

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI  
**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**RECOMANDĂRI DE REPERTORII GITHUB**

propusă de  
*Gabor Silviu*

**Sesiunea:** *iulie, 2017*

**Coordonator științific**  
**Conf. Dr. Adrian Iftene**

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**  
**FACULTATEA DE INFORMATICĂ**

## **RECOMANDĂRI DE REPERTORII GITHUB**

*Gabor Silviu*

**Sesiunea:** *iulie, 2017*

Coordonator științific

**Conf. Dr. Adrian Iftene**

# DECLARAȚIE PRIVIND ORIGINALITATEA ȘI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licență cu titlul “Recomandări de repertorii GitHub” este scrisă de mine și 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 și în traducere proprie din altă limbă, sunt scrise între ghilimele și 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 și dețin referințe precise;
- rezumarea ideilor altor autori precizează referința precisă la textul original.

Iași, 03.07.2017

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 și 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 Iași să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, 03.07.2017

Absolvent Gabor Silviu

---

## **Acord privind proprietatea dreptului de autor**

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

Încheierea acestui acord este necesară din următoarele motive:

În viitor, doresc continuarea dezvoltării acestei aplicații, precum și extinderea pe alte platforme.

Iași,

Decan *Adrian Iftene*

---

Absolvent *Gabor Silviu*

---

# Cuprins

<b>1.Introducere.....</b>	<b>9</b>
1.1.Motivație.....	9
1.2.GitHub.....	10
1.3.Concluzii.....	10
<b>2.Definiții și exemple.....</b>	<b>11</b>
2.1.Clusterizare de documente folosind învățarea automată.....	11
2.2.Preprocesarea instanțelor.....	14
2.3.Diagrame UML.....	18
2.4.Concluzii.....	20
<b>3.Tehnologii folosite.....</b>	<b>21</b>
3.1.Java EE.....	21
3.2.Maven.....	22
3.3.Spring.....	23
3.3.1.Spring IoC Container.....	24
3.3.2.Spring Security.....	25
3.3.3.Spring Data JPA.....	26
3.3.4.Spring RESTful API.....	28
3.4.Hibernate.....	28
3.5.Spark MLlib.....	29
3.6.Plant UML.....	30
3.7.Apache Wicket.....	30
3.8.Bootstrap.....	31
<b>4.Arhitectura aplicației.....</b>	<b>32</b>
4.1.Serviciul de crawling.....	33
4.2.Serviciul de recomandări.....	34
4.2.1.Clusterizare.....	35
4.2.2.Recomandare.....	37
4.3.Serviciul de generare de diagrame.....	37

4.4.Layer-ul de prezentare.....	38
4.5.Concluzii.....	39
<b>5.Studii de caz.....</b>	<b>39</b>
5.1.Caz de utilizare favorabil.....	39
5.2.Caz de utilizare nefavorabil.....	41
5.3.Limitări ale aplicației și analiza erorilor.....	42
<b>6.Concluzii finale.....</b>	<b>42</b>
6.1.Contribuții personale și rezultate.....	43
6.2.Direcții de viitor.....	44
<b>7.Link-uri utile.....</b>	<b>45</b>

# 1. Introducere

## 1.1. Motivație

Prezenta lucrare de licență dorește a oferi o soluție pentru dezvoltarea mai ușoară a proiectelor de diferite tipuri, pornind de la o idee sau de la un proiect deja conceput și distribuit pe diferite platforme ce găzduiesc aplicații dezvoltate de comunitatea GitHub<sup>1</sup>. Pe scurt, ideea acestei lucrări gravitează în jurul conceptului de recomandare pe baza unor descrieri de tip text, ce conțin un scurt rezumat în legătura cu funcționalitatea ce se dorește a fi exploatată, precum și limbajele de programare implicate sau tehnologiile ce se urmăresc a fi utilizate.

Decizia alegerii acestei teme a fost luată în urma numeroaselor discuții cu coordonatorul de licență, luând în considerare necesitatea firmelor de IT care au nevoie de un punct de plecare în urma discuțiilor cu clienții care își manifestă dorința (nu neapărat clară) de implementare a unor funcționalități ale unor aplicații 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 înțelegerea modului de funcționare a aplicației, în clarificarea felului în care au fost îmbinate tehnologiile utilizate pentru a ajunge la produsul final, dar și în lămurirea metodelor prin care aplicația curentă ar putea fi îmbunătățită.

Primul capitol conține detalii despre noțiunile folosite în cadrul descrierii funcționalităților aplicației, la nivel de bază (similaritate cosinus<sup>2</sup>, norma Euclidiană<sup>3</sup>, document clustering<sup>4</sup>, stemming<sup>5</sup>). De asemenea, vor fi menționați și descriși, la nivel conceptual, algoritmi utilizați pentru realizarea recomandărilor (Bisecting K-means<sup>6</sup>, K-means<sup>7</sup>, TF-IDF<sup>8</sup>), precum și librăriile care oferă implementarea lor.

În capitolul doi, sunt descrise, în mod individual, fiecare framework utilizat pentru

---

<sup>1</sup> <https://github.com/>

<sup>2</sup> [https://en.wikipedia.org/wiki/Cosine\\_similarity](https://en.wikipedia.org/wiki/Cosine_similarity)

<sup>3</sup> [https://en.wikipedia.org/wiki/Norm\\_\(mathematics\)](https://en.wikipedia.org/wiki/Norm_(mathematics))

<sup>4</sup> [https://en.wikipedia.org/wiki/Document\\_clustering](https://en.wikipedia.org/wiki/Document_clustering)

<sup>5</sup> <https://en.wikipedia.org/wiki/Stemming>

<sup>6</sup> <http://minethedata.blogspot.ro/2012/08/bisecting-k-means.html>

<sup>7</sup> [https://en.wikipedia.org/wiki/K-means\\_clustering](https://en.wikipedia.org/wiki/K-means_clustering)

<sup>8</sup> <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>



realizarea părții practice a prezentei lucrări de licență: Spring Framework<sup>9</sup>, Apache Wicket<sup>10</sup>, Spark MLlib<sup>11</sup>, PlantUML<sup>12</sup>, Hibernate<sup>13</sup>, Maven<sup>14</sup>.

Penultimul capitol constituie o imagine de ansamblu asupra arhitecturii aplicației și oferă detalii despre fiecare modul în parte: librării folosite, algoritmi care stau la bază, relația cu alte module și modul lor de comunicare.

Ultimul capitol conține informații în legătură cu problemele întâmpinate, posibilitățile de îmbunătățire a recomandărilor (perspective asupra altor algoritmi ce ar putea fi folosiți), precum și extensiile ce ar putea constitui o direcție pe viitor pentru îmbunătățirea interactivității cu utilizatorii aplicației.

## **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 Bisecting K-means din cadrul librăriei Apache Spark MLlib, am căzut de acord cu profesorul de licență să folosesc drept instanțe descrierile unor repertorii, luate aleator, folosind un crawler construit de mine care apelează API-ul expus de GitHub. De menționat este faptul că, deoarece repertoriile au fost descărcate aleator, nu s-au putut controla instanțele de proiecte împreună cu descrierile lor, motiv pentru care sistemul de recomandare nu poate funcționa în mod optim (există descrieri care nu au relevanță în ceea ce privește conținutul repertoriului sau există proiecte ale căror descrieri sunt prezentate mult prea general).

## **1.3. Concluzii**

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

---

<sup>9</sup> <https://projects.spring.io/spring-framework/>

<sup>10</sup> <https://wicket.apache.org/>

<sup>11</sup> <https://spark.apache.org/docs/latest/mllib-guide.html>

<sup>12</sup> <http://plantuml.com/>

<sup>13</sup> <http://hibernate.org/>

<sup>14</sup> <https://maven.apache.org/>

## 2. Definiții și exemple

### 2.1. Clusterizare de documente folosind învățarea automată

Algoritmii de învățare nesupervizată din cadrul domeniului de Învățare automată impun o structură peste mulțimi de date neetichetate.

Clusterizarea este cea mai comună formă de învățare nesupervizată și reprezintă diferența majoră între clusterizare și clasificare. În cadrul clusterizării, ceea ce determină apartenența la un anumit cluster este distribuția și natura datelor, spre deosebire de clasificare, unde clasificatorul învață asocierea dintre obiecte și/sau clase provenite din așa-numitele "date de antrenament" (o mulțime de date corect etichetate la mână), iar apoi replică comportamentul învățat asupra datelor neetichetate. Câteva dintre aplicațiile frecvent întâlnite în clusterizare sunt: găsirea documentelor similare, organizarea de colecții mari de documente, detectarea de conținut duplicat, sisteme de recomandare și optimizare de căutări.

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

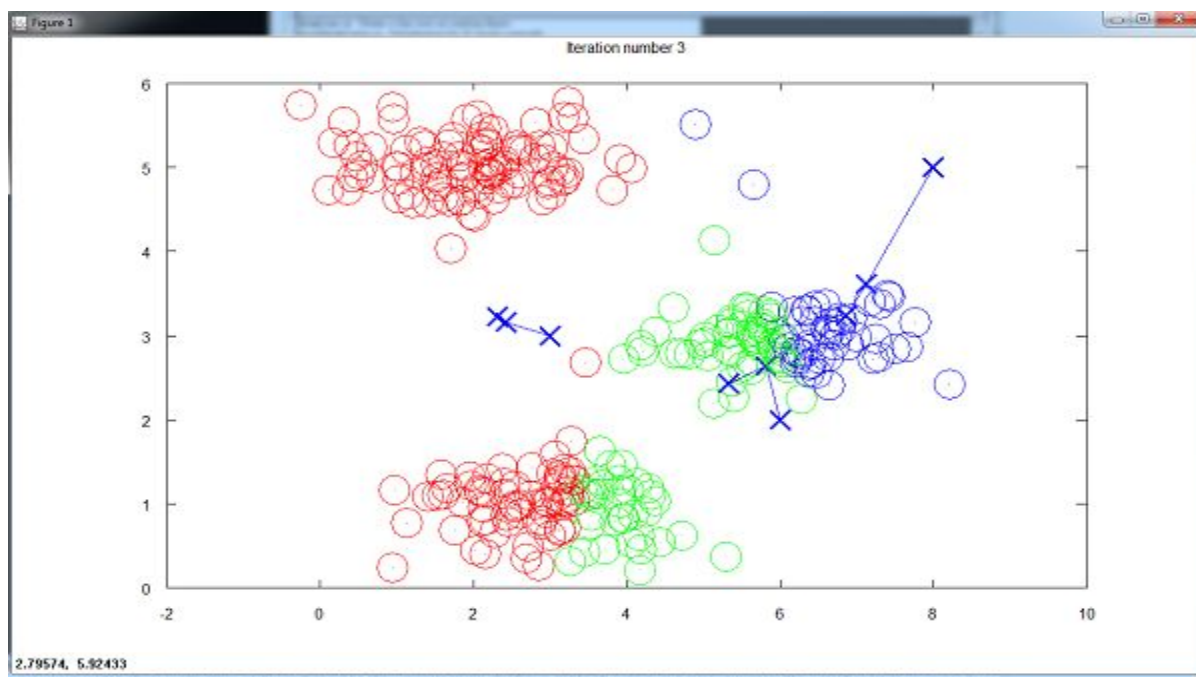


Figura 1: Schiță în care datele sunt reprezentate ca puncte, iar figurile în formă de "x" constituie centroizii a căror poziție este stabilită la iterația numărul 3 [1]

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

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

*Input:*  $x_1, x_2, \dots, x_n \in \mathbb{R}^d$ , cu  $n \geq K$   
*Output:* o anumită K-partiție pentru  $\{x_1, x_2, \dots, x_n\}$   
*Procedură:*

[Inițializare/Iterația 0:]  $t \leftarrow 0$ ;  
 se fixează în mod arbitrar  $\mu_1^0, \dots, \mu_K^0$ , centroizii inițiali ai clusterelor și, apoi, se asignează fiecare instanță  $x_i$  la centroidul cel mai apropiat, formând astfel clusterelor  $C_1^0, \dots, C_K^0$ .

[Recurzivitate:] Se execută iterația  $++t$ :  
*Pasul 1:* se calculează noile poziții ale centroizilor:

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

*Pasul 2:* se reasignează fiecare  $x_i$  la [clusterul cu] centroidul cel mai apropiat, adică se stabilește noua componență a clusterelor la iterația  $t$ :  $C_1^t, \dots, C_K^t$ ; [Terminare:] până când o anumită condiție este îndeplinită (de exemplu: până când pozițiile centroizilor – sau: componența clusterelor – nu se mai modifică de la o iterație 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 spațiu. Totodată, obiectivul unei scheme bune de clusterizare a documentelor este cel de a minimiza distanța intra-cluster între documente, în timp ce se maximizează distanța inter-cluster.

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 folosește de algoritmul K-means. Acest algoritm își începe execuția prin a pune întreaga mulțime de documente într-un singur cluster, apoi clusterul original este partiționat în alte două cluster folosind K-means (cu  $K = 2$ ). Clusterul care are cea mai ridicată similaritate intra-cluster este salvat, apoi celălalt cluster este recursiv împărțit în alte două cluster folosind același algoritm (K-means, cu  $K = 2$ ). Această operație se realizează până când este obținut numărul dorit de cluster.

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

```
Input: K - numărul de cluster, D - primele N documente obținute prin  
similaritatea vectorilor din spațiu  
Output: cele K cluster  
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  
            folosește K-means pentru a împărți clusterul C în două  
            sub-cluster,  $C_1$  și  $C_2$   
            if ( similaritatea_intra_cluster( $C_1$ ) >  
                similaritatea_intra_cluster( $C_2$ ) )  
                salvează clusterul  $C_1$   
                C =  $C_2$   
            else  
                salvează clusterul  $C_2$   
                C =  $C_1$   
            end_if
```

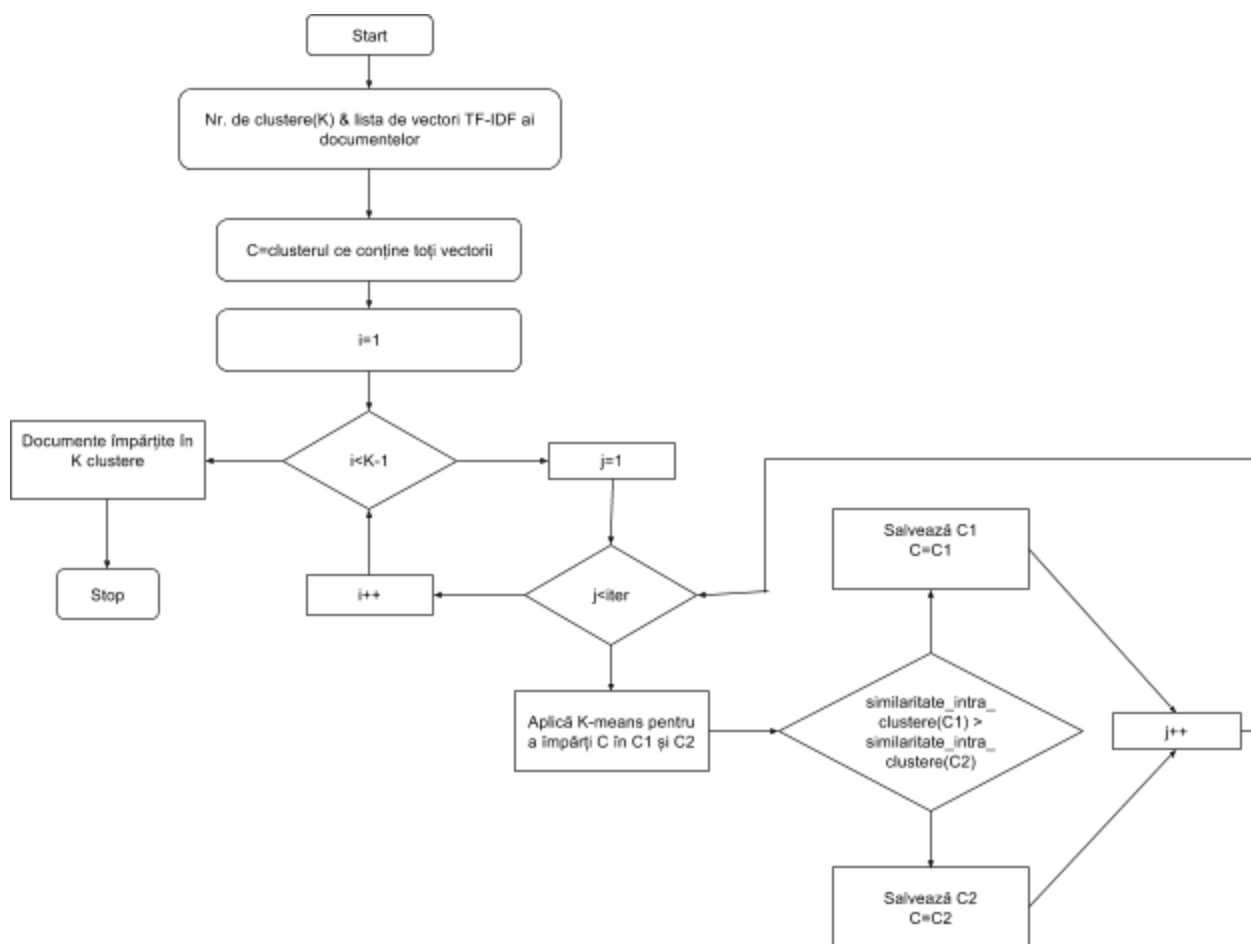


Figura 2: Schiță a modului de funcționare pentru Bisecting K-means

### ➤ Complexitatea timp

Bisecting K-Means folosește 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  $I$  reprezintă numărul de iterații până la convergență. În concluzie, Bisecting K-Means rulează în timp liniar cu dimensiunea documentelor.

## 2.2. Preprocesarea instanțelor

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

➤ Cleanser

Un Cleanser este o clasă ce extinde clasa Transformer prezentă în librăria Spark MLlib care are rolul de a șterge semnele de punctuație ("white-spaces") dintr-o descriere a unui proiect, proces necesar pentru acțiunea ulterioară de izolare corectă a cuvintelor. Această acțiune este realizată prin intermediul unei expresii regulate, înlocuind fiecare potrivire cu un șir de caractere vid.

➤ Tokenizer

Tokenizer este o clasă ce aparține aceleiași librării mai sus menționate, cu o implementare oferită de Apache pentru a "sparge în cuvinte" un șir de caractere, rezultând o listă de șiruri ce constituie cuvintele care alcătuiesc seria de caractere inițială. Astfel are loc izolarea fiecărui cuvânt în parte, rezultat ce va constitui un input pentru următoarea transformare.

➤ Stop Words Remover

StopWordsRemover definește un set de cuvinte care apar cu o frecvență foarte mare într-o anumită limbă definită și care constituie părți de propoziție nerelevante în cadrul unei descrieri, reprezentând doar componente adjuvante în exprimarea fluentă a unei idei. Întrucât această mulțime nu definește un aspect important în rezultatul final (ba chiar îl poate influența într-un mod greșit), rolul acestei clase oferite de Apache este următorul: lexemele din această colecție care se regăsesc în descriere vor fi eliminate.

➤ Stemmer

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

➤ Algoritmul TF-IDF [5]

Acest algoritm constituie cea mai importantă transformare de preprocesare a instanțelor de descrieri întrucât aici are loc conversia unei liste de șiruri 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 importanță a unui cuvânt pentru un document prezent într-o colecție sau într-un corpus. Este des întâlnit în momentul în care se dorește a se utiliza un factor de greutate în regăsirea de informații sau în text-mining.

Valoarea TF-IDF asociată unui cuvânt crește proporțional cu numărul de apariții ale cuvântului respectiv într-un text, dar această valoare este echilibrată și de frecvența cuvântului din corpus, ceea ce aduce un echilibru prin faptul că unele cuvinte apar mai frecvent în mod obișnuit. 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" și "inverse document frequency".

### 1. *Term Frequency*

În cazul acestui tip de statistică,  $tf(t, d)$ , abordarea constă în a memora numărul de apariții 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 informații pe care un cuvânt le poate furniza, adică deducem dacă termenul respectiv este comun sau rar în întreaga colecție de documente. Matematic vorbind, este fracția inversă și logaritmic scalată a documentelor ce conțin cuvântul respectiv, obținută prin divizarea numărului total de documente la numărul documentelor ce conțin 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 împărțirea 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  $t$  aparținând unui document  $d$  dintr-o colecție  $D$  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.

➤ Normalizer (normalizare în  $L_2$ )

Fie vectorul  $v = (v_1, v_2, \dots, v_n)$ . Rezultatul normalizării vectorului  $v$  în  $L_2$  este vectorul  $v' = (v'_1, v'_2, \dots, v'_n)$ , unde  $v'_i = \frac{v_i}{\|v\|_2} \quad \forall i = 1..n$ , 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  $u$  și  $v$  ca fiind:

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

unde  $u'$  și  $v'$  sunt vectorii rezultați în urma normalizării în  $L_2$  a vectorilor  $u$  și  $v$ .

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

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

Din relația (1), putem deduce faptul că *distanța Euclidiană ridicată la pătrat* între doi vectori normalizați în  $L_2$ ,  $u$  și  $v$ , este proporțională cu *distanța cosinus (similaritatea cosinus)* între aceiași vectori.

În urma observațiilor menționate mai sus, putem afirma faptul că, pentru a măsura similaritatea cosinus între doi vectori TF-IDF asociați 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 obține drept măsură de distanță o măsură echivalentă (ca proporție) cu similaritatea cosinus : *distanța Euclidiană normalizată*. Menționez faptul că această ultimă transformare este necesară întrucât K-means folosește drept măsură de distanță *distanța 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 proporție cu *similaritatea cosinus*.

În concluzie, prin simpla normalizare în  $L_2$  a vectorilor obținuți prin algoritmul TF-IDF, vom putea obține, în loc de *distanța Euclidiană*, o măsură de similaritate proporțională cu *distanța cosinus*. [8]

### 2.3. Diagrame UML[9]

UML<sup>15</sup> (Unified Modeling Language) este un limbaj standard, conceput de OMG (Object Management Group) în Ianuarie 1997 și folosit pentru a specifica, vizualiza, construi și documenta artefactele unui sistem software. Inițial, a fost construit cu scopul de a surprinde comportamentul sistemelor atât software, cât și 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 își facă apariția. În acel moment, nu existau metodologii standard de organizare și de consolidare a dezvoltării orientate-obiect, motiv pentru care standardul UML și-a făcut apariția.

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 toți modelatorii, dar în același timp să fie simplu de înțeles și ușor de folosit.

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

---

<sup>15</sup> [https://ro.wikipedia.org/wiki/Unified\\_Modeling\\_Language](https://ro.wikipedia.org/wiki/Unified_Modeling_Language)

realizarea unui sistem de succes.

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 și contemporan.

➤ Diagrama UML de clase[11]

În contextul programării orientate-obiect, o diagramă UML de clase este folosită pentru reprezentarea vizuală a claselor și a interdependențelor, taxonomiei și a relațiilor de multiplicitate dintre ele. Diagramele de clasă sunt folosite și pentru reprezentarea concretă a unor instanțe de clasă, așadar obiecte, și 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. Opțional, în interiorul unei clase, pot fi reprezentate atributele și metodele acesteia (doar semnătura).
- *Interfața* – este reprezentată sub forma unui dreptunghi în interiorul căruia este notat numele interfeței. Ca și alternativă, o interfață mai poate fi reprezentată în mod simplificat sub forma unui cerc în dreptul căruia este trecut numele interfeței. Interfața poate fi considerată un caz particular de clasă.
- *Relația* - este reprezentată sub forma unei drepte orientate sau nu, care leagă două clase. O relație de asociere dintre două clase semnifică o relație de colaborare sau de înrudire între clasele conectate. Tipurile de relații 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), compoziție (reprezentată ca o dreaptă orientată, la capătul căreia se află un romb umplut), implementare (reprezentată ca o dreaptă orientată punctată al cărei capăt conține o săgeată goală) și moștenire (reprezentată ca o dreaptă orientată al cărei capăt conține o săgeată goală).

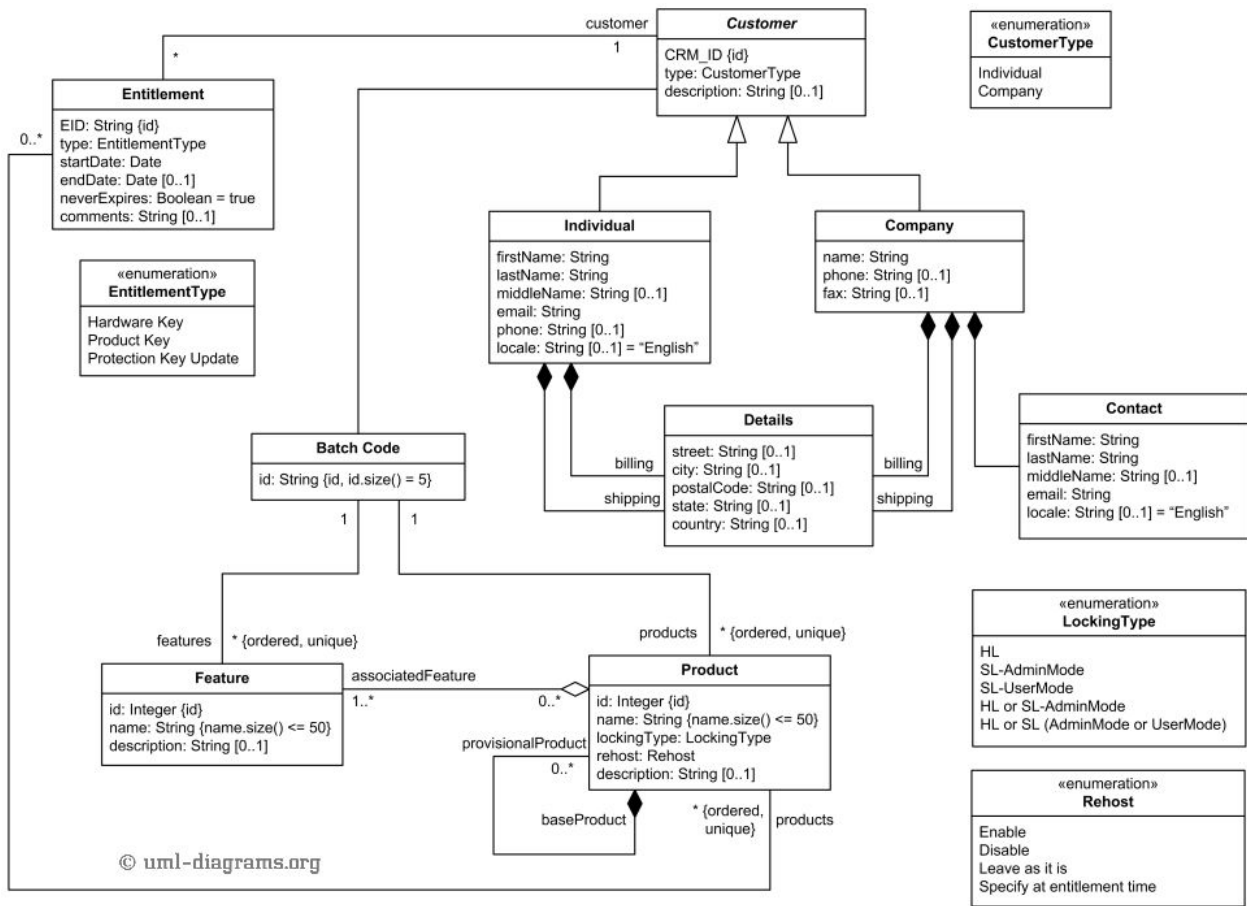


Figura 3: Exemplu de diagramă de clase care surprinde elementele componente, împreună cu relațiile dintre ele, ale unui sistem destinat domeniului de "online shopping"[10]

## 2.4. Concluzii

În cadrul acestui capitol, am prezentat sumar algoritmii care intervin în sistemul de recomandare, stagiile de preprocesare a informațiilor, precum și suportul matematic necesar înțelegerii corecte a modului de funcționare al transformărilor. Pentru etapa de învățare automată, am definit algoritmii K-means și Bisecting K-means, cel din urmă fiind folosit pentru clusterizarea de documente (secțiunea 2.1), precum și algoritmul TF-IDF (secțiunea 2.2) de transformare a unor descrieri în vectori din spațiul  $\mathbf{R}^n$ , vectori utilizați de algoritmii anterior menționați. De asemenea, am prezentat necesitatea transformărilor șirurilor de caractere (secțiunea 2.2) prin metodele de Cleansing, Tokenization, Stop Words Removal, Stemming și  $L_2$  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 înțelegerii mai bune a utilizatorului despre conținutul 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 colecție de tehnologii și API-uri pentru platforma Java, proiectate pentru a veni în sprijinul aplicațiilor de tip ”Enterprise” care, la rândul lor, pot fi clasificate, în general, ca aplicații de scară largă, distribuite, tranzacționale și disponibile pe o perioadă îndelungată de timp. Aceste aplicații au rolul de constitui suportul pentru cerințele critice din partea de „business”.

Furnizorii produselor realizate cu Java EE sunt, de obicei, servere de aplicații, servere web sau furnizori de sisteme de baze de date care pun la dispoziție clase ce implementează interfețele definite în specificațiile propuse de Java EE.

#### ➤ *Diferențe între Java EE și Java SE [14]*

Tehnologia Java este atât un limbaj de programare, cât și 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 aplicații scrise în limbajul de programare Java.

Toate platformele Java sunt alcătuite dintr-o mașină virtuală Java (Java Virtual Machine, JVM<sup>16</sup>) și dintr-o interfață de programare a aplicației (Application Programming Interface, API<sup>17</sup>). Java Virtual Machine este un program pentru platforme software și hardware, care rulează aplicații dezvoltate prin intermediul tehnologiei Java. Un API este o colecție de componente software care pot fi folosite pentru crearea altor componente sau aplicații software. Fiecare platformă Java furnizează câte o mașină virtuală și un API, ceea ce permite aplicațiilor scrise pentru platforma respectivă să poată fi executate în orice sistem compatibil, beneficiind, totodată, de avantajele limbajului de programare Java: independență de platformă, putere, stabilitate, ușurință de dezvoltare și securitate.

---

<sup>16</sup> [https://en.wikipedia.org/wiki/Java\\_virtual\\_machine](https://en.wikipedia.org/wiki/Java_virtual_machine)

<sup>17</sup> [https://ro.wikipedia.org/wiki/Application\\_Programming\\_Interface](https://ro.wikipedia.org/wiki/Application_Programming_Interface)

## ❑ Java SE

În momentul în care majoritatea oamenilor se gândesc la limbajul de programare Java, se referă, de fapt, la Java SE API. Acest API furnizează funcționalitatea de bază al acestui limbaj: definește totul, începând de la tipurile și obiectele de bază, până la clase de nivel înalt folosite pentru rețelistică, securitate, acces la baza de date, dezvoltare de interfețe grafice (GUI) și parsare de XML. În plus, față de API-ul nucleu, platforma Java SE este alcătuită dintr-o mașină virtuală, din unelte de dezvoltare, tehnologii de deployment, precum și din alte librării sau seturi de instrumente folosite frecvent în aplicațiile Java.

## ❑ Java EE

Platforma Java EE este construită peste platforma Java SE și furnizează atât un API, cât și un mediu pentru dezvoltarea și execuția aplicațiilor scalabile, sigure, multi-nivelate, de mari dimensiuni și securizate în rețea.

### 3.2. Maven

Maven [15] este un proiect găzduit de Apache Software Foundation [16] și 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 dependențele ei. Spre deosebire de uneltele precedente precum Apache Ant[17], Maven[15] folosește convenții pentru procesul de build și doar excepțiile trebuie menționate. Un fișier XML descrie proiectul de construit, dependențele acestuia legate de modulele și de componentele externe, ordinea în care acestea trebuie procesate, dar și 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 și 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<sup>18</sup> (Project Object Model) furnizează toate informațiile legate de configurarea

---

<sup>18</sup> [https://en.wikipedia.org/wiki/Apache\\_Maven#Project\\_Object\\_Model](https://en.wikipedia.org/wiki/Apache_Maven#Project_Object_Model)

unui proiect. Configurațiile generale acoperă denumirea proiectului, proprietarul acestuia și dependențele de alte proiecte. Se pot configura, de asemenea, faze individuale pentru procesul de build, stagii implementate ca plug-in-uri.

Aplicația de față, 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 conține informațiile legate de compilarea tuturor modulelor proiectului de față, lucru care are ca efect posibilitatea compilării întregului proiect dintr-o singură linie de comandă.

Fișierul POM responsabil cu configurația unui proiect are denumirea "pom.xml" și, orientativ, are următorul conținut:

```
<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>

  <!-- dependențe 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 execuția și compilarea testelor -->

      <scope>test</scope>

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

### 3.3. Spring

Spring [19] este un framework care asigură un model de configurare și de programare inteligent pentru aplicații moderne scrise în Java, indiferent de platforma folosită. Un element

cheie pe care Spring se bazează este suportul infrastructural la nivel de aplicație: Spring este orientat mai mult pe ”instalarea” aplicațiilor de tip întreprindere, în așa fel încât echipa de dezvoltare să își îndrepte atenția asupra logicii de afaceri de la nivelul de aplicație, fără a mai fi necesar să intre în configurațiile nenesare ale mediului de ”deployment”.

Câteva dintre caracteristicile framework-ului sunt: dependency injection, programarea orientată pe aspecte (AOP) care include modulul de administrare a tranzacțiilor, aplicațiile web MVC, serviciile web RESTful, suportul fundamental pentru JDBC<sup>19</sup>, JPA<sup>20</sup>, JMS<sup>21</sup>, ș.a.m.d.

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

### 3.3.1. Spring IoC Container[20]

IoC<sup>22</sup> (Inversion of Control), cunoscut ca și dependency injection<sup>23</sup> (DI), este un proces prin care obiectele își definesc dependențele (prin dependențe se înțeleg celelalte obiecte cu care lucrează) prin argumentele constructorilor, prin argumentele unei metode factory, sau prin proprietăți fixate pe instanța obiectului, după ce obiectul este construit sau returnat de o metodă de tip factory. Recipientul (container-ul), apoi, injectează acele dependențe în momentul în care este creat bean-ul. Acest proces constă în inversiunea (de unde denumirea de IoC) bean-ului în sine, ce controlează instanțierea sau locația dependențelor sale, folosind construirea directă a claselor sau utilizând un mecanism precum pattern-ul Service Locator<sup>24</sup>.

În Spring, obiectele care alcătuiesc bazele aplicației și care sunt administrate de Spring IoC Container poartă denumirea de *beans*<sup>25</sup>. Un *bean* este un obiect instanțiat, asamblat și, altminteri, administrat de un Spring IoC Container. Altfel spus, un *bean* este, de fapt, unul dintre multitudinea de obiecte din aplicație. Bean-urile, împreună cu dependențele lor, sunt reflectate în metadatele de configurare folosite de container-ul în cauză.

---

<sup>19</sup> [https://en.wikipedia.org/wiki/Java\\_Database\\_Connectivity](https://en.wikipedia.org/wiki/Java_Database_Connectivity)

<sup>20</sup> [https://en.wikipedia.org/wiki/Java\\_Persistence\\_API](https://en.wikipedia.org/wiki/Java_Persistence_API)

<sup>21</sup> <http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html>

<sup>22</sup> [https://en.wikipedia.org/wiki/Inversion\\_of\\_control](https://en.wikipedia.org/wiki/Inversion_of_control)

<sup>23</sup> [https://en.wikipedia.org/wiki/Dependency\\_injection](https://en.wikipedia.org/wiki/Dependency_injection)

<sup>24</sup> [https://en.wikipedia.org/wiki/Service\\_locator\\_pattern](https://en.wikipedia.org/wiki/Service_locator_pattern)

<sup>25</sup> <https://en.wikipedia.org/wiki/JavaBeans>

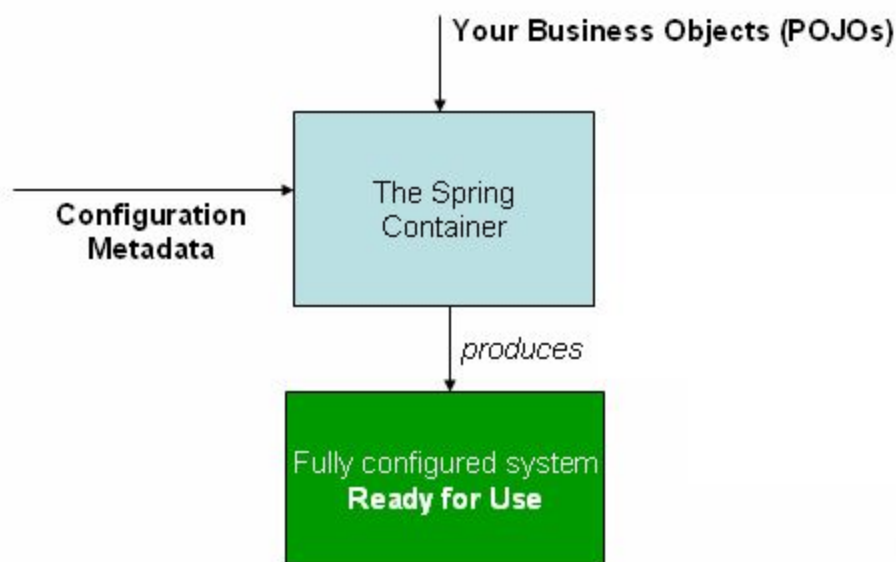


Figura 4: Modalitatea prin care bean-urile sunt configurate și instanțiate de Spring IoC Container[20]

### 3.3.2. Spring Security [21]

Modulul de securitate oferit de Spring asigură servicii de securitate pentru aplicații dezvoltate pe platforma Java EE. Spring Security este folosit, în special, întrucât caracteristicile de securitate legate de specificațiile pentru Servlet<sup>26</sup>-urile Java EE sau pentru EJB<sup>27</sup> (Enterprise Java Beans) prezintă o oarecare lipsă în ceea ce privește profunzimea necesară pentru scenariile din aplicațiile tipice de întreprindere. Este, de asemenea, bine de menționat faptul că aceste standarde nu sunt portabile la nivel de WAR<sup>28</sup> (Web Application Archive) sau EAR<sup>29</sup> (Enterprise Application Archive). Prin urmare, schimbarea mediului în care rulează serverul presupune un nivel mare de muncă dedicat reconfigurării securității aplicației în noul mediu de lucru. Folosind Spring Security, aceste probleme sunt depășite și, de asemenea, acest modul vine cu multe alte caracteristici de securitate utile și personalizabile.

Se știe faptul că două cele mai mari arii în ceea ce privește securitatea unei aplicații sunt *autentificarea* și *autorizarea* (sau *controlul accesului*), domenii care constituie țintele de bază pentru Spring Security. *Autorizarea* se referă la procesul de a decide dacă unui utilizator îi este permis să execute o acțiune într-o aplicație, însă, pentru a ajunge la punctul în care o decizie de

<sup>26</sup> [https://en.wikipedia.org/wiki/Java\\_servlet](https://en.wikipedia.org/wiki/Java_servlet)

<sup>27</sup> [https://en.wikipedia.org/wiki/Enterprise\\_JavaBeans](https://en.wikipedia.org/wiki/Enterprise_JavaBeans)

<sup>28</sup> [https://en.wikipedia.org/wiki/WAR\\_\(file\\_format\)](https://en.wikipedia.org/wiki/WAR_(file_format))

<sup>29</sup> [https://en.wikipedia.org/wiki/EAR\\_\(file\\_format\)](https://en.wikipedia.org/wiki/EAR_(file_format))



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

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

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

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

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

- se crează o clasă abstractă, de bază, pentru repository, care furnizează operații CRUD<sup>30</sup> pentru entități
- 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 și care să le invoce. Mai rău, ar trebui să repetăm acest proces de fiecare dată când se dorește 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 soluție pentru această problemă întrucât obiectivul acestui modul este cel de a îmbunătăți vizibil implementarea layer-elor de acces la date, reducând cantitatea de efort depus în acest sens. Astfel, dezvoltatorului nu-i rămâne decât să definească interfețele 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 abstracție peste un anumit JPA provider definit. Astfel, folosind această librărie, layer-ul de repository-uri, în aplicația de

---

<sup>30</sup> [https://en.wikipedia.org/wiki/Create,\\_read,\\_update\\_and\\_delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete)

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

- *Spring Data JPA* furnizează suport pentru crearea de repository-uri JPA prin extinderea interfețelor 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.

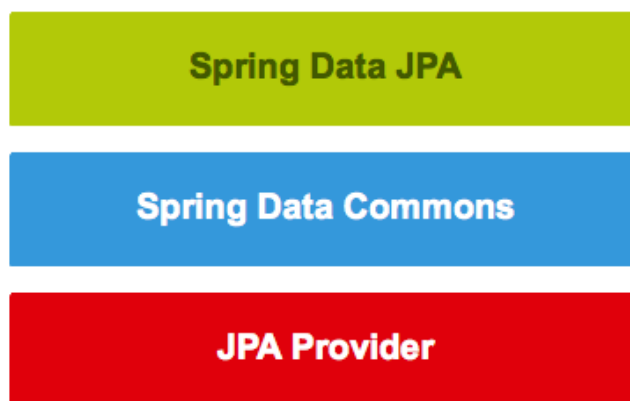


Figura 5: Structura layerelor de repository-uri dintr-un proiect ce utilizează Spring Data JPA[22]

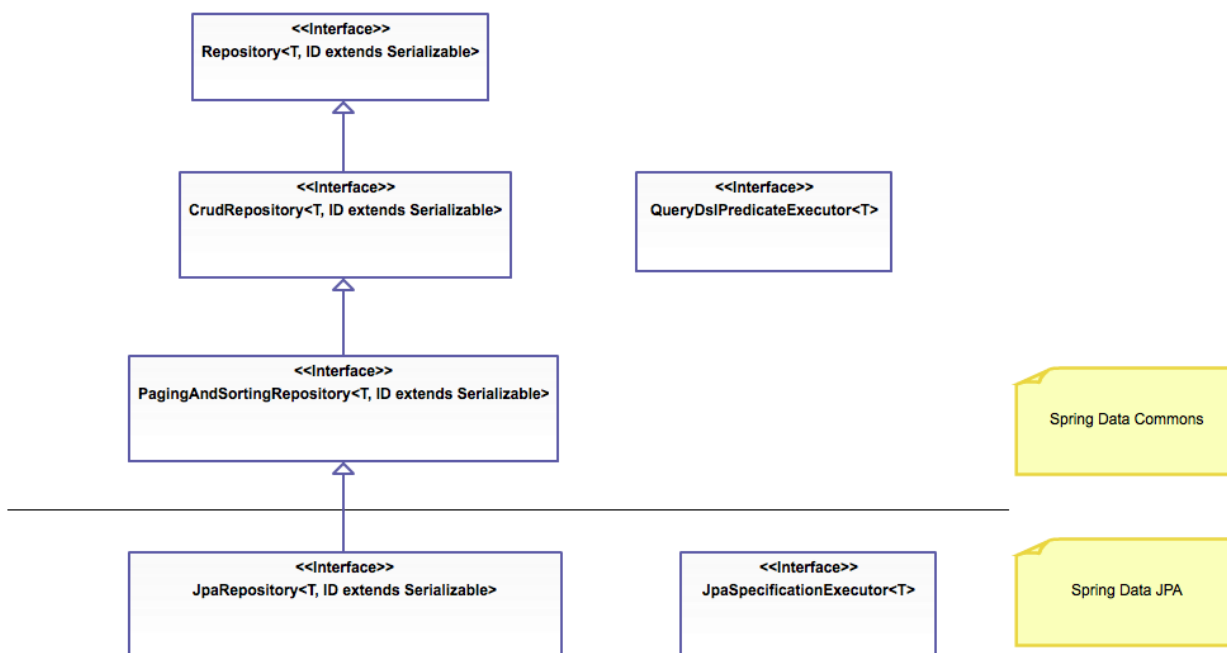


Figura 6: Ierarhia de repository-uri din Spring<sup>31</sup>

<sup>31</sup> <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 aplicația noastră să pară mai complicată, iar într-un fel așa este. Chiar dacă aduce un strat adițional peste cel de repository-uri, ne eliberează de scrierea de cod supraîncărcat, ceea ce este un compromis ce nu poate fi ignorat în aplicația noastră.

#### **3.3.4. Spring RESTful API [23]**

Serviciile web de tip REST constituie o modalitate de furnizare o interoperabilității între sisteme conectate la Internet. Aceste servicii permit sistemelor care solicită cereri să acceseze și să manipuleze reprezentări text de resurse web, folosind un set de operații predefinit. Există și alte forme de servicii web, care expun propriile mulțimi de operații, precum WSDL<sup>32</sup> sau SOAP<sup>33</sup>

Spring oferă, de asemenea, un modul care facilitează crearea de servicii REST<sup>34</sup> (Representational Web Transfer), reducând drastic cantitatea de cod scris necesar expunerii unui astfel de serviciu.

### **3.4. Hibernate [24]**

Hibernate este o unealtă ORM<sup>35</sup> (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 relațională.

Proprietatea de bază pentru Hibernate este asocierea dintre clase definite în Java și tabele în baza de date, dar și asocierea dintre tipurile de date din Java și tipurile de date SQL<sup>36</sup>. De asemenea, oferă facilități precum interogări sau regăsire de date, generând apeluri SQL<sup>2</sup> și scutind, astfel, dezvoltatorul de manipularea manuală și de conversia de obiecte a setului rezultat. Această asociere este implementată în urma configurării unui fișier XML<sup>37</sup> sau folosind adnotări Java.

Totodată, este posibilă definirea unor relații de tipul one-to-many sau many-to-many între

---

<sup>32</sup> <https://www.w3.org/TR/wsdl>

<sup>33</sup> <https://en.wikipedia.org/wiki/SOAP>

<sup>34</sup> [https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer)

<sup>35</sup> [https://en.wikipedia.org/wiki/Object-relational\\_mapping](https://en.wikipedia.org/wiki/Object-relational_mapping)

<sup>36</sup> <https://www.w3schools.com/SQL/default.asp>

<sup>37</sup> <https://www.w3schools.com/xml/>

clase, lucru care se răsfrânge automat și asupra schemelor bazelor de date (independent de dialectul folosit).

În jargonul Hibernate, o *entitate* este un obiect de sine stătător prezent în mecanismul de persistență al framework-ului, care poate fi manipulat independent de alte obiecte. În contrast, o componentă este subordonată unei entități și 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 entității de tipul Album, mergând pe premisa că Piese 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 învățare automată ce aparține de Spark<sup>38</sup>, al cărei obiectiv principal este cel de a face ca algoritmi de învățare automată să devină scalabili și ușor de folosit.

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

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

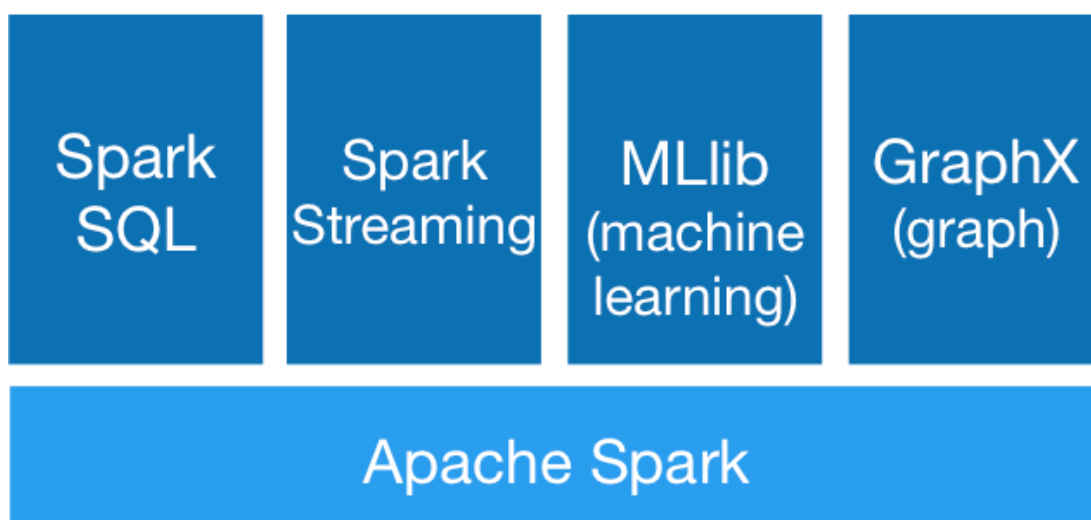


Figura 7: Spark împuternicește un set de librării, printre care și MLlib[27]

---

<sup>38</sup> <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, folosește software-ul Graphviz<sup>39</sup> pentru a expune aceste diagrame.

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

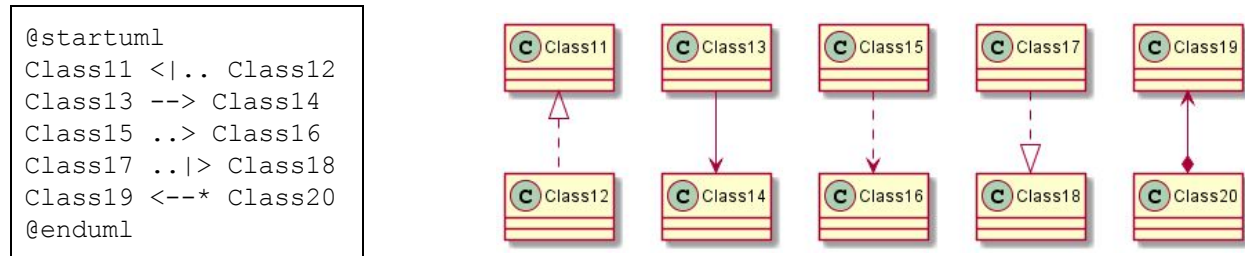


Figura 8: Diagrama de clase generată folosind PlantUML[28]

### 3.7. Apache Wicket [30]

Apache Wicket este un framework de aplicații web bazat pe componente, folosit în cadrul limbajului de programare Java și este conceptual similar cu JavaServer Faces<sup>40</sup> și Tapestry<sup>41</sup>.

Aplicațiile Wicket sunt, de fapt, arbori de componente, care se folosesc de așa-numiți ”ascultători delegați” pentru a răspunde la cereri HTTP pentru link-uri și formulare, la fel cum componentele din Swing<sup>42</sup> răspund la evenimentele trimise de mouse sau prin apăsarea de taste.

Wicket folosește XHTML<sup>43</sup> pentru șablonare (ce forțează o separare clară între prezentare și logica de business). Fiecare componentă este legată de un element din XHTML și 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 susținută, în spate, de propriul model, care constituie starea

<sup>39</sup> <http://www.graphviz.org/>

<sup>40</sup> [https://en.wikipedia.org/wiki/JavaServer\\_Faces](https://en.wikipedia.org/wiki/JavaServer_Faces)

<sup>41</sup> <http://tapestry.apache.org/>

<sup>42</sup> [https://en.wikipedia.org/wiki/Swing\\_\(Java\)](https://en.wikipedia.org/wiki/Swing_(Java))

<sup>43</sup> <https://ro.wikipedia.org/wiki/XHTML>

componentei în sine. Framework-ul nu are cunoștințe despre felul în care componentele interacționează cu propriile modele, care sunt tratate ca obiecte opace și automat serializate și persistate între cereri. Modelele mai complexe, totuși, ar putea fi construite ca fiind detașabile și furnizează un mecanism pentru a-și aranja propria memorare și restaurare la începutul și la finalul fiecărui ciclu de cerere. Întrucât Wicket nu impune niciun layer special de persistență a obiectelor sau de ORM, aplicația web de față poate folosi acest framework în combinație cu obiecte Hibernate, Spring Beans<sup>44</sup> sau obiecte POJO<sup>45</sup> ca modele.

În Wicket, toată starea server-ului este automat administrată. Fiecare componentă server-side din pagină menține 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 și a aplicațiilor web, care conține șabloane HTML<sup>46</sup> și CSS<sup>47</sup> destinate tipografiei, formularelor, butoanelor, componentelor de navigare, precum și, opțional, unor extensii JavaScript<sup>48</sup>. Spre deosebire de multe alte framework-uri, interesul este orientat doar spre dezvoltarea pe partea de front-end<sup>49</sup>.

#### ➤ *Foi de stil*

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

#### ➤ *Componente reutilizabile*

Pe lângă elementele HTML obișnuite, Bootstrap conține alte elemente de interfață utilizate frecvent. Componentele sunt implementate drept clase CSS, care trebuie aplicate pe

---

<sup>44</sup> <https://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html>

<sup>45</sup> [https://en.wikipedia.org/wiki/Plain\\_old\\_Java\\_object](https://en.wikipedia.org/wiki/Plain_old_Java_object)

<sup>46</sup> [https://ro.wikipedia.org/wiki/HyperText\\_Markup\\_Language](https://ro.wikipedia.org/wiki/HyperText_Markup_Language)

<sup>47</sup> <https://www.w3.org/Style/CSS/Overview.en.html>

<sup>48</sup> <https://ro.wikipedia.org/wiki/JavaScript>

<sup>49</sup> [https://en.wikipedia.org/wiki/Front-end\\_web\\_development](https://en.wikipedia.org/wiki/Front-end_web_development)

anumite elemente HTML dintr-o pagină.

#### ➤ Componente JavaScript

Bootstrap vine la pachet cu câteva componente JavaScript sub forma unor plugin-uri jQuery<sup>50</sup>. Acestea oferă elemente de interfață adiționale, precum căsuțe de dialog, tooltip-uri sau carousel-uri<sup>51</sup>. Ele extind funcționalitatea câtorva elemente de interfață existente, cum ar fi funcția de autocompletare pentru câmpurile de input.

## 4. Arhitectura aplicației

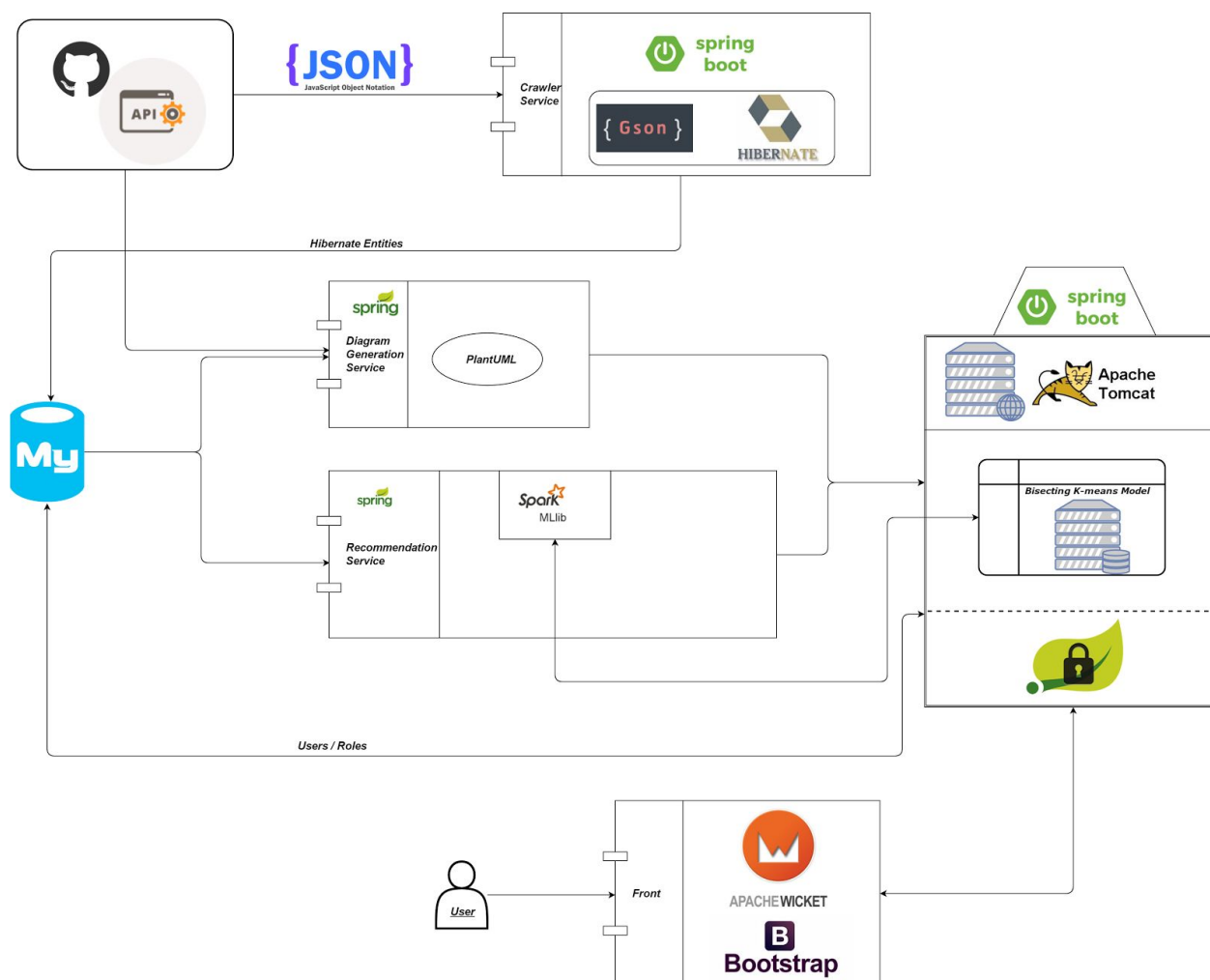


Figura 9: Arhitectura aplicației

<sup>50</sup> <https://ro.wikipedia.org/wiki/JQuery>

<sup>51</sup> [https://www.w3schools.com/bootstrap/bootstrap\\_carousel.asp](https://www.w3schools.com/bootstrap/bootstrap_carousel.asp)

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 a pune la dispoziție fluxul de lucru al aplicației de față.

#### 4.1. Serviciul de crawling

GitHub expune un API care facilitează căutarea de repository-uri publice, ba mai mult, suplimentează informații 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 fișierul "README.MD", link-ul către repertoriu, ș.a.m.d.

Un simplu exemplu de informații 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 îngroșate constituie informațiile utile oferite de API pentru prezenta aplicație, accesate și asociate cu o entitate Hibernate pentru a putea fi salvate în baza de date.

Această facilitate oferită de GitHub<sup>52</sup> vine și cu o limitare a numărului de cereri neautentificate (60 cereri/oră), motiv pentru care am fost nevoit să creez 10 token-uri<sup>53</sup> de acces care să restrângă numărul de cereri la 5000 cereri/oră per token. Astfel, pentru fiecare request<sup>54</sup> trimis la interval de 2 secunde (pentru a nu supraîncărca API-ul), token-ul este schimbat prin intermediul unui Observer<sup>55</sup>, ceea ce a mărit numărul maxim de cereri care se pot trimite la 50 000/oră.

În urma acestei etape, s-au obținut 1892 de proiecte dezvoltate în diferite limbaje de programare, număr suficient de mare pentru a testa utilitatea și gradul de corectitudine al serviciului de recomandare. De menționat, totuși, este faptul că, întrucât instanțele de proiecte nu au putut fi controlate de API în ceea ce privește descrierea lor, unele dintre acestea nu sunt suficient de detaliate pentru a reflecta conținutul proiectului.

## 4.2. Serviciul de recomandări

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

Astfel, sub-procesul de *clusterizare* are loc la prima rulare a aplicației, accesând un API expus în acest sens sub forma unui URL care activează un serviciu ce creează (în urma aplicării algoritmului Bisecting K-means<sup>56</sup> pe setul de date din baza de date) un model salvat în storage-ul serverului, pe baza căruia are loc distribuția repository-urilor către clustere. Pentru fiecare instanță de repository din baza de date, în urma clusterizării, se actualizează și numărul cluster-ului la care aceasta aparține. 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 aplicației.

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

---

<sup>52</sup> <https://en.wikipedia.org/wiki/GitHub>

<sup>53</sup> [https://en.wikipedia.org/wiki/Access\\_token](https://en.wikipedia.org/wiki/Access_token)

<sup>54</sup> <https://v4.datafeedr.com/documentation/373>

<sup>55</sup> [https://www.tutorialspoint.com/design\\_pattern/observer\\_pattern.htm](https://www.tutorialspoint.com/design_pattern/observer_pattern.htm)

<sup>56</sup> <https://stackoverflow.com/questions/6871489/bisecting-k-means-clustering-algorithm-explanation>

modelul care oferă drept output identificatorul cluster-ului prezis de algoritm, ce constituie grupul de instanțe la care ar trebui ca repository-ul să aparțină în urma utilizării măsurii de distanță față de celelalte proiecte, reprezentată de *similaritatea cosinus*. Întrucât identificatorul cluster-ului nu este suficient pentru a oferi recomandarea, pentru fiecare instanță de repertoriu aparținând cluster-ului respectiv, se măsoară distanța cosinus dintre acel repository și descrierea dată de proiectul utilizatorului, apoi acest scor este folosit pentru a sorta, în ordinea similarității, repertoriile indicate. Primul repository va constitui, astfel, cea mai bună recomandare de proiect existent în aplicația de față.

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

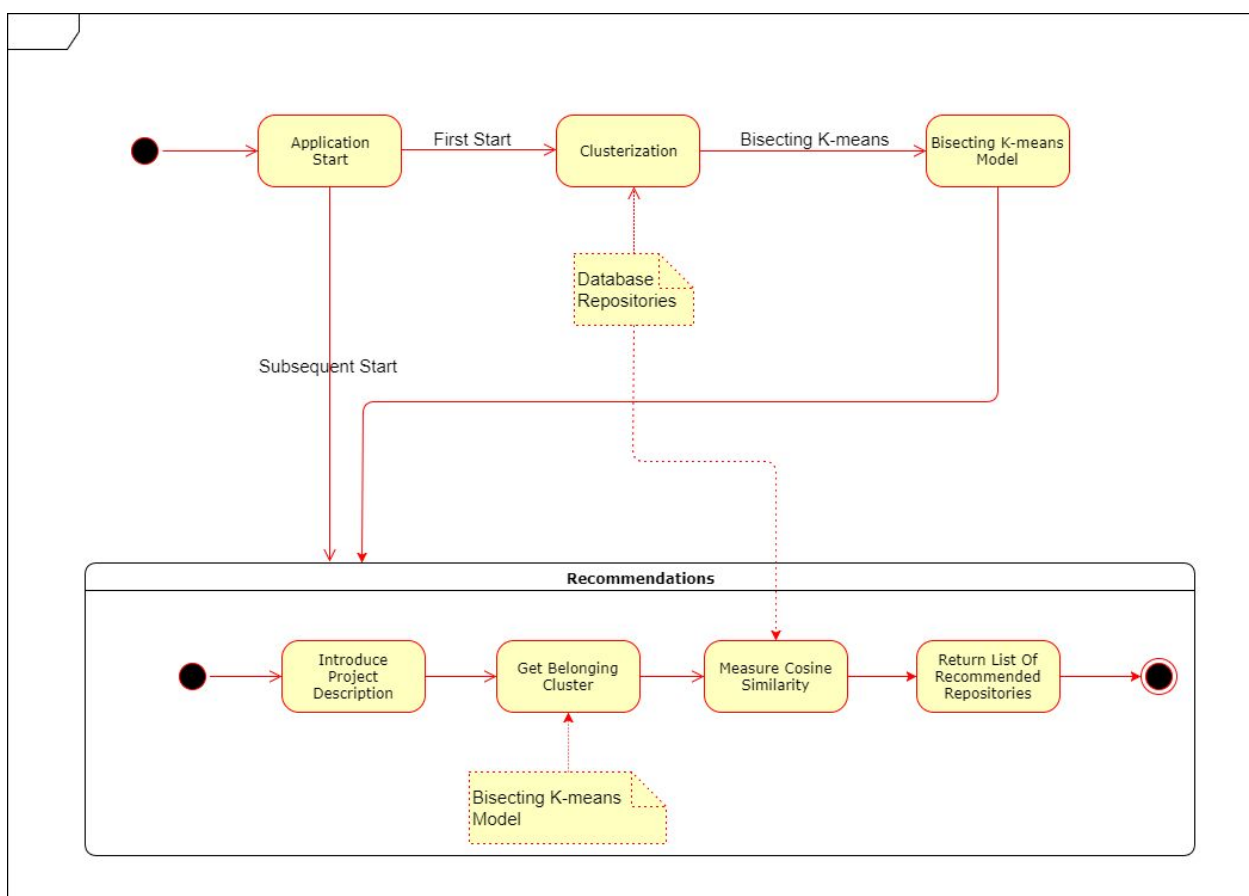


Figura 10: Diagrama de stare a serviciul de recomandare

#### 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 construiește modelul necesar predicției cluster-ului la care aparțin noile proiecte furnizate de utilizatori. Următoarea diagramă de stare descrie exact pașii executați pentru clusterizare.

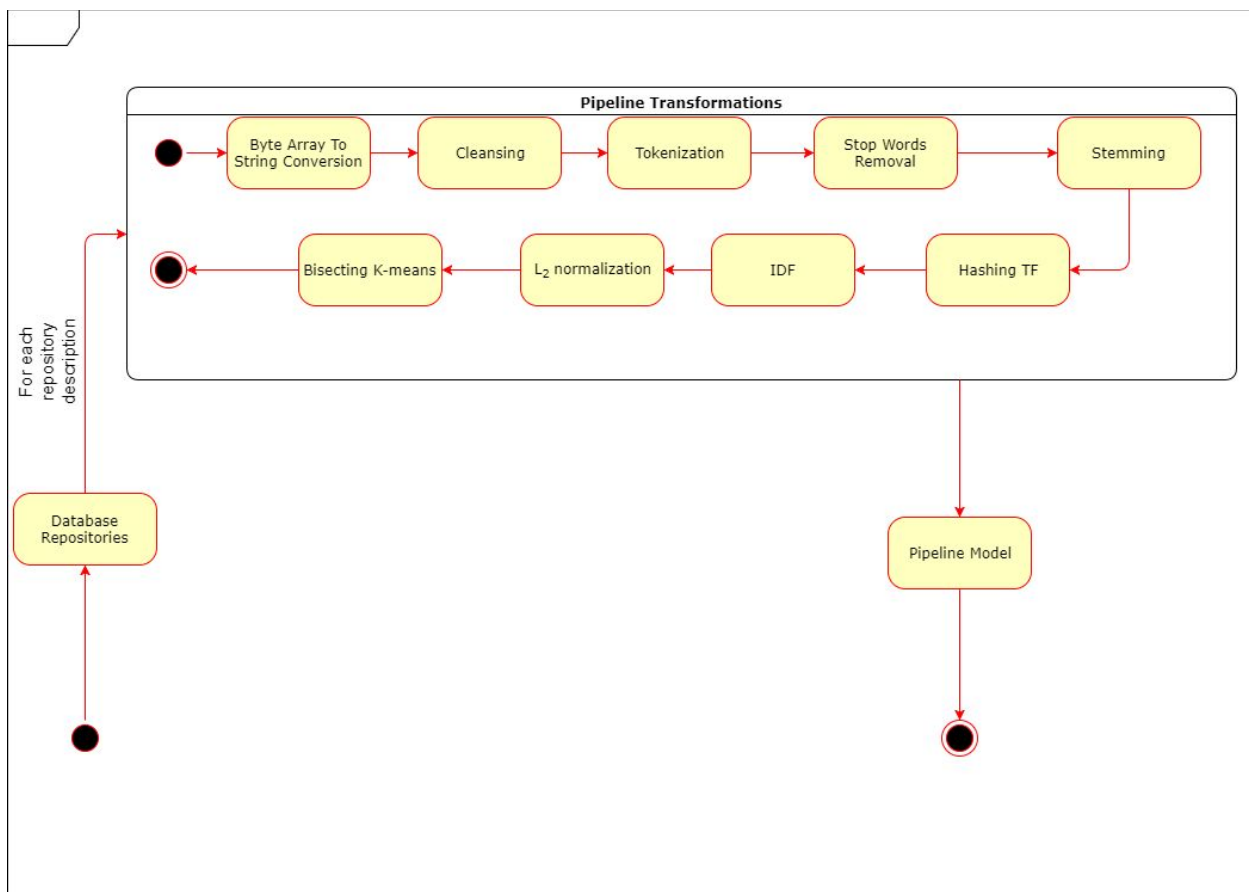


Figura 11: Diagrama de stare pentru sub-procesul de clusterizare

În diagrama de mai sus, s-a adăugat o transformare nemenționată până acum (*Byte Array To String Conversion*), schimbare realizată în cadrul aceluiași pipeline cu celelalte modificări și 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 existenței unor caractere nerecunoscute de vendor-ul bazei de date, lucru care ar conduce la apariția 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 instanței în cauză în vectorul normalizat, dar, în același timp, proiectul este asignat și către cluster-ul care conține proiectele asemănătoare ca funcționalitate. După ce este luată decizia alocării către un cluster, pentru fiecare instanță ce aparține 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 selecția repository-urilor din baza de date aferentă procesului de furnizare a recomandărilor.

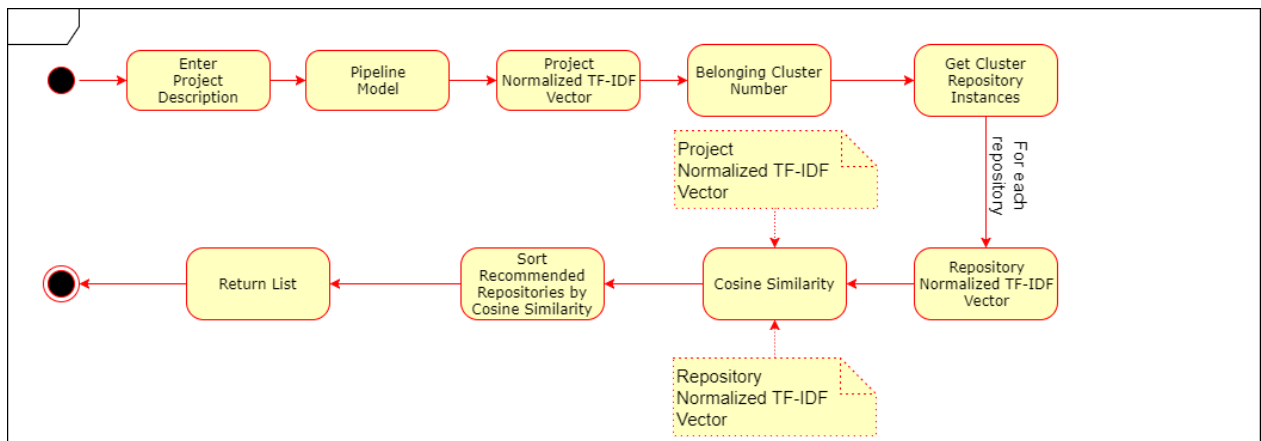


Figura 12: Diagrama de stare pentru sub-procesul de recomandare de repository-uri

#### 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*<sup>57</sup> ce constă în construirea și folosirea de expresii regulate pentru a potrivi și prelua denumiri de clase, semnături de metode, declarații de variabile și relațiile dintre clase.

*PlantUML*<sup>58</sup> împreună cu extensia *PlantUML Dependency*<sup>59</sup> oferă posibilitatea de a parsa fișiere sursă dintr-un proiect Java cu scopul de a genera diagrama de clase împreună cu relațiile

<sup>57</sup> [https://ro.wikipedia.org/wiki/Inginerie\\_invers%C4%83](https://ro.wikipedia.org/wiki/Inginerie_invers%C4%83)

<sup>58</sup> <http://plantuml.com/>

<sup>59</sup> <http://plantuml-depend.sourceforge.net/>

dintre ele. Întrucât extensia nu are o expresie regulată care să poată extrage atât declarările de variabile, cât și semnatura metodelor, am creat încă două expresii regulate care vin cu o soluție pentru această problemă.

Extensia *PlantUML Dependency* funcționează în felul următor: după ce fișierele sursă sunt analizate, se construiește un arbore care definește relațiile dintre clase, rezultând un fișier text care respectă regulile din cadrul framework-ului de bază, *PlantUML*. În momentul în care acest fișier este generat, sunt reanalizate fișierele pentru a insera denumirile de metode, declarările de variabile și modificatorii de acces, apoi este generată diagrama UML finală în format SVG<sup>60</sup>, trimisă apoi către utilizatorul care a solicitat vizionarea acesteia.

#### 4.4. Layer-ul de prezentare

Pentru prezentare, s-a expus atât o interfață web, cât și un API.

Interfața web este construită folosind un șablon oferit de Themeum<sup>61</sup> (ce are la bază, la rândul său, framework-ul de front-end Bootstrap<sup>62</sup>), adaptată cu Wicket<sup>63</sup>, iar administrarea sesiunilor este realizată folosind framework-ul Spring Security<sup>64</sup>.

Pentru expunerea API-ului, am folosit modulul *Spring RESTful Web Service*<sup>65</sup>, iar rutele expuse sunt următoarele:

- */train* : accesarea acestei rute activează serviciul de antrenare a modelului folosit pentru clusterizare, iar fișierul JSON returnat conține instanțele din baza de date și clusterelor către care repertoriile au fost distribuite;
- */recommend?id=\${id}&description=\${description}&recommendations=\${number\_of\_recommendations}* : returnează primele *\${number\_of\_recommendations}* recomandări pentru un proiect cu descrierea *\${description}* și ID-ul *\${id}*;
- */predict?id=\${id}&description=\${description}* : distribuie un proiect nou cu descrierea *\${description}* și ID-ul *\${id}*, pe baza modelului Pipeline descris în capitoul 4.2.1, către cluster-ul la care ar trebui asociat. JSON-ul returnat conține doar numărul cluster-ului

---

<sup>60</sup> [https://www.w3schools.com/html/html5\\_svg.asp](https://www.w3schools.com/html/html5_svg.asp)

<sup>61</sup> <http://demo.themeum.com/html/oxygen/>

<sup>62</sup> <http://getbootstrap.com/>

<sup>63</sup> <https://wicket.apache.org/>

<sup>64</sup> <https://projects.spring.io/spring-security/>

<sup>65</sup> <https://spring.io/guides/gs/rest-service/>

către care instanța dată este distribuită.

#### **4.5. Concluzii**

În cadrul acestui capitol, am prezentat arhitectura generală a aplicației împreună cu detaliile de funcționare pentru fiecare modul/serviciu în parte și felul în care modulele descrise relaționează pentru a oferi funcționalitatea întregii aplicații. De asemenea, au fost menționate framework-urile care stau la baza serviciilor, precum și felul în care implementările expuse de aceste librării au fost îmbinate pentru a oferi utilizatorului experiența oferită de aplicația în cauză.

## 5. Studii de caz

Să presupunem că o companie de IT își exprimă intenția de a dezvolta un proiect ce presupune o implementare despre care nu există o cunoștință vastă asupra felului în care ar trebui realizată. În acest caz, se poate apela la aplicația de față, furnizând o simplă descriere a ceea ce se dorește a se implementa, iar în funcție de ceea ce există deja implementat ca proiect în baza de date, se furnizează o recomandare ce poate reprezenta un punct de plecare în dezvoltarea proiectului. Suplimentar, dacă recomandarea constituie o implementare Java, aplicația vine în ajutorul utilizatorului, oferind posibilitatea descărcării unei diagrame de clase pentru o înțelegere mai bună asupra a ceea ce reprezintă, de fapt, proiectul. Totodată, pentru fiecare recomandare, este afișat un scor ce reprezintă gradul de similaritate între ceea ce dorește utilizatorul să obțină și ceea ce există deja implementat ca funcționalitate.

### 5.1. Caz de utilizare favorabil

Pentru cazul ce urmează a fi prezentat, s-a căutat un proiect ce folosește “*învățarea automată pentru predicții de locații*”. Se observă că primul proiect recomandat este cel mai relevant, întrucât conține o descriere mai detaliată despre funcționalitate, care se pliază mult mai bine pe dorințele utilizatorului. Deși are legătură cu domeniul căutat, restul de descrieri, după cum se observă, sunt mai generale, deci nu ar putea reflecta cu exactitate funcționalitatea cerută, motiv pentru care scorul de similaritate este mult mai mic față de prima recomandare.

## Get repository recommendations

project that uses machine learning for location prediction

Get Recommendations

#	Repository Name	Languages	Similarity	Description	Repository Link	Download UML class diagram
1	whereami	PYTHON	0.491	Uses WiFi signals :signal_strength: and machine learning to predict where you are	<a href="#">View</a>	
2	smile	JAVA, HTML, SCALA, CSS, JAVASCRIPT	0.243	Statistical Machine Intelligence & Learning Engine	<a href="#">View</a>	<a href="#">Download</a>
3	MLAlgorithms	PYTHON	0.218	Minimal and clean examples of machine learning algorithms	<a href="#">View</a>	
4	awesome-machine-learning	PYTHON	0.214	A curated list of awesome Machine Learning frameworks, libraries and software.	<a href="#">View</a>	
5	NakedTensor	PYTHON	0.208	Bare bone examples of machine learning in TensorFlow	<a href="#">View</a>	
6	Theano-Tutorials	PYTHON	0.171	Bare bones introduction to machine learning from linear regression to convolutional neural networks using Theano.	<a href="#">View</a>	
7	have-fun-with-machine-learning	PYTHON	0.168	An absolute beginner's guide to Machine Learning and Image Classification with Neural Networks	<a href="#">View</a>	
8	h2o-2	JAVA, PYTHON, CSS, HTML, JAVASCRIPT	0.167	h2o = fast statistical, machine learning & math runtime for bigdata	<a href="#">View</a>	<a href="#">Download</a>

Figura 13: Screenshot din aplicație cu primele 8 recomandări pentru descrierea furnizată

Un alt exemplu de caz favorabil este ales folosind o descriere mai generală, pentru a demonstra corectitudinea serviciului de recomandare chiar și la un nivel abstract. Astfel, se poate observa că, în următoarea figură, recomandările pentru un proiect de *”deep learning tutorial”*, pe lângă faptul că sunt relevante, au un scor de similaritate foarte apropiat.



Get repository recommendations						
deep learning tutorial						
Get Recommendations						
#	Repository Name	Languages	Similarity	Description	Repository Link	Download UML class diagram
1	TensorFlow-and-DeepLearning-Tutorial	PYTHON, JAVASCRIPT	0.865	TensorFlow & Deep Learning Tutorial	<a href="#">View</a>	
2	pytorch-tutorial	PYTHON	0.662	PyTorch Tutorial for Deep Learning Researchers	<a href="#">View</a>	
3	DeepLearningTutorials	PYTHON	0.52	Deep Learning Tutorial notes and code. See the wiki for more info.	<a href="#">View</a>	
4	MachineLearning	PYTHON, C	0.491	Basic Machine Learning and Deep Learning	<a href="#">View</a>	
5	srez	PYTHON	0.395	Image super-resolution through deep learning	<a href="#">View</a>	
6	learning-to-learn	PYTHON	0.392	Learning to Learn in TensorFlow	<a href="#">View</a>	
7	chainer	PYTHON, C	0.391	A flexible framework of neural networks for deep learning	<a href="#">View</a>	
8	tensorlayer	PYTHON	0.383	TensorLayer: Deep Learning and Reinforcement Learning Library for Researcher and Engineer.	<a href="#">View</a>	

Figura 14: Screenshot din aplicație cu primele 8 recomandări pentru descrierea furnizată

## 5.2. Caz de utilizare nefavorabil

În acest subcapitol, prezentăm un caz în care serviciul de recomandare nu poate oferi un randament ridicat. Astfel, considerând faptul că un utilizator ar dori să caute un proiect cu o viziune de dezvoltare asupra unor ”orașe deștepte”, recomandările oferite de serviciu sunt cele ilustrate în *figura 15*. Din imagine se poate observa că până și scorul de similaritate este unul infim, ceea ce indică faptul că cea mai bună recomandare ce a putut fi oferită în cazul de față nu poate fi luată în considerare cu un grad ridicat de siguranță întrucât nu reflectă funcționalitatea cerută de utilizator. Dacă ne uităm și la descrierea primei recomandări, observăm clar că propunerea oferită nu este deloc relevantă în ceea ce privește cerințele utilizatorului.

Dacă dorim să săpăm și mai adânc pentru a aduce la suprafață motivele pentru care randamentul dat este atât de mic, este îndeajuns să observăm faptul că în baza de date, practic, nu există niciun proiect legate de orașe, dar poate fi luat în calcul și faptul că descrierea este una abstractă și nu oferă informații mai detaliate despre cerințele proiectului.

smart city development vision						
Get Recommendations						
#	Repository Name	Languages	Similarity	Description	Repository Link	Download UML class diagram
1	UltimateAndroid	JAVA, HTML	0.174	UltimateAndroid is a rapid development framework for developing your apps	<a href="#">View</a>	<a href="#">Download</a>
2	pwntools	PYTHON	0.108	CTF framework and exploit development library	<a href="#">View</a>	
3	pinax	PYTHON	0.088	a Django-based platform for rapidly developing websites	<a href="#">View</a>	
4	hug	PYTHON, HTML	0.057	Embrace the APIs of the future. Hug aims to make developing APIs as simple as possible, but no simpler.	<a href="#">View</a>	
5	material-dialogs	JAVA	0	A beautiful, fluid, and customizable dialogs API.	<a href="#">View</a>	<a href="#">Download</a>
6	realm-java	JAVA, HTML	0	Realm is a mobile database: a replacement for SQLite & ORMs	<a href="#">View</a>	<a href="#">Download</a>
7	okhttputils	JAVA	0	okhttp的辅助类	<a href="#">View</a>	<a href="#">Download</a>
8	intellij-community	JAVA, PYTHON, HTML, JAVASCRIPT, CSS, C, SCALA, RUBY	0	IntelliJ IDEA Community Edition	<a href="#">View</a>	<a href="#">Download</a>
9	Side-Menu.Android	JAVA	0	Side menu with some categories to choose.	<a href="#">View</a>	<a href="#">Download</a>
10	ZhihuDailyPurify	JAVA	0	Purified version of Zhihu Daily - 更纯净的知乎日报	<a href="#">View</a>	<a href="#">Download</a>
11	Small	JAVA, RUBY, HTML, SWIFT, CSS	0	A small framework to split app into small parts	<a href="#">View</a>	<a href="#">Download</a>
12	hackpad	JAVA, JAVASCRIPT, CSS, SCALA,	0	Hackpad is a web-based realtime wiki.	<a href="#">View</a>	<a href="#">Download</a>

Figura 15: Screenshot din aplicație cu primele 12 recomandări pentru descrierea furnizată

### 5.3. Limitări ale aplicației și analiza erorilor

Momentan, prezenta aplicație este limitată în ceea ce privește unele funcționalități din cauza unor constrângeri ce țin de următoarele aspecte:

- proiectele luate de pe GitHub drept instanțe de test pentru recomandările oferite de aplicația curentă *nu respectă nicio schemă* în ceea ce privește *descrierea funcționalităților*, chiar mai mult, unele sunt foarte scurte și generale. Acest fapt are drept urmare o influență negativă asupra clusterizării descrierilor, ceea ce ar putea conduce la grad scăzut al calității recomandărilor;
- diagramele UML nu pot fi generate decât pentru proiectele care au fost dezvoltate pentru platforma Java, întrucât metoda de reverse engineering aplicată asupra acestora se pliază numai pe sintaxa acestui limbaj de programare;
- diagramele nu pot fi generate pentru proiecte de dimensiune mari întrucât necesită un timp foarte îndelungat de procesare, iar diagrama rezultată ar fi prea detaliată, lucru care nu ar face decât să dezorienteze utilizatorul în încercarea lui de înțelegere mai profundă a

proiectului;

- numărul de proiecte descărcate de pe GitHub este unul relativ mic, iar varietatea domeniilor pentru care acestea au fost dezvoltate este la un nivel foarte scăzut, motiv pentru care recomandările care constituie o propunere bună sunt puține și, astfel, criteriul de căutare este restrâns.

## 6. Concluzii finale

Așa cum am menționat și în capitolul de introducere, scopul acestei aplicații este cel de a oferi un punct de start proiectelor ce se doresc a fi dezvoltate de firmele IT, un element esențial în cadrul procesului de analiză și de concepere a unei arhitecturi care să se plieze pe funcționalitățile cerute. Cu ajutorul repository-urilor deja existente luate de pe GitHub, am reușit să simulăm o astfel de aplicație care să ofere recomandări în sensul dezvoltărilor de noi proiecte, ba chiar se poate observa un rezultat mai mult decât satisfăcător, având în vedere problemele majore întâmpinate și care au fost deja menționate în capitolul 5.3.

### 6.1. Contribuții personale și rezultate

În dezvoltarea părții practice a prezentei lucrări de licență, am întâmpinat o multitudine de dificultăți și probleme, însă, în cele din urmă, am reușit să îmbin o serie de elemente ce țin de recomandări de repertorii și de generare de diagrame astfel încât să obțin un rezultat care să se ridice la nivelul așteptărilor proprii.

Astfel, am reușit să găsesc un model de transformare a descrierilor text care să poată constitui un input corect din punct de vedere logic și matematic pentru algoritmul de clusterizare Bisecting K-means în așa fel încât output-ul oferit de acesta să fie unul cât mai sigur pentru repartizarea corectă a noilor instanțe la clusterelor ale căror reprezentanți sau membri ar putea constitui un set de recomandări. De asemenea, modelul matematic din spatele serviciului de recomandări a fost analizat și testat de către mine din punctul de vedere al corectitudinii, iar felul în care am reușit să îmbin aceste mici detalii având la dispoziție câteva din uneltele oferite de librăriile în cauză constituie o altă parte din contribuție.

Modulul de crawling ce folosește, în spate, o serie de librării care constituie un adjuvant în manipularea datelor în format JSON, precum și asocierea informațiilor oferite de API-ul GitHub cu entități Hibernate pentru a putea salva repertoriile în baza de date, constituie alte contribuții personale aduse proiectului.

Am reușit să adaug extensiei PlantUML o funcționalitate suplimentară prin care pot fi preluate semnatura metodelor și declarațiile de variabile pentru procesul de reverse engineering aferent generării de diagrame UML, ba mai mult, am putut oferi către descărcare această imagine

prin expunerea unui serviciu într-o interfață web. Totodată, am integrat template-ul Oxygen ce utilizează framework-ul Bootstrap cu librăria Apache Wicket pentru a putea expune o interfață user-friendly prin care să poată fi testată aplicația în sine. În plus, am reușit să adaug un layer de securitate pentru modulul de front-end, adaptând framework-ul Spring Security la aplicația curentă.

Serviciile menționate până acum le-am expus, deopotrivă, printr-un API creat cu Spring Web Services. Arhitectura aplicației a fost gândită integral de mine, iar procesul de îmbinare a tehnologiilor care au dus la realizarea proiectului curent constituie, și el, un aport personal, alcătuind, în final, o aplicație business modulară reușită.

## **6.2. Direcții de viitor**

Proiectul de față poate fi extins cu o multitudine de elemente, dintre care cele mai importante sunt:

- conceperea și impunerea unei scheme de reprezentare a repository-urilor pentru o clusterizare mai corectă în așa fel încât recomandările să fie mai relevante;
- adăugarea unei extensii a modulului de generare a diagramelor pentru alte limbaje de programare orientate obiect (C++, C#, ș.a.m.d);
- împărțirea utilizatorilor în două grupe: companii și utilizatori normali, diversificând, astfel, funcționalitățile sau restricțiile;
- utilizarea unui cluster Spark extern pentru operații complexe și consumatoare de timp, aspect ce ar aduce o notă de eficiență proceselor de lungă durată (clusterizarea, transformările de text, calcule matematice);
- introducerea unui panou de administrație pentru managementul facil al utilizatorilor și al proiectelor existente în aplicație, precum și dezvoltarea unui modul de analiză a statisticilor ce țin de procesul de recomandare;
- implementarea unui job care să actualizeze modelul construit în urma aplicării algoritmului Bisecting K-means pe setul de date deja existent cu eventualele noi repository-uri adăugate.

## 7. 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](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\\_\(mathematics\)](https://en.wikipedia.org/wiki/Norm_(mathematics))
- [7] [https://en.wikipedia.org/wiki/Cosine\\_similarity](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](http://control.aut.utcluj.ro/hmihai/doku.php?id=uml:uml_clase)
- [12] <http://www.oracle.com/technetwork/java/javase/overview/index.html>
- [13] <https://stackoverflow.com/questions/106820/what-is-java-ee>
- [14] <http://docs.oracle.com/javase/6/firstcup/doc/gkhoy.html>
- [15] [https://en.wikipedia.org/wiki/Apache\\_Maven](https://en.wikipedia.org/wiki/Apache_Maven)
- [16] [https://ro.wikipedia.org/wiki/Funda%C8%9Bia\\_Apache](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](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\\_\(front-end\\_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))*

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