UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

RESTAURAREA DE DIACRITICE ÎN FIȘIERE TEXT COMPLEXE

LUCRARE DE DIPLOMĂ

Prezentată ca cerinţă parţială pentru obţinerea
titlului de *Inginer*
în domeniul *Electronică, Telecomunicaţii şi Tehnologia Informaţiei*
programul de studii *Rețele și Software pentru Telecomunicații*

Profesori coordonatori:
Ș.L. Horia Cucu
Prof. Dr. Ing. Corneliu Burileanu

Student:
Andra-Irina Ivan

București
2016

Anexa 1

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul Telecomunicaţii

Aprobat Director de Departament :
Prof. Dr. Ing. Silviu CIOCHINĂ

TEMA PROIECTULUI DE DIPLOMĂ
a studentei IVAN P. Andra-Irina, 442D

1. Titlul temei: **Restaurarea de diacritice în fişiere text complexe**

2. Contribuţia practică, originală a studentului va consta în:
Pornind de la un exemplu de serviciu web de restaurare de diacritice deja existent, se doreşte realizarea unui nou serviciu web, independent de primul, care sa permită restaurarea de diacritice pentru documentele Word de tip .docx. Utilizatorului i se va permite, prin intermediul unei interfeţe atragătoare, să uploadeze documentul dorit apăsând un buton, să pornească restaurarea de diacritice apăsând un al doilea buton şi să downloadeze rezultatul utilizând un al treilea buton. De asemenea, i se va permite, dacă acesta va dori, să trimită documentul rezultat, ce prezintă diacritice, utilizând serviciul de e-mail, unui alt utilizator.
În plus, se mai doreşte eloborarea unei aplicaţii Android care să realizeze restaurarea de diacritice. I se va permite utilizatorului să introducă un text cu ajutorul tastaturii virtuale şi să pornească restaurarea de diacritice prin apăsarea unui buton. La finalizarea restaurării aplicaţia va întoarce textul cu diacritice pe care utilizatorul îl va putea utiliza mai departe.

3. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline: Programarea calculatoarelor, Programare orientată pe obiecte, Tehnici de programare în internet.

4. Realizarea practică/ proiectul rămân în proprietatea: *Laborator Cercetare SpeeD, student Ivan P. Andra-Irina*

5. Proprietatea intelectuală asupra proiectului aparţine: *Laborator Cercetare SpeeD, student Ivan P. Andra-Irina*

6. Locul de desfăşurare a activităţii: UPB - ETTI, Sala 3 de Calculatoare

7. Data eliberării temei: august 2015

CONDUCĂTOR LUCRARE: **STUDENT:**

Ş.L. Horia CUCU *Andra-Irina IVAN*
Prof. Dr. Ing. Corneliu BURILEANU

DECLARAȚIE DE ONESTITATE ACADEMICĂ

Prin prezenta declar că lucrarea cu titlul "Restaurarea de diacritice în fişiere text complexe", prezentată în cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei a Universităţii „Politehnica" din Bucureşti ca cerinţă parţială pentru obţinerea titlului de *Inginer* in domeniul Inginerie Electronică şi Telecomunicaţii, programul de studii *Reţele şi Software pentru Telecomunicaţii* este scrisă de mine şi nu a mai fost prezentată niciodată la o facultate sau instituţie de învăţământ superior din ţară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referinţe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi fac referinţă la sursă. Reformularea în cuvinte proprii a textelor scrise de către alţi autori face referinţă la sursă. Înţeleg că plagiatul constituie infracţiune şi se sancţionează conform legilor în vigoare.

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

Bucureşti, *28.06.2016*

Andra-Irina Ivan

Cuprins

Listă de Figuri 9
Listă de Tabele 10
Listă de Acronime 11
Introducere 13
Motivația lucrării 13
Obiectivele lucrării 15
CAPITOLUL 1 Noțiuni teoretice privind restaurarea diacriticelor 17
1.1 Modele de limbă statistice 17
1.1.1 Construcția modelelor de limbă bazate pe n-grame 18
1.1.2 Principala problemă asociată construcției modelelor de limbă de tip n-gram și abordarea ei 18
1.1.3 Evaluarea modelelor de limbă 19
1.2 Descrierea metodei folosite pentru restaurarea diacriticelor 20
1.3 Alte metode de restaurare a diacriticelor 21
1.4 Performanțele metodei folosite pentru restaurarea diacriticelor 23
CAPITOLUL 2 Tehnologii software utilizate în cadrul realizării serviciului web 27
2.1 Tehnologii de programare 27
2.1.1 Java 27
2.1.2 JavaScript 28
2.1.2.1 jQuery 28
2.1.2.2 AJAX 29
2.1.3 HTML 30
2.1.4 CSS 31
2.1.5 Bootstrap 31
2.2 Protocoale 32
2.2.1 HTTP 32
2.2.2 SMTP 33
2.3 Stilul arhitectural - REST 33
2.4 Biblioteci utilizate în cadrul procesului de restaurare a diacriticelor pentru fișiere .docx 34
2.4.1 Apache POI 34
2.4.2 SRI-LM 34
2.4.3 JavaNLP 36
2.5 Server-ul de aplicații - GlassFish 36
CAPITOLUL 3 Serviciul web de restaurare a diacriticelor 37
3.1 Descriere generală 37
3.2 Arhitectura sistemului și modul de funcționare 40

3.3 Implementarea sistemului 41
3.3.1 Implementarea paginii web 41
3.3.2 Implementarea blocului „Prelucrare fișier .docx” 43
3.3.3 Implementarea blocului „Trimitere fișier rezultat pe mail” 44
3.3.4 Implementarea blocului ce se ocupă de ștergere 45
3.4 Serviciu RESTful 46
3.5 Problemele apărute în timpul proiectării serviciului web și rezolvarea lor 47
3.6 Situații în care nu se realizează restaurarea diacriticelor 48
3.7 Studiu durată inițializare disambig 50
CAPITOLUL 4 Aplicația Android 53
4.1 Descriere generală 53
4.2 Componentele unei aplicații Android 57
4.3 Implementarea aplicației 58
4.4 Problemele aplicației și rezolvarea lor 63
Concluzii 65
Contribuții Personale 66
Activitate ulterioară 67
Bibliografie 69
Anexa 1 73
Anexa 2 89

LISTĂ DE FIGURI

Figura 1.1 Arhitectura sistemului de restaurare de diacritice, Sursa [3].....22
Figura 2.1 AJAX – Modul de funcționare, Sursa [14].....29
Figura 3.1 Utilizatorul introduce o informație incorectă în câmpul destinat adresei de e-mail....38
Figura 3.2 Utilizatorul este înștiințat de faptul că procesul de restaurare este în curs de desfășurare 38
Figura 3.3 Utilizatorul este înștiințat de faptul că poate descărca fișierul rezultat 38
Figura 3.4 Mesajul de eroare 38
Figura 3.5 Scenariul de utilizare al serviciului web de restaurare a diacriticelor pentru fișiere .docx.....39
Figura 3.6 Arhitectura sistemului 40
Figura 3.7 Modul de funcționare al sistemului 40
Figura 3.8 Structura unui document .docx.....44
Figura 3.9 Fișierul .docx fără diacritice 49
Figura 3.10 Fișierul .docx restaurat 49
Figura 3.11 Evoluția timpului necesar restaurării.....51
Figura 4.1 Schema bloc a sistemului de restaurare a diacriticelor.....54
Figura 4.2 Interfața aplicației.....54
Figura 4.3 Mesajele de informare asupra motivelor pentru care procesul de restaurare nu poate fi pornit sau pentru care nu se poate trece la etapa următoare 55
Figura 4.4 Desfășurarea și finalizarea procesului de restaurare 56
Figura 4.5 Rezultatul apăsării butonului „Trimite mail!”.....56
Figura 4.6 Ierarhie de obiecte de tipul View sau ViewGroup, Sursa [45].....58
Figura 4.7 Iconița „Options.....59
Figura 4.8 Ierarhia porțiunii destinate introducerii textului dorit și afișării rezultatului obținut..59

LISTĂ DE TABELE

Tabel 1.1 Fragment dintr-o hartă probabilistică, Sursa [3].....21

Tabel 1.2 Performanțele la nivel de caracter ale metodei de restaurare folosite, Sursa [3].....24

Tabel 1.3 Comparația rezultatelor metodei folosite cu cele ale celorlalte metode prezentate, Sursa [3].....24

Tabel 3.1 Rezultatele experimentului 51

LISTĂ DE ACRONIME

AJAX – Asynchronous JavaScript and XML
API – Application Programming Interface
CDN – Content Delivery Network
CEO – Chief Executive Officer
ChER – Character Error Rate
CSJS - Client-Side JavaScript
CSS – Cascading Style Sheets
EE – Enterprise Edition
HTML – HyperText Markup Language
HTTP – Hypertext Transfer Protocol
ID – Identifier
JVM – Java Virtual Machine
LM – Language Model
MIME – Multipurpose Internet Mail Extensions
MS – Microsoft
MSA – Mail Submission Agent
MTA – Mail Transfer Agent
MUA – Mail User Agent
NLP – Natural Language Processing
OOV – Out Of Vocabulary
POS – Part Of Speech
REST – Representational State Transfer
SMS – Short Message Service
SMTP – Simple Mail Transfer Protocol
SSJS - Server-Side JavaScript
SSL – Secure Sockets Layer
TLS – Transport Layer Security
TTS – Text To Speech
UML – Unified Modeling Language
URI – Uniform Resource Identifier
URL – Uniform Resource Locator
WAR – Web Archive File
WER – Word Error Rate
WORA – Write Once Run Anywhere

WWW – World Wide Web

W3C – World Wide Web Consortium

XML – EXtensible Markup Language

INTRODUCERE

MOTIVAȚIA LUCRĂRII

În Dicționarul explicativ al limbii române un semn diacritic este definit ca un „semn grafic care dă unei litere a alfabetului o valoare specială”. Wikipedia oferă însă o definiție mai explicită, potrivit căreia un semn diacritic este „un semn tipografic adăugat la o literă pentru a indica o diferență în pronunție sau pentru a deosebi sensurile a două cuvinte altfel scrise identic”.

Scrierea fără diacritice, de cele mai multe ori, schimbă sensul unor cuvinte sau propoziții, ducând astfel la exprimări ambigue și/sau cu un înțeles total diferit. Având în vedere aceste lucruri, pentru a se evita posibilitatea ca un document să poată avea mai multe interpretări, au fost elaborate și adoptate legi potrivit cărora este obligatoriu ca textul materialelor emise de instituțiile statului să fie scris corect, cu toate semnele diacritice necesare. De asemenea, scrierea fără diacritice poate avea drept consecință obținerea unor cuvinte care nu există în limba română, cum ar fi: „dictionar”, „tantar”, „catel”. Cele trei exemple anterioare reprezintă varianta fără diacritice a cuvintelor: „dicționar”, „țânțar”, „cățel”[1].

Literele cu semne diacritice folosite în limba română sunt: Ă, Â, Î, Ș, Ț alături de corespondentele lor minuscule ă, â, î, ș, ț. Din punct de vedere grafic, ș și ț au asociate două forme: cea cu virgulă (ș, ț) și cea cu sedilă (ş, ţ). În 2003 Institutului de Lingvistică al Academiei Române a stabilit că varianta corectă este cea cu virgulă, care se folosește exclusiv în limba română. Însă, având în vedere faptul că anumite sisteme nu pot afișa corect varianta

cu virgulă, aceasta este adesea înlocuită cu cea cu sedilă care este specifică unor limbi precum turca.

Semnele diacritice sunt prezente în toate textele ce se regăsesc în materialele scrise, însă nu același lucru se poate afirma și în cazul textelor disponibile prin Internet ce sunt accesate de către diferite aplicații software. De exemplu, există site-uri de știri ce nu folosesc diacritice, precum protv.ro Făcând abstracție de faptul că textele fără diacritice pot conține expresii ambigue și prin urmare nu pot fi înțelese pe deplin, utilizarea lor de către anumite aplicații sau sisteme care pun accentul pe folosirea/generearea unor texte corecte, cu toate semnele diacritice necesare, presupune apelarea la o metodă de inserare automată a diacriticelor. O astfel de metodă poate fi folosită, de exemplu, pentru a restaura diacriticele textului generat de un sistem de recunoaștere automată de vorbire ce nu folosește diacritice.

La ora actuală există numeroase servicii web care oferă posibilitatea realizării conversiei unui text fără diacritice într-un text cu diacritice. Printre acestea se numără binecunoscutele site-uri diacritice.com și diacritice.opa.ro ale căror servicii pot fi accesate numai prin intermediul interfeței grafice. Acest lucru face dificilă apelarea funcționalității oferite de către alte aplicații software sau sisteme precum cel precizat anterior.

Printre dezavantajele serviciilor web amintite se numără faptul că este impusă o limită în ceea ce privește numărul de caractere (și prin urmare, numărul de cuvinte) al textului introdus. Astfel, în cazul în care utilizatorul accesează serviciul pentru a restaura un text de mari dimensiuni (de exemplu patruzeci – cincizeci de pagini), acesta se va vedea nevoit să introducă textul pe porțiuni, ceea ce va provoca un discomfort nu numai din cauza faptului că va trebui să repete aceeași operație de mai multe ori ci și din cauza faptului că acest lucru va însemna un timp de așteptare mai mare până la obținerea întregului rezultat dorit.

Un alt dezavantaj este reprezentat de cel referitor la faptul că aceste servicii web nu iau în considerare aspectul textului introdus (culoare, font, dimensiune). De cele mai multe ori aceste sisteme sunt accesate în scopul facilitării operației de întocmire a unui referat, a unei lucrări științifice, etc. care sunt redactate folosind un procesor de text (precum Microsoft Word) ce oferă o gamă largă de posibilități de formatare. În cazul unui text ce prezintă numeroase porțiuni cu aspect diferit, utilizarea unor astfel de servicii web nu va simplifica munca utilizatorului, ci mai mult o a va îngreuna deoarece acesta va fi nevoit să reformateze fiecare porțiune de text în parte. Se observă astfel necesitatea unui serviciu sau a unei aplicații care să permită restaurarea diacriticelor pentru un text oricât de mare și care să țină cont, în același timp, de aspectul acestuia. Astfel lucrarea de față propune un serviciu web de restaurare de diacritice care să elimine cele două dezavantaje menționate anterior și care este destinat fișierelor cu extensia .docx.

Din 2007, când CEO-ul Apple, Steve Jobs, a lansat primul smartphone modern caracterizat printr-un design simplu și o putere de procesare impresionantă pentru un dispozitiv mobil, și până în 2010, vânzările de smartphone-uri le-au depășit pe cele de calculatoare, ajungându-se ca în 2014 să se înregistreze peste opt sute de milioane de unități vândute la nivel mondial[2].

Având o dimensiune mai mică ca a unui laptop și o putere de procesare asemănătoare desktop-urilor, smartphone-urile au devenit indispensabile căci, pe lângă faptul că oferă posibilitatea unei interacțiuni în timp real între două persoane aflate la sute și chiar mii de kilometri distanță, ele prezintă și alte funcționalități care până nu demult erau specifice computerelor precum: navigarea pe Internet, accesarea adresei de e-mail, partajarea de documente, efectuarea de cumpărături online, etc.

Având în vedere importanța pe care au căpătat-o smartphone-urile în viața unei persoane, precum și puterea mare de procesare a acestora, teza de față mai propune o aplicație Android,

cu scop demonstrativ, ce pune în evidență capacitatea acestor telefoane mobile evoluate de a comunica cu alte aplicații și sisteme externe. Trebuie menționat faptul că aplicația prezentată are drept scop restaurarea textului introdus de utilizator, permițându-i apoi acestuia să trimită mai departe textul restaurat prin folosirea serviciului de poștă electronică.

OBIECTIVELE LUCRĂRII

Având în vedere cele menționate anterior, această lucrare de diplomă își propune realizarea și descrierea unui serviciu web de restaurare de diacritice pentru fișiere .docx dar și a unei aplicații android de restaurare de diacritice.

Serviciul web prezentat poate fi accesat de către alte aplicații sau sisteme prin intermediul protocolului HTTP și este destinat restaurării diacriticelor pentru fișiere .docx ce pot conține nu numai text ci și imagini și tabele simple, textul acestora din urmă fiind și el restaurat. Utilizatorul poate accesa serviciul web prin intermediul unei interfețe grafice aspectuoase și ușor de utilizat iar fișierul .docx rezultat este trimis la o adresă de e-mail specificată de utilizator dar poate fi descărcat și direct, printr-un simplu click, dacă utilizatorul va dori să aștepte până la finalizarea procesului de restaurare.

În ceea ce privește aplicația Android, aceasta oferă posibilitatea utilizatorului de a introduce un text fără diacritice prin intermediul tastaturii virtuale, ce va fi transformat, după apăsarea unui buton, într-un text cu diacritice de către un serviciu web cu care aplicația comunică prin protocolul HTTP. Utilizatorul va fi înștiințat de faptul că procesul de restaurare este în curs de desfășurare, rezultatul fiind afișat pe ecranul smartphone-ului.

În primul capitol se va descrie metoda folosită pentru restaurarea diacriticelor și se va compara cu alte metode ce vor fi și ele menționate pe scurt. Metoda prezentată realizează conversia unui text fără diacritice într-unul cu diacritice cu ajutorul unui model de limbă și a unei hărți a cuvintelor. Dintre cele două, se va pune accentul pe modelul de limbă, prezentându-se modul de construcție, problemele asociate precum și factorii săi de evaluare.

Cel de-al doilea capitol va fi dedicat tehnologiilor software și librăriilor folosite pentru dezvoltarea serviciului web. Se vor specifica anumite informații asupra acestora iar detaliile referitoare la contribuția lor în realizarea aplicației vor fi oferite în capitolul trei.

În capitolul trei și patru se vor prezenta în detaliu serviciul web de restaurare de diacritice și aplicația Android. În cazul serviciului web, accentul se va pune pe modul de funcționare și pe modul de implementare, fiind prezentate în detaliu anumite componente mai importante iar în cazul aplicației Android, în prim-plan va fi modul de implementare.

Această lucrare va fi finalizată prin prezentarea unor concluzii generale, a contribuțiilor personale precum și a activității ulterioare.

CAPITOLUL 1

NOȚIUNI TEORETICE PRIVIND RESTAURAREA DIACRITICELOR

1.1 MODELE DE LIMBĂ STATISTICE

Un model de limbă estimează probabilitatea ca o secvență de cuvinte C=c_1, c_2, c_3, ..., c_n să fie o secvență validă a limbii sursă (în cazul tezei de față, a limbii române). Mai exact, având în vedere următoarele propoziții: „Catelusul are blanita alba” și „Cățelușul are blănița albă”, modelul de limbă are rolul de a asocia o probabilitatea mai mare celei de-a doua propoziții[3]. Acest lucru este bineînțeles firesc deoarece prima propoziție conține cuvinte ce nu aparțin limbii române precum „catelusul”, „blanita” și „alba”. Varianta corectă a acestora, ce se regăsește și în Dicționarul exlicativ al limbii române este: „cățelușul”, „blănița” și „albă”, ele fiind prezente în cea de-a doua propoziție.

Pentru a determina probabilitatea p(C), asociată secvenței de cuvinte C=c_1, c_2, c_3, ..., c_n, se folosește următoarea formulă[3]:

$$p(C) = p(c_1, c_2, c_3, \dots, c_n) = p(c_1)p(c_2|c_1) \dots p(c_n|c_1, c_2, \dots, c_{n-1}) \qquad \textbf{(1.1)}$$

Prin urmare, calcularea probabilității secvenței de cuvinte *C* se bazează pe determinarea, pentru fiecare cuvânt c_i cu i=1...n, a probabilității asociate ținând cont de cele i-1 cuvinte precedente, mai precis de o istorie a cuvintelor care îl preced. Având în vedere că istoria cuvintelor precedente nu poate cuprinde un număr infinit de elemente, aceasta va fi limitată la un număr *m* de cuvinte. Astfel, au apărut așa numitele modele de limbă bazate pe n-grame. Alegerea lui *m* depinde de dimensiunea corpusurilor de antrenare, cu cât acestea sunt mai mari, cu atât se poate crea o istorie mai mare și prin urmare și *m* va fi mai mare. Cele mai folosite modele de limbă sunt cele bazate pe trigrame, în cazul cărora, pentru estimarea unui cuvânt este necesară o istorie formată din două cuvinte. Acest lucru presupune existența unei colecții statistice asociate unor secvențe de trei cuvinte, așa numitele 3-grame (trigrame)[3].

1.1.1 Construcția modelelor de limbă bazate pe n-grame

Pentru construirea unui model de limbă bazat pe n-grame se folosește un corpus de text suficient de mare și calculul de probabilități cu ajutorul principiului probabilității maxime. În cazul modelelor de limbă bazate pe bigrame, probabilitățile care vor trebui calculate pentru fiecare pereche de cuvinte (c_i, c_j) sunt $p(c_j|c_i)$[3].

$$p(c_j|c_i) = \frac{număr\ apariții\ (c_i,c_j)}{\sum_c număr\ apariții\ (c_i,c)} \quad \textbf{(1.2)}$$

Formula 1.2 pune în evidență faptul că pentru estimarea probabilității $p(c_j|c_i)$ este necesar să se determine cât de des este urmat cuvântul c_i de cuvântul c_j și nu de alte cuvinte[3].

În cazul unui model bazat pe trigrame probabilitățile care vor trebui calculate pentru fiecare grupare de trei cuvinte (c_i, c_j, c_k) sunt $p(c_k|c_i, c_j)$[3].

$$p(c_k|c_i, c_j) = \frac{număr\ apariții\ (c_i,c_j,c_k)}{\sum_c număr\ apariții\ (c_i,c_j,c)} \quad \textbf{(1.3)}$$

Formula 1.3 pune în evidență faptul că pentru estimarea probabilității $p(c_k|c_i, c_j)$ este necesar să se determine cât de des este urmat cuvântul c_i de cuvintele c_j și c_k.

Pentru determinarea cu acuratețe a probabilităților prezentate este necesară o cantitate mare de date de antrenare (sute de milioane de cuvinte sau chiar miliarde). De asemenea, modelele de limbă bazate pe n-grame de ordin mai mare necesită și ele o cantitate mare de date de antrenare[3].

1.1.2 Principala problemă asociată construcției modelelor de limbă de tip n-gram și abordarea ei

Data sparseness reprezintă principala problemă întâlnită în cazul construcției modelelor de limbă bazate pe n-grame chiar și atunci când se folosesc pentru antrenare corpusuri de text de dimensiuni mari. Acest lucru se întâmplă deoarece există posibilitatea ca anumite n-grame să nu se regăsească în corpusul de antrenare, indiferent de dimensiunea acestuia, dar să se regăsească în corpusul de test, folosit la evaluare. Având în vedere principiul plauzibilității maxime, probabilitatea asociată n-gramelor în cauză va fi în mod evident 0 așa cum reiese și din formulele 1.2 sau 1.3. Problema este cu atât mai accentuată, cu cât ordinul n-gramelor este mai mare și are ca și consecință necesitatea ajustării probabilităților care au fost estimate pe baza numărului de apariții ale n-gramelor în corpusul de antrenare[3].

Metode de netezire

Metodele de netezire sunt folosite în cadrul procesului de ajustare și extrag o parte din probabilitatea asociată n-gramelor întâlnite pe parcursul procesului de antrenare, redistribuind-o n-gramelor care sunt întâlnite pentru prima dată în etapa de evaluare, în corpusul de test. Se vor defini trei metode de netezire ce diferă prin modul în care se face redistribuția probabilității[3].

Metoda de netezire *add-one smoothing* adaugă un număr fix, cum ar fi unu, la fiecare numărare a n-gramelor. Astfel, n-gramele care nu apar în corpusul de antrenare dar care sunt alcătuite din cuvinte ce se regăsesc în vocabular vor avea asociate probabilități nenule. Această metodă oferă, însă, o credibilitate nejustificată n-gramelor care nu apar în corpusul de antrenare. Pentru a se evita acest lucru, numărul fix ce va fi adaugat va fi unul mai mic ca unu, notat α, rezultând astfel o metodă de netezire numită *add-α smoothing*[3].

Metoda *Good-Turing* se bazează pe numărul real de apariții (k) și pe ținerea în evidență a numărului de apariții, acestea fiind folosite la ajustarea numărul de apariții (k*) pentru toate n-gramele care au mai fost întâlnite și care nu au mai fost întâlnite:

$$k^* = (k+1)\frac{N_{k+1}}{Nk} \quad \textbf{(1.4)}$$

N_k este numărul de n-grame care apar de k ori în corpusul de antrenare[3].

Metoda *Good-Turing* nu este de încredere pentru un k mare, caz în care N_k de regulă este 0. Rezolvarea acestui lucru constă în nerealizarea ajustării numărului de apariții pentru n-gramele frecvente[3].

Metodele de back-off

O altă abordare pentru a rezolva problema cunoscută sub numele de *data sparseness* presupune utilizarea mai multor modele de limbă pentru a crea un model de limbă interpolat care să poată beneficia de toate părțile constitutive. Dacă au fost construite modele de limbă bazate pe n-grame (p_n) de diferite ordine (de exemplu de ordin unu, doi sau trei) atunci un model de limbă interpolat (p_I) a putea fi construit ca o combinație liniară[3]:

$$p_I(c_3|c_1,c_2) = \lambda_1 p_1(c_3) \times \lambda_2 p_2(c_3|c_2) \times \lambda_3 p_3(c_3|c_1,c_2) \quad \textbf{(1.5)}$$

Coeficienții λ sunt subunitari, pozitivi iar suma lor este egala cu 1.

Metodele de *back-off* subliniază faptul că, pentru a determina probabiliatea unei n-grame care nu se regăsește în corpusul de antrenare, se poate lua în considerare si probabilitatea oferită de modelele de limbă de ordin inferior. În acest caz, problema de optimizare este reprezentată de alegerea echilibrului corect între modelele de ordin superior și cele de ordin inferior, în cazul în care acestea vor fi folosite. Mai multe metode de *back-off* au fost propuse începând cu *metoda de netezire Witten-Bell* care pune accentul pe diversitatea cuvintelor care urmează o istorie. Cea mai folosită metodă la ora actuală este *metoda de netezire Kneser-Key* care ia în calcul diversitatea istoriilor pentru o n-gramă particulară. Extensia acestei metode este *metoda modificată de netezire Kneser-Ney* ce folosește o metodă numită reducere absolută pentru a micșora probabilitatea cumulată a evenimentelor întâlnite[3].

1.1.3 Evaluarea modelelor de limbă

Având în vedere toate cele afirmate până în acest moment, se poate deduce faptul că rolul unui model de limbă este acela de a prezice următorul cuvânt ținând cont de predecesorii săi. Această capacitate de prezicere este cea care trebuie luată în considerare în vederea comparării modelelor de limbă dar și a îmbunătățirii lor[3].

Perplexitatea

Evaluarea puterii de predicție a unui model de limbă se face măsurând probabilitatea pe care acesta o asociază secvențelor de test. Un model de limbă bun va asocia o probabilitate mare unui text corect și o probabilitate mică unui text incorect. În acest caz, cea mai întâlnită metrică de evaluare este *perplexitatea*[3].

Perplexitatea derivă din entropia încrucișată, ce poate fi calculată pe baza unui model de limbă particular dar și a unei secvențe particulare de cuvinte C=c_1, c_2, c_3, ..., c_n după cum urmează[3]:

$$H(p_{LM}) = -\frac{1}{n}\log p_{LM}(c_1,c_2,\dots,c_n) = -\frac{1}{n}\sum_{i=1}^{n}\log p_{LM}(c_i|c_1,c_2,\dots,c_{i-1}) \quad \textbf{(1.6)}$$

Formula ce pune în evidență modul în care perplexitatea derivă din entropia încrucișată este[3]:

$$PPL(p_{LM}) = 2^{H(p_{LM})} \quad \textbf{(1.7)}$$

O perplexitate mai mare asociată unei anumite secvențe de cuvinte înseamnă o capacitate mai mică de predicție a secvenței pentru respectivul model de limbă. Perplexitatea poate fi

calculată atât pentru un text de evaluare (de testare) cât și pentru un text de antrenare, având semnificații diferite în aceste două cazuri. Perplexitatea asociată textului de testare evaluează capacitatea de generalizare și predicție a modelului de limbă iar perplexitatea asociată textului de antrenare măsoară gradul de potrivire al modelului de limbă cu datele de antrenare[3].

Cuvintele din afara vocabularului

Metodele de netezire prezentate anterior iau în calcul n-gramele care nu se regăsesc în corpusul de antrenare dar care sunt alcătuite din cuvinte ce se regăsesc în acesta. Aceste metode nu pot fi folosite pentru a ajusta modelul de limbă astfel încât să asocieze o probabilitate nenulă unui cuvânt care nu face parte din vocabularul inițial. Un astfel de cuvânt, de fapt, astfel de cuvinte sunt numite *cuvinte din afara vocabularului* (termenul din limba engleză fiind *out of vocabulary*, de unde și acronimul *OOV*). Ele nu pot fi prezise de modelul de limbă și prin urmare îngreunează procesul de evaluare a modelului de limbă. Perplexitatea lor este infinită, motiv pentru care nu poate fi adunată la perplexitatea celorlalte n-grame pentru obținerea perplexității asociate secvenței de cuvinte. Prin urmare și procentul de cuvinte din afara vocabularului trebuie specificat și luat în considerare, pe lângă perplexiatete, în cazul procesului de comparare a două modele de limbă[3].

$$OOV[\%] = \frac{Număr\ OOVuri}{Număr\ total\ cuvinte} \times 100 \quad \textbf{(1.8)}$$

Aparițiile n-gramelor

Și aparițiile n-gramelor pot fi folosite pentru evaluarea capacităților de predicție a unui model de limbă bazat pe n-grame. Anterior s-a arătat faptul că metodele de *back-off* folosesc modele de limbă bazate pe n-grame pentru abordarea problemei numite *data sparseness*. Astfel, un model de limbă bazat pe trigrame încearcă să estimeze cuvântul următor pe baza unei istorii alcătuite din două cuvinte precedente (caz în care avem de-a face cu un model de trigrame), dar se poate întoarce, respectiv poate folosi o istorie alcătuită dintr-un singur cuvânt precedent (caz în care avem de-a face cu un model de bigrame) sau poate să nu se bazeze pe o istorie (caz în care avem de-a face cu model de unigrame). În ceea ce privește modelul de trigrame, procentul de apariție al trigramelor reprezintă o măsură a numărului de cazuri în care modelul de limbă poate folosi istoria alcătuită din două cuvinte precedente în comparație cu numărul de cazuri în care modelul de limbă necesită folosirea unor istorii alcătuite dintr-un număr mai mic de cuvinte sau chiar nule pentru a determina probabilitatea trigramei curente[3]:

$$apariții\ trigrame[\%] = \frac{\#apariții\ trigrame}{\#cuvinte} \times 100 \quad \textbf{(1.9)}$$

Aparițiile n-gramelor reprezintă o metrică de evaluare auxiliară, însă foarte utilă pentru compararea modelelor de limbă specifice unor anumite domenii. Un procent mai mare al aparițiilor presupune un model de limbă mai adaptat la domeniu[3].

1.2 DESCRIEREA METODEI FOLOSITE PENTRU RESTAURAREA DIACRITICELOR

Procesul de restaurare a diacriticelor poate fi privit ca unul de dezambiguizare[4].

Un proces de dezambiguizare are drept scop transformarea unui flux de cuvinte asociate vocabularului V_1 într-un flux corespondent de cuvinte, asociate vocabularului V_2 ținând cont de o hartă probabilistică de tipul una-la-mai-multe. Aceasta din urmă cuprinde numeroasele posibilități de transformare a unui cuvânt din vocabularul V_1 într-unul din vocabularul V_2 precum și probabilitățile asociate fiecărei posibilități[4].

Prin urmare, procesul de restaurare de diacritice transformă o secvență de cuvinte ambigue (în acest caz, cuvinte ce conțin parțial diacritice sau care nu conțin deloc diacritice) într-o

secvență corespondentă de cuvinte ce nu prezintă ambiguități și care prin urmare sunt corecte (în cazul nostru, cuvinte cu diacritice)[4].

Metoda de restaurare folosită pentru realizarea părții practice a lucrării de față folosește o colecție de librării în C++, numită SRI-LM, pentru a realiza transfomarea unui text fără diacritice, sau care le conține parțial, într-unul cu diacritice. Acest lucru nu poate fi, însă, realizat fără folosirea unui model de limbă construit tot cu ajutorul aceleiași colecții de librării și a unei hărți probabilistice create cu ajutorul unei aplicații Java[3].

Pentru un corpus de text în care diacriticele lipsesc parțial sau complet, estimarea formei cu diacritice se face pentru fiecare cuvânt în parte. Dacă forma fără diacritice a cuvântului nu este găsită în harta probabilistică atunci acesta nu va fi modificat[3]. Altfel, se estimează forma cu diacritice d* pentru cuvântul ambiguu din punct de vedere al prezenței diacriticelor d', având în vedere secvența precedentă de N cuvinte cu diacritice, D, prin găsirea formei cu diacritice d_i care maximizează următoarea formulă[4]:

$$d^* = \arg\max_{d_i} p(d_i|D) \times p(d_i|d') \quad \textbf{(1.10)}$$

Prima probabilitate este dată de modelul de limbă bazat pe n-grame iar cea de-a doua de harta probabilistică. Modelul de limbă poate fi construit folosind un corpus de text ce conține cuvinte corecte din punct de vedere al prezenței diacriticelor. Același corpus poate fi folosit pentru a estima probabilitățile ce se regăsesc în harta probabilistică cu ajutorul formulei[4]:

$$p(d_i|d') = \frac{număr\ apariții\ d_i}{\sum_j număr\ apariții\ d_j} \quad \textbf{(1.11)}$$

unde d_j reprezintă toate formele cu diacritice ale lui d'.

În tabelul următor se va prezenta un extras dintr-o hartă probabilistică de tipul una-la-mai-multe.

...
colectiva: colectiva 0.008 colectivă 0.992
dacia: dacia 1.0
ingrosam: îngroșam 0.2 îngroșăm 0.8
pana: pana 0.005 pană 0.008 până 0.987
sarmana: sărmana 0.847 sărmană 0.153
...

Tabel 1.1 Fragment dintr-o hartă probabilistică, Sursa [3]

Având în vedere cele menționate asupra procesului de restaurare a diacriticelor și a modului de construire a modelului de limbă și a hărții probabilistice, se poate considera valabilă ca și arhitectură a sistemului de restaurare folosit arhitectura prezentată în Figura 1.1.

1.3 ALTE METODE DE RESTAURARE A DIACRITICELOR

De-a lungul timpului au fost dezvoltate mai multe metode de restaurare a diacriticelor pentru limba română. Restaurarea, în cazul unora, se face doar pe baza contextului caracterelor, în timp ce în cazul altora, rezultate mai bune se obțin dacă se ia în calcul contextul secvenței de cuvinte. De asemenea, metodele se diferențiază și prin cantitatea necesară de resurse de antrenare, care de cele mai multe ori determină costul dezvoltării sistemului de restaurare[3].

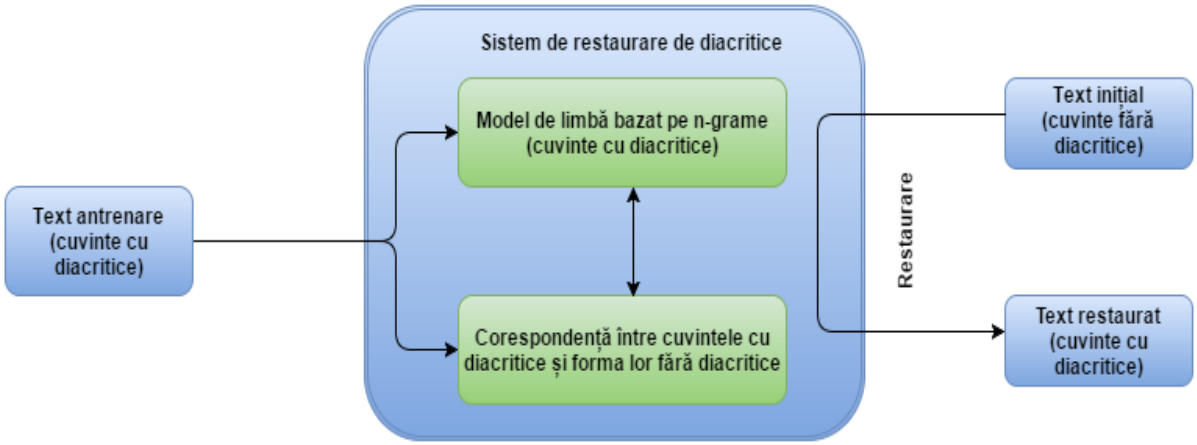

Figura 1.1 Arhitectura sistemului de restaurare de diacritice, Sursa [3]

Rada Mihalcea și Vivi Năstase propun în lucrarea *„Letter Level Learning for Language Independent Diacritics Restoration”* o metodă de restaurare a diacriticelor bazată pe învățarea la nivel de caracter și nu la nivel de cuvânt. Aceasta propune ca accentul să nu mai fie pus pe învățarea generalizărilor care se aplică la nivel de cuvânt, care s-ar traduce în reguli precum „*peste* ar trebui să se transforme în *pește* atunci când este urmat de un substantiv”, ci pe găsirea unor generalizări la nivel de literă care s-ar traduce în reguli de forma „*s* urmat de *i* și precedat de un spațiu alb ar trebui să se transforme în *ș*”. Astfel de reguli sunt mai generale și au o aplicabilitate mai mare atunci când dicționarele disponibile sunt de mici dimensiuni, sunt întâlnite numeroase cuvinte necunoscute în textul de intrare sau când nu sunt disponibile instrumente pentru analiza morfologică sau sintactică[5].

În ceea ce privește analiza limbajului, literele constituie cel mai mic nivel posibil de granularitate și prin urmare au cel mai mare potențial de generalizare. Prin urmare, în loc să folosească un model de limbă bazat pe n-grame, metoda folosește un model de caracter bazat pe n-grame[3]. Ea a fost evaluată pentru patru limbi diferite: ceha, ungara, poloneza și româna, oferind o precizie de peste 98% la nivel de caracter. De asemenea, nu necesită niciun alt instrument sau nicio altă resursă, în afară de un corpus de text cu diacritice. Având în vedere simplitatea algoritmului, viteza de procesare este și ea mare[5].

Următoarea metodă de restaurare a diacriticelor a fost prezentată de Dan Tufiș și Adrian Chițu în lucrarea *„Automatic Diacritics Insertion in Romanian Texts”*, fiind îmbunătățită mai apoi de către Dan Tufiș și Alexandru Ceaușu în lucrarea *„DIAC^{+}: A Professional Diacritics Recovering System”*. Ea utilizează o tehnică de etichetare a părților de vorbire (POS) pentru dezambiguizarea diferitelor ipoteze ale cuvintelor cu diacritice și necesită mai multe resurse de tip NLP față de metoda prezentată anterior, însă oferă rezultate mai bune. Metoda a fost integrată într-un sistem de restaurare a diacriticelor de sine stătător – Diac^{+} care este disponibil prin Internet fiind destinat completării funcționalităților oferite de MS-Office[3].

Îmbunătățirile prezentate de Dan Tufiș și Alexandru Ceaușu în lucrarea *„DIAC^{+}: A Professional Diacritics Recovering System”* presupun corectarea ortografiei dar și adăugarea unei metode de procesare a cuvintelor necunoscute. De asemenea, modelul de limbă este și el îmbunătățit, fiind folosit un corpus de antrenare de trei ori mai mare față de varianta precedentă, iar etichetarea beneficiază de mai multă acuratețe. Varianta inițială utiliza un model de limbă combinat (impunându-se astfel etichetarea pentru fiecare model de limbă disponibil, pentru ca, mai apoi, rezultatele să fie combinate). Cea actualizată este mai rapidă deoarece se bazează pe un singur pas de etichetare, evitându-se astfel pierderile de timp datorate combinării rezultatelor etichetării pentru fiecare model de limbă disponibil[6]. Detaliile asupra modului în care se realizează etichetarea sunt prezentate în același lucrare

menționată anterior, lucrarea de față punând accentul pe metoda de restaurare folosită și nu pe celelalte.

Ultima metodă a fost prezentată de Cătălin Ungureanu, Dragoș Burileanu, Vladimir Popescu, Cristian Negrescu și Aurelian Derviș în lucrarea *„Automatic diacritics restoration for a TTS-based e-mail reader application”*. Ea necesită un corpus de dimensiuni medii pentru antrenarea diverselor modele de limbă și pentru a crea harta cuvintelor ce face legătura între forma fără diacritice a cuvintelor și toate formele cu diacritice și oferă rezultate comparabile cu cele ale metodei anterioare.

1.4 PERFORMANȚELE METODEI FOLOSITE PENTRU RESTAURAREA DIACRITICELOR

Pentru evaluarea performanțelor metodei folosite au fost luate în considerare două corpusuri de text: misc2 (1.2M cuvinte) și 9am30art(13k cuvinte). Cel din urmă conține primele 30 de articole ale corpusul 9am.

Corpusul 9am a fost construit în urma colectării tuturor știrilor publicate de site-ul de știri 9am în perioada Noiembrie 2004 – Martie 2011. A rezultat astfel cel mai mare corpus de antrenare disponibil pentru limba română, acesta conținând 3.5 milioane de fraze respectiv 63 de milioane cu cuvinte[3].

Printre metricile folosite în cadrul evaluării sistemului se numără: rata de eroare la nivel de cuvânt (WER), rata de eroare la nivel de caracter (ChER) şi măsura-F. Pentru a determina rata de eroare la nivel de cuvânt se vor lua în considerare erorile de inserție, de substituție și de ștergere. În mod similar, pentru a determina ChER se vor lua în calcul aceleași erori apărute de acestă dată la nivel de caracter și nu la nivel de cuvânt ca în cazul WER. În cazul ideal, WER și CheER sunt 0[3]. Prin urmare, o metodă de restaurare a diacriticelor este cu atât mai bună cu cât aceste două metrici sunt mai apropiate de 0.

$$WER[100] = \frac{Nr.erori\ inserție + Nr.erori\ de\ substituție + Nr\ erori\ de\ ștergere}{Nr.cuvinte\ text\ de\ referință} \times 100 \qquad \textbf{(1.12)}$$

Mărimea-F se definește ca media armonică a preciziei și gradul de comparație. Precizia este dată de raportul dintre numărul de diacritice corect inserate și numărul de diacritice din textul ipotetic (textul rezultat). Gradul de comparație este dat de raportul dintre numărul de diacritice inserate corect și numărul de diacritice din textul de referință. Pentru aceste metrici valoare ideală este 100%.

Analizând tabelul etichetat Tabel 1.2 se observă că se obțin performanțe mai bune pentru caracterele fără diacritice față de varianta lor cu diacritice. De asemenea, se poate afirma faptul că, în ceea ce privește clasa de ambiguitate i/î aceasta este rezolvată aproape perfect. Nu același lucru poate fi afirmat despre clasa de ambiguitate a/ă/â. Ambiguitatea a/ă se numără printre cele mai dificile probleme ale limbii române deoarece toate substantivele și adjectivele ce prezintă o formă accentuată ce se termină cu litera a, prezintă o formă neaccentuată ce se termină cu litera cu ă.

Pentru o evaluare completă a sistemului de restaurare folosit nu este suficientă menționarea rezultatelor oferite de acesta în urma realizării unui anumit șir de experimente, fiind nesară compararea acestora cu cele oferite de alte metode de restaurare. În tabelul etichetat Tabel 1.3 se va prezenta o astfel de comparație. Se va lua în considerare următoarea notație: metoda 3 - metoda prezentată de Cătălin Ungureanu, Dragoș Burileanu, Vladimir Popescu, Cristian Negrescu și Aurelian Derviș în lucrarea *„Automatic diacritics restoration for a TTS-based e-mail reader application”*.

Clasa de ambiguitate	Litera	Corpus evaluare: misc2			Corpus evaluare: 9am30art		
		Precizia [%]	*Grad comparație [%]*	*Mărimea -F [%]*	*Precizia [%]*	*Grad comparație [%]*	*Mărimea -F [%]*
a/ă/â	a	98.28	97.71	97.99	99.32	98.20	98.75
	ă	94.42	96.13	95.27	93.73	97.54	95.60
	â	98.79	97.55	98.16	97.00	99.08	98.03
i/î	i	99.97	99.88	99.92	100	99.96	99.98
	î	99.26	99.65	99.45	100	100	100
s/ș	s	99.75	99.62	99.69	99.21	99.89	99.55
	ș	98.71	99.14	98.92	99.66	97.67	98.65
t/ț	t	99.52	99.62	99.57	99.78	99.96	99.87
	ț	97.74	97.21	97.47	99.70	98.51	99.10

Tabel 1.2 Performanțele la nivel de caracter ale metodei de restaurare folosite, Sursa [3]

Experiment	Metoda de restaurare a diacriticelor	Corpus evaluare: misc2			Corpus evaluare: 9am30art		
		WER [%]	ChER [%]	Mărimea-F [%]	WER [%]	ChER [%]	Mărimea-F [%]
1	Metoda folosită	1.99	0.48	98.73	1.50	0.31	99.18
2	Diac^{+*}	n/a	n/a	n/a	1.61	0.34	99.11
3	Diac^{+**}	n/a	n/a	n/a	2.28	0.51	98.67
4	Metoda 3	2.13	0.25	99.34	2.37	0.50	98.95

Tabel 1.3 Comparația rezultatelor metodei folosite cu cele ale celorlalte metode prezentate, Sursa [3]

Evaluarea sistemului Diac^{+} a impus anumite dificultăți prin faptul că deși acesta realizează restaurarea pentru marea majoritate a cuvintelor ambigue (cuvinte care pot fi scrise după mai multe tipare ale diacriticelor: pana/pană/până) și a cuvintelor care nu sunt ambigue (cuvinte care fie nu au diacritice, fie au un tipar unic al diacriticelor: alb, pădure) el îi cere utilizatorului să aleagă forma corectă doar pentru cuvintele cu diacritice de tip POS. Astfel s-a considerat următoarea abordare[3]:

- Pentru al doilea experiment: a fost selectată manual prima alternativă oferită de sistem;

- Pentru al treilea experiment: a fost lăsată forma fără diacritice, ca și cum sistemul nu ar fi știut ce abordare să folosească.

Se observă că nu au putut fi oferite rezultate pentru al doilea și al treilea experiment atunci când evaluarea s-a făcut folosind corpusul mai mare: misc2. Acest lucru se datorează faptului că funcționalitatea Diac^{+} disponibilă în MS-Office nu a putut restaura diacriticele pentru cele

1.2 milioane de cuvinte. Prin urmare s-a impus necesitatea evaluării metodelor folosind un corpus mai mic: 9am30art[3].

Analizând rezultatele prezentate în tabelul etichetat Tabel 1.3 se poate observa că pentru corpusul misc2 (corpusul mai mare), din punct de vedere al ratei de eroare la nivel de caracter, rezultatele cele mai bune sunt oferite de metoda 3. În schimb, din punct de vedere al ratei de eroare la nivel de cuvânt, metoda folosită pentru realizarea părții practice a lucrării de față este mai performantă. Prin urmare, se poate deduce faptul că o metodă reușește să restaureze corect anumite diacritice ce nu sunt restaurate de către cealaltă[3].

În ceea ce privește sistemul Diac^{+}, se obțin rezultate mai proaste în comparație cu cele oferite de celelalte metode dacă este menținută forma fără diacritice. În schimb, dacă se alege manual prima formă cu diacritice ce se numără printre alternativele oferite de sistem, se obțin rezultate mai bune față de cele oferite de metoda 3, însă mai slabe față de cele ale metodei folosite în lucrarea de față[3].

Ca o ultimă observație, corpusul de evaluare este și el important având în vedere că în urma primului experiment și al utimului experiment se obțin rezultate diferite pentru fiecare corpus. Prima metodă oferă rezultate mai bune pentru 9am30art, în timp ce ultima metodă oferă rezultate mai bune pentru misc2. Având în vedere că misc2 este mult mai mare, se poate afirma faptul că rezultatele obținute în cazul său sunt mai convingătoare. Totuși rezultatele diferite obținute pentru 9am30art care este și el suficient de mare pun în evidență faptul că nu se pot compara sisteme de restaurare diferite decât în cazul folosirii aceluiași cadru de experimentare pentru ambele sisteme[3].

CAPITOLUL 2

TEHNOLOGII SOFTWARE UTILIZATE ÎN CADRUL REALIZĂRII SERVICIULUI WEB

2.1 TEHNOLOGII DE PROGRAMARE

2.1.1 *Java*

Java este un limbaj de programare de nivel înalt, orientat pe obiecte, ce permite execuția mai multor instrucțiuni în același timp (instucțiunile nu sunt executate secvențial) şi care a fost dezvoltat pe baza principiului WORA, făcându-se astfel posiblă rularea codului scris folosind acest limbaj de programare pe toate platformele care suportă Java fără a fi necesară recompilarea[7].

Fişierele ce conțin codul sursă scris în limbaj Java, cele cu extensia .java, sunt transformate în fișiere cu extensia .class, ce conțin codul sursă reprezentat într-un format numit *bytecode*. Această transformare se face cu ajutorul compilatorului, fişierele .class rezultate fiind cele care sunt executate de interpretorul Java[8].

Portabilitatea acestui limbaj, mai precis, independența față de platforma de lucru se datorează faptului că, la ora actuală, multe sisteme de operare au asociate interpretoare Java şi medii de execuție specifice, numite *Java Virtual Machine*(JVM)[8].

Programarea orientată pe obiecte a apărut din dorința de eliminare a dezavantajelor programării clasice, sturcturate, printre care se numără următoarele: reutilizarea codurilor este dificilă, programele de dimensiuni mari sunt greu depanat[9]. Ea se bazează pe conceptul de obiect care este definit printr-o mulțime de atribute, ce nu reprezintă altceva decât proprietăți ale obiectului (de exemplu, un atribut al obiectului cățel ar fi rasă și coloare blană), și printr-o mulțime de metode, ce descriu modul în care se comportă obiectul (în cazul obiectului cățel,

numele uneia dintre metode ar fi latră). Diferența dintre programarea structurată și cea orientată pe obiecte este reprezentată de faptul că programarea clasică are drept obiectiv prelucrarea datelor, punând accentul pe programe și funcții, în timp ce programarea orientată pe obiecte pune accentul pe definirea obiectelor, cu alte cuvinte, pune accentul pe atribute și metode[9].

Java prezintă numeroase domenii de aplicabilitate, fiind limbajul de programare folosit pentru realizarea aplicațiilor Android; a aplicațiilor web; a unor instrumente software și de dezvoltare precum Eclipse, Netbeans IDE dar și pentru realizarea unor aplicații științifice precum cele de procesare a limbajului natural.

2.1.2 JavaScript

JavaScript este un limbaj de programare de nivel înalt, orientat pe obiecte, dinamic, diferit de Java atât din punct de vedere al conceptului cât și al modului de proiectare, fiind utilizat pentru dezvoltarea aplicațiilor web. Este cunoscut, în special, ca un limbaj de *scripting* asociat paginilor web, fiind folosit pentru a controla comportamentul acestora.

Există trei forme ale limbajului JavaScript:

- CSJS (Client-Side JavaScript) – reprezintă o versiune extinsă a JavaScript folosită pentru îmbunătățirea funcționalităților paginilor web și a navigatoarelor web folosite de client;
- SSJS (Server-Side JavaScript) – reprezintă o versiune extinsă a JavaScript folosită pentru accesarea în fundal a bazelor de date, a sistemelor de fișiere și a server-elor;
- Core JavaScript – reprezintă limbajul JavaScript de bază, CSJS și SSJS fiind dependente de acesta.

Cea mai utilizată versiune este CSJS, codul rezultat fiind inclus direct în documentul HTML sau introdus într-un fișier de tip .js care va fi referit în documentul HTML. Se creează astfel o pagină HTML dinamică și interactivă, ce poate prezenta următoarele funcționalități: verificarea faptului că adresa de e-mail introdusă în câmpul corespunzător al unui formular este una corespunzătoare (nu se verifică existența ei, ci faptul că respectă șablonul clasic al unei adrese de e-mail), verificarea faptului că un câmp al formularului a fost completat, etc. Codul JavaScript este executat abia după finalizarea completării formularului și apăsării butonului de trimitere.

Principalul avantaj al acestui limbaj de programare este reprezentat de faptul că, fiind un limbaj interpretat de către navigatoarele web, nu este necesar un compilator[10].

2.1.2.1 jQuery

jQuery este o bibliotecă JavaScript mică și rapidă dar care pune la dispoziția programatorului numeroase funcții pe care acesta le poate utiliza pentru simplificarea codului scris în limbaj JavaScript. Cu alte cuvinte, un simplu apel al unei funcții asociate acestei biblioteci poate conduce la obținerea unei funcționalități care altfel ar fi necesitat scrierea mai multor linii de cod. Acest lucru este susținut și de motto-ul pe baza căruia a fost dezvoltată biblioteca și anume *„Write less, do more!”*.

Printre principalele caracteristici ale bibliotecii jQuery se numără[11]:

- facilitează traversarea elementelor documentului HTML precum și manipularea lor;
- oferă o metodă elegantă de a trata anumite evenimente precum apăsarea unui link;
- prezintă numeroase funcții prin intermediul cărora pot fi realizate și incluse animații în pagina web, cu un minim de efort;
- simplifică apelurile de tip AJAX.

Pentru ca funcțiile specifice acestei biblioteci să poată fi folosite este necesară fie descărcarea ei de pe site-ul web asociat, jQuery.com, pe propriul computer și apoi includerea în fișierul HTML, fie includerea directă în fișierul HTML însă ținând cont de faptul că biblioteca se află pe server-ele unei rețele de tip CDN. CDN este o rețea distribuită de servere care livrează pagini web și alte tipuri de astfel de conținut (conținut web) utilizatorului.

Dintre cele două variante, de preferat este cea de-a doua deoarece, prin vizitarea altor pagini web, cei mai mulți utilizatori au descărcat deja biblioteca jQuery. Astfel, aceasta va fi încărcată direct din cache atunci când sunt vizitate alte pagini ce folosesc funcții jQuery și prin urmare timpul de încărcare al paginii web vizitate va fi mai mic. De asemenea, cea de-a doua variantă va fi preferată deoarece, în cazul unei rețele de tip CDN, resursa cerută de utlilizator va fi oferită de server-ul cel mai apropiat de acesta, din punct de vedere geografic, ceea ce duce la micșorarea timpului de încărcare al paginii accesate[12].

Sintaxa utilizată atunci când se dorește apelarea unei funcții din biblioteca jQuery indică faptul că principalul scop al acesteia este acela de a selecta elementul paginii HTML asupra căruia va acționa una din funcțiile sale[13].

Sintaxa este[13]: *$(selector).action()*. Semnul *„$”* specifică faptul că se va folosi o funcție specifică jQuery, *selector* este folosit pentru identificarea elementului HTML (de exemplu, acest câmp poate fi reprezenat de id-ul elementului) iar *action()* nu reprezintă altceva decât funcția apelată. Astfel, dacă se dorește ascunderea tuturor paragafelor dintr-o pagină HTML, linia de cod ce va trebui folosită este: $(“p”).hide().

2.1.2.2 AJAX

AJAX este o tehnică folosită pentru crearea de pagini web dinamice și rapide. Aceasta permite actualizarea asincronă a paginilor web ce se realizează prin schimbul unui număr mic de pachete de date cu server-ul. Cu alte cuvinte, paginile web pot fi actualizate fără a fi necesară reîncărcarea acestora. Nu același lucru se întâmplă și în cazul paginilor care nu folosesc AJAX, ele fiind reîncărcate în browser de fiecare dată când o porțiune a paginii este actualizată.[14].

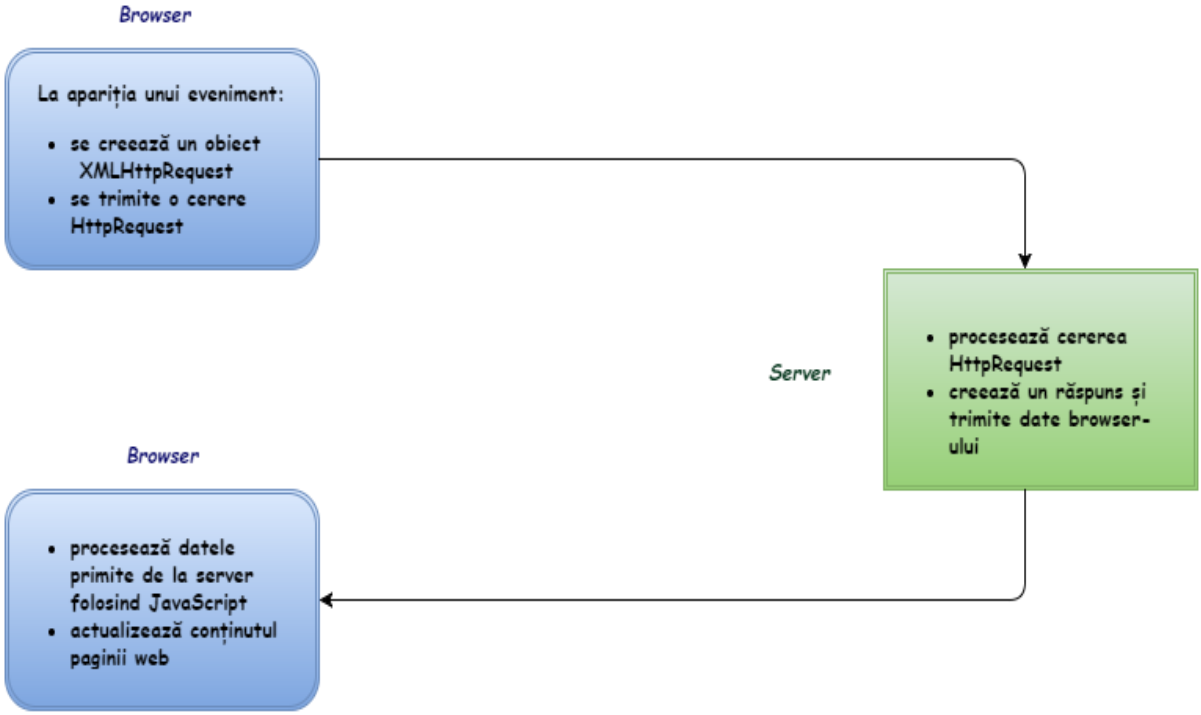

Figura 2.1 AJAX – Modul de funcționare, Sursa [14]

Așa cum se poate observa în Figura 2.1, în care este prezentat modul de funcționare al tehnicii AJAX, la apariția unui eveniment, precum apăsarea unui buton, se creează un obiect XMLHttpRequest, folosit pentru a schimba date cu server-ul în fundal. Astfel, părți ale paginii web pot fi actualizate fără a fi necesară reîncărcarea întregii pagini[14]. După crearea obiectului XMLHttpRequest, se va trimite o cerere HttpRequest către server folosind două metode ale obiectului XMLHttpRequest: open(), care inițializează cererea, și send(), care trimite efectiv cererea. Cererea trimisă este asincronă, prin urmare, în timp ce se așteaptă răspunsul din partea server-ului se pot executa alte scripturi, urmând ca răspunsul să poată fi procesat în momentul în care acesta este primit. În momentul în care răspunsul este primit, acesta este procesat iar pe baza informațiilor primite se actualizează pagina web (porțiunea din pagina web).

2.1.3 HTML

HTML este un limbaj de marcare folosit pentru crearea paginilor web, ce definește conținutul paginilor și modul de organizare și afișare al componentelor acestora prin utilizarea etichetelor și a atributelor[15]. Etichetele sunt reprezentate de cuvinte cheie plasate între caracterele „<”, „>” și sunt folosite pentru a defini elemente HTML. Marea majoritate a etichetelor formează perechi, ceea ce însemnă că folosirea uneia dintre etichete presupune, implicit, și folosirea celeilalte. O astfel de pereche este alcătuită dintr-o etichetă de început și una de încheiere. Ambele coțin același cuvânt cheie plasat între caracterele „<” și „ >”, diferența dintre ele fiiind dată de faptul că, în cazul etichetei de încheiere, cuvântul cheie este precedat de caracterul „/”. Un exemplu de astfel de pereche ar fi și , folosită pentru definirea unui paragrapf. Există însă și etichete ce nu prezintă o pereche. De exemplu, pentru definirea unei imagini este suficientă folosirea etichetei <img>[16].

Pentru ca o pagină web să poată fi interpretată corect de către browser, și ,prin urmare, să poată fi afișată, trebuie să prezinte etichetele esențiale:<html> și perechea </html>, <head> și corespondentul </head>, <body> și eticheta de încheiere </body>. Prima pereche este folosită pentru a i se specifica browser-ului faptul că are de interpretat o pagină HTML. A treia pereche marchează secțiunea numită *head* ce conține informații asupra paginii precum titlul acesteia, referințele către fișerele JavaScript, etc iar utlima pereche marchează secțiunea numită *body* care conține toate componentele paginii vizibile în browser[17].

Având în vedere cele menționate anterior, codul utlizat pentru realizarea unei pagini web cu titlul „Acesta este un test”, ce afișează textul „Acesta este un paragraf” este:

```
<html>
    <head>
        <title>Acesta este un test</title>
    </head>
    <body>
        <p>Acesta este un paragraf</p>
    </body>
</html>
```

În ceea ce privește atributele, ce sunt specificate în eticheta de început, acestea sunt folosite pentru a oferi informații suplimentare asupra uneia dintre componentele paginii web (un paragraf, o imagine, etc). În general, atributele au asociat un nume și o valoare. Astfel, pentru alinierea centrală a unui paragraf se va folosi atributul cu numele *align* și valoarea *center* după cum urmează:

* Acesta este un paragraf *

2.1.4 *CSS*

Primele variante ale limbajului HTML nu prezentau etichete pentru formatarea conținutului paginii web, scopul limbajului fiind doar acela de a descrie conținutului paginii sub forma: <h1>Aceasta este o rubrică<h1>, Acesta este un paragraf . Odată cu apariția versiunii HTML 3.2, în care au fost introduse etichete precum <font> dar și atributele pentru specificarea culorii, procesul de dezvoltare a site-urilor web cu multe pagini a devenit unul greoi doarece era necesară introducerea informațiilor referitoare la font și la culoare pentru fiecare pagină în parte. Au apărut astfel cazurile defavorabile în care aceleași informații erau introduse separat în mai multe fișiere cu extensia html asociate site-ului web. Pentru ca timpul pierdut în realizarea acestei operații să poată fi eliminat W3C a creat CSS, Cascading Style Sheets[18].

CSS este un limbaj de stilizare folosit pentru a descrie modul de prezentare a unui document scris în limbajul HTML. Cu alte cuvinte, CSS descrie cum vor fi afișate elementele unei pagini web. La ora actuală, controlul aspectului unei pagini se face numai cu ajutorul CSS, ceea ce înseamnă că de la poziționarea unei componente pe pagină și până la culoarea textului și a fundalului, totul este implementat folosind acest limbaj.

CSS elimină dezavantajul menționat anterior și simplifică procesul de dezvoltare al unui site web cu mai multe pagini prin faptul că toate formele de stilizare ce vor fi aplicate elementelor HTML vor fi menționate într-un fișier extern cu extensia .css. Astfel, pentru schimbarea aspectului mai multor pagini web vor fi necesare modificări doar în fișierul .css.

Pentru ca diferitele forme de stilizare precizate în fișierele .css să poată fi încorporate într-o pagină web este necesară includerea acestora în pagină, mai exact, în fișierul .html, astfel:

<head>

<link rel="stylesheet" type="text/css" href="Style.css">

</head>

A doua linie de cod este folosită pentru a crea o legătură (un link) către fișierul .css. Atributul *rel* specifică relația dintre documentul curent (în cazul nostru pagina web) și documentul către care s-a creat legătura (fișier .css) și ia valoarea *stylesheet*, așa cum era de așteptat iar atributul *type* specifică tipul media al fișierului extern.

2.1.5 Bootstrap

Bootstrap este cel mai popular framework HTML, CSS și JavaScript folosit pentru dezvoltarea paginilor web a căror interfață se adaptează la rezoluția ecranului dispozitivului (desktop, laptop, tabletă, smartphone) de pe care este accesată pagina[19].

Bootstrap oferă numeroase facilități, punând la dispoziția dezvoltatorilor șabloane HTML și CSS pentru formulare, butoane, tabele, diferite animații, etc. Astfel, dezvoltatorul este scutit de o parte din muncă, rămânându-i sarcina de a particulariza șablonul prespectiv, de exemplu, prin utilizarea propriului fișier .css.

Pentru ca facilitățile oferite de Bootstrap să poată fi folosite este necesară includerea fișierelor .css și .js specifice Boostrap în fișierul .html. Cea mai avantajoasă variantă, așa cum s-a arătat în secțiunea 2.1.2.1 este cea a includerii directe ținând cont de faptul că fișierele se află pe server-ele din rețeaua Bootstrap CDN.

2.2 PROTOCOALE

2.2.1 HTTP

HTTP este un protocol de nivel aplicație destinat sistemelor distribuite între care se stabilește o relație de colaborare[20]. Practic, HTTP este un protocol de comunicație folosit pentru transferul datelor (text, imagini, fișiere audio, fișiere video, etc) între două sau mai multe aplicații distribuite. Portul folosit pentru comunicare este de regulă portul TCP 80, însă pot fi folosite și alte porturi. Mai mult, HTTP este un protocol de tip cerere-răspuns ce reprezintă baza comuncării în cazul unui sistem de tipul client-server. În ceea ce privește clientul, acesta poate fi reprezentat, de exemplu de browser-ul web, iar server-ul poate fi reprezentat de o aplicație ce rulează pe un alt calculator[21]. Clientul trimite o cerere HTTP către server iar acesta va răspunde, răspunsul HTTP putând conține datele cerute de client. Varianta 1.0 a protocolului realiza o nouă conexiune de fiecare dată când o cerere HTTP era emisă. Acest lucru a fost însă rectificat, astfel varianta 1.1 permite folosirea aceleiași conexiuni pentru realizarea mai multor schimburi de tip cerere-răspuns[21].

O cerere HTTP simplă poate arăta în felul următor[22]:

GET /files/colors.txt HTTP/1.1

Host: www.colors.com

User-Agent: Mozilla/4.0

Analizând prima linie a cererii, se observă că metoda folosită de către aceasta este GET, ceea ce înseamnă că se dorește obținerea resursei identificată prin URI-ul */files/colors.txt*. De asemenea, se mai observă că versiunea protocolului HTTP folosită este 1.1.

Următoarele două linii nu sunt altceva decât antete. Antetul *Host* identifică server-ul pe care se află resursa iar *User-Agent* specifică bowser-ul utilizat pentru realizarea cererii. Pintre celelelalte antete care mai pot fi folosite se numără , de exemplu, cel care oferă informații referitoare la tipurile de date care pot fi interpretate corect de client[22].

Având în vedere cererea prezentată anterior, răspunsul server-ului poate fi următorul[22]:

HTTP/1.1 200 OK

Lat-Modified: Mon, 23 Jul 2015, 08:41:57 GMT

Content-Length: 24

Content-Type: text/plain

red, orange, blue, green

Prima linie indică versiunea protocolului HTTP folosită și status-ul cererii HTTP. Fiind prezent codul 200, se poate trage concluzia că cererea a fost acceptată, fișierul cerut urmând a fi transmis. Urmează apoi trei antete care specifică data și ora la care au fost făcute ultimele modificări asupra fișierului, lungimea (exprimată în octeți) și tipul acestuia. Antele sunt urmate de o linie goală și apoi de conținutul fișierului respectiv.

Pe lângă metoda *GET*, folosită pentru a obține resursa identificată cu ajutorul URI-ului, cel mai des sunt folosite următoarele metode: *PUT* – creează sau modifică o resursă, *DELETE* - șterge resursa identificată cu ajutorul URI-ului, *POST* – oferă date ce urmează a fi procesate de resursa identificată prin URI-ul asociat.

Un răspuns la o cerere HTTP va conține întotdeauna un cod ce va indica starea cererii respective. Există mai multe tipuri de astfel de coduri, ele fiind împărțite în clase[23]: *1xx* – conține codurile de informare; *2xx* – codurile acestei clase sunt folosite pentru a indica faptul că cererea a fost primită, înțeleasă și acceptată, cel mai cunoscut cod al acestei clase este 200 care indică faptul că cererea a fost procesată cu succes; *3xx* – codurile acestei clase indică faptul că trebuie parcurși pași suplimentari de către client pentru ca cererea să poată fi îndeplinită cu succes, un exemplu de cod al acestei clase ar fi 301 care indică faptul că resursei accesate i-a fost asociat un nou URI, permanent iar, în acest caz, pasul suplimentar ar fi reprezentat de utilizarea unui alt URI (cel nou asociat); *4xx* – codurile acestei clase sunt folosite atunci când server-ul consideră că a fost comisă o eroare de către client, de exemplu acesta a inițiat o cerere greșită, cel mai des întâlnit cod al acestei clase este 404 care indică faptul că nu a fost găsită resursa corespunzătoare URI-ului specificat în cerere, *5xx* – codurile acestei clase indică faptul că au apărut erori ale server-ului în timpul procesării cererii, un exemplu de cod al acestei clase este 500 care indică apariția unei erori interne în ceea ce privește serverul, acesta fiind incapabil să îndeplinească cererea.

2.2.2 SMTP

Să presupunem următoarea situație: Ana ce are asociată adresa de e-mail ana@abc.org îi trimite un e-mail Ioanei care are asociată adresa ioana@xyz.org. Ana își va compune e-mail-ul cu ajutorul MUA-ului propriu și apoi îl va trimite MTA-ului local. Acesta din urmă va trimite mesajul e-mail către server-ul de mail al domeniului abc.org folosind SMTP. După ce va identifica server-ul de mail al domeniului xyz.org, server-ul de mail al domeniului abc.org va transmite mai departe mesajul e-mail folosind tot SMTP. În final, mesajul va fi stocat în mailbox-ul destinatarului.

Trebuie specificat faptul că Mail User Agent este o aplicație folosită pentru a accesa și a administra mailbox-ul utilizatorului. În schimb, Mail Transfer Agent este folosit pentru transferul mesajelor e-mail între client și server.

Având în vedere exemplul anterior, se poate observa că scopul protocolului de nivel aplicație Simple Mail Transfer Protocol (SMTP) este acela de transfera mesajele de e-mail eficient și în mod fiabil. El este independent de sistemele care participă la comuncație, necesitând doar un canal care să asigure transmisia ordonată a datelor[24]. Implicit SMTP folosește portul 25.

Exemplul prezentat în introducerea acestei secțiuni a fost simplificat, în realitate, mesajul de e-mail transmis de MUA poate să nu fie recepționat direct de către MTA ci de către MSA (Mail Submission Agent). MSA este o aplicație care colaborează cu MTA pentru a asigura livrarea mesajului e-mail și folosește, de asemenea, SMTP. MTA și MSA pot folosi portul implicit, însă MSA are asociat un port dedicat și anume 587[25]. Prin urmare, SMTP poate folosi și portul 587 pentru a transmite mesajele de e-mail de la MUA la MTA.

2.3 STILUL ARHITECTURAL - REST

Representational State Transfer este un stil arhitectural folosit pentru proiectarea serviciilor web[26]. Principalul element pe care se bazează acest stil arhitectural este resursa. O resursă poate fi reprezentată de un document, o imagine, etc., practic, de orice concept ce reprezintă ținta unui hipertext[27].

Serviciile web proiectate cu ajutorul REST prezintă o arhitectură de tip client-server, în cazul căreia clientul și server-ul schimbă reprezentări ale resurselor folosind, de regulă, protocolul HTTP[28]. O reprezentare este o succesiune de octeți la care se adaugă metadate pentru descrierea octeților. De exemplu, documentul HTML este o reprezentare a unei pagini web[27].

Printre principalele avantaje ale unui serviciu RESTful se numără[26]:

- independența față de platformă, astfel, sistemul de operare al server-ului poate diferi de cel al clientului;
- independența față de limbajul de programare, astfel, o aplicație client scrisă în limbaj Java poate comunica fără probleme cu o aplicație de tip server scrisă în limbaj C++ sau C#;
- poate fi utilizat fără probleme în prezența unui firewall;
- este integrat foarte ușor în alte servicii web deja existente;
- este simplu de implementat.

Totuși, un astfel de serviciu prezintă și un dezavantaj, și anume faptul că folosește cookie-uri[26]. Un cookie este un mic fișier text stocat pe calculatorul utilizatorului ce-i oferă posibilitatea site-ului web careia ii este asociat de a obține anumite informații asupra clientului, ca de exemplu: preferințe, e-mail, nume utilizator, parolă, număr de telefon. Cookie-urile nu sunt sigure din moment ce sunt păstrate în text clar și , prin urmare, pot fi accesate de către oricine. Ele pot fi criptare și decriptate manual, însă acest lucru implică scrierea de cod în plus ce afectează performanța aplicației deoarece este necesar un timp suplimentar pentru criptare/decriptare[29].

2.4 Biblioteci utilizate în cadrul procesului de restaurare a diacriticelor pentru fișiere .docx

2.4.1 Apache POI

Apache POI este un API ce pune la dispoziția programatorilor o colecție de funcții destinate creării fișierelor MS Office dar și modificării și afișării conținutului acestora. Este o bibliotecă dezvoltată și distribuită de către Apache Software Foundation[30].

Câteva componente ale API-ului[30]:

- POIFS (Poor Obfuscation Implementation File System) – folosit pentru citirea explicită a diferitelor fișiere de tip MS Office, fiind baza tuturor celorlalte componente ale API-ului Apache POI;
- HSSF (Horrible Spreadsheet Format) – folosit pentru citirea și scrierea fișierelor MS-Excel de tip .xls;
- XSSF (XML Spreadsheet Format) – folosit pentru citirea și scrierea fișierelor MS-Excel de tip .xlsx;
- HPSF (Horrible Property Set Format) – extrage seturi de proprietăți ale fișierelor MS-Office;
- HWPF (Horrible Word Processor Format) – folosit pentru citirea și scrierea fișierelor MS-Word de tip .doc;
- XWPF (XML Word Processor Format) – folosit pentru scrierea și citirea fișierelor MS-Word de tip .docx;
- HSLF (Horrible Slide Layout Format) – folosit pentru citirea, crearea și editarea prezentărilor PowerPoint.

2.4.2 SRI-LM

SRI-LM este o colecție de biblioteci C++, programe executabile și script-uri ajutătoare destinată creării modelelor de limbă statistice dar și testării acestora din urmă. Utilitarul este folosit pentru conceperea și evaluarea unei varietăți de modele de limbă, cum ar fi cele bazate pe n-grame, însă permite și realizarea altor operații precum: manipularea listelor cu n-grame. SRI-LM tratează tot ce este încadrat de două spații albe ca pe un cuvânt și nu realizează

normalizarea și etichetarea textului. Aceste operații depind de textul de intrare și sunt îndeplinite cu ajutorul unor filtre care preprocezează datele[31].

În ceea ce privește restaurarea diacriticelor, aceasta este realizată cu ajutorul instrumentului numit *disambig*, pus la dispoziție de SRI-LM. Practic, *disambig* transformă un flux de cuvinte al vocabularului V_1 (în cazul de față, cuvinte fără dacritice) într-un flux corespondent de cuvinte al vocabularului V_2 (în cazul de față, cuvinte cu diacritice) cu ajutorul unui model de limbă bazat pe n-grame și a unei hărți probabilistice de tipul una-la-mai-multe.

Sintaxa comenzii folosite prentru restaurarea diacriticelor este următoarea: **disambig** *opțiuni* iar printre opțiuni se numără[32]:

- **-text** *file* folosită pentru specificarea căii către fișierul .txt care conține textul inițial, fără diacritice;
- **-map** *file* folosită pentru specificarea căii către fișierul .txt ce conține corespondețele cuvânt fără diacritice – cuvânt/cuvinte cu diacritice. Practic, opțiunea este folosită pentru specificarea căii către harta probabilistică de tipul una-la-mai-multe;
- **-lm** *file* – folosită pentru specificarea căii către modelul de limbă bazat pe n-grame;
- **-order** *n* – folosită pentru setarea ordinului n-gramelor. Implicit n este egal cu 2, ceea ce înseamnă că se utilizează un model de limbă bazat pe bigrame;
- **-keep-unk** – dacă se folosește această opțiune, cuvintele necunoscute ale textului de intrare și anume, cele care nu se regăsesc în harta cuvintelor, nu vor fi marcate folosind eticheta <unk> ci se vor regăsi nemodificate în corpusul rezultat în urma restaurării;
- **-fb** - folosit pentru a activa opțiunea de întoarcere la un model de limbă mai mic bazat pe n-grame atunci când nu este găsită o anumită succesiune de cuvinte în modelul de limbă mai mare.

Fie un fișier .txt ce conține următoarele linii:

gand la gand cu bucurie

vreau sa imi cumpar un smartphone

haina nu face pe om

Presupunând că numele fișerului anterior este *corpus.preprocessed* iar cele ale fișierelor asociate modelului de limbă, respectiv hărții cuvintelor sunt: *languageModel* și *wordsMap*, comanda utilizată pentru pornirea procesului de restaurare a diacriticelor ar putea arăta în felul următor:

disambig –lm /cale/fișier/LanguageModel –order 3 –map /cale/fișier/WordsMap –text /cale/fișier/corpus.preprocessed –keep-unk –fb

Se observă faptul că s-a optat pentru opțiunea de a păstra nemodificate cuvintele necunoscute (cum ar fi, în exemplul anterior, smartphone), pentru opțiunea de a utiliza un model de limbă care are la bază trigramele dar și pentru opțiunea de a utiliza un model de limbă mai mic atunci când o succesiune de trei cuvinte nu este regăsită în modelul de limbă mai mare. Având în vedere cele menționate, rezultatul procesului de restaurare va fi:

gând la gând cu bucurie

vreau sa îmi cumpăr un smartphone

haina nu face pe om

Dacă nu s-ar fi optat pentru folosirea opțiunii –keep-unk, a doua linie ar fi arătat în felul următor: *vreau să îmi cumpăr un <unk>*.

2.4.3 JavaNLP

Analizând exemplul oferit în cadrul secțiunii anterioare, 2.4.2, se poate observa faptul că textul din fișierul *corpus.preprocessed* nu prezintă semne de punctuație, caractere speciale sau majuscule. Totuși, textul inițial oferit de utilizator, în aproape 100% din cazuri, conține semne de punctuație, majuscule și caractere speciale. Eliminarea manuală a acestora de către utilizator nu reprezintă o variantă fiabilă deoarece, deși acest proces este unul rapid în cazul unei propoziții scurte, el devine unul imposibl de realizat într-un timp foarte scurt atunci când vine vorba despre un text cu zeci de mii sau milioane de propoziții. Prin urmare, se observă necesitatea unei metode care să permită „curățarea" rapidă a textului oferit de utilizator.

Horia Cucu prezintă în lucrarea „*Contribuţii la un sistem de recunoaştere de vorbire continuă, cu vocabular extins, independent de vorbitor pentru limba română"* o bibliotecă numită JavaNLP ale cărei funcții sunt destinate realizării operației de „curățare". Se vor prezenta în cele ce urmează numai acele operații necesare procesului de restaurare de diacritice, JavaNLP punând la dispoziție mai multe metode de curățare, cum ar fi cele utilizate de un sistem de recunoaștere de vorbire.

În ceea ce privește procesul de restaurare a diacriticelor, prima operație de „curățare", numită normalizare, are drept scop eliminarea cazurilor în care se folosesc mai multe caractere unicode pentru a exprima același lucru așa cum se întâmplă cu „ș" și „ş", „ț" și „ţ", „î" și „î", etc. Așadar, toate caracterele pentru care se pot defini astfel de cazuri sunt înlocuite cu principala variantă aleasă. De exemplu, se va înlocui „ș" cu „ş", „ț" cu „ţ", „î" cu „î", etc[3].

A doua operație de „curățare" presupune extragerea semnelor de punctuație și a caracterelor speciale, ele fiind înlocuite cu un spațiu. Apar excepții în cazul caracterului „-", care nu este extras dacă face parte dintr-o construcție de forma *„cuvânt$_1$-cuvânt$_2$"*, ca de exemplu: să-mi, într-o, dă-mi, etc[3].

În ceea ce privește ultima operație de curățare, majusculele sunt extrase fiind înlocuite cu varianta lor minusculă[3].

În concluzie, prin aplicarea succesivă a celor trei operații de „curățare" se va obține un flux de cuvinte ce va îndeplini toate condițiile anterior menționate pentru a putea fi restaurat.

2.5 SERVER-UL DE APLICAȚII - GLASSFISH

GlassFish este un server de aplicații care permite dezvoltatorilor să conceapă tehnologii convenabile și scalabile, destinate companiilor. De asemenea, oferă servicii suplimentare ce pot fi instalate în funcție de preferințe[33].

GlassFish introduce conceptul numit domeniu. Un domeniu nu este altceva decât un motor Java EE. Odată ce domeniul este pornit, el reprezintă server-ul. Astfel, domeniul va putea găzdui aplicațiile Java EE proiectate de dezvoltatori[34].

Pentru ca o aplicație web să poată fi găzduită de către un domeniu al server-ului, este necesară cearea unui WAR ce va conține întreaga aplicație web, însă în fomă comprimată. După obținerea WAR-ului acesta va fi implemetat în domeniu cu ajutorului unei comenzi simple.

CAPITOLUL 3

SERVICIUL WEB DE RESTAURARE A DIACRITICELOR

3.1 DESCRIERE GENERALĂ

Serviciul web prezentat în acest capitol este destinat restaurării diacriticelor într-un fișier .docx. Astfel, principala sarcină a acestuia va fi restaurarea de diacritice ținând cont de necesitatea păstrării aspectului fișierului .docx oferit de utilizator. Acesta din urmă poate să nu conțină deloc diacritice sau poate să prezinte doar anumite părți fără diacritice, poate să fie un document Word simplu, conținând doar text, sau poate conține imagini sau tabele, caz în care procesul de restaurare va trebui să ia în calcul și textul asociat acestor elemente. În ceea ce privește imaginile, textul asociat este reprezentat de cel din titluri, iar în ceea ce privește tabelele, textul asociat este cel din celule.

Diferitele funcții ale serviciului sunt puse la dispoziția utilizatorului prin intermediul unei interfețe grafice atractive. Astfel, după ce a completat corect câmpul care are asociat eticheta „E-mail”, utilizatorul va putea alege fișierul .docx dorit, pornind procesul de restaurare. Dacă informația introdusă în câmpul corespunzător adresei de e-mail nu prezintă structura unei astfel de adrese, atunci, utilizatorul nu va putea alege fișierul .docx dorit și prin urmare procesul de restaurare nu va porni, așa cum reise și din Figura 3.1. În cazul contrar, utilizatorul va putea alege fișerul dorit iar procesul de restaurare va porni, afișându-se un mesaj de informare, așa cum se poate observa și în Figura 3.2. După încheierea procesului de restaurare fișierul rezultat, ce va conține diacritice, va fi trimis la adresa de e-mail specificată. Totuși, el va putea fi și descărcat prin apăsarea unui buton numit „Descarcare”, așa cum sugerează Figura 3.3, dacă utilizatorul va alege să nu părăsească pagina până la finalizarea restaurării și a trimiterii e-mail-ului. În cazul în care apar erori în timpul procesului de restaurare , acesta va fi oprit iar utilizatorul va fi înștiințat printr-un mesaj corespunzător, așa cum arată Figura 3.4.

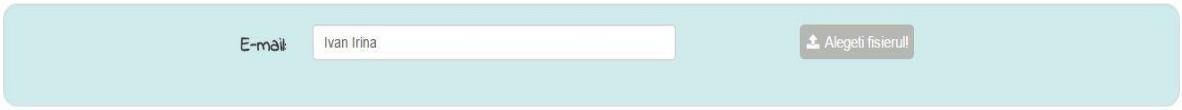

Figura 3.1 Utilizatorul introduce o informație incorectă în câmpul destinat adresei de e-mail

Figura 3.2 Utilizatorul este înștiințat de faptul că procesul de restaurare este în curs de desfășurare

Figura 3.3 Utilizatorul este înștiințat de faptul că poate descărca fișierul rezultat

Figura 3.4 Mesajul de eroare

Interacțiunea dintre utilizator și serviciul web poate fi descrisă pe scurt și cu ajutorul unei diagrame de fluxuri (flow chart). O astfel de diagramă este o reprezentare grafică a unui algoritm sau a modului de desfășurare a unui proces[35] și folosește simboluri distincte pentru a descrie diferiții pași asociați algoritmului sau procesului în cauză[36].

Printre simbolurile ce pot fi folosite pentru realizarea unei diagrame de fluxuri se numără[35]:

- Ovalul – folosit pentru a indica începutul respectiv finalul unui algoritm sau al unui proces;
- Dreptunghiul – folosit pentru a evidenția o acțiune;
- Rombul – folosit pentru a marca un punct de decizie;
- Paralelogramul – folosit pentru a reprezenta datele de intrare sau de ieșire;

Astfel, o diagramă de fluxuri a fost folosită pentru a evidenția modul de utilizare al serviciului web, așa cum se poate observa analizând Figura 3.5. Trebuie precizat faptul că, după părăsirea paginii web asociate serviciului, utilizatorul poate accesa propriul mailbox pentru a verifica primirea mesajului e-mail ce are atașat fișierul .docx rezultat. În acest fel el va putea să descarce fișierul restaurant, dacă a părăsit pagina înainte de a i se oferi posibilitatea de a salva direct documentul. În această secțiune, accentul nu a fost pus pe modul în care se realizează restaurarea diacriticelor pentru fișierele .docx. Detalii mai amănunțite vor fi oferite în secțiunea următoare, 3.2.

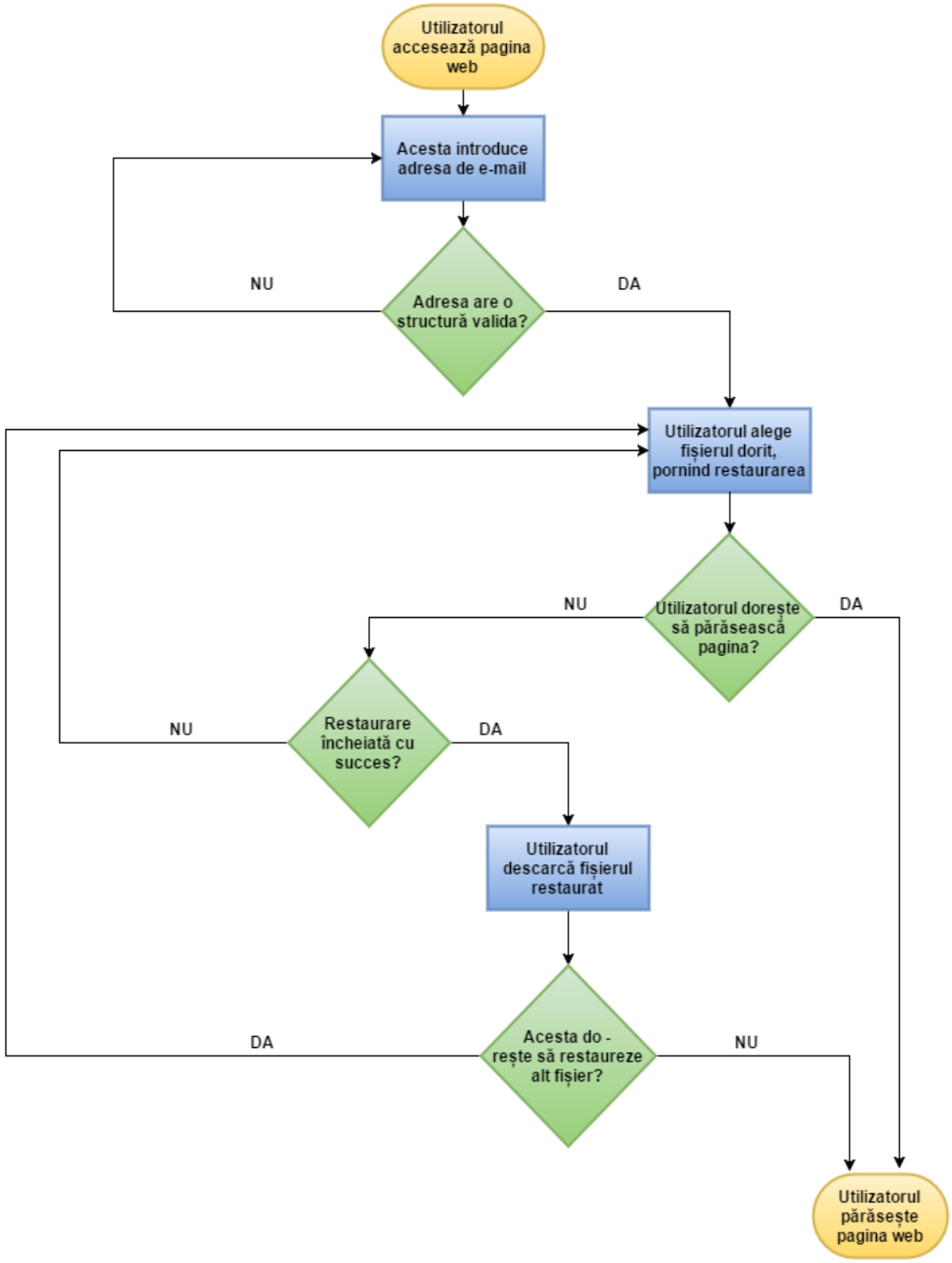

Figura 3.5 Scenariul de utilizare al serviciului web de restaurare a diacriticelor pentru fișiere .docx

3.2 ARHITECTURA SISTEMULUI ȘI MODUL DE FUNCȚIONARE

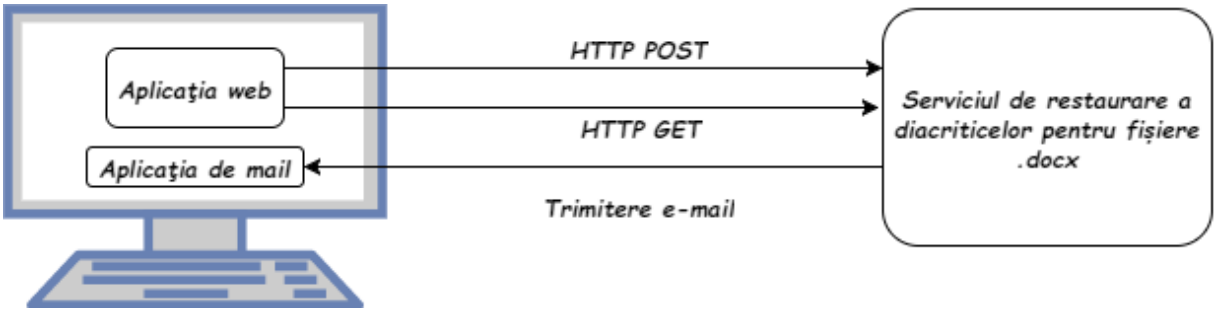

Figura 3.6 Arhitectura sistemului

În Figura 3.6 este prezentată arhitectura sistemului ce pune în evidență modul în care comunică principalele elemente ale acestuia. Astfel, odată ce pagina web asociată serviciului este încărcată în browser iar utilizatorul poate alege fișierul dorit, se va trimite, pentru pornirea procesului de restaurare, o cerere HTTP POST către serviciul web, aceasta conținând adresa de e-mail introdusă de utilizator și fișierul ales. După ce serviciul realizează restaurarea, fișierul va putea fi descărcat direct, însă va fi necesară trimiterea unei cereri HTTP GET pentru a se realiza acest lucru. Mai mult, serviciul va trimite fișierul rezultat și la adresa de e-mail specificată la început de utilizator.

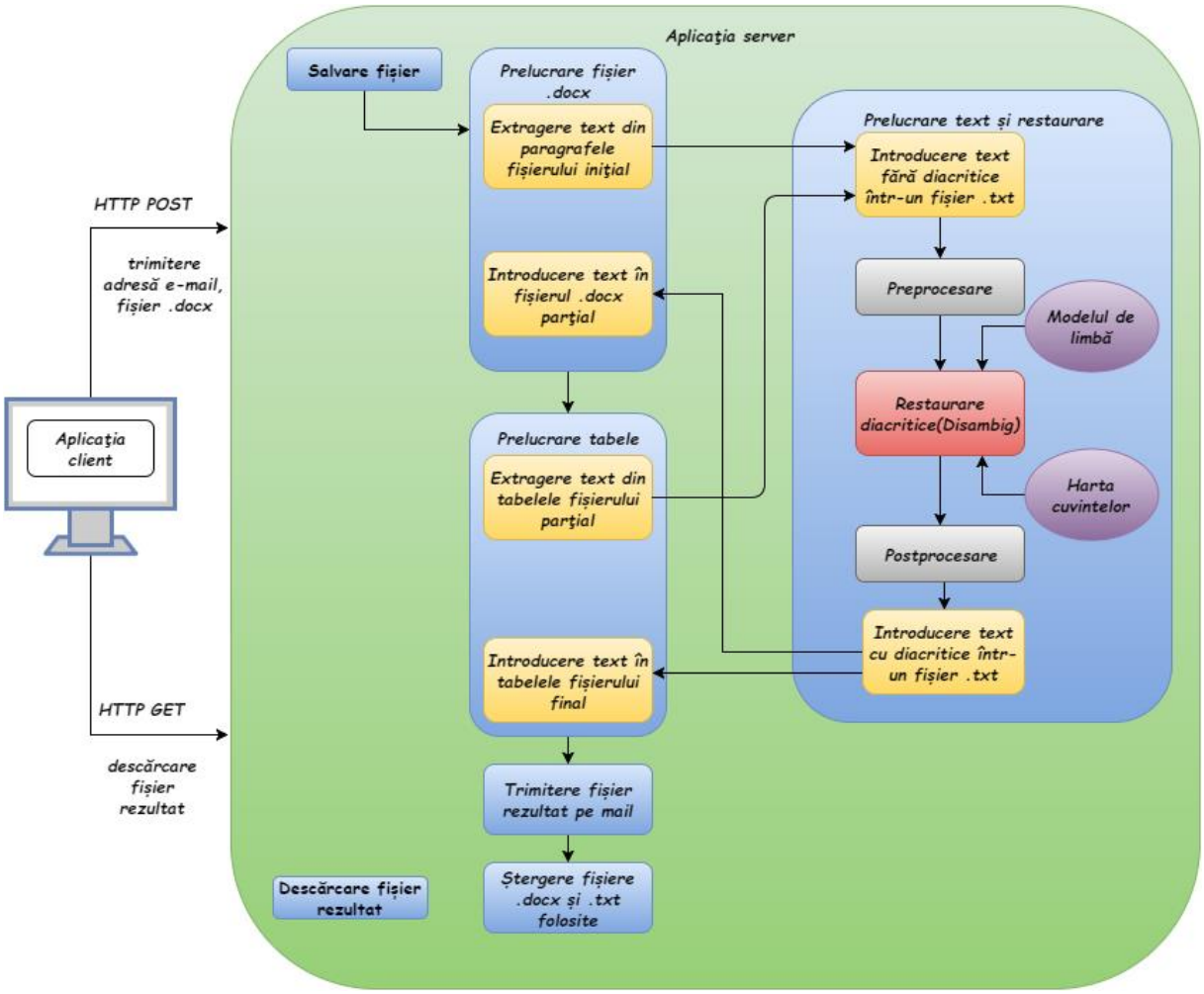

Figura 3.7 Modul de funcționare al sistemului

În ceea ce privește modul de funcționare al serviciului, acesta este prezentat în Figura 3.7. Astfel, după ce utilizatorul introduce o adresă de e-mail cu o structură corectă, el va putea alege fișierul .docx pe care dorește să-l restaureze. Odată ce fișierul este ales, se va trimite către server o cerere HTTP POST, ce va conține adresa de e-mail specificată și fișierul respectiv. Cererea va fi prelucrată, obținându-se informații asupra adresei de e-mail și a fișierului în cauză. Se va salva apoi fișierul pe server utilizând funcțiile puse la dispoziție de blocul de salvare și se va trece mai departe la prelucrarea acestuia. Prelucrarea va consta în extragerea textului din fiecare paragraf ce conține doar text și introducerea acestuia într-un fișier .txt asociat, fiind făcută de blocul numit *„Prelucrare fișier .docx”*. După ce se finalizează această etapă, se trece la „curățarea” textului din fișierul .txt de intrare ce are ca scop normalizarea caracterelor, extragerea semnelor de punctuație și a caracterelor speciale și înlocuirea majusculelor cu formele lor minuscule. Textul rezultat va fi restaurat de către instrumentul *disambig*, obținându-se textul cu diacritice asupra căruia se aplică operația de postprocesare ce presupune adăugarea înapoi a majusculelor și a semnelor de punctuație și caracterelor speciale. Se va crea apoi un fișier .docx parțial prin copierea fișierului inițial și introducerea în locurile corespunzătoare a textului cu diacritice. În continuare, se vor utiliza funcțiile blocului numit *„Prelucrare tabele”* pentru a stabili dacă fișierul parțial prezintă sau nu tabele. În cazul în care fișierul conține tabele, se va extrage textul din fiecare celulă și se vor repeta etapele precizate anterior pentru paragrafele ce conțin doar text, în final rezultând fișierul restaurat. Dacă în fișierul parțial nu se regăsesc tabele, atunci acesta va fi pur și simplu copiat în fișierul final. Fișierul rezultat va fi trimis apoi la adresa de e-mail specificată de utilizator. Acesta din urmă, pe tot parcursul timpului asociat parcurgerii etapelor menționate anterior, va fi înștiințat, printr-un mesaj precum cel din Figura 3.2, de faptul că procesul de restaurare este în curs de desfășurare. După ce mail-ul este trimis, se șterg fișierele ajutătoare, fișierul inițial și se pornește procesul de ștergere a fișierului restaurat, ștergere ce se realizează după un interval de treizeci de minute. În final, utilizatorul este înștiințat de faptul că procesul de restaurare s-a încheiat și i se oferă posibilitatea de descărcare, dacă acesta dorește. Pentru a descărca fișierul, el va trebui să apese pe butonul „Descărcare”. Se va trimite astfel o cerere HTTP GET către server ce va fi prelucrată de blocul numit *„Descărcare fișier”*.

3.3 IMPLEMENTAREA SISTEMULUI

Serviciul web de restaurare a diacriticelor a fost implementat ca un serviciu RESTful cu ajutorul limbajului de programare Java. Pentru ca un utilizator neexperimentat să poată beneficia de funcțiile oferite de serviciu, s-a optat și pentru implementarea unei pagini web atrăgătoare, prin folosirea limbajului HTML și a framework-ului Bootstrap. Diferitele elemente ale paginii au devenit funcționale cu ajutorul colecției de metode puse la dispoziție de limbajul JavaScript. În continuare vor fi oferite câteva detalii asupra modului de implementare al paginii web, a funcțiilor acesteia dar și a funcțiilor blocurilor prezentate în Figura 3.7.

3.3.1 Implementarea paginii web

Așa cum a fost precizat anterior, pentru implementarea paginii web s-a folosit limbajul HTML în corespondență cu framework-ul Bootstrap, codul sursă fiind prezentat în Anexa1. Inițial, s-a încercat o implementare doar cu ajutorul limbajului HTML, obținându-se o pagină web cu un aspect rudimentar al cărei principal dezavantaj era reprezentat de faptul că, în urma micșorării ferestrei browser-ului, pagina nu era redimensionată. Astfel anumite elemente ale paginii nu mai erau vizibile (de exemplu: butonul de decărcare) și prin urmare câteva funcții ale acesteia nu mai puteau fi accesate. Prin urmare, s-a luat decizia de a se folosi framework-ul Bootstrap, nu numai pentru a proiecta o pagină web care să se adapteze la dimensiunile

ferestrei browser-ului, ci și pentru a proiecta o pagină mult mai atrăgătoare, deoarece, prin folosirea șabloanelor oferite de Bootstrap, accentul s-a pus pe dezvoltarea propriului fișier .css pentru înfrumusețarea aspectului elementelor și nu pe operația de proiectare a acestora.

Pentru conceperea diferitelor funcții asociate paginii web s-au folosit metodele puse la dispoziție de jQuery și AJAX, codul sursă asociat funcțiilor fiind și el prezentat în Anexa 1. De exemplu, s-a folosit elementul cheie al tehnicii AJAX și anume obiectul XMLHttpRequest pentru a permite pornirea procesului de restaurare după apăsarea butonului de alegere a fișierului și alegerea efectivă a acestuia. Un astfel de obiect a fost creat în felul următor:

var serverRequest = new XMLHttpRequest();

El prezintă două metode: open() și send() folosite pentru specificarea tipului cererii necesare pentru pornirea procesului de restaurare și respectiv, pentru trimiterea cererii către server. Cererea trimisă va fi una de tipul HTTP POST și va conține adresa de e-mail a utilizatorului și fișierul .docx ales, acestea fiind obținute prin intermediul formularului prezentat în Figura 3.2. Elementele precizate anterior vor fi trimise mult mai ușor, cu ajutorul metodei send(), dacă se creează un obiect de tipul FormData. Un astfel de obiect va conține un set de perechi de tipul id câmp formular/valoare câmp, ce vor fi adaugate cu ajutorul metodei append() în felul următor:

var formData = new FormData();

formData.append("email", email);

formData.append("docxfile", file);

Trimiterea adresei de e-mail și a fișierului se va face în felul următor:

serverRequest.send(formData);

Însă nu înainte de a se stabili tipul cererii cu ajutorul metodei *open()*:

serverRequest.open("POST",
"http://ivan.studenti.speed.pub.ro/RestaurareDiacritice/webresources/diacriticsrestorer/submit", true);

Se observă astfel că cererea va fi una de tipul HTTP POST și că va avea drept țintă URL-ul precizat. True reprezintă valoarea câmpului async și specifică faptul că cererea este una asincronă, cu alte cuvinte, specifică faptul că se pot executa alte operații până în momentul în care apare răspunsul server-ului (de exemplu, în cazul de față, se afișează un mesaj prin care utilizatorul este înștiințat de faptul că procesul de restaurare este în curs de desfășurare).

Fiecare obiect XMLHttpRequest prezintă atributul readyState, ce oferă informații asupra statusului obiectului, putând lua următoarele valori: 0 – obiectul a fost creat dar nu a fost apelată metoda open(), 1 – metoda open() a fost invocată cu succes, 2 – a fost invocată metoda send() și s-au primit antetele răspunsului HTTP, 3 – începe primirea corpului răspunsului HTTP, 4 – s-a primit răspunsul[37]. De fiecare dată când readyState își schimbă valoarea este declanșat evenimentul numit onreadystatechange, folosit pentru a specifica ceea ce se va întâmpla atunci când răspunsul server-ului va fi primit.

Având în vedere cele menționate, secvența de cod utilizată pentru a specifica faptul că se afișează un mesaj de așteptare până la finalizarea procesului de restaurare, urmând ca atunci utilizatorul să fie informat asupra faptului că poate descărca fișierul rezultat, este următoarea:

$("#converting").show();//afișare mesaj: proces de restaurare în curs de desfășurare

serverRequest.onreadystatechange = function (){

```
        if (serverRequest.readyState === 4)
                {
                $("#converting").hide();//ascunderea mesajului anterior
                 if (serverRequest.status === 200)
                        {
                            $("#converted").show();//afișare mesaj: restaurare efectuată
                        }
                else{
                     if (serverRequest.status === 500) {
                        $("#file_error").show();//mesaj de eroare
                     }
                 }
        }
};
```

3.3.2 Implementarea blocului „Prelucrare fișier .docx”

Fișierele text complexe, precum cele cu extensia .docx, diferă față de fișierele .txt nu numai prin faptul că pot prezenta elemente ce nu reprezintă text (imagini, diagrame, tabele .etc) ci și prin faptul că oferă utilizatorului numeroase posibilități de formatare (de la schimbarea culorii textului și până la așezarea în pagină). Instrumentul disambig, ce realizează restaurarea diacriticelor, nu a fost proiectat pentru a ține cont de formatări, prin urmare a fost necesară găsirea unei metode de restaurare a diacriticelor fără a modifica aspectul fișierului .docx.

Având în vedere metoda de restaurare a diacriticelor pentru fișiere .txt propusă de Ana-Maria Vladu în lucrarea *„Serviciu web de restaurare a diacriticelor”*, s-a luat decizia ca textul fișierului .docx oferit de utilizator să fie extras și apoi introdus într-un fișier .txt asupra căruia să se aplice metoda prezentată în lucrarea amintită. Ținând cont de faptul că fișierul rezultat trebuia să fie unul .docx și nu unul .txt, s-a decis ca textul restaurat să fie introdus în fișierul .docx final, ținând cont de poziția acestuia în documentul inițial dar și de formatări. Toate aceste prelucrări, corespunzătoare blocului *„Prelucrare fișier .docx”*, au fost realizate prin folosirea funcțiilor bibliotecii Apache POI, funcții ce sunt specifice citirii și scrierii documentelor cu extensia .docx.

Metoda de prelucrare a fișierelor .docx prezentată în această lucrare și al cărei cod sursă este listat în Anexa 1(metodele restoreDiacriticsForText și restoreDiacriticsForTabels), ține cont de structura unui astfel de document și pune în evidență faptul că un paragraf al acestuia este alcătuit din porțiuni mai mici. O astfel de porțiune este caracterizată de faptul că fragmentele de text asociate prezintă același set de proprietăți, așa cum se poate observa și în Figura 3.8. De fapt, acest mod de definire a unui paragraf pune în evidență faptul că pe lângă text, acesta conține și informații referitoare la formatare.

Astfel, extragerea textului se face, de fapt, din fiecare porțiune a unui paragrapf. Pentru a se putea realiza acest lucru este necesară, mai întâi, obținerea unei liste de tipul XWPFParagraph care să conțină toate paragrafele documentului. Se va parcurge apoi această listă iar în cazul fiecărui paragraf se va defini o altă listă, de data aceasta de tipul XWPFRun care va conține porțiunile ascociate paragrafului. Și această listă va fi parcursă cu scopul de a se extrage, pentru fiecare porțiune în parte, textul asociat care va fi introdus într-un fișier .txt.

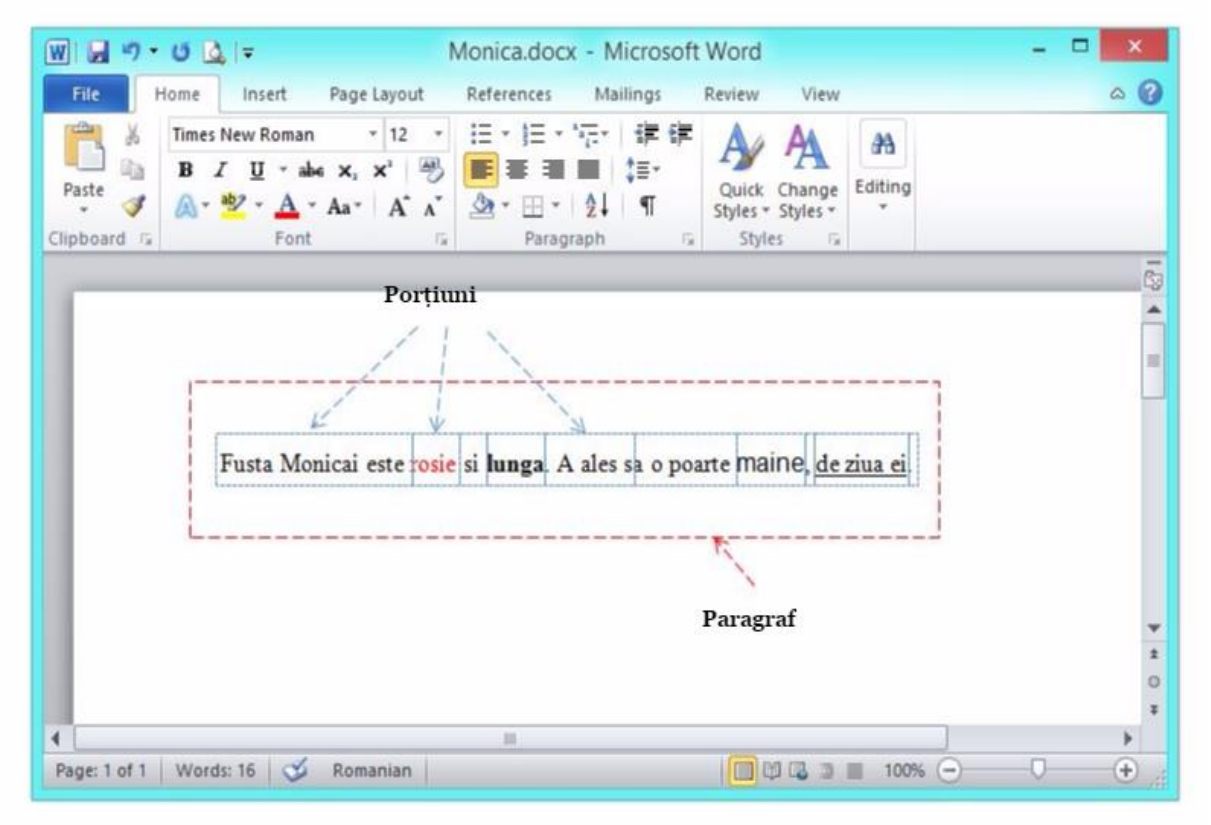

Figura 3.8 Structura unui document .docx

După ce blocul *„Prelucrare text și restaurare”* își îndeplinește funcțiile, generând un fișier .txt în care se regăsește textul cu diacritice, se va parcurge din nou lista de paragrafe iar, pentru fiecare paragraf în parte, lista de porțiuni, pentru a se înlocui textul fără diacritice cu textul restaurant, în final obținându-se un fișier .docx cu diacritice.

În ceea ce privește prelucrarea documentelor Word ce conțin, pe lângă text, și tabele se va proceda asemănător, deoarece fiecare celulă a tabelului conține unul sau mai multe paragrafe. Cu alte cuvinte, fișierul .docx rezultat în urma etapei menționate mai sus, al cărui text va conține diacritice, cu excepția celui asociat tabelelor, va fi parcurs pentru a extrage textul din fiecare celulă.

Pe lângă tabele, un fișier .docx mai poate conține și imagini. Având în vedere că o imagine nu conține text, problema impusă de existența acesteia într-un document Word a fost soluționată în felul următor: se verifică dacă porțiunile conțin sau nu text, înainte de a se trece la etapa de introducere a textului în fișierul .txt, așa cum se poate observa și în urma analizării codului sursă prezentat în Anexa 1.

3.3.3 Implementarea blocului „Trimitere fișier rezultat pe mail”

Scopul acestui bloc este acela de a construi și de a trimite la adresa specificată de utilizator mesajul de e-mail, ce are atașat fișierul rezultat în urma finalizării procesului de restaurare a diacriticelor.

Formatul unui mesaj e-mail ce conține numai text prezintă mai multe câmpuri numite antete și un corp al mesajului. În general, câmpurile amintite anterior sunt referite mult mai simplu, doar prin folosirea cuvântului „antet”. Astfel, corpul mesajului poate fi opțional, însă antetul este obligatoriu. Cu alte cuvinte, mesajul nu poate fi trimis dacă nu prezintă antet, însă poate fi trimis dacă porțiunea asociată corpului mesajului lipsește[38]. Acest lucru se datorează faptului că antetul conține informații importante precum adresa de e-mail a expeditorului sau

adresa de e-mail a destinatarului. Revenind la corpul mesajului, acesta este separat de antet printr-o linie goală și este alcătuit dintr-o secvență de caractere ASCII.

Atunci când se dorește transmiterea unui mesaj e-mail care să conțină pe lângă text și imagini sau documente Word, cum este cazul serviciului web prezentat în acest capitol, se va apela la standardul MIME. Acesta extinde formatul mesajului e-mail prezentat anterior pentru a permite includerea mai multor secțiuni în corpul mesajului, secțiuni ce pot conține elemente text sau elemente ce nu reprezintă text, precum cele menționate mai sus. Totuși, aceste secțiuni vor trebui separate iar acest lucru se va face prin intermediul unui parametru numit boundary, ce constă într-o secvență de caractere diferită de cele care se regăsesc în secțiunile corpului mesajului și precedată de două cratime „--”[38].

Pentru a trimite fișierul restaurat la adresa de e-mail menționată de utilizator a fost necesară conceperea mesajului ținând cont de structura prezentată în paragraful anterior, codul sursă rezultat fiind și el prezentat în Anexa 1(metoda SendMail din clasa Submit). Protocolul folosit a fost SMTP iar portul asociat, 587. În ceea ce privește server-ul de mail, s-a ales folosirea celui oferit de Google. De asemenea, s-a optat pentru transformarea conexiunii existente între client și server într-una sigură prin activarea opțiunii STARTTLS. Astfel prin folosirea standardului SSL sau TLS s-a asigurat integritatea și protecția datelor transmise. Atât SSL cât și TLS oferă o metodă de criptare a canalului de comunicație dintre client și server, diferența dintre ele fiind reprezentată de faptul că TLS este succesorul SSL[39].

Pentru a putea fi realizate toate aspectele precizate mai sus, a fost necesară crearea unui obiect de tipul Session astfel: *Session session = Session.getDefaultInstance(prop, null);* , unde prop este un obiect de tipul Properties ce oferă informații sesiunii în cauză asupra server-ului de mail ales, portului folosit, precum și alte informații necesare autentificării astfel încât să se permită accesul la mailbox-ul asociat expeditoului. Practic, prin intermediul sesiunii default create se pot realiza toate operațiile necesare transmiterii e-mail-ului, de la construirea antetului și corpului mesajului și până la transmiterea efectivă a acestuia. O astfel de sesiune, ce poate conține informații confidențiale, poate fi accesată de către toate funcțiile ce se execută în același mediu JVM dacă obiectul asociat funcțiilor, de tipul Authenticator, este același cu obiectul de același tip asociat sesiunii (în cazul de față ambele obiecte trebuie să fie null). Trebuie precizat faptul că pentru a se putea trimite fișierul restaurat pe mail a fost necesară crearea unei adrese de e-mail: diacriticsrestorer@gmail.com care să fie asociată expeditorului.

Construirea antetului a presupus specificarea adresei de e-mail a expeditorului și a destinatarului precum și a subiectului e-mail-ului, care în cazul de față este „Rezultat proces restaurare”. În ceea ce privește corpul mesajului, acesta a fost construit ca având două secțiuni: prima secțiune fiind asociată textului „Primiti acest e-mail deoarece ati folosit Serviciul web de restaurare de diacritice al http://speed.pub.ro/.” ce are drept scop informarea utilizatorului asupra motivului pentru care a primit e-mail-ul iar în ceea ce privește a doua secțiune, aceasta a fost destinată fișierului .docx restaurat.

Odată construit mesajul s-a trecut la ultima etapă și anume la transmiterea acestuia prin protocolul SMTP. Pentru aceasta a fost necesară stabilirea unei conexiuni cu server-ul ales, care a fost închisă abia după ce e-mail-ul a fost transmis.

3.3.4 Implementarea blocului ce se ocupă de ștergere

Principalul rol al acestui bloc este acela de a asigura ștergerea fișierului .docx inițial, a fișierelor .docx și .txt parțiale dar și de a declanșa procesul de ștergere a documentului Word restaurat. Acesta va fi șters după treizeci de minute de la crearea sa pentru a-i oferi posibilitatea utilizatorului de a-l descărca de mai multe ori. De asemenea, toate aceste ștergeri

se fac pentru a se evita situațiile în care informațiile oferite de utilizator sunt accesate de către altcineva.

Liniile de cod folosite, ce au făcut posibilă ștergerea fișierului .docx restaurat, după un anumit interval de timp, sunt:

String[] deleteCommand = {"bash", "-c", "sleep 1800 " + "&& rm " + _resultPath + fileName};

Process deleteProcess = Runtime.getRuntime().exec(deleteCommand);

Analizând cele două linii, utilizate de către metoda deleteResultFiles prezentată în Anexa 1(clasa Submit), se poate trage concluzia că serviciul web rulează pe un server ce prezintă un sistem de operare Linux și că, prin urmare, se folosește o comandă Linux pentru a se realiza ștergerea.

Fiind vorba de o comandă Linux, automat trebuie precizat termenul „bash". Bash este un interpretor de comenzi, ce procesează comenzile introduse de utlizator direct în terminal dar care poate citi și procesa și comenzile dintr-un script.

Amânarea momentului ștergerii a putut fi realizată prin folosirea comenzii sleep, ce a necesitat specificarea perioadei de timp în care execuția procesului de ștergere a fost suspendată. Dorindu-se un interval de așteptare de treizeci de minute, perioada anterior menționată a fost considerată de 1800 de secunde ($1800 = 30 \times 60(s)$). Dacă procesul de așteptare nu este întrerup și se termină cu succes se va realiza apoi ștergerea efectivă a fișierului a cărui cale este specificată la finalul comenzii ce a fost numită *deleteCommand*.

Pentru ca procesul de ștergere să poată fi pornit, va fi creat un obiect de tipul Process deoarece clasa Process ce face parte din pachetul java.lang pune la dispoziția programatorilor metode ce permit pornirea execuției unui proces, verificarea statusului acestuia, așteptarea terminării execuției procesului, omorârea acestuia, etc.

În ceea ce privește ștergerea celorlalte fișiere menționate la începutul secțiunii de față, nu se mai folosește o comandă Linux. Astfel, ștergerea se va face mai simplu, doar cu ajutorul metodei delete() a clasei File.

3.4 SERVICIU RESTFUL

Așa cum a fost precizat și in secțiunea 2.3, REST este un stil arhitectural de proiectare al serviciilor web ce prezintă ca și principal element resursa. O resursă poate fi reprezentată de o imagine, un fișier text dar și de o metodă, o funcție și are asociat un URI.

Analizând cererea HTTP POST prezentată în secțiunea 3.3.1, se observă folosirea URL-ului http://ivan.studenti.speed.pub.ro/RestaurareDiacritice/webresources/diacriticsrestorer/submit, ce este asociat metodei sau funcției care salvează și prelucrează fișierul .docx de intrare, realizează restaurarea, trimite e-mail-ul și realizează operația de ștergere. Astfel, funcția este privită ca o resursă. Mai sus s-a precizat faptul că o resursă are asociat un URI, și totuși în exemplul prezentat se folosește un URL. Acest lucru se întâmplă datorită faptului că un URL este practic un URI care specifică și metoda de accesare a resursei[41]. Astfel, pentru resursa identificată prin URI-ul parțial /webresources/diacriticsrestorer/submit, URL-ul asociat este cel menționat anterior.

Având în vedere că serviciul de restaurare a diacriticelor prezentat este unul RESTful, construirea URL-ului asociat unei resurse se face cu ajutorul adnotației *@Path*. Aceasta se regăsește înaintea definirii clasei în care este declarată metoda ce reprezintă ținta URL-ului

dar și înainte de definirea metodei. În cazul de față clasa este numită *Submit* iar metoda în cauză *submit*, codul sursă al acestora fiind prezentat în Anexa 1.

```
@Path("/diacriticsrestorer")
public class Submit {
@POST
@Path("/submit")
@Consumes(MediaType.MULTIPART_FORM_DATA)
    public Response submit(@Context HttpServletRequest request) throws Exception {
```

Urmărind al doilea exemplu prezentat anterior se poate observa prezența adnotațiilor *@POST* și *@Consumes*. Prima adnotație, *@POST*, subliniază faptul că metoda submit procesează cereri HTTP POST iar a doua adnotație, *@Consumes*, precizează tipul reprezentărilor transmise de client și pe care resursa le poate consuma sau, mai bine zis, le poate procesa.

Clasa Submit mai prezintă o metodă ce poate fi privită ca o resursă:

```
@GET
@Path("download")
@Consumes(MediaType.TEXT_PLAIN)
@Produces("application/vnd.openxmlformats-officedocument.wordprocessingml.document")
    public Response downloadFile(@Context HttpServletRequest request) throws Exception {
```

Așa cum subliniază și numele, metoda este folosită pentru a descărca de pe server fișierul .docx rezultat și folosește o nouă adnotație: *@Produces*. Aceasta specifică tipul reprezentărilor produse de resursă și transmise clientului. În cazul de față avem de-a face cu un document .docx.

Atât în cazul metodei *submit* cât și în cazul metodei *downloadFile* se poate observa prezența adnotației *@Context* care injectează obiectul request de tipul HttpServletRequest, folosit pentru a obține informații asupra cererii HTTP.

În final, trebuie precizat faptul că serviciul RESTful a fost configurat cu ajutorul clasei ApplicationConfig care a fost adnotată astfel: @javax.ws.rs.ApplicationPath("webresources"). Prin urmare, având în vedere faptul că URL-ul asociat metodei submit conține și setul de caractere „webresources”, și adnotația anterioară este luată în calcul în vederea construirii URL-ului.

3.5 Problemele apărute în timpul proiectării serviciului web și rezolvarea lor

Problemele apărute în timpul proiectării serviciului web au fost de cele mai multe ori generate de faptul că un document Word nu va fi niciodată scris corect în totalitate. Vor exista întotdeauna greșeli apărute din grabă sau din neatenție precum introducerea unui spațiu alb (space) după terminarea unui paragraf sau înaintea paragrafului.

Prin extragerea textului din fișierul .docx oferit de utilizator, astfel de spații, incorect plasate, nu dispar, ci sunt copiate și ele în fișierul .txt ce urmează a fi prelucrat pentru a se putea face apoi restaurarea diacriticelor. Așa cum s-a menționat și în secțiunea 2.4.3, prelucrarea sau,

mai degrabă, „curățarea textului” se face cu ajutorul metodelor puse la dispoziție de către biblioteca JavaNLP. Printre operațiile de „curățare” se numără extragerea semnelor de punctuație, a caracterelor speciale și a spațiilor (nu și a spațiilor dintre cuvinte) și introducerea acestora întru-un alt fișier .txt. Să presupunem că numele fișierului .txt în care se află textul extras din documentul Word este *nume.in*. În aceste condiții elementele extrase, anterior menționate, sunt introduse în fișierul text *nume.punctuation*. În acest fișier se vor regăsi trei coloane: pe prima coloană va fi trecut elementul extras, de exemplu: „.”, „?”, pe a doua coloană va fi trecut numărul liniei pe care acesta se afla înainte de a fi extras iar pe a treia coloană poziția sa în cadrul șirului de caractere ce se regăsește pe linia respectivă. Următoarea operație de „curățare”, executată imediat după încheierea celei prezentate mai sus, este cea de înlocuire a majusculelor cu varianta lor minusculă, ținându-se o evidență a literelor mari existente în textul inițial cu ajutorul un fișier text numit *nume.case*, ce prezintă aceeași structură ca și *nume.punctuation*.

S-a menționat faptul că prima operație de „curățare” include și excluderea spațiilor și introducerea lor în fișierul *nume.punctuation*, însă nu s-a precizat faptul că acest lucru nu este valabil și în cazul în care se regăsește un singur spațiu la sfârșitul paragrafului sau la începutul paragrafului. Instrumentul disambig, elimină însă aceste spații și, prin urmare, ele nu se vor mai regăsi în fișierul .txt restaurat numit *nume.rsd-preliminary*. Înainte de a copia textul restaurat în fișierul .docx final, se introduc înapoi majusculele, semnele de puctuație și caracterele speciale ținând cont de informațiile din fișierele *nume.case* și *nume.punctuation*, rezultând fișierul .txt final numit *nume.out*. Acest fișier nu va conține nici el spațiile considerate la începutul acestui paragraf, astfel că, în momentul înlocuirii textului fără diacritice cu textul cu diacritice vor apărea erori deoarece se va încerca înlocuirea unui șir de caractere mai mare cu un șir de caractere mai mic. În alte cazuri acest lucru ar fi posibil, însă în cazul de față apar erori deoarece fiecare caracter trebuie înlocuit cu versiunea sa din fișierul .txt. Prin urmare a fost necesară modificarea anumitor porțiuni de cod ale bibliotecii JavaNLP, pentru a se lua în considerare și situația prezenței unui spațiu la începutul sau la sfârșitul paragrafului. Astfel, în urma modficării făcute și aceste spații sunt introduse în fișierul *nume.punctuation* și prin urmare nu mai apar probleme în cazul introducerii textului cu diacritice în fișierul .docx final.

O altă problemă a fost generată de prezența rândurilor goale în fișierul .docx oferit de utilizator sau a rândurilor care conțin doar semne de punctuație, caractere speciale sau doar spații deoarece și acestea, după operația de curățare, devin rânduri goale. Instrumentul diasmbig, așa cum elimină și spațiile de la începutul sau sfârșitul unui paragraf, elimină și rândurile goale, ceea ce a creat o problemă în momentul generării fișierului *nume.out*. Prin urmare s-a luat decizia de a extrage și liniile goale, ținând o evidență a acestora într-un fișier asemănător lui *nume.case* sau *nume.punctuation*, urmând ca acestea să fie din nou introduse pentru crearea fișierului *nume.out*.

Modificările făcute codului JavaNLP au soluționat probleme apărute și au făcut posibilă funcționarea corectă a serviciului așa cum se poate observa și în urma anlizării figurilor etichetate Figura 3.9 și Figura 3.10.

3.6 SITUAȚII ÎN CARE NU SE REALIZEAZĂ RESTAURAREA DIACRITICELOR

O astfel de situație este cea a alăturărilor de forma cuvânt1-cuvânt2, precum om-mașină, ce nu se regăsesc în Dicționarul explicativ al limbii române. Acestea nu sunt restaurate datorită faptului că operația de „curățare” efectuată cu ajutorul metodelor puse la dispoziție de JavaNLP nu înlătură cratima dintr-o astfel de structură pentru a nu se ajunge în situația în care alăturări de forma într-o, dintr-o, să-mi, ce se regăsesc în dicționar, să nu fie restaurate.

De asemenea, în cazul în care fişierul .docx introdus de utilizator conţine un tabel mai complex , ce prezintă într-o celulă un alt tabel, textul tabelului din celula respectivă nu va fi restaurant deoarece, în momentul proiectării serviciului web s-a considerat faptul că o celulă a unui tabel va conţine doar text şi nu s-au luat în considerare astfel de cazuri. Pe viitor, se doreşte însă modificarea serviciului astfel încât restaurarea diacriticelor să se poată face şi în cazul unor astfel de fişiere .docx.

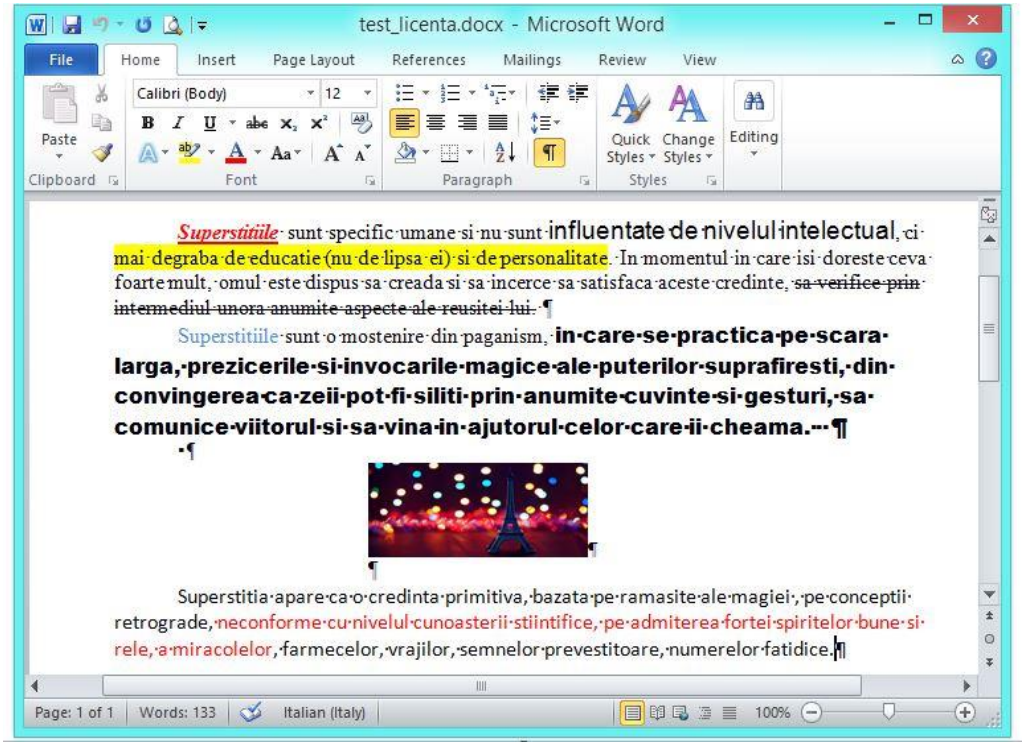

Figura 3.9 Fişierul .docx fără diacritice

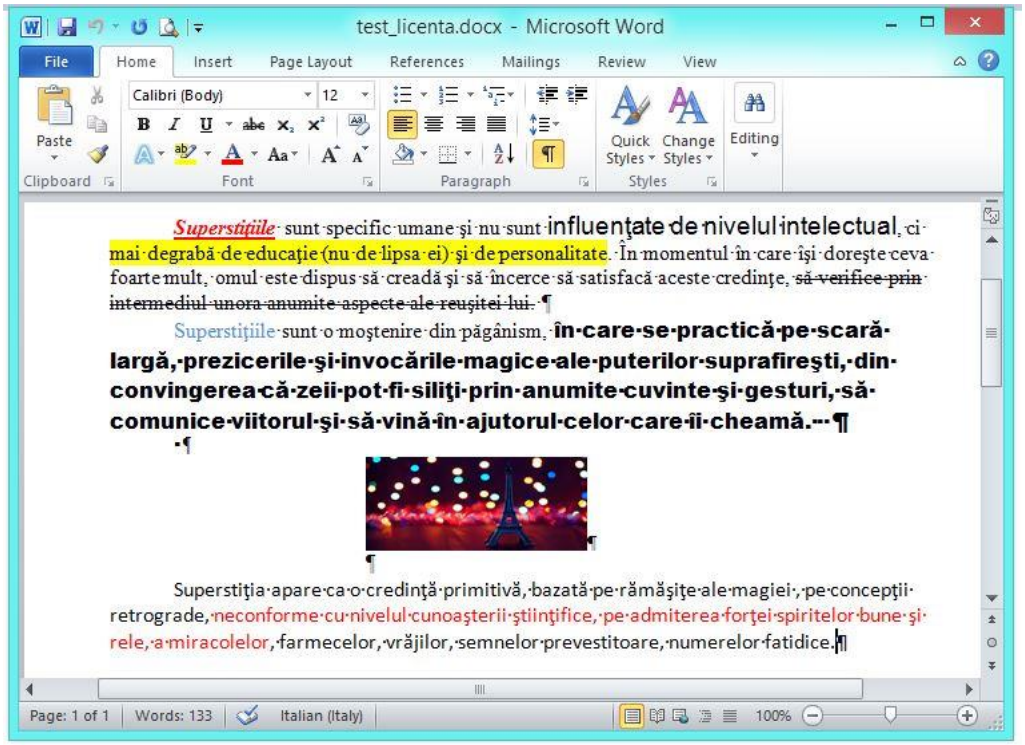

Figura 3.10 Fişierul .docx restaurat

3.7 STUDIU DURATĂ INIȚIALIZARE DISAMBIG

Deși serviciul web prezentat în acest capitol ușurează munca utilizatorului deoarece permite restaurarea automată a diacriticelor pentru fișierele .docx, acesta prezintă un dezavantaj important și anume: timpul necesar restaurării documentului oferit de utilizator este de ordinul minutelor și nu de ordinul secundelor, așa cum ar fi de dorit.

În lucrarea de față, restaurarea diacriticelor, ce presupune transformarea unui flux de cuvinte fără diacritice într-un flux de cuvinte cu diacritice, este realizată de către instrumentul disambig cu ajutorul modelului de limbă și a hărții cuvintelor. Acestea, la rândul lor au fost construite tot cu ajutorul acestui instrument respectiv cu ajutorul unei aplicații Java și sunt reprezentate de două fișiere .txt, ambele de dimensiune mare. Pentru ca disambig să poată genera textul cu diacritice, este necesar ca acesta să încarce cele două fișiere text amintite mai sus: harta cuvintelor și modelul de limbă. Având în vedere dimensiunea destul de mare a fiecărui fișier în parte, s-a emis ipoteza că durata mare de restaurare a documentului Word se datorează, în parte, și faptului că încărcarea acestor două fișiere consumă și ea destul de mult timp. De fapt, s-a presupus faptul că încărcarea celor două fișiere durează mai mult ca restaurarea în sine. Pentru a se verifica dacă ipoteza considerată este sau nu adevărată, s-a luat decizia de a se realiza un studiu care să pună în evidență durata de inițializare asociată instrumentului disambig.

Astfel, pentru a se realiza studiul de față, a fost necesară colectarea mai multor texte, aparținând unor domenii diferite, precum cel al protecției muncii, cel jurnalistic sau cel medical. După efectuarea primului pas, ce a constat în adunarea textului destinat realizării experimentului prezentat în această secțiune, a urmat realizarea propriu-zisă a exeprimentului, comanda folosită pentru restaurarea diacriticelor având următoarea structură: *disambig –lm calea/model_de_limbă -order 3 –map calea/harta_cuvintelor -text calea/fișier_text_intrare –keep-unk*. Trebuie precizat faptul că testele au fost făcute pe server-ul pe care este instalat serviciul și că mai multe detalii referitoare la opțiunile folosite în cadrul comenzii anteriorare sunt oferite în secțiunea 2.4.2 a acestei lucrări.

Așadar, exeprimentul a constat în restaurarea pe rând a unui număr din ce în ce mai mare de cuvinte, astfel încât să se poată determina dacă ipoteza considerată este sau nu adevărată. Rezultatele acestuia sunt prezentate în tabelul etichetat Tabel 3.1. În urma analizării acestui tabel se poate observa că numărul minim de cuvinte folosit a fost unu. Scopul includerii acestui caz a fost acela de a determina timpul necesar încărcării modelului de limbă și a hărții cuvintelor, care în urma realizării restaurării s-a dovedit a fi egal cu 25.8 secunde. S-a trecut apoi la restaurarea diacriticelor pentru un număr tot mai mare de cuvinte, până când s-a ajuns la un număr egal cu 53.000 și s-a observat că diferența între timpul obținut în acest caz și cel obținut în cazul restaurării unui singur cuvânt este de doar 6.7 secunde. Trebuie precizat faptul că textul de 53.000 de cuvinte a fost extras dintr-un document Word de 110 pagini ce nu conținea tabele sau imagini. De altfel, toate textele au fost extrase din documente Word diferite ce nu conțineau imagini. Analizând și celelalte rezultate se poate observa faptul că timpul necesar restaurării propriu-zise a unui document de 18.400 de cuvinte sau 4.700 de cuvinte este de 4.6 respectiv 2.3 secunde. Se remarcă astfel că, pe măsură ce numărul de cuvinte crește rapid, creșterea timpului necesar restaurării este lentă, așa cum se poate observa și în Figura 3.11. Astfel pentru a se ajunge la un timp asociat restaurării propriu-zise egal cu 25.8 secunde, cel mai probabil va fi necesar un document Word de patru, cinci ori mai mare sau chiar mult mai mare decât cel de 110 pagini. Cum, în general, un document redactat de un utilizator obișnuit nu are atât de multe pagini, de fapt, nu depășește 150 – 200 pagini (luând în considerare cazul extrem: lucrare de dizertație, de doctorat, etc), se poate afirma că ipoteza considerată este validă.

Având în vedere rezultatele prezentate, se poate considera că pentru restaurarea textului paragrafelor documentului oferit de utilizator este necesar un timp mai mare de 30 de secunde. Cum de cele mai multe ori un fișier .docx conține și tabele și ținând cont de modul de funcționare prezentat pe scurt în Figura 3.7, va mai fi necesar un alt timp de peste 30 de secunde pentru restaurarea textului asociat tabelelor. Prin urmare, doar pentru restaurarea textului din paragrafe și tabele este necesar mai mult de un minut. Lând în considerare și restul operațiilor prezentate în Figura 3.7, ce necesită și ele un anumit număr de secunde, timpul scurs de la alegerea fișierului .docx și până la momentul apariției mesajului ce sugerează faptul că fișierul a fost restaurat ajunge să fie de câteva minute. Prin urmare, se impune ca, pe viitor, să se găsească o modalitate de micșorare a acestui timp.

Nr. crt.	Număr cuvinte	Timpul necesar restaurării (s)
1	1	25.8
2	600	26.3
3	1400	26.9
4	2500	27.4
5	4700	28.1
6	6600	28.4
7	9600	28.9
8	18400	30.4
9	53000	32.5

Tabel 3.1 Rezultatele experimentului

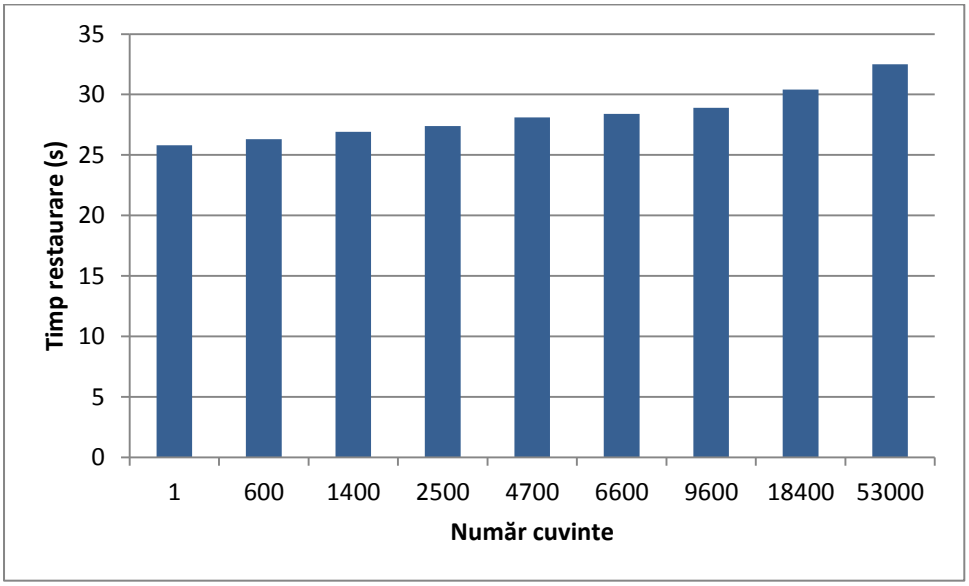

Figura 3.11 Evoluția timpului necesar restaurării

CAPITOLUL 4

APLICAȚIA ANDROID

4.1 DESCRIERE GENERALĂ

Smartphone-urile au devenit nelipsite din viața omului modern deoarece, pe lângă avantajul dimensiunii reduse, prezintă și avantajul deținerii unei puteri de procesare asemănătoare cu cea a desktop-urilor. Acest lucru a făcut posibilă utilizarea telefoanelor mobile moderne pentru îndeplinirea unor acțivități care, până nu demult, necesitau utilizarea unui desktop sau a unui laptop. Astfel, smartphone-urile au ajuns să fie folosite pentru navigarea pe Internet, trimiterea e-mail-urilor, redactarea documentelor, etc. Având în vedere acest lucru, se preconizează faptul că în anul 2017 numărul utilizatorilor de telefoane mobile, la nivel global, va ajunge la 5.1 miliarde.

La ora actuală, marea majoritate a smartphone-urilor folosesc ca și sistem de operare Android. Android este un sistem de operare open source ce se bazează pe nucleul Linux, fiind destinat telefoanelor mobile sau tabletelor cu touchscreen. Aplicațiile destinate telefoanelor mobile ce folosesc acest sistem de operare sunt scrise în limbaj Java, astfel orice persoană familiarizată cu acest limbaj de programare își poate concepe propria aplicație Android.

Având în vedere faptul că serviciul web descris în capitolul trei a fost realizat folosind, în mare parte, limbajul de programare Java și ținând cont de ascensiunea tot mai rapidă a smatphone-urilor, s-a dorit dezvoltarea unei aplicații Android care să-i ofere posibilitatea utilizatorului de a restaura diacriticele pentru un text pe care urmează să îl trimită prin e-mail. Aplicația pezentată în acest capitol a fost testată atât pe un dispozitiv virtual cât și pe unul real, îndeplinindu-și funcțiile în ambele cazuri. Figurile ce vor fi prezentate în continuare vor demonstra funcționalitatea aplicației pe dispozitivul virtual.

Trebuie menționat faptul că aplicația comunică cu un serviciu web existent, el fiind cel care realizează practic restaurarea diacriticelor, oferindu-i textul restaurat aplicației care pur și simplu îl afișează. Serviciul web la care se face referire este cel dezvoltat de Ana-Maria Vladu în lucrarea „Serviciu web de restaurare a diacriticelor”. Având în vedere cele menționate, schema bloc a sistemului de restaurare a diacriticelor va fi cea prezentată în figura următoare.

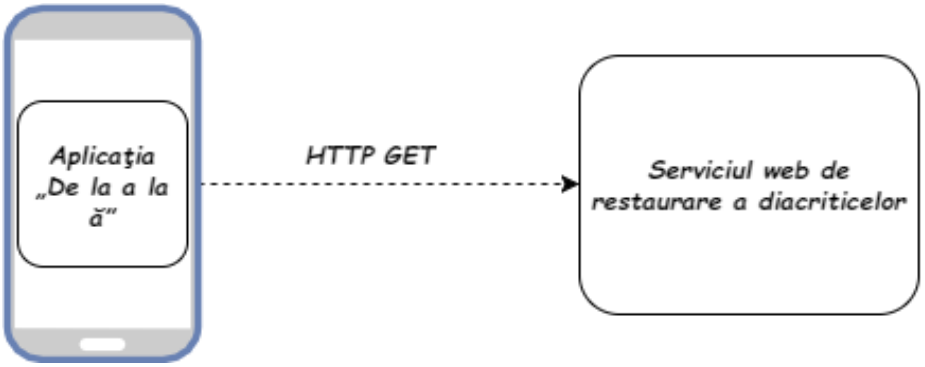

Figura 4.1 Schema bloc a sistemului de restaurare a diacriticelor

Detaliile amănunțite referitoare la funcționarea serviciului web de restaurare a diacriticelor sunt prezentate în lucrarea scrisă de Ana-Maria Vladu, deși câteva informații referitoare la modul de restaurare a textului din fișiere .txt au fost oferite și în secțiunea 3.2 a acestei lucrări. Principalul dezavantaj al serviciului dezvoltat de către Ana-Maria Vladu este reprezentat de faptul că liniile goale și spațiile de la începutul sau sfârșitul unei propoziții sunt eliminate. Cum textul introdus de către utilizator cu ajutorul tastaturii virtuale este destinat transmiterii prin e-mail, este posibil ca acesta să conțină linii și spații goale. Pentru a nu impune utilizatorului restricții referitoare la modul în care trebuie scris textul ce urmează a fi restaurant, s-au făcut anumite modificări prezentate în secțiunea 3.5, îmbunătățindu-se astfel serviciul web existent.

Figura 4.2 Interfața aplicației

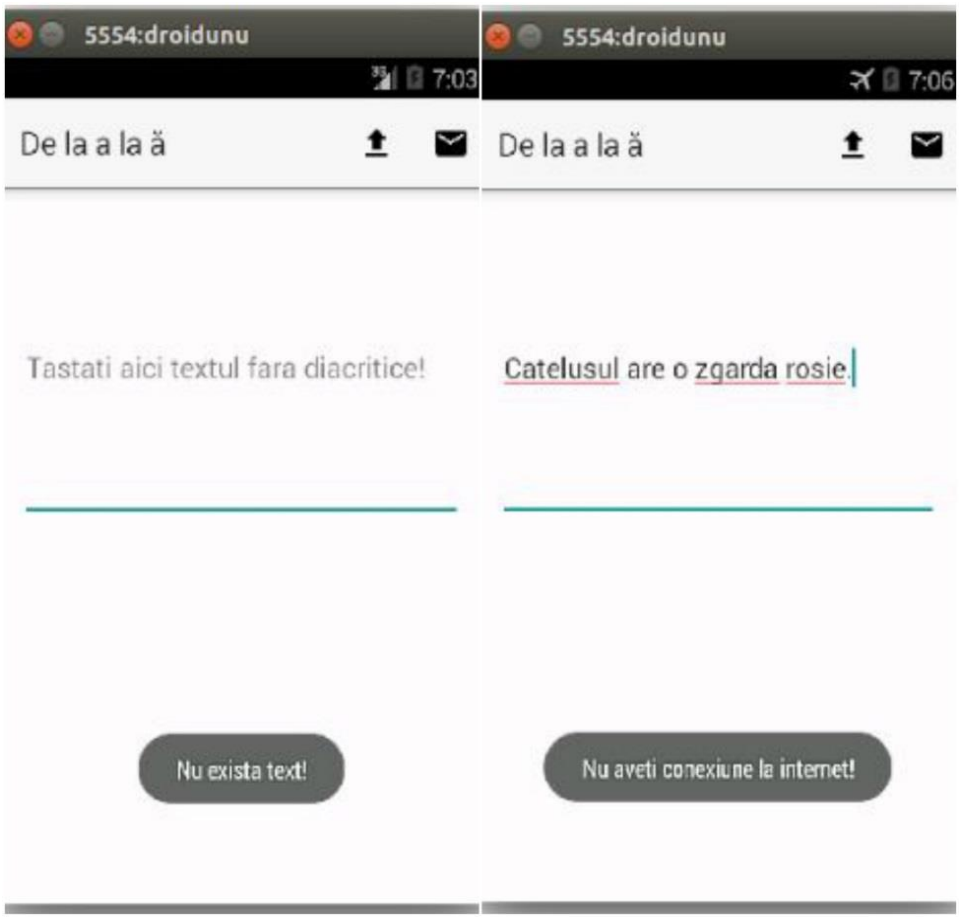

Figura 4.3 Mesajele de informare asupra motivelor pentru care procesul de restaurare nu poate fi pornit sau pentru care nu se poate trece la etapa următoare

În continuare se va face o descriere succintă a aplicației, fără a se intra prea mult în detaliu asupra modului în care se realizează restaurarea. Astfel, în momentul în care utilizatorul deschide aplicația, acesta va fi întâmpinat de o interfață simplă precum cea prezentată în Figura 4.2. Pentru a putea porni restaurarea diacriticelor, utilizatorul va fi nevoit să introducă textul dorit în secțiunea corespunzătoare și apoi să aleagă opțiunea destinată pornirii restaurării din meniul aplicației, cea marcată printr-o săgeată neagră în figura menționată anterior. Dacă utilizatorul nu introduce un text sau telefonul mobil nu este conectat la Internet și se încearcă trimiterea textului spre serviciul web, aplicația va afișa mesaje de informare corespunzătoare fiecărei situații menționate, prin care anunță cauza pentru care procesul de restaurare nu a putut fi pornit. Astfel, în cazul în care nu a fost introdus un text în secțiunea corespunzătoare, utilizatorul va fi informat de acest lucru prin intermediul mesajului „Nu exista text!”. Dacă, în schimb, nu este stabilită conexiunea la Internet, mesajul de informare afișat este „Nu aveti conexiune la Internet!”. Scopul acestor mesaje, prezentate în Figura 4.3, este acela de a avertiza utilizatorul asupra problemelor apărute, astfel încât să se facă modificările necesare pornirii procesului de restaurare.

În cazul în care niciuna dintre situațiile menționate anterior nu își face apariția, procesul de restaurare este pornit, utilizatorul fiind înștiințat asupra faptului că acesta este în plină desfășurare așa cum arată și Figura 4.4.

După finalizarea restaurării, aplicația va obține textul cu diacritice de la serviciul web și îl va afișa într-o secțiune separată de cea destinată textului inițial, după cum arată și Figura 4.4. Se va putea trece apoi la etapa următoare, de trimitere a e-mail-ului, prin selectarea din meniul aplicației a opțiunii corespunzătoare, prezentată și ea în Figura 4.4.

Figura 4.4 Desfășurarea și finalizarea procesului de restaurare

Figura 4.5 Rezultatul apăsării butonului „Trimite mail!"

În urma selectării opțiunii „Trimitere mail!", utilizatorului i se va oferi posibilitatea de a alege aplicația de mail pe care dorește să o utilizeze pentru a trimite e-mail-ul. După ce alegerea este făcută, tot ce va mai trebui să facă acesta va fi să completeze câmpurile corespunzătoare

destinatarului şi subiectului mesajului şi să expedieze apoi mail-ul, aşa cum se poate observa şi prin analizarea figurii de mai sus. De asemenea, se mai poate observa faptul că, în urma selectării opțiunii, textul mesajului a fost completat automat.

4.2 COMPONENTELE UNEI APLICAȚII ANDROID

Printre componentele unei aplicații Android se numără:

- *Activitatea* – asigură funcțiile asociate unei ferestre a aplicației. Având în vedere că o aplicație Android poate avea mai multe ferestre, ea va prezenta, implicit, mai multe activități. De exemplu, o aplicație de poștă electronică poate avea o activitate care să permită afișarea mesajelor e-mail recepționate, o activitate destinată compunerii unui mesaj de tip e-mail sau o activitate ce face posibilă citirea e-mail-urilor. Trebuie menționat faptul că, deși aceste activități cooperează pentru ca aplicația de poștă electronică să poată funcționa corect, ele sunt independente. Prin urmare, o altă aplicație va putea porni o activitate a aplicației de mail dacă aceasta din urmă permite acest lucru, fără a porni și celelalte activități[42]. Activitățile unei aplicații au asociată o stivă a activităților. Astfel, atunci când este accesată a nouă activitate, ea este plasată în vârful stivei, devenind, în acest fel, activitatea ce va fi rulată de către aplicație. Activitatea anterioară va coborâ în stivă, revenind în prim-plan abia după părăsirea noii activități[43]. Se poate afirma faptul că o activitate are patru stări[43]: dacă se află în prim-plan, cu alte cuvinte, dacă este în vârful stivei, atunci activitarea este în starea *running*; dacă fereastra activității nu mai este în prim-plan, însă rămâne vizibilă (în prim-plan se află o activitate cu o fereastră mai mică), atunci activitatea intră în starea de *aşteptare*; dacă fereastra activității este complet acoperită de fereastra unei alte activități atunci activitatea este *întreruptă*; dacă activitatea se află în așteptare sau este întreruptă, sistemul îi poate cere să-și termine prelucrările sau o poate *distruge* pentru a elibera memoria, astfel, atunci când fereastra sa este afișată din nou utilizatorului, activitatea va fi restartată sau se va restaura starea anterioară a acesteia.
- *Serviciile* – rulează în fundal şi realizează operații de lungă durată. Astfel, un serviciu poate colecta informații de pe Internet, fără ca interacțiunea unui utilizator cu o activitate să fie blocată[44].
- *Receptorii de mesaje sau Broadcast Receivers* – răspund la mesajele difuzate de alte aplicații sau de către sistem. De exemplu, o aplicație poate difuza mesaje pentru a anunța alte aplicații de faptul că anumite date au fost descărcate și sunt gata de utilizare, receptorul de mesaje fiind cel care interceptează mesajul și inițiază măsurile corespunzătoare[44].
- *Furnizorii de conținut* – așa cum le spune și numele, transmit datele asociate unei aplicații către alte aplicații, care au trimis, în prealabil, o cerere în acest sens[44].
- *Intent* – un Intent este un mesaj asincron prin care sunt activate anumite funcții ale unei componente aparținând aplicației respective sau altei aplicații. Practic, un astfel de mesaj activează următoarele componente: serviciile, receptorii de mesaje și activitățile[42].
- *Fragmentele* – reprezintă o porțiune a interfeței unei activități[43].
- *Resursele* - sunt separate de codul sursă, fiind reprezentate de imagini, fișiere audio și de orice element ce ajută la proiectarea unei interfețe atractive a aplicației. Ele se definesc cu ajutorul fișierelor XML. Separarea acestora de codul sursă permite actualizarea diferitelor caracteristici ale aplicației fără modificarea codului dar și pobilitatea oferirii de resurse alternative corespunzătoare unor diferite configurații de dispozitive. De exemplu, se pot specifica mai multe imagini, de diferite dimensiuni,

pentru dispozitive cu diferite dimensiuni ale ecranului. Fiecare resursă are asociat un ID unic ce este folosit pentru referirea resursei din codul sursă sau din alte fișiere XML. Astfel, unei imagini numite diacritice.jpeg (stocată în fișierul res/drawable) i se asociază ID-ul R.drawable.diacritice[42].

- *Fișierul Manifest* – orice aplicație Android prezintă fișierul AndroidManifest.xml. Acesta conține toate componentele aplicației. Pe lângă faptul că este folosit pentru declararea componentelor, el mai este folosit pentru următoarele lucruri: specificarea permisiunilor, cum ar fi cele necesare accesului la Internet, citirii contactelor utilizatorului; declararea nivelului API minim necesar aplicației; declararea caracteristicilor hardware și software utilizate sau cerute de către aplicație, precum serviciile bluetooth, camera foto, etc; specificarea unor biblioteci necesare aplicației cum ar fi Google Maps[42].

4.3 IMPLEMENTAREA APLICAȚIEI

Interfața grafică reprezintă unul dintre elementele cheie ale aplicației Android, fiind, în general, construită folosind o ierarhie de obiecte de tipul View și ViewGroup.

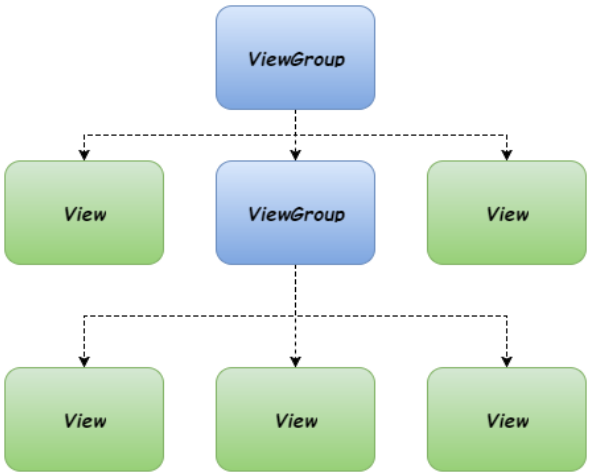

Figura 4.6 Ierarhie de obiecte de tipul View sau ViewGroup, Sursa [45]

Un obiect de tipul View poate fi reprezentat de un buton, o imagine, un câmp de text, iar unul de tipul ViewGroup nu reprezintă altceva decât un container invizibil ce definește modul în care elementele componente sunt dispuse. De exemplu, acestea pot fi dispuse sub forma unei liste verticale sau sub forma unui ansamblu de linii și coloane (grid)[45].

Implementarea interfeței grafice se face cu ajutorul limbajului XML, acesta fiind potrivit pentru crearea unei ierarhii de elemente. În ceea ce privește aplicația prezentată în acest capitol, interfața asociată prezintă un meniu și o porțiune destinată introducerii textului dorit și afișării rezultatului obținut în urma restaurării, așa cum se poate observa și în Figura 4.2.

Meniul conține două opțiuni, una este folosită pentru pornirea procesului de restaurare sau, mai exact, pentru trimiterea textului utilizatorului către serviciul de restaurare a diacriticelor și cealaltă este folosită pentru accesarea uneia dintre aplicațiile de mail existente pe dispozitiv-ul pe care este instalată aplicația Android, numită sugestiv „De la a la ă”. Având în vedere faptul că pentru proiectarea interfeței se folosește limbajul XML, opțiuile amintite anterior sunt definite într-un fișier .xml care, în cazul de față, se numește start.xml. Fiecare opțiune va avea asociat un ID, o iconiță corespunzătoare și un text folosit pentru a explica pe scurt rolul opțiunii, asocierile fiind posibile datorită folosirii atributelor android:id, android:icon respectiv android:title, așa cum se poate observa și în urma analizării secvenței următoare:

```
<item
    android:id="@+id/send_text"
    android:icon="@drawable/ic_file_upload_black_24dp"
    android:title="@string/send_text"
    app:showAsAction="ifRoom"/>
```

Se remarcă, de asemenea, folosirea unui alt atribut, și anume: app:showAsAction, ce definește modul în care elementul (opțiunea) va apărea în cadrul meniului. În cazul de față, valoarea acestuia este ifRoom, sugerând faptul că opțiunea va fi afișată direct pe bara de meniu dacă va fi suficient loc, fiind astfel vizibilă fără a fi necesară acesarea opțiunii „Options” a cărui iconiță este prezentată în figura următoare.

Figura 4.7 Iconița „Options

Secvența de cod prezentată anterior este asociată opțiunii de trimitere a textului către serviciul web de restaurare a diacriticelor și are asociat ID-ul send_text și textul „Trimite text!” ce este definit în fișierul strings.xml și este referit prin numele send_text.

În ceea ce privește porțiunea destinată introducerii textului dorit și afișării rezultatului obținut în urma restaurării, ierarhia folosită pentru implementarea acesteia este prezentată în figura următoare.

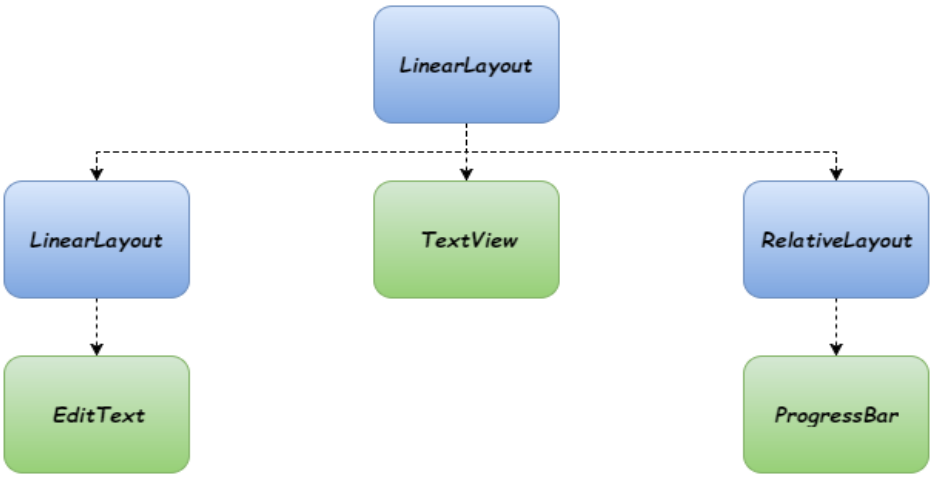

Figura 4.8 Ierarhia porțiunii destinate introducerii textului dorit și afișării rezultatului obținut

În Figura 4.8, LinearLayout și RelativeLayout sunt obiecte de tipul ViewGroup, motiv pentru care ele au fost reprezentate cu albastru, iar EditText, TextView și ProgressBar sunt obiecte de tipul View, motiv pentru care au fost reprezentate cu verde. LinearLayout permite plasarea elementelor componente, de tipul View, fie pe o direcție verticală, fie pe una orizontală, în funcție de valoarea atributului android:orientation. Astfel, dacă android:orientation va avea valoarea “vertical”, direcția de plasare a elementelor va fi una verticală, în schimb, dacă acesta va avea valoarea “horizontal”, direcția de plasare va fi una orizontală. Trebuie precizat faptul că elementele unui LinearLayout vor apărea pe ecranul dispozitivului utilizatorului în ordinea în care au fost definite în fișierul .xml, în cazul de față, în fișierul activity_start.xml. RelativeLayout permite plasarea unui element relativ la alte componente sau chiar la el însuși. Atunci când un element este plasat relativ la un alt element component, el poate fi poziționat la dreapta sau la stânga elementului considerat drept referință. În schimb, dacă el este plasat

relativ la container-ul RelativeLayout, el poate fi poziționat în partea de sus, în partea de jos sau în partea centrală a container-ului. Acestea fiind spuse, se observă că un element EditText, ce-i permite utilizatorului să introducă textul dorit, este plasat într-un container de tipul LinearLayout. Secvența de cod asociată fiind următoarea:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >
    <EditText
        android:id="@+id/edit_text"
        android:layout_width="match-parent"
        android:layout_height="wrap_content"
        android:scrollbars="vertical"
        android:hint="@string/first_text"
        android:lines="8" />
</LinearLayout>
```

S-a ales ca elementul LinearLayout să fie la fel de lat ca și ecranul dispozitivului pe care rulează aplicația, prin atribuirea valorii "match_parent" atributului android:layout_width, și ca acesta să aibă înălțimea minima necesară pentru afișarea conținutului său, prin atribuirea valorii "wrap_content" atributului android:layout_height. După cum se poate observa, aceleași alegeri au fost făcute și pentru elementul EditText. Pentru ca utilizatorul să-și poată da seama unde trebuie introdus textul dorit, fără diacritice, secțiunea corespunzătoare trebuie marcată. Acest lucru a fost făcut prin afișarea unui text implicit în secțiunea corespunzătoare, atunci când niciun text nu a fost încă introdus. Textul implicit ales a fost: „Tastati aici textul fara diacritice!", fiind definit în fișierul strings.xml și apoi referit pentru ca atributul android:hint să aibă asociat textul implicit. După cum se poate observa, numărul de rânduri ocupate de textul implicit este egal cu unu. Astfel, din motive pur estetice, s-a ales ca numărul de rânduri alocate elementului EditText să fie egal cu opt, ajungându-se la forma prezentată în Figura 4.2. După stabilirea acestui număr de rânduri, s-a observat faptul că la introducerea unui număr mai mare de rânduri, primele linii ale textului nu mai sunt vizibile. Prin urmare, a fost identificată necesitatea introducerii unui scroll bar care să permită vizualizarea liniilor care altfel nu erau vizibile. Astfel, s-a folosit atributul android:scrollbars căruia i s-a atribuit valoarea "vertical".

Elementul TextView este folosit pentru afișarea textului restaurant, primit de la serviciul web, și va avea aceleași valori pentru atributele android:layout_width, android:layout_height și android:scrollbars ca cele stabilite pentru elementul EditText.

Pentru ca utilizatorul să poată fi înștiințat asupra faptului că procesul de restaurare este în curs de desfășurare, se va folosi o bară de progres. Atunci când aceasta va fi vizibilă, ea va fi plasată în centrul container-ului RelativeLayout, lucru obținut prin atribuirea valorii "true" ambelor atribute: android:layout_centerHorizontal și android:layout_centerVertical. Trebuie menționat faptul că atributele android:layout_width și android:layout_height, pentru elementul ProgressBar, au ambele valoarea wrap_content iar pentru elementul RelativeLayout au valoarea match_parent respectiv wrap_content. Aceste lucruri pot fi observate și prin analizarea următoarei secvențe de cod:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <ProgressBar
        android:id="@+id/progressBar1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true" />
</RelativeLayout>
```

Până în acest punct au fost prezentate detalii referitoare la modul de proiectare a interfeței aplicației Android, nefiind încă prezentate detaliile referitoare la modul de proiectare a funcțiilor aplicației. Astfel, în cele ce urmează vor fi prezentate informații referitoare la modul de implementare a funcțiilor aplicației „De la a la ă”, codul sursă al acestora fiind prezentat în Anexa 2.

Cea mai importantă opțiune din meniul aplicației este „Trimite textul!”, deoarece în urma selectării acesteia se va trimite o cerere HTTP GET către serviciul web de restaurare a diacriticelor, însă doar dacă utilizatorul a introdus un text în câmpul corespunzător și dispozitivul pe care rulează aplicația este conectat la Internet. Serviciul web la care se conectează aplicația este unul RESTful, ceea ce înseamnă că pornirea procesului de restaurare poate fi făcută prin simpla accesare a unui URL ce prezintă forma următoare: http://ivan.studenti.speed.pub.ro/RestaurareDiacritice/webresources/androiddiacriticsrestorer/process?inputText=textul_dorit, în care textul_dorit reprezintă textul fără diacritice pe care utilizatorul dorește să-l restaureze. Având în vedere această formă a URL-ului, informațiile asupra cererii HTTP, ce va fi trimisă serviciului web, vor fi stabilite cu ajutorul funcției următoare:

```
private void requestData(String uri) {
    EditText editText = (EditText) findViewById(R.id.edit_text);
    //textul din EditText este asociat variabilei message
    String message = editText.getText().toString();
    RequestToWebService r=new RequestToWebService();
    r.setMethod("GET");
    r.setUri(uri);
    r.setParameters("inputText", message);
    BackgroundTask task = new BackgroundTask();
    task.execute(r);
}
```

Analizând liniile de cod prezentate mai sus, se poate observa faptul că se folosesc trei metode ale obiectului r de tipul RequestToWebService: setMethod – pentru specificarea faptului că cererea HTTP este una de tipul GET, setUri – pentru specificarea următorului URL:

http://ivan.studenti.speed.pub.ro/RestaurareDiacritice/webresources/androiddiacriticsrestorer/ process şi setParameters – pentru specificarea parametrului de tip Query ce urmează a fi trimis servciului web de restaurare a diacriticelor: ?inputText=textul_dorit. A fost necesară conceperea metodei setParameters datorită faptului că textul_dorit nu va fi niciodată acelaşi, fiecare utilizator introducând propriul text.

Odată ce sunt stabilite detaliile asupra cererii HTTP, aceasta va trebui să fie trimisă serviciului web iar utilizatorul va trebui să fie informat cu ajutorul unui progress bar de faptul că procesul de restaurare a diacriticelor este în curs de desfăşurare. Trebuie precizat faptul că orice aplicaţie Android prezintă un fir de execuţie principal care controlează elementele interfeţei grafice a aplicaţiei şi care nu permite efectuarea mai multor operaţii în acelaşi timp. Cu alte cuvinte, pentru a efectua o altă operaţie, precum introducerea unui text nou în secţiunea corespunzătoare, este necesară aşteptarea terminării efectuării operaţiei precedente, cum ar fi obţinerea textului restaurat. De cele mai multe ori există posibilitatea ca timpul de efectuare a unei operaţii să fie de câteva secunde, aşa cum se întâmplă în cazul trimiterii unei cereri HTTP, consecinţa fiind faptul că utilizatorul nu va putea accesa alte funcţii ale aplicaţiei. Practic, aplicaţia va fi blocată în tot acest timp. Astfel, pentru a se evita acest lucru în cazul aplicaţiei „De la a la ă”, singura activitate a acesteia, numită Start, va defini o clasă membru ce va moşteni metodele clasei AsyncTask. Această clasă va fi numită BackgroundTask, nume ce sugerează faptul că funcţiile asociate vor fi executate în fundal, însă având în vedere că este stabilită o relaţie de moştenire cu clasa AsyncTask, rezultatele oferite de unele funcţii vor putea fi afişate utilizatorului doarece clasa AsyncTask prezintă metode de accesare a firului principal de execuţie.

Clasa BackgroundTask, ce prezintă metode folosite pentru trimiterea cererii HTTP GET, afişarea barei de progres şi pentru obţinerea textului restaurat, a fost definită în interiorul activităţii Start, în felul următor:

```
private class BackgroundTask extends AsyncTask<RequestToWebService, String, String> {
    @Override
    protected void onPreExecute() {
    output = (TextView) findViewById(R.id.textView);
    output.setText("");
    pbar.setVisibility(View.VISIBLE);
    }
    @Override
    protected String doInBackground(RequestToWebService... params) {
    String content= HttpManager.getData(params[0]);
    return content;
    }
    @Override
    protected void onPostExecute(String result) {
    updateDisplay(result);
    pbar.setVisibility(View.INVISIBLE);
    }
```

}

Analizând liniile de cod de mai sus se observă faptul că AsyncTask are asociate trei tipuri generice: primul reprezintă tipul parametrilor folosiți de operația ce se efectuează în fundal, al doilea reprezintă tipul datelor afișate în timpul efectuării operației iar al treilea reprezintă tipul rezultatului oferit de operația efectuată în fundal[46]. În ceea ce privește metoda onPreExecute aceasta este invocată înaintea începerii efectuării operației în fundal și are drept scop afișarea progress bar-ului și eliminarea textului din secțiunea destinată afișării textului restaurat. Ștergerea textului este realizată deoarece există posibilitatea ca utilizatorul să dorească să transmită prin e-mail un alt text care să nu conțină textul deja existent în acea secțiune. Pentru efectuarea operației din fundal se folosește metoda doInBackground ce are drept parametru un obiect de tipul RequestToWebService. Revenind la metoda doInBackground, aceasta definește un obiect de tipul String ce va conține textul cu diacritice oferit de către serviciul web. Pentru ca utilizatorul să poată vizualiza textul restaurat se va folosi metoda onPostExecute ce va primi ca parametru rezultatul oferit de metoda doInBackground, pe care îl va afișa. De asemenea, metoda onPostExecute va face invizibilă bara de progres. Se remarcă, în plus, folosirea în cadrul metodei doInBackgroung a metodei getData aparținând clasei HttpManager. Această funcție ajută la obținerea URL-ului final, folosit de către aplicația Android prezentată în această lucrare pentru pornirea procesului de restaurare a diacriticelor. URL-ul menționat este (textul_dorit fiind cel al utilizatorului) : http://ivan.studenti.speed.pub.ro/RestaurareDiacritice/webresources/androiddiacriticsrestorer/process?inputText=textul_dorit. Mai mult, funcția transmite cererea HTTP GET și citește corpul răspunsului oferit de serviciul web, obținând astfel textul cu diacritice. Trebuie amintit faptul că un răspuns la o cerere HTTP GET este format din mai multe antete și un corp ce conține informațiile solicitate prin intermediul cererii HTTP GET.

Odată ce textul cu diacritice este afișat în secțiunea corespunzătoare, acesta va putea fi trimis prin e-mail. Pentru a se realiza acest lucru, utilizatorul va trebui să aleagă opțiunea „Trimite mail!” din meniul aplicației. În urma alegerii acestei opțiuni se va verifica dacă dispozitivul este conectat la Internet și dacă există text în secțiunea destinată textului restaurat. În cazul în care una dintre verificări nu este realizată cu succes, de exemplu dispozitivul nu este connectat la Internet se va crea un Toast pentru a se afișa un mesaj, în cazul exemplului de față acesta fiind: „Nu aveti conexiune la internet!”. Linia de cod folosită pentru a crea Toast-ul este:

Toast.makeText(this, "Nu aveti conexiune la internet!", Toast.LENGTH_LONG).show();

Pentru accesarea aplicațiilor de mail instalate pe dispozitiv se folosește un Intent astfel:

Intent emailIntent=new Intent(Intent.ACTION_SEND);

emailIntent.putExtra(Intent.EXTRA_TEXT, output.getText());

emailIntent.setType("message/rfc822");

startActivity(Intent.createChooser(emailIntent, "Alege aplicatia de email dorita!"));

4.4 PROBLEMELE APLICAȚIEI ȘI REZOLVAREA LOR

Așa cum s-a precizat și în secțiunea 4.1, aplicația „De la a la ă” este destinată restaurării diacriticelor pentru textul introdus de utilizator și permite introducerea unui alt text în secțiunea corespunzătoare în timpul desfășurării procesului de restaurare, utilizatorul fiind înștiințat de faptul că restaurarea se află în derulare prin intermediul unui progress bar. Dacă după ce a fost pornit procesul de restaurare, utilizatorul introduce un alt text în secțiunea corespunzătoare și alege din nou opțiunea „Trimite text!”, s-a observat că după obținerea

versiunii restaurate a primului text, bara de progres dispărea iar utilizatorul nu mai era înştiinţat de faptul că şi celălalt text era în curs de restaurare. Într-un final, dacă utilizatorul avea suficientă răbdare, acesta putea observa faptul că şi cel de-al doilea text restaurat era afişat. Deşi şi următorul text introdus era restaurat şi afişat, s-a încercat rezolvarea problemei referitoare la dispariţia progress bar-ului, astfel încât utilizatorul să ştie faptul că şi cea de-a doua cerere HTTP către servciul web este procesată. Astfel au fost modificate metodele onPreExecute şi onPostExecute asociate clasei membru BackgroundTask, în cazul cărora tasks este o listă de obiecte de tipul BackgroundTask:

```
protected void onPreExecute() {
                    output = (TextView) findViewById(R.id.textView);
                    output.setText("");
                    if(tasks.size()==0){
                    pbar.setVisibility(View.VISIBLE);
                    }
                    tasks.add(this);
}
protected void onPostExecute(String result) {
                    updateDisplay(result);
                    tasks.remove(this);
                    if(tasks.size()==0){
                    pbar.setVisibility(View.INVISIBLE);
                    }
}
```

Se observă faptul că bara de progres este afişată dacă nicio cerere HTTP GET nu a fost încă trimisă şi eliminată dacă toate cererile sunt procesate. Prin urmare, ea nu va mai fi eliminată imediat după primirea şi procesarea răspunsului asociat primei cereri HTTP, utilizatorul fiind astfel informat de faptul că şi următoarea cerere HTTP este prelucrată.

CONCLUZII

Așa cum a fost precizat și în Introducere, scrierea fără diacritice schimbă sensul unor cuvinte și duce, astfel, la exprimări ambigue și/sau cu un înțeles total diferit. Totuși, la ora actuală, marea majoritate a utilizatorilor, din dorința de a economisi timp, aleg să scrie fără diacritice, bazându-se pe faptul că anumite cuvinte pot fi înțelese dacă se ia în considerare întreg contextul. Această abordare nu mai poate fi utilizată, însă, în cazul redactării unor documente oficiale, deoarece există legi care impun ca acestea să fie redactate corect, cu toate semnele diacritice necesare.

Prin urmare, cu scopul de a veni în ajutorul utilizatorilor care preferă să scrie fără diacritice dar care trebuie să redacteze documente oficiale, au fost elaborate diferite servicii web precum diacritice.com sau diacritice.opa.ro care permit introducerea textului dorit, lipsit de diacritice, oferind în final varianta restaurată a acestuia, cu diacritice. Printre principalele dezavantaje ale acestor servicii se numără faptul că numărul maxim de caractere (și, prin urmare, numărul de cuvinte) care pot fi introduse de către utilizator este limitat dar și de faptul că aceste servicii web nu iau în considerare aspectul textului introdus (culoare, font, dimensiune). Astfel, deși serviciile web amintite au fost proiectate și create cu scopul de a ușura munca utilizatorului, mai mult o vor îngreuna datorită faptului că acesta va fi nevoit să restaureze textul documentului, scris folosind un editor de text precum Microsoft Word, pe porțiuni și apoi să reformateze textul respectiv, ceea ce va reprezenta un consum mai mare de timp față de cel necesar scrierii cu diacritice încă de la început. Pentru a elimina dezavantajele

menționate mai sus, lucrarea de față a prezentat un serviciu web destinat restaurării diacriticelor în cazul fișierelor text complexe, cum sunt cele cu extensia .docx.

Serviciul prezentat îi permite utilizatorului să restaureze în același timp întregul fișier .docx dorit fără ca acesta sa fie nevoit mai apoi să reformateze anumite porțiuni de text. De asemenea, s-a dorit ca acesta să-i trimită utilizatorului rezultatul pe mail dar să-i ofere și posibilitatea de a descărca direct fișierul restaurat. Astfel, în cazul în care utilizatorul accesează serviciul web, de pe un calculator ce nu-i aparține, acesta va putea descărca documentul restaurat și pe calculatorul personal datorită faptului că fișierul rezultat a fost trimis și prin e-mail. De asemenea, serviciul web prezentat în teza de față are asociat o interfață atrăgătoare deoarece s-a dorit ca accentul să nu fie pus numai pe funcțiile oferite ci și pe aspectul paginii web asociate. Detalii referitoare la modul de funcționare și de implemenare al serviciului au fost oferite în capitolul trei.

Deși teza de față s-a concentrat pe modul de procesare al documentelor Word și pe modul de realizarea a unei pagini web dinamice și atrăgătoare, au trebuit studiate și anumite detalii referitoare la modul în care se realizează restaurarea diacriticelor, detalii ce au fost prezentate în capitolul unu.

Având în vedere că smartphone-urile au devenit indispensabile din viața omului modern, aceeași problemă referitoare la scrierea fără diacritice este prezentă și în cazul redactării mesajelor ce urmează a fi trimise pe mail. De cele mai multe ori, un astfel de mesaj trebuie scris corect, cu toate semnele diacritice necesare, deoarece este un mesaj formal, adresat, spre exemplu, unui partener de afaceri, unui profesor, unui director al unei firme, etc. Astfel, pentru a ușura munca utilizatorilor de smartphone-uri care redactează un astfel de mesaj, lucrarea de față a prezentat în capitolul patru o aplicație Android destinată restaurării diacriticelor. Această aplicației nu numai că transformă textul fără diacritice introdus de utilizator într-unul cu diacritice ci și accesează una dintre aplicațiile de poștă electronică instalate pe smartphone, simplificând și mai mult munca utilizatorului.

Astfel, având în vedere cele prezentate mai sus și în Introducere, lucrarea de față și-a atins scopul propus și anume acela de a prezenta un serviciu web de restaurare a diacriticelor destinat fișierelor .docx dar și o aplicație Android de restaurare a diacriticelor.

CONTRIBUȚII PERSONALE

Pornind de la un serviciu web de restaurare de diacritice existent, prezentat de către Ana-Maria Vladu în lucrarea *„Serviciu web de restaurare a diacriticelor”*, autorul a dezvoltat un nou serviciu web, independent de primul, care să permită restaurarea diacriticelor pentru fișierele cu extensia .docx și care să beneficieze de o interfață mult mai atrăgătoare față de cea a serviciului menționat anterior. În plus, pe lângă modalitatea de descărcare directă a fișierului, s-a implementat și o modalitate de a trimite fișierul rezultat în urma restaurării la adresa de e-mail oferită de utilizator.

Serviciul web prezentat în această lucrare a eliminat unele din dezavantajele serviciului deja existent. Astfel, utilizatorul nu mai este nevoit să restaureze textul documentului Word și apoi sa introducă textul în document și să realizeze reformatările necesare, el putând să introducă direct fișierul .docx și să obțină la final fișierul restaurat, cu toate formatările corespunzătoare. Trebuie precizat faptul că fișierul .docx introdus de utilizator poate conține imagini și tabele simple, textul asociat acestora fiind și el restaurat.

O altă contribuție, a fost aceea de a modifica unele din funcțiile bibliotecii JavaNLP astfel încât restaurarea să țină cont de liniile și spațiile goale existente în textul inițial. Așadar, ele nu mai sunt eliminate așa cum se întâmpla în cazul serviciului anterior ci se regăsesc în

fișierul final. Acest lucru reprezintă un mare avantaj deoarece utilizatorul nu va mai fi nevoit să reformateze textul.

Nu în ultimul rând, autorul a mai dezvoltat o aplicației Android, prezentată în capitolul patru, care să-i ofere posibilitatea utilizatorului de a restaura diacriticele pentru un text pe care dorește să-l trimită prin e-mail.

ACTIVITATE ULTERIOARĂ

Principalul dezavantaj al servciului web de restaurare a diacriticelor destinat fișierelor .docx dar și al aplicației Android este reprezentat de faptul că timpul necesar restaurării este destul de mare, așa cum s-a arătat și în secțiunea 3.7. Acest lucru se datorează faptului că cele două fișiere utilizate de instrumentul disambig pentru a realiza restaurarea au o dimensiune mare și, prin urmare, încărcarea lor consumă destul de mult timp. Astfel, una dintre activitățile ulterioare va fi reprezentată de căutarea unei modalități de micșorare a acestui timp.

În ceea ce privește serviciul web, un alt dezavantaj este reprezentat de faptul că restaurarea diacriticelor nu se realizează pentru anumite celule ale unui tabel mai complicat. De exemplu, dacă fișierul .docx prezintă un tabel ce conține într-una dintre celule un alt tabel, textul asociat tabelului din celulă nu va fi restaurat. Acest lucru se întâmplă datorită faptului că nu s-a luat în considerare faptul că într-o celulă a unui tabel se poate regăsi un alt tabel și nu direct text sau o imagine. Prin urmare, următoarea activitate ulterioară este reprezentată de rezolvarea acestei probleme.

Revenind la aplicația Android, aceasta permite trimiterea textului restaurat prin e-mail, însă se dorește ca aceasta să permită trimiterea textului și prin SMS. De asemenea, se dorește ca, pe viitor, aplicația să fie modificată astfel încât să fie destinată și folosirii în cazul în care orientarea ecranului este de tipul „landscape”, ceea ce va presupune și o modificare a interfeței.

BIBLIOGRAFIE

[1]Scrierea cu diacritice: indicator al gradului de cultură şi de profesionalism, http://legestart.ro/scrierea-cu-diacritice-indicator-al-gradului-de-cultura-si-de-profesionalism/ ,accesat la data: 01.06.2016

[2]Cum au evoluat dispozitivele mobile și cum influențează procesele de business, https://www.trust4mobile.ro/cum-au-evoluat-dispozitivele-mobile-si-cum-influenteaza-procesele-de-business/, accesat la data: 02.06.2016

[3]Cucu, H., „*Contribuţii la un sistem de recunoaştere de vorbire continuă, cu vocabular extins, independent de vorbitor pentru limba română”*, Teză de doctorat, pp. 25-29, pp. 70, pp. 72-77, 2011

[4]Cucu, H., Buzo A., Besacier L., Burileanu C., *„SMT-based ASR domain adaptation methods for under-resourced languages: Application to Romanian”*, Speech Communications 56, pp. 5-6, 2014

[5]Mihalcea, R., Năstase, V., *„Letter Level Learning for Language Independent Diacritics Restoration”*, Proceedings of the 6th conference on Natural language learning - Volume 20, pp. 1-7, 2002

[6]Tufiș, D., Ceauşu, A., *„DIAC^{+}: A Professional Diacritics Recovering System”*, Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), pp. 167-174, 2008

[7]Java(Programming language),https://en.wikipedia.org/wiki/Java_(programming_language), accesat la data: 06.06.2016

[8]Java, http://www.webopedia.com/TERM/J/Java.html, accesat la data: 06.06.2016

[9] Programarea orientata pe obiecte in C++:clase, obiecte, functii membru, structuri si uniuni in analogie cu clasele, functii speciale, http://software.ucv.ro/~mburicea/Cap3POO.htm, accesat la data: 06.06.2016

[10]JavaScript-Overview, http://www.tutorialspoint.com/javascript/javascript_overview.htm, accesat la data: 06.06.2016

[11]jQuery-Overview, http://www.tutorialspoint.com/jquery/jquery-overview.htm, accesat la data: 06.06.2016

[12]jQuery Get Started, http://www.w3schools.com/jquery/jquery_get_started.asp, accesat la data: 06.06.2016

[13]jQuery Syntax, http://www.w3schools.com/jquery/jquery_syntax.asp, accesat la data: 06.06.2016

[14]AJAX Introduction, http://www.w3schools.com/ajax/ajax_intro.asp, accesat la data: 06.06.2016

[15]HTML, http://www.webopedia.com/TERM/H/HTML.html, accesat la data: 07.06.2016

[16]What is HTML?, http://www.simplehtmlguide.com/whatishtml.php, accesat la data de: 07.06.2016

[17]HTML, http://www.computerhope.com/jargon/h/html.htm, accesat la data: 07.06.2016

[18]CSS-Introduction, http://www.w3schools.com/css/css_intro.asp, accesat la data: 07.06.2016

[19]Bootstrap 3 Tutorial, http://www.w3schools.com/bootstrap/default.asp, accesat la data: 07.06.2016

[20]RFC2616, *Hypertext Transfer Protocol – HTTP/1.1*, The Internet Society, 1999

[21]HTTP-Overview, http://www.tutorialspoint.com/http/http_overview.htm, accesat la data: 08.07.2016

[22]HTTP requests, http://eloquentjavascript.net/1st_edition/chapter14.html, accesat la data: 08.07.2016

[23]10 Status Code Definitions, https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html, accesat la data: 08.06.2016

[24]RFC821, Simple Mail Transfer Protocol, Information Sciences Institute, University of Southern California, 1982

[25]Mail Submission Agent, https://en.wikipedia.org/wiki/Mail_submission_agent, accesat la data:08.06.2016

[26]What is REST?, http://rest.elkstein.org/, accesat la data: 09.06.2016

[27]Introduction to Representational State Transfer (REST),https://www.exoplatform.com/docs/public/index.jsp?topic=%2FPLF43%2FWS.Introduction.html, accesat la data:09.06.2016

[28]What Are RESTful Web Services?,http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html, accesat la data: 09.06.2016

[29]What is a cookie? Advantages and disadvantages of cookies?,http://www.webcodeexpert.com/2013/03/what-is-cookie-advantages-and.html, accesat la data: 09.06.2016

[30]Apache POI Overview, http://www.tutorialspoint.com/apache_poi/, accesat la data: 09.06.2016

[31]Stolcke, A., *„SRILM – An extensible language modeling toolkit”*, Proceedings International Conference on Spoken Language Processing, pp. 1-4, 2002

[32]Stolcke, A., Disambig, http://www.speech.sri.com/projects/srilm/manpages/disambig.1.html, accesat la data: 10.06.2016

[33]Definition – What does GlassFish mean?, https://www.techopedia.com/definition/27238/glassfish, accesat la data: 10.06.2016

[34]Concept of a GlassFish Domain…,https://blogs.oracle.com/bloggerkedar/entry/concept_of_a_glassfish_domain, accesat la data: 10.06.2016

[35]Flowchart, https://en.wikipedia.org/wiki/Flowchart, accesat la data: 12.06.2016

[36]What is a Flow Chart?, http://www.breezetree.com/articles/what-is-a-flow-chart.htm, accesat la data: 12.06.2016

[37]readyState property, https://msdn.microsoft.com/en-us/library/ms534361(v=vs.85).aspx, accesat la data: 13.06.2016

[38]Frequently asked questions about MIME and content conversion in Exchange 2000 Server and in Exchange Server 2003, https://support.microsoft.com/en-us/kb/836555, accesat la data: 15.06.2016

[39]SSL vs TLS vs STARTTLS, https://www.fastmail.com/help/technical/ssltlsstarttls.html, accesat la data: 15.06.2016

[40]Class Session, http://docs.oracle.com/javaee/6/api/javax/mail/Session.html, accesat la data: 15.06.2016

[41]Uniform Resource Identifier, https://en.wikipedia.org/wiki/Uniform_Resource_Identifier, accesat la data: 15.06.2016

[42]Application Fundamentals, https://developer.android.com/guide/components/fundamentals.html, accesat la data: 16.06.2016

[43]Activity, https://developer.android.com/reference/android/app/Activity.html, accesat la data: 16.06.2016

[44]Android – Application Components, http://www.tutorialspoint.com/android/android_application_components.htm, accesat la data: 16.06.2016

[45]Building a simple user interface, https://developer.android.com/training/basics/firstapp/building-ui.html, accesat la data: 20.06.2016

[46]AsyncTask, https://developer.android.com/reference/android/os/AsyncTask.html, accesat la data: 21.06.2016

ANEXA 1

index.html

```
<html>
    <head>
        <title>Restaurare diacritce</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- Latest compiled and minified CSS -->
        <link                                                                    rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
        <!-- Optional theme -->
        <link   rel="stylesheet"   href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css">
        <!--font awesome -->
        <link    rel="stylesheet"    href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-
awesome.min.css">
        <!--my css file-->
        <link rel="stylesheet" type="text/css" href="Styles.css" />
        <!-- fonts -->
        <link      href="http://fonts.googleapis.com/css?family=Indie      Flower"      rel="stylesheet"
type="text/css">
        <link href="http://fonts.googleapis.com/css?family=Lobster" rel="stylesheet" type="text/css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>
        <!-- Latest compiled and minified JavaScript -->
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
        <!--my script-->
        <script type="text/javascript" src="Script.js"></script>
    </head>
    <body>
        <div class="container-fluid">
            <img src="images/var4.jpg" class="img-responsive">
        </div>
        <div class="container-fluid">
            <div class="well well_design">
                <form class="form-horizontal" id="form" role="form" enctype="multipart/form-data">
                    <div class="form-group">
                        <label    class="control-label    col-sm-3    grid_text_font"    for="email">E-
mail:</label>
                        <div class="col-sm-4">
                            <input type="email" class="form-control" id="email" placeholder=
```

```
                                "Tastati adresa de e-mail..." >
                        </div>
                        <div class="col-sm-1"></div>
                        <div class="col-sm-2">
                            <input            type="file"            name="file"            id="file"
accept="application/vnd.openxmlformats-officedocument.wordprocessingml.document"
class="inputfile_disabled" disabled>
                            <label    class    ="trim_text"    for="file"><i    class="fa    fa-upload
upload_logo"></i><span id="label_span">Alegeti fisierul!</span></label>
                        </div>
                    </div>
                </form>
            </div>
        </div>
        <div class="container">
            <div class="alert alert-info" id="converting" style="display: none;">
                <div class="container-fluid">
                    <div class="row">
                        <div class="col-sm-4 restriction">
                            <strong class="grid_heading_font" id="converting_file"></strong>
                        </div>
                        <div class="col-sm-8">
                            <font color="white" class="converting_info_design">
                            <img src="images/ajax-loader.gif"> In curs de restaurare...</font>
                            <font color="black">S-ar putea sa dureze cateva minute.</font>
                        </div>
                    </div>
                </div>
            </div>
            <div class="alert alert-success" id="converted" style="display: none;">
                <div class="container-fluid">
                    <div class="row">
                        <div class="col-sm-3 restriction">
                            <strong class="grid_heading_font" id="converted_file"></strong>
                            <br>
                            <br>
                            <p class="converted_info_design"><font color="white">Restaurat
                                </font></p>
                        </div>
                        <div class="col-sm-7"></div>
                        <div class="col-sm-2">
                            <br>
                            <div id="download_link"></div>
                        </div>
                    </div>
                </div>
            </div>
```

```
            <div class="alert alert-danger" id="file_error" style="display: none;">
                <a href="#" class="close" data-dismiss="alert" aria-label="close">&times;</a>
                <strong class="grid_heading_font">Fisierul nu a putut fi restaurat!</strong>
                Asigurati-va ca nu ati introdus un fisier gol sau incercati sa eliminati liniile si
spatiile goale si reluati procesul de restaurare.
            </div>
            <!--
            <div class="alert alert-danger" id="abnormal_email" style="display: none;">
                <a href="#" class="close" data-dismiss="alert" aria-label="close">&times;</a>
                <strong class="grid_heading_font">Mail-ul nu a fost trimis!</strong>
                A aparut o problema interna.Va rugam sa reincercati.
            </div>
            -->
        </div>
        <div class="my_row">
            <div class="my_column three">
                <h4     class="grid_heading_font"><span     class="glyphicon     glyphicon-question-sign
logo_details"></span> Cum functioneaza?</h4>
                <p   class="grid_text_font">Dupa   ce   ati   introdus   adresa   de   e-mail   in   campul
corespunzator, alegeti fisierul .docx dorit,
                    pornind astfel restaurarea.</p>
            </div>
            <div class="my_column three">
                <h4      class="grid_heading_font"><span      class="glyphicon      glyphicon-info-sign
logo_details"></span> Cum poti obtine rezultatul?</h4>
                <p class="grid_text_font">Fisierul rezultat (ce contine diacritice) va fi trimis la
adresa specificata dar va putea fi
                    si descarcat direct printr-un simplu click.</p>
            </div>
            <div class="my_column three">
                <h4      class="grid_heading_font"><span      class="glyphicon      glyphicon-thumbs-up
logo_details"></span> Nu modifica fisierul initial</h4>
                <p  class="grid_text_font">Acest  serviciu  restaureaza  diacriticele  fara  a  modifica
aspectul fisierului initial. In plus este
                    si foarte simplu de utilizat!</p>
            </div>
            <div class="my_column three">
                <h4       class="grid_heading_font"><span       class="glyphicon       glyphicon-lock
logo_details"></span> Sigur</h4>
                <p class="grid_text_font">Fisierul va fi procesat pe server-ul nostru, fiind sters
peste 30 de minute pentru a nu fi accesat de catre altcineva.</p>
            </div>
        </div>
        <footer class="container-fluid text-center">
            <div class="copyright footer_design">© speed.pub.ro 2016</div>
        </footer>
    </body>
</html>
```

Script.js

```
(function ($) {
    $(document).ready(function () {
        if (!($("#email").val() === "")) {
            if (isValidEmailAddress($("#email").val())){
                $("#file").prop('disabled', false);
                $("#file").addClass('inputfile_design');
            }else{
                $("#file").prop('disabled', true);
            }
        }
        $("#email").on("input", function () {
            if ($("#email").val() === "") {
                $("#converted").hide();
                $("#file").prop('disabled', true);
                $("#file").removeClass('inputfile_design');
            } else {
                if (!isValidEmailAddress($("#email").val())) {
                    $("#converted").hide();
                    $("#file").prop('disabled', true);
                    $("#file").removeClass('inputfile_design');
                    $("#label_span").text("Alegeti fisierul!");
                } else {
                    $("#file").addClass('inputfile_design');
                    $("#file").prop('disabled', false);
                }
            }
        });
        $("#file").on("change", function () {
            var file = $("input[type='file']").get(0).files[0];
            var fileName = $("input[type='file']").val().split('/').pop().split('\\').pop();
            $("#label_span").text(fileName);
            var email = $("#email").val();
            var serverRequest = new XMLHttpRequest();
            var formData = new FormData();
            formData.append("email", email);
            formData.append("docxfile", file);
            serverRequest.open("POST",
"http://ivan.studenti.speed.pub.ro/RestaurareDiacritice/webresources/diacriticsrestorer/submit", true);
            $("#abnormal_email").hide();
            $("#file_error").hide();
            $("#converted").hide();
            $("#download_link").empty();
            $("#converting_file").text(fileName);
            $("#converting").show();
            serverRequest.onreadystatechange = function ()
```

```
            {
                if (serverRequest.readyState === 4)
                {
                    $("#converted_file").text(fileName);
                    $("#converting").hide();
                    if (serverRequest.status === 200)
                    {
                        $("#download_link").prepend('<a
href="http://ivan.studenti.speed.pub.ro/RestaurareDiacritice/webresources/diacriticsrestorer/download?n
ame=' + fileName +
                                '"><i class=' + '"' + 'fa fa-download' + '">' + '</i>' +'Descarcare'+
'</a>');
                        $("#converted").show();
                    }
                    else
                    {
                        if (serverRequest.status === 500) {
                            $("#file_error").show();
                        }
                    }
                }
            };
            serverRequest.send(formData);
        });
    });
})(jQuery);
function isValidEmailAddress(emailAddress) {
    var pattern = new RegExp(/^(("[\w-+\s]+")|([\w-+]+(?:\.[\w-+]+)*)|("[\w-+\s]+")([\w-+]+(?:\.[\w-
+]+)*))(@((?:[\w-+]+\.)*\w[\w-+]{0,66})\.([a-z]{2,6}(?:\.[a-z]{2})?)$)|(@\[?((25[0-5]\.|2[0-
4][\d]\.|1[\d]{2}\.|[\d]{1,2}\.))((25[0-5]|2[0-4][\d]|1[\d]{2}|[\d]{1,2})\.){2}(25[0-5]|2[0-
4][\d]|1[\d]{2}|[\d]{1,2})\]?$)/i);
    return pattern.test(emailAddress);
};
```

Styles.css

```
.logo_details {
    color: #151B8D;
    font-size: 20px;
}
.button_logo{
    color: white;
    font-size: 15px;
}
.grid_text_font{
    font-size: 20px;
    font-family: Indie Flower, sans-serif;
}
.grid_heading_font{
    font-size: 20px;
    font-family: Lobster, sans-serif;
}
.well_design {
    border-radius: 15px;
    background:  #CFECEC;
    margin-top: 50px;
    margin-bottom: 20px;
}
.converting_info_design{
    border-radius: 2px;
    background: #151B8D;
    padding: 2px;
```

```
}
.converted_info_design{
    border-radius: 2px;
    background: #347235;
    padding: 2px;
    width: 66px;
}
.footer_design{
    margin-top: 20px;
    margin-bottom: 20px;
}
.upload_logo{
    margin-right: 6px;
    margin-left: 2px;
}
a:link {
    font-size: 15px;
    color:white;
    background-color: #347235;
    text-decoration: none;
    border-radius: 5px;
    padding: 6px;
    cursor: pointer;
}
a:visited {
    font-size: 15px;
    color:white;
    background-color: #347235;
    text-decoration: none;
}
a:hover {
    font-size: 15px;
    color:white;
    background-color: #254117;
    text-decoration: none;
}
a:active {
    font-size: 15px;
    color:white;
    background-color: #254117;
    text-decoration: none;
}
.restriction{
    word-wrap: break-word;
}
.trim_text{
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
}
.inputfile_disabled{
    width: 0.1px;
    height: 0.1px;
    opacity: 0;
    overflow: hidden;
    position: absolute;
    z-index: -1;
    display: inline-block;
}
.inputfile_disabled + label{
    font-size: 15px;
    border-radius: 5px;
    font-weight: normal;
    color: white;
    padding: 6px;
    background-color: #B6B6B4;
    display: inline-block;
}
.inputfile_design{
    width: 0.1px;
    height: 0.1px;
    opacity: 0;
    overflow: hidden;
    position: absolute;
    z-index: -1;
    display: inline-block;
}
.inputfile_design + label{
    font-size: 15px;
    border-radius: 5px;
    font-weight: normal;
    color: white;
    padding: 6px;
    background-color: #151B8D;
    display: inline-block;
    cursor: pointer;
}
.inputfile_design:focus + label{
    outline: 1px dotted #000;
    outline: -webkit-focus-ring-color auto 5px;
```

```
    background-color: #151B54;
}
.inputfile_design +label:hover {
    background-color: #151B54;
}
.my_row{
    padding: 40px;
    width: 100%;
    min-height: 1px;
    float: left;
}
.my_column{
    min-height: 40px;
    border-radius: 8px;
    margin-left: 2.4%;
    margin-right: 2.4%;
    display: inline-block;
    float: left;
}
.my_column:first-child{
    margin-left: 0px;
}
.my_column:last-child{
    margin-right: 0px;
}
.my_column.three{
    width: 21.4%;
}
@media (max-width: 580px){
    .my_column{
        width: 100% !important;
        margin: 5px 0px 5px 0px;
    }
}
```

Submit

```
package diacriticsrestorer;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Iterator;
import java.util.List;
import java.util.Properties;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.activation.FileDataSource;
import javax.mail.BodyPart;
import javax.mail.Message;
import javax.mail.Multipart;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.ws.rs.core.Context;
import javax.ws.rs.Consumes;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.WebApplicationException;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
import org.apache.commons.io.FilenameUtils;
import org.apache.poi.xwpf.usermodel.IBodyElement;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.apache.poi.xwpf.usermodel.XWPFParagraph;
import org.apache.poi.xwpf.usermodel.XWPFRun;
import org.apache.poi.xwpf.usermodel.XWPFTable;
import org.apache.poi.xwpf.usermodel.XWPFTableCell;
import org.apache.poi.xwpf.usermodel.XWPFTableRow;

@Path("/diacriticsrestorer")
public class Submit {
    static String _txtFilesFolderPath;
    static String _uploadPath;
    static String _resultPath;
    static {
        try {
            Property.loadProps();
            _uploadPath = Property.props.getProperty("_uploadpath");
            _resultPath = Property.props.getProperty("_resultpath");
            _txtFilesFolderPath = Property.props.getProperty("_txtfilesfolserpath");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    String file_name = null;
    static boolean restorationForTextFinished;
    static boolean restorationForTabelsFinished;

    @POST
    @Path("/submit")
    @Consumes(MediaType.MULTIPART_FORM_DATA)
```

```
public Response submit(@Context HttpServletRequest request) throws Exception {
    InputStream uploadedInputStream = null;
    String email = null;
    boolean mailSent = false;
    List<FileItem> items = new ServletFileUpload(new DiskFileItemFactory()).parseRequest(request);
    for (FileItem item : items) {
        String fieldName = item.getFieldName();
        switch (fieldName) {
            case "docxfile":
                uploadedInputStream = item.getInputStream();
                file_name = item.getName();
                break;
            case "email":
                email = item.getString();
                break;
        }
    }
    String newFileName;
    if (file_name.contains(" ")) {
        String[] name_part = file_name.split(" ");
        newFileName = name_part[0];
        for (int i = 1; i < name_part.length; i++) {
            newFileName = newFileName + "\\" + " " + name_part[i];
        }
    } else {
        newFileName = file_name;
    }
    saveToDisk(uploadedInputStream, file_name);
    restorationForTextFinished = restoreDiacriticsForText(file_name);
    restorationForTabelsFinished = restoreDiacriticsForTabels(file_name);
    if (restorationForTextFinished == true && restorationForTabelsFinished == true) {
        deleteHelpFiles(_txtFilesFolderPath);
        deleteHelpFiles(_uploadPath);
        SendMail(email, file_name);
        deleteResultFiles(newFileName);
        String intermediateFileName = FilenameUtils.removeExtension(newFileName);
        deleteResultFiles(intermediateFileName + "_partial.docx");
        return Response.status(Status.OK).build();
    } else {
        return Response.status(Status.INTERNAL_SERVER_ERROR).build();
    }

}
private void saveToDisk(InputStream uploadedInputStream, String fileToUpload) {
    String uploadedFileLocation = _uploadPath + fileToUpload;
```

```
        try {
            OutputStream out = new FileOutputStream(new File(uploadedFileLocation));
            int read = 10;
            byte[] bytes = new byte[1024];
            out = new FileOutputStream(new File(uploadedFileLocation));
            while ((read = uploadedInputStream.read(bytes)) != -1) {
                out.write(bytes, 0, read);
            }
            out.flush();
            out.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public boolean restoreDiacriticsForText(String fileName) {
        try {
            boolean emptyParagraphs = true;
            InputStream docxInputStream = new FileInputStream(new File(_uploadPath + fileName));
            XWPFDocument docx = new XWPFDocument(docxInputStream);
            String name = FilenameUtils.removeExtension(fileName);
            File outputDocxFile = new File(_resultPath + name + "_partial.docx");
            FileOutputStream docxOutputStream = new FileOutputStream(outputDocxFile);
            List<XWPFParagraph> docxParagraphs = docx.getParagraphs();
            int docxParagraphsNumber = docxParagraphs.size();
            for (int a = 0; a < docxParagraphsNumber; a++) {
                List<XWPFRun> paragraphruns = docxParagraphs.get(a).getRuns();
                if (paragraphruns.size() != 0) {
                    emptyParagraphs = false;
                }
            }
            if (emptyParagraphs == false) {
                String _pathcorpus = _txtFilesFolderPath + name + ".in";
                String _pathcorpusOut = _txtFilesFolderPath + name + ".out";
                FileWriter inputTxt = new FileWriter(_pathcorpus);
                BufferedWriter txtWriter = new BufferedWriter(inputTxt);
                for (int a = 0; a < docxParagraphsNumber; a++) {
                    List<XWPFRun> paragraphRuns = docxParagraphs.get(a).getRuns();
                    for (int b = 0; b < paragraphRuns.size(); b++) {
                        int i = paragraphRuns.get(b).getTextPosition();
                        String text = paragraphRuns.get(b).getText(i);
                        if (text != null) {
                            txtWriter.write(text);
                            emptyParagraphs = false;
                        }
                    }
```

```
                txtWriter.write("\n");
            }
            txtWriter.close();
            DiacriticsProcess.diacritics_process(name);
            BufferedReader finalFile = new BufferedReader(new FileReader(_pathcorpusOut));
            for (int a = 0; a < docxParagraphsNumber; a++) {
                List<XWPFRun> paragraphRuns = docxParagraphs.get(a).getRuns();
                try {
                    String line = finalFile.readLine();
                    if (line != null) {
                        if (line.length() > 0) {
                            char[] characters = line.toCharArray();
                            int char_index = 0;
                            int g = 0;
                            for (int b = 0; b < paragraphRuns.size(); b++) {
                                int f = paragraphRuns.get(b).getTextPosition();
                                String text = paragraphRuns.get(b).getText(f);
                                StringBuilder newText = new StringBuilder();
                                if (text != null) {
                                    char[] textcharacters = text.toCharArray();
                                    int chars_nr = 0;
                                    while (chars_nr < text.length()) {
                                        newText.append(characters[char_index]);
                                        char_index++;
                                        chars_nr++;
                                    }
                                    paragraphRuns.get(b).setText(newText.toString(), 0);
                                }
                            }
                        }
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            finalFile.close();
        }
        docx.write(docxOutputStream);
        docxOutputStream.close();
        return true;
    } catch (Exception ex) {
        ex.printStackTrace();
        return false;
    }
}
```

```
    public boolean restoreDiacriticsForTabels(String fileName) {
        try {
            boolean noTables = true;
            String name = FilenameUtils.removeExtension(fileName);
            InputStream docxInputStream = new FileInputStream(new File(_resultPath + name + "_partial.docx"));
            XWPFDocument docx = new XWPFDocument(docxInputStream);
            File outputDocxFile = new File(_resultPath + name + ".docx");
            FileOutputStream docxOutputStream = new FileOutputStream(outputDocxFile);
            Iterator<IBodyElement> initialbodyElementIterator = docx.getBodyElementsIterator();
            while (initialbodyElementIterator.hasNext()) {
                IBodyElement initialElement = initialbodyElementIterator.next();
                if ("TABLE".equalsIgnoreCase(initialElement.getElementType().name())) {
                    List<XWPFTable> initialTableList = initialElement.getBody().getTables();
                    if (initialTableList.size() != 0) {
                        noTables = false;
                    }
                }
            }
            if (noTables == false) {
                String _pathcorpus = _txtFilesFolderPath + name + "_partial" + ".in";
                String _pathcorpusOut = _txtFilesFolderPath + name + "_partial" + ".out";
                FileWriter inputTxt = new FileWriter(_pathcorpus);
                BufferedWriter txtWriter = new BufferedWriter(inputTxt);
                Iterator<IBodyElement> bodyElementIterator = docx.getBodyElementsIterator();
                while (bodyElementIterator.hasNext()) {
                    IBodyElement element = bodyElementIterator.next();
                    if ("TABLE".equalsIgnoreCase(element.getElementType().name())) {
                        List<XWPFTable> tableList = element.getBody().getTables();
                        for (int table_nr = 0; table_nr < tableList.size(); table_nr++) {
                            XWPFTable table = tableList.get(table_nr);
                            List<XWPFTableRow> rowsList = table.getRows();
                            for (XWPFTableRow row : rowsList) {
                                List<XWPFTableCell> cellsList = row.getTableCells();
                                for (XWPFTableCell cell : cellsList) {
                                    int cell_para = cell.getParagraphs().size();
                                    List<XWPFParagraph> cellParagraphs = cell.getParagraphs();
                                    for (int v = 0; v < cell_para; v++) {
                                        List<XWPFRun> cellRuns = cellParagraphs.get(v).getRuns();
                                        for (int w = 0; w < cellRuns.size(); w++) {
                                            int i = cellRuns.get(w).getTextPosition();
                                            String text = cellRuns.get(w).getText(i);

                                            if (text != null) {
                                                txtWriter.write(text);
                                            }
```

```
                        }
                        txtWriter.write("\n");

                    }

                }
            }
        }
    }
}
txtWriter.close();
DiacriticsProcess.diacritics_process(name + "_partial");
BufferedReader finalFile = new BufferedReader(new FileReader(_pathcorpusOut));
Iterator<IBodyElement> newbodyElementIterator = docx.getBodyElementsIterator();
while (newbodyElementIterator.hasNext()) {
    IBodyElement element = newbodyElementIterator.next();
    if ("TABLE".equalsIgnoreCase(element.getElementType().name())) {
        List<XWPFTable> tableList = element.getBody().getTables();
        for (int table_nr = 0; table_nr < tableList.size(); table_nr++) {
            XWPFTable table = tableList.get(table_nr);
            List<XWPFTableRow> rowsList = table.getRows();
            for (XWPFTableRow row : rowsList) {
                List<XWPFTableCell> cellsList = row.getTableCells();
                for (XWPFTableCell cell : cellsList) {
                    int cell_para = cell.getParagraphs().size();
                    List<XWPFParagraph> cellParagraphs = cell.getParagraphs();
                    for (int v = 0; v < cell_para; v++) {
                        List<XWPFRun> cellRuns = cellParagraphs.get(v).getRuns();
                        try {
                            String line = finalFile.readLine();
                            if (line != null) {
                                if (line.length() > 0) {
                                    char[] characters = line.toCharArray();
                                    int char_index = 0;
                                    for (int w = 0; w < cellRuns.size(); w++) {
                                        int f = cellRuns.get(w).getTextPosition();
                                        String text = cellRuns.get(w).getText(f);
                                        StringBuilder newText = new StringBuilder();
                                        if (text != null) {

                                            int chars_nr = 0;
                                            while (chars_nr < text.length()) {
                                                newText.append(characters[char_index]);
                                                char_index++;
                                                chars_nr++;
```

```
                                                    }
                                                cellRuns.get(w).setText(newText.toString(), 0);
                                                }

                                            }
                                        }
                                    }
                                } catch (Exception e) {
                                    e.printStackTrace();
                                }

                            }
                        }
                    }
                }
            }

            finalFile.close();
        }
        docx.write(docxOutputStream);
        docxOutputStream.close();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
@GET
@Path("download")
@Consumes(MediaType.TEXT_PLAIN)
@Produces("application/vnd.openxmlformats-officedocument.wordprocessingml.document")
public Response downloadFile(@Context HttpServletRequest request) throws Exception {
    String fileName = request.getParameter("name");

    String filepath = new String(_resultPath + fileName);
    File file = new File(filepath);

    if (!file.exists()) {
        throw new WebApplicationException(404);
    }
    String name = "filename=" + "\"" + fileName + "\"";
    return Response.ok(file)
            .header("Content-Disposition",
                    "attachment;" + name).build();
```

```
    }

    public void deleteResultFiles(String fileName) {
        try {

            String[] deleteCommand = {"bash", "-c", "sleep 1800 " + "&& rm " + _resultPath + fileName};
            Process deleteProcess = Runtime.getRuntime().exec(deleteCommand);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void deleteHelpFiles(String directoryPath) {
        File folder = new File(directoryPath);
        File[] listOfFiles = folder.listFiles();

        for (int i = 0; i < listOfFiles.length; i++) {
            if (listOfFiles[i].isFile()) {
                listOfFiles[i].delete();
            }
        }
    }
    public void SendMail(String email, String filetosend) {
        try {
            String host = new String();
            String from = new String("diacriticsrestorer@gmail.com");
            String pass = new String("[redacted]");
            String subject = new String("Rezultat proces restaurare");
            String message = new String("Primiti acest e-mail deoarece ati folosit Sericiul web de
restaurare de diacritice al http://speed.pub.ro/.");
            Properties prop = System.getProperties();
            prop.put("mail.smtp.starttls.enable", "true");
            prop.put("mail.smtp.host", "smtp.gmail.com");
            host = "smtp.gmail.com";
            prop.put("mail.smtp.user", from);
            prop.put("mail.smtp.password", pass);
            prop.put("mail.smtp.port", 587);
            prop.put("mail.smtp.auth", "true");
            Session session = Session.getDefaultInstance(prop, null);
            MimeMessage mailMessage = new MimeMessage(session);//mesajul care poate identifica tipurile
si antetele MIME (continut+antet)
            mailMessage.setFrom(new InternetAddress(from));
            mailMessage.setRecipient(Message.RecipientType.TO, new InternetAddress(email));
            mailMessage.setSubject(subject);
            BodyPart mailMessageBodyPart = new MimeBodyPart();
            mailMessageBodyPart.setText(message);
            Multipart multipart = new MimeMultipart();
```

```
            multipart.addBodyPart(mailMessageBodyPart);
            String file = _resultPath + filetosend;
            mailMessageBodyPart = new MimeBodyPart();
            DataSource source = new FileDataSource(file);
            mailMessageBodyPart.setDataHandler(new DataHandler(source));
            mailMessageBodyPart.setFileName(filetosend);
            multipart.addBodyPart(mailMessageBodyPart);
            mailMessage.setContent(multipart);
            Transport transport = session.getTransport("smtp");
            transport.connect(host, from, pass);
            transport.sendMessage(mailMessage, mailMessage.getAllRecipients());
            transport.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

ANEXA 2

Start – Activitatea aplicației Android

```
package com.speedlab.restaurarediacritice;
import android.support.v7.app.ActionBarActivity;
import android.text.method.ScrollingMovementMethod;
import java.util.ArrayList;
import java.util.List;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

@SuppressLint("NewApi")
public class Start extends ActionBarActivity {
        public final static String EXTRA_MESSAGE = "com.speedlab.restaurarediacritice.MESSAGE";
        TextView output;
        EditText input;
        ProgressBar pbar;
        List<BackgroundTask> tasks;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_start);
                output = (TextView) findViewById(R.id.textView);
                output.setMovementMethod(new ScrollingMovementMethod());
                input = (EditText) findViewById(R.id.edit_text);
                input.setMovementMethod(new ScrollingMovementMethod());
```

```
                pbar = (ProgressBar) findViewById(R.id.progressBar1);
                pbar.setVisibility(View.INVISIBLE);
                input.setText("");
                output.setText("");
                tasks= new ArrayList();
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.start, menu);
                return true;
        }
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.send_text) {
                        InputMethodManager inputManager = (InputMethodManager)
                    getSystemService(Context.INPUT_METHOD_SERVICE);

                        inputManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
                       InputMethodManager.HIDE_NOT_ALWAYS);
                        if (isOnline()) {
                                if(input.getText().length()==0){
                                        Toast.makeText(this,        "Nu        exista        text!",
Toast.LENGTH_LONG).show();
                                }else
        requestData("http://ivan.studenti.speed.pub.ro/RestaurareDiacritice/webresources/androiddiacri
ticsrestorer/process");
                                }
                        } else {
                                Toast.makeText(this,    "Nu    aveti    conexiune    la    internet!",
Toast.LENGTH_LONG).show();Toast.makeText(this,     "Nu     aveti     conexiune     la     internet!",
Toast.LENGTH_LONG).show();
                        }
                }
                if(id== R.id.send_mail){
                        InputMethodManager inputManager = (InputMethodManager)
                    getSystemService(Context.INPUT_METHOD_SERVICE);
                       inputManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
                       InputMethodManager.HIDE_NOT_ALWAYS);
                        if(isOnline()){
                        if(output.getText().length()==0){
                                Toast.makeText(this, "Nu exista text!", Toast.LENGTH_LONG).show();
                        }else{
```

```
                                Intent emailIntent=new Intent(Intent.ACTION_SEND);
                                emailIntent.putExtra(Intent.EXTRA_TEXT, output.getText());
                                emailIntent.setType("message/rfc822");
                                startActivity(Intent.createChooser(emailIntent,  "Alege  aplicatia  de
email dorita!"));
                                input.setText("");
                                output.setText("");
                        }
                        }else{
                                Toast.makeText(this,    "Nu    aveti    conexiune    la    internet!",
Toast.LENGTH_LONG).show();}
                }
                return super.onOptionsItemSelected(item);
        }
        @Override
        public void onUserLeaveHint() {
            finish();
            super.onUserLeaveHint();
        }
        private void requestData(String uri) {
        EditText editText = (EditText) findViewById(R.id.edit_text);
        String message = editText.getText().toString();
        RequestToWebService r=new RequestToWebService();
        r.setMethod("GET");
        r.setUri(uri);
        r.setParameters("inputText", message);
        BackgroundTask task = new BackgroundTask();
        task.execute(r);
        }
        protected boolean isOnline() {
                ConnectivityManager           connectivitym           =           (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
                NetworkInfo netInfo = connectivitym.getActiveNetworkInfo();
                if (netInfo != null && netInfo.isConnectedOrConnecting()) {
                        return true;
                } else {
                        return false;
                }
        }
        protected void updateDisplay(String message) {
                output.append(message);
        }
        private class BackgroundTask extends AsyncTask<RequestToWebService, String, String> {
                @Override
                protected void onPreExecute() {
                        output = (TextView) findViewById(R.id.textView);
```

```
                output.setText("");
                if(tasks.size()==0){
                pbar.setVisibility(View.VISIBLE);
                }
                tasks.add(this);
        }
        @Override
        protected String doInBackground(RequestToWebService... params) {
                String content= HttpManager.getData(params[0]);
                return content;
        }
        @Override
        protected void onPostExecute(String result) {
                updateDisplay(result);
                tasks.remove(this);
                if(tasks.size()==0){
                pbar.setVisibility(View.INVISIBLE);
                }
        }
```

RequestToWebService

```
package com.speedlab.restaurarediacritice;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.HashMap;
import java.util.Map;
public class RequestToWebService {
        private String uri;
        private String method;
        private Map<String,String> parameters=new HashMap<>();
        public String getUri() {
                return uri;
        }
        public void setUri(String uri) {
                this.uri = uri;
        }
        public String getMethod() {
                return method;
        }
        public void setMethod(String method) {
                this.method = method;
        }
        public Map<String, String> getParameters() {
                return parameters;
        }
```

```
	public void setParameters(Map<String, String> parameters) {
		this.parameters = parameters;
	}
	public void setParameters(String key, String value){
		parameters.put(key, value);
	}
	public String getEncodedParameters(){
		StringBuilder sb=new StringBuilder();
		for(String key: parameters.keySet()){
			String value=null;
			try {
				value = URLEncoder.encode(parameters.get(key),"UTF-8");
			} catch (UnsupportedEncodingException e){
				e.printStackTrace();
			}
			if(sb.length()>0){
				sb.append("&");
			}
			sb.append(key+"="+value);
		}
		return sb.toString();
	}
}
```

HttpManager

```
package com.speedlab.restauraredіacritice;
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import android.util.Log;
public class HttpManager {
	public static String getData(RequestToWebService request) {
		BufferedReader reader = null;
		String uri=request.getUri();
		if(request.getMethod().equals("GET")){
			uri+="?"+request.getEncodedParameters();
		}
		try {
			URL url = new URL(uri);
			HttpURLConnection con = (HttpURLConnection) url.openConnection();
			con.setRequestMethod(request.getMethod());
			StringBuilder sb = new StringBuilder();
```

```
                        reader = new BufferedReader(new InputStreamReader(con.getInputStream()));
                        String line;
                        while ((line = reader.readLine()) != null) {
                                sb.append(line + "\n");
                        }
                        return sb.toString();
                } catch (Exception e) {
                        e.printStackTrace();
                        return "Conectare esuata!";
                } finally {
                        if (reader != null) {
                                try {
                                        reader.close();
                                } catch (IOException e) {
                                        e.printStackTrace();
                                        return "Au aparut probleme in timpul obtinerii textului!";
                                }
                        }
                }
        }
}
```