26/11/23

Webir Spotify Lookup

Informe Final – Grupo 7

Integrantes:
Tomas Rodríguez
Nicolas Inzua
William Tift
Federico Correa
Stefano Graziani

Docente:
Libertad Tansini

1. Problema abordado

El problema abordado se centra en mejorar la experiencia de búsqueda de canciones de un usuario de Spotify a través de una aplicación fullstack que amplíe las funcionalidades básicas que provee la plataforma musical. Como será explicado posteriormente, la API para desarrolladores de Spotify brinda una base ideal para acceder a sus funcionalidades básicas y construir una solución a partir de ellas. Con el objetivo de completar la experiencia musical, hemos integrado también un buscador de letras de canciones, la cual es una funcionalidad que Spotify no posee nativamente. Esta funcionalidad se agrega como forma de poner a disposición del usuario toda la información que pueda interesarle buscar, desde descubrir nueva música hasta entender cada palabra de sus letras favoritas.

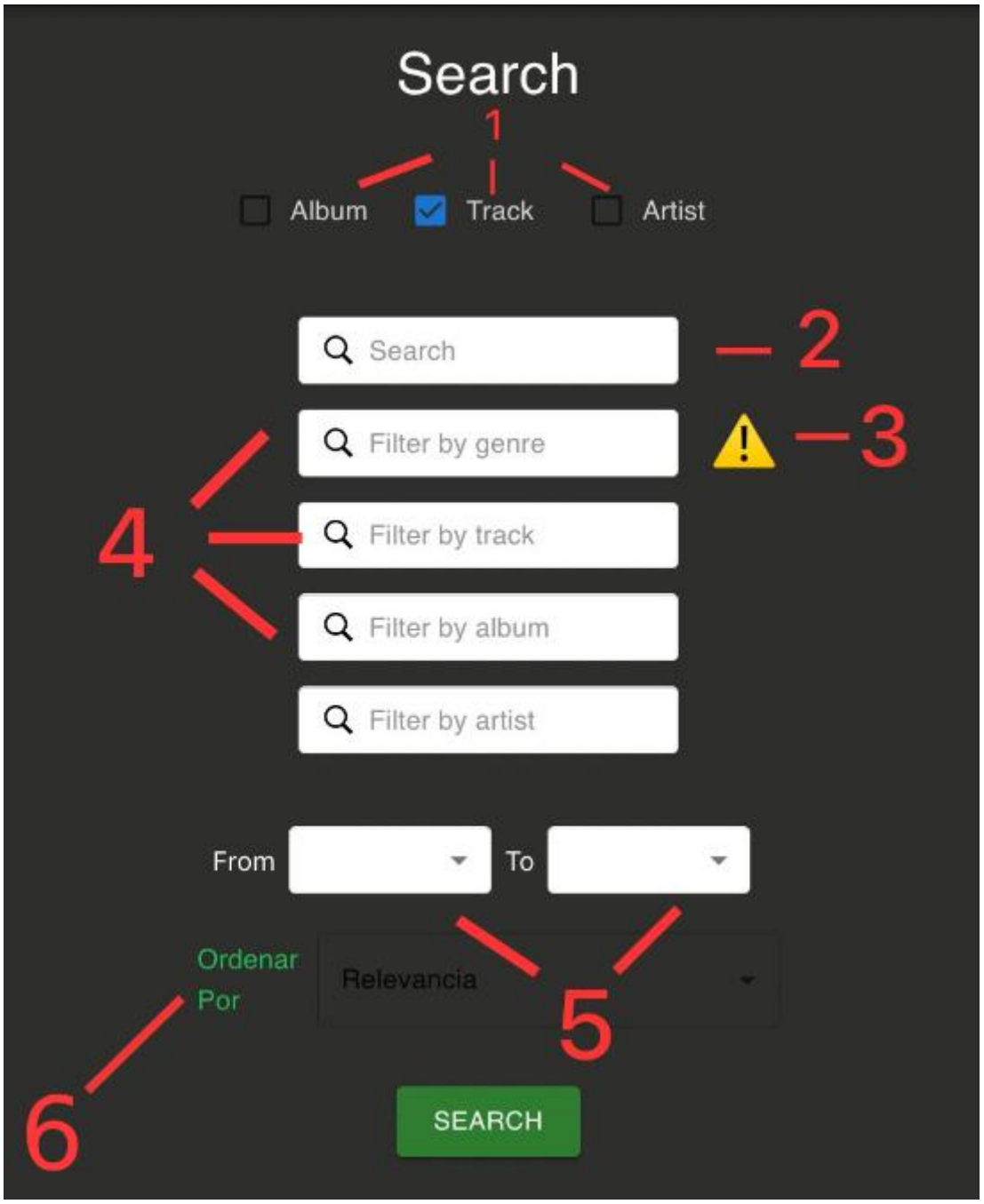
2. Componentes del sistema

Frontend

Buscador principal (Spotify)

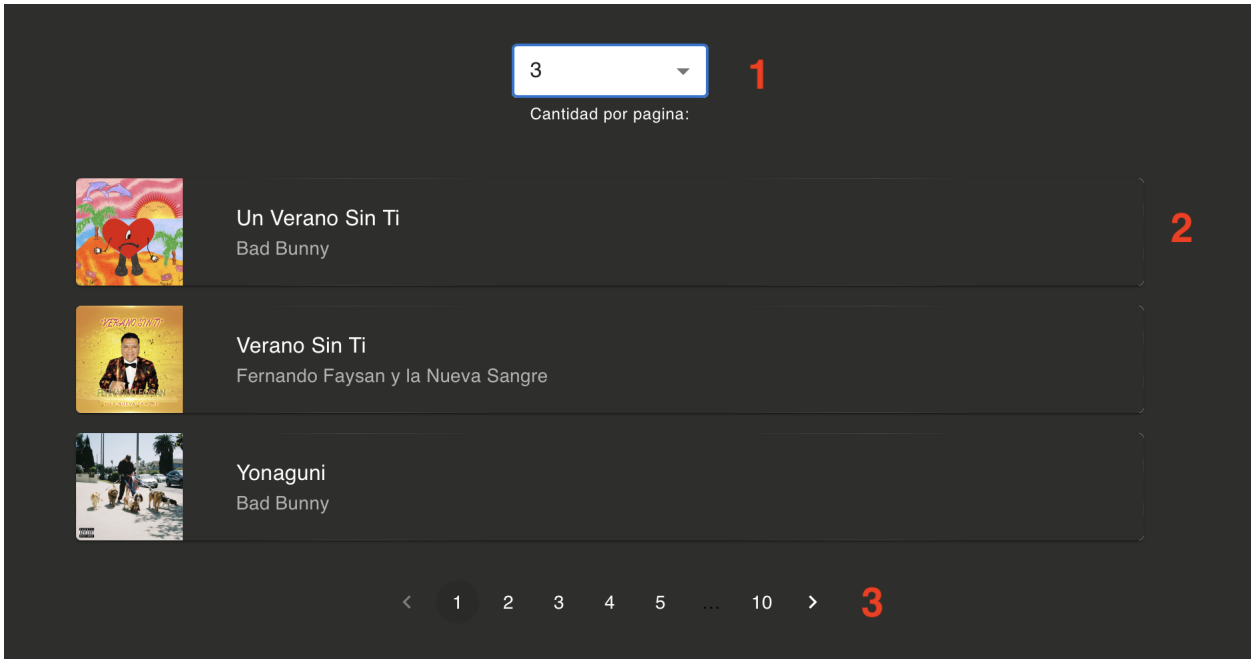
Este componente se encarga de procesar la búsqueda del usuario para Álbumes, Artistas o Canciones únicamente.

Pantalla de búsqueda:



1. Selector de tipos: Nuestra aplicación contará con un selector que permite a los usuarios especificar qué tipo de resultados están buscando en Spotify. Es importante destacar que al menos un tipo de resultado debe ser seleccionado para habilitar la función de búsqueda. Esto asegura que la búsqueda sea relevante para las necesidades del usuario.
2. Buscador principal: Aquí es donde los usuarios ingresarán su consulta de búsqueda. Este buscador principal es la interfaz principal para interactuar con la API de Spotify y recuperar los resultados deseados.
3. Indicador de filtro no disponible: Si un usuario realiza una selección que no es válida o si falta información en su búsqueda, nuestro sistema proporcionará un indicador en forma de tooltip que explicará por qué ciertos filtros o inputs están desactivados. Esto ayudará a los usuarios a comprender cómo mejorar su búsqueda.
4. Filtros disponibles: Para refinar aún más los resultados de búsqueda, ofreceremos varios filtros que los usuarios pueden aplicar. Estos filtros pueden incluir, por ejemplo, género musical, artista, canción, letra, etc.
5. Filtro por rango de años: Uno de los filtros disponibles será la capacidad de restringir los resultados a un rango de años específico. Esto permitirá a los usuarios encontrar canciones lanzadas en un período particular.
6. Selector de ordenamiento: Brinda al usuario la capacidad de ordenar los resultados a desplegar. Los valores posibles son “Relevancia” (implementado por el algoritmo de Spotify), “Reciente” (implementado acorde a la metadata recibida sobre los resultados de la consulta) y “Alfabético”

Pantalla de resultados:

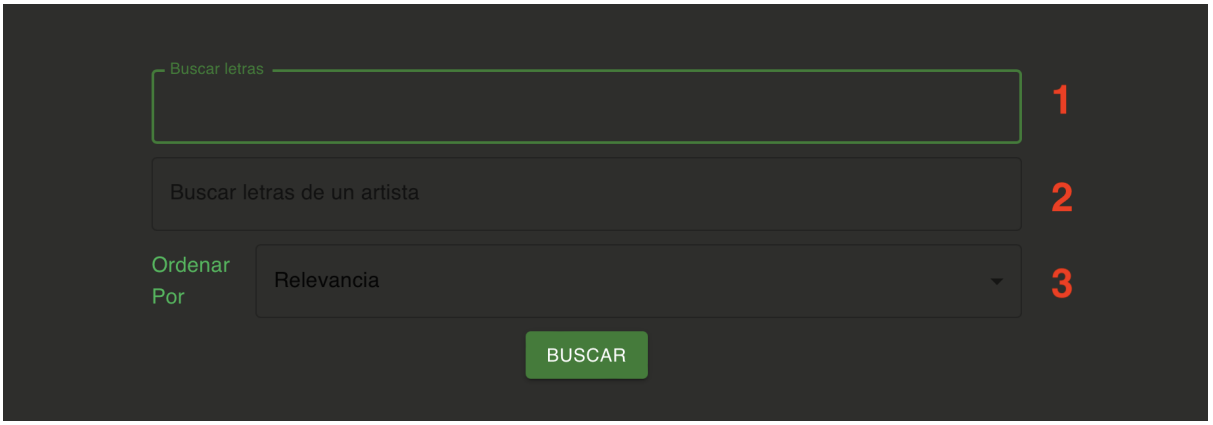


1. Selector de tamaño de página: Aquí es donde el usuario puede decidir el tamaño de la página en la cual se muestran los resultados. Para evitar sobrecargar a los usuarios con una gran cantidad de resultados, implementaremos un indicador que muestra cuántos resultados se están mostrando actualmente y si hay un límite en la cantidad de resultados que se pueden recuperar en una sola búsqueda.
2. Item de resultado: Aquí se puede visualizar un ítem de resultado obtenido, con información básica como la foto, título y artista de la canción.
3. Selector de índice de página: Aquí es donde el usuario puede seleccionar en qué página desea ubicarse. Para manejar listas largas de resultados, implementaremos un sistema de paginación que permitirá a los usuarios navegar a través de múltiples páginas de resultados.

Buscador secundario (Letras)

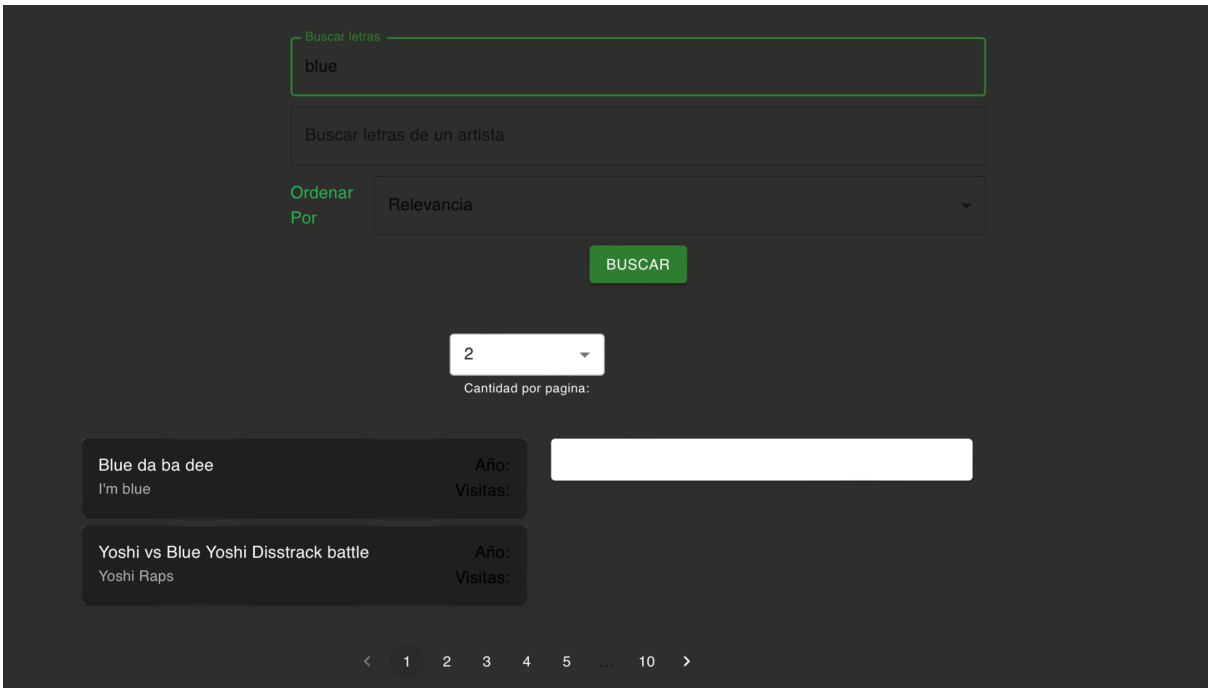
Este componente se encarga de procesar la búsqueda del usuario según letra de la canción.

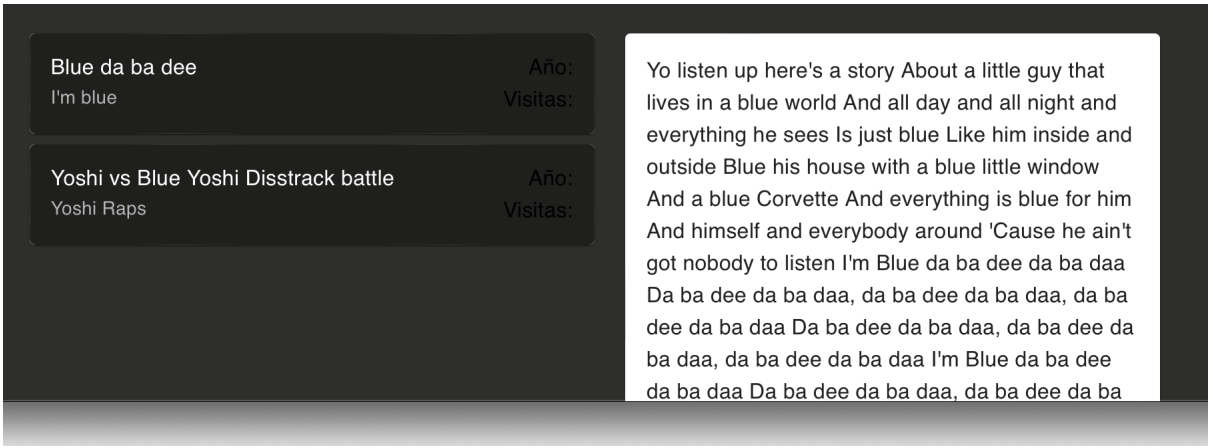
Pantalla de búsqueda:



1. Buscador de letra: Aquí es donde los usuarios ingresarán su consulta de búsqueda
2. Filtro de artista: Aquí es donde opcionalmente el usuario puede especificar el artista de la canción para la cual está buscando la letra.
3. Selector de ordenamiento: Aquí es donde el usuario puede seleccionar el ordenamiento que quiere que sigan los resultados de la página, las opciones disponibles son: relevancia, popularidad, reciente.

Pantalla de resultados:





Como se puede ver se reutilizan varios componentes de la UI de resultados de la pantalla de búsqueda principal. Esta pantalla de resultados cuenta con paginación y los ítems obtenidos en los resultados pueden ser presionados para consultar la letra de los mismos.

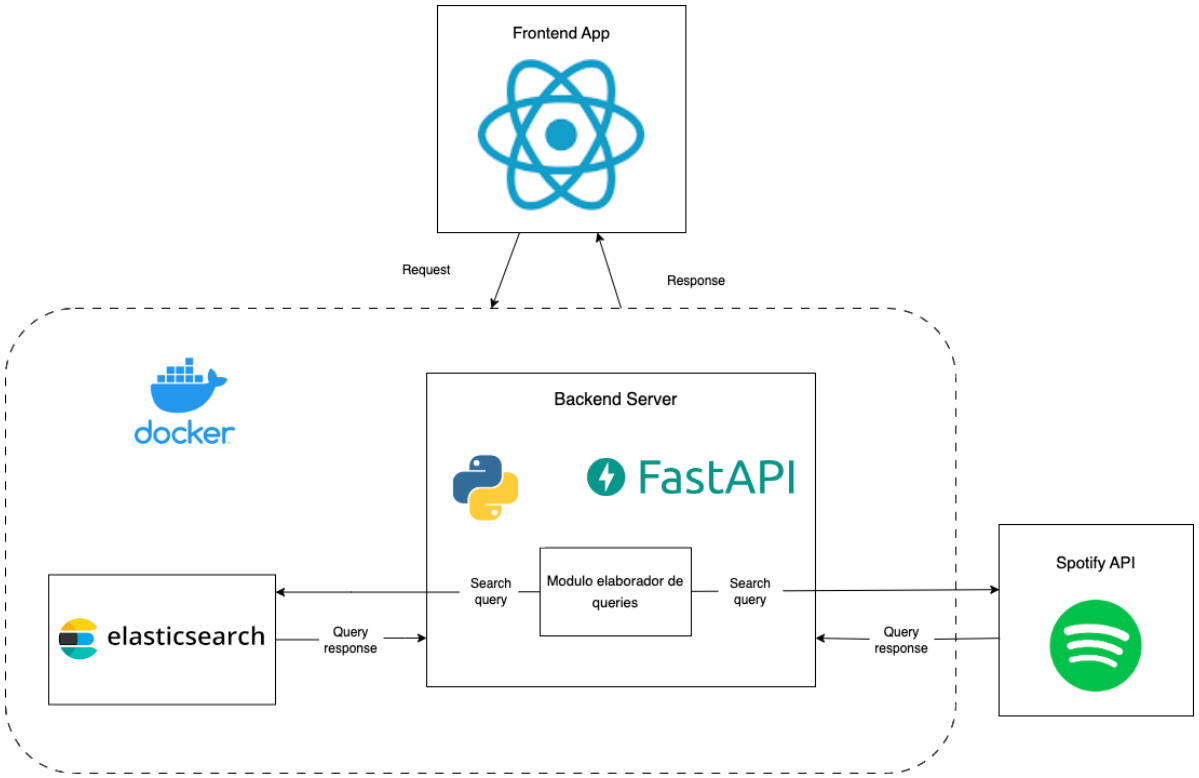
Backend

1) React
    - Utilizaremos el framework de React para desarrollar el frontend de nuestra aplicación, lo que nos permitirá crear una interfaz de usuario interactiva y receptiva.
2) FastAPI (Framework Web para Python)
    - En el backend, utilizaremos FastAPI, un framework web de Python, para gestionar las solicitudes y respuestas de la API, así como para la lógica de negocio que implica procesar las consultas de búsqueda de los usuarios.
3) Módulo elaborador de Queries
    - Implementaremos un módulo en nuestro backend que se encargará de elaborar las consultas adecuadas tanto para la API de Spotify como para ElasticSearch en función de los parámetros proporcionados por los usuarios
4) Proveedores de datos
    5) ElasticSearch[1]
        - Se utilizará para almacenar de forma accesible exclusivamente información sobre letras de canciones. En este se indexará un dataset que contiene datos sobre letras de canciones[2], los datos que contiene este mismo no están relacionado con los resultados obtenidos de Spotify.
    6) Spotify API[3]
        - Utilizaremos “Spotipy”[4], un framework de Python diseñado específicamente para trabajar con la API de Spotify. Spotipy simplificará la comunicación con la API de Spotify, lo que nos permitirá obtener datos precisos y relevantes para mostrar a los usuarios.

---

[1] https://elasticsearch-py.readthedocs.io/en/v8.10.0/
[2] Los datos serán provenientes de datasets encontrados en la plataforma Kaggle. Por ejemplo https://www.kaggle.com/datasets/carlosgdcj/genius-song-lyrics-with-language-information
[3] https://developer.spotify.com/documentation/web-api
[4] https://spotipy.readthedocs.io/

Diagrama de componentes



3. Resultados

Encontraremos que los resultados obtenidos están alineados con nuestras expectativas. Sin embargo, tuvimos que resolver ciertos problemas que surgieron a la hora de diseñar e implementar la aplicación. Estos desafíos incluyen:

Arquitectura:
Diseñar la arquitectura para la indexación y consulta en Elasticsearch presentó el desafío de encontrar la combinación adecuada de módulos y contenedores con servicios. Nos surgio la duda si era necesario o no integrar a su vez una base de datos distinta a la provista por elasticsearch para almacenar los resultados de spotify, pero decidimos no incorporar este componente, ya que por el momento la API de spotify no nos presentaba dificultades en cuanto la cantidad de consultas que se le pueden hacer para extraer los datos.

Elección del Dataset Auxiliar:
La elección del dataset de letras planteó el problema de cuántos datos queríamos manejar, ya que era bastante extenso (12GB). Dado el tamaño del proyecto, la carga y lectura de datos podría ser lenta e ineficiente. En consecuencia, decidimos limitar la cantidad de datos extraídos.

Indexación en Elasticsearch:
Definir las columnas del dataset a indexar implicó la tarea de seleccionar la información que resultará relevante para el usuario final. La clave estuvo en elegir un conjunto mínimo pero que nos de la información suficiente para lo que queríamos hacer. Como información adicional de la indexación, podemos mencionar su velocidad:

Entradas a indexar	Tiempo tomado
1000	12s
10000	1m 1s
20000	1m 42s
30000	2m 12s

Como podemos ver, el tiempo de indexación no resulta lineal. Ya que el dataset utilizado tiene más de medio millón de letras, podemos concluir que el tiempo de indexación sería considerable pero probablemente no desmesurado (se estima que menor a una hora).

Elaboración de Queries para Elasticsearch:
Las consultas son estáticas en cuanto al input del usuario. Se puntúa el doble un match en la lirica que en el nombre del artista. Este es un ejemplo de una consulta básica sin cambios en el ordenamiento

```
"query": {
  "bool": {
```

```
    "must": [
      {"match": {"lyrics": {"query": lyric, "boost": 2.0}}},
      {"match": {"artist": artist}}
    ]
  }
}
```

Elaboración de Queries para Spotify:
Desarrollar las consultas fue un desafío tedioso, ya que las consultas que podemos hacer a través de la api de spotify tiene filtros condicionales, por ejemplo “Los filtros de tag:new y tag:hipster solo se pueden utilizar al buscar album”, esto llevó a tener que implementar controles sobre qué tipo de filtro habilitamos para limitar desde el front end los campos de filtrado y para que el backend pueda realizar las consultas de forma correcta y que no ocurran casos donde el usuario usaba un filtro que no sirve para el tipo de datos que está buscando. Este es un ejemplo de una consulta donde se buscan resultados de albumes o artistas creados entre los años 2010-2020, en el genero de rock y con un nombre de artista y album especifico

`“year:2010-2020+genre:rock+artist:Radiohead+album:In+Rainbows&type=artist,album”`

4. Conclusiones

Se logró crear una aplicación que cumple con la idea propuesta al inicio del proyecto, y se considera que el problema fue abordado correctamente.

El producto final a la vez es altamente flexible y se le ve mucho potencial para poder seguir elaborando sobre él a modo de seguir incorporando aprendizajes. Además, nos acercó a conocer más de cerca los contenidos impartidos en el curso y a diversas. tecnologías con las que nunca se había tenido experiencia

También fue muy interesante entender desde la perspectiva de un diseñador como proveer al usuario con diversas herramientas y funcionalidades de búsqueda puede resultar abrumador para la experiencia del mismo y también complejo de implementar (dada la gran interdependencia de algunas de las variables y su posterior traducción por parte del módulo elaborador de queries).

Se puede destacar también que el manejo de grandes volúmenes de información estructurada con la ayuda de ElasticSearch fue muy sencillo, y las facilidades que brinda para indexar y consultar la información también agregan un gran atractivo. Se considera que la exploración que se hizo de las funcionalidades de ElasticSearch en este trabajo fueron bastante menores en comparación con todas las que provee y que posee un gran potencial.

5. Trabajo Futuro

Algunas posibles mejoras para tener en cuenta a futuro podrían ser:

Complejizar la elaboración de consultas a ElasticSearch:
Motivación: Mejorar la precisión y la relevancia de los resultados de búsqueda para ofrecer una mejor experiencia de usuario. Implementar consultas más complejas y capacidades de filtrado en Elasticsearch para permitir buscar letras de canciones con mayor especificidad, atendiendo a criterios más personalizados.
Desafíos: El principal reto aquí radica en diseñar consultas que sean tanto eficientes como efectivas, teniendo en cuenta las limitaciones y la información del dataset. Además, se deberá asegurar que el sistema sea intuitivo para el usuario final, evitando sobrecargar la interfaz con opciones complicadas.

Uso de un dataset en español:
Motivación: La inclusión de un dataset en español ampliaría significativamente el alcance de la aplicación, permitiendo a un público más amplio buscar y disfrutar de las letras de canciones en este idioma.
Desafíos: Elasticsearch no soporta nativamente las stop words en español, lo que podría afectar la calidad de los resultados de búsqueda. Sería necesario utilizar algún plugin para gestionar las stop words en español.

Escalado de la aplicación con un dataset más grande:
Motivación: Escalar la aplicación para manejar un dataset más grande permitiría ofrecer una biblioteca de letras de canciones más extensa y diversa, mejorando así la experiencia del usuario y la utilidad de la aplicación.
Desafíos: El principal desafío aquí es mantener o mejorar el rendimiento y la estabilidad de la aplicación a medida que aumenta el tamaño del dataset. Esto podría requerir optimizaciones nuevas, mejoras en el diseño y posiblemente el uso de técnicas de indexación y caché más avanzadas

Expansión y mejora de la UI para resultados de Spotify:
Motivación: Proporcionar una presentación más intuitiva y detallada de canciones, artistas y álbumes mejorará la interacción del usuario.
Desafíos: Desarrollar una interfaz que integre eficazmente distintos tipos de resultados brindados por Spotify (artistas, canciones o álbumes) junto con su extensa metadata, sin sobrecargar visualmente al usuario, manteniendo la facilidad de uso y la estética agradable.

Integración con más fuentes de información:
Motivación: Complementar los resultados de búsqueda con información adicional, por ejemplo, biografías de artistas, fotos, y más datos relevantes para ofrecer una experiencia más rica y completa al usuario.
Desafíos: Identificar y seleccionar fuentes de terceros confiables y relevantes para integrarlas con la aplicación. Esto implica manejar diversas APIs o datasets, asegurando la compatibilidad y la actualización constante de la información.