RAPORTARE ȘTIINȚIFICĂ

Proiect complex ReTeRom. Proiect component CoBiLiRo

Activitatea A2.3

Realizarea de convertoare de format pentru armonizarea diferitelor reprezentări ale partenerilor la reprezentarea standard agreată în consorțiu

Faza de predare: septembrie 2019

Autori: Ionuț Pistol, Andrei Scutelnicu, Cristian Pădurariu;
Colaborator extern: Șerban Boghiu

1. Rezumatul etapei

A doua etapă (2019) a proiectului CoBiLiRO prevede realizarea infrastructurii pentru gestionarea și stocarea corpusului bimodal. Activitățile complementare prevăzute includ descrierea și implementarea unor soluții de armonizare a reprezentărilor colecțiilor existente text/vorbire (metadata și adnotări). Ulterior, aceste soluții urmează să fie utilizate pentru a armoniza formatele colecțiilor existente și a le încărca pe platforma CoBiLiRO. Similar celorlalte etape, este prevăzută o activitate de diseminare atât la evenimente științifice cât și în mass-media. De interes special pentru platforma dezvoltată este respectarea drepturilor de autor și a anonimizării solicitate pentru contributorii de resurse pe platformă.

2. Rezumatul activității

Această activitate a avut ca obiectiv implementarea mecanismelor de conversie la formatul CoBiLiRo (descris în raportul activității A1.3) a celor trei formate utilizate de parteneri (descrise în raportul activității A2.2). Aceste convertoare sunt aplicate, în prezent, automat, în momentul adaugării unei resurse pe platforma CoBiLiRo (descrisă în raportul activității A2.1). Platforma suportă și conversia fișierelor din arhive. Cum procesul de conversie nu este fără pierdere de informații, unele formate fiind mai detaliate decât standardul propus în raportul A1.3, fișierele originale sunt păstrate pe server împreună cu fișierele convertite. În cazul unor modificări ulterioare a standardului propus, mecanismele de conversie pot fi adaptate și fișierele originale reconvertite.

3. Descrierea științifică și tehnică

3.1 Introducere

Rezultatele principale ale activităților A1.2 și A1.3 au fost identificarea a trei formate predominant utilizate în reprezentarea resurselor partenerilor, respectiv propunerea unui format standard pentru resursele de pe platforma CoBiLiRo. În raportul A2.2 au fost propuse soluții tehnice pentru conversia resurselor disponibile în unul din cele trei formate la standardul descris în raportul activității A1.3, soluții ușor adaptabile la eventuale viitoare actualizări ale platformei. Conform metodei indicate în raportul A2.2, pot fi concepute în viitor și alte procedee de conversie, în cazul în care apar noi resurse relevante pentru platformă ce nu sunt disponibile în unul din cele trei formate indicate.

Acest raport continuă în Secțiunea 3.2 cu o descriere pe scurt a unor actualizări ale standardului CoBiLiRo, în urma experienței căpătate studiind cerințele platformei dar și formatele resurselor disponibile. În secțiunea 3.3 sunt descrise implementările și modalitatea de funcționare pentru cele trei convertoare, iar în secțiunea 4 sunt prezentate soluțiile tehnice adoptate pe serverul CoBiLiRo în vederea conversiei unui volum mare de date, eventual contribuite sub forma unei arhive.

3.2 Completări asupra formatului CoBiLiRo

Formatul CoBiLiRo, așa cum a fost descris în raportul A1.3, prevede două tipuri posibile de alinieri text-vorbire. Tipul *"file"* este prevăzut pentru situația în care există mai multe fișiere audio, pentru care avem textul corespunzător. Al doilea tip de aliniere prevăzut de formatul CoBiLiRo se numește *"start-stop"* și se referă la alinieri făcute la nivelul unui singur fișier. Așadar, dacă resursa încărcată cuprinde un singur fișier audio și mai multe texte extrase din această înregistrare, vom apela la acest tip de aliniere. Fiecare secțiune de text va fi însoțită de momentul în care a început rostirea sa în cadrul înregistrării (*start*) și momentul încheierii rostirii sale (*stop*).

În raportul A2.2 [2] am descris formatul TEXTGRID care prezintă mai multe fișiere audio, fiecare având propria aliniere "*start-stop*". Acest scenariu ne plasează între cele tipuri de alinieri ale formatului CoBiLiRo, deoarece avem mai multe fișiere audio, dar și o aliniere de tip *"start-stop"* pentru fiecare din fișierele text. Astfel, propunem extinderea formatului existent adăugând un nou tip de aliniere de tip *"file-start-stop"*. Acesta va avea, la fel ca tipul *"file"*, un tag <*unit*> cu un atribut *"speechFile"*, dar și o serie de tag-uri <*subunit*>, cu atributele *"start"* și *"stop"*. Figura 1 include un exemplu de astfel de metadată.

```
<unit>
   <speech speechFile="file1.wav"/>
    <subunit>
        <speech start="0.0000" stop = "8.5673"/>
        <text>...</text>
    </subunit>
    ...
     <subunit>
        <speech start="8.5673" stop="15.6652"/>
        <text>...</text>
     </subunit>
```

```
</unit>
...
<unit>
   <speech speechFile=”file2.wav”/>
   <subunit>
        <speech start=”0.0000” stop = ”4.1231”/>
        <text>...</text>
    </subunit>
    …
    <subunit>
        <speech start=”3.7234” stop=”10.423”/>
        <text>...</text>
    </subunit>
</unit>
```

Figura 1: Exemplu de notare a unui semnal de vorbire în format WAV și tipul de segmentare “file-start-stop”

3.3 Soluții de armonizare propuse

Pentru fiecare format va fi descris pe scurt procesul complet de conversie, cu exemple. Sunt menţionate și posibile deficiențe (pierdere de informații, viteză scăzută), cu posibile soluții.

3.3.1 Header

Procesul de definire a *header*-ului specific unei resurse, așa cum a fost descris în referatul A1.3 [1], este următorul:

- Contributorul va accesa pagina de creare a resursei;
- Contributorul va completa formularul asociat, vizibil în figura 2;
- Contributorul va încărca fișierele asociate resursei;
- În final va acționa butonul “Crează” ce va permite transmiterea informației către server. Aplicația salvează informațiile din formular într-un obiect (*ResourceViewModel*), iar fișierele resursă într-o listă (*List<IFormFile>*).

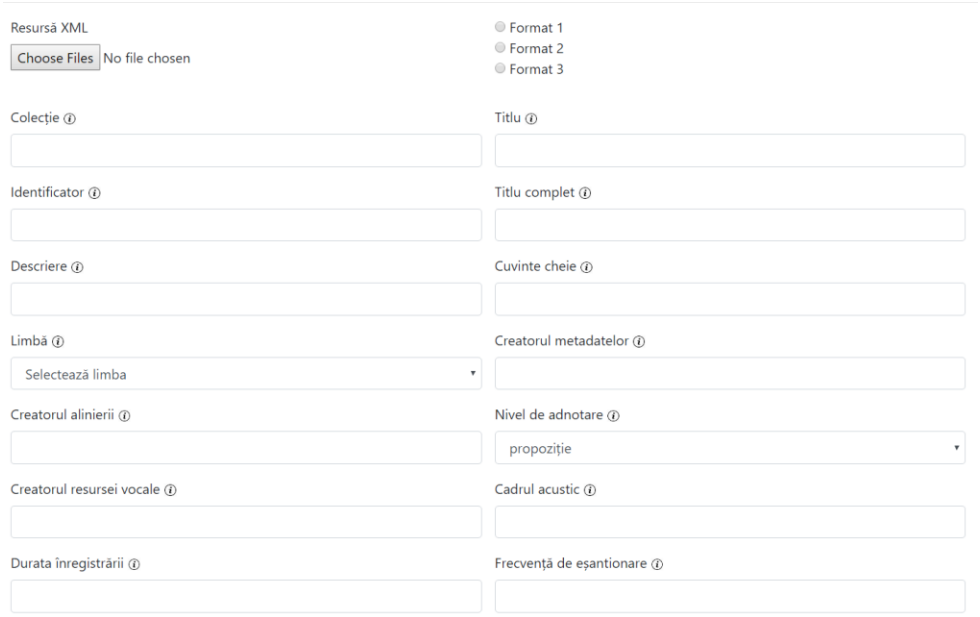

Figura 2: Formularul de descriere a unei resurse

Primul pas în conversia formatelor existente la formatul CoBiLiRo este acela al mapării stocate în obiectul *ResourceViewModel* la câmpurile aferente din header-ul noului format.

Pentru această etapă vom utiliza un *AutoMapper*, care realizează conversia automată între două obiecte aparținând unor clase diferite, pe baza unor reguli definite de programator. Utilizând această opțiune vom realiza conversia de la obiectul *ResourceViewModel* la obiectul de tip *TeiHeader*, care conține o serie de atribute ce corespund fiecărui element din header. Regulile de mapare sunt definite sub formă de perechi de tip *clasa1.atribut => clasa2.atribut,* care sugerează corespondența dintre atributul sursă și atributul destinație.

3.3.2 Formatul PHS/LAB

După cum a fost descris în raportul A2.2 [2], formatul PHS/LAB este compus din serii de câte patru fișiere cu extensiile *wav, phs, lab, txt*. Fișierul *wav* conține o înregistrare audio, fișierul *txt* conține varianta nealterată a celor rostite în cadrul înregistrării, fișierul *lab* reprezintă varianta procesată a textului, după eliminarea semnelor de punctuație și convertirea literelor mari în litere mici, iar fișierul *phs* conține o listă a tuturor literelor din text împreună cu momentele de start și de final ale rostirii lor.

Conversia acestui format la formatul CoBiLiRo implică crearea header-ului (conform metodei descrise mai sus, comună tuturor formatelor) și a tag-urile de tip *unit*, ce vor păstra fișierele audio, în care se regăsesc înregistrările sonore, și, respectiv, transcrierile lor în text.

Ținând cont de faptul că fiecare fișier dintr-un astfel de calup de patru au același nume, vom crea grupuri de câte patru fișiere cu același nume. Astfel, vom ști care sunt fișierele text

asociate unei anumite resurse audio. În tag-ul *unit* vom adăuga textul găsit în fișierul *.txt,* iar atributul *speechFile*, va lua valoarea numelui fișierului *.wav.*

Pentru a realiza acest lucru a fost construită o funcție ce primește ca parametru lista fișierelor resursă. Aceasta este împărțită, în funcție de numele lor, în liste de câte patru fișiere. Astfel, se poate ști care fișiere *wav, phs, lab* și *txt,* aparțin cărei resurse.

În continuare, serverul citește fișierele *txt* din fiecare grup identificat. După crearea elementului *xml,* numit *unit,* se crează un subelement numit *text* ce primește ca valoare textul citit din fișier. Se crează apoi un element de tip atribut, numit *speechFile*, ce va lua valoarea numelui fișierului audio corespunzător. Acest atribut este atașat tag-ului *unit*, conform specificațiilor formatului CoBiLiRo, alături de un atribut *id,* ce ajută la identificare unică a fiecărui *unit*.

După realizarea acestei rutine pentru fiecare grup de câte patru fișiere identificate, se adăugă într-un singur document *xml* header-ul obținut în prima fază și tag-urile *unit* create la pasul anterior.

Fișierul *xml* rezultat în urma conversiei, împreună cu fișierele audio asociate sunt salvate pe disc și reprezentă varianta finală, convertită a resursei.

Pierderile suferite în urma conversie se referă în principal la fișierele *phs*, a căror informație nu se regăsește în formatul CoBiLiRo. De asemenea, nici textele preprocesate găsite în fișierele *lab* nu vor apărea în varianta finală. Fișierele originale sunt însă păstrate pe serverul aplicației, putând fi oricând utilizate în cazul unei schimbări a procedurii de conversie sau a formatului CoBiLiRo.

3.3.3 Formatul 2 (MULTEXT/TEI)

Acest format este reprezentat printr-un fișier *xml* și o serie de fișiere audio cu extensia *wav*. *Xml*-ul conține o serie de metadate și textul asociat fiecărui fișier *wav*. Astfel, conversia va consta în extragerea textului aferent fiecărei înregistrări și salvarea lui în formatul CoBiLiRo. Salvarea elementelor header-ului se face conform procedeului descris în secțiunea 3.1. Ca urmare vom explica în continuare modul de construcție a tag-urilor *unit*.

Am creat o funcție ce primește ca parametru lista fișierelor resursă. Printre acestea se căută fișierul *xml* ce conține informația textuală. Conținut *xml*-ului este încărcat în memorie, după se crează o interogare care ne va returna tag-urile *div* ce conținut atributul *type* cu valoarea *block.* În aceste taguri se regăsesc textele care ne interesează. Pentru a extrage numele fișierului asociat fiecărui text trebuie accesate valorile atributelor *url* ale tag-urilor *xref,* ce apar ca subtag-uri al acelorași *div*-uri.

După extragerea textului și a numelui fișierului audio ce conține înregistrarea sa, se crează tag-urile *unit*. Acestea primesc ca atribut *speechFile* valoarea numelui fișierului audio. Subelementul *text* va primi ca valoare textul extras.

În final se plasează în același *xml* header-ul creat anterior și tag-urile *unit* ce conțin asocierile între text și fișierele audio. Ca resursă finală, se salvează pe disc *xml*-ul rezultat împreună cu fișierele audio corespunzătoare.

Conversia se face fără pierdere de informație relevantă. Se pierd doar metadatele din fișierul *xml* inițial care nu reprezintă informație relevantă pentru scopul nostru. Fișierele audio și textul corespunzător rămân și în forma finală a resursei.

3.3.4 Formatul TEXTGRID

Acest format este reprezentat de perechi de câte 3 fișiere: *TextGrid, wav* si *txt*. Din fișierul *TextGrid* se extrag următoarele informații: *xmin* și *xmax*, intervalul și textul, adică vocalele conținute în respectiva secvență. Fișierul *wav* conține întreaga înregistrare audio. Fișierul .txt conține, după header, câte o linie pentru fiecare vocală, unde sunt marcați parametrii acustici ai fiecărei vocale: durata (ms), energia (dB) și frecvența (extrasă în câte trei puncte). De asemenea, în partea de jos a fișierului, sunt marcate momentele în timp unde au fost citite valorile.

După extragerea textului și a numelui fișierului audio ce conține înregistrarea sa, se crează tag-urile *unit,* care, drept atribut *speechFile*, vor lua valorile numelor fișierelor audio.

Mai jos, în Figura 3, sunt prezentați pașii prin care trece o cerere HTTP către server. După ce metadatele și fișierele au fost încărcate, pagina trimite o cerere de tip *POST*. Durata cererii este de 653 ms, care cuprinde: încărcarea conținutului, autentificarea și autorizarea cererii, executarea metodei *Create* (maparea la formatul *Tei*), generarea resursei *xml și* salvarea în baza de date MySql a fișierelor încărcate și apoi redirectionarea utilizatorului la pagina cu resurse.

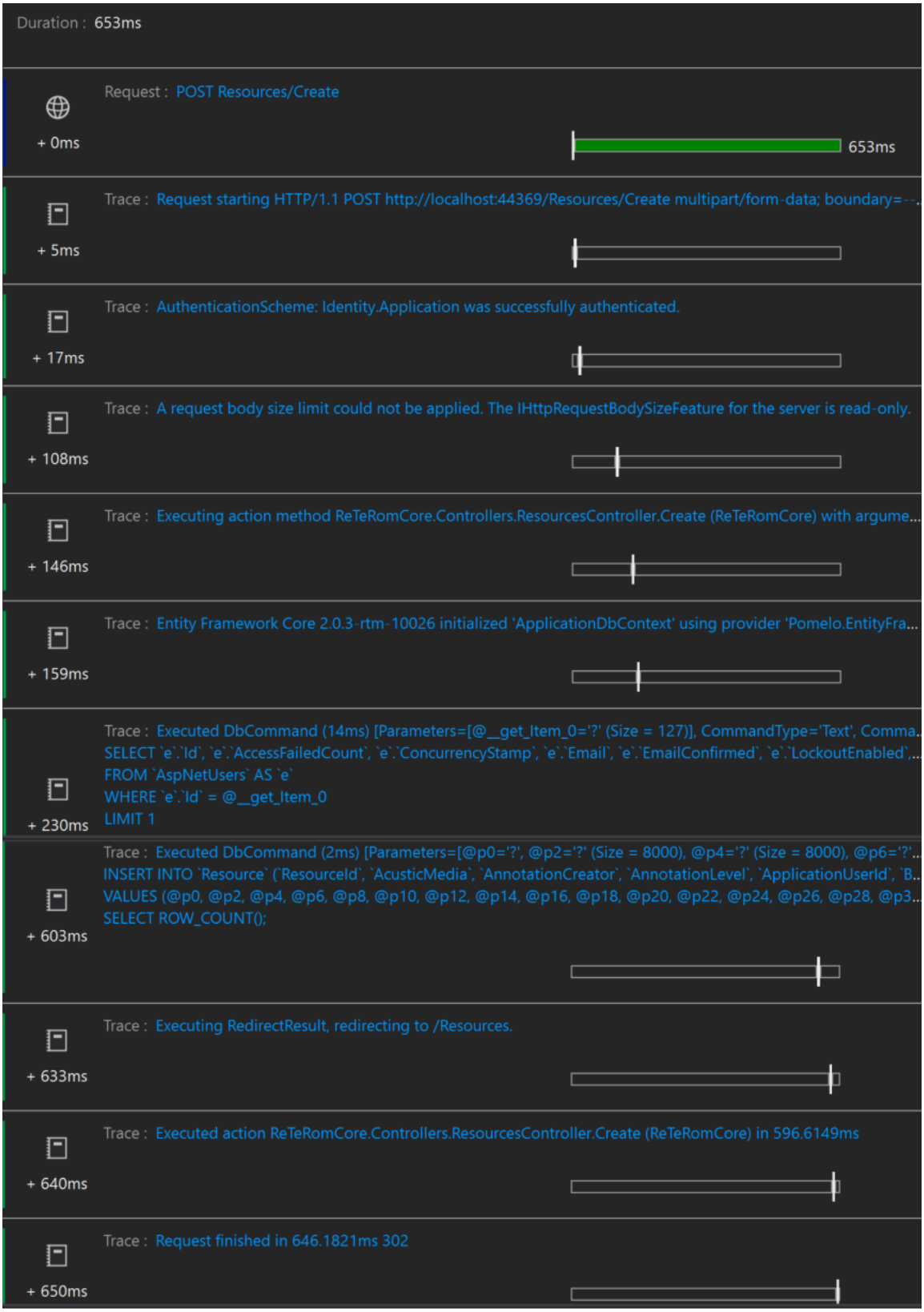

Figura 3: Etape în adăugarea unei resurse noi pe server

În Figura 3 putem observa pașii prin care trece un *request* (*650ms*):

- În prima fază, *request*-ul este autentificat și autorizat de *layer*-ul de securitate folosind parametrii de autentificare din *header*-ul *request*-ului.
- Dacă *request*-ul trece de autorizare, se trece la pasul următor, prin care se crează o instanță a "*AplicationDbContext*", ce facilitează comunicarea cu baza de date, folosind ca ORM Pomelo Entity Framework.
- În următorul pas sunt luate date cu privire la persoana care crează resursa, apoi se adaugă noua înregistrare în baza de date.
- Dacă salvarea a fost făcută cu succes, utilizatorul este redirecționat apoi la pagina cu toate resursele

3.4 Soluții tehnice pentru conversia automată

Serverul pe care este instalată aplicația platformei este modelul HP ProLiant ML350 Gen9, cu un procesor E5-2650 v3 de 2.30GHz, care conține 40 de core-uri.

Având acest număr de core-uri, aplicațiile se pot împărți pe fire de execuție (thread-uri). O aplicație nu consumă mai mult de 5 fire de execuție la o rulare. Având la dispoziție 40 de core-uri, nu vor fi ocupate simultan toate, întrucât 15 core-uri vor fi necesare pentru rularea sistemului de operare și a altor programe necesare acestuia și se recomandă să existe un minim de 5 core-uri care să fie lăsate libere pentru orice altă procesare care poate fi realizată de către server - update, generare de log-uri etc. Astfel, avem la dispoziție un număr de 20 de core-uri pentru aplicație, adică 4 procese a câte 5 fire de execuție.

Principalul motiv pentru care s-a ales framework-ul ASP.NET Core a fost faptul că tehnologia este *open-source, necesită resurse de calcul rezonabile, este rapidă și modulară*. Acest framework este în plină dezvoltare și ne pune la dispoziție o serie de librării și pachete *NuGet* ce ne ajută la dezvoltarea rapidă a platformei.

Unul din beneficiile majore aduse de .NET Core este portabilitatea. Acest lucru permite găzduirea aplicației pe orice sistem de operare.

Un alt motiv pentru care am folosit această tehnologie a fost faptul că oferă o securitate a datelor (protecție pentru atacuri de tip *SQL Injection* și *Cross-site request forgery)* și un mod de a securiza API-urile REST folosind *JSON Web Token.*

Performanța ne este garantată de faptul ca aplicația poate fi scalată pe anumite servicii (de exemplu, putem aloca un număr mai mare de instanțe pentru serviciile de conversie - care vor fi mai costisitoare din punct de vedere al procesării).

Pentru programarea la nivel de client - *client-side* s-a fost folosit framework-ul jQuery. Acesta permite manipularea DOM-ului și accesul la anumite animații și validări specifice.

S-a ales să se folosească ca suport pentru bazele de date serverul MariaDB, pentru că este unul dintre cele mai populare servere de baze de date din lume. Este realizat de dezvoltatorii originali ai MySQL și garantat să rămână open source (gratuit). Deoarece serverul pe care va rula aplicația este cu sistem de operare Linux, acest server virtual pentru baze de date este printre

puținele care poate rula sub Linux. Un alt motiv pentru care s-a ales MariaDB este securitatea informației. Sistemul de securitate al informației este foarte bine pus la punct, atât prin prisma sistemului de operare - Linux, cât și din cea a configurării mașinii virtuale pe care va rula MariaDB.

Sistemele de operare în reţea (NOS) furnizează funcţii de reţea, servicii de reţea, capacitate multiuser, multitasking, securitate, partajarea resurselor şi administrarea centralizată a reţelei:

- Permite accesul mai multor utilizatori;
- Execută aplicaţii multi-user;
- Este robust şi redundant;
- Oferă securitate crescută în comparaţie cu sistemele de operare de tip desktop.

NOS trebuie să aibă un nucleu robust pentru a evita erorile şi întreruperile. Este deosebit de important ca nucleul NOS să poată gestiona mai multe procese, împiedicând astfel blocarea altor părţi ale sistemului. NOS utilizează sistemele de fişiere care permit stocarea rapidă şi sigură a informaţiilor.

Cele mai folosite NOS sunt Windows şi Linux. Linux-ul a fost conceput iniţial ca un sistem de operare în reţea, distribuit gratuit (sub licenţă GNU/GPL), utilizând pentru administrare o interfață în linie de comandă (CLI). Ulterior, când a devenit mai cunoscut, a început să migreze către sisteme de tip desktop, fiind însoțit și de o interfață grafică prietenoasă (GUI). Menționăm că există distribuții sau versiuni de Linux comerciale, dar marea majoritate sunt distribuite gratuit, incluzând şi codul sursă.

Distribuția folosită de noi în configurarea serverului este de tipul RedHat, sistemul de operare numindu-se CentOS - versiunea 7.6.181.

Ca soluție pentru încărcarea fișierelor mari am ales sa stabilim o limită de încărcare a acestora de 3Gb. Încărcarea se va putea realiza din linie de comandă, utilizând comanda *scp* sau prin utilizarea unui program de tip transfer fișiere gen: *WinSCP, FileZilla*. Dacă utilizatorul va încerca să încarce fișiere mai mari de 3Gb, se recomandă ca soluție arhivarea fișierului în pachete multiple, folosind utilitarul *WinRar* sau *7Z*. Aceste utilitare pot împărți o arhivă în pachete de maxim 500Mb/arhivă, iar procesul de încărcare este optimizat, având avantajul că dacă se întrerupe conexiunea de internet, procesul poate fi reluat de la ultima arhivă încărcată.

În cazul în care arhiva nu ajunge integral pe server (upload-ul eșuează din motive cum ar fi întreruperea conexiunii la sursa arhivei, spațiu insuficient, probleme hardware cu serverul), eroarea este semnalată automat de un script ce rulează pe server și care re-începe automat procesul de descărcare.

Pentru situația în care upload-ul resursei se va face sub formă unei arhive de tip *zip,* a fost implementată o rutină de dezarhivare a conținutului și salvarea lui temporară pe disc până la terminarea conversiei. De asemenea, avem posibilitatea de a caută recursiv prin directoare în eventualitatea în care fișierele nu se regăsesc direct în arhivă. După dezarhivare, fișierele vor fi încărcate într-o listă de tip *List<IFormFile>* pentru a putea fi folosite drept intrări pentru convertoare, care necesită liste de acest tip. La momentul încărcării unei resurse se verifică dacă aceasta este formată dintr-un singur fișier de tip *zip*. În caz afirmativ se va apela rutina de dezarhivare ce va returna lista de fișiere dorită.

La momentul scrierii acestui livrabil, funcționalitatea este testată doar pe arhive de tip *zip*.

În ceea ce privește validarea resursei încărcate, propunem, în primă instanță, verificarea prezenței tuturor fișierelor necesare conversiei. Spre exemplu, pentru o resursă în formatul 1 verificăm dacă acesta conține pentru fiecare fișier *wav* fișierele *lab, phs* și *txt* corespunzătoare. În caz contrar contributorul va primi un mesaj de tipul *"Resursă incompletă: Lipsesc fișierele x.lab, x.phs"*.

3.5 Concluzii

În acest raport au fost descrise trei procese de conversie automată a formatelor identificate pentru resursele bimodale disponibile, formate descrise în raportul asociat activității A1.2, la standardul convenit pentru platforma CoBiLiRo, descris în raportul corespunzător activității A1.3 [1]. De remarcat este și faptul că în urma observațiilor făcute de parteneri în procesul de implementare a platformei și a convertoarelor, standardul CoBiLiRo a suferit modificări, descrise pe scurt în acest raport. În cazul acestor conversii este necesară și completarea informațiilor disponibile în fișierele originale, în special în zona de header, unde informații precum autorul, sursa textului și a înregistrării audio sunt esențiale pentru o platformă de distribuire precum cea dezvoltată în cadrul proiectului. În cazul identificării ulterioare și a altor formate relevante, soluții similare de armonizare pot fi propuse și implementate.

Tot în acest raport sunt descrise soluțiile tehnice adoptate pe serverul CoBiLiRo, ce găzduiește resursele convertite, în vederea prelucrării rapide a unui volum mare de date, eventual incluse într-o arhivă. Pentru flexibilitate, în cazul apariției unor formate noi sau a unor situații ce necesită noi modificări ale standardului, fișierele originale sunt și ele păstrate pe server în eventualitatea unei conversii viitoare. Procesul de conversie s-a dovedit a fi rapid, în momentul elaborării acestui raport existând deja un volum semnificativ de resurse convertite pe serverul proiectului, resurse ce sunt descrise în raportul activității A2.4.

Toate obiectivele incluse în plan la această activitate au fost realizate.

Bibliografie

[1] Cristea, D., Scutelnicu A. (2018, noiembrie), Raport *"Activitatea A1.3: Proiectarea funcțională și arhitecturală a infrastructurii care va găzdui resursele și instrumentele de prelucrare și acces ale consorțiului și realizarea unui prototip"*, proiect RETEROM

[2] Pistol. I., Pădurariu C., Boghiu Ș., Scutelnicu A., Raport *"Activitatea A2.2: Raport asupra soluțiilor de armonizare a reprezentărilor colecțiilor existente text /vorbire (metadate și adnotări)",* proiect RETEROM