UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICA

LUCRARE DE LICENŢĂ

SISTEM PENTRU RECOMANDARE DE FILME

Conducător ştiinţific
[Grad, titlu si nume coordonator]

Absolvent
RADU DRAGOS STEFAN

2021

ABSTRACT

Due to the mass growth of information on the internet, it is quite hard to get to what we are interested in or to find something to our liking. For example in the field of cinema people often don't find a movie to their liking. The aim of this work is to create an algorithm to recommend movies to users, and to embed this algorithm in a website where the user can receive movie recommendations. The innovative part is how the difference between films with close similarities is made, and how a user can create their own list from which the algorithm is able to make film recommendations.

The first step in developing a paper and an application is to establish a goal and propose a solution, so the first chapter establishes the existing problem and briefly describes the chosen solution.

Chapter two presents an overview of the algorithm that makes movie recommendations. Here the general steps of the algorithm are explained in turn.

Chapter three presents all the use cases of the application: such as logging in, registering, adding a movie to the list, etc. This is quite an important chapter because it also contains a sequence diagram for each case so that the reader can understand how the application works behind the scenes. The chapter ends with a description of how the user can use the application.

In chapters four and five the technologies used for back-end application development are described in turn: Node Js, Javascript, MySQL, Python and Flask, as well as the front-end ones: HTML, CSS, React and Typescript.

Chapter six presents in detail how the application was developed, first describing how the database design was thought out, then describing the first server which is written in Node Js followed by describing the Flask server with which the movie recommendation is made, the chapter ends with the description of the front-end part.

The last one presents a small conclusion, but also future improvements that the application and the algorithm could have.

Cuprins

Listă de figuri

Capitolul 1

Introducere

Datorită creșterii în masă a informațiilor de pe internet, este destul de greu de a ajunge la cea ce ne interesează sau să găsim ceva pe placul nostru, de exeplu în domeniul cinematografic de multe ori oameni nu găsesc un film pe placul lor. Scopul acestei lucrări este de a crea un algoritm cu ajutorul căreia să se recomande filme utilizatorilor în funcție de alte filme, iar acest algoritm să fie incoporat într-un site web unde utilizatorul să poată să primească recomandări de filme. Partea inovativă a acestei lucrarifiind modul în care se diferențiază filmele cu similaritate apropiată, dar și modul în care un utilizator își poate creea el o propie lista din care să se facă recomandări de filme.

Primul pas în dezvoltarea unei lucrări și a unei aplicații este stabilirea unui scop și să se propună o soluție, așa că în primul capitol se stabilește problema propusă și se descrie pe scurt soluția propusă.

În capitolul doi se prezintă o privire de asamblu a algoritmului care face recomandări de filme unde sunt explicații pe rând pașii generali ai algoritmului.

Capitolul trei prezintă toate cazuile de utilizare ale apicatiei: cum ar fi autentificare, înregistrare, adăugare film la lista, etc. Acesta este un capitol destul de important deoarece în el se regăsește și câte o diagramă seventiala pentru fiecare caz de utilizare astfel încât cititorul să poată înțelege cum funcționează aplicația în spate. Capitolul se termină cu o descriere a modului în care utilizatorul poate folosi aplicația.

În capitolul patru și cinci se descriu pe rând tehnologiile folosite pentru dezvoltarea aplicației pe back-end: Node Js, Javascript, MySQL, Python, Flask, dar și cele de front-end: HTML, CSS, React și Typescript.

În capitolul șase se prezintă în detaliu modul în care aplicația a fost dezoltata. În primul rând se descrie cum a fost gândit proiectarea bazei de date, apoi descrierea primului server care este scris în Node Js urmat de descrierea serverului de Flask cu ajutorul căruia se face recomandarea de filme, capitolul încheindu-se cu descrierea părții de front-end.

În ultimul se prezintă o mică concluzie, dar și viitoare îmbunătățiri pe care aplicația și algoritmul le-ar putea avea.

1.1 Motivatie

În ultimii ani internetul este într-o continuă extindere pe aproximativ toate domeniile. După cum se spune că totul are avantajele și dezavantajele sale prin urmare, odată cu extinderea domeniilor vine supraîncărcare de informații și dificultăți în extragerea datelor. Pentru a putea trece mai ușor peste această problemă sistemele de recomandări joacă un rol destul de important.

Cadrul sistemului de recomandări joacă un rol vital în navigarea pe internet de astăzi, fie că este vorba de cumpărarea unui produs de pe un site de comerț electronic sau vizionarea unui film laun serviciu video la cerere. În viața noastră de zi cu zi, depindem de recomandările oferite de alte persoane, fie din gură din gură, fie din recenziile sondajelor generale. Oamenii folosesc adesea sisteme de recomandare pe web pentru a lua decizii cu privire la elementele legate de alegerea lor. Sistemele de recomandare sunt instrumente și tehnici software al căror scop este de a face recomandări utile și sensibile unei colecții de utilizatori pentru articole sau produse care ar putea să îi intereseze.

Sistemele de recomandare au devenit răspândite în ultimii ani, deoarece se ocupă de problema suprasolicitării informațiilor, sugerând utilizatorilor cele mai relevante produse dintr-o cantitate masivă de date. Pentru produsele media, recomandările de filme online colaborative încearcă să-i ajute pe utilizatori să acceseze filmele preferate prin captarea precisă a vecinilor similari dintre utilizatori sau filme din evaluările lor istorice comune. Cu toate acestea, din cauza datelor rare, selectarea vecinilor devine din ce în ce mai dificilă odată cu creșterea rapidă a filmelor și a utilizatorilor.

Cu alte cuvinte, sistemul de recomandare sau sistemele de recomandare aparțin unei clase de sisteme de filtrare a informațiilor care vizează prezicerea „preferinței” sau „ratingului” acordate unui articol.

Capitolul 2

Cadru de lucru

2.1 Problema propusă

Se dorește crearea unui algoritm destul de eficient care să poată recomandă o lista de filme pentru unul sau mai multe filme. Algoritmul trebuie pus la dispoziția utilizatorilor pintr-o interfață, tot managmantul aplicației fiind făcut de către un back-end. Cu aplicația respectivă utilizatorul ar trebui să poată să își caute filme, să le adauge la o lista, să poată vedea informații și cel mai important să poată primi diferite recomandări de filme.

2.2 Soluție

Soluția e compusă din mai multe părți. În primul rând, e nevoie de o bază de date populate cu filme care să fie bine descrise astfel algoritmul să poată recomanda cele mai bune filme. După un timp în care aplicația va fi folosită aceasta va da un randament și mai bun deoarece utilizatorii vor da note filmelor, iar algoritmul va diferenția filmele cu valorile apropiate ale similarităților. Următorul pas și poate unul din cele mai dificile este dezvoltarea algoritmului care va face recomandările de filme. Ultima parte va consta în crearea unei interfețe unde utilizatorul poate interacționa cu algoritmul, alegerea este un site web deoarece acesta poate fi accesat și de pe telefon dar și de pe un calculator, iar utilizatorului îi va fi simplu să își poată întreține propria lui listă de flme. Pe baza acestor filme se vor face recomandări cu ajutorul algoritumului.

Capitolul 3

Algoritmul de recomandare

3.1 Prelucrarea datelor

Un prim pas în algoritmul de recomandare de filme este de a prelucra datele pentru a obține cele mai bune valori în calculul final al scorului. După ce avem toate datele primul pas este să înlocuim datele nule cu caracterul spațiu pentru a nu conta în calculul final. Apoi un pas destul de important în procesarea datelor este să eliminăm cuvintele de legătură ca de exmplu: 'an', 'be', 'some', etc. Cel mai important câmp din care trebuie scoase aceste cuvinte este descrierea deoarece la final se vor crea vectori din cuvintele pe care le avem în datele noastre, iar cuvintele de legătură nu sunt deloc importante în calculul final noi având nevoie doar de cuvintele estențiale. Ultimul lucru care trebuie făcut în procesarea datelor este de a scoate separatorii de cuvinte, în datele noastre fiind doar virgulă, din colonele: gen, actori, scriitori, directori și descriere.

3.2 Sacul de cuvinte și transformrea cuvintelor în vectori

Următorul pas în algoritmul de recomandare de filme este de a creea ceea ce se numește "Bag of words". Modelul "Bag of words" este o reprezentare pentru procesarea limbajului natural și regăsirea informațiilor care simplifică lucrurile. Un text (cum ar fi o propoziție sau un document) este reprezentat în această paradigmă ca un sac 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 creea sacul de cuvinte se lipesc toate toate datele de care avem nevoie în calculul scorului final: descrierea fără cuvinte de legătură, gen, actori, scriitori, directori și descriere, într-un singur câmp pe care îl numin "bag" această operație făcându-se pentru fiecare film, deci fiecare film va avea pro-

priul bag. Următorul pas este de a pune toate bag-urile la un loc și de a scoate duplicatele la urmă rămânând cu n cuvinte care apar în datele din toate filmele. Pe baza acestor cuvinte se va creea pentru fiecare film câte un vector de dimensiune n fiecare poziție reprezentând frecvența cuvântului din filmul curent. Ca exemplu pentru explicația de mai sus luăm trei propoziții:

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

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

Figura 3.1: Propoziții pentru exemplu

După ce scoatem cuvintele de legătură: lui, îi, să, se, prima propoziție va arată Daniel, place, uite, filme , iar a două Marius place uite fotbal. După ce se lipesc cele două propoziții câmpul cuvintelor va fi: Daniel, place, uite, filme, Marius, fotbal. Apoi vectori care se vor creea pentru cele două propoziții vor arată în felul următor:

(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 3.2: Propoziții pentru exemplu

3.3 Similaritatea cosinus

Matricea de similaritate cosinus este utilizată pentru a determina cât de asemănătoare sunt documentele, indiferent de mărimea lor. Aceasta estimează matematic cosinusul 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 că 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(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.

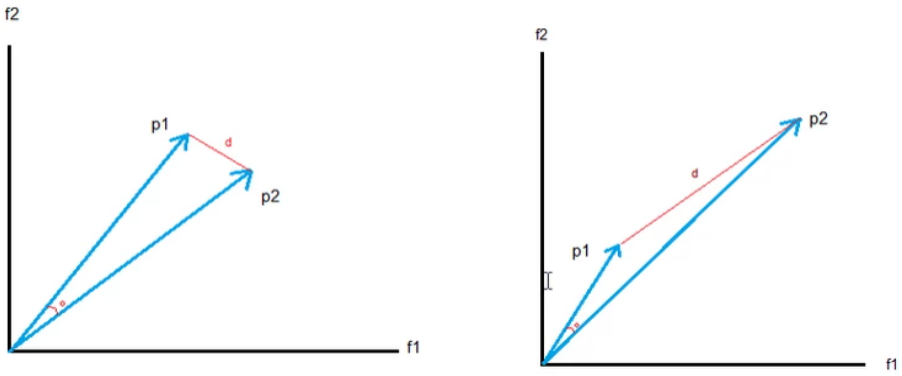

Figura 3.3: Distanța euclidiană vs. distanța unghiulară

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

3.4 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 rating-ul general a unui film va face diferenta dintre filme cu similaritate apropiata fiind adaugat la scoru matricea de similaritati o valoare in functie de rating. La final se vor lua cele mai mari n scoruri , n reprezentând numărul de filme pe care vrem să le recomandăm, exceptându-l pe primul deoarece cel mai bun scor îl va avea filmul însuși.

Capitolul 4

Modelare și proiectare

4.1 Cazurile utilizatorului și diagrame

Există un sigur tip de utilizator care poate avea interacțiune cu aplicația, pe viitor se poate dezvolta încă un tip de utiizatoracesta având scopul de a administra aplicația. Fiecare utilizator are un nume unic și o parolă pentru a se conecta la aplicație, acolo unde el poate să: caute filme, să își creeze propia lui lista cu filme, să dea notă la un film sau să vadă detalii despre un film acolo unde sunt prezente și recomandări pentru filmul respectiv.

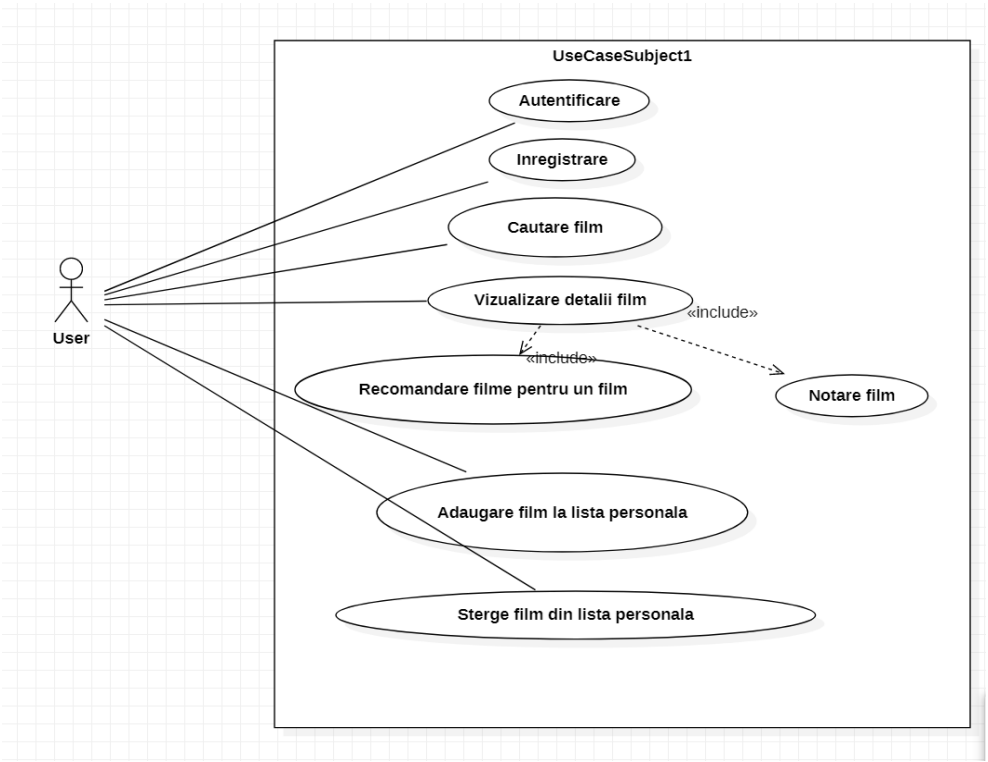

Figura 4.1: Diagrama de cazuri

Figura 1: Diagrama secvențială înregistrare

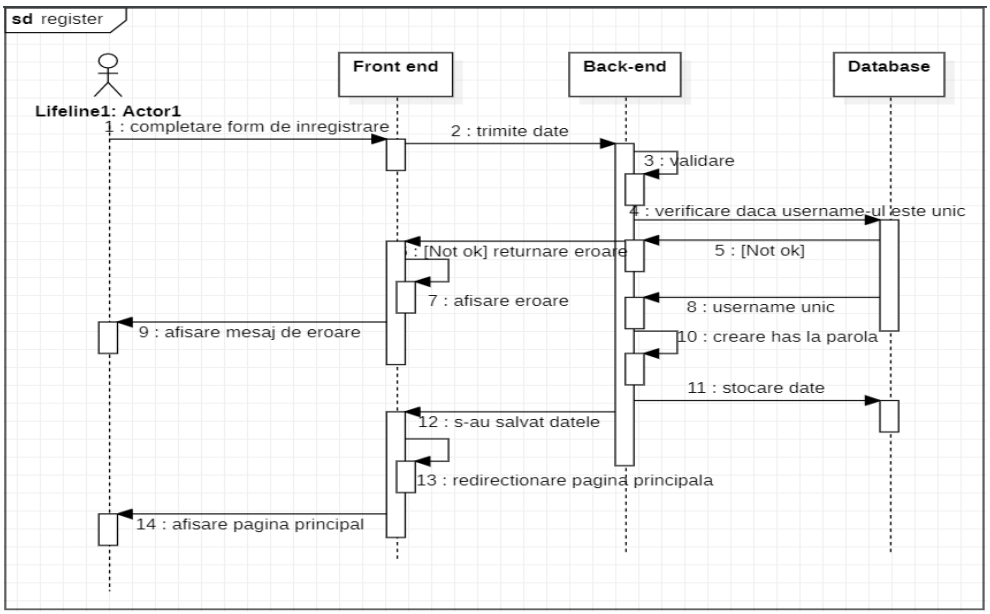

Figura 4.2: Diagrama secvențială pentru înregistrare

Figura 2: Diagrama secvențială autentificare

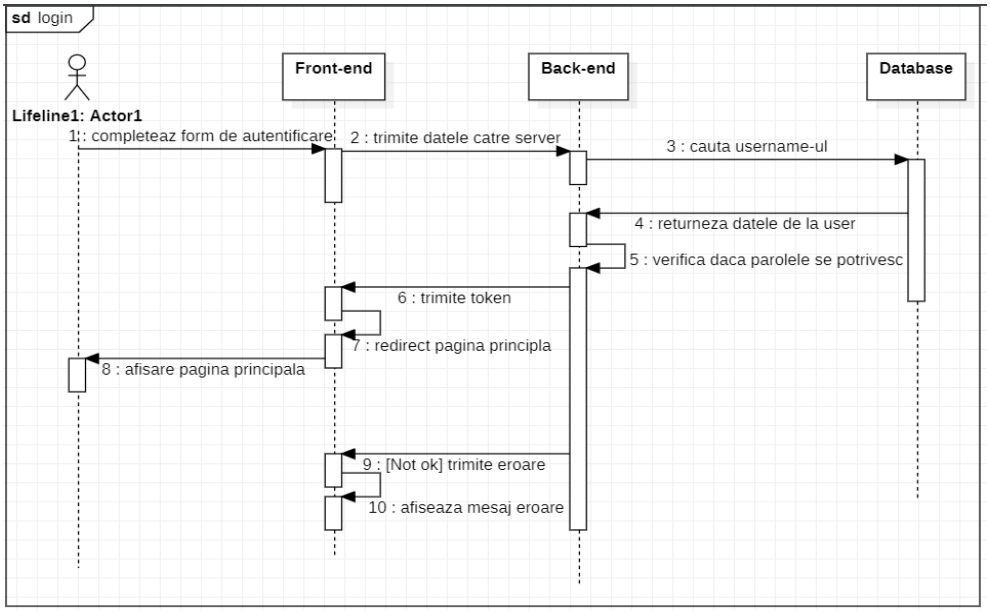

Figura 4.3: Diagrama secvențială pentru autentificare

Figura 3: Diagrama secvențială adaugare film la lista personală

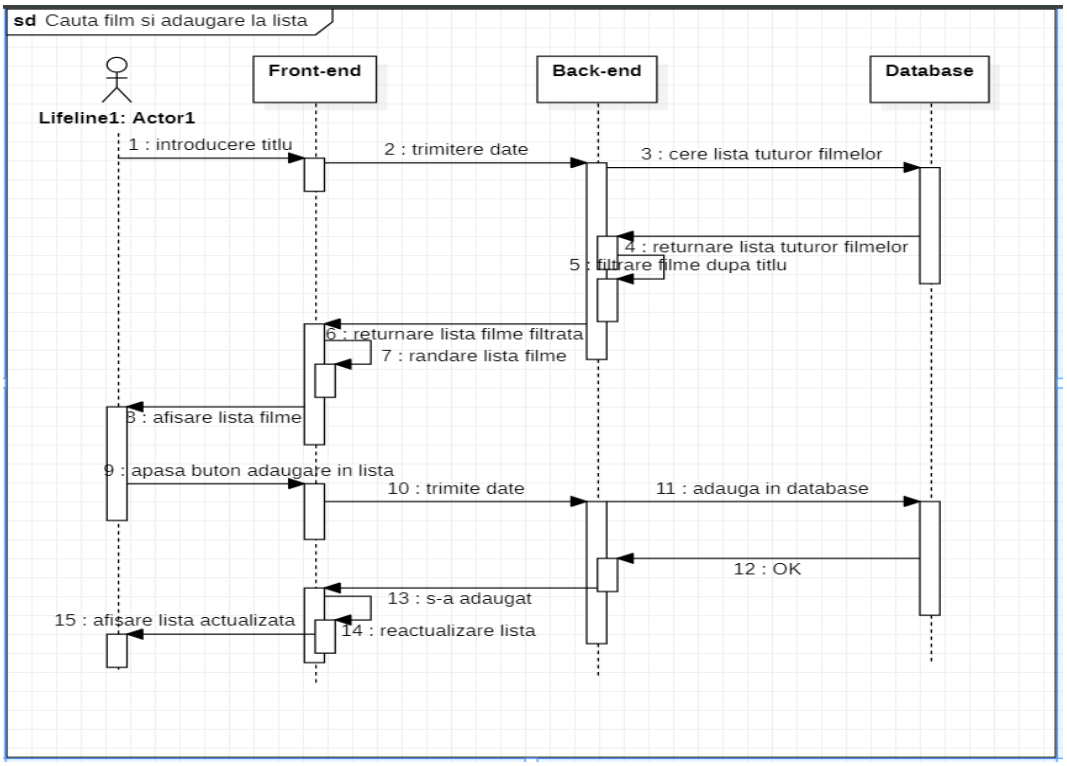

Figura 4.4: Diagrama secvențiala adaugare film la lista

Figura 4: Diagrama secvențială adaugare film la lista personală

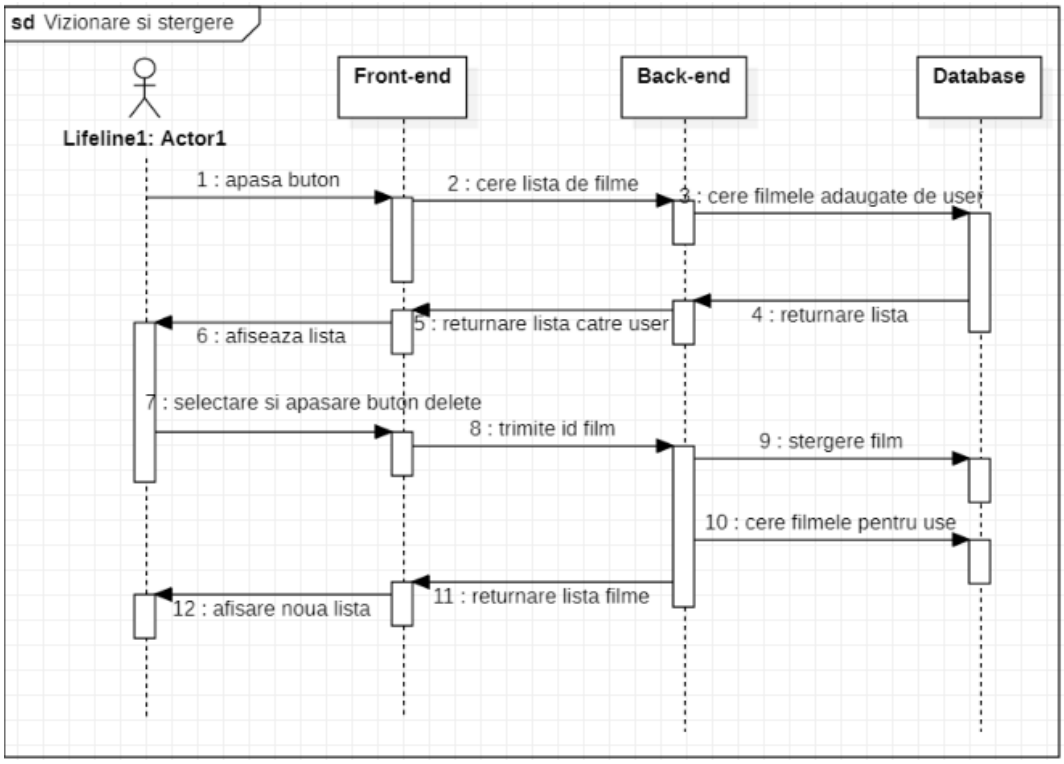

Figura 4.5: Diagrama secvențială adaugare film la listă

Figura 5: Acordare rating

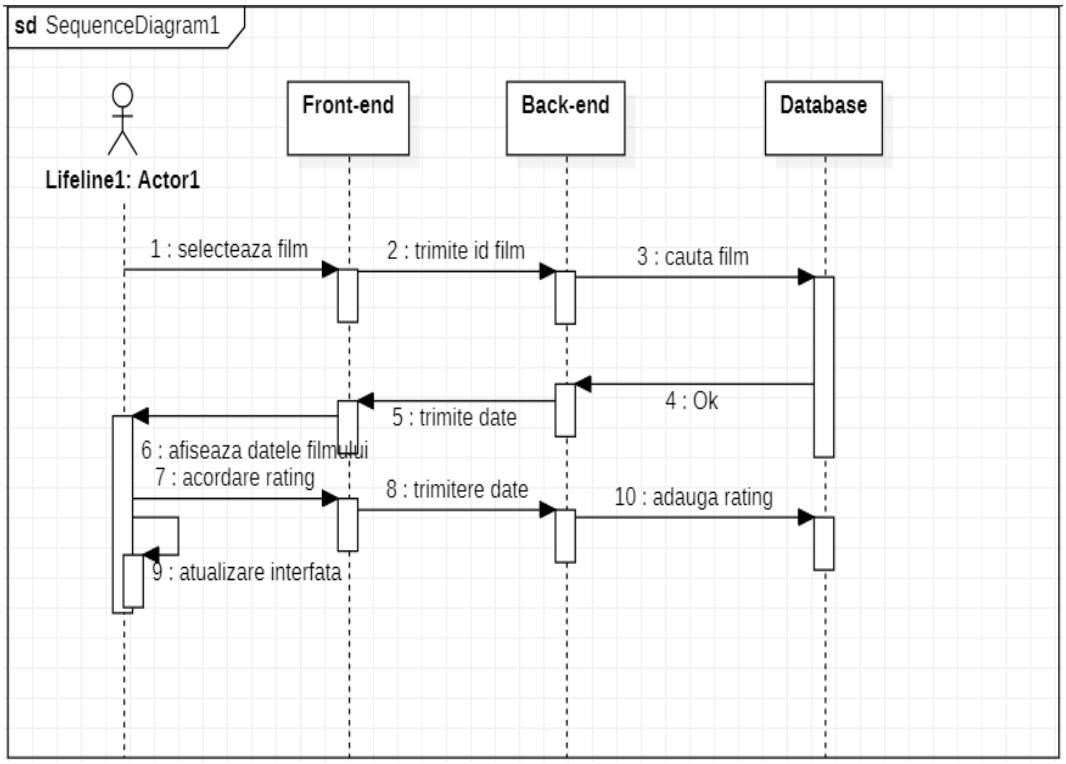

Figura 4.6: Diagrama secvențială adaugare film la listă

Figura 6: Recomandare filme pentru un film

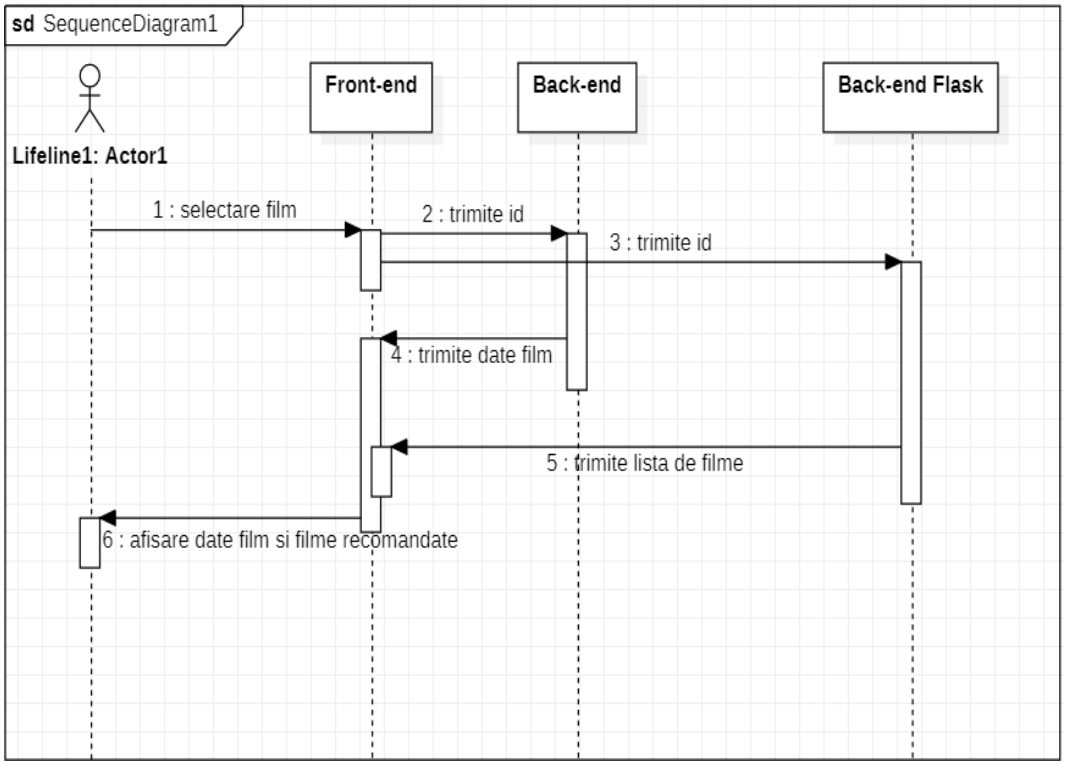

Figura 4.7: Diagrama secvențială adaugare film la listă

Figura 6: Recomandare filme pentru un o lista de filme

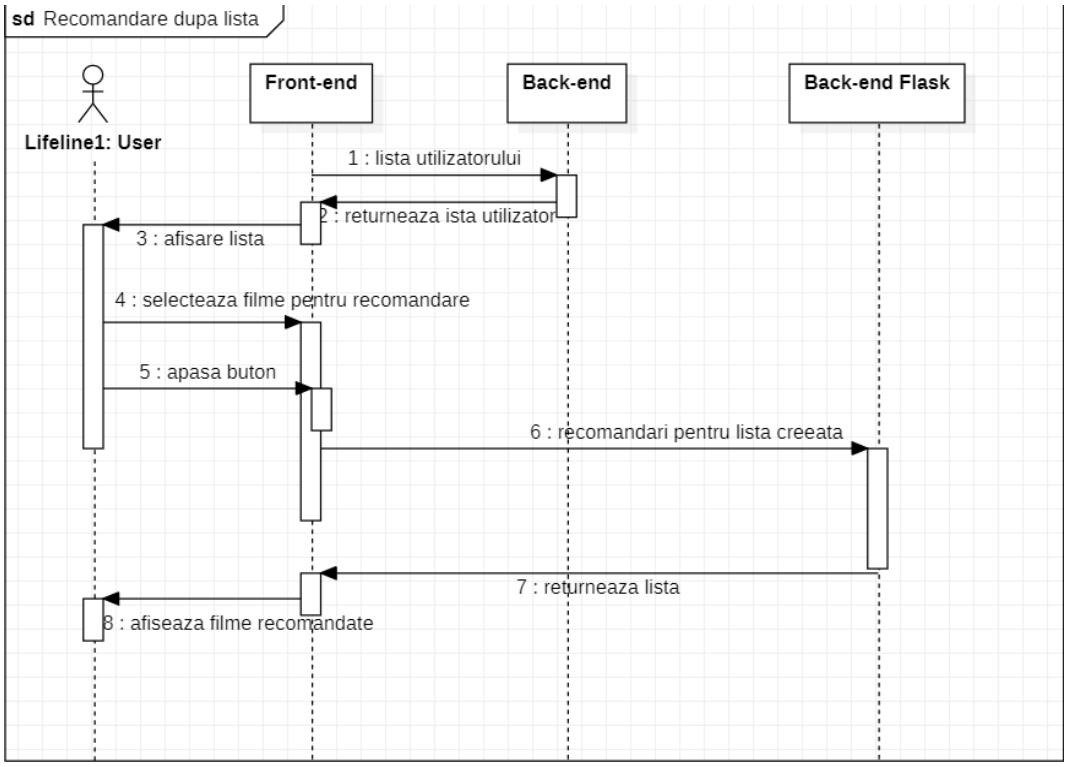

Figura 4.8: Diagrama secvențială recomandare filme pentru o lista de filme

4.2 Utilizarea aplicației

Când utilizatorul aceseaza siteul acesta este redirecționat către pagină de autentificare, dacă acesta nu are cont poate apăsa pe butonul Register și va fi redirecționat 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 redirecționat înpoi către pagină de autentificare, unde se poate autentifica cu noile credențiale.

După autentificare, utilizatorul este redirecționat 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 redirecționat către pagina unde se pot vedea 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 ceilanți utilizatori și cel mai important lucru se poate vedea patru filme recomandate pentru filmul respectiv. Tot pe această pagină se poate acordă 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 lista 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 lista și același buton de detalii că pe pagina principală. Tot pe această pagină utilizatorul își poate creea o sub lista din lista personală, iar la apăsarea butonului de recomandare acesta va primi o serie de filme recomandate pentru ce a ales în sublista lui.

Capitolul 5

Capitolul 3: Tehnologiile de pe Back-end

5.1 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 programatorilor să folosească JavaScript pentru a crea instrumente in linia de comandă și scripturi pentru servere cu ajutorul carora se generează conținutul dinamic pe paginiile web înainte ca pagina să fie transmisă către browserul utilizatorului.

JavaScript a fost creat ca un limbaj de programare care se baza pe browser si care avea un singur fir de execuție. A fi single-threaded înseamnă că, în același proces, se execută un singur set de instrucțiuni la un moment dat (în acest caz, browserul, sau doar fila curentă în browserele moderne). Datorita acestui lucru s-au ușurat lucrurile pentru dezvoltatorii care folosesc acest limbaj. 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 in spate 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. Modulele NPM sunt comparabile cu Ruby Gems în sensul că sunt o colecție de componente reutilizabile

disponibile în mod public, care pot fi instalate cu ușurință dintr-un depozit online și includ gestionarea versiunilor și a dependențelor. Oricine poate contribui la ecosistemul de module prin publicarea propriului modul pentru a fi inclus în depozitul npm.

5.2 Javascript

Limbajul de programare JavaScript și-a petrecut cea mai mare parte a vieții în interiorul browserele web. A început ca un limbaj de scripting de bază pentru ajustarea celor mai fine aspecte ale paginilor web, dar acum a evoluat într-un limbaj sofisticat cu o gamă largă 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. Dezvoltatorii pot folosi acum JavaScript în loc de Ruby, Python, Java și alte limbaje pentru a crea aplicații pe partea serverului. Deși JavaScript poate să nu fie cel mai bun limbaj pentru toată lumea, Node.js oferă o mulțime de avantaje. Pentru început, motorul JavaScript V8 este rapid, iar Node.js suportă codarea asincronă, care accelerează codul evitând în același timp problemele de multithreading. Dezvoltatorii nu trebuie să facă niciun fel de schimbare de context atunci când trec de la client și Server.

5.3 MySQL

MySQL este un sistem de gestionare a bazelor de date relaționale dezvoltat de MySQL AB din Suedia și publicat sub Licența Publică Generală GNU. Este cel mai utilizat sistem de gestionare a bazelor de date open-source în prezent și reprezintă o parte importantă a stivei LAMP (Linux, Apache, MySQL, PHP).

Deși este folosit foarte des împreună cu limbajelel 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 interfaţă, să utilizeze bazele de date MySQL cum ar fi ASP sau Visual Basic (Molnar).

Multe manuale susțin că MySQL este simplu de învățat și de utilizat în comparație cu alte programe de gestionare a bazelor de date; de exemplu, comanda exit este simplă: ”exit” sau ”quit”.

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

MySQL este un server de baze de date care este extrem de rapid, fiabil și simplu de utilizat. A fost creat pentru a gestiona baze de date uriașe mult mai rapid decât alternativele anterioare.

Software-ul de baze de date MySQL este un sistem client/server care include un server MySQL cu mai multe fire și o varietate de aplicații client și biblioteci, precum și instrumente de administrare și o varietate de interfețe de programare a aplicațiilor.

5.4 Python

Python este un limbaj de programare dinamic semantic, de nivel înalt, orientat pe obiecte și interpretat. Structurile sale de date integrate de nivel înalt, împreună cu tastarea dinamică și legarea dinamică, îl fac ideal pentru crearea rapidă de aplicații, precum și pentru utilizarea ca limbaj de scripting sau ca punct de legătură între componentele existente. Sintaxa de bază a limbajului Python, ușor de învățat, acordă prioritate lizibilității, reducând costurile de întreținere a software-ului. Modulele și pachetele sunt suportate de Python, ceea ce facilitează modularitatea programelor și reutilizarea codului. Interpretorul Python și biblioteca standard extinsă pot fi descărcate și distribuite gratuit în formă sursă sau binară pentru toate platformele majore.

Python este popular în rândul programatorilor datorită productivității îmbunătățite pe care o oferă. Ciclul editare-testare-depanare este foarte rapid, deoarece nu există o fază de compilare. Depanarea scripturilor Python este simplă: o problemă de segmentare nu este niciodată cauzată de o greșeală sau de o intrare incorectă. În schimb, atunci când interpretorul găsește o greșeală, aruncă o excepție. Interpretul produce o urmă de stivă dacă aplicația nu reușește să detecteze eroarea. Se pot verifica variabilele locale și globale, se pot evalua expresii arbitrare, se pot crea puncte de întrerupere, se poate parcurge codul rând pe rând și așa mai departe cu un depanator la nivel de sursă. Depanatorul este construit în Python, demonstrând capacitățile introspective ale limbajului. Pe de altă parte, adăugarea câtorva comenzi de tastare la codul sursă este frecvent cea mai rapidă metodă de depanare a unui program: ciclul rapid editare-testare-depanare face ca această tehnică de bază să fie foarte eficientă.

5.5 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 abstractizare a bazei de date, validarea formularelor sau orice alte componente în care bibliotecile 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 formularelor, gestionarea încărcărilor, diverse tehnologii de autentificare deschise și mai multe instrumente comune legate de cadru.

Capitolul 6

Capitolul 4: Tehnologiile de pe Front-end

6.1 HTML

HTML, sau HyperText Markup Language, este un limbaj de marcare standard pentru texte care sunt destinate a fi vizualizate într-un browser web. Foile de stil în cascadă (Cascading Style Sheets - CSS) și limbajele de programare, cum ar fi JavaScript, pot ajuta în acest sens.

Browserele web acceptă documente HTML de pe un server web sau fișiere stocate local și le convertesc în pagini web multimedia. Inițial, HTML a furnizat indicii pentru aspectul documentului și a descris în mod logic structura unei pagini web.

Etichete precum ¡image /¿ și ¡input /¿ inserează imediat materiale în pagină. Alte etichete, cum ar fi ¡p¿, înconjoară și oferă informații despre conținutul documentului și pot cuprinde subelemente, cum ar fi alte etichete. Etichetele HTML nu sunt afișate de browsere, dar sunt folosite pentru a înțelege conținutul paginii.

HTML permite limbajelor de scripting, cum ar fi JavaScript, să încorporeze programe care modifică comportamentul și conținutul paginilor online. CSS determină aspectul și aspectul materialului. Din 1997, World Wide Web Consortium (W3C), care gestiona standardele HTML și care acum gestionează standardele CSS, a recomandat utilizarea CSS în locul HTML-ului de prezentare explicit.

6.2 CSS

CSS (Cascading Style Sheets) este un limbaj de foi de stil pentru a specifica aspectul unui document scris într-un limbaj de marcare precum HTML. Alături de HTML și JavaScript, CSS este o componentă cheie a paginilor web.

CSS permite separarea prezentării și a conținutului, inclusiv a aspectului, cu-

lorilor și fonturilor. Această separare poate îmbunătăți accesibilitatea conținutului și poate oferi mai multă libertate și control în definirea caracteristicilor de prezentare.Mai multe pagini web pot partaja formatarea prin definirea CSS corespunzătoare într-un fișier .css separat, ceea ce reduce la minimum complexitatea și duplicarea conținutului structural, permițând în același timp ca fișierul .css să fie memorat în memoria cache pentru a îmbunătăți performanța de încărcare a paginilor pe site-urile care partajează fișierul și formatarea acestuia.Posibilitatea de a afișa aceeași pagină de marcaje în mai multe stiluri pentru tehnici de redare distincte, cum ar fi pe ecran, pe hârtie, prin voce (prin intermediul unui browser bazat pe vorbire sau al unui cititor de ecran) și pe dispozitive tactile bazate pe Braille, este, de asemenea, posibilă prin separarea formatării și a conținutului. În cazul în care materialul este accesibil pe un dispozitiv mobil, CSS oferă reguli de formatare alternativă.

Denumirea de cascadă provine de la schema de prioritate specificată pentru a determina ce regulă de stil se aplică dacă mai multe reguli se potrivesc unui anumit element. Această schemă de prioritate în cascadă este previzibilă. Consorțiul World Wide Web menține standardele CSS (W3C). RFC 2318 specifică tipul de media Internet text/css (tip MIME) pentru utilizare cu CSS (martie 1998). Pentru documentele CSS, W3C oferă un serviciu gratuit de validare CSS. În afară de HTML, alte limbaje de marcare acceptă utilizarea CSS, inclusiv XHTML, XML simplu, SVG și XUL.

6.3 React

Considerat un JavaScript framework ca si Angular sau Vue.js,de fapt React este o biblioteca source. Cel mai mult este folosit pentru proiecte mari deoarece modul de lucru cu componente ajuta la intretinerea mai usoara a aplicatiei, dar de multe ori este foosit is la aplicatii cu osingura pagina. Acesta a fost implementat imediat în fluxul de știri Facebook în 2011, după ce a fost creat de Jordan Walke, un dezvoltator de software de la Facebook. Povestea continuă un an mai târziu, când Instagram, deținută de Facebook, a introdus React. Această colecție alimentează în prezent sute de mii (dacă nu chiar milioane) de site-uri web, iar multe altele sunt lansate în fiecare zi. Această colecție alimentează în prezent sute de mii (dacă nu chiar milioane) de site-uri web, multe altele fiind lansate în fiecare zi. De fapt, odată cu lansarea React, adoptarea unor biblioteci JavaScript mici, dar puternice, a crescut vertiginos. Utilizatorii cer din ce în ce mai mult site-uri web mai rapide și mai dinamice, în timp ce dezvoltatorii optează pentru configurații contemporane, flexibile, care nu includ un număr mare de linii de cod. Pentru multe persoane, ReactJS este alegerea evidentă. Să trecem în revistă motivele cheie pentru care folosim React pentru a explica de ce. React este o bibliotecă open source, în ciuda faptului că este clasificat ca un cadru JavaScript precum Angular sau Vue.js. Este utilizat în principal pentru

aplicații cu o singură pagină și interfețe web mari și sofisticate.

Simplitatea lui ReactJS este poate primul motiv pentru care atât de mulți oameni îl folosesc în proiectele lor. Deoarece React este o bibliotecă JavaScript, dezvoltatorilor care sunt deja familiarizați cu funcțiile JS le va fi mai ușor să înceapă cu ReactJS. Dezvoltatorii folosesc această bibliotecă pentru a crea interfețe folosind o sintaxă asemănătoare cu cea din HTML, numită JSX. Codul HTML și CSS este generat ca o consecință. API-ul React este simplu, dar puternic, iar învățarea câtorva metode fundamentale este tot ce aveți nevoie pentru a începe. Atunci când doriți să utilizați React alături de cadre JS suplimentare, cum ar fi Redux, Material UI sau Enzyme, există o mică curbă de învățare. Deși astfel de biblioteci nu fac parte din stiva React, ele oferă funcționalități suplimentare și facilitează manipularea componentelor React. Majoritatea bibliotecilor populare sunt complet documentate și nu ar trebui să ofere dificultăți niciunui dezvoltator.

Pentru a dezvolta interfețe utilizator, React JS utilizează tehnologiile JSX și Virtual DOM. Dezvoltatorii pot realiza aplicații pentru majoritatea platformelor și, deoarece pot vedea imediat efectele codului lor, pot vedea și înțelege mai bine ceea ce fac. React JS utilizează un limbaj cunoscut sub numele de JSX. Acesta este un plugin React care vă permite să amestecați HTML alături de JavaScript, oferind codului dvs. o mai mare flexibilitate. Majoritatea browserelor actuale sunt compatibile cu React, permițând dezvoltatorilor să își actualizeze și să își testeze DOM-ul pe mai multe platforme. Document Object Model (DOM) este o interfață de program de aplicație care este prescurtarea pentru DOM virtual (API).Acesta permite programelor să citească conținutul oricărui site web și să îl modifice în funcție de gusturile și cerințele programatorului. Orice site web care nu utilizează React JS își modifică și schimbă DOM-ul folosind HTML. React JS poate avea un DOM virtual, care este o replică a DOM-ului original utilizat de aplicație sau de site-ul web, datorită implementării JSX. Diferența majoră dintre DOM și DOM virtual este că primul dezvăluie modificările doar după ce pagina a fost reîmprospătată, în timp ce al doilea face acest lucru în timp real, fără a necesita o reîncărcare.

6.4 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 lăudat 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 Eclipsă, 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 limbaj de programare dezvoltat și întreținut 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 transcompilează 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 partea 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 7

Specificarile aplicatiei

Pe partea de back end sunt prezente două servere, unul scris în JavaScript cu ajutorul framework-ului Node js care servește partea de front-end cu toate datele de care are nevoie și un server scris în Python cu ajutorul framework-ului Flask cu ajutorul căruia aflu filme recomandate pentru un film sau o lista de filme.

7.1 Baza de date

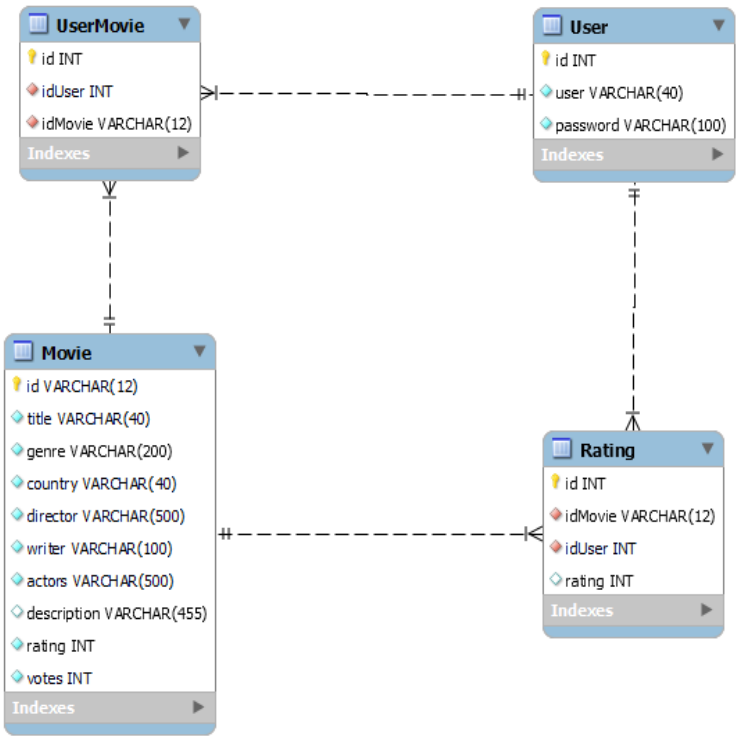

Figura 7.1: Diagrama baza de date.

Un lucru foarte important în dezvoltarea aplicației a fost alegerea bazei de date, am optat pentru pentru MySql versiunea 8.0. În tabelul de User voi salva datele

de inregisrare a unui utilizator , în tabelul Movie sunt prezente datele care descriu un film prin coloanele: title,genre,country,directors,writers,actors,description,votes și rating, în tabelul Rating salvez datele unui rating dat de un user pe laga valoare ratingului salvez și id-urile de la film și user pentru a ști cine da rating și la ce film totodată pe partea de back-end fac și o actualizare a rating ului și numărului de voturi din tabelul Movie asta pentru o a face aplicația să se miște mai rapid, iar în tabelul UserMovie salvez id ul de la un user și id ul de la un film reprezentând un film pe care utilizatorul l-a vizualizat.

7.2 Server Node Js

Serverul de Node Js este folosit pentru a se face înregistrare, login, returanrea tuturor filmelor care conțîn un numit sub titlu, adăugarea unui film la lista utilizatorului, returnarea a tuturor filmelor unui utilizator și adăugarea de rating a unui film. Cu ajutorul pachetului Express se stabilesc rutele și se pornesc serverul. Când se face un request pe o ruta, se apelează funcția asignată din Service, care are rol de a modela datele in funcție de cum avem nevoie. Pentru a comunica cu baza de date ne folosim de Repository, acesta face legătură cu baza de date.

Standardul REST („representational state transfer”) a fost creat pentru a ghida design-ul și arhitectură 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 conțîn informații importante de identificare referitoare la metadatele cererii, autorizarea, identificatorul uniform al resurselor (URI), cache, cookie-uri. 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

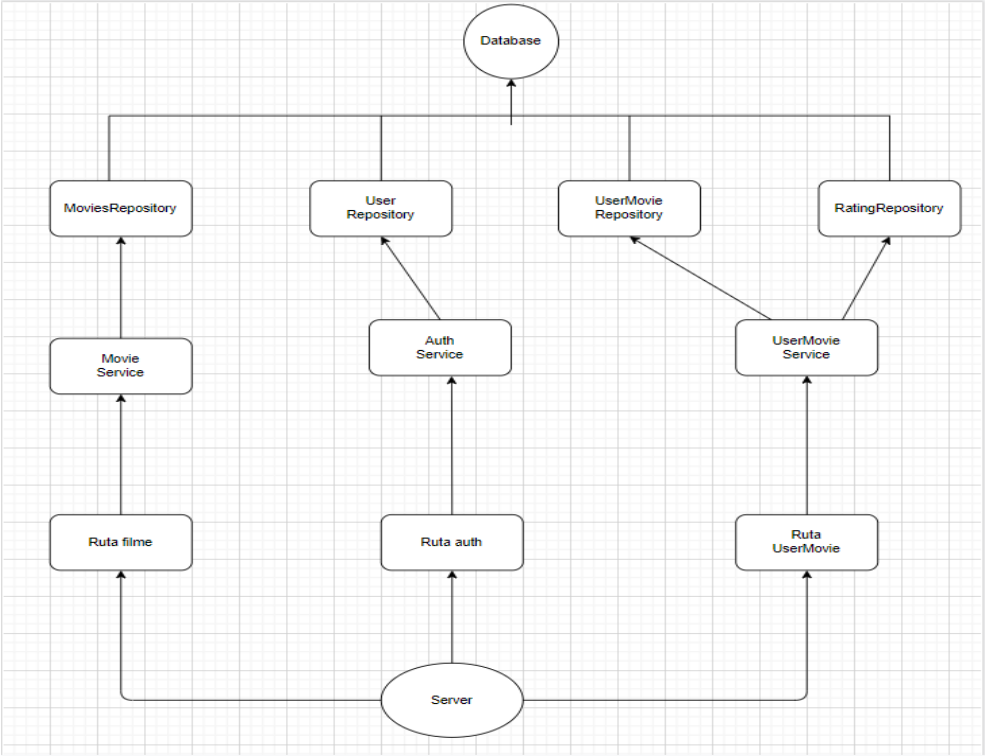

Figura 7.2: Arhitectura server Node Js .

7.2.1 Inregistrare

Procesul de înregistrare este unul simplu, pașii fiind următorii:

1. Se trimit datele către server
2. Se validează datele primite
3. Dacă datele sunt valide se verifică dacă numele utilizatorului se află în baza de date deoarece numele trebuie să fie unic pentru fiecare utiliztor
4. Dacă totul este bine până la acest pas se face has la parolă pentru o mai bună securitate

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

Figura 7.3: Has la parola.

5. Următorul pas este de a apela functția din repositoy care adaugă un user in 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 7.4: Adaugare user in baza de date.

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

7.2.2 Autentificare

Pașii pentru procesul de autentificare sunt următorii

1. Se trimit datele catre server
2. Se valideaza datele primite
3. Se verifica daca username-ul se afla in 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 7.5: Cautare dupa username in baza de date

4. Daca se găsește user-ul in baza de date se verifică parola trimisă in requst cu cea din baza de date apoi se trimite raspunsul in functie de potrivirea celor doua parole, daca rapunsul 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 incorect" });
response.status(200).send({ status: "ok", token: id });
```

Figura 7.6: Verificare parola si trimitere raspuns

7.2.3 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 7.7: Functia din repository care cauta filmele

7.2.4 Adaugarea unui film la lista

Pașii pentru adaugare sunt:

1. Exista o ruta care trebuie sa primeasca intr-un body id-ul user-ului si id-ul unui film
2. Se apeleaza functia din service respectiva rutei
3. Se adauga cele doua id uri in baza de date
4. In cazul unei erori se returneaza codul 400, iar daca totul s-a executat asa cum trebuie se returneaza codul 200 si statusul true.

```
const { idUser, idMovie } = req.body;
try {
  await addUserMovie(idUser, idMovie);
  res.status(200).send({ status: true });
} catch (err) {
  console.log(err.message);
  res.status(400).send({ err: err.message });
}
```

Figura 7.8: Functia pentru ruta de adaugat filme

5. In cazul in care se doreste stergerea filmului respectiv din lista se apeleaza ruta ”/del” cu cele 2 id-uri

7.2.5 Acordare rating

Pașii pentru adaugare sunt:

1. Exista o ruta care trebuie sa primeasca intr-un body id-ul user-ului , nota data si id-ul filmului
2. Se apeleaza functia din service respectiva rutei
3. Se adauga cele doua id uri si nota in baza de date
4. In baza de date cu filme se modifica la id-ul filmului: se incrementeaza cu unu numarul de votanti si se aduna la rating nota data, acest lucru ajuta la calcularea mediei generale a filmului fara a mai face un select cu toate notele unui film.

5. In cazul unei erori se returneaza codul 400, iar daca totul s-a executat asa cum trebuie se returneaza codul 200 si statusul true.

```
const command = `INSERT INTO Rating( idMovie, idUser, rating) VALUES ('${idMovies}',${idUser},${rating})`;
const commandUp = `Update Movie Set rating= rating + ${rating} , votes=votes+1 Where id = '${idMovies}'`;
const prom = new Promise((resolve, reject) => {
  pool.query(commandUp, (err, data) => {
    if (err) throw new Error(err);
  });
  pool.query(command, (err, data) => {
    if (err) throw Error;
    resolve(data);
  });
});
return prom;
```

Figura 7.9: Functia pentru ruta de adaugat filme

6. In cazul in care se doreste stergerea filmului respectiv din lista se apeleaza ruta ”/del” cu cele 2 id-uri

7.3 Server Flask

Seveverul scris cu ajutorul framework-ului Flask este cel care face recomandari pentru un film sau pentru o lista de filme. La pornirea acestui server trebuie sa se stepte un anumit timp deoarece la pornire acesta ia toate datele de la baza de date apoi le proceseaza si calculeaza matricea de scoruri, face toate acestea pentru ca atunci cand un client cere recomandari pentru un film raspunsul sa fie minim si sa nu fie nevoie sa faca toate aceste calcule. Exista un dezanjantaj la aceasta metoda, daca intervin modificari la baza de date in acest timp nu vor influenta scorurile, de exemplu daca un film va fi votat cu multe voturi nu se regasi in scorul final sau daca un film este sters din baza de date algoritmul din acest server inca il poate recomanda. Exista totusi variante pentru aceste probleme, se poate creea o functie care sa reincarce datele iar la apelarea aceesteia sa se recalculeze matricele de scoruri, iar o a doua varianta putin mai neconvetionala se poate opri si reporni server ul acesta recalculand scorurile. Primul lucru pe care il face serverul dupa citire este procesarea datelor, dupa care creeaza sacul de cuvinte si vectorii care reprezinta filmele, urmatorul pas fiind calcularea scorului.Server-ul are doua rute una in care returneaza recomandaripentru un film si o ruta care returneza recomandari pentru o lista de filme.

Pentru a face recomandari pentru un film algoritumul ia din linia din matricea de scoruri cele mai mari n scoruri, n reprezentand numarul de filme pe care algoritmul trebuie sa il recomande, apoi returneaza o lista cu detaliile filmelor. Atunci cand se cere recomandari pentru o lista de filme algoritmul ia din matricea de scoruri mai

multe filme pentru fiecare film, la final sortandu-le dupa scor si returneaza n filme cu cele mai bune scoruri.

```
stop = set(stopwords.words('english'))
df['keywords'] = df['description'].str.split().apply(lambda x: [item for item in x if item not in stop])
df['keywords'] = df['keywords'].str.join(' ')
df['genre'] = df['genre'].str.replace(',', '')
df['actors'] = df['actors'].str.replace(',', '')
df['writer'] = df['writer'].str.replace(',', '')
df['director'] = df['director'].str.replace(',', '')
df['keywords'] = df['keywords'].str.lower()
```

Figura 7.10: Prelucrarea datelor

```
cv = CountVectorizer()
count_matrix = cv.fit_transform(df1['bow'])
cos_sim = cosine_similarity(count_matrix)

for raw in cos_sim:
    index=0
    for x in raw:
        if df1['votes'][index] !="0":
            x=x+int(df1['votes'][index])/(int(df1['votes'][index])*100)
        index+=1
    break
```

Figura 7.11: Calcularea matricii de scoruri

7.4 Front-end

Front-endul este reprezentat de un site web scris cu ajutorul framework-ului React Js. Proiectul este impartit pe pagini care la randul lor contin diferite componente. Aceasta metoda a componentelor este foarte utila atunci cand se refoloseste cod, este ca si o functie care este scrisa o singura data dar este apelata in mai multe locuri. Exista doua tipuri de pagini, publice si private, cele publice pot fi accesate de un user fara a fi neoie de autentificare, iar cele publice necesita o autentificare. Stilizarea site-ului web se face cu ajutorul CSS-ului in doua moduri, una fiin prin importarea fisierelor de CSS in componente si pagini si una cu ajutorul injectarii in componente cu ajutorul JavaScript. Pentru a putea naviga intre pagini exista un

router in care se specifica pentru fiecare ruta din aplicatie pagina care trebuie sa fie afisata si daca aceasta este publica sau privata, in cazul in care este privata iar user-ul nu s-a autentificat acesta va fi redirectionat la una din rutele publice. Acesta respectă standardul REST și se folosește de librăria axios pentru a face cereri către server.

7.4.1 Autentificare si inregistrare

Pașii sunt:

1. Atunci cand utilizatorul acceseaza site-ul web este redirectionat la pagina de autentificare unde trebuie sa introduca userul si parola
2. Daca nu are un cont poate apasa re butonul de inregistrare unde va fi redirectionat catre pagina de inregistrare:
   (a) Dupa completarea folmularului de inregistrare si pasarea butonului datele vor fi validate
   (b) Urmatorul pas este de a trimite datele la server
   (c) Daca se primeste status 200 userv-ul va fi redirectionat catre pagina de autentificare
   (d) Daca se primeste alt status se va afisa eroarea, iar pasii de inregistrare trebuie repetati pana se introduc date valide
3. Dupa ce toate datele de autentificare au fost introduse ele se trimit la server, iar daca datele sunt corecte se primeste ca si raspuns token-ul care fa fi salvat in memorie si utilizatorul va fi redirectionat catre pagina principala
4. Daca de la server se primeste raspuns negativ se va afisa eroarea, iar pasii trebuie repetati.

7.4.2 Cautare filme

Pașii sunt:

1. Dupa ce utilizatorul s-a autentificat acesta este redirectionat catre pagina principala unde poate cauta filme dupa titlu
2. Dupa ce introduce titlul pe care acesta il cata si apasa pe buton vor aparea toate filmele care in titlu contin sirul introdus de utilizator
3. Fiecare film are doua optiuni, mai multe detalii si adauga la lista

4. Daca se alege optiunea de mai multe detalii acesta va fi redirectionat catre pagina unde poate vedea detaliile explicite ale filmului si poate cel mai important se face si o recomandare de patru filme pentru filmul respectiv.Tot pe aceasta pagina se poate da un rating filmului astfel influentand rating-ul filmului.

5. Daca se alege optiunea de adaugare la lista acesta se va adauga la lista utilizatorului, vizibila pe alta pagina a aplicatiei, iar aceasta optiune va disparea

7.4.3 Lista cu filme

1. Pe aceasta pagina se incarca toate filmele utilizatorului cu doua optiuni, se poate sterge filmul din lista sau se poate face redirectionare catre pagina cu detalii explicite ale filmului.

2. Tot pe aceasta pagina se poate creea o lista pentru care se poate face recomandari de filme:

   (a) Se pasa pe butonul cu plus, iar titlul filmului va aparea intr-o lista pozitionata in partea de sus a pagini, filmul poate fi si deselectat deoarece dupa adaugare va aparea un buton pentru aceasta optiune

   (b) Dupa ce toate filmele au fost selectate se apasa pe buton

   (c) Se vor afisa toate filmele recomandate pentru lista de filme aleasa

Capitolul 8

Concluzii

8.1 Ce s-a obtinut

Scopul aplicatiei a fost dezvoltarea unui algoritm care sa recomande filme si crearea unei aplicatii care sa ajute un utilizator sa isi poata nota toate filmele relevante pentru el, de exemplu filmele pe care el le-a vizualizat, totodata aceasta platforma sa recomande sa indrume utilizatorul spre alte filme facandu-i recomandari. Lucrarea si-a atins aceste scopuri puse la ineputul lucrari atat algoritmic cat si business. Obiectivul de business nu a fost unul principal mai mult necesar fiind pentru a putea interactiona cu algoritmul de recomandari.

8.2 Ce s-a obtinut

Un prim lucru ar fi experienta utilizatorului cu aplicatia dezvoltata, design-ul nu este cel mai reusit nefiin luati in calcul oameni cu dizabilitati nepunanduse accent pe utilizabilitatea interfetei.Tot odata aplicatia ar putea fi dezvoltata pentru a permite utilizatorului sa isi gestioneze mai bine filmele.

Pe partea de algoritm, acesta functioneaza si de multe ori face recomandari foarte utile,dar sunt momente cad da si rateuri cauza putand fiind multitudinea de cuvinte pe care filmele le au. O mai buna prelucrare a datelor ar ajuta algoritmul sa fie mult mai efficient.