Recuperación de Información
y Recomendaciones en la Web

Curso 2023

Grupo 06

Informe Final

Autores:

Florencia Carle 5.355.244-8
Gonzalo Piria 5.151.202-4
Lucas de Sierra 5.170.666-7
Facundo Spira 5.092.383-0

Supervisor:
Libertad Tansini

Fecha de Entrega: 20 de noviembre, 2023.

Índice

1 Definición del problema

Debido a la alta oferta de plataformas de streaming, resulta un problema para los consumidores poder saber en cuál de ellas se encuentra el contenido que están buscando. Series y películas rara vez se encuentran en más de una plataforma, por lo que los usuarios deben buscar en cada una para poder verlas.

Existen además dos variables que realizan esta búsqueda más compleja; por un lado se encuentra la dificultad de que si la persona no se encuentra suscrita a alguna plataforma, la búsqueda de contenido en dicho sitio puede resultar más trabajosa. Por otro lado, si el sujeto es usuario de varias plataformas, este carece de un espacio donde todas sus visualizaciones se tomen en cuenta a la hora de generar recomendaciones de nuevo contenido para ver.

Es por esto que el problema se centra en crear un espacio donde, de manera centralizada, un usuario pueda no solamente encontrar películas o series de acuerdo a sus intereses, sino que también podrá conocer donde visualizar dicho contenido así como poder conocer información general de este.

Cabe además destacar que el proyecto se realiza tanto en el marco de la materia "Recuperación de Información y Recomendaciones en la Web" como de la materia "Integración de Datos", por lo que un problema adicional al que el grupo se enfrenta es la integración de información pertinente de diversas fuentes.

2 Solución propuesta

Con el fin de buscar una solución a los problemas planteados, se creó un página web donde los usuarios pueden encontrar diferentes películas y series en conjunto con las plataformas digitales donde pueden ser visualizadas en Uruguay. Se muestra además información del contenido audiovisual que puede ser de importancia para el usuario, como el trailer, reparto, duración, géneros, calificación promedio de los usuarios, entre otros.

La aplicación permite a las personas crear su propio usuario, con del fin de poder generar recomendaciones de contenido personalizadas. Para esto, dentro de la web, los usuarios podrán opcionalmente calificar las series o películas que hayan visualizado. Luego, en base a estas calificaciones se utiliza la técnica de *filtrado colaborativo* para realizar recomendaciones.

3 Fuentes de Datos

Los datos de series y películas que vamos a utilizar los obtendremos de dos sitios diferentes: TMDb (The Movie Database) [9] y IMDb [4]. Ambos sitios proveen APIs REST las cuales cuentan con información sobre series y películas, con distintos niveles de granularidad. Por otro lado, utilizaremos Wikidata [13] para ampliar la información de los actores.

3.1 API de TMDb

TMDb, o "The Movie Database", es una plataforma en línea que se centra en películas y programas de televisión. Ofrece una extensa base de datos con detalles sobre títulos, elenco, equipo de producción, calificaciones y sinopsis. Provee además una API que puede usarse libremente siempre y cuando se le atribuya y se utilice sin fines comerciales.

3.2 API de IMDb

IMDb (Internet Movie Database) es una plataforma en línea propiedad de Amazon que ofrece una amplia base de datos sobre películas, programas de televisión, actores y profesionales del entretenimiento. Los usuarios pueden encontrar información detallada sobre películas, programas, elenco, equipo de producción, calificaciones y críticas.

3.3 Wikidata

Wikidata es una base de datos colaborativa y gratuita que almacena una amplia gama de información estructurada y enlazada sobre una gran variedad de temas, incluyendo personas, lugares, obras de arte, eventos históricos y mucho más. Es parte de la iniciativa Wikimedia y se utiliza para complementar y enriquecer la información de otros proyectos de Wikimedia, como Wikipedia. Utilizaremos este sitio para extender la información que se tiene de los actores/actrices de las películas. En particular nos interesa obtener nombre, foto, ocupación y premios recibidos.

4 Integración de los datos

En primer lugar, utilizando la API de TMDb, se obtuvo un volumen considerable (1234 en total) de películas y series en orden de popularidad. Utilizando otros endpoints del mismo sitio, se expandió esta información, agregando los proveedores de streaming que cuentan con el título e información adicional del mismo.

Una vez obtenidos estos datos, se utilizó un endpoint de TMDb que permite, dado un ID de serie o película, obtener el ID correspondiente en IMDb. Este ID se utilizó para consumir la API de IMDb y poder obtener información adicional sobre la serie/película.

Luego, se debió integrar la información obtenida de ambas fuentes. Para cada título, existían datos exclusivos en cada API, por ejemplo, TMDb poseía información de los servicios de streaming, mientras que en IMDb se encontraban datos curiosos sobre la producción, frases célebres y la lista de actores de la obra. Sin embargo, existe una serie de campos que se encuentra en ambos sitios y pueden tener discrepancias. Para esto se seguió la siguiente metodología:

- Para las discrepancias entre presupuesto, recaudación y duración, se tomó el promedio entre las dos obtenidas.
- Para la fecha de estreno de la película/serie, se guarda en el esquema integrada la más antigua
- Un desafío interesante consistió en integrar la lista de géneros de una película/serie y la trama de la misma. Sucedía que ambos sitios dos devolvían cosas diferentes, que muchas veces se solapaban y agregaban algo de información que el otro sitio no tenía. Luego, en este caso, la estrategia de integración se basó en el uso de inteligencia artificial, particularmente utilizando grandes modelos de lenguaje (LLM) para procesar ambas respuestas y obtener una respuesta integrada. Para esto se utilizó el último modelo de OpenAI [6] hasta la fecha, `gpt-4-1106-preview`, con una instrucción en particular. En el anexo A se puede ver un ejemplo con un pedido y respuesta del modelo utilizando dicha instrucción.

Por último, una vez obtenida la lista de actores provista por IMDb, se utiliza Wikidata para expandir la información de estos. Desde IMDb se obtiene una lista de los actores que participan de la obra (elenco). Para cada actor nos interesa obtener su nombre, foto, lista de ocupaciones y lista de premios recibidos (si es que ganó alguno). Desde IMDb solo obtenemos una lista de IDs para estos actores. Luego, utilizando estos valores, escribimos una query SPARQL [8] para consumir la información de Wikidata. La respuesta contará con los campos que se detallan anteriormente, y nos permitirá enlazar las películas y series con información más detallada de los actores que participan en la misma. La query SPARQL que se utilizará puede verse en el anexo B.

5 Herramientas utilizadas y arquitectura

5.1 Tecnologías

5.1.1 Frontend

El frontend de la aplicación se construyó con **Next.js** [12], un framework de React que, entre otras cosas, añade la opción de utilizar Server Side Rendering (SSR). La elección de Next por sobre una aplicación React ordinaria se realizó para poder aprovechar la capacidad que nos provee el SSR, realizando pre-fetching de las solicitudes al backend y así dando una mejor experiencia de usuario.

Como lenguaje subyacente se utilizó **Typescript** [5], un lenguaje de programación hecho por encima de Javascript, el cual provee tipado a las aplicaciones. La elección de Typescript por sobre Javascript se basa puramente en una mejora en la experiencia de desarrollo, haciendo que sea más fácil trabajar en equipo, y ayudando a detectar errores en tiempo de compilación (transpilación) evitando que estos ocurran en ejecución.

Se utilizó además una librería llamada **Clerk.js** [3] para facilitar la autenticación y el manejo de usuarios. Clerk provee componentes para aplicaciones React que facilitan la integración de los flujos de registro, inicio de sesión y personalización de la cuenta.

5.1.2 Backend

El backend de la aplicación se construyó con **Ruby on Rails** [7], un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby.

Esta tecnología se basa en el patrón de diseño MVC y el principio "convención antes que configuración". El resultado de estas decisiones arquitectónicas son aplicaciones muy bien estructuradas.

Además, Rails provee un ORM integrado, ActiveRecord, que permite realizar consultas a la base de datos de forma extremadamente fácil y sin necesidad de escribir nada de SQL. En particular, ActiveRecord provee varios mecanismos que permiten:

- Representar modelos y sus datos.
- Representar asociaciones entre estos modelos.
- Representar jerarquías de herencia a través de modelos relacionados.
- Validar modelos antes de que se guarden en la base de datos.
- Realizar operaciones en la base de datos de manera orientada a objetos.

5.2 Bases de datos

- Para almacenar información sobre los usuarios, y potencialmente información adicional utilizaremos **Postgres** [10]. Esta elección es debido a que Postgres es una base de datos relacional altamente confiable y robusta que se integra de manera excelente con Ruby on Rails y ActiveRecord, proporcionando un sólido soporte para la gestión de datos
- Por otro lado, se utilizó **ElasticSearch** [2] para guardar la información procesada y trabajada sobre las películas/series. Esta elección se debió a varias razones, entre las que destacamos:
  - La velocidad y eficiencia en el manejo de grandes volúmenes de datos de ElasticSearch, lo que lo hace ideal para indexar y consultar bases de datos extensas.
  - Las capacidades de búsqueda de ElasticSearch, que incluyen la búsqueda de texto completo y opciones avanzadas de filtrado, permitirán la sugerencia de películas relevantes a los usuarios.

  En resumen, la versatilidad, velocidad y capacidades de búsqueda de ElasticSearch hacen que sea una herramienta muy útil para construir una aplicación de recomendaciones de películas y series de forma dinámica y efectiva.

5.3 Diagrama de arquitectura

Todo lo mencionado en las partes anteriores puede resumirse en el siguiente diagrama de arquitectura de la fig. 1.

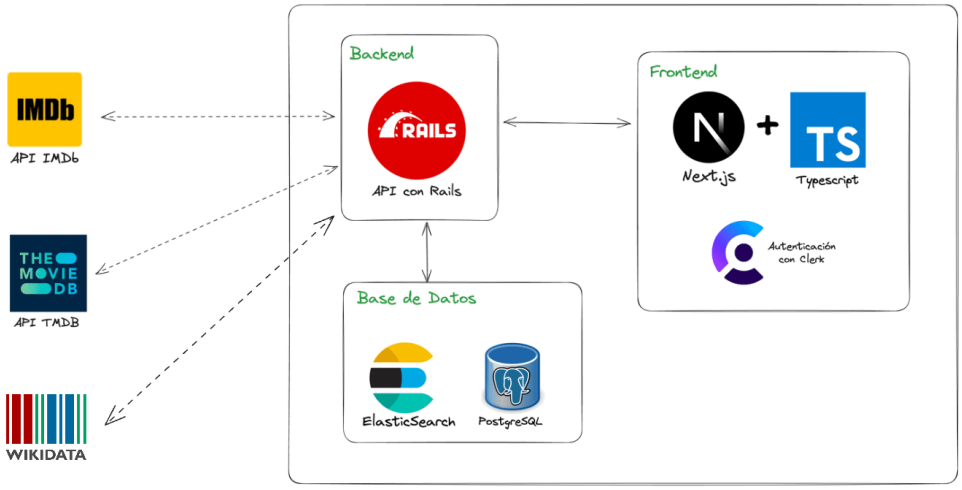

Figura 1: Diagrama de Arquitectura

5.4 Despliegue de la aplicación

Para poder consumir la aplicación decidimos desplegarla. El Frontend se desplegó en Vercel [11], una plataforma en la nube que se especializa en la implementación y el alojamiento de aplicaciones web. Ofrece una solución integral para desarrolladores que desean desplegar sus proyectos de manera rápida y sencilla. La misma puede encontrarse en la siguiente dirección: `https://syncnema.vercel.app`.

Por otro lado, tanto la API desarrollada en Rails, como la base de datos Postgres y Elasticsearch se desplegaron en **MiNube** [1] de Antel, utilizando la "Plataforma como Servicio" (PaaS) **Elastic Cloud**. Durante el proceso tuvimos algunos problemas con la plataforma. Para varias de las cosas que queríamos hacer la plataforma no resultaba del todo intuitiva. Además, la documentación con la que se contaba era un poco pobre, lo que dificultaba aún más el proceso.

Luego de probar con diferentes servidores, pudimos realizar el despliegue de la API utilizando un servidor NGINX. El mismo se conecta con un repositorio de Github donde se encuentra la API, y luego conectándonos por SSH al contenedor podemos correr los comandos de instalación para poder luego finalmente levantar la aplicación. También utilizamos un nodo con una base de datos Postgres, y un nodo con una imagen de Docker con ElasticSearch. Un resumen de la infraestructura utilizada puede verse en la fig. 2 tomada de la consola de *MiNube*.

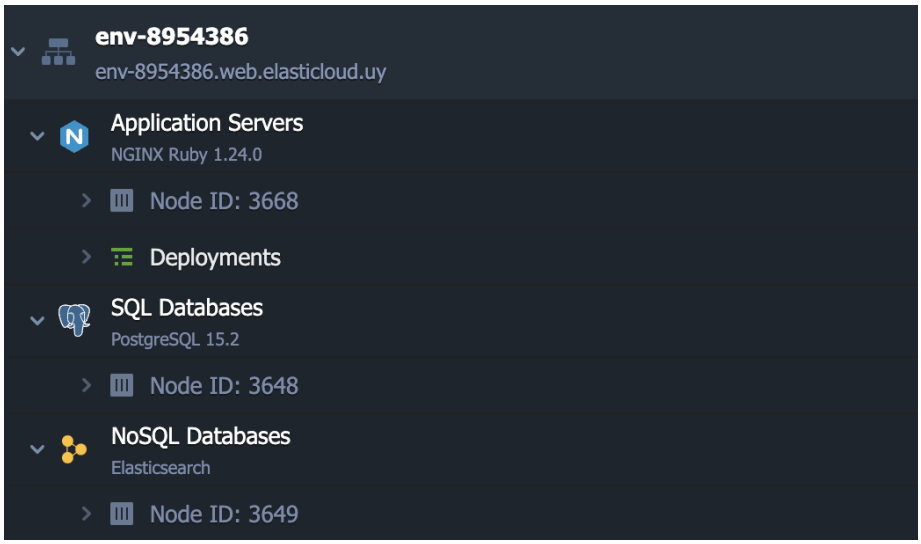

Figura 2: Recursos utilizados en MiNube para el despliegue del Backend

6 Funcionalidades y uso

6.1 Flujo de la aplicación - Recuperación de Información

Página de Home

Al ingresar a la página, el usuario se encontrará con la pantalla *Home*, donde le serán presentados títulos que podrían gustarle, en conjunto con una lista de tanto series como películas que se encuentran en tendencia actualmente. Si el usuario no se encuentra logueado, se le dará también la opción de iniciar sesión. Una imagen de dicha página puede verse en la fig. 3.

Las recomendaciones que se le hacen al usuario son de dos tipos diferente:

- Si el usuario **no** se encuentra logueado, se le recomiendan películas y series de las que se encuentran en tendencia.
- Si el usuario se encuentra logueado, se utiliza la técnica de filtrado colaborativo, basándonos en sus calificaciones y las calificaciones de otros usuarios. En la sección siguiente se detalla sobre la implementación de la misma.

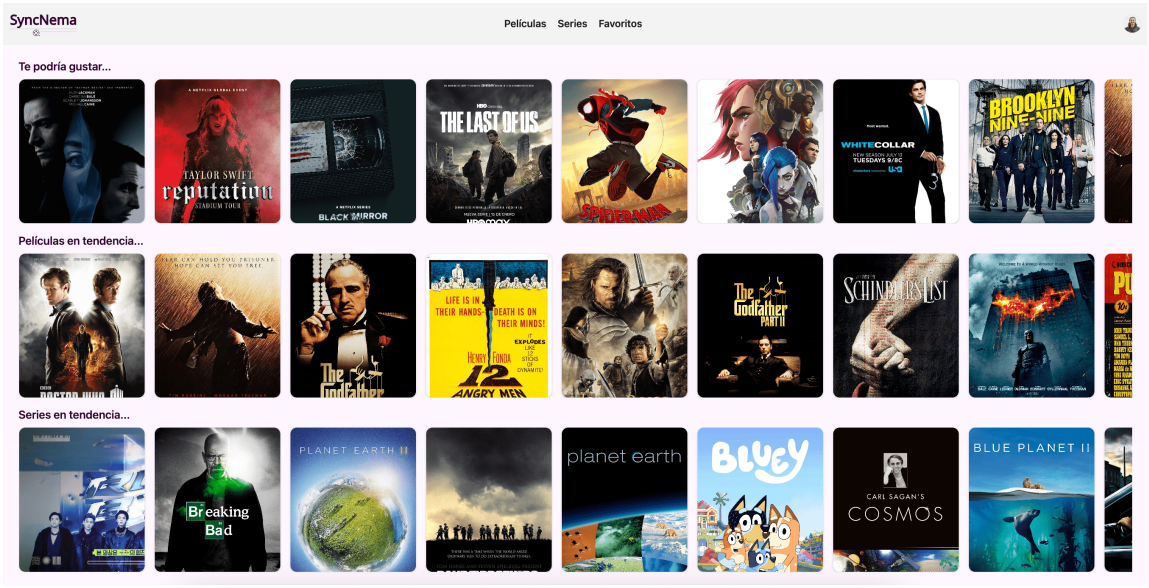

Figura 3: Página principal de la aplicación

Desde esta pantalla, el usuario podrá elegir un título de los presentados, opción que lo redirigirá a los detalles del contenido seleccionada, o bien navegar a una de tres pantallas adicionales: Películas, Series o Favoritos.

Página de detalle de una serie/película

Si se entra a una serie o película en particular, el usuario podrá ver el detalle de la misma. Aquí podrá ver información adicional como la trama, fecha de estreno, duración, géneros, nombre del creador, servicios de streaming en donde se puede ver el contenido, la calificación del contenido, y la lista de actores que participaron en la realización del mismo. Podemos ver un ejemplo en la fig. 4.

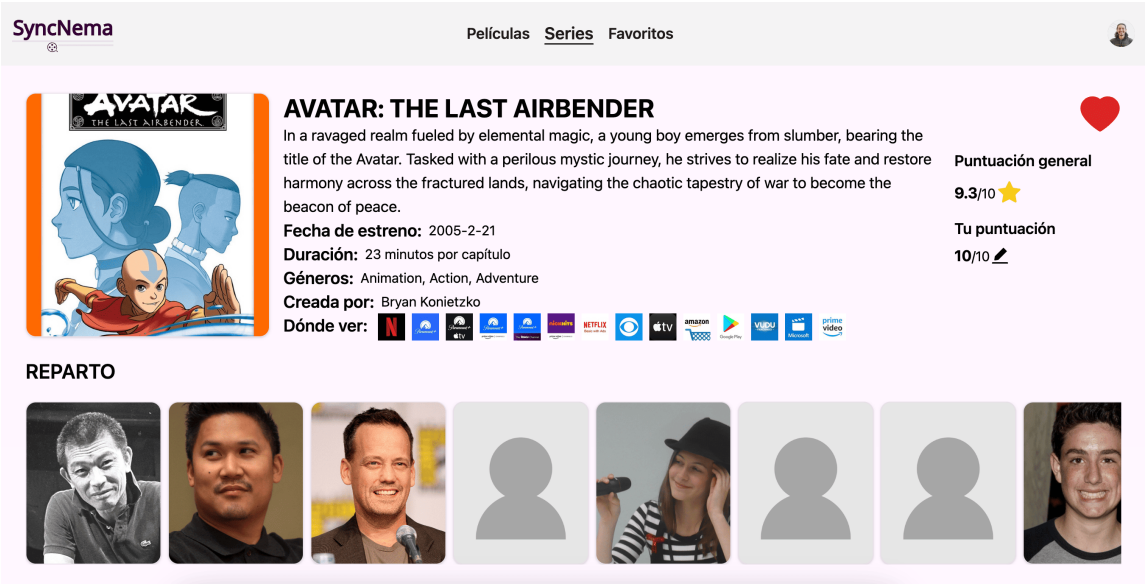

Figura 4: Detalle de una serie en particular

Además, en caso de que el usuario se encuentre logueado, podrá realizar lo siguiente:

- Agregar o quitar la película/serie de favoritos utilizando el botón con el corazón.
- Calificar (y más adelante cambiar la calificación) utilizando un selector de estrellas en un rango del 0 al 10. La interfaz para puntuar una película o serie puede verse en la fig. 5

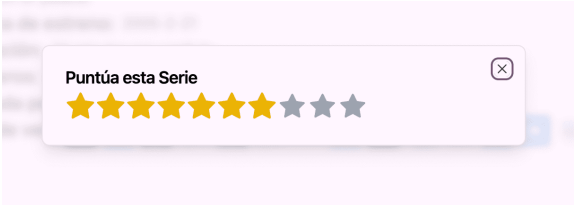

Figura 5: Interfaz para calificar una película o serie

Página de listado de series y películas

Las pantallas de Películas o Series son muy similares, diferenciándose tan solo en el tipo de contenido que muestran. En estas pantallas el usuario cuenta con una barra de búsqueda (la cuál permite buscar según el nombre del contenido, nombre del creador y nombre de los actores del elenco), así como un filtrado por género y/o por plataforma que le interesa. Podemos ver la interfaz en la fig. 6.

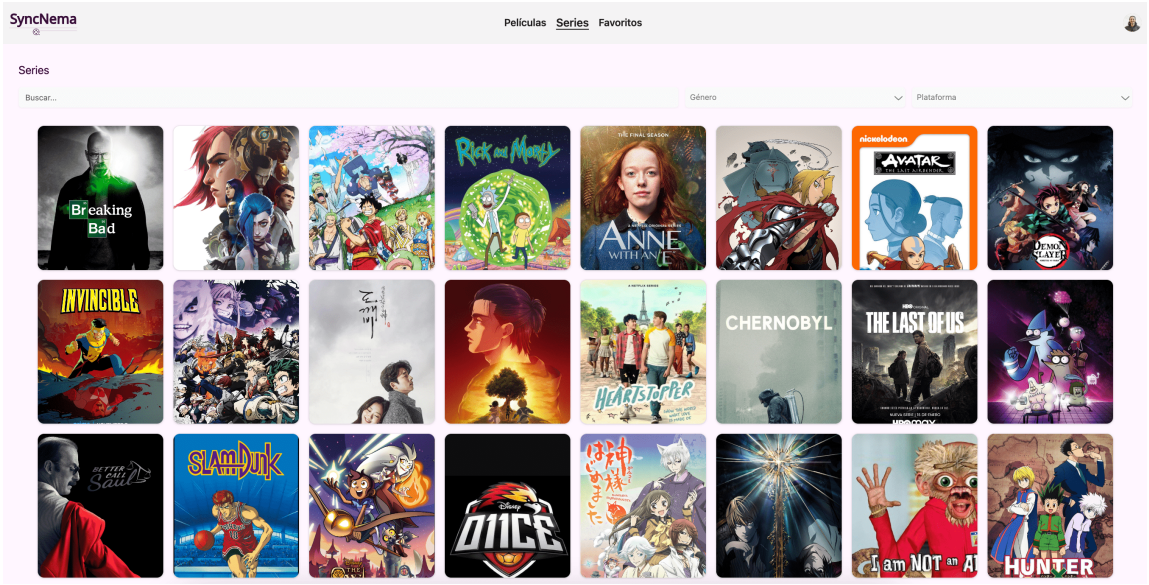

Figura 6: Página con listado de series

La API utiliza ElasticSearch para poder devolver la información a los usuarios lo más rápido posible. Para esto se hace una *query Bool* con una clausula *must* y otra *should*. La *must* siempre se debe cumplir y dentro de la *should* se debe cumplir al menos 1 sub-cláusula si se pasó el parámetro "query" o ninguna si este no esta presente.

- La clausula *must* esta conformada por 3 subcláusulas. Una de tipo *match_phrase* donde se matchea el parámetro *type* con la columna *type* del contenido (para diferenciar películas de series). Una de tipo *query_string* donde se hace un *OR* de los géneros y se busca en el campo $combined_genres$. Por último una de tipo *nested*, que busca dentro de los atributos nested donde se guardan las plataformas, y hace una query *query_string* donde se hace un *OR* de las plataformas.
- La clausula *should* consta de 4 subcláusulas, las primeras 3 son de tipo *query_string*, y utilizan el parámetro "query" para buscar en los campos *title director* y *creator*. La última es de tipo *nested*, que busca dentro del atributo nested donde se guardan los actores, y hace un *match_phrase* para buscar coincidencias con sus nombres.

Pantalla de favoritos

La última pantalla con la que cuenta la aplicación es la pantalla de "Favoritos", la cuál muestra los títulos que el usuario seleccionó como tales. Esta pantalla se encuentra subdividida en el tipo se contenido, es decir, si es una serie o película. Puede verse la interfaz en la fig. 7.

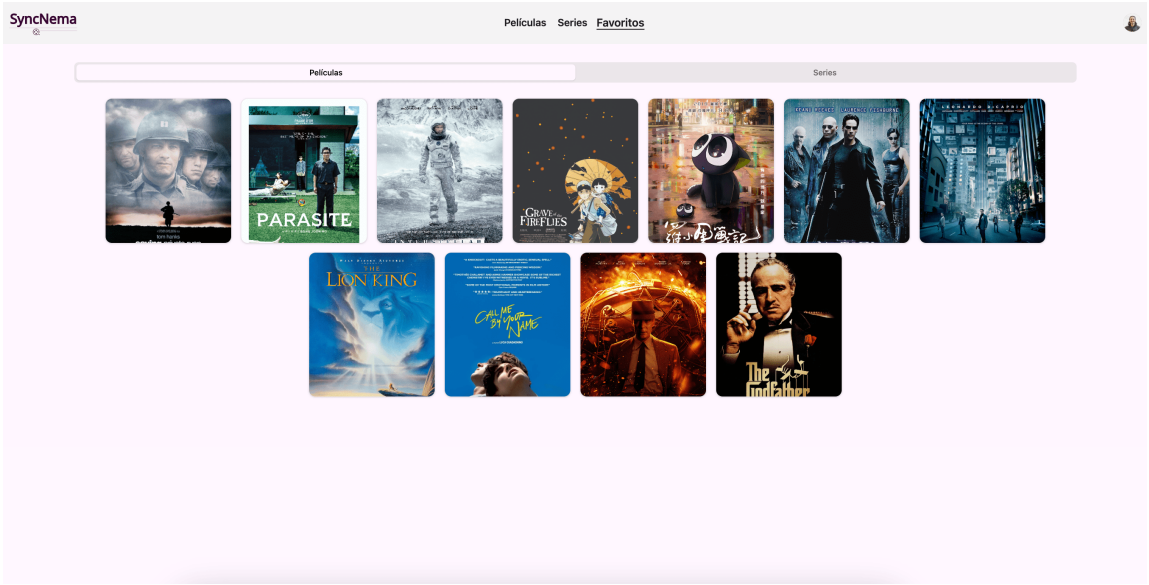

Figura 7: Página de favoritos

6.2 Filtrado colaborativo

Como comentamos anteriormente, utilizamos la técnica de filtrado colaborativo para realizar recomendaciones de series y películas a los usuarios. Intentamos utilizar una gema de Rails llamada *disco*. Dicha gema utiliza filtrado colaborativo y factorización matricial de alto rendimiento para realizar recomendaciones de forma eficiente. Sin embargo, no pudimos utilizarla debido a que resultó no ser compatible con las versiones que estábamos utilizando en el deploy, y la plataforma MiNube no nos permitía actualizar la misma.

Por tanto, decidimos implementar una versión a mano. La idea intuitiva era recomendarle a un usuario películas y series que hayan sido calificadas de forma positiva por usuarios que son "parecidos" a ellos. Consideramos que dos usuarios son parecidos si tienen gustos parecidos. En este contexto esto significa que hayan calificado de forma similar un conjunto de películas/series. De esta forma, en caso de que dos usuarios sean parecidos, podemos recomendarle a uno, cosas que al otro le hayan gustado.

Para implementarlo, realizamos lo siguiente. Dado un usuario

- Se toman las puntuaciones que hizo y las películas y series que puntuó.
- Se buscan puntuaciones de otros usuarios a ese set de películas y series.
- Se calcula una métrica de similitud (que dado una puntuación a una película y otra puntuación a la misma da un numero mayor cuanto mas parecidas son) para las puntuaciones de estos usuarios comparándolas con la del usuario seleccionado.
- Se agrupan estas puntuaciones por usuario y se asigna una métrica de similitud total que suma todas las métricas individuales para cada usuario. Luego se ordena este array agrupado en base a que usuario tiene mayor puntuación de similitud.
- Se recorre este agrupamiento y para cada usuario se buscan puntuaciones a otras películas que el usuario original no haya ya puntuado, guardándolas en el orden obtenidas.
- Dado el array final de puntuaciones se calcula una métrica para cada una sumando la métrica de similitud del usuario y el numero de esta puntuación (por ejemplo si el usuario tenia métrica de similitud 10 y puntúo a cierta película con un 8, la nueva métrica dará 18, mientras que si otro usuario tenia una métrica de similitud 9 pero puntúo una película con un 10, va a dar 19 por lo que se mostrara primero) y se devuelven las películas asociadas a estas puntuaciones en orden, sin repetidas.
- En caso de no contar con suficiente cantidad de recomendaciones para devolver, se completa esta lista con películas y series que actualmente se encuentran en tendencia.

7 Conclusiones

En conclusión, se logró desarrollar una aplicación que cumple con los objetivos que nos planteamos. Además, consideramos que fue un proyecto muy interesante que nos permitió llevar muchos de los conceptos vistos en el curso a la práctica. Nos resultó de particular interés ya que pudimos probar nuevas tecnologías, como lo es el uso de ElasticSearch y integraciones con OpenAI y APIs externas para obtener series y películas.

Se encontró además que la aplicación parece tener potencial de crecimiento, tanto en términos de utilidad del producto para el usuario como de posibilidades de mejoras y trabajo a futuro, por lo que podría ser interesante seguir iterando sobre esta.

8 Trabajo futuro

Como posible trabajo a futuro, la aplicación puede se ampliada de diversas maneras.

- Se puede realizar una extensión a los filtros que el usuario tiene disponibles para encontrar el contenido que busca, como pueden ser filtros asociados con los actores que forman parte del reparto, premios que haya ganado el título, duración, etc. Además se puede incluir una manera de ordenar el contenido por diferentes categorías como las mencionadas previamente.

- Podría ampliarse aún más la información provista al usuario, por ejemplo mostrando páginas de detalle de los miembros del reparto, donde pueda verse información sobre los actores y otros títulos de los que haya formado parte. También podría proveerse información sobre cada capítulo en las temporadas de una serie.

- A la hora de realizar recomendaciones, podría también tenerse en cuenta las series y películas vistas por el usuario, hayan sido o no calificadas por este, así como agregar una pantalla de vistos, donde se le muestre al usuario los títulos que haya visualizado.

- Podría resultar beneficioso agregar una página que permita a los usuarios ver una lista con todas las películas y series que calificó, pudiendo desde allí mismo cambiar la calificación obtenida.

- Además, sería interesante poder agregar algún mecanismo que permita la traducción del contenido de la página a diversos idiomas, así como poder determinar la localización del usuario para poder mostrar las plataformas que estén disponibles en su país, sin tener que estar atados a Uruguay.

- Por último, se podría intentar encontrar alguna biblioteca existente que implemente el filtrado colaborativo para poder mejorar esta funcionalidad.

Referencias

[1] Antel. (2023). Mi Nube Antel. Retrieved November 12, 2023, from Minubeantel.uy website: https://minubeantel.uy/index.php?NAME_PATH=Elastic_Cloud

[2] Búsqueda gratuita y abierta: Los creadores de Elasticsearch, ELK y Kibana — Elastic. (n.d.). Retrieved September 10, 2023, from www.elastic.co website: https://www.elastic.co/es/

[3] Clerk. (n.d.). Clerk — Authentication and User Management. Retrieved September 10, 2023, from Clerk website: https://clerk.com/

[4] IMDb API. (2023). Retrieved October 22, 2023, from Workers.dev website: https://search.imdbot.workers.dev/

[5] Microsoft. (2015). TypeScript - JavaScript that scales. Retrieved September 10, 2023, from Typescriptlang.org website: https://www.typescriptlang.org/

[6] OpenAI. (2019, April 25). OpenAI. Retrieved from OpenAI website: https://openai.com/

[7] Ruby on Rails. (n.d.). Retrieved September 10, 2023, from Ruby on Rails website: https://rubyonrails.org/

[8] SPARQL 1.1 Query Language. (2013). Retrieved October 22, 2023, from W3.org website: https://www.w3.org/TR/sparql11-query/

[9] The Movie Database. (2018). Retrieved October 22, 2023, from Themoviedb.org website: https://www.themoviedb.org/

[10] The PostgreSQL Global Development Group. (2019). PostgreSQL: The world's most advanced open source database. Retrieved September 10, 2023, from Postgresql.org website: https://www.postgresql.org/

[11] Vercel. (2023). Develop. Preview. Ship. For the best frontend teams – Vercel. Retrieved November 12, 2023, from vercel.com website: https://vercel.com/

[12] Vercel. (n.d.). Next.js by Vercel - The React Framework. Retrieved September 10, 2023, from nextjs.org website: https://nextjs.org/

[13] Wikidata. (n.d.). Wikidata. Retrieved October 22, 2023, from www.wikidata.org website: https://www.wikidata.org

A Ejemplo de integración de la trama de una película

A.1 Pedido

Se realiza el siguiente pedido utilizando el modelo `gpt-4-1106-preview`. Se realizó en inglés ya que la información de películas y series obtenidas era en inglés.

```
You are a very enthusiastic cinephile who loves to combine movie plots
from different sites. Given the following movie plots, create another one
that merges all the important information. Try to not make it longer than
100 words.

Plot 1: As U.S. troops storm the beaches of Normandy, three brothers lie
dead on the battlefield, with a fourth trapped behind enemy lines. Ranger
captain John Miller and seven men are tasked with penetrating German-held
territory and bringing the boy home.

Plot 2: Following the Normandy Landings, a group of U.S. soldiers go
behind enemy  lines to retrieve a paratrooper whose brothers have been
killed in action.

Combined plot:
```

A.2 Respuesta

Para este caso en particular se obtuvo la siguiente respuesta, la cuál fue agregada al esquema integrado.

```
Following the Normandy Landings, a group of U.S. soldiers, led by Ranger
captain John Miller, are assigned the mission of retrieving a paratrooper
trapped behind enemy lines. This paratrooper, whose brothers have been
killed in action, must be brought home safely as the soldiers navigate
German-held territory. The team of seven men faces numerous challenges
and risks as they endeavor to fulfill their mission and reunite the
paratrooper with his family.
```

B Query SPARQL para enlazado con Wikidata

Dadao una lista de IDs de IMDb, obtenemos la información sobre el actor presente en Wikidata utilizando la siguiente consulta SPARQL.

```
SELECT
    ?actorName ?actorImage
    (GROUP_CONCAT(DISTINCT ?award;separator="|") AS ?actorAwards)
    (GROUP_CONCAT(DISTINCT ?occupation;separator="|") AS ?occupations)

WHERE {
  ?actor wdt:P31 wd:Q5 ;
         wdt:P345 ?imdbId ;
         rdfs:label ?actorName .

  FILTER(LANG(?actorName) = "es")

  OPTIONAL {
    ?actor wdt:P18 ?actorImage .
  }

  OPTIONAL {
    ?actor wdt:P166/rdfs:label ?award .
    FILTER(LANG(?award) = "es")
  }

  OPTIONAL {
    ?actor wdt:P106/rdfs:label ?occupation .
    FILTER(LANG(?occupation) = "es")
  }

  FILTER (?imdbId IN ("nm7838852", "nm6185683"))
}
GROUP BY ?actorName ?actorImage
```