

Tabelul 2.8. Rezultatele experimentelor de fine-tuning

Experiment	Model bază	Date	Epoci FT	Loss train	Loss eval	Reprezentare
FT speaker_14	ConfigB ep.315	1.419 ex. (~72 min)	184	17,71	17,39	Grafeme
FT vorbitor ext. (C)	ConfigC avg.	1.449 ex. (~97 min)	140	17,11	**16,34**	Grafeme
FT vorbitor ext. (D)	ConfigD avg.	1.449 ex. (~97 min)	140	18,15	17,23	Grafeme
FT vorbitor ext. (D foneme)	ConfigD avg.	1.449 ex. (~97 min)	280	17,99	—	Foneme IPA
FT speaker_14 (D foneme)	ConfigD avg.	1.419 ex. (~68 min)	268	17,89	—	Foneme IPA

Se remarcă faptul că experimentele cu reprezentare fonetică IPA au necesitat un număr semnificativ mai mare de pași de antrenare față de variantele grafematice: aproximativ **24.000 de pași globali** față de **7.000** pentru fine-tuning-ul grafematic pe un volum similar de date. Această diferență reflectă complexitatea mai ridicată a spațiului de optimizare fonetic — modelul trebuie să reajusteze maparea fonemelor IPA la caracteristicile acustice ale vorbitorului țintă, față de simpla adaptare a reprezentării grafematice.

Fine-tuning-ul pe speaker_14 a condus la o reducere a loss-ului de la 18,72 (modelul general la epoca 315) la 17,71, confirmând eficiența abordării cu un volum redus de date (~72 de minute). Fine-tuning-ul pe vorbitorul extern (~97 de minute de înregistrări de înaltă calitate) a produs cel mai bun rezultat din seria de experimente (eval loss **16,34** pornind din ConfigC), demonstrând că un corpus de fine-tuning de calitate superioară poate compensa volumul redus de date. Comparând cele două experimente pe vorbitorul extern, ConfigC averaged oferă un punct de pornire mai bun față de ConfigD averaged, consistent cu rezultatele superioare ale ConfigC în etapa de antrenare generală.

Investigarea fine-tuning-ului cu reprezentare fonetică IPA.

O direcție suplimentară de cercetare a vizat testarea fine-tuning-ului cu **reprezentare fonetică IPA** pornind din ConfigD, motivată de ipoteza că un model antrenat cu foneme ar putea generaliza mai bine la vorbitori noi. Primele experimente pe vorbitorul extern au produs însă rezultate nesatisfăcătoare: vorbirea sintetizată prezenta artefacte prozodice severe, semănând cu grupuri de foneme aruncate haotic, fără coerență recognoscibilă. Au fost investigate mai multe variante pentru a identifica sursa problemei:

- Fine-tuning cu `use_phonemes=False` pornind din ConfigD (grafeme suprapuse pe model fonetic) — vorbire inteligibilă dar cu WER ridicat (72,6%), cauzat de incompatibilitatea dintre vocabularul grafematic al fine-tuning-ului și cel fonetic IPA al modelului general
- Fine-tuning cu `use_phonemes=True` și `phoneme_language="ro"` pornind direct din ConfigD averaged (foneme end-to-end) — vorbire neinteligibilă, cu artefacte prozodice severe
- Variante cu learning rate diferit și număr variat de epoci — fără îmbunătățiri semnificative

Analiza acestor rezultate a condus la ipoteza că problema nu ține de configurarea fine-tuning-ului, ci de natura vorbitorului țintă. Modelul general ConfigD a calibrat spațiul latent fonetic exclusiv pe cei 41 de vorbitori SWARA. Introducerea unui **vorbitor extern** impune simultan

două sarcini incompatibile: adaptarea la o nouă identitate vocală și menținerea consistenței fonetice IPA, rezultând în degradarea spațiului latent.

Ipoteza a fost validată printr-un experiment de control pe **speaker_14**, vorbitor intern SWARA cu ~68 de minute de înregistrări de studio, al cărui profil vocal fusese inclus în antrenarea generală ConfigD. Fine-tuning-ul fonetic pe speaker_14 a produs vorbire inteligibilă, confirmând că pentru un vorbitor cunoscut modelul rafinează o reprezentare existentă, în loc să creeze una nouă de la zero.

Întrucât fine-tuning-ul fonetic pe vorbitorul extern nu a produs rezultate utilizabile, etapele ulterioare de evaluare și optimizare s-au concentrat pe modelele **FT vorbitor ext. (ConfigC)** și **FT vorbitor ext. (ConfigD)**, care au demonstrat cea mai bună combinație între inteligibilitate și fidelitate vocală. De remarcat că experimentele cu foneme IPA au necesitat un număr semnificativ mai mare de pași de antrenare (~24.000 față de ~7.000 pentru varianta grafematică), datorită complexității mai ridicate a spațiului de optimizare fonetic față de cel grafematic.

2.7. Evaluarea sistemului de sinteză vocală

Evaluarea performanței sistemului de sinteză vocală se realizează printr-o abordare obiectivă, utilizând metrici matematice și algoritmi de procesare a semnalelor audio. Această etapă validează capacitatea modelului de a genera vorbire inteligibilă, de a păstra identitatea vocală a vorbitorului țintă și de a identifica erorile sistematice de pronunție, inclusiv problemele de tip *Out-of-Vocabulary (OOV)*.

2.7.1. Metodologia de evaluare și metrici utilizate

Înaintea etapei de evaluare, modelele antrenate sunt supuse unui proces de **mediere a checkpoint-urilor** (*checkpoint averaging*). Tehnica constă în calculul mediei aritmetice a greutăților ultimelor $k = 5$ checkpoint-uri salvate, producând un model final mai stabil față de variațiile stochastice din ultimele epoci:

$$\theta_{avg} = \frac{1}{k} \sum_{i=1}^{k} \theta_i$$

Pentru cuantificarea calității sintezei și a fidelității identității vocale, sunt utilizați următorii indicatori principali:

– **Word Error Rate (WER):** Măsoară inteligibilitatea vorbirii generate prin transcrierea automată a eșantioanelor cu modelul ASR **Whisper `medium`** [20]. Este calculat ca:

$$\text{WER} = \frac{S + D + I}{N}$$

unde S = substituții, D = ștergeri, I = inserții, N = numărul total de cuvinte de referință. Se aplică o etapă de **normalizare a textului** prin biblioteca `num2words` înainte de calcul, pentru eliminarea penalizărilor artificiale introduse de conversia numeralelor în cifre arabe

de către modelul ASR. De exemplu, enunțul de referință „Nota de plată se ridică la o sută douăzeci și trei de lei" este transcris de Whisper ca „Nota de plată se ridică la 123 lei", generând erori care nu reflectă nicio deficiență reală de articulare a modelului TTS. Normalizarea elimină această clasă de erori prin conversia reciprocă a cifrelor în cuvinte înaintea comparației. Suplimentar, semnele de punctuație sunt eliminate din ambele texte anterior calculului, deoarece Whisper generează frecvent virgule și puncte în transcriere, introducând penalizări false la nivel de token.

– **Character Error Rate (CER):** Oferă granularitate mai fină față de WER, operând la nivel de caracter:

$$\text{CER} = \frac{S_c + D_c + I_c}{N_c}$$

CER este deosebit de relevant pentru identificarea erorilor de lipit cuvinte adiacente (*word boundary errors*), frecvente în modelele antrenate pe corpusuri cu variabilitate fonetică limitată.

– **Real-Time Factor (RTF):** Raportul dintre timpul de procesare și durata semnalului audio generat:

$$\text{RTF} = \frac{t_{proc}}{t_{audio}}$$

Un RTF < 1,0 indică capacitate de sinteză în timp real, esențială pentru aplicațiile de tip chatbot.

– **Similaritatea vorbitorului:** Evaluată prin modelul **ECAPA-TDNN** [5] din biblioteca `SpeechBrain` [21], reprezintă similaritatea cosinus dintre embedding-urile vocale ale eșantioanelor sintetizate și cele reale. Scorul măsoară fidelitatea „amprentei" vocale, unde o valoare mai apropiată de 1,0 indică o identitate vocală bine conservată.

2.7.2. Rezultatele evaluării

O evaluare preliminară pe 10 enunțuri a confirmat superioritatea ConfigC față de ConfigD (WER normalizat 19,5% față de 27,8%), motivând extinderea la un set de 100 de enunțuri structurate pe șase categorii de complexitate. Evaluarea finală s-a realizat pe acest set extins, acoperind atât scenarii *in-domain* (HoReCa) cât și enunțuri de română generală (*out-of-domain*). Distribuția setului este detaliată în Tabelul 2.9.

Tabelul 2.9. Distribuția setului de evaluare (100 enunțuri)

Categorie	**Nr. enunțuri**	**Caracteristici**
Rezervări simple	15	1–2 sloturi, propoziții scurte
Rezervări complexe	15	Dată, oră, locație, confirmare
Comenzi pizza	20	Nume proprii italiene, mărimi
Prețuri și numere	10	Sume în lei, procente, ore
Formule de politețe	20	Confirmări, salutări, răspunsuri sistem
Română generală	20	Out-of-domain, diacritice, numere
TOTAL	**100**	

Rezultatele globale sunt prezentate în Tabelul 2.10, iar distribuția per categorie în Tabelul 2.11.

Tabelul 2.10. Rezultatele globale ale evaluării — modele originale vs. augmentate. Valorile WER/CER pentru modelele augmentate sunt calculate cu normalizarea inițială; valorile corectate sunt prezentate în Tabelul 2.15.

Model	**WER norm. (%)**	**CER norm. (%)**	**RTF**	**Sim. vorbitor**
FT vorbitor ext. (ConfigC)	32,8	10,4	0,028	**0,740**
FT vorbitor ext. (ConfigD)	72,6	29,5	0,026	0,683
FT ConfigC augmentat	**19,3**	**5,0**	0,030	0,521
FT ConfigD augmentat	21,2	5,8	0,031	0,513

Tabelul 2.11. WER și CER normalizat per categorie — toate modelele. Valorile coloanei FT-C-aug utilizează normalizarea inițială; valoarea corectată WER pizza este 29,7% (Tabelul 2.15).

Categorie	FT-C		FT-D		FT-C-aug		FT-D-aug	
	WER	**CER**	**WER**	**CER**	**WER**	**CER**	**WER**	**CER**
Rezervări simple	21,8	4,4	64,4	23,0	**16,0**	**3,0**	16,8	4,4
Rezervări complexe	17,4	4,7	68,8	26,3	**10,8**	**2,7**	17,2	4,4
Comenzi pizza	74,6	26,3	90,4	39,3	**39,9**	**9,4**	41,5	9,3
Prețuri și numere	23,7	7,5	68,1	26,6	**15,5**	**4,8**	23,2	8,4
Formule de politețe	22,6	7,0	64,2	27,0	15,1	4,6	**11,2**	**3,9**
Română generală	25,6	8,3	74,2	30,9	**14,0**	**4,4**	16,2	4,8

Analiza rezultatelor evidențiază două concluzii principale. **ConfigD fără augmentare** înregistrează performanțe dramatic inferioare față de ConfigC (WER 72,6% față de 32,8%), cu zero enunțuri perfect transcrise, sugerând incompatibilități introduse de fine-tuning-ul grafematic pornind dintr-un model antrenat cu foneme. **Augmentarea datelor**, constând în adăugarea a 90 de eșantioane sintetice ElevenLabs care acoperă terminologia italiană OOV și formule de rezervare cu vocabular rar, conduce la îmbunătățiri semnificative pe toate categoriile, cel mai spectaculos progres înregistrându-se la comenzile de pizza, unde WER scade de la 74,6% la **29,7%** pentru FT-C-aug (valoare obținută cu normalizarea corectă, detaliată în Secțiunea 2.7.3).

Totuși, augmentarea introduce un compromis între inteligibilitate și fidelitatea vocală. Scorul de similaritate al vorbitorului scade de la **0,740** (FT-C original) la **0,521** (FT-C-aug), indicând că adăugarea datelor sintetice ElevenLabs a afectat „amprenta" vocală a modelului. Această degradare se datorează discrepanței spectrale dintre înregistrările reale ale vorbitorului extern (realizate în studio profesional) și eșantioanele sintetice ElevenLabs, care prezintă caracteristici acustice diferite (silențe digitale, timbru ușor diferit). Deși preprocesarea (trimming + fade in/out) a redus parțial această discrepanță, efectul rămâne perceptibil la nivel de embedding vocal.

Analiza per categorie evidențiază trei surse sistematice de eroare:

a) **Problema Out-of-Vocabulary (OOV):** Categoria comenzilor de pizza înregistrează cel mai ridicat WER (74,6% pentru FT-C), cauzat de pronunția deficitară a denumirilor italiene — *Diavola*, *Margherita*, *Carnivora*, *Quattro Formaggi* — absente din vocabularul corpusului SWARA. Modelul ASR Whisper transcrie aceste cuvinte în forme eronate (ex. „Quattro Formaggi" → „toatro formăcii"), amplificând WER-ul raportat față de eroarea perceptuală reală. Strategiile de corecție prin dicționar fonetic sunt tratate în detaliu în Secțiunea 2.7.3.

b) **Atacurile consonantice inițiale rare:** Cuvinte precum „Rezervarea" sunt transcrise

sistematic ca „Ezervarea” sau „Prezervarea”, indicând o acoperire fonetică insuficientă a consoanelor inițiale rare în corpusul SWARA.

c) **Lipirea cuvintelor adiacente:** Modelul tinde să genereze secvențe fără pauze clare între cuvinte (ex. „Diavola mare” → „Diavolamare”), penalizând WER-ul disproporționat față de eroarea perceptuală reală. CER-ul de **5,0%** pentru FT-C-aug confirmă că, la nivel de caracter, modelul reproduce corect aproximativ 95% din foneme.

Pe categoriile fără vocabular specializat, CER-ul modelului augmentat scade la **3,0–4,6%**, comparabil cu literatura de specialitate pentru modele antrenate în condiții similare.

Rezultatele obținute pot fi contextualizate în raport cu lucrarea înrudită a lui Răgman et al. [22], care raportează valori WER de 2–6% și scoruri SECS de până la 0,92 pentru VITS antrenat pe corpusul SWARA 1.0. Comparația directă nu este echitabilă: evaluarea din lucrarea de față utilizează enunțuri *out-of-domain* (HoReCa), un vorbitor extern necunoscut modelului general și un corpus mai divers dar mai zgomotos (SWARA 1.0 + 2.0, 41 de vorbitori). Scenariul adoptat este mai complex și mai reprezentativ pentru condițiile reale de deployment ale unui sistem conversațional.

Augmentarea datelor de fine-tuning cu eșantioane sintetice

Pentru adresarea problemei OOV, datele de fine-tuning au fost augmentate cu **90 de eșantioane sintetice** generate prin platforma **ElevenLabs** (*Professional Voice Clone*), utilizând vocea vorbitorului extern ca sursă de clonare. Eșantioanele acoperă explicit categoriile cu cele mai ridicate rate de eroare: denumirile de pizza (*Diavola*, *Margherita*, *Carnivora*, *Quattro Formaggi*), formule de rezervare și formule de politețe cu vocabular rar.

Fișierele audio generate au fost preprocesate în trei etape: conversia din MP3 la WAV mono la 22.050 Hz; aplicarea unui algoritm de **trimming** automat (prag $\epsilon = 10^{-3}$) pentru eliminarea silențelor digitale de la capete, care introduceau instabilitate de fază în vocoderul HiFi-GAN; și aplicarea unui **fade in/out de 10 ms** pentru eliminarea tranziției bruște la zero. Datele augmentate (~6 minute) au fost adăugate la corpusul original (~97 minute), rezultând **1.466 de perechi audio-text**. Fine-tuning-ul augmentat pornește din averaged_model ConfigC, cu $lr = 5 \times 10^{-5}$, `batch_size`=16, 300 epoci, loss final **16,59**.

Îmbunătățirea nu este liniară cu cantitatea datelor augmentate: primele 90 de eșantioane aduc un câștig semnificativ, dar adăugarea unui volum similar suplimentar produce randamente descrescătoare, efect tipic când datele sintetice dintr-un model comercial sunt adăugate într-un volum limitat. Suplimentar, augmentarea a redus și frecvența erorilor de tip atac consonantic inițial (ex. „Rezervarea” transcris ca „Ezervarea”), categorie identificată ca sursă sistematică de erori în modelele neaugmentate.

Analiza proiecțiilor t-SNE

Validarea spațiului latent s-a realizat prin proiecții **t-SNE** (*t-distributed Stochastic Neighbor Embedding*), care reduc dimensionalitatea embedding-urilor de vorbitor de la un vector de 256 de dimensiuni la o reprezentare plană 2D.

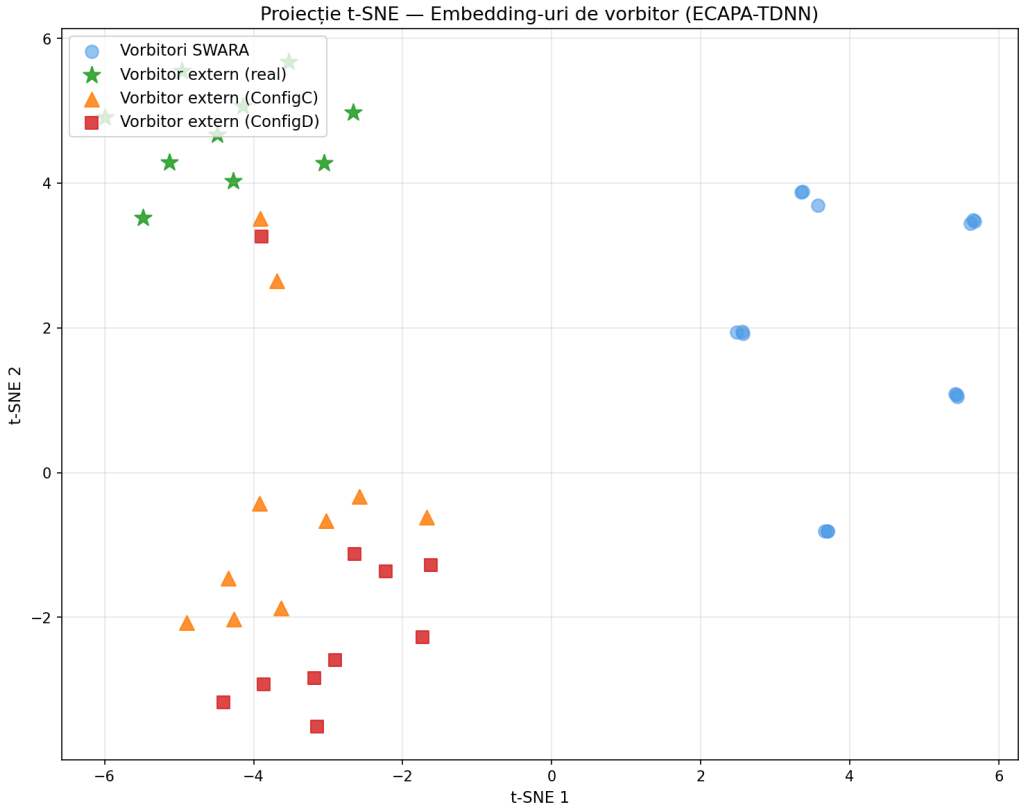


Figura 2.5. Proiecție t-SNE a embedding-urilor de vorbitor (ECAPA-TDNN): distincția între vorbitorii SWARA, vorbitorul extern real și eșantioanele sintetizate prin modelele augmentate.

Figura 2.5 evidențiază că embedding-urile vorbitorului extern real (verde) formează un cluster distinct față de vorbitorii SWARA (albastru), confirmând separarea identităților vocale. Eșantioanele sintetizate prin modelele augmentate ConfigC (portocaliu) și ConfigD (roșu) se poziționează în aceeași regiune cu înregistrările reale, confirmând că fine-tuning-ul a capturat caracteristicile vocale individuale. Dispersia mai pronunțată față de modelele neaugmentate este consistentă cu scorurile de similaritate obținute (0,521 față de 0,513), reflectând compromisul dintre inteligibilitatea îmbunătățită și fidelitatea vocală introdus de datele sintetice ElevenLabs.

2.7.3. Corecția pronunției termenilor din vocabularul de specialitate italian

O categorie distinctă de erori identificată în evaluarea cantitativă o reprezintă pronunția incorectă a termenilor din vocabularul de specialitate de origine italiană utilizați în meniul restaurantului: *Margherita*, *Diavola*, *Carnivora*, *Quattro Formaggi* și *Capricciosa*. Acești termeni constituie cuvinte împrumutate (out-of-vocabulary) din perspectiva fonetică a limbii române, nefiind prezente în corpusul de antrenare SWARA. Analiza erorilor pe categoria *Comenzi pizza* a evidențiat o rată WER normalizat de 29,7% pentru modelul FT-C-aug fără preprocesare, semnificativ mai ridicată față de celelalte categorii evaluate (11,3%–17,8%), confirmând că termenii OOV italieni reprezintă principala sursă de erori în domeniu.

Au fost identificate două tipare de erori distincte. Primul tipar constă în erori de pronunție propriu-zise, în care modelul generează audio incorect fonetic pentru termenul respectiv (de exemplu, *Margherita* sintetizat ca „margirita"). Al doilea tipar este fenomenul de lipire a cuvintelor (*word-sticking*), în care termenul italian și adjectivul de dimensiune adiacent sunt redați fără pauză naturală, producând un token audio unificat (de exemplu, „Diavola mică" sintetizat ca „diavolamică"). Cele două tipuri de erori contribuie diferit la rata WER: o eroare de pronunție generează o substituție de cuvânt, în timp ce o eroare de lipire generează o substituție și o ștergere simultan, dublând penalizarea metrică per ocurență.

a) Dicționar fonetic pentru modelul grafemic.

Strategia de corecție explorată a urmat abordarea propusă pentru sistemele TTS cu vocabular extins din literatura de specialitate, constând în construirea unui dicționar de substituție text care mapează termenul original la o transcriere fonetică aproximativă în limba română, aplicată înaintea apelului la modelul de sinteză. Această abordare este eficientă în sistemele TTS clasice (rule-based), unde textul trece printr-un fonetizator explicit înainte de etapa de sinteză acustică, fonetizatorul procesând forma substituită și producând secvențe fonetice corecte pentru modelul acustic.

În cazul modelelor neuronale end-to-end de tip VITS, textul este procesat direct de un encoder antrenat pe distribuțiile statistice ale textului natural. Introducerea unor secvențe artificiale cu spații interioare (denumite în continuare substituții *split-word*), precum „marghe rita" în locul lui „Margherita", creează bigramuri și trigramuri absente din datele de antrenare, determinând instabilitate în procesul de decodificare. Experimental, s-a constatat că această strategie nu numai că nu îmbunătățește pronunția termenului respectiv, ci produce artefacte audio care se propagă și asupra unităților lingvistice adiacente în propoziție. De exemplu, transcrierea Whisper a produsului TTS pentru propoziția „Vreau o Diavola mare și două Margherita mici, vă rog" a generat ipoteza „Vreau o diavol la mare si doua mari de rita mici, va rog", în care inclusiv termenul *Diavola*, neatins de dicționar, a fost afectat de instabilitatea introdusă de secvența artificială adiacentă. Rata WER pe categoria *Comenzi pizza* a crescut de la 29,7% la 45,9% prin aplicarea substituțiilor de tip split-word, confirmând incompatibilitatea acestei strategii cu arhitectura VITS end-to-end.

S-a investigat în continuare o strategie alternativă bazată pe substituții cu un singur token (fără spații artificiale), în care termenul italian este înlocuit cu o aproximare grafică formată dintr-un singur cuvânt care respectă regulile fonetice ale limbii române. Variantele candidate au fost evaluate perceptual prin generarea de probe audio și ascultare directă, comparând diferite forme pentru fiecare termen. În urma acestei calibrări empirice, au fost selectate substituțiile prezentate în Tabelul 2.12.

Tabelul 2.12. Dicționar fonetic pentru modelul grafemic FT-C-aug. Termenii *Diavola* și *Carnivora* nu necesită substituție, evaluarea perceptuală indicând pronunție acceptabilă la nivel fonetic.

Termen original	**Substituție fonetică**
Margherita	Margheritta
Quattro Formaggi	Cuatro Formaggi
Quattro	Cuatro
Capricciosa	Caprícioza

Termenii *Diavola* și *Carnivora* nu au necesitat substituție, evaluarea perceptuală indicând că modelul îi pronunță acceptabil la nivel fonetic în context izolat; erorile lor provin în principal din fenomenul de word-sticking, tratat separat. Aplicarea dicționarului cu substituții single-token a condus la o îmbunătățire moderată a ratei WER pe categoria *Comenzi pizza*: de la 29,7% la 26,5% (−3,2 puncte procentuale), fără a afecta negativ celelalte categorii evaluate. WER global normalizat a scăzut marginal de la 14,9% la 14,3%.

b) Tentativele de corecție a fenomenului de word-sticking.

Fenomenul de word-sticking, prin care termenul italian și adjectivul de dimensiune adiacent sunt redați ca un singur token audio, a reprezentat o sursă persistentă de erori în categoria *Comenzi pizza*. S-au investigat mai multe strategii de inserție a unei pauze prosodice prin introducerea de semne de punctuație în textul de intrare, anterior apelului la modelul TTS.

Inserția de virgulă (substituția „Diavola mică” → „Diavola, mică”) a produs o creștere a ratei WER la 45,9% pe categoria pizza, iar analiza ipotezelor Whisper a evidențiat destabilizarea pronunției termenilor adiacenți. Inserția de cratimă (substituția „Diavola mică” → „Diavola - mică”) a produs o creștere și mai pronunțată, la 48,0% WER, cu artefacte de tipul „diavolat amare” și „carni vor a amici”. Tabelul 2.13 centralizează rezultatele tuturor strategiilor investigate.

Tabelul 2.13. Comparație între strategiile de corecție a pronunției OOV și baseline-ul fără dicționar, investigate pe modelul grafemic FT-C-aug. Simbolul “–” indică valori neraportate separat, deteriorate global față de baseline.

Configurație	**WER global (%)**	**WER pizza (%)**
FT-C-aug fără dicționar	14,9	29,7
FT-C-aug cu dict. split-word	–	45,9
FT-C-aug cu dict. single-token	14,3	26,5
FT-C-aug cu dict. + virgulă	–	45,9
FT-C-aug cu dict. + cratimă	–	48,0

În ambele cazuri, caracterele de punctuație introduse au fost procesate direct de encoderul VITS, generând secvențe token nefamiliare modelului și producând efecte similare cu cele observate la substituțiile split-word. Acest comportament este în contrast cu cel al sistemelor TTS clasice, unde semnele de punctuație sunt tratate explicit de modulul de analiză prosodică anterior sintezei. Fenomenul de word-sticking a rămas o limitare neremediat prin preprocesare text pentru modelul grafemic, și este raportat ca atare în rezultatele finale.

c) Dicționar fonetic pentru modelul fonemic.

În cazul modelului sp14-D-aug, bazat pe foneme IPA generate de eSpeak-NG, strategia de corecție se aplică anterior fonetizatorului, permițând substituirea termenilor problematici cu forme grafice pe care eSpeak-NG le fonetizează corect în contextul regulilor fonetice ale limbii române. Analiza fonetizării automate a termenilor italieni cu eSpeak-NG în configurație pentru limba română (opțiunea `-v ro`) a produs rezultatele prezentate în Tabelul 2.14.

Tabelul 2.14. Fonetizarea automată a termenilor italieni de eSpeak-NG pentru limba română. Patru din cinci termeni sunt fonetizați rezonabil, beneficiind de particularitățile comune ale fonetismului românesc și italian.

Termen	**Pronunție generată (eSpeak-NG)**	**Evaluare**
Margherita	/mar-ghe-ri-ta/	corect
Diavola	/dja-vo-la/	acceptabil
Carnivora	/kar-ni-vo-ra/	corect
Quattro Formaggi	/kwa-tro for-mad/	parțial corect
Capricciosa	/ka-prik-tso-sa/	**incorect**

Patru din cinci termeni sunt fonetizați rezonabil de eSpeak-NG, beneficiind de particularitățile comune ale fonetismului românesc și italian: „gh” înainte de „e” redă // dur în ambele limbi, iar „qu” generează grupul /kw/ în contexte românice. Singurul termen care necesită corecție este *Capricciosa*: grupul grafic „cci” determină eSpeak-NG să genereze grupul consonantic /kt/ în loc de /t/ simplu, producând fonetizarea incorectă /kapiktosa/. Substituția „Capricciosa” → „Capriciosa” (un singur „c”) rezolvă problema, eSpeak-NG producând /kapitosa/, confirmat prin rulare directă.

S-a investigat suplimentar posibilitatea injectării directe de secvențe IPA prin mecanismul de notație literală al eSpeak-NG (`[[foneme]]`), care permite specificarea explicită a fonemelor pentru anumite segmente de text, bypassing complet fonetizatorul. Testele au confirmat că notația funcționează pentru foneme simple ASCII (`[[kwatro]]` → /kwatro/), însă eșuează pentru caractere IPA cu diacritice (`[[kwato formadi]]` → /kw form/), limitând aplicabilitatea practică a acestei abordări la termeni cu fonologie simplă.

De asemenea, s-a investigat inserția de virgulă pentru corecția word-sticking în cazul modelului fonemic. Spre deosebire de modelul grafemic, eSpeak-NG procesează virgula ca

limită prozodică explicită, generând o întrerupere în fluxul IPA (linii separate în output eSpeak). Evaluarea perceptuală a indicat un comportament superior față de modelul grafemic în teste izolate. Totuși, evaluarea cantitativă a evidențiat că virgula a crescut rata WER pe categoria pizza de la 36,4% la 40,5%, producând artefacte similare celor din modelul grafemic (de exemplu, „Diavola” → „de avă la” în contextul „Diavola, mică”). Concluzia este că destabilizarea decodificatorului VITS la granițele prozodice artificiale este independentă de tipul de preprocesare (grafemic sau fonemic), reprezentând o caracteristică intrinsecă a arhitecturii neuronale end-to-end.

Dicționarul fonemic final reținut pentru evaluare conține exclusiv substituția validată empiric: *Capricciosa* → „Capriciosa”. Aplicarea acestui dicționar minimal a condus la o îmbunătățire a WER pe categoria *Comenzi pizza* de la 40,8% la 36,4% (−4,4 puncte procentuale) și a WER global de la 19,6% la 18,5%.

d) Corecția metodologiei de evaluare.

În cursul experimentelor, s-a identificat o eroare sistematică în pipeline-ul de evaluare WER: funcția de normalizare a textului nu elimina semnele de punctuație din transcrierea Whisper înainte de comparație. Deoarece Whisper produce frecvent punctuație în transcrieri (virgule, puncte, semne de exclamare), token-uri precum „rog!” sau „Mulțumesc.” erau comparate cu formele corecte „rog” și „Mulțumesc”, generând erori false. Corecția a constat în adăugarea unui pas de eliminare a caracterelor non-alfanumerice în funcția de normalizare, anterior calculului WER. Impactul a fost semnificativ: WER global al modelului FT-C-aug fără dicționar a scăzut de la 19,3% (valoare raportată anterior) la 14,9%, iar WER pe pizza de la 39,9% la 29,7%. Toate rezultatele finale raportate utilizează normalizarea corectă.

e) Validarea prin evaluare ASR bilingvă.

Pentru a distinge între erorile de calitate ale sintezei TTS și erorile de evaluare introduse de ASR-ul forțat pe limba română, s-a realizat o analiză complementară în care aceleași fișiere audio au fost transcrise atât cu Whisper forțat pe română (`language="ro"`), cât și cu Whisper forțat pe italiană (`language="it"`), pentru cele 20 de enunțuri din categoria *Comenzi pizza*. Rezultatele au evidențiat că termenii italieni sunt corect recunoscuți de ASR-IT în cazuri în care ASR-RO produce transcrieri eronate:

Referință: Dați-mi vă rog o pizza Quattro Formaggi mare.
Whisper-RO: Dați-mi vă rog o pizza cu atro format jimare.
Whisper-IT: Dati-mi, va'rogo, pizza **quattro formaggi** mare.

Referință: O pizza Diavola mică și o pizza Margherita medie, vă rog.
Whisper-RO: o pizza diavolamica si o pizza margherita medie, vă rog!
Whisper-IT: O pizza **diavola mica** si o pizza **margherita media**? Va rog!

Aceste exemple demonstrează că audio-ul generat de TTS conține informația fonică necesară recunoașterii corecte a termenilor italieni, dar modelul de limbă al ASR-ului român penalizează pronunțiile fonologic corecte în italiană ca erori în contextul românesc. De remarcat că Whisper-IT rezolvă în același timp și fenomenul de word-sticking (*Diavola mică* transcris separat, nu ca „diavolamica”), confirmând că lipirea este un artefact al transcrierii ASR-RO care reinterpretează grupul fonic prin fonetica română, nu un defect al audio-ului sintetizat. WER

calculat cu ASR-RO reprezintă prin urmare o limită inferioară a inteligibilității reale pentru termenii OOV de origine italiană.

f) Rezultate finale.

Tabelul 2.15. Rezultate WER normalizat cu și fără dicționar fonetic pentru modelele evaluate. Coloana Δ pizza indică variația față de baseline-ul fără dicționar al fiecărui model.

Configurație	WER global (%)	WER pizza (%)	Δ pizza
FT-C-aug (fără dicționar)	14,9	29,7	–
FT-C-aug (cu dicționar)	**14,3**	**26,5**	−3,2
sp14-D-aug (fără dicționar)	19,6	40,8	–
sp14-D-aug (cu dicționar)	18,5	36,4	−4,4

Modelul FT-C-aug cu dicționar fonetic de tip single-token constituie configurația finală recomandată, cu WER global de 14,3% și WER de 26,5% pe categoria comenzilor de pizza. Fenomenul rezidual de word-sticking, care contribuie la erorile rămase în această categorie, reprezintă o limitare arhitecturală a modelelor VITS end-to-end pentru termenii OOV urmați de modificatori lexicali: spre deosebire de sistemele TTS clasice, encoderul neural procesează direct secvențele de text și nu dispune de un modul explicit de gestionare a granițelor prosodice inter-lexicale pentru combinații de cuvinte absente din datele de antrenare. Remedierea acestui fenomen ar necesita fie date de antrenare suplimentare care să acopere tiparul respectiv, fie o arhitectură cu control explicit al prosodiei la nivel de cuvânt.

2.7.4. Evaluarea perceptuală a calității sintezei

Evaluarea cantitativă prin metrici automate (WER, CER) prezintă o limitare intrinsecă în contextul sistemelor TTS destinate aplicațiilor conversaționale: metricile bazate pe recunoaștere automată a vorbirii nu captează fidel calitatea perceptuală percepută de utilizatorii finali. Pentru a completa evaluarea obiectivă și a valida alegerea modelului pentru integrarea în sistem, s-a realizat o evaluare perceptuală cu evaluatori umani, utilizând o metodologie inspirată din standardul **MUSHRA** (*MUltiple Stimuli with Hidden Reference and Anchor*) [11].

Design-ul evaluării și participanți.

Evaluarea a vizat compararea a **patru sisteme TTS** reprezentând cele două configurații principale (grafemică și fonemică) înainte și după augmentarea datelor de fine-tuning, descrise în Tabelul 2.16.

Sistemele A și B utilizează aceeași voce (vorbitorul extern), permițând cuantificarea efectului augmentării pe configurația grafemică. Sistemele C și D realizează aceeași comparație

Tabelul 2.16. Sistemele TTS evaluate perceptual

Sistem	Configurație	Vorbitor	Augmentat
Sistem A	ConfigC (grafeme)	Vorbitor extern	Nu
Sistem B	ConfigC (grafeme)	Vorbitor extern	Da
Sistem C	ConfigD (foneme)	Vorbitor intern	Nu
Sistem D	ConfigD (foneme)	Vorbitor intern	Da

pentru configurația fonemică pe vorbitorul intern. Compararea B cu D oferă o perspectivă asupra preferinței utilizatorilor pentru una dintre cele două voci în contextul aplicației de restaurant.

Setul de stimuli a cuprins **6 enunțuri** reprezentative pentru domeniul HoReCa, distribuite pe trei categorii: rezervări (enunțurile 1–2), comenzi pizza cu termeni OOV italieni (enunțurile 3–4) și formule diverse — prețuri, politețe (enunțurile 5–6). Același set a fost sintetizat cu toate cele 4 sisteme, rezultând 24 de fișiere audio evaluate.

Evaluatorii au primit câte 4 stimuli audio etichetați aleator (ordine diferită per evaluator, pentru a elimina efectul de poziție) și au notat fiecare pe o scală continuă 0–100, răspunzând la întrebarea: *„Cât de potrivit sună această voce pentru un asistent vocal de restaurant?”*. Scala a fost ancorată cu descriptori verbali: Rău (0–20), Slab (20–40), Acceptabil (40–60), Bun (60–80), Excelent (80–100). Evaluarea a fost realizată printr-o interfață web dedicată, accesibilă de la distanță, cu salvare automată a rezultatelor în format CSV.

Au participat **17 evaluatori** fără cunoștințe tehnice de specialitate în procesarea vorbirii, fiecare completând toate cele 6 enunțuri pentru toate cele 4 sisteme. Au rezultat **388 de evaluări valide**, după excluderea unui set de răspunsuri utilizat pentru calibrare.

Rezultate și analiză statistică.

Scorurile medii globale și distribuția per categorie sunt prezentate în Tabelul 2.17 și Tabelul 2.18.

Tabelul 2.17. Scoruri perceptuale globale (17 evaluatori, $N = 97$ eșantioane per sistem)

Sistem	Medie (0–100)	Deviație standard
Sistem B (ConfigC augmentat, vorbitor extern)	**78,3**	19,6
Sistem D (ConfigD augmentat, vorbitor intern)	70,2	18,6
Sistem A (ConfigC fără aug., vorbitor extern)	56,5	24,4
Sistem C (ConfigD fără aug., vorbitor intern)	48,8	23,2

Analiza statistică prin testul t de Student pentru eșantioane perechi (*paired t-test*) confirmă semnificația statistică a diferențelor observate. Comparând variantele fără și cu augmentare,

Tabelul 2.18. Scoruri perceptuale per categorie de enunțuri

Sistem	Rezervări	Pizza (OOV)	Altele
Sistem B (ConfigC aug., vorbitor extern)	**73,2**	**82,5**	**79,1**
Sistem D (ConfigD aug., vorbitor intern)	67,8	72,7	70,1
Sistem A (ConfigC fără aug., vorbitor extern)	62,1	35,9	70,9
Sistem C (ConfigD fără aug., vorbitor intern)	49,6	38,0	58,7

diferențele sunt extrem de semnificative atât pentru configurația grafemică (Sistem A vs. B: $p < 0{,}001$) cât și pentru cea fonemică (Sistem C vs. D: $p < 0{,}001$). Comparația directă dintre cele două sisteme augmentate confirmă statistic preferința utilizatorilor pentru Sistemul B față de Sistemul D ($p = 0{,}003$).

Cel mai spectaculos efect al augmentării se observă în categoria **comenzi pizza (OOV)**: scorurile cresc de la 35,9 la 82,5 pentru Sistemul B (+46,6 puncte) și de la 38,0 la 72,7 pentru Sistemul D (+34,7 puncte). Această îmbunătățire dramatică confirmă perceptual concluzia evaluării cantitative — pronunția termenilor italieni OOV este net superioară după augmentarea setului de date cu terminologie specifică domeniului. De remarcat că, înaintea augmentării, configurația fonemică (Sistemul C, 38,0) a depășit ușor configurația grafemică (Sistemul A, 35,9) pe această categorie, datorită capacității fonetizatorului eSpeak-NG de a aproxima reguli de pronunție pentru termeni de origine italiană. Pe categoriile fără termeni OOV, augmentarea produce îmbunătățiri mai moderate (+11–13 puncte), sugerând că modelele de bază sintetizează satisfăcător textul românesc standard.

Comparând cele două voci în varianta augmentată, Sistemul B (78,3) este preferat față de Sistemul D (70,2) cu o diferență semnificativă statistic ($p = 0{,}003$), rezultat consistent cu scorurile WER superioare obținute în evaluarea obiectivă (14,3% față de 18,5%). Evaluarea perceptuală validează astfel alegerea **Sistemului B — ConfigC augmentat** ca model optim pentru sistemul final, oferind cel mai bun compromis între valorile obiective de inteligibilitate (WER 14,3%) și calitatea perceptuală globală evaluată de utilizatori (scor 78,3/100).

În urma evaluării obiective și a optimizărilor descrise, alegerea modelului optim depinde de prioritatea aplicației. Dacă **inteligibilitatea** este criteriul principal, modelul **FT ConfigC augmentat cu dicționar fonetic** oferă cel mai bun WER normalizat (**14,3%**) și WER de **26,5%** pe categoria comenzilor de pizza. Dacă **fidelitatea vocală** este prioritară, modelul **FT vorbitor ext. ConfigC** păstrează un scor de similaritate de **0,740**, cu prețul unui WER mai ridicat. Pentru aplicația de față — un sistem conversațional vocal în domeniul HoReCa — inteligibilitatea și acoperirea lexicală sunt criteriile determinante, motiv pentru care modelul **FT ConfigC augmentat cu dicționar fonetic** a fost selectat pentru integrare. Modelul a fost ulterior expus ca serviciu web HTTP, descris în secțiunea următoare.

2.7.5. Comparație cu sistemul comercial Zevo TTS

Pentru contextualizarea performanțelor modelului antrenat în raport cu o soluție comercială disponibilă, s-a realizat o evaluare comparativă față de sistemul **Zevo TTS** [30], utilizat și ca punct de referință în cadrul platformei de laborator. Evaluarea s-a realizat pe același set de 100 de enunțuri descris în Secțiunea 2.3.2 — acoperind atât scenarii *in-domain* (rezervări, comenzi, prețuri, formule de politețe) cât și enunțuri de română generală *out-of-domain* — utilizând același model Whisper `medium` și aceleași funcții de normalizare a textului, asigurând comparabilitatea deplină a rezultatelor.

Sistemul Zevo TTS [30], accesat prin API WebSocket, a generat audio cu vocea românească standard *maria*. Rezultatele comparative sunt prezentate în Tabelul 2.19.

Tabelul 2.19. Comparație WER între modelul antrenat și Zevo TTS pe setul de 100 de enunțuri (Whisper medium)

Sistem	**WER global (%)**	**WER pizza (%)**	**WER non-pizza (%)**
FT-C-aug (cu dicționar)	**14,3**	26,5	**11,2**
Zevo TTS (*maria*)	23,6	**24,0**	23,5

Modelul antrenat obține un WER global de **14,3%**, față de **23,6%** al sistemului Zevo TTS, o diferență de 9,3 puncte procentuale care confirmă superioritatea modelului antrenat pe ansamblul setului de evaluare. Pe cele 80 de enunțuri fără terminologie italiană (in-domain și out-of-domain), avantajul este și mai pronunțat: 11,2% față de 23,5%, reflectând adaptarea profundă a modelului la fonetica și prosodia limbii române prin antrenarea pe corpusul SWARA.

Pe categoria comenzilor de pizza (20 de enunțuri), diferența se reduce semnificativ: Zevo TTS înregistrează **24,0%** față de **26,5%** pentru modelul antrenat, o diferență de doar 2,5 puncte procentuale. Această ușoară inferioritate pe termenii OOV italieni se explică prin distribuția datelor de antrenare: sistemul Zevo TTS este antrenat pe corpusuri extinse care includ probabil terminologie culinară de origine italiană, în timp ce corpusul SWARA nu conține astfel de cuvinte. De remarcat că WER de 26,5% al modelului antrenat subestimează inteligibilitatea reală pe această categorie, din cauza penalizărilor ASR-RO pentru pronunții italiene corecte fonologic, demonstrat prin analiza bilingvă din Secțiunea 2.7.3.

Comparația confirmă că modelul antrenat este competitiv față de soluția comercială pe întreg setul de evaluare, cu o singură categorie de inferioritate marginală (termeni OOV italieni, $\Delta = 2{,}5$ pp) explicată prin acoperirea lexicală mai restrânsă a corpusului de antrenare. Extinderea corpusului de fine-tuning cu terminologie specifică domeniului HoReCa constituie o direcție de cercetare ulterioară pentru eliminarea acestei discrepanțe reziduale.

2.8. Expunerea modelului ca serviciu web HTTP

Pentru integrarea componentei de sinteză vocală în arhitectura sistemului conversațional, modelul de fine-tuning selectat (FT vorbitor extern, ConfigC) a fost expus ca un **serviciu web HTTP** utilizând framework-ul **Flask** (Python). Această abordare permite decuplarea modulului TTS de restul sistemului, facilitând apelarea sa din orice componentă a arhitecturii (server IVR, modul de dialog sau interfață web) printr-un API REST simplu, fără dependențe directe față de framework-ul Coqui TTS sau de mediul Python al serverului de antrenare.

Serviciul rulează pe serverul `rtx-4090-01` la adresa `http://192.168.1.115:5002` și expune două endpoint-uri. Primul, `GET /health`, returnează starea serviciului în format JSON: {`"status": "ok", "model": "FT Vorbitor Extern ConfigC"`}. Al doilea, `POST /synthesize`, reprezintă endpoint-ul principal de sinteză. Acesta acceptă un corp JSON cu câmpul `text` și returnează un fișier audio în format WAV, cu antetul `X-RTF` indicând Real-Time Factor-ul sintezei pentru cererea respectivă:

```
POST /synthesize
Content-Type: application/json
{"text": "Bună ziua, bine ați venit la restaurantul nostru!"}
```

O problemă identificată în etapa de testare a fost comportamentul modelului față de cifrele arabe din textul de intrare, aceeași limitare observată și în etapa de evaluare (Secțiunea 2.7). Soluția implementată este identică: o etapă de normalizare automată cu biblioteca `num2words` aplicată înainte de sinteză, care convertește cifrele în reprezentarea lor textuală în română, asigurând o sinteză corectă indiferent de formatul textului primit de la componentele din amonte ale sistemului.

Modelul TTS este încărcat o singură dată la pornirea serviciului și menținut în memoria GPU, eliminând latența de inițializare pentru cererile ulterioare. Această decizie de design este esențială pentru aplicații conversaționale, unde o latență consistentă este preferabilă uneia variabile. Timpul de răspuns pentru o cerere tipică de sinteză (10–20 de cuvinte) este de aproximativ **0,3–0,5 secunde**, corespunzător unui RTF mediu de **0,022**, valoare mult inferioară pragului de timp real (RTF $< 1{,}0$), ceea ce confirmă adecvarea modelului pentru integrarea în sisteme conversaționale interactive.

Pentru validarea funcționalității serviciului înaintea integrării cu componentele IVR, a fost implementată o interfață web minimală servită la rădăcina serviciului (`GET /`), care permite introducerea unui text arbitrar și redarea audio a rezultatului sintetizat direct în browser, fără instalarea unor instrumente suplimentare, prezentată în Figura 2.6.

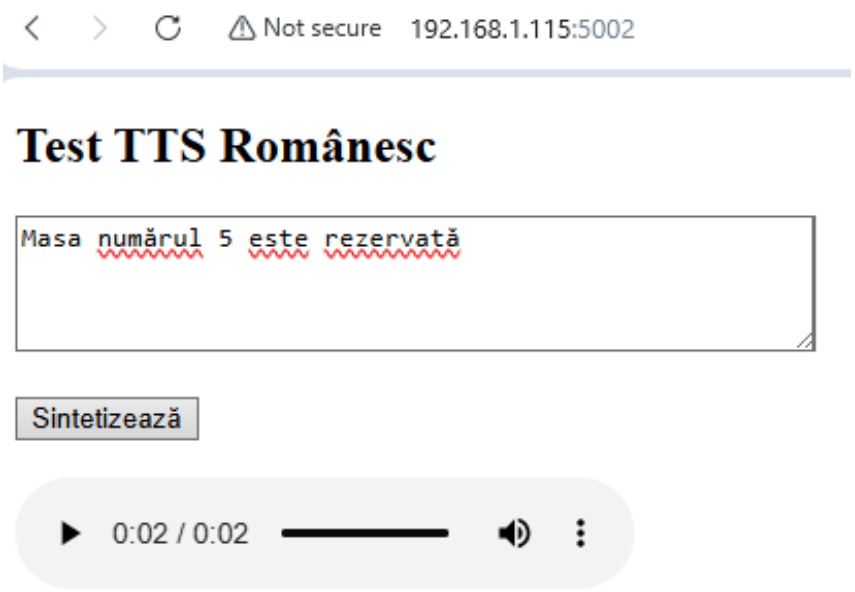


Figura 2.6. Interfața web de testare a serviciului TTS HTTP.

2.9. Sinteza capitolului

În cadrul acestui capitol a fost proiectată și implementată componenta de sinteză vocală a sistemului conversațional. Prin antrenarea de la zero a modelului VITS pe corpusul SWARA consolidat și căutarea sistematică a hiperparametrilor pe patru configurații experimentale, s-a identificat configurația C ca optimă, cu o valoare finală a funcției de pierdere de 19,10. Fine-tuning-ul pe vorbitorul extern a produs un model funcțional cu RTF mediu de 0,022, confirmat adecvat pentru sinteza în timp real.

Principala provocare abordată a fost pronunția termenilor italieni OOV din meniul restaurantului. Strategia de augmentare prin distilarea cunoștințelor vocale dintr-un model comercial (ElevenLabs)a redus WER-ul pe categoria comenzilor de pizza de la 74,6% la 29,7% (respectiv 26,5% cu dicționar fonetic), iar evaluarea perceptuală MUSHRA cu 17 evaluatori a validat calitatea modelului augmentat cu un scor global de 78,3/100 ($p < 0{,}001$). Modelul FT ConfigC augmentat cu dicționar fonetic, cu WER global de 14,3% și RTF de 0,022, a fost selectat pentru integrarea în arhitectura sistemului IVR și expus ca serviciu web HTTP independent.

3. PROIECTAREA ȘI IMPLEMENTAREA COMPONENTEI DE ÎNȚELEGERE A LIMBAJULUI NATURAL

Acest capitol detaliază proiectarea și implementarea componentei de înțelegere a limbajului natural (NLU), necesară pentru a interpreta solicitările utilizatorilor. Sunt prezentate fundamentele utilizării modelelor de limbaj de mari dimensiuni (LLM) cu scopul clasificării intențiilor (*intent recognition*) și extragerii entităților (*slot filling*), precum și metodologia de proiectare și îmbunătățire a instrucțiunilor (*prompt engineering*) pentru obținerea rezultatelor dorite. Se regăsesc de asemenea descrierea setului de date de validare și protocolul experimental utilizat pentru compararea mai multor modele. Scenariul în care se desfășoară aceste experimente este conform domeniului proiectului: rezervarea și comanda pentru restaurante.

3.1. Fundamente teoretice și selecția modelelor LLM

Evoluția de la sisteme NLU bazate pe reguli către soluții moderne este susținută de arhitectura **Transformer**, care modelează dependențe la distanță prin mecanisme de auto-atenție (*self-attention*) [27]. În practică, acest tip de arhitectură permite interpretarea unei solicitări în context global, ceea ce este esențial pentru identificarea intenției dominante și a sloturilor asociate (ex. locație, timp, număr de persoane, produse). Spre deosebire de abordările tradiționale — clasificare separată pentru intenție și modele dedicate pentru slot filling — LLM-urile pot realiza simultan clasificarea intenției și extragerea sloturilor prin învățare în context (*in-context learning*) și construire de prompturi (*prompting*) [2]. În acest proiect, rezultatul este constrâns într-un format structurat sub forma unui obiect JSON, incluzând variabile precum locația, data/ora (inclusiv expresii relative), numărul de persoane și produsele comandate, pentru a permite validare automată și integrare directă în pipeline-ul aplicației.

Pentru componenta NLU au fost evaluate patru modele LLM, selectate astfel încât să acopere atât soluții comerciale accesibile prin API, cât și un model open-weights utilizat printr-un furnizor de inferență. Toate modelele evaluate sunt capabile să proceseze limba română și să genereze ieșiri structurate sub constrângeri explicite de format. Compararea a urmărit următoarele criterii: (i) acuratețea extragerii intenției și a sloturilor semantice, (ii) corectitudinea extragerii comenzilor (structuri repetitive), (iii) stabilitatea respectării formatului JSON impus și (iv) observații practice relevante pentru integrarea într-un sistem conversațional (robustețe la prompt și consistență). [9]

Tabelul 3.1. Modelele LLM utilizate în evaluarea componentei NLU

ID	LLM name	LLM version / model ID	Release date
LLM-1	Google Gemini 3	Gemini 3 (Complex / reasoning setting)	18.11.2025
LLM-2	OpenAI GPT-5.2	GPT-5.2 (Auto / ChatGPT mode)	11.12.2025
LLM-3	Anthropic Claude Sonnet 4.5	Claude Sonnet 4.5	29.09.2025
LLM-4	Meta Llama 3.3 70B	`llama-3.3-70b-versatile`	06.12.2024

Modelele comerciale (GPT-5.2, Claude Sonnet 4.5 și Gemini 3) reprezintă starea actuală a artei în ceea ce privește performanța generală și suportul pentru output structurat,în timp ce Llama 3.3 70B [7] oferă un punct de comparație relevant pentru evaluarea unui model *open-weights*, cu un grad mai ridicat de dependență față de formularea promptului. Selecția finală a modelului utilizat în sistemul integrat este discutată în Secțiunea 3.5, pe baza performanței end-to-end și a profilului de erori observat experimental.

3.2. Arhitectura componentei NLU și *Prompt Engineering*

Arhitectura modulului NLU se bazează pe transformarea transcrierii normalizate (rezultată din modulul ASR) într-o reprezentare semantică structurată. Procesul este ghidat de un prompt care impune reguli explicite de extragere și un format de ieșire determinist, astfel încât rezultatul să poată fi validat și consumat direct de pipeline-ul aplicației.

Pentru interoperabilitate și validare automată, sistemul este constrâns să producă un obiect JSON cu nouă câmpuri fixe și valori canonicalizate [15], acoperind patru intenții posibile (*rezervare*, *comanda*, *informatii* și *necunoscut* (out-of-scope)) alături de informații despre locație, timp, comandă și tipul de informație solicitat. Schema este prezentată în Listingul 3.1.

```
{
  "id": "integer",
  "intentie": "rezervare | comanda | informatii | necunoscut",
  "locatie": "string | null",
  "data": "string | null",
  "ora": "string | null",
  "nr_persoane": "integer | null",
  "comanda": [ {"pizza": "string", "size": "mica | medie | mare | null"} ],
  "adresa": "string | null",
  "tip_info": "program | contact | adresa | null"
}
```

Listing 3.1. Structura JSON utilizată pentru extragerea datelor

Pentru a evalua sensibilitatea modelelor la formularea instrucțiunilor, au fost definite trei variante de prompt reprezentând o progresie în complexitatea constrângerilor: prompting zero-shot cu constrângeri (*zero-shot constrained prompting*) (P1), prompting cu exemple (*few-shot prompting*) (P2) și prompting bazat pe politici explicite (*policy prompting*) (P3). Promptul P1, cel mai scurt dintre cele trei, este prezentat integral în Listingul 3.2.

```
Returnează DOAR un JSON array valid (fără text suplimentar). Nu folosi markdown.
Pentru fiecare input, întoarce un obiect cu EXACT cheile (în această ordine):
"id","intentie","locatie","data","ora","nr_persoane","comanda","adresa","tip_info"

Reguli generale:
- Output exclusiv JSON array, fără explicaţii.
- Nu adăuga alte chei. Nu omite chei. Dacă nu ştii, pune null.
- Păstrează ordinea elementelor ca în listă.

Domenii permise:
- "intentie" in ["rezervare","comanda","informatii","necunoscut"]
- "tip_info" in ["program","contact","adresa", null]
- "comanda" este null sau listă de obiecte {"pizza": string, "size": "mica"|"medie"|"mare"|null}
- "nr_persoane" este integer sau null

Canonicalizare (ca în gold):
- Pentru "locatie" şi "pizza": foloseşte formă canonică cu iniţiale mari (Title Case) şi diacritice.
  Exemple: "Unirii", "Aviatorilor", "Militari", "Drumul Taberei", "Calea Victoriei", "Piaţa Romană".
- Dacă inputul conţine "romană" (cu/fără "piaţa"), setează "locatie"="Piaţa Romană".
- Pentru "size": foloseşte doar lowercase ("mica","medie","mare") sau null.
- Dacă sunt cerute cantităţi multiple, repetă obiectul în listă (nu agrega).

Timp şi date:
- Păstrează expresiile relative: "azi", "mâine", "acum", "diseară", "seară", "prânz", "weekend" etc.
- Dacă nu este menţionată nicio informaţie temporală, pune null la "data" şi "ora".

INPUTS: [ID=1] ... [ID=20] ...
```

Listing 3.2. Promptul P1 — zero-shot, constrângeri și canonicalizare

Față de P1, **Promptul P2** (few-shot) păstrează neschimbate toate regulile și constrângerile, adăugând exclusiv un bloc de două exemple concrete de intrare/ieșire imediat înainte de secțiunea `INPUTS`. Exemplele acoperă două scenarii reprezentative — o comandă cu cantități multiple și o rezervare cu expresie temporală relativă — ancorând comportamentul modelului fără a introduce constrângeri suplimentare. Blocul adăugat față de P1 este prezentat în Listingul 3.3.

```
EXEMPLE (doar pentru a ilustra formatul şi regulile):

Input:  [ID=901] bună comand o diavola mare şi două vegetariana
Output: {"id":901,"intentie":"comanda","locatie":null,"data":null,
         "ora":null,"nr_persoane":null,"comanda":[{"pizza":"Diavola",
         "size":"mare"},{"pizza":"Vegetariana","size":null},
         {"pizza":"Vegetariana","size":null}],"adresa":null,"tip_info":null}

Input:  [ID=902] hei diseară la romană mai e liber pentru patru
Output: {"id":902,"intentie":"rezervare","locatie":"Piaţa Romană",
         "data":"azi","ora":"diseară","nr_persoane":4,"comanda":null,
         "adresa":null,"tip_info":null}
```

```

ACUM procesează INPUTS şi returnează un JSON array cu 20 obiecte:
INPUTS: [ID=1] ... [ID=20] ...
```

Listing 3.3. Promptul P2 — blocul de exemple few-shot adăugat față de P1

Promptul P3 (zero-shot, rubrică procedurală) diferă structural față de P1 și P2: în locul unei liste plate de reguli, introduce o *rubrică cu ordine strictă de prioritate* pentru determinarea intenției, verificând pe rând scenariile out-of-scope, comanda, informatii și rezervare. Suplimentar, adaugă o regulă critică (`tip_info` poate fi non-null *exclusiv* când `intentie="informatii"`) și normalizări explicite pentru toate formele de expresie temporală, inclusiv cazuri ambigue (*„acum”* în contexte diferite, ore rostite literar, date calendaristice parțiale). Promptul P3 este prezentat integral în Listingul 3.4.

```
Returnează DOAR un JSON array valid (fără text suplimentar). Nu folosi
    markdown.
Pentru fiecare input, întoarce un obiect cu EXACT cheile (în această ordine
    ):
"id","intentie","locatie","data","ora","nr_persoane","comanda","adresa","
    tip_info"

Valori permise:
- "intentie" in ["rezervare","comanda","informatii","necunoscut"]
- "tip_info" in ["program","contact","adresa", null]
- "nr_persoane" integer sau null
- "comanda" null sau lista {"pizza": string, "size": "mica"|"medie"|"mare"|
    null}
- campuri neidentificabile sigur => null

REGULA CRITICA:
- "tip_info" poate fi NON-null DOAR daca "intentie"="informatii". Altfel "
    tip_info"=null.

CANONICALIZARE:
- "locatie" si "pizza" in Title Case.
- daca apare "romana" (cu sau fara "piata") => "locatie"="Piata Romana"
- "size" doar: "mica"|"medie"|"mare"|null (lowercase)
- cantitati multiple: repeta obiectul in lista (nu agrega cantitatea)
- "adresa" completeaza DOAR daca apare explicit "adresa X" sau "la adresa X
    "

RUBRICA INTENTIE (ordine stricta):
1. OUT-OF-SCOPE => "necunoscut": plata/card/tichete, alergeni, livrare,
   parcare, anulare, reclamatii, joburi. Nota: "acceptati comenzi acum" =>
    necunoscut.
2. Daca exista pizza/cantitati/marimi => "comanda" (+ "adresa" daca e
    explicita).
3. Daca intreaba despre program/orar/deschis => "informatii", "tip_info"="
    program"
   Daca intreaba despre telefon/contact    => "informatii", "tip_info"="
    contact"
   Daca intreaba despre adresa/localizare  => "informatii", "tip_info"="
    adresa"
4. Daca cere masa/rezervare/disponibilitate => "rezervare"
5. Altfel => "necunoscut"

DATA/ORA:
- "maine" => data="maine"; "azi" => data="azi"; "weekend" => data="weekend"
```

```
- "diseara" => data="azi" si ora="diseara"
- "seara" (fara diseara) => ora="seara" (data doar daca e mentionata)
- "pranz"/"pe la pranz" => ora exact cum apare
- "pe la sase/sapte/opt" => ora exact cum apare
- "acum": disponibilitate => data="acum", ora=null
          "vin acum"/"deschis acum" => data="azi", ora="acum"
          out-of-scope => data=null, ora="acum"
- Date DD.MM.YYYY: foloseste exact.
- Zi + luna fara an (ex: 15 martie) => normalizeaza la "DD.MM.2026".
- "ora HH:MM" => pastreaza HH:MM
- "ora optsprezece" => "18:00"; "ora douazeci si treizeci" => "20:30"

CONSTRÂNGERI:
- Nu adauga alte chei. Nu omite chei. Respecta ordinea cheilor.
- Output: JSON array cu obiecte in aceeasi ordine ca INPUTS.

INPUTS: [ID=1] ... [ID=20] ...
```

Listing 3.4. Promptul P3 — zero-shot, rubrică procedurală

Cele trei variante acoperă paradigme distincte de prompt engineering: P1 definește formatul și regulile de bază, P2 adaugă exemple pentru a ancora cazurile tipice, iar P3 introduce o rubrică procedurală care elimină ambiguitățile rămase, în special pentru câmpurile temporale. Performanța comparativă a celor trei variante este discutată în Secțiunea 3.5.

3.3. Setul de date pentru testare (NLU Dataset)

Pentru evaluarea componentei NLU a fost construit un set de date sintetic format din 100 de solicitări în limba română, concepute pentru a reflecta interacțiuni realiste într-un context de restaurant. Setul include atât cereri scurte și directe, cât și enunțuri complexe, colocviale sau eliptice, frecvente în dialogul oral. Fiecare intrare este reprezentată prin: (i) textul natural al solicitării, (ii) o transcriere normalizată (litere mici, numere exprimate în cuvinte, fără punctuație) și (iii) etichetarea de referință (*gold*) utilizată pentru evaluare. Distribuția pe complexitate este de 20% solicitări simple — cereri cu o singură intenție și puține sloturi — și 80% solicitări complexe, cu mai multe sloturi, timp relativ sau explicit, cantități multiple, formulări colocviale și cazuri *out-of-scope*.

Etichetarea de referință a fost realizată manual, pe baza unui set de reguli deterministe definit anterior evaluării. Pentru fiecare solicitare sunt extrase exclusiv informațiile menționate explicit în enunț, fără inferențe implicite. În cazurile ambigue sau în afara scopului funcțional al sistemului (ex. plăți, livrare, alergeni, anulare), solicitările sunt etichetate cu intenția *necunoscut*, pentru a reflecta comportamentul dorit într-un sistem conversațional real (clarificare sau redirecționare).

Pentru a ilustra diversitatea și gradul de complexitate al solicitărilor, sunt prezentate câteva exemple reprezentative. Acestea acoperă principalele categorii semantice urmărite de componenta NLU, respectiv cereri de rezervare, comenzi, solicitări de informații și cazuri *out-of-scope*. Exemplele ilustrează atât formulări concise, cât și enunțuri colocviale sau eliptice, caracteristice dialogului oral.

Rezervare simplă. *„Vreau o rezervare la Unirii."* Transcriere normalizată: *vreau o rezervare la*

unirii Ieșire de referință: `intentie=rezervare, locatie=Unirii`.

Comandă directă. *„O pizza Diavola, vă rog.”* Transcriere normalizată: *o pizza diavola vă rog* Ieșire de referință: `intentie=comanda`, cu un element în lista `comanda`.

Rezervare cu sloturi multiple. *„Salut, suntem cinci și ne gândeam la o masă la Unirii.”* Transcriere normalizată: *salut suntem cinci și ne gândeam la o masă la unirii* Ieșire de referință: `intentie=rezervare, locatie=Unirii, nr_persoane=5`.

Solicitare de informații cu timp relativ. *„Alo, azi la Piața Romană cum e programul?”* Transcriere normalizată: *alo azi la piața romană cum e programul* Ieșire de referință: `intentie=informatii, tip_info=program, data=azi`.

Caz *out-of-scope*. *„Pot plăti cu cardul?”* Transcriere normalizată: *pot plăti cu cardul* Ieșire de referință: `intentie=necunoscut`.

Aceste exemple evidențiază prezența expresiilor temporale relative (*azi, diseară*), a cantităților exprimate în limbaj natural și a formulărilor colocviale, elemente care contribuie la creșterea dificultății sarcinii de extragere. Lista completă a celor 100 de solicitări, împreună cu etichetările de referință, este disponibilă în anexă.

3.4. Metodologia de evaluare a componentei NLU

Evaluarea componentei NLU bazate pe modele de tip LLM este formulată ca o problemă combinată de clasificare și extracție de informație. Fiecare model este evaluat pe întregul set de 100 de solicitări cu fiecare dintre cele trei prompturi (P1–P3), rezultând un total de 12 rulări experimentale. Calculul metricilor a fost automatizat printr-un pipeline alcătuit dintr-un script de orchestrare (rulare batch pentru toate configurațiile) și un script de evaluare (compararea ieșirilor cu etichetarea gold), care generează câte un raport JSON per configurație.

Scriptul de evaluare este implementat în Python și operează pe fișiere JSON produse de modele, comparându-le cu etichetarea gold la nivel de câmp. Compararea câmpului `comanda` utilizează un algoritm de potrivire pe multiset: perechile `(pizza, size)` sunt normalizate la lowercase înainte de comparație, astfel încât diferențele de capitalizare (ex. `"diavola"` vs. `"Diavola"`) să nu fie penalizate ca erori semantice. Scorul F1 este calculat pe baza intersecției multiseturilor, permițând evaluarea corectă a comenzilor cu cantități multiple reprezentate prin repetiție de obiecte. Suplimentar, scriptul detectează automat subtipuri de erori din taxonomie: `date_time_swap` (inversarea valorilor între câmpurile `data` și `ora`), `order_size_mismatch` (nume de pizza corect, mărime greșită) și `order_item_mismatch` (pizza greșită sau cantitate incorectă). Parsarea ieșirilor este concepută tolerant față de abateri minore de format, tratând `[]` și `null` ca echivalente pentru câmpul `comanda`, separând astfel erorile de formatare de cele semantice.

Metricile utilizate sunt:

– **Intent Accuracy**: procentul de solicitări pentru care câmpul `intentie` corespunde etichetării de referință.

– **Field Accuracy pe sloturi**: acuratețea per câmp pentru sloturile `locatie`, `data`, `ora`, `nr_persoane`, `adresa` și `tip_info`.
– **Order Extraction Score**: scorul F1 mediu pentru lista `comanda`, calculat ca potrivire pe mulțimi a perechilor `(pizza, size)`, pentru a permite comenzi multiple și ordine diferită a itemilor.
– **Strict Record Accuracy**: procentul de solicitări pentru care toate câmpurile (intenție, sloturi și comandă) sunt extrase corect simultan, reflectând un scenariu end-to-end.

Suplimentar, este colectată o **taxonomie a erorilor** (ex. `data_mismatch`, `ora_mismatch`, `intent_mismatch`, `missed_order`, `spurious_order`), utilizată pentru analiza calitativă a comportamentului modelelor.

3.5. Rezultate experimentale și analiză comparativă

Rezultatele agregate pentru toate configurațiile model–prompt sunt prezentate în Tabelul 3.2. Se observă un tipar consistent: pentru modelele comerciale (GPT, Claude, Gemini), trecerea de la P1 la P3 produce o creștere substanțială a performanței end-to-end. **Strict Record Accuracy** crește de la aproximativ 64–66% (P1) la 90–94% (P3), iar îmbunătățirea este corelată în principal cu stabilizarea câmpurilor temporale `data` și `ora`. Pentru Llama 3, P3 îmbunătățește semnificativ extragerea temporală (Data 90%, Ora 93%), însă persistă erori mai frecvente pe intenție și locație, ceea ce limitează scorul end-to-end la 67%.

Tabelul 3.2. Rezultate agregate (100 exemple) pentru fiecare model și prompt

Model / Prompt	Strict Rec. Acc.	Intent Acc.	Data Acc.	Ora Acc.
Claude P1	64%	92%	75%	89%
Claude P2	79%	96%	81%	93%
Claude P3	94%	98%	96%	98%
Gemini P1	66%	93%	75%	84%
Gemini P2	72%	100%	76%	92%
Gemini P3	91%	99%	96%	95%
GPT P1	64%	91%	74%	86%
GPT P2	76%	98%	80%	92%
GPT P3	90%	97%	96%	95%
Llama P1	44%	79%	68%	84%
Llama P2	49%	79%	70%	88%
Llama P3	67%	82%	90%	93%

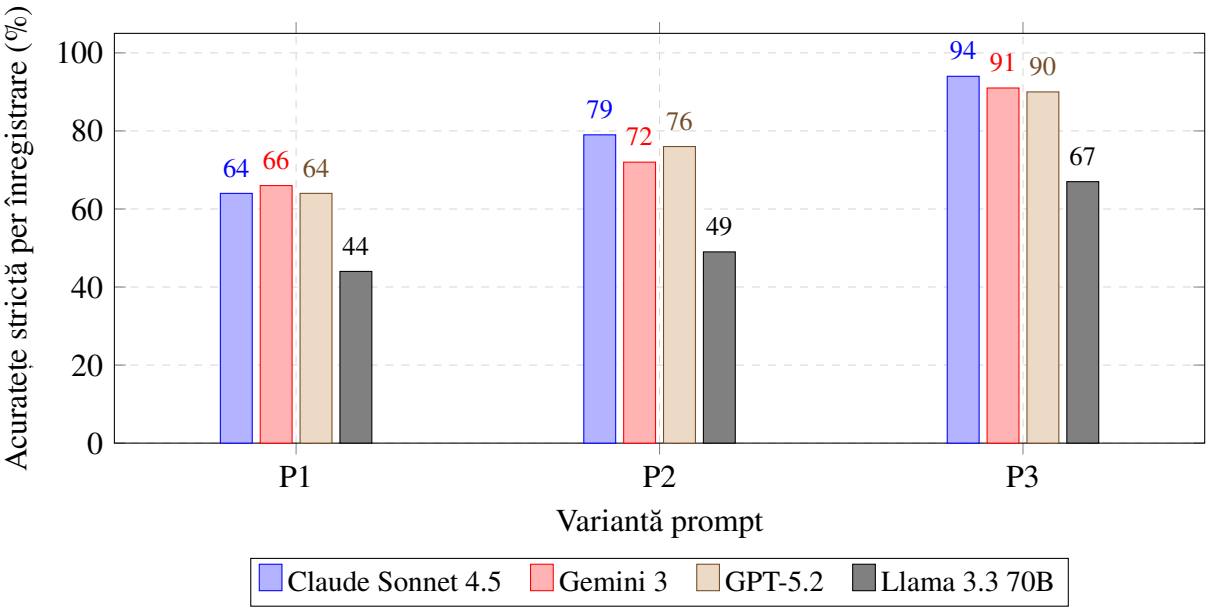


Figura 3.1. Evoluția Strict Record Accuracy pentru variantele de prompt P1–P3

Pe componenta `comanda`, modelele comerciale obțin scoruri foarte ridicate (F1 în intervalul 98–100%), ceea ce sugerează că recunoașterea numelor de pizza și a mărimilor este o sarcină relativ stabilă în cadrul acestui domeniu. Llama 3 are performanță mai modestă (F1 aproximativ 83–87%), cu erori recurente de mărime (`order_size_mismatch`), comenzi omise (`missed_order`) și comenzi spurioase (`spurious_order`).

Taxonomia erorilor indică faptul că, pentru GPT, Claude și Gemini în P1 și P2, cele mai frecvente erori sunt mismatch-urile temporale: `data_mismatch` (19–26 cazuri din 100) și `ora_mismatch` (7–16/100). Promptul P3 reduce aceste erori drastic (tipic 4–5/100), confirmând utilitatea rubricii procedurale și a regulilor de normalizare pentru date/ore. Eroarea `date_time_swap` — inversarea valorilor între câmpurile `data` și `ora` — apare izolat (1 caz la Claude P1) și dispare complet începând cu P2, ceea ce indică faptul că exemplele few-shot sunt suficiente pentru a ancora distincția dintre cele două câmpuri temporale.

Pentru Llama 3, erorile sunt mai distribuite și reflectă o sensibilitate mai ridicată la formularea promptului. Erorile dominante sunt `intent_mismatch` (18–21/100 în funcție de prompt) și `locatie_mismatch` (13–14/100), aceasta din urmă fiind absentă complet la modelele comerciale. Analiza per-exemplu relevă că erorile de locație sunt cauzate în principal de canonicalizarea defectuoasă a alias-urilor: modelul returnează forme prescurtate (ex. *Romană* în loc de *Piața Romană*) sau omite locația complet în contexte ambigue. Erorile specifice comenzilor (`spurious_order`, `missed_order`, `order_size_mismatch`) persistă în toate cele trei variante de prompt, cu valori între 3 și 7 cazuri fiecare, și sunt corelate cu confuziile de intenție — modelul clasifică uneori o comandă drept rezervare și invers, propagând eroarea asupra sloturilor asociate. Acest profil sugerează că Llama 3.3 70B, deși capabil să proceseze limba română, prezintă o consistență mai redusă în respectarea constrângerilor de canonicalizare. O altă sursă de confuzie o reprezintă similitudinea dintre clasele de intenție adiacente.

3.6. Evaluarea unui model LLM local (CPU-only)

Suplimentar evaluării modelelor comerciale accesibile prin API, a fost investigată fezabilitatea utilizării unui model LLM rulat local, exclusiv pe procesor (CPU-only), fără acces la acceleratoare hardware dedicate.Evaluarea urmărește să verifice dacă un model CPU-only reprezintă o alternativă viabilă față de API-urile comerciale.

Experimentele au fost realizate pe un laptop cu procesor Intel Core i7-10750H (2.60 GHz) și 8 GB RAM, utilizând platforma **Ollama** pentru inferență locală. Au fost testate trei modele de dimensiuni reduse, compatibile cu constrângerea de memorie disponibilă, prezentate în Tabelul 3.3. Toate modelele au fost evaluate cu promptul P3 și cu parametrul `format: json` activat în API-ul Ollama pentru a constrânge sintactic formatul de ieșire. Fiecare solicitare a fost trimisă individual, măsurând latența per exemplu cu `time.time()` în jurul fiecărui apel.

Tabelul 3.3. Modele LLM testate în configurație CPU-only

Model	Parametri	Dimensiune	RAM ollama.exe	CPU ollama.exe
`llama3.2:3b`	3B	2.0 GB	~2.1 GB	~49%
`gemma2:2b`	2B	1.6 GB	~1.7 GB	~44%
`qwen2.5:3b`	3B	2.0 GB	~2.0 GB	~38%

Rezultatele complete ale evaluării sunt prezentate în Tabelul 3.4. Modelul `gemma2:2b` obține cea mai bună performanță dintre cele trei, cu o acuratețe strictă de 13% și o acuratețe pe intenție de 80%.

Tabelul 3.4. Rezultate modele CPU-only (P3, 100 exemple)

Model	Strict Acc.	Intent Acc.	Locatie Acc.	Pizza F1	Latency/ex	Total
`gemma2:2b`	**13%**	80%	57%	78%	17.9s	~30 min
`llama3.2:3b`	8%	79%	44%	81%	13.3s	~22 min
`qwen2.5:3b`	8%	79%	44%	81%	~16s	~27 min

Analiza calitativă a ieșirilor modelelor locale relevă trei categorii de erori specifice, absente în evaluarea modelelor comerciale. Prima și cea mai gravă este agregarea cantităților în câmpul `pizza` în loc de repetiția obiectelor conform schemei: `llama3.2:3b` produce în mod sistematic structuri de forma {`"pizza": "trei margherita"`} sau {`"pizza": "cinci diavola"`}, încălcând explicit regula din promptul P3. Această eroare apare la toate comenzile cu cantități mai mari de 1 și explică în mare parte Pizza F1 inițial scăzut pentru acest model. A doua categorie este subreprezentarea prin repetiție incompletă: chiar și atunci când modelul încearcă să repete obiectele, produce mai puține copii decât sunt menționate în input (ex. ID=36: gold 4 obiecte Diavola, predicție 1 obiect; ID=49: gold 10 obiecte, predicție 2). A treia categorie, specifică modelelor `gemma2:2b` și `qwen2.5:3b`, este returnarea câmpului `comanda` ca obiect dict în loc de listă ({`"pizza": "diavola"`} în loc de [{`"pizza": "Diavola"`}]), ceea ce face ca scorul F1 să fie 0 chiar și pe comenzi cu un singur item corect — o eroare de format pur, nu semantică. Notă: valorile Pizza F1 raportate în tabel au fost calculate cu un script de evaluare care normalizează numele pizzelor la lowercase înainte de comparație, eliminând astfel penalizarea pentru diferențe de capitalizare între gold și predicție.

Suplimentar, `gemma2:2b` produce în 6 cazuri valori de intenție în afara schemei definite, precum `"masă"`, `"anulare"` sau reproducerea literală a inputului (`"acceptați comenzi acum"`), în loc să mapeze la una dintre cele patru clase permise. Principalele surse de eroare pentru `gemma2:2b` sunt `locatie_mismatch` (43/100), `nr_persoane_mismatch` (36/100) și `data_mismatch` (31/100). Modelul `llama3.2:3b` prezintă în plus probleme de encoding

al caracterelor românești, generând forme corupte precum *Piaüa* sau *Piaóa*, ceea ce explică acuratețea de locație de doar 44%.

Rezultatele indică faptul că modelele de dimensiuni reduse rulate exclusiv pe CPU nu sunt viabile pentru componenta NLU în configurația actuală a sistemului. Sistemul prezintă două limitări fundamentale: capacitatea semantică insuficientă pentru urmărirea constrângerilor din promptul P3 și latența ridicată (13–18 secunde per exemplu), incompatibilă cu cerințele unui sistem conversațional în timp real.

3.7. Sinteză comparativă și direcții de optimizare

Tabelul 3.5 sintetizează performanța tuturor configurațiilor evaluate, punând față în față modelele comerciale (cu promptul P3) și modelele locale CPU-only.

Tabelul 3.5. Comparație modele comerciale (P3) vs. modele locale CPU-only

Model	Strict Acc.	Intent Acc.	Locatie Acc.	Pizza F1	Latency/ex	Rulare
Claude Sonnet 4.5 (P3)	**94%**	98%	100%	99%	N/A	API cloud
Gemini 3 (P3)	91%	99%	100%	99%	N/A	API cloud
GPT-5.2 (P3)	90%	97%	100%	98%	N/A	API cloud
Llama 3.3 70B (P3)	67%	82%	100%	87%	N/A	API cloud
`gemma2:2b`	13%	80%	57%	78%	17.9s	CPU local
`llama3.2:3b`	8%	79%	44%	81%	13.3s	CPU local
`qwen2.5:3b`	8%	79%	44%	81%	~16s	CPU local

N/A: latența modelelor comerciale nu a fost instrumentată în cadrul acestei evaluări, deoarece depinde de factori externi necontrolabili (încărcarea serverului, rețea, regiunea de inferență).

Analiza comparativă evidențiază un decalaj substanțial între cele două categorii: cel mai bun model local (`gemma2:2b`, 13% strict accuracy) se situează la peste 75 de puncte procentuale față de Claude Sonnet 4.5 P3 (94%) consistent cu literatura de specialitate [9]. Profilul de erori este însă calitativ diferit — modelele comerciale eșuează aproape exclusiv pe câmpuri temporale, în timp ce modelele locale acumulează erori pe toate dimensiunile simultan: canonicalizare, format, intenție și extragerea comenzilor.

Analiza erorilor identificate în cadrul evaluării P1–P3 și al modelelor locale sugerează câteva direcții de îmbunătățire. Regula pentru `„acum"` prezintă inconsistențe între variante, afectând câmpurile `data` și `ora`. Modelele locale nu au respectat convenția de repetiție a obiectelor pentru cantități multiple, chiar și în prezența regulii explicite din P3, iar canonicalizarea locațiilor și a numelor de pizza ar putea fi consolidată prin furnizarea unui enum închis explicit în loc de exemple ilustrative.

Experimentele prezentate în acest capitol demonstrează că utilizarea unui LLM comercial cu un prompt structurat corespunzător reprezintă o soluție viabilă pentru componenta NLU a sistemului conversațional. Claude Sonnet 4.5 cu promptul P3 obține cea mai bună performanță end-to-end (94% Strict Record Accuracy), modelele locale CPU-only dovedindu-se neviabile atât din perspectiva acurateței, cât și a latenței.

4. PROIECTAREA ȘI IMPLEMENTAREA SERVERULUI IVR

Acest capitol descrie integrarea componentelor dezvoltate anterior (modelul TTS antrenat pe corpusul SWARA (Capitolul 2), modulul de înțelegere a limbajului natural bazat pe LLM (Capitolul 3) și modulul de gestiune a dialogului) într-un sistem IVR (Interactive Voice Response) funcțional, capabil să susțină o conversație vocală completă printr-un apel telefonic real.

4.1. Arhitectura sistemului: componente și schimb de mesaje

Sistemul integrat este compus din șase componente principale care comunică prin protocoale standardizate, ilustrate în Figura 4.1. **Platforma Twilio** [26] gestionează infrastructura telefonică: primește apelul de la utilizator, îl redirecționează către serverul IVR printr-un webhook HTTP și stabilește un canal WebSocket bidirecțional pentru transmisia audio în timp real. Pachetele audio sunt transmise în format μ-law la 8000 Hz, codificate în Base64. **Tunelul ngrok** expune serverul IVR local către internet, generând un URL public accesibil platformei Twilio, necesar deoarece serverul de calcul se află într-o rețea privată fără adresă IP publică. **Serverul IVR** (`ivr_gateway.py`, Flask, portul 5000) constituie componenta centrală: gestionează conexiunile WebSocket cu Twilio, orchestrează fluxul conversației și coordonează apelurile către celelalte module. **Serviciul TTS** (`tts_service.py`, Flask, portul 5002), descris în Secțiunea 2.8, furnizează audio sintetizat cu modelul VITS antrenat, pornind de la text primit prin HTTP POST. **Modulul STT** transcrie audio-ul recepționat din apel folosind API-ul Zevo Live STT [29], accesat prin WebSocket. **Modulul de dialog și NLU** (`chatbot_meu.py`) implementează logica conversației: extragerea intențiilor și a sloturilor prin Azure OpenAI GPT-4.1 [19], verificarea disponibilității în baza de date și confirmarea acțiunilor.

Fluxul unui apel complet parcurge următoarele etape: utilizatorul sună la numărul Twilio alocat; platforma redirecționează apelul către serverul IVR prin webhook-ul configurat dinamic la fiecare pornire a serverului; serverul returnează un document TwiML care instruiește Twilio să stabilească o conexiune WebSocket securizată (`wss://`) la endpointul `/realtime`; prin această conexiune, serverul redă mesajul de întâmpinare sintetizat vocal, recepționează răspunsul utilizatorului, îl transcrie și generează un răspuns contextual, repetând ciclul până la finalizarea conversației.

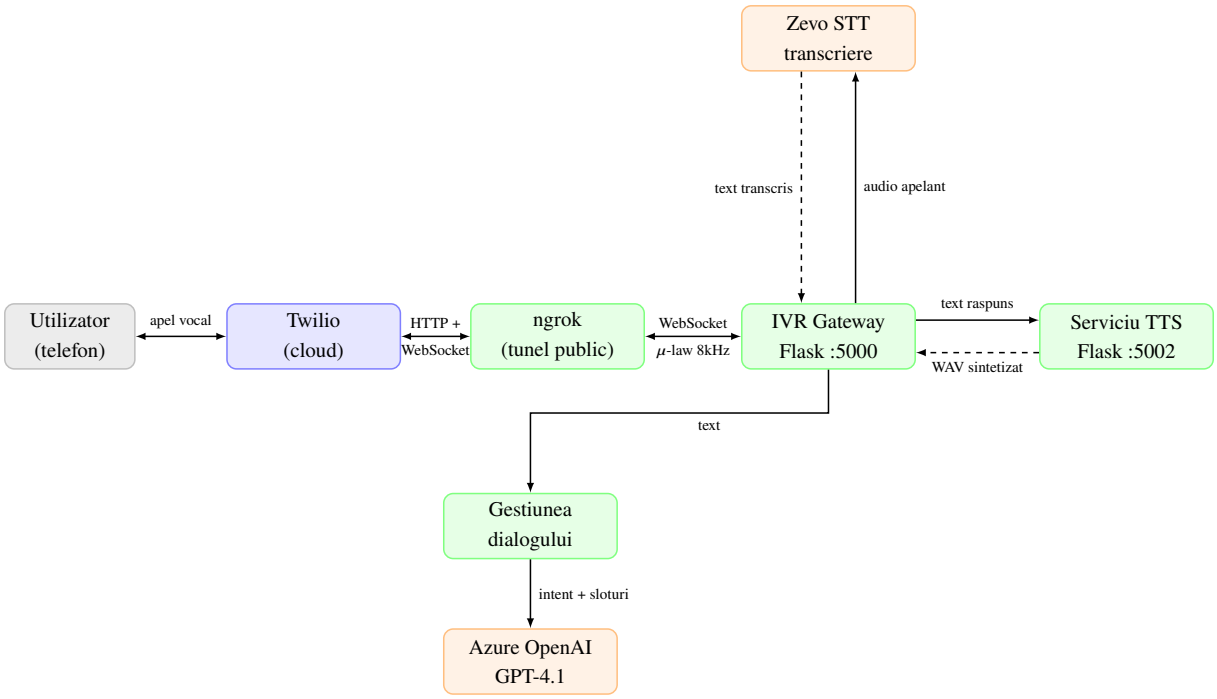


Figura 4.1. Arhitectura sistemului IVR conversational integrat. Săgețile continue indică fluxul de date dus, iar cele punctate răspunsul întors. Componentele locale rulează pe serverul de calcul; comunicarea cu Twilio se realizează prin tunelul ngrok, iar procesarea vocii implică serviciile externe Zevo STT și Azure OpenAI, respectiv modelul TTS antrenat local.

Rularea sistemului necesită configurarea a trei servicii active simultan, gestionate prin sesiuni `tmux` pe serverul de calcul. Fișierul `.env` conține credențialele platformei Twilio și URL-ul public al serverului (`SERVER_URL`). Ordinea de pornire este: (1) serviciul TTS pe portul 5002, care necesită încărcarea modelului VITS în memoria GPU (aproximativ 30 de secunde); (2) tunelul ngrok, care generează URL-ul public; (3) serverul IVR, care actualizează automat webhook-ul Twilio și pornește serverul Flask pe portul 5000.

4.2. Modulul de gestiune a dialogului

Logica conversațională a sistemului este implementată în modulul `chatbot_meu.py`, care orchestrează întregul flux de dialog pentru domeniul restaurantului și constituie componenta centrală a inteligenței sistemului IVR. Modulul utilizează Azure OpenAI GPT-4.1 [19] pentru interpretarea limbajului natural și structurează interacțiunea în trei scenarii principale: rezervare, comandă și solicitare de informații.

La inițializare, modulul încarcă două fișiere de configurare externe. `dialogues_meu.txt` conține replicile sistemului indexate prin chei simbolice (`welcome_message`, `prompt_comanda`, `confirm_reservation` etc.) permițând modificarea textelor fără intervenție în logica codului. `knowledge_base_meu.json` stochează starea persistentă a sistemului: meniul restaurantului (5 tipuri de pizza cu prețuri), locațiile disponibile (6 locații din București), rezervările existente și

comenzile plasate. La confirmarea unei rezervări sau comenzi, baza de cunoștințe este actualizată pe disc prin funcția `save_knowledge_base`, asigurând persistența datelor între sesiuni.

Interpretarea limbajului natural se realizează prin funcția `parse_intent`, care apelează GPT-4.1 cu prompturi specializate în funcție de contextul conversațional. Promptul de tip 1 clasifică intenția principală a apelantului în una dintre clasele `rezervare`, `comandă`, `informații` sau `necunoscut`. Promptul de tip 2 extrage simultan toți parametrii necesari rezervării – locație, număr de persoane, dată în format `DD.MM.YYYY` și oră în format `HH:MM` – returnând un obiect JSON cu valori `null` pentru câmpurile absente, ceea ce permite colectarea selectivă a informațiilor lipsă prin întrebări ulterioare. Promptul de tip 3 identifică perechile (pizza, dimensiune)din comenzile verbale libere, realizând normalizarea față de lista canonică din baza de cunoștințe (ex. *„diavă”* $\rightarrow$ *Diavola*, *„la medie”* $\rightarrow$ *Medie*).

Fluxul conversațional începe cu clasificarea intenției apelantului, după care execuția se ramifică în unul dintre cele trei scenarii. În scenariul de **rezervare**, sistemul încearcă să extragă toți cei patru parametri dintr-un singur enunț liber; câmpurile lipsă sunt colectate prin întrebări individuale (`choose_location`, `choose_date`, `choose_time`, `choose_number_ppl`), fiecare cu validare și reîncercare în caz de răspuns neinteligibil. Verificarea disponibilității se realizează prin funcția `check_availability`, care calculează suprapunerea temporală a rezervărilor existente pentru aceeași locație și dată, considerând o fereastră de două ore per rezervare și o capacitate maximă de 50 de persoane simultan. Dacă slotul solicitat nu este disponibil, utilizatorul este informat și procesul se reia. La confirmare verbală explicită, rezervarea este adăugată în baza de cunoștințe și persistată pe disc.

În scenariul de **comandă**, sistemul oferă opțional prezentarea meniului, după care solicită comanda printr-un enunț liber. GPT-4.1 extrage lista de perechi (pizza, dimensiune); pentru fiecare dimensiune lipsă sau nerecunoscută în lista canonică `[‘‘Mică’’, ‘‘Medie’’, ‘‘Mare’’]`, sistemul întreabă explicit. Apelantul poate adăuga pizza suplimentare iterativ, răspunzând la întrebarea „Mai aveți alte pizza de adăugat?” gestionată printr-un prompt GPT binar. La final, sistemul colectează adresa de livrare și observațiile opționale (interfon, etaj), citește un sumar al comenzii și așteaptă confirmarea explicită înainte de a persista comanda. În scenariul de **informații**, sistemul solicită tipul de informație dorită – program de lucru sau date de contact – și redă răspunsul corespunzător din fișierul de dialoguri.

Toate interacțiunile vocale sunt realizate exclusiv prin funcțiile `say_and_display` și `listen_and_transcribe`, fără referințe directe la perifericele audio. Abstractizarea interfeței de intrare/ieșire a permis ulterior integrarea în contextul IVR fără nicio modificare a logicii interne, prin simpla înlocuire a celor două funcții cu versiunile bazate pe WebSocket Twilio, descrisă în Secțiunea 4.3.

4.3. Integrarea componentelor în fluxul de procesare

Punctul de plecare al implementării l-a constituit exemplul de server IVR furnizat în cadrul laboratorului 6 al cursului, bazat pe biblioteca `flask-sock` pentru gestionarea conexiunilor WebSocket și pe SDK-ul Python al platformei Twilio. Exemplul inițial implementa două

funcționalități de bază: redarea unui fișier audio preexistent pe disc și înregistrarea a cinci secunde de audio din apel, salvate local. Extinderea a implicat trei modificări principale față de această implementare: înlocuirea redării fișierelor statice cu sinteza vocală în timp real prin serviciul HTTP descris în Secțiunea 2.8, înlocuirea salvării brute a audio-ului cu transcrierea prin Zevo STT și integrarea modulului complet de dialog descris în Secțiunea 4.2.

Prima provocare tehnică a fost reconcilierea formatelor audio incompatibile între componentele sistemului. Twilio Media Streams utilizează codarea G.711 [12] μ-law la 8000 Hz, cu 8 biți per eșantion, în timp ce serviciul TTS generează audio la 22050 Hz, cu 16 biți per eșantion, în format PCM, iar API-ul Zevo STT operează la 16000 Hz. Conversia între aceste formate se realizează cu modulul `audioop` din biblioteca standard Python. Pentru trimiterea audio sintetizat către apelant, rata de eșantionare este redusă de la 22050 Hz la 8000 Hz prin interpolare liniară (`audioop.ratecv`), după care eșantioanele PCM sunt convertite în codare μ-law (`audioop.lin2ulaw`) și codificate Base64 pentru transmisie prin WebSocket. În direcția inversă, audio-ul primit de la Twilio în format μ-law este convertit la PCM și resamplingul de la 8000 Hz la 16000 Hz înainte de a fi trimis la Zevo STT, conform Listingului 4.1.

```
# TTS -> Twilio: resampling 22050 Hz -> 8000 Hz, PCM -> mu-law
pcm_8k, _ = audioop.ratecv(pcm_22k, src_width, 1, 22050, 8000, None)
mulaw     = audioop.lin2ulaw(pcm_8k, 2)
payload   = base64.b64encode(mulaw).decode("utf-8")

# Twilio -> STT: mu-law -> PCM, resampling 8000 Hz -> 16000 Hz
pcm_8k     = audioop.ulaw2lin(mulaw_payload, 2)
pcm_16k, _ = audioop.ratecv(pcm_8k, 2, 1, 8000, 16000, None)
```

Listing 4.1. Conversia audio între formatul TTS (22050 Hz PCM) și formatul Twilio (8000 Hz μ-law), în ambele direcții.

A doua modificare principală a vizat integrarea modulului de dialog în contextul IVR. Deoarece `chatbot_meu.py` expune toate interacțiunile vocale exclusiv prin funcțiile `say_and_display` și `listen_and_transcribe`, integrarea s-a realizat prin tehnica *monkey-patching* (înlocuirea dinamică a funcțiilor): la inițierea fiecărui apel, cele două funcții sunt înlocuite dinamic cu versiunile IVR, fără nicio modificare a logicii interne a modulului. Funcția `ivr_say` apelează serviciul TTS prin HTTP intern, aplică conversia de format descrisă anterior și trimite audio-ul prin WebSocket-ul Twilio; funcția `ivr_listen` recepteză cinci secunde de audio de la apelant, aplică conversia inversă și transcrie cu Zevo STT. Odată înlocuite cele două funcții, întregul flux de dialog funcționează nemodificat prin canalul telefonic:

```
bot.say_and_display       = ivr_say
bot.listen_and_transcribe = ivr_listen
bot.main()
```

Listing 4.2. Redirecționarea I/O-ului modulului de dialog către canalul IVR prin monkey-patching.

4.4. Interfața de monitorizare și demonstrație

În scop demonstrativ, serverul IVR a fost extins cu o interfață web de monitorizare a conversațiilor în timp real, accesibilă la endpointul `/demo`. Pagina interoghează periodic

endpointul `/conversation` (la interval de o secundă), care returnează un obiect JSON cu starea curentă a apelului și istoricul mesajelor acumulate în sesiunea activă. La fiecare apel nou, istoricul este reinițializat, asigurând că interfața reflectă exclusiv conversația curentă. Mesajele botului și ale apelantului sunt afișate în stilul unei conversații de tip chat, cu indicarea orei și a rolului fiecărui participant, permițând urmărirea în timp real a fluxului conversațional fără acces la terminalul serverului.

Figura 4.2 ilustrează interfața în timpul unui apel de test în care apelantul solicită o comandă: botul extrage tipul și dimensiunea pizzei prin GPT-4.1, gestionează iterativ adăugarea de pizza suplimentare și confirmă în final comanda. Interfața a fost utilizată în cadrul sesiunilor de testare pentru a verifica corectitudinea transcrierilor și a răspunsurilor generate, fără a necesita acces direct la log-urile serverului.

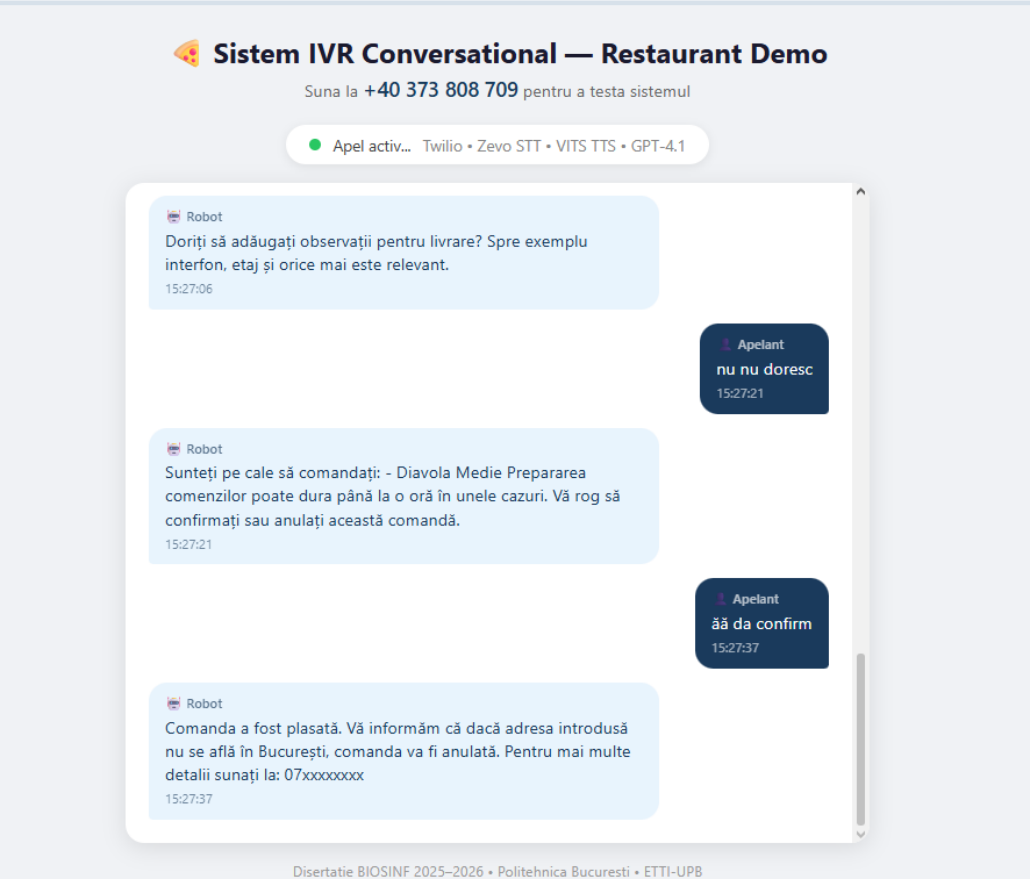


Figura 4.2. Interfața web de monitorizare a conversației IVR în timp real. Mesajele sintetizate de robot apar în stânga (albastru deschis), iar transcrierile vocii apelantului în dreapta (albastru închis).

4.5. Limitări și concluzii

Calitatea audio percepută în cadrul unui apel telefonic este în mod inerent inferioară calității obținute în condiții de laborator. Banda de frecvență disponibilă în rețeaua telefonică clasică se limitează la 4000 Hz (la o rată de eșantionare de 8000 Hz), față de 11025 Hz disponibili la rata de

eșantionare de 22050 Hz utilizată de modelul TTS. Compresia suplimentară introdusă de codarea G.711 [12] μ-law amplifică această degradare. Această limitare este inerentă oricărui sistem TTS integrat într-o platformă de telefonie clasică și nu reflectă calitatea intrinsecă a modelului, evaluată separat prin metodele descrise în Capitolul 2.

În ansamblu, sistemul IVR implementat demonstrează viabilitatea integrării componentelor dezvoltate într-un pipeline conversațional funcțional end-to-end. Tehnica de monkey-patching aplicată modulului de dialog a permis reutilizarea integrală a logicii conversaționale fără modificări ale codului intern. Sistemul a fost validat end-to-end printr-un apel telefonic real, iar interfața de monitorizare a facilitat urmărirea fluxului conversațional în timpul sesiunilor de testare.

5. CONCLUZII ȘI DIRECȚII VIITOARE DE DEZVOLTARE

Lucrarea de față a propus proiectarea și implementarea unui sistem vocal conversațional destinat domeniului HoReCa, cu accent pe optimizarea procesării limbii române pe infrastructură locală. Sistemul integrează modular trei componente principale (sinteza vocală (TTS), înțelegere a limbajului natural (NLU) și un server telefonic interactiv (IVR)), fiecare fiind evaluată independent și ulterior unificată într-un prototip funcțional accesibil prin apel telefonic real.

5.1. Concluzii pe componente implementate

5.1.1. Sinteza vocală (TTS)

Componenta de sinteză vocală a fost construită pornind de la arhitectura VITS [13], antrenată de la zero pe corpusul SWARA consolidat, un set de date de vorbire românească cuprinzând 41 de vorbitori și 2.792 de propoziții unice după deduplicare. Experimentele au demonstrat că preprocesarea riguroasă a datelor zgomotoase din versiunea SWARA 2.0 (reducere de zgomot prin subtracție spectrală adaptivă) aduce un câștig acustic superior față de simpla schimbare a reprezentării lingvistice din grafeme în foneme IPA.

Pentru adresarea vocabularului lipsă (OOV), reprezentat de terminologia italiană din meniu, s-a identificat fenomenul de word-sticking (lipirea termenilor străini de modificatorii lexicali adiacenți) ca o limitare arhitecturală intrinsecă a modelelor VITS end-to-end. Strategia de augmentare prin distilarea cunoștințelor vocale dintr-un model comercial (ElevenLabs Professional Voice Clone) a permis corectarea pronunției, determinând o scădere drastică a WER-ului pe categoria comenzilor de pizza de la 74,6% la 29,7% (respectiv 26,5% cu dicționar fonetic). Experimentele cu dicționare fonetice au arătat că substituțiile de tip single-token îmbunătățesc moderat inteligibilitatea, în timp ce despărțirea cuvintelor în tokeni multipli destabilizează predictorul de durate.

5.1.2. Înțelegerea limbajului natural (NLU)

Evaluarea comparativă a componentei NLU pe un set de test de 100 de solicitări în limba română a evidențiat un decalaj substanțial (de peste 75 de puncte procentuale) între modelele comerciale cloud și cele open-weights de dimensiuni reduse rulate local pe CPU. Prin aplicarea ingineriei iterative a prompturilor, modelul Claude Sonnet 4.5 guvernat de o rubrică procedurală strictă (P3) a atins o acuratețe strictă de înregistrare de 94%, stabilizând extragerea sloturilor

temporale. În schimb, modelele locale (gemma2:2b, llama3.2:3b, qwen2.5:3b via Ollama) s-au dovedit inviabile pentru scenariul de deployment în timp real, înregistrând o latență prohibitivă de 13–18 secunde per exemplu și erori sistematice de formatare structurală și canonicalizare a entităților.

5.1.3. Integrarea serverului IVR

Modulele au fost integrate cu succes într-un server IVR funcțional conectat la platforma Twilio [26], utilizând serviciul Zevo STT [29] pentru recunoașterea vorbirii. Integrarea modulului de dialog s-a realizat curat prin tehnica de monkey-patching a funcțiilor abstracte de I/O, permițând reutilizarea neschimbată a logicii conversaționale în contextul canalului telefonic WebSocket. Incompatibilitățile severe de format audio dintre componente (sinteză PCM la 22.050 Hz, Twilio G.711 μ-law la 8.000 Hz și STT la 16.000 Hz) au fost rezolvate stabil prin rutine de conversie și resampling adaptiv realizate cu modulul `audioop` din biblioteca standard Python.

5.2. Contribuții originale

Principalele contribuții personale ale acestei lucrări sunt:

- **Optimizarea pipeline-ului acustic VITS:** Realizarea căutării hiperparametrilor pe patru configurații experimentale utilizând corpusul consolidat SWARA (60 de ore, 41 de vorbitori), demonstrând impactul pozitiv al filtrării acustice avansate și al eliminării enunțurilor eliptice asupra convergenței modelului.
- **Metodologie de augmentare targetată pentru termeni OOV:** Conceperea unei strategii de distilare a cunoștințelor vocale capabilă să corecteze vocabularul de specialitate cu doar 90 de eșantioane sintetice, validată cantitativ prin scăderea WER de la 74,6% la 39,9%.
- **Izolarea și validarea bilingvă a limitărilor de decodificare:** Caracterizarea fenomenului de word-sticking și introducerea unei metodologii de validare ASR bilingvă (română-italiană) pentru a disocia erorile acustice brute de penalizările contextuale ale modelelor de limbă monotone.
- **Validare perceptuală MUSHRA:** Proiectarea și implementarea unei platforme web de testare subiectivă cu 17 evaluatori umani, confirmând statistic ($p < 0,001$) utilitatea augmentării prin obținerea unui scor global de 78,3/100.
- **Studiu comparativ cloud vs. edge pe sarcini NLU:** Construirea unui set de date de test de complexitate ridicată (80% enunțuri colocviale/eliptice) și maparea benchmark-ului de performanță și latență pentru 7 arhitecturi LLM sub constrângeri stricte de format JSON.
- **Dezvoltarea arhitecturii integrate IVR:** Construirea serverului gateway cu expunerea TTS ca serviciu web HTTP independent (RTF mediu excelent de 0,022), rezolvarea dynamic-routing-ului audio în Python și implementarea unei interfețe web de monitorizare în timp real a apelurilor active.

5.3. Limitări și direcții viitoare de dezvoltare

Prototipul actual prezintă câteva limitări care pot constitui puncte de plecare pentru cercetări ulterioare:

a) **Lățimea de bandă audio:** Calitatea audio percepută în telefonie este limitată la 8.000 Hz în format μ-law, ceea ce degradează rezoluția nativă de 22.050 Hz a modelului TTS. O direcție viitoare este tranziția de la telefonia clasică la protocoale WebRTC cu codec Opus pentru a păstra fidelitatea semnalului neural.

b) **Detecția dinamică a activității vocale (VAD):** Fereastra fixă de înregistrare de 5 secunde utilizată în prezent trunchiază răspunsurile lungi sau adaugă latență inutilă. Integrarea unui modul VAD local ar permite determinarea dinamică a finalului discursului apelantului.

c) **Controlul explicit al granițelor prozodice:** Modificarea encoderului VITS sau reducerea dimensiunii `hidden_channels` de la 192 la 128 pentru a exercita un efect de regularizare mai puternic pe corpusuri multi-speaker și a preveni fenomenul de word-sticking.

d) **Optimizarea NLU local offline:** Testarea și optimizarea unor modele open-weights de dimensiuni medii (ex. Llama 3.1 8B) prin tehnici de cuantizare avansată (GGUF, AWQ) rulate pe acceleratoare grafice dedicate la marginea rețelei (edge computing) pentru a aduce latența locală sub o secundă.

e) **Persistența datelor:** Baza de cunoștințe este stocată în prezent într-un fișier JSON, soluție adecvată pentru prototip dar limitată pentru un sistem de producție. Migrarea la o bază de date relațională ar permite gestionarea concurentă a rezervărilor și scalabilitatea sistemului.

În ansamblu, lucrarea demonstrează că un sistem vocal conversațional funcțional pentru limba română poate fi construit pe infrastructură locală accesibilă, cu componente open-source și modele comerciale accesate prin API. Rezultatele obținute confirmă viabilitatea abordării pentru domeniul HoReCa, iar direcțiile identificate oferă un punct de plecare concret pentru dezvoltarea unui sistem de producție.

BIBLIOGRAFIE

[1] Mathieu Bernard, Jamil Tiah, „Phonemizer: Text to Phones Phonemization Framework in Python", *Journal of Open Source Software*, vol. 6, nr. 68, p. 3954 2021.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, „Language Models are Few-Shot Learners", *Advances in Neural Information Processing Systems (NeurIPS)* 2020.

[3] Edresson Casanova, Julian Weber, Christopher Shulby, Arnaldo Candido Junior, Eren Gölge, Moacir Antonelli Ponti, „YourTTS: Towards Zero-Shot Multi-Speaker TTS and Zero-Shot Voice Conversion for everyone", *Proceedings of the 39th International Conference on Machine Learning (ICML)* Proceedings of Machine Learning Research, vol. 162, PMLR, pp. 2709–2720 2022.

[4] M. Cohn, G. Zellou, „Perception of concatenative vs. neural text-to-speech (TTS): Differences in intelligibility in noise and language attitudes", *Proceedings of Interspeech*, pp. 1733–1737 2020.

[5] Brecht Desplanques, Jenthe Thienpondt, Kris Demuynck, „ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification", *Interspeech* 2020.

[6] Reece H. Dunn, *eSpeak NG Text-to-Speech*, `https://github.com/espeak-ng/espeak-ng`. [Online] Versiunea 1.52.0.1, accesat 2026.

[7] Aaron Grattafiori et al., „The Llama 3 Herd of Models", *arXiv preprint arXiv:2407.21783*, vol. nr. 2024.

[8] Michael Hansen, *Piper: A fast, local neural text to speech system*, `https://github.com/rhasspy/piper`. [Online] [Online; accesat Ianuarie 2026].

[9] Mutian He, Philip N. Garner, „Can ChatGPT Detect Intent? Evaluating Large Language Models for Spoken Language Understanding", *arXiv preprint arXiv:2305.13512*, vol. nr. 2023.

[10] Idiap Research Institute, *Coqui TTS: A deep learning toolkit for Text-to-Speech*, `https://github.com/idiap/coqui-ai-TTS`. [Online] [Accesat: Ian. 2026].

[11] ITU-R, *Method for the subjective assessment of intermediate quality levels of coding systems*, Recommendation BS.1534-3, International Telecommunication Union, 2015, Disponibil: https://www.itu.int/rec/R-REC-BS.1534/en.

[12] ITU-T, *Pulse code modulation (PCM) of voice frequencies*, Recommendation G.711, International Telecommunication Union, 1988, Disponibil: https://www.itu.int/rec/T-REC-G.711/en.

[13] Jaehyeon Kim, Jungil Kong, Juhee Son, „Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech", *Proceedings of the 38th International Conference on Machine Learning (ICML)*, PMLR, pp. 5530–5540 2021.

[14] Jungil Kong, Jaehyun Kim, Jaekyoung Bae, „HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis", *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, Curran Associates, Inc., pp. 17022–17033 2020.

[15] Michael Xieyang Liu, Frederick Liu, Alexander Fiannaca, Terry Y. Koo, Lucas Dixon, Michael A. Terry, „"We Need Structured Output": Towards User-centered Constraints on Large Language Model Output", *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '24)* 2024.

[16] T. Liu, „Clip-TTS: Contrastive Text-Content and Mel-Spectrogram, A High-Quality Text-to-Speech Method Based on Contextual Semantic Understanding", *arXiv preprint arXiv:2502.18889*, vol. nr. 2025.

[17] Y. Liu, Z. Wang, B. Zhao, S. Fan, L. Zou, „A Mel Spectrogram Enhancement Paradigm Based on CWT in Speech Synthesis", *Applied Sciences*, vol. 12, nr. 15, p. 7829 2022.

[18] B. Lőrincz, A. Stan, M. Giurgiu, „An objective evaluation of the effects of recording conditions and speaker characteristics in multi-speaker deep neural speech synthesis", *Procedia Computer Science*, vol. 192, nr. Pp. 2501–2510 2021.

[19] Microsoft Azure, *Azure OpenAI Service Documentation*, https://learn.microsoft.com/en-us/azure/ai-services/openai/. [Online] [Accesat: Iun. 2026].

[20] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, Ilya Sutskever, „Robust Speech Recognition via Large-Scale Weak Supervision", *International Conference on Machine Learning (ICML)* 2023.

[21] Mirco Ravanelli et al., *SpeechBrain: A General-Purpose Speech Toolkit*. [Online].

[22] Teodora Răgman, Adrian Bogdan Stânea, Horia Cucu, Adriana Stan, „How Open is Open TTS? A Practical Evaluation of Open Source TTS Tools", *IEEE Access*, vol. 13, nr. Pp. 203415–203428 2025.

[23] A. Stan, F. Dinescu, C. Tiple, Ş. Meza, B. Orza, M. Chirilă, M. Giurgiu, „The SWARA Speech Corpus: A Large Parallel Romanian Read Speech Dataset", *Proceedings of the 9th International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pp. 1–6 2017.

[24] Sudeshnm, *Coqui TTS: Deep Dive Into an Open-Source Text-to-Speech Framework*, Medium - Towards Data Science. [Online] Disponibil: https://medium.com/@sudeshnm/coqui-tts-deep-dive-into-an-open-source-text-to-speech-framework-3456789 [Accesat: Ian. 2026].

[25] X. Tan, T. Qin, F. Soong, T. Y. Liu, „A survey on neural speech synthesis", *arXiv preprint arXiv:2106.15561*, vol. nr. 2021.

[26] Twilio Inc., *Twilio Voice: Programmable Voice Documentation*, https://www.twilio.com/docs/voice. [Online] [Accesat: Iun. 2026].

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, „Attention Is All You Need", *Advances in Neural Information Processing Systems (NeurIPS)* 2017.

[28] H. Zen, K. Tokuda, A.W. Black, „Statistical parametric speech synthesis", *Speech Communication*, vol. 51, nr. 11, pp. 1039–1064 2009.

[29] Zevo Technologies, *Zevo Live STT API Documentation*, https://drive.google.com/file/d/1iOck17DKhxJ_dciFGGJhVCD9spt9Z4PG/view. [Online] [Accesat: Iun. 2026].

[30] Zevo Technology SRL, *Ghid de utilizare Zevo Live TTS API, Versiunea 1.4*, Accesat în iunie 2026, 2023, Disponibil: https://zevo-tech.com.