Aplicație bazată pe microservicii pentru identificarea persoanelor cu interese similare

Radu-Valentin Cornea
UNIVERSITATEA TEHNICĂ
„Gheorghe Asachi" din IAȘI
FACULTATEA DE AUTOMATICĂ
ȘI CALCULATOARE
Iași, România
radu-valentin.cornea@student.tuiasi.ro
Coordonator științific:
Ș.l. dr. inf. Tiberius Dumitriu

***Abstract*—În această lucrare se propune prezentarea unor metode de recomandare a anumitor persoane, în funcție de preferințele pe care le au în comun cu ceilalți utilizatori, folosind metode din sfera inteligenței artificiale. Accentul cade pe grupuri de interes din sfera educațională, dar poate fi extins la orice alte domenii.**

Studiul redă câteva aspecte generale ale aplicațiilor bazate pe microservicii, modalități de securizare a aplicației, tehnologiile folosite, motivația realizării unei aplicații diferite de cele existente, arhitectura realizată și rezultatele obținute. Experimentele efectuate presupun aplicarea unor algoritmi de inteligență artificială, precum K-Nearest Neighbors (KNN) pentru a i se recomanda unui nou utilizator, pe baza preferințelor sale, un grup de persoane (cunoscute sau nu) ce au interese similare. Pentru aceasta se utilizează valori diferite ale parametrului K, precum și mai multe metrici (Jaccard, euclidiană, cosine etc.) cu scopul de a determina cea mai potrivită recomandare. Rezultatele obținute sunt prezentate sub formă de tabele și grafice care evidențiază avantajele acestei abordări.

Pe baza experiențelor anterioare obținute în urma recomandărilor, unui utilizator nou i se poate asocia rapid un grup de persoane cu interese similare.

Cuvinte cheie—Aplicații software, Aplicații educaționale, Inteligență artificială

I. INTRODUCERE

Proiectul constă într-o aplicație pentru interconectarea oamenilor, în funcție de preferințele lor, utilizând diverși algoritmi. Unii dintre algoritmii utilizați sunt mai simpli, iar alții folosesc inteligență artificială. Se dorește și recomandarea utilizatorilor în funcție de alte filtre, precum distanța geografică. Scopul proiectului este de a uni oamenii mai ușor în scop educațional în funcție de preferințele legate de tehnologii sau concepte teoretice, însă proiectul ar putea fi folosit și în alte arii decât cele educaționale.

Există o mulțime de studii pe tema recomandărilor, unii dintre autori precum Xin Wei [1], Zeinab Shahbazi [2], Qian Zhang [3], gândindu-se deja la sisteme de recomandări bazate pe metode ale inteligenței artificiale și pe psihologia educației, folosind resurse precum imagini, texte, clipuri și link-uri. În funcție de cum se interacționează pe site-uri cu resursele menționate, studenții au fost clasificați în: studenți activi, studenți cu potențial și studenți inactivi. Toate aceste trei grupuri au fost împărțite pe baza modului în care interacționau cu paginile respective, urmând ca fiecare dintre ei să primească resurse educative potrivite [1]. Există foarte multe domenii de aplicare pentru algoritmii de recomandare și pot fi folosiți nu doar în scopuri educative, ci și divertisment, precum filmele [3].

În urma unui studiu de piață, s-a constatat că, în ciuda faptului că există aplicații care încearcă să recomande persoane potrivite după anumite criterii, acestea nu realizează în totalitate obiectivul dorit. Multe dintre aplicațiile găsite pe Play Store includ opțiunea de alegere de preferințe (de exemplu, mâncare, hobby-uri, muzică), însă filtrele de utilizatori sunt inexistente, neputând primi utilizatori similari unei ținte, cel puțin din punctul de vedere al preferințelor [4]. Singura aplicație care s-a constatat că ar face o parte dintre aceste funcționalități de recomandare este Meetup, dar acolo au loc recomandări de evenimente, nu de persoane. Panion ar fi fost un exemplu bun, dar în prezent nu mai funcționează publicului larg.

Pe baza acestui studiu de piață, s-a ajuns la concluzia că o astfel de aplicație pentru recomandarea persoanelor în funcție de preferințele lor ar fi necesară. Obiectivul principal este acela de a găsi și de a filtra cât mai mulți utilizatori potriviți cu filtrele aplicate, dar și de a avea un produs funcțional, sigur și securizat, care să fie ușor de înțeles, plăcut și intuitiv de folosit.

O posibilitate de a rezolva această problemă este dată de utilizarea microserviciilor și inteligenței artificiale. Arhitecturile bazate pe microservicii oferă scalabilitate, rulare independentă, performanțe adiționale, avantajul de a fi ușor de menținut, costuri reduse etc. [5], în timp ce inteligența artificială oferă posibilități multiple de modelare și antrenare a datelor după anumite seturi de date. Câțiva dintre acești algoritmi ce pot fi folosiți în acest sens sunt: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Singular Value Decomposition (SVD), Random Forest, filtrul colaborativ, filtrul bazat pe conținut, abordări hibride [4, 5].

Tocmai de aceea, în încercarea de a rezolva problema propusă, tehnologiile și conceptele teoretice utilizate în această aplicație sunt:

- Backend: Spring, Spring Security, JWTs, REST, Kotlin, MariaDB, MongoDB, criptare, decriptare;
- Frontend: Thymeleaf, JavaScript, CSS, HTML;
- AI: KNN, Python, Flask, sklearn, pandas;
- Aplicații software: IntelliJ IDEA, PyCharm, Visual Studio Code, Postman, Compass, DBeaver.

În secțiunea următoare se discută despre arhitectura aplicației, detalii importante cu privire la tehnologiile folosite, modul în care a fost implementată aplicația și motivațiile din spatele acestora. În capitolul al treilea se discută despre experimentele realizate, rolul lor și însemnătatea lor, dar tot aici au loc interpretarea rezultatelor. La sfârșit sunt menționate câteva concluzii cu privire la experimentele realizate și rezultatele obținute. La final este prezentată bibliografia.

II. Arhitectură și implementare

A. *Arhitectura aplicației*

O posibilitate de a implementa soluția propusă, așa cum s-a discutat în capitolul anterior, este folosirea microserviciilor împreună cu inteligența artificială. Componentele principale identificate sunt: *Aplicația Web*, *Gateway*-ul, serviciul pentru gestionarea identităților *IDM (Identity Management)*, cel pentru controlul profilelor utilizatorilor (*Profile Service*) și unul pentru algoritmii de inteligență artificială (*Algorithms Service*). Alte servicii utile aplicației, dar neimplementate încă, sunt cele pentru identificarea locației, comunicarea între utilizatori, și pentru suport-ul utilizatorului (în cazul în care acesta are nevoie de ajutor, să poată depune cereri).

S-au ales microserviciile pentru că oferă avantajul îmbunătățirii procesului de dezvoltare, prin împărțirea sistemului în componente mai mici, mai ușor de gestionat (Fig. 1). Microserviciile permit eșuarea independentă, o modularizare mai bună, scalabilitatea, alocarea mai eficientă de resurse, conducând astfel la o aplicație software îmbunătățită.

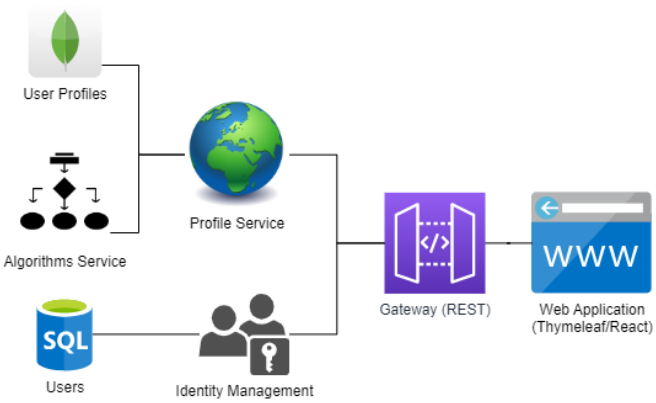

Fig. 1. Diagrama de servicii

Codul a fost despărțit pe module și clase, în unele situații fiind necesare șabloanele de proiectare. Șablonul de proiectare *strategie* a fost folosit pentru a decide modul în care utilizatorii vor fi recomandați unei ținte. În prezent, sunt patru strategii de a decide acest aspect: căutări directe în lista utilizatorilor prin aplicarea paradigmei funcționale, celelalte strategii bazându-se pe algoritmul KNN și metrici ale acestuia.

Componenta *Aplicație Web* reprezintă punctul de acces de la nivelul clientului. Rolul componentei este de a oferi utilizatorului o experiență pozitivă pe site. S-a folosit Thymeleaf pentru a construi paginile web, precum și JavaScript pentru a realiza cereri către server, CSS pentru stilizarea paginilor și HTML pentru scheletul acestora.

Gateway-ul este componenta ce leagă Backend-ul aplicației de Frontend, ce poate fi asemănat cu șablonul de proiectare *fațadă*. Rolul lui este de a adăuga un strat ce conține componente de securitate, dar și de a redirecta cererile clienților către serverele potrivite. La nivelul acestui *Gateway* au loc verificările tokenilor criptați JWT, din care se extrag claim-urile de interes și se caută autoritățile de interes.

Serviciul IDM se ocupă cu identificarea utilizatorilor și autorităților acestora, dar și de crearea conturilor unor noi utilizatori. Scopul serviciului este de a oferi diferitor utilizatori accesul la unele rute pe care alții nu îl au. În spate există baza de date MariaDB folosită pentru a stoca aceste detalii legate de conturile utilizatorilor, dar și legate de tokenii emiși și starea lor. Această bază de date s-a ales pentru că este Open-Source, dar și pentru că oferă consistență datelor.

Serviciul Profile funcționează împreună cu *IDM*, dar este destinat utilizatorilor care deja sunt autentificați, sau care își crează cont. Un profil de utilizator este posibil ca, în viitor, să aibă altă structură și alte câmpuri. De aceea, acesta are în spate o bază de date MongoDB utilă pentru a gestiona fragmente de date. Inițial, dorința era de a separa detaliile din profilul utilizatorilor de cele pentru autentificare. În prezent, serviciul se ocupă de profilele utilizatorilor, putând verifica și edita detaliile cu privire la propriul cont, dar și de a primi recomandări în funcție de profilele celorlalți.

Algorithms reprezintă punctul de acces pentru diverși algoritmi de inteligență artificială. Rolul lui este de a recomanda utilizatori în funcție de preferințele celui în cauză, și de preferințele celorlalți. Momentan, acesta se folosește de KNN și metricile cosine, euclidiană și Jaccard.

Toate aceste servicii urmăresc respectarea principiilor SOLID, atât pentru clase, cât și microservicii, obținând astfel o granularitate fină. Fiecare are propriul proiect, server și bază de date unde e cazul, urmărindu-se decuplarea aplicației dezvoltate. Aplicația se folosește de modelul client-server, ceea ce înseamnă că există și un protocol de comunicare. Aici s-a preferat protocolul HTTP, tocmai pentru că procesul de dezvoltare devine facil, dar și din cauză că anumite cereri așteptau răspunsul pentru a finaliza operațiile. Cererile și răspunsurile dintre servicii se obțin prin REST API, fiecare dintre servicii folosindu-se de framework-ul Spring Boot, mai puțin Algorithms, unde s-a folosit Flask.

B. *Modalități de securizare a aplicației*

Pentru siguranța datelor utilizatorilor, s-au utilizat diverse mecanisme de protecție a datelor, precum JWT-urile, păstrate la nivelul clientului prin Cookies, în format criptat (tocmai pentru a se evita schimbarea câmpurilor din interiorul lor), ele fiind mai întâi decriptate la nivelul server-ului, apoi se validează formatul și semnătura acestora, urmând abia apoi să se realizeze validările pe câmpurile lor (de exemplu, există un câmp expiry), autorizarea realizându-se abia la sfârșit, în funcție de autoritatea pe care o deține utilizatorul respectiv (Fig. 2 și Fig. 3).

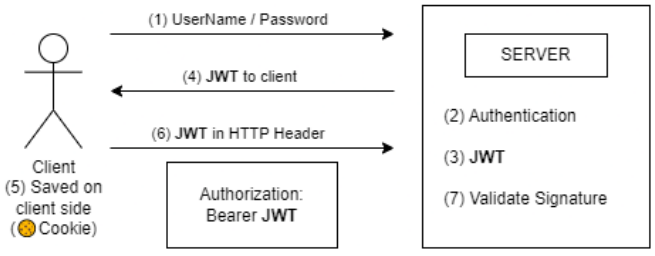

Fig. 2. Interacțiunea dintre client și server folosind JWT

Fig. 3. Validarea semnăturii unui JWT

Pentru a persista parolele, acestea sunt ținute în baza de date criptate cu ajutorul funcției BCrypt, iar în cazul în care un client se autentifică, la verificare, parola introdusă este criptată și ea și comparată cu valoarea din baza de date. S-ar fi putut folosi alte metode pentru autentificare și autorizare, precum OAuth2, SAML, OpenID, dar în scop demonstrativ și pentru a asigura securizarea aplicației, s-au preferat JWT-urile și Spring Security.

Ecosistemul Spring pune la dispoziție mecanisme de securizare a aplicației, precum Spring Security. Acesta poate fi folosit pentru autentificare, dar și autorizare. Acesta se folosește de lanțuri de filtre (filter chains) și dispune de o sintaxă proprie, Spring Expression Language (SpEL), pentru autorizarea într-un mod facil al utilizatorilor în funcție de diverși parametri, precum hasIpAddress, hasAuthority, principal și mulți alții [6].

C. *Setul de date*

Bazele de date sunt folosite pentru a ține cont și memora detalii cu privire la utilizatori și tokenii folosiți. În cazul repornirii aplicației, e de așteptat ca utilizatorii și tokenii lor să rămână la fel ca ultima dată când a fost pornit server-ul. La nivelul serverelor au fost necesare crearea unor entități și tabele. Scopul folosirii acestor entități, alături de Spring JPA și Hibernate, precum și adnotările Spring Boot, este acela de a realiza într-un mod cât mai facil operații precum înregistrări de utilizatori noi. Bazele de date relaționale (SQL) sunt utile când se dorește ca datele să fie consistente, ceea ce se potrivește pentru persistarea datelor de autentificare ale utilizatorilor. Bazele de date nerelaționale (NoSQL) sunt utile că se dorește ca anumite câmpuri să fie opționale.

Pentru a realiza acestea, au fost necesare dependențele Maven, configurarea Spring JPA Hibernate, Spring Datasource, precum și driver-ul MariaDB sau MongoDB, în fișierul de configurare în funcție de metoda de persistare aleasă. Apoi se pot crea interfețele de tip repositories cu adnotarea respectivă (Repository). Aici se vor crea funcții cu interogările SQL/NoSQL, folosind adnotarea Query. În cele din urmă, se realizează entitățile-tabel, cu rolul de a asocia diverse obiecte cu bazele de date. În cazul acestor entități trebuie incluse adnotările utilizate (Entity, Table). Aceste clase includ variabile ce asociază coloanele sau câmpurile din tabele.

Alte adnotări utilizate sunt Id, GeneratedValue, Column, Transient, ManyToMany, JoinTable. În spate, aceste adnotări ușurează procesul de creare al aplicației, dezvoltatorul nemaifiind obligat să includă secvențe de cod în plus.

Pe baza acestora, s-au realizat tabelele, care arată ca în figura alăturată (Fig. 4). Fiecare utilizator are un username și parolă (câmpurile), dar și autorități și tokeni pe care i-a folosit (din celelalte tabele). Utilizatorul este folosit pentru autentificare, autoritățile pentru autorizare. Tokenii parte dintre aceste detalii, dar sunt ținuți criptați aici, cât și la client.

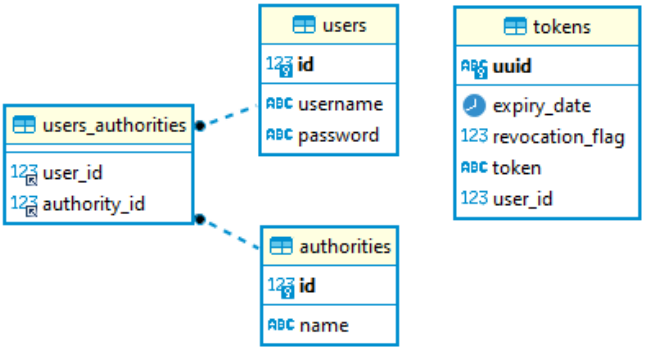

Fig. 4. Diagrama Entity-Relation

Asocierea dintre utilizatorul simplu din *IDM* cu profilul din serviciul *Profile* este făcută prin stocarea câmpului idmId în interiorul profilului, atunci când este creat la înregistrare. Această asociere este importantă, pentru că, atunci când se crează un nou utilizator, este de așteptat ca și profilul lui să fie creat. Similar, atunci când utilizatorul vrea să își șteargă contul, trebuie să fie șters și contul lui din cealaltă bază de date. S-a adoptat acest model tocmai pentru a separa preocupările serviciilor.

```
_id: ObjectId('644a158836f4276994d6d770')
idmId: 84
firstName: "Victoria"
lastName: "Campbell"
email: "victoria.campbell@yahoo.com"
▾ preferences: Array
    0: "react"
    1: "javascript"
    2: "css"
    3: "html"
    4: "node.js"
_class: "com.project.profile.data.entities.UserProfile"
```

Fig. 5. Structura profilului utilizatorului

D. *Interfață grafică*

Unul dintre framework-urile utilizate pentru interfața grafică este Thymeleaf. Motivația este simplă: comunică foarte bine cu ecosistemul Spring Boot și oferă facilități pentru metode HTTP precum POST, la nivelul server-ului:

```
@PostMapping("/register")
fun register(@ModelAttribute userModel: UserDTO):
ModelAndView {

if(!userValidatorService.isUserRegisterValid(userM
odel))
        return ModelAndView("register-fail")
    return ModelAndView("register-success")
}
```

Iar la niviel de view, acesta ar putea fi un mod de a-l folosi:

```
<form th:action="@{/register}" th:object="${user}"
method="post">
<input id="register_username" type="text"
placeholder="e.g. SuperUser23" th:field="*{username}"/>
```

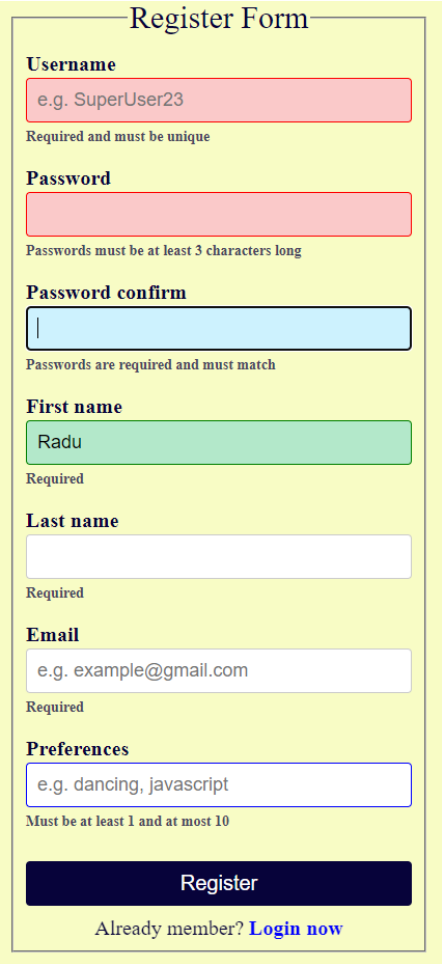

Fig. 6. Pagina de înregistrare

III. Experimente și rezultate

Algoritmul KNN este deseori folosit în gruparea elementelor în funcție de anumite clase. Algoritmul se potrivește pentru sistemele de tip recomandări, fiind util aplicației curente. Pentru KNN s-au utilizat metricile: Jaccard, euclidiană și cosine.

Prin aceste experimente se urmăresc: compararea metricilor cosine, euclidiană și Jaccard; determinarea valorii k cea mai potrivită pentru KNN (3, 5, 7, 11, lungimea datelor de antrenare); dar și urmărirea potrivirii utilizatorilor în funcție de o țintă cu preferințe oarecare, încât modelul să fie unul nu doar funcțional, ci și corect.

A. *Preprocesarea datelor*

Pentru a testa modelul KNN și metricile menționate, precum și efectivitatea acestora, au fost necesare următoarele preprocesări:

- Transformarea preferințelor utilizatorilor în date numerice (preferințele, inițial reprezintă string-uri; în funcție de toate preferințele utilizatorilor, s-a realizat operația de mapare a acestora în numere întregi pozitive, de exemplu, ["c++", "java", "html"] se transformă în [0, 1, 2], reprezentând indexul mapării);
- Pe baza preferințelor, seturile de date se vor împărți în două subseturi: de antrenare (70%), și de testare (30%), astfel încât toate preferințele să se regăsească în datele reunite;
- Metrica dorită a fi utilizată;
- Valoarea lui k pentru a utiliza modelul KNN.

Pentru a urmări mai ușor arhtiectura aplicației în ceea ce privește algoritmul KNN, s-a realizat următoarea diagramă, începând cu utilizatorul și preferințele lui, trecând prin lanțul de servicii în care se realizează procesările datelor, iar la sfârșit obținându-se utilizatorii potriviți preferențial țintei:

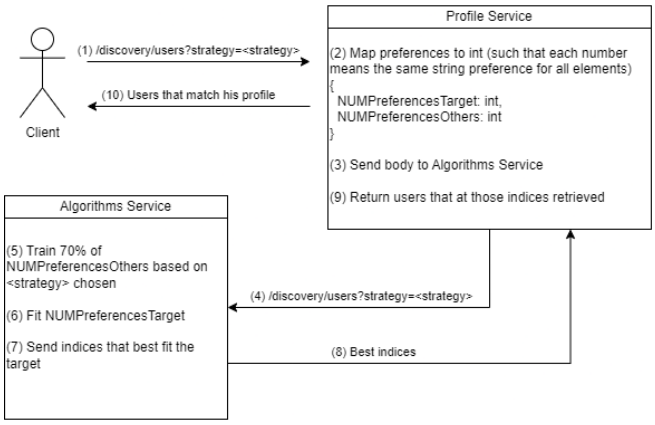

Fig. 7. Interacțiunea dintre client și servere folosind algoritmul KNN

B. *Modelul KNN*

În urma antrenării modelului, este așteptată obținerea de utilizatori cu preferințe similare țintei, ținând cont de datele de intrare alese la început.

Pentru compararea celor trei metrici se utilizează același set de date de antrenare respectiv testare în calcularea acurateței. Atunci trebuie ales același k, iar preferințele de antrenare și potrivire trebuie să fie aceleași. Similar se procedează și când se urmărește compararea valorilor lui k.

În urma rulării, se vor obține vecinii cei mai apropiați utilizatorului cu preferințele alese. Intrând mai adânc în detalii, la nivelul constructorului se binarizează preferințele și se ține cont de aceste detalii, astfel încât să poată fi comparate corect folosind mai multe metrici. Atunci când se apelează funcția train(k), se antrenează modelul folosind metrica aleasă și preferințele binarizate. Binarizarea vectorilor are rolul de a simplifica din calcule și îmbunătăți performanța. Prin binarizare, valorile continue sunt transformate în valori binare de 0 și 1. Astfel, indicii utilizatorilor cu preferințele apropiate se obțin prin binarizarea vectorului de preferințe nou inclus (kneighbors([target_user_binary])). Similar se pot obține și distanțele utilizatorilor. Dar ce este de interes în final este vectorul de preferințe similare țintei, care se poate obține prelucrând aceste date menționate.

Similaritatea cosine este folosită în sisteme de recomandări. Matematic, ea reprezintă cosinusul unghiului dintre doi vectori și e folosită drept metrică de evaluare a distanței dintre doua puncte din plan. Cu cât distanța dintre cele două puncte este mai mare, cu atât similaritatea dintre puncte este mai mică [7]. Formula similarității cosinusului se regăsește mai jos, cu A, B vectorii comparați:

$$cosine\ similarity = Sc(A,B) = \frac{A \cdot B}{||A||\,||B||} \quad (1)\ [8]$$

$$cosine\ similarity = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2)$$

Distanța euclidiană reprezintă lungimea unui segment dintre două puncte. Cu cât distanța dintre cele două puncte este mai mare, cu atât similaritatea este mai mică [9]. Mai jos se află formula de calcul a acestei distanțe, A și B fiind cei doi vectori în plan.

$$euclidian\ distance = d(A,B) = ||A - B||_2 \quad (3)$$

$$euclidian\ distance = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2} \quad (4)$$

Similaritatea Jaccard este o metrică de proximitate pentru a calcula asemănarea dintre două obiecte, cum ar fi două documente text. Matematic, se definește drept cardinalul intersecțiilor preferințelor supra cardinalul reuniunii celor două mulțimi. Cu cât rezultatul fracției tinde către o valoare supraunitară, cu atât cele două seturi de date sunt mai similare [10]. Formulele de calcul pentru metrica Jaccard sunt acestea:

$$Jaccard\ similarity = J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (5)$$

$$Jaccard\ similarity = J(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (6)$$

C. Rezultate experimentale

În tabelul de mai jos se găsesc trei vectori de preferințe (în format continuu, după maparea lor în formatul acesta), alături vecinii cei mai apropiați, obținuți pentru fiecare metrică în parte. De exemplu, primul rând din tabel are următorul sens: utilizatorul cu preferințele ["c++", "javascript", "html"] va fi convertit în [5, 19, 32], 5 însemnând "c++", 19 fiind "javascript", iar 32 fiind "html". Aceasta este etapa de mapare a preferințelor. În urma rulării algoritmului pentru fiecare dintre metrici, pe baza preferințelor numerice, s-au obținut vectorii de preferințe numerice, ca în tabel. Pentru cosine, în cazul acesta, vectorul de preferințe se poate transla astfel: [["javascript", "kotlin", "c#", "web", "c++"], ["mongodb", "javascript", "web", "c++", "python", "kubernetes"], ["mongodb", "javascript", "web", "c++", "python", "machine learning"], ["go", "javascript", "web", "c++", "python", "ai"], ["go", "mongodb", "javascript", "web", "c++", "python", "kubernetes"]].

TABLE I. EVALUAREA METRICILOR, CU K=3

Preferință țintă	**Metricile evaluate**		
	Cosine	***Euclidiană***	***Jaccard***
[5, 19, 32]	[[19, 99, 81, 4, 5], [58, 19, 4, 5, 6, 66], [142, 19, 4, 5, 6, 14]]	[[19, 99, 4], [19, 145, 146], [21]]	[[19, 99, 81, 4, 5], [58, 19, 4, 5, 6, 66], [142, 19, 4, 5, 6, 14]]
[16]	[[16], [13, 5, 14, 15, 6, 4, 16, 17], [18, 19, 6, 4, 5, 14, 20, 21, 17, 16]]	[[16], [113], [21]]	[[16], [13, 5, 14, 15, 6, 4, 16, 17], [18, 19, 6, 4, 5, 14, 20, 21, 17, 16]]
[141, 5, 19, 55, 13]	[[142, 19, 4, 5, 6, 15], [142, 19, 4, 5, 6, 14], [142, 19, 4, 5, 6, 66, 67]]	[[18], [8], [19, 99, 4]]	[[142, 19, 4, 5, 6, 14], [142, 19, 4, 5, 6, 15], [142, 19, 4, 5, 6, 66, 67]]

Similar primului tabel, a fost întocmit încă unul cu valoarea k=5, tocmai pentru a se putea observa existența diferențelor nu doar între metrici, ci și a valorilor lui k date (Tabelul II).

TABLE II. EVALUAREA METRICILOR, CU K=5

Preferință țintă	**Metricile evaluate**		
	Cosine	***Euclidiană***	***Jaccard***
[5, 19, 32]	[[19, 99, 81, 4, 5], [58, 19, 4, 5, 6, 14], [58, 19, 4, 5, 6, 66], [142, 19, 4, 5, 6, 14], [58, 19, 4, 5, 6, 15]]	[[18], [19, 145, 146], [8], [19, 29, 212], [19, 29, 212]]	[[19, 99, 81, 4, 5], [142, 19, 4, 5, 6, 15], [58, 19, 4, 5, 6, 15], [142, 19, 4, 5, 6, 14], [58, 19, 4, 5, 6, 14]]
[16]	[[16], [13, 5, 14, 15, 6, 4, 16, 17], [18, 19, 6, 4, 5, 14, 20, 21, 17, 16], [116, 117, 80, 118, 119], [159, 160]]	[[16], [113], [18], [21], [161, 162]]	[[16], [13, 5, 14, 15, 6, 4, 16, 17], [18, 19, 6, 4, 5, 14, 20, 21, 17, 16], [58, 19, 8, 31, 29, 50, 28], [205, 206, 207, 208]]
[141, 5, 19, 55, 13]	[[142, 19, 4, 5, 6, 15], [142, 19, 4, 5, 6, 14], [142, 19, 4, 5, 6, 66, 67], [13, 5, 14, 15, 6, 4, 16, 17], [19, 8]]	[[19, 8], [8], [215, 19, 209], [19, 29, 212], [19, 29, 212]]	[[142, 19, 4, 5, 6, 15], [142, 19, 4, 5, 6, 14], [142, 19, 4, 5, 6, 66, 67], [13, 5, 14, 15, 6, 4, 16, 17], [19, 8]]

Pe baza celor două tabele se poate constata că cele mai bune metrici dintre cele trei testate pentru potrivirea utilizatorilor sunt cosine și Jaccard. Motivația este simplă: în metrica euclidiană, fiind necesară conversia în vectori binarizați, multe dintre preferințe vor fi setate pe valoarea 0 și doar foarte puține vor avea 1. Matematic vorbind, se caută preferințe aflate la distanțe euclidiene cât mai mici de preferința țintă, dar asta nu garantează că vor fi preferințe dintre cele pe care le deține utilizatorul. Totuși, din punct de vedere al distanței euclidiene, acei utilizatori sunt similari.

S-au rulat 100 de teste prin care se dorește compararea rezultatelor obținute dintre cele 3 metrici, dar și dintre valori diferite ale lui k. Datele colectate sunt legate de numărul de utilizatori ale căror preferințe se încadrează într-o anumită categorie față de preferințele utilizatorului țintă. În total, sunt 5 categorii în care se pot încadra preferințele (foarte în comun, în comun, în moderație, puțin în comun, foarte puțin în comun). Inițial, toate aceste grupuri pornesc de la 0 indivizi. Iterând prin vectorul cu vectori de preferințe obținuți pe baza algoritmului KNN, în funcție de valoarea elementului curent (preferința vecină), se compară gradul de similitudine cu preferința țintă și adăugă profilul în categoria corectă. Fiecare grup crește în număr în funcție de gradele de similitudine. Acestea au fost încadrate astfel:

$$very\ low = \begin{cases} very\ low + 1, & 0 \le similitude < 0.1 \\ very\ low, & otherwise \end{cases}$$

$$low = \begin{cases} low + 1, & 0.1 \le similitude < 0.5 \\ low, & otherwise \end{cases}$$

$$moderate = \begin{cases} moderate + 1, & 0.5 \le similitude < 0.7 \\ moderate, & otherwise \end{cases}$$

$$high = \begin{cases} high + 1, & 0.7 \le similitude < 0.8 \\ high, & otherwise \end{cases}$$

$$very\ high = \begin{cases} very\ high + 1, & 0.8 \le similitude < 1.0 \\ very\ high, & otherwise \end{cases}$$

Similitudinea reprezintă gradul de asemănare dintre ținta curentă și vecinul ei la acea iterație. Dacă e mai mare, înseamnă că cei doi utilizatori sunt mai potriviți preferențial. Pentru acest algoritm, similitudinea este necesară pentru a putea încadra utilizatori în aceste categorii.

Se rulează testele pentru fiecare dintre date, obținând rezultate pentru k ori câte metrici sunt, conform tabelelor următoare (Tabelul III, Tabelul IV, Tabelul V):

TABLE III. REZULTATE COSINE, PENTRU VALORI DIVERSE ALE PARAMETRULUI K

k	100 teste, cosine, pentru fiecare k				
	Very High	***High***	***Moderate***	***Low***	***Very Low***
3	1816	1236	3930	5511	1907
5	1863	1984	5554	10116	4483
7	1867	2234	7033	14856	7610
11	1867	2252	8620	25452	14609
110	1867	2252	9086	89176	425619

În urma rulării aceluiași test și încadrări similitudinale, au fost realizate statisticile de mai sus. Important din acesta este valoarea coloanei *Very High* și a coloanei *High*, tocmai pentru că acestea sunt valorile ce indică cei mai potriviți utilizatori. Așadar, acuratețea va reprezenta suma dintre valorile procentuale ale acestor coloane raportate la totalul de comparații efectuate.

Așa cum se observă în tabel, valorile acestor coloane rămân aproximativ identice (se stabilizează) de la anumite valori ale lui k. Este foarte important și numărul de utilizatori ce sunt luați în considerare. Aplicația, având doar 159 de utilizatori pentru care să ruleze algoritmul, este evident că valori mai mici ale lui k ar fi mai potrivite. Este posibil ca pentru o aplicație cu mulți utilizatori să aibă nevoie de valori mai mari ale lui k.

TABLE IV. REZULTATE EUCLIDIANĂ, PENTRU VALORI DIVERSE ALE PARAMETRULUI K

k	100 teste, euclidiană, pentru fiecare k				
	Very High	***High***	***Moderate***	***Low***	***Very Low***
3	380	609	7867	5544	0
5	380	609	12235	10776	0
7	380	609	15472	17139	0
11	380	609	20105	31706	0
110	380	609	30917	446624	49470

Similar tabelului anterior, a fost realizată o astfel de observație și pentru metrica euclidiană. Spre deosebire de metrica de tip cosine, acesta înregistrează valori ale acurateței mai slabe. De notat ar fi și faptul că, doar de la anumite valori pentru k algoritmul obține valori pentru *Very Low* fără niciun individ, fapt ce indică o posibilă nepotrivire cu această metrică sau o utilizare eronată a acesteia.

TABLE V. REZULTATE JACCARD, PENTRU VALORI DIVERSE ALE PARAMETRULUI K

k	100 teste, Jaccard, pentru fiecare k				
	Very High	***High***	***Moderate***	***Low***	***Very Low***
3	792	900	2000	8685	2023
5	792	947	3111	14346	4804
7	792	951	3704	19874	8279
11	792	951	3814	31005	16238
110	792	951	3814	74353	448090

Iar în cazul metricii Jaccard s-au obținut valori ale acurateței. Similar celorlalte tabele, de la anumite valori ale lui k se înregistrează același număr de utilizatori cu preferințe în comun. Aceasta poate însemna că, acea valoare a lui k este cea mai potrivită algoritmului pentru acel număr de utilizatori.

După acesta, pe baza tabelelor astfel obținute, se pot agrega datele, asociându-se cu metricile respective (k va lua valori de la [3, 5, 7, 11, 110] și se vor aduna pentru metrica respectivă toți utilizatorii). Următoarele două tabele sunt realizate cu scopul de a compara metricile (Tabelul VI), precum și valorile lui k alese (Tabelul VII).

TABLE VI. REZULTATE AGREGATE CU K=[3, 5, 7, 11, 110]

Metrica	100 teste, k=[3, 5, 7, 11, 110]					
	Very High	***High***	***Moderate***	***Low***	***Very Low***	***Total***
Cosine	9280	9958	34223	145111	454228	652800
Euclidiană	1900	3045	86596	511789	49470	652800
Jaccard	3960	4700	16443	148263	479434	652800

Pe baza celor 100 de teste, s-au obținut și următoarele detalii cu privire la modelul KNN și valorile k ale acestuia, adunându-se valorile fiecărei metrici pentru fiecare k.

TABLE VII. REZULTATE AGREGATE PENTRU FIECARE METRICĂ

k	100 teste, pentru fiecare metrică					
	Very High	***High***	***Moderate***	***Low***	***Very Low***	***Total***
3	2988	2745	13797	19740	3930	43200
5	3035	3540	20900	35238	9287	72000
7	3039	3794	26209	51869	15889	100800
11	3039	3812	32539	88163	30847	158400
110	3039	3812	43817	610153	923179	1584000

Scăderea acurateței pe măsură ce crește numărul de preferințe comparate nu indică decât faptul că există o suprapotrivire. Aceasta poate fi rezolvată fie reducând valorile lui k, fie adăugând mai mulți utilizatori (de ordinul sutelor sau chiar miilor) în aplicație care sa aibă preferințe relativ comune celorlalți.

Graficul de mai jos încearcă să rezume cele discutate anterior, punând în evidență atât metricile folosite, valorile lui k utilizate, cât și acuratețea corespunzătoare, pentru fiecare categorie.

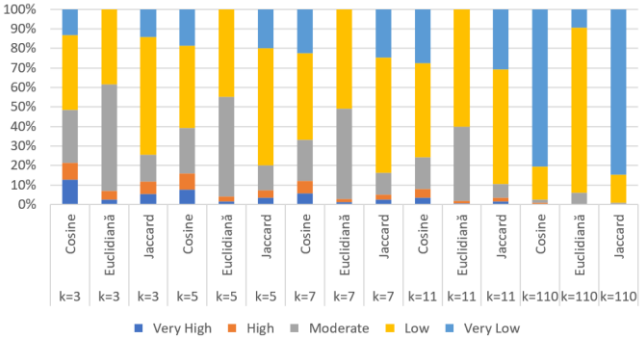

Fig. 8. Compararea valorilor tuturor metricilor și tuturor valorilor lui k, procentual

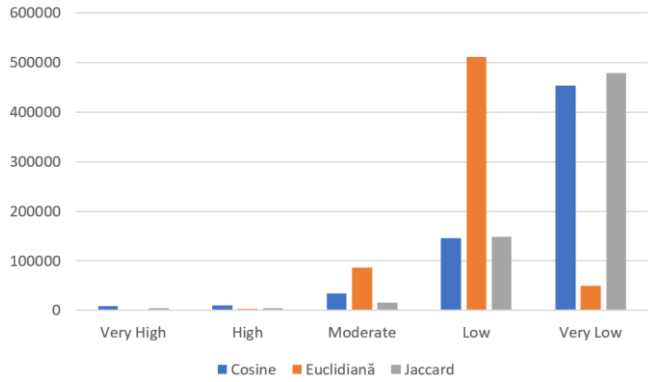

Fig. 9. Compararea celor trei metrici și a încadrării preferințelor lor

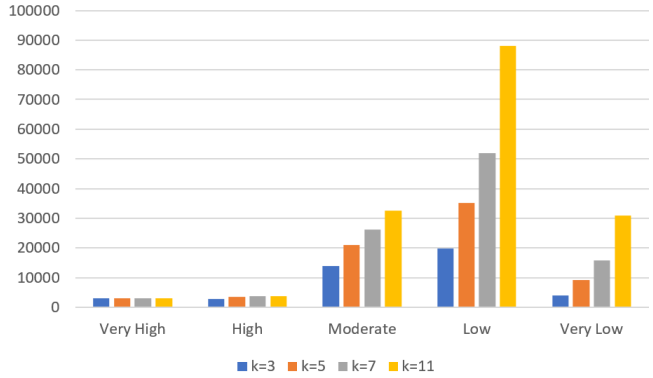

Fig. 10. Compararea valorilor lui k=[3, 5, 7, 11] și a preferințelor lor

În urma acestor experimente, s-a constatat că metrica euclidiană este mai puțin potrivită pentru această problemă, atât din cauză că se recomandă preferințe ce nu sunt neapărat comune cu ținta, cât și din cauză că sunt recomandați relativ puțini oameni cu un număr de preferințe în comun mai mare. Acestea se datorează tocmai distanței matematice euclidiene dintre vectorii binarizați. De aceea, cei mai mulți dintre ei se încadrează la cei cu preferințe moderat potrivite.

IV. Concluzii

Sunt numeroase studii legate de recomandări și inteligență artificială, dar în ciuda acestui fapt, aplicații asemănătoare cu tema propusă nu au fost găsite cu succes, în sensul că multe dintre acestea fie realizează recomandările la nivel minimalist, fie au doar o funcționalitate precară, încât utilizatorii recomandați să fie aleși, de fapt, stohastic.

Există numeroase metode pentru a rezolva problemele din viața de zi cu zi, iar abordările ce utilizează inteligența artificială nu sunt o excepție. În cadrul proiectului, s-au realizat cu succes recomandările pentru utilizatori folosind metricile Jaccard, cosine, și chiar și euclidiană. Există mai multe modalități de a recomanda diverși utilizatori în funcție de preferințele lor, dar în momentul de față, abordarea KNN reușește să rezolve problema aceasta corect și cu succes.

În comparație cu metrica euclidiană, cosine este mult mai optimistă, înregistrând mult mai multe preferințe potrivite cu cele ale țintei. Iar în cazul metricii Jaccard, s-au remarcat tendințe preferențiale mai mari decât în cazul metricii cosine. Spre deosebire de metrica euclidiană, Jaccard reușește să obțină utilizatori potriviți țintei, din punct de vedere al preferințelor.

Aplicația propune o metodă de a uni oameni cu preferințe similare în scop educațional, dar nu limitat numai la acest domeniu, putând fi extins ușor și către altele.

References

[1] Xin Wei, Shiyun Sun, Dan Wu, Liang Zhou, Personalized Online Learning Resource Recommendation Based on Artificial Intelligence and Educational Psychology, Frontiers in Psychology, Sec. Educational Psychology Volume 12, 2021, https://doi.org/10.3389/fpsyg.2021.767837

[2] Zeinab Shahbazi, Yung-Cheol Byun, Agent-Based Recommendation in E-Learning Environment Using Knowledge Discovery and Machine Learning Approaches, Mathematics 2022, 10(7), 1192, https://doi.org/10.3390/math10071192

[3] Qian Zhang, Jie Lu, Yaochu Jin, Artificial intelligence in recommender systems, Complex Intell. Syst. 7, 439–457 (2021), https://doi.org/10.1007/s40747-020-00212-w

[4] William Park, Online dating might not help you to find the one. But the data from dating apps offers some tantalising insights., https://www.bbc.com/future/article/20191112-how-dating-app-algorithms-predict-romantic-desire

[5] Florian Auer, Valentina Lenarduzzi, Michael Felderer, Davide Taibi, From monolithic systems to Microservices: An assessment framework, Information and Software Technology, Volume 137, 2021, https://doi.org/10.1016/j.infsof.2021.106600

[6] Spring docs contributors, Expression-Based Access Control, https://docs.spring.io/spring-security/reference/servlet/authorization/expression-based.html, ultima accesare: aprilie 2023

[7] Darshan M., What is cosine similarity and how is it used in machine learning?, Mystery Vault, https://analyticsindiamag.com/cosine-similarity-in-machine-learning/#:~:text=Cosine%20similarity%20is%20used%20as%20a%20metric%20in,find%20the%20similarity%20of%20texts%20in%20the%20document., ultima accesare: mai 2023

[8] Selva Prabhakaran, Cosine Similarity – Understanding the math and how it works (with python codes), https://www.machinelearningplus.com/nlp/cosine-similarity/?utm_content=cmp-true, ultima accesare: mai 2023

[9] Nikos kalikis, Text Similarity: Euclidian Distance VS Cosine Similarity !!!, https://nikoskalikis.medium.com/text-similarity-euclidian-distance-vs-cosine-similarity-3a1167f686a, ultima accesare: mai 2023

[10] Fatih Karabiber, Jaccard Similarity, https://www.learndatasci.com/glossary/jaccard-similarity/