Recuperación de Información y Recomendaciones en la Web

Curso 2022

Grupo 1

Guillermo Waltes	5103218-3
Julieta Caffera	5235540-1
Karen Rasmussen	5077838-6
Luis Felipe Zamora	4922307-5
Martín Schnaiderman	4775066-6

Indice:

Descripción del problema

Con la sobrecarga de plataformas de streaming la búsqueda de contenido (de calidad) se ha vuelto un problema de interés para los usuarios. Encontrar series o películas que resulten interesantes para los diversos grupos de personas, no es una tarea sencilla y si bien la mayoría de los servicios de streaming intentan solucionar este problema entendemos que lo hacen sin éxito. En general buscan priorizar el contenido propio y de esa forma reducir los costos variables evitando pagar a las empresas dueñas de ese contenido las regalías correspondientes. También sucede que se sobre-simplifican las recomendaciones, enfocándose únicamente en los protagonistas o director de la misma sin tener en cuenta el rating de estas.

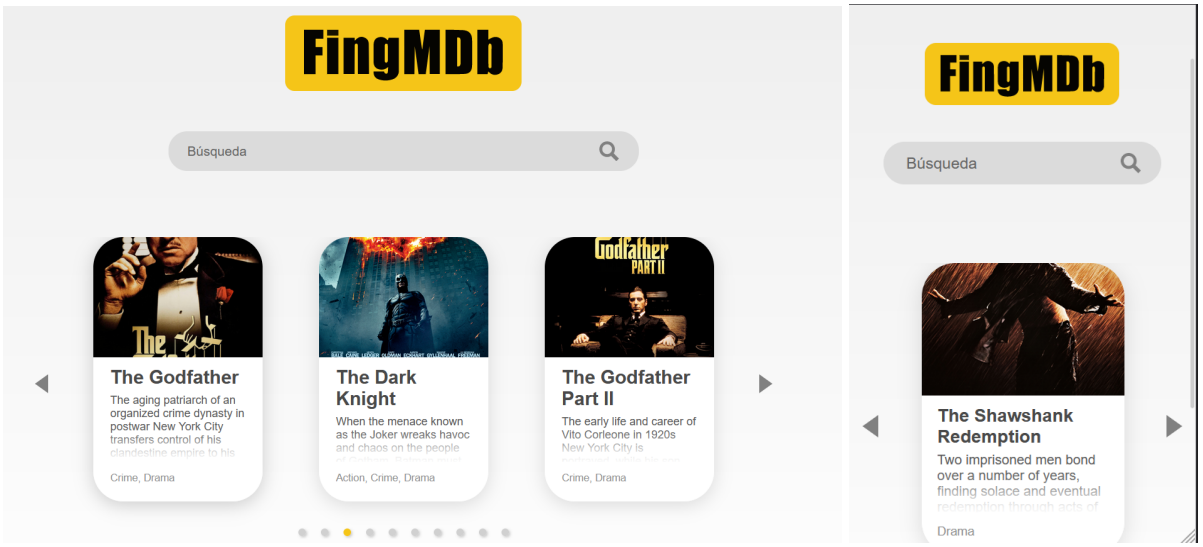
Entendemos que no es fácil determinar la calidad de una película en general y como se dijo antes estas plataformas buscan su propio beneficio y no el de los usuarios, por esto se apuntó a generar un buscador que utilice ratings más neutrales realizados por usuarios. Creemos que si utilizamos el rating de millones de usuarios que incluso están por fuera de las plataformas, los resultados serían más certeros y sin sesgos de plataformas.

Propuesta de solución

Como solución proponemos una web que muestre información relacionada a una película o serie. La aplicación también permite que se pueda filtrar por género y año, ya que entendemos que el primero es de los aspectos más importantes al momento de elegir una película. De cualquier forma creemos que lo mejor para el usuario es brindarle bastante información y que este sea quien tome la decisión final en cuanto a lo que desea ver. Para seleccionar “las mejores” nosotros utilizaremos la lista de 250 mejor puntuadas por la audiencia en IMDB ya que los miembros del equipo coinciden en gusto con muchas de las películas de esa lista y en general son películas muy populares (o lo fueron en el caso de los clásicos).

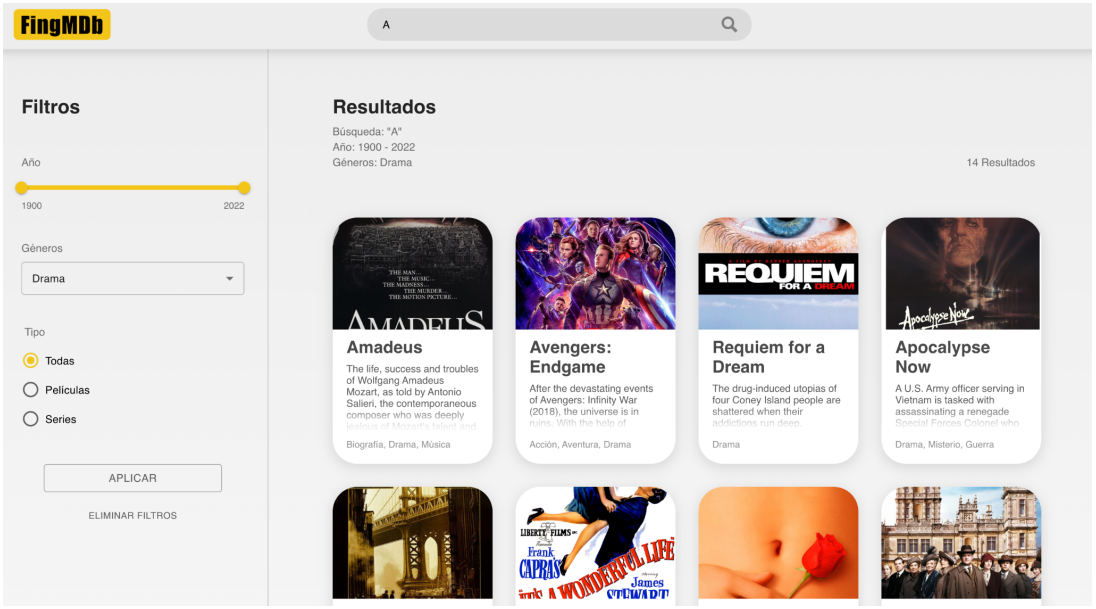
Flujo de la aplicación

La pantalla inicial de la página web es simple y minimalista, similar a la interfaz de Google que recibe al usuario con la barra de búsqueda. En esta, el usuario puede buscar un texto cualquiera que será enviado al backend de la aplicación para que esta busque en Elasticsearch. La aplicación en su totalidad tiene como objetivo ser funcional tanto en móvil como en escritorio por lo que se adaptó la misma para cumplir este requerimiento.



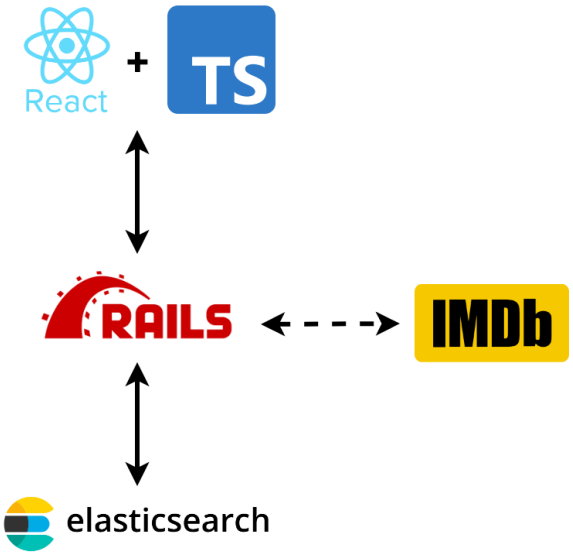
Como se puede observar también se cuenta con un carrousel que muestran las películas más populares de FingMDB con un diseño similar al de Netflix en cuanto a estética y funcionamiento del carrousel. Cada elemento del mismo consiste en una especie de tarjeta que muestra la imagen de la película y el inicio de la descripción del mismo. Al clickear una de estas tarjetas seremos redirigidos a la ficha de la película donde veremos toda la información detallada.

En cambio, si se realiza una búsqueda utilizando la barra de búsqueda, se redirige la página a una pantalla distinta donde se ordenan las películas de mayor a menor relevancia según el prompt de búsqueda.
A la izquierda de esta pantalla también se pueden utilizar filtros para seleccionar películas según el año y el género.
Al momento de devolver los resultados, la aplicación también muestra los filtros aplicados (si es que se aplicó alguno) y la cantidad de películas/series encontradas.



Arquitectura de la solución y herramientas a utilizar

En cuanto al diseño de la solución desde el punto de vista técnico utilizamos un frontend web en React + Typescript.
Para el backend se decidió utilizar Ruby on Rails para la creación de un API REST, ambos deployados en Heroku. Este será el intermediario en todas las consultas con Elasticsearch además de ser el responsable de hacer la recuperación de los datos de las películas y series a partir de la API de IMDB[1]. Este mismo componente se encargará de devolver los datos recuperados desde Elasticsearch.
El diagrama a continuación describe la arquitectura a utilizar en nuestra solución.



Parece relevante mencionar que la línea punteada entre IMDb y Rails tiene como objetivo ejemplificar que existirá una comunicación entre ellos para la recuperación de datos pero no necesariamente serán consultados en tiempo real. Su objetivo es el de conseguir una gran cantidad de datos de películas y series inicialmente y almacenarlas en algún índice de Elasticsearch.

Implementación

Películas de IMDb

Para la implementación de la solución se utilizó una api de IMDb[1],esta fue utilizada para poder acceder a la información, para poder utilizarse fue necesario registrarse y obtener una key que fue usada para poder acceder a los endpoints que brinda la api.
Un ejemplo de la forma que accedimos a los datos fue mediante:
https://imdb-api.com/es/API/Top250Movies/[key].

En nuestro caso utilizamos 3 endpoints del servicio REST, el anteriormente mencionado nos permitió obtener datos generales sobre el top 250 de películas, análogamente uno para series y por último un endpoint que nos permitió obtener información detallada por id de recurso (pelicula o serie).
En elasticsearch se almacenó en un único índice la información detallada de todas las películas y series.

A la hora de realizar esta acción nos encontramos con la siguiente problemática, el servicio de imdb utilizado contaba con un límite de solicitudes por día, este era de 100 request, por lo que fue necesario para poder cargar más de 100 películas y series, realizar la carga en días distintos.

Elasticsearch

La manera en que almacenamos los datos que traemos de la API es utilizando la base de datos que brinda Elasticsearch.
Esto se hizo ya que provee una forma fácil de almacenar grandes cantidades de datos, y es altamente eficiente realizando búsquedas en estos. En nuestra aplicación es de vital importancia ofrecer al usuario los resultados de forma rápida.
Elasticsearch también nos permitió realizar queries de búsquedas complejas, ponderando resultados por rating. Dentro de los objetivos iniciales estuvo la idea de implementar una ponderación por preferencias de género del usuario, pero debido a tiempos no se terminó implementando. Sin embargo cabe destacar que en Elasticsearch no tendría mayor complicación la implementación por preferencia de género, sino que el tiempo invertido mas bien en la implementación de perfiles de usuarios con preferencias asociadas.
Una vez que se cargan los datos en la base de Elasticsearch, están listos para ser utilizados por la web. A la hora de desplegar los datos, la plataforma web (front-end) debe consumir estos a partir de un endpoint de nuestro back-end.

Para realizar las búsquedas y filtrados por género en elasticsearch, se utilizó la siguiente query:

```
GET movies_test/_search
{
"query": {
      "bool": {
        "must": [
          {
            "match_phrase_prefix": {
              "title": {
                "query": "{{term}}",
                "slop":2
              }
            }
          },
          {
            //filtro opcional de género
            "match": {
              "genres": {
                "query": "{{genre}}",
                "operator": "and"
              }
            }
          }
        ],
        "should": [
          { "match_phrase_prefix":
            {
              "plot": "{{term}}"
            }
          },
          {
            "range": {
              "imDbRating": {
                "gte":"8",
                "boost": 2
              }
            }
          }
        ],
        //filtro opcional de año
        "filter":[
          {
             "range":  {
                 "year": {
                   "gte" : "{{min_year}}" ,
                    "lte":"{{max_year}}"
                 }
              }
          }]}}}
```

Procedemos a explicar brevemente la query mencionada. Lo que tenemos aquí es una query de tipo *bool* con cláusulas *must*, *should* y *filter*.

La cláusula *must* nos sirve para establecer propiedades que si o si se deben cumplir en los resultados que se obtengan, esto nos permite elegir el conjunto de datos inicial sobre el cual se van a aplicar las demás cláusulas (*should* y *filter*).

En este caso se dice que debe aparecer el término a buscar en el título de las películas/series resultado. La propiedad *slop* permite que no tenga que aparecer exactamente igual a como el usuario lo escribió sino que puede tener una distancia de hasta 2 palabras. Por ejemplo si tenemos la siguiente serie “How I meet your mother”, si el usuario busca "How your" (que en el título original ambas palabras tienen entre medio dos más), aparecerá este resultado. Cabe destacar que el slop también funciona para cambiar orden de palabras, por ejemplo si tenemos la siguiente película “The Godfather”, si el usuario busca God The, aparecerá este resultado gracias al *slop* de 2.

El segundo elemento *must*, lo que busca es aplicar un filtro por género (siempre y cuando el usuario decida filtrar por géneros). El operator *and* indica que si o si deben aparecer todos los géneros que eligió el usuario para filtrar.

En cuanto a las cláusulas *should*, estas nos permiten ordenar nuestros resultados, empujando más arriba aquellos que pueden ser más relevantes. En nuestro caso con la primera cláusula decidimos empujar hacia arriba si en la descripción aparece el término que busca el usuario. Sin embargo esto no es lo más relevante, sino que elegimos darle aún mayor relevancia (*boost 2*, esto multiplica el score del elemento por 2 si es que cumple con la cláusula) al rating de la película , si esta tiene un rating mayor a 8 estará más arriba en los resultados.

Finalmente tenemos los filters, los cuales nos permiten filtrar entre los resultados obtenidos hasta ahora, eliminando todos aquellos que no cumplan las condiciones, en este caso es utilizado para filtrar el año por rango, dejando así las películas que cumplen estar entre min_year y max_year.

Trabajo a futuro

A pesar de estar satisfechos con los resultados, creemos que existen mejoras que se podrían implementar a futuro para agregar aún más funcionalidades y mejorar la experiencia de usuario:

- Se podría agregar un registro de sesión para que se registren usuarios y de esa manera poder recomendarles películas/series de acuerdo a sus gustos y de una manera más personalizada.
- Se podrían buscar las películas por actor/director en vez de solamente por el título de la película.
- Se podría agregar más cantidad de películas y/o series.
- Sería positivo crear un proceso período que actualice una vez al mes de manera asincrónica la base de datos con información de nuevas películas y/o series.
- Sería bueno agregar datos sobre los actores/directores de las películas/series.

- Una problemática que enfrentamos fue que no se almacenó la query dentro de Elasticsearch como template, sino que es construida por el backend dinámicamente según los filtros opcionales que recibe desde el front end y luego enviada. Un punto de mejora sería investigar si es posible almacenar la query en Elasticsearch como template y que algunos filtros se omitan según que parametros se le envían. Esto sería un punto de mejora para desacoplar el backend más aún de Elasticsearch, ya que de esta forma simplemente se enviaría el nombre del template y los valores de los distintos parámetros.

Conclusiones

El trabajo nos pareció interesante y muy enriquecedor, nos permitió trabajar con una de las herramientas de búsquedas más conocidas y utilizada por grandes empresas como Netflix, Uber, Microsoft y Mercado Libre. A su vez, nos permitió ver en funcionamiento una herramienta que aplica por detrás todos los conceptos comentados en el curso.
El mayor desafío en nuestra opinión, se dio al momento de decidir cuándo y cómo se iba almacenar la información en Elasticsearch. Algunos otros desafíos que tenían cierto riesgo ya que nunca habíamos trabajado con Elasticsearch, fue la integración con este y el crear las querys para que nos devuelva la información necesaria tal cuál queríamos.

Por otro lado,creemos que el sitio sería de gran interés para los usuarios finales, ya que les permite obtener información sobre las películas y series más populares del momento de una manera rápida y sencilla. Creemos que la interfaz de usuario es intuitiva y amigable a la vista por lo que los usuarios la pueden preferir antes que otros sitios.

Referencias

1. API IMDb - https://imdb-api.com/api
2. ElasticSearch - https://www.elastic.co/elasticsearch/
3. Heroku - https://www.heroku.com/
4. Ruby on Rails - https://rubyonrails.org/
5. React - https://es.reactjs.org/