Aplicación Web de Selección de Películas Implementando Algoritmos de aprendizaje.

Alexis Gabriel Diaz Fajardo

Resumen– El constante cambio en el ámbito tecnológico ha llevado a una expansión sin precedentes de aplicaciones de entretenimiento, generando un desafío al seleccionar contenido en un panorama saturado. Este artículo representa un proyecto destinado a desarrollar una aplicación web que tiene como objetivo simplificar la búsqueda de películas, enfocándose en las preferencias del usuario. A lo largo del artículo, se exploran los objetivos y funciones clave de la aplicación, detallando la metodología y tecnologías seleccionadas, con especial énfasis en la optimización del código para mejorar el tiempo de respuesta. Se presentan los resultados obtenidos hasta el momento y se extraen conclusiones. Para la implementación de recomendaciones personalizadas, se consideran bibliotecas como TensorFlow.js, junto con el uso de la matriz usuario-elemento para el análisis de recomendaciones, empleando algoritmos como la descomposición de valores singulares (SVD), similitud del coseno y Autoencoders. La arquitectura se despliega en un entorno de servidor Node.js, utilizando tecnologías como Express.js para manejar las solicitudes HTTP, además se aprovechan bases de datos NoSQL para almacenar, gestionar la información y se incluye una API para facilitar el acceso externo a la funcionalidad de la aplicación.

Palabras claves– Aplicación Web, Sistemas de Recomendación, TensorFlow, Descomposición de valores singulares(SVD), similitud del coseno, Autoencoders, nodeJS, Express.js, HTTP, Base de datos NoSQL, API

Abstract– The constant change in the technological landscape has led to an unprecedented expansion of entertainment applications, generating a challenge when selecting content in a saturated landscape. This article represents a project aimed at developing a web application that aims to simplify the search for movies, focusing on user preferences. Throughout the article, the key objectives and functions of the application are explored, detailing the methodology and selected technologies, with a special emphasis on optimizing the code to improve response time. The results obtained so far are presented, and conclusions are drawn. For the implementation of personalized recommendations, libraries such as TensorFlow.js are considered, along with the use of the user-item matrix for recommendation analysis, employing algorithms such as Singular Value Decomposition (SVD), Cosine similarity and Autoencoders. The architecture is deployed in a Node.js server environment, utilizing technologies such as Express.js to handle HTTP requests, and taking advantage of NoSQL databases to store, manage information, and includes an API to facilitate external access to the application's functionality.
Keywords– Web Application, Recommendation Systems, TensorFlow, Singular Value Decomposition (SVD), Cosine similarity, Autoencoders, Node.js, Express.js, HTTP, NoSQL Database, API.

1 Introducción

El mundo continúa cambiando y más en el ámbito tecnológico, donde el número de aplicaciones orientadas al contenido de entretenimiento es más amplio que nunca y tomar una decisión sobre que película ver puede generar una sensación de estrés y más cuando disponemos de tiempo limitado. Con la multiplicación de plataformas de stream y un repertorio en constante crecimiento de

• E-mail de contacto: d.alexis.g@hotmail.com
• Mención realizada: Tecnologías de la Información
• Trabajo Tutorizado por :Juan Antonio Martinez Carrascal (Área de Ciencias de la Computación e Inteligencia Artificial)
• Curs 2023/24

películas y programas de televisión[1], las personas buscan a menudo películas que coincidan con sus gustos e intereses personales. En este contexto se evidencia la necesidad de una solución que simplifique y mejore la experiencia de búsqueda de películas.

Ante este escenario, surge la necesidad imperante de una solución que simplifique y mejore la experiencia de búsqueda de películas. En este articulo, se presenta el un proyecto que tiene como objetivo desarrollar una aplicación web centrada en la selección de películas de acuerdo con las preferencias individuales del usuario. Esta aplicación no solo facilitará la búsqueda según criterios específicos, como género, título y valoraciones de otros usuarios, sino que también proporcionará recomendaciones personalizadas mediante la implementación de algoritmos de aprendizaje automático. Los usuarios podrán crear listas de preferencias, calificar películas y, a través de la recopilación de datos, recibir sugerencias que se adapten a sus gustos personales.

La base de trabajo de este proyecto es el conjunto de datos de "The Movies DataSet" disponible en la plataforma Kaggle [2], proporcionando una extensa variedad de películas que servirá como fundamento el desarrollo de la aplicación. Además, para garantizar una cobertura amplia en la búsqueda de imágenes, se emplearon diversas APIs, incluyendo The Movie Database (TMDb) y FilmToro.

A lo largo de este artículo, se explorarán los objetivos y funciones clave de la aplicación. Se discutirá la metodología y las tecnologías seleccionadas para su desarrollo, proporcionando una justificación detallada de su implementación. Se hablara también de la arquitectura de despliegue, la optimización del código para mejorar el Tiempo de respuesta. Además, se presentarán los resultados obtenidos y se extraerán conclusiones hasta ahora.

2 Objetivos

El objetivo de este proyecto es desarrollar una aplicación web en que mejore significativamente la experiencia de búsqueda de películas, proporcionando a los usuarios recomendaciones personalizadas basadas en sus preferencias individuales. A través de la implementación de algoritmos de aprendizaje automático, la aplicación permitirá a los usuarios filtrar y explorar el extenso catálogo de películas, brindándoles opciones alineadas con sus gustos cinematográficos. Además, se busca optimizar el rendimiento de la aplicación mediante la adopción de prácticas de codificación eficientes y la mejora del tiempo de respuesta. Este proyecto busca ofrecer una solución centrada en el usuario para simplificar la elección de películas.

3 Estado del arte

Como mencionamos anteriormente, existen múltiples plataformas de selección de películas, a continuación se describen las mas populares. [3]

PlayPilot Es una plataforma que permite a los usuarios buscar y descubirr contenido de streaming en diferentes servicios como Netflix, HBO, Amazon Prime Videos, entre otros. Ofrece una interfaz facil de usar y una amplia variedad de opciones de busqueda y filtrado.[4]

Betaseries Es una plataforma que se enfoca en series de televisión y permite a los usuarios hacer un seguimiento de los episodios que han y no han visto. También ofrece recomendaciones personalizadas y una comunidad de usuarios para discutir y compartir opiniones sobre la serie. [5]

IMDb Es una plataforma que ofrece información sobre películas, series de televisión, actores y directores. También ofrece calificaciones y reseñas de usuarios como recomendaciones personalizadas. [6]

Hulu Es una plataforma de streaming que ofrece una amplia variedad de contenido, incluyendo películas, series de televisión y programación en vivo. También ofrece opciones de personalización y recomendaciones basadas en el historial de visualización del usuario. [7]

Filmaffinity Es una plataforma que ofrece información sobre películas, incluyendo calificaciones y reseñas de usuarios. También cuenta con una sección de lo mas visto, lo mejor y lo peor votado en el momento, lo que mantiene a los usuarios actualizados sobre las tendencias y popularidad del contenido real. [8]

MovieFone Es una plataforma que ofrece información sobre películas, incluyendo horarios de proyección en cines cercanos. También ofrece calificaciones y reseñas de usuarios, así como recomendaciones personalizadas. [9]

Las plataformas ofrecen una amplia variedad de opciones para buscar y descubrir contenido de streaming, así como recomendaciones personalizadas y comunidades de usuarios para discutir y compartir opiniones sobre las películas y series de televisión, además, muchas de estas plataformas utilizan sistemas de recomendación basados en algoritmos de aprendizaje automático para ofrecer recomendaciones mas precisas y personalizadas a los usuarios. [10]

A pesar de la existencia de numerosas plataformas, existe una necesidad continua para simplificar y enriquecer la experiencia de búsqueda de películas. Con esta premisa como base, se presenta un proyecto que se centra en el desarrollo de una aplicación web de selección de películas. Este proyecto se distingue en competir de las plataformas existentes al enfocarse en proporcionar una experiencia de usuario mas completa y personalizada. Para lograrlo, se pone especial énfasis en aspectos como interfaz de usuario atractiva, seguridad, eficiencia y tiempos de respuesta mas rápido, empleando tecnologías modernas y enfoques innovadores.

4 Tecnologías

Hemos realizado cuidadosas elecciones de tecnologías y frameworks que se complementan entre sí para el desarrollo de la aplicación. Cada elección se basa en su capacidad para cumplir con los objetivos del proyecto y garantizar la eficiencia y calidad de la aplicación.

4.1. Lenguaje de programación

Para el desarrollo del backend de la App, hemos optado por utilizar Node.js [11], esta lenguaje es ampliamente conocido por su eficiencia y escalabilidad, en términos de manejo de solicitudes y conexiones, lo que lo hace adecuado para aplicaciones que requieren alta eficiencia en la gestión de solicitudes web y conexiones a tiempo a real. A pesar de no estar diseñado para manejar grandes volúmenes de datos por sí solo, lo complementamos con bibliotecas y herramientas adecuadas como Express.js y MongoDB.

Para la implementación de la funcionalidad de recomendación personalizada, se consideraran bibliotecas como TensorFlow.js [12] o Brain.js [13] que proporcionan herramientas y modelos pre-entrenados. Además para el análisis de recomendaciones, se utilizara ”la matriz usuario-elemento”, es una representación que muestra la interacción entre usuarios y elementos en donde la matriz se utiliza para registrar si un usuario ha interactuado o no con una película [14]. Existen varios algoritmos populares para factorizar una matriz, como el algoritmo de descomposición de valores singulares (SVD) [15], la similitud del coseno[16] y así como los autoencoders[17] para la reducción de dimensionalidad en caso de utilizar redes neuronales [18]. Las implementaciones de estos algoritmos la encontramos en varias bibliotecas de Python [19].

Es importante destacar que, al desarrollar la API en Node.js, se establecerá una comunicación fluida y eficiente con Python. Esto permitirá la transferencia de información de manera efectiva entre ambas plataformas, lo que resulta esencial para el funcionamiento de los algoritmos de recomendación y el procesamiento de datos.

Utilizaremos también la biblioteca de validación Zod [20], para definir esquemas de validación y validar datos de entrada. Esto será útil para prevenir errores y garantizar que los datos sean correctos antes de ser enviados a la aplicación, mejorando la integridad de nuestros datos y la seguridad de la aplicación.

4.2. Framework’s

Express.js
Será utilizado como el framework de desarrollo web para el backend. Es conocido por su simplicidad y velocidad, lo que nos permitirá crear de manera eficiente endpoints y controladores para gestionar las solicitudes HTTP de los usuarios. [21]

Tailwind CSS
Es un marco de diseño CSS altamente configurable que facilita la creación de interfaces de usuario, tiene como principal objetivo reducir la cantidad de código para estilizar la aplicación. Su enfoque nos permitirá simplificar y reutilizar interfaces estiladas de manera rápida y eficiente, lo que será esencial para lograr una interfaz de usuario atractiva. [22]

Vue.js

Para la construcción de interfaces de usuario hemos optado por utilizar Vue.js, un framework Javascript que facilita la creación de componentes. Con Vue.js podemos garantizar una experiencia de usuario fluida y altamente receptiva al crear componentes modulares y reutilizables en nuestra aplicación. [23]

4.3. Hospedaje

En cuanto el alojamiento de nuestra aplicación, hemos seleccionados plataformas que se ajustan a nuestras necesidades y con lo que estamos familiarizados.

1. Backend y Front-end en FL0.
2. Base de Datos en MongoDB Atlas (Cloud).

4.4. Base de Datos

MongoDB
Para almacenar y gestionar la amplia variedad de datos de películas, utilizaremos MongoDB, una base de datos NoSQL. Su capacidad de escalabilidad horizontal y su flexibilidad para trabajar con altos volúmenes de datos. Tomamos en cuenta que la velocidad en la recuperación y manipulación de datos es esencial para una experiencia de usuario fluida [24].

4.5. Otras Tecnologías

Además de las tecnologías principales mencionadas anteriormente, hemos seleccionado herramientas adicionales que desempeñaran un papel fundamental en el éxito de nuestro proyecto.

Control de versión
GIT será nuestra elección para el control de versiones y la gestión del código fuente. Además de la familiaridad con esta plataforma, nos permitirá mantener un historial detallado de cambios de nuestro código. [25]

Estas opciones de alojamiento aseguraran que nuestra aplicación esté disponible en línea de manera confiable y que pueda manejar las demandas de nuestros usuarios. Además, la familiaridad con estas plataformas nos permitirá optimizar su configuración y rendimiento de manera efectiva.

5 Desarrollo

5.1. Migración de la Base de Datos.

El proceso de migración de la base de datos inicial, que se encontraba en formato CSV, se completó con éxito. Los datos se han trasladado a MongoDB, lo que ha permitido un almacenamiento mas estructurado y eficiente, proporcionando la base para el desarrollo del proyecto.

5.2. Instalación y Configuración

El proyecto se ha instalado y configurado localmente en Node.js. Las dependencias se han gestionado de manera eficiente para permitir un entorno de desarrollo ágil y sencillo.

La aplicación ha sido respaldado en un repositorio GitHub A.1, lo que asegura una capa adicional de seguridad y control de versiones. Este respaldo permitirá restaurar y seguir el desarrollo del proyecto en caso de cualquier interrupción o bug de la aplicación. Este se ha compartido con el tutor, garantizando que se mantenga al tanto del progreso y las actualizaciones del proyecto.

Hasta la fecha se ha logrado implementar un servidor básico con rutas iniciales para recuperar información de las películas. Además, se ha creado un entorno de pruebas para validar la funcionalidad y operaciones básicas del sistema.

5.3. Estructura de la aplicación.

Se ha implementado una estructura especifica para facilitar su desarrollo y despliegue Modelo-Vista-Controlador (MVC)[26]. Se han definido varios elementos fundamentales:

Controladores: se han añadido controladores para gestionar las operaciones relacionadas con las funcionalidades de la aplicación.

Modelo: Gestiona la conexión a la base de datos MongoDB y las operaciones relacionadas con la colección de nuestra Base datos.

Routes: Maneja las rutas y peticiones asociadas a la aplicación.

Middleware: Se definen los intermediarios entre el servidor y el cliente, se ha implementado Cors por posibles inconvenientes y la implementación de sesiones.

Schema: contiene esquemas de validación (por ejemplo, movie.js y user.js) para asegurar la integridad de datos en la aplicación, empleando reglas específicas para campos como título de película, ID, nombre de usuario, correo electrónico y contraseña. Estos esquemas son fundamentales para garantizar la robustez y seguridad del sistema.

App.js Donde se configura el servidor, middleware [27], y configuraciones del servidor.

Test-Endpoints.http: Se encuentran los pruebas realizadas de las solicitudes HTTP de las rutas de la API que se han implementado, facilitando la prueba y verificación de los endpoints.

Script generateImagenes.js: Este script fue creado para almacenar las rutas de las imágenes de cada película, aunque no abarca la totalidad, contribuye a disminuir el tiempo de carga en la aplicación. Al ejecutarlo, podemos mantener actualizadas las imágenes en nuestra base de datos.

Durante la ejecución del proyecto, se utilizó una metodología de desarrollo ágil para gestionar y visualizar las tareas diarias, se implementó una tabla Kanban que proporcionó una herramienta efectiva para la gestión de tareas y la colaboración en el desarrollo del proyecto.

La tabla Kanban utilizada a lo largo del desarrollo. Proporcionando una visión visual adicional del progreso y las actividades del equipo a lo largo del proyecto. A.4

5.4. Despliegue e implementación de Funcionalidades

Se logró la implementación exitosa del despliegue de la base de datos en Atlas MongoDB, proporcionando una solución robusta y escalable para la gestión de datos. Atlas MongoDB ofrece características como escalabilidad automática, seguridad avanzada y monitoreo en tiempo real.

Simultáneamente, el código backend fue desplegado en FL0, una plataforma que ha demostrado ser eficiente para el hosting de aplicaciones. Aunque el sistema está operativo en línea, se identificó un comportamiento que vale la pena mencionar. Si no se recibe ninguna petición durante un período específico, la página puede cerrarse momentáneamente debido a las políticas de suspensión automática de FL0. Esto es una medida normal para optimizar recursos, y la aplicación se reactiva automáticamente ante nuevas solicitudes.

Estos despliegues son esenciales para garantizar la disponibilidad constante de la aplicación y la gestión eficiente de la base de datos en un entorno de producción.

La aplicación sigue una estructura modular, haciendo uso del framework Vue.js para la construcción de la interfaz de usuario y Tailwind CSS para la gestión de estilos y diseño. La organización del proyecto cumple con las mejores practicas de desarrollo web, facilitando la administración y comprensión del código. La estructura implementada es:

public: Carpeta donde se encuentra el código front-end, taildwindcss implementado y los componentes de Vue.js

Formularios: Ficheros login y registrar.html que carga los formularios para registrarse y loguearse en la aplicación.

src: Carpeta recursos donde están las configuraciones de postCSS [28] con Tailwind CSS, para poder implementar el estilado en los componentes de Vue.js

componentes.js Archivo Vue.js, que contiene los componentes para construir la aplicación web de películas , los componentes a destacar son

- **RootComponent:** En este componente, se definen las variables principales y se establece el funcionamiento fundamental de la aplicación. Aquí, nos comunicamos con nuestro servidor mediante peticiones HTTP para obtener todos los datos de las películas. Además, gestionamos la paginación a través del scroll para reducir y mejorar el tiempo de respuesta. Implementamos también el manejo de sesiones para usuarios que han iniciado sesión, y realizamos la llamada a nuestros componentes principales **FilterMovie**, **MovieItem**.

- **MovieItem** Es la sección donde se muestra las películas, **FilterMovies** Emite los eventos de búsquedas, renderizando y

mostrando las películas a buscar, tiene como componente hijo **DescriptionMovie**.

- **DescriptionMovie** Muestra los detalles e información general de la películas y las plataformas a tiempo real en donde se encuentran disponibles por medio de su componente hijo widgetJustWatch.

- **FilterMovies:** Este componente gestiona las interacciones del usuario, incluyendo la búsqueda y selección de películas, permitiendo una filtración precisa de las películas según las categorías utilizadas.
- Visualización de Plataformas por cada Película: Se implementó un componente **WidgetJustWatch**, el cual integra la funcionalidad de JustWatch, una plataforma que ofrece información detallada sobre la disponibilidad de la película en diversas plataformas de transmisión. Dado que no poseíamos información en nuestra base de datos, se optó por usar esta funcionalidad que nos proporciona en tiempo real las plataformas en las que está disponible A.3.

Durante la implementación del componente 'WidgetJustWatch' y el script de las imágenes , en el caso de las imágenes se optó por utilizar por predeterminado 404 not found, se encontraron algunos desafíos notables que afectaron la precisión de la visualización de plataformas para ciertos títulos. Algunos de estos desafíos incluyen:

Títulos en Inglés y Español: La presencia de títulos en ambos idiomas puede resultar en búsquedas menos precisas, ya que la API de JustWatch podría no encontrar una coincidencia exacta para títulos que tienen variantes en diferentes idiomas.

No Encontrar Coincidencias Exactas: Algunos títulos pueden no tener coincidencias precisas en la base de datos de JustWatch, lo que resulta en la falta de información sobre la disponibilidad en las plataformas de transmisión.

A pesar de los desafíos mencionados, es fundamental resaltar que el componente WidgetJustWatch continúa ofreciendo información valiosa sobre la disponibilidad de películas en diversas plataformas, contribuyendo significativamente a la experiencia del usuario. Estamos trabajando activamente en la mejora y resolución de estos desafíos para perfeccionar aún más la aplicación y optimizar la experiencia de los usuarios, ya que consideramos que esta área es crucial para el potencial de nuestra aplicación.

La implementación de un endpoint para buscar imágenes es un paso crucial. Aunque la base de datos proporciona rutas (paths) para las películas, no siempre garantiza la disponibilidad de los recursos. Con el fin de mejorar la experiencia del usuario y asegurar una visualización adecuada de las películas, hemos introducido un nuevo endpoint /image/:path. Esta solución se diseñó para abordar la problemática de disponibilidad de las rutas de imágenes proporcionadas por la base de datos. Decidimos implementar este endpoint a través de la interfaz de usuario, permitiendo que al crear el componente visual de nuestras películas, este componente busque automáticamente la imagen asociada en caso de que esté disponible. Esta decisión se tomó para proporcionar una solución ágil al usuario.

Este nuevo endpoint desempeña un papel crucial al buscar las imágenes de las películas no solo en la base de datos local, sino también a través de dos fuentes externas: la API de The Movie Database (TMDB) y el sitio 'filmtoro.cz'. La lógica detrás de esta implementación se basa en la consideración de que, aunque la base de datos proporciona rutas de imagen, la disponibilidad de los recursos no está garantizada.

El método 'getImage' del componente 'MovieItem' utiliza este nuevo endpoint para obtener la imagen de una película. Primero, intenta recuperar la imagen de TMDB a través de una solicitud a la API. En caso de que esta solicitud no sea exitosa, la aplicación realiza un segundo intento, buscando la imagen en 'filmtoro.cz'. Este enfoque dinámico mejora significativamente la cobertura de obtención de imágenes, solucionando la falta de recursos para muchas películas.

Es importante mencionar que, aunque esta implementación aborda la mayoría de las películas, no puede garantizar la disponibilidad total de recursos para todas las películas debido a la variabilidad de las fuentes externas. Se ha incorporado un manejo de errores robusto para adaptarse a posibles fallos en ambas solicitudes y garantizar una presentación coherente en la interfaz de usuario. Actualmente, lo mantenemos **desactivado** en la fase de desarrollo, ya que hemos priorizado el enfoque en funcionalidades y la optimización del tiempo de respuesta.

6 Algoritmos de recomendación

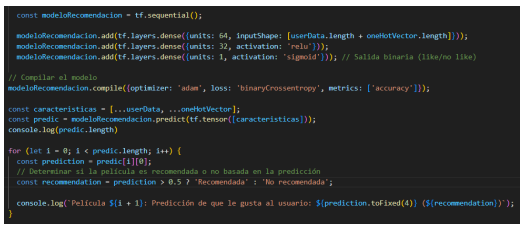
Durante el desarrollo de este proyecto, se llevó a cabo una exploración y prueba de algoritmos utilizando TensorFlow y la Descomposición de Valores Singulares (SVD) en el entorno de Node.js. Sin embargo, surgieron problemas de implementación que afectaron la efectividad de estos enfoques. Por consiguiente, se optó por utilizar la similitud de coseno [33] como alternativa. En este artículo, exploraremos los desafíos encontrados durante el intento inicial de utilizar TensorFlow y SVD, así como los motivos que condujeron a la elección de la similitud de coseno como solución alternativa.

6.1. TensorFlow

Utilizando TensorFlow en el entorno de Node.js, se procedió a construir un modelo de recomendación basado en los likes del usuario almacenados en nuestra BD. Se empleó un modelo secuencial con capas densas para procesar las características de entrada y predecir la probabilidad de que a un usuario le guste una película determinada.

Se crearon modelos de entrada para representar los datos del usuario y las películas. Para el usuario, se construyó un vector de características que representaba todas las películas disponibles, con un valor de 1 en las posiciones correspondientes a las películas que le gustaban al usuario y 0 en las demás. Esto se logró mediante la creación de un vector llamado userData, con una longitud igual al número total de películas disponibles, inicializándolo con ceros. Luego, para cada película que le gustaba al usuario, se actualizaba el valor correspondiente en userData a 1.

Para construir las características de entrada que representan los géneros de las películas que le gustan al usuario, se utilizó una técnica conocida como codificación one-hot. Esto implica representar cada género de película como un vector binario, donde un elemento tiene un valor de 1 y los demás tienen un valor de 0.



```
const modeloRecomendacion = tf.sequential();

modeloRecomendacion.add(tf.layers.dense({units: 64, inputShape: [userData.length + oneHotVector.length]}));
modeloRecomendacion.add(tf.layers.dense({units: 32, activation: 'relu'}));
modeloRecomendacion.add(tf.layers.dense({units: 1, activation: 'sigmoid'})); // Salida binaria (like/no like)

// Compilar el modelo
modeloRecomendacion.compile({optimizer: 'adam', loss: 'binaryCrossentropy', metrics: ['accuracy']});

const caracteristicas = [...userData, ...oneHotVector];
const predic = modeloRecomendacion.predict(tf.tensor([caracteristicas]));
console.log(predic.length)

for (let i = 0; i < predic.length; i++) {
  const prediction = predic[i][0];
  // Determinar si la película es recomendada o no basada en la predicción
  const recommendation = prediction > 0.5 ? 'Recomendada' : 'No recomendada';

  console.log(`Película ${i + 1}: Predicción de que le gusta al usuario: ${prediction.toFixed(4)} (${recommendation})`);
}
```

Fig. 1: Modelo de aprendizaje en base a los likes del cliente

A continuación, se agregaron varias capas densas al modelo utilizando el método add() del objeto modeloRecomendacion. La primera capa densa, con 64 unidades, fue definida con un parámetro inputShape que especificaba la forma de los datos de entrada. Esta forma incluía la longitud de dos vectores de características: userData y oneHotVector, que representaban las preferencias del usuario y los géneros de las películas, respectivamente.

Luego, se añadió una segunda capa densa con 32 unidades y una función de activación ReLU, seguida de una capa densa de una sola unidad con función de activación sigmoid. Esta última capa producía una salida binaria que indicaba si al usuario le gustaría o no la película.

Finalmente, se compiló el modelo utilizando el método compile(), donde se especificó el optimizador ('adam'), la función de pérdida ('binaryCrossentropy') y las métricas a evaluar (['accuracy']).

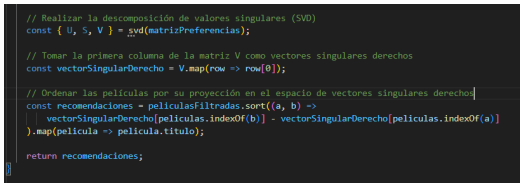
A pesar de los esfuerzos realizados para resolver estos problemas, se encontró que la complejidad de la implementación y la falta de experiencia en el manejo de TensorFlow en el entorno de Node.js dificultaban la resolución de los problemas identificados.

6.2. Descomposición de Valores Singulares

La descomposición de valores singulares (SVD) es una técnica fundamental en el campo del análisis de datos y el aprendizaje automático. Se utiliza comúnmente en una variedad de aplicaciones, incluidos los sistemas de recomendación, el procesamiento de imágenes, la reducción de la dimensionalidad y la resolución de problemas de mínimos cuadrados. Se empleó para modelar las preferencias de los usuarios (likes) y las características de las películas en un espacio dimensional reducido, con el objetivo de generar recomendaciones personalizadas de películas.

La SVD. permite descomponer una matriz de datos en tres matrices , U, S, y V, donde U y V contienen vectores singulares izquierdos y derechos respectivamente y S la diagonal de los valores singulares.

Para preparar los datos, usamos todas las películas, las películas de preferencia de nuestro usuario y filtramos los géneros únicos de su agrado. Una vez que limpiamos y preparamos nuestros, creamos nuestra Matriz de preferencia e implementamos el modelo.



```
// Realizar la descomposición de valores singulares (SVD)
const { U, S, V } = svd(matrizPreferencias);

// Tomar la primera columna de la matriz V como vectores singulares derechos
const vectorSingularDerecho = V.map(row => row[0]);

// Ordenar las películas por su proyección en el espacio de vectores singulares derechos
const recomendaciones = peliculasFiltradas.sort((a, b) =>
  vectorSingularDerecho[peliculas.indexOf(b)] - vectorSingularDerecho[peliculas.indexOf(a)]
).map(pelicula => pelicula.titulo);

return recomendaciones;
}
```

Fig. 2: Modelo SVD en base a los likes del cliente

A pesar de la implementación del modelo utilizando SVD, los resultados obtenidos no fueron coherentes y no reflejaron las preferencias del usuario de manera precisa. Se identificaron varios problemas durante la implementación del modelo, que podrían haber contribuido a los resultados inconsistentes. Estos problemas incluyeron la presencia de datos NaN en la matriz de datos utilizada para la SVD, posibles errores en el preprocesamiento de datos y dificultades en la interpretación de los resultados de la descomposición SVD.

6.3. Similitud de Coseno

Durante el desarrollo de este proyecto de investigación, se exploraron diversas técnicas y enfoques para la implementación de un sistema de recomendación eficaz. Uno de los enfoques consi-

derados fue el uso de la similitud de coseno como una medida para comparar la similitud entre las preferencias del usuario y las características de las películas.

La similitud de coseno es una técnica ampliamente utilizada en el campo del aprendizaje automático y la minería de datos para calcular la similitud entre vectores en un espacio multidimensional. En el contexto de un sistema de recomendación de películas, este enfoque implica representar tanto las preferencias del usuario como las características de las películas como vectores de características y luego calcular la similitud de coseno entre ellos.

$$similarity = cos(\theta) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}$$

Fig. 3: Formula similitud del coseno

Para la implementación de este algoritmo, se utilizó un conjunto de datos que incluye las características de todas las películas. Estas características se basaron en los gustos del usuario tomando en cuenta el genero de la película. Estas características se almacenaron en un archivo CSV, proporcionado por Kaggle. Se utilizo python para procesar estos datos y calcular la similitud del coseno entre las películas.

```
import ast
import pandas as pd    La importación "pandas" no se ha podido resolver desde el origen
from rake_nltk import Rake    No se ha podido resolver la importación "rake_nltk".
import numpy as np    No se ha podido resolver la importación "numpy".
from sklearn.metrics.pairwise import cosine_similarity    La importación "sklearn.metrics.pairwise" no se ha podido res
from sklearn.feature_extraction.text import CountVectorizer    La importación "sklearn.feature_extraction.text" no se h


#generos = sys.argv[1]

#generos = "adventure fantasy family comedy drama romance action horror thriller crime"

df = pd.read_csv('data/movies_metadataAlta.csv')
df['Key_words'] = ""
df['genres'] = df['genres'].apply(ast.literal_eval)

def extract_and_lower_genres(genres_list):
    return [genre['name'] for genre in genres_list]


df['Key_words'] = df['genres'].apply(extract_and_lower_genres)


df.to_csv('data/csvLimpio_dataframe.csv', index=False)
```

Fig. 4: Preparación de Datos

Debido a la gran cantidad de datos y el tiempo necesario par calcular la similitud se desarrolló un script que se ejecuta periódicamente para actualizar la tabla de similitud entre las 45,000 películas. Para reducir el tiempo de procesamiento y el tamaño de los datos, la matriz de similitud se almacenó localmente en un archivo npz (formato numpy), lo que permitió reducir el tamaño de los datos de 16 GB a 900 MB.

La comunicación entre el servidor Node.js y el script de Python se logró mediante solicitudes HTTP. Cuando se solicitan recomendaciones, el servidor Node.js realiza una solicitud HTTP y ejecuta el script de Python, que organiza y ordena la información para calcular las recomendaciones basadas en la similitud del coseno. Una vez completado, el script devuelve los IDs de las películas recomendadas al servidor Node.js para su presentación al usuario.

```
import pandas as pd    La importación "pandas" no se ha podido resolver desde el origen
from rake_nltk import Rake    No se ha podido resolver la importación "rake_nltk".
import numpy as np    No se ha podido resolver la importación "numpy".
from sklearn.metrics.pairwise import cosine_similarity    La importación "sklearn.metrics.pairwise" no se ha podido res
from sklearn.feature_extraction.text import CountVectorizer    La importación "sklearn.feature_extraction.text" no se h
import h5py    No se ha podido resolver la importación "h5py".

df = pd.read_csv('data/csvLimpio_dataframe.csv')

print(df[['Key_words']].head(-1))
def convertir_a_cadena(lista):
    return "[" + " ".join(lista) + "]"

count = CountVectorizer()
count_matrix = count.fit_transform(df['Key_words'])
cosine_sim = cosine_similarity(count_matrix, count_matrix)

print("matriz generada")
np.savez_compressed('data/cosine_sim_compressedFinal.npz',cosine_sim)


print("guardado")
```

Fig. 5: Creación de Matriz de similitud

```
import pandas as pd    La importación "pandas" no se ha podido resolver desde el origen
from rake_nltk import Rake    No se ha podido resolver la importación "rake_nltk".
import numpy as np    No se ha podido resolver la importación "numpy".
from sklearn.metrics.pairwise import cosine_similarity    La importación "sklearn.metrics.pairwise" no se ha podido res
from sklearn.feature_extraction.text import CountVectorizer    La importación "sklearn.feature_extraction.text" no se h
import h5py    No se ha podido resolver la importación "h5py".

df = pd.read_csv('data/csvLimpio_dataframe.csv')

print(df[['Key_words']].head(-1))
def convertir_a_cadena(lista):
    return "[" + " ".join(lista) + "]"

count = CountVectorizer()
count_matrix = count.fit_transform(df['Key_words'])
cosine_sim = cosine_similarity(count_matrix, count_matrix)

print("matriz generada")
np.savez_compressed('data/cosine_sim_compressedFinal.npz',cosine_sim)


print("guardado")
```

Fig. 6: Llamada al script de python desde Node.js

Cuando un usuario solicita recomendaciones de películas, el servidor Node.js realiza una llamada al script de Python utilizando una solicitud HTTP. Esta llamada se realiza mediante una función que retorna una promesa, lo que permite al servidor Node.js esperar a que el script de Python termine de ejecutarse antes de procesar los datos.

Una vez que el script de Python ha terminado de ejecutarse y ha devuelto los resultados, el servidor Node.js procesa los datos recibidos. En este punto, se hace más fácil obtener los IDs de las películas recomendadas, para así enviarlos al cliente y este cree los componentes respectivos de cada película.

```
import pandas as pd    La importación "pandas" no se ha podido resolver desde el origen
from rake_nltk import Rake    No se ha podido resolver la importación "rake_nltk".
import numpy as np    No se ha podido resolver la importación "numpy".
from sklearn.metrics.pairwise import cosine_similarity    La importación "sklearn.metrics.pairwise" no se ha podido res
from sklearn.feature_extraction.text import CountVectorizer    La importación "sklearn.feature_extraction.text" no se h
import h5py    No se ha podido resolver la importación "h5py".

df = pd.read_csv('data/csvLimpio_dataframe.csv')

print(df[['Key_words']].head(-1))
def convertir_a_cadena(lista):
    return "[" + " ".join(lista) + "]"

count = CountVectorizer()
count_matrix = count.fit_transform(df['Key_words'])
cosine_sim = cosine_similarity(count_matrix, count_matrix)

print("matriz generada")
np.savez_compressed('data/cosine_sim_compressedFinal.npz',cosine_sim)


print("guardado")
```

Fig. 7: Script del recomendador de Películas en Python

Actualmente, esta funcionalidad está implementada únicamente en el backend del sistema, haciendo una petición http al servidor ‘https://aplicacion-movies-dev-pfra.1.ie-1.fl0.io/movies/recomendationMovies/emailUser‘ . Cuando un usuario realiza una solicitud de recomendaciones, el backend realiza una llamada correspondiente al script de Python y espera a que se completen los cálculos antes de enviar los resultados al cliente.

Este enfoque de implementación garantiza que los datos de recomendación sean precisos y estén completamente procesados antes de enviarlos al cliente. Sin embargo, en el futuro, se pueden ex-

plorar opciones para mejorar la eficiencia y la velocidad de la solicitud de recomendaciones, como el uso de tecnologías de procesamiento en tiempo real o el almacenamiento en caché de resultados previamente calculados.

Aunque inicialmente se consideró la implementación de un modelo de aprendizaje automático más complejo utilizando TensorFlow, surgieron desafíos técnicos que dificultaron su implementación. En lugar de persistir en la resolución de estos problemas, se optó por explorar alternativas más simples y eficientes como la descrita anteriormente.

7 Optimización de código

Con el objetivo de optimizar el tiempo de respuesta de la aplicación, se tomó la decisión de eliminar la solicitud HTTP para acceder a la información de todas las películas, que asciende a aproximadamente 45,000 registros. En su lugar, se implementó un script que se ejecuta periódicamente, para actualizar la información en la base de datos y generar un archivo JSON en el servidor. De esta manera, la aplicación envía directamente este archivo JSON, lo que ha demostrado mejorar de manera gradual el tiempo de respuesta. Ahora, la carga de un gran número de películas se realiza en mucho menor tiempo, proporcionando una experiencia más eficiente para los usuarios.

8 Conclusiones

Durante el desarrollo del proyecto, se han alcanzado los objetivos establecidos, demostrando la implementación exitosa de diversas funcionalidades. Se destaca la implementación de algoritmos de aprendizaje como TensorFlow y SVD. Aunque se experimentaron dificultades en la implementación completa y efectiva de estos algoritmos, se considera que esta experiencia brindó valiosos aprendizajes sobre su funcionamiento y aplicación en el contexto del proyecto. Además, se logró avanzar significativamente en la mejora y refinamiento de la interfaz de usuario de la aplicación, proporcionando así una experiencia más satisfactoria para los usuarios finales.

El proyecto ha permitido adquirir una comprensión más profunda de los algoritmos de recomendación y su aplicación práctica en el desarrollo de sistemas personalizados. Aunque hubo desafíos y obstáculos en el camino, el proceso de superarlos proporcionó valiosas lecciones que podrán aplicarse en proyectos futuros. Además, la capacidad de adaptarse y buscar soluciones alternativas demostró la versatilidad y la habilidad para enfrentar los desafíos en el desarrollo de software. En última instancia, el proyecto ha sido una experiencia enriquecedora que ha contribuido al crecimiento profesional y técnico.

Mirando hacia el futuro, las lecciones y habilidades adquiridas durante este proyecto nos servirán como base sólida para enfrentar desafíos similares en proyectos futuros. Nos hemos dado cuenta de la importancia de la adaptabilidad y la flexibilidad en el desarrollo de software, y estamos preparados para aplicar estos principios en situaciones donde la manipulación de grandes volúmenes de información sea necesaria. Además, reconocemos el potencial del campo de la inteligencia artificial y el aprendizaje automático, y estamos ansiosos por explorar y mejorar nuestras habilidades en este ámbito. Nuestro objetivo es aprovechar nuestros tropiezos en la realización de este proyecto como oportunidades de aprendizaje y crecimiento, impulsándonos hacia adelante con mayor determinación y confianza en nuestras capacidades.

9 Agradecimientos

Quisiera expresar mi profundo agradecimiento a mi tutor por brindarme las bases y la dirección esencial en la realización de este Trabajo de Fin de Grado. A lo largo de mi curso académico, agradezco a los maestros que han compartido sus conocimientos y lecciones, guiándome hasta este punto. Mi sincero agradecimiento también a todas las personas que han facilitado mi camino y, especialmente, a mi madre por su constante apoyo y voto de confianza. Finalmente, reconozco y agradezco a mí mismo por el esfuerzo perseverante, superando desafíos y alcanzando esta etapa crucial en mi trayectoria académica. ¡Gracias a todos por ser parte fundamental en mi vida académica.

Referencias

[1] Romanenko, Vasyl. (2020, mayo). *Live streaming apps: Design. Development. Support. softengi.com*. https://softengi.com/blog/video-streaming-apps-design-development-support/. Accedido: 2023-9. Idioma: español.

[2] (2023, octubre). *kaggle The Movies Dataset. kagle*. https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset. Accedido: 2023-10. Idioma: español.

[3] (2023, octubre). *SimilarWeb Competidores. SimilarWeb*.

[4] (2023, noviembre). *Playpilot*. https://www.playpilot.com/es/. Accedido: 2023-11. Idioma: español.

[5] (2023, noviembre). *BetaSeries*. https://www.betaseries.com/. Accedido: 2023-11. Idioma: español.

[6] (2023, noviembre). *IMDb - Internet Movie Database*. https://www.imdb.com/. Accedido: 2023-11. Idioma: español.

[7] (2023, noviembre). *Hulu*. https://www.hulu.com/welcome. Accedido: 2023-11. Idioma: español.

[8] (2023, noviembre). *Filmaffinity*. *Sitio web*. https://www.filmaffinity.com/. Accedido: 2023-11. Idioma: español.

[9] (2023, enero). *Moviefone*. *Sitio web*. https://www.moviefone.com/. Accedido: 2023-1. Idioma: español.

[10] (2021, agosto). *Como los servicios de streaming usan algoritmos*. En *Arts Managament & Technology Laboratory*. https://amt-lab.org/blog/2021/8/algorithms-in-streaming-services. Accedido: 2023-10. Idioma: español.

[11] (2023, mayo). *Django vs node.Js - what's the difference? (pros and cons)*. *Radixweb*. https://radixweb.com/blog/django-vs-nodejs. Accedido: 2023-10. Idioma: español.

[12] *TensorFlow.js*. *TensorFlow*. https://www.tensorflow.org/js?hl=es-419. Accedido: 2023-10.

[13] *Brain.Js: GPU accelerated Neural Networks in JavaScript*. *Brain.js.org*. https://brain.js.org//. Accedido: 2023-10.

[14] Towards Data Science. (2023, noviembre 4). *Recomendacion de sistemas: Matriz de Factorizacion*. https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b. Accedido: 2023-11-04. Idioma: español.

[15] Wikipedia. (2023, noviembre 4). *Descomposición en Valores Singulares*. https://es.wikipedia.org/wiki/Descomposici

[16] Graph Everywhere. (2024, Febrero). *Algoritmo de similitud de coseno*. https://www.grapheverywhere.com/algoritmo-de-similitud-de-coseno/: :text=LaIdioma: español.

[17] (2023, noviembre 4). *Autoencoders*. https://abdatum.com/tecnologia/autoencoders. Accedido: 2023-11. Idioma: español.

[18] Real Python. (2023, noviembre 4). *Building a Recommendation System with Collaborative Filtering*. https://realpython.com/build-recommendation-engine-collaborative-filtering/algorithms-for-matrix-factorization. Accedido: 2023-11. Idioma: español.

[19] (2023, noviembre). *Página oficial de Python*. https://www.python.org/. Accedido: 2023-11. Idioma: español.

[20] (2023, octubre). *Npm: Zod. npm*. https://www.npmjs.com/package/zod. Accedido: 2023-10. Idioma: español.

[21] (2023, mayo). *Express - Node.js web application framework*. *Expressjs.com*. https://expressjs.com/. Accedido: 2023-10. Idioma: español.

[22] (2023, octubre). *Crea rápidamente sitios web modernos HTML*. *Tailwindcss.com*. https://tailwindcss.com/. Accedido: 2023-10. Idioma: español.

[23] (2023, octubre). *Vue.Js - the progressive JavaScript framework*. *Vuejs.org*. https://vuejs.org/. Accedido: 2023-10. Idioma: español.

[24] (2023, octubre). *MongoDB Atlas database*. *MongoDB*. https://www.mongodb.com/atlas/database. Accedido: 2023-10. Idioma: español.

[25] *GitHub*. https://github.com/. Consultado: 2023-11. Idioma: inglés.

[26] Mozilla Developer Network. (2023, noviembre 4). *MVC - Modelo-Vista-Controlador*. https://developer.mozilla.org/es/docs/Glossary/MVC. Consultado: 2023-11. Idioma: español.

[27] CódigoFacilito. (2023, noviembre 4). *Middleware en Express*. https://codigofacilito.com/articulos/middleware-express. Consultado: 2023-11. Idioma: español.

[28] *Tailwind CSS - Installation*. https://tailwindcss.com/docs/installation. Accedido: 2023-12.

[29] *Features • GitHub actions*. Idioma: español.

[30] Na. (2019, agosto). *Sistemas de recomendación. Aprende Machine Learning*. https://www.aprendemachinelearning.com/sistemas-de-recomendacion/. Accedido: 2023-10. Idioma: español.

[31] IEBSchool. (2023, noviembre 4). *Qué son Metodologías Ágiles (Agile Scrum)*. https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/. Accedido: 2023-11. Idioma: español.

[32] Asana. (2023, noviembre 4). *Qué es Kanban?*. https://asana.com/es/resources/what-is-kanban. Accedido: 2023-11. Idioma: español https://es.wikipedia.org/wiki/DescomposiciAccedido: 2023-11-04. Idioma: español.

[33] TowardsDataScience(2024,Febrero). *How to Build From Scratch a Content-Based Movie Recommender with Natural Language Processing*. *Towards Data Science*.

https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243. Publicado: 2023. Idioma: inglés.

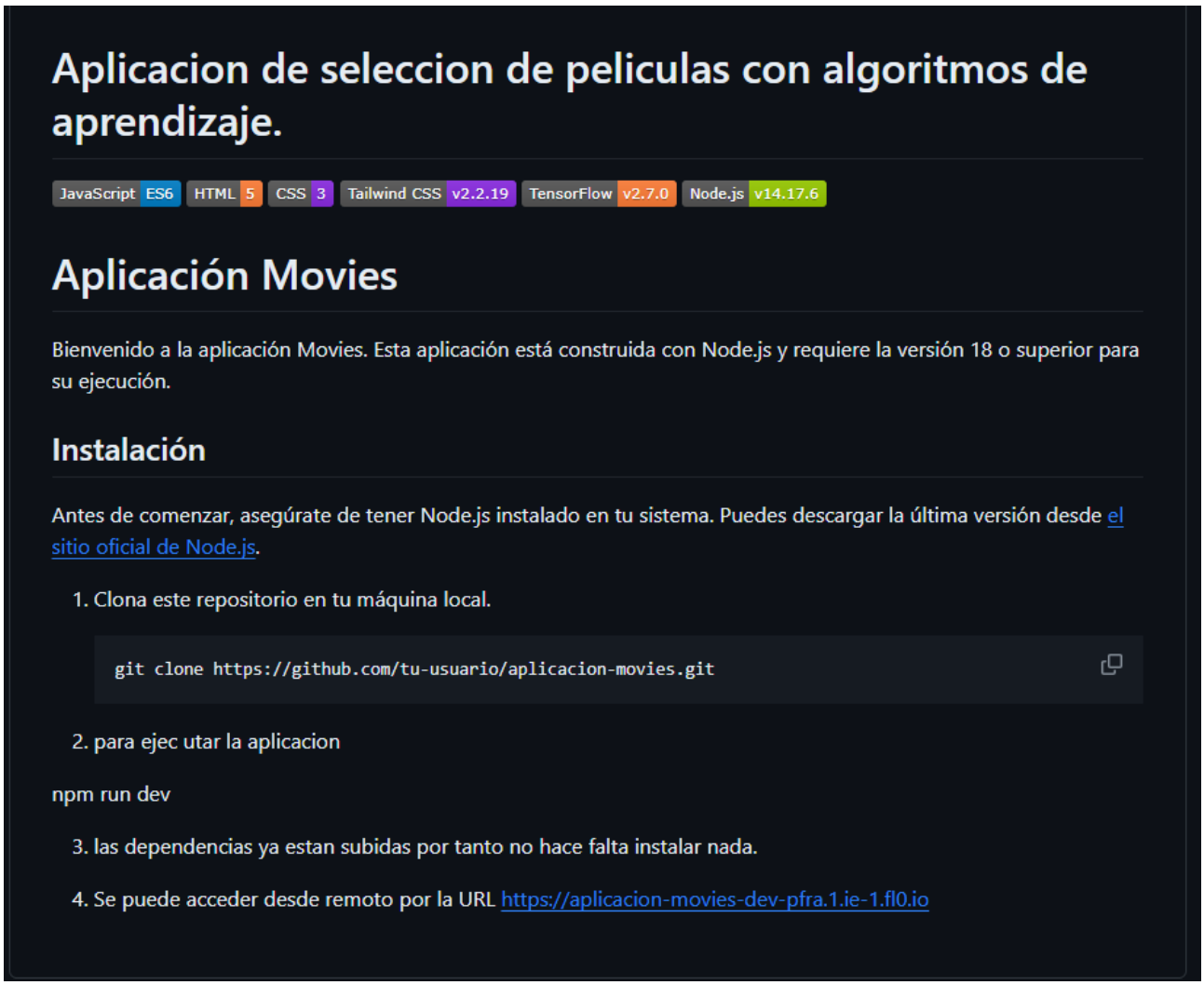


Fig. 1: Descripción de repositorio en GitHub

A Apéndice

A.1. Código Fuente en GitHub

Aquí está el enlace a nuestro repositorio de GitHub que contiene el código fuente de nuestra aplicación:

- Repositorio de GitHub: https://github.com/alekzihz/aplicacion-movies

A.2. Planificación

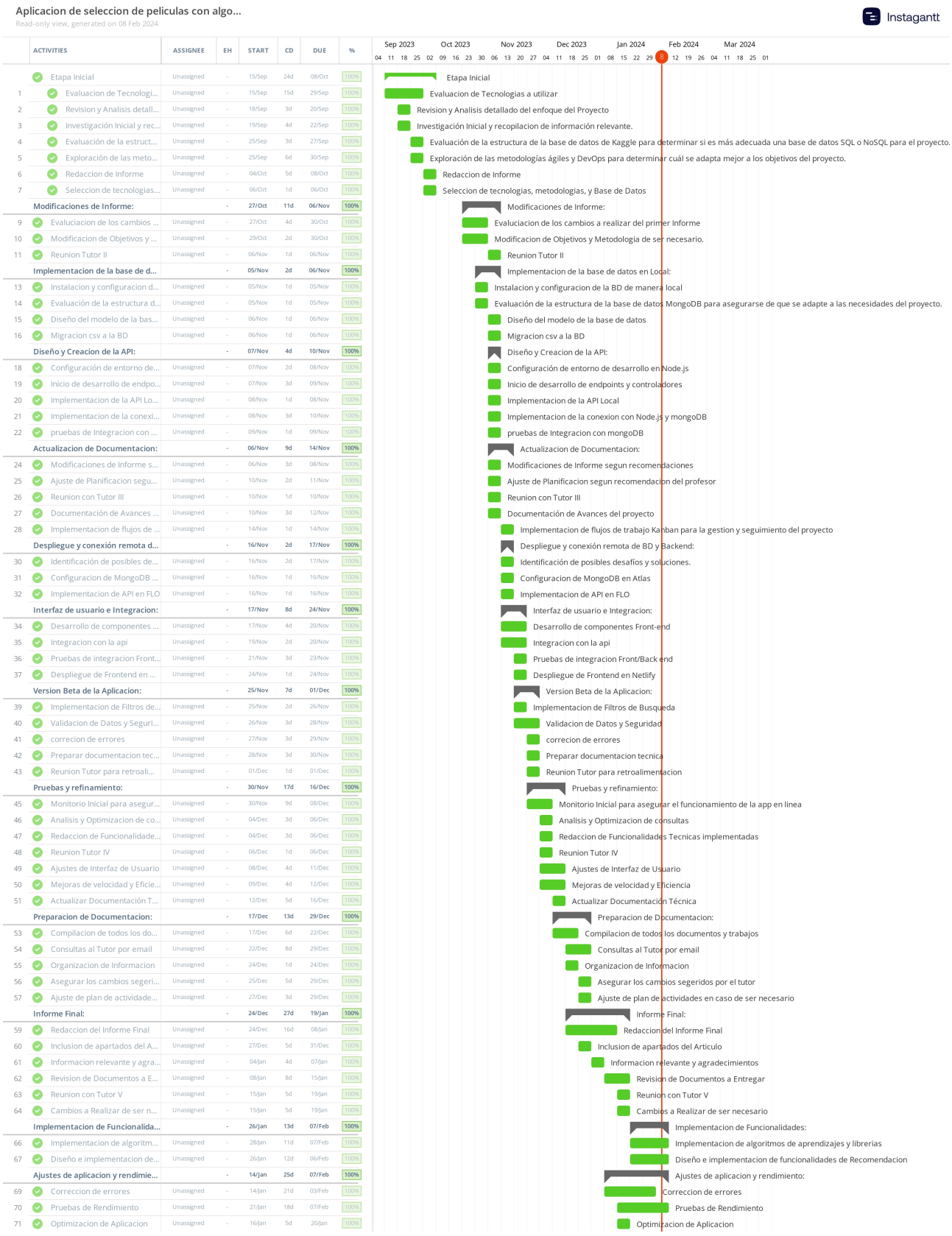


Fig. 2: Diagrama de Gantt de las actividades realizadas en el proyecto

A.3. Widget Justwatch

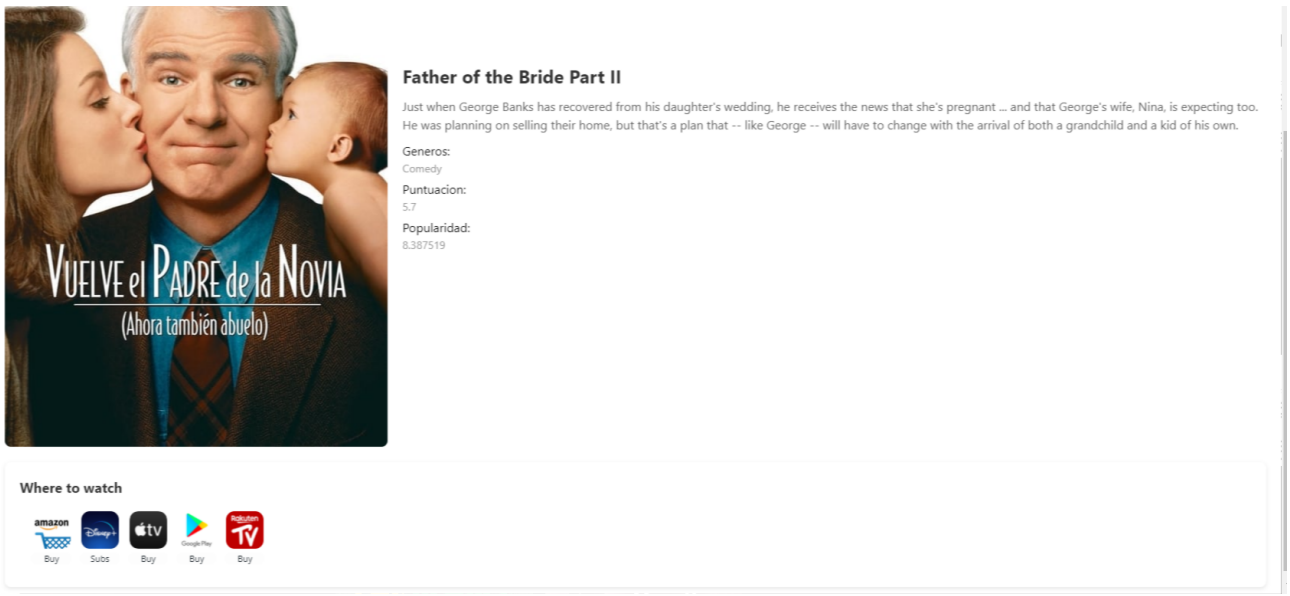


Fig. 3: Plataformas de stream disponibles por el Widget Justwatch

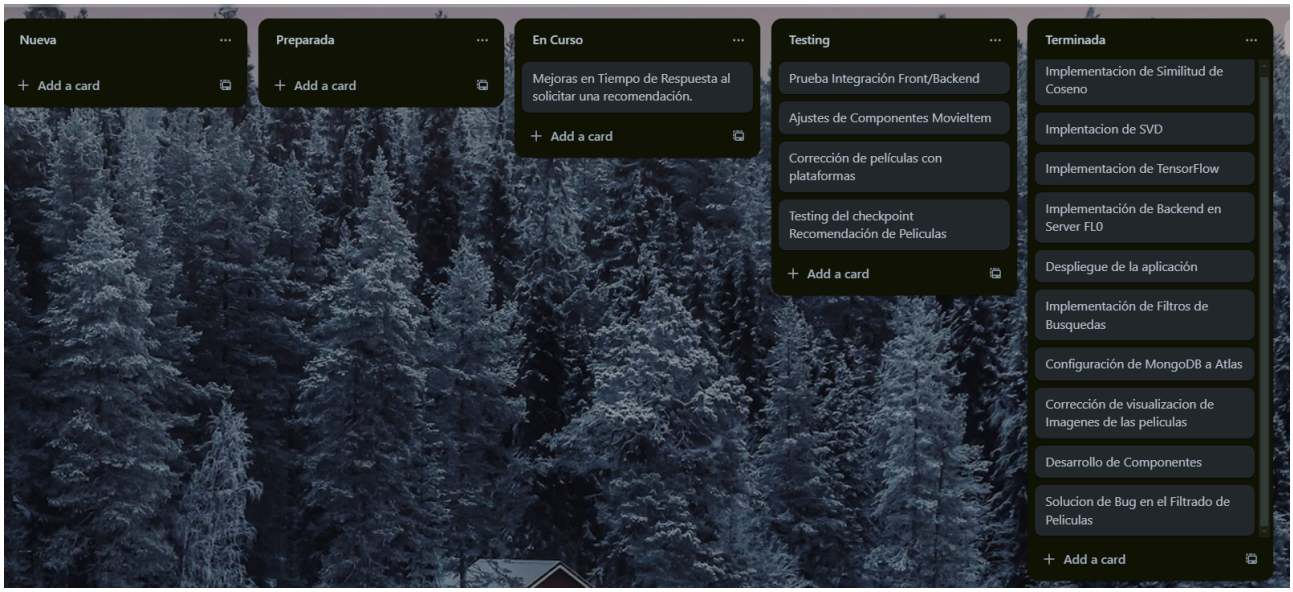


Fig. 4: Tabla Kanban Utilizada para la gestión del desarrollo de la aplicación

A.4. Tabla Kanban

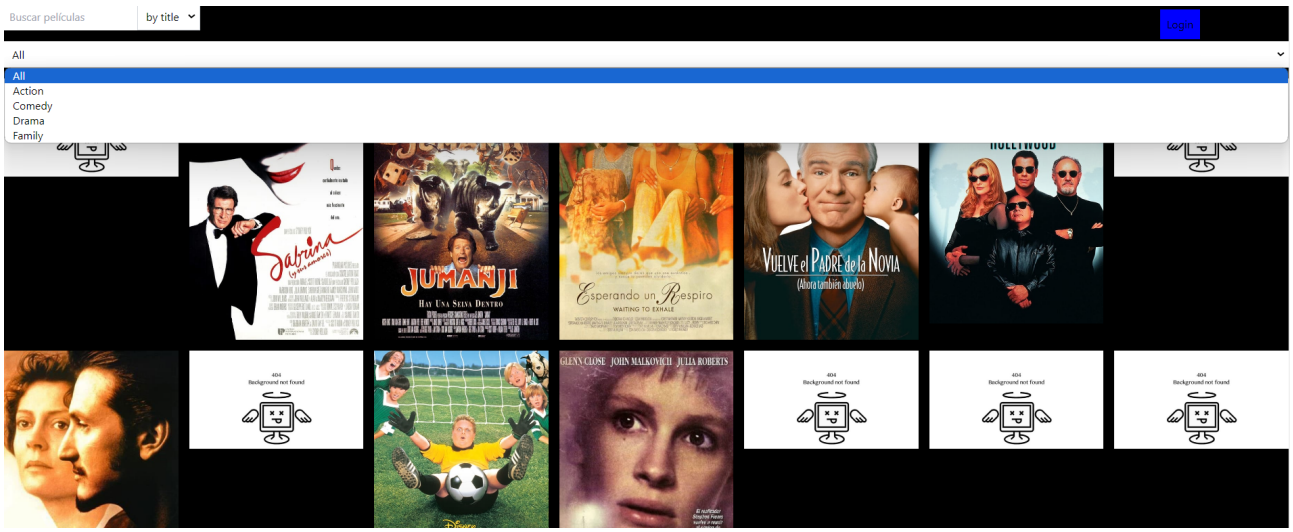


Fig. 5: Pagina principal de la Aplicación Web

A.5. Pagina Principal de la aplicación

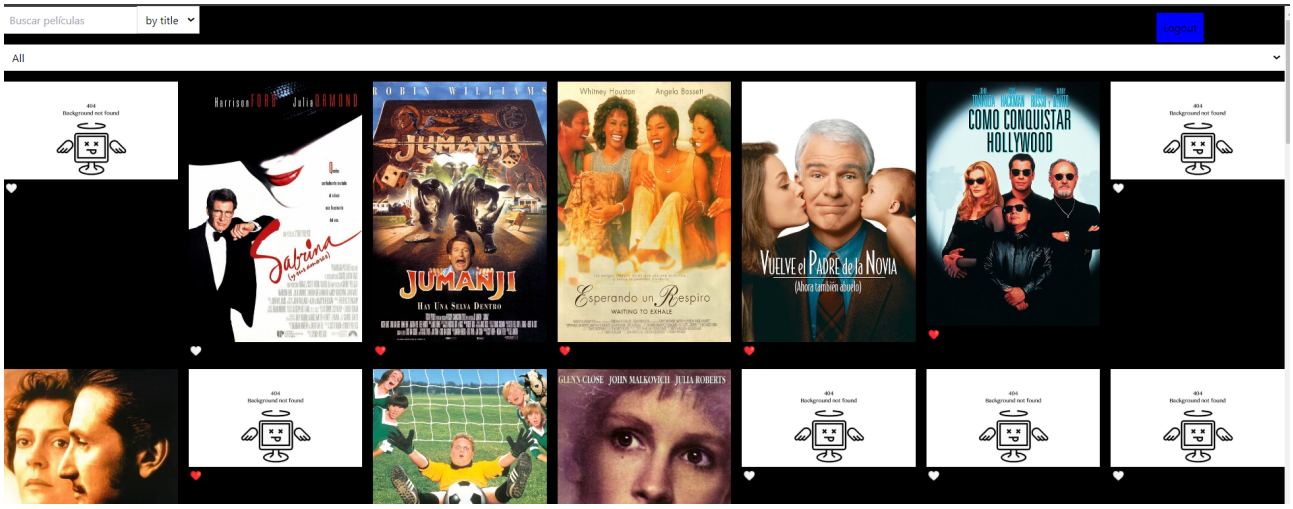


Fig. 6: Pagina principal del usuario

A.6. Pagina Principal Usuario