UNIVERSITATEA “ALEXANDRU IOAN CUZA“ IAŞI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Coordonator ştiinţific

Conf. Dr. Adrian Iftene

Absolvent

Gabor Silviu

IAŞI

2017

UNIVERSITATEA “ALEXANDRU IOAN CUZA“ IAŞI

FACULTATEA DE INFORMATICĂ

RECOMANDĂRI DE REPERTORII GITHUB

Coordonator ştiinţific

Conf. Dr. Adrian Iftene

Absolvent

Gabor Silviu

IAŞI

2017

DECLARAȚIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licență cu titlul "Recomandări de repertorii GitHub" este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi dețin referința precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
- codul sursă, imagini etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor şi dețin referințe precise;
- rezumarea ideilor altor autori precizează referința precisă la textul original.

Iaşi, ${day}.07.2017

Absolvent Gabor Silviu

_____

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul "Recomandări de repertorii GitHub", codul sursă al programelor şi celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

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

Iaşi, ${day}.07.2017

Absolvent Gabor Silviu

_____

Acord privind proprietatea dreptului de autor

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, format executabil şi sursă, să aparţină autorului prezentei lucrări, Gabor Silviu.

Încheierea acestui acord este necesară din următoarele motive:

În viitor, doresc continuarea dezvoltării acestei aplicaţii, precum şi extinderea pe alte platforme.

Iaşi,

Decan *Adrian Iftene*

_____

Absolvent *Gabor Silviu*

_____

Cuprins

1. Introducere

1.1. Motivaţie

Prezenta lucrare de licenţă doreşte a oferi o soluţie pentru dezvoltarea mai uşoară a proiectelor de diferite tipuri, pornind de la o idee sau de la un proiect deja conceput şi distribuit pe diferite platforme ce găzduiesc aplicaţii dezvoltate de comunitatea GitHub[1]. Pe scurt, ideea acestei lucrări gravitează în jurul conceptului de recomandare pe baza unor descrieri de tip text, ce conţin un scurt rezumat în legătura cu funcţionalitatea ce se doreşte a fi exploatată, precum şi limbajele de programare implicate sau tehnologiile ce se urmăresc a fi utilizate.

Decizia alegerii acestei teme a fost luată în urma numeroaselor discuţii cu coordonatorul de licenţă, luând în considerare necesitatea firmelor de IT care au nevoie de un punct de plecare în urma discuţiilor cu clienţii care îşi manifestă dorinţa (nu neapărat clară) de implementare a unor funcţionalităţi ale unor aplicaţii deja existente sau ale unor proiecte care vor trebui dezvoltate încă de la început.

Lucrarea este structurată în patru mari capitole, care vor constitui un adjuvant în înţelegerea modului de funcţionare a aplicaţiei, în clarificarea felului în care au fost îmbinate tehnologiile utilizate pentru a ajunge la produsul final, dar şi în lămurirea metodelor prin care aplicaţia curentă ar putea fi îmbunătăţită.

Primul capitol conţine detalii despre noţiunile folosite în cadrul descrierii funcţionalităţilor aplicaţiei, la nivel de bază. De asemenea, vor fi menţionaţi şi descrişi, la nivel conceptual, algoritmii utilizaţi pentru realizarea recomandărilor, precum şi librăriile care oferă implementarea lor.

În capitolul doi, sunt descrise, în mod individual, fiecare framework utilizat pentru realizarea părţii practice a prezentei lucrări de licenţă.

Cel de-al treilea capitol constituie o imagine de ansamblu asupra arhitecturii aplicaţiei şi oferă detalii despre fiecare modul în parte: librării folosite, algoritmii care stau la bază, relaţia cu alte module şi modul lor de comunicare.

Ultimul capitol conţine informaţii în legătură cu problemele întâmpinate, posibilităţile de îmbunătăţire a recomandărilor (perspective asupra altor algoritmi ce ar putea fi folosiţi), precum şi extensiile ce ar putea constitui o direcţie pe viitor pentru îmbunătăţirea interactivităţii cu

utilizatorii aplicaţiei.

1.2. GitHub

GitHub este un serviciu web de găzduire pentru proiecte software.

Întrucât sistemul de recomandare are la bază un model construit în urma unei clusterizări folosind algoritmul K-means din cadrul librăriei Apache Spark MLlib, am căzut de acord cu profesorul de licență să folosesc drept instanţe descrierile unor repertorii, luate aleator, folosind un crawler construit de mine care apelează API-ul expus de GitHub. De menţionat este faptul că, deoarece repertoriile au fost descărcate aleator, nu s-au putut controla instanţele de proiecte împreună cu descrierile lor, motiv pentru care sistemul de recomandare nu poate funcţiona în mod optim (există descrieri care nu au relevanţă în ceea ce priveşte conţinutul repertoriului sau există proiecte ale căror descrieri sunt prezentate mult prea general).

1.3. Concluzii

Acest capitol constituie o scurtă introducere în cadrul aplicaţiei prezentate, descriindu-se sumar componentele constitutive, împreună cu problemele care pot avea o influenţă asupra corectitudinii rezultatului final.

2. Definiţii şi exemple

2.1. Clusterizare de documente folosind învăţarea automată

Algoritmii de învăţare nesupervizată din cadrul domeniului de Învăţare automată impun o structură peste mulţimi de date neetichetate.

Clusterizarea este cea mai comună formă de învăţare nesupervizată şi reprezintă diferenţa majoră între clusterizare şi clasificare. În cadrul clusterizării, ceea ce determină apartenenţa la un anumit cluster este distribuţia şi natura datelor, spre deosebire de clasificare, unde clasificatorul învaţă asocierea dintre obiecte şi/sau clase provenite din aşa-numitele ”date de antrenament” (o mulţime de date corect etichetate la mână), iar apoi replică comportamentul învăţat asupra datelor ne-etichetate. Câteva dintre aplicaţiile frecvent întâlnite în clusterizare sunt: găsirea documentelor similare, organizarea de colecţii mari de documente, detectarea de conţinut duplicat, sisteme de recomandare şi optimizare de căutări.

K-means este un algoritm proiectat pentru a identifica grupuri de date între care există o strânsă legătură, în funcţie de caracteristicile alese ce vor constitui aspectele definitorii pentru instanţele din mulţimea considerată.

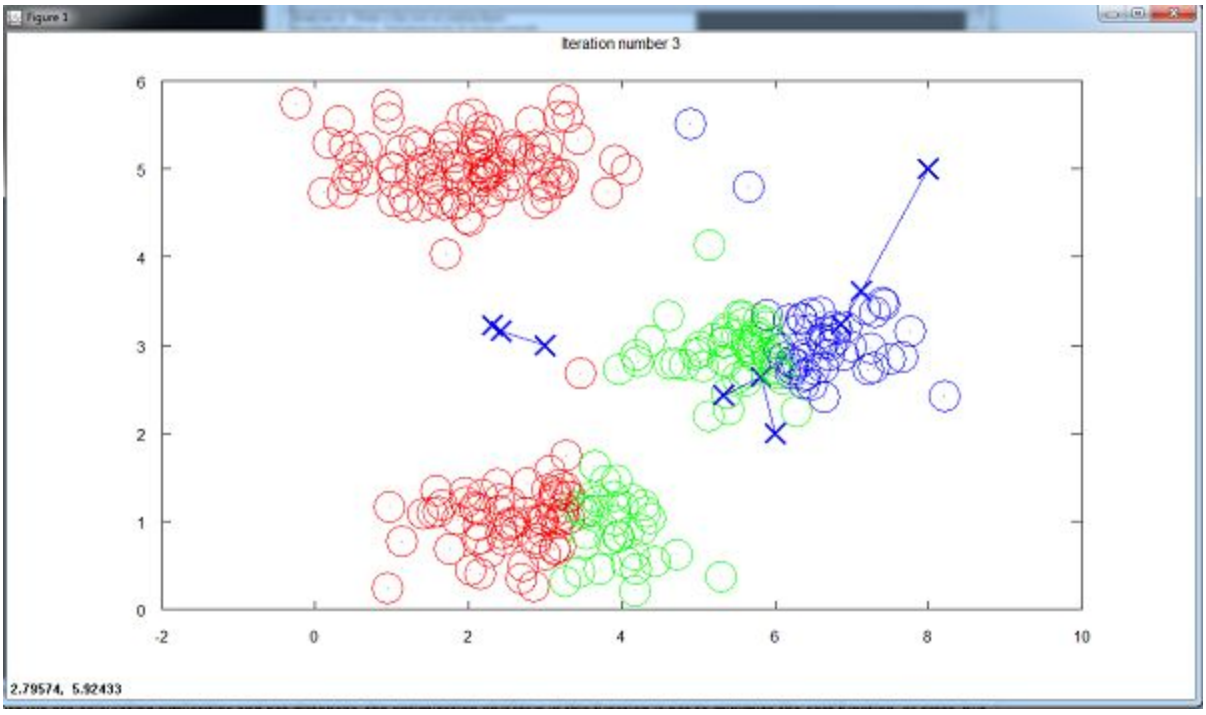

Figura 1: Schiţă în care datele sunt reprezentate ca puncte, iar figurile în formă de ”*x*” constituie centroizii a căror poziţie este stabilită la iteraţia numărul 3 [1]

În imaginea de mai sus, culorile reprezintă delimitarea grupurilor realizată de algoritmul K-means în timpul iteraţiei numărul 3.

Un pseudo-cod pentru acest algoritm [2] este următorul:

Input: $x_1, x_2, \ldots, x_n \in R^d$, cu $n \geq K$

Output: o anumită K-partiţie pentru $\{x_1, x_2, \ldots, x_n\}$

Procedură:

[*Iniţializare/Iteraţia 0:*] $t \leftarrow 0$;

se fixează în mod arbitrar $\mu_1^0, \ldots, \mu_K^0$, centroizii iniţiali ai clusterelor şi, apoi, se asignează fiecare instanţă x_i la centroidul cel mai apropiat, formând astfel clusterele $C_1^0, \ldots, C_K^0$.

[*Recursivitate*:] Se execută iteraţia ++t:

Pasul 1: se calculează noile poziţii ale centroizilor:

$$\mu_j^t = \frac{1}{|C_j^{t-1}|} \sum_{x_i \in C_j^{t-1}} x_i, \ \forall j = 1 \ldots K$$

Pasul 2: se reasignează fiecare x_i la [clusterul cu] centroidul cel mai

apropiat, adică se stabileşte noua componenţă a clusterelor la iteraţia t: $C_1^t, \ldots, C_K^t$; [*Terminare*:] până când o anumită condiţie este îndeplinită (de exemplu: până când poziţiile centroizilor — sau: componenţa clusterelor — nu se mai modifică de la o iteraţie la alta).

În cazul clusterizării de texte, sarcina este diferită: setul de date îl constituie un grup de documente text, motiv pentru care nu este imediat evident modul cum ar trebui reprezentate aceste puncte în spaţiu. Totodată, obiectivul unei scheme bune de clusterizare a documentelor este cel de a minimiza distanţa intra-clustere între documente, în timp ce se maximizează distanţa inter-clustere.

Din aceste considerente, vom face o scurtă introducere în cadrul unei extensii pentru acest algoritm: Bisecting K-means.

- **Algoritmul Bisecting K-means**

Această metodă constituie un fel de clusterizare ierarhică care se foloseşte de algoritmul K-means. Acest algoritm îşi începe execuţia prin a pune întreaga mulţime de documente într-un singur cluster, apoi clusterul original este partiţionat în alte două clustere folosind K-means (cu $K = 2$). Clusterul care are cea mai ridicată similaritate intra-clustere este salvat, apoi celălalt cluster este recursiv împărţit în alte două clustere folosind acelaşi algoritm (K-means, cu $K = 2$). Această operaţie se realizează până când este obţinut numărul dorit de clustere.

Un pseudo-cod pentru acest algoritm [3] este următorul:

```
Input: K - numărul de clustere, D - primele N documente obţinute prin
similaritatea vectorilor din spaţiu
Output: cele K clustere
Procedură:
      pune cele N documente într-un singur cluster, C;
      Buclă: for i=1 to K-1 do
            Buclă: for j=1 to ITER do
                  foloseşte K-means pentru a împărţi clusterul C în două
                  sub-clustere, C1 şi C2
                  if ( similaritea_intra_clustere(C1) >
                  similaritea_intra_clustere(C2) )
                        salvează clusterul C1
                        C = C2
                  else
                        salvează clusterul C2
                        C = C1
                  end_if
```

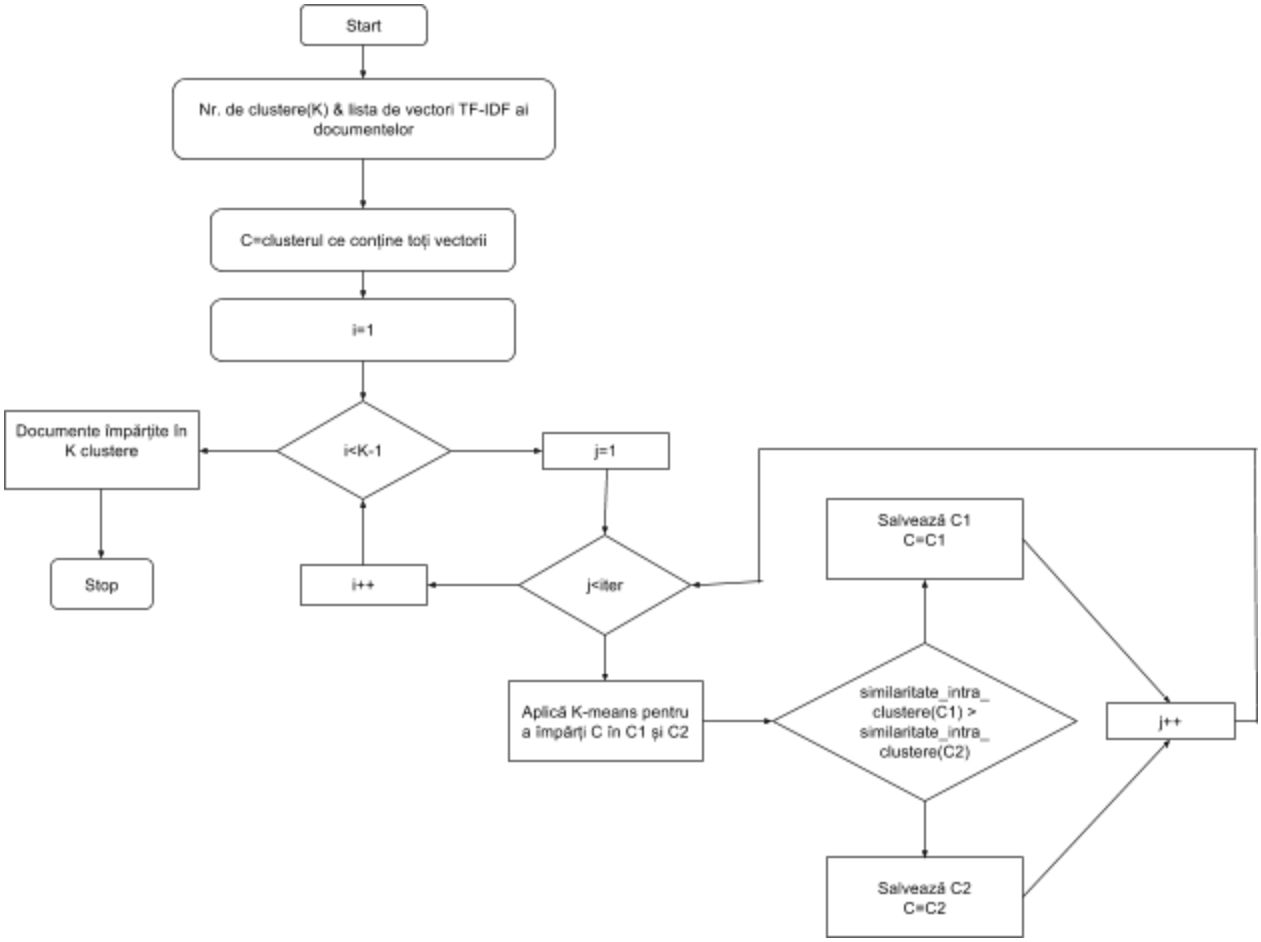

Figura 2: Schiţă a modului de funcţionare pentru Bisecting K-means

➢ *Complexitatea timp*

Bisecting K-Means foloseşte K-Means pentru a genera doua sub-clustere. Întrucât K-Means rulează în O(N), complexitatea-timp a algoritmului va fi O((K-1)IN), unde I reprezinta numărul de iteraţii până la convergenţă. În concluzie, Bisecting K-Means rulează în timp liniar cu dimensiunea documentelor.

2.2. Preprocesarea instanţelor

Întrucât în cazul clusterizării de texte nu este foarte evident modul cum ar trebui reprezentate documentele în spaţiu, vom prezenta transformările care au loc pentru fiecare instanţă de document, astfel încât, dintr-o simplă descriere de proiect, va rezulta, în final, un vector caracteristic în spaţiul $\mathbf{R^n}$ ce va reprezenta conţinutul descrierii elementare.

➢ *Cleanser*

Un Cleanser este o clasă ce extinde clasa Transformer prezentă în librăria Spark MLlib care are rolul de a şterge semnele de punctuaţie (”white-spaces”) dintr-o descriere a unui proiect, proces necesar pentru acţiunea ulterioara de izolare corectă a cuvintelor. Această acţiune este realizată prin intermediul unei expresii regulate, înlocuind fiecare potrivire cu un şir de caractere vid.

➢ *Tokenizer*

Tokenizer este o clasă ce aparţine aceleiaşi librării mai sus menţionate, cu o implementare oferită de Apache pentru a ”sparge în cuvinte” un şir de caractere, rezultând o listă de şiruri ce constituie cuvintele care alcătuiesc seria de caractere iniţială. Astfel are loc izolare fiecărui cuvânt în parte, rezultat ce va constitui un input pentru următoarea transformare.

➢ *Stop Words Remover*

StopWordsRemover defineşte un set de cuvinte care apar cu o frecvenţă foarte mare într-o anumita limbă definită şi care constituie părţi de propoziţie nerelevante în cadrul unei descrieri, reprezentând doar componente adjuvante în exprimarea fluentă a unei idei. Întrucât această mulţime nu defineşte un aspect important în rezultatul final (ba chiar îl poate influenţa într-un mod greşit), rolul acestei clase oferite de Apache este următorul: lexemele din această colecţie care se regăsesc în descriere vor fi eliminate.

➢ *Stemmer*

Un Stemmer este clasa a cărei sarcini este cea de a aduce la forma din dicţionar cuvintele din descriere. Acest proces este necesar întrucât algoritmul TF-IDF prezentat mai jos calculează frecvenţa de apariţie a cuvintelor pentru a construi o pondere ce reprezintă nivelul de importanţă al cuvântului în prezentarea proiectului. Întrucât cuvintele pot fi folosite ca diferite părţi de propoziţie, dar pot face trimitere la aceeaşi idee, fără o astfel de transformare, acestea vor fi tratate drept cuvinte separate şi, drept urmare, există posibilitatea unui rezultat eronat. Această clasă se foloseşte de o implementare oferită de librăria Snowball [4].

➢ *Algoritmul TF-IDF* [5]

Acest algoritm constituie cea mai importantă transformare de preprocesare a instanţelor de descrieri întrucât aici are loc conversia unei liste de şiruri de caractere într-un vector în care fiecare componentă reprezintă un număr real.

TF-IDF (abreviere de la ”term frequency - inverse document frequency”) reprezintă o statistică numerică destinată reflectării gradului de importanţă a unui cuvânt pentru un document prezent într-o colecţie sau într-un corpus. Este des întâlnit în momentul în care se doreşte a se utiliza un factor de greutate în regăsirea de informaţii sau în text-mining.

Valoarea TF-IDF asociată unui cuvânt creşte proporţional cu numărul de apariţii ale cuvântului respectiv într-un text, dar această valoare este echilibrată şi de frecvenţa cuvântului din corpus, ceea ce aduce un echilibru prin faptul că unele cuvinte apar mai frecvent în mod obişnuit. Astăzi, TF-IDF reprezintă una din cele mai populare scheme de asociere a unei ponderi pentru un termen, iar algoritmul este folosit în aproximativ 83% din sistemele de recomandare pe bază de text în contextul librăriilor digitale.

TF-IDF este produsul a două statistici, ”term frequency” şi ”inverse document frequency”.

1. *Term Frequency*

În cazul acestui tip de statistică, *tf(t, d)*, abordarea constă în a memora numărul de apariţii ale termenului *t* în documentul *d*. Notând cu $f_{t,d}$ această numerare, schema devine una simplă:

$$tf(t,\ d) = f_{t,d}$$

2. *Inverse Document Frequency*

Inverse Document Frequency este o statistică ce măsoară cantitatea de informaţii pe care un cuvânt le poate furniza, adică deducem dacă termenul respectiv este comun sau rar în întreaga colecţie de documente. Matematic vorbind, este fracţia inversă şi logaritmic scalată a documentelor ce conţin cuvântul respectiv, obţinută prin divizarea numărului total de documente la numărul documentelor ce conţin termenul luat în considerare, apoi calculat logaritmul acestui coeficient:

$$idf(t,\ D) = \frac{N}{|\{d \in D \mid t \in d\}|}$$

unde:

- N reprezintă numărul total de documente din corpus;
- $|\{d \in D \mid t \in d\}|$ reprezintă numărul de documente în care apare termenul *t*. Dacă termenul nu se află în corpus, acest lucru ar putea duce la împărţirea la zero, motiv pentru care este normal a se ajusta numitorul la $1 + |\{d \in D \mid t \in d\}|$.

În final, valoarea TF-IDF pentru un termen *t* aparţinând unui document *d* dintr-o colecţie *D* se calculează astfel:

$$tfidf(t,\ d,\ D) = tf(t,\ d) \cdot idf(t,\ D)$$

Astfel, pentru fiecare descriere de-a unui proiect, va rezulta un vector în care fiecare element reprezintă valoarea TF-IDF asociată fiecărui termen ce apare în descriere.

➢ <u>*Normalizer (normalizare în L_2)*</u>

Fie vectorul $v = (v_1,\ v_2, \dots, v_n)$. Rezultatul normalizării vectorului *v* în L_2 este vectorul $v' = (v_1',\ v_2', ..., v_n')$, unde $v_i' = \frac{v_i}{\|v\|_2}\ \forall i = 1..n$, iar $\|v\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$ reprezintă *norma Euclidiană* [6] a vectorului *v*.

De asemenea, definim *similaritatea cosinus*[7] între doi vectori *u* şi *v* ca fiind:

$$cos(\theta) = \frac{u \cdot v}{\|u\|_2 \cdot \|v\|_2} = u' \cdot v'$$

unde u' şi v' sunt vectorii rezultaţi în urma normalizării în L_2 a vectorilor *u* şi *v*.

Întrucât pentru oricare vector $v \in R^n$ normalizat în L_2 avem $\|v\|_2 = 1$, definim *distanţa Euclidiană ridicată la pătrat*[8] între doi vectori *u* şi *v* normalizaţi în L_2 ca fiind:

$$\|u - v\|_2^2 = (u - v)^T \cdot (u - v) = u^T u - 2 \cdot u^T \cdot v + v^T \cdot v = 2 - 2 \cdot u^T \cdot v = 2 - 2 \cdot cos < (u, v)\ (1).$$

Din relaţia (1), putem deduce faptul că *distanţa Euclidiană ridicată la pătrat* între doi vectori normalizaţi în L_2, *u* şi *v,* este proporţională cu *distanţa cosinus* (*similaritatea cosinus*) între aceiaşi vectori.

În urma observaţiilor menţionate mai sus, putem afirma faptul că, pentru a măsura similaritatea cosinus între doi vectori TF-IDF asociaţi unor două descrieri de proiecte, este îndeajuns să aplicăm o transformare a acestor vectori (folosind algoritmul a cărei implementare este oferită de Apache prin clasa Normalizer) ce constă în normalizarea lor în L_2 pentru ca, mai

apoi, folosind algoritmul K-means din cadrul librăriei Apache Spark MLlib, să putem obţine drept măsură de distanţă o măsură echivalentă (ca proporţie) cu similaritatea cosinus. Menţionez faptul că această ultimă transformare este necesară întrucât K-means foloseşte drept măsură de distanţă *distanţa Euclidiană* (care nu poate fi schimbată din cauza felului în care algoritmul de clusterizare a fost implementat de către cei de la Apache), iar pentru implementarea sistemului de recomandare, am avut nevoie de o măsură egală ca proporţie cu *similaritatea cosinus*.

În concluzie, prin simpla normalizare în L_2 a vectorilor obţinuţi prin algoritmul TF-IDF, vom putea obţine, în loc de *distanţa Euclidiană,* o măsură de similaritate proporţională cu *distanţa cosinus*.[8]

2.3. Diagrame UML[9]

UML[1] (Unified Modeling Language) este un limbaj standard, conceput de OMG (Object Management Group) în Ianuarie 1997 şi folosit pentru a specifica, vizualiza, construi şi documenta artefactele unui sistem software. Iniţial, a fost construit cu scopul de a surprinde comportamentul sistemelor atât software, cât şi non-software.

➢ *Obiectivele UML*

”O imagine valorează cât 1000 de cuvinte” este expresia care descrie cel mai bine limbajul UML. Conceptele orientate-obiect au fost introduse mult înainte ca acest standard să îşi facă apariţia. În acel moment, nu existau metodologii standard de organizare şi de consolidare a dezvoltării orientate-obiect, motiv pentru care standardul UML şi-a făcut apariţia.

Există numeroase motive pentru care UML a fost dezvoltat, dar cel mai important aspect este acela de a defini un limbaj de modelare pentru un scop general care să poată fi folosit de toţi modelatorii, dar în acelaşi timp să fie simplu de înţeles şi uşor de folosit.

Diagramele UML nu sunt destinate numai dezvoltatorilor, ci şi utilizatorilor de business, oamenilor de rând sau oricărei persoane interesate în a înţelege un sistem. Întrucât acesta (sistemul) poate fi atât software, cât şi non-software, întărim ideea că UML nu este, de fapt, o metodă de dezvoltare, ci, mai degrabă, este un însoţitor al unor procese ce are drept scop realizarea unui sistem de succes.

[1] https://ro.wikipedia.org/wiki/Unified_Modeling_Language

În concluzie, obiectivul acestui limbaj poate fi definit ca un simplu mecanism de modelare cu rolul de a modela toate sistemele practice posibile într-un mediu complex şi contemporan.

➢ *Diagrama UML de clase*[11]

În contextul programării orientate-obiect, o diagramă UML de clase este folosită pentru reprezentarea vizuală a claselor şi a interdependenţelor, taxonomiei şi a relaţiilor de multiplicitate dintre ele. Diagramele de clasă sunt folosite şi pentru reprezentarea concretă a unor instanţe de clasă, aşadar obiecte, şi a legăturilor concrete dintre acestea.

Principalele elemente ale unei diagrame de clase sunt:

- *Clasa* – este reprezentată sub forma unui dreptunghi în interiorul căruia este notat numele clasei. Opţional, în interiorul unei clase, pot fi reprezentate atributele şi metodele acesteia (doar semnătura).
- *Interfaţa* – este reprezentată sub forma unui dreptunghi în interiorul căruia este notat numele interfeţei. Ca şi alternativă, o interfaţă mai poate fi reprezentată în mod simplificat sub forma unui cerc în dreptul căruia este trecut numele interfeţei. Interfaţa poate fi considerată un caz particular de clasă.
- *Relaţia* - este reprezentată sub forma unei drepte orientate sau nu, care leagă două clase. O relaţie de asociere dintre două clase semnifică o relaţie de colaborare sau de înrudire între clasele conectate. Tipurile de relaţii posibile între două clase sunt: asociere (reprezentată în diagramă drept o dreaptă simplă), agregare (reprezentată ca o dreaptă orientată, la capătul căreia se află un romb gol), compoziţie (reprezentată ca o dreaptă orientată, la capătul căreia se află un romb umplut), implementare (reprezentată ca o dreaptă orientată punctată al cărei capăt conţine o săgeată goală) şi moştenire (reprezentată ca o dreaptă orientată al cărei capăt conţine o săgeată goală).

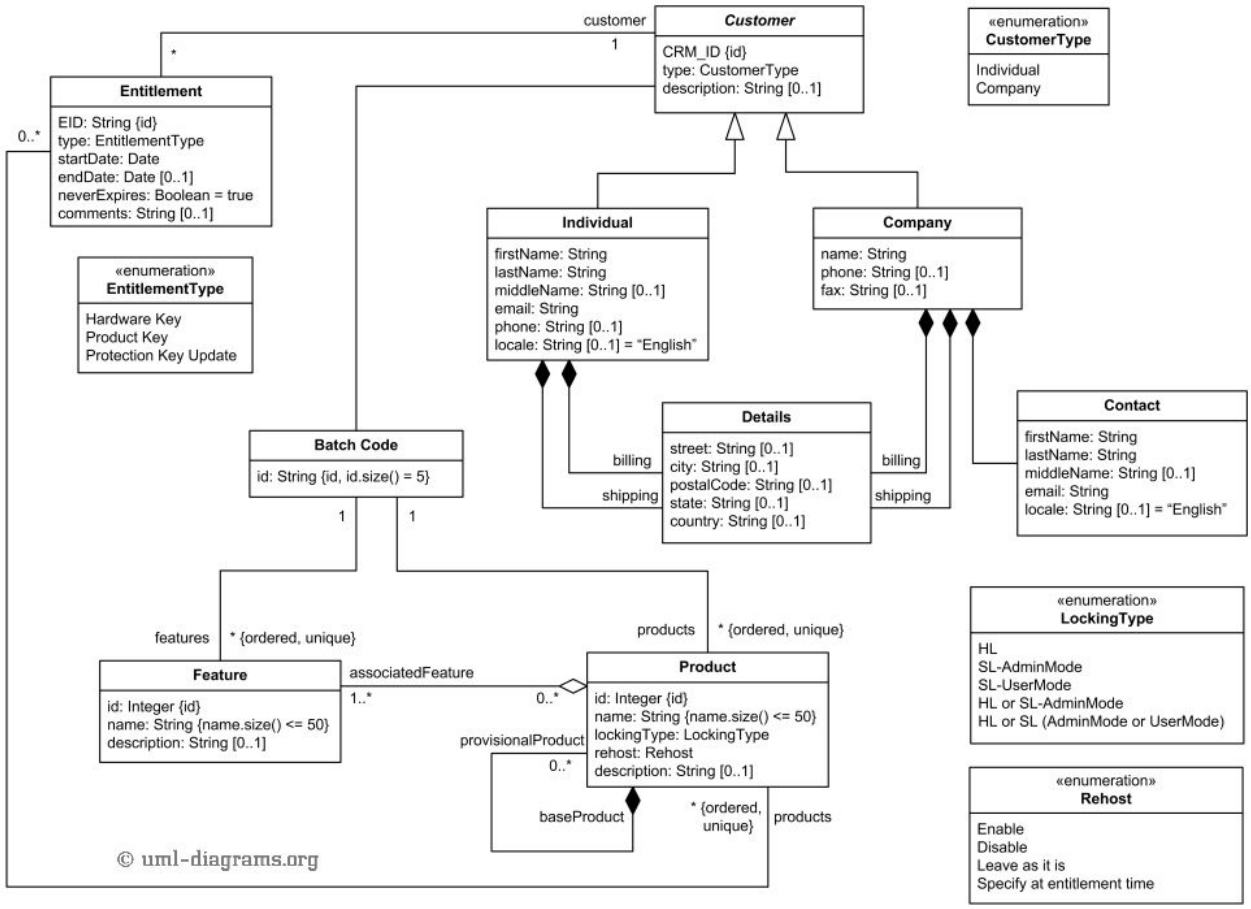

Figura 3: Exemplu de diagramă de clase care surprinde elementele componente, împreună cu relaţiile dintre ele, ale unui sistem destinat domeniului de "online shopping"[10]

2.4. Concluzii

În cadrul acestui capitol, am prezentat sumar algoritmii care intervin în sistemul de recomandare, stagiile de preprocesare a informaţiilor, precum şi suportul matematic necesar înţelegerii corecte a modului de funcţionare al transformărilor. Pentru etapa de învăţare automată, am definit algoritmii K-means şi Bisecting K-means, cel din urmă fiind folosit pentru clusterizarea de documente (secţiunea 2.1), precum şi algoritmul TF-IDF (secţiunea 2.2) de transformare a unor descrieri în vectori din spaţiul $\mathbf{R}^n$, vectori utilizaţi de algoritmii anterior menţionaţi. De asemenea, am prezentat necesitatea transformărilor şirurilor de caractere (secţiunea 2.2) prin metodele de Cleansing, Tokenization, Stop Words Removal, Stemming şi L_2 Normalization, iar nu în ultimul rând, am oferit un scurt suport teoretic pentru limbajul de modelare UML, limbaj folosit pentru a oferi suplimentar o diagramă de clase necesară înţelegerii mai bune a utilizatorului despre conţinutul proiectului recomandat.

3. Tehnologii folosite

3.1. Java EE

Java Platform Enterprise Edition (Java EE)[12] este un standard folosit în dezvoltarea de software pentru afaceri. J2EE[13] reprezintă o colecţie de tehnologii şi API-uri pentru platforma Java, proiectate pentru a veni în sprijinul aplicaţiilor de tip ”Enterprise” care, la rândul lor, pot fi clasificate, în general, ca aplicaţii de scară largă, distribuite, tranzacţionale şi disponibile pe o perioadă îndelungată de timp. Aceste aplicaţii au rolul de constitui suportul pentru cerinţele critice din partea de „business”.

Furnizorii produselor realizate cu Java EE sunt, de obicei, servere de aplicaţii, servere web sau furnizori de sisteme de baze de date care pun la dispoziţie clase ce implementează interfeţele definite în specificaţiile propuse de Java EE.

➢ *Diferenţe între Java EE şi Java SE* [14]

Tehnologia Java este atât un limbaj de programare, cât şi o platformă. Limbajul de programare Java este un limbaj de nivel înalt, orientat-obiect, având o sintaxă proprie. Platforma Java este un mediu de lucru în care sunt rulate aplicaţii scrise în limbajul de programare Java.

Toate platformele Java sunt alcătuite dintr-o maşină virtuală Java (Java Virtual Machine, JVM[2]) şi dintr-o interfaţă de programare a aplicaţiei (Application Programming Interface, API[3]). Java Virtual Machine este un program pentru platforme software şi hardware, care rulează aplicaţii dezvoltate prin intermediul tehnologiei Java. Un API este o colecţie de componente software care pot fi folosite pentru crearea altor componente sau aplicaţii software. Fiecare platformă Java furnizează câte o maşină virtuală şi un API, ceea ce permite aplicaţiilor scrise pentru platforma respectivă să poată fi executate în orice sistem compatibil, beneficiind, totodată, de avantajele limbajului de programare Java: independenţă de platformă, putere, stabilitate, uşurinţă de dezvoltare şi securitate.

❑ **Java SE**

În momentul în care majoritatea oamenilor se gândesc la limbajul de programare Java, se

[2] https://en.wikipedia.org/wiki/Java_virtual_machine
[3] https://ro.wikipedia.org/wiki/Application_Programming_Interface

referă, de fapt, la Java SE API. Acest API furnizează funcţionalitatea de bază al acestui limbaj: defineşte totul, începând de la tipurile şi obiectele de bază, până la clase de nivel înalt folosite pentru reţelistică, securitate, acces la baza de date, dezvoltare de interfeţe grafice (GUI) şi parsare de XML. În plus, faţă de API-ul nucleu, platforma Java SE este alcătuită dintr-o maşină virtuală, din unelte de dezvoltare, tehnologii de deployment, precum şi din alte librării sau seturi de instrumente folosite frecvent în aplicațiile Java.

- **Java EE**

Platforma Java EE este construită peste platforma Java SE şi furnizează atât un API, cât şi un mediu pentru dezvoltarea şi execuţia aplicaţiilor scalabile, sigure, multi-nivelate, de mari dimensiuni şi securizate în reţea.

3.2. Maven

Maven [15] este un proiect găzduit de Apache Software Foundation [16] şi constituie o unealtă folosită pentru automatizare la nivel de build, în special folosit în cadrul proiectelor Java: în primul rând, descrie cum este construită aplicația, apoi descrie dependenţele ei. Spre deosebire de uneltele precedente precum Apache Ant[17], Maven[15] foloseşte convenții pentru procesul de build şi doar excepţiile trebuiesc menţionate. Un fişier XML descrie proiectul de construit, dependenţele acestuia legate de modulele şi de componentele externe, ordinea în care acestea trebuiesc procesate, dar şi plug-in-urile necesare. Proiectul de la Apache vine laolaltă cu obiective predefinite în sensul în care are loc efectuarea unor sarcini bine definite, precum compilarea codului sau împachetarea acestuia.

Maven descarcă librăriile Java şi plug-in-urile Maven în mod dinamic, dintr-una sau mai multe repository-uri (cum ar fi Maven 2 Central Repository [18]), apoi le stochează într-un cache local. Tehnologii alternative precum Gradle nu se bazează pe XML, dar păstrează conceptele Maven introduse.

Un POM[4] (Project Object Model) furnizează toate informaţiile legate de configurarea unui proiect. Configuraţiile generale acoperă denumirea proiectului, proprietarul acestuia şi dependenţele de alte proiecte. Se pot configura, de asemenea, faze individuale pentru procesul de

[4] https://en.wikipedia.org/wiki/Apache_Maven#Project_Object_Model

build, stagii implementate ca plug-in-uri.

Aplicaţia de faţă, întrucât este de dimensiune medie, este divizată în module (sau sub-proiecte), fiecare modul având la rândul său propriul POM. Există un POM rădăcină care conţine informaţiile legate de compilarea tuturor modulelor proiectului de faţă, lucru care are ca efect posibilitatea compilării întregului proiect dintr-o singură linie de comandă.

Fişierul POM responsabil cu configuraţia unui proiect are denumirea ”pom.xml” şi, orientativ, are următorul conţinut:

```
<project>
  <modelVersion>4.0.0</modelVersion>

  <!-- coordonatele proiectului, adică un grup de valori care identifică în
mod unic proiectul-->

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>

  <!-- dependenţe de librării -->

  <dependencies>
    <dependency>

      <!-- coordonatele librăriei necesare -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- următorul tag semnifică faptul ca librăria de mai sus este folosită
în execuţia şi compilarea testelor -->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```

3.3. Spring

Spring [19] este un framework care asigură un model de configurare şi de programare inteligent pentru aplicaţii moderne scrise în Java, indiferent de platforma folosită. Un element cheie pe care Spring se bazează este suportul infrastructural la nivel de aplicaţie: Spring este orientat mai mult pe ”instalarea” aplicaţiilor de tip întreprindere, în aşa fel încât echipa de dezvoltare să îşi îndrepte atenţia asupra logicii de afaceri de la nivelul de aplicaţie, fără a mai fi

necesar să intre în configuraţiile ne-necesare ale mediului de ”deployment”.

Câteva dintre caracteristicile framework-ului sunt: dependency injection, programarea orientată pe aspecte (AOP) care include modulul de administrare a tranzacţiilor, aplicaţiile web MVC, serviciile web RESTful, suportul fundamental pentru JDBC[5], JPA[6], JMS[7], ş.a.m.d.

În continuare, vom prezenta modulele din cadrul framework-ului Spring folosite în prezenta lucrare de licenţă.

3.3.1. Spring IoC Container[20]

IoC[8] (Inversion of Control), cunoscut ca şi dependency injection[9] (DI), este un proces prin care obiectele îşi definesc dependenţele (prin dependenţe se înţeleg celelalte obiecte cu care lucrează) prin argumentele constructorilor, prin argumentele unei metode factory, sau prin proprietăţi fixate pe instanţa obiectului, după ce obiectul este construit sau returnat de o metodă de tip factory. Recipientul (container-ul), apoi, injectează acele dependenţe în momentul în care este creat bean-ul. Acest proces constă în inversiunea (de unde denumirea de IoC) bean-ului în sine, ce controlează instanţierea sau locaţia dependenţelor sale, folosind construirea directă a claselor sau utilizând un mecanism precum pattern-ul Service Locator[10].

În Spring, obiectele care alcătuiesc bazele aplicaţiei şi care sunt administrate de Spring IoC Container poartă denumirea de *beans*[11]. Un *bean* este un obiect instanţiat, asamblat şi, altminteri, administrat de un Spring IoC Container. Altfel spus, un *bean* este, de fapt, unul dintre multitudinea de obiecte din aplicaţie. Bean-urile, împreună cu dependenţele lor, sunt reflectate în metadatele de configurare folosite de container-ul în cauză.

[5] https://en.wikipedia.org/wiki/Java_Database_Connectivity
[6] https://en.wikipedia.org/wiki/Java_Persistence_API
[7] http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html
[8] https://en.wikipedia.org/wiki/Inversion_of_control
[9] https://en.wikipedia.org/wiki/Dependency_injection
[10] https://en.wikipedia.org/wiki/Service_locator_pattern
[11] https://en.wikipedia.org/wiki/JavaBeans

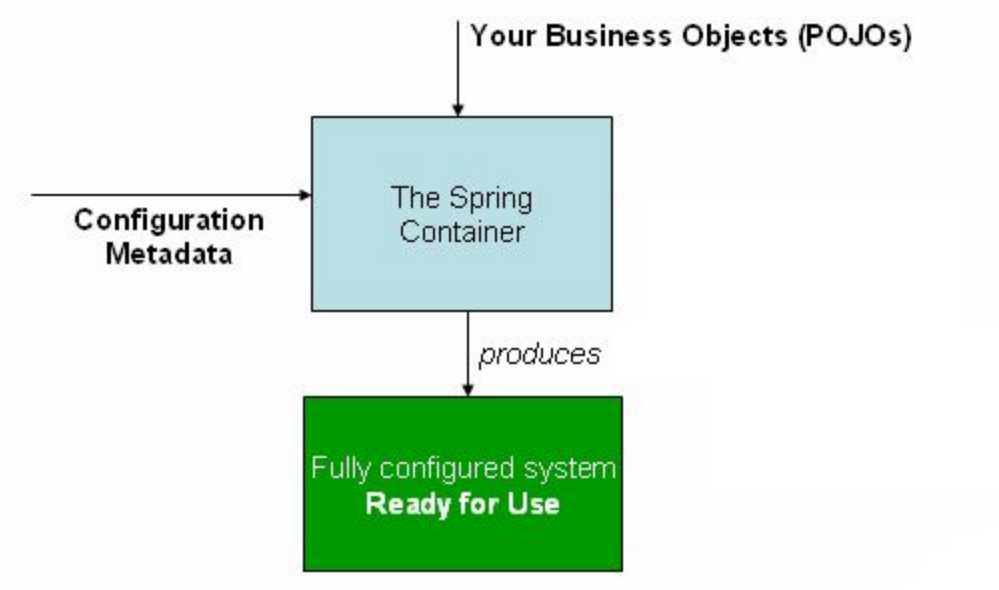

Figura 4: Modalitatea prin care bean-urile sunt configurate şi instanţiate de Spring IoC Container[20]

3.3.2. Spring Security [21]

Modulul de securitate oferit de Spring asigură servicii de securitate pentru aplicaţii dezvoltate pe platforma Java EE. Spring Security este folosit, în special, întrucât caracteristicile de securitate legate de specificaţiile pentru Servlet[12]-urile Java EE sau pentru EJB[13] (Enterprise Java Beans) prezintă o oarecare lipsă în ceea ce priveşte profunzimea necesară pentru scenariile din aplicaţiile tipice de întreprindere. Este, de asemenea, bine de menţionat faptul că aceste standarde nu sunt portabile la nivel de WAR[14] (Web Application Archive) sau EAR[15] (Enterprise Application Archive). Prin urmare, schimbarea mediului în care rulează serverul presupune un nivel mare de muncă dedicat reconfigurării securităţii aplicaţiei în noul mediu de lucru. Folosind Spring Security, aceste probleme sunt depăşite şi, de asemenea, acest modul vine cu multe alte caracteristici de securitate utile şi personalizabile.

Se ştie faptul că două cele mai mari arii în ceea ce priveşte securitatea unei aplicaţii sunt *autentificarea* şi *autorizarea* (sau *controlul accesului*), domenii care constituie ţintele de bază pentru Spring Security. *Autorizarea* se referă la procesul de a decide dacă unui utilizator îi este permis să execute o acţiune într-o aplicaţie, însă, pentru a ajunge la punctul în care o decizie de

[12] https://en.wikipedia.org/wiki/Java_servlet
[13] https://en.wikipedia.org/wiki/Enterprise_JavaBeans
[14] https://en.wikipedia.org/wiki/WAR_(file_format)
[15] https://en.wikipedia.org/wiki/EAR_(file_format)

autorizare este necesară, identitatea utilizatorului a fost deja recunoscută şi stabilită de procesul de *autentificare*.

Spring Security oferă un set de capacităţi de autorizare foarte bine puse la punct, în aşa fel încât se extinde pe trei mari arii de interes: autorizarea cererilor web, autorizarea metodelor (dacă pot fi invocate sau nu) şi autorizarea accesului la instanţele de obiecte ale utilizatorilor.

Având în vedere cele menţionate mai sus, acest modul este folosit în cadrul autentificării în aplicaţia a cărei arhitecturi urmează a fi ulterior prezentată. Deşi în momentul de faţă, această componentă nu este valorificată la capacitatea maximă în proiectul prezent descris, ea constituie un bun candidat de asistenţă pentru o dezvoltare pe viitor în sensul gestiunii securităţii modulelor acestui proiect.

3.3.3. **Spring Data JPA** [22]

Crearea de repository-uri care folosesc Java Persistence API este un proces împovărător care necesită foarte mult timp şi care presupune scriere de cod foarte încărcat. S-ar putea elimina o parte din acest cod complicat urmărind nişte paşi simpli:

- se crează o clasă abstractă, de bază, pentru repository, care furnizează operaţii CRUD[16] pentru entităţi
- se crează o clasă concretă ce extinde clasa abstractă de bază

Problema în această abordare este că încă este necesar a scrie codul care crează interogările în baza de date şi care să le invoce. Mai rău, ar trebui să repetăm acest proces de fiecare dată când se doreşte a se crea o nouă interogare în baza de date, ceea ce ar constitui o pierdere mare de timp.

Spring Data JPA vine cu o soluţie pentru această problemă întrucât obiectivul acestui modul este cel de a îmbunătăţi vizibil implementarea layer-elor de acces la date, reducând cantitatea de efort depus în acest sens. Astfel, dezvoltatorului nu-i rămâne decât să definească interfeţele de repository-uri împreună cu metodele necesare, iar Spring va oferi implementarea necesară, în mod automat.

Spring Data JPA este un framework care aduce un nou layer de abstracţie peste un anumit JPA provider definit. Astfel, folosind această librărie, layer-ul de repository-uri, în aplicaţia de

[16] https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

noastră, conţine, la rândul ei, alte trei straturi descrise în felul următor:

- *Spring Data JPA* furnizează suport pentru crearea de repository-uri JPA prin extinderea interfeţelor definite în acest modul;
- *Spring Data Commons* furnizează infrastructura distribuită de datastore-ul specific proiectelor din Spring Data;
- *JPA Provider* implementează Java Persistence API definit în standard.

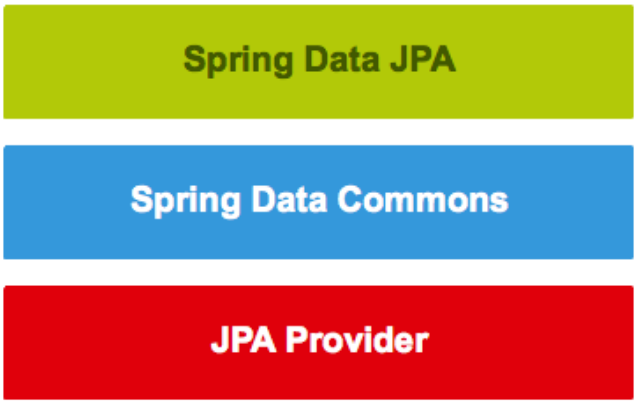

Figura 5: Structura layerelor de repository-uri dintr-un proiect ce utilizează Spring Data JPA[22]

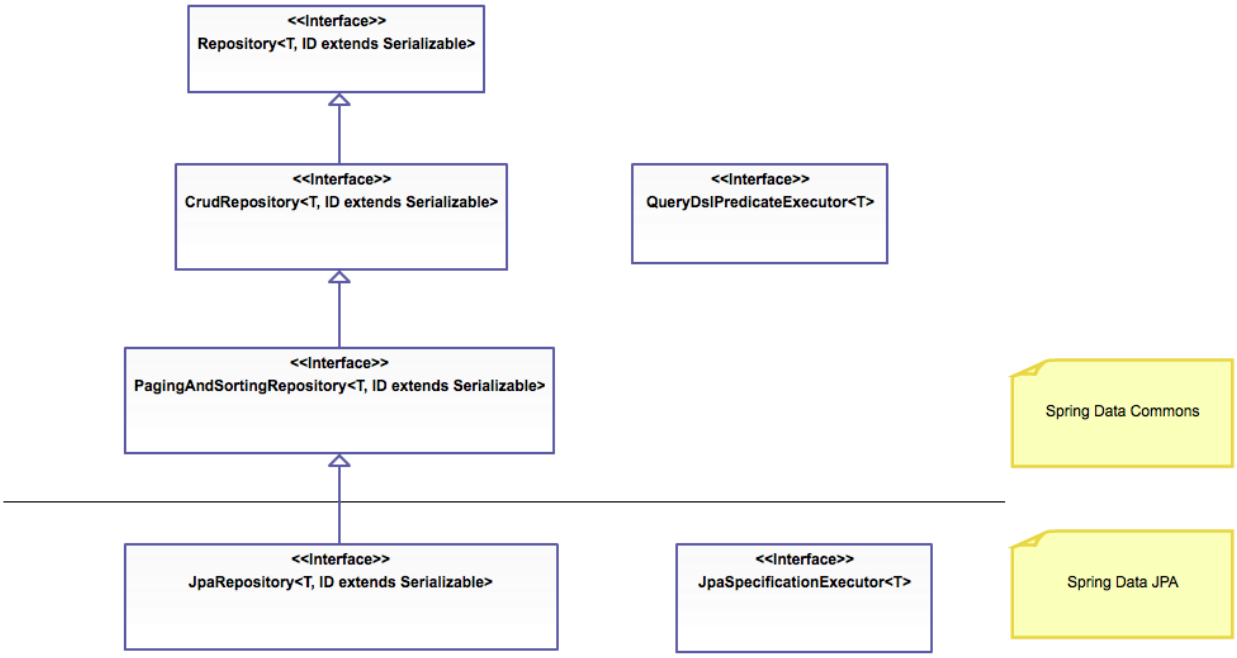

Figura 6: Ierarhia de repository-uri din Spring[17]

[17] https://www.petrikainulainen.net/wp-content/uploads/springdatajrepositories.png

La prima vedere, impresia pe care o lasă Spring Data JPA este aceea că ar putea face ca aplicaţia noastră să pară mai complicată, iar într-un fel aşa este. Chiar dacă aduce un strat adiţional peste cel de repository-uri, ne eliberează de scrierea de cod supraîncărcat, ceea ce este un compromis ce nu poate fi ignorat în aplicaţia noastră.

3.3.4. Spring RESTful API [23]

Serviciile web de tip REST constituie o modalitate de furnizare o interoperabilităţii între sisteme conectate la Internet. Aceste servicii permit sistemelor care solicită cereri să acceseze şi să manipuleze reprezentări text de resurse web, folosind un set de operaţii predefinit. Există şi alte forme de servicii web, care expun propriile mulţimi de operaţii, precum WSDL[18] sau SOAP[19]

Spring oferă, de asemenea, un modul care facilitează crearea de servicii REST[20] (Representational Web Transfer), reducând drastic cantitatea de cod scris necesar expunerii unui astfel de serviciu.

3.4. Hibernate [24]

Hibernate este o unealtă ORM[21] (Object-Relational Mapping) pentru limbajul de programare Java care furnizează un framework pentru asocierea unui model creat folosind paradigma obiectuală la o bază de date relaţională.

Proprietatea de bază pentru Hibernate este asocierea dintre clase definite în Java şi tabele în baza de date, dar şi asocierea dintre tipurile de date din Java şi tipurile de date SQL[22]. De asemenea, oferă facilităţi precum interogări sau regăsire de date, generând apeluri SQL[2] şi scutind, astfel, dezvoltatorul de manipularea manuală şi de conversia de obiecte a setului rezultat. Această asociere este implementată în urma configurării unui fişier XML[23] sau folosind adnotări Java.

Totodată, este posibilă definirea unor relaţii de tipul one-to-many sau many-to-many între clase, lucru care se răsfrânge automat şi asupra schemelor bazelor de date (independent de

[18] https://www.w3.org/TR/wsdl
[19] https://en.wikipedia.org/wiki/SOAP
[20] https://en.wikipedia.org/wiki/Representational_state_transfer
[21] https://en.wikipedia.org/wiki/Object-relational_mapping
[22] https://www.w3schools.com/SQL/deFault.asp
[23] https://www.w3schools.com/xml/

dialectul folosit).

În jargonul Hibernate, o *entitate* este un obiect de sine stătător prezent în mecanismul de persistenţă al framework-ului, care poate fi manipulat independent de alte obiecte. În contrast, o componentă este subordonată unei entităţi şi poate fi manipulată numai dacă se ia în considerare entitatea primară. De exemplu, un Album poate reprezenta o entitate, dar un obiect de tipul Piesă asociat unui Album ar reprezenta o componentă a entităţii de tipul Album, mergând pe premisa că Piesele pot fi memorate sau regăsite în baza de date numai prin intermediul obiectului de tip Album.

3.5. Spark MLlib [25]

MLlib este librăria de învăţare automată ce aparţine de Spark[24], al cărei obiectiv principal este cel de a face ca algoritmii de învăţare automată să devină scalabili şi uşor de folosit.

La nivel mai înalt, pune la dispoziţie o serie de unelte precum:

- *Algoritmi de învăţare automată*: algoritmi de învăţare comuni precum cei de clasificare, regresie, clusterizare şi de collaborative filtering;
- *Featurization*: extragere de caracteristici, transformări, reducere a dimensiunilor şi selecţie;
- *Pipelines*: salvarea şi încărcarea algoritmilor şi a modelelor;
- *Utilităţi*: algebră liniară, statistici, gestionarea datelor.

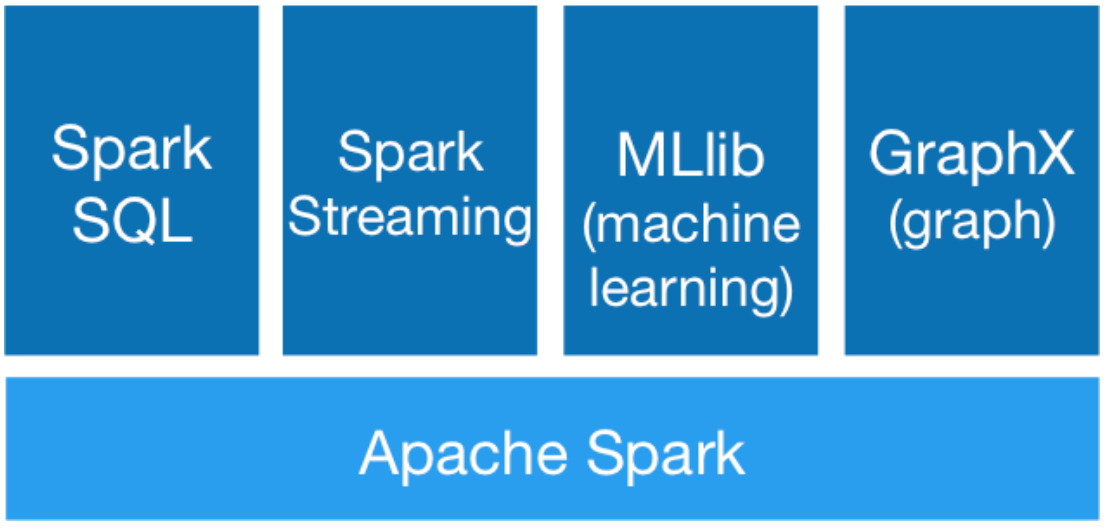

Figura 7: Spark împuterniceşte un set de librării, printre care şi MLlib[27]

[24] https://spark.apache.org/

3.6. Plant UML [26]

PlantUML este o unealtă open-source ce permite utilizatorilor să creeze diagrame UML, pornind de la un limbaj de tipul plain-text. De asemenea, foloseşte software-ul Graphviz[25] pentru a expune aceste diagrame.

Ca exemplu, plain-textul următor generează diagrama din *Figura 8.*

```
@startuml
Class11 <|.. Class12
Class13 --> Class14
Class15 ..> Class16
Class17 ..|> Class18
Class19 <--* Class20
@enduml
```

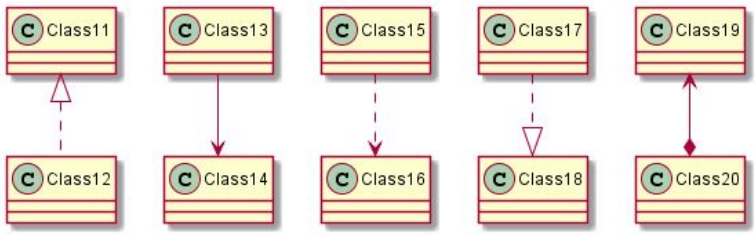

Figura 8: Diagrama de clase generată folosind PlantUML[28]

3.7. Apache Wicket [30]

Apache Wicket este un framework de aplicaţii web bazat pe componente, folosit în cadrul limbajului de programare Java şi este conceptual similar cu JavaServer Faces[26] şi Tapestry[27].

Aplicaţiile Wicket sunt, de fapt, arbori de componente, care se folosesc de aşa-numiţii ”ascultători delegaţi” pentru a răspunde la cereri HTTP pentru link-uri şi formulare, la fel cum componentele din Swing[28] răspund la evenimentele trimise de mouse sau prin apăsarea de taste.

Wicket foloseşte XHTML[29] pentru şablonare (ce forţează o separare clară între prezentare şi logica de business). Fiecare componentă este legată de un element din XHTML şi devine responsabil de randarea acelui element în output-ul final. Folosind tag-uri speciale, un grup de componente pot fi abstractizate într-o singură componentă, denumită panou, care, apoi, poate fi refolosită ca întreg în acea pagină, în alte pagini, sau chiar în alte panouri.

Fiecare componentă este susţinută, în spate, de propriul model, care constituie starea componentei în sine. Framework-ul nu are cunoştinţe despre felul în care componentele interacţionează cu propriile modele, care sunt tratate ca obiecte opace şi automat serializate şi persistate între cereri. Modelele mai complexe, totuşi, ar putea fi construite ca fiind detaşabile şi

[25] http://www.graphviz.org/
[26] https://en.wikipedia.org/wiki/JavaServer_Faces
[27] http://tapestry.apache.org/
[28] https://en.wikipedia.org/wiki/Swing_(Java)
[29] https://ro.wikipedia.org/wiki/XHTML

furnizează un mecanism pentru a-şi aranja propria memorare şi restaurare la începutul şi la finalul fiecărui ciclu de cerere. Întrucât Wicket nu impune niciun layer special de persistenţă a obiectelor sau de ORM, aplicaţia web de faţă poate folosi acest framework în combinaţie cu obiecte Hibernate, Spring Beans[30] sau obiecte POJO[31] ca modele.

În Wicket, toată starea server-ului este automat administrată. Fiecare componentă server-side din pagină menţine o ierarhie imbricată de componente, cu stare, în care fiecare model de componentă este, în cele din urmă, un POJO.

3.8. **Bootstrap** [29]

Bootstrap este un framework web de front-end, open-source, folosit în proiectarea atât a site-urilor web, cât şi a aplicaţiilor web, care conţine şabloane HTML[32] şi CSS[33] destinate tipografiei, formularelor, butoanelor, componentelor de navigare, precum şi, opţional, unor extensii JavaScript[34]. Spre deosebire de multe alte framework-uri, interesul este orientat doar spre dezvoltarea pe partea de front-end[35].

➢ *Foi de stil*

Bootstrap furnizează un set de foi de stiluri care oferă definiţii de stiluri de bază pentru toate componentele-cheie din HTML. Acestea asigură un aspect modern, uniform pentru formatarea de texte, tabele şi elemente din formulare.

➢ *Componente reutilizabile*

Pe lângă elementele HTML obişnuite, Bootstrap conţine alte elemente de interfaţă utilizate frecvent. Componentele sunt implementate drept clase CSS, care trebuiesc aplicate pe anumite elemente HTML dintr-o pagină.

➢ *Componente JavaScript*

Bootstrap vine la pachet cu câteva componente JavaScript sub forma unor plugin-uri

[30] https://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html
[31] https://en.wikipedia.org/wiki/Plain_old_Java_object
[32] https://ro.wikipedia.org/wiki/HyperText_Markup_Language
[33] https://www.w3.org/Style/CSS/Overview.en.html
[34] https://ro.wikipedia.org/wiki/JavaScript
[35] https://en.wikipedia.org/wiki/Front-end_web_development

jQuery[36]. Acestea oferă elemente de interfață adiționale, precum căsuțe de dialog, tooltip-uri sau carousel-uri[37]. Ele extind funcționalitatea câtorva elemente de interfață existente, cum ar fi funcția de autocompletare pentru câmpurile de input.

4. Arhitectura aplicației

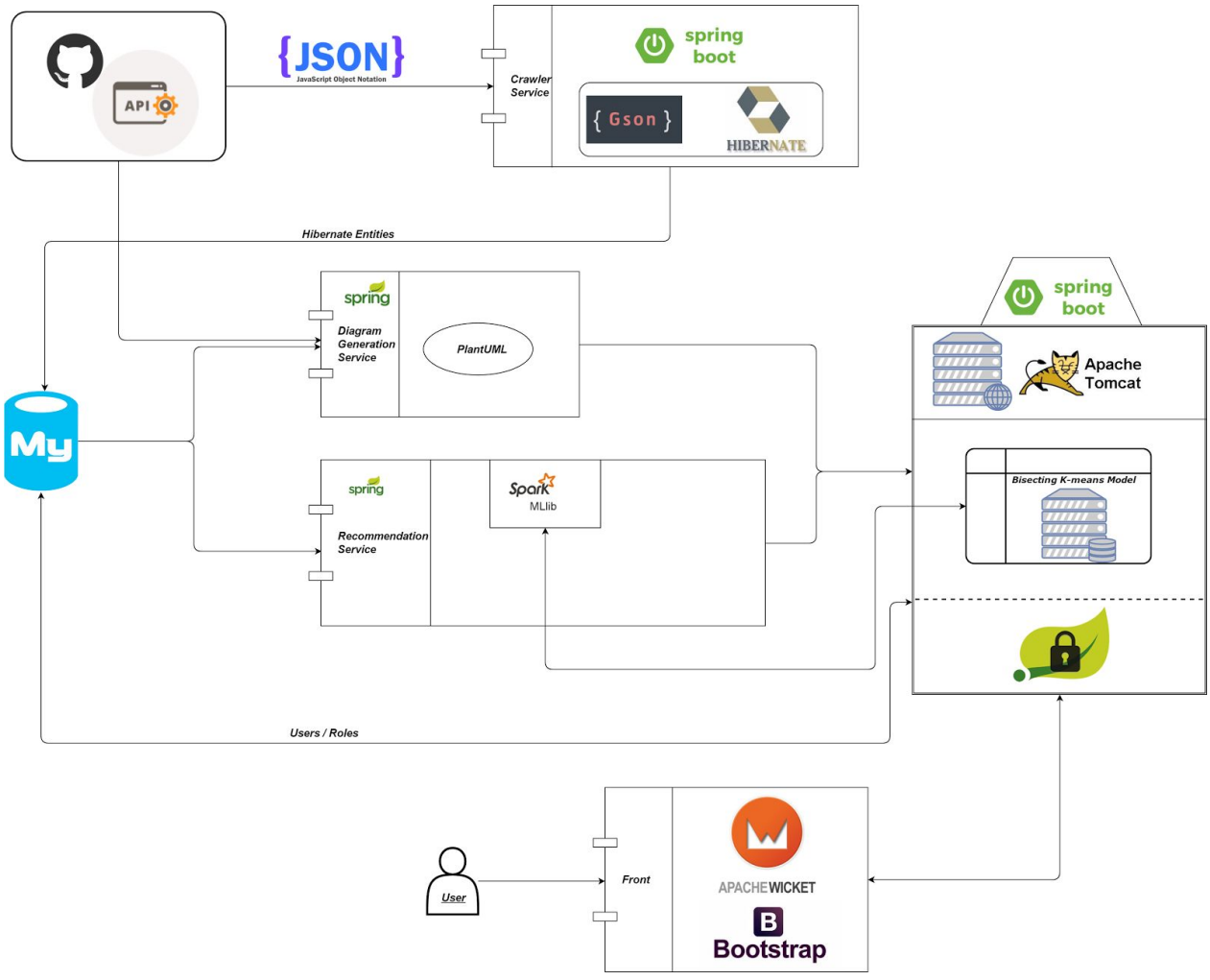

Figura 9: Arhitectura aplicației

În *Figura 9* este prezentată arhitectura aplicației, împreună cu modulele și serviciile constitutive, dar și cu tehnologiile care au stat la baza dezvoltării acestora. În continuare, va fi descris mai detaliat arhitectura fiecărui serviciu/modul și felul în care acesta funcționează pentru

[36] https://ro.wikipedia.org/wiki/JQuery
[37] https://www.w3schools.com/bootstrap/bootstrap_carousel.asp

a pune la dispoziţie fluxul de lucru al aplicaţiei de faţă.

4.1. Serviciul de crawling

GitHub expune un API care facilitează căutarea de repository-uri publice, ba mai mult, suplimentează informaţii în legătură cu acel repository, cum ar fi: limbajele de programare utilizate în dezvoltarea proiectului, descrierea acestuia, posibilitatea de descărcare în format ZIP a întregului proiect, accesul la fişierul ”README.MD”, link-ul către repertoriu, ş.a.m.d.

Un simplu exemplu de informaţii în format JSON despre un repository este următorul:

```
{
      "id": 21289110,
      "name": "awesome-python",
      "full_name": "vinta/awesome-python",
      "owner": {
        "login": "vinta",
        "id": 652070
        "url": "https://api.github.com/users/vinta",
        "repos_url": "https://api.github.com/users/vinta/repos",
        "events_url": "https://api.github.com/users/vinta/events{/privacy}",
        "received_events_url":
"https://api.github.com/users/vinta/received_events",
        "type": "User",
        "site_admin": false
      },
      "url": "https://api.github.com/repos/vinta/awesome-python",
      "description": "A curated list of awesome Python frameworks, libraries,
software and resources",
"forks_url": "https://api.github.com/repos/vinta/awesome-python/forks",
      "languages_url":
"https://api.github.com/repos/vinta/awesome-python/languages",
      "contents_url":
"https://api.github.com/repos/vinta/awesome-python/contents/{+path}",
      "downloads_url":
"https://api.github.com/repos/vinta/awesome-python/downloads",
      "issues_url":
"https://api.github.com/repos/vinta/awesome-python/issues{/number}",
      "pulls_url":
"https://api.github.com/repos/vinta/awesome-python/pulls{/number}",
      "language": "Python"
      ……..
      }
```

Valorile îngroşate constituie informaţiile utile oferite de API pentru prezenta aplicaţie, accesate şi asociate cu o entitate Hibernate pentru a putea fi salvate în baza de date.

Această facilitate oferită de GitHub[38] vine şi cu o limitare a numărului de cereri neautentificate (60 cereri/oră), motiv pentru care am fost nevoit să creez 10 token-uri[39] de acces care să restrângă numărul de cereri la 5000 cereri/oră per token. Astfel, pentru fiecare request[40] trimis la interval de 2 secunde (pentru a nu supraîncărca API-ul), token-ul este schimbat prin intermediul unui Observer[41], ceea ce a mărit numărul maxim de cereri care se pot trimite la 50 000/oră.

În urma acestei etape, s-au obţinut 1892 de proiecte dezvoltate în diferite limbaje de programare, număr suficient de mare pentru a testa utilitatea şi gradul de corectitudine al serviciului de recomandare. De menţionat, totuşi, este faptul că, întrucât instanţele de proiecte nu au putut fi controlate de API în ceea ce priveşte descrierea lor, unele dintre acestea nu sunt suficient de detaliate pentru a reflecta conţinutul proiectului.

4.2. Serviciul de recomandări

Procesul de recomandare este unul complex întrucât necesită două etape diferite, desfăşurate în intervale distincte de timp.

Astfel, sub-procesul de *clusterizare* are loc la prima rulare a aplicaţiei, accesând un API expus în acest sens sub forma unui URL care activează un serviciu ce creează (în urma aplicării algoritmului Bisecting K-means[42] pe setul de date din baza de date) un model salvat în storage-ul serverului, pe baza căruia are loc distribuţia repository-urilor către clustere. Pentru fiecare instanţă de repository din baza de date, în urma clusterizării, se actualizează şi numărul cluster-ului la care aceasta aparţine. Modelul poate fi actualizat printr-un job pornit la un interval fix de timp, pentru a-l reconstrui folosind noile repertorii create/aduse în baza de date a aplicaţiei.

Sub-procesul de *recomandare* nu poate avea loc decât dacă există deja un model construit în etapa anterioară. Astfel, în momentul în care o nouă descriere de proiect este furnizată, textul este transformat într-un vector TF-IDF normalizat în L_2, acesta din urmă fiind trimis către modelul care oferă drept output identificatorul cluster-ului prezis de algoritm, ce constituie grupul de instanţe la care ar trebui ca repository-ul să aparţină în urma utilizării măsurii de distanţă faţă

[38] https://en.wikipedia.org/wiki/GitHub
[39] https://en.wikipedia.org/wiki/Access_token
[40] https://v4.datafeedr.com/documentation/373
[41] https://www.tutorialspoint.com/design_pattern/observer_pattern.htm
[42] https://stackoverflow.com/questions/6871489/bisecting-k-means-clustering-algorithm-explanation

de celelalte proiecte, reprezentată de *similaritatea cosinus*. Întrucât identificatorul cluster-ului nu este suficient pentru a oferi recomandarea, pentru fiecare instanţă de repertoriu aparţinând cluster-ului respectiv, se măsoară distanţa cosinus dintre acel repository şi descrierea dată de proiectul utilizatorului, apoi acest scor este folosit pentru a sorta, în ordinea similarităţii, repertoriile indicate. Primul repository va constitui, astfel, cea mai bună recomandare de proiect existent în aplicaţia de faţă.

În figura următoare, este descris flow-ul serviciului de recomandare.

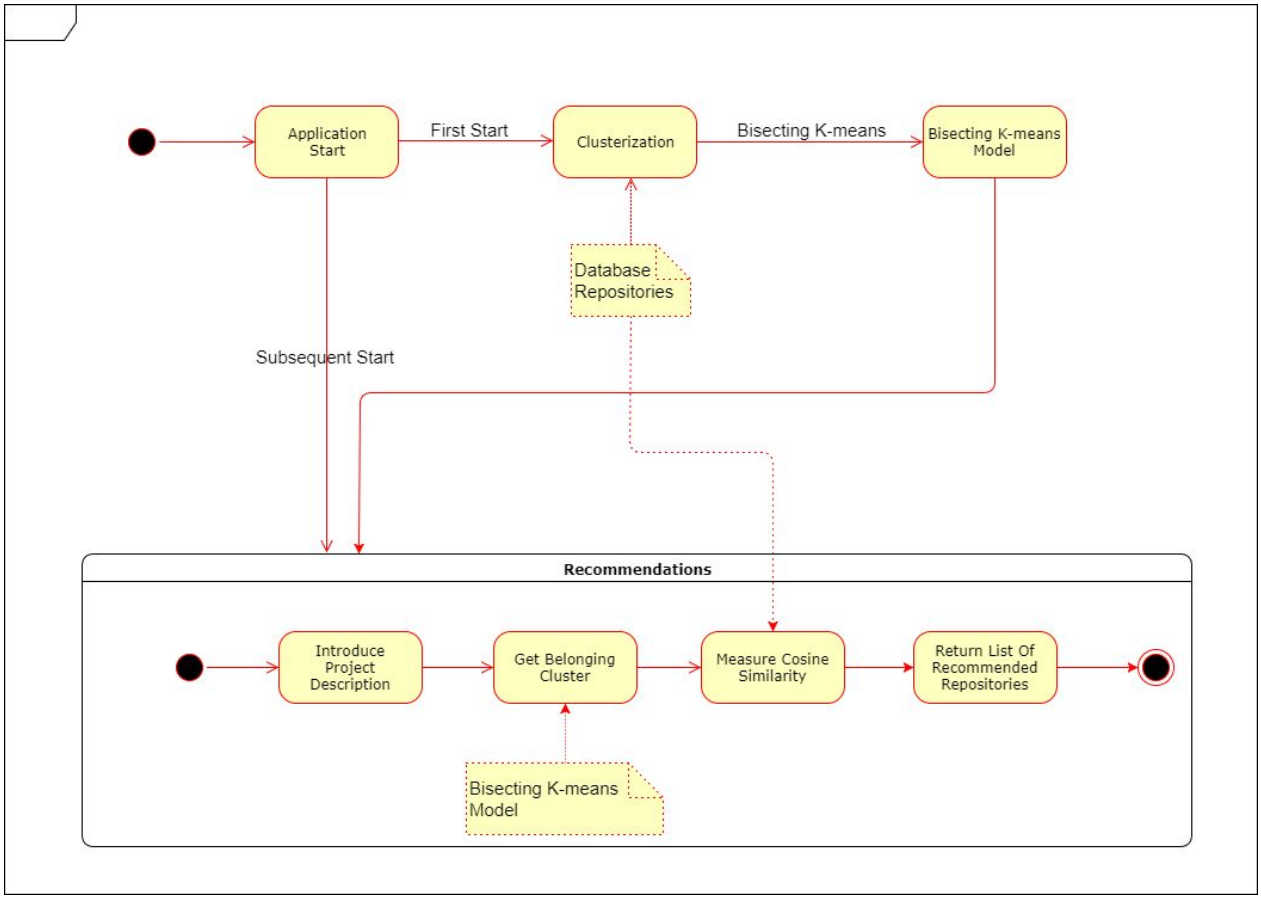

Figura 10: Diagrama de stare a serviciul de recomandare

4.2.1. Clusterizare

Pentru acest proces, se execută, în prealabil, o transformare a descrierii text (transformare prezentată mai detaliat în capitolul 2.2), rezultând vectorul TF-IDF normalizat în L_2. Apoi, vectorul reprezentând repository-ul în cauză este folosit ca input pentru algoritmul Bisecting K-means care construieşte modelul necesar predicţiei cluster-ului la care aparţin noile proiecte furnizate de utilizatori. Următoarea diagramă de stare descrie exact paşii executaţi pentru

clusterizare.

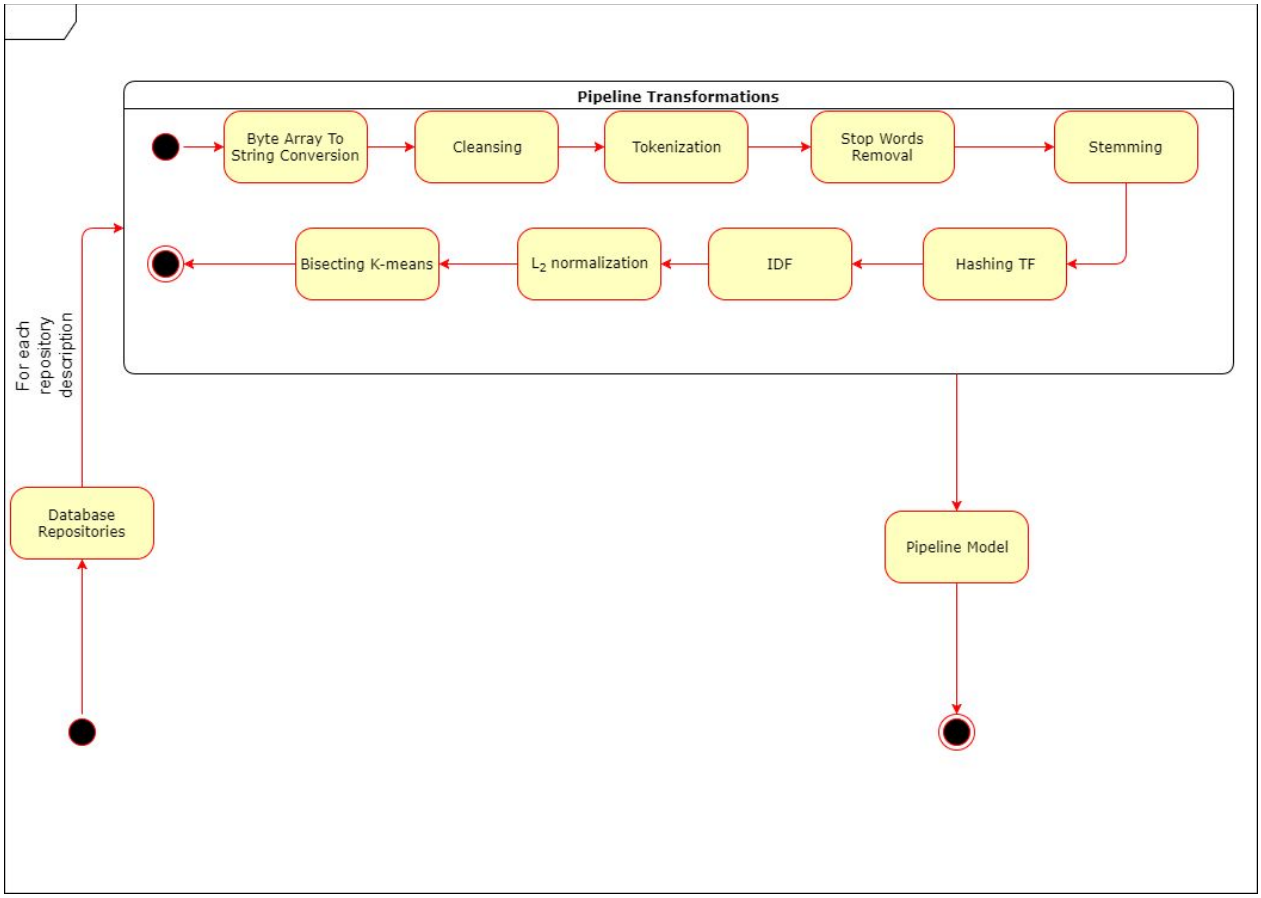

Figura 11: Diagrama de stare pentru sub-procesul de clusterizare

În diagrama de mai sus, s-a adăugat o transformare nemenţionată până acum (*Byte Array To String Conversion*), schimbare realizată în cadrul aceluiaşi pipeline cu celelalte modificări şi folosită pentru a transforma byte array-ul corespunzător descrierii repository-ului salvat în baza de date într-un string. Salvarea descrierilor s-a făcut în acest fel întrucât există posibilitatea existenţei unor caractere nerecunoscute de vendor-ul bazei de date, lucru care ar conduce la apariţia unei erori în momentul în care se încearcă inserarea unor intrări de acest tip.

4.2.2. Recomandare

Pentru construirea listei de repository-uri recomandate în urma furnizării de către utilizator a unui proiect, se utilizează modelul construit la pasul anterior pentru transformarea instanţei în cauză în vectorul normalizat, dar, în acelaşi timp, proiectul este asignat şi către

cluster-ul care conţine proiectele asemănătoare ca funcţionalitate. După ce este luată decizia alocării către un cluster, pentru fiecare instanţă ce aparţine de acel grup, se măsoară *similaritatea cosinus* cu proiectul furnizat de către utilizator. Acest scor va constitui criteriul de sortare a recomandărilor în lista returnată de serviciu, listă cu o dimensiune maximă de 20 de repertorii.

În următoarea figură, este descrisă modalitatea prin care are loc selecţia repository-urilor din baza de date aferentă procesului de furnizare a recomandărilor.

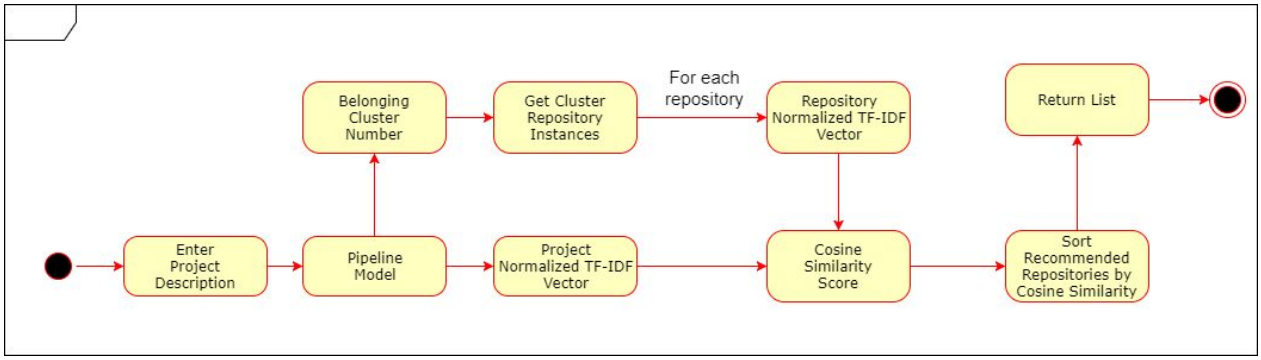

Figura 12: Diagrama de stare pentru sub-procesul de recomandare de repository-uri

4.3. Serviciul de generare de diagrame

Serviciul în cauză poate fi folosit doar în momentul în care proiectul recomandat este scris în limbajul de programare Java. Astfel, se aplică un proces de *reverse engineering*[43] ce constă în construirea şi folosirea de expresii regulate pentru a potrivi şi prelua denumiri de clase, semnături de metode, declaraţii de variabile şi relaţiile dintre clase.

PlantUML[44] împreună cu extensia *PlantUML Dependency*[45] oferă posibilitatea de a parsa fişiere sursă dintr-un proiect Java cu scopul de a genera diagrama de clase împreună cu relaţiile dintre ele. Întrucât extensia nu are o expresie regulată care să poată extrage atât declarările de variabile, cât şi signatura metodelor, am creat încă două expresii regulate care vin cu o soluţie pentru această problemă.

Extensia *PlantUML Dependency* funcţionează în felul următor: după ce fişierele sursă sunt analizate, se construieşte un arbore care defineşte relaţiile dintre clase, rezultând un fişier text care respectă regulile din cadrul framework-ului de bază, *PlantUML*. În momentul în care acest fişier este generat, sunt reanalizate fişierele pentru a insera denumirile de metode,

[43] https://ro.wikipedia.org/wiki/Inginerie_invers%C4%83
[44] http://plantuml.com/
[45] http://plantuml-depend.sourceforge.net/

declarările de variabile şi modificatorii de acces, apoi este generată diagrama UML finală în format SVG[46], trimisă apoi către utilizatorul care a solicita vizionarea acesteia.

4.4. Layer-ul de prezentare

Pentru prezentare, s-a expus atât o interfaţă web, cât şi un API.

Interfaţa web este construită folosind un şablon oferit de Themeum[47] (ce are la bază, la rândul său, framework-ul de front-end Bootstrap[48]), adaptată cu Wicket[49], iar administrarea sesiunilor este realizată folosind framework-ul Spring Security[50].

Pentru expunerea API-ului, am folosit modulul *Spring RESTful Web Service*[51], iar rutele expuse sunt următoarele:

4.5. Concluzii

[46] https://www.w3schools.com/html/html5_svg.asp
[47] http://demo.themeum.com/html/oxygen/
[48] http://getbootstrap.com/
[49] https://wicket.apache.org/
[50] https://projects.spring.io/spring-security/
[51] https://spring.io/guides/gs/rest-service/

Link-uri utile

[1] *http://jonathanzong.com/blog/2013/02/02/k-means-clustering-with-tfidf-weights*

[2] *Curs de Învățare Automată - Liviu Ciortuz: ML.ex-book.SLIDES.Cluster.pdf*

[3] *https://www.ijarcsse.com/docs/papers/Volume_5/2_February2015/V5I2-0229.pdf*

[4] *http://snowball.tartarus.org/*

[5] *https://en.wikipedia.org/wiki/Tf%E2%80%93idf*

[6] *https://en.wikipedia.org/wiki/Norm_(mathematics)*

[7] *https://en.wikipedia.org/wiki/Cosine_similarity*

[8]*https://stats.stackexchange.com/questions/146221/is-cosine-similarity-identical-to-l2-normalized-euclidean-distance*

[9] *https://www.tutorialspoint.com/uml/index.htm*

[10] *http://www.uml-diagrams.org/index-examples.html*

[11] *http://control.aut.utcluj.ro/hmihai/doku.php?id=uml:uml_clase*

[12] *http://www.oracle.com/technetwork/java/javaee/overview/index.html*

[13] *https://stackoverflow.com/questions/106820/what-is-java-ee*

[14] *http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html*

[15] *https://en.wikipedia.org/wiki/Apache_Maven*

[16] *https://ro.wikipedia.org/wiki/Funda%C8%9Bia_Apache*

[17] *http://ant.apache.org/*

[18] *https://repo.maven.apache.org/maven2/*

[19] *https://projects.spring.io/spring-framework/*

[20] *https://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html*

[21] *https://projects.spring.io/spring-security/*

[22]*https://www.petrikainulainen.net/programming/spring-framework/spring-data-jpa-tutorial-introduction/*

[23] *https://en.wikipedia.org/wiki/Representational_state_transfer*

[24] *http://hibernate.org/orm/*

[25] *https://spark.apache.org/docs/latest/ml-guide.html*

[26] *http://plantuml.com/*

[27] *https://spark.apache.org/*

[28] *http://plantuml.com/class-diagram*

[29] *https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)*

[30] "*Instant Apache Wicket 6" - João Sávio Ceregatti Longo, Felipe Fedel Pinto*

E in regula cum scrii :)

Observatii:

1) Partea de inceput cu definitii si prezentari de algoritm trebuie sa fie plina de trimiteri catre note bibliografice. Momentan ai cateva, dar adauga mai multe (chiar daca te repeti)
2) Astept o versiune in care sa ai cat mai multe din elementele din cuprins completate