

UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICA

LUCRARE DE LICENȚĂ

SISTEM PENTRU RECOMANDARE DE FILME

Conducător științific
[Grad, titlu și nume coordonator]

Absolvent
RADU DRAGOS STEFAN

2021

ABSTRACT

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

Cuprins

1	Capitolul 1: Algoritmul de recomandare	1
1.1	Prelucrarea datelor	1
1.2	Sacul de cuvinte si transformarea cuvintelor in vectori	1
1.3	Similaritatea cosinus	3
1.4	Calcularea scorului final	3
2	Modelare si proiectare	4
2.1	Cazurile utilizatorului si diagrame	4
2.2	Utilizarea aplicatiei	6
3	Capitolul 3: Tehnologiile de pe Back-end	8
3.1	Node Js	8
3.2	Javascript	9
3.3	MySQL	9
3.4	Python	10
3.5	Flask	11
4	Capitolul 4: Tehnologiile de pe Front-end	12
4.1	React	12
4.2	TypeScript	13
5	Specificari Back-end	15
5.1	Baza de date	15
5.2	Server Node Js	16
5.2.1	Inregistrare	17
5.2.2	Autentificare	18
5.2.3	Lista filmelor care contin un subtitlu	19

Capitolul 1

Capitolul 1: Algoritmul de recomandare

1.1 Prelucrarea datelor

Un prim pas in algoritmul de recomandare de filme este de a prelucra datele pentru a obtine cele mai bune valori in calculul final al scorului. Dupa ce avem toate datele un prim pas este sa inlocuim datele nule cu caracterul spatiu pentru a nu conta in calculul final. Apoi un pas destul de important in procesarea datelor este sa eliminam cuvintele de legatura ca de exemplu: 'an', 'be', 'some', etc. Cel mai important camp din care trebuie scoase aceste cuvinte este descrierea deoarece la final se vor crea vectori din cuvintele pe care le avem in datele noastre, iar cuvintele de legatura nu sunt deloc importante in calculul final noi avand nevoie doar de cuvintele esentiale. Ultimul lucru care trebuie facut in procesarea datelor este de a scoate separatorii de cuvinte, in datele noastre fiind doar virgula, din colonele: gen, actori, scriitori, directori si descriere.

1.2 Sacul de cuvinte si transformarea cuvintelor in vectori

Urmatorul pas in algoritmul de recomandare de filme este de a crea cea ce se numeste "Bag of words". Modelul "Bag of words" este o reprezentare pentru procesarea limbajului natural și regăsirea informațiilor care simplifică lucrurile (IR). Un text (cum ar fi o propoziție sau un document) este reprezentat în această paradigmă ca un sac (multiset) al cuvintelor sale, care ignoră sintaxa și chiar ordinea cuvintelor, păstrând în același timp multiplicitatea. Viziunea computerizată a utilizat, de asemenea, conceptul de sac de cuvinte. Pentru a crea sacul de cuvinte se lipsesc toate datele de care avem nevoie in calculul scorului final: descrierea fara cu-

vinte de legatura, gen, actori, scriitori, directori si descriere, intr-un singur camp pe care il numim "bag" aceasta operatie facandu-se pentru fiecare film, deci fiecare film va avea propriul bag. Urmatorul pas este de a pune toate bag-urile la un loc si de a scoate duplicatele la urma ramanand cu n cuvinte care apar in datele din toate filmele. Pe baza acestor cuvinte se va crea pentru fiecare film cate un vector de dimensiune n fiecare pozitie reprezentand frecventa cuvantului din filmul curent. Ca exemplu pentru explicatia de mai sus luam trei propozitii:

(1) Lui Daniel ii place sa se uite la filme.

(2) Lui Marius ii place sa se uite la fotbal.

Figura 1.1: Propozitii pentru exemplu

Dupa ce scoatem cuvintele de legatura: lui, ii, sa, se, propozitiile prima propozitie va arata Daniel, place, uite, filme, iar a doua Marius place uite fotbal. Dupa ce se lipsesc cele doua propozitii campul cuvintelor va fi: Daniel, place, uite, filme, Marius, fotbal. Apoi vectori care se vor crea pentru cele doua propozitii vor arata in felul urmator:

(1) "Daniel":1, "place":1, "uite":1, filme:"1", "Marius":0, "fotbal":0

(2) "Daniel":0, "place":1, "uite":1, filme:"0", "Marius":1, "fotbal":1



(1)[1, 1, 1, 1, 0, 0]

(2)[0, 1, 1, 0, 1, 1]

Figura 1.2: Propozitii pentru exemplu

1.3 Similaritatea cosinus

Metrica de similaritate cosinus este utilizată pentru a determina cât de asemănătoare sunt documentele, indiferent de mărimea lor. Aceasta estimează matematic cosinul unghiului format de doi vectori proiectați într-un spațiu multidimensional. Similaritatea cosinusului este utilă deoarece, chiar dacă două documente comparabile sunt separate de o distanță euclidiană mare (din cauza dimensiunii documentelor), este probabil ca acestea să fie similare. Pentru acest model, se folosește similaritatea cosinusului pentru a defini "similitudinea" dintre filme. Motivul pentru care nu se folosește doar distanța dintre vectori ca scor de similaritate, chiar dacă ar putea părea rezonabil, lungimile vectorilor pot afecta distanța euclidiană, dar nu și distanța cosinus ($1 - \cos(\text{unghi})$). Deoarece lungimile a doi vectori definesc dimensiunea acestora (de exemplu, cantitatea de date), dar unghiul definește mai bine caracteristicile lor în lipsa unor cuvinte mai bune. Prin urmare, distanța unghiulară este mai potrivită pentru a defini similitudinea lor.

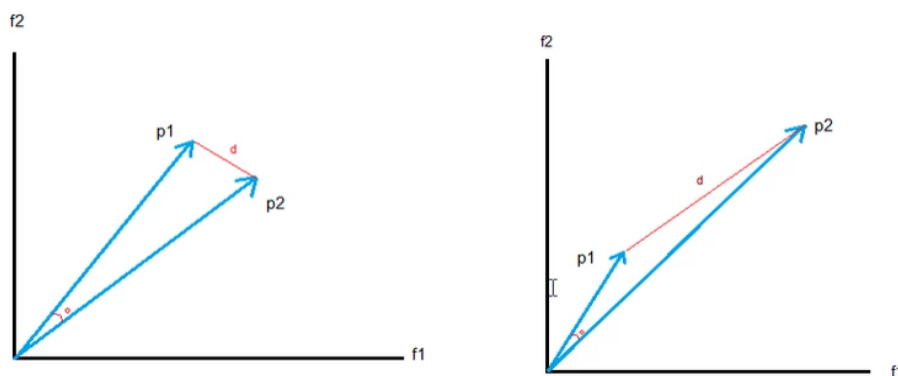


Figura 1.3: Distanța euclidiană vs. distanța unghiulară

După cum se poate vedea, distanța euclidiană devine mult mai mare pe al doilea grafic, chiar dacă unghiul este destul de mic.

1.4 Calcularea scorului final

După ce se calculează o matrice în care perechea (i,j) reprezintă similaritatea cosinus dintre două filme se calculează încă o matrice care va reprezenta scorul final. Similaritatea cosinus va fi îmbunătățită în funcție de cât de apreciat este un film, astfel zece procente din scorul final le va reprezenta media dintre suma voturilor și numărul de votanți. La final se vor lua cele mai mari n scoruri, n reprezentând numărul de filme pe care vrem să le recomandăm, exceptândul pe primul deoarece cel mai bun scor îl va avea filmul însuși.

Capitolul 2

Modelare si proiectare

2.1 Cazurile utilizatorului si diagrame

Exista un sigur tip de utilizator care poate avea interactiune cu aplicatia, pe viitor se poate dezvolta inca un tip de user acesta avand scopul de a administra tabela cu filme. Fiecare utilizator are un nume unic si o parola pentru a se conecta la aplicatie, acolo unde el poate sa: caute filme, sa isi creeze propria lui lista cu filme, sa dea nota la un film sau sa vada detalii despre un film acolo unde sunt prezente si recomandari pentru filmul respectiv.

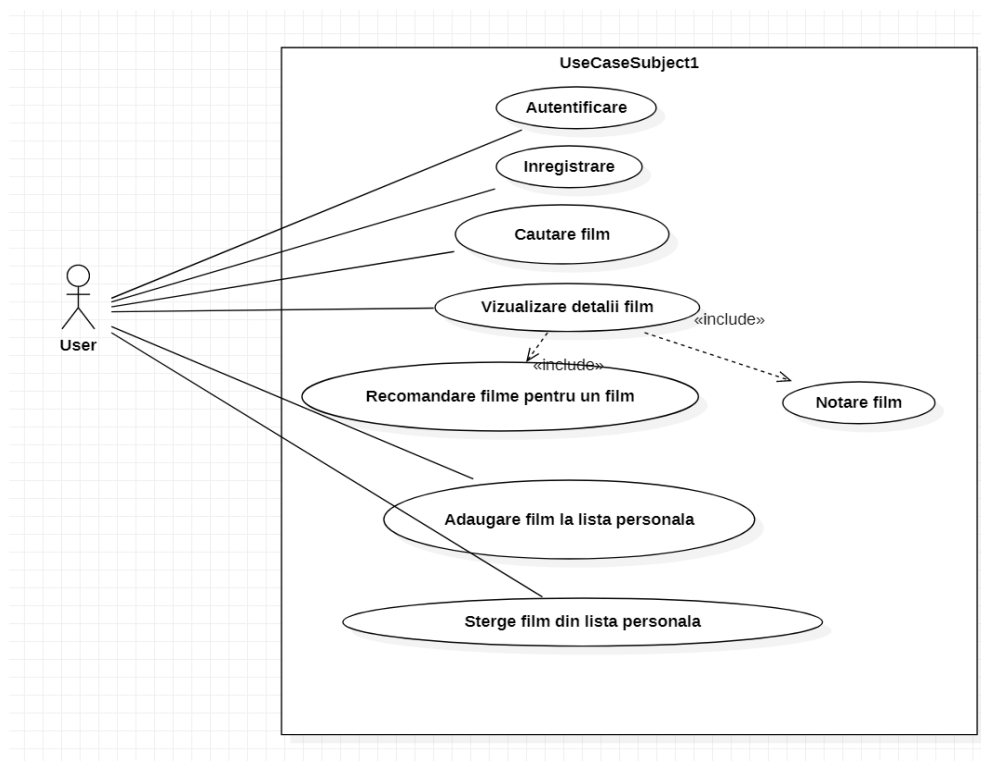


Figura 2.1: Diagrama de cazuri

Figura 1: Diagrama secventiala inregistrare

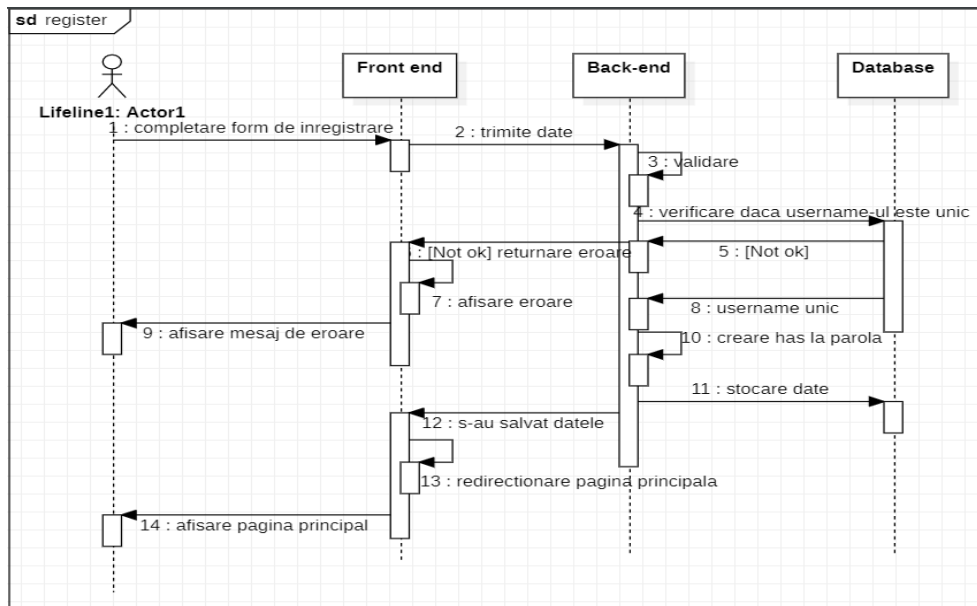


Figura 2.2: Diagrama secventiala pentru inregistrare

Figura 2: Diagrama secventiala autentificare

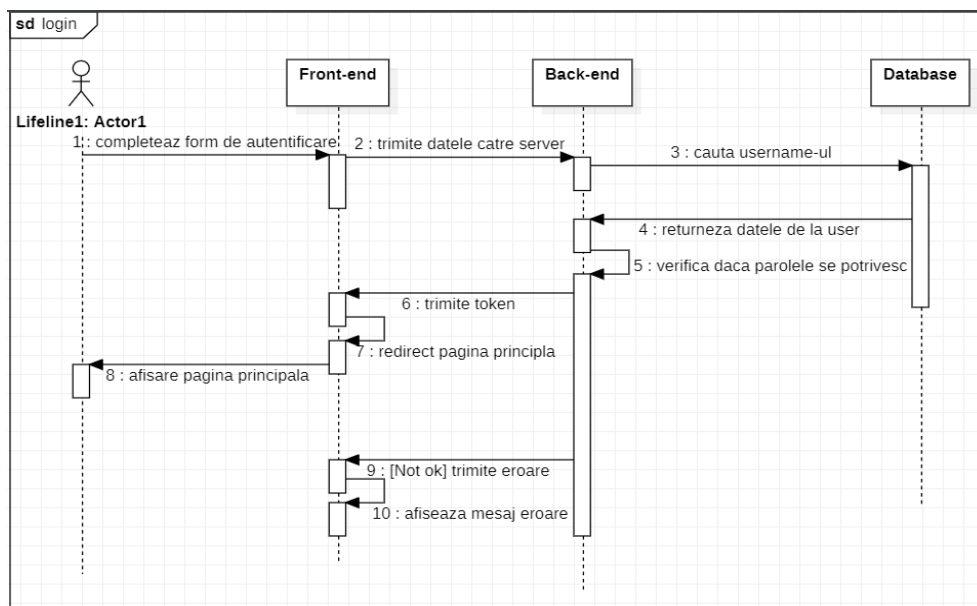
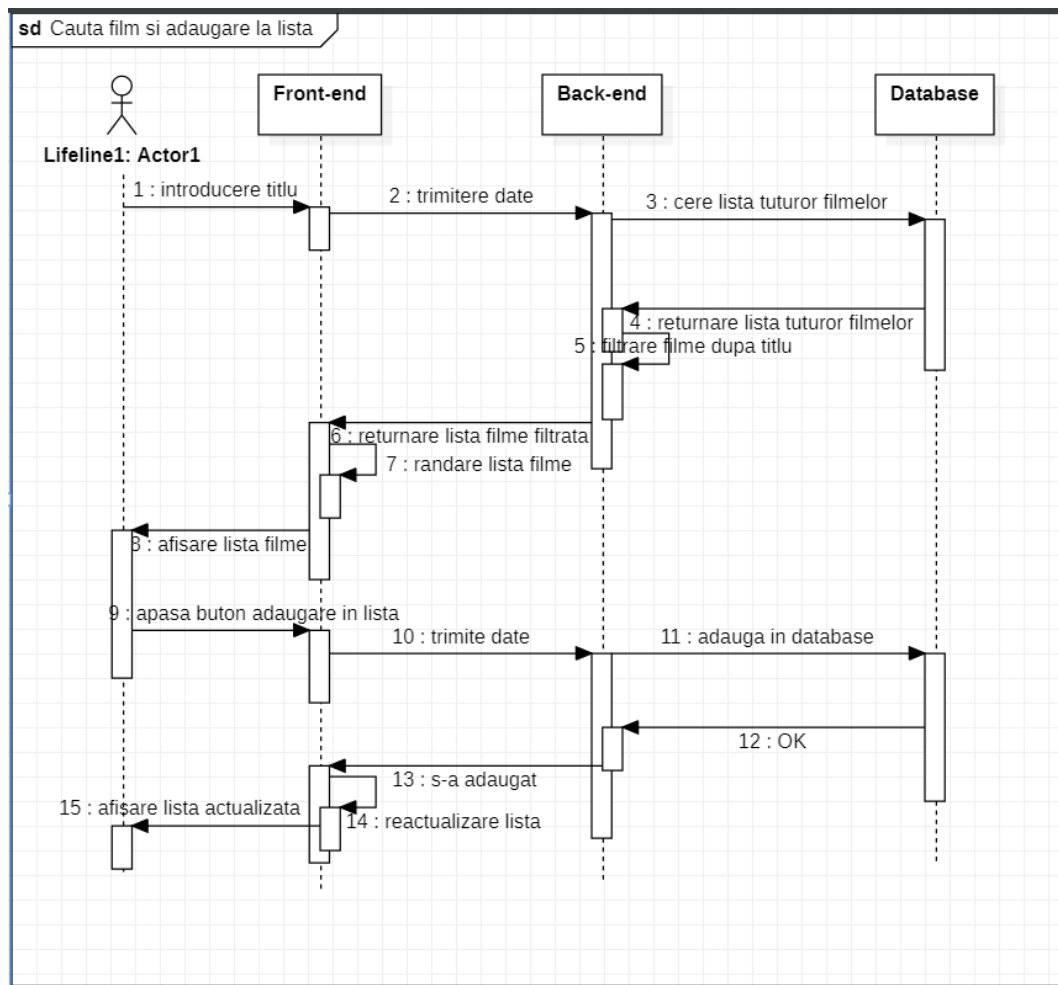


Figura 2.3: Diagrama secventiala pentru autentificare

Figura 3: Diagrama secventiala adaugare film la lista personala**Figura 2.4:** Diagrama secventiala adaugare film la lista

2.2 Utilizarea aplicatiei

Când utilizatorul acesează siteul acesta este redirectionat către pagina de autentificare, dacă acesta nu are cont poate apăsa pe butonul Create account și va fi redirectionat către o pagină care conține un formular de înregistrare. După ce toate datele au fost completate și înregistrarea a fost făcută cu succes acesta va fi redirectionat înapoi către pagina de autentificare, unde se poate autentifica cu noile credențiale.

După autentificare, utilizatorul este redirectionat către pagina principală unde în dreapta are un meniu de navigare și în centrul paginii o bară de căutare. Pe pagina principală utilizatorul poate să își caute filme, după introducerea unui titlu și apăsarea butonului de căutare i se vor afișa o listă cu toate filmele ce conțin în titlu șirul de caractere introduse de utilizator. Pe această pagină un film poate avea una sau două posibilități: să se vadă detalii despre film sau să fie adăugat la lista

personală de filme.

Dacă se apasă pe butonul de Detalii la un film utilizatorul este redirectionat către pagina unde se pot vedea mai detaliile unui film, cum ar fi: titlu, descriere, actorii care joacă în film, țara de origine a filmului, directorii filmului, genurile, scriitorii care au venit cu idei pentru film, rating-ul general al filmului care reprezintă media notelor date de ceilalți utilizatori și cel mai important lucru se poate vedea patru filme recomandate pentru filmul respectiv. Tot pe această pagină se poate acorda o notă filmului influențând de altfel și nota generală a filmului.

Dacă se apasă pe butonul de adăugare la lista personală filmul va fi adăugat, iar butonul respectiv va dispărea. Pentru a vedea această listă din meniul din dreapta se poate naviga către pagina în care se prezintă lista cu filme adăugate de utilizator. Aceste filme tot două butoane: unul de ștergere care șterge filmul din listă și același buton de detalii ca pe pagina principală.

Capitolul 3

Capitolul 3: Tehnologiile de pe Back-end

3.1 Node Js

Node js este un ecosistem de JavaScript , open-source , cross-platform si care este bazat pe motorul V8 de la Chrome acesta executand cod JavaScript in afara unui browser web. Node.js permite dezvoltatorilor să folosească JavaScript pentru a scrie instrumente de linie de comandă și pentru scripturi de pe partea de server, rularea scripturilor pe partea de server pentru a produce conținut dinamic al paginii web înainte ca pagina să fie trimisă în browserul web al utilizatorului.

JavaScript a fost conceput ca un limbaj de programare cu un singur fir care rulează într-un browser. A fi un singur fir înseamnă că numai un singur set de instrucțiuni este executat în orice moment în același proces (browserul, în acest caz, sau doar fila curentă în browserele moderne). Acest lucru a făcut lucrurile mai ușoare pentru implementare și pentru dezvoltatorii care folosesc limba. JavaScript a fost inițial un limbaj util numai pentru adăugarea unor interacțiuni la paginile web, validări de formulare și așa mai departe - nimic care să nu necesite complexitatea multithreading-ului. Cum funcționează sub capotă este destul de interesant. În comparație cu tehnicile tradiționale de servire web în care fiecare conexiune (cerere) generează un fir nou, preluând RAM de sistem și, în cele din urmă, maximizând cantitatea de RAM disponibilă, Node.js funcționează pe un singur fir, utilizând apeluri non-blocante, permițându-i să suporte zeci de mii de conexiuni simultane ținute în bucla evenimentului

Cand vorbim de node js este foarte important de mentionat este suportul incorporat pentru gestionarea pachetelor utilizand NPM(node package manager), un instrument care vine în mod implicit la fiecare instalare Node.js. Ideea modulelor NPM este destul de similară cu cea a Ruby Gems : un set de componente reutiliza-

bile disponibile public, disponibile prin instalare ușoară printr-un depozit online, cu gestionarea versiunilor și dependenței. Ecosistemul modulului este deschis tuturor și oricine își poate publica propriul modul care va fi listat în depozitul npm.

3.2 Javascript

În cea mai mare parte a vieții sale, limbajul de programare JavaScript a trăit în interiorul web browsere. A început ca un limbaj de scriptare simplu pentru modificarea detaliilor mici ale pagini web, dar a devenit un limbaj complex, cu o mulțime de aplicații și biblioteci. Mulți furnizori de browsere precum Mozilla și Google au început să pompeze resurse în timpii de rulare rapid JavaScript, iar browserele au obținut motoare JavaScript mult mai rapide ca un rezultat. În 2009, a apărut Node.js. Node.js a scos V8, puternicul motor JavaScript Google Chrome, din browser și i-a permis să ruleze pe servere. În browser, dezvoltatorii nu au avut de ales decât să aleagă JavaScript. Pe lângă Ruby, Python, Java, și alte limbi, dezvoltatorii ar putea alege acum JavaScript atunci când dezvoltă aplicații de pe server. Este posibil ca JavaScript să nu fie limbajul perfect pentru toată lumea, dar Node.js are adevărat beneficii. În primul rând, motorul V8 JavaScript este rapid, iar Node.js încurajează un stil de codare asincron, făcând codul mai rapid evitând în același timp coșmarurile cu mai multe fire. JavaScript avea, de asemenea, o mulțime de biblioteci utile datorită popularității sale. Cu excepția Cel mai mare beneficiu al Node.js este capacitatea de a partaja codul între browser și server. Dezvoltatorii nu trebuie să facă niciun fel de schimbare de context atunci când trec de la client și Server.

3.3 MySQL

MySQL este un sistem de gestiune al bazelor de date relațional, produs de compania suedeză MySQL AB și distribuit sub Licență Publică Generală GNU. Este cel mai popular SGBD open-source la ora actuală, fiind o componentă cheie a stivei LAMP(Linux, Apache, MySQL, PHP).

Deși este folosit foarte des împreună cu limbajele de programare JAVA, PHP, cu MySQL pot construi aplicații în orice limbaj major. Există multe scheme API disponibile pentru MySQL ce permit scrierea aplicațiilor în numeroase limbaje de programare pentru accesarea bazelor de date MySQL. O interfață de tip ODBC denumită MyODBC permite altor limbaje de programare ce folosesc această interfață, să utilizeze bazele de date MySQL cum ar fi ASP sau Visual Basic (Molnar).

În multe cărți de specialitate este precizat faptul că MySQL este destul de ușor de învățat și folosit în comparație cu multe din aplicațiile de gestiune a bazelor de

date, ca exemplu comanda de ieșire fiind una simplă și evidentă: „exit” sau „quit”.

Pentru a administra bazele de date MySQL se poate folosi modul linie de comandă sau se pot descărca de pe internet diferite programe ce creează o interfață grafică: MySQL Administrator și MySQL Query Browser. Un alt instrument de management al acestor baze de date este aplicația SQL Manager.

Serverul de baze de date MySQL este foarte rapid, fiabil și ușor de utilizat. Inițial a fost dezvoltat pentru a manipula baze de date de dimensiuni mari mult mai rapid decât soluțiile existente.

MySQL Database Software este un sistem client/server ce constă într-un server MySQL multi-threaded care suportă diferite programe client și biblioteci, unelte administrative și o gamă largă de interfețe pentru programarea aplicațiilor.

3.4 Python

Python este un limbaj de programare interpretat, orientat spre obiecte, la nivel înalt, cu semantică dinamică. Structurile sale de date încorporate la nivel înalt, combinate cu tastarea dinamică și legarea dinamică, îl fac foarte atractiv pentru dezvoltarea rapidă a aplicațiilor, precum și pentru utilizarea ca limbaj de scriptare sau lipici pentru a conecta componentele existente împreună. Sintaxa simplă, ușor de învățat a Python accentuează lizibilitatea și, prin urmare, reduce costul întreținerii programului. Python acceptă module și pachete, ceea ce încurajează modularitatea programului și reutilizarea codului. Interpretul Python și biblioteca standard extinsă sunt disponibile sub formă sursă sau binară fără taxe pentru toate platformele majore și pot fi distribuite în mod liber.

Adesea, programatorii se îndrăgostesc de Python din cauza productivității crescute pe care o oferă. Deoarece nu există nicio etapă de compilare, ciclul de editare-test-depanare este incredibil de rapid. Depanarea programelor Python este ușoară: o eroare sau o intrare greșită nu va provoca niciodată o eroare de segmentare. În schimb, atunci când interpretul descoperă o eroare, ridică o excepție. Când programul nu prinde excepția, interpretul imprimă o urmă de stivă. Un depanator la nivel de sursă permite inspectarea variabilelor locale și globale, evaluarea expresiilor arbitrare, setarea punctelor de întrerupere, trecerea prin cod o linie la un moment dat și așa mai departe. Depanatorul este scris chiar în Python, mărturisind puterea introspectivă a lui Python. Pe de altă parte, deseori cel mai rapid mod de a depana un program este să adăugați câteva instrucțiuni de tipărire la sursă: ciclul rapid de editare-test-depanare face această abordare simplă foarte eficientă.

3.5 Flask

Flask este un framework web scris în Python care este clasificat ca un microframe, deoarece nu necesită anumite instrumente sau biblioteci. Nu are un strat de abstrac-tizare a bazei de date, validarea formularelor sau orice alte componente în care bi-bliotecile terțe preexistente oferă funcții comune. Cu toate acestea, Flask acceptă extensii care pot adăuga caracteristici ale aplicației ca și cum ar fi implementate în Flask în sine. Există extensii pentru mapere relaționale obiect, validarea formule-lor, gestionarea încărcărilor, diverse tehnologii de autentificare deschise și mai multe instrumente comune legate de cadru.

Capitolul 4

Capitolul 4: Tehnologiile de pe Front-end

4.1 React

Desconsiderat un JavaScript framework precum Angular sau Vue.js, React este de fapt o bibliotecă de tip open source. Este folosit mai ales pentru interfețe web mari, complexe dar și pentru aplicații single-paged. Creat de Jordan Walke, un software engineer la Facebook, a fost implementat rapid în news feed-ul Facebook, în 2011. Un an mai târziu, Instagram, aplicația deținută de Facebook, a implementat React și de aici începe povestea. În ziua de azi, sute de mii (poate chiar milioane) de website-uri sunt susținute de această bibliotecă și alte mii se nasc în fiecare zi. De fapt, de la lansarea React, am observat o creștere explozivă în utilizarea de bibliotecă JavaScript mici ca dimensiuni, dar puternice. Utilizatorii vor să utilizeze din ce în ce mai multe pagini web mai rapide și mai dinamice, în timp ce dezvoltatorii optează pentru medii moderne și flexibile fără tone de linii de cod în pachet. De aceea ReactJS este o alegere evidentă pentru mulți. Ca să explicăm de ce, haideți să recapitulăm principalele motive pentru care folosim React. Desi considerat un JavaScript framework precum Angular sau Vue.js, React este de fapt o bibliotecă de tip open source. Este folosit mai ales pentru interfețe web mari, complexe dar și pentru aplicații single-paged.

Primul lucru care face atât de mulți oameni să folosească ReactJS în proiectele lor este probabil simplitatea sa. React este o bibliotecă JavaScript, astfel încât dacă un dezvoltator este familiarizat cu funcțiile JS, va avea un start mai ușor cu ReactJS. Cu această bibliotecă, dezvoltatorii definesc interfețe cu o sintaxă asemănătoare HTML numită JSX. Drept urmare, este produs cod HTML și CSS. API-ul React este foarte mic, dar puternic și tot ce trebuie să faci înainte de a începe este să înveți câteva funcții de bază. Un pic din curba de învățare apare când doriți să utilizați React cu

alte biblioteci JS, cum ar fi Redux, Material UI sau Enzyme. Deși nu fac parte din stiva React, astfel de biblioteci adaugă funcții suplimentare și vă permit să gestionați mai ușor componentele React. Cele mai comune biblioteci sunt bine documentate și nu ar trebui să creeze probleme niciunui dezvoltator.

React JS folosește câteva extensii numite JSX și Virtual DOM pentru a crea interfețe utilizator. Acestea permit dezvoltatorilor să le creeze pentru majoritatea platformelor și, datorită faptului că pot vedea rezultatele codului lor instantaneu, pot avea o vizualizare și o înțelegere mai bună a ceea ce fac. React JS folosește ceva numit JSX. Aceasta este o extensie a React care permite utilizarea HTML cu JavaScript și acest lucru face ca codul dvs. să fie mai versatil. React este compatibil cu majoritatea browserelor moderne, ceea ce îi ajută pe dezvoltatori să își schimbe și să testeze DOM pe diferite platforme. Virtual DOM: DOM este prescurtarea pentru Document Object Model, care este o interfață de program de aplicație (API). Permite programelor să citească conținutul oricărui site web, astfel încât să poată fi modificat în funcție de preferințele și nevoile programatorului. Orice site web care nu folosește React JS folosește HTML pentru a modifica și modifica DOM-ul său. Datorită implementării JSX, React JS poate avea un DOM virtual, care este o copie a DOM-ului original utilizat de aplicație sau site-ul web. Principala diferență dintre DOM și DOM-ul virtual este că primul arată doar modificările după ce pagina este încărcată din nou, în timp ce acesta din urmă le arată în timp real, fără a fi nevoie să reîncărcați.

4.2 TypeScript

TypeScript a fost făcut public pentru prima dată în octombrie 2012 (la versiunea 0.8), după doi ani de dezvoltare internă la Microsoft. La scurt timp după anunț, Miguel de Icaza a laudat limba însăși, dar a criticat lipsa de sprijin IDE matur în afară de Microsoft Visual Studio, care nu era disponibil pe Linux și OS X în acel moment. Astăzi există sprijin în alte IDE, în special în Eclipse, printr-un plug-in contribuit de Palantir Technologies. Diversi editori de text, inclusiv Emacs, Vim, Webstorm, Atom și a Microsoft Cod Visual Studio acceptă, de asemenea, TypeScript. TypeScript este o limbă de programare dezvoltată și întreținută de Microsoft. Este un sintactic strict superset de JavaScript și adaugă opțional tastarea statică la limbă. TypeScript este conceput pentru dezvoltarea de aplicații mari și transpilează la JavaScript. Deoarece TypeScript este un superset de JavaScript, programele JavaScript existente sunt, de asemenea, programe TypeScript valabile. TypeScript poate fi folosit pentru a dezvolta aplicații JavaScript pentru ambele părți ale clientului și partea de server executare (ca și în cazul Node.js sau Deno). Există mai multe opțiuni disponibile pentru transcompilare. Poate fi folosit fie TypeScript Checker implicit, sau Babel compilatorul poate fi invocat pentru a converti TypeScript în JavaScript. TypeScript

acceptă fișiere de definiție care pot conține informații de tip ale bibliotecilor JavaScript existente, la fel C ++ fișierele antet poate descrie structura existentului fișiere obiect. Aceasta permite altor programe să utilizeze valorile definite în fișiere ca și cum ar fi entități TypeScript tipizate static. Există fișiere antet terțe pentru biblioteci populare, cum ar fi jQuery, MongoDB, și D3.js. Anteturi TypeScript pentru Node.js sunt de asemenea disponibile module de bază, care permit dezvoltarea programelor Node.js în TypeScript.

Capitolul 5

Specificari Back-end

Pe partea de back end sunt prezente doua servere, unul scris in JavaScript cu ajutorul framework-ului Node js care serveste partea de front-end cu toate datele de care are nevoie si un server scris in Python cu ajutorul framework-ului Flask cu ajutorul caruia aflu filme recomandate pentru un film sau o lista de filme.

5.1 Baza de date

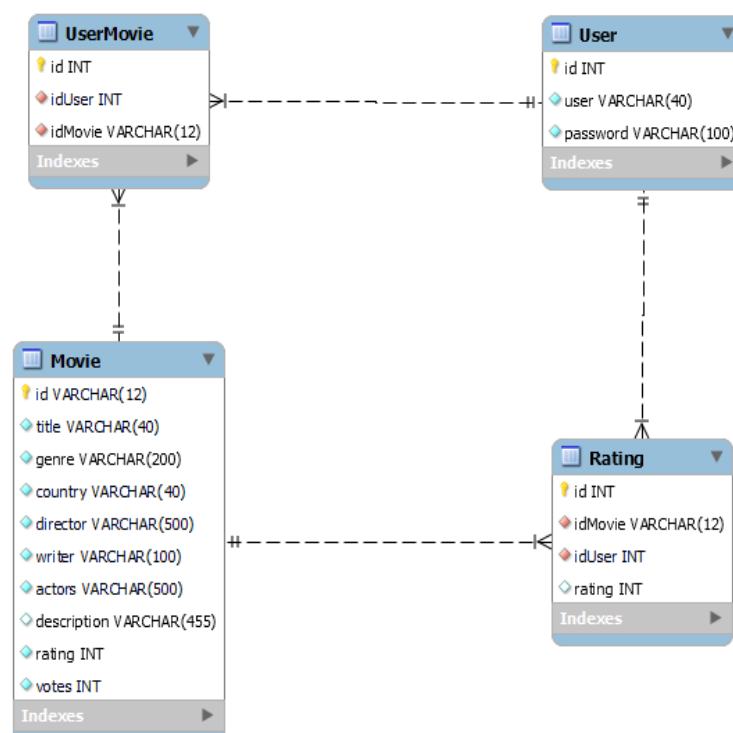


Figura 5.1: Diagrama baza de date.

Un lucru foarte important in dezvoltarea aplicatiei a fost alegerea bazei de date, am optat pentru MySQL versiunea 8.0. In tabelul de User voi salva datele

de inregistrare a unui utilizator , in tabelul Movie sunt prezente datele care descriu un film prin coloanele: title, genre, country, directors, writers, actors, description, votes si rating, in tabelul Rating salvez datele unui rating dat de un user pe laga valoare ratingului salvez si id-urile de la film si user pentru a stii cine da rating si la ce film totodata pe partea de back-end fac si o actualizare a rating ului si numarului de voturi din tabelul Movie asta pentru o a face aplicatia sa se miste mai rapid, iar in tabelul UserMovie salvez id ul de la un user si id ul de la un film reprezentand un film pe care utilizatorul l-a vizualizat.

5.2 Server Node Js

Serverul de Node Js il folosesc pentru a face inregistrare, login, returnarea tuturor filmelor care contin un numit sub titlu, adaugarea unui film la lista utilizatorului, returnarea a tuturor filmelor unui utilizator si adaugarea de rating a unui film. Cu ajutorul pachetului Express stabilesc rutele si pornesc serverul. Cand se face un request pe o ruta, se apeleaza functia asignata din Service, care are rol de a modela datele in functie de cum avem nevoie. Pentru a comunica cu baza de date ne folosim de Repository, acesta face legatura cu baza de date.

Standardul REST („representational state transfer”) a fost creat pentru a ghida design-ul și arhitectura serviciilor web. REST descrie cum ar trebui să se comporte internetul. Orice serviciu care reușește să se încadreze în parametrii și regulile descrise de acest standard se numește un serviciu RESTful. Atunci când o cerere de client se face printr-un API RESTful, acesta transferă o reprezentare a stării resursei către solicitant sau punct final. Aceste informații sau reprezentări sunt livrate într-unul din mai multe formate prin HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP sau text simplu. JSON este cel mai popular limbaj de programare folosit, deoarece, în ciuda numelui său, este limbaj-agnostic, precum și lizibil atât de oameni, cât și de mașini. Altceva de reținut: anteturile și parametrii sunt, de asemenea, importanți în metodele HTTP ale unei cereri HTTP RESTful API, deoarece conțin informații importante de identificare referitoare la metadatele cererii, autorizarea, identificatorul uniform al resurselor (URI), cache, cookie-uri și Mai Mult. Există anteturi de solicitare și anteturi de răspuns, fiecare cu propriile informații de conexiune HTTP și coduri de stare.

Serverul respectă standardul REST API. Crearea entităților se face cu PUT, interogările cu GET, ștergerea cu DELETE, iar restul operațiilor cu POST. Specificațiile API în format OpenAPI se pot găsi la documentație API

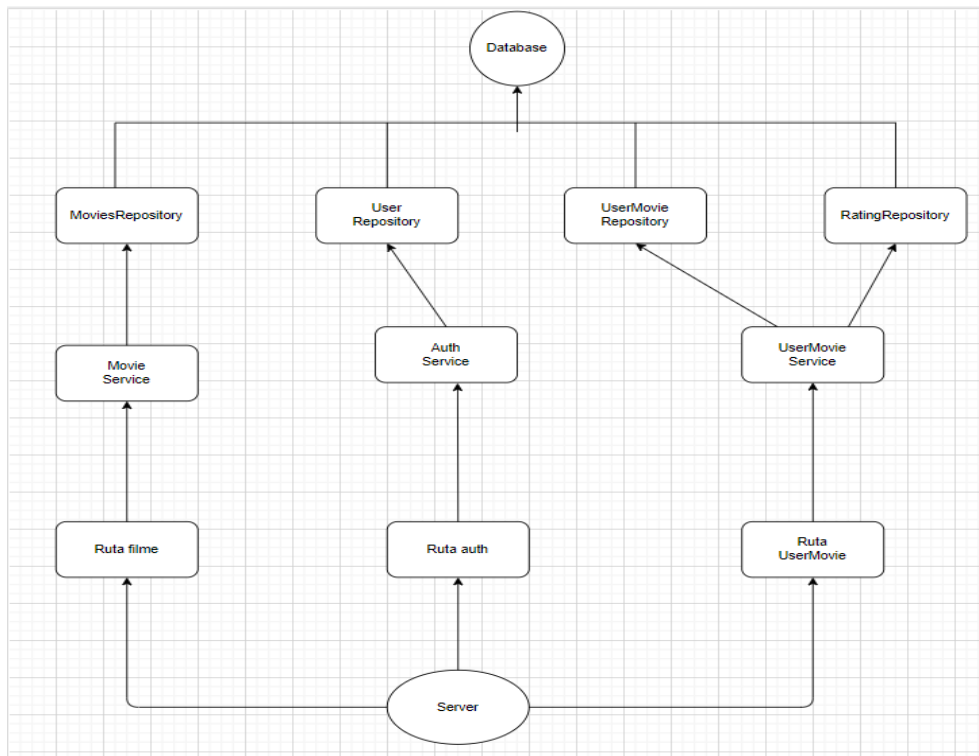


Figura 5.2: Arhitectura server Node Js .

5.2.1 Inregistrare

Procesul de inregistrare este unul simplu, pasii fiind urmatoarii:

1. Se trimite datele catre server
2. Se valideaza datele primite
3. Daca datele sunt valide se verifica daca numele utilizatorului se afla in baza de date deoarece numele trebuie sa fie unic pentru fiecare utilizator
4. Daca totul este bine pana la acest pas se face hash la parola pentru o mai buna securitate

```
//Hash password
const salt = await bcrypt.genSalt(10);
const hashedPassword = await bcrypt.hash(password, salt);
const user = new User(username, hashedPassword);
```

Figura 5.3: Hash la parola.

5. Următorul pas este de a apela funcția din repository care adaugă un user în baza de date.

```
const addUser = async (user) => {  
  const { username, password } = user;  
  const command = `INSERT INTO User (user, password) VALUES ('${username}', '${password}')`;   
  const prom = new Promise((resolve, reject) => {  
    pool.query(command, (err, data) => {  
      if (err) throw Error;  
      resolve(data);  
    });  
  });  
};
```

Figura 5.4: Adaugare user în baza de date.

6. Dacă adăugarea s-a efectuat cu succes se returnează un mesaj de succes altfel se returnează un mesaj de eroare.

5.2.2 Autentificare

Pașii pentru procesul de autentificare sunt următorii

1. Se trimite datele către server
2. Se validează datele primite
3. Se verifică dacă username-ul se află în baza de date

```
const findUser = async (username) => {  
  const command = `Select * from User Where user = '${username}'`;   
  const prom = new Promise((resolve, reject) => {  
    pool.query(command, (err, data) => {  
      if (err) throw Error;  
      resolve(data);  
    });  
  });  
  return prom;  
};
```

Figura 5.5: Căutare după username în baza de date

4. Daca se găsește user-ul in baza de date se verifică parola trimisă in request cu cea din baza de date apoi se trimite raspunsul in functie de potrivirea celor doua parole, daca raspunsul este valid se va trimite id-ul user-ului care va reprezenta token-ul pentru front-end, daca parolele nu corespund se va trimite status 400 si un mesaj de eroare.

```
//Verify
const databaseUser = new User(userList[0].user, userList[0].password);
const id = userList[0].id;
const validPass = bcrypt.compare(databaseUser.password, password);
if (!validPass) response.status(400).send({ error: "Password is incorrect" });
response.status(200).send({ status: "ok", token: id });
```

Figura 5.6: Verificare parola si trimitere raspuns

5.2.3 Lista filmelor care contin un subtitlu

Pașii pentru returnarea listii sunt:

1. Dupa se se apeleaza ruta specifica se ia din body titlul
2. Se apeleaza functioa din repository care returneaza lista
3. In repository se iau toate filmele din baza de date apoi se parcurg pentru a pastra doar filmele care contin subtitlul dat.

```
const searchMovies = async (title) => {
  const command = `SELECT * FROM Movie`;
  const prom = new Promise((resolve, reject) => {
    pool.query(command, (err, data) => {
      if (err) reject(err);
      const result = data.reduce((acc, elem) => {
        if (elem.title.includes(title)) acc.push(elem);
        return acc;
      }, []);
      resolve(result);
    });
  });
  return prom;
};
```

Figura 5.7: Functia din repository care cauta filmele