Bases de Datos Avanzada: Actividad 4

Profesor: Manuel Alba
Alumno: Samuel Angulo
Ayudante: César Muñoz

Mayo de 2024

Índice

1. Introduction

Esta actividad consta de dos partes, la primera se basa en familiarizarse con la plataforma de Elasticsearch, manejar grandes volúmenes de datos en ella y hacer consultas sobre los mismos.

2. Desarrollo

2.1. Obtención de los datos y normalización

Se debe descargar a partir de un enlace provisto por el profesor un archivo que contiene 2GB en letras de canciones junto a datos asociados como los autores, el título de la canción y la URL de donde se obtuvo dicha letra.

Una vez descargados, por comodidad del formato se transforman los CSV en JSON y se inserta el prefijo "music-xyz.ª cada archivo transformado, donde "xyz.ᵉˢ el respectivo nombre del archivo anterior en formato CSV. Por ejemplo, "bachata.csv"pasa a ser "music-bachata.json".

Dicho script para tranformar y normalizar a formato JSON es el siguiente:

```
const csvFolder = './songs';
const outputFolder = './json';

async function csvToJsonStream(csvFilePath, outputFolder) {
    return new Promise((resolve, reject) => {
        const results = [];
        const fileName = path.basename(csvFilePath, '.csv');
        const jsonFilePath = path.join(outputFolder, `music-${fileName}.json`);

        if (!fs.existsSync(outputFolder)) {
            fs.mkdirSync(outputFolder);
        }

        fs.createReadStream(csvFilePath)
            .pipe(csv({ separator: '|', trim: true }))
            .on('data', (data) => results.push(data))
            .on('end', () => {
                fs.writeFileSync(jsonFilePath, JSON.stringify(results, null, 4));
                resolve();
```

```
            })
            .on('error', (error) => reject(error));
    });
}
```

2.2. Subida de gran volumen

Una vez los datos han sido manipulados localmente llega el momento de poblar el sistema de Elasticsearch y sus índices con cada uno de los archivos. Con el requisito que cada uno de los archivos debe ser un índice en Elasticsearch, el requisito de tener el prefijo "music-ïba en este apartado y se decidió solucionar en la normalización previamente trabajada y explicada.

En primera instancia se tuvieron que crear credenciales para poder utilizar la API de Elasticsearch, la cuales fueron almacenadas en un archivo *.env* dentro del espacio de trabajo y vuyo modelo es el siguiente:

```
ELASTIC_ENDPOINT="aqui va el endpoint otorgado por elastic"
API_KEY="aqui va la API KEY que se genera para el entorno donde se crearán
y llenarán los indices"
```

Y así, con dichas credenciales se podrá iniciar la instancia necesaria para poder manipular los índices mediante la API. A continuación la forma en la que se instancia:

```
import { Client } from "@elastic/elasticsearch";

export const client = new Client({
    node: process.env.ELASTIC_ENDPOINT,
    auth: {
        apiKey: process.env.API_KEY
    }
});
```

Lo anterior está encapsulado en un archivo llamado .[el]astic.js"para facilitar su uso en archivos ajenos.

Para la subida de los archivos JSON a la plataforma se investigó los métodos existentes. Y así, cayendo en las siguientes dos formas:

La primera forma, demostrada a continuación, puesto que tiene métodos válidos y disponibles en la API de Elasticsearch no contaba con la capacidad de poder llenar a partir de mucha información de forma recurrente cada índice, porque lanzaba error de memoria. Esta es mas bien, una implementación para casos más acotados.

```
jsonFileNames.forEach(async element => {
    await client.indices.create({
        index: 'music-'+element,
        body: { }
        }
    );
});
```

Sin embargo, investigando se encontró un método más ótpimo para poder manejar grandes volúmenes de datos y poder llenar los índices con ello.

A continuación, la implementación ilustrada en la documentación de Elasticsearch:

```
const dataset = [
  {"name": "Snow Crash", "author": "Neal Stephenson",
  "release_date": "1992-06-01", "page_count": 470},
  {"name": "Revelation Space", "author": "Alastair Reynolds",
  "release_date": "2000-03-15", "page_count": 585},
  {"name": "1984", "author": "George Orwell",
  "release_date": "1985-06-01", "page_count": 328},
  {"name": "Fahrenheit 451", "author": "Ray Bradbury",
  "release_date": "1953-10-15", "page_count": 227},
  {"name": "Brave New World", "author": "Aldous Huxley",
  "release_date": "1932-06-01", "page_count": 268},
  {"name": "The Handmaid's Tale", "author": "Margaret Atwood",
  "release_date": "1985-06-01", "page_count": 311},
];

// Index with the bulk helper
const result = await client.helpers.bulk({
  datasource: dataset,
  onDocument: (doc) => ({ index: { _index: 'index_name' }}),
});
```

Y dado a la característica de esta actividad, se debe poder insertar muchos archivos en muchos índices, por lo que la implementación propia queda de la siguiente manera:

```
import { client } from "./elastic.js";
import fs from 'fs';
import path from 'path';
import csv from 'csv-parser';

const csvFolder = './songs';
const outputFolder = './json';


function getJsonFileNamesWithPrefix(jsonFolder) {
    try {
        const fileNames = fs.readdirSync(jsonFolder);
        const jsonFileNames = fileNames
            .filter(fileName => fileName.endsWith('.json'))
            .map(fileName => path.basename(fileName, '.json'));
        return jsonFileNames;
    } catch (error) {
        console.error('Error al obtener los nombres de los archivos JSON:', error.mess
        return [];
    }
}

const getSongData = async (jsonFolder, fileName) => {
    try {
        const jsonFilePath = path.join(jsonFolder, `${fileName}.json`);
        const jsonArray = JSON.parse(fs.readFileSync(jsonFilePath, 'utf8'));
        return jsonArray;
    } catch (error) {
        console.error('Error al leer el archivo JSON:', error.message);
        return [];
    }
}

fs.readdir(csvFolder, async (err, files) => {
    if (err) {
        console.error('Error al leer la carpeta:', err);
        return;
    }

    const csvFiles = files.filter(file => file.endsWith('.csv'));
```

```
    for (const csvFile of csvFiles) {
        const csvFilePath = path.join(csvFolder, csvFile);
        await csvToJsonStream(csvFilePath, outputFolder);
    }

    const jsonFiles = getJsonFileNamesWithPrefix(outputFolder);

    for (const element of jsonFiles) {
        const songData = await getSongData(outputFolder, element);
        await client.helpers.bulk({
            datasource: songData,
            pipeline: 'ent-search-generic-ingestion',
            onDocument: (doc) => ({index: {_index: element}})
        });
        console.log(`Índice '${element}' LLENADO con éxito.`);
    }
});
```

Como se puede observar, se sigue utilizando los valores *csvFolder* y *outputFolder*, las cuales son la carpeta decargada y donde se guardaron los archivos noramlizados, respectivamente.

Así mismo, se hace uso de dos funciones auxiliares. La primera es *getJsonFileNamesWithPrefix* para poder obtener los nombres de los archivos y crear los índices con esos mismos, y la segunda es *getSongData* cuyo propósito es recuperar el contenido de cada uno de los archivos para después ser urilizado en el cuerpo del índice.

De esta forma, se crean y llenan exitosamente todos índices para poder seguir a la siguiente fase.

2.3. Búsqueda solicitada

Esta etapa, fue más fácil de ejecutar por la explicación dada por el profesor en anteriores clases. Dirigiéndose al módulo *Dev Tools* de Elasticsearch, se es capaz de hacer peticiones de manera sencilla.

Esta parte de la actividad consta de buscar en toda la colección de índices las canciones cuyas letras tengan "la ciudad de la furia.[en] ellas. Se infiere, dado el contexto de la aplicación que la búsqueda debe ser precisa, por lo que se debe considerar que sea tal cual esa frase la que se encuentre dentro de las colecciones y se soluciona mediante la palabra reservada *match_phrase*.

Dicha petición es realizada mediante la siguiente *query*:

```
GET /*/_search
{
  "query": {
    "match_phrase": {
      "Lyric": "ciudad de la furia"
    }
  }
}
```

La petición lanza el siguiente resultado:


```
{
  "took": 79,
  "timed_out": false,
  "_shards": {
    "total": 269,
    "successful": 269,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 19,
      "relation": "eq"
    },
    "max_score": 28.456871,
    "hits": [
      {
        "_index": "music-rock-alternativo",
        "_id": "81zYm48BCxzS6OMuCNba",
        "_score": 28.456871,
        "_ignored": [
          "Lyric.keyword"
        ],
        "_source": {
          "Artist": "Aterciopelados",
          "Lyric": "    Me verás volar  Por la ciudad de la furia
          [...] Un hombre alado prefiere esta noche   ",
          "Title": "En La Ciudad de La Furia",
```

```
      "Url": "https://www.letras.mus.br/aterciopelados
      /en-la-ciudad-de-la-furia/en-la-ciudad-de-la-furia-print.html"
    }
  },
  {
    "_index": "music-rock",
    "_id": "hl3Ym48BCxzS6OMut33s",
    "_score": 26.453403,
    "_ignored": [
      "Lyric.keyword"
    ],
    "_source": {
      "Artist": "Soda Stereo",
      "Lyric": "    Me verás volar por la ciudad de la furia
      [...] Me verás volver Me verás volver A la ciudad de la furia",
      "Title": "En La Ciudad de La Furia",
      "Url": "https://www.letras.mus.br/soda-stereo
      /37255/en-la-ciudad-de-la-furia-print.html"
    }
  },
  {
    "_index": "music-rock",
    "_id": "vV3Ym48BCxzS6OMut33s",
    "_score": 26.453403,
    "_ignored": [
      "Lyric.keyword"
    ],
    "_source": {
      "Artist": "Soda Stereo",
      "Lyric": "    Me verás volar por la ciudad de la furia
      [...] A la ciudad de la furia   ",
      "Title": "En La Ciudad de La Furia",
      "Url": "https://www.letras.mus.br/soda-stereo/37255
      /en-la-ciudad-de-la-furia-print.html"
    }
  },
  {
    "_index": "music-rock",
    "_id": "lF3Ym48BCxzS6OMut33s",
    "_score": 25.892635,
    "_ignored": [
      "Lyric.keyword"
    ],
```

```
    "_source": {
      "Artist": "Soda Stereo",
      "Lyric": "Me verás volar Por la ciudad de la furia
      [...] Entre la niebla Un hombre alado Extraña la noche   ",
      "Title": "La Ciudad de La Furia",
      "Url": "https://www.letras.mus.br/soda-stereo
      /la-ciudad-de-la-furia/la-ciudad-de-la-furia-print.html"
    }
  },
  {
    "_index": "music-rock",
    "_id": "yV3Ym48BCxzS6OMut33s",
    "_score": 25.892635,
    "_ignored": [
      "Lyric.keyword"
    ],
    "_source": {
      "Artist": "Soda Stereo",
      "Lyric": "Me verás volar Por la ciudad de la furia
      [...] Un hombre alado Extraña la noche   ",
      "Title": "La Ciudad de La Furia",
      "Url": "https://www.letras.mus.br/soda-stereo
      /la-ciudad-de-la-furia/la-ciudad-de-la-furia-print.html"
    }
  },
  {
    "_index": "music-disco",
    "_id": "hVTPm48BCxzS6OMuVmPO",
    "_score": 25.603989,
    "_ignored": [
      "Lyric.keyword"
    ],
    "_source": {
      "Artist": "Gustavo Cerati",
      "Lyric": "Me verás volar Por la ciudad de la furia
      [...] Un hombre alado Prefiere la noche   ",
      "Title": "En la Ciudad de la Furia",
      "Url": "https://www.letras.mus.br/gustavo-cerati
      /1280439/en-la-ciudad-de-la-furia-print.html"
    }
  },
  {
    "_index": "music-hip-hop-rap",
```

```
    "_id": "UEbSm48BXpgaVw2eJsZA",
    "_score": 25.155663,
    "_ignored": [
      "Lyric.keyword"
    ],
    "_source": {
      "Artist": "C.R.O ",
      "Lyric": "Yeah Ah, Orodembow Ah, baby
      [...] A la ciudad de la furia es donde voy
      Al menos sé que aún te tengo Cuervos",
      "Title": "Cuervos",
      "Url": "https://www.letras.mus.br/cro-bardero
      /cuervos/cuervos-print.html"
    }
  },
  {
    "_index": "music-poprock",
    "_id": "p1rVm48BCxzS6OMu9NEQ",
    "_score": 24.752705,
    "_ignored": [
      "Lyric.keyword"
    ],
    "_source": {
      "Artist": "Soda Stereo",
      "Lyric": "Me verás volar por la ciudad de la furia
      [...] Me verás volver Me verás volver
      A la ciudad de la furia   ",
      "Title": "En La Ciudad de La Furia",
      "Url": "https://www.letras.mus.br/soda-stereo
      /37255/en-la-ciudad-de-la-furia-print.html"
    }
  },
  {
    "_index": "music-poprock",
    "_id": "3lrVm48BCxzS6OMu9NEQ",
    "_score": 24.752705,
    "_ignored": [
      "Lyric.keyword"
    ],
    "_source": {
      "Artist": "Soda Stereo",
      "Lyric": "Me verás volar por la ciudad de la furia
      [...] Me verás volver Me verás volver A la ciudad de la furia",
```

```
          "Title": "En La Ciudad de La Furia",
          "Url": "https://www.letras.mus.br/soda-stereo
          /37255/en-la-ciudad-de-la-furia-print.html"
        }
      },
      {
        "_index": "music-pop",
        "_id": "iUjVm48BXpgaVw2ebBcp",
        "_score": 24.734192,
        "_ignored": [
          "Lyric.keyword"
        ],
        "_source": {
          "Artist": "Shakira",
          "Lyric": "Me verás volar Por la ciudad de la furia
          [...] Un hombre alado Prefiere la noche   ",
          "Title": "La Ciudad de La Furia",
          "Url": "https://www.letras.mus.br/shakira
          /1280354/la-ciudad-de-la-furia-print.html"
        }
      }
    ]
  }
}
```


Para efectos de la integridad del documento se ha resumido con [...] las letras y destacado las partes donde aparece lo que se está buscando.

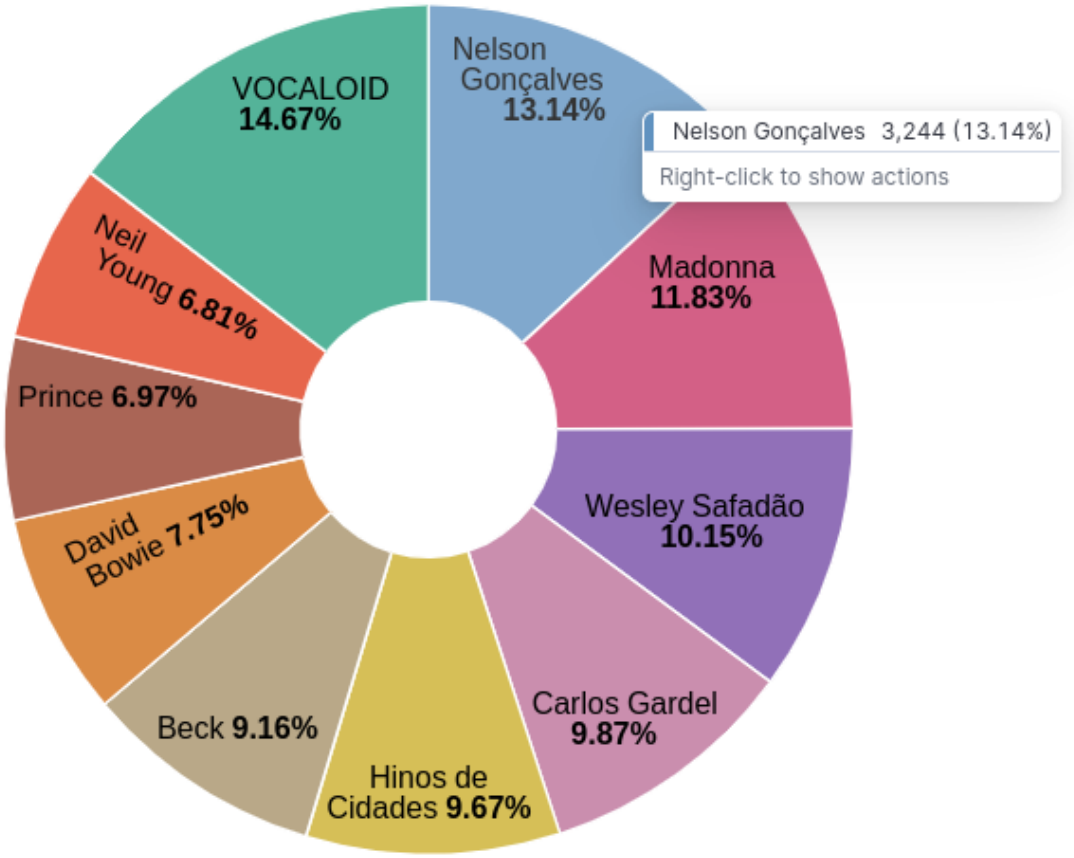
2.4. Creación de graficos

Una vez realizada la búsqueda se continúa con la siguiente fase de la actividad, la cual se basa en generar gráficos a partir de los datos almacenados. Aqui es cuando se encuentra la utilidad a la normalización de los nombres de los índices a "music-xyz", puesto que elastic necesita que se eligan los índices con los que se quiere trabajar con los gráficos y se pueden seleccionar todos los índices previamente subidos estableciendo como patron "music-*", donde * significa que utlizaremos todos los índices que hagan *match* con ese patrón, es decir, todos los que nos interesan.

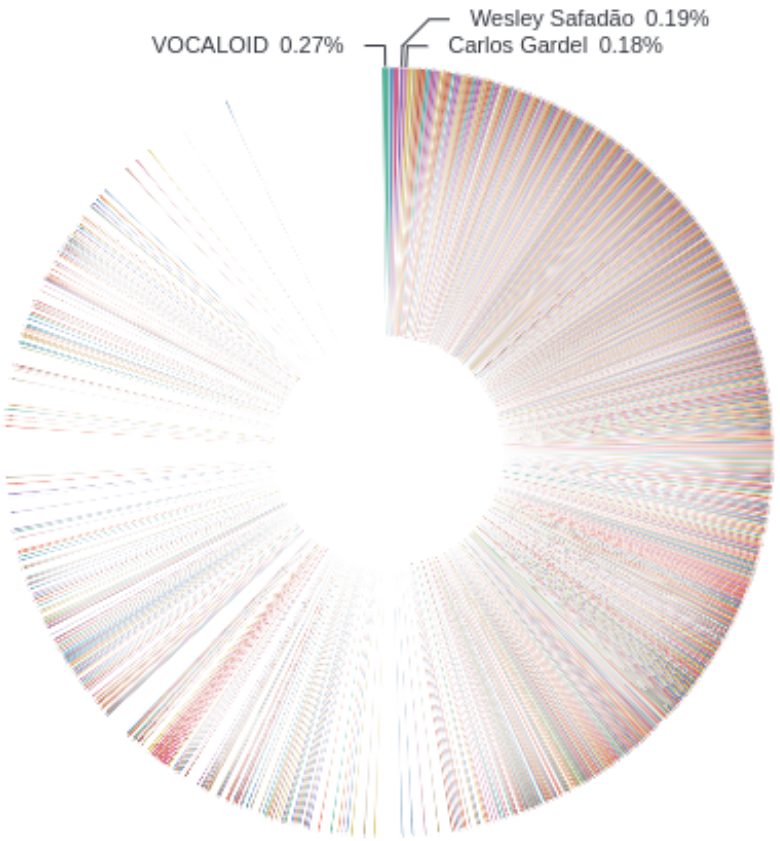
A continuación se muestran los gráficos generados a partir de Kibana, el servicio que ofrece Elastic search para visualización de datos.

2.4.1. Canciones por artista.

Se pide contar y representar mediante un gráfico de dona cuántas canciones tiene cada artista, a continuación el resultado:

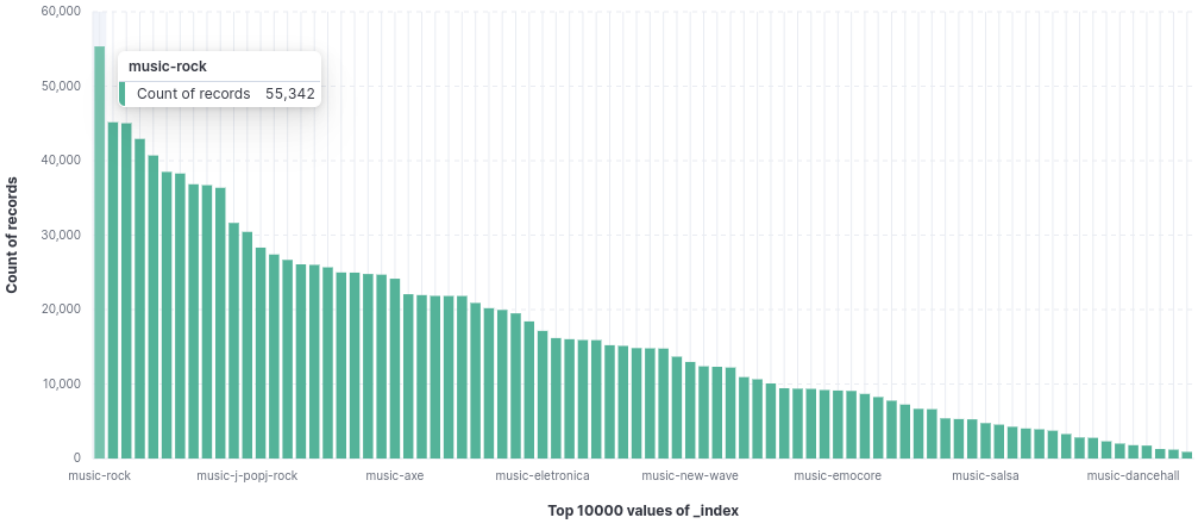


La anterior imagen es una versión simplificada y acotada de los resultados para poder visualizar lo que se nos entrega, a continuación una versión con una cantidad más grande de datos, pero más dificil de entender.



2.4.2. Canciones por género.

También se pide generar un gráfico de barras para visualizar las canciones por géneros, teniendo en consideración que cada índice representa un género. Por el contrario a la anterior representación, esta fue más fácil de generar con la totalidad por su cantidad.



2.4.3. 100 títulos más usados en las canciones.

La actividad también asignaba generar otro diagrama, pero esta vez a elección propia en temática y representación. Personalmente me interesó ver cuántas la predominancia de ciertos títulos en las canciones y ver cuáles eran los más famosos. A continuación su respresentación con un gráfico de nube.

3. Conclusión

En esta actividad, hemos explorado diversas etapas del manejo de datos en una plataforma de Elasticsearch. Desde la obtención y normalización de datos hasta la creación de gráficos para visualizar información relevante.

La normalización de los datos, transformándolos de archivos CSV a JSON, fue un paso fundamental para prepararlos para su inserción en Elasticsearch. Esto nos permitió manipular grandes volúmenes de datos de manera eficiente y escalable.

La subida de datos a Elasticsearch fue un proceso interesante, donde se aprendió a trabajar con la API de Elasticsearch. Se investigó diferentes métodos para cargar los datos, desde la creación de índices de forma individual hasta la utilización de la función *bulk* para manejar grandes volúmenes de información de manera más eficiente.

La búsqueda en Elasticsearch nos permitió encontrar información específica dentro de nuestros datos de manera rápida y precisa. Se hizo uso de una consulta para buscar canciones que contuvieran una frase específica en sus letras, demostrando la potencia de las capacidades de búsqueda de Elasticsearch.

Finalmente, la creación de gráficos nos brindó una manera visual y comprensible de analizar nuestros datos. Se usó Kibana para generar gráficos que nos mostraran la distribución de canciones por artista, género y títulos más usados, lo que nos permitió obtener insights valiosos sobre nuestros datos.

En resumen, esta actividad proporcionó una visión completa del proceso de manejo de datos en Elasticsearch, desde la adquisición y normalización hasta la visualización y análisis.