Desenvolupament d'una plataforma web per a la recomanació de pel·lícules

Ferran Martínez Reyes

Resum—Aquest projecte té com a finalitat la creació d'una plataforma web enfocada a millorar l'experiència dels usuaris en la descoberta de noves pel·lícules. Mitjançant l'ús d'un sistema de recomanacions basat en intel·ligència artificial, volem analitzar les valoracions dels usuaris per poder crear recomanacions personalitzades de pel·lícules. La implementació comprèn la recopilació de dades de pel·lícules a partir de l'ús d'una API (The Movie Database), el disseny i implementació tant del client com dels servidors de la web i del sistema, la creació d'una interfície d'usuari amigable, el desenvolupament d'un algorisme KNN que serà l'encarregat de generar les recomanacions per als usuaris, i la publicació de la web a internet mitjançant docker i AWS. L'avaluació del sistema també serà una part important per mesurar l'eficàcia de l'algorisme i la satisfacció dels usuaris.

Paraules Clau —Plataforma Web, Cinema, Recomanacions, API, The Movie Database, Interfície d'Usuari, Algorisme KNN, Pel·lícules

Abstract—This project aims to create a web platform focused on enhancing users' experience in discovering new movies. By utilizing an artificial intelligence-based recommendation system, we seek to analyze user ratings to generate personalized movie recommendations. The implementation involves collecting movie data through an API (The Movie Database), designing and implementing both the web and system servers, creating a user-friendly interface, developing a KNN algorithm responsible for generating user recommendations, and deploying the website on the internet using Docker and AWS. Evaluating the system will be crucial to measure the algorithm's effectiveness and user satisfaction.

Keywords—Web Platform, Cinema, Recommendations, API, The Movie Database, User Interface, KNN Algorithm, Movies

1 Introducció – Context del treball

Amb la creixent disponibilitat de plataformes de streaming i la infinita varietat de pel·lícules disponibles, sovint és difícil trobar aquella pel·lícula perfecta que s'ajusti als nostres gustos personals. Cada cop ens passem més estona buscant una pel·lícula que ens captivi que no pas gaudint-la. És aquí on rau la importància d'aquest projecte, minimitzar aquest temps de cerca i simplificar la vida dels espectadors. El que ens cal per dur-ho a terme és oferir una experiència personalitzada per a cada usuari.

En moltes ocasions, hem observat que els nostres amics tenen èxit en les seves recomanacions de pel·lícules, ja que coneixen els nostres gustos i preferències. Aquesta experiència personalitzada és la clau per a una bona recomanació. No obstant això, no sempre podem dependre dels altres per obtenir suggeriments. És aquí on aquest projecte entra en joc, amb la creació d'una eina que conegui els gustos dels usuaris i generi recomanacions automàticament.

A través d'aquest projecte, s'espera aportar una petita millora a l'experiència d'usuari d'aquells amants del cinema que es troben indecisos sobre què veure, oferint-los un sistema que els conegui millor i els proporcioni recomanacions més precises.

2 Objectius

Els objectius del projecte es divideixen en diverses etapes amb l'objectiu de garantir una implementació completa i eficaç del sistema de recomanació de pel·lícules:

Objectiu 1: Recopilació de dades

Aquest objectiu implica la recopilació d'informació rellevant de pel·lícules mitjançant l'ús d'una API, en aquest cas, The Movie Database. Es buscarà una àmplia varietat de dades, inclosos valoracions d'usuaris, gèneres, directors i altres elements importants per al sistema.

• *E-mail de contacte: 1565491@uab.cat*
• *Menció realitzada: Enginyeria del Software*
• *Treball tutoritzat per: Sergio Bachiller Rubia.*
• *Curs 2023/24*

Objectiu 2: Desenvolupament del frontend

En aquest objectiu, es procedirà amb el disseny i desenvolupament de la interfície d'usuari de la plataforma web. L'objectiu és crear una interfície atractiva, fàcil d'utilitzar i que permeti als usuaris explorar i valorar les pel·lícules de manera intuïtiva. Es tindran en compte aspectes com la navegació, la presentació de dades i l'experiència global de l'usuari.

Objectiu 3: Desenvolupament del backend

Aquest objectiu implica la creació de la part del servidor de la plataforma web, que processarà les dades recopilades i implementarà l'algorisme de recomanació KNN. Es garantirà que el backend sigui eficient, escalable i sigui capaç de gestionar les sol·licituds del frontend de manera efectiva. A més, s'implementaran funcionalitats com la gestió d'usuaris amb JWT i l'emmagatzematge de dades amb MongoDB.

Objectiu 4: Optimitzacions i desplegament

També tenim com a objectiu l'optimització tant del frontend com del backend per millorar el rendiment general de la plataforma. Així com a assegurar-nos que les recomanacions satisfan els nostres usuaris. Posteriorment, es procedirà amb el desplegament del sistema a través de Docker i AWS per permetre l'accessibilitat de la plataforma en línia.

3 Estat de l'art

El camp de la recomanació de pel·lícules és una àrea amb constants avenços. A continuació, es presenta una revisió detallada dels mètodes i tecnologies utilitzades en projectes semblants, analitzant algorismes, eines i solucions arquitectòniques. A més, es destaca com aquest projecte es diferencia en aquest context.

3.1 Algorismes de recomanacions

3.1.1. Filtratge Col·laboratiu

És un mètode que fa recomanacions basades en les preferències d'usuaris semblants. Els algorismes de Filtratge Col·laboratiu estan basats en l'usuari.

3.1.2. Filtratge Basat en Contingut

Aquest mètode es basa en les característiques de les pel·lícules, com per exemple el gènere, l'any, l'idioma, els actors, i les preferències que té l'usuari sobre aquestes.

3.1.2. Filtratge Híbrid

Integra diversos mètodes, combinant avantatges del filtratge col·laboratiu i de filtratge basat en contingut. Això millora precisió del model.

3.2. Tecnologies Sovint Utilitzades

3.2.1. Bases de Dades de Pel·lícules:

La majoria dels projectes utilitzen una API com The Movie Database o IMDb per obtenir informació actualitzada sobre les pel·lícules, incloent-hi valoracions, sinopsi, el repartiment, i altres metadades.

3.2.2. Frameworks Frontend:

React.js és àmpliament utilitzat per desenvolupar interfícies d'usuari interactives i reactives. Altres opcions com Angular o Vue.js també són comunes.

3.2.3. Frameworks Backend:

Node.js és popular pel seu rendiment escalable en el desenvolupament del backend. Flask o Django amb Python són també opcions freqüents.

3.3. Solucions de Recomanacions Avançades

Grans empreses, com ara Netflix, incorporen models de machine learning més avançats, com ara xarxes neuronals, per entendre patrons complexos en les preferències dels usuaris. Això permet oferir recomanacions més precises. Alguns dels factors que també es tenen en compte per a les recomanacions en aquest àmbit són:

- Historials de visualització
- Comportament de clics i valoracions
- Hora del dia
- Ubicació geogràfica
- Feedback de les recomanacions

4 Metodologia i eines

Per al desenvolupament de la part del Client del nostre projecte hem fet servir les tecnologies web bàsiques com HTML, CSS i JavaScript per crear la interfície d'usuari. A més, optarem per utilitzar React com a framework principal per a la creació d'una aplicació d'una sola pàgina (SPA). Ja que React té molta flexibilitat a l'hora de gestionar components reutilitzables i actualitzacions eficients del DOM. A més, no cal que ens iniciem en typescript (com és el cas d'Angular) i té una corba d'aprenentatge molt més fàcil.

Pel que fa a la base de dades, hem optat per utilitzar MongoDB, ja que té una gran capacitat per gestionar dades no estructurades, com les dades de les pel·lícules i les valoracions dels usuaris. MongoDB està basat en documents JSON, que podem adaptar fàcilment a les necessitats del nostre projecte, i això ens permetrà emmagatzemar i consultar les dades de manera eficient.

Pel que fa a l'arquitectura del projecte, l'objectiu és seguir el patró Model-Vista-Controlador (MVC) amb l'objectiu de mantenir el codi ben estructurat i fàcilment mantenible. Aquesta arquitectura divideix l'aplicació en tres parts principals: el Model, que gestiona les dades i la lògica de negoci; la Vista, que és responsable de la interfície d'usuari; i el Controlador, que coordina les interaccions entre el Model i la Vista.

Un aspecte crític per al nostre projecte és la necessitat de

disposar de dades reals de pel·lícules per això es va optar per l'ús d'una API de base de dades de pel·lícules, en el nostre cas l'API de la pàgina <<themoviedatabase.org>>. Amb una API com aquesta vam poder millorar la quantitat de pel·lícules de la web, així com una més fàcil obtenció de la imatge de portada o d'altres metadades de la pel·lícula.

A l'inici del projecte es va optar a buscar bases de dades de pel·lícules i valoracions d'usuaris autèntiques i existents, però davant l'escassetat d'aquestes a la web es va optar finalment per introduir les valoracions a mà amb ajuda de les opinions de familiars i amics.

Pel que fa al backend utilitzarem Node.js per crear una RESTful API que gestionarà les sol·licituds del client i comunicarà amb la nostra base de dades MongoDB. A més, farem servir Python i Flask per a facilitar la creació d'un servidor a part que s'encarregarà del sistema de recomanació.

El desenvolupament del projecte es realitza en un entorn Windows, utilitzant WebStorm. A més, es manté un repositori Git a GitHub per controlar i gestionar el codi font.

Respecte als mètodes de gestió de projectes, s'utilitzarà el mètode àgil Kanban, molt útil per a projectes amb un enfocament més flexible i que no requereixen un equip com en Scrum, sinó que un sol desenvolupador pot treballar-hi. Per això, utilitzarem l'eina Trello, un sistema de gestió de tasques que ajuda a dur un seguiment de quines tasques s'han de dur a terme en cada moment.

5 Anàlisi de requisits del sistema

El projecte, com cap altre, ha de començar per l'anàlisi de requisits del sistema que volem desenvolupar, aquesta és una fase crucial del cicle de vida del desenvolupament del software i implica la identificació, documentació i validació de les funcions i característiques que el sistema ha de tenir. Només d'aquesta manera podrem garantir que el disseny del sistema sigui la més adequada per satisfer totes les necessitats del sistema.

Els requisits del sistema es classifiquen en funcionals i no funcionals

5.1 Requisits Funcionals

Descriuen les funcionalitats específiques que el sistema ha de proporcionar.

- **RF1: Registre d'Usuaris:** El sistema ha de permetre als usuaris registrar-se mitjançant un nom d'usuari i una contrasenya

- **RF2: Inici de Sessió:** El sistema ha de permetre als usuaris iniciar sessió amb les seves credencials.

- **RF3: Valoració de Pel·lícules:** Els usuaris han de poder valorar les pel·lícules mitjançant una puntuació o una valoració numèrica.

- **RF4: Emmagatzematge de Valoracions:** Les valoracions dels usuaris han de ser emmagatzemades en la base de dades.

- **RF5: Generació de Recomanacions:** El sistema ha de ser capaç de generar recomanacions de pel·lícules basades en les valoracions dels usuaris i l'afinitat amb altres usuaris.

- **RF6: Consulta de Pel·lícules:** Els usuaris han de poder buscar i veure informació sobre les pel·lícules disponibles al sistema.

- **RF7: Gestió de Compte:** Els usuaris han de poder gestionar el seu compte, inclosa la modificació de contrasenya i la configuració del perfil.

- **RF8: Interfície d'Usuari Intuïtiva:** L'aplicació ha de proporcionar una interfície d'usuari fàcil d'entendre i d'utilitzar.

5.2 Requisits No Funcionals

En lloc de descriure què fa el sistema, defineixen com ha de ser el sistema

- **RNF1: Seguretat:** Les dades dels usuaris, com les contrasenyes i les valoracions, han de ser emmagatzemades de manera segura i protegides mitjançant autenticació i autorització.

- **RNF2: Rendiment**: El sistema ha de ser eficient i capaç de gestionar una gran quantitat de dades i usuaris sense un temps excessiu de càrrega.

- **RNF3: Disponibilitat:** El sistema ha de ser altament disponible per als usuaris, amb una disponibilitat propera al 100%.

- **RNF4: Precisió de les Recomanacions:** Les recomanacions han de ser precises i rellevants per als usuaris, basades en les seves valoracions i les de similars.

- **RNF5: Interfície d'Usuari Responsiva:** L'aplicació ha de ser accessible des de dispositius mòbils i altres pantalles petites amb una interfície d'usuari responsiva.

- **RNF6: Gestió de Sessions**: La gestió de sessions ha de ser eficaç per garantir la seguretat i la persistència de la sessió de l'usuari.

6 Disseny de la base de dades

6.1 Estructura de la BD

La base de dades MongoDB utilitzada per al projecte consta de diverses col·leccions que emmagatzemen dades essencials per al seu funcionament.

USERS
Emmagatzema informació d'usuaris com ara correu electrònic, nom d'usuari (valor unic, identificador irrepetible) i contrasenya (emmagatzemada amb hash)

MOVIERATINGS
Registra les valoracions d'usuaris per a les pel·lícules, incloent-hi identificadors, valoracions i usuari associat.

RECOMMENDATIONS
Col·lecció que utilitzarà l'algorisme de Python per emmagatzemar les recomanacions de pel·lícules dels usuaris un cop s'hagi fet el càlcul.

MOVIES
Conté dades relacionades amb les pel·lícules, incloent-hi identificadors, títols i altres atributs.

Nota 1: Val la pena explicar que no totes les pel·lícules existents estaran a la nostra base de dades. La nostra aplicació utilitza TMDB per a cercar-les i només les afegim a la nostra BD per emmagatzemar el seu ID un cop un usuari ha valorat una pel·lícula.

Nota 2: Les valoracions es vinculen a les pel·lícules i als usuaris a través d'identificadores únics, creant relacions que faciliten l'extracció de dades relacionades quan es necessiti.

6.2 Recuperació de dades

Només els usuaris autenticats tenen accés a certes funcionalitats, com l'emmagatzematge de valoracions o a obtenir recomanacions de pel·lícules

7 Disseny i arquitectura del sistema

Pel que fa a l'estructura del sistema, per un lloc desenvoluparem un servidor RESTful API que gestionarà les sol·licituds del client i es comunicarà amb la nostra base de dades MongoDB. Farem servir Node.js amb l'ús del framework Express per crear-la.

Per altra banda, el generador de recomanacions s'implementarà utilitzant Python i el framework Flask, Hem optat per utilitzar Python per a aquesta tasca, ja que ofereix eines i llibreries ben desenvolupades per a tasques de processament i anàlisi de dades, cosa que facilita la creació del sistema de recomanació.

Un aspecte destacat del nostre projecte és la modularitat. Tant el RESTful API com el generador de recomanacions es poden desenvolupar de manera independent. Això significa que podem treballar en aquestes dues components de manera separada i, després, unir-los fàcilment a través del protocol HTTP. Aquesta flexibilitat ens ha permès adaptar el desenvolupament a les necessitats del projecte.

A continuació podem es mostra el diagrama de components del sistema:

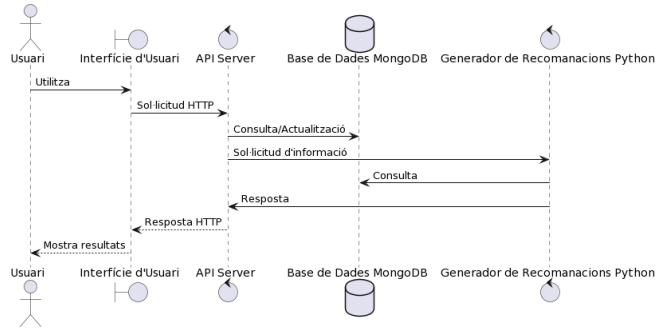
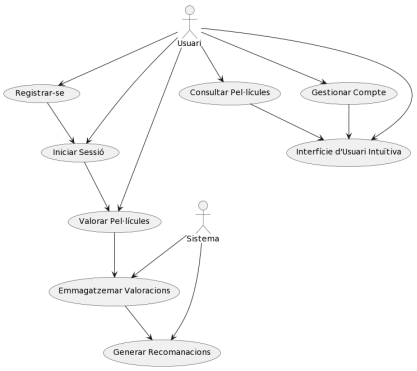


Fig. 1 i 2. Diagrama de components del sistema (a dalt) i diagrama d'especificació de sistema (a baix) (Font pròpia)



Abans de començar amb el desenvolupament de la web vam començar amb aquest disseny que és el que hem utilitzat per tal de tenir una guia a l'hora de crear l'APP:

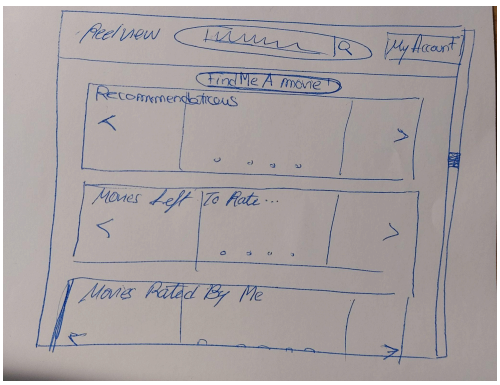


Fig. 3. Disseny preliminar de la idea principal de la pàgina inicial de l'aplicació (Font pròpia)

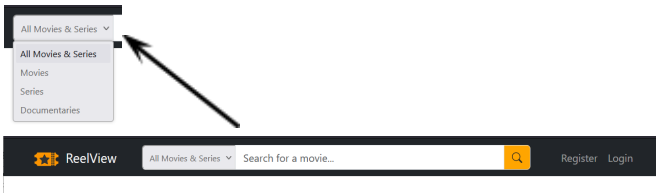
Veiem que Tenim 3 carrusels que ens mostren les recomanacions, les pel·lícules per valorar i les ja valorades. Així com la barra de cerca i altres elements de la web.

8 Desenvolupament

8.1 FrontEnd

8.1.1. Elements Importants

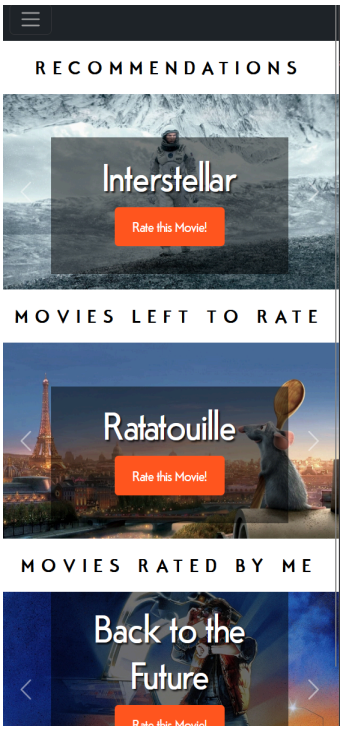
Barra de Cerca i navegació:
La barra de cerca és una funció clau que permet als usuaris trobar ràpidament les pel·lícules desitjades.
Permet a l'usuari accedir a les opcions del seu compte, registrar-se, iniciar sessió, tancar sessió, etc.
Permet filtrar per pel·lícules, sèries, documentals, etc.



Pàgina Inicial amb Carrusels de pel·lícules
Els carrusels de la pàgina principal permet als usuaris explorar les pel·lícules sense haver de realitzar una cerca específica.

Hi trobem tres diferents:

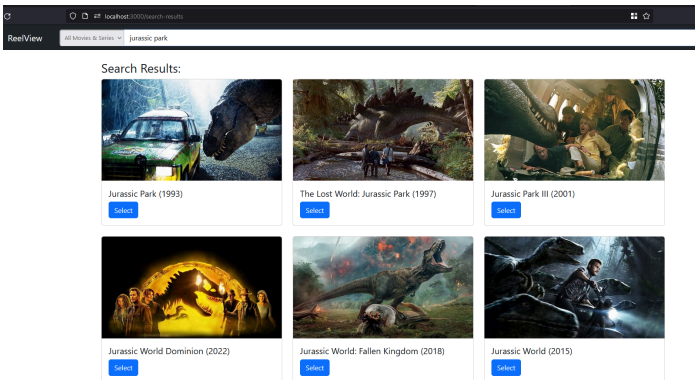
- Un carrusel presenta les pel·lícules més recomanades per a aquest usuari.
- Un altre carrusel amb pel·lícules pendents per valorar per aquest usuari
- Un altre amb les pel·lícules ja valorades per aquest usuari



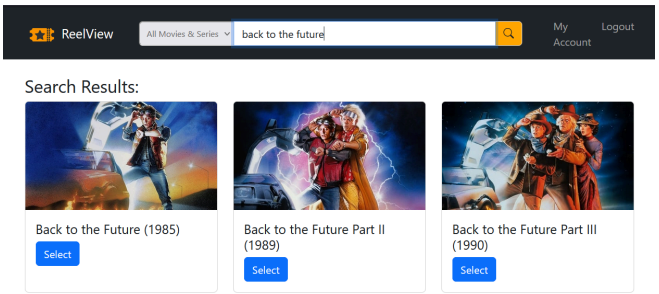
Pàgina de resultats de cerca

La pàgina de resultats de cerca mostra resultats detallats amb imatges i informació rellevant sobre les pel·lícules relacionades amb la cerca de l'usuari.
Facilita l'accés ràpid als detalls i a la valoració de les pel·lícules trobades.

Cerca per a Jurassic Park:

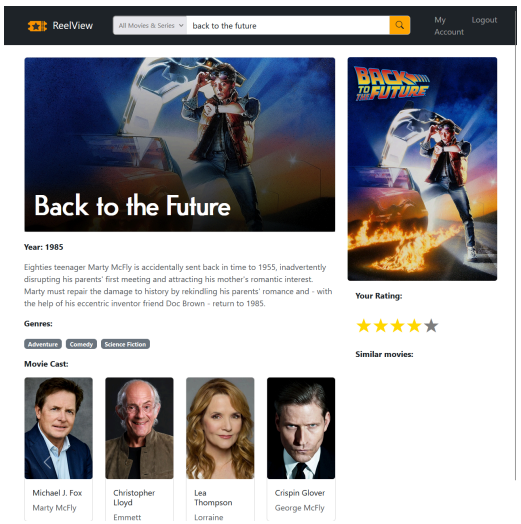


Cerca per a Back To The Future:



Pàgina de detalls de pel·lícula

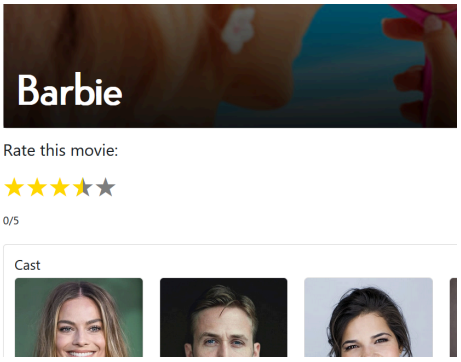
Les pàgines de detalls de les pel·lícules ofereixen una visió completa amb informació com la sinopsi, el repartiment, els generes, etc.



Gestió de valoracions

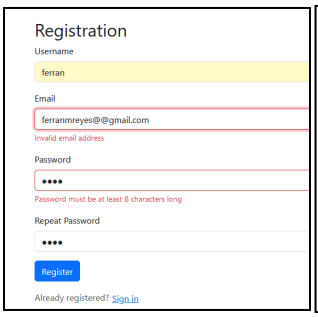
Una interfície fàcil d'utilitzar permet als usuaris valorar les pel·lícules directament des de la pàgina de detalls.

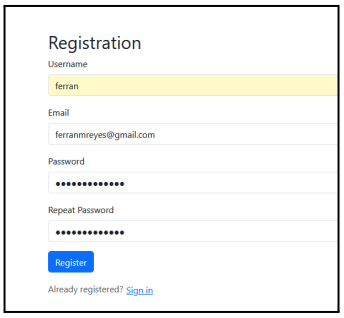
Utilitza una llibreria externa per a react anomenada "react-rating-stars-component" per a implementar el sistema de valoracions basat en un rànquing de 5 estrelles:

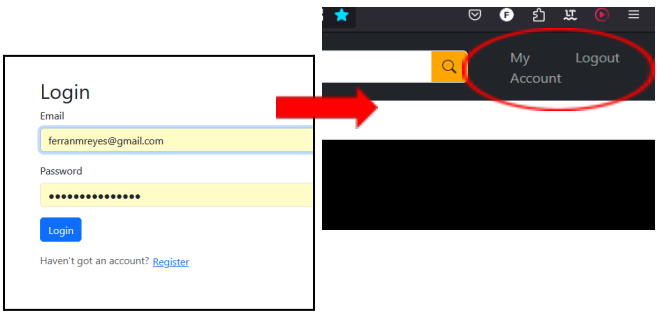


Pàgines de registre i inici de sessió

- Mostren errors si els camps no s'omplen bé
- Només permeten el registre i inici de sessió si tots els camps són com haurien de ser i consulten a la base de dades de l'existència de l'usuari.







Un cop iniciada la sessió ja podem valorar pel·lícules, ja que sense iniciar sessió no es generarà cap token i, per tant, no podrem cridar a rutes protegides, com la ruta de valoració de pel·lícules.

Sabem que hem iniciat sessió perquè ens redirigeix a la pàgina principal i a la barra de navegació ens mostra l'opció de tancar la sessió.

8.1.2. Eines utilitzades al client

Axios:

És una llibreria JavaScript que facilita la realització de peticions HTTP des de clients basats en navegador o des de Node.js. És molt utilitzada en aplicacions React per interactuar amb servidors per obtenir o enviar dades.

Bootstrap:

És una llibreria de disseny front-end que ofereix un conjunt de components i estils CSS predefinits per a la creació d'interfícies d'usuari sense necessitat d'escriure moltes línies de codi personalitzat.

React Router DOM:

És una llibreria React que facilita la implementació del sistema de navegació en aplicacions React basades en el navegador. Permet definir rutes per a diferents components i gestionar el canvi de pàgines sense recarregar la pàgina completa.

React Rating Stars Component:

Llibreria React que facilita la creació i incorporació de components d'estrelles de valoració interactiva a les aplicacions React.
És útil per permetre als usuaris indicar les seves valoracions de pel·lícules o altres elements amb una interfície gràfica.

Exemple d'ús:

```
const RatingComponent = () => {
  return (
        <StarRating
        count={5}
        value={4}
        onChange={(newValue) =>
console.log(`Nova valoració: ${newValue}`)}
        size={24}
        activeColor="#ffd700"
        />
  );
};
```

8.2 BackEnd

8.2.1. Implementació del Servidor API REST:

Descripció de l'Arquitectura:

El servidor API REST implementat utilitza Express.js com a framework de Node.js. El servidor escolta al port 5000 i incorpora el middleware CORS per gestionar el compartiment de recursos entre dominis.

Decisions de Disseny i Patrons:

- **Principis RESTful**:

  El servidor segueix els principis RESTful organitzant rutes per a diferents funcionalitats com ara autenticació, valoracions d'usuaris i cerca de pel·lícules.

- **Escala**:

  L'estructura del servidor està dissenyada per ser modular, permetent fàcilment la incorporació de rutes i middleware per a una escalabilitat futura. Express.js, amb el seu disseny minimalista, facilita l'addició de noves funcionalitats.

- **Mesures de Seguretat**:

  La seguretat és una prioritat amb la inclusió de bcrypt per al hash de contrasenyes durant el registre i l'inici de sessió d'usuaris. S'utilitzen "JSON Web Tokens" (JWT) per a l'autenticació d'usuaris, assegurant la transmissió segura de la informació confidencial.

8.2.2. Implementació de l'API The Movie Database (TMDB):

El servidor es connecta amb la base de dades de pel·lícules The Movie Database (TMDB) per tal de poder accedir a una base de dades amb una gran quantitat de pel·lícules disponibles.

Com que TMDB ofereix totes les metadades de les pel·lícules: resums, imatges, pòsters, actors… ens és molt útil, ja que introduir pel·lícules de manera manual seria un procés molt tediós, i val més la pena reutilitzar bases de dades ja existents per tal de construir la nostra.

La integració comença amb l'obtenció d'una clau d'API de TMDB. Aquesta clau és necessària per fer sol·licituds autenticades a l'API de TMDB. La clau d'API s'utilitza durant la instanciació de l'objecte moviedb, permetent al servidor fer sol·licituds autoritzades a l'API de TMDB.

Maneig de les Respostes de l'API de TMDB (Filtre):

És crucial gestionar les respostes de l'API de TMDB de manera efectiva. Això implica analitzar les dades de TMDB, extreure informació rellevant i presentar-la en un format amigable per a l'usuari.

A TMDB hi ha milers de pel·lícules, i no ens interessa mostrar aquelles que no són res conegudes, ja que fan que la cerca estigui plagada de resultats que no valen la pena.

De manera que he creat un filtre per a mostrar només aquelles que tinguin una popularitat major a 20% o que tinguin una valoració mitjana de 6 (sempre que hi hagi almenys 3000 vots)

```
const movies = movieResults.results
        .filter(movie => movie.popularity >
20 || (movie.vote_average > 6 &&
movie.vote_count > 3000))
        .map(movie => ({
        id: movie.id,
        original_title: movie.original_title,
        backdrop_path: movie.backdrop_path,
        release_date: movie.release_date,
        }));
```

El filtre està obert a millores, però he trobat que aquest és un bon mètode per a mostrar només les pel·lícules més conegudes o aquelles amb valoracions relativament altes.

8.2.3. Eines utilitzades al Servidor API REST

Bcrypt:

La llibreria bcrypt s'ha integrat per gestionar de manera segura la hash de les contrasenyes dels usuaris.
Abordar la seguretat de les contrasenyes és crucial en qualsevol aplicació web. Amb bcrypt, s'assegura un emmagatzemament segur i eficaç de contrasenyes mitjançant l'algorisme de hash. Aquesta mesura ajuda a prevenir vulnerabilitats i protegir les dades dels usuaris contra possibles atacs.

JsonWebToken (JWT):

JSON Web Tokens (JWT) és un estàndard àmpliament utilitzat per autenticar i autoritzar usuaris sense emmagatzemar dades de sessió al servidor, ja que aquestes contenen informació delicada de l'usuari.

JWT s'utilitza per verificar que l'usuari és qui diu ser i està autenticat i autoritzat per accedir al recurs sol·licitat (és a dir, les rutes protegides per autenticació).

8.2.4. Ús de JWT i autenticació

A les aplicacions web tradicionals, quan iniciem sessió, el servidor emmagatzema informació sobre nosaltres en una base de dades o caché i ens envia un identificador de sessió que normalment s'emmagatzema en una cookie. En aquest cas, cada vegada que sol·licitem un recurs, el servidor cerca les dades de la nostra sessió a la base de dades mitjançant l'ID de sessió per comprovar si estem autenticats i autoritzats per accedir.

Amb JWT, en canvi, quan un usuari inicia sessió, el servidor crea un token que conté informació sobre l'usuari. En lloc d'emmagatzemar les dades de la sessió al servidor, s'envia el token al client. Aleshores, el client emmagatzema el token localment. A partir d'aquí, el client inclou el token a les seves sol·licituds futures per demostrar la seva autenticació

Quan ha passat un temps determinat i el client intenta accedir a un recurs protegit, aquest pot rebre un error HTTP 401 (Unauthorised) que indica al client que ja no pot accedir a aquest recurs perquè el token ha caducat. És quan llavors, el client ha d'enviar una sol·licitud de refresh perquè el servidor li enviï un nou token vàlid. El client ha de tornar a demanar el recurs sol·licitat automàticament amb el nou token:

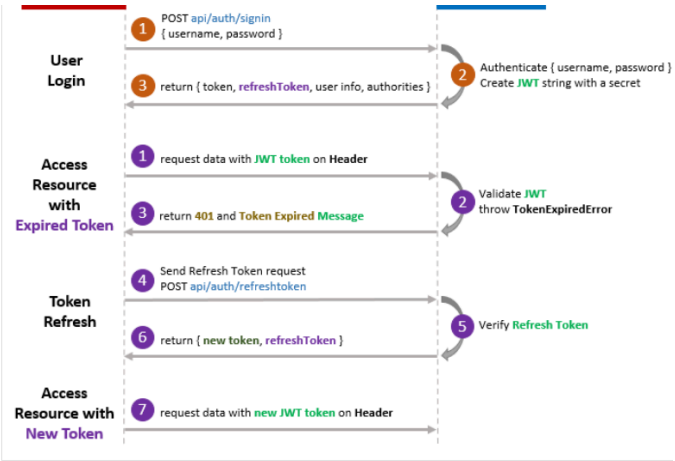


Fig. 5 Diagrama que mostra el flux d'execució del refresh d'un token (Font: https://www.bezkoder.com/angular-12-refresh-token/) [29]

Implementació del Mecanisme de Refresh Token i JWT des del SERVER:

1. Generació i Enviament del JWT (Inici de sessió):

Quan l'usuari inicia la sessió amb èxit, el servidor genera un JWT que conté informació com l'ID d'usuari, el mail i el nom d'usuari. Aquest token és signat amb una clau secreta i s'envia al client perquè l'emmagatzemi localment.

2. Verificació del JWT (Middleware verifyAccessToken):

Les rutes protegides, com "/api/saveRating", requereixen l'autenticació per mitjà del JWT, ja que usuaris no autenticats no poden valorar pel·lícules. El middleware verifyAccessToken verifica si el token proporcionat pel client és vàlid mitjançant la clau secreta. Si la verificació és exitosa, es permet l'accés a la ruta protegida. En cas contrari, es retorna un error d'accés denegat o un error de token no vàlid.

La següent ruta és una ruta protegida per la funció middleware "*verifyAccessToken*":

```
app.post( path: '/api/saveRating', verifyAccessToken, async (req, res) => {
    try {
        const { movieId, userID, userRating } = req.body;

        const client = await MongoClient.connect(url,  options: { useNewUrlPar
        db = client.db( dbName: 'reelviewDB');
        const ratingsCollection = db.collection( name: 'movieRatings');
```

3. Actualització del JWT mitjançant el Refresh Token:

Quan el token d'accés caduca (és a dir, quan s'arriba al temps límit definit), es pot refrescar i obtenir un de nou sense haver de tornar a iniciar la sessió. Per a això tenim una ruta al servidor "/auth/refresh" que rep un token i, mitjançant la verificació de JWT, genera un de nou amb un temps límit renovat. Això permet que l'usuari continuï autenticat sense haver de proporcionar les credencials cada cop que es caduca.

Implementació del Refresh Token i JWT des del CLIENT:

Quan es fa una sol·licitud a una ruta protegida mitjançant un token d'accés expirat, el client pot utilitzar un refresh token per obtenir un nou token d'accés sense necessitat de tornar a iniciar la sessió. Aquí s'expliquen els passos realitzats pel client:

1. Enviar el Token d'Accés amb la Sol·licitud Original:

Si per exemple intentem puntuar una pel·lícula cridant a l'URL del servidor corresponent. Com es tracta d'una ruta protegida, hem de passar el token al header del nostre POST dins de l'atribut "Authorization".

```
// Tornem a intentar la sol·licitud original amb el n
const retryResponse = await fetch(SAVE_RATING_URL, {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
        'Authorization': `${refreshedToken}`,
    },
    body: JSON.stringify({
        movieId: movie.id,
        userID: username,
        userRating: newRating,
    }),
});
```

2. Gestionar la Resposta en Cas d'Expiració del Token d'Accés:

En aquesta etapa, si la resposta indica que el token ha expirat (codi d'estat 401), el client crida la funció "refreshAccessToken" per obtenir un de nou. A continuació, es torna a intentar la sol·licitud original amb el nou token d'accés.

En un component a part (authUtils.js), amb la intenció de què es pugui reutilitzar a qualsevol altre component de l'aplicació, tenim la funció "*refreshAccessToken*", que és la que s'encarrega de fer la sol·licitud a la ruta "/auth/refresh"
Un cop tanquem la sessió s'eliminarà el token del navegador, juntament amb l'ID d'usuari que teníem guardat.

8.3 Server Python

8.3.1 Algorisme de recomanacions

En el nostre cas, l'algorisme de recomanacions és del tipus Filtratge Col·laboratiu. És a dir, ens basem en el fet que si un usuari A i un usuari B puntuen positivament diverses pel·lícules podem concloure que ambdós usuaris són semblants, i per tant una pel·lícula que ha agradat a l'usuari A, molt probablement agradarà a l'usuari B:

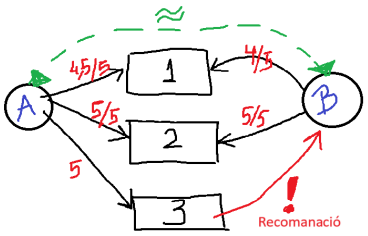


Fig. 6 Dibuix que mostra com un usuari A valora pel·lícules (1, 2) de manera similar a B, i com a conseqüència el sistema genera una recomanació de la pel·lícula 3 a l'usuari B (Font Pròpia)

El nostre sistema està destinat a ser usat per un gran nombre de persones, per tant, per buscar similituds entre usuaris hem d'utilitzar algorismes més avançats, com és el KNN.

L'algorisme KNN:

L'algorisme de KNN, o K Nearest Neighbors, és un algorisme de classificació i regressió supervisada que utilitza aprenentatge automàtic. La idea fonamental d'aquest algorisme és classificar un punt de dades basant-se en els veïns més propers d'aquest punt. En el nostre cas ens permetrà cercar els usuaris més semblants a un usuari donades les seves valoracions.

Una vegada trobats els veïns més propers, l'algorisme genera recomanacions per a l'usuari. Les pel·lícules que els veïns han valorat positivament i que l'usuari encara no ha vist són candidates a ser recomanades.

Procediment del càlcul de recomanacions:

Carregar les Dades:
Les dades sobre les valoracions dels usuaris es carreguen des de la base de dades MongoDB i s'organitzen en una matriu de valoracions.

Configurar el Model KNN:
Es crea un model KNN utilitzant la llibreria scikit-learn. En el nostre cas, s'ha configurat amb 5 veïns, mesurant la similitud amb la distància cosinus i utilitzant l'algoritme de força bruta (*brute-force*).

Càlcul dels Veïns Més Propers:
Per a cada usuari, es calculen els veïns més propers basant-se en les seves valoracions anteriors.

Generació de Recomanacions:
A partir dels veïns més semblants, es calcula el recommendationScore de cada pel·lícula que l'usuari encara no ha valorat. Només aquelles amb un score superior a 2,5/5 (un 50% de possibilitats) es consideren com a recomanacions.

Guardar les Recomanacions:
Per cada usuari desem les recomanacions calculades a la col·lecció "recommendations" a la base de dades de MongoDB.

8.3.2 Freqüència de Càlcul de Recomanacions

La freqüència de càlcul de les recomanacions, és a dir, la freqüència per la qual actualitzem les nostres recomanacions és important, ja que si la fem molt sovint això comporta una gran càrrega de treball per al servidor en casos on tinguem molts usuaris. Però si la fem amb molt poca freqüència pot resultar que les noves valoracions dels usuaris no es tinguin en compte en les actuals recomanacions.

Aquesta freqüència principalment depèn de diversos factors, com la dinàmica d'actualització de les dades i la quantitat de nous usuaris o valoracions al sistema.

D'aquesta manera s'ha decidit que actualitzarem les nostres recomanacions automàticament cada nit a les 2 de la matinada mitjançant un script automatitzat que té el servidor de Python. Aquesta freqüència d'actualització és un bon compromís entre poca càrrega de treball al servidor, i prou freqüent per a no estar desactualitzat.

8.3.3 Eines Utilitzades al servidor Python

En la implementació del servidor Python, s'han utilitzat diverses llibreries i frameworks per facilitar el desenvolupament de la funcionalitat requerida. Les principals eines són:

Flask:

Un framework web minimalista per a Python que permet construir ràpidament aplicacions web. Flask proporciona funcionalitats essencials per gestionar les rutes, les sol·licituds HTTP i les respostes del servidor.

Pymongo:

Una llibreria de Python per interactuar amb bases de dades MongoDB. S'ha utilitzat per connectar-se a la base de dades MongoDB que conté la informació sobre els usuaris, les pel·lícules i les valoracions.

Scikit-learn (sklearn):

Una llibreria de màquines vectorials de suport per a Python que inclou eines per a la mineria i l'anàlisi de dades. En aquest cas, s'ha utilitzat l'algorisme de veïns més propers (KNN) per a la recomanació.

9 Proves, Resultats i Millores

A continuació es mostra la llista de les valoracions que han donat els usuaris a certes pel·lícules

[Veure Apendix A]

Per a fer proves amb l'algorisme de recomanacions i confirmar el seu correcte funcionament podem provar d'eliminar algunes valoracions d'un usuari i veure si calcula correctament les recomanacions de les pel·lícules sense valorar. Per fer això eliminem les algunes de les valoracions de l'usuari 'Martina' que segons el KNN té una forta semblança amb l'usuari 'Maite'. S'eliminen: Interstellar, Ratatouille, Inglourious Basterds, 300, Avengers Infinity War, Hachi i Norbit.

El resultat de l'algorisme es veu reflectit a la col·lecció de recomanacions

[Veure Apendix B]

Veiem com les pel·lícules que millor score tenen son efectivament les esperades amb una puntuació alta (Interstellar, amb un 3.5 de puntuació i Hachi, amb un 3.25 de puntuació)

Per altra banda, les pel·lícules amb pitjor score, són les que pitjor serien valorades per l'usuari (Norbit, amb un 0.375 i Inglourious Basterds, amb un 1.5 de puntuació)

Tot i això, veiem que el model té els seus petits errors, per exemple, Avengers Infinity War hauria de tenir de les pitjors puntuacions, però es tracta d'un cas puntual on l'usuari Martina ha donat una puntuació excessivament baixa en tractar-se d'una pel·lícula de superherois.

Aquest és un dels casos on vindria bé tenir en compte la preferència de gèneres de l'usuari a l'hora de calcular les recomanacions.

Aquesta podria ser una de les millores a dur a terme sobre la nostra aplicació, és a dir, tenir en compte també el contingut i les característiques de les pel·lícules, com el gènere, l'any, l'idioma, els actors, etc. Aconseguint així un filtratge híbrid que integri diversos mètodes, combinant avantatges del filtratge col·laboratiu i de filtratge basat en contingut i així aconseguiríem millorar la precisió del model.

Pel que fa a la nostra pàgina web hem obtingut un sistema funcional amb una interfície senzilla i fàcil d'utilitzar tot i que encara podríem afegir més funcionalitats, com mostrar pel·lícules similars, mostrar tràilers, deixar comentaris, modificació de comptes, etc.

Per valorar l'experiència d'usuari s'ha provat el sistema amb familiars i amics i sembla que la majoria ha estat satisfet amb les recomanacions, tot i que sí que és cert que en un entorn real on hi hagi molts més usuaris, hi haurà més valoracions a tenir en compte, el qual millorarà l'exactitud de les recomanacions.

10 Conclusions

Per concloure el projecte cal recalcar que s'han dut a terme correctament tots els objectius del treball, seguint les metodologies esmentades amb un resultat final d'èxit. A continuació repassem els nostres objectius:

Objectiu Principal [Superat]:

Hem pogut arribar a crear una plataforma web enfocada a millorar l'experiència dels usuaris en la descoberta de noves pel·lícules mitjançant l'ús d'un sistema de recomanacions basat en intel·ligència artificial.

Objectiu 1: Recopilació de dades [Superat]

Hem aconseguit dades reals de diferents familiars i amics. S'ha dut a terme la recollida de les dades de les valoracions de pel·lícules i s'han vinculat amb les ID de pel·lícules de l'API de TMDB, per a posteriorment afegir les valoracions des de l'aplicació. S'ha utilitzat TMDB com a font de BD per a obtenir pel·lícules i les seves dades.

Objectiu 2: Desenvolupament del frontend [Superat]

S'ha dissenyat i implementat una interfície d'usuari per a la plataforma web que permet explorar i valorar les pel·lícules de manera intuïtiva i simple. L'ús de llibreries de react ens ha permès implementacions més senzilles, com la del sistema de valoració per estrelles.

Objectiu 3: Desenvolupament del backend [Superat]

Durant el procés de desenvolupament, s'han abordat diversos aspectes; S'ha dut a terme la creació de la part d'ambdós servidors, tant del RESTful API com del servidor Python amb l'algorisme de recomanacions i el KNN. S'ha implementat també l'API de The Movie Database en el nostre servidor. En tot moment s'ha buscat la màxima eficiència i escalabilitat així com la correcta interacció amb el client. A més, s'ha implementat funcionalitats com la gestió d'usuaris amb JWT entre altres.

Objectiu 4: Optimitzacions i desplegament [Parcialment Superat]:

Pel que fa a les optimitzacions, s'han dut a terme proves internes per assegurar la robustesa i la fiabilitat del sistema. No obstant això, per a una validació més exhaustiva, seria necessari dur a terme proves amb un grup més ampli d'usuaris. Malauradament, tot i que s'ha utilitzat Docker per a aïllar els servidors, no hem realitzat el desplegament en AWS, ja que no ha pogut ser completat dins del marc d'aquest treball.

◆

En conclusió, ha sigut un treball que m'ha introduït a noves tecnologies i mètodes d'implementació i que m'ha permès obrir-me a noves idees i projectes que semblaven més difícils i ara em semblen més accessibles.

11 Bibliografia

[1] "Angular vs React: A Detailed Side-by-Side Comparison," Kinsta®, Nov. 22, 2021. Available: https://kinsta.com/blog/angular-vs-react/

[2] E. Trossat, "Personalised recommendations for films and TV series: Webedia's Data Science team explains how they…," Medium, Sep. 10, 2021. Available: https://webedia.io/personalised-recommendations-for-films-and-tv-series-webedias-data-science-team-explains-how-they-62392537fb59.

[3] "Express routing," Expressjs.com, 2017. Available: https://expressjs.com/en/guide/routing.html

[4] "Difference between Virtual DOM and Real DOM," GeeksforGeeks, Sep. 05, 2022. Available: https://www.geeksforgeeks.org/difference-between-virtual-dom-and-real-dom/

[5] "Deploying Docker containers on ECS," Docker Documentation, Sep. 06, 2023. Available: https://docs.docker.com/cloud/ecs-integration/.

[6] "Introducing JSX – React," legacy.reactjs.org. Available: https://legacy.reactjs.org/docs/introducing-jsx.html

[7] "Wikiwand - Single-page application," Wikiwand. Available: https://www.wikiwand.com/en/Single-page_application.

[8] "Affinity Matrix," DeepAI, May 17, 2019. Available: https://deepai.org/machine-learning-glossary-and-terms/affinity-matrix.

[9] "Educative Answers - Trusted Answers to Developer Questions," Educative. Available: https://www.educative.io/answers/what-is-the-difference-between-virtual-and-real-dom-react.

[10] "Node JS vs PHP: Which Backend to Choose for Your Project in 2022?," Radixweb, Mar. 29, 2022. Available: https://radixweb.com/blog/node-js-vs-php

[11] RedHat, "What is a REST API?," www.redhat.com, May 08, 2020. Available: https://www.redhat.com/en/topics/api/what-is-a-rest-api

[12] N. M, "Express vs Flask: A Comparison of Speed," Medium, Oct. 28, 2021. Available: https://python.plainenglish.io/lets-find-which-is-fast-node-js-vs-flask-python-5ab68b3c3559.

[13] T. P. Team, "What Is a REST API? Examples, Uses, and Challenges," Postman Blog, Jul. 09, 2020. Available: https://blog.postman.com/rest-api-examples/

[14] "Welcome to Flask — Flask Documentation (3.0.x)," flask.palletsprojects.com. Available: https://flask.palletsprojects.com/en/3.0.x/

[15] "I rebuilt the same web API using Express, Flask, and ASP.NET. Here's what I found.," freeCodeCamp.org, Feb. 29, 2020. Available: https://www.freecodecamp.org/news/i-built-a-web-api-with-express-flask-aspnet/

[16] M. Kuroda, Y. Mori, and M. Iizuka, "Initial Value Selection for the Alternating Least Squares Algorithm," Studies in classification, data analysis, and knowledge organization, Jan. 2020, doi: https://doi.org/10.1007/978-981-15-3311-2_18

[17] "Wikiwand - Singular value decomposition," Wikiwand. Available: https://www.wikiwand.com/en/Singular_value_decomposition.

[18] MongoDB. "Getting Started," MongoDB Documentation. Available: https://www.mongodb.com/docs/manual/tutorial/getting-started

[19] Stack Overflow. "Best way to store password in database," Available: https://stackoverflow.com/questions/1054022/best-way-to-store-password-in-database

[20] Scrimba. "React Project Structure," Available: https://scrimba.com/articles/react-project-structure/

[21] MongoDB. "db.collection.find() Method," MongoDB Documentation. Available: https://www.mongodb.com/docs/manual/reference/method/db.collection.find/

[22] Stack Overflow. "Best way to connect to MongoDB using Node.js," Available: https://stackoverflow.com/questions/38485575/best-way-to-connect-to-mongodb-using-node-js

[23] MongoDB. "Connect to MongoDB with MongoDB Compass," MongoDB Documentation. Available: https://www.mongodb.com/docs/compass/current/connect/

[24] ImproNunciable/moviedb Repository on GitHub. Available: https://github.com/impronunciable/moviedb

[25] react-rating-stars-component on npm. Available: https://www.npmjs.com/package/react-rating-stars-component

[26] ""Fabrik Font, 1001fonts,. Available: https://www.1001fonts.com/fabrik-font.html

[27] Vaadata. "How to Securely Store Passwords in Database." Available: https://www.vaadata.com/blog/how-to-securely-store-passwords-in-database/

[28] The Movie Database (TMDb) API Documentation. "Getting Started," TMDb API. Available: https://developer.themoviedb.org/reference/intro/getting-started

[29] "Angular 12 Refresh Token," Bezkoder. Available: https://www.bezkoder.com/angular-12-refresh-token/

[30] Aman Makwana, "Understanding Recommendation System and KNN with Project: Book Recommendation System," Medium, Available: https://aman-makwana101932.medium.com/understanding-recommendation-system-and-knn-with-project-book-recommendation-system-c648e47ff4f6

[31] "Using JWT for Authentication in React," OpenReplay Blog. Available: https://blog.openreplay.com/using-jwt-for-authentication-in-react

12 Apèndix

APÈNDIX A

la llista de les valoracions que han donat els usuaris a certes pel·lícules

A	B	C	D	E	F	G	H	I	J
ID	TITLE	Ferran	Martina	Maite	Nando	Ivan	Eloi	Eric	Jack
105	Back To The Future	5	3	3.5	3	4	4.5	4	4
329	Jurassic Park	4	2.5	3	2.5	3	3.5	5	3.5
13	Forrest Gump	5	2.5	5	5	4	3	4.5	4
680	Pulp Fiction	5	2	5	5	3.5	3	4.5	3.5
313369	La La Land	4.5	4	5	3.5	2.5	2	3	4
157336	Interstellar	4.5	5	5	4	5	4	5	4
346698	Barbie	3.5	4	3.5	2	1.5	0.5	2.5	4
335797	Sing!	2.5	4	2.5	2	2.5	3.5	3	3.5
2062	Ratatouille	4.5	5	4	4.5	3.5	4	4.5	4
211672	Minions	2	3.5	3	3.5	2.5	3.5	3	4
164	Breakfast At Tiffany's	4	1.5	5	3.5	2.5	2.5	2.5	3
744	Top Gun	4	2.5	5	5	3	3.5	4	2.5
11631	Mamma Mia!	4	4	5	2.5	1.5	1.5	1.5	3
857	Saving Private Ryan	1.5	2	4.5	5	3.5	4	4.5	2.5
77338	The Intouchables	3	3	4.5	4	4.5	3	3	3.5
16869	Inglourious Basterds	4	1.5	2	4	5	4.5	4.5	3
244786	Whisplash	5	3	2.5	4	3.5	3.5	4.5	3.5
1271	300	1	0.5	2.5	5	4.5	3.5	3.5	1
424	Schindler's List	3.5	3	4.5	4.5	4	3	3.5	3
807	Seven	3.5	4	4.5	4	4.5	3.5	5	2.5
10193	Toy Story 3	4	4.5	3	3.5	5	5	3.5	3.5
299534	Avengers Endgame	0.5	1.5	2.5	1.5	5	5	4	5
424694	Bohemian Rhapsody	4	2	3.5	4.5	5	2.5	2.5	2.5
496243	Parasite	3.5	2.5	3.5	3.5	4.5	3.5	5	3.5
8587	The Lion King	4	3.5	3.5	3.5	4	4.5	3.5	3.5
299536	Avengers Infinity War	2	0.5	2.5	1.5	5	5	4.5	5
557	Spiderman	3.5	2	3	3	5	4.5	3.5	4
1726	Iron Man	1	1	2.5	0.5	4	5	3.5	4
458156	John Wick 3	2.5	2.5	3.5	3	4.5	5	3.5	2
475557	Joker	4	2	2.5	3.5	4	4	4.5	3.5
504608	Rocketman	4	3	4	3.5	3.5	2.5	3	3.5
10681	Wall-E	4.5	3.5	3.5	3.5	4	5	3.5	4
28178	Hachi: A Dog's Tale	3.5	4.5	4	3.5	4	3	4.5	4
9757	Norbit	0.5	0.5	0	0	2.5	3.5	3.5	4

APÈNDIX B

El resultat de l'algorisme reflectit a la col·lecció de recomanacions

```
_id: ObjectId('65c203e7a3e03ed77f48f78b')
userID: "martina"
movieId: 157336
recommendationScore: 3.5

_id: ObjectId('65c203e7a3e03ed77f48f78c')
userID: "martina"
movieId: 28178
recommendationScore: 3.25

_id: ObjectId('65c28c11d6a6d39917fe31a6')
userID: "martina"
movieId: 299536
recommendationScore: 2.875

_id: ObjectId('65c28c11d6a6d39917fe31a7')
userID: "martina"
movieId: 2062
recommendationScore: 2.5

_id: ObjectId('65c28d632265bc8daad30682')
userID: "martina"
movieId: 1271
recommendationScore: 2

_id: ObjectId('65c28d632265bc8daad30683')
userID: "martina"
movieId: 16869
recommendationScore: 2

_id: ObjectId('65c28d632265bc8daad30684')
userID: "martina"
movieId: 9757
recommendationScore: 0.375
```