



universidad
de león



Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Sistemas de Información de Gestión y Business Intelligence

SISTEMA RECOMENDACIÓN DE EVENTOS MUSICALES



Miguel Ángel Pérez López

León, febrero de 2023

ÍNDICE

1. INTRODUCCIÓN.....	4
2. DESCRIPCIÓN DEL PROBLEMA.....	5
2.1 OBJETO DE ESTUDIO.....	5
2.2 MOTIVACIÓN.....	5
2.3 RESUMEN.....	5
3. HERRAMIENTAS UTILIZADAS.....	6
3.1 FRONTEND.....	6
3.2 BACKEND.....	7
3.3 BASE DE DATOS.....	8
3.4 OTRAS HERRAMIENTAS DE USO GENERAL.....	9
4. LA APLICACIÓN EN DETALLE.....	10
4.1 BASE DE DATOS.....	10
4.2 BACKEND.....	16
4.3 FRONTEND.....	17
4.3.1 ESTRUCTURA GENERAL.....	17
4.3.2 VISTAS.....	18
4.3.2.1 EXPANSIÓN DE UN CONCIERTO.....	18
4.3.2.2 LOGIN.....	19
4.3.2.3 PRINCIPAL.....	20
4.3.3 COMPONENTES.....	20
4.3.3.1 CARD ARTIST.....	20
4.3.3.2 CARD CONCERT.....	21
4.3.3.3 CARD ROOM.....	21
4.3.3.4 FOOTER.....	22
4.3.3.5 TOP BAR.....	22
5. ALGORITMOS DE RECOMENDACIÓN EMPLEADOS.....	24
5.1 ALGORITMO BASADO EN CONTENIDO PARA CONCIERTOS SIMILARES.....	24
5.2 ALGORITMO DE FILTRADO COLABORATIVO PARA ARTISTAS SIMILARES.....	26
5.3 ALGORITMO BASADO EN LA ACTIVIDAD DEL USUARIO PARA CONCIERTOS SIMILARES.....	27
5.4 OTROS ALGORITMOS INTERESANTES.....	32
5.4.1 SALAS MEJOR VALORADAS.....	32

5.4.2 ARTISTAS CON MÁS SEGUIDORES.....	32
5.4.3 ARTISTAS CON MÁS CONCIERTOS.....	32
5.4.4 SALAS CON MÁS CONCIERTOS.....	32
5.4.5 CONCIERTOS CON MÁS ME GUSTAS.....	32
5.4.6 CONCIERTOS QUE PERTENECEN A LA CIUDAD DEL USUARIO.....	33
6. ANÁLISIS DE RESULTADOS.....	33
6.1 ALGORITMO BASADO EN CONTENIDIO PARA CONCIERTOS SIMILARES.....	33
6.1.1 EJEMPLO DE USO PARA UN CONCIERTO CONCRETO.....	33
6.2 ALGORITMO DE FILTRADO COLABORATIVO PARA ARTISTAS SIMILARES.....	35
6.2.1 EJEMPLO PARA 3 USUARIOS.....	35
6.3 ALGORITMO BASADO EN LA ACTIVIDAD DEL SUUARIO PARA CONCIERTOS SIMILARES.....	37
6.3.1. EJEMPLO PARA UN PERFIL.....	37
7. ANÁLISIS DAFO.....	39
7.1 DEBILIDADES.....	39
7.2 AMENAZAS.....	40
7.3 FORTALEZAS.....	40
7.4 OPORTUNIDADES.....	40
8. LÍNEAS DE FUTURO.....	41
9. LECCIONES APRENDIDAS.....	41
10. ANEXOS.....	42
11. BIBLIOGRAFÍA Y RECURSOS.....	42

1.Introducción

En la presente memoria se detalla la aplicación web LoveLiveConcerts. Una aplicación desarrolla para la asignatura de Sistemas de Información de Gestión y Business Intelligence (SIBI), perteneciente al primer semestre del cuarto curso del Grado en Ingeniería informática de la Universidad de León.

Inicialmente se describirá el problema al que se pretende dar solución, con los objetivos y la motivación de la que se parte. Seguidamente se expondrán las herramientas que se han empleado durante el proceso del desarrollo, los algoritmos implementados, así como su funcionamiento y el posterior análisis de los resultados obtenidos. Complementando la memoria, se detallará un análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) del proyecto y se proponen varias líneas de futuro para completar su funcionalidad

Por último, se concluye haciendo un repaso de las distintas lecciones aprendidas y experiencia adquirida por mi persona a lo largo de la elaboración de este proyecto.

2.DESCRIPCIÓN DEL PROBLEMA

2.1. OBJETO DE ESTUDIO

Por tanto, el objeto de este proyecto es el desarrollo de una aplicación web, que teniendo en cuenta una serie de datos sobre los me gusta de los usuarios a conciertos y de sus preferencias, así como de sus gustos, emita recomendaciones sobre otros conciertos u otros artistas que puede dar a conocer.

2.2. MOTIVACIÓN

Con el intento de unificar diferentes requisitos que exige un amante de la música en directo, esa es la motivación principal. Conseguir un producto que te recomiende conciertos y artistas siguiendo el gusto de cada usuario y así profundizar en los usuarios, ofreciendo un uso exclusivo.

2.3. RESUMEN

Es una aplicación web basada en Node JS que utiliza una base de datos de grafos de conocimiento y que precisa el uso de algoritmos de Inteligencia Artificial para recomendar productos a usuarios haciendo uso del Big Data.

3. HERRAMIENTAS UTILIZADAS

En esta sección se explica la arquitectura de la aplicación y una descripción de las distintas herramientas utilizadas para el posible desarrollo del proyecto.

Las aplicaciones web operan bajo el modelo Cliente-Servidor, el cliente ejecuta el código frontend y el servidor el código backend. Para la persistencia de la información se utiliza en este caso una base de datos. Así quedan definidas las tres artes que la componen.

En concreto, las herramientas utilizadas para cada parte fueron:



3.1. FRONTEND

Es la capa que se encuentra al lado del cliente, es decir, todo lo que los usuarios pueden ver en la pantalla de la aplicación. Para el desarrollo de esta parte se utiliza el framework React.js basado en el lenguaje de JavaScript, haciendo el propio uso ya, de HTML y CSS, ambos incluidos en este entorno de trabajo.

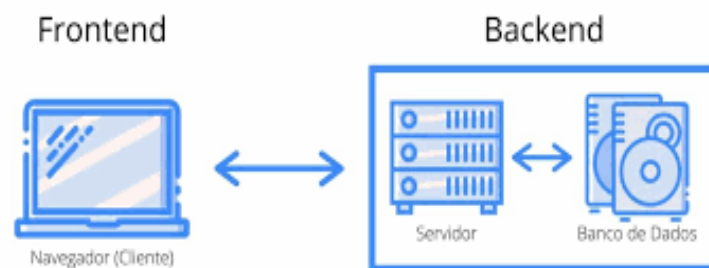
Con esto no es suficiente para realizar una buena página web a nivel de usuarios. Para ello, se utilizaron librerías externas:

- **AXIOS:** es una librería de JavaScript que puede ejecutarse en el navegador y que nos permite hacer sencillas las operaciones como cliente HTTP, por lo que podremos configurar y realizar solicitudes a un servidor y recibiremos respuestas fáciles de procesar. Es decir, es la herramienta que nos permite hacer la conexión entre el frontend y el backend.
- **MATERIAL UI:** es una librería de componentes de React.js, de código abierto creada por Google, basada en Material UI, la cual nos brinda una serie de pautas y lineamientos que sientan la base para crear diseños para aplicaciones web.
- **ICONS MATERIAL:** es una librería como Material UI pero basada en iconos que puedes implementar en tu aplicación sin la necesidad de descargar imágenes.
- **REACT-ROUTER-DOM:** es una librería que se utiliza en React.js y facilita el proceso de definir las rutas de navegación dentro de una aplicación.

**AXIOS**

3.2. BACKEND

El backend tiene que ver con la lógica de negocio de la aplicación, es decir, contiene todo el código de los algoritmos de recomendación, los endpoints que enrutan las peticiones provenientes del cliente(frontend) y contestan las respuestas, la lógica correspondiente al tratamiento de los datos y el acceso a la persistencia de los datos, es decir, es el manejador de la base de datos.



Para esta parte, se optó por las siguientes tecnologías:



- **NODE JS:** es un entorno de ejecución de código JavaScript que permite la creación de aplicaciones. Se puede usar el mismo lenguaje y las mismas herramientas que se usan en el navegador con la ventaja de usarlas en el servidor. Tiene un amplio paquete de oportunidades para crear diversas funciones.

- EXPRESS JS: es un framework de desarrollo flexible y simple para Node.js. Express nos permite crear el mecanismo de autenticación de un usuario, recibir las peticiones HTTP que se mandan desde el lado del cliente, establecer conexión con la base de datos, y además potencia la organización dentro del servidor para los diferentes componentes.
- NEO4J DRIVER: es el componente que permite enlazar el backend con la base de datos aportando que te puedas conectar con un usuario y una contraseña de usuario de la base de datos.

Ambas se basan en el lenguaje JavaScript, que es un lenguaje ligero y más sencillo de aprender, donde a la vez es muy permisivo.

3.3. BASE DE DATOS

En este apartado existen muchas bases de datos y alguna con esquemas y distinciones en la forma del almacenamiento de los datos. Por un lado, están las bases de datos típicamente conocidas por relacionales como MySQL, Oracle, etc. Y por otro, están las no relacionales, donde se incluyen las bases de datos orientadas a grafos como Neo4j. Una diferencia fundamental es la forma de tratar las relaciones de los datos en los dos modelos de almacenaje. En las bases de datos orientadas a grafos, son registros diferentes e individuales mientras que, en las relacionales, los datos del mismo tipo se almacenan todos en la misma tabla. También, son diferenciadas porque tienen relaciones individuales lo que hace que sean mucho más rápidas para los datos que están conectados por las relaciones y más accesibles de una forma más sencilla. Las consultas a la base de datos son mucho más cortas y concisas, algo que es para tener en cuenta.

En este caso concreto, ya que los usuarios tienen relaciones directas con conciertos (me gusta) o con los artistas que sigue, así como las valoraciones a las diferentes salas, géneros favoritos y géneros de los artistas, son muchas relaciones entre sí. Por lo que es muy interesante y apropiado usar la base de datos de Neo4j ya que prioriza tantas relaciones entre sí y hace que aporte recomendaciones más exactas con la utilización de todos estos datos.

Personalmente, nunca había escuchado hablar de Neo4j por lo que mis conocimientos eran nulos. Para aprender acerca de Neo4j hice varios certificados aporta la propia compañía en su página web.

Cabe destacar la importancia de Cypher que es el lenguaje para hacer las consultas (insertar, modificar, eliminar y seleccionar), está basado en SQL pero de una forma mucho más sencilla y aplicable a gran escala. La plataforma de Neo4j permite a los desarrolladores desplegar la base de datos en 3 formas diferentes: Neo4j Desktop, Aura y Sandbox. Yo he elegido la de Desktop porque ha sido la forma más sencilla de poder importar los datos desde un dataset solo con la inclusión de estos ficheros en la configuración de la propia base de datos.



Además, Neo4j te permite utilizar librerías internas para diferentes funcionalidades. En el proyecto se utilizan las siguientes:

- **APOC:** La biblioteca APOC (Amazing Procesudres on Cypher) consta de muchos (alrededor de 450) procedimientos y funciones para ayudar con muchas tareas diferentes en áreas como integración de datos, algoritmos gráficos o conversión de datos.
- **Graph Data Science Library:** La biblioteca Neo4j Graph Data Science (GDS) proporciona amplias capacidades analíticas centradas en algoritmos gráficos. La biblioteca incluye algoritmos para detección de comunidades, centralidad, similitud de nodos, búsqueda de rutas y predicción de enlaces, así como procedimientos de catálogo de gráficos diseñados para admitir flujos de trabajo de ciencia de datos y tareas de aprendizaje automático en sus gráficos. Todas las operaciones están diseñadas para una escala y paralelización masivas, con una API general y personalizada diseñada para el procesamiento global de gráficos y estructuras de datos comprimidos en memoria altamente optimizados.

3.4. OTRAS HERRAMIENTAS DE USO GENERAL

A lo largo del proceso de desarrollo, se ha trabajado en el editor de código Visual Studio Code. También, se ha aprovechado de GitHub para el control de las diferentes versiones y de las actualizaciones del código.

OSF: Cuaderno de trabajo en el que llevar el progreso actualizado y un seguimiento que facilita la redacción de la memoria y el almacenamiento del proyecto en su totalidad, así como su posterior publicación.



4. LA APLICACIÓN EN DETALLE

En el siguiente apartado se realiza una descripción del diseño de la aplicación en cada una de sus partes. Como ya se intuye a partir del punto anterior, la estructura de la aplicación se divide en tres partes, la base de datos, el cliente o *front-end* y el servidor o *back-end*.

4.1. BASE DE DATOS

Para poder realizar un sistema de recomendación que funcione correctamente y llegue a realizar un trabajo serio es necesario tener una buena base de datos.

Como el tema del que me encargaba era todo lo relacionado con la música en directo, como los conciertos y artistas, estuve buscando datasets de conciertos futuros de artistas en plataformas como Kaggle, incluso en datos abiertos del gobierno, ayuntamientos y no encontré nada que me sirviera para tener una base de datos compacta. Por lo que me decidí por hacer WebScrapping. Empleé bastante tiempo en buscar páginas web donde conseguir esta información, pero sin éxito ya que se hacía muy complicado el navegar dentro de cada web entre diferentes artistas y, además, en la mayoría faltaban datos importantes, como el lugar del concierto, la fecha, precio..., es decir, datos que se necesitaban para los algoritmos.

Como solución ante estos problemas, cree mi propio dataset con datos aleatorios estimados por mí mismo formando conciertos artificiales con artistas de diferentes géneros en salas por todo España. De esta forma, podía asegurarme tener un dataset compacto y aplicable a los requisitos necesarios para hacer efectivas las recomendaciones de los algoritmos.

Busqué información en internet acerca de artistas de diferentes géneros y salas de diferentes ciudades de España. Aquí, sí que intenté respetar al máximo la realidad, dado que, aunque, el precio, fecha y hora de comienzo fueran datos aleatorios, y los conciertos fueran ficticios, al menos respetar que las salas existan y concuerden con la ciudad relacionada y, que los artistas pertenezcan al género con el que se les caracteriza. El motivo de esto es que, aunque el concierto en la realidad no se vaya a celebrar, al menos, cuando un usuario le da me gusta a conciertos de determinados artistas, respetar así sus gustos y poder realizar recomendaciones realmente válidas acerca de artistas y conciertos de esos géneros. El fin de esto es poder respetar la veracidad de los datos dentro de los conciertos ficticios.

PROCESAMIENTO DEL DATASET

- Cree 10 listas diferentes en la que cada una pertenece a un género diferente. Los géneros disponibles son: pop, flamenco, rock, indie, metal, latino, hiphop, reggaetón, bachata y dembow. Cada lista contiene artistas del género al que se dedica su música. Hay listas con 20 artistas, otras con 15, y alguna con 5.

```
listaArtistasPop = ["Manuel Carrasco", "Dani Martín", "David Bisbal", "Melendi", "Alex Ubago",
    "Alejandro Sanz", "Pablo Lopez", "Chanel Terrero", "David Bustamante", "Blas Cantó",
    "Alvaor Solen", "Nil Moliner", "Cepeda", "Vanessa Martín", "Antonio Orozco",
    "Conchita", "Marc Seguí", "Pol Granch", "Pole.", "Beret"]
listaArtistasFlamenco = ["C. Tangana", "Omar Montes", "India Martínez", "Estopa",
    "Diego El Cigala", "El Barrio", "Joaquín Sabina", "Niña Pastori", "Los Delinquentes",
    "Arcangel", "Paco de Lucía", "Gipsy Kings", "Fondo Flamenco", "Dáviles de Novelda",
    "Camela", "Los Chichos", "Los Chunguitos", "Los Rebutijos", "Elena Vargas"]
listaArtistasRock = ["Fito y Fitipaldis", "Talco", "Desakato", "SKA-P", "Boikot",
    "Celtas Cortos", "Tanxugueiras", "Los de Marras", "La Fuga", "Sinkope",
    "Porretas", "Kaotiko", "Mago de Oz", "Ciudad Jara", "Valira",
    "", "", "", "", ""]
listaArtistasIndie = ["Leiva", "La M.O.D.A.", "Lori Meyers", "Vetusta Morla", "IZAL",
    "Love Of Lesbian", "Dorian", "Supersubmarina", "Viva Suecia", "Sidonie",
    "Taburete", "Macaco", "Sidecars", "La Pegatina", "Rulo y la Contrabanda",
    "Carlos Sadness", "Shinova", "Rozalen", "Miss Cafeína", "Ivan Ferreiro"]
listaArtistasMetal = ["Corvus V", "Black Bomber", "Sober", "Deshonra", "The Broken Horizon",
    "Metalica", "Guns N Roses", "Pink Floyd", "AC/DC", "Nirvana",
    "", "", "", "", "",
    "", "", "", "", ""]
listaArtistasLatino = ["Marc Anthony", "J Balvin", "Jennifer Lopez", "Juanes", "Maluma",
    "Anuel AA", "Freid", "Ozuna", "Camilo", "Myke Towers",
    "RVFV", "Justin Quiles", "Ovy On The Drums", "Bad Bunny", "Eladio Carrion",
    "Nicky Jam", "CNCO", "Morat", "Plan B", "Bandana"]
listaArtistasHipHop = ["Nikone", "Natos y Waor", "FERNANDOCOSTA", "Recycled J", "Ajax y Prok",
    "Israel B", "Young Beef", "Morad", "Rels B", "Duki",
    "Kase. O", "Los Chikos del Maiz", "Nach", "Violadores del Verso", "Chojin",
    "SFDK", "Delaossa", "Mala Rodriguez", "Tote King", "Rayden"]
listaArtistasReggaeton = ["Enrique Iglesias", "Juan Magan", "DaSoul", "Ana Mena", "Rosalía",
    "Lola Indigo", "Bad Gyal", "Danny Romero", "Daddy Yankee", "Don Omar",
    "Shakira", "Quevedo", "Luis Fonsi", "Karol G", "Ricky Martin",
    "Raw Alejandro", "Sebastian Yatra", "Mora", "Ryan Castro", "Tiago PZK"]
listaArtistasBachata = ["Manuel Turizo", "Romeo Santos", "Prince Royce", "Luis Vargas", "Daniel Santacruz"]
listaArtistasDembow = ["El Cherry Scum", "Lirico en la Casa", "El Alfa", "Farruko", "Miky Woodz"]
```

- Cree otra lista con las salas disponibles y la ciudad a la que pertenecen cada una. Hay 40 salas distintas disponibles de diversas ciudades de España.

```
listaSalas = [{"Espacio Vias", "Leon"}, {"Palau Sant Jordi", "Barcelona"}, {"Plaza de Toros", "Roquetas de Mar"},
{"Estadio Jose Copete", "Albacete"}, {"Noches de la Maestranza", "Sevilla"},
{"Gran Canaria Arena", "Las Palmas de Gran Canaria"}, {"Recinto Ferial", "Santa Cruz de Tenerife"},
{"Palacio de Deportes Martin Carpena", "Malaga"}, {"Palacio de Toros", "Alicante"},
{"Plaza de Toros", "Leon"}, {"Wizink Center", "Madrid"}, {"Coliseum", "A Coruña"},
{"Plaza de Toros", "Valencia"}, {"Plaza de Toros", "Murcia"}, {"Navarra Arena", "Pamplona"},
{"Estadio La Cartuja", "Sevilla"}, {"Estadio Iberoamericano", "Huelva"},
{"Pabellon de la Magdalena", "Aviles"}, {"Campo de Futbol", "Almendralnejo"}, {"Muelle Ciudad", "Cadiz"},
{"Trui Son Fusteret", "Mallorca"}, {"Bilbao Miribilla", "Bilbao"}, {"Santa Maria Polo Club", "Sotogrande"},
{"Marenostrum Fuengirola", "Fuengirola"}, {"Centro Hipico", "Mairena del Aljarafe"},
{"Sala Capitol", "Santiago de Compostela"}, {"Palacio de Congresos", "Leon"}, {"La Iguana Club", "Vigo"},
{"Auditorio de Ponferrada", "Ponferrada"}, {"Civitas Metropolitano", "Madrid"},
{"La cubierta de Leganes", "Leganes"}, {"Sala Karma", "Pontevedra"}, {"Sala Canacol", "Madrid"},
{"Black Note Club", "Valencia"}, {"Auditorio de Zaragoza", "Zaragoza"},
{"Auditorio Marbella Arena", "Marbella"}, {"Coliseum Burgos", "Burgos"},
{"Estadio la Romareda", "Zaragoza"}, {"Plaza de Toros de Las Ventas", "Madrid"}, {"Sala Studio 54", "Leon"}]
```

Los datos estimados:

- Cree una lista de posibles horas de comienzo de los conciertos entre las 10:00h de la mañana y las 4:00h de la madrugada, con 1h de diferencia entre cada hora de comienzo.

```
horasDisponibles = ["10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "18:00", "19:00", "20:00",
"21:00", "22:00", "23:00", "00:00", "01:00", "02:00", "03:00", "04:00"]
```

- En el campo del precio, se va a generar por cada concierto aleatoriamente un número entre 0 (entrada gratuita) y 70 (precio máximo, 70 euros) que será el que designa el precio de cada concierto.

```
# Price
numero3 = random.randint(0, 70)
```

- Para la fecha del concierto lo que hice fue establecer unos márgenes de fechas, siendo la primera fecha disponible el día 1 de febrero de 2023 y la última fecha disponible el día 31 de diciembre de 2023.

```
inicio = datetime(2023, 2, 1)
final = datetime(2023, 12, 31)
```

Para la creación de cada concierto

- Las listas con los artistas, las salas y los datos estimados los cargue en un programa que cree en Python para combinarlos y así de esta forma, crear los conciertos. El procedimiento que seguí fue el siguiente:
 - Fui leyendo artista por artista de cada género y por cada uno, generando sus propios conciertos de modo que, cada artista obtenga 20 conciertos ficticios con las siguientes características (Importante, las características nombradas a continuación son individuales para cada uno de los 20 conciertos creados):
 - El ID de cada concierto es un simple número que se va sumando automáticamente de uno en uno, siendo único para cada concierto.
 - El nombre del concierto es el mismo que el nombre del artista.
 - La fecha de la celebración del concierto se genera aleatoriamente por cada uno, dando de resultado una fecha comprendida entre los márgenes comentados anteriormente.

```
# Date
random_date = inicio + (final - inicio) * random.random()
```

- Para la hora de comienzo del concierto, se genera aleatoriamente un numero entre 0 y 19, que son los valores posibles para este campo. Es decir, hay 19 horarios diferentes en la lista, entonces el numero aleatorio corresponde a una hora.
- Para las salas, seguimos el mismo formato que con las horas. Se genera aleatoriamente un numero entre 0 y 40, que es el número de salas disponibles, y el entero generado es el que corresponde con la sala para el concierto. De aquí obtenemos la ciudad ya que, con la sala ya viene especificado la ciudad donde se encuentra.
- En el caso del precio, como ya he comentado antes, el rango está entre 0 y 70, y el valor obtenido será aleatorio.
- El género será el marcado por la lista a la que pertenezca cada artista.
- Script se puede ver en el archivo auxiliar

De este modo, contamos con:

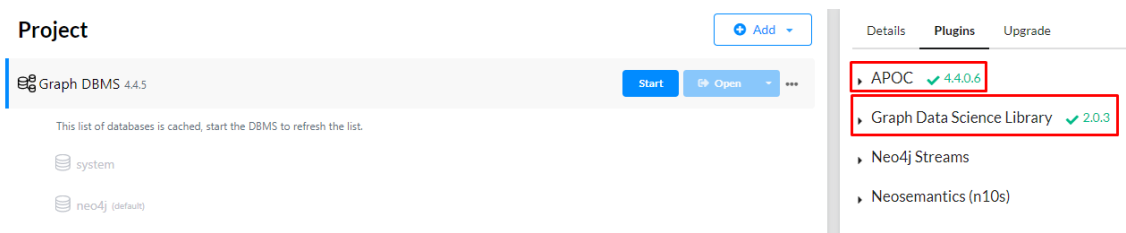
- 160 artistas
- 10 géneros
- 40 salas repartidas entre diversas ciudades
- 19 horarios de conciertos diferentes
- Fechas aleatorias entre el 01/02/2023 y el 31/12/2023 (muchos días posibles)
- 3200 conciertos

Ya hecho el procesado de datos se exporta en formato CSV para que lo pueda importar Neo4j.

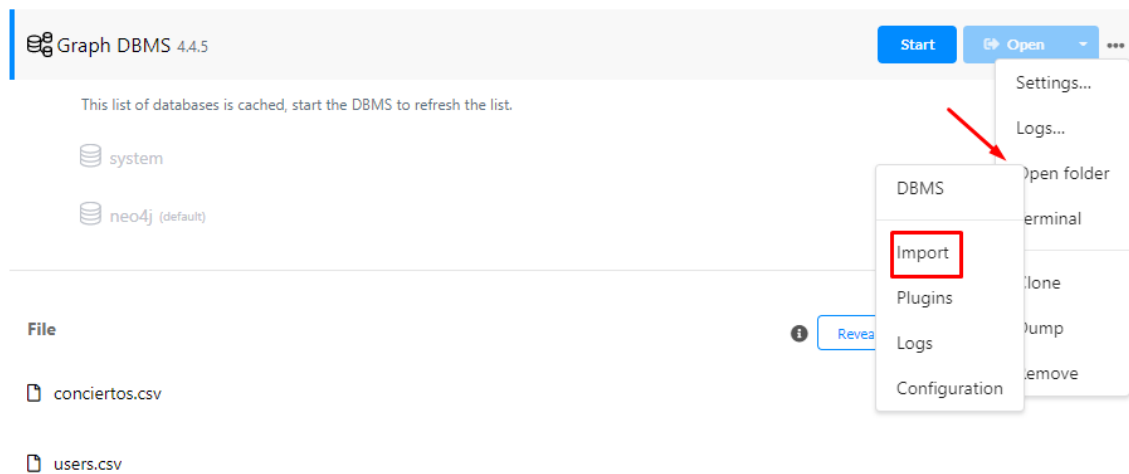
CREACIÓN DEL GRAFO DE CONOCIMIENTO

Teniendo el archivo .csv ultimado y preparado, el siguiente paso resuelta el diseño del grafo de conocimiento, la importancia del dataset y la construcción del código cypher necesario para hacer dicho grafo efectivo.

En este paso, el proceso es tan sencillo como crear una base de datos en Neo4j Desktop y habilitar las librerías extra que nos provee la propia base de datos, las cuales ya hemos comentado anteriormente.



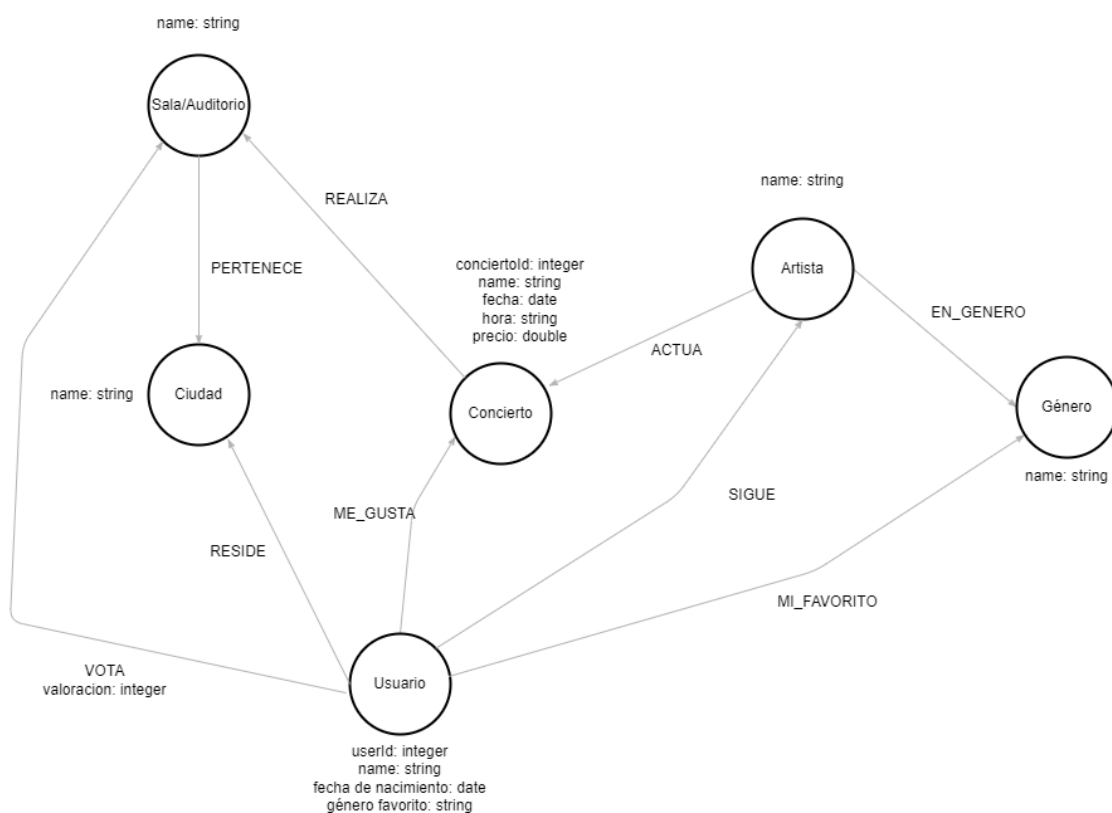
Otro paso importante, es agregar los archivos que vayamos a importar, en el directorio import del Neo4j Desktop.



Sin estos archivos en el directorio, a la hora de leerlos, Cypher no conseguirá importar nada.

ESTRUCTURA DEL GRAFO DE CONOCIMIENTO

El grafo de conocimiento está compuesto de 6 nodos y 9 relaciones entre ellos como se puede ver a continuación:



Los nodos en el grafo están definidos de la siguiente manera:

- Sala. Es el nodo que determina cada sala existente en la base de datos.
- Ciudad. Nodo para determinar la ciudad donde vive el usuario y donde pertenece cada sala.
- Usuario. Nodo principal donde se guarda información de cada usuario de la aplicación
- Concierto. Nodo donde se guarda información de cada concierto y que se relaciona con el resto de los nodos que precisan de sus datos.
- Artista. Nodo en el que se guarda información de cada artista.
- Genero. Es el nodo que se utiliza para determinar el género.

Las relaciones en el grafo están definidas de la siguiente manera:

- VOTA. Esta relación se encarga de unir la valoración del usuario acerca de una sala cualquiera.
- RESIDE. Aquí relaciona el nodo Usuario con la ciudad en la que reside, solo puede tener una relación.
- PERTENECE. Esta relación sirve para indicar la ciudad a la que pertenece una sala. Diferentes salas pueden pertenecer a la misma ciudad.
- REALIZA. Determina la unión de la sala donde se realiza cada concierto.
- ME_GUSTA. Relaciona los conciertos que le gustan a cada usuario.
- ACTUA. Sirve para relacionar los conciertos en los que actúa cada artista.
- SIGUE. Esta relación determina los artistas a los que sigue cada usuario.
- MI_FAVORITO. Esta relación determina el género favorito de cada usuario.
- EN_GENERO. Sirve para indicar el género al que se dedica cada artista.

CÓDIGO CYPHER

Una vez que tenemos la base de datos creada y preparada para importar los datos, y sabemos cómo va a ser la estructura de nuestros nodos y relaciones, el paso siguiente es ejecutar el código Cypher que se encarga de la construcción del grafo y cada una de las sentencias ejecutadas se encuentran en un archivo auxiliar llamado ____ .

Este es un ejemplo de código para importar los conciertos

```
LOAD CSV WITH HEADERS FROM 'file:///concierotos.csv' AS row  
WITH row  
WHERE row.concertID IS NOT NULL  
MERGE (c:Concierto {  
    concertID: toInteger(row.concertID),  
    name: row.name,  
    date: row.date,  
    hourStart: row.hourStart,  
    price: row.price  
})  
MERGE (l:City {
```

```
        city: row.city
    })
    MERGE (a:Artist {
        name: row.artist
    })
    MERGE (g:Genre {
        genre: row.genre
    })
    MERGE (r:Room {
        room: row.room
    })
    MERGE (a)-[:ACTUA]->(c)
    MERGE (c)-[:REALIZA]->(r)
    MERGE (r)-[:PERTENECE]->(l)
    MERGE (a)-[:EN_GENERO]->(g)
```

4.2. BACK-END

Como se detalló anteriormente, el backend está compuesto por distintos puntos de comunicación o endpoints, cada uno de los cuales ofrece acceso a una funcionalidad concreta:

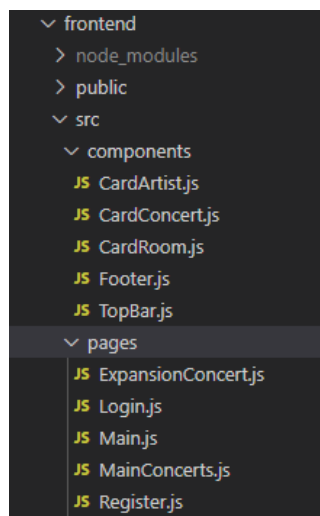
- **/register**. Crea un nodo Usuario en la Base de Datos. Recibe como parámetros el nombre de usuario, contraseña, nombre, apellidos, fecha de nacimiento, sexo, ciudad de residencia y el género favorito.
- **/login**. Comprueba que los credenciales introducidos sean correctos. Recibe como parámetros el usuario y la contraseña.
- **/getConcertsCityUser**. Obtiene los conciertos que se celebran en la ciudad en la que reside el usuario.
- **/allArtists**. Devuelve todos los artistas que hay almacenados en la base de datos.
- **/allRooms**. Devuelve todas las salas que hay almacenadas en la base de datos junto a la ciudad a la que pertenece.
- **/allConcerts**. Devuelve todos los conciertos que hay almacenados en la base de datos junto a todas sus características, incluido el género, la sala, la ciudad y el artista.
- **/moreLikeConcerts**. Devuelve los conciertos que más me gusta tienen.
- **/moreConcertsRooms**. Devuelve las salas con más conciertos.
- **/artistWithMoreConcerts**. Devuelve los artistas con más conciertos.
- **/artistsWithMoreFollowers**. Devuelve los artistas con más seguidores.
- **/getMoreValuationRooms**. Devuelve las salas mejor valoradas.
- **/getValuationRooms**. Este endpoint lo que hace es buscar todas las valoraciones de un usuario sobre una sala

- **/getConcert.** Se le pasa por parámetro el ID del concierto y devuelve todas las características de este.
- **/getConcertsLike.** Obtiene todos los conciertos que le ha dado me gusta un usuario.
- **/getArtistsFollowing.** Devuelve todos los artistas a los que sigue el usuario con la sesión activa.
- **/followArtist.** Sirve para dar seguir a un artista. Se pasa el ID por parámetro.
- **/stopFollowArtist.** Deja de seguir a un artista. Se pasa el ID por parámetro.
- **/likeConcert.** Sirve para dar me gusta a un concierto. Se pasa el ID por parámetro.
- **/dislikeConcert.** Quita el me gusta a un concierto. Se pasa el ID por parámetro.
- **/valorarSala.** Valora una sala que el usuario quiera. Se le pasa el ID de la sala y la valoración en cuestión.
- **/modificarValoraciónSala.** Sirve para modificar la valoración de una sala ya valorada. Se le pasas el ID de la sala y la valoración a modificar.
- **/eliminarValSala.** Elimina la valoración de una sala. Se le pasa el ID por parámetro.
- **/recommendedArtists.** Endpoint que da lugar al algoritmo 2 explicado posteriormente.
- **/concertsSimilarities.** Endpoint que da lugar al algoritmo 1 explicado posteriormente.
- **/actividadUsuario.** Endpoint que da lugar al algoritmo 3 explicado posteriormente.
- Otras funciones, que sirven de ayuda para los endpoints ya nombrados.

4.3. FRONT-END

4.3.1. ESTRUCTURA GENERAL

La estructura del frontend se ha creado siguiendo el modelo de componentes y páginas habiendo bastante diferencia entre ambos conceptos. Los componentes son reutilizables para utilizar en cualquiera de las páginas. Las páginas son cada una de las ventanas que dispone la aplicación y donde se muestran diferentes componentes.



4.3.2. VISTAS

4.3.2.1. Expansión de un concierto (localhost:3000/concierto)

Esta página proporciona una vista del concierto seleccionado por el usuario de forma más detallada y donde puedes ver los conciertos similares recomendados por uno de los algoritmos.

[VOLVER ATRAS](#)

Aitana

Hora de comienzo: 21:00

Fecha: 27/11/2022

Sala: WiZink Center

Ciudad: Madrid

Artista: Aitana

Genero: Pop

Precio: 35 e.

Conciertos recomendados

Alejandro Sanz

Hora de comienzo: 19:00

Fecha: 01/12/2022

Sala: WiZink Center

Precio: 10 e.

[ME GUSTA](#)[VER CONCIERTO](#)

C Tangana

Hora de comienzo: 22:00

Fecha: 15/11/2022

Sala: WiZink Center

Precio: 30 e.

[ME GUSTA](#)[VER CONCIERTO](#)

La MODA

Hora de comienzo: 21:00

Fecha: 26/11/2022

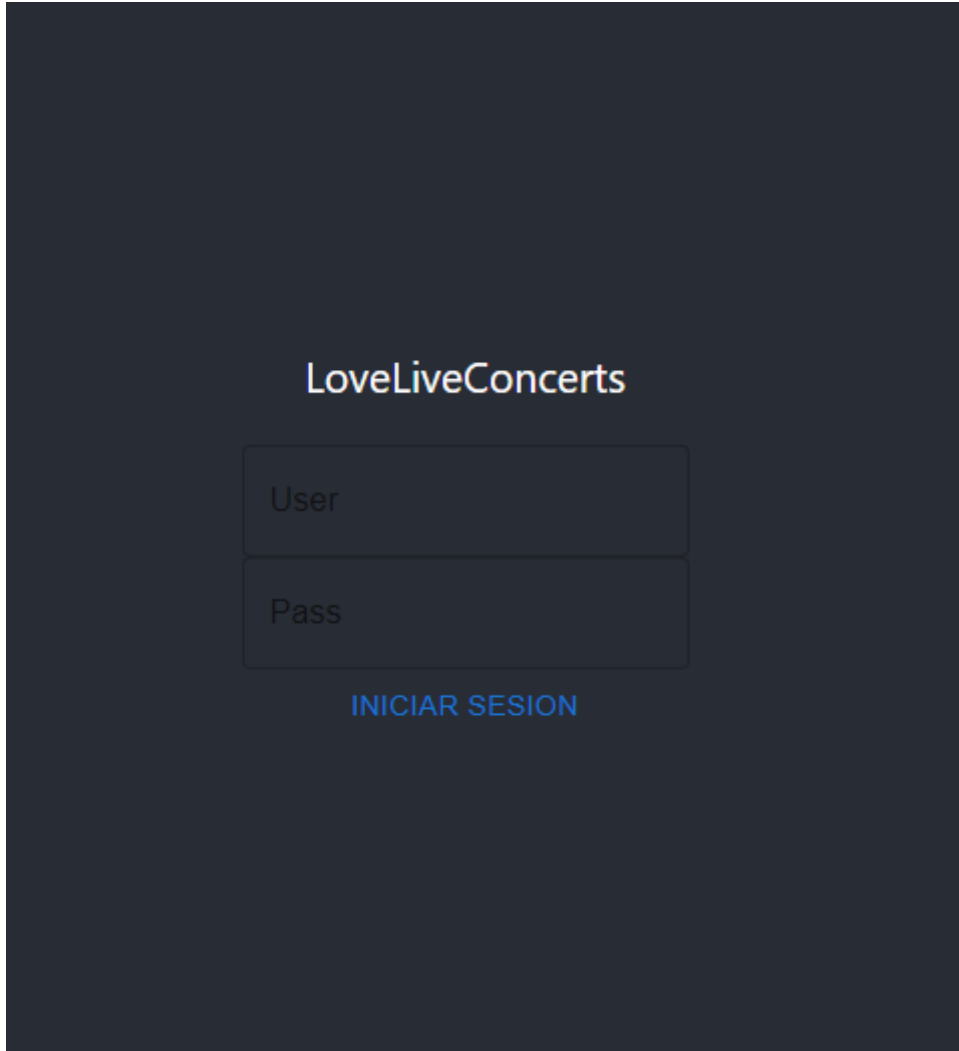
Sala: WiZink Center

Precio: 25 e.

[ME GUSTA](#)[VER CONCIERTO](#)

4.3.2.2. Login (localhost:3000)

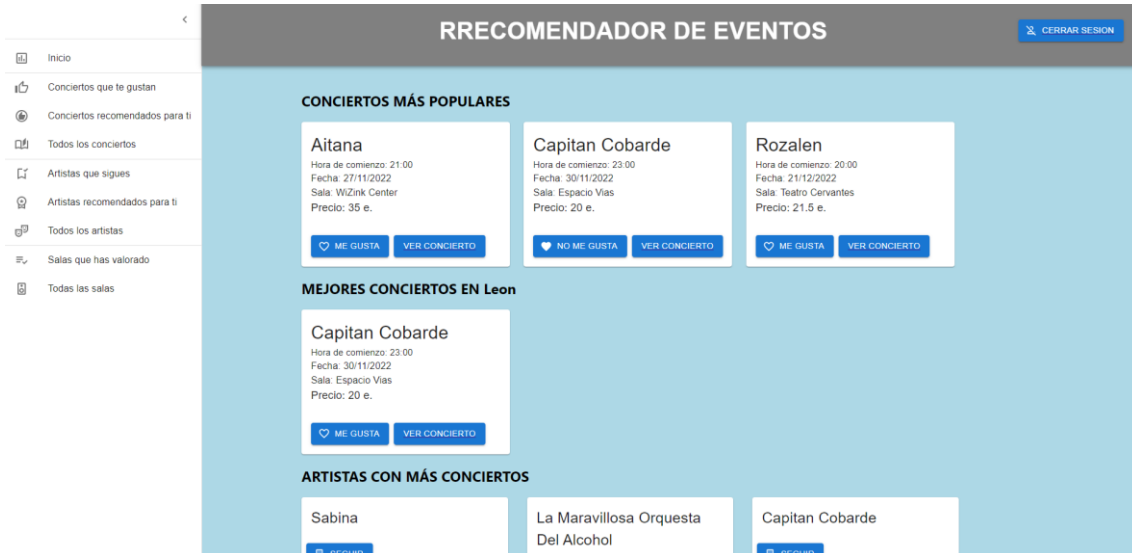
Es la página a la que te dirige la aplicación cuando la ejecutas y donde cada usuario inicia sesión con sus credenciales y da paso al interior de la aplicación.



The screenshot shows a dark-themed login interface for 'LoveLiveConcerts'. The title 'LoveLiveConcerts' is centered at the top in a light blue font. Below it are two input fields: 'User' and 'Pass', both with light blue borders and placeholder text. At the bottom, there is a blue button labeled 'INICIAR SESION'.

4.3.2.3. Principal (localhost:3000/incio)

Esta es la pantalla principal desde donde puedes acceder a toda la aplicación y realizar cualquier acción sobre conciertos, artistas y salas.

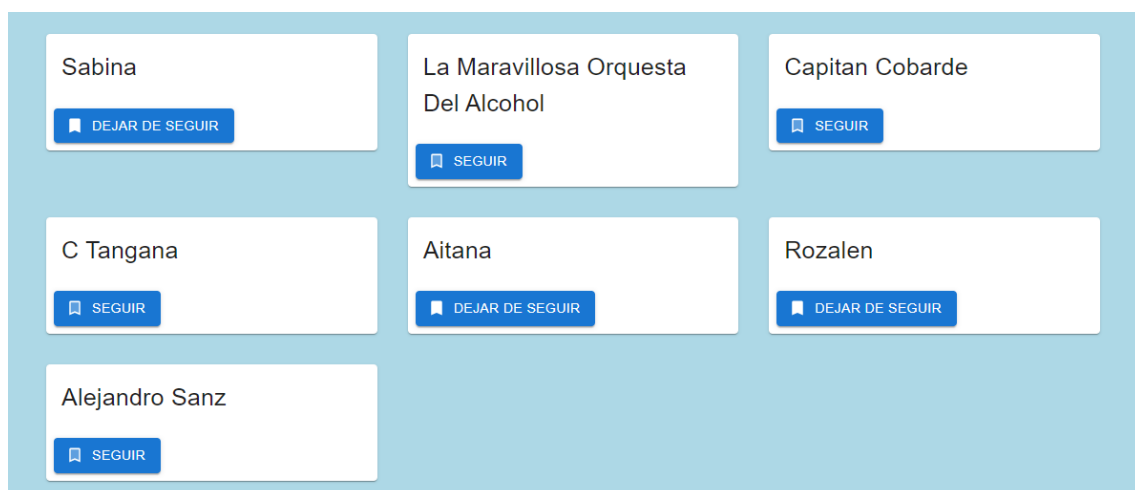


4.3.3. COMPONENTES

4.3.3.1. CARD ARTIST

Este componente se utiliza para visualizar cada uno de los artistas que tenemos en la base de datos mostrando los datos que disponemos de cada uno.

Se visualizaría de la siguiente forma:



Dentro de cada Card, te da la posibilidad de seguir a un artista en caso de que no lo sigas todavía, o en su defecto, que lo dejes de seguir.

Aquí se ven las dos funciones del botón:

```
const followArtist = () => {
  axios.post('/api/followArtist', {idArtists: item.name}).then(response => {
    if(response.data.r === 1) {
      console.log("Artista dado a seguir");
      setButtonFollow(-1);
    }
  }).catch();
};

const stopFollowArtist = () => {
  axios.post('/api/stopFollowArtist', { data: { idArtists: item.name } }).then(response => {
    if(response.data.r === 1) {
      console.log("Concierto quitado el me gusta");
      setButtonFollow(1);
    }
  }).catch();
};
```

4.3.3.2. CARD CONCERT

Para la Card del concierto, se visualizan los datos más comunes de cada concierto, reservándose así el resto de datos para cuando ves un concierto en la vista ampliada.

<p>Sabina</p> <p>Hora de comienzo: 22:00 Fecha: 23/11/2022 Sala: Gran Canaria Precio: 5 e.</p> <p>ME GUSTA VER CONCIERTO</p>	<p>Alejandro Sanz</p> <p>Hora de comienzo: 19:00 Fecha: 01/12/2022 Sala: WiZink Center Precio: 10 e.</p> <p>ME GUSTA VER CONCIERTO</p>	<p>Aitana</p> <p>Hora de comienzo: 21:00 Fecha: 27/11/2022 Sala: WiZink Center Precio: 35 e.</p> <p>NO ME GUSTA VER CONCIERTO</p>
---	---	--

Tienes la opción de dar me gusta o de quitarlo, así como de poder ver el concierto de una forma más amplia y detallada.

4.3.3.3. CARD ROOM

En esta card podemos observar los datos de cada sala, así como la valoración del usuario si ya la ha valorado. También te la opción de que elimines la valoración o que la puedas modificar.

<p>Gran Canaria</p> <p>★★★★☆</p> <p>MODIFICAR VALORACION</p> <p>ELIMINAR VALORACION</p>	<p>WiZink Center</p> <p>☆☆☆☆☆</p> <p>VALORAR SALA</p>	<p>Espacio Vias</p> <p>☆☆☆☆☆</p> <p>VALORAR SALA</p>
<p>Teatro Cervantes</p> <p>☆☆☆☆☆</p> <p>VALORAR SALA</p>		

Aquí se visualizan la llamada al backend para realizar las diferentes funciones necesarias:

```
// PONER LA VALORACION
const valorarSala = () => {
  axios.post('/api/valorarSala', {idRoom: item.name, val: valueRating}).then(response => {
    if(response.data.r === 1) {
      console.log("Sala valorada");
      setButtonValuation(1);
    }
  }).catch();
};

const modificarValoracionSala = () => {
  axios.post('/api/modificarValoracionSala', {idRoom: item.name, val: valueRating}).then(response => {
    if(response.data.r === 1) {
      console.log("Concierto quitado el me gusta");
    }
  }).catch();
};

const eliminarValSala = () => {
  axios.post('/api/eliminarValSala', { data: {idRoom: item.name} }).then(response => {
    if(response.data.r === 1) {
      console.log("Concierto quitado el me gusta");
      setButtonValuation(-1);
      setValueRating(0);
    }
  }).catch();
};
```

4.3.3.4. FOOTER

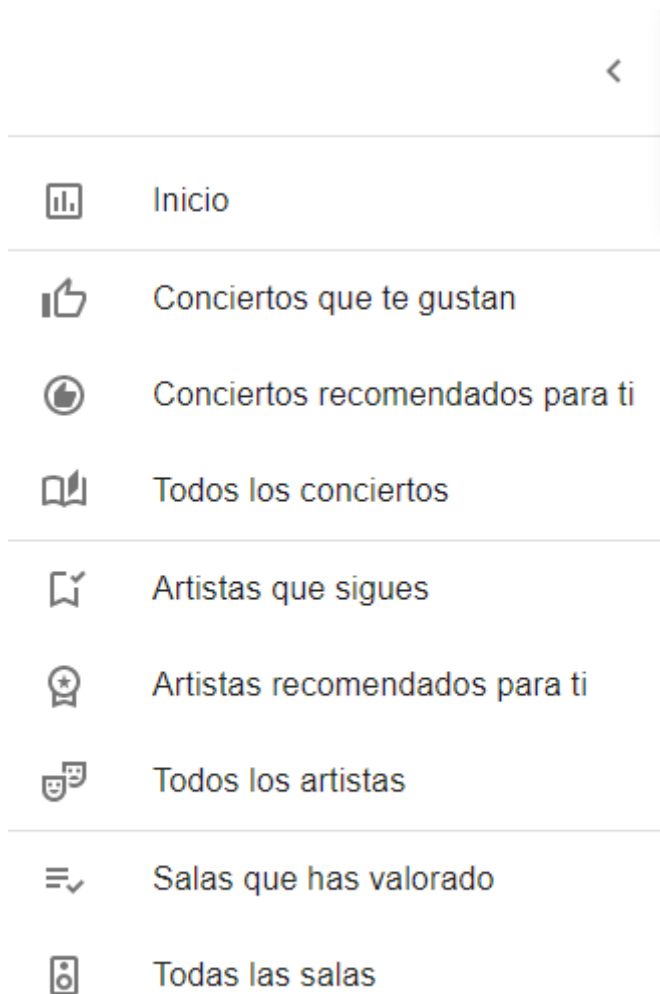
Se ha creado un pie de pagina para reservar los derechos de imagen y poder poner información adicional en un futuro.

Love Live Concerts
Copyright © Your Website 2023.

4.3.3.5. TOP BAR

Este componente es el que determina la barra superior de la pantalla principal junto al menú lateral izquierdo.





Aquí es donde se denomina todas las opciones del menú izquierdo:

```
140 >
141 <IconButton onClick={handleDrawerClose}>
142 |   <ChevronLeftIcon />
143 </IconButton>
144 </Toolbar>
145 <Divider />
146 <List component="nav">
147 |   <ListItemButton onClick={() => selectItem("Menu_1")}>
148 |     <ListItemIcon>
149 |       <AssessmentOutlinedIcon />
150 |     </ListItemIcon>
151 |     <ListItemText primary="Inicio" />
152 |   </ListItemButton>
153 |   <Divider />
154 |   <ListItemButton onClick={() => selectItem("Menu_2")}>
155 |     <ListItemIcon>
156 |       <ThumbUpAltOutlinedIcon />
157 |     </ListItemIcon>
158 |     <ListItemText primary="Conciertos que te gustan" />
159 |   </ListItemButton>
160 |   <ListItemButton onClick={() => selectItem("Menu_3")}>
161 |     <ListItemIcon>
162 |       <RecommendOutlinedIcon />
163 |     </ListItemIcon>
164 |     <ListItemText primary="Conciertos recomendados para ti" />
165 |   </ListItemButton>
166 </List>
```

5.ALGORITMOS DE RECOMENDACIÓN EMPLEADOS

En el presente apartado se realiza una descripción de los algoritmos empleados por nuestro motor de recomendaciones.

En el proyecto se utilizan 3 algoritmos distintos para realizar las recomendaciones de artistas y conciertos al usuario.

5.1. ALGORITMO BASADO EN CONTENIDO PARA CONCIERTOS SIMILARES

El sistema de recomendación basado en contenido lo que pretende es buscar los conciertos más similares al que se muestra en pantalla y ofrecer al usuario los resultados en la parte inferior. El motivo de este algoritmo es que, el usuario está inspeccionando un concierto que probablemente le interesa o es de su agrado, entonces, se considera apropiado recomendar conciertos muy similares al que el usuario ha seleccionado, teniendo en cuenta las propiedades del ítem seleccionado más importantes.

- **Explicación de Similitud Coseno**

Para comenzar, vamos a explicar qué empleamos para el cálculo de las predicciones de conciertos similares.

Para el cálculo matemático de este algoritmo se emplea la Similitud de Coseno, que es una medida de similitud entre dos vectores distintos de cero de un espacio de producto interno.

Se define para igualar el coseno del ángulo entre ellos, que también es el mismo producto interno de los dos mismos vectores normalizados para que ambos tengan longitud 1. El coseno de 0° es 1, y es menor que 1 para cualquier ángulo en el intervalo $(0, \pi]$ radianes. Por lo tanto, es un juicio de orientación y no de magnitud: dos vectores con la misma orientación tienen una similitud de coseno de 1, dos vectores orientados a 90° entre sí tienen una similitud de 0, y dos vectores diametralmente opuestos tienen una similitud de -1, independientemente de su magnitud. La similitud del coseno se usa particularmente en el espacio positivo, donde el resultado está claramente delimitado en $[0,1]$. El nombre deriva del término "coseno de dirección": en este caso, los vectores unitarios son máximamente "similares" si son paralelos y máximamente "diferentes" si son ortogonales (perpendiculares). Esto es análogo al coseno, que es la unidad (valor máximo) cuando los segmentos subtenden un ángulo cero y cero (no correlacionado) cuando los segmentos son perpendiculares.

La definición matemática es:

Dados dos vectores de atributos, A y B, la similitud del coseno, $\cos(\theta)$, se representa mediante un producto escalar y la magnitud como

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

dónde A_i y B_i son componentes del vector A y B respectivamente.

Entonces lo primero que tenemos que hacer para poder aplicar la similitud es extraer la información necesaria, es decir, los dos vectores a comparar para sacar el coeficiente.

- **Extracción de la información**

Buscamos el concierto que se muestra por pantalla y extraemos los datos como el género, la sala y la ciudad, que van a ser las características por las que saquemos la similitud. De este modo, tendríamos el vector A, que sería para todos los cálculos el mismo vector dado que el objetivo es comparar un concierto con el resto existentes en la base de datos.

Lo que varía en cada interacción es el vector B, que se formará con los datos de cada concierto comparando si las características son iguales o no, con el concierto seleccionado por el usuario. Para formar este vector, con valores $[0, 2]$, se hace uso de las cláusulas COLLECT, CASE, THEN, ELSE... (pertenecientes a uno de los plugins explicados en la parte de la base de datos). Sirven para comparar los valores de ambos conciertos y saber si, son iguales o no, y dar los valores correspondientes al vector B.

Con los dos vectores creados se utiliza la función de la librería GDS que sirve para realizar los cálculos matemáticos entre A y B denominada `gds.similarity.cosine([vector A, vector B])` que toma como parámetros ambos vectores.

- **Filtrado de los resultados obtenidos**

Una vez que tenemos todos los cálculos realizados, filtramos la información obtenida. Solo nos quedamos con aquellos conciertos que tengan una similitud mayor a 0, ya que, en el caso de que sea 0, quiere decir que no hay ninguna característica, entre los dos conciertos, que sea similar. Posterior a este paso, ordenamos los conciertos en base al coeficiente

obtenido y de esta forma, ya tenemos los pasos más similares situados al principio de la lista y serán los que se muestren debajo del concierto elegido por el usuario.

5.2. ALGORITMO DE FILTRADO COLABORATIVO PARA ARTISTAS SIMILARES

El sistema de recomendación de filtrado colaborativo pretende recomendar al usuario artistas en los que otros usuarios que usan la aplicación comparten gustos acerca de determinados artistas. Es decir, usuarios que siguen a artistas en común y sus preferencias son parecidas. De esta forma, hace que los usuarios dispongan de recomendaciones de artistas que les puedan interesar y que no conozcan. El algoritmo utilizado es de filtrado colaborativo, lo que quiere decir que, todos los cálculos que se realizan son, con otros usuarios con los que más parecido tienen. En este caso, se usa el algoritmo de similitud de Jaccard.

- **Explicación del algoritmo de Similitud de Jaccard**

El algoritmo de similitud Jaccard es un algoritmo diseñado para aplicar los conceptos de similitud desarrollados por Paul Jaccard que ayuda a medir las similitudes entre conjuntos. Este se define como el tamaño de la intersección dividida por el tamaño de la unión de dos conjuntos. Este se rige bajo la siguiente fórmula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Este algoritmo desarrollado en el ambiente de trabajo de Neo4j posee dos tipos de ejecución para diferentes tipos de conjuntos de datos. Existe el análisis con base a funciones, que se utiliza normalmente para estudiar la similitud entre pequeños números de conjuntos y adicionalmente se cuenta con la ejecución del algoritmo por procedimientos que está pensada para analizar similitudes en conjuntos de datos grandes.

- **Extracción de la información**

Para este algoritmo, vamos a tener en cuenta los artistas a los que sigue en usuario en activo respecto al resto de usuarios de la aplicación. Es de suma importancia, la relación del usuario con los artistas a los que sigue y gracias a la forma de enlazar la información de Neo4j, toma tanta importancia el algoritmo utilizado y los resultados obtenidos son muy exactos.

La función necesita saber los artistas seguidos por el usuario principal y obtiene los artistas que siguen el resto de usuarios. Guarda el ID de cada artista almacenándolos en dos vectores (un vector para el usuario activo y otro para cada usuario diferente). Por último, lo que se hace es comparar el vector del usuario activo con el resto de los vectores de usuarios y almacena el coeficiente de similitud.

Para esta parte se utiliza la función de la librería GDS `gds.similarity.jaccard([vector A, vector B])` que toma como parámetros ambos vectores.

- **Filtrado de los resultados obtenidos**

Para finalizar, se ordenan los usuarios con mayor coeficiente de similitud y se filtran los datos de forma que se devuelvan aquellos artistas a los que el usuario siga y el usuario activo NO siga. Esto es para evitar que recomiende un artista que ya sigue dado que no tendría sentido.

5.3. ALGORITMO BASADO EN LA ACTIVIDAD DEL USUARIO PARA CONCIERTOS SIMILARES

Este algoritmo reúne partes de los algoritmos anteriores para conseguir recomendaciones más compactas, con más datos y basándose en la actividad del usuario.

La base de este algoritmo es reunir una serie de características para definir el perfil del usuario, saber a que le da mayor prioridad y realizar recomendaciones con certeza de que eso es lo que el usuario busca. Es más completo que los anteriores porque reúne todas las condiciones a las que el usuario le da importancia, busca conocer a cada usuario y extrae las características de los me gusta que den a los conciertos. Tiene en cuenta todos los datos existentes en la aplicación en lo que el gusto del usuario y las características de los conciertos se comparan y busca relaciones entre las preferencias y los gustos, teniendo diferentes valores de importancia.

- **Explicación del algoritmo de Similitud de Pearson**

El coeficiente de correlación de Pearson, también llamado coeficiente de correlación lineal o simplemente coeficiente de correlación, es una medida estadística que indica la relación entre dos variables.

Para calcular el coeficiente de correlación de Pearson entre dos variables se debe dividir la covarianza de dichas variables por la raíz cuadrada del producto de sus varianzas.

De manera que el coeficiente de correlación de Pearson trata de cuantificar la dependencia lineal entre dos variables aleatorias cuantitativas. A priori, valorar numéricamente la correlación entre dos variables es complicado porque resulta difícil determinar qué pareja de variables está más correlacionada entre sí, así pues, el objetivo de coeficiente de correlación de Pearson es dar un valor a la relación entre variables para luego poder comparar entre ellas.

El valor del índice de correlación de Pearson está entre -1 y +1, ambos incluidos. Más abajo veremos cómo se interpreta el valor del coeficiente de correlación de Pearson.

El coeficiente de correlación de Pearson de dos variables estadísticas es igual al cociente entre la covarianza de las variables y la raíz cuadrada del producto de la varianza de cada variable.

Por lo tanto, la fórmula para calcular el coeficiente de correlación de Pearson es la siguiente:

Coeficiente de correlación
de Pearson

$$\rho_{XY} = \frac{Cov(X, Y)}{\sqrt{Var(X) \cdot Var(Y)}}$$

$$-1 \leq \rho_{XY} \leq 1$$

Para determinar el coeficiente de correlación de Pearson es imprescindible que sepas calcular la covarianza entre dos variables y la varianza de una variable.

En estadística, la covarianza es un valor que indica el grado de variación conjunta de dos variables aleatorias. Es decir, la covarianza sirve para analizar la dependencia entre dos variables.

La covarianza es igual al sumatorio de los productos de las diferencias entre los datos de las dos variables y sus respectivas medias partido por el número total de datos.

Covarianza

$$Cov(X, Y) = \frac{\sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})}{n}$$

En estadística, la varianza es una medida de dispersión que indica la variabilidad de una variable aleatoria. La varianza es igual a la suma de los cuadrados de los residuos partido por el número total de observaciones.

Ten en cuenta que como residuo se entiende la diferencia entre el valor de un dato estadístico y la media del conjunto de datos.

En la teoría de la probabilidad, el símbolo de la varianza es la letra griega sigma elevada al cuadrado (σ^2). Aunque también se suele representar como $\text{Var}(X)$, siendo X la variable aleatoria de la cual se calcula la varianza.

En general, la interpretación del valor de la varianza de una variable aleatoria es sencilla. Cuanto más grande sea el valor de la varianza, más dispersos están los datos. Y al revés, cuanto más pequeña sea el valor de la varianza, menos dispersión habrá en la serie de datos. Sin embargo, al interpretar la varianza hay que prestar atención con los valores atípicos (outliers), ya que pueden distorsionar el valor de la varianza.

Varianza

$$\text{Var}(X) = \frac{\sum_{i=1}^n (x_i - \bar{X})^2}{n}$$

$$\text{Var}(X) = \frac{(x_1 - \bar{X})^2 + (x_2 - \bar{X})^2 + \dots + (x_n - \bar{X})^2}{n}$$

En código se resumiría así:

```
function covarianza(x, y) {  
    var cov = 0;  
    var mediaX = mediaAritmetica(x);  
    var mediaY = mediaAritmetica(y);  
    for(var i = 0; i < x.length; i++) {  
        cov += (x[i] - mediaX)*(y[i] - mediaY);  
    }  
    cov = cov / x.length;  
    return cov;  
}  
  
function varianza(x) {  
    var v = 0;  
    var media = mediaAritmetica(x);  
    for (var i = 0; i < x.length; i++) {  
        v += Math.pow(x[i]-media, 2);  
    }  
    v = v / x.length;  
    return v;  
}  
  
function mediaAritmetica(x) {  
    var media = 0;  
    for(var i = 0; i < x.length; i++) {  
        media += x[i];  
    }  
    media = media/x.length;  
    return media;  
}
```

```
// PEARSON  
let pearson = covarianza(vProfileUser, vConcert)/Math.sqrt(varianza(vProfileUser)*varianza(vConcert));
```

- **Creación del perfil del usuario**

Lo primero que realiza el algoritmo es buscar todos los conciertos que le gustan al usuario para poder crear el perfil con las preferencias del usuario y saber a que le da importancia a la hora de dar me gusta a un concierto. Para esta parte, es importante la ciudad en la que vive el usuario y el genero favorito, porque son características importantes ya que son dos datos que conoce la aplicación y se puede determinar si los conciertos que busca seguir el usuario son de su género o sino le importa demasiado el género, y si solo va a conciertos de su ciudad o no le importa ir a conciertos de otros lugares. Por eso son dos características importantes ya que deciden las intenciones de cada usuario. También, se hace uso de varias funciones internas para buscar los artistas a los que sigue el usuario y a las salas que ha valorado.

Ahora entramos en una parte importante del algoritmo. Se crean variables contables para determinar la importancia de cada característica. Todas las puntuaciones se guardan en

estás variables genéricas. Por cada concierto, se compara si el género concuerda con el favorito del usuario, y en ese caso, se obtiene una puntuación de 3 y si no, será 0. Para la ciudad de residencia, se lleva a cabo el mismo proceso, en caso de coincidir la ciudad en la que se celebra el concierto con la que vive el usuario, se suma una puntuación de 3, sino, 0.

Para los horarios, se diferencian 3 principalmente, horario de mañana, tarde y noche comprendidos en:

- Mañana. Desde las 10:00h hasta las 15:00h sin incluir
- Tarde. Desde las 15:00h hasta las 21:00h sin incluir
- Noche. Desde las 21:00h hasta las 4:00h

En el caso del precio, se almacena en una variable la suma de todos los precios de todos los conciertos.

Si el usuario sigue al artista de cada concierto, la puntuación será de 2. Para las valoraciones de las salas, lo que se hace es multiplicar * 0.4 dado que, la puntuación es comprendida entre [0, 5] y la ponderación máxima obtenida será de 2 si la valoración es 5.

Una vez que tenemos todas las sumas hechas de todos los conciertos en base a las características de cada uno a los que el usuario ha dado me gusta, normalizaremos estos datos haciendo la media aritmética.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{x_1 + x_2 + \dots + x_n}{n}$$

Se dividirán las sumas de las puntuaciones del género(si es el mismo o no), la ciudad (si es donde reside el usuario o no), los horarios, los artistas a los que sigue, las valoraciones de la sala entre el número total de conciertos analizados. Para el precio, es un poco diferente ya que, vamos a obtener una media, pero es necesario premiar a los conciertos con un precio inferior a lo que el usuario se podría gastar de media yendo a conciertos. Para esto, se ha ideado una fórmula que satisface estas necesidades (se calcula esta fórmula por cada precio de cada concierto):

PrecioObtenido = precioObtenido + (1 – (0.5/precio medio) * precio concierto actual)

Precio medio: media aritmética del precio de todos los conciertos.

Esto básicamente lo que hace es, por ejemplo, si el precio medio es 10 euros y el precio actual es 5. El cálculo nos daría 0.75, premiando que es está por encima de la media. En el caso de que el precio actual, fuese 10 y la media 10, el valor daría 0.5 y en cualquier caso, que el precio actual sea mayor que la media, daría por debajo de 0.5. Si es 20 el precio actual y 10 la media sería el primer valor en tomar 0, porque hay la misma diferencia de 0 euros a 10euros, que de 10 euros a 20 euros, entonces 0 euros la puntuación sería máxima [1] y siendo 20 euros la puntuación mínima [0], por supuesto, más de 20 euros, serán todo [0].

Por cada concierto se obtiene una puntuación que se suma a una variable para finalmente calcular la media aritmética resultando un valor entre [0,1].

Todos estos datos conforman el vector del perfil del usuario a lo que más preferencias le da el usuario, y el vector a comparar con cada uno de los conciertos.

- **Filtrado y recomendación**

En este caso, lo que hace el algoritmo es buscar los conciertos a los que el usuario no le ha dado me gusta y comparar las características con el perfil del usuario. Se estiman los mismos valores que para el crear el perfil, pero ahora, solo con los datos de cada concierto, no se calculan medias ni nada.

Por último, se calcula la correlación de Pearson y se guardan los baremos obtenidos para ordenarlos y recomendar los valores que más se aproximen a [1].

5.4. OTROS ALGORITMOS INTERESANTES

En el funcionamiento de los algoritmos que voy a mencionar a continuación, es importante destacar que todos estos cálculos y derivaciones son calculadas en base a los datos que hay dentro de la aplicación y se estiman únicamente con la actividad de los usuarios. Es decir, estos datos se extraen de la propia aplicación.

5.4.1. SALAS MEJOR VALORADAS

Las salas mejor valoradas dentro de la aplicación pueden ser interesantes para aquellos usuarios que no sepan acerca de alguna sala, aquí podrán ver la valoración media de cada sala, si es que tiene alguna valoración y sacar sus propias conclusiones.

5.4.2. ARTISTAS CON MÁS SEGUIDORES

Los usuarios podrán comprobar por sí mismos, los artistas más fluorescentes del panorama musical, así de esta forma, podrán descubrir artistas a los que no siguen pero que tienen una gran repercusión.

5.4.3. ARTISTAS CON MÁS CONCIERTOS

Por mera curiosidad, aquí se podrá saber cuáles son los artistas que más conciertos tienen dentro de la gira de cada uno.

5.4.4. SALAS CON MÁS CONCIERTOS

Aquí se podrá ver las salas que más conciertos albergan, algo que resulta curioso ya que, aunque, no tenemos ningún otro algoritmo que recomiende salas dentro de la aplicación web, aquí podremos encontrar aquellas salas con más conciertos, sinónimo de que cuantos más, mejor será la sala.

5.4.5. CONCIERTOS CON MÁS ME GUSTAS

Es interesante mostrar al usuario los conciertos que más me gusta tiene dentro de la aplicación por el resto de los usuarios. Esto evidentemente, cuantos más usuarios usen

la plataforma mejores datos y más concretos serán los resultados. Es una forma de dar a conocer, que es lo que más llama la atención a todos los usuarios de la comunidad.

5.4.6. CONCIERTOS QUE PERTENECEN A LA CIUDAD DEL USUARIO

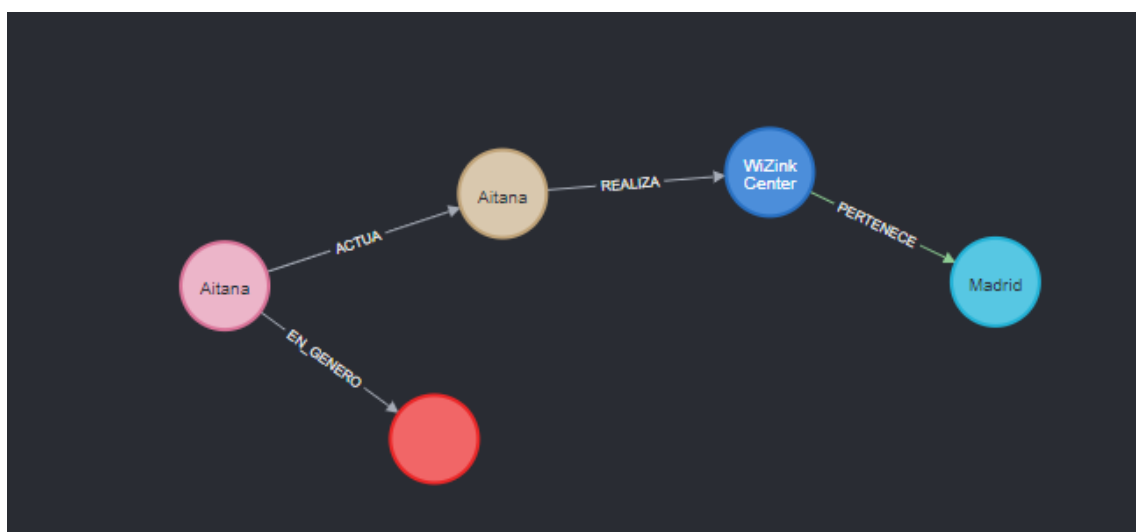
Para esta recomendación, se le propone al usuario conciertos aleatorios que se van a realizar en la ciudad donde reside y sin importar ninguna otra característica. En algún caso, puede resultar curioso porque puede haber un concierto en la ciudad del usuario que aunque no coincida con sus gustos, puede resultarle atractivo, por la importancia del grupo o por ser algún evento único en su ciudad. Y evidentemente, con ninguno de los algoritmos de recomendación, le va a salir en novedades.

6. ANÁLISIS DE RESULTADOS

6.1. ALGORITMO BASADO EN CONTENIDO PARA CONCIERTOS SIMILARES

6.1.1. EJEMPLO DE USO PARA UN CONCIERTO EN CONCRETO

En este caso, vamos a calcular la similitud sobre el siguiente concierto de Aitana (Pop):

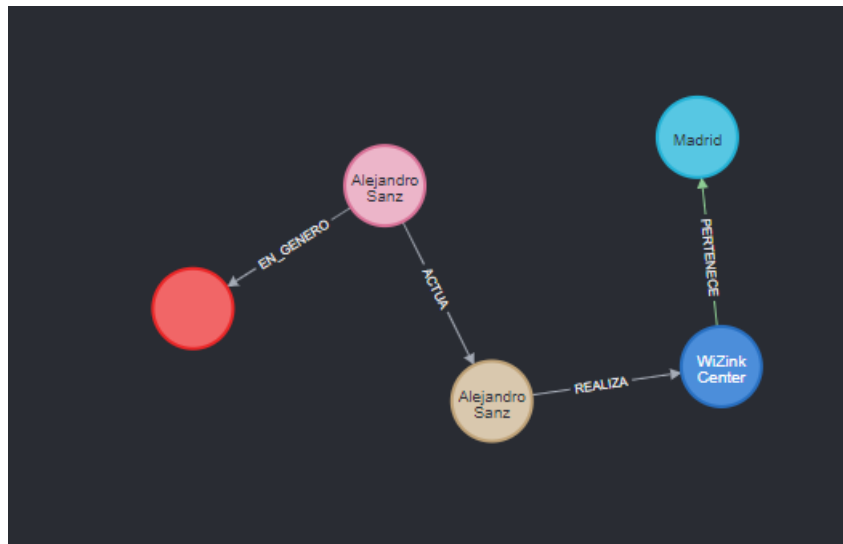


Como ya hemos comentado en la explicación del algoritmo, nos va a buscar los conciertos más similares respecto al mostrado en la imagen basándose en el género, sala y ciudad. En este caso el género es pop, la sala es el Wizink Center y la ciudad es Madrid.

Analizamos los resultados que obtenemos:

	cAux.concertID	cAux.name	cAux.hourStart	cAux.pnce	cAux.date	lAux.city	gAux.genre	aAux.name	rAux.room	coseno
1	7	"Alejandro Sanz"	"19:00"	"10"	"01/12/2022"	"Madrid"	"Pop"	"Alejandro Sanz"	"Wizink Center"	1.0
2	4	"C Tangana"	"22:00"	"30"	"15/11/2022"	"Madrid"	"Urbano latino/Flamenco y muska tradicional"	"C Tangana"	"Wizink Center"	0.8164965
3	2	"La MODA"	"21:00"	"25"	"26/11/2022"	"Madrid"	"Indie"	"La Maravillosa Orquesta Del Alcohol"	"Wizink Center"	0.8164965

Como podemos comprobar, el concierto de Alejandro Sanz tiene los 3 mismos valores que el concierto de Aitana:



Por eso, el resultado de la similitud de Coseno es igual a 1.

En los otros casos, como las preferencias tienen el mismo valor, el resultado de la similitud es el mismo y es que la única diferencia es que el género es diferente.

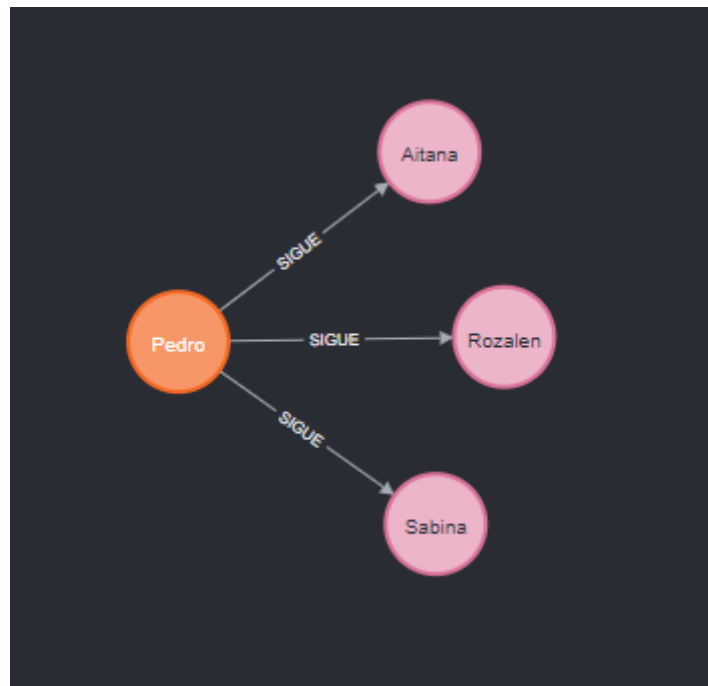
Únicamente recomienda estos tres conciertos porque el resto de los coeficientes de similitud son igual a 0 y como ya dijimos antes, esos conciertos no nos interesa recomendarlos.

6.2. ALGORITMO DE FILTRADO COLABORATIVO PARA ARTISTAS SIMILARES

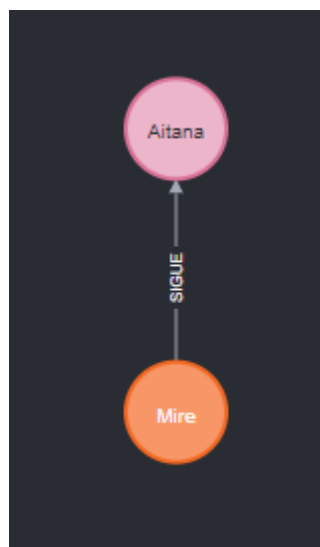
6.2.1. EJEMPLO PARA 3 USUARIOS

Para ver cuál es el rendimiento de este algoritmo he creado 3 usuarios diferentes con una serie de restricciones:

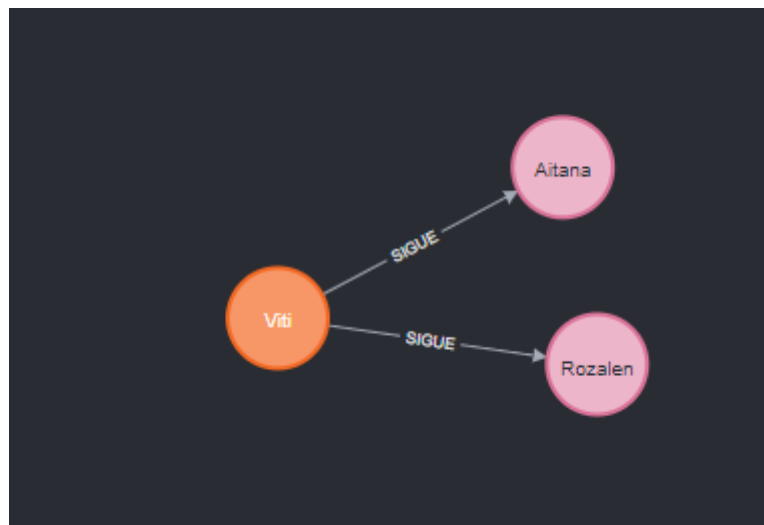
El primer usuario denominado Pedro sigue a 3 artistas: Aitana, Rozalén y Sabina



El segundo usuario llamado Mire, sigue a un solo artista que es Aitana



Y el tercer y último usuario es Viti y sigue a dos artistas: Aitana y Rozalen



Como podemos observar, hay ciertos parecidos porque, aunque no siguen al mismo número de artistas, realmente son solo 3 artistas en total. Con esto buscamos encontrar la similitud entre usuarios y sus gustos.

Si el cálculo del algoritmo lo empleamos para Pedro, ciertamente como hemos comentado al explicarlo, no le recomendaría nada porque el parecido con otros usuarios existe, pero él ya sigue a todos los artistas que los otros usuarios siguen.

En el caso de Viti, tiene cierto parecido con Pedro ya que tienen en común dos artistas y aunque con Mire tiene en común uno solo, es lógico que se le recomiende el otro artista que sigue Pedro pero aún no sigue Viti, porque es probable que le guste Sabina.

Los resultados obtenidos son los que esperábamos:

	(a.name)	g.genre
1	"Sabina"	"Pop/Rock"

En el caso de Mire, los resultados lógicamente serían:

	(a.name)	g.genre
1	"Rozalen"	"Rap/Pop/Flamenco"
2	"Sabina"	"Pop/Rock"

Como primera opción le tiene que recomendar Rozalén porque es la artista que tiene en común con los otros dos usuarios y en último caso, Sabina. Y así es, como el algoritmo nos lo recomienda.

6.3. ALGORITMO BASADO EN LA ACTIVIDAD DEL USUARIO PARA CONCIERTOS SIMILARES

6.3.1. EJEMPLO PARA UN PERFIL

En este algoritmo utiliza muchos más datos y por lo tanto es más preciso con las recomendaciones. En primer lugar, vamos a tener en cuenta el perfil de un usuario que le da mucha preferencia a su género favorito y a la ciudad en la que reside, porque bueno es bastante fiel a su filosofía y no tiene excesivos recursos como para ir a conciertos fuera de la ciudad en la que reside. Únicamente vamos a tener en cuenta dos conciertos, porque son los únicos me gusta que tenga dados. Obtenemos lo siguiente:

	genero fav	generos	ciudad resi	ciudades	horarioM	horarioT	horarioN	precio	val sala	val artista
Perfil(Puntuacion maxima)	3	1	2	1	1	1	1	1	2	2
Concierto1	3	1	2	1	0	0	1	0,5	0	0
Concierto2	3	1	2	1	1	0	0	0	0	0

Como podemos comprobar, el género favorito le da importancia, a la ciudad también y las dos columnas de géneros y de ciudades, sirven para que un usuario le da me gusta a un concierto de una ciudad o genero diferentes a sus preferencias, pueda tener en cuenta con menos ponderación otras ciudades o géneros que también le gustan. En los horarios, vemos que hay 2 diferentes, al precio no le da mucha importancia y a la sala y al artista tampoco.

Haciendo la media aritmética obtenemos el perfil del usuario:

	genero fav	generos	ciudad resi	ciudades	horarioM	horarioT	horarioN	precio	val sala	val artista
Perfil(Puntuacion maxima)	3	1	2	1	1	1	1	1	2	2
Concierto1	3	1	2	1	0	0	1	0,5	0	0
Concierto2	3	1	2	1	1	0	0	0	0	0
(Perfil) Media puntuaciones conciertos	3	1	2	1	0,5	0	0,5	0,25	0	0

Ahora que ya sabemos el perfil que tiene denominado el usuario procedemos a comparar con otros conciertos y así poder recomendarle alguno.

Como podemos deducir, cuantos más me gusta haya dado, mejor definido tendrá el perfil y así mejores recomendaciones se podrán realizar y más críticas.

Realizamos las predicciones sobre dos conciertos. Uno con el mismo género que el favorito del usuario y con la ciudad donde reside y otro sin estas características. Vemos cómo influye en los resultados:

	genero fav	generos	ciudad resi	ciudades	horarioM	horarioT	horarioN	precio	val sala	val artista
Perfil(Puntuacion maxima)	3	1	2	1	1	1	1	1	2	2
Concierto1	3	1	2	1	0	0	1	0,5	0	0
Concierto2	3	1	2	1	1	0	0	0	0	0
(Perfil) Media puntuaciones conciertos	3	1	2	1	0,5	0	0,5	0,25	0	0
Concierto de prediccion 1	3	1	2	1	1	0	0	0	1	0
Concierto de prediccion 2	0	1	0	1	1	0	0	0	1	0

En el que menos se parece la correlación es negativa pero algún aspecto se asemeja aunque poco importante. Por eso lo bueno de este algoritmo, es bastante crítico con los datos que tiene en cuenta y para tener un buen coeficiente ha de tener gran parecido al perfil del usuario. En el primer caso vemos como se asemeja bastante y la correlación de Pearson es casi de 1, pero no llega a serlo.

Para este perfil de usuario se podría, estimar infinidad de opciones y eso que no tenemos tan claro el perfil, porque, repito, cuantos mas conciertos con mas me gusta mucho mejor y más nutrido va a ser el motor de recomendación.

Otro ejemplo distinto con características diferentes del usuario:

	genero fav	generos	ciudad resi	ciudades	horarioM	horarioT	horarioN	precio	val sala	val artista	
Perfil(Puntuacion maxima)	3	1	2	1	1	1	1	1	2	2	
Concierto1	0	1	0	1	0	0	1	0,5	1	2	
Concierto2	3	1	2	1	1	0	0	0	0	2	
(Perfil) Media puntuaciones conciertos	1,5	1	1	1	0,5	0	0,5	0,25	0,5	2	
Concierto de prediccion 1	3	0	2	1	1	0	0	0	1	2	0,744783522
Concierto de prediccion 2	0	1	0	1	1	0	0	0	1	0	-0,107314094

Aquí vemos que el perfil del usuario ya es está en más rebeldía, entonces es más complicado de determinar una similitud aproximado al máximo, ya que, con dos conciertos, hay bastante diferencia entre el valor [3] y el valor [1.5], que al final es lo que se compara aquí.

7. ANALISIS CRITICO(DAFO)



7.1. DEBILIDADES

- Los datos que se utilizan en la base de datos son ficticios por lo que no se puede ver la respuesta de los algoritmos sobre conciertos que vayan a ocurrir en la realidad.
- En el aspecto de la interfaz se le ha dado poca importancia debido a que el periodo de desarrollo del proyecto es muy concreto, entonces se le ha dado más importancia a otros aspectos como los algoritmos.
- Hay algún algoritmo que precisa de pocos datos para hallar el cálculo de similitudes entonces es un aspecto por mejorar.

7.2. AMENAZAS

- Dependencia de los datos otras páginas web. En este caso, dependemos de los datos que suban los artistas a sus páginas web o alguna otra página de información ya que ellos son los que van a tener las fechas de los conciertos.
- Aparición de nuevos competidores. Esto es algo común ya que hoy en día no hay una aplicación completa que recomienda conciertos, pero si hay aplicaciones que buscan realizarse con algoritmos potentes y bien contruidos.

7.3. FORTALEZAS

- Disponemos de un modelo de base de datos escalable ya que cuantos más datos más potenciamos nuestros algoritmos y no importa el tamaño de datos ya que se ha dedicado tiempo a pensar un modelo que satisfaga cualquier modificación o mejora en el futuro ya que se puede aplicar a nuevos algoritmos.
- Existen muchos estudios del INE o incluso, realizados por ayuntamientos para determinar los porcentajes de mujeres o hombres, que van a conciertos, por edades, sexo, a que género van más, todo esto se puede utilizar para hacer hincapié en determinadas recomendaciones.

7.4. OPORTNUIDADES

- Es un mercado sin explotar porque aun no hay aplicaciones que utilicen grandes motores de recomendación para conciertos y festivales, por lo que se puede tener grandes oportunidades de mercado en ese aspecto.
- Adaptación rápida a otros eventos de ocio y aplicable a diferentes celebraciones del mismo sector. Al final lo interesante, es reunir en la misma aplicación eventos que se refieran al ocio y poder tener más donde elegir.
- No hay un límite para dar información a los usuarios y dar datos a los usuarios (mejores conciertos, mejores artistas), lo bueno es que se tiene la oportunidad de estar en constante mejora sacando estadísticas y buenas prácticas del gran volumen de datos.

8. LÍNEAS DE FUTURO

- Aplicación móvil. Para que el sistema de recomendación fuera mas popular y accesible habría que hacer una aplicación móvil. Porque sería mucho más rápido el acceso que en una pagina web. Gozaría de notificaciones avisando al usuario de un concierto cercano.
- Mejora de la base de datos. Este sería un buen punto importante por tratar debido a que la base de datos se nutre de datos aleatorios y que no son ciertos. Así se relanzaría la aplicación, de forma que sería mucho más real y se podría observar la realidad de los algoritmos empleados.
- Crear una agenda con los conciertos. Una de las actualizaciones que tambien es interesante es la de crear una agenda con los conciertos a los que va a asistir el usuario, de esta forma, se puede organizar mucho más fácil, también tiene controlado de alguna forma a los conciertos que ha ido de cada artista, ciudad, fecha. Por otro lado, podría tener constancia de los precios y cuanto se ha gastado. Podría ser de bastante utilidad.
- Mejora de algoritmos y creación de nuevos. Una de las mayores propuestas y que daría un gran salto de calidad sería la creación de un nuevo algoritmo que te realice un tour de conciertos, de los días y el presupuesto fijado por el usuario. En este aspecto, dibujar un mapa y calcular las distancias entre diferentes puntos, ciudades y salas.

9. LECCIONES APRENDIDAS

La verdad que desde un principio fue un trabajo que me gustó y que me inspiró confianza. Fue un trabajo que me motivó y me gustó realizar porque era un descubrimiento de un nuevo mundo del que desconocía y que me ha gustado mucho.

Gracias a ese proyecto he aprendido como funcionan las bases de datos de grafos de conocimiento, que nunca había tratado con ellas. Otra parte importante, es sacar los certificados que he realizado para tener conocimientos y poder aplicarlos. Al final, el saber no ocupa lugar, y los certificados demuestran lecciones aprendidas también, a la vez que este trabajo. Te ayuda a aplicar eso que vas aprendiendo y es una forma ideal para poder aplicar todos estos conceptos nuevos.

El trabajar en algo nuevo y de forma autodidacta, y formativa, con la monitorización y orientación de un tutor, también es una forma muy buena de aprender. Es algo que he desarrollado, y que estoy seguro de que me vendrá bien para proyectos futuros.

Aplicar las matemáticas a los algoritmos, conocer nuevos algoritmos, es otra de las lecciones que me llevo. El aplicar la Inteligencia Artificial al Big Data, un nuevo mundo que antes desconocía y que ahora, por suerte, entiendo como funcionan todos los sistemas de recomendación existentes que usamos día a día.

El uso de las herramientas recomendadas por el profesor como el OSF, el cuaderno virtual donde puedes apuntar cualquier información que te pueda servir útil, incluso para demostrar que has hecho algo o puntualizar sobre alguna tarea. Kaggle, un descubrimiento de datasets que se puede dar uso en cualquier otro ámbito.

10. ANEXOS

- **Framework:** un framework es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software.
- **Dataset:** El término dataset en sí es un término extranjero, un anglicismo, que hemos incorporado a nuestra lengua como un término más en los países hispanohablantes. Su traducción a nuestra lengua sería “conjunto de datos” y es una colección de datos habitualmente tabulada.
- **Web Scrapping:** Web scraping o raspado web, es una técnica utilizada mediante programas de software para extraer información de sitios web.¹ Usualmente, estos programas simulan la navegación de un humano en la World Wide Web ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en una aplicación.

11. BIBLIOGRAFÍAS Y RECURSOS

https://hmong.es/wiki/Cosine_similarity

<https://www.grapheverywhere.com/algoritmo-de-similaridad-de-jaccard/>

<https://www.probabilidadyestadistica.net/varianza/#calculadora-de-la-varianza>

<https://www.probabilidadyestadistica.net/covarianza/#calculadora-de-la-covarianza>

<https://www.probabilidadyestadistica.net/coeficiente-de-correlacion-de-pearson/#calculadora-del-coeficiente-de-correlacion-de-pearson>