UNIVERSITATEA "POLITEHNICA" DIN BUCUREŞTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAŢII ŞI TEHNOLOGIA INFORMAŢIEI

SERVICIU WEB DE ANALIZĂ STATISTICĂ A PRESEI ONLINE. ANALIZA DATELOR, GENERAREA ŞI VIZUALIZAREA STATISTICILOR

PROIECT DE DIPLOMĂ

prezentat ca cerinţă parţială pentru obţinerea titlului de
Inginer în domeniul *Inginerie Electronică şi Telecomunicaţii*
programul de studii *Reţele şi Software de Telecomunicaţii*

Conducător ştiinţific:
Ş.L. Dr. Ing. Horia CUCU

Absolvent:
Alexandru TOMA

Bucureşti
2015

Anexa 1

Universitatea "Politehnica" din Bucuresti
Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei
Departamentul Telecomunicatii

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

TEMA PROIECTULUI DE DIPLOMA
a studentului Toma Alexandru, grupa 441 D

1. Titlul temei: *Serviciu web de analiza statistica a presei online. Analiza datelor, generarea si vizualizarea statisticilor.*

2. Contributia practica, originala a studentului va consta in:
Acest proiect de diploma isi propune realizarea conceptuala si implementarea unui serviciu web de achizitie si analiza a continutului media online, fiind constituit din 4 componente principale:

1) *Componenta ce vizeaza achizitia si preprocesarea continutului media – aceasta componenta a fost realizata ca parte a unui alt proiect;*
2) *Componenta de indexare a bazei de date - aceasta componenta a fost realizata ca parte a unui alt proiect;*
3) *Interfata grafica – se va realiza o interfata web prin intermediul careia vor fi vizualizate rezultatele analizei asupra continutului media achizitionat in prealabil;*
4) *Componenta ce faciliteaza cautarea in baza de date indexata – se va implementa un serviciu REST complet, cu rolul de a transfera structura de filtre si criterii formulata in pagina web catre modulul care realizeaza reprezentarea grafica a datelor.*

Studentul va descrie conceptual sistemul mai sus mentionat si va insista cu precadere asupra implementarii interfetei web si a serviciului REST asociat.

3. Proiectul se bazeaza pe cunostinte dobandite in principal la urmatoarele 3-4 discipline: *Programarea Calculatoarelor, Programare Obiect-Orientata, Tehnologii de Programare în Internet*

4. Proprietatea intelectuala asupra proiectului apartine: *Studentului, Conducatorului de lucrare*

5. Locul de desfasurare a activitatii: *UPB-ETTI, Sala 3 de Calculatoare*

6. Realizarea practica/ proiectul raman in proprietatea: *Studentului, UPB*

7. Data eliberarii temei: *1.06.2014*

CONDUCATOR LUCRARE: **STUDENT:**

S.L. Horia Cucu **Toma Alexandru**

DECLARAŢIE DE ONESTITATE ACADEMICĂ

Prin prezenta declar că lucrarea cu titlul "Serviciu web de analiză statistică a presei online. Analiza datelor, generarea şi vizualizarea statisticilor", prezentată în cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei a Universităţii "Politehnica" din Bucureşti ca cerinţă parţială pentru obţinerea titlului de *Inginer* în domeniul *Inginerie Electronică şi Telecomunicaţii*, programul de studii *Reţele şi Software de 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, 22.06.2015

Absolvent *Alexandru Toma*

_____

CUPRINS

LISTA FIGURILOR

LISTA TABELELOR

LISTA ACRONIMELOR

AJAX - Asynchronous JavaScript and XML

API - Application Program Interface

BOW - Bag of Words

CEO - Chief Executive Officer

CORBA - Common Object Request Broker Architecture

DOM - Document Object Model

HTML -HyperText Markup Language

HTTP - HyperText Transfer Protocol

IP - Internet Protocol

JDBC - Java DataBase Connectivity

JDK - Java Development Kit

JPEG - Joint Photographic Experts Group

JS - JavaScript

JVM - Java Virtual Machine

LSA - Latent Semantic Analysis

MVCC- Multiversion Concurrency Control

NLP - Natural Language Processing

PHP - PHP: Hypertext Preprocessor

REST - Representational State Transfer

RSS - Rich Site Summary

RMI - Remote-Method Invocation

SGML- Standard Generalized Markup Language

SO - PMI - Semantic Orientation - Pointwise Mutual Information

SpeeD - Speech and Dialogue

SQL - Structured Query Language

SVM - Support Vector Machine

TTL - Time to Live

TV - Television

URI - Uniform Resource Identifier

URL - Uniform Resource Locator

UTF-8 - Unicode Transformation Format (8-Bit)

WEB - World Electronic Base

WORA-Write Once, Run Anywhere

XML - EXtensible Markup Language

CAPITOLUL 1

INTRODUCERE

1.1 MOTIVAŢIA TEMEI

Premergător realizării prezentei lucrări de diplomă, am desfăşurat un studiu privitor la conţinutul media online, din care a rezultat că în mediul online există o cantitate însemnată de informaţie, dar care este utilizată un interval de timp relativ restrâns, după care interesul utilizatorului scade, iar informaţiile stocate în Internet nu se mai folosesc.

Studiul a fost realizat utilizând *Google Trends API*[1]. Acesta a presupus accesarea serviciului *Google Trends* oferit de către Google şi evidenţierea interesului manifestat de publicul larg cu privire la un anumit subiect, într-un interval temporal ales arbitrar.

În cadrul studiului am ales următoarele date de intrare:

- intervalul temporal vizat: Ianuarie 2014 – Ianuarie 2015;
- regiunea vizată: România;
- subiectul vizat: Simona Halep.

Google Trends este un serviciu web, proprietate a companiei Google, care pe baza motorului de căutare cu acelaşi nume, indică cât de des un anumit termen este căutat în raport cu volumul total de căutări în diferite regiuni ale lumii şi în diferite limbi. Pentru vizualizarea rezultatelor este utilizat un sistem de coordonate carteziene, în care abscisa reprezintă intervalul de timp

specificat, iar ordonata redă cât de des un termen este căutat în raport cu numărul total de căutari, la nivel global.

Având în vedere aceste caracteristici, *Google Trends* devine relevant în studiul tendinţelor existente la un moment dat în mediul online. Astfel, aplicând serviciul mai sus menţionat asupra datelor de intrare specificate, obţinem rezultatele reprezentate în Fig. 1.1.1, respectiv Fig. 1.1.2.

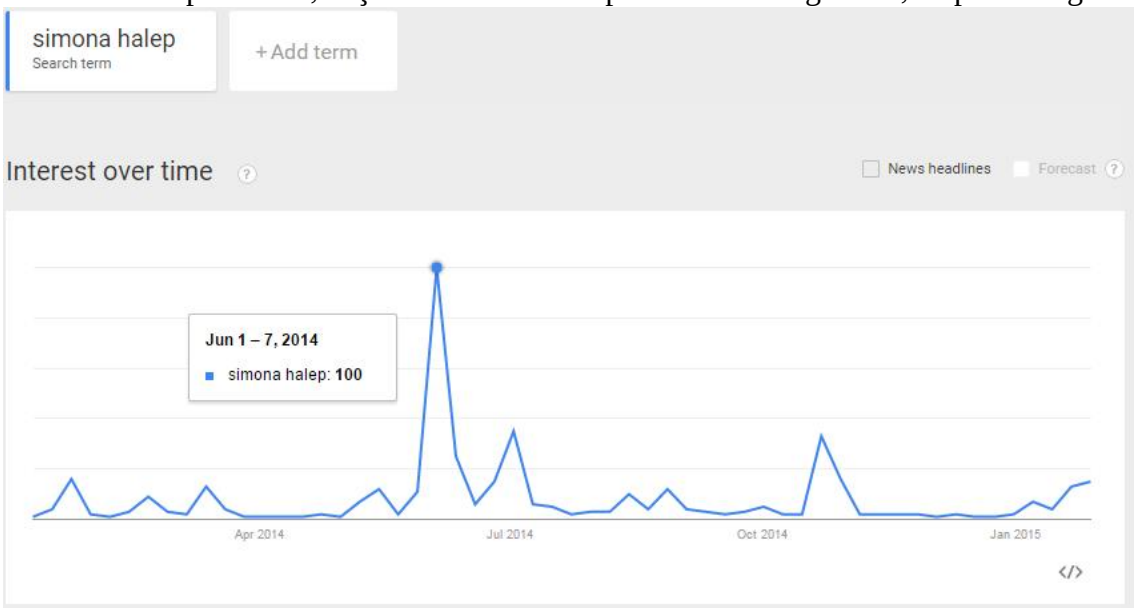

Fig. 1.1.1 Rezultatul studiului referitor la conţinutul media online (a)

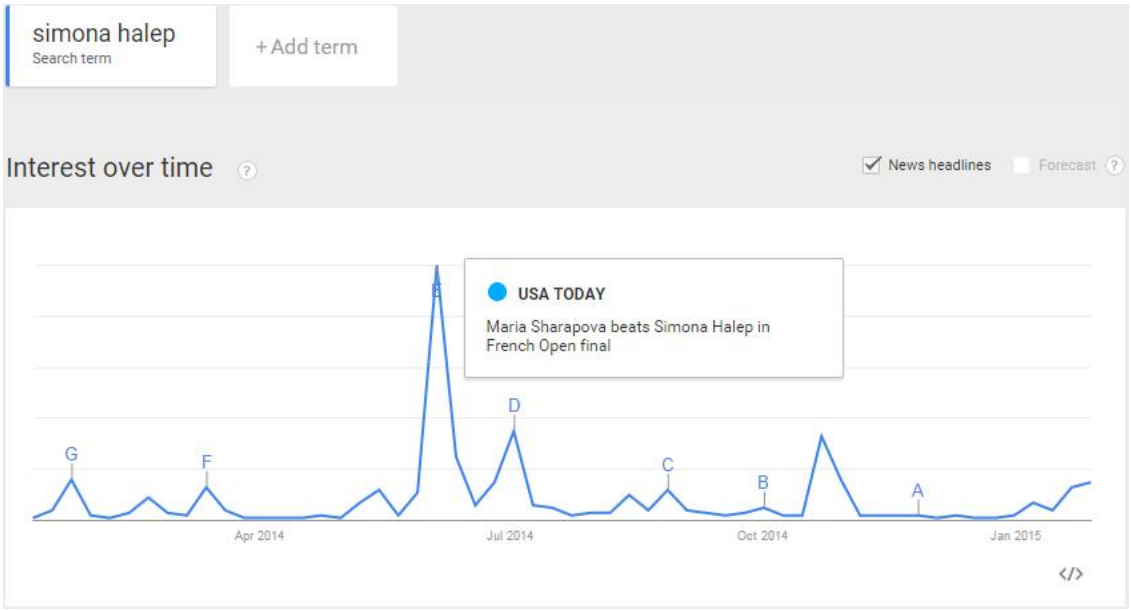

Fig. 1.1.2 Rezultatul studiului referitor la conţinutul media online (b)

Dacă luăm în considerare Fig. 1.1.1, se observă că punctul maxim de interes manifestat de către utilizatori privind subiectul *Simona Halep*, în intervalul temporal vizat, este înregistrat în perioada 1-7 iunie 2014, în care numărul de căutări ale termenului în raport cu volumul total de căutări în contextul regional al României, atinge valoarea limită superioară de 100. Depăşind această perioadă, se inregistrează un trend descendent până la următorul eveniment definitoriu, care o are în centru pe Simona Halep.

Pe de altă parte, activând opţiunea de a arăta cele mai importante titluri ale ştirilor în intervalul temporal vizat, *Google Trends* asociază fiecărui eveniment major, o ştire dintr-un cotidian important, realizând astfel o corelaţie între punctele maxime de interes, respectiv evenimente importante şi informaţiile referitoare la acestea, stocate în mediul online. Astfel, dacă privim Fig. 1.1.2, observăm că punctul maxim de interes mai sus menţionat este corelat cu un eveniment relatat într-un articol publicat online de către platforma de ştiri *USA Today*: *„Maria Sharapova beats Simona Halep in French Open Final”*.

Rezultatele studiului au evidenţiat comportamentul consumatorului de conţinut media online, dar şi faptul că, aplicând acest algoritm în care corelaţia dintre un eveniment important şi informaţiile referitoare la acesta se face prin asocierea în mod aleator a unei ştiri cu evenimentul respectiv, se ajunge în situaţia în care o cantitate însemnată de informaţie stocată online nu mai este folosită ulterior.

În acest context, este necesară realizarea conceptuală şi implementarea unui serviciu web de achiziţie şi analiză statistică a conţinutului media online, care să permită stocarea unei cantităţi însemnate de informaţie într-o bază de date şi analiza ulterioară a acesteia. Serviciul web şi-ar demonstra utilitatea în situaţia în care o companie ar vrea să vadă ce impact produce o campanie de marketing în presa scrisă online şi implicit cum influenţează aceasta interesele companiei (i.e., este vorba despre realizarea unor corelaţii între subiecte). De asemenea, acest sistem de analiză ar putea fi utilizat în mod subiectiv, atunci când de pildă un politician vrea să vadă cât de des este menţionat în articolele de ştiri, în campania electorală.

Se vizează realizarea unui sistem expus utilizatorului sub forma unui serviciu web ce poate fi accesat programatic, în sensul că un utilizator poate avea acces la acesta folosind diferite utilitare software. În cadrul acestei lucrări de diplomă s-a implementat o interfaţă web care exemplifică funcţionalitatea unui astfel de sistem.

1.2 OBIECTIVE

În contextul în care în mediul online există o însemnată cantitate de informaţie media, folosită un interval temporal determinat doar de comportamentul utilizatorului, este necesară realizarea conceptuală şi implementarea unui serviciu web de achiziţie şi analiză statistică a conţinutului media online, care să permită stocarea unei cantităţi însemnate de informaţie într-o bază de date şi analiza ulterioară a acesteia, ilustrând totodată conceptul de „big data”. „Big data” descrie seturi ample de date care pot fi analizate programatic, pentru a descoperi modele, tendinţe şi asocieri, în special cu privire la comportamentul uman şi la interacţiunile interumane.

Obiectivul central al acestei lucrări de diplomă este de a expune utilizatorului, prin intermediul unei interfeţe web, diferite opţiuni de analiză asupra conţinutului media achiziţionat în prealabil şi stocat într-o bază de date (i.e., vezi CAPITOLUL 4, subsecţiunea 4.1). Dintre acestea amintim: analiza sentimentelor, realizarea unor corelaţii, sau chiar metoda de counting.

De altfel, prezenta lucrare de diplomă îşi propune ca pe lângă realizarea conceptuală şi implementarea sistemului de analiză a presei scrise online, să asigure corpusul de text necesar în proiecte ulterioare, ce vizează dezvoltarea sistemelor de recunoastere automată a vorbirii. În corelaţie cu activitatea desfăşurată de laboratorul de cercetare *SpeeD* (*Speech and Dialogue*), analiza ar putea fi extinsă la presa audiovizuală. Astfel s-ar putea dezvolta un sistem care în loc să achiziţioneze corpus de text, ar face achiziţia şi analiza de conţinut audiovizual (i.e., articole de ştiri în format video şi/sau audio).

1.3 SUMARIZAREA LUCRĂRII DE DIPLOMĂ

În secţiunea curentă a acestei lucrări de diplomă a fost discutată motivaţia care a condus la realizarea acestei lucrări, dar şi principalele obiective urmărite.

Capitolul 2 urmăreşte introducerea unor noţiuni generice despre tehnologiile software utilizate în realizarea proiectului.

Capitolul 3 va fi oferi o privire de ansamblu asupra sistemului Media Analytics, în sensul că va fi prezentat modul în care sistemul de analiză este expus utilizatorului, arhitectura sistemului, dar şi componenta ce vizează achiziţia datelor.

Capitolul 4 urmăreşte să scoată în evidenţă contribuţiile personale ale studentului, prezentând în detaliu componenta ce vizează analiza datelor, generarea şi vizualizarea rezultatelor.

Capitolul 5 reprezintă ultima parte a lucrării de diplomă şi va cuprinde o subsecţiune de concluzii la care s-a ajuns în urma realizării proiectului de diplomă, o subsecţiune ce presupune rezumarea contribuţiilor personale ale studentului şi nu în ultimul rând, o parte în care sunt prezentate eventualele îmbunătăţiri şi funcţionalităţi ce pot fi adăugate ulterior sistemului.

CAPITOLUL 2

TEHNOLOGII SOFTWARE

2.1 JAVA

Java este un limbaj de programare de uz general, concurent, bazat pe clase, obiect-orientat şi special conceput astfel încât sa aibă cât mai puţine dependenţe de implementare. Acesta a fost menit să permită dezvoltatorilor de aplicaţii să scrie o singură dată codul, ca mai apoi să poată rula aplicaţia oriunde - este vorba despre conceptul WORA („write once, run anywhere”), ce presupune ca o aplicaţie Java al cărei cod a fost compilat în prealabil, să poată fi rulată pe orice platformă care suportă JVM (Java Virtual Machine), fără a fi nevoie de o recompilare a codului.

Prin caracterul său inovativ, Java aduce laolaltă proprietăţi prezente în alte limbaje de programare, dintre care amintim:

- obiect-orientat;
- portabil;
- securizat;
- capabil să ruleze mai multe fire de execuţie;
- aplicabil în Internet;
- uşor de utilizat.

În continuare, voi prezenta ce impact are fiecare dintre aceste proprietăţi în realizarea unui proiect software de mare anvergură.

Obiect-orientat

Multe limbaje pioniere, precum C şi Pascal, au fost limbaje de programare procedurale. Procedurile (adesea numite *funcţii*) erau blocuri de cod care făceau parte dintr-un modul sau dintr-o aplicaţie. Procedurile primeau parametri, de regulă tipuri primitive de date, cum ar fi tipuri întregi, caractere, şiruri de caractere, valori în virgulă mobilă. Deoarece codul era tratat separat faţă de date, apărea necesitatea utilizării structurilor de date, iar procedurile îşi puteau schimba oricând conţinutul. Aceasta a constituit o sursă de apariţie a problemelor, întrucât anumite proceduri puteau avea efecte neprevăzute asupra altor zone ale programului, iar depistarea procedurii care producea aceste probleme implica un consum semnificativ de timp, în special în cazul programelor mari.

Java este un limbaj de programare obiect-orientat. Un astfel de limbaj manipulează obiecte, care conţin atât date (variabile membru), cât şi cod (metode). Fiecare obiect aparţine unei anumite clase, care este o schiţă ce descrie ce variabile membru şi ce metode poate oferi acel obiect. În Java, aproape fiecare variabilă este un obiect de un anumit tip sau altul – de exemplu String.

În prezent există o serie de limbaje de programare orientate pe obiect. Unele dintre ele, cum ar fi Smalltalk sau Java sunt proiectate din start sa fie orientate pe obiect, spre deosebire de altele, cum ar fi C++, care este hibrid, parţial obiect-orientat şi parţial procedural. In C++ încă se pot suprascrie conţinutul structurilor de date şi obiectele, cauzând defectarea aplicaţiei. Java, în schimb, interzice accesul direct în memorie, ceea ce il face un sistem mult mai robust.

Portabil

Majoritatea limbajelor de programare sunt proiectate pentru un anumit tip de microprocesor, respectiv pentru un anumit sistem de operare. Atunci când codul sursă este compilat, este de fapt convertit în cod maşină, care poate fi executat doar pe un anumit tip de microprocesor, acest cod fiind extrem de rapid.

Pe de altă parte, există şi limbaje de programare care produc cod interpretabil. Codul este citit de o aplicaţie software (interpretorul), care realizează operaţii specifice asupra acestuia. De multe ori, codul interpretat nu are nevoie să fie compilat – acesta este translatat în cod masină pe masură ce este executat. Din acest motiv, codul interpretat este destul de lent, dar de multe ori portabil pe diferite sisteme de operare şi arhitecturi de microprocesor.

Java preia avantajele celor două tehnici, astfel că un cod Java este compilat într-un cod maşină independent de platforma de rulare, care este numit *Java bytecode*. Un interpretor special numit *Java Virtual Machine*, citeşte bytecode-ul şi îl procesează. Fig. 2.1.1 arată codul de asamblare al unei aplicaţii simple Java. În cazul acesta, bytecode-ul indicat prin săgeată este reprezentat sub formă de text, însă atunci când este compilat, va fi reprezentat sub formă de bytes, pentru a conserva spaţiul.

```
--- HelloWorld.java  ------------------------------------------------
1:    public class HelloWorld
2:    {
3:        public static void main(String args[])
4:        {
5:            System.out.println ("Hello World");
➪ 00000000   getstatic                  java/lang/System.out
  00000003   ldc                        "Hello World"
  00000005   invokevirtual              java/io/PrintStream.println
6:        }                             (java/lang/String)
  00000008   return
7:    }
```

Fig. 2.1.1 Bytecode-ul în limbaj de asamblare pentru „Hello World”

Modul de abordare al Java oferă anumite avantaje spre deosebire de alte limbaje de programare care produc cod interpretabil. În primul rând, codul sursă este protejat, astfel că nu poate fi văzut sau modificat – doar bytecode-ul este vizibil utilizatorilor. În al doilea rând, mecanismele de securitate pot scana bytecode-ul în vederea verificării modificărilor survenite sau dacă codul este malițios, complementând astfel celelalte mecanisme de securitate ale Java. În plus, un cod sursă Java poate fi compilat o singura data, dupa care poate rula pe orice sistem ce suporta JVM (i.e., este vorba despre principiul „write once, run anywhere"). O aplicaţie este portabilă atunci când nu este necesară decât scrierea acesteia o singură dată, dupa care poate fi rulată pe o gamă largă de sisteme.

Securizat

Deoarece applet-urile Java sunt descărcate de la distanţă şi executate într-un browser, securitatea datelor este de mare interes în Java. Bineînţeles că nu este de dorit ca applet-urile să acceseze documentele personale sau să şteargă fişierele utilizatorilor.

La nivel de API există restricţii puternice de securitate asupra applet-urilor în ceea ce priveşte manipularea fişierelor şi accesul în reţea, precum şi suport pentru semnături digitale pentru a verifica integritatea codului descărcat, iar la nivel de bytecode se fac verificări evidente privitoare la acţiuni maliţioase, cum ar fi manipularea stivei sau bytecode invalid.

În Java, mecanismele de securitate asigură protecţia împotriva încălcărilor accidentale sau intenţionate de securitate, dar este important de precizat că niciun sistem nu este perfect. Cea mai slabă verigă din lanţul sistemului este reprezentată de JVM, deoarece o maşină virtuală cu deficienţe de securitate cunoscute este predispusă la atacuri informatice.

Capabil să ruleze mai multe fire de execuţie

În C există conceptul de procese multiple. O aplicaţie se poate împărţi în mai multe copii separate, care rulează simultan. Fiecare copie reproduce codul şi datele, rezultând în creşterea consumului de memorie, iar crearea fiecărui proces implică un apel către sistemul de operare, care consumă deopotrivă ciclii suplimentari.

O abordare mult mai bună este de a utiliza mai multe fire de execuţie, denumite pe scurt threaduri, care folosesc mai puţină memorie şi totodată solicită mai puţin microprocesorul. Java are suport pentru multiple fire de execuţie, implementat chiar în limbaj. Acest suport este uşor de folosit în Java, iar implementarea de threaduri în aplicaţii şi applet-uri este destul de uzuală.

Aplicabil în Internet

Java a fost proiectat să susţină programarea în Internet. Java API asigură suport extins pentru reţea, de la socket-uri şi adrese IP, până la URL(Uniform Resource Locator) şi HTTP (Hyper Text Transfer Protocol). Dezvoltarea unei aplicaţii web în Java este relativ simplă, iar codul este complet portabil între diferite platforme de lucru.

Java include de asemenea suport pentru programare mai complexă în Internet, cum ar fi Remote-method invocation (RMI), CORBA şi Jini. Aceste tehnologii de sisteme distribuite fac din Java o alegere atractivă pentru sistemele distribuite de mare amploare.

Uşor de utilizat

Java a fost dezvoltat pornind de la limbajul C++, considerat a fi un limbaj de programare complex, cu caracteristici precum moştenirea multiplă, şabloane şi pointeri, care sunt elemente contraproductive. Java, pe de altă parte, este mai apropiat de un limbaj obiect-orientat „pur". Accesul la pointeri de memorie a fost eliminat, utilizând în schimb referinţe către obiecte. De asemenea suportul pentru moştenirea multiplă a fost îndepărtat, ceea ce implică o viziune mult mai clară în ceea ce priveşte realizarea şi utilizarea claselor.

În contextul Java, interfaţa de programare a aplicaţiilor (API), este o colecţie predefinită de pachete, clase şi interfeţe cu metodele, câmpurile şi constructorii acestora. Similar unei interfeţe cu utilizatorul, ce facilitează interacţiunea dintre om şi computer, un API poate fi considerat o interfaţă software care facilitează interacţiunea.

În Java, cele mai comune sarcini de programare sunt realizate prin intermediul claselor şi pachetelor din API, care sunt utile în minimizarea numărului de linii scrise în porţiunile de cod.

Java Development Kit (JDK) este format din trei componente de bază, după cum urmează:

- Compilatorul Java;
- Maşina virtuală Java (JVM);
- Interfaţa de programare a aplicaţiilor în Java (API).

Interfaţa de programare a aplicaţiilor în Java, inclusă în JDK, descrie funcţia fiecărei componente constituente – o mare parte dintre aceste componente sunt predefinite şi utilizate frecvent. Astfel, programatorul este capabil să utilizeze codul prescris, prin intermediul API-ului Java. După referirea claselor şi pachetelor disponibile din API, programatorul poate apela foarte uşor codul asociat acestora, necesar implementării.[23]

În afara faptului că API-urile ajută programatorii să determine funcţiile claselor şi pachetelor, prin intermediul acestora pot fi determinaţi parametri sau alte informaţii necesare implementării codului Java predefinit în cadrul acelor API.

Sumarizând cele prezentate în această sub-secţiune, Java oferă dezvoltatorilor software multe avantaje. În timp ce o mare parte dintre acestea sunt prezente în alte limbaje de programare, Java le combină într-un singur limbaj, îndeplinind toate criteriile necesare implementării unui proiect software complex.

2.2 WEB CRAWLERS, RSS FEEDS

Un crawler web este o aplicaţie software independentă care navighează sistematic prin Internet, de obicei cu scopul de a indexa paginile web. El mai este numit şi "Web spider", "Automatic Indexer" sau "Web Scutter". Motoarele de căutare utilizează un astfel de serviciu pentru a-şi actualiza conţinutul din baza de date şi pentru actualizarea indecşilor unor site-uri web. De asemenea crawler-ele pot fi folosite pentru validarea ancorelor web sau a codului HTML.

Un crawler web porneşte de obicei de la o listă cu legături ale unor pagini web pe care trebuie să le viziteze. Pe măsură ce crawler-ul vizitează o pagină web, identifică toate legăturile din acea pagină, şi le adaugă în lista de legături pe care urmează să le viziteze. Paginile pot fi vizitate recursiv după un anumit set de reguli.

Un volum mare de date implică o limitare a crawler-ului la a descărca un număr limitat de pagini web într-un anumit timp, aşa că el este nevoit să prioritizeze ordinea de parsare şi descărcare a paginilor web. De asemenea generarea legăturilor pe partea de server aduce o nouă dificultate crawler-elor web, deoarece acestea pot ajunge să descarce pagini duplicate.[4]

Prin urmare, un crawler realizează „harta” unui site web, punând la dispoziţia utilizatorului, sub forma unui arbore ierarhic, toate legăturile către paginile ce intră în componenţa site-ului web. Totuşi, această abordare nu este potrivită atunci când se doreşte automatizarea unui proces de achiziţie, deoarece un crawler web nu oferă opţiuni de condiţionare a introducerii de conţinut nou în baza de date, rezultând la fiecare rulare ulterioară a procesului, conţinut duplicat (i.e., vezi CAPITOLUL 3, subsecţiunea 3.3).

Pe de altă parte, RSS (Rich Site Summary) este o familie de formate de fluxuri web, realizate în format XML şi folosite pentru Web Syndication. RSS este folosit (printre altele) pentru ştiri, weblog-uri şi podcasting.

Web feed-urile oferă conţinut web sau sumare de conţinuturi web, împreună cu legături către conţinutul complet al respectivei surse de informaţii şi alte metadate. RSS oferă această informaţie sub forma unui fişier XML, numit *feed RSS*, *webfeed*, *stream RSS* sau *canal RSS*.

În plus, față de facilitarea partajării ştirilor (termen american: "syndication"), feed-urile web permit cititorilor fideli anumitor pagini să fie informați la actualizarea conținutului de pe aceste pagini web, prin folosirea unui soft special numit *aggregator*.

În timp ce partea cea mai importantă a mass-mediei încă încearcă să înțeleagă potențialul RSS, redactorii de ştiri folosesc RSS ca să ocolească sursele de ştiri tradiționale. Prin sistemul RSS, utilizatorii și jurnaliștii au la dispoziție surse constante de ştiri, fără să mai fie nevoiți să petreacă timp căutând.[5]

Din punct de vedere programatic, informaţiile structurate sub forma unui fişier XML pot fi extrase relativ uşor printr-un proces de parsare a fişierului, folosind un API Java specializat. Acest subiect va fi detaliat în secţiunea care vizează achiziţia datelor.

2.3 BAZE DE DATE, POSTGRESQL

În sensul cel mai larg, o bază de date reprezintă o colecţie de date corelate din punct de vedere logic, care reflectă un anumit aspect al lumii reale şi este destinată unui anumit grup de utilizatori. În acest sens, bazele de date pot fi create şi menţinute manual (e.g, fişele de evidenţă a cărţilor dintr-o bibliotecă, aşa cum erau folosite cu ani în urmă) sau computerizat, aşa cum sunt majoritatea bazelor de date folosite în momentul de faţă. O definiţie într-un sens mai restrâns a unei baze de date este următoarea:

O bază de date este o colecţie de date creată şi menţinută computerizat, care permite operaţii de introducere, ştergere, actualizare şi interogare a datelor.

Faţă de vechile metode de înregistrare a datelor privind diferite activităţi pe documente scrise sau chiar în fişiere pe disc, sistemele de baze de date oferă avantaje considerabile, ceea ce explică utilizarea extinsă a acestora. Câteva dintre avantajele oferite sunt prezentate în continuare.

- Compactitate ridicată: volumul ocupat de sistemele de baze de date este mult mai redus decât volumul ocupat de documente scrise sau de fişiere necorelate.
- Viteză mare de regăsire şi actualizare a informaţiilor.
- Redundanţa scăzută a datelor memorate, care se obţine prin partajarea datelor între mai mulţi utilizatori şi aplicaţii. În stocarea pe fişe sau în fişiere a datelor, fiecare aplicaţie conţinea propriile seturi de date. În sistemele de baze de date, mai multe aplicaţii pot folosi date comune, memorate o singură dată. De exemplu, o aplicaţie de personal şi o aplicaţie de rezultate la examene dintr-o universitate care exploatează o singură bază de date, pot folosi aceleaşi informaţii referitoare la structurarea facultăţilor şi a secţiilor.
- Posibilitatea de introducere a standardelor privind modul de stocare a datelor, ceea ce permite interschimbul informaţiilor între diferite organizaţii.
- Menţinerea integrităţii datelor prin politica de securitate (drepturi de acces diferenţiate în funcţie de rolul utilizatorilor), prin gestionarea tranzacţiilor şi prin refacerea datelor în caz de funcţionare defectuoasă a diferitelor componente hardware sau software.

- Independenţa datelor faţă de suportul hardware utilizat: sistemele de gestiune a bazelor de date oferă o vedere externă a datelor, care nu se modifică atunci când se schimbă suportul de memorare fizic, ceea ce asigură imunitatea structurii bazei de date şi a aplicaţiilor la modificări ale sistemului hardware utilizat.[6]

PostgreSQL este un sistem de gestionare a bazelor de date relaţionale, care pune accent pe extensibilitate şi asigurarea conformităţii cu standardele. Ca server de baze de date, funcţia sa principală este de a stoca datele în siguranţă şi de a pune la dispoziţia altor aplicaţii software aceste date, atunci când sunt formulate cereri în această privinţă. Poate trata sarcini de lucru, de la aplicaţii ce vizează un singur computer, până la aplicaţii ample, care rulează în Internet şi implică accesarea concurentă de către mai mulţi utilizatori a aceleaşi baze de date.

PostgreSQL implementează majoritatea standardelor SQL:2011 şi oferă o serie de avantaje care vin în sprijinul dezvoltatorilor software, în detrimentul altor sisteme de gestionare a bazelor de date.

Fiind un sistem open-source de gestionare a bazelor de date relaţionale, PostgreSQL nu este controlat de nicio corporaţie sau orice altă entitate privată, punând la dispoziţia utilizatorilor codul sursă al sistemului. Prin aceasta, PostgreSQL preîntâmpină problemele legate de supra-implementare vizate de unii furnizori de baze de date proprietare şi permite dezvoltatorilor să realizeze cercetări de concept şi numeroase implementări provizorii, fără a impune costuri de acordare a licenţei software.

De asemenea, acest sistem de gestionare a bazelor de date oferă fiabilitate şi stabilitate, acesta fiind un avantaj major atunci când se doreşte implementarea unor soluţii software expuse utilizatorilor sub forma unui serviciu. În plus, Postgres este aplicabil tuturor platformelor (i.e., este cross-platform), fiind nativ compatibil cu aproape orice sistem Unix şi Windows şi este conceput pentru medii de lucru ce implică un volum mare de date (i.e., foloseşte o strategie de stocare a datelor bazată pe versionarea înregistrărilor pentru evitarea problemelor de concurenţă, numită MVCC – Multiversion Concurrency Control).[78]

2.4 SERVICII DE INDEXARE A DATELOR

Conceptul de indexare folosit pentru căutarea datelor este definit ca fiind procesul de adnotare a informaţiei existente într-o colecţie de date, prin adăugarea de informaţii suplimentare relative la conţinutul acesteia, informaţii numite şi indici de conţinut. Această etapă este necesară accesării colecţiei de date, deoarece permite catalogarea automată în funcţie de conţinut a datelor.[9]

Aşadar, un index dintr-o bază de date reprezintă o structură de date care sporeşte viteza operaţiilor de interogare cu scopul de a recupera datele, în detrimentul creşterii capacităţii de stocare necesară menţinerii structurii de indecşi.

Sistemele de gestionare a bazelor de date relaţionale precum MySQL, dispun de un serviciu de indexare a datelor de tip „Full Text”. Acesta este rapid, însă nu este la fel de uşor de configurat precum librăriile dedicate, compatibile Java (e.g., Apache Lucene).

Apache Lucene este o librărie software open source, care oferă servicii de extragere a datelor dintr-o colecţie de date (i.e., indexare, căutare, ierarhizare a rezultatelor), scrisă iniţial în Java de către Doug Cutting şi adaptată ulterior altor limbaje de programare, inclusiv C#, C++, Python, Ruby, PHP, Perl şi Delphi. În timp, s-a dovedit a fi adecvată pentru orice aplicaţie care necesită implementarea unui serviciu de indexare şi căutare într-o bază de date. Lucene a fost recunoscut pentru utilitatea sa în implementarea motoarelor de căutare în Internet, dar şi local, pe un singur site.[10]

Elementele fundamentale care stau la baza Lucene sunt: indexul, documentul, câmpul şi termenul.

- Un index conţine o secvenţă de documente.
- Un document conţine o secvenţă de câmpuri.
- Un câmp conţine o secvenţă de termeni.
- Un termen este un şir de caractere.

În momentul în care se doreşte indexarea datelor, este creat un index ce va fi populat cu elemente de tip document. Elementele de tip document vor conţine câmpurile după care se indexează colecţia de date, denumite arbitrar de către utilizator şi având în componenţă elemente de tip text: pereche de şiruri de caractere, primul fiind numele câmpului, iar cel de-al doilea textul cuprins în câmpul respectiv – astfel sunt luaţi în evidenţă termenii asupra cărora se vor realiza cereri de căutare.

Indecşii Lucene se încadrează în familia de indecşi cunoscuţi ca „indecşi inversaţi”. Acest lucru se datorează faptului că indecşii pot lista, pentru un anumit termen, documentele care îl conţin, iar această abordare este inversă relaţiei naturale, în care documentele listează termeni, tocmai pentru a accelera procesul de căutare într-o bază de date indexată.

În contextul prezentei lucrări de diplomă, Lucene s-a dovedit a fi adecvat procesului de indexare desfăşurat asupra bazei de date populate în prealabil cu conţinut media. Aceasta datorită flexibilităţii în implementare şi caracteristicilor sale particulare. În schimb, menţinerea indecşilor unei astfel de baze de date implică duplicarea datelor şi stocarea acestora pe disc, ceea ce presupune asigurarea unui spaţiu de stocare considerabil şi reprezintă dezavantajul major al unui astfel de serviciu.

2.5 XML, HTML, JS, REST

XML (Extensible Markup Language), descendent al SGML (Standard Generalized Markup Language) este un meta-limbaj utilizat în activitatea de marcare structurală a documentelor. Acesta defineşte un set de reguli pentru crearea formatelor text ce permit structurarea datelor, scopul său primar fiind de a pune accent pe simplitate, universalitate şi uşurinţa utilizării sale în Internet. Deşi XML se concentrează în principal pe documente, acesta este utilizat pe scară largă pentru reprezentarea structurilor de date arbitrare, cum ar fi cele utilizate în serviciile web.

Un document XML este format din *marcaje (tag-uri)* şi date caracter şi este structurat sub forma unui arbore ierarhic. Cuvântul *marcaj (markup)* a fost folosit iniţial pentru a descrie anumite adnotări (i.e., note marginale în cadrul unui text) cu intenţia de a indica tehnoredactorului cum trebuie listat un anumit pasaj. Generalizând, marcajul poate fi definit drept orice acţiune de a interpreta explicit o porţiune de text. Un *marcaj (tag)* este un şir de caractere delimitat de caracterele "<" şi ">" (e.g., nodul rădăcină, descendenţii acestuia, atributele fiecărui element). *Datele caracter* reprezintă conţinutul marcajelor.[11] Un exemplu al utilizării marcajelor în cadrul unui fişier XML poate fi vizualizat în Fig. 2.5.1, respectiv Fig. 2.5.2.

Orice aplicaţie XML derivă dintr-un fişier text şi poate fi creată foarte simplu utilizând orice editor de text, în care se include marcajul specific XML: `<?xml version='2.0' encoding='utf-8'?>` pentru a specifica tipul documentului, precum şi marcajul ce defineşte arbitrar nodul rădăcină: `<news-counts> ... </news-counts>` sau `<metadata> ... </metadata>`, conform exemplelor prezentate în Fig. 2.5.1, respectiv în Fig. 2.5.2.

Pe de altă parte, dacă se doreşte dezvoltarea unei aplicaţii ce necesită structurarea şi actualizarea permanentă a datelor, dacă se ia în considerare folosirea XML, atunci utilizarea editoarelor de text nu mai reprezintă o soluţie viabilă. Există în acest scop utilitare Java capabile să creeze, să actualizeze şi să parcurgă documentele XML pentru a oferi rezultate concludente utilizatorului

(e.g., XML DOM Parser, Jsoup). În informatică, parcurgerea şi analizarea unui text, cu identificarea atomilor care îi corespund, în raport cu o gramatică formală poartă numele de *parsare*. Un parser este o componentă a unui interpretor sau compilator, în cadrul cărora identifică structura textului de intrare şi o aduce într-o formă potrivită pentru prelucrări ulterioare. În conjucţie cu utilizarea formatului XML, prin parsare pot fi extrase sub formă de text valorile atributelor aflate sub incidenţa anumitor marcaje. Dacă considerăm ca exemplu documentul XML prezentat în Fig. 2.5.2, pot fi extrase prin parsare valorile atributelor *count*, respectiv *date*, ce intră sub incidenţa marcajului `<item></item>` şi mai departe valorile atributelor *link* şi *title* aflate sub incidenţa marcajului `<news></news>`.

Bineînţeles, folosind astfel de utilitare, se poate pleca de la proiectarea unui model simplu de document XML, conform Fig. 2.5.1 şi se poate ajunge la forme complexe ale documentului XML, conform Fig. 2.5.2.

```
▼<news-counts>
    <item count="4" date="2012.01.01"/>
    <item count="2" date="2012.01.02"/>
    <item count="1" date="2012.01.03"/>
    <item count="1" date="2012.01.04"/>
    <item count="3" date="2012.01.05"/>
    <item count="1" date="2012.01.06"/>
    <item count="1" date="2012.01.07"/>
    <item count="2" date="2012.01.08"/>
    <item count="2" date="2012.01.09"/>
```

Fig. 2.5.1 Rezultatul realizării unui document XML, utilizând Jsoup (1)

```
▼<metadata>
  ▼<item count="6" date="2015.01.01">
     <news link="" title="ANALIZĂ 2015, anul de foc pentru renegocierea redevențelor la petrol și gaze naturale. Guvernul a
     făcut deja mai multe concesii companiilor din sector"/>
     <news link="" title="În scandalul plagiatului lui Ponta, așteptăm reacția președintelui Iohannis"/>
     <news link="" title="Cum a ajuns Daniel Constantin de la favoritul lui Voiculescu la indezirabil"/>
     <news link="" title="Ediție specială Adevărul Live cu Ioan Talpeș: „România a fost tratată aproape ca un inamic de
     Rusia, încă din anii '90”"/>
     <news link="" title="Scrisoarea dură a unor clujeni, despre „Ponta copy-paste”: „Plagiatul nu este guturaiul unor
     indivizi afectați de  grandomanie, ci e cancer“"/>
     <news link="" title="Noduri în papură"/>
   </item>
  ▼<item count="7" date="2015.01.02">
     <news link="" title="Președintele Iohannis participă duminică la liturghia Catedralei Mitropolitane Ortodoxe din
     Sibiu"/>
     <news link="" title="O renunțare și mai multe acceptări"/>
     <news link="" title="SONDAJ IRES Cât cred românii în reforma PSD"/>
     <news link="" title="O nouă formațiune politică: Partidul Facebook"/>
     <news link="" title="Caz umanitar. Povestea Alinei, femeia care își crește băiețelul de doi ani într-o bucătărie"/>
     <news link="" title="Cătălin Predoiu, la Adevarul Live: Nu există tensiuni între mine și președintele Iohannis"/>
     <news link="" title="Ce restanțe are statul în domeniul energetic pentru 2015: fuziunea Elcen-RADET, negocierile pentru
     reactoare, definitivarea legislației la regenerabile"/>
   </item>
```

Fig. 2.5.2 Rezultatul realizării unui document XML, utilizând Jsoup (2)

HTML (HyperText Markup Language) este o formă de marcare orientată către prezentarea documentelor text pe o singura pagină, utilizând un software de redare specializat, numit *agent utilizator HTML*, cel mai bun exemplu de astfel de software fiind *browserul web*. HTML furnizează mijloacele prin care conținutul unui document poate fi adnotat cu diverse tipuri de metadate și indicații de redare. Indicațiile de redare pot varia de la decorațiuni minore ale textului, cum ar fi specificarea faptului că un anumit cuvânt trebuie subliniat sau că o imagine trebuie introdusă, până la scripturi sofisticate, hărți de imagini și formulare. Metadatele pot include informații despre titlul și autorul documentului, informații structurale despre cum este împărțit documentul în diferite segmente, paragrafe, liste, titluri etc. și informații cruciale care permit ca documentul să poată fi legat de alte documente pentru a forma astfel hiperlink-uri (sau web-ul).

Ca şi în cazul XML, un document HTML poate fi citit și editat utilizând un editor de text simplu. Totuşi scrierea și modificarea paginilor în acest fel solicită cunoștințe solide de HTML și este consumatoare de timp. Editoarele grafice cum ar fi *Macromedia Dreamweaver*, *Adobe GoLive* sau *Microsoft FrontPage* permit ca paginile web sa fie tratate asemănător cu documetele Word, dar cu observația că aceste programe generează un cod HTML care este de multe ori de proastă calitate.

Componența unui document HTML este:

1. versiunea HTML a documentului
2. zona *head* cu etichetele `<head></head>`
3. zona *body* cu etichetele `<body><body>` sau `<frameset></frameset>`

Toate paginile HTML încep şi se termină cu etichetele `<html>` şi `</html>`. În interiorul acestor etichete găsim perechile `<head>`, `</head>` şi `<body>`, `</body>`. `head` conține titlul paginii între etichetele `<title>` şi `</title>`, descrieri de tip `<meta>`, stiluri pentru formatarea textului, script-uri şi legături către fisiere externe (de exemplu script-uri, fişiere de tip CSS sau *favicon*). Etichetele de tip *meta* conțin cuvinte cheie, descrierea paginii, date despre autor, informații utile motoarelor de căutare și au următorul format: `<META NAME="nume" CONTENT="continut">`, iar `body` găzduiește practic toate etichetele afișate de browser pe ecran.

Elementele de marcare în HTML pot descrie o marcare structurală (e.g., `<h1></h1>`), o marcare pentru prezentare (e.g., `<strong></strong>`), marcare pentru hyperlink sau pot descrie elemente speciale (i.e., widget).[12]

În contextul realizării sistemului de analiză a presei scrise online, utilizarea formatului HTML a vizat atât extragerea integrală a textului şi a metadatelor (e.g., autor, categorie) din paginile HTML ale ştirilor, folosind pentru parsare utilitarul Jsoup, cât şi crearea paginii web a aplicaţiei folosind NetBeans şi descrierea manuală a documentului HTML.

JS (JavaScript) este un limbaj de programare obiect orientat, bazat pe conceptul prototipurilor. Este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul Javascript din aceste pagini fiind rulat de către browser. Limbajul este binecunoscut pentru folosirea sa în construirea siturilor web, dar este folosit și pentru accesul la obiecte încastrate (embedded objects) în alte aplicații.

Cea mai des întâlnită utilizare a JavaScript este în scriptarea paginilor web. Programatorii web pot îngloba în paginile HTML script-uri pentru diverse activități cum ar fi verificarea datelor introduse de utilizatori sau crearea de meniuri și alte efecte animate.

Browserele reţin în memorie o reprezentare a unei pagini web sub forma unui arbore de obiecte și pun la dispoziție aceste obiecte script-urilor JavaScript, care le pot citi şi manipula. Arborele de obiecte poartă numele de *Document Object Model* sau *DOM*. Există un standard W3C pentru DOM-ul pe care trebuie să îl pună la dispoziție un browser, ceea ce oferă

premiza scrierii de script-uri portabile, care să funcționeze pe toate browserele. În practică, însă, standardul W3C pentru DOM este incomplet implementat. Deși tendința browserelor este de a se alinia standardului W3C, unele din acestea încă prezintă incompatibilități majore, cum este cazul Internet Explorer.

O tehnică de construire a paginilor web tot mai întâlnită în ultimul timp este AJAX, abreviere de la „Asynchronous JavaScript and XML". Această tehnică constă în executarea de cereri HTTP în fundal, fără a reîncărca toată pagina web, și actualizarea numai anumitor porțiuni ale paginii prin manipularea DOM-ului paginii. Tehnica AJAX permite construirea unor interfețe web cu timp de răspuns mic, întrucît operația (costisitoare ca timp) de încărcare a unei pagini HTML complete este în mare parte eliminată.[13]

REST este acronimul de la *Representational State Transfer* şi reprezintă un model architectural pentru crearea serviciilor web.

Serviciile web de tip REST folosesc toate componentele ce au făcut web-ul un mare succes, aplicând arhitectura web-ului asupra serviciilor web. Deşi REST nu este un standard, el se foloseşte de standarde (e.g., HTTP, URI – Uniform Resource Identifier, XML/HTML/JPEG). Practic REST presupune construirea unui serviciu web folosind HTTP, XML şi URI aşa cum a fost construit şi web-ul.

În REST este vorba despre arhitectură, nu despre design. REST este un set de reguli la care o arhitectură ar trebui să se conformeze. Pentru că nu este suficient de detaliat pentru a defini o arhitectură reală vom spune despre REST că este un stil arhitectural. Web-ul este un exemplu real de arhitectură care urmează în linii mari regulile REST.

O arhitectură este construită din diferite componente proiectate (design pieces). Aceste componente interacţionează unele cu altele, în diverse moduri. Orice componentă poate trimite un mesaj la orice altă componentă iar regulile referitoare la semnificaţia mesajului sau felul în care mesajul este tratat să rămână la latitudinea emiţătorului şi receptorului.

Într-o arhitectură REST se deosebesc două trăsături importante :

- datele asupra cărora clientul îi spune serverului să opereze se află în URI
- operaţia pe care serverul o face asupra datelor este descrisă direct de metoda HTTP

REST este o altă abordare a serviciilor web. Din punct de vedere tehnic, arhitectura de tip REST se descrie prin câteva noţiuni de bază : resursă, URI, reprezentare, interfaţă uniformă. Într-o arhitectură de tip REST totul este o resursă. Orice poate fi referit ca un obiect este o resursă. În general tot ce poate fi stocat în calculator şi reprezentat ca un flux de octeţi este o resursă. Legarea acestor componente într-un mod cât mai eficient duce la crearea unei arhitecturi REST. O imagine de ansamblu asupra serviciilor de tip REST este prezentată în diagrama de mai jos:[14]

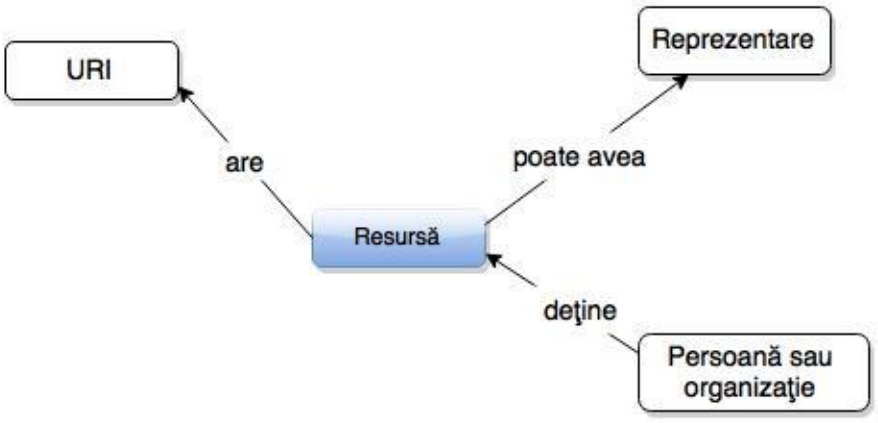

Fig. 2.5.3 Modelul conceptual REST

REST a fost utilizat în conjucţie cu JavaScript pentru implementarea funcţiei de analiză a sistemului descris în această lucrare de diplomă.

CAPITOLUL 3

O PRIVIRE DE ANSAMBLU ASUPRA SISTEMULUI. ACHIZIŢIA DATELOR

Având în vedere motivaţia temei de diplomă, precum şi obiectivele prezentate în CAPITOLUL 1, pentru a spori gradul de utilizare a conţinutului media online, este necesară realizarea conceptuală şi implementarea unui serviciu web de achiziţie şi analiză statistică a acestui tip de informaţie. În continuare, vom numi acest serviciu web: *Media Analytics*.

3.1 FUNCŢIONALITATEA SISTEMULUI

Media Analytics este expus utilizatorului sub forma unei pagini web (i.e., aplicaţie web), care realizează interfaţa dintre acesta şi componenta de analiză. Această pagină poate fi vizualizată prin accesarea linkului următor: http://dev.speed.pub.ro:10082/MediaAnalytics-Search/ .

Pentru exemplificarea funcţiei de analiză asupra datelor, utilizatorul are la dispoziţie o fereastră de căutare care conţine o serie de opţiuni pentru filtrarea datelor de intrare: selectarea portalului de ştiri, selectarea categoriei, dar şi a intervalului temporal pentru care se doreşte analiza datelor.

În momentul de faţă, utilizatorul serviciului are la dispoziţie trei portaluri de ştiri asupra cărora se poate desfăşura procesul de analiză: Adevărul.ro, Wall-Street.ro, respectiv 9AM.ro. De

asemenea, pentru fiecare ziar în parte, au fost extrase categoriile, astfel încât să fie posibilă aplicarea de filtre şi pe acestea.

Bineînţeles, dacă se doreşte analiza asupra tuturor ziarelor, respectiv tuturor categoriilor, aplicaţia web pune la dispoziţia utilizatorului opţiunea de a selecta toate ziarele, respectiv toate categoriile.

Intervalul temporal asupra căruia poate fi desfăşurată analiza datelor nu depinde de data publicării articolelor achiziţionate în prealabil, însă pentru obţinerea unor rezultate concludente este de preferat să avem în vedere ceea ce se află stocat în baza de date. În momentul de faţă, baza de date conţine articole publicate în cele trei ziare, începând cu data de 1 ianuarie 2014. Baza de date este actualizată constant, prin intermediul funcţiei de achiziţie automată a articolelor. Fig. 3.1.1 prezină pagina de start a aplicaţiei web, iar Fig. 3.1.2 prezintă modul de selecţie şi de aplicare a filtrelor.

Fig. 3.1.1 Pagina de start a aplicaţiei web

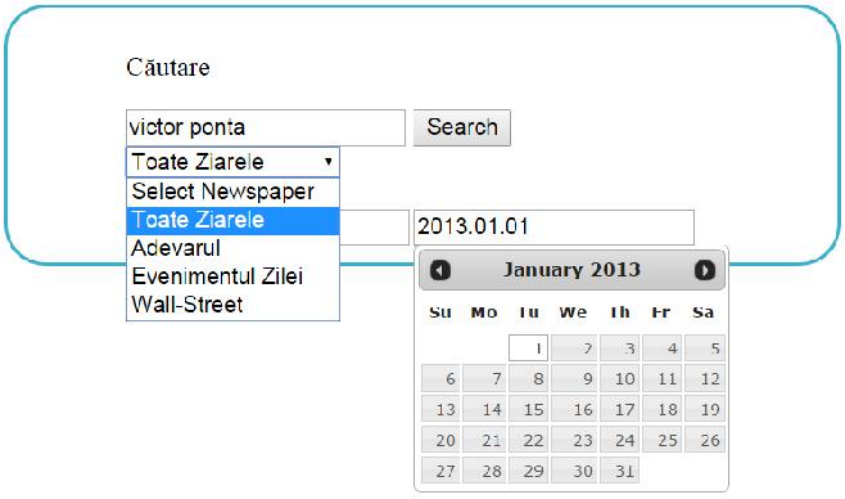

Fig. 3.1.2 Modul de selecţie şi de aplicare a filtrelor

Odată ce utilizatorul a creat structura de filtre dorită, selectând în primul rând termenul/termenii căutaţi, apoi portalul/portalurile de ştiri, categoria şi intervalul de timp vizat, poate lansa cererea către serviciul de analiză a presei scrise online prin apăsarea „butonului” *Search*. Serverul va întoarce în pagină rezultatul cererii, care va fi afişat în cadrul unui grafic. Pentru acest grafic, abscisa reprezintă intervalul de timp vizat, iar ordonata redă numărul de articole în care au fost găsiţi termenii căutaţi, pentru fiecare zi din intervalul specificat. Rezultatul cererii formulate în Fig. 3.1.2 poate fi vizualizat în Fig. 3.1.3.

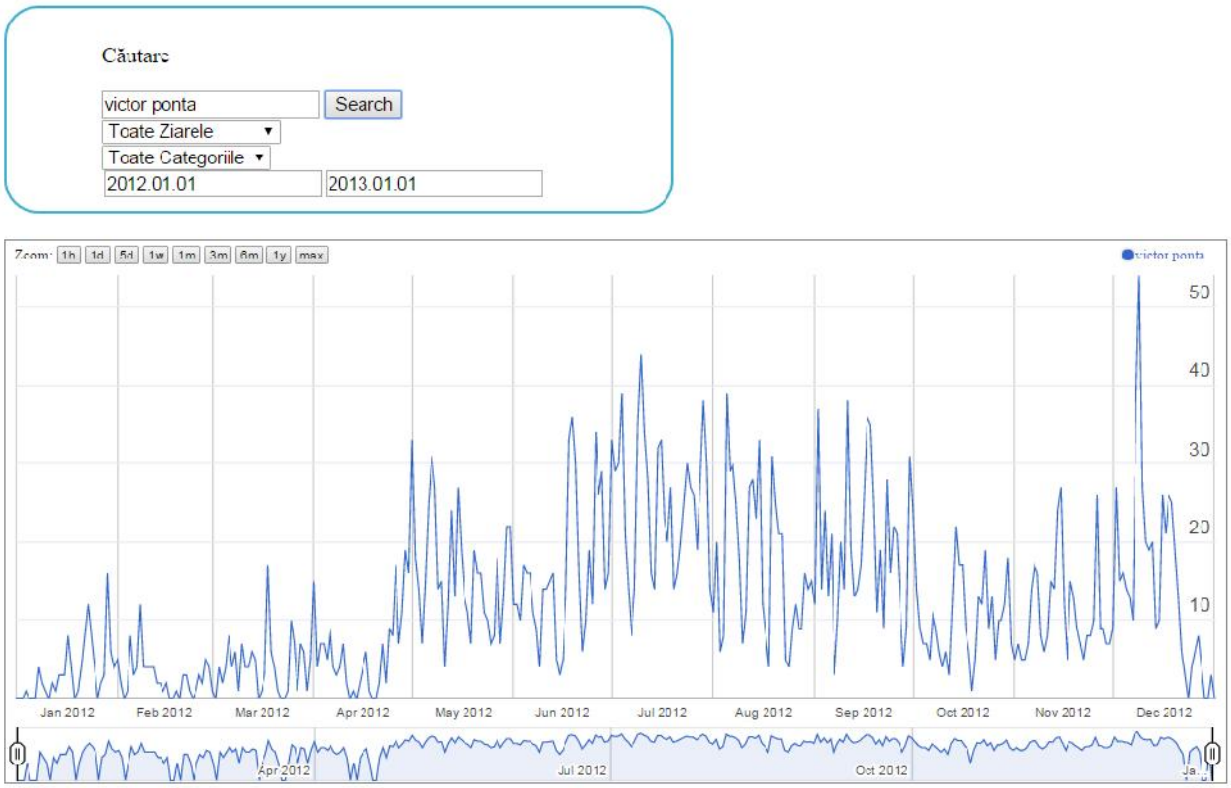

Fig. 3.1.3 Rezultatul cererii formulate în Fig. 3.1.2

Din figura de mai sus, se poate observa că intereseul opiniei publice, exprimat prin intermediul articolelor din presa scrisă online, este concentrat pe mai multe intervale temporale, înregistrând un vârf în data de 9 decembrie 2012. Acest lucru poate fi urmărit prin selectarea punctelor înregistrate pe grafic, pentru fiecare zi în parte (e.g., Fig. 3.1.4).

Utilizatorul are opţiunea de a mări orice zonă de pe grafic pentru a inspecta în detaliu rezultatul oferit. Acest lucru se poate face atât prin folosirea butonului *Zoom* poziţionat în partea stânga sus, cât şi prin derularea convenabilă a intervalului temporal, utilizând instrumentele din partea de jos a graficului.

Atât urmărirea pantelor înregistrate pe grafic, cât şi opţiunea de zoom pot fi vizualizate în Fig. 3.1.4.

În această sub-secţiune a fost detaliat modul în care serviciul *Media Analytics* este expus utilizatorului. Prin interfaţa web intuitivă şi uşor de utilizat, acesta poate efectua o analiză bazată pe metoda de tip counting (i.e., vezi CAPITOLUL 4, subsecţiunea 4.1.3). Rezultatul analizei poate fi vizualizat prin intermediul unui grafic afişat în cadrul paginii web.

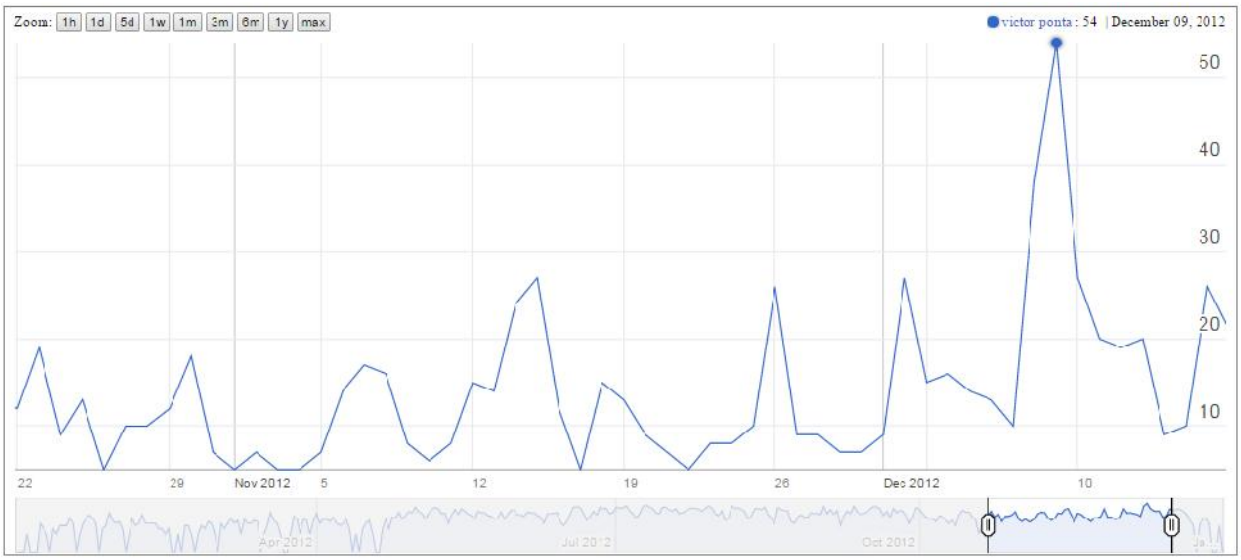

Fig. 3.1.4 Opţiunea de detaliere a unei porţiuni din grafic

3.2 ARHITECTURA SISTEMULUI

Schema bloc funcţională a sistemului „Media Analytics” este descrisă in Fig. 3.2.1.

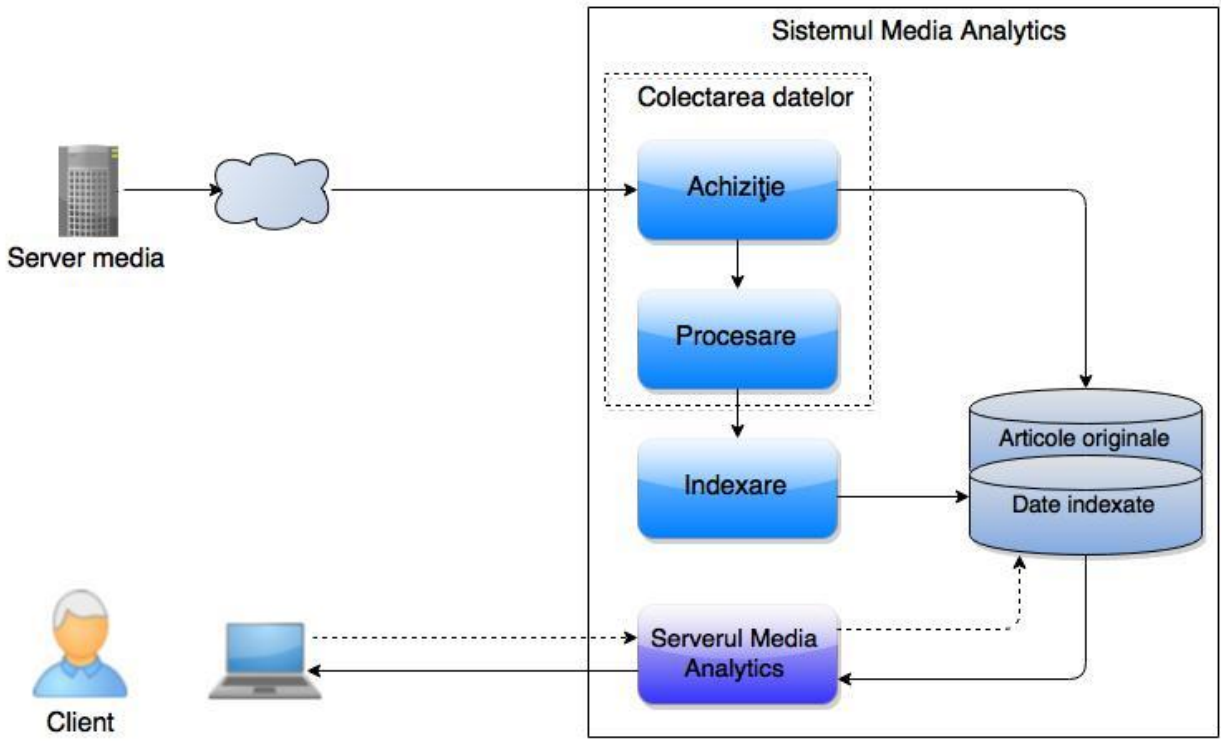

Fig. 3.2.1 Schema bloc funcţională a sistemului Media Analytics

Media Analytics este un sistem complex de achiziţie şi analiză a presei scrise online, care din punct de vedere arhitectural este constituit din patru componente principale:

1) Componenta ce vizează achiziţia şi preprocesarea conţinutului media;
2) Componenta de indexare a bazei de date;
3) Interfaţa grafică;
4) Componenta ce facilitează căutarea în baza de date indexată.

Dintre acestea, componentele ce vizează achiziţia şi preprocesarea conţinutului media, respectiv indexarea bazei de date, au fost realizate ca parte a unui alt proiect şi nu fac obiectul acestei lucrari de diplomă. Prin urmare, această lucrarea de diplomă îşi propune să realizeze o prezentare succintă a acestora.

Conform Fig. 3.2.1, serverul media reprezintă orice entitate care furnizează conţinut media online (e.g., portaluri de ştiri). Achiziţia şi preprocesarea conţinutului media sunt realizate de două subcomponente distincte, dar care iau parte la acelaşi proces: colectarea datelor. Prin urmare, acestea au fost grupate într-o singură componentă şi anume componenta ce vizează achiziţia şi preprocesarea conţinutului media. Această componentă are rolul de a colecta conţinut media (i.e., corpus de text) furnizat de o serie de portaluri de ştiri, dar şi metadate ale articolelor achiziţionate, pe care apoi le stochează într-o bază de date. Concurent are loc un proces de „curăţare" de text, ce vizează printre altele restaurarea şi uniformizarea diacriticelor conform UTF-8 (acolo unde este cazul), transformarea numerelor în cuvinte (i.e., 20 este transformat în douăzeci etc.), respectiv scrierea desfăşurată a abrevierilor şi acronimelor (i.e., RO este transformat în România, ONU este transformat în Organizaţia Naţiunilor Unite, etc.).

Pentru că interogarea unei baze de date ample în vederea căutării unor termeni ar dura foarte mult, este necesară implementarea unei componente de indexare a bazei de date (i.e, vezi CAPITOLUL 2, subsecţiunea 2.4). Dacă nu se folosea indexarea, la căutarea unui termen, baza de date ar fi fost accesată inutil şi astfel timpul în care era întors rezultatul ar fi fost mult prea mare. Prin urmare, a fost dezvoltată o componentă de indexare, care poate indexa datele fie în mod automat – concurent cu achiziţia datelor, fie retrospectiv, pe o bază de date creată în prealabil. Indecşii creaţi în urma procesului de indexare sunt stocaţi fizic pe disc, formând o bază de date indexată, ce va fi folosită în procesul de analiză.

Media Analytics, privit ca serviciu web, rulează pe un server dedicat (i.e., server de aplicaţii - *Tomcat*). Acesta este serverul Media Analytics, către care se vor îndrepta toate cererile HTTP venite din partea utilizatorilor. Dacă un utilizator doreşte efectuarea unei analize (e.g., vezi subsecţiunea 3.1 a acestui capitol), atunci acesta va accesa baza de date indexată prin intermediul unei cereri de tip HTTP GET, îndreptată către serverul Media Analytics. Serverul va interoga colecţia de indecşi şi va răspunde în cel mai scurt timp, returnând rezultatul analizei în cadrul aplicaţiei client ce rulează în browserul utilizatorului (i.e., este vorba despre interfaţa web).

Datorită funcţiilor distincte pe care le îndeplinesc, cele patru componente pot fi grupate în două module: modulul de achiziţie a datelor, respectiv modulul de analiză a datelor. Modulul de achiziţie a datelor conţine componenta ce vizează achiziţia şi preprocesarea conţinutului media şi componenta de indexare a bazei de date, iar modulul de analiză a datelor include interfaţa grafică, respectiv componenta ce facilitează căutarea în baza de date indexată.

3.3 COMPONENTA CE VIZEAZĂ ACHIZIŢIA ŞI PREPROCESAREA CONŢINUTULUI MEDIA

3.3.1 Achiziţia conţinutului media

În scopul implementării componentei de achiziţie a conţinutului media s-au studiat două opţiuni: utilizarea unui crawler web, respectiv utilizarea feed-urilor RSS.

S-a pornit prin a studia capabilităţile unui crawler web, iar în urma analizei acestora s-a constatat că un astfel de serviciu realizează harta completă a unui site web, fără a oferi însă posibilitatea de a automatiza procesul de achiziţie. Limitarea acestuia a rezultat din faptul că accesarea paginilor HTML se făcea retrospectiv, fără a avea opţiunea de a condiţiona strict introducerea de conţinut nou în baza de date. Prin urmare, stocarea conţinutului media într-o bază de date era greu de gestionat.

A fost abordată cea de-a doua variantă: utilizarea feed-urilor RSS. Aşadar, operaţia de achiziţie a conţinutului media constă în colectarea informaţiilor furnizate de o serie de portaluri de ştiri, prin intermediul feed-urilor RSS. Feed-urile RSS sunt expuse utilizatorilor în format XML, ce oferă o reprezentare structurată a datelor, fiind uşor de parcurs şi de interpretat (i.e., vezi CAPITOLUL 2, subsecţiunile 2.2, respectiv 2.5). Aceasta constituie un avantaj major în extragerea informaţiilor vizate. Extragerea acestor informaţii presupune aplicarea unor operaţii de parsare a paginilor XML, folosind o librărie Java dedicată (i.e., este vorba despre XML DOM Parser). Astfel, pot fi preluate o serie de metadate asociate articolelor, printre care se regăseşte şi data publicării acestora. Data publicării articolelor constituie un element extrem de important în analiza datelor, deoarece aceasta este folosită pentru a condiţiona introducerea de conţinut nou în baza de date şi ajută la formarea unui „timeline” pentru datele colectate.

Pentru achiziţie s-a avut în vedere o serie de portaluri de ştiri, care au fost analizate din punct de vedere structural şi dintre care doar trei s-au dovedit a fi corespunzătoare în cadrul procesului de parsare: Adevarul.ro[15], 9AM.ro[16], respectiv Wall-Street.ro[17]. Rezultatul acestei analize este redat în Tabelul 3.3.1.

Tabelul 3.3.1 Lista portalurilor de ştiri analizate

RSS URL	Probleme	Soluţii
http://www.antena3.ro/rss http://www.click.ro/rss.xml http://jurnalul.ro/rss http://www.ziarulunirea.ro/feed	Acestea folosesc marcajul XML ![CDATA], astfel că tot conţinutul unor astfel de secţiuni este ignorat de către parser.	N/A
http://adevarul.ro/rss http://www.9am.ro/rss/ultima-ora.xml http://img.wall-street.ro/rssfeeds	Feed-urile RSS se regăsesc în format XML şi pot fi parsate cu uşurinţă utilizând *DOM Parser*.	Crearea de parsere dedicate, ce parsează documente XML.
http://www.gandul.info/rss-all.html http://feeds.feedburner.com/bzi http://www.ziarulatac.ro/rss	Feed-urile RSS se regăsesc în format HTML, structura acestor portaluri de ştiri diferă semnificativ de la un site la altul şi necesită implementarea unui parser dedicat pentru fiecare portal în parte.	Crearea unui parser dedicat pentru fiecare portal de ştiri , ce parsează pagini HTML.

În esenţă, Media Analytics este un sistem automat de analiză a presei scrise online şi din acest motiv, componentele constituente ale acestuia trebuie să îndeplinească această funcţie. Pe de altă parte însă, atât achiziţia cât şi indexarea datelor pot fi realizate şi într-o manieră retrospectivă (i.e., achiziţia datelor se poate face retrospectiv, prin accesarea arhivelor de ştiri puse la dispoziţie de portaluri, iar indexarea poate avea loc asupra unei baze de date creată anterior).

În continuare, voi exemplifica modul în care are loc achiziţia datelor, luând ca referinţă feed-ul RSS al ziarului *Adevărul*. Structura documentului XML asociat acestui feed RSS este prezentată în cele ce urmează.

Orice document XML destinat unui serviciu conţine un antet, acesta fiind de fapt nodul rădăcină al documentului, în care regăsim informaţii privind tipul şi versiunea serviciului oferit (i.e., în acest caz este vorba despre serviciul RSS, iar informaţiile sunt definite prin intermediul atributelor din marcajul `<rss></rss>` şi reprezentate în Fig. 3.3.1).

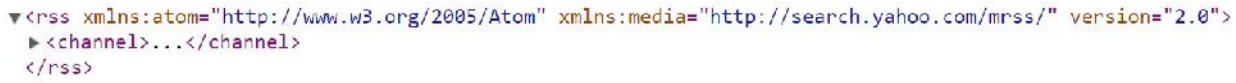

Fig. 3.3.1 Antetul unui feed RSS

Mai departe, conţinutul canalului RSS este delimitat de marcajul `<channel></channel>`. Acesta poate fi vizualizat în Fig. 3.3.2, respectiv Fig. 3.3.3.

Fig. 3.3.2 Conţinutul canalului RSS (1)

Fig. 3.3.3 Conţinutul canalului RSS (2)

Secţiunea prezentată în Fig. 3.3.2 conţine informaţii referitoare la configuraţia paginii web ce expune utilizatorului serviciul RSS pentru portalul de ştiri *Adevărul* (e.g., titlul şi linkul către această pagină, limba în care sunt afişate datele, un câmp TTL, relaţii de legătură ce trimit către

cotidianul *Adevărul*, etc), iar Fig. 3.3.3 prezintă structura efectivă a feed-ului RSS. Se observă că fiecare articol este încadrat de marcajele `<item></item>`, ce conţine metadate descrise de marcaje specifice (e.g., titlul şi linkul către articolului respectiv, scurtă descriere, data publicării, etc.).

Reamintim în această secţiune că Media Analytics este integral implementat în Java. Prin urmare, detaliile privind componentele sistemului vor fi strict legate de acest limbaj de programare.

Prin parsarea documentului XML sunt extrase metadatele ce caracterizează o ştire (i.e., este vorba despre *title*, *link*, *description*, *pubDate*). Acestea sunt preluate de sub marcajul `<item></item>` şi sunt introduse în baza de date. După cum am precizat în paragrafele anterioare, pubDate joacă un rol important în automatizarea achiziţiei de conţinut media, deoarece prin aceasta se stabileşte un „lastDate" funcţie de care este condiţionată introducerea de conţinut nou în baza de date.

În scopul accesării serviciului RSS pentru fiecare portal de ştiri, în baza de date a fost implementat un tabel sub forma unei liste de site-uri ale cărui înregistrări constituie informaţii de legătură (e.g., numele portalului de ştiri şi linkul asociat, linkul către fluxul RSS, intervalul de update al fluxului RSS, etc.). Acest tabel este reprezentat în Fig. 3.3.4.

	id [PK] bigserial	name text	link text	rssfeed text	printable boolean	time bigint	lastdate timestamp w
1	1	Adevarul	adevarul.ro	http://adev	TRUE	2400000	2014-01-01
2	2	WallStreet	img.wall-street.r	http://img.	TRUE	126000000	2014-01-01
3	3	Nineam	http://www.9am.rc	http://www.	FALSE	108000000	2014-01-01
*							

Fig. 3.3.4 Lista portalurilor de ştiri, reprezentată în baza de date

Orice interogare a bazei de date presupune realizarea unei conexiuni cu aceasta. Conectarea la o bază de date se realizează exclusiv prin intermediul JDBC. JDBC (Java DataBase Connectivity) este o interfaţă standard SQL de acces la baze de date. Acesta furnizează un acces uniform la baze de date relaţionale. JDBC este constituit dintr-un set de clase şi interfeţe scrise în Java, furnizând un API standard pentru proiectanţii de aplicaţii ce utilizează baze de date. Acest lucru face posibilă scrierea aplicaţiilor de baze de date folosind un API Java pur.

Înainte de realizarea conexiunii trebuie încărcat driverul pentru baza de date. Acest lucru se face prin apelarea metodei `Class.forName()`(e.g., `Class.forName("org.postgresql.Driver");`). Având în vedere că în JDBC o bază de date este identificată printr-un URL, dacă se foloseşte PostgreSQL, URL-ul se poate găsi sub una dintre formele:

- `jdbc:postgresql:database`
- `jdbc:postgresql://host/database`
- `jdbc:postgresql://host:port/database`

unde parametrii specificaţi au următoarea semnificaţie: `host` reprezintă numele de host al severului (i.e., implicit acesta este localhost), `port` reprezintă numărul de port deschis pe server, caracteristic bazei de date (e.g., în mod implicit, pentru PostgreSQL, are valoarea 5432), iar `database` este numele bazei de date. În afară de aceşti parametrii standard de conexiune pot fi specificaţi o serie de parametrii care permit de pildă selectarea utilizatorului bazei de date în numele căruia se face conexiunea, parola asociată acestuia, dar şi parametrii specifici unei conexiuni SSL (Secure Sockets Layer). Mai departe, pentru conectarea efectivă la baza de date, utilizatorul trebuie să obţină de la JDBC o instanţă de tip Connection. Pentru aceasta, trebuie folosită metoda `DriverManager.getConnection()`, astfel:

`Connection db = DriverManager.getConnection(url, username, password) ;`.[18]

Pentru a asigura portabilitatea aplicaţiei a fost dezvoltat un modul în Java ce oferă posibilitatea configurării acesteia, indiferent de sistemul pe care rulează (i.e., `config.Finals.java`) dar şi un alt modul prin intermediul căruia pot fi manipulate baze de date (i.e., `controllers.DatabaseController.java`). Prin urmare parametrii necesari conexiunii cu baza de date vor fi preluaţi dintr-un fişier de configurare.

Parsarea unui feed RSS începe prin interogarea bazei de date, mai precis tabelul *sites*, de unde este preluat linkul către feed-ul RSS. Această operaţie se realizează executând un *sql querry* în Java. Mai departe, acest link este utilizat pentru conectarea la feed-ul RSS al portalului de ştiri. Având în vedere că obiectul `rssFeedLink` este de tip `URL` şi conţine linkul feedului RSS, conectarea la acesta este realizată prin deschiderea unui stream de date:

```
InputStream stream = rssFeedLink.openStream();
```

iar pentru accesarea şi manipularea documentului XML, acesta din urmă trebuie încărcat într-un obiect de tip XML DOM:

```
Document doc = dBuilder.parse(stream);
```

Ulterior este creată o listă de noduri, pentru a putea selecta din document informaţiile vizate (i.e., în cazul de faţă, elementele de sub marcajul `<item></item>`):

`NodeList nList = doc.getElementsByTagName("item");` şi prin parcurgerea acestei liste sunt selectate metadatele menţionate în paragrafele anterioare. Odată preluate din documentul XML, acestea sunt introduse în baza de date, pentru fiecare articol în parte.

Pentru manipularea articolelor a fost creată clasa *Stire*, având un constructor ce preia ca parametri *title*, *url*, *siteId*, unde *title* reprezintă titlul ştirii, *url* reprezină URL-ul ştirii, iar *siteId* este un element care păstrează o evidenţă a portalurilor de ştiri, pentru a şti exact de pe ce site a fost extrasă ştirea respectivă (i.e., vezi Fig. 3.3.4).

Pentru fiecare ştire există un URL, acesta reprezentând link-ul către pagina HTML a ştirii. Paginile HTML trebuie parsate la rândul lor, pentru extragerea textului propriu-zis al ştirilor, dar şi a altor metadate care nu sunt specificate în mod obişnuit în feed-ul RSS (e.g., autor, categorie). Pentru a obţine toate aceste informaţii s-a studiat în detaliu structura paginii HTML. De asemenea s-au căutat variante în care formatul paginii web este cât mai „curat” (i.e., majoritatea paginilor HTML în care sunt reprezentate articolele conţin reclame sau videoclipuri care trebuie ignorate în procesul de parsare) şi în acest scop, acolo unde este cazul, în loc să se acceseze pagina web a ştirii, se ia în considerare versiunea printabilă a acesteia. Desigur, acest procedeu nu este convenabil în cazul în care pagina printabilă conţine insuficiente metadate comparabil cu pagina web propriu-zisă a ştirii. Această situaţie este întâlnită şi în cazul editorialului *Adevărul*: câmpul ce conţine autorul este dificil de parsat, deoarece este încadrat între marcaje `<div></div>` multiple, iar categoria este absentă în versiunea printabilă.

În urma analizei, s-a luat decizia de a accesa pagina web a ştirii şi s-au observat următoarele:

- categoria poate fi extrasă direct din URL-ul ştirii
- autorul este indicat prin marcajul `<span class="a-name"></span>`
- descrierea şi textul ştirilor se regăsesc sub marcajele `<h2 class="articleOpening">`, respectiv `<div class="article-body"></div>`

Prin urmare s-a dezvoltat un parser HTML dedicat care extrage aceste metadate din corpul paginii HTML. Procesul de parsare este similar cu cel folosit în cadrul documentelor XML, folosind însă utilitarul Java *Jsoup*.

Astfel, categoria în care se încadrează articolul vizat este extrasă direct din URL. Dacă considerăm arbitrar un articol publicat pe portalul Adevărul, URL-ul acestuia este de forma:

```
http://adevarul.ro/category/additional-info/article title_Id/index.html
```

şi poate fi prelucrat în Java, având în vedere faptul că orice URL poate fi reprezentat sub formă de string, iar un string poate fi împărţit într-o serie de stringuri constituente.

```
(1)String linkUnparsed = stire.getUrl().toString();
(2)String linkParsed[] = linkUnparsed.split("\\/");
(3)category = linkParsed[3];
(4)stire.setCategory(category);
```

În primă instanţă este obţinut URL-ul ştirii, care este convertit în String, preluat mai departe şi parsat după elementul despărţitor „/”, obţinând un vector ale cărui elemente sunt stringurile constituente (i.e., http:,blank, adevarul.ro, category). În final, pentru obţinerea câmpului *category*, este preluat elementul de pe poziţia a patra din vector (i.e., category) şi introdus în baza de date ca metadată.

Pentru selectarea celorlalte metadate, conţinutul paginii HTML este încărcat într-un document, care urmează să fie parsat:

```
Document newsDoc = Jsoup.connect(stire.getUrl().toString()
                                                    + "?mobile=0").get();
```

Este vizată varianta desktop a ştirilor şi în acest scop este dezactivată opţiunea mobilă predefinită prin adăugarea la URL a particulei „?mobile=0”. Într-un paragraf anterior am specificat modul în care este regăsit autorul articolului în pagina HTML. Atunci când un parser traversează un document de orice tip, foloseşte un selector cu o sintaxă bine definită. Pentru HTML, această sintaxă presupune interpretarea marcajelor şi atributelor specifice atsfel:

- `#id`: găseşte elementele după ID-ul specificat, e.g. `#logo`
- `.class`: găseşte elementele după numele clasei, e.g. `.a-name`, `.articleOpening`
- `[attribute]`: găseşte elementele cu atributul, e.g. `[href]`

Bineînţeles, se pot realiza combinaţii precum `el#id`, `el.class`, `el[href]`(e.g., `div#logo`, `div.a-name`, `div.articleOpening`, `a[href]`).

În concluzie, autorul, decrierea şi textul articolelor au fost extrase astfel:

```
//Get AUTHOR
Elements newsAuthor = newsDoc.select(".a-name a[rel]");
if (newsAuthor.text().length() > 3) {
    author = newsAuthor.text();
}
stire.setAuthor(author);
//Get DESCRIPTION
Elements newsDescription = newsDoc.select(".articleOpening");
description = newsDescription.text();
stire.setDescription(description);
//Get TEXT
Elements newsText = newsDoc.select(".articleOpening , .article-body");
String temp = newsText.text().trim().replaceAll("\\. ", ".\n");
```

În cazul în care se doreşte descărcarea retrospectivă a articolelor, sunt accesate arhivele portalurilor de ştiri şi este desfăşurat un proces similar de parsare a paginii web asociate. Procedeul însă este mai complex, în sensul că au loc o serie de operaţii de formare a URL-ului ce caracterizează o astfel de pagină, astfel încât să fie corespunzător modelului caracteristic fiecărui site. Prin urmare au fost create parsere dedicate pentru achiziţionarea retrospectivă a ştirilor pentru fiecare portal de ştiri în parte. Structura acestui URL pentru cotidianul *Adevărul* este reprezentată mai jos:

http://adevarul.ro/cauta/?terms=victor%20ponta&fromDate=2012-1-1&toDate=2015-6-20

3.3.2 Preprocesarea conţinutului media

În ceea ce priveşte calitatea textului achiziţionat, acesta poate fi afectat de erori de formatare atunci când este încărcat pe pagina web a portalului de ştiri. De cele mai multe ori se întâmplă ca anumite paragrafe ale textului să conţină puţine cuvinte, iar dacă sunt reprezentate în format *justified* atunci aceste cuvinte apar spaţiate pe un singur rând. De asemenea, textul poate devia de la standardul UTF-8, astfel că nu mai este respectată o uniformitate a caracterelor (e.g., în reprezentarea ASCII a caracterelor nu există diacritice). Un alt aspect important care trebuie luat în considerare este că anumite site-uri nu oferă text formatat cu diacritice, iar acestea trebuie restaurate.

Pentru a evita stocarea în baza de date a unui text ce prezintă una sau mai multe dintre problemele descrise mai sus, a fost implementată o soluţie prin care este realizată „curăţarea" textului. În acest scop s-a folosit un utilitar complex de NLP dezvoltat de laboratorul SpeeD (i.e., este vorba despre utilitarul JavaNLP2, implementat sub forma unei librării Java). Acest utilitar realizează printre altele reducerea spaţierii între cuvinte la un singur spaţiu, restaurarea diacriticelor, adaptarea textului la standardul UTF-8 astfel încât să fie asigurată o uniformitate a caracterelor, scrierea detaliată a abrevierilor şi acronimelor, precum şi transformarea numerelor în cuvinte.

Procesul de „curăţare" a textului este foarte important în analiza datelor. În acest context, trebuie realizată o uniformizare a textului achiziţionat de pe site-urile media, pentru a asigura o căutare mult mai clară şi relevantă a datelor în colecţia de indecşi.

3.4 COMPONENTA DE INDEXARE A BAZEI DE DATE

În subsecţiunea 3.2 a acestui capitol, ce vizează descrierea arhitecturii sistemului, a fost evidenţiată importanţa implementării unei componente de indexare a datelor. Indexarea datelor aparţine de asemenea modulului de achiziţie (i.e., conform clasificării efectuate în finalul subsecţiunii 3.2) şi poate fi realizată concurent cu achiziţia datelor, dar şi retrospectiv, pe o colecţie de date creată în prealabil.

Indexarea datelor presupune stocarea pe disc a unui structuri de indecşi ce va conţine informaţii de tip document. Documentul reprezintă o colecţie de câmpuri care pun în evidenţă metadatele supuse procesului de indexare, precum şi conţinutul propriu-zis al acestora (i.e., termenii ce alcătuiesc un câmp). Alegerea metadatelor ce sunt supuse procesului de indexare se face în funcţie de cerinţele aplicaţiei ce facilitează operaţia de căutare în baza de date indexată. De pildă, în cadrul serviciului Media Analytics a fost indexat conţinutul articolelor de ştiri (i.e., textul ştirilor) – deoarece termenii sunt căutaţi în corpul ştirii, data publicării, categoria în care se poate încadra un termen căutat şi identificatorul portalului de ştiri – deoarece acestea sunt filtre aplicate în procesul de căutare, titlul şi descrierea articolului de ştiri – pentru dezvoltarea ulterioară a serviciului. Modul în care este realizat procesul de indexare este prezentat mai jos.

```
public void addToIndex(Stire stire) {
    try {
        Document document = new Document();
        document.add(new TextField("title", stire.getTitle(), Field.Store.YES));
        document.add(new TextField("description", stire.getDescription(), Field.Store.YES));
        document.add(new TextField("text", stire.getTextAsString(), Field.Store.YES));
        document.add(new TextField("category", stire.getCategory(), Field.Store.YES));
        document.add(new TextField("date", DateTools.dateToString(stire.getPubDate(), DateTools.Resolution.DAY),
                Field.Store.YES));
        document.add(new IntField("site_id", stire.getSiteId(), Field.Store.YES));
        writer.addDocument(document);
        writer.commit();

    }
}
```

Pentru fiecare ştire în parte, metoda `addToIndex` adaugă la colecţia de indecşi, în `document`, informaţiile specificate, în următoarea manieră: primul parametru al constructorului `TextField` reprezintă numele câmpului şi descrie ce metadate au fost indexate în câmpul respectiv, cel de-al doilea parametru reprezintă conţinutul metadatelor care se indexează, iar ultimul parametru este o directivă specifică *Lucene*. `writer` este o instanţă a clasei `IndexWriter`, care scrie efectiv datele indexate într-un director. Colecţia de indecşi rezultată va fi folosită mai departe de modulul de analiză a datelor, ce va fi descris în secţiunea următoare a lucrării de diplomă.

CAPITOLUL 4

ANALIZA DATELOR, GENERAREA ŞI VIZUALIZAREA STATISTICILOR

4.1 MECANISME DE ANALIZĂ A DATELOR

4.1.1 Analiza sentimentelor

Analiza sentimentelor (sentiment analysis sau opinion mining) se referă la utilizarea prelucrării limbajului natural, analizei de text precum și lingvisticii computaționale pentru a identifica şi extrage informații cu caracter subiectiv din materiale sursă. La modul general analiza sentimentelor urmărește să identifice atitudinea unei anumite persoane (atitudine exprimată în scris sau oral) referitoare la o anumită topică sau la contextul unui document. Atitudinea poate fi propria judecată , starea emoțională sau comunicarea emoțională intenționată de autor.

O sarcină de bază a acestui tip de analiză este de a clasifica polaritatea unui text dat, la nivel de document, propoziţie sau trăsatură distinctivă/aspect – dacă opinia exprimată în legătură cu respectivul document, propoziţie sau trăsatura distinctivă/aspectul unei entităţi este pozitivă, negativă sau neutră. La nivel superior, o clasificare a sentimentelor „dincolo de polaritate” caută, de exemplu stări emoţionale precum furie, tristeţe sau fericire.

Primele cercetări în acest domeniu îi aduc în prim-plan pe Turney[19] şi Pang[20], care au aplicat diferite metode pentru a determina polaritatea recenziilor unor produse, respectiv filme. Activitatea s-a desfăşurat la nivel de document. Polaritatea unui document se poate clasifica de asemenea pe o scară multiplă, direcţie adoptată de către Pang[21] şi Snyder[22] (printre alţii). Pang a extins sarcina de bază privind clasificarea unei recenzii de film de la a fi pozitivă sau negativă la estimarea unor evaluări pe o scară de la 3 la 4 stele, în timp ce Snyder a desfăşurat o analiză în profunzime asupra recenziilor unor restaurante, estimând evaluări privind diferite aspecte ale restaurantului dat, precum calitatea mâncării şi atmosfera (pe o scară de la una la cinci stele). Chiar dacă în cele mai multe metode de clasificare statistică, clasa neutră este ignorată, în ipoteza că textele neutre din punct de vedere al polarităţii se află în apropiere de graniţa de clasificator binar, mai mulţi cercetători sugerează că, la fel ca în orice problemă de polaritate, trebuie identificate trei categorii de funcţii ale proceselor afective. Mai mult, se poate demonstra că anumiţi clasificatori, cum ar fi Max Entropy[23] sau clasificatorul SVM[24] (Support Vector Machine), pot beneficia de introducerea clasei neutre pentru a îmbunătăți acuratețea generală a clasificării.

O metodă diferită de determinare a sentimentului este folosirea unui sistem de clasificare în care pentru cuvintele asociate frecvent cu un sentiment negativ, neutru sau pozitiv, se stabileşte un număr pe o scară de la -10 la 10 (cel mai negativ către cel mai pozitiv) şi atunci când un fragment de text nestructurat este analizat utilizând NLP (Natural Language Processing), conceptele rezultate în urma analizei vor fi la rândul lor analizate pentru înţelegerea cuvintelor şi modului în care acestea se raportează la conceptele respective. Mai departe, fiecare concept primeşte un scor bazat pe gradul în care cuvintele analizate se raportează la conceptul respectiv, dar si pe scorul propriu fiecărui cuvânt, stabilit anterior. Acest lucru permite înţelegerea mai aprofundată a analizei sentimentelor, bazată pe o scară în 11 puncte.[25]

O altă direcţie de cercetare este identificarea gradului de subiectivitate/obiectivitate al unui text. În acest caz, sarcina este de a determina dacă un text dat (uzual o propoziţie) este obiectiv sau subiectiv. Această chestiune poate fi uneori mai dificilă decât clasificarea polarităţii[26]: subiectivitatea cuvintelor şi frazelor poate depinde de contextul în care apar, iar documentele cu caracter obiectiv pot conţine propoziţii subiective (e.g., un articol de ştiri citând opiniile celor intervievaţi). Mai mult, după cum s-a menţionat de către Su[27], rezultatele depind în mare măsură de definiţia subiectivităţii abordată în momentul adnotării textelor. Totuşi, Pang[28] a arătat că eliminarea propoziţiilor obiective dintr-un document înainte de a-i clasifica polaritatea, poate ajuta la îmbunătăţirea performanţei.

Un model de analiză mai atentă a sentimentelor se bazează pe analiza trăsăturilor/aspectului.[29] Aceasta se referă la determinarea opiniilor sau sentimentelor exprimate în legătură cu diferite trăsături sau aspectul unei entităţi (e.g., un telefon mobil, un aparat foto digital sau o bancă). Trăsătura sau aspectul constituie un atribut sau o componentă a unei entităţi (e.g., ecranul unui telefon mobil, calitatea imaginii unui aparat de fotografiat). Această chestiune implică câteva sub-probleme: identificarea entităţilor relevante, extragerea trăsăturilor/aspectelor ce ţin de acestea şi determinarea dacă o opinie exprimată în legătură cu fiecare trăsătură/aspect este pozitivă, negativă sau neutră.[30]

Abordările existente în analiza sentimentelor pot fi grupate în patru categorii principale: detectarea cuvintelor cheie, afinitate lexicală, metode statistice şi tehnici la nivel de concept. Detectarea cuvintelor cheie clasifică textul în conformitate cu categorii afective bazate pe prezenţa cuvintelor emotive distincte, precum *fericit*, *trist*, *speriat* sau *plictisit*. Afinitatea lexicală nu numai că detectează cuvintele emotive distincte, dar şi asociază prezumtiv cuvintelor arbitrare, o „afinitate" pentru anumite emoţii. Metodele statistice se bazează pe elemente de machine learning, precum LSA (Latent Semantic Analysis), SVM, BOW (bag of words)şi SO –

PMI (Semantic Orientation – Pointwise Mutual Information). Metode mai sofisticate incearcă sa detecteze posesorul unui sentiment (i.e., persoana care menţine acea stare afectivă) şi ţinta (i.e., entitatea spre care este îndreptată starea afectivă). Pentru a extrage opinia din context şi a obţine trăsatura asupra căreia se îndreaptă, sunt folosite relaţii gramaticale între cuvinte. Relaţiile de dependenţă gramaticală sunt obţinute prin parsarea în profunzime a textului. Spre deosebire de metodele pur sintactice, tehnicile la nivel de concept se bazează pe elemente de reprezentare a cunoştinţelor, precum ontologii şi reţele semantice, fiind de asemenea capabile să detecteze semantici exprimate într-o manieră subtilă, e.g., prin analiza unor concepte care nu transmit în mod explicit informaţii relevante, dar care sunt implicit legate de alte concepte care fac acest lucru.

În afară de tehnicile software folosite în analiza sentimentelor, este necesară şi intervenţia factorului uman, în condiţiile în care sistemele automate nu sunt în măsură să analizeze tendinţele dezvoltate de un comentator individual, acestea fiind clasificate incorect din punct de vedere al sentimentelor exprimate.

Acurateţea unui astfel de sistem de analiză se referă, în principiu, la cât de bine clasificările acestuia sunt în acord cu aprecierile factorului uman. Astfel, potrivit cercetărilor, în mod obişnuit, evaluatorii umani convin în 79% din situaţii.[31]

Dacă punem în conjuncţie analiza sentimentelor si Web 2.0, în contextul dezvoltării platformelor sociale precum blogurile sau reţelele sociale, opinia exprimată online s-a transformat într-un fel de monedă virtuală pentru întreprinderile care vor să-şi comercializeze produsele, să identifice noi oportunităţi şi să îşi gestioneze reputaţia. Deoarece întreprinderile urmăresc automatizarea procesului de „filtrare a zgomotului", înţelegerea conversaţiilor, respectiv identificarea conţinutului relevant şi folosirea adecvată a acestuia, societatea se orientează spre domeniul analizei sentimentelor.

Problema este că cei mai mulţi algoritmi de analiză a sentimentelor folosesc termeni simpli pentru a exprima sentimentul despre un produs sau serviciu. Cu toate acestea, factorii culturali, nuanţele lingvistice şi contextele diferite, fac extrem de dificilă sarcina de a transforma un fragment de text scris într-un simplu sentiment pro sau contra. Faptul că de multe ori oamenii nu sunt de acord cu privire la sentimentul exprimat în legatură cu fragmentul de text dat, ilustrează cât de dificilă este pentru un computer aceasta sarcină de a clasifica în mod corect opiniile exprimate. Cu cât fragmentul de text este mai scurt, cu atât sporeşte dificultatea acestei sarcini; din acest motiv se impune realizarea analizei asupra unui corpus mare de text.

Analiza sentimentelor poate fi efectuată, de asemenea, asupra conţinutului vizual (i.e., imagini, videoclipuri). Una dintre primele abordări în această direcţie este *SentiBank*, care pentru reprezentarea conţinutului vizual pune în corespondenţă adjective şi substantive, formând liste de corespondenţă. Structura unei astfel de liste poate fi vizualizată în Fig. 4.1.1.

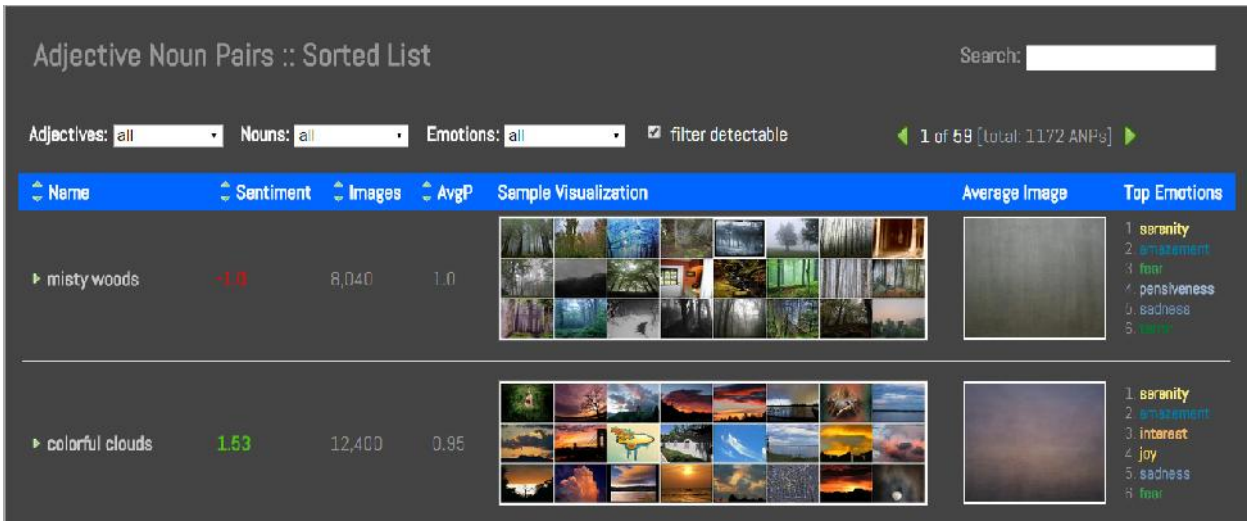

Fig. 4.1.1 Structura unei liste de corespondenţă, utilizată în analiza conţinutului vizual

Conform Fig. 4.1.1, o listă de corespondenţă oferă informaţii în legătură cu procesul de analiză a conţinutului vizual, specificând asocierea adjectiv-substantiv, adoptată pe baza analizei a „i" imagini, precum şi un scor ce exprimă sentimentul transmis, unde „i" este un număr natural diferit de zero.

4.1.2 *Realizarea unor corelaţii*

Un alt tip de analiză ce poate fi efectuată asupra conţinutului media este realizarea unor corelaţii între subiecte de interes general sau între entităţi media.

În anul 2013, cercetătorii americani au efectuat un studiu[32] care a confirmat corelaţia între Twitter şi audienţele TV (Television).

Potrivit platformei de analiză media *SocialGuide*, în 2012, 32 de milioane de utilizatori unici din S.U.A au scris pe Twitter despre programele TV agreate.

Prin analiza postărilor de pe Twitter despre programele live TV, studiul a confirmat o relaţie de legătură între Twitter şi audienţele înregistrate de programele TV. Studiul a identificat de asemenea Twitter ca fiind una dintre cele trei variabile statistic semnificative (în plus faţă de evaluările din anul precedent şi cheltuielile de publicitate) ce se aliniază cu audienţa TV. Andrew Somosi, CEO al SocialGuide afirma la momentul respectiv: „Ne aşteptam sa observăm o corelaţie între Twitter şi audienţa TV, însă acest studiu cuantifică tăria acestei relaţii.".

Corelaţia este influenţată în mare parte de creşterea semnificativă a consumului de conţinut media şi de avântul tehnologic în ceea ce priveşte dispozitivele electronice. Cercetătorii au estimat că 80% din posesorii americani de tablete şi smartphone-uri care urmăresc programe TV, îşi folosesc dispozitivele în acest scop cel puţin de câteva ori pe lună. De asemenea, 40% din utilizatorii de tablete şi smartphone-uri vizitează reţelele sociale în timp ce urmăresc programele TV.

Este important de remarcat cât de bine se aliniază Twitter la audienţele înregistrate de programele TV. Studiul Nielsen/SocialGuide a confirmat că sporirea volumului de posturi pe Twitter se află în corelaţie cu creşterea audienţei programelor TV pentru diferite grupe de vârstă, dezvăluind o corelaţie puternică în cazul publicului mai tânăr. Mai exact, cercetătorii au constatat că pentru tinerii cu vârste între 18 şi 34 de ani, o creştere de 8.5% în volumul posturilor de pe Twitter (i.e., reprezentată cu albastru în Fig. 4.1.2), corespunde unei creşteri de 1% a audienţei TV pentru episoadele în premieră a serialelor. Pe de altă parte, o creştere de 14.0% în volumul posturilor de pe Twitter (i.e., reprezentată cu albastru în Fig. 4.1.2) este asociată cu o

creştere de 1% a audienţei programelor TV, oferite de publicul cu vârste între 35 şi 49 de ani, reflectând o relaţie mai puternică între Twitter şi TV, pentru publicul mai tânăr.

Mai mult, corelaţia dintre posturile de pe Twitter şi audienţele TV se consolidează în cazul episoadelor plasate la mijlocul sezonului (i.e., reprezentate cu verde în Fig. 4.1.2), pentru ambele categorii de vârstă. O creştere de 4.2%, respectiv 8.4%, în volumul posturilor de pe Twitter, este asociată cu o creştere de 1% a audienţelor TV pentru categoria 18-34 ani, respectiv 35-49 de ani. Rezultatele studiului sunt prezentate în Fig. 4.1.2[33].

Realizarea unor corelaţii între subiecte presupune analiza simultană a două sau mai multe subiecte de interes general, expuse în mass-media. Analiza efectuată poate fi o analiză a sentimentelor asupra subiectelor vizate, de unde poate rezulta impresia generală a publicului cu privire la acestea, sau o analiză de tip „counting".

Corelaţiile între subiecte îşi demonstrează utilitatea atunci când companiile îşi propun analize şi studii de piaţă, dar şi cercetări de marketing, situaţii în care realizarea unor corelaţii cu privire la concurenţă, sau corelaţii între evenimente definitorii pentru respectivele companii se dovedesc a fi de o importanţă majoră.

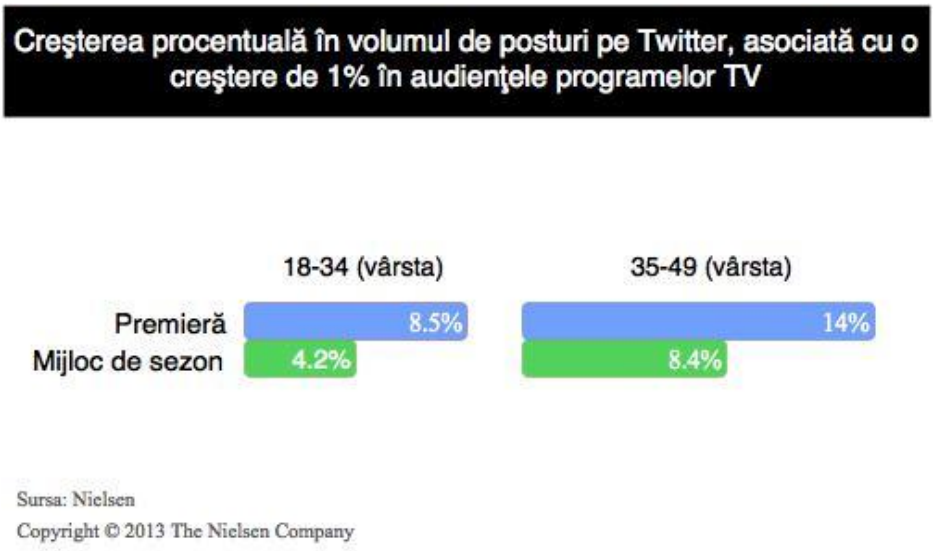

Fig. 4.1.2 Rezultatele studiului Nielsen/SocialGuide

4.1.3 *Metoda de counting*

Această metodă de analiză presupune determinarea frecvenţei de apariţie a unor termeni în mediul online, analizând în general interesul utilizatorilor cu privire la acei termeni şi pe baza analizei se pot face previziuni în timp real.

Google utilizează o formă a acestei metode în cadrul serviciului *Google Trends*, în care sunt cuantificate căutarile unice în legătură cu anumiţi termeni şi prin analiza acestora sunt oferite previziuni, dar şi o viziune retrospectivă asupra evenimentelor importante care au marcat opinia publică.

Caracterul predictiv al Google Trends a fost observat în diferite arii ale ştiinţelor sociale. De exemplu, prin monitorizarea numărului de căutări pe Google pentru termeni aflaţi în conjucţie cu prevenirea şi vindecarea gripei, au fost observate focare de gripă în curs de dezvoltare. Aceasta demonstrează importanţa unei astfel de analize în observarea şi prevenţia unor evenimente care pot avea un impact major în viaţa cotidiană.

Metoda de counting vizează, de asemenea, evidenţierea interesului manifestat de către utilizatori în legatură cu anumite subiecte expuse în mass-media. Analiza este realizată prin estimarea numărului de apariţii ale unor termeni în articole distincte publicate în mediul online. În acest mod, se poate aprecia zona de interes către care se orientează mass-media într-un anumit interval temporal şi cum influenţează informaţia expusă opinia utilizatorilor.

4.2 COMPONENTA CE FACILITEAZĂ CĂUTAREA ÎN BAZA DE DATE INDEXATĂ

Componenta ce facilitează căutarea în baza de date indexată a fost creată pentru a implementa funcţia de analiză a sistemului *Media Analytics*, respectând specificaţiile modelului arhitectural REST (i.e., vezi CAPITOLUL 2, subsecţiunea 2.5).

Pentru ca această componentă să-şi îndeplinească funcţiile, este necesar ca celelalte două componente (i.e., componenta ce vizează achiziţia şi preprocesarea conţinutului media, respectiv componenta de indexare a bazei de date) să conlucreze pentru asigurarea unei baze de date indexată. Această componentă a fost implementată cu ajutorul utilitarului *Lucene*, care stă de altfel la baza procesului de indexare a datelor.

Atunci când un utilizator doreşte să efectueze o analiză a conţinutului media, putem spune că acesta construieşte, în pagina web, o structură de filtre. Fizic, această structură de filtre este creată de către componenta ce facilitează căutarea în baza de date indexată.

În momentul în care este solicitată analiza datelor, componenta de căutare accesează şi citeşte colecţia de indecşi prin intermediul metodei `createIndexSearcher()`, reprezentată în fragmentul de cod de mai jos. În cadrul acestei metode, pe baza directorului fizic care conţine colecţia de indecşi, este creat un director virtual (i.e., `dir`). Acest director va fi folosit în continuare pentru realizarea operaţiilor ulterioare de citire şi de căutare a indecşilor.

```
public void createIndexSearcher(){

        dir = FSDirectory.open(indexDirFile);
        indexReader  = DirectoryReader.open(dir);
        indexSearcher = new IndexSearcher(indexReader);
    }
```

Structura de filtre este creată progresiv, pe baza criteriilor de căutare introduse în pagina web de către un utilizator. Criteriile de căutare sunt de fapt atributele clasei `Query`, care caracterizează orice interogare efectuată în pagina web. Atributele (e.g., keywords, newspapers, category, startDate, endDate) sunt definite ca liste ce conţin elemente de tip `String`, deoarece procesul de căutare suportă introducerea de termeni multiplii (i.e., utilizatorul poate căuta un termen compus din mai multe cuvinte, poate filtra căutarea după toate ziarele, respectiv toate categoriile).

Din punct de vedere programatic, structura de filtre este o instanţă a clasei `BooleanQuery`, necesară în orice sistem care utilizează filtre în căutarea datelor. Spunem că procesul de formare a acestei structuri este progresiv, deoarece pe măsură ce utilizatorul selectează criterii de căutare a datelor, noi termeni sunt adăugaţi în acel obiect de tip `BooleanQuery`. Acest procedeu poate fi exemplificat luând în considerare un fragment de cod din metoda `search()`, care primeşte ca parametru o instanţă a clasei `Query` şi returnează numărul de ocurenţe ale termenilor căutaţi.

```
BooleanQuery booleanQuery = new BooleanQuery();
for(String keyword:query.getKeywords()){
        booleanQuery.add(new TermQuery(new Term("text", keyword)),
                                        BooleanClause.Occur.MUST);
}
```

Fiecare element *keyword* din lista *keywords* (i.e., atribut al clasei `Query`) este adăugat acelui *booleanQuery*, sub forma unui obiect de tipul `TermQuery`, cu specificaţia că pentru acesta se foloseşte clauza `Occur.MUST`. Clauza specifică modul în care are loc căutarea în indecşi a termenilor adăugaţi. În acest caz, `Occur.MUST` poate fi interpretat ca un *ŞI logic*, impunând ca rezultatul să conţină obligatoriu toţi termenii căutaţi, nu numai unul dintre aceştia (i.e., la polul opus se află `Occur.SHOULD` ,interpretat în mod similar ca fiind un *SAU logic*). *Keyword* reprezintă cuvântul introdus în câmpul de căutare din pagina web. În mod similar, în booleanQuery se adaugă şi celelalte criterii (i.e., newspapers, category, etc.), până când se obţine structura de filtre completă. Este important de menţionat că pentru toate filtrele s-a folosit clauza `Occur.MUST`, astfel că utilizatorul va trebui să formuleze de fiecare dată o structură de filtre completă. În caz contrar, atunci când se va face căutarea nu va fi respectată structura lui `booleanQuery`, iar rezultatul căutării va fi necorespunzător (i.e., va fi afişat un grafic fără niciun element).

Căutarea se face prin intermediul metodei `search()` specifică utilitarului Lucene, care primeşte ca parametri o structură de filtre de tip `BooleanQuery` (e.g., `booleanQuery`) şi un obiect de tip `TopScoreDocCollector` (e.g., `collector`). Acest *collector* înregistrează numărul de apariţii ale termenilor căutaţi în colecţia de indecşi.

Metodele amintite mai sus (i.e., `createIndexSearcher()`, respectiv `search()`) aparţin clasei `MediaSearcher()`.

Conceptele ce caracterizează modelul arhitectural REST au fost prezentate în CAPITOLUL 2, subsecţiunea 2.5 a prezentei lucrări de diplomă. Reamintim că într-un serviciu bazat pe modelul arhitectural REST totul este o resursă, care este identificată printr-un URI. De asemenea, datele asupra cărora clientul îi spune serverului să opereze se află în URI, iar operaţia pe care serverul o face asupra datelor este descrisă direct de metoda HTTP specificată (e.g., PUT, GET, POST, DELETE). HTTP *GET* accesează resursa (i.e., o citeşte) fără a o altera.

Orice serviciu bazat pe REST şi implementat în Java (i.e., prin intermediul librăriei *Jersey*), foloseşte aşa numitele adnotări, pentru a-şi accesa resursele. Acestea sunt reprezentate prin indicativul „@”, urmat de o directivă specifică (e.g., @PATH(/path), @GET, @Produces(MediaType.TEXT_PLAIN), @Consumes(type)). Toate aceste adnotări sunt aplicate unei metode Java, care va fi în final apelată în browser prin accesarea unui URL specific.

- @PATH setează calea pentru URL-ul de bază folosit pentru a accesa serviciul (i.e., aceasta este doar o particulă a URL-ului – nu este suficient să fie specificată doar calea).
- @GET indică faptul că metoda va răspunde unei cereri de tip HTTP GET.
- @Produces specifică ce tip de MIME (Multipurpose Internet Mail Extensions) este produs în urma apelării metodei (i.e., în cazul de mai sus este produs text (“text/plain”).
- @Consumes specifică ce tip de MIME este consumat de metoda respectivă.

Media Analytics este realizat în concordanţă cu modelul arhitectural REST. În consecinţă, clasa care implementează serviciul de căutare şi de analiză a datelor este conformă acestui model. Este vorba despre clasa `Searcher`, adnotată corespunzător prin `@Path("/searcher")`.

`Searcher()` conţine metoda `search()`, adnotată `@Path("/search")`, care preia ca parametru un obiect de tip `HttpServletRequest` şi returnează un `String`. Metoda răspunde cererilor de tip HTTP GET, consumă text şi produce un document XML. Prin `@Context`, Jersey va apela metoda la fiecare `request` venit din partea unui client (i.e., are loc injectarea câmpurilor aflate în context). Modul în care sunt realizate aceste adnotări, este redat în fragmentul de cod de mai jos:

```
@Path("/searcher")
public class Searcher {

    @Path("/search")
    @GET
    @Produces(MediaType.TEXT_XML)
    @Consumes(MediaType.TEXT_PLAIN)
    public String search(@Context HttpServletRequest request)
```

Orice cerere venită din partea unui client va fi interpretată ca fiind un request. Prin intermediul JavaScript, structura de filtre este preluată din pagina web şi descompusă în elementele constituente (i.e., keyword, category, newspaper, etc.). Elementele structurii de filtre sunt extrase din documentul HTML asociat paginii web, prin selectarea indentificatorului corespunzător, astfel: `keyword = document.getElementById("keyword").value;`, unde document face referire la documentul HTML, iar elementul la care se referă este un `div`, indentificat printr-un `id` specific (e.g., în acest caz, „keyword"). Aceste elemente au rolul de a forma, împreună cu un URL de bază, URL-ul caracteristic utilizat pentru accesarea serviciului. URL-ul de bază este de forma:

`http://dev.speed.pub.ro:10082/MediaAnalytics-Search/rest/searcher/search` , în care particulele constituente sunt definite anterior în două fişiere XML (i.e., context.xml, respectiv web.xml). Acest URL este reţinut de variabila `baselink`. Pentru obţinerea URL-ului caracteristic, elementele structurii de filtre extrase anterior sunt concatenate succesiv, utilizând simboluri specifice (e.g., „?", „&"):

```
termination = "?keyword=" + keyword + "&category=" + cat + ...
```

URL-ul astfel obţinut va fi accesat pentru a trata request-ul. În acest scop, este creată o variabilă de tip `XMLHttpRequest`: `var xmlHttp = new XMLHttpRequest();` şi este accesat URL-ul creat anterior: `xmlHttp.open("GET", baselink + termination, false);`. Urmează ca request-ul să fie trimis către serverul *Media Analytics*: `xmlHttp.send();`. Deoarece sunt utilizate cereri de tip `XMLHttpRequest`, răspunsul întors de către server ajunge tot în interiorul scriptului, unde urmează sa fie procesat.

În acest context este apelată metoda `Searcher.search()` (i.e., adnotată prin `@Path("/search")`) care decapsulează request-ul în elementele sale constituente (i.e., keyword, category, newspaper, etc.), instanţiază un obiect din clasa `MediaSearcher` şi iniţiază căutarea în baza de date indexată.

4.3 ANALIZA DATELOR

Algoritmul implementat în cadrul sistemului Media Analytics vizează analiza sentimentelor conform metodei de counting (i.e., prezentată în subsecţiunea 4.1.3 a acestui capitol).

Acesta presupune preluarea intervalului temporal specificat de către un utilizator în pagina web şi returnarea, pentru fiecare zi din interval, a numărului de elemente ce sunt în concordanţă cu structura de criterii şi filtre construită.

Pe baza rezultatului obţinut în urma căutării, pentru fiecare zi din intervalul temporal specificat, este completată o matrice cu M linii şi N coloane (i.e., unde M reprezintă numărul de zile din intervalul temporal, iar N=2). Matricea este completată pe linii astfel:

$$\textbf{array} = \begin{pmatrix} 2012.01.01 & 33 \\ 2012.01.02 & 29 \\ 2012.01.03 & 39 \\ . & . \\ . & . \\ . & . \\ 2012.12.09 & 54 \end{pmatrix}$$

unde un element a_{i1} al matricei reprezintă data calendaristică, iar un element a_{i2} indică numărul de ocurenţe din data respectivă.

Am ales o reprezentare matriceală a rezultatului, pentru a structura cât mai bine datele şi pentru a putea păstra corespondenţa „dată calendaristică – număr de ocurenţe”, necesară mai departe în analiza datelor.

REST permite ca resursele să aibă diferite reprezentări (e.g., text, XML, JSON, etc.). Clientul REST poate solicita o reprezentare specifică prin intermediul protocolului HTTP. Pentru ca serviciul Media Analytics să fie conform modelului arhitectural REST, am ales ca răspunsul primit în urma cererii HTTP GET să fie reprezentat sub forma unui document XML.

Documentul XML este creat utilizând metode specifice din librăria Java *DOM Parser*. În primă instanţă este preluată matricea de rezultate şi este parcursă pe linii. Apoi, pentru fiecare linie (i.e., vector) din matrice, sunt extrase părţile componente (e.g., v[0] = 2012.01.01 , v[1] = 33) şi sunt adăugate ca atribute ale marcajului `<item></item>` din documentul XML. Operaţia se realizează până la epuizarea liniilor din matrice. În final, ca răspuns la cererea HTTP GET, se obţine un document XML cu structura din Fig. 4.3.1.

```
▼<news-counts>
    <item count="4" date="2012.01.01"/>
    <item count="2" date="2012.01.02"/>
    <item count="1" date="2012.01.03"/>
    <item count="1" date="2012.01.04"/>
    <item count="3" date="2012.01.05"/>
    <item count="1" date="2012.01.06"/>
    <item count="1" date="2012.01.07"/>
    <item count="2" date="2012.01.08"/>
    <item count="2" date="2012.01.09"/>
```

Fig. 4.3.1 Răspunsul sistemului în urma unei cereri de tip HTTP GET

Se observă că elementul rădăcină al documentului XML este indicat prin intermediul marcajului `<news-counts></news-counts>` şi înglobează rezultatul căutării între marcajele `<item></item>`.

Pentru a interpreta rezultatul căutării este necesar ca răspunsul sub formă de XML să fie întors în JavaScript pentru a fi procesat. Acest lucru are loc în mod automat, având în vedere că răspunsul este de tip `XMLHttpRequest()`. Răspunsul este reţinut de variabila `responseText` şi este parsat în vederea extragerii atributelor de sub marcajul `<item></item>` (i.e., dată, număr de ocurenţe):

```
var responseText = xmlHttp.responseText;
var xml = $.parseXML(responseText);
var x = xml.getElementsByTagName('item'); //variabilă ce reţine toate elementele marcate prin <item></item>.
```

Putem spune că în urma parsării documentului XML în JavaScript, au fost obţinute elementele necesare şi suficiente pentru a reprezenta grafic rezultatul analizei: data calendaristică, respectiv numărul de ocurenţe din data respectivă.

Reprezentarea grafică a rezultatului se face prin intermediul unui script implementat în JavaScript. Este vorba despre un API pus la dispoziţie de Google, ce facilitează reprezentarea grafică a datelor de intrare: Annotation Chart.[34]

Annotation Chart este doar un suport pentru reprezentarea grafică a datelor. Pentru afişarea în mod dinamic a graficului am avut în vedere modificarea scriptului pus la dispoziţie de catre Google, astfel încât să fie îndeplinite funcţiile sistemului Media Analytics.

Pentru manipularea graficului este necesară definirea unui secţiuni în pagina HTML (e.g., sugestiv `<div id="chart_div" style='width: 900px; height: 400px;'></div>`) asupra căreia este aplicat scriptul care generează reprezentarea grafică (i.e., `drawChart()`). După cum am precizat, datele extrase din documentul XML sunt oferite ca parametri în API-ul *Google Annotation Chart*. Pentru ca afişarea grafică a analizei să se facă în mod dinamic (i.e., graficul să se transforme în concordanţă cu modificarea criteriilor de căutare) am iterat prin elementele documentului XML, am extras datele necesare şi le-am transmis ca parametri în cadrul graficului. Modul în care am realizat aceste operaţii este redat în fragmentul de cod de mai jos:

```
var data = new google.visualization.DataTable();
data.addColumn('date', 'Date');
data.addColumn('string', keyword);
data.addColumn('string', 'News Title');
data.addColumn('string', 'News Text');
for (itemIndex = 0; itemIndex < x.length; itemIndex++) {
    var count = (x.item(itemIndex).getAttribute('count'));
    var date = (x.item(itemIndex).getAttribute('date'));
    if (Number(count) !== 0)
        data.addRows([
            [new Date(date), Number(count), '', '']
        ]);
    else
        data.addRows([
            [new Date(date), Number(count), undefined, undefined]
        ]);
}
```

De asemenea, în reprezentarea grafică a analizei nu am folosit opţiunea de adnotare a graficului, deoarece momentan aceasta nu corespunde modului de implementare a sistemului.

Reprezentarea grafică a analizei efectuate (i.e., analiza sentimentelor bazată pe metoda de counting) are loc efectiv în urma apelării metodei `draw(data, options)` din cadrul API-ului *Google Annotation Chart*: `chart.draw(data, options);`.

4.4 INTERFAŢA GRAFICĂ

Media Analytics este expus utilizatorului sub forma unei aplicaţii web, care realizează interfaţa dintre acesta şi componenta de analiză.

Pentru structurarea paginii web am folosit HTML, iar pentru separarea conţinutului HTML de stilul de afişare în pagină am utilizat CSS (Cascaded Style Sheet).

Pagina web urmăreşte structura unui document HTML standard. În secţiunea `<head></head>` a fost definită calea către fişierul CSS extern paginii (i.e., acesta este situat în directorul proiectului), stiluri CSS care nu au putut fi preluate din exterior, respectiv scripturile care

rulează în pagină. Secţiunea `<body></body>` a fost structurată folosind elemente de tip `<div></div>`, ce descriu funcţia fiecărei subsecţiuni.

Clientul interacţionează cu interfaţa grafică într-un mod simplu, deoarece aceasta este intuitivă, oferind o fereastră de căutare care conţine o serie de opţiuni pentru filtrarea datelor de intrare: selectarea portalului de ştiri, selectarea categoriei, dar şi a intervalului temporal pentru care se doreşte analiza datelor.

Meniul de căutare face parte din subsecţiunea descrisă de elementul `<div class="createBorder">`. Atributul `createBorder` reprezintă un stil de reprezentare definit în fişierul *index.css*, în care sunt stabiliți o serie de parametri cu rolul de a crea caseta de căutare (e.g., border, background, width, border-radius, etc.). Câmpurile care stabilesc filtrele aplicate la intrarea sistemului sunt elemente de tip `<input></input>`, fiecare dintre acestea fiind indentificate corespunzător. De exemplu, câmpul în care trebuie specificat termenul căutat, este identificat cu `id="keyword"`:

```
<input type="text" name="keyword" id="keyword" required placeholder="What to search?" >
```

Aceste indentificatoare sunt folosite în procesul de căutare şi analiză a datelor (i.e., vezi secţiunile 4.2, respectiv 4.3 ale acestui capitol).

Selectarea intervalului temporal se realizează pe baza unor scripturi JavaScript descrise în pagină, care menţin în permanenţă actualizat un calendar, astfel încât utilizatorul să poată alege intervalul temporal dorit. Modul în care a fost definit scriptul care formatează data de început a căutării este redat în fragmentul de cod de mai jos:

```
<script type="text/javascript">
    $(function() {
        var pickerOpts = {
            dateFormat: "yy.mm.dd"
        };
        $("#datepicker1").datepicker(pickerOpts);
    });
</script>
```

Pentru menţinerea atualizată a datei, acest script are nevoie de accesarea resurselor stocate online, pe serverul *jQuery user interface*.

Reprezentarea grafică a rezultatului se face prin intermediul unui script implementat integral în JavaScript (i.e., vezi subsecţiunea 4.3 a acestui capitol). Efectele grafice şi modul în care interfaţa grafică este expusă utilizatorului au fost prezentate detaliat în CAPITOLUL 3, secţiunea 3.1.

CAPITOLUL 5

CONCLUZII

5.1 CONCLUZII GENERALE

În prezent, societatea informaţională se confruntă cu aşa numitul concept „Big Data”. Acesta descrie seturi ample de date, care pot fi analizate programatic, pentru a descoperi modele, tendinţe şi asocieri, în special cu privire la comportamentul uman şi la interacţiunile interumane.

Analiza datelor implică procese complexe de NLP şi în principiu are loc o analiză a sentimentelor, care urmărește să identifice atitudinea unei anumite persoane (i.e., atitudine exprimată în scris sau oral) referitoare la o anumită topică sau la contextul unui document.

Dintre mecanismele de analiză a datelor descrise în CAPITOLUL 4, subsecţiunea 4.1 a prezentei lucrări de diplomă, am ales implementarea analizei bazate pe metoda de counting, datorită raportului favorabil între relevanţa rezultatelor obţinute şi gradul de dificultate al implementării. Rezultatele obţinute devin relevante atunci când algoritmul de analiză este aplicat asupra unei baze de date amplă, iar metoda prezintă un grad redus de dificultate, deoarece implementează cea mai simplă formă de stabilire a unor statistici (i.e., determinarea frecvenţei de apariţie a unor termeni în mediul online, analizând în general interesul utilizatorilor cu privire la acei termeni).

Obiectivul central al acestei lucrări de diplomă a fost de a expune utilizatorului, prin intermediul unei interfeţe web, diferite opţiuni de analiză asupra conţinutului media achiziţionat în prealabil şi stocat într-o bază de date. Prin urmare, s-a dezvoltat un sistem de analiză a presei scrise

online, capabil să ofere funcţia de analiză a datelor pe baza metodei de counting. De asemenea, a fost asigurată o bază de date care conţine text procesat necesar în antrenarea modelelor acustice, utilizate în sisteme de RAV (i.e., Recunoaştere Automată a Vorbirii).

Având în vedere varietatea articolelor achiziţionate, analiza sentimentelor se extinde în diferite domenii (e.g., economie, politică, cultură, etc.). Astfel, serviciul web de analiză se dovedeşte a fi util în situaţia în care o companie ar vrea să vadă ce impact produce o campanie de marketing în presa scrisă online, sau în situaţia în care o persoană influentă vrea să afle cât de des este menţionată în articolele de ştiri, în corelaţie cu desfăşurarea unor evenimente importante care o includ.

5.2 CONTRIBUŢII PERSONALE

Contribuţiile personale ale autorului prezentei lucrări de diplomă sunt descrise în CAPITOLUL 3, respectiv CAPITOLUL 4 astfel:

a) Descrierea conceptuală a întregului sistem şi prezentarea succintă a modulului de achiziţie a datelor (i.e., CAPITOLUL 3).

b) Implementarea algoritmului de analiză a datelor, a componentei ce vizează căutarea în baza de date indexată şi a interfeţei web ce permite stabilirea criteriilor de căutare şi afişarea rezultatelor (i.e., CAPITOLUL 4, subsecţiunile 4.2, 4.3, respectiv 4.4).

5.3 ÎMBUNĂTĂŢIRI VIITOARE ALE SISTEMULUI

Activităţile viitoare privind dezvoltarea sistemului Media Analytics se vor concentra pe integrarea celorlalte mecanisme în algoritmul de analiză a datelor.

În primă instanţă, se urmăreşte realizarea corelaţiei între subiecte. În loc de construirea unei singure structuri de filtre, pentru fiecare subiect analizat va fi creată o structură de filtre specifică. În urma analizării datelor filtrate de aceste structuri, vor fi generate corespunzător o serie de documente XML (i.e., care reprezintă rezultatul analizei de tip counting pentru fiecare dintre subiectele analizate). În plus, va trebui modificat scriptul (i.e., care realizează reprezentarea grafică a rezultatelor), astfel încât să suporte reprezentarea pe acelaşi grafic a mai multor funcţii.

În al doilea rând, se vizează extinderea analizei sentimentelor asupra conţinutului audiovizual. O primă etapă în realizarea acestei funcţii ar fi identificarea furnizorilor de conţinut audiovizual şi a modului în care poate fi achiziţionat un astfel de conţinut media (e.g., pentru portalurile de ştiri care oferă astfel de conţinut, trebuie ca pe lângă ştirea în format audio/video, să existe şi transcrierea textuală a acesteia). Un sistem de analiză a conţinutului audiovizual ar oferi date în format audio, împreună cu transcrierea textuală a acestora, necesare în implementarea unui sistem de RAV.

De asemenea, se urmăreşte ca atunci când utilizatorul selectează un punct pe grafic, sistemul să afişeze detalii suplimentare privind analiza subiectului vizat (e.g., în contextul actual, atunci când utilizatorul selectează un punct pe grafic, ar trebui ca serviciul web să întoarcă şi titlurile, însoţite de text pentru toate ştirile din ziua respectivă).

BIBLIOGRAFIE

[1] [Trends] Google Trends API (https://www.google.com/trends)

[2] [David Reilly, 1999] *Inside Java: The Java Programming Language* (http://www.javacoffeebreak.com/articles)

[3] [JAPI] Java API (http://www.techopedia.com/definition/25133/application-programming-interface-java-api)

[4] [WCRL] Web Crawler (http://www.scs.ubbcluj.ro/~pmsd0954/referat.html#resursa2)

[5] [RSSF] RSS Feed (https://ro.wikipedia.org/wiki/RSS)

[6] [Dan Galaţchi, 2015] *Inginerie software pentru comunicaţii - Note de curs, 2015*

[7] [Postgres] PostgreSQL (https://en.wikipedia.org/wiki/PostgreSQL)

[8] [PostAV] PostgreSQL Advantages (http://www.postgresql.org/about/advantages/)

[9] [Bogdan Ionescu, 2013] *Conceptul de Indexare automată după conţinut în contextul datelor multimedia*, 2013

[10] [APLUC] Apache Lucene (https://en.wikipedia.org/wiki/Lucene)

[11] [Mihai Gabroveanu, 2006] *Introducere în XML, 2006* (http://inf.ucv.ro/~mihaiug/courses/xml)

[12] [HTML] HyperText Markup Language (http://ro.wikipedia.org/wiki/HyperText_Markup_Language)

[13] [JS] JavaScript (http://ro.wikipedia.org/wiki/JavaScript)

[14] [Ana-Cristina Olteanu] *Arhitectura serviciilor web*

[15] [ADV] Ziarul online Adevărul (http://adevarul.ro)

[16] [9AM] Ziarul online Nine AM(http://www.9am.ro/)

[17] [WST] Ziarul online Wall-Street(http://Wall-Street.ro)

[18] [DBCON] Conectarea la baza de date PostgreSQL (https://jdbc.postgresql.org/documentation/80/connect.html)

[19] [Turney, 2002] Turney, Peter. "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews". *Proceedings of the Association for Computational Linguistics*. pp. 417–424.

[20] [Pang, 2002] Pang, Bo; Lee, Lillian; Vaithyanathan, Shivakumar. "Thumbs up? Sentiment Classification using Machine Learning Techniques". *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp. 79–86.

[21] [Pang, 2005] Pang, Bo; Lee, Lillian. "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales". *Proceedings of the Association for Computational Linguistics (ACL)*. pp. 115–124.

[22] [Snyder, 2007] Snyder, Benjamin; Barzilay, Regina. "Multiple Aspect Ranking using the Good Grief Algorithm". *Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (HLT-NAACL)*. pp. 300–307.

[23] [Vryniotis, 2013] Vryniotis, Vasilis. *The importance of Neutral Class in Sentiment Analysis*.

[24] [Koppel, 2006] Koppel, Moshe; Schler, Jonathan. "The Importance of Neutral Examples for Learning Sentiment". *Computational Intelligence 22*. pp. 100–109

[25] [Thelwall, 2010] Thelwall, Mike; Buckley, Kevan; Paltoglou, Georgios; Cai, Di; Kappas, Arvid."Sentiment strength detection in short informal text". *Journal of the American Society for Information Science and Technology* **61** (12): 2544–2558.

[26] [Mihalcea, 2007] Mihalcea, Rada; Banea, Carmen; Wiebe, Janyce. "Learning Multilingual Subjective Language via Cross-Lingual Projections" (PDF). *Proceedings of the Association for Computational Linguistics (ACL)*. pp. 976–983

[27] [Su, 2008] Su, Fangzhong; Markert, Katja. "From Words to Senses: a Case Study in Subjectivity Recognition". *Proceedings of Coling 2008, Manchester, UK*.

[28] [Pang, 2004] Pang, Bo; Lee, Lillian. "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts". *Proceedings of the Association for Computational Linguistics (ACL)*. pp. 271–278.

[29] [Hu, 2004] Hu, Minqing; Liu, Bing. "Mining and Summarizing Customer Reviews".*Proceedings of KDD 2004*

[30] [Liu, 2005] Liu, Bing; Hu, Minqing; Cheng, Junsheng (2005). "Opinion Observer: Analyzing and Comparing Opinions on the Web". *Proceedings of WWW 2005*.

[31] [Ogneva] Ogneva, M. "How Companies Can Use Sentiment Analysis to Improve Their Business".

[32] [NSG] Studiul *Nielsen/SocialGuide*, 2013

[33] [NSGRes] Rezultatele studiului *Nielsen/SocialGuide, 2013*

[34] [GAC] Google Annotation Chart (https://developers.google.com/chart/interactive/docs/gallery/annotationchart)