MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

**UNIVERSITATEA TEHNICĂ**
DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**
**DEPARTAMENTUL CALCULATOARE**

# Sistem de gestiune și recomandare a filmelor

LUCRARE DE LICENŢĂ

Absolvent: **Diana-Ioana GIURGIU**
Conducător ştiinţific: **Asist. Ing. Cosmina IVAN**

**2020**

# Cuprins

# Capitolul 1

# Introducere - Contextul proiectului

Proiectul își propune crearea și implementarea unei aplicații web ce permite utilizatorului să acceseze informații despre filme și seriale și să primească recomandări pe baza preferințelor.

Acest capitol își propune să realizeze o scurtă introducere în tematicile abordate de către această lucrare și să ofere contextul și motivația din spatele realizării acestei lucrări.

## 1.1 Contextul proiectului

Tehnologia a avut un impact uriaș asupra lumii, în special a industriei cinematografice, pe măsură ce aceasta avansează, devine din ce în ce mai realistă și diversificată. O dată cu evoluția tehnologiei, industria filmului a încercat să țină pasul cu aceasta îmbunătățind conținutul creat recent și promovând filmele create unui public cât mai larg. Însă, în toate aceste noi apariții de conținut este foarte dificil pentru public să țină pasul și să aleagă ce va vizualiza în continuare. Mai multe platforme au început să creeze recomandări pe baza preferințelor anterioare ale unui utilizator pentru a-l încuraja să consume în continuare conținut. Dar, o dată cu apariția mai multor alternative la anumite soluții, tot mai mulți oameni se confruntă cu o dificultate în găsirea unui conținut potrivit. Aici sistemele de recomandare au venit în ajutorul indivizilor și au încurajat realizarea deciziilor.

Sistemele de recomandare au adus un uriaș avantaj, nu doar industriei filmului și divertismentului ci și altor domenii precum comerțului încurajând consumerismul individului dar cel mai mare avantaj l-au avut oamenii de rând care au avut oportunitatea de a găsi ceva pe placul lor încă de pe prima pagină pe care au intrat ceea ce i-a scutit din timpul pierdut în căutare.

Tot mai multe platforme media au început să implementeze sisteme de recomandare pentru a încuraja utilizatori să vizioneze mai mult. Netflix, este o platformă de vizionare filme și seriale pe bază de membership, care are un sistem de recoman-

dare cu o acuratețe ridicată, ce oferă recomandări concise cu posibilitatea de a vizualiza un scurt trailer înainte de a viziona filmul. De asemenea, Netflix utilizează in sistemul de recomandare atât tehnici **Item based** cât și **User based**, obținând astfel un sistem dinamic.

## 1.2 Motivația proiectului

Motivația proiectului provine din dorința de a cunoaste mai multe despre aceste tipuri de sisteme de recomandare si modul în care acestea sunt realizate. Am fost interesată constant de modul în care un sistem se poate implementa cât și de acuratețea cu care oferă sugestii.

Ca și utilizator al platformelor media ce au la bază mecanisme de recomandare, de multe ori am încercat să identific mecanismul de funcționare studiind diverse resurse bibliografice. Un exemplu concret provine de la Spotify[1], o aplicație de streaming pentru muzică și podcasturi populară, ce se bazează pe un sistem de publish-subscribe unde orice device ce utilizează sistemul este marcat ca subscriber iar serviciile ce rulează în cadrul aplicației sunt Publisher. Utilizatorii primesc informații în funcție de preferințele lor sau ale prietenilor din listă. Acest sistem prezintă și o funcționalitate de ”Discover Weekly” care are rolul de a recomanda utilizatorului noi melodii ce i-ar putea plăcea, conform [2] recomandările sunt personalizate dar lipsite de context, bazându-se pe comportamentul observat al utilizatorilor. Nu pot spune că toate recomandările au fost pe baza preferințelor mele, dar nu am considerat că au fost alese special pentru mine ci pentru o gamă largă de utilizatori cu preferințe asemănătoare.

Astfel, am dorit să realizez propriul sistem de recomandare pentru a vedea modul de operare și cum este posibilă implementarea acestuia. Dar am dorit să realizez și un sistem de gestiune a filmelor văzute de utilizator.

## 1.3 Conținutul lucrării

În acest subcapitol se prezintă conținutul lucrării pe capitole și descrierea sumară a acestora.

- Capitolul 1: Introducere -În acest capitol se realizează o prezentare a contextului proiectului și a motivației ce a dus la studierea acestui subiect.
- Capitolul 2: Obiectivele proiectului- În acest capitol sunt prezentate obiectivele proiectului ce s-au dorit a fi atinse la finalul proiectului
- Capitolul 3: Studiu Bibliografic- În acest capitol sunt prezentate noțiuni de ML și modele de sisteme de recomandare existente ce au ajutat în oferirea unui ghid pentru realizarea licenței.

- Capitolul 4: Analiza si Fundamentare Teoretica- Acest capitol prezintă arhitectura care stă la baza realizării sistemului, cerințele funcționale și non-funcționale de la care pornește proiectul împreună cu o descriere scurtă a tehnologiile utilizate în dezvoltare.
- Capitolul 5: Proiectare de detaliu si Implementare se prezintă implementarea proiectului pas cu pas și a algoritmilor mai importanți ce au fost utilizați și implementați.
- Capitolul 6: Testare și Validare- În acest capitol se prezintă tehnicile care au ajutat la testarea aplicației și frameworkurile folosite pentru validarea cerintelor.
- Capitolul 7: Manual de Utilizare și Instalare- prezintă resursele necesare pentru instalarea sistemului și pornirea acestuia pentru funcționare. De asemena, se precizează și modul în care acesta va fi utilizat.
- Capitolul 8: Concluzii- se vor analiza rezultatele obținute și se vor descrie noi extensii ce pot fi implementate pentru a îmbunătății sistemul.

# Capitolul 2

# Obiectivele Proiectului

Acest capitol prezintă obiectivele principale care stau la baza proiectului și obiectivele secundare care au ajutat la implementarea acestuia.

## 2.1 Obiectivul principal

Principalul scop al acestui proiect îl constituie crearea unui sistem de recomandare pentru a oferi alternative posibile pentru a decide următorul film ce va fi vizionat. Sistemul este o aplicație web ce va avea utilizatori ce se pot autentifica pentru a gestiona filmele văzute, pentru a obține recomandări și pentru a-și exprima aprecierile față de un film vizionat.

## 2.2 Obiective secundare

Obiectivele ce se urmăresc pentru a fi atinse în cadrul acestui proiect sunt următoarele:

- Studiul mai multor tipuri de baze de date NoSQL cu scopul alegerii uneia pentru realizarea implementării.
- Studiul și analiza mai multor tipuri de sisteme de recomandare ce ar putea fi utilizate pentru realizarea acestuia și a sistemelor ce au fost deja implementate pentru a se realiza o alegere potrivită asupra unei soluții existente a unui algoritm de recomandare concret.
- Studiul mai multor frameworkuri JavaScript pentru alegerea unuia în procesul dezvoltării și implementării aplicației.
- Construirea unui sistem de gestiune web ca prototip ilustrativ pe baza alegerilor realizate.

# Capitolul 3

# Studiu Bibliografic

În acest capitol vor fi prezentate și analizate câteva sisteme reprezentative pentru oferirea informaţiilor cu privire la filme sau seriale şi recomandarea acestora. Aceste sisteme reprezintă modele pentru realizarea proiectului propus apartinand aceleiaşi teme.

Marea majoritate a sistemelor industriale[3] cunoscute conțin sisteme de recomandare realizate pe baza algoritmilor de Machine Learning existenți dar propun soluții hibride pentru realizarea acestora. Sistemele mai puțin avansate propun recomandări fie printr-un motor mai puțin avansat utilizând similaritățile utilizatorilor, fie utilizând similaritățile dintre filme. În continuare se va prezenta conceptul de Machine Learning(ML) și se vor explica mai mulți algoritimi cu scopul de a-l alege pe cel mai adecvat pentru sistemul ce se dorește a fi dezvoltat.

## 3.1 Machine Learning

Această secțiune prezintă conceptul de machine learning și algoritmii care sunt mai specifici problemei acestui proiect. Machine Learning reprezintă un set de tehnici ce integrează algoritmi folosit de unele sisteme pentru a rezolva anumite sarcini fără a avea nevoie de instrucţiuni explicite. Mai pe scurt, Machine Learning se referă la crearea unui model matematic ce poate rezolva o anumită problemă după ce îşi dezvoltă experienţa necesară rezolvării. Alte persoane definesc Machine Learning ca fiind procesul de a învăța calculatoarele să se comporte sau să imite comportamentul oamenilor prin oferirea de informații și realizarea de decizii pe baza acestora [4].

Modul de învățare al algoritmilor de Machine Learning s-a dezvoltat în două ramuri, învățare supervizată și nesupervizată [5] , fiecare având mai multe tehinici, dintre acestea se evidențiază clasificarea și clusterizarea, dintre care se vor prezenta în detaliu algoritmii comparați.

**Învăţare supervizată**

Modul de operare al acestei metode este de a oferi algoritmului date de intrare marcate cu datele corecte de ieşire. Algoritmul învaţă ce este aşteptat ca ieşire pentru un anumit set de date de intrare. Există două mari tipuri de învăţare supervizată:

1. Tipul unul se numește clasificare, este caracterizat de clasificarea datelor de ieșire într-o categorie predefinită (precum: negru, alb, rotund, patrat)

2. Tipul doi se numește regresie, fiind caracterizat de faptul că datele de ieșire reprezintă o variabilă.

**Învăţare nesupervizată**

Modul de operare al acestei metode este de a lăsa sistemul să ”gândească” pe baza asemănărilor din setul de date. Datele nu sunt marcate în nici un fel, iar scopul este ca algoritmul să găsească un tipar. Există două mari tipuri de învăţare ne-supervizată:

1. Primul tip se numește asociere: modelul încearcă să împartă setul de date de intrare în grupuri, după frecvenţa cu care au loc în acelaşi timp .

2. Cel de al-doilea tip se numește clustering: modelul încearcă să împartă setul de date de intrare în grupuri, după asemănările dintre date.

Dintre aceste două tipuri de învățare s-au ales doi dintre cei mai cunoscuți algoritmi pentru a se face o comparație între aceștia, cu scopul de a alege unul pentru dezvoltare. Dar mai întâi se va discuta despre un algoritm utilizat de o gamă largă de motoare de recomandare. Unul dintre cele mai utilizate moduri de dezvoltare al acestui tip de aplicații îl reprezintă Filtrarea Colaborativă.

### 3.1.1 Filtrarea Colaborativă

Multe sisteme de recomandare se bazează pe Filtrarea Colaborativă (Colaborative Filtering) pentru a face sugestii utilizatorului, fiind considerată cea mai concretă şi precisă tehnică de recomandare[6].

Filtrarea Colaborativă se bazează pe ipoteza că persoanele care au un istoric de a oferi aceeaşi notă aceloraşi filme pot avea preferinţe asemănătoare [6].

Spre exemplu, în lucrarea [7] sunt propusw şi analizate mai multe posibităţi de sisteme de recomandare. Netlix este o platforma web, mobile și desktop de streaming a filmelor, serialelor, documentarelor şi desenelor animate. Pe lângă filtrarea colaborativă este prezentată şi filtrarea pe bază de conţinut şi filtrarea hibridă. Măsurarea performanţei este realizată prin rădăcina pătrată a erorii (Root

Mean Square Error, RMSE) [8], unde fiecare eroare este ridicată la pătrat, înainte de a fi adunată cu celelalte, rezultatul final devine astfel o rădăcină pătrată a erorilor.

Majoritatea sistemelor de recomandare folosesc clusterizarea pentru a ameliora problema scalabilităţii şi deoarece au o precizie mai bună comparativ cu celelalte [6].

### 3.1.2 K-Nearest-Neighbors

K-Nearest-Neighbors(KNN) este un algoritm de învățare supervizată ce pleacă de la premiza că grupurile cu interese asemănătoare se află la o distanță apropiată unele de altele. Se începe prin alegerea unui număr de ”vecini”, apoi se calculează distanța dintre punctul referit și celelalte puncte din setul de date, se ordonează acesta și se aleg vecinii[9].

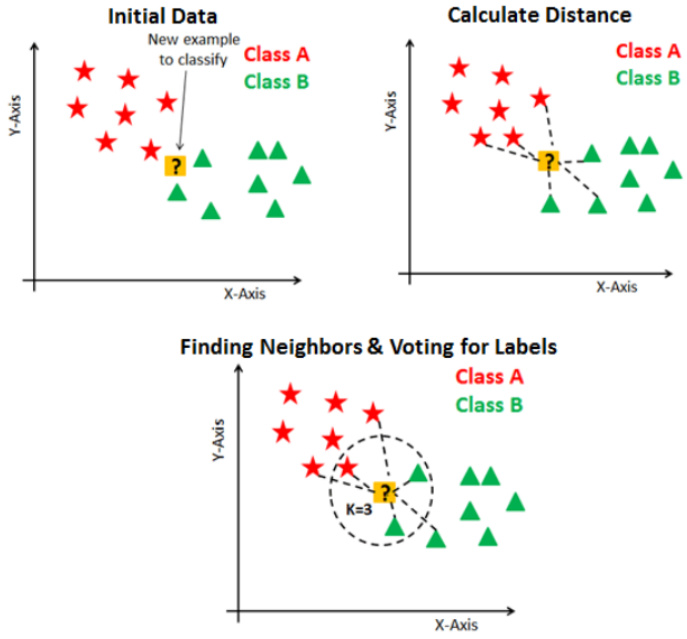


Figura 3.1: Ilustrare algoritm KNN

### 3.1.3 K-means clustering

Clusterizarea are scopul de a partiţiona o mulţime de obiecte în grupuri diferite, unde instanţele dintr-un grup sunt similare într-un anumit sens.

Clusterizarea K-means aparţine învăţării nesupervizate, având scopul de a găsi grupuri în setul de date; aceste grupuri sunt create dacă distanţa pătrată din-

tre centroidul clusterelor şi fiecare punct din cluster este minimă. Acest lucru presupune cunoaşterea numărului de clustere ce se pot forma sau definirea acestora în mod imparţial. Un exemplu concret pentru a demonstra modul de funcționare a acestui algoritm este prezentat în 3.2. Se definește o mulțime de puncte și un număr K(K=3) de clustere ales imparțial:

1. Pentru fiecare x din mulțimea de puncte se va asigna o culoare diferită în funcție de centroid-ul de care este mai apropiat.
2. Se reface centrul fiecărui cluster prin calcularea mediei dintre punctele ce aparțin în acest moment clusterului.
3. Se repetă primii doi pași până nu se mai modifică centroidul.

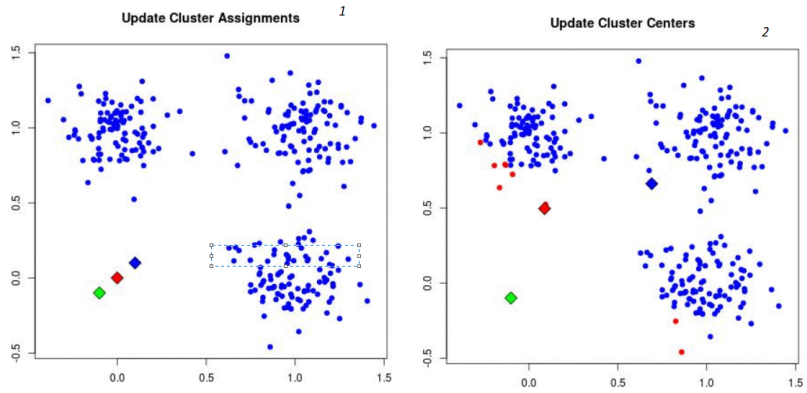


Figura 3.2: Primii 2 pași

La final, după ce nu se mai distanțează centroidul în nici o parte, se vor împărți punctele în 3 clustere fiecare de culori diferite, centroidul aflându-se în mijlocul clusterelor.

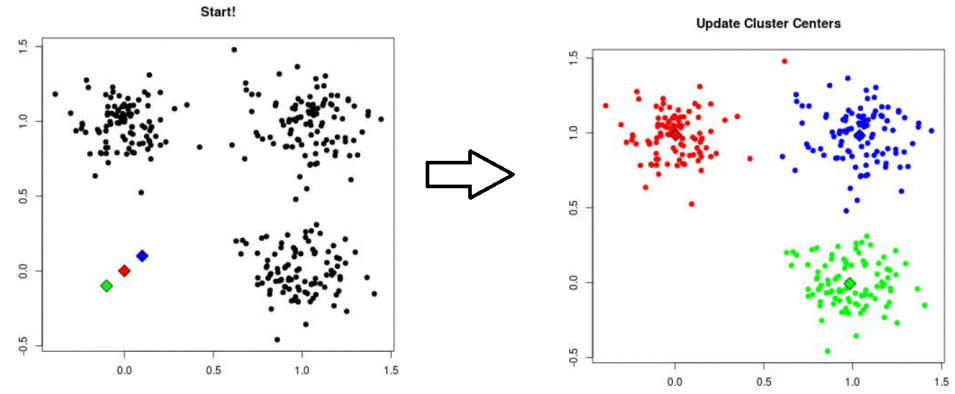


Figura 3.3: Rezultatul final al K-means

## 3.2 Sisteme cu caracteristici similare

Pentru a încuraja utilizatori să gestioneze filmele și serialele au fost create numeroase aplicaţii web care oferă informaţii cu privire la generalităţi despre acestea. În crearea aplicaţiei s-a avut în vedere analiza comparativă prezentată mai jos, a unor aplicaţii cu caracteristici similare pentru extragerea unui set de funcţionalităţi potrivite. După identificarea soluţiilor au fost identificate concepte similare acestor aplicaţii precum:

**Reviews** : reprezintă acordarea unui calificativ, de la 1 la 5, pentru a arăta nivelul de apreciere al filmului de către cinefili. Este marcat în mod intuitiv printr-un număr de 5 stele ce se schimbă la trecerea mouse-ului peste ele.

**Comments**: reprezintă acordarea unui comentariu unui film pentru exprimarea unei opinii personale asupra filmului.

**Recommendations**: reprezintă funcţia de recomandarea a unui film sau serial care se califică preferinţelor utilizatorului.

**Trailer**: reprezintă posibilitatea de afişare a trailerului unui film, dacă există, pe pagina de informaţii despre film.

**NextEp**: reprezintă notificarea utilizatorului de apariţia unui nou episod din serialul favorit.

### 3.2.1 IMDB

IMDB[10] este o bază de date web ce stochează informaţii despre filme, seriale şi jocuri video cât şi despre producţiile de filmare. Majoritatea datelor din baza de date sunt furnizate de colaboratori voluntari. Acesta prezintă şi o secţiune extinsă despre actorii prezenţi în filme şi despre viaţa publică a acestora.

În ceea ce priveşte funcţionalităţile acestei aplicaţii, site-ul permite utilizatorilor să adauge sau să modifice informaţii pe site doar după ce acestea sunt verificate. Utilizatorii înregistraţi își aleg propriul nume de profil, iar cei mai mulţi operează anonim. Utilizatorii înregistraţi au o pagină de profil care arată de cât timp sunt membrii precum şi notele acordate filmelor (în cazul în care utilizatorul decide să le afişeze). Un utilizator are o listă, *Watchlist*, unde marchează care sunt filmele vizionate de acesta şi există posibilitatea configurării unei noi liste ce poate cuprinde şi actori. Aceste liste sunt setate, by default , în privat dar pot fi partajate public pe profilul utilizatorului la cerere.

De asemenea, site-ul are un parteneriat cu *Primevideo* prin care poate indica utilizatorului un loc de vizionare a filmului. Primevideo reprezintă un serviciu de streaming a filmelor disponibile membrilor ce deţin un membership Amazon Prime[11]. Mai mult, IMDB a lansat recent serviciul IMDB Pro[12] ce funcționează pe bază de membership și prezintă funcționalități mai avansate față de un utilizator normal.

### 3.2.2 Rotten Tomatoes

Rotten Tomatoes[13] este o platformă web ce conţine informaţii despre filme şi seriale. Un utilizator poate să își creeze un cont prin completarea informațiilor necesare (email si parola) sau conectarea prin Facebook, ceea ce înseamnă că aplicaţia va avea acces la datele existente pe facebook precum: nume, prenume, poza, etc.

Deosebirea față de celelalte sisteme este aceea că propune un sistem de review diferit numit *Tomatometer* unde scorul reprezintă procentul de recenzii critice profesionale care sunt pozitive pentru un anumit film sau emisiune de televiziune. Pentru o reprezentare mai intuitivă şi mai ilustrativă a acestui scor a fost ales ca sistem de reprezentare afișarea unei roșii sau a unei pete verzi lângă scorul oferit, cum se poate observa şi în figura de mai jos 3.4.

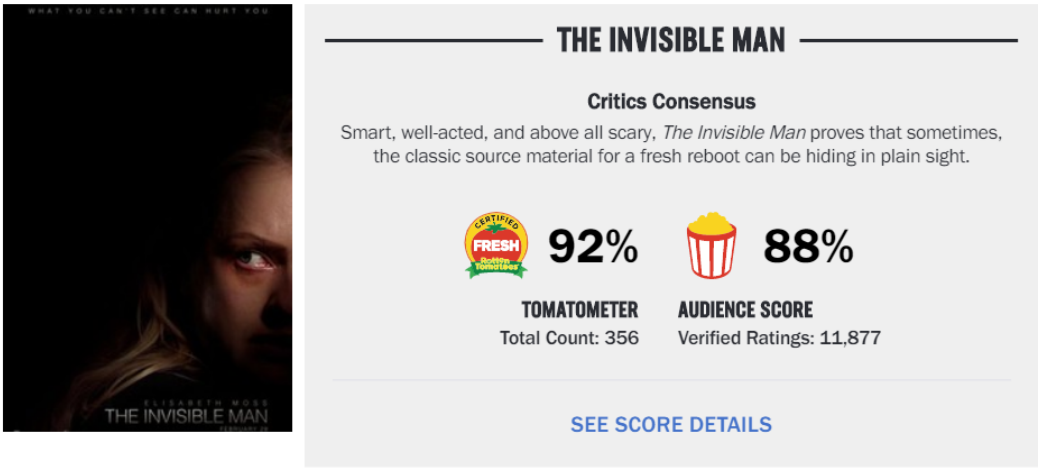


Figura 3.4: Reprezentarea scorului în RT

Raportat la cerinţele identificate anterior ca fiind valoroase pentru un sistem acesta conţine funcţionalitățile de afişare a unui trailer, acordarea de reviews şi comentarii. Dar s-a constatat că nu implementează și suportul pentru oferirea de recomandări utilizatorilor ci doar posibilitatea de a vedea comentarii și scorul obținut nu şi posibilitatea oferirii de recomandări pentru fiecare utilizator, utilizatorul are posibilitatea de a vedea comentariile şi scorul obținut de unele filme.

### 3.2.3 Flixboss

Flixboss[14] este o platformă web ce acţionează ca un motor de căutare rapid a filmelor existente pe Netflix şi sortate în funcţie de review de pe IMDB şi din locaţia utilizatorului. Acesta acţionează doar pe filmele existente pe Netflix şi conţine

informaţii despre ce va apărea nou pe Netflix şi ce va fi șters. Utilizatorul are posibilitatea de a oferi un review cât şi un commentariu pentru a îşi exprima aprecierea faţă de un film sau serial.

În referinţă cu funcţionalităţile prezentate mai sus, acesta conţine funcţionalităţile de afişare a unui trailer, acordarea de review şi comentarii, dar nu şi posibilitatea oferirii de recomandări pentru fiecare utilizator, ci utilizatorul are de a vedea comentariile şi scorul obţinut de unele filme, dar toate acestea sunt structurate doar pe filmele sau serialele existente pe Netflix şi care există în regiunea utilizatorului.

### 3.2.4 Trakt.tv

Trakt.tv[15] este o platformă atât web cât şi mobile care gestionează filmele şi serialele văzute, prin integrarea cu media center-ul existent în posesia utilizatorului pentru realizarea scrobbling-ului. Scrobbling reprezintă funcţionalitatea de a memora ce vizualizează utilizatorul, în media center, în mod automat fără fi necesar input de la utlizator[16].

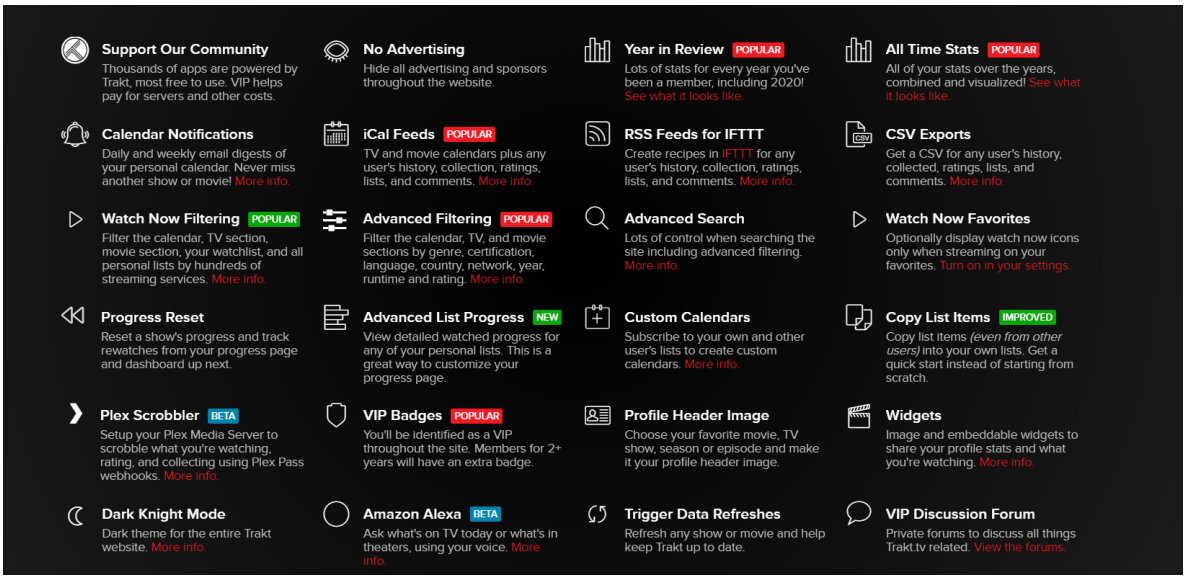


Figura 3.5: Lista de funcţionalităţi Trakt.tv

Utilizatorul are posibilitatea de a realiza o filtrare avansată, a primi push notifications în cazul apariţiei unui nou episod, verifica calendarul pentru gestiunea episoadelor sau filmelor ce vor apărea, a integra cu diferite sisteme media precum Amazon Alexa pentru aflarea celor mai noi informaţii în legătură cu premierele recente [17], oferire de suport video pentru vizionarea filmului pe baza locaţiei utilizatorului şi realizarea recomandărilor pe baza istoricului vizionării utilizatorului şi a

istoricului prietenilor acestuia.

Raportat la cerinţele identificate anterior ca fiind valoroase pentru un sistem, acesta conţine aproape toate funcţionalităţile dorite în afara posibilităţii de a vedea un trailer despre film. Un plus adus de Trakt.tv este integrarea cu diferite sisteme de operare printre care se numără IoS, Android și Desktop prin diferite aplicații realizate în parteneriat cu acesta, împreună cu integrarea cu diferite aplicații de streaming precum Netflix, Disney+, Amazon Prime, Horizon Go.

### 3.2.5 Concluzii

În concluzie Trakt.tv, cu toate că este o aplicaţie relativ nouă, este cea care se integrează cel mai bine în diferite sisteme şi pe diferite dispozitive dar nu are o bază de date de sine stătătoare ca IMDB sau RottenTomatoes. Așa cum se observă și în 3.1, Trakt.tv domină prin funcționalitățile sale, dar nici celelalte sisteme nu sunt de neglijat. Funcționalitatea de recomandări, cu toate că este relativ nouă, fiind avansată doar după evoluția sistemelor de recomandare, este prezentă în marea majoritate a sistemelor. Funcțiile de Review și Comments sunt prezente în toate sistemele ceea ce sugerează că sunt necesare pentru un sistem complet. Baza de date, depinde de nivelul de realizare a aplicației, o aplicație mai performantă precum Trakt.tv are o bază de date extinsă ce cuprinde mai multe informații în timp ce o aplicație mai redusă, precum Flixboss are o bază de date mai mică.

Tabelul 3.1: Analiza sisteme

| | IMDB | Rotten Tomatoes | Trakt.tv | Flixboss | Sistemul meu |
|---|---|---|---|---|---|
| Reviews | DA | DA | DA | DA | DA |
| Comments | DA | DA | DA | DA | DA |
| Recommendations | NU | DA | DA | NU | DA |
| Trailer | DA | DA | NU | DA | DA |
| NextEp | NU | NU | DA | DA | DA |
| Baza de date | Proprie, extinsa | Proprie, extinsa | IMDB, TMDB | Netflix | TMDB+IMDB |

Aplicaţia ce va fi creată doreşte să integreze funcţionalităţile prezentate mai sus pentru a oferi utilizatorului o experienţă cât mai plăcută și să realizeze recomandări pentru preferințele utilizatorului.

# Capitolul 4

# Analiză şi Fundamentare Teoretică

Acest capitol va prezenta conceptual architectura şi perspectiva tehnologică a proiectului concluzionând cu deciziile de dezvoltare luate. De asemenea, se prezintă cerinţele funcţionale şi non-funcționale împreună cu diagrama generală a cazurilor de utilizare şi cazurile de utilizare principale ale aplicaţiei.

## 4.1 Arhitectura sistemului

În acest capitol se prezintă conceptual arhitectura sistemului, perspectiva technologică şi deciziile de dezvoltare luate pe baza acestora.

### 4.1.1 Arhitectura conceptuală

În figura 4.1 este prezentată arhitectura pe baza căreia sunt dezvoltate componentele ce vor face parte din aplicația finală. Arhitectura cuprinde 3 componente:

1. Frontend:
    - Stratul de prezentare al aplicației, include interfața grafică ce relaționează cu clientul,
    - Realizat prin React,
    - Comunică prin HTTP requeste și response cu Backend,
    - Comunică cu serviciul de autentificare din Firebase,
2. Backend:
    - Stratul de bussiness al aplicației, include serviciile realizate de aplicație și prelucrarea datelor,
    - Realizat prin Spring Boot și Java,

- Comunică prin HTTP request și response cu Frontend,
- Comunică cu baza de date prin serviciul de Firestore,
- Comunică cu funcția de recomandare printr-un api,
- Funcționalitatea de recomandare este realizată cu Pythond și Flask,

3. Firebase :

- Baza de date în care sunt stocate datele și care are propriile servicii de autentificare,
- Comunică cu Backend și cu Frontend prin servicii diferite.

Fiecare componentă a fost realizată utilizând diferite limbaje iar comunicațiile dintre acestea sunt realizate prin HTTP. Astfel, frontend este dezvoltat în React care comunică prin HTTP cu backend care a fost realizat atât cu java cât și cu python, iar comunicația cu baza de date este reaizată de ambele module.

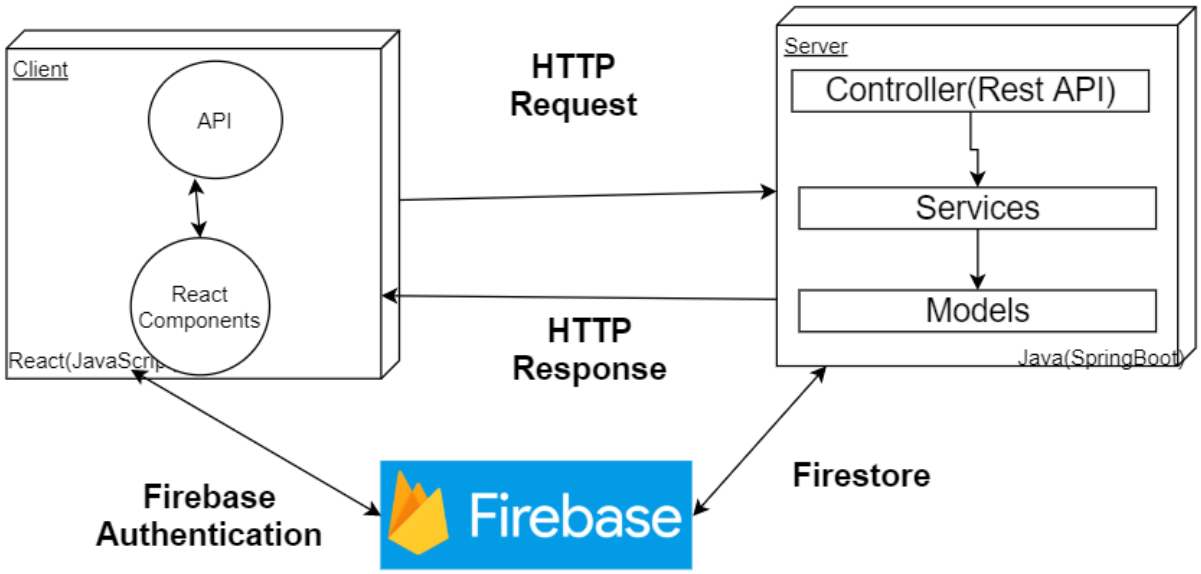


Figura 4.1: Arhitectura conceptuală

## 4.2 Perspectiva Technologică

### 4.2.1 Firebase

Firebase[18] reprezintă un serviciu cloud ce pune la dispoziţia dezvoltatorilor un set de tool-uri ce acoperă o gamă largă de servicii pentru a uşura munca

acestora. Scopul acestui serviciu este de a oferi un mediu sigur pentru crearea şi îmbunătăţirea aplicaţiilor atât web cât şi mobile.

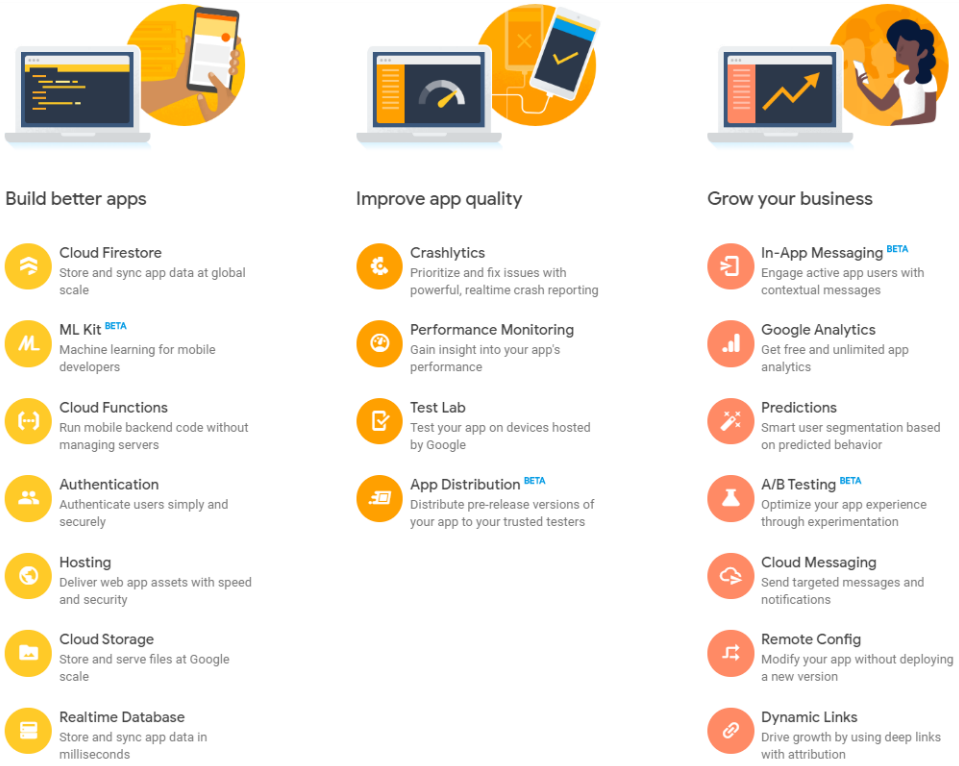


Figura 4.2: Lista de servicii Firebase

Gama de servicii oferite şi securizate de Firebase pentru dezvoltarea de aplicaţii sunt:

1. Authetication- utilizat pentru autentificare utilizatorilor si identificarea lor
2. Realtime Database- bază de date NoSQL în timp real
3. Cloud Firestore- bază de date NoSQL în timp real
4. Cloud Storage -stocarea datelor în cloud
5. Cloud Functions- funcții cloud
6. Firebase Hosting- hosting web
7. ML Kit- Kit de Machine Learning comun

Dintre aceste servicii au fost utilizate Authentication şi Cloud Firestore în aplicaţia în curs de dezvoltare.

### 4.2.2 Firebase Authentication

Firebase Authentication [19] este un serviciu oferit de Firebase prin care se asigură înregistrarea utilizatorilor, logarea şi identificarea acestora pe baza email-ului sau a conturilor deja existente de Google sau Facebook. Identificarea utilizatorilor este prioritară deoarece permite aplicaţiei să salveze datele utilizatorului în siguranţă în cloud şi să se prezinte conţinut specific acestora. Firebase propune două posibilități de implementare a acestui serviciu, prin propriul UI, FirebaseUI Auth sau prin utilizarea unui SDK( Software Development Kit) care oferă posibilitatea de a interacționa cu serverul pentru a realiza diferite acțiuni.

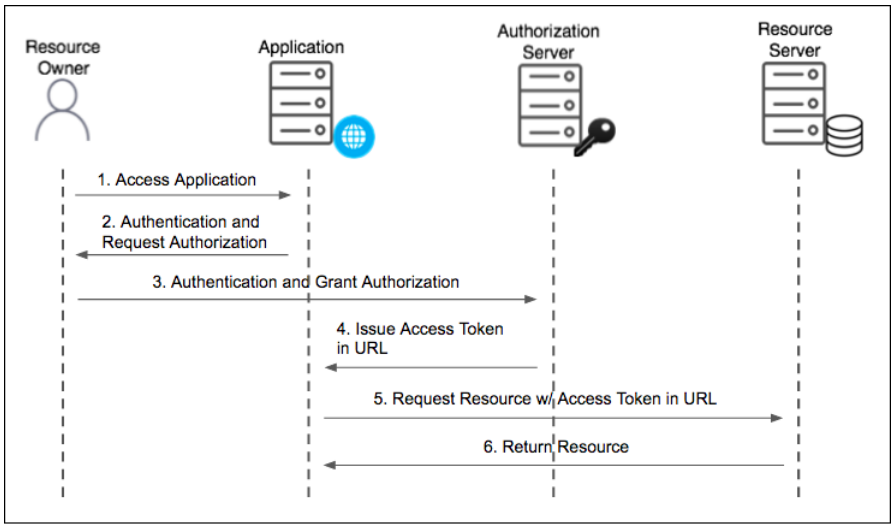


Figura 4.3: Schemă autentificare Firebase

### 4.2.3 Realtime Database

Serviciul de Realtime Database reprezintă una din cele mai cunoscute servicii pe care Firebase le deține, deoarece acesta sincronizează, în timp real, informația stocată ca JSON pentru fiecare client, urmând modelul din figura 4.4. Realtime Database are suport atât pentru Mobile cât și pentru Web development prezentând moduri simple de realizare a funcțiilor pentru conectarea la baza de date. Este o bază de date NoSQL ce stochează informațiile ca JSON dar devine greu de scalat la un cumul mare de date. Are o securitate proprie prin scrierea de reguli ce limitează accesul utilizatorilor la baza de date.


Figura 4.4: RealTime Database

### 4.2.4 Cloud Firestore

Cloud Firestore [20] reprezintă o bază de date NoSQL găzduită în cloud, utilizat pentru stocarea informațiilor din aplicațiile mobile sau web. Datele unui utilizator sunt stocate în documente ce sunt organizate în colecții, aceste documente conțin o denumire unică. Documentul poate conține și subcolecții de elemente sau poate face referire la anumite documente aflate în alte colecții din baza de date. Acesta oferă suport atât pentru aplicațiile mobile pentru IoS și Android cât și pentru aplicațiile Web pentru limbajele Python, Java și JavaScript. În figura 4.10 este reprezentat modelul de date pentru tabela Movies, se poate observa că se pot realiza filtrări, modificări și ștergerea datelor din interfața Firebase.

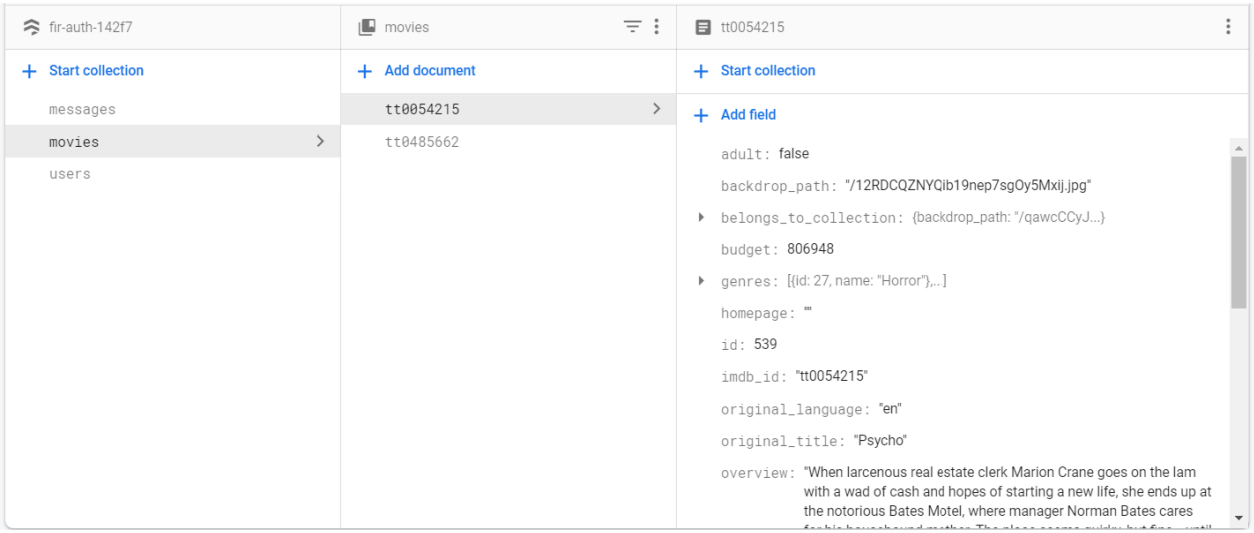


Figura 4.5: Model de date pentru Cloud Firestore

### 4.2.5 TMDB Api

The Movie Database [21] este un API (Application Programming Interface) creat cu scopul de a oferi acces rapid și sigur la informațiile despre filmele existente în baza de date. Pe lângă informații detaliate despre filmele sau serialele existente, acesta oferă suport pentru găsirea actorilor ce joacă în ele.

Aceste informații sunt obținute prin crearea unui controller REST API ce realizează apelul la server. Pentru a avea acces la aceste informații un dezvoltator trebuie să obțină o cheie de acces, API key, ce se obține prin crearea unui cont și solicitarea acestei chei. Apelurile la server sunt refuzate dacă cheia de acces este greșită sau dacă lipsește unul dintre parametrii necesari realizării request-ului. Există diferite forme de apeluri ce pot fi realizate la API pentru a obține diferite informații. Spre exemplu, pentru căutarea unui film este nevoie de cheia API și cuvântul după care se face căutarea, pe lângă acestea se pot adăuga informații suplimentare pentru filtrarea rezultatelor precum anul lansării sau limba .

Listing 4.1: Forma URL pentru call la TMDB API

```
https://api.themoviedb.org/3/search/movie?api_key=*******&query=searchedWord
```

Rezultatul unui apel este reprezentat de un obiect JSON ce conține informațiile filmului. Acesta poate conține mai multe câmpuri cu informații despre film precum:

1. adult: ne oferă informații cu privire la publicul targetat (adulți sau toate vârstele),
2. budget: bugetul filmului,
3. genres: un șir cu genurile prezente în film,
4. id: identificatorul filmului din baza de date a TMDB,
5. imdb id: identificatorul filmului din imdb,
6. poster path: path-ul unde este salvată imaginea posterului,
7. title: titlul filmului,
8. vote average: media voturilor ,
9. vote count : numărul de voturi.

Unele date au fost utilizate doar în scop informațional, în timp ce altele au fost amestecate împreună cu alte funcții pentru a obține funcționalitățile ce se doresc a fi implementate în aplicația în curs de dezvoltare.

Am utilizat TMDB API pentru a obține informațiile importante despre filmele sau serialele căutate, pentru afișarea listei celor mai populare filme, pentru găsirea filmelor și afișarea posterului acestora.

### 4.2.6 IMDB Api

Internet Movie Database (IMDB) [22] Api este un serviciu ce oferă informații detaliate despre filme seriale, anime și actori având mai multe funcționalități comparativ cu TMDB, printre acestea numărându-se găsirea subtitrărilor unui film, filmele ce rulează în prezent în cinema sau găsirea trailerului unui film.

IMDB API monitorizează toate call-urile realizate și le contorizează verificând ca nici un utilizator să nu depășească numărul admis de call-uri din planul ales. IMDB API are planuri separate pentru diferite servicii și pentru durata de utilizare diferită dar, are și un plan gratis care permite utilizarea restrânsă a resurselor. În figura 4.6 este prezentată diagrama de monitorizare a apelurilor din Api, se poate observa că numărul request-urilor este ridicat cu toate că se realizează doar call-urile necesare pentru funcționarea corectă a unei singure componente.

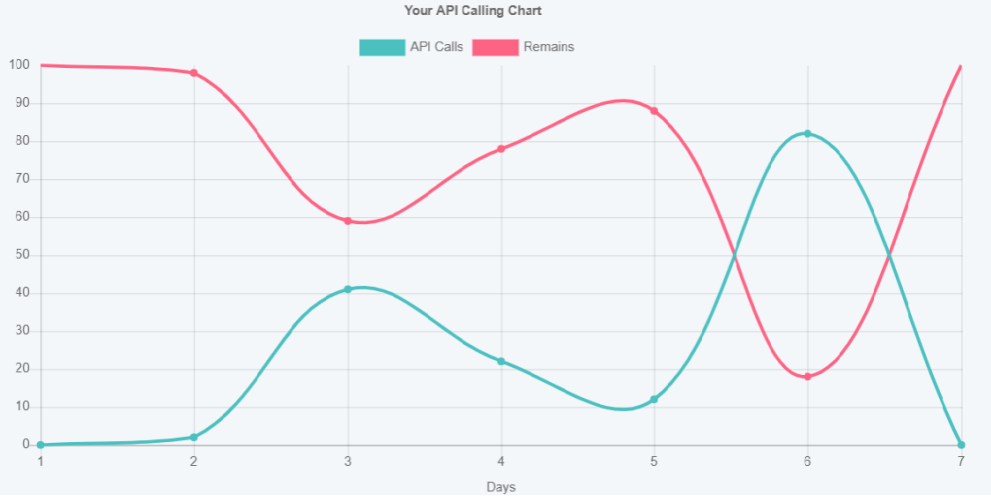


Figura 4.6: Monitorizarea apelurilor API realizate în ultima săptămână

Am utilizat IMDB Api pentru a implementa funcționalitatea de afișare a trailerului, obținând id-ul videoului din baza de date a IMDB. De asemenea, s-a folosit funcția de obținere a episoadelor pentru a fi afișate și pentru a oferi utilizatorului un mod ușor de a căuta un episod specific.

Nu a fost utilizat tot serviciul de IMDB api deoarece numărul de requesturi necesar funcționării corecte a aplicației ar depăși planul care este utilizat de contul personal. Am ales să utilizez o combinație între acest serviciu și TMDB API pentru a minimiza numărul de requesturi pentru încadrarea în planul serviciului oferit.

### 4.2.7 Apache Maven

Apache Maven [23] un instrument utilizat în crearea și gestionarea proiectelor software din limbajul Java. Utilizează un fișier POM.xml (Project Object Model) pen-

tru a menține informațiile necesare configurării proiectului, facilitând transmiterea proiectului de pe un device pe un altul.

Maven realizează o distincție clară între codul ce realizează implementarea funcționalităților și codul ce realizează testarea acestora; această distincție se datorează structurii proiectului prezentată în figura 4.7. Această separare între codurile scrise cu diferite obiective, este evidențiată și prin comenzile Maven ce pot executa diferite părți de cod. Spre exemplu, comanda **mvn test** execută doar porțiunea din partea de test și raportează la final rezultatul testelor.


Figura 4.7: Structura unui proiect Maven

Maven este integrat în diferite IDE-uri (Integrated Development Environment) ceea ce permite rularea, crearea și managementul proiectelor din cadrul mediului de dezvoltare să se mapeze pe structura unui proiect specific Maven. Dintre aceste IDE-uri se evidențiază **IntelliJ IDEA** care va fi folosit pentru dezvoltarea aplicației împreună cu framework-ul Spring Boot.

### 4.2.8 Spring Boot Framework

Spring Boot Framework [24] are scopul de a ușura dezvoltarea proiectelor folosind Java prin evitarea unor pași de configurare. De asemenea, este ușor de utilizat fiind intuitiv, evitând configurații XML complicate și oferind suport pentru crearea serviciilor REST. Configurarea se bazează pe adnotațiile specificate în clase. Spre exemplu, clasa principală conține adnotația **@SpringBootApplication** pentru a marca punctul de început iar, celelalte componente au diferite adnotații printre care se numără:

1. **@Controller** - pentru a marca un endpoint cu diferite metode,
2. **@Service** - pentru marcarea realizării funcționalității de Service, în care are loc procesarea datelor

3. **@Repository** - pentru a marca un o interfață specifică tipului de date utilizat, poate fi de tip JPA sau CRUD

Pe lângă adnotațiile claselor există configurări speciale pentru câmpurile dintr-o clasă model, cum ar fi:

1. **@Required** - pentru a marca necesitatea câmpului,
2. **@Autowired** - pentru a realiza instanțierea automată a unor componente,
3. **@Bean** - pentru crearea de Spring Beans asupra metodelor,

Pentru transmiterea datelor din back-end la front-end am utilizat adnotațiile REST:

1. **@GetMapping**- pentru obținerea de date din baza de date,
2. **@PostMapping**- pentru inserarea datelor în baza de date,
3. **@PutMapping**- pentru updatarea datelor în baza de date,
4. **@DeleteMapping**- pentru ștergerea datelor din baza de date.

Acestea au fost combinate împreună cu alte adnotații pentru a realiza functionalitatile mai complicate precum obținerea recomandărilor utilizatorilor.

Am ales SpringBoot datorită ușurinței cu care am realizat diferite componente și simplității implementării. Am utilizat diferite adnotații pentru a realiza funcțiile dorite, în special cele REST pentru transmiterea datelor prin HTTP.

### 4.2.9 React

Pentru partea de front-end a aplicației am avut de ales între Angular, React și Vue toate, utilizând JavaScript. După o cercetare atentă a acestora, am ales React pentru implementare deoarece oferă suport în dezvoltare, este intuitivă și permite integrarea cu diferite librării JavaScript. React este o librărie bine dezvoltată ce utilizează JavaScript pentru implementarea interfețelor utilizatorilor. Aceasta permite utilizarea concomitentă a diverselor component de UI(User Interface) pentru a oferi clienților o interacțiune plăcută cu aplicația și asigurarea securității acesteia. Conform lucrării [25] React prezintă performanțe favorabile în ceea ce privește viteza executării diverselor comenzi și utilizarea memorie. În React datele sunt transmise de sus în jos marcând un flow de date într-un singur sens, prin urmare orice componentă părinte poate transmite datele unei componente copil dar, componenta copil nu poate transmite datele înapoi. Dar, este nevoie și de un sens invers de transmisie a datelor, sens care poate fi obținut prin utilizarea diferitelor funcționalități de callback precum eventHandlers ce notifică schimbarea unui câmp. Pentru realizarea rutării și securizarea rutelor este necesară instalarea unei librării separete, React Router, care adaugă securizează rutarea.

Pe lângă aceste pachete, s-au instalat mai multe componente, separate, cu scopul ușurării utilizării aplicației și a reprezentării intuitive a diverselor functionalitati precum React-Star-Rating, sau React-youtube. React-youtube este un pachet ce funcționează ca un strat subțire peste Youtube IFrame Player Api și oferă posibilitatea rulării videourilor de pe Youtube în cadrul acestuia, dacă este cunoscut videoId-ul sau url-ul acestuia. Instalarea acestui pachet în proiect s-a realizat cu ajutorul npm(Node Package Manager), care este un manager de pachete ce permite adăugarea și instalarea dependințelor necesare proiectului.

## 4.3 Actorii sistemului

În această secțiune sunt prezentați actorii ce interacționează cu sistemul urmând, ca în capitolul următor să fie prezentate cazurile de utilizare mai importante.

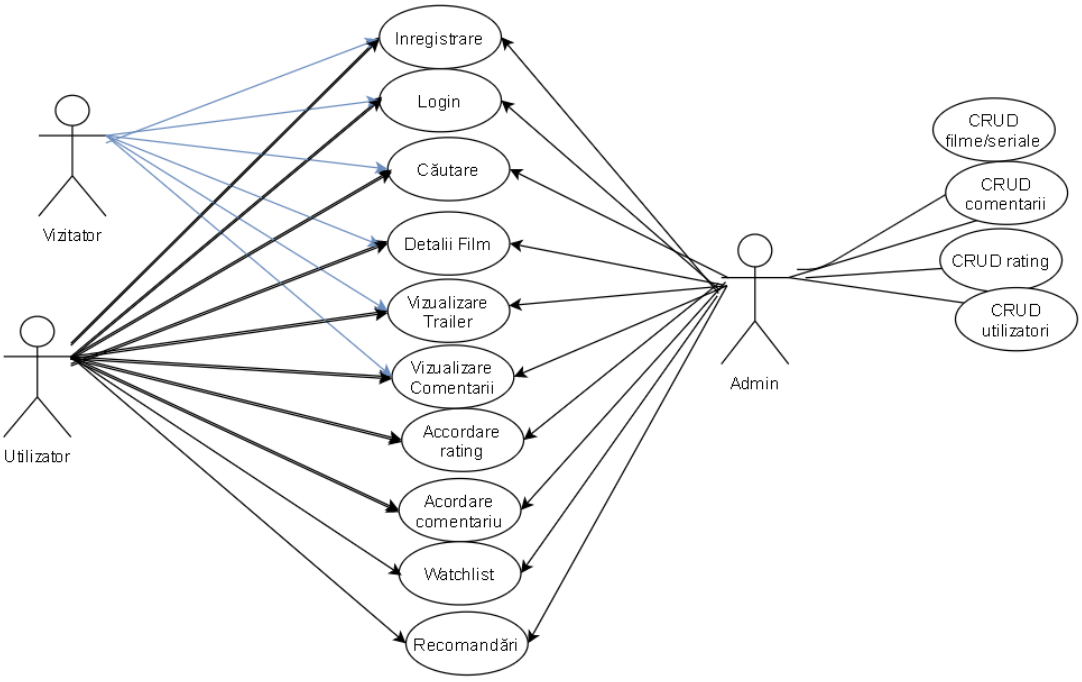


Figura 4.8: Funcționalitățile principale ale adminului, utilizatorii și vizitatorii

Unele funcționalități pot fi utilizate doar de către un utilizator autentificat, precum adăugarea la liste de preferințe a unui film sau serial, obținerea de recomandări, adăugarea unui comentariu sau review la un film. Alte funcționalități pot fi utilizate de toti utilizatorii precum căutarea, vizualizarea detaliilor unor filme sau seriale, vizualizarea comentariilor sau adăugarea de recenzii.

Un vizitator poate interacționa cu pagina home unde poate vedea filmele ce se află în top, poate afla mai multe detalii despre filme apăsând pe ele, sau poate căuta un film.

Un utilizator poate realiza aceleași operații ca și un vizitator, în plus poate crea o listă de filme favorite(Favourite) sau o listă de filme vicionate (Watchist), poate obține recomandări și poate acorda un review sau un comentariu la un film pentru a-și arăta aprecierea.

Un administrator are posibilitatea creării, updatării, modificării și ștergerii datelor din baza de date, inclusiv a utilizatorilor, comentariilor și ratingurilor.

## 4.4 Cerinţe funcţionale

Cerinţele funcţionale reprezintă funcţiile pe care ar trebui să le realizeze sistemul. Am împărțit această categorie în 3 tabele în funcție de tipul de utilizator. Fiecare tabel prezintă ce tip de utilizator este și care sunt acțiunile posibilie ale acestuia.

| Acțiunile posibile ale unui utilizator autentificat | |
|---|---|
| FR-1 | **Utilizator autentificat** |
| FR-1.1 | Utilizatorul poate să modifice parola |
| FR-1.2 | Utilizatorul poate să modifice profilul |
| FR-1.3 | Utilizatorul poate să se deconecteze |
| FR-1.4 | Utilizatorul poate să își șteargă contul |
| FR-1.5 | Utilizatorul poate să primească recomandări pe baza preferințelor |
| FR-1.6 | Utilizatorul poate urmări trailerul unui film |
| FR-1.7 | Utilizatorul poate acorda review filmelor vizionate |
| FR-1.8 | Utilizatorul poate adăuga un comentariu filmelor vizionate |
| FR-1.9 | Utilizatorul poate adăuga la Favourite serialele la care dorește să fie notificat în caz de noi episoade |
| FR-1.10 | Utilizatorul poate căuta un film pe baza titlului, a genului sau al scorului de pe IMDB |
| FR-1.11 | Utilizatorul poate să vadă detalii despre film (Trivia, Photos, short videos, director, actori) |
| FR-1.12 | Utilizatorul poate să citească comentariile celorlalți utilizatori |

| Acțiunile posibile ale unui utilizator ne-autentificat | |
|---|---|
| FR-2 | **Utilizator ne-autentificat** |
| FR-2.1 | Utilizatorul poate urmări trailerul unui film |
| FR-2.2 | Utilizatorul poate să citeasca comentariile celorlalţi utilizatori |
| FR-2.3 | Utilizatorul poate căuta un film pe baza titlului, a genului sau al scorului de pe IMDB |
| FR-2.4 | Utilizatorul poate să vadă detalii despre film (Trivia, Photos, short videos, director, actori) |
| FR-2.5 | Utilizatorul poate să îşi creeze un cont |
| Acțiunile posibile ale unui administrator | |
| FR-3 | **Administrator** |
| FR-3.1 | Administratorul poate să șteargă un utilizator |
| FR-3.2 | Administratorul poate să modifice detaliile unui film |
| FR-3.3 | Administratorul poate să modifice detaliile unui serial |
| FR-3.4 | Administratorul poate să șteargă un film |
| FR-3.5 | Administratorul poate să șteargă un serial |
| FR-3.6 | Administratorul poate să șteargă comentariul unui utilizator |
| FR-3.7 | Administratorul poate să vadă toți utlizatorii existenți în sistem |

## 4.5 Cerinţe non-funcţionale

Cerinţele non-funcţionale sunt cerinţele ce se referă la atributele non-funcţionale ale unei aplicaţii precum viteză, securitate, portabilitate, utilizabilitate, mentenanţă.

| Cerinţe non-funcţionale | |
|---|---|
| Fr-1 | **Securitate** |
| FR-1.1 | Criptarea parolelor și înlocuirea acestora la fiecare 30 de zile |
| FR-1.2 | Securizarea rutelor și autorizarea paginilor proprii ale utilizatorilor |
| Fr-2 | **Utilizabilitatea** |
| FR-2.1 | Interfaţa este uşor de folosit și este intuitivă |
| FR-2.2 | Utilizatorul poate folosi aplicaţia şi fără a se autentifica/a crea cont |

## 4.6 Descrierea detaliată a cazurilor de utilizare

S-au ales cele mai importante cazuri de utilizare pentru a fi dezvoltate și pentru a se prezenta fluxul din cadrul aplicației. Aceste cazuri sunt despărțite în cazuri ale utilizatorului autentificat și ale utilizatorului ne-autentificat, amintim mai jos care sunt acestea:

- Utilizator autentificat, reprezintă utilizatorul ce este înregistrat în aplicație și are acces la serviciile oferite de acesta:

1. **UC1**: Log In, reprezintă activitatea de logare a unui utilizator deja înregistrat în aplicație,
2. **UC2**: Adăugarea unui film la Watchlist, reprezintă adăugarea unui film deja vizionat la lista de ”Văzute”,
3. **UC3**: Acordarea unui review reprezintă exprimarea apricierilor față de filmul deja vizionat
4. **UC4**: Obținerea unor recomandări, reprezintă realizarea recomandărilor pe baza preferințelor

- Utilizator ne-autentificat reprezintă utilizatorul ce nu este înregistrat și nu este recunoscut de sistem, prin urmare are acces la o gamă restrânsă de servicii:

1. **UC5**: Înregistrarea, reprezintă activitate de înregistrare și autentificare a unui utilizator în aplicație,
2. **UC6**: Căutarea unui film în aplicație reprezintă căutarea după titlul unui film.

### 4.6.1 Cazuri de utilizare pentru un utilizator autentificat

***UC1***

**Use case:** Log in
**Actor:** Utilizator înregistrat
**Descriere:** Utilizatorul doreşte să se logheze în aplicaţie pentru a folosi toate serviciile pe care le oferă aceasta
**Flow:**
1. Utilizatorul introduce emailul și parola cu care şi-a creat cont,
2. Utilizatorul apasă butonul de "Sign In",
3. Operaţia reuşeşte și utilizatorul este redirecționat la pagina de profil.

**Flow alternativ:** În cazul în care emailul și parola sunt scrise greşit:
1. Credenţialele folosite de user sunt invalide
2. Userul este rugat să încerce din nou
3. Flow-ul revine la pasul întâi.

**Precondiții:**
Utilizatorul se află pe pagina de login.
**Postcondiții**
Utilizatorul s-a logat cu succes în aplicaţie.

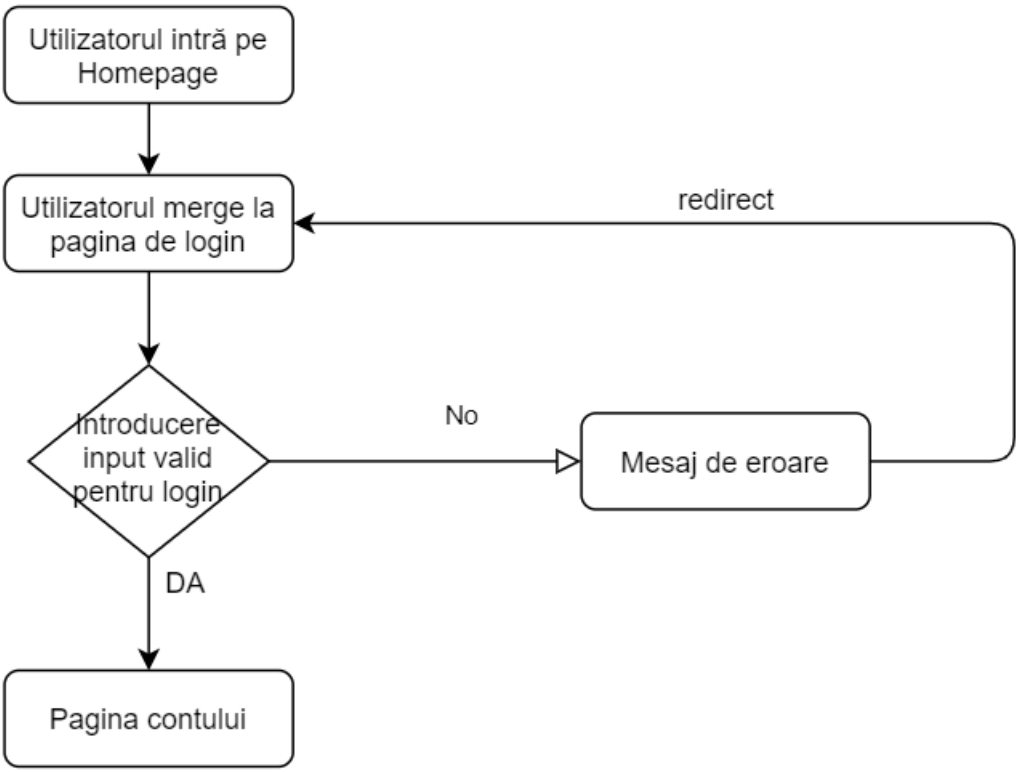


Figura 4.9: Diagrama de flow pentru login

***UC2***

**Use case:** Adăugarea unui film la WatchList
**Actor:** Utilizator înregistrat
**Descriere:** Utilizatorul doreşte să adauge un film la watchlist pentru a putea fi notificat în cazul în care apar noutăţi.
**Flow:**
1. Utilizatorul apasă pe butonul de "add to watchlist" de lângă titlu.
2. Filmul este adăugat la watchlist.
3. Operaţia reuşeşte si utilizatorul poate fi notificat în viitor.

**Flow alternativ:** -
**Preconditii:**
Utilizatorul se află pe pagina de informaţii despre film.
Utilizatorul este logat în aplicaţie.
**Postconditii**
Filmul apare în lista de vizionare a utilizatorului.
În cazul unor noutăţi utilizatorul va fi notificat.

***UC3***

**Use case:** Lăsarea unui review
**Actor:** Utilizator înregistrat
**Descriere:**Utilizatorul doreşte să lase un review (o notă) unui film.
**Flow:**

1. Utilizatorul se află pe pagina de informaţii a filmului.
2. Utilizatorul selectează numărul de stele.
3. Utilizatorul apasă butonul "Rate".
4. Review este înregistrat.

**Flow alternativ:** -
**Preconditii:**
Utilizatorul se află pe pagina de informaţii a filmului.
Utilizatorul este înregistrat în aplicaţie.
**Postconditii**
Review-ul unui film a fost realizat cu succes.

***UC4***

**Use case:** Obținerea unor recomandări
**Actor:** Utilizator înregistrat
**Descriere:** Utilizatorul doreşte să vizioneze noi filme utilizând funcționalitatea de recomandare.
**Flow:**

1. Utilizatorul se află pe pagina Home.
2. Utilizatorul alege butonul de recomandare
3. Utilizatorul obține recomandările

**Flow alternativ:** În cazul în care nu deține filme vizionate sau favorite în listă acesta va trebui să își adauge unele dintre filmele recomandate

1. Utilizatorul este redirectat la o listă de filme pentru a le adăuga în wish-list.
2. Utilizatorul are posibilitatea să caute mai multe filme pentru a adăuga la wishlist.

**Preconditii:**
Utilizatorul se află pe pagina Home.
Utilizatorul este înregistrat în aplicaţie.
Utilizatorul are o listă de filme favorite în care se află cel puțin 5 filme.
**Postconditii**
Utilizatorul este redirecționat pe o pagină în care se află recomandările acestuia.

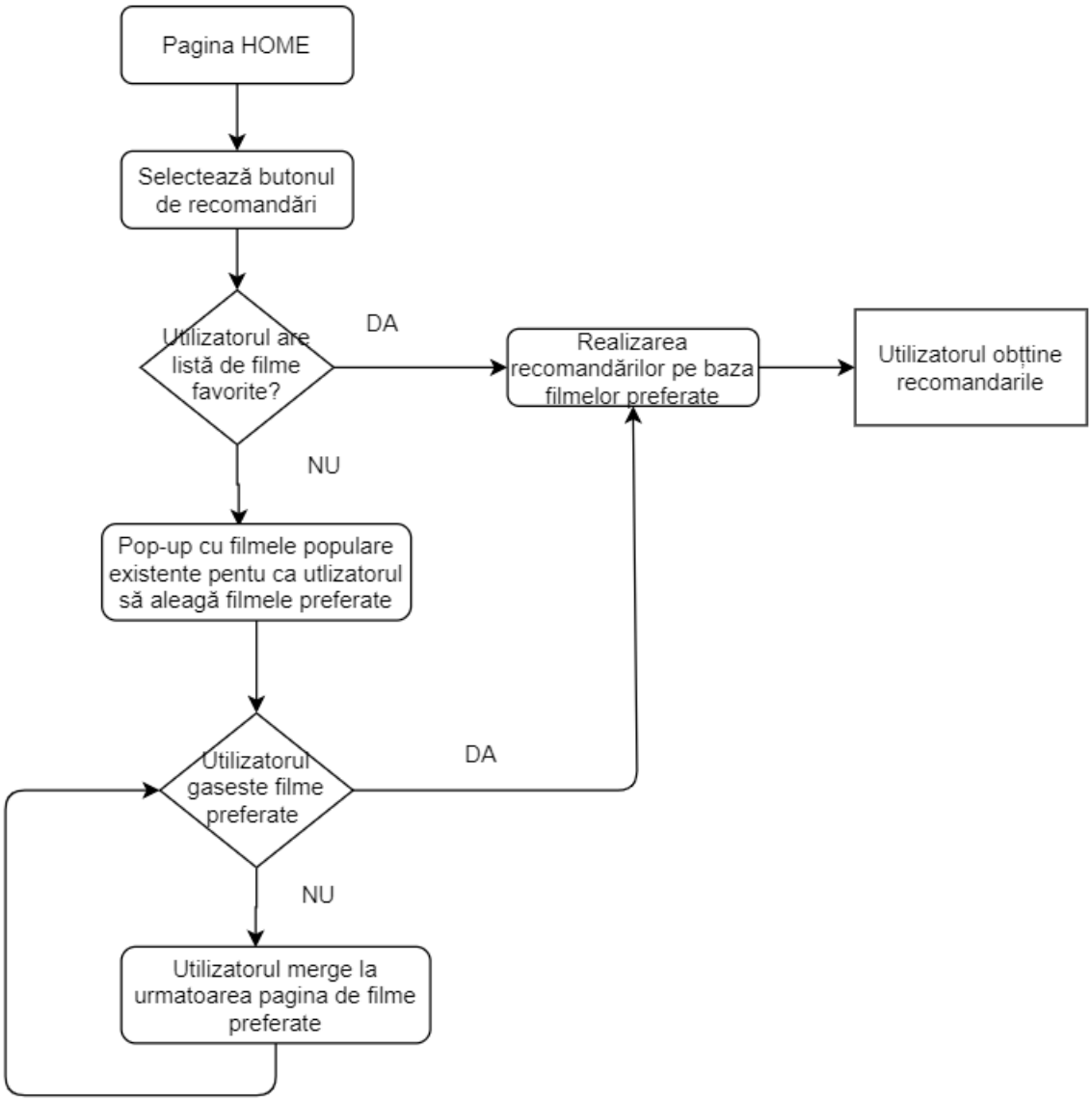


Figura 4.10: Diagrama de flow pentru recomandarea filmelor

### 4.6.2 Cazuri utilizator ne-autentificat

***UC5***

**Use case:** Înregistrare
**Actor:** Utilizator neînregistrat
**Descriere:** Utilizatorul doreşte să îşi creeze un cont în aplicaţie pentru a folosi toate serviciile pe care le oferă aplicația.
**Flow:**

1. Utilizatorul apasă butonul de "Register" şi este redirecționat pe pagina de înregistrare.

2. Utilizatorul completează toate câmpurile necesare creării unui cont cu date valide.

3. Utilizatorul apasă butonul de register.

4. Utilizatorul primește un email pentru a verfica contul.

5. Utilizatorul poate intra în cont dupa terminarea flow-ului de login, înregistrarea s-a realizat cu succes.

**Flow alternativ:**

1. În cazul în care adresa de email este greșită sau există deja în baza de date:
   1.1 Emailul folosit de user este invalid iar userul este notificat de acest lucru.

   1.2 Userul este rugat să incerce din nou.

   1.3 Flow-ul revine la pasul 1.

2. În cazul în care numele userului există:

   2.1 Se vor oferi utilizatorului diferite posibilități pentru a alege numele.

   2.2 Userul este rugat să încerce din nou

3. În cazul în care parola este invalidă:

   3.1 Userul este notificat de acest lucru.

   3.2 Userul este rugat să incerce din nou. 3.3 Flow-ul revine la pasul 1.

**Preconditii:**

Utilizatorul se află la pagina "Home " a aplicaţiei.

**Postconditii**

Utilizatorul primeşte email pentru a îşi verificat contul, iar apoi se poate loga cu credenţialele introduse.

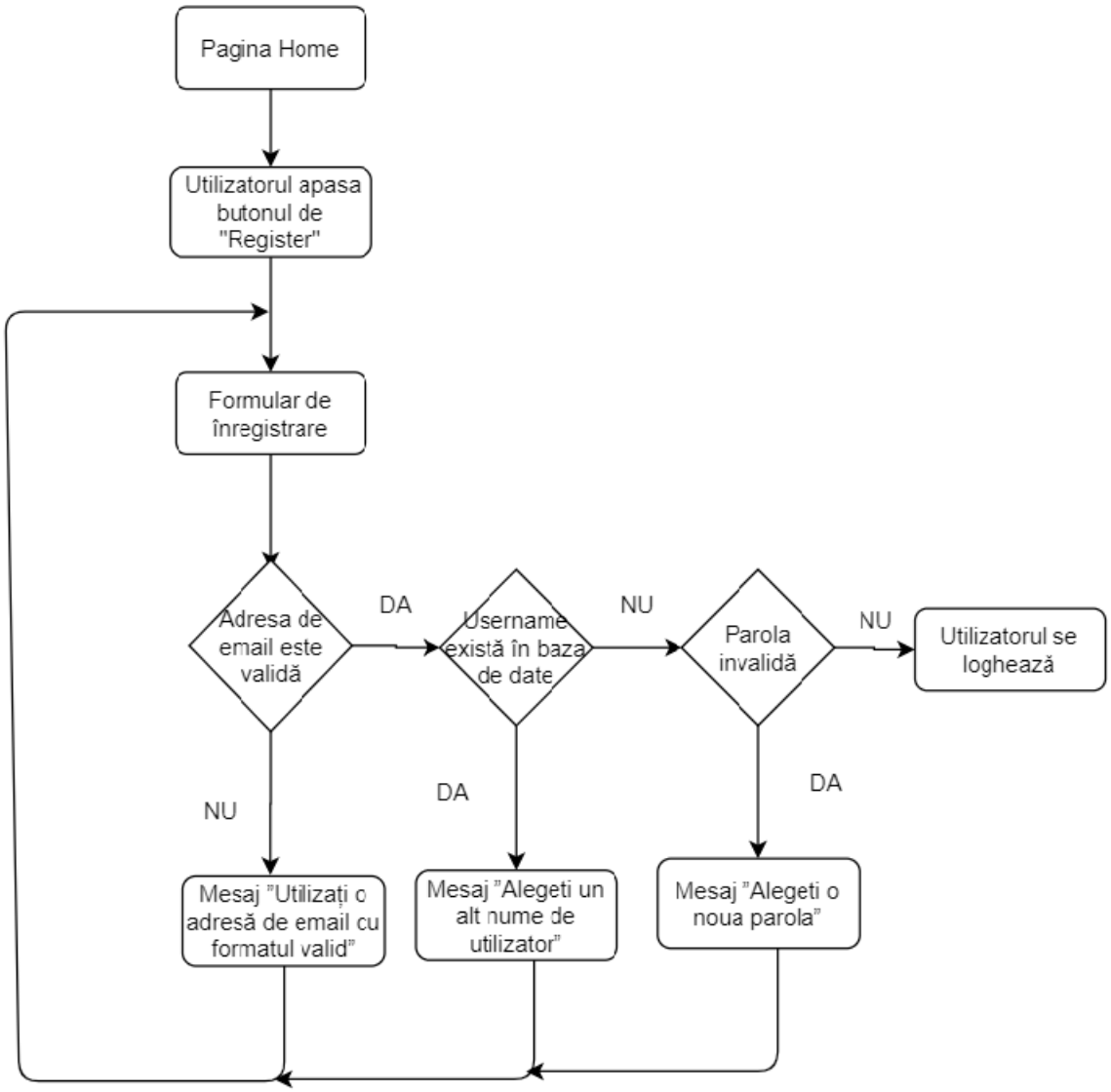


Figura 4.11: Diagrama de flow pentru înregistrarea unui utilizator

***UC6***

**Use case:** Căutarea unui film în aplicaţie
**Actor:** Orice tip de utilizator
**Descriere:** Utilizatorul doreşte să caute un film în aplicaţie.
**Flow:**

1. Utilizatorul completează bara de căutare cu numele filmului
2. Utilizatorul apasă butonul de "Search"

3. Utilizatorul este redirecționat către o pagină de căutare unde vor apărea filmele ce au în titlu sau în descriere cuvintele căutate.

**Flow alternativ:** În cazul în care nu este găsit nici un film cu denumirea căutată:

1. Utilizatorul este redirecționat către o pagină albă cu textul "Nu s-au găsit rezultate, este cuvăntul corect?"

**Precondiții:**

Utilizatorul se află pe pagina Home .

**Postcondiții**

Utilizatorul găseşte filmul căutat.

# Capitolul 5

# Proiectare de Detaliu şi Implementare

Acest capitol are rolul de a prezenta implementarea funcționalităților descrise în capitolul 4 prin utilizarea cunoștințelor detaliate în capitolul 3. Scopul final al acestui capitol este de a realiza un prototip al aplicației și de a detalia asupra unor decizii luate. Acest capitol este împărțit în două mari subcapitole reprezentând sistemul de gestiune și sistemul de recomandare.

## 5.1 Sistemul de gestiune

Acest subcapitol reprezintă implementarea sistemului de gestiune și deciziile luate pe baza tehnologiilor folosite. Acest subcapitol se împarte în trei părți:

- Baza de date Firestore
- Implementarea în Backend
- Afișarea în Frontend

### 5.1.1 Modelul de date Firestore

Firestore 4.10 reprezintă una dintre funcționalitățile de la Firebase pe care le-am utilizat în realizarea proiectului. Am decis să realizez 5 colecții de date ce conțin cele mai importante clase, fiecare clasă conținând documente diferite cu tipuri de date diferite sub forma JSON (JavaScript Object Notation). Colecțiile conțin atât date simple precum Integer (Întregi), String, Double cât și date înlănțuite reprezentând obiecte.

Colecția **users** este cea mai complicată colecție deoarece conține marea majoritate a datelor utilizatorilor precum și liste de obiecte înlănțuite.

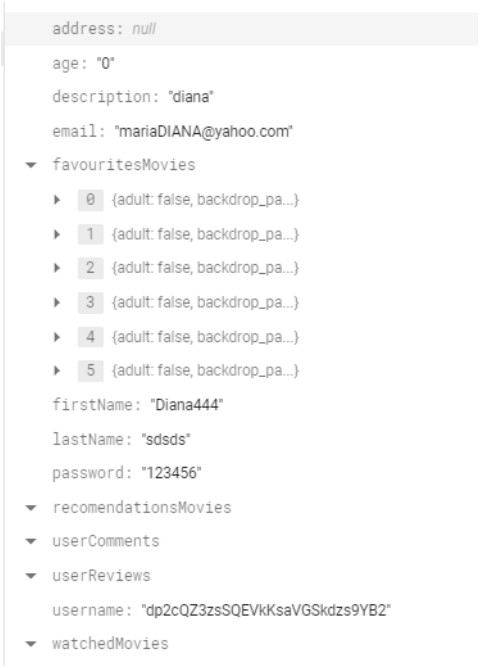


Figura 5.1: Câmpurile din tabela users

Colecțiile **movies** și **shows** conțin date complete despre filmele sau serialele cărora li s-au adăugat comentarii sau review cu scopul de a găsi mai ușor și mai rapid datele necesare în cazul afișării acestora.

Colecțiile **comments** și **reviews** stochează toate datele existente în legătură cu aprecierile lăsate de către utilizator, având ca identificator pentru fiecare document id-ul unic al utilizatorului pentru a ușura găsirea informațiilor.

Pentru realizarea unei diagrame a bazei de date am utilizat Hackolade care este un tool nou pe bază de membership ce permite utilizarea acestuia pentru o perioadă de test. Acest tool este util tuturor bazelor de date NoSQL, permițând atât forward engineering cât și backwards enginnering. Este ușor de folosit și permite exportarea propriilor baze de date împreună cu documentația. Modelul bazei de date este afișat în 5.2

S-a ales salvarea datelor de review și comment ale utilizatorilor separat deoarece volumul de date al tabelei user crește excesiv iar obținerea tuturor datelor despre comentarii sau review este îngreunată. Prin urmare, tabelele Comment și Review vor fi referențiate în tabela User.

Baza de date se conectează la backend prin pachetul de inițializare ce conține informațiile de conectare și clasa care realizează conexiune la Firestore. Pentru a se realiza conexiunea, datele de inițializare trebuie citite din fișierul admin SDK **fir-auth-142f7-firebase-adminsdk-wtsk0-3f0d0233b7.json**.În figura de mai jos sunt prezentate câmpurile specifice unui fișier admin SDK pentru conectarea la baza de date, conținând credențialele de access la aceasta.

Listing 5.1: Fisierul pentru conectarea la Firebase

```
{ "type":"service_account" ,
  "project_id": "id-ul proiectului",
  "private_key_id": " id cheii private necesara pentru acesul la baza
     de date",
  "private_key": "id cheii private necesara pentru acesul la baza de
     date",
  "client_email": "email-ul creat in cadrul firestore si de pe care se
     pot folosi serviciile de trimitere de email",
  "client_id": "id-ul clientului ",
  "auth_uri": "",
  "token_uri": "token pentru accesul la aplicatie",
  "auth_provider_x509_cert_url":" ",
  "client_x509_cert_url":""
}
```

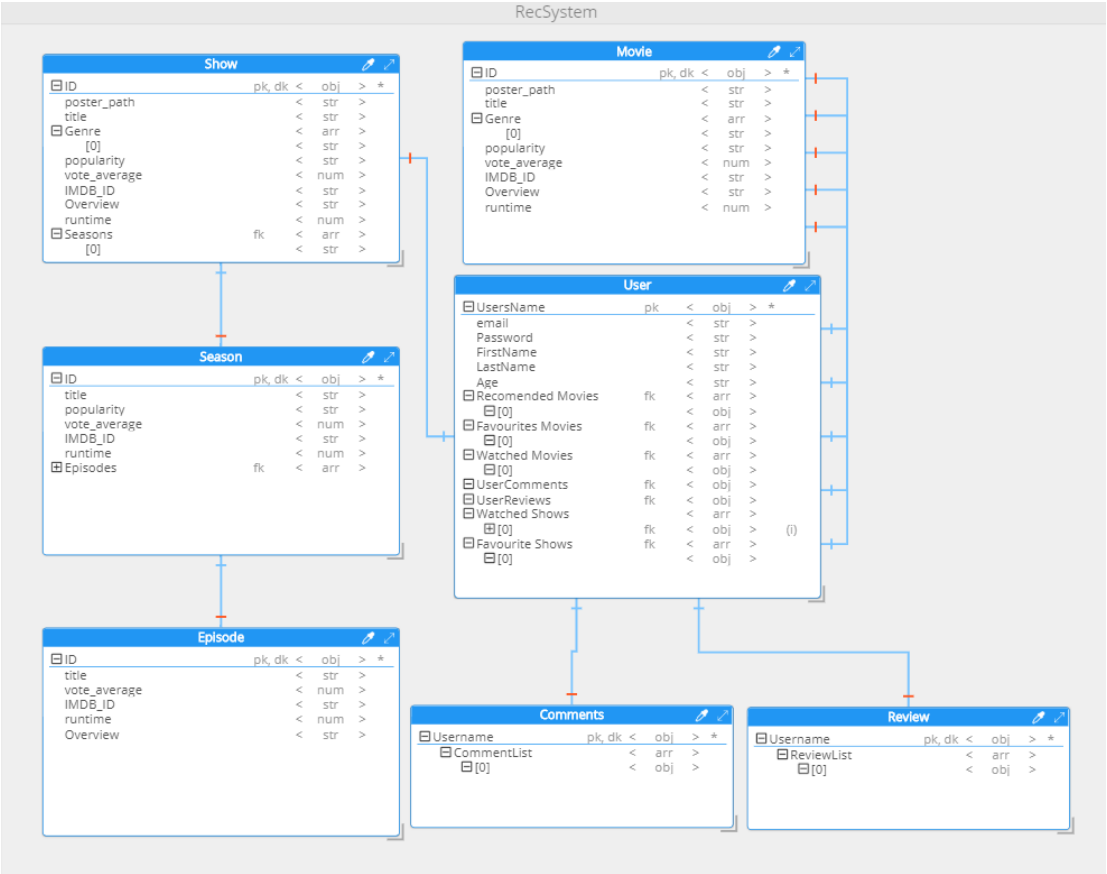


Figura 5.2: Baza de date realizată prin Hackolade

Accesul la baza de date se realizează pe baza unor reguli ce sunt create de către administratorul bazei de date pentru a modera accesul la informații pe baza tokenului de identificare. Aceste reguli sunt revizuite periodic de către Firebase, iar dacă acestea sunt declarate ”nesigure” de către sistem, administratorul este avertizat printr-un email pentru a le verifica și modifica.

### 5.1.2 Procesarea datelor în Backend

Datele extrase din baza de date sunt mai apoi procesate în backend și trimise fie în front fie înapoi în baza de date. Backend conține 4 pachete necesare transmiterii și modificării informațiilor:

1. Initializer package: clasa de inițializare și realizare a legăturii la baza de date
2. Entities package :clasele model ce conțin aceleași câmpuri ca tabelele din baza de date
3. Services package: clasele de procesare a informațiilor pentru realizarea diferitelor operații asupra acestora precum: adăugare, modificare, ștergere, citire
4. Controller package: clasele ce expun endpointul de la serviciul de backend, prezintă metode ce leagă serviciul de backend atât de frontend, baza de date cât și de serviciile de API utilizate.

#### Initializare package

Pachetul Initializare conține o singură clasă ce realizează legătura la baza de date pe baza

#### Entities package

Pachetul Entities conține mai multe clase ce se mapează pe modelul de date din baza de date Firestore. Conform 5.3 clasa Movie conține instanțe din mai multe clase din pachet pentru a crea Obiectul Movie.

#### Services package

Pachetul Services conține implementarea mai multor funcții pentru realizarea diferitelor acțiuni asupra bazei de date. Exemple de funcții :

- **addFavouriteMovie** : adăugarea unui film la favorite se realizează prin găsirea filmului și a utilizatorului în baza de date și adăugarea acestuia la lista de favorite urmând ca acesta să fie updatat în baza de date prin operația de save.

- **getMovieByID** : Obținerea unui film după id-ul transmis, realizează un call la API TMDB pentru a obține toate datele filmului, apoi realizează salvarea acestuia în baza de date la colecția movies.

- **saveComments** : Salvarea unui comentariu în baza de date este realizată prin găsirea listei de comentarii ce corespunde utilizatorului specificat și adăugarea la aceasta.

### Controller package

Pachetul Controller realizează expunerea de API (Application Programming Interface) pentru a răspunde la cererile primite de la Frontend sau pentru a comunica cu baza de date. Pachetul Controller conține mai multe clase, fiecare specifică pentru endpointul pe care îl expune și diferite metode ce conțin parametrii diferiți pentru realizarea call-urilor. S-au utilizat atăt parametrii în header cât și obiecte transmise în call.

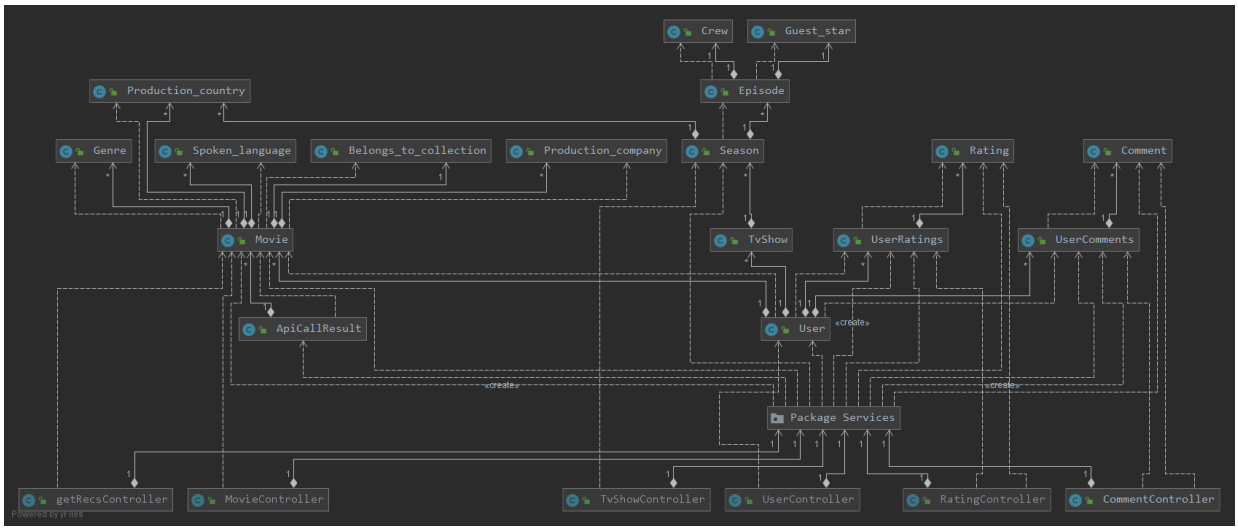


Figura 5.3: Clasele din Backend din toate pachetele

### Integrarea cu TMDB API

Integrarea cu The Movie Database (TMDb) a fost posibilă doar după obținerea cheii de API. Aceasta a fost salvată separat și s-au realizat funcții de citire a acesteia din fișier pentru a ușura portabilitatea aplicației între diferite servere. Call-ul trebuie să urmeze modelul din capitolul anterior, cu el se realizează o conexiune HTTP prin care se comunică cu serverul TMDB și care returnează un JSON ce conține obiectele dorite.

Un exemplu de răspuns la apelul pentru obținerea unor informații detaliate despre un film pe baza id-ului:

Listing 5.2: Raspuns de la un apel catre TMDB API


```
{ "adult": "false",
  "budget": "200000000",
  "genres": [
    {
      "id": 18,
      "name": "Drama"
    },
    {
      "id": 10749,
      "name": "Romance"
    }
  ],
  "id": "597",
  "imdb_id": "tt0120338",
  "original_language": "en",
  "original_title": "Titanic",
  "overview": "101 year old Rose DeWitt Bukater tells the story of her
     life aboard the Titanic, 84years later. A young Rose boards the
     ship with her mother and fiance. Meanwhile, Jack Dawson and
     Fabrizio De Rossi win third class tickets aboard the ship. Rose
     tells the whole story from Titanic departure through to its death
     on its first and last voyage on April 15,1912.",
  "popularity": "32.046",
  "poster_path": "/9xjZS2rlVxm8SFx8kPC3aIGCOYQ.jpg",
  "release_date": "1997-11-18",
  "title": "Titanic",
  "video": "false",
  "vote_average": "7.8",
  "vote_count": "17146"
}
```


### 5.1.3 Afișarea datelor în Frontend

Frontul este principalul mod de interacționare între utilizator și client, prin urmare s-a încercat realizarea unei interfețe cât mai interactivă și intuitivă prin utilizarea de componente din diferite pachete de simboluri din React precum React-Star-Ratings, material-ui, ant-design pentru a reprezenta grafic acțiunile ce se pot

realiza în cadrul proiectului. Mai mult, am utilizat pachetele de Firebase și Firestore pentru a realiza login-ul și register-ul.

O dată cu intrarea pe pagină a utilizatorului, acesta are posibilitatea să caute un film sau serial prin introducerea numelui în câmpul de căutare, mai mult acesta are posibilitatea să filtreze printre diferite genuri și să afișeze alfabetic sau după popularitatea filmelor.

Accesul la unele pagini este dat de un token de acces deoarece aceste pagini sunt specifice pentru utilizator și sunt încărcate cu date specifice acestuia. Pentru ca utilizatorul să se logheze s-a implementat login-ul pe baza unui token generat de firebase. Acest token se generează automat la utilizarea serviciului de autentificare din firebase și este reprezentat de un șir de litere sau cifre diferit. O dată cu logarea utilizatorului, acesta rămâne logat până la de-logare sau până expiră sesiunea acestuia.

Utilizatorul are o pagină a contului unde poate vedea acțiunile sale precedente și poate accesa funcționalitățile prezentate. Mai mult, acesta poate să adauge un film la favorite prin apăsarea butonului inimă din cadrul paginii filmului, sau să îl adauge la vizionat prin apăsarea butonului plus. Posibilitatea de a arăta aprecierea unui utilizator este de asemenea implementată în butonul review. Pentru vizualizarea comentariilor existente despre un film,utilizatorul trebuie să apese pe butonul de mesaje albastru. Toate aceste componente se realizează după același șablon, se face un call la endpointul de backend al serviciului cu id-ul filmului și id-ul utilizatorului și dacă este necesar un obiect de tipul Comment sau Review, se procesează informațiile, urmând ca acestea să fie memorate în backend de către serviciul de salvare.

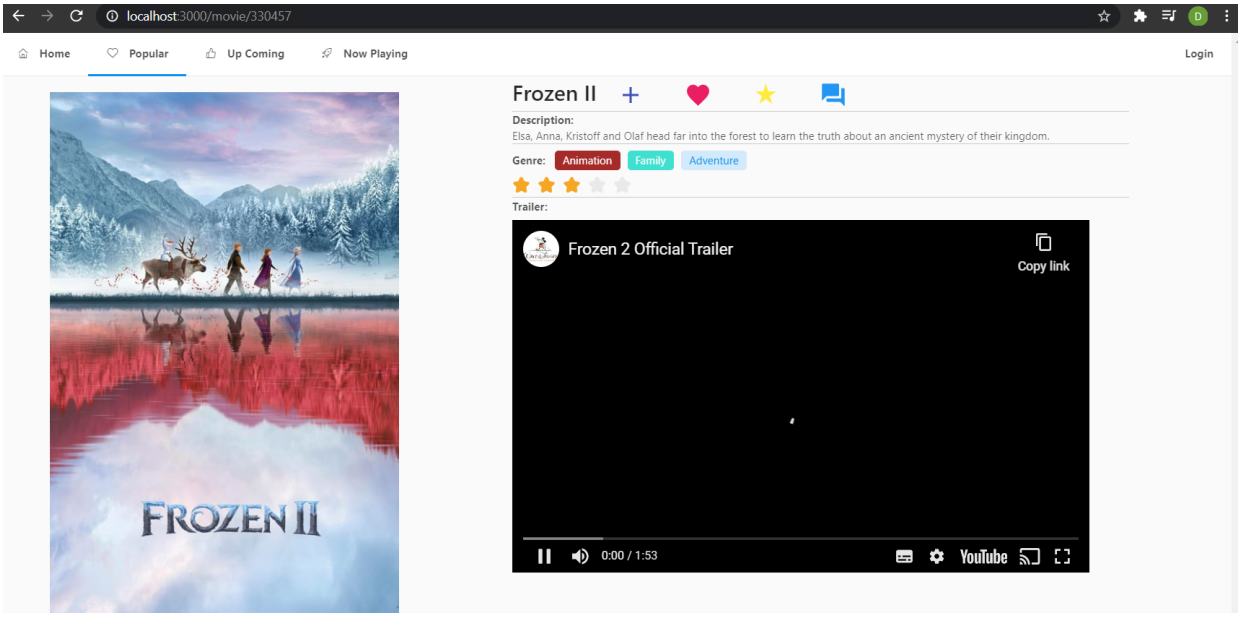


Figura 5.4: Modelul unei pagini din aplicație

Pentru afișarea informațiilor despre trailer a fost nevoie de realizarea unui call la IMDB pentru a afla date despre acesta. Apoi, afișarea acestuia s-a făcut prin pachetul React-Youtube ce conține o interfață pentru afișarea trailerului. Pe pagina de detalii a filmului se găsesc și informații despre acesta, posterul, genul și un scurt rezumat despre acesta 5.4. Modelul de pagină de la film a fost implementat și pentru serial, realizându-se legătura la episod prin call la IMDB API pentru a primi mai multe informații.

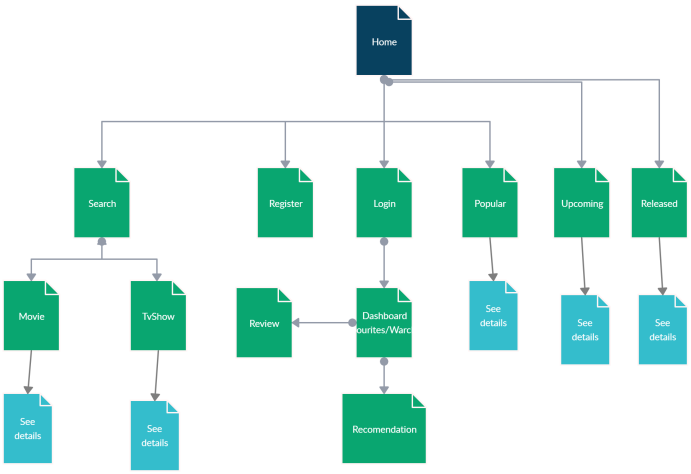


Figura 5.5: Ierarhia paginilor

În figura 5.5 este prezentat modelul ierarhic al paginilor, modul prin care se ajunge de la pagina de home până la paginile de review, favourite, watched sau recommended. Paginile marcate cu verde sunt pagini ce au posibilitatea de a redirecta la alte pagini, în timp ce paginile marcate cu albastru sunt paginile ce nu au posibilități de dezvoltare sau de accesare unei alte pagini. Rezultatul unei căutări se face prin afișarea mai multor card Templateuri ce conțin posterul filmului, ratings și numele acestuia în partea de jos, iar când se selectează, acesta duce la pagina de detalii a filmului.

## 5.2 Sistemul de recomandare

Acest subcapitol reprezintă implementarea sistemului de recomandare și deciziile luate pe baza tehnologiilor folosite. Pentru realizarea sistemul s-a folosit setul de date MovieLens și mai multe librării cu scopul de a crește precizia sistemului.

A fost ales ca algoritm de implementare K-Nearest Neighbors deoarece acesta face asocieri atât între utilizatori cât și între filme pe baza ratingului. Pentru filme s-a utilizat setul de date MovieLens care conține și partea de rating a utilizatorilor, pe care se antrenează setul de date. Iar pentru filme am obținut un set de date al TMDB de pe Kaggle ce conține un fișier în care se află și ratingul și numărul de voturi și id-ul unui serial.

### 5.2.1 Setul de date

MovieLens conține două fișiere .csv(Comma Separated Value) ce au diferite informații despre o gamă largă de filme.

1. movies.csv- conține informații despre filme:
    - movieId: identificatorul filmului
    - title: titlul filmului
    - genres: genurile filmului
2. ratings.csv: conține mai multe intrări ale utilizatorilor specificând ratingul și identificatorul filmului apreciat
    - userId: identificatorul utilizatorului
    - movieId: identificatorul fimului
    - rating: scorul filmului

Pentru aplicarea algoritmului KNN (K-Nearest Neighbors) este necesară prelucrarea datelor din cele două fișiere pentru obținerea unui singur set de date,

datele ce lipsesc din fișier vor fi completate cu 0 deoarece utilizatorul nu le-a apreciat. Se poate crește precizia prin filtrarea acestor date, alegând filmele cele mai populare ca număr de voturi, pentru asta se numără câți utilizatori au oferit un rating filmului și se realizează un rating final al acestuia ce conține media voturilor. Identic a fost realizată procesarea datelor pentru fișierul de seriale și implementarea sistemului este aceeași diferență făcând greutatea obținerii de date despre seriale. Pentrul seriale am utilizat un set de date de la TMDB, s-a realizat împărțirea ratingului cu numărul de utilizatori ce au votat împreună cu crearea unui fișier dummy de antrenare a algoritmului pentru a obține informațiile necesare.

### 5.2.2 Algoritmul K-Nearest-Neighbors(KNN)

K-Nearest-Neighbors este un algoritm ce realizează gruparea filmelor pe baza ratingului oferit de diferiți utilizatori. Pentru asta se realizează o matrice cu titlul filmului și ratingul oferit de utiliatori din setul de date de mai sus.

Pentru realizarea recomandărilor pe baza KNN am folosit funcțiile existente în scikit-learn pentru a obține distanțele dintre filmele existente în baza de date și au fost filtrate rezultând cele mai apropiate filme față de filmul căutat. Apoi, s-a realizat îmbinarea listei cu filme ținând cont de ratingul dat de utilizatorul care necesită infomațiile. După ce a avut loc crearea matricii cu utilizatori și filme, a fost definit modelul Knn din scikit-learn utilizând parametrii următori:

1. metric: reprezintă expresia matematică ce se va folosi pentru calcularea distanțelor
2. algorithm: tipul de algoritm ce poate fi folosit, poate fi ales automat prin utilizarea parametrului ”auto” sau sunt posibile următoarele alegeri:
    - ball-tree [26] : urmează modelul unui arbore unde frunzele și nodurile sunt sub formă de clustere rotunde, astfel că datele sunt împărțite în mai multe clustere.

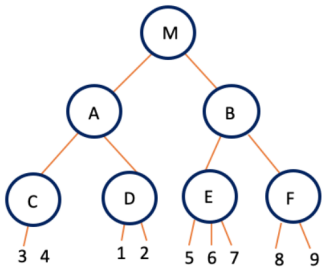


Figura 5.6: Reprezentarea Ball-Tree

    - kd-tree [26]: reprezentat de un arbore binar ce se încheie mereu în două noduri și este mai potrivit procesării datelor ce au două coordonate, acesta

utilizează distanța mediană de la prima axă (X) apoi distanța mediană de la a doua axă (Y). Apoi se realizează plotarea acesteia și împărțirea datelor pe fiecare ramură.

- brute-force: este prezentat în lucrarea [27] ca fiind cea mai simplă versiune a K-Nearest-Neighbors, fiind cuprins din trei pași:
  (a) Pasul unu reprezintă calcularea distanțelor dintre obiectul căutării și toate punctele din setul de date
  (b) Pasul doi reprezintă sortarea acestor distanțe prin selectarea ”vecinilor” cei mai apropiați
  (c) Pasul trei este pasul în care se realizează clasificarea vecinilor.

3. neighbors: numărul vecinilor care sunt căutați de sistem.
4. jobs : numărul de joburi ce funcționează în paralel pentru ceilalți vecini, -1 reprezintă utilizarea tuturor proceselor în timp ce 1 reprezintă utilizarea numai a unuia.

Am ales numărul de joburi rulate în paralel să fie maxim pentru ca utilizatorul să primească în timp util recomandările. Am ales ca algoritm de găsire a vecinilor, de tip brute-force deoarece timpul realizării unei interogări asupra unui volum mediu de date este mai bun iar eficacitatea obținerii rezultatului este similară. Am ales ca metrică similaritatea cosină, aceasta măsoară asemănările dintre 2 vectori pentru a determina dacă cei doi se îndreaptă înspre aceeași direcție .

Listing 5.3: Model pentru KNN

```
model_knn = NearestNeighbors(metric='cosine', algorithm='brute',
    n_neighbors=10, n_jobs=-1)
```

### 5.2.3 Comunicarea cu sistemul de gestiune

Algoritmul de KNN se află într-un fișier Python și rulează în acest mediu, utilizând librării tipice limbajului. Informațiile sunt posibile să fie transmise prin două moduri:

1. Jython - o implementare a limbajului Python în limbaj Java, fiind folosit ca interpretator pentru a compila și executa codul Python dintr-un context Java, dar nu conține suport pentru toate librăriile din Python, una dintre aceste librării este Pandas.
2. Rest API - expunerea unui endpoint pentru realizarea de call-uri HTTP cu frameworkul Flask.

Pentru implementarea acestui proiect s-a ales crearea unui endpoint cu Flask și expunerea acestuia, deoarece în cadrul Jython nu există suport pentru utilizarea librăriei Pandas, librărie ce este utilizată la citirea fișierelor din setul de date și pre-procesarea datelor.

În Flask a fost realizat un endpoint cu o metodă **GET** ce primește ca parametru o listă de nume, listă ce este procesată înainte de a fi trimisă sistemului de recomandare. Se transmite ca răspuns un șir de denumiri ale filmelor alese ca răspuns, urmând ca acestea să fie procesate de sistemul de backend prin utilizarea TMDB Api și ca să fie inserate în baza de date, de unde vor fi accesate de către utilizator.

# Capitolul 6

# Testare şi Validare

Acest capitol prezintă tipurile de testare utilizate pentru verificarea realizării corecte ale funcționalităților sistemului. S-a realizat atât testare manuală cât și testare automată utilizând frameworkuri specializate. De asemenea, testarea s-a bazat atât pe testare funcțională pentru verificarea codului cât și pe testare non-funcțională pentru a verifica securitatea și nivelul de utilizabilitate al aplicației propuse.

## 6.1 Testare funcțională

Testarea funcțională are ca scop testarea caracteristicilor funcționale și funcțiile aplicației. În testarea funcțională s-au realizat testele, din această categorie, prin utilizarea testării automate și a frameworkurilor specifice: JUnit, Selenium, Serenity.

### 6.1.1 Unit Testing

S-a realizat Unit Testing pentru a verifica funcționarea fiecărei metode din aplicație pentru metodele de bază precum CRUD (Create, Read, Update, Delete) utilizând tehnica Cutiei Albe (White Box Testing). Acest tip de testare a fost realizate predominant în timpul dezoltării, astfel că o dată cu dezvoltarea unei noi metode aceasta este verificată mai întâi prin unit testing și mai apoi prin celelalte faze ale testării.

JUnit este frameworkul prin care a fost realizat acest tip de testare. Junit este un mod ușor, rapid și intuitiv prin care se pot realiza teste pentru descoperirea eventualelor probleme de pe partea de backend înainte ca acestea să apară la celelalte faze ale testării. Prezintă posibilitatea creării unor clase de test în care se pot realiza diferite tipuri de testări și care pot comunica între ele. Verificarea ca un test să fi trecut se realizează prin aserțiuni ce reprezintă rezultatul dorit. Dacă

aserțiunile nu sunt corecte, testul va fi fără succes (Failed) altfel, va apărea o bară cu verde ce reprezintă progresul și faptul că testul a trecut ( Passed). Apoi, se trece la următorul test de executat. La finalul execuției tuturor testelor se va afișa un raport cu rezultatele testării.

### 6.1.2 Integration Testing

Testarea de Integrare este următorul pas ce se realizează după ce componenta trece de Unit Testing. Acest tip de testare verifică comportamentul unei componente raportat la celelalte componente permițând descoperirea bugurilor. Prin intermediul JUnit, cu ajutorul unor clase distincte tipului de testare, s-a realizat acesta. S-au verificat testele ce au picat și s-au realizat corecțiile necesare funcționării corecte a codului.

### 6.1.3 Sistem Testing

Testarea de sistem realizează testarea sistemului în totalitatea sa, reprezentând mai mult decât testarea de integrare deoarece este testat tot sistemul realizat până în acel punct. Acest tip de testare s-a realizat pe partea de frontend, prin utilizare Selenium Web driver în scrierea testelor.

S-a utilizat Selenium Webdriver [28] pentru a realiza testarea automată. Selenium este un webdriver ce permite utilizarea diferiților browseri pentru a rula testele automate. Configurarea acestuia de a rula pe diferiți browseri s-a facut în pom.xml (fisier de configurări) iar apelarea acestuia s-a realizat prin crearea unei clase (BaseTest) care este extinsă de toate celelalte clase de test. Această clasă are o singură metodă de setup pentru a rula browserul și a crea diferite configurări filei din browser (maximizare, minimizare, wait). Este adnotată cu @Before pentru a arăta că se face înainte ca testul propriu-zis să înceapă.

Pe lângă Selenium a mai fost folosit si Serenity BDD[29],o librărie Open-Source cu diferite funcții și elemente din care s-au utilizat anumite funcții și elemente precum WebElementFacade, care este un webElement ce se poate găsi în pagină pe baza proprietății (button, input) și a numelui acestuia.

Pentru acest tip de testare am realizat ”Pagini” în backend ce reprezintă paginile prin care trece un utilizator pentru realizarea logării în aplicație. Astfel, au fost necesare 3 pagini:

- HeaderPage reprezintă Headerul de la ecranul de Home, unde se află butonul de LogIn necesar pentru a fi apăsat pentru a trece în ecranul următor.

- LoginPage reprezintă pagina de Login cu toate elementele existente pe ea. Pe această pagina se va face inserarea emailului și a parolei în spațiul corect și apăsarea butonului de Login.

- MyAccountPage este pagina pe care utilizatorul este redirecționat după un login cu succes. Se va verifica acest lucru prin aserțiunea că adresa de email de pe prima pagină este aceeași cu cea introdusă la login.

Exemplul unei pagini folosite mai sus:

Listing 6.1: Pagina de Loginlanguage

```
{
@DefaultUrl("http://localhost:3000/login")
public class LoginPage extends PageObject {
   public LoginPage() {
      super();
   }
   @FindBy(css = "input[name='email']")
   private WebElementFacade emailAddressField;
   @FindBy(css = "input[name='password']")
   priate WebElementFacade passwordField;
   @FindBy(css = "button[type='submit']")
   private WebElementFacade loginButton;
   public void inputEmailValueInField(String email) {
      emailAddressField.waitUntilClickable().sendKeys(email);
   }
   public void inputPasswordValueInField(String password) {
      passwordField.waitUntilClickable().sendKeys(password);
   }
   public void pressLoginButton() {
      loginButton.waitUntilClickable().click();
   }
}

}
```

Apoi, s-a împărțit în pași fiecare acțiune realizată pentru a ajunge la rezultatul final, logarea utilizatorului. Toti acești pași au fost puși intr-un step-group cu denumirea ”PerformLogin” iar fiecare metodă din acest stepgroup este formată din metode de granularitate mai mică, subpași pentru a ajunge la obiectivul final.

### 6.1.4 Regression Testing

Regression Testing reprezintă testarea după ce au fost adăugate funcționalități proiectului pentru a se verifica funcționarea corectă nu doar a noilor funcționalități cât și a celor dezvoltate anterior. Acest tip de testare a fost utilizat și reutilizat pe parcursul dezvoltării aplicației pentru a verifica dacă sistemul funcționează corect

și pentru a corecta eventualele bug-uri ce ar putea apărea fie prin integrarea unei noi componente fie care au scăpat în procesele anterioare.

## 6.2 Testare non-funcțională

Acest tip de testare are ca scop verificarea caracteristicilor non-funcționale ale aplicației pentru a îmbunătăți sistemul. S-au utilizat Positive Testing și Negative testing pentru implementare și validare a aplicației.

### 6.2.1 Testare Pozitivă

Positive Testing are scopul de a verifica dacă funcționalitățile aplicației se comportă conform asteptărilor. S-a realizat acest tip de testare în timpul cazurilor de test, user story-urilor si în cazul testării funcționale.

Tabelul 6.1: Expemplificare Testare Pozitivă

| Tipul inputului | Test Case Description | Remarks |
|---|---|---|
| Email | Trebuie să conțină forma unui input de tip email: valoare@valoare.com orice altă formă nu este acceptată | Inserare valori corecte ale unui email |
| | Trebuie să contină @ și . | 2 Inserare email cu @ și . |
| | Nu sunt permise alte caractere speciale precum # $ % | 3 Introducere doar caractere normale |
| Telefon | Sunt acceptate doar valori numerice, orice alte valori nu vor fi validate | Inserare doar valori numerice |
| | Nu sunt acceptate caractere speciale precum # $ % & * | Nu se vor Insera caractere speciale |
| Password input | Inserare input corect pentru parola specificată contului | Se vor insera maxim 32 caractere sau mai putin |
| | Maxim 32 de caractere pentru parola pot fi inserate | Se vor inser maxim 8 caractere sau mai putin |
| | Minim 8 caractere | |

### 6.2.2 Testare Negativă

Negative Testing are scopul de a verifica dacă functionalitățile aplicației se comportă corect cu toate că datele de test introduse sunt greșite sau invalide. S-a

testat acest lucru prin introducerea datelor invalide în campul de email, număr de telefon sau nume.

Tabelul 6.2: Exemplificare Testare Negativă

| Tipul inputului | Test Case Description | Remarks |
|---|---|---|
| Email | Trebuie să conțină forma unui input de tip email: valoare@valoare.com orice altă formă nu este acceptată | Se încearcă alte forme precum email.com sau parola.com email |
| | Trebuie să contină @ și . | În email lipsește @ sau . |
| | Nu sunt permise alte caractere speciale precum # $ % | Se inserează caractere speciale |
| Telefon | Sunt acceptate doar valori numerice, orice alte valori nu vor fi validate | Se inserează caractere |
| | Nu sunt acceptate caractere speciale precum # $ % & * | Se inserează caractere speciale |
| Password input | Inserare input corect pentru parola specificată contului | Inserare parola gresită |
| | Maxim 32 de caractere pentru parola pot fi inserate | se vor insera mai multe caractere |
| | Minim 8 caractere | Se inserează sub 8 caractere |

### 6.2.3 End-to-End Testing

Reprezintă testarea workflowului aplicației de la începutul creării obiectului până își încheie ciclul de viață. Există tooluri pentru aceste tip de testare precum testCraft si KatalanStudio. Un exemplu de end-to-end testing care a fost realizat pe scenariul ”Obținerea de recomandări unui utilizator autentificat”.

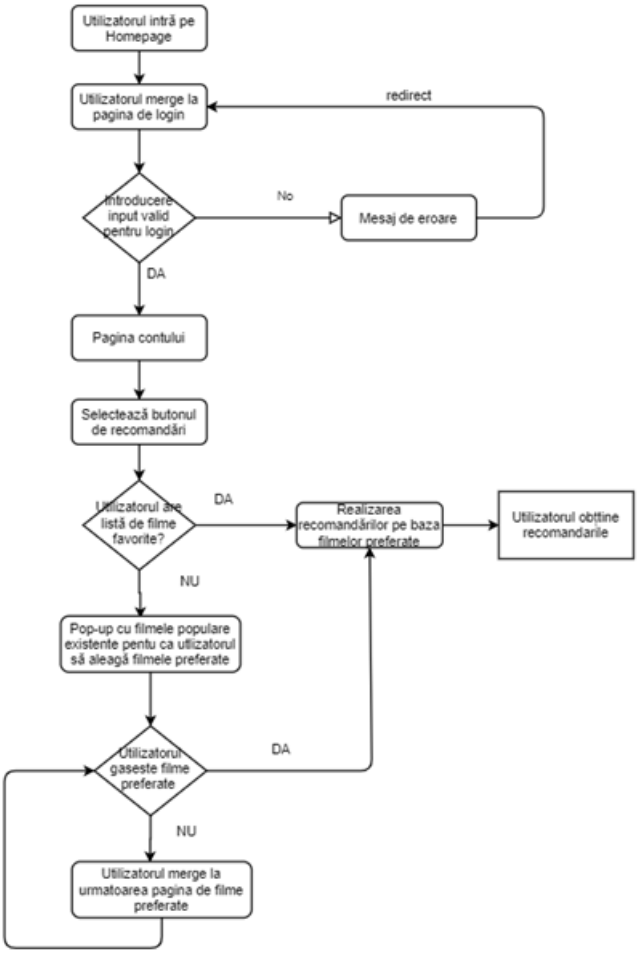


Figura 6.1: Flow de testare end-to-end

## 6.3 User Story

User Stories sunt scurte descrieri ale unei noi funcționalități spuse din perspectiva persoanei ce dorește ca acesta să fie implementat, de obicei un utilizator sau client. Urmează un model strict pentru scrierea acestora:

**Ca utilizator(admin, user) vreau să fac (realizez o acțiune) pentru a (ajunge la obiectivul dorit)**. De multe ori sunt necesare și criterii de acceptanță pentru a defini stadiul de completare al unei funcționalități. Acest tip de testare este specifică proiectelor agile ce se dezvoltă incremental prin adăugarea de noi funcționalități. Acestea au fost realizate în cadrul testării manuale, testare ce s-a realizat pe toată durata dezvoltării acestui sistem și pe care m-am bazat cel mai mult pentru descoperirea de buguri sau probleme. Cele mai importante user-story ce au fost testate sunt următoarele:

1. Ca un utilizator autentificat mă pot loga pentru a accesa recomandările create pentru mine.
2. Ca un utilizator neautentificat pot căuta după denumire pentru a găsi filmul dorit.
3. Ca un utilizator neautentificat pot să mă înregistrez pentru a îmi face un cont.
4. Ca un utilizator admin pot să mă inregistrez pentru vedea toti utilizatorii din baza de date.
5. Ca un utilizator admin pot adăuga filme în baza de date pentru a o menține up-to-date cu noile apariții cinematice.
6. Ca un utilizator autentificat pot marca un serial ca favorit pentru a fi notificat în momentul apariției unui nou episod.

# Capitolul 7

# Manual de Instalare şi Utilizare

În acest capitol vor fi prezentate resursele minime necesare pentru instalarea, rularea și utilizarea aplicației împreună cu programele necesare pentru funcționare și conturile necesare obținerii credențialelor.

## 7.1 Resursele necesare

Resursele necesare pentru hardware sunt reprezentate de un calculator sau un laptop cu minim 8 Gb RAM și spațiul necesar stocării proiectelor. În ceea ce privește necesitățile software, acestea sunt următoarele:

1. Kit de Development: 1.8
2. Framework: SpringBoot, JUnit, Flask, Serenity, Selenium
3. Build Automation: Maven
4. IDEs: IntelliJ IDEA, WebStorm, PyCharm

## 7.2 Manualul de instalare

În acest sub-capitol se vor prezenta pașii necesari pentru a instala backend-ul și credențialele necesare pentru configurare.

Înainte de instalarea Backend-ului sunt necesare configurări ale sistemului de operare, în cazul în care nu sunt deja realizate, urmând pașii de instalare a Java 8 de la adresa următoare de pe oracle și crearea variabilei de mediu JAVA-HOME urmând tutorialul de pe JavaTutorial. Apoi, pentru Maven este necesară download-area de la adresa de maven și configurarea variabilei de mediu MAVEN-HOME. Mai sunt necesare și instalarea limbajului python urmând tutorialul de la `https://www.`

ics.uci.edu/~pattis/common/handouts/pythoneclipsejava/python.html. Pentru instalarea Frontend-ului sunt necesarea instalarea Node.js și npm (Node Package Manager) de la https://nodejs.org/en/download/ urmând pașii de instalare. Apoi, se verifică că instalarea a reușit prin comenzile **node –v , npm –v**.

### 7.2.1 Instalarea Backend-ului

Pentru obținerea proiectului java backend este necesară downloadarea acestuia de pe github, de la adresa https://github.com/GiurgiuDiana/LICENTABACKEND.git fie printr-un pull request fie prin downloadarea proiectului sub format .zip și dezarhivarea acestuia. În folderul rădăcină se deschide un cmd (Command prompt) și se rulează comanda **mvn spring-boot:run**. Pentru proiectul python backend se urmează aceiași pași dar adresa de download este https://github.com/GiurgiuDiana/PYTHON_LICENTA.git. În folderul rădăcină se deschide un cmd și se execută comenzile **flask run**.

### 7.2.2 Instalarea Frontendului

Pentru proiectul de frontend se urmează pașii executați la proiectul de backend, utilizând adresa https://github.com/GiurgiuDiana/frontendLicenta.git. În folderul rădăcină se deschide un cmd și se execută comanda **npm install** pentru instalarea pachetelor dependință și **npm start** pentru pornirea serviciului.

### 7.2.3 Integrarea cu alte API-uri

Se prezintă credențialele necesare ce au fost folosite în acest proiect și obținerea acestora prin diverse platforme. Au fost necesare 3 credențiale:

1. Firebase: pentru accesul la baza de date și pentru utilizarea serviciilor este necesară crearea unui cont pe https://firebase.google.com/ apoi, se accesează consola unde se crează un proiect web, se completează toate câmpurile necesare. După crearea proiectului, se alege tipul de aplicație dorit(Web) și se completează câmpurile necesare. Apoi, din tab-ul ”Project Settings” se obțin credențialele de acces, se salvează într-un fișier .env ce va fi apoi adăugat la proiectul de backend și frontend.

2. TMDB API: pentru accesul la baza de date a TMDB pentru a obține informații se intră pe pagina https://www.themoviedb.org/ se creează un cont nou și se obține cheia de API ce se adaugă la proiect.

3. IMDB API: pentru accesul la baza de date a IMDB pentru a obține informații legate despre trailer, se intră pe pagina https://imdb-api.com/ se creează

un cont și se alege un plan de pricing (simple plan) și se obține cheia necesară realizării conexiunii.

## 7.3 Manual de utilizare

Pentru ca un utilizator să poată accesa aplicația este necesar ca acesta să deschidă un browser web (Chrome, Mozilla, Opera) și să accesseze adresa `http://localhost:3000//`, aceasta este adresa primei pagini, pe prima pagină se află și bara de căutare împreună cu un top al primelor 20 de filme cele mai votate. În bara de navigație apar 5 posibilități de navigare a utilizatorului, navigarea spre Tab-ul cu filmele sau cu serialele populare, cele ce urmează a fi scoase pe piață, filmele sau serialele ce au ieșit recent, loginul și tab-ul de home. Mai jos este o reprezentare a paginii de filme ce urmează a intra pe micile ecrane.

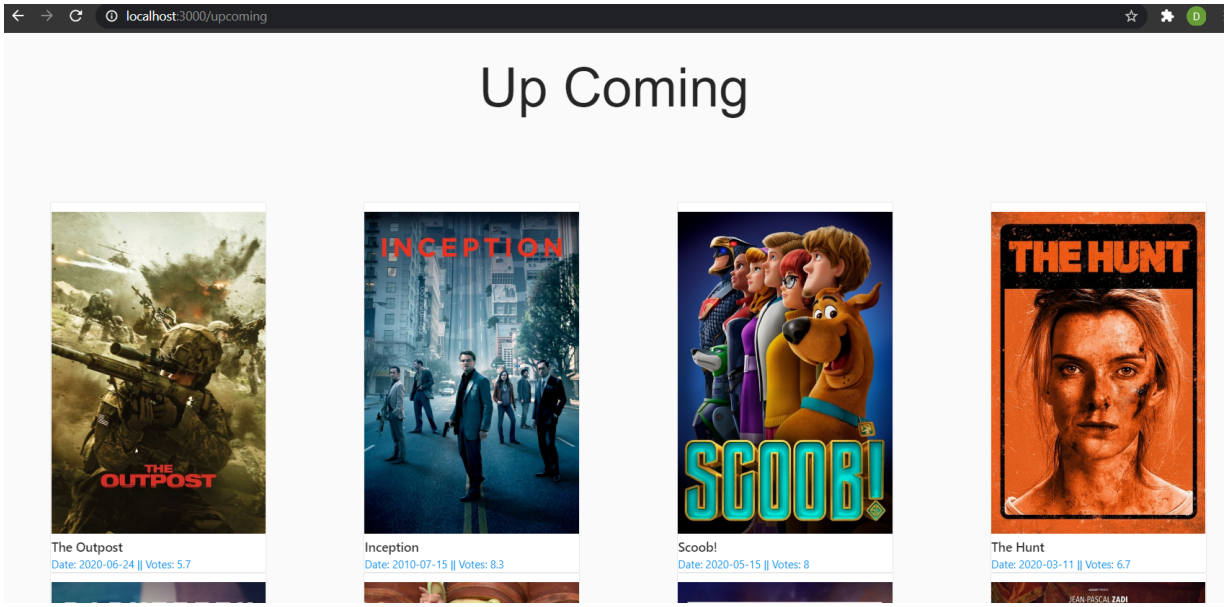


Figura 7.1: Filmele ce vor apărea în curând pe micile ecrane

Utilizatorul ne-autentificat poate alege să caute numele unui film sau serial sau poate vizualiza informațiile despre acesta. Pentru a realiza un review sau pentru a adăuga la o listă filmul, utilizatorul trebuie să se autentifice prin crearea unui cont accesând `http://localhost:3000//register` și completând toate informațiile necesare. Dacă acesta are deja un cont el trebuie doar să se autentifice cu credențialele contului pe pagina de login.

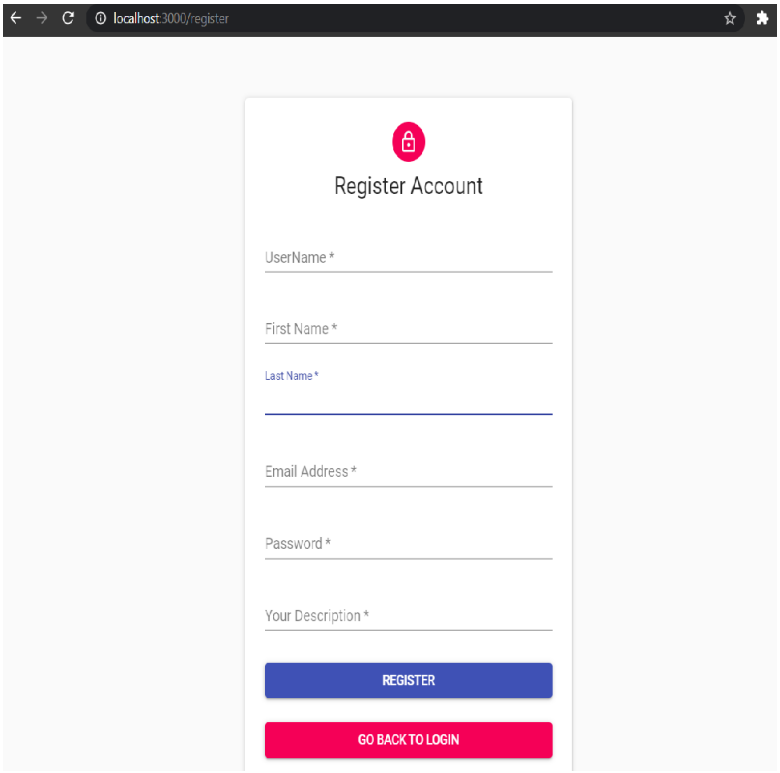


Figura 7.2: Pagina de register

Odată autentificat utilizatorul are posibilitatea de a adăuga filme și seriale în propria listă de favorite și are posibilitatea de a primi recomandări specifice preferințelor acestuia. Uilizatorul mai are posibilitatea de a-și modifica credențialele și de a adăuga rating și comentarii filmelor.

Un utilizator de tip administrator are acces la o pagină de tip dashboard prin intermediul căreia primește controlul sistemului și poate șterge sau modifica elemente. Această pagină permite navigarea către tabele de elemente diferite cu scopul de a le modifica conținutul.

Pentru a obține recomandări este necesar ca utilizatorul să fi adăugat cel puțin câteva filme în lista de favorite. Pentru a adăuga filme în lista de favorite, utilizatorul trebuie să fie logat, acesta poate să vadă dacă este logat de pe orice pagină a aplicației, deoarece îi va apărea numele în colțul din dreapta sus, dacă nu este logat, va apărea ”Login” în colțul din dreapta. Pentru a adauga la favorite un film, utilizatorul trebuie sa apese pe butonul sub forma de inima din dreptul titlului.

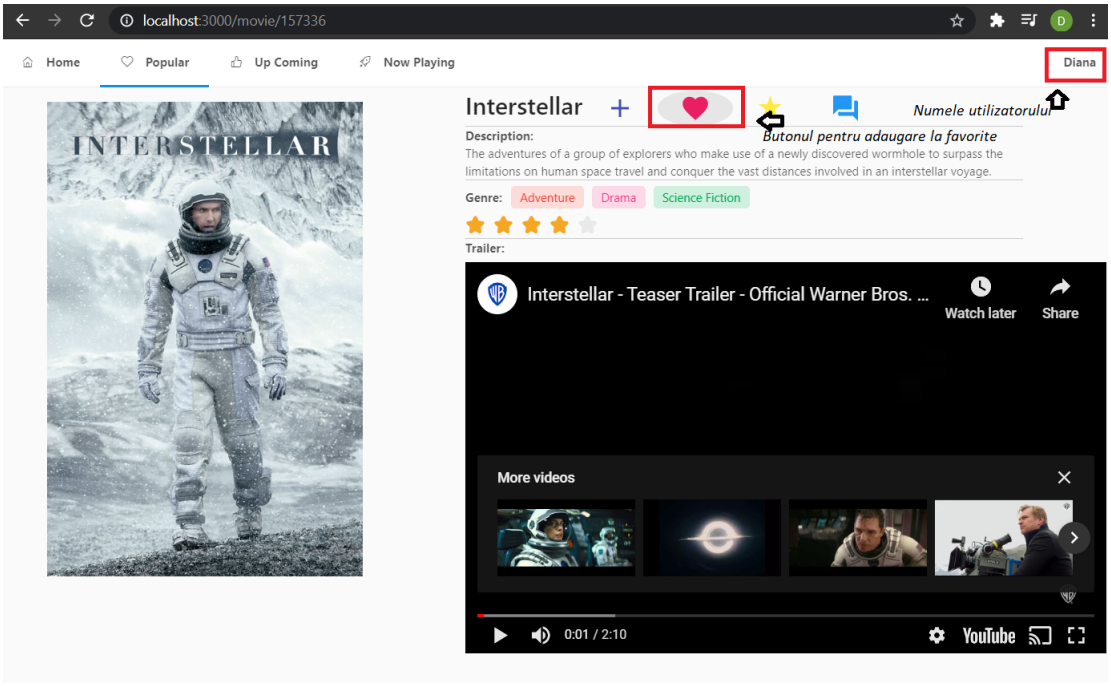


Figura 7.3: Ilustrare a butonului ce trebuie apasat pentru un utilizator logat

Dacă utilizatorul are deja filmele favorite, tot ce trebuie să facă este să apese pe butonul de Get Recs. Acest buton poate fi accesat atât de pe pagina de Dashboard a utilizatorului cât și din dropdownul de sub numele utilizatorului, apoi va fi redirecționat spre pagina de recomandări pentru a vedea filmele ce îî sunt recomandate.

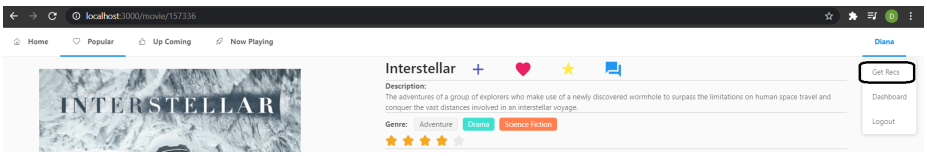


Figura 7.4: Butonul de Get Recs

De aici, utilizatorul poate să apese pe fiecare imagine si va fi redirectionat catre pagina filmului recomandat, ce se afla la adresa http://localhost:3000/recomendations.

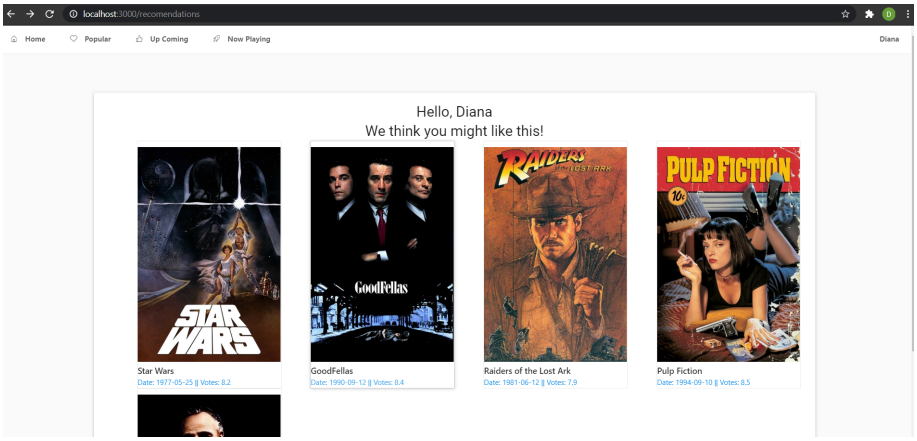


Figura 7.5: Ilustrare a obtinerii de recomandari

# Capitolul 8

# Concluzii

Acest capitol va prezenta concluziile desprinse ca urmare a realizării sistemului făcând referință la obiectivele propuse, stadiul de îndeplinire și observațiile ce s-au realizat pe parcurs. Se vor prezenta și posibile dezvoltări ulterioare ale proiectului cu scopul de a realiza îmbunătățiri și corecturi în cazul apariție problemelor.

## 8.1 Analiza rezultatelor obținute

Obiectivul central al acestui proiect, acela de a realiza un sistem de gestiune și de recomandare al filmelor și serialelor pe baza preferințelor anterioare utilizatorului, a fost îndeplinit, iar nivelul de acuratețe al recomandărilor a fost satisfăcător.

O problemă foarte importantă a sistemului reprezintă lipsa seturilor de date recente pe care să se poată implementa un algoritm de recomandare și care să țină pasul cu evoluția mediului. Soluțiile găsite pentru a minimiza problema lipsei datelor au fost crearea de date dummy, acolo unde acestea nu existau din motive obiective și încercarea imitării comportamentului utilizatorului. O altă problemă a reprezentat maparea aplicației pe API-urile existente pentru a propune o soluție concretă și integrată, ce poate să aducă date de la diferite baze de date și să le îmbine.

## 8.2 Dezvoltări ulterioare

Dezvoltarea care primează este construcția unui set mai extins și mai recent de date ce ar putea fi folosit atât pentru îmbunătățirea preciziei algoritmului cât și pentru extinderea bazei de date. Dar, un set mai extins de date ridică problema procesării și stocării acestora, deoarece realizarea de query-uri sau operații pe acestea se complică. Un răspuns la această problemă ar fi stocarea în Cloud a datelor cu ajutorul funcției Cloud Storage ce este oferită de Firebase.

O altă posibilitate de dezvoltare reprezintă crearea unei aplicații mobile pentru extinderea numărului de utilizatori, de asemenea integrarea mobilă este ușoară datorită bazei de date utilizate, ce conține atât suport mobile cât și suport Web. Mai mult, telefoanele mobile se află într-o continuă evoluție, în fiecare an apar pe piață noi modele, iar marea majoritate a persoanelor consumatoare de conținut media utilizează un telefon smart, prin urmare, dezvoltarea mobile reprezintă o cale sigură pentru a oferi mai multe posibilități de extindere.

O direcție importantă de dezvoltare care apare pentru acest proiect reprezintă integrarea cu o alta platformă ce oferă servicii de streaming, așa cum au făcut sistemele cele mai populare din punct de vedere al utilizatorilor de rând, IMDB, Flixboss, Trakt.tv. Această dezvoltare este una dintre cele mai favorabile sistemului dezvoltat, oferind posibilitatea extinderii fără realizarea de mari schimbări.

În concluzie, această aplicație a fost un mijloc de extindere a cunoștințelor și de studiu al sistemelor de recomandare, oferite pe piață cât și o provocare în implementare datorită utilizării de noi tehnologii.

# Bibliografie


[1] V. Setty, G. Kreitz, R. Vitenberg, M. van Steen, G. Urdaneta, and S. Gimåker, *The hidden pub/sub of spotify*, 06 2013, pp. 231–240. [Online]. Available: https://www.csc.kth.se/~gkreitz/spotifypubsub/spotifypubsub.pdf

[2] D. Jannach, I. Kamehkhosh, and G. Bonnin, *Music Recommendations: Algorithms, Practical Challenges and Applications*, 11 2018, pp. 481–518. [Online]. Available: https://www.researchgate.net/publication/329392345_Music_Recommendations_Algorithms_Practical_Challenges_and_Applications

[3] C. Gómez-Uribe and N. Hunt. (2015, 12) The netflix recommender system. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/2843948

[4] V. Jatana. (2018, 10) Machine learning for beginners. [Online]. Available: https://www.researchgate.net/publication/328614973_Machine_Learning_For_Beginners

[5] Y. Kawahara. (2020, 04) Understanding the underlying algorithms and theories of machine learning. [Online]. Available: https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf

[6] N. F. Z. W. Zan Wang, Xue Yu. An improved collaborative movie recommendation system using computational intelligence. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1045926X14000901

[7] L. E. M. Fernandez. (2018) Recommendation system for netflix. [Online]. Available: https://beta.vu.nl/nl/Images/werkstuk-fernandez_tcm235-874624.pdf

[8] A. Gunawardana and G. Shani. (2009) A survey of accuracy evaluation metrics of recommendation tasks. [Online]. Available: http://jmlr.csail.mit.edu/papers/volume10/gunawardana09a/gunawardana09a.pdf

[9] Knn classification using scikit-learn. [Online]. Available: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

[10] Imdb. [Online]. Available: https://www.imdb.com/

[11] What is prime video?-amazon prime insider. [Online]. Available: https://www.amazon.com/primeinsider/video/prime-video-qa.html

[12] Imdb. [Online]. Available: https://en.wikipedia.org/wiki/IMDb#IMDbPro

[13] Rottentomatoes. [Online]. Available: https://www.rottentomatoes.com/

[14] Flixboss. [Online]. Available: https://flixboss.com/

[15] Trakt.tv. [Online]. Available: https://trakt.tv/

[16] Trakt.tv scrobbler. [Online]. Available: https://trakt.docs.apiary.io/#introduction/terminology

[17] Alexa, ask trakt tv what's on tv today? [Online]. Available: https://blog.trakt.tv/alexa-ask-trakt-what-is-on-tv-today-9820b7a81c8

[18] Firebase. [Online]. Available: https://firebase.google.com

[19] Firebase authetication. [Online]. Available: https://firebase.google.com/docs/auth/web/custom-auth

[20] Firebase firestore. [Online]. Available: https://firebase.google.com/docs/firestore

[21] Tmdb api. [Online]. Available: https://developers.themoviedb.org/3

[22] Imdb api. [Online]. Available: https://imdb-api.com/

[23] Apache maven. [Online]. Available: http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

[24] Spring boot. [Online]. Available: https://spring.io/projects/spring-boot

[25] React tutorial. [Online]. Available: https://reactjs.org/tutorial/tutorial.html

[26] M. Prasad and V. Jarugumalli, *Performance Evaluation: Ball-Tree and KD-Tree in the Context of MST*, 10 2012, vol. 62.

[27] D. Verma, N. Kakkar, and N. Mehan, *Comparison of Brute-Force and K-D Tree Algorithm*, 2014.

[28] Selenium webdriver. [Online]. Available: https://www.selenium.dev/projects/

[29] Serenity. [Online]. Available: http://www.thucydides.info/#/whatisserenity

# Anexa A

# Secţiuni relevante din cod

Listing A.1: Obținerea de recomandări prin HTTP request la endpointul din python

```
public ArrayList<String> getRecsDetails(String imdb_id) throws
    URISyntaxException, IOException {
   URIBuilder builder = new URIBuilder();
   String imdb=imdb_id.replaceAll("The","");
   URL url = new
      URL("http://127.0.0.1:5000/?movieName="+imdb.replaceAll("\\s",""));
   HttpURLConnection con = (HttpURLConnection) url.openConnection();
   con.setDoOutput(true);
   con.setRequestMethod("GET");
   con.setRequestProperty("Content-Type", "application/json");
   BufferedReader br = new BufferedReader(new
      InputStreamReader((con.getInputStream())));
   String output;
   ArrayList<String>onlyTheMovies=new ArrayList<>();
   ArrayList<String[]> array= new ArrayList();
   while ((output = br.readLine()) != null) {
         String[] add=output.substring(1).split(";");
      array.add(add);
   }
   for(String[] temp:array1) {
      for (int i = 0; i < 5; i++) {
         onlyTheMovies.add(temp[i].split(",")[0]);
      }
   }
   return onlyTheMovies;
}
```

# Lista figurilor

# Lista tabelelor