UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ÎN LIMBA ROMÂNĂ

LUCRARE DE LICENŢĂ

Conectarea persoanelor bazată pe interese comune folosind sisteme de recomandare

Conducător științific
Asist. Drd. Coste Claudia-Ioana

Absolvent
Bucilă Mihai-Cristian

2025

ABSTRACT

Abstract: un rezumat în limba engleză cu prezentarea, pe scurt, a conținutului pe capitole, punând accent pe contribuțiile proprii și originalitate

Cuprins

Capitolul 1

Introducere

Hobby-ul este o activitate plăcută pe care o persoană o face regulat, cu dorința de a se relaxa. Acesta nu trebuie confundat cu profesia, deoarece nu urmărește vreun obiectiv profesional sau financiar.

1.1 Motivație

1.2 Obiective

1.3 Structura lucrării

Această lucrare este structurată în 5 capitole care prezintă atât aspecte teoretice, cât și aspecte practice ale subiectului abordat. Capitolul 1 prezintă o introducere a temei propuse, cu atenția îndreptată spre obiectivele lucrării și motivația acesteia. În capitol 2 se oferă detalii despre câteva soluții existente ce rezolvă problema dată. Capitolul 3 este dedicat bazei teoretice. Soluția software se descrie în capitolul 4. În capitolul 5 sunt prezentate concluziile și perspectivele de viitor.

Capitolul 2

Soluții similare

[?]

Capitolul 3

Fundamente teoretice în sisteme de recomandare

3.1 Filtrare bazată pe conținut

Filtrarea bazată pe conținut este o tehnică de recomandare care analizează atributele unui element sau ale unei persoane pentru a genera sugestii. În general, această metodă este utilizată cu scopul de a oferi recomandări personalizate în funcție de profilul utilizatorilor. Sistemul identifică similarități între profiluri și sugerează elemente sau persoane care corespund cel mai bine preferințelor utilizatorului. [1]

3.1.1

3.2 Filtrare colaborativă

Capitolul 4

Aplicație software

4.1 Arhitectura aplicației

Aplicația este alcătuită din 3 componente: interfața utilizator (frontend), partea de server (backend) și sistemul de recomandare. Partea de frontend este dezvoltată în Ionic React, tehnologie ce îmbină librăria React, scrisă în limbajul de programare JavaScript, utilizată pentru a construi interfețe utilizator și Ionic, un instrument ce ajută la dezvoltarea aplicațiilor cross-platform. Folosind Ionic, aplicația poate fi instalată pe mai multe platforme — web, mobile (Android, iOS), desktop — utilizând același cod sursă. Această abordare eficientizează dezvoltarea și întreținerea aplicațiilor, permițând o experiență de utilizare uniformă, indiferent de dispozitiv folosit.

4.1.1 Frontend

Interfața utilizator a aplicației este compusă din rute, pagini și componente. Paginile corespund unor ecrane complete din aplicație, iar componentele sunt părți reutilizabile ale aplicației, precum un formular sau buton personalizat. Componentele primesc proprietăți și returnează elemente de interfață utilizator. Rutele definesc ce pagină trebuie afișată atunci când utilizatorul accesează o anumită adresă (URL). De exemplu, ruta `/login` afișează pagina de autentificare, ruta `/register` afișează pagina de înregistrare ș.a.m.d.

Frontend-ul comunică cu serverul prin apeluri HTTP, datele fiind transmise în formatul JSON. În cadrul aplicației a fost creat fișierul `api.ts` care grupează toate cererile care sunt trimise către backend. Acest serviciu oferă funcții precum `register`, `login` și `saveHobbies`. Aplicația folosește biblioteca Axios, o soluție populară și eficientă pentru manipularea cererilor HTTP.

4.1.2 Backend

Partea de server este scrisă în Java utilizând Spring Boot, un instrument care ajută la simplificarea procesului de dezvoltare al aplicațiilor. Acest framework permite crearea rapidă de API-uri REST, o integrare facilă cu baza de date și o organizare logică a codului în componente. Alegerea acestei tehnologii a fost motivată de ecosistemul extins pe care îl oferă, configurarea redusă și structura modulară.

Proiectul este organizat conform principiilor arhitecturii stratificate, adaptate pentru o aplicație REST. Așadar, codul sursă este împărțit în mai multe directoare (pachete) care separă responsabilitățile logice:

- controller: include clase care gestionează cererile HTTP venite de la frontend. Acestea definesc rutele API și interacționează direct cu serviciile.
- service: conține implementările operațiilor specifice fiecărei funcționalități (înregistrare utilizator, adăugare hobby etc.). Acest strat realizează legătura între controller și repository.
- repository: definește interfețe care extind `JpaRepository`. Acestea se concentrează asupra operațiilor CRUD (Create, Read, Update, Delete) cu efect direct asupra bazei de date.
- domain: conține entitățile din domeniul aplicației. Acestea sunt adnotate cu specificații precum `@Entity`, `@Table` etc. pentru a le mapa în tabele din baza de date.

Pe lângă aceste pachete clasice, proiectul include și alte componente utile:

- dto (Data Transfer Object): conține clase care precizează formatul datelor transmise între client și server.
- mapper: cuprinde clase cu metode statice folosite la conversia obiectelor de tip domeniu în obiecte de tip DTO și vice-versa.
- exception: definește clase care se ocupă de tratarea erorilor. Acestea extind clasa `Exception` cu scopul de a crea excepții personalizate.
- security: gestionează autentificarea și autorizarea. Oferă implementare pentru JWT (Json Web Token).

Model de date

Entitățile de bază ale aplicației sunt User, Hobby și HobbyType. Fiecare dintre acestea sunt reprezentate printr-o clasă Java, mapată la o tabelă din baza de date. Sunt definite clar relațiile între obiecte: un hobby are mai multe tipuri, un utilizator are mai multe hobby-uri etc.

Baza de date

Conectarea la baza de date se face cu ajutorul fișierului de configurare `application.properties`, unde sunt specificate datele de acces precum URL, utilizator și parolă. Baza de date folosită este PostgreSQL, consacrată pentru fiabilitate și performanță. Pentru maparea obiectelor s-a utilizat Java Persistence API împreună cu framework-ul Hibernate, permițând manipularea entităților Java ca tabele din baza de date.

Comunicare cu sistemul de recomandare

Securitate

4.1.3 Sistemul de recomandare

4.2 Funcționalități

4.2.1 Înregistarea utilizatorului

Această funcționalitate permite unui user nou să își creeze cont în aplicație. Pagina conține un formular cu câmpurile: nume, prenume, email sau parolă. La acționarea butonului Register, aplicația validează datele scrise de utilizator, iar în cazul în care nu există erori, trimite o cerere HTTP de tip POST către endpoint-ul `/auth/register` expus de backend. În caz contrar, programul oferă user-ului feedback imediat prin mesaje de eroare sugestive. Pe backend, datele se revalidează pentru a asigura securitatea și integritatea acestora. Contul utilizatorul se salvează în baza de date doar dacă nu există deja unul cu aceeași adresă de email, altfel se aruncă o excepție corespunzătoare. Serverul trimite răspuns, după caz, fie datele user-ului creat, fie un mesaj de eroare.

4.2.2 Autentificarea utilizatorul

Pentru a accesa restul funcționalităților, utilizatorul trebuie să se autentifice. Pagina dedicată logării cuprinde un formular cu două câmpuri: adresa de email și parolă. În urma apăsării butonului Login, aplicația validează local datele introduse. În lipsa erorilor, se inițiază o cerere HTTP POST către endpoint-ul `/auth/login`. Pe partea de server, datele se verifică din nou, iar dacă acestea sunt corecte și corespund unui cont existent, se trimite un răspuns ce conține datele utilizatorului logat și un token JWT generat. În cazul unor date invalide (parolă incorectă, email inexistent), sistemul oferă un mesaj de eroare specific. După autentificare, token-ul JWT este stocat, iar utilizatorul este redirecționat către pagina principală.

4.2.3 Alegerea hobby-urilor

După ce utilizatorul este autentificat, aplicația verifică dacă acesta are hobby-uri. Dacă nu sunt identificate hobby-uri pentru user-ul logat, programul prezintă ecranul de alegere a hobby-urilor. Acesta conține un buton pentru salvare, o bară de căutare și listă de componente ce afișează numele și tipul fiecărui hobby, alături de un checkbox. Lista hobby-urilor este primită de la server prin intermediul unui apel HTTP de tip GET către endpoint-ul `/hobbies`. Utilizatorul poate filtra lista după numele hobby-urilor folosind bara de căutare. După ce user-ul selectează hobby-urile dorite și apasă butonul Save, se trimite o cerere HTTP de tip POST pentru a salva hobby-urile selectate.

4.3 Testare

4.4 Manual de utilizare

La prima interacțiune cu aplicația, utilizatorul are acces la două ecrane, la cel de logare și la cel de înregistrare. În funcție de calitatea user-ului (nou sau înregistrat) poate alege să se înregistreze sau să se autentifice.

Pentru a se loga, utilizatorul trebuie să introducă email-ul și parola cu care s-a înregistrat în prealabil. După ce a introdus aceste câmpuri

În cazul în care utilizatorul logat nu are adăugate hobby-uri, un ecran dedicat acestei acțiuni va fi afișat imediat după logare. Aici user-ul poate alege din lista de hobby-uri existente, prin intermediul unui Checkbox. De asemenea, hobby-urile pot fi filtrate după nume, folosind bara de căutare. După selectare, aceastea pot fi salvate prin apăsarea butonului Save.

Capitolul 5

Concluzii

Concluzii ...

Bibliografie

[1] P. Kumar and R. S. Thakur, "Recommendation system techniques and related issues: a survey," *International Journal of Information Technology*, vol. 10, pp. 495–501, 2018.