



iimas



Proyecto Final

Bases de Datos No Estructuradas

Axel Daniel Malváez Flores : 3183151261

Santiago Licea Becerril : 421048344

Gabriel Zadquiel Peralta Rionda : 317691742

7 de Junio del 2023

Introducción

En la actualidad, es muy común que las personas ya no tengan que asistir al cine, a algún teatro o simplemente rentar una película de *Blockbuster*, basta con simplemente prender tu televisor y seleccionar tu streaming favorito y disfrutar de miles de películas con tan solo pagar una suscripción mensual. No obstante, surge un problema debido a la falta de integración de todas las películas en una sola plataforma digital. Esto provoca una búsqueda tediosa en distintas plataformas, lo cual resulta molesto para los usuarios que simplemente desean relajarse y ver una película sin el estrés que esta situación genera.

Por esta razón, el proyecto final propuesto para la materia de base de datos no estructuradas consiste en desarrollar un sistema de base de datos que aproveche los diferentes manejadores de base de datos que se han visto en clase para la solución de este problema.

La estructuración de nuestras bases de datos es fundamental, ya que busca reunir toda la información de las películas que siempre hemos querido ver en un solo lugar. La base de datos incluirá enlaces a las plataformas donde se puede ver nuestra película favorita, y almacenará la información completa de cada película, como su fecha de lanzamiento, actores, directores y sinopsis. Para lograrlo, se aprovechará la versatilidad y velocidad de una base de datos key-value para almacenar y proporcionar de forma eficiente los links en donde se encuentran las películas y por otro lado utilizaremos una base de datos orientada a objetos, que permite almacenar de manera sencilla los datos de las películas.

Actualmente existen múltiples plataformas para la visualización de películas. El problema de tan amplia variedad de películas es que no todas las películas se encuentran integradas en una única plataforma digital, por lo que es una tarea poco práctica y estresante buscar en qué plataforma se puede reproducir una película. Por ello es que el proyecto final que se ha propuesto para la materia de base de datos no estructuradas es una base de datos que aproveche todos los manejadores de base de datos vistos en clase.

Además, otro desafío es la dificultad de elegir qué película ver cuando no se tiene una en mente. Por ello, otro objetivo del proyecto es emplear una base de datos orientada a grafos, que nos ayudará a encontrar la película perfecta para cada usuario sin tener que agobiarnos con la tarea de selección.

Por último, es importante tener en cuenta la actividad de los usuarios dentro de la plataforma. Al analizar su tiempo de conexión, podremos tomar decisiones sobre cómo mejorar el servicio y planificar el uso de servidores de manera eficiente. En este sentido, una base de datos columnar resulta especialmente útil, debido a su estructura basada en columnas, lo que facilita las consultas específicas a los registros en función de una columna determinada.

Objetivo del proyecto

Utilizar todas las herramientas y conocimientos adquiridos en el curso para poder resolver el problema de tener fácil acceso a toda la información que queramos de una película de una manera eficiente y rápida aprovechando todas las ventajas que nos presentan las bases de datos NoSQL.

Propuesta de solución

Ya que toda la información que se quiere almacenar presenta distintos tipos de estructuras, lo ideal para la planificación de las soluciones será, implementar distintos manejadores de bases de datos, pues cada una de las estructuras de base de datos presenta estructuras particulares que ayudan a resolver las distintas problemáticas que se presentan.

Key-Value Store

Una base de datos Key-Value Store (almacenamiento clave-valor) es una opción adecuada para almacenar enlaces debido a su estructura simple y eficiente. En este tipo de base de datos, los datos se almacenan como pares clave-valor, donde cada clave se asigna a un valor correspondiente.

En el contexto de almacenar enlaces de películas, podemos utilizar los enlaces como claves y asociarlos con los identificadores de las películas como valores. Esto permite un acceso rápido y directo a los enlaces específicos de las películas, ya que se puede buscar y recuperar el valor (en este caso, el enlace) a través de su clave correspondiente de forma eficiente.

Además, las bases de datos Key-Value Store suelen ser altamente escalables y pueden manejar grandes volúmenes de datos sin problemas. Esto es especialmente útil en el caso de almacenar enlaces de películas, ya que puede haber una gran cantidad de enlaces en la base de datos.

Column family

Una base de datos de tipo Column Family (familia de columnas) es adecuada para almacenar la actividad de los usuarios que se conectan a una plataforma debido a su estructura optimizada para consultas por columna y su capacidad de manejar grandes volúmenes de datos.

En el caso de la actividad de los usuarios, se pueden crear column families para diferentes aspectos de la actividad, como tiempo de conexión, acciones realizadas, fechas, etc. Esto permite una consulta rápida y eficiente de datos específicos al acceder directamente a las columnas relevantes.

Además, las bases de datos Column Family son altamente escalables y pueden manejar grandes cantidades de datos, lo que es esencial cuando se almacena la actividad de múltiples usuarios en una plataforma. Además, ofrecen una alta disponibilidad y rendimiento, lo que permite un acceso rápido a los datos de actividad de los usuarios.

Document Store

Una base de datos orientada a documentos es una elección beneficiosa para almacenar toda la información de una película, incluyendo el año de estreno, los actores, los directores y otros elementos relacionados. Esta elección se debe a varias ventajas que proporciona este tipo de base de datos.

En primer lugar, la flexibilidad en la estructura es una de las razones clave. Las bases de datos orientadas a documentos permiten almacenar información en formato de documentos, como JSON o XML. Esto significa que no se requiere un esquema rígido predefinido, lo que otorga la libertad de almacenar diferentes atributos para cada película según sea necesario. No estás limitado a un conjunto fijo de columnas, y puedes agregar, modificar o eliminar campos fácilmente a medida que evolucionan los requisitos de información de las películas.

Además, este enfoque facilita el almacenamiento y la consulta eficientes de la información. Los documentos se almacenan de forma coherente, lo que simplifica la recuperación y actualización de datos relacionados. No es necesario unir tablas o realizar consultas complicadas para acceder a los datos relevantes de una película en particular. Esto agiliza el proceso y mejora el rendimiento de las consultas, especialmente cuando se trabaja con grandes volúmenes de información.

Por último, este enfoque es especialmente útil cuando se trabaja con datos semi-estructurados o no estructurados, como las sinopsis de las películas. Al no tener restricciones estrictas en la estructura de datos, puedes manejar fácilmente información de diferentes formatos y tamaños sin perder la coherencia de la base de datos.

Base de datos orientada a grafos

La presentación de una base de datos orientada a grafos es importante para desarrollar un sistema de recomendación de películas que involucre usuarios, películas, géneros, actores y directores debido a varias razones clave.

En primer lugar, los grafos permiten representar y visualizar relaciones complejas entre entidades. En un sistema de recomendación de películas, es esencial tener en cuenta las conexiones y relaciones entre los diferentes elementos. Por ejemplo, un usuario puede tener preferencias específicas por ciertos géneros o actores, y las películas pueden tener relaciones basadas en los géneros, los actores o los directores involucrados. Un grafo puede mostrar claramente estas relaciones y facilitar la identificación de patrones que ayuden a realizar recomendaciones más precisas.

Además, una base de datos orientada a grafos permite realizar consultas y algoritmos de recomendación más complejos y efectivos. Los grafos permiten realizar búsquedas y análisis de trayectorias para descubrir caminos de conexión entre entidades.

Por último, una base de datos orientada a grafos fomenta la flexibilidad y la evolución del sistema. Puedes agregar y modificar fácilmente relaciones y atributos en el grafo a medida que se descubren nuevas conexiones o se agregan nuevos elementos. Esto permite adaptarse a los cambios en los gustos y preferencias de los usuarios, así como a las tendencias en la industria cinematográfica.

Riak KV

En el caso de la implementación de “Riak KV” se desea almacenar los links en donde encontraremos las películas que se tienen en la base de datos para poder dar un acceso cuando se quiera visualizar una película. Esta implementación se decidió hacer de esta manera ya que la versatilidad de una base de datos Key-Value nos proporciona la facilidad de guardar cualquier objeto en ella.

Toda la base de datos estará almacenada en el Bucket-Type llamado Películas1, donde se le agregara la propiedad en la que la última modificación gana.

En cuanto a la inserción de los elementos, la estructura en que se guardaran las urls serán:

```
buckets/<decada_de_estreno_de_la_pelicula>/keys/<nombre_pelicula><DDMMYYYY>
```

Es decir cada década será un bucket y las llaves de las películas será el nombre de la película junto con su fecha de estreno de la película.

Las siguientes películas serán cargadas a la base de datos:

D1980

1. The shining23051980
 - a. <https://w4.cuevana3.ai/23414/the-shining>

D1990

2. ToyStory213111999
 - a. <https://w4.cuevana3.ai/11522/toy-story-2>
3. Mulan05061998
 - a. <https://www.youtube.com/watch?v=c9IUPbTi7w0>
4. ForestGump06071994
 - a. <https://www.youtube.com/watch?v=luVZ628yBqY>

D2000

5. Cars30062006
 - a. <https://www11.cuevana3.ch/2809/cars>
6. ElHombreAraña17052002
 - a. <https://www11.cuevana3.ch/569/el-hombre-arana>

7. BatmanBegins15062005
 - a. https://youtu.be/rQ_vmBH5qrM
8. TheDarkKnight18072008
 - a. <https://w4.cuevana3.ai/962/the-dark-knight-html>
9. TheHangover05062009
 - a. <https://w4.cuevana3.ai/5052/the-hangoonline>
10. IRobot15072004
 - a. [Watch I, Robot | Star+ \(starplus.com\)](#)
11. Shooter23032007
 - a. [Watch Shooter | Netflix](#)
12. Shrek215052004
 - a. [Watch Shrek 2 | Netflix](#)

D2010

13. LosJuegosDelHambre23032012
 - a. <https://www11.cuevana3.ch/3317/los-juegos-del-hambre>
14. JohnWick30102014
 - a. <https://www11.cuevana3.ch/764/john-wick-otro-dia-para-matar>
15. LaNocheDelDemonio17062010
 - a. <https://www11.cuevana3.ch/3154/la-noche-del-demonio>
16. ElOrigen23072010
 - a. <https://www11.cuevana3.ch/3146/el-origen>
17. CuestionDeTiempo06122013
 - a. <https://www11.cuevana3.ch/3445/cuestion-de-tiempo>
18. ElLoboDeWallStreet10012014
 - a. <https://www11.cuevana3.ch/3407/el-lobo-de-wall-street>
19. BatmanVsSupermanDownOfJustice25032016
 - a. <https://www11.cuevana3.ch/1352/batman-vs-superman-el-origen-de-la-justicia>
20. LaLaLand09122016
 - a. <https://w4.cuevana3.ai/1344/la-la-land-ihy0v>
21. ThreeMetersAboveTheSky03122010
 - a. <https://w4.cuevana3.ai/4516/tres-metros-sobre-el-cielo>
22. TheConjuring210062016
 - a. <https://w4.cuevana3.ai/1036/the-conjuring-2>
23. TheHangoverPartII26052011
 - a. <https://w4.cuevana3.ai/5048/the-hangopart-ii>
24. Interstellar05112014
 - a. [Prime Video: Interstellar](#)
25. GrownUps25062010
 - a. [Watch Grown Ups | Netflix](#)
26. Inception22072010
 - a. [Prime Video: Inception](#)
27. TheImitationGame14112014
 - a. [Watch The Imitation Game | Netflix](#)
28. RapiDOSFuriosos515042011
 - a. [Prime Video: Fast Five](#)

29. Rush27092013

a. [Watch Rush | Netflix](#)

30. JohnWick24102014

a. [Prime Video: John Wick](#)

Ahora para iniciar la inserción de datos, primero se realizan los siguientes comandos para iniciar riak y crear el bucket-type:

Iniciar Riak:

```
sudo riak start
```

Verificar que Riak está activo:

```
riak ping
```

Obtener las propiedades de la base:

```
curl -v http://127.0.0.1:8098/types/default/props
```

```
(base) zadquiel@pc-ubuntu-dm:~$ sudo riak start
[sudo] password for zadquiel:
(base) zadquiel@pc-ubuntu-dm:~$ riak ping
pong
(base) zadquiel@pc-ubuntu-dm:~$ curl -v http://127.0.0.1:8098/types/default/props
* Trying 127.0.0.1:8098...
* Connected to 127.0.0.1 (127.0.0.1) port 8098 (#0)
> GET /types/default/props HTTP/1.1
> Host: 127.0.0.1:8098
> User-Agent: curl/7.84.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Vary: Accept-Encoding
< Server: MochiWeb/2.20.0 WebMachine/1.11.1 (greased slide to failure)
< Date: Tue, 06 Jun 2023 00:50:32 GMT
< Content-Type: application/json
< Content-Length: 503
<
* Connection #0 to host 127.0.0.1 left intact
{"props":{"allow_mult":false,"basic_quorum":false,"big_vclock":50,"chash_keyfun":{"mod":"riak_core_utili
"fun":"chash_std_keyfun"},"dvv_enabled":false,"dw":"quorum","last_write_wins":false,"linkfun":{"mod":"ri
k_kv_wm_link_walker"},"fun":"mapreduce_linkfun"},"n_val":3,"node_confirms":0,"notfound_ok":true,"old_vclo
k":86400,"postcommit":[],"pr":0,"precommit":[],"pw":0,"r":"quorum","repl":true,"rw":"quorum","small_vclo
k":50,"sync_on_write":"backend","w":"quorum","write_once":false,"young_vclock":20}}(base) zadquiel@pc-ub
untu-dm:~$
```

En esta captura se inicia Riak y se verifica su estado.

Creación del Bucket type:

```
riak admin bucket-type create Peliculas1
'{"props":{"last_write_wins":false,"allow_mult":false}}'
```

Activación del Bucket type:

```
riak admin bucket-type activate Peliculas1
```

Mostrar status

```
riak admin bucket-type status Peliculas1
```

```
(base) zadquiel@pc-ubuntu-dm:~$ riak admin bucket-type create Peliculas1 '{"props":{"last_write_wins":false,"allow_mult":false}}'
Peliculas1 created
(base) zadquiel@pc-ubuntu-dm:~$ riak admin bucket-type activate Peliculas1
Peliculas1 has been activated
(base) zadquiel@pc-ubuntu-dm:~$ riak admin bucket-type status Peliculas1
Peliculas1 is active

young_vclock: 20
w: quorum
sync_on_write: backend
small_vclock: 50
rw: quorum
r: quorum
pw: 0
precommit: []
pr: 0
postcommit: []
old_vclock: 86400
notfound_ok: true
node_confirms: 0
n_val: 3
linkfun: {modfun,riak_kv_wm_link_walker,mapreduce_linkfun}
last_write_wins: false
dw: quorum
dvv_enabled: true
chash_keyfun: {riak_core_util,chash_std_keyfun}
big_vclock: 50
basic_quorum: false
allow_mult: false
active: true
claimant: 'riak@127.0.0.1'
(base) zadquiel@pc-ubuntu-dm:~$
```

En la captura anterior estamos creando y activando el bucket-type llamado Peliculas1 en el cual le estamos agregando las propiedades en las que la última modificación de un elemento gana.

Inserción de los datos:

Ejemplo de inserción de la película Los Juegos del Hambre.

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/LosJuegosDelHambre23032012
\
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/3317/los-juegos-del-hambre'
```

Explicación del comando:

El comando curl es una herramienta de línea de comandos que permite realizar solicitudes HTTP. En este caso, el comando curl se utiliza para realizar una solicitud PUT a través del método -XPUT a la URL

<http://localhost:8098/types/Peliculas1/buckets/D2010/keys/LosJuegosDelHambre23032012>.

Partes del comando:

- -XPUT: Especifica que la solicitud es de tipo PUT, lo que significa que se utilizará para insertar o actualizar datos en el servidor.
- <http://localhost:8098/types/Peliculas1/buckets/D2010/keys/LosJuegosDelHambre23032012>: Es la URL de destino a la que se enviará la solicitud PUT. En este caso, se

está insertando el objeto con clave LosJuegosDelHambre23032012 en el bucket D2010 del bucket type Peliculas1.

- -H "Content-Type: text/plain": Especifica el encabezado Content-Type de la solicitud, que indica el tipo de contenido que se está enviando en el cuerpo de la solicitud. En este caso, se establece como text/plain, lo que indica que el contenido es texto sin formato.
- -d 'https://www11.cuevana3.ch/3317/los-juegos-del-hambre': Especifica los datos que se enviarán en el cuerpo de la solicitud. En este caso, se proporciona la dirección URL 'https://www11.cuevana3.ch/3317/los-juegos-del-hambre', que se insertará como el valor del objeto en Riak.

Inserción de todas la películas:

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D1980/keys/TheShining23051980 \
```

```
-H "Content-Type: text/plain" \
-d 'https://w4.cuevana3.ai/23414/the-shining'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D1990/keys/ToyStory213111999 \
```

```
-H "Content-Type: text/plain" \
-d 'https://w4.cuevana3.ai/11522/toy-story-2'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D1990/keys/Mulan05061998 \
```

```
-H "Content-Type: text/plain" \
-d 'https://www.youtube.com/watch?v=c9IUPbTi7w0'
```

```
curl -XPUT
```

```
http://localhost:8098/types/Peliculas1/buckets/D1990/keys/ForestGump06071994 \
```

```
-H "Content-Type: text/plain" \
-d 'https://www.youtube.com/watch?v=luVZ628yBqY'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2000/keys/Cars30062006 \
```

```
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/2809/cars'
```

```
curl -XPUT
```

```
http://localhost:8098/types/Peliculas1/buckets/D2000/keys/ElHombreAraña17052002 \
```

```
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/569/el-hombre-arana'
```

```
curl -XPUT
```

```
http://localhost:8098/types/Peliculas1/buckets/D2000/keys/BatmanBegins15062005 \
```

```
-H "Content-Type: text/plain" \
-d 'https://youtu.be/rQ_vmBH5qrM'
```

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2000/keys/TheDarkKnight18072008 \
-H "Content-Type: text/plain" \
-d 'https://w4.cuevana3.ai/962/the-dark-knight-html'
```

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2000/keys/TheHangover05062009 \
-H "Content-Type: text/plain" \
-d 'https://w4.cuevana3.ai/5052/the-hangoonline'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2000/keys/IRobot15072004 \
-H "Content-Type: text/plain" \
-d 'Watch I, Robot | Star+ (starplus.com)'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2000/keys/Shooter23032007 \
-H "Content-Type: text/plain" \
-d 'Watch Shooter | Netflix'
```

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/LosJuegosDelHambre23032012
\
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/3317/los-juegos-del-hambre'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2010/keys/JohnWick30102014 \
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/764/john-wick-otro-dia-para-matar'
```

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/LaNocheDelDemonio17062010 \
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/3154/la-noche-del-demonio'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2010/keys/EIOrigen23072010 \
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/3146/el-origen'
```

```
curl -XPUT
```

```
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/CuestionDeTiempo06122013 \
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/3445/cuestion-de-tiempo'
```

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/EILoboDeWallStreet10012014 \
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/3407/el-lobo-de-wall-street'
```

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/BatmanVsSupermanDownOfJustice25032016 \
-H "Content-Type: text/plain" \
-d 'https://www11.cuevana3.ch/1352/batman-vs-superman-el-origen-de-la-justicia'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2010/keys/LaLaLand09122016 \
-H "Content-Type: text/plain" \
-d 'https://w4.cuevana3.ai/1344/la-la-land-ihy0v'
```

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/ThreeMetersAboveTheSky0312
2010 \
-H "Content-Type: text/plain" \
-d 'https://w4.cuevana3.ai/4516/tres-metros-sobre-el-cielo'
```

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/TheConjuring210062016 \
-H "Content-Type: text/plain" \
-d 'https://w4.cuevana3.ai/1036/the-conjuring-2'
```

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/TheHangoverPartII26052011 \
-H "Content-Type: text/plain" \
-d 'https://w4.cuevana3.ai/5048/the-hangopart-ii'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2010/keys/Interstellar05112014 \
-H "Content-Type: text/plain" \
-d 'Prime Video: Interstellar'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2010/keys/GrownUps25062010
\
-H "Content-Type: text/plain" \
```

```
-d 'Watch Grown Ups I Netflix'
```

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2010/keys/Inception22072010 \
-H "Content-Type: text/plain" \
-d 'Prime Video: Inception'
```

Inserción de ejemplo:

```
curl -XPUT
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/JohnWick241020
14 \
-H "Content-Type: text/plain" \
-d 'Prime Video: John Wick'
```

Consultas:

Consultar el link de la película “Los juegos del hambre”.

Comando de consulta:

```
curl -XGET
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/LosJuegosDelHambre23032012
```

Resultado de la consulta:

```
(base) zadquiel@pc-ubuntu-dm:~$ curl -XGET http://localhost:8098/types/Peliculas1/buckets/D2010/keys/LosJ
uegosDelHambre23032012
https://www11.cuevana3.ch/3317/los-juegos-del-hambre(base) zadquiel@pc-ubuntu-dm:~$
```

Consultar todas las películas que se encuentran en el bucket D2010.

Comando de la consulta:

```
curl -XGET http://localhost:8098/types/Peliculas1/buckets/D2010/keys?keys=true
```

Resultado de la consulta:

```
(base) zadquiel@pc-ubuntu-dm:~$ curl -XGET http://localhost:8098/types/Peliculas1/buckets/D2010/keys?keys
=true
{"keys":["Inception22072010","LaNocheDelDemonio17062010","JohnWick30102014","LosJuegosDelHambre23032012",
"CuestionDeTiempo06122013","LaLaLand09122016","GrownUps25062010","Interstellar05112014","ElOrigen23072010",
"TheHangoverPartII26052011","ElLoboDeWallStreet10012014","BatmanVsSupermanDownOfJustice25032016","Three
MetersAboveTheSky03122010","TheConjuring210062016"]}(base) zadquiel@pc-ubuntu-dm:~$
```

Comando extras:

Eliminar una película

```
curl -XDELETE
http://localhost:8098/types/Peliculas1/buckets/D2010/keys/LosJuegosDelHambre23032012
```

Actualizar el valor de una película

```
curl -XPUT http://localhost:8098/types/Peliculas1/buckets/D2010/keys/IRobot15072004 \
```

```
-H "Content-Type: text/plain" \
-d
'https://www.starplus.com/es-419/video/24bcd80-0597-4b90-ad5e-6de3b7ba70eb?distributionPartner=google'
```

MongoDB

Para la base de datos de MongoDB, crearemos tres colecciones: Películas, Usuarios, y Géneros. Veamos algunos ejemplos del esqueleto del documento para cada una.

Para Películas:

```
{
  "_id": "Blancanieves01011937", //nombresfechaEstreno
  "nombre": "Blancanieves",
  "fechaEstreno": "01/01/1937",
  "genero": ["id_género_11", "id_género_7"],
  "actores": [
    {
      "nombre": "Adriana Caselotti",
      "fechaNacimiento": "06/05/1916",
      "rol": "Blancanieves (voz)",
      "paisDeOrigen": "Estados Unidos"
    },
    {
      "nombre": "Lucille La Verne",
      "fechaNacimiento": "07/11/1872",
      "rol": "Reina Malvada/Gitana (voz)",
      "paisDeOrigen": "Estados Unidos"
    }
  ],
  "director": {
    "nombre": "David Hand",
    "fechaNacimiento": "23/01/1900",
    "paisDeOrigen": "Estados Unidos"
  },
  "sinopsis": "Blancanieves es una película de animación y fantasía que cuenta la historia de una joven princesa llamada Blancanieves. Después de que su malvada madrastra, la Reina, se entera de que Blancanieves es la más bella del reino, intenta deshacerse de ella. Blancanieves se refugia en una cabaña en el bosque, donde conoce a siete enanitos."
}
```

Como vemos, tendremos relaciones one to few para los géneros y los actores de la película. MongoDB nos permite tener bastante flexibilidad al querer modificar este esqueleto añadiendo más campos. Por ejemplo, podríamos añadir más de un director de darse el caso.

Ahora veamos un ejemplo del esqueleto para la colección Usuarios:

```
{
  "_id": "Juan150520034421050928",
  "nombre": "Juan Perez",
  "email": "juan.perez@gmail.com",
  "fechaNacimiento": "15/05/2003",
  "genero": "Hombre",
  "telefono": 4421050928
}
```

Los id's van a ser los mismos para todas las bases de datos.

Para Generos:

```
{
  "_id": "id_género_1",
  "nombre": "Acción"
}
```

Usaremos estos mismos id's para los géneros en todas las bases de datos. Optamos por hacerlo así ya que hay muchas formas de escribir a cada género, por lo que hacía falta garantizar la estandarización de los nombres.

En la línea de comandos de MongoDB, procedimos a crear las tres colecciones:

```
db.createCollection('Películas')
db.createCollection('Usuarios')
db.createCollection('Generos')
```

Después, con el comando insertMany(), añadimos todos los documentos que creamos a su respectiva colección. Los documentos completos de las colecciones Películas y Usuarios se adjuntarán junto con este reporte.

Para los Generos:

```
db.Generos.insertMany([
  { "_id": "id_género_1", "nombre": "Acción" },
  { "_id": "id_género_2", "nombre": "Aventura" },
  { "_id": "id_género_3", "nombre": "Comedia" },
  { "_id": "id_género_4", "nombre": "Drama" },
  { "_id": "id_género_5", "nombre": "Terror" },
  { "_id": "id_género_6", "nombre": "Ciencia ficción" },
  { "_id": "id_género_7", "nombre": "Fantasía" },
  { "_id": "id_género_8", "nombre": "Romance" },
  { "_id": "id_género_9", "nombre": "Suspense" },
  { "_id": "id_género_10", "nombre": "Documental" },
  { "_id": "id_género_11", "nombre": "Animación" },
  { "_id": "id_género_12", "nombre": "Crimen" },
  { "_id": "id_género_13", "nombre": "Misterio" },
  { "_id": "id_género_14", "nombre": "Western" },
])
```

```
{ "_id": "id_género_15", "nombre": "Guerra" }
]);
```

Para las Películas:

```
db.Peliculas.insertMany([
{
  "_id": "batmanbegins15062005",
  "nombre": "Batman Begins",
  "fechaEstreno": "15/06/2005",
  "genero": ["id_género_1", "id_género_2", "id_género_4"],
  "duracion": 140,
  "actores": [
    {
      "nombre": "Christian Bale",
      "fechaNacimiento": "30/01/1974",
      "rol": "Bruce Wayne / Batman",
      "paisDeOrigen": "Reino Unido"
    },
    {
      "nombre": "Michael Caine",
      "fechaNacimiento": "14/03/1933",
      "rol": "Alfred Pennyworth",
      "paisDeOrigen": "Reino Unido"
    },
    {
      "nombre": "Liam Neeson",
      "fechaNacimiento": "07/06/1952",
      "rol": "Ra's al Ghul",
      "paisDeOrigen": "Irlanda"
    },
    {
      "nombre": "Katie Holmes",
      "fechaNacimiento": "18/12/1978",
      "rol": "Rachel Dawes",
      "paisDeOrigen": "Estados Unidos"
    },
    {
      "nombre": "Gary Oldman",
      "fechaNacimiento": "21/03/1958",
      "rol": "Jim Gordon",
      "paisDeOrigen": "Reino Unido"
    },
    {
      "nombre": "Cillian Murphy",
      "fechaNacimiento": "25/05/1976",
      "rol": "Dr. Jonathan Crane / Espantapájaros",
```

```

        "paisDeOrigen": "Irlanda"
    },
    {
        "nombre": "Morgan Freeman",
        "fechaNacimiento": "01/06/1937",
        "rol": "Lucius Fox",
        "paisDeOrigen": "Estados Unidos"
    }
],
"director": {
    "nombre": "Christopher Nolan",
    "fechaNacimiento": "30/07/1970",
    "paisDeOrigen": "Reino Unido"
},
"sinopsis": "Batman Begins es una película que narra los orígenes del icónico superhéroe Batman. Bruce Wayne (interpretado por Christian Bale) busca venganza por la muerte de sus padres y decide convertirse en Batman para luchar contra el crimen en Ciudad Gótica. Con la ayuda del leal mayordomo Alfred (interpretado por Michael Caine) y el entrenamiento del misterioso Ra's al Ghul (interpretado por Liam Neeson), Batman se enfrenta a los peligrosos villanos que amenazan la ciudad, incluido el espeluznante Espantapájaros (interpretado por Cillian Murphy).",
},
...
]);

```

Para los Usuarios:

```

db.Usuarios.insertMany([
{
    "_id": "Juan150520034421050928",
    "nombre": "Juan Perez",
    "email": "juan.perez@gmail.com",
    "fechaNacimiento": "15/05/2003",
    "genero": "Hombre",
    "telefono": 4421050928
},
{
    "_id": "Maria030419885512093876",
    "nombre": "Maria Gonzalez",
    "email": "maria.gonzalez@hotmail.com",
    "fechaNacimiento": "03/04/1988",
    "genero": "Mujer",
    "telefono": 5512093876
},

```



```
...  
]);
```

Veamos las capturas con las confirmaciones de todos estos comandos:

```
test> db.createCollection('Usuarios')  
{ ok: 1 }  
test> db.createCollection('Películas')  
{ ok: 1 }  
test> db.createCollection('Generos')  
{ ok: 1 }
```

```
acknowledged: true,  
insertedIds: {  
  '0': 'id_género_1',  
  '1': 'id_género_2',  
  '2': 'id_género_3',  
  '3': 'id_género_4',  
  '4': 'id_género_5',  
  '5': 'id_género_6',  
  '6': 'id_género_7',  
  '7': 'id_género_8',  
  '8': 'id_género_9',  
  '9': 'id_género_10',  
  '10': 'id_género_11',  
  '11': 'id_género_12',  
  '12': 'id_género_13',  
  '13': 'id_género_14',  
  '14': 'id_género_15'  
}  
}  
test>
```

```
{
  acknowledged: true,
  insertedIds: {
    '0': 'Juan150520034421050928',
    '1': 'Maria030419885512093876',
    '2': 'Luisa100620007224087631',
    '3': 'Pedro280219953336092837',
    '4': 'Laura200720016647213982',
    '5': 'Diego140319748184092573',
    '6': 'Carolina220820076675089276',
    '7': 'Fernando070519934427109843',
    '8': 'Isabel011119963338790426',
    '9': 'Alejandro250719826642539176',
    '10': 'Santiago180419924421050928',
    '11': 'Isabella120319875512093876',
    '12': 'Gabriel050719797224087631',
    '13': 'Valentina240519863336092837',
    '14': 'Maximiliano140219996647213982',
    '15': 'Valeria080719778184092573',
    '16': 'Sebastian220820056675089276',
    '17': 'Lucia010119854427109843',
    '18': 'Emilio021219963338790426',
    '19': 'Camila050220006642539176',
    '20': 'Gabriel071119925551234567',
    '21': 'Daniel150319995559876543',
    '22': 'Esmeralda250919905552468135',
    '23': 'Santiago120619955553698524',
    '24': 'Azucena030419955557418529',
    '25': 'MiguelHidalgo061119995513290489',
    '26': 'JuanaPerez150919855522187634',
    '27': 'PedroLopez280319925546298102',
    '28': 'AnaGarcia110719885578152098',
    '29': 'LuisHernandez030519965567489123',
    '30': 'CarolinaRuiz191219915589764321',
    '31': 'JorgeSanchez090819825553126789',
    '32': 'MariaLopez220619975532987654',
    '33': 'RicardoGomez140219895598765432',
    '34': 'LauraMartinez271019935509238476'
  }
}
test> 
```

```

{
  acknowledged: true,
  insertedIds: {
    '0': 'batmanbegins15062005',
    '1': 'thedarkknight18072008',
    '2': 'batmanvsuperman25032016',
    '3': 'lalaland09122016',
    '4': 'threemetersabovethesky03122010',
    '5': 'toystory219111999',
    '6': 'TheShining23051980',
    '7': 'theconjuring210062016',
    '8': 'thehangover05062009',
    '9': 'TheHangoverPartII06052011',
    '10': 'losjuegosdelhambre23032012',
    '11': 'elhombrearana03052002',
    '12': 'johnwick24102014',
    '13': 'lanochedeldemonio13042011',
    '14': 'cars09062006',
    '15': 'Mulan06061998',
    '16': 'quisierasermillonario05062023',
    '17': 'forestgump06071994',
    '18': 'cuestiondetiempo10092013',
    '19': 'lobodewallstreet17122013',
    '20': 'interstellar07112014',
    '21': 'irobot16072004',
    '22': 'shooter23032007',
    '23': 'grownups24062010',
    '24': 'inception16072010',
    '25': 'TheImitationGame10122014',
    '26': 'Shrek204062004',
    '27': 'RápidosyFuriosos520052006',
    '28': 'Rush27092013',
    '29': 'Blancanieves01011937'
  }
}
test>

```

Consultas:

Busquemos los nombres de las películas creadas antes del año 2000:

```

db.Peliculas.aggregate([
  {
    $addFields: {
      fechaEstreno: {
        $dateFromParts: {
          year: { $toInt: { $substr: ["$fechaEstreno", 6, 4] } },
          month: { $toInt: { $substr: ["$fechaEstreno", 3, 2] } },
          day: { $toInt: { $substr: ["$fechaEstreno", 0, 2] } }
        }
      }
    }
  }
])

```

```

    },
    {
      $match: {
        fechaEstreno: { $lt: ISODate("2000-01-01") }
      }
    },
    {
      $project: {
        nombre: 1
      }
    }
  ]
})

```

En esta consulta primero tuvimos que convertir las fechas a formato ISO 8601, después con \$match elegimos los documentos con fechas menores a “2000-01-01” y finalmente proyectamos el resultado.

```

test> db.Peliculas.aggregate([
...   {
...     $addFields: {
...       fechaEstreno: {
...         $dateFromParts: {
...           year: { $toInt: { $substr: ["$fechaEstreno", 6, 4] } },
...           month: { $toInt: { $substr: ["$fechaEstreno", 3, 2] } },
...           day: { $toInt: { $substr: ["$fechaEstreno", 0, 2] } }
...         }
...       }
...     }
...   },
...   {
...     $match: {
...       fechaEstreno: { $lt: ISODate("2000-01-01") }
...     }
...   },
...   {
...     $project: {
...       nombre: 1
...     }
...   }
... ])
[
  { _id: 'toystory219111999', nombre: 'Toy Story 2' },
  { _id: 'TheShining23051980', nombre: 'The Shining' },
  { _id: 'Mulan06061998', nombre: 'Mulan' },
  { _id: 'forestgump06071994', nombre: 'Forest Gump' },
  { _id: 'Blancanieves01011937', nombre: 'Blancanieves' }
]

```

Ahora creemos una consulta para ver qué usuarios no son de Estados Unidos:

```

db.Usuarios.find({ paisDeOrigen: { $ne: 'Estados Unidos' } }, { nombre:
1 })

```

Obtuvimos:

```
test> db.Usuarios.find({ paisDeOrigen: { $ne: 'Estados Unidos' } }, { nombre: 1 })
[
  { _id: 'Juan150520034421050928', nombre: 'Juan Perez' },
  { _id: 'Maria030419885512093876', nombre: 'Maria Gonzalez' },
  { _id: 'Luisa100620007224087631', nombre: 'Luisa Martinez' },
  { _id: 'Pedro280219953336092837', nombre: 'Pedro Ramirez' },
  { _id: 'Laura200720016647213982', nombre: 'Laura Sanchez' },
  { _id: 'Diego140319748184092573', nombre: 'Diego Romero' },
  { _id: 'Carolina220820076675089276', nombre: 'Carolina Herrera' },
  { _id: 'Fernando070519934427109843', nombre: 'Fernando Lopez' },
  { _id: 'Isabel011119963338790426', nombre: 'Isabel Torres' },
  { _id: 'Alejandro250719826642539176', nombre: 'Alejandro Mendez' },
  { _id: 'Santiago180419924421050928', nombre: 'Santiago Ramirez' },
  { _id: 'Isabella120319875512093876', nombre: 'Isabella Torres' },
  { _id: 'Gabriel050719797224087631', nombre: 'Gabriel Mendez' },
  { _id: 'Valentina240519863336092837', nombre: 'Valentina Gonzalez' },
  {
    _id: 'Maximiliano140219996647213982',
    nombre: 'Maximiliano Romero'
  },
  { _id: 'Valeria080719778184092573', nombre: 'Valeria Sanchez' },
  { _id: 'Sebastian220820056675089276', nombre: 'Sebastian Herrera' },
  { _id: 'Lucia010119854427109843', nombre: 'Lucia Lopez' },
  { _id: 'Emilio021219963338790426', nombre: 'Emilio Torres' },
  { _id: 'Camila050220006642539176', nombre: 'Camila Mendez' }
]
```

Ahora encontremos a los directores que dirigieron más de una película en nuestra base de datos:

```
db.Peliculas.aggregate([
  {
    $group: {
      _id: "$director.nombre",
      peliculas: { $push: "$nombre" },
      count: { $sum: 1 }
    }
  },
  {
    $match: {
      count: { $gt: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      director: "$_id",
      peliculas: 1
    }
  }
])
```

Obtenemos:

```

test> db.Peliculas.aggregate([
...   {
...     $group: {
...       _id: "$director.nombre",
...       peliculas: { $push: "$nombre" },
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $match: {
...       count: { $gt: 1 }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       director: "$_id",
...       peliculas: 1
...     }
...   }
... ])
[
  {
    peliculas: [ 'Batman Begins', 'The Dark Knight', 'Interstellar', 'Inception' ],
    director: 'Christopher Nolan'
  },
  {
    peliculas: [ 'The Hangover', 'The Hangover Part II' ],
    director: 'Todd Phillips'
  },
  {
    peliculas: [ 'The Conjuring 2', 'La Noche Del Demonio' ],
    director: 'James Wan'
  }
]

```

Después, creamos una consulta para obtener el nombre de las películas cuyo director no es de Estados Unidos:

```

db.Peliculas.aggregate([
  {
    $match: {
      "director.paisDeOrigen": { $ne: "Estados Unidos" }
    }
  },
  {
    $project: {
      _id: 0,
      nombre: 1
    }
  }
])

```

Obtuvimos:

```
test> db.Peliculas.aggregate([
...   {
...     $match: {
...       "director.paisDeOrigen": { $ne: "Estados Unidos" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       nombre: 1
...     }
...   }
... ])
[
  { nombre: 'Batman Begins' },
  { nombre: 'The Dark Knight' },
  { nombre: 'Three Meters Above the Sky (Tres metros sobre el cielo)' },
  { nombre: 'The Conjuring 2' },
  { nombre: 'John Wick' },
  { nombre: 'La Noche Del Demonio' },
  { nombre: 'Quisiera ser millonario' },
  { nombre: 'Cuestión de Tiempo' },
  { nombre: 'Interstellar' },
  { nombre: 'I, Robot' },
  { nombre: 'Inception' },
  { nombre: 'The Imitation Game' },
  { nombre: 'Shrek 2' },
  { nombre: 'Rápidos y Furiosos 5' }
]
```

Finalmente, busquemos a las películas con género `id_género_3` = 'Comedia':

```
db.Peliculas.find({ genero: "id_género_3" }, { nombre: 1 })
```

Obtuvimos:

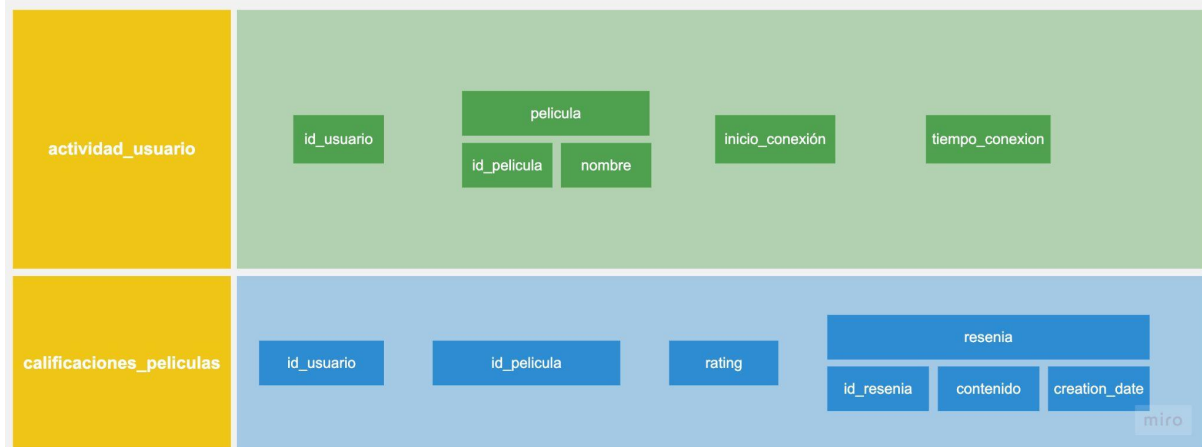
```
test> db.Peliculas.find({ genero: "id_género_3" }, { nombre: 1 })
[
  { _id: 'toystory219111999', nombre: 'Toy Story 2' },
  { _id: 'thehangover05062009', nombre: 'The Hangover' },
  { _id: 'TheHangoverPartII06052011', nombre: 'The Hangover Part II' },
  { _id: 'cars09062006', nombre: 'Cars' },
  { _id: 'grownups24062010', nombre: 'Grown Ups' },
  { _id: 'Shrek204062004', nombre: 'Shrek 2' }
]
```

Cassandra

La base de datos propuesta está diseñada para permitir a los usuarios interactuar con películas y compartir sus opiniones sobre ellas, con el objetivo de facilitar la recomendación de películas a otros usuarios. La base de datos utiliza Apache Cassandra, una base de datos distribuida altamente escalable y de alto rendimiento, ideal para escenarios de escritura intensiva y consultas rápidas.

El diagrama de nuestra base de datos es la siguiente:

Interacción de los usuarios



La base de datos consta de dos tablas principales: **actividad_usuario** y **calificaciones_peliculas**. La tabla **actividad_usuario** contiene información sobre los usuarios que utilizan el sistema, mientras que la tabla **calificaciones_peliculas** registra las calificaciones y reseñas de las películas proporcionadas por los usuarios.

En la tabla **actividad_usuario**, se registran los siguientes atributos: *id_usuario*, que identifica de manera única a cada usuario en el sistema; *inicio_conexion*, que registra la fecha y hora en la que un usuario inicia sesión; y *tiempo_conexion*, que representa la duración de la sesión de un usuario.

La tabla **calificaciones_peliculas** almacena la información relacionada con las calificaciones y reseñas de las películas. Los atributos principales son: *id_usuario* y *id_pelicula*, que identifican al usuario y a la película respectivamente; *rating*, que representa la calificación dada por el usuario a la película; y *resenia*, que contiene la información adicional proporcionada por el usuario en forma de una súper columna.

La súper columna *resenia* en la tabla **calificaciones_peliculas** contiene atributos adicionales, como *id_resenia*, que identifica de manera única cada reseña; *contenido*, que almacena el texto de la reseña; y *creation_date*, que indica la fecha y hora en que se creó la reseña.

Además de proporcionar una plataforma para que los usuarios califiquen y reseñen películas, la base de datos también ofrece la capacidad de realizar análisis basados en los datos registrados. Con la información de *inicio_conexion* y *tiempo_conexion*, es posible realizar análisis de comportamiento de los usuarios, como identificar patrones de uso, determinar la duración promedio de las sesiones de los usuarios y generar recomendaciones personalizadas de películas.

En resumen, la base de datos de Cassandra para interacción y calificación de películas permite a los usuarios compartir sus opiniones sobre películas, proporcionando un sistema de recomendación basado en las calificaciones y reseñas. Además, los datos registrados

ofrecen oportunidades para realizar análisis de comportamiento de los usuarios y mejorar la experiencia general del negocio de películas.

Por lo que comenzaremos a crear nuestra base de datos en Cassandra, la cuál hace uso de keyspaces por lo que inicialmente debemos de crear este espacio de trabajo donde tendremos nuestra información almacenada en columnas:

```
CREATE KEYSPACE UsersInteraction WITH replication = {'class':  
'SimpleStrategy' , 'replication_factor' : 1};  
  
USE usersinteraction ;
```

```
cqlsh:socialnetworkunam> CREATE KEYSPACE UsersInteraction WITH replication = {'class': 'SimpleStrategy' , 'replication_factor' : 1};  
cqlsh:socialnetworkunam>  
cqlsh:socialnetworkunam> USE usersinteraction ;  
cqlsh:usersinteraction> █
```

Posteriormente creamos las súper columnas que necesitaremos: **Pelicula** y **Resenia**.

```
CREATE TYPE pelicula (  
id_pelicula text,  
nombre text);  
CREATE TYPE resenia (  
id_resenia text,  
contenido text,  
creation_date timestamp);
```

```
cqlsh:usersinteraction> CREATE TYPE pelicula (  
... id_pelicula text,  
... nombre text,  
[ ... ];  
cqlsh:usersinteraction> CREATE TYPE resenia (  
... id_resenia text,  
... contenido text,  
... creation_date timestamp  
[ ... ];
```

Ahora, lo que sigue será crear nuestras tablas en la base de datos, estas son las siguientes:

```
CREATE TABLE actividad_usuario (  
id_usuario text,  
pelicula_vista pelicula,  
inicio_conexion timestamp,  
tiempo_conexion duration,  
PRIMARY KEY (id_usuario, inicio_conexion)  
) WITH comment='Tabla de actividad';
```

```
CREATE TABLE calificaciones_peliculas (
  id_usuario text,
  id_pelicula text,
  rating int,
  resenias resenia,
  PRIMARY KEY (id_usuario, id_pelicula)
) WITH comment='Tabla perfiles';
```

```
cqlsh:usersinteraction> CREATE TABLE actividad_usuario (
... id_usuario text,
... pelicula_vista pelicula,
... inicio_conexion timestamp,
... tiempo_conexion duration,
... PRIMARY KEY (id_usuario, inicio_conexion)
... ) WITH comment='Tabla de actividad';
cqlsh:usersinteraction> CREATE TABLE calificaciones_peliculas (
... id_usuario text,
... id_pelicula text,
... rating int,
... resenias resenia,
... PRIMARY KEY (id_usuario, id_pelicula)
... ) WITH comment='Tabla perfiles';
```

Una vez ya creadas nuestras tablas comenzamos a insertar nuestra información tal y como la aplicación que estemos manejando lo indique. En este caso nuestra información es **simulada** y por lo tanto la insertamos toda de una sola carga. Los ID's correspondientes a nuestros usuarios y películas están descritos en nuestra base de datos en Mongo. Esta base de datos en Cassandra está consistente y concuerda con los ID's de dichos usuarios y películas de las demás bases de datos. Dado que tenemos una gran cantidad de objetos insertados, pues a cada usuario (34) le agregamos 5 registros de actividad a continuación veremos la inserción de actividad para un usuario y los demás estarán disponibles en el archivo anexo a este documento (*inserts-cassandra.txt*):

- Tabla *actividad_usuario*:

```
INSERT INTO actividad_usuario (id_usuario, pelicula_vista, inicio_conexion,
tiempo_conexion) VALUES ('Gabriel071119925551234567', {id_pelicula:
'Interstellar05112014', nombre: 'Interstellar'}, '2023-01-01T08:00:00', 2h);
```

```
INSERT INTO actividad_usuario (id_usuario, pelicula_vista, inicio_conexion,
tiempo_conexion) VALUES ('Gabriel071119925551234567', {id_pelicula:
'IRobot15072004', nombre: 'I, Robot'}, '2023-01-05T11:30:00', 1h30m);
```

```
INSERT INTO actividad_usuario (id_usuario, pelicula_vista, inicio_conexion,
tiempo_conexion) VALUES ('Gabriel071119925551234567', {id_pelicula:
'Shooter23032007', nombre: 'Shooter'}, '2023-01-10T15:15:00', 1h45m);
```

```
INSERT INTO actividad_usuario (id_usuario, pelicula_vista, inicio_conexion,
tiempo_conexion) VALUES ('Gabriel071119925551234567', {id_pelicula:
```

```
'GrownUps25062010', nombre: 'Grown Ups'}, '2023-01-15T18:45:00', 1h20m);
```

```
INSERT INTO actividad_usuario (id_usuario, pelicula_vista, inicio_conexion,
tiempo_conexion) VALUES ('Gabriel071119925551234567', {id_pelicula:
'Inception22072010', nombre: 'Inception'}, '2023-01-20T22:00:00', 1h50m);
```

- Tabla *calificaciones_peliculas*:

```
INSERT INTO calificaciones_peliculas (id_usuario, id_pelicula, rating, resenias)
VALUES ('Gabriel071119925551234567', 'TheDarkKnight18072008', 9, {id_resenia:
'resenia1', contenido: 'Me encantó esta película. La actuación de Heath Ledger
como el Joker fue increíble.', creation_date: '2023-01-10 12:00:00'}));
```

Dado que esta es la base de datos que nos ayudará a ver cómo es el comportamiento de los usuarios de nuestra plataforma de películas. Queremos conocer a qué hora del día y cuáles son los días que más se conectan, así mismo con las reseñas queremos ver cuáles son las películas que tienen más y menos aprobación por parte del público usuario de la plataforma. Este análisis lo realizaremos utilizando un Jupyter Notebook en Python. Primero lo que haremos será exportar las tablas de nuestra base de datos en Cassandra a un archivo con extensión **.csv** :

- Tabla **actividad_usuario** :

```
cqlsh:usersinteraction> COPY usersinteraction.actividad_usuario TO '/Users/danielmalvaez/Downloads/data-users-activity.csv' WITH HEADER = true;
Using 7 child processes

Starting copy of usersinteraction.actividad_usuario with columns [id_usuario, inicio_conexion, pelicula_vista, tiempo_conexion].
Processed: 175 rows; Rate: 1741 rows/s; Avg. rate: 427 rows/s
175 rows exported to 1 files in 0.427 seconds.
cqlsh:usersinteraction>
```

- Tabla **calificaciones_peliculas** :

```
cqlsh:usersinteraction> cqlsh:usersinteraction> COPY usersinteraction.calificaciones_peliculas TO '/Users/danielmalvaez/Downloads/data-movie-ratings.csv' WITH
Using 7 child processes

Starting copy of usersinteraction.calificaciones_peliculas with columns [id_usuario, id_pelicula, rating, resenias].
Processed: 35 rows; Rate: 715 rows/s; Avg. rate: 113 rows/s
35 rows exported to 1 files in 0.330 seconds.
```

Ahora, iniciaremos este análisis con algunas consultas tanto en Cassandra como en Python (visualización), para realizar alguna exploración de nuestros datos:

- Todas las películas cuyo rating sea 10:

```
SELECT * FROM calificaciones_peliculas WHERE rating = 10 ALLOW FILTERING;
```

```
cqlsh:usersinteraction> SELECT * FROM calificaciones_peliculas WHERE rating = 10 ALLOW FILTERING;

id_usuario | id_pelicula | rating | resenias
-----|-----|-----|-----
Luisa100620087224087631 | TheDarkKnight18072008 | 10 | {id_resenia: 'resenia18', contenido: '¡Una película de Batman increíble! La actuación de Heath Ledger como el Joker es fenomenal.', creation_date: '2023-06-06 06:00:00.000000+0000'}
CarolinaRuiz191219915589764321 | ToyStory213111999 | 10 | {id_resenia: 'resenia11', contenido: 'Amo esta película. Es un clásico de la infancia.', creation_date: '2023-01-04 23:55:00.000000+0000'}
MiguelHidalgo06111995513290409 | JohnWick24102014 | 10 | {id_resenia: 'resenia6', contenido: '¡Esta película es impresionante! Las escenas de acción son increíbles.', creation_date: '2023-01-06 02:10:00.000000+0000'}
Camila050220086642539176 | Inception22072010 | 10 | {id_resenia: 'resenia35', contenido: 'Inception es una obra maestra del cine. La trama es intrigante y te mantiene al borde de tu asiento en todo momento. Las actuaciones son excelentes y los efectos visuales son impresionantes. Sin duda, una de mis películas favoritas.', creation_date: '2023-06-06 06:00:00.000000+0000'}
Gabriel050719797224087631 | TheDarkKnight18072008 | 10 | {id_resenia: 'resenia28', contenido: 'Una obra maestra del cine de superhéroes. Heath Ledger como el Joker es increíble.', creation_date: '2023-06-06 06:00:00.000000+0000'}
```

- Ahora queremos saber cuáles han sido las actividades que hemos realizado los primeros días del año:

```
SELECT * FROM actividad_usuario WHERE inicio_conexion < '2023-01-02'
ALLOW FILTERING;
```

```
cqlsh:usersinteraction> SELECT * FROM actividad_usuario WHERE inicio_conexion < '2023-01-02' ALLOW FILTERING;
```

id_usuario	inicio_conexion	pelicula_vista	tiempo_conexion
Valentina240519863336092837	2023-01-02 01:30:00.000000+0000	(id_pelicula: 'TheHangover05062009', nombre: 'The Hangover')	2h5m30s
CarolinaRuiz191219915589764321	2023-01-01 16:30:00.000000+0000	(id_pelicula: 'Interstellar05112014', nombre: 'Interstellar')	1h30m10s
Daniel150319995559876543	2023-01-01 15:30:00.000000+0000	(id_pelicula: 'Interstellar05112014', nombre: 'Interstellar')	2h15m
MiguelHidalgo061119995513290400	2023-01-01 14:30:00.000000+0000	(id_pelicula: 'Shrek215052004', nombre: 'Shrek 2')	2h30m15s
Gabriel071119925551234567	2023-01-01 14:00:00.000000+0000	(id_pelicula: 'Interstellar05112014', nombre: 'Interstellar')	2h
JorgeSanchez0908019825553126789	2023-01-01 14:15:00.000000+0000	(id_pelicula: 'ToyStory21311999', nombre: 'Toy Story 2')	1h30m45s
MariaLopez220619975532987654	2023-01-01 15:30:00.000000+0000	(id_pelicula: 'Interstellar05112014', nombre: 'Interstellar')	2h15m30s
Sebastian220820056675089276	2023-01-02 00:30:00.000000+0000	(id_pelicula: 'LaLaLand09122016', nombre: 'La La Land')	2h8m
RicardoGomez140219895598765432	2023-01-01 14:00:00.000000+0000	(id_pelicula: 'Inception22072010', nombre: 'Inception')	2h30m45s
JuanPerez150919855522187634	2023-01-01 15:00:00.000000+0000	(id_pelicula: 'IRobot15072004', nombre: 'I, Robot')	2h30m15s
Gabriel050719797224087631	2023-01-02 02:15:00.000000+0000	(id_pelicula: 'IRobot15072004', nombre: 'I, Robot')	1h53m30s

- Queremos obtener aquellos ID's de los usuarios que han hecho reseñas con un rating de 8 o más

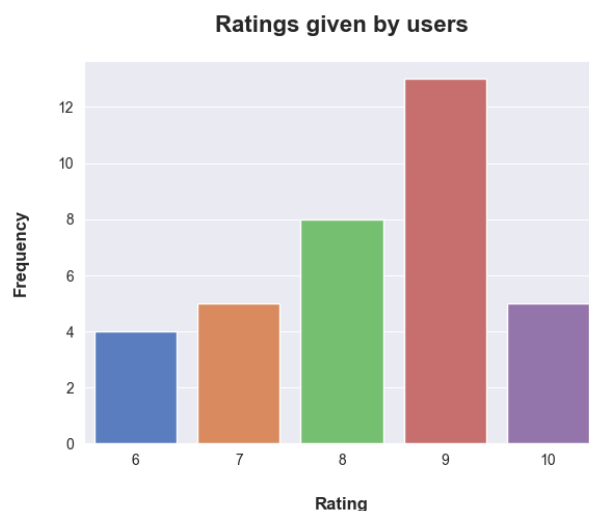
```
SELECT id_usuario FROM calificaciones_peliculas WHERE rating > 8 ALLOW
FILTERING;
```

```
cqlsh:usersinteraction> SELECT id_usuario FROM calificaciones_peliculas WHERE rating > 8 ALLOW FILTERING;
```

id_usuario
Valentina240519863336092837
Luisa100620007224087631
AnaGarcia110719805578152098
Alejandro250719826642539176
CarolinaRuiz191219915589764321
Valeria080719778184092573
MiguelHidalgo061119995513290400
Gabriel071119925551234567
JorgeSanchez0908019825553126789
Emilio021219963338790426
Camila050220006642539176
Isabella120319875512093076
Maria030419885512093076
PedroLopez220319925546298102
Diego140319748184092573
Sebastian220820056675089276
Azuena030419955557418529
Gabriel050719797224087631

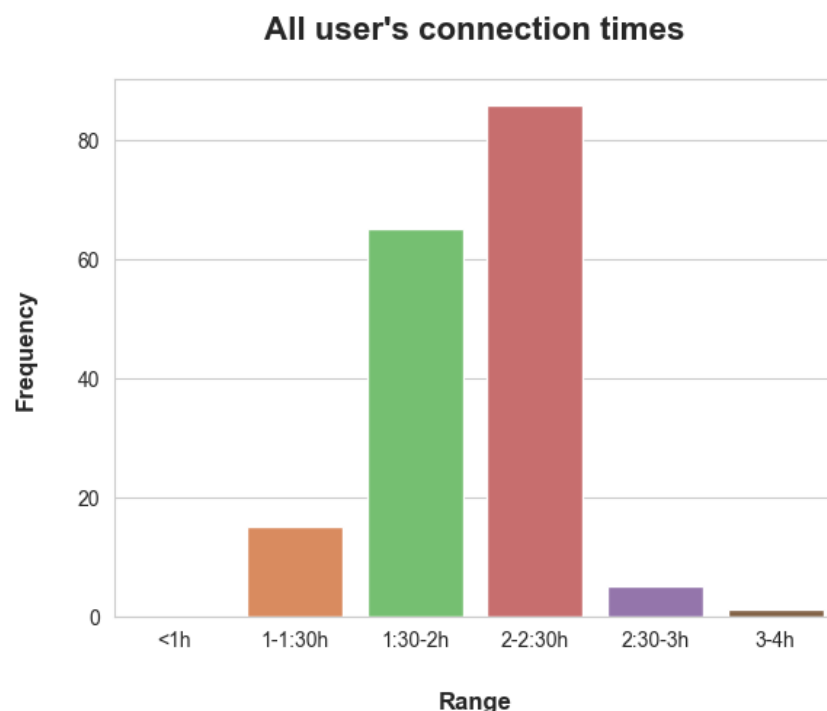
(18 rows)

Ya que realizamos algunas consultas en la terminal de Cassandra, lo que haremos posteriormente será realizar analítica descriptiva de ambas tablas y tratar de sacar información valuable sobre nuestros usuarios utilizando las librerías **Pandas**, **Matplotlib** y **Seaborn** de Python (para la graficación y visualización). Queremos ver cómo se distribuyen los *ratings* que nuestros usuarios le han dado a las películas para ver qué tan bien o mal están, esta información fue obtenida de la tabla **calificaciones_peliculas**:



Con esta información, nos podemos dar cuenta de que la mayoría de las personas que han publicado reseñas sobre nuestras películas, han sido reseñas con un rating bastante alto. Tenemos que tener en cuenta también que las reseñas que tenemos hasta este mes, ninguna ha sido menor a un rating de 6.

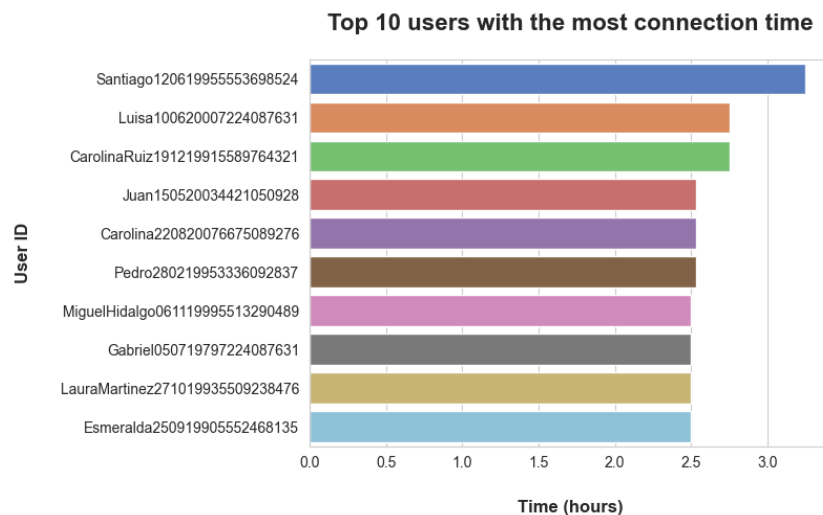
Posteriormente, en nuestra tabla **actividad_usuario** tenemos información importante tal y como lo es el tiempo de conexión. Estos datos son importantes si queremos realizar un análisis de qué tanto tiempo nuestros clientes están en la plataforma viendo alguna película, esto nos ayuda para poder clasificar a nuestros clientes más recurrentes y los no tan recurrentes y poder generar estrategias que impacten positivamente en aquellos que no están viendo tanto tiempo las películas y fidelizar a aquellos que sí pasen mucho tiempo.



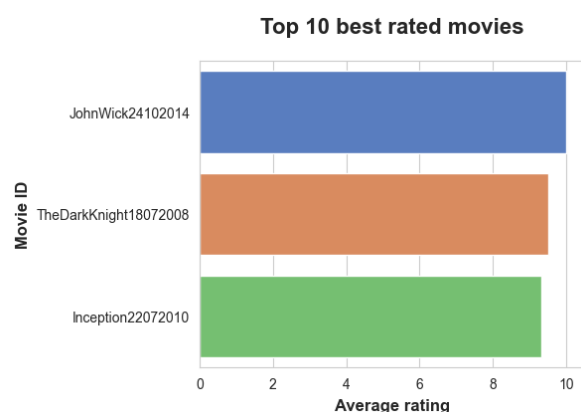
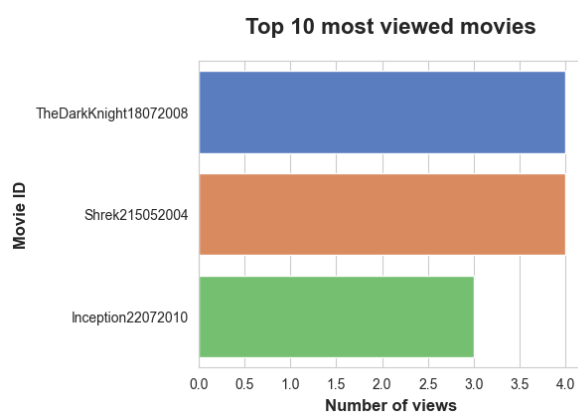
Aquí podemos ver qué tanto nuestros usuarios pasan el tiempo en nuestra plataforma. Según nuestros datos, la duración promedio de las películas en nuestra plataforma es de aproximadamente 2 a 2:30 horas. Por lo tanto, en general, nuestros usuarios pasan entre 2 y 2:30 horas cada vez que se conectan. Sin embargo, también hemos observado que hay usuarios que tienen una preferencia por ver películas más cortas y solo pasan entre 1 y 1:30 horas en la plataforma. Aunque este grupo de usuarios es menor en comparación con los que pasan más tiempo, todavía representan una parte significativa de nuestra base de usuarios. Afortunadamente, ninguno de nuestros usuarios pasa menos de 1 hora en la plataforma, lo que indica que todos encuentran contenido de su interés y disfrutan de la experiencia la plataforma ofrece. Por otro lado, también hemos notado que algunos usuarios son más dedicados y pasan más tiempo en nuestra plataforma, pues algunos llegan a pasar de 3 a 4 horas disfrutando de nuestro contenido. Aunque este grupo es más reducido, son usuarios muy comprometidos y leales a nuestra plataforma. En resumen, la mayoría de nuestros usuarios pasan entre 2 y 2:30 horas en nuestra plataforma debido a la duración promedio de las películas. Sin embargo, también tenemos usuarios que pasan menos tiempo (1-1:30 horas)

y algunos que pasan más tiempo (3-4 horas), lo que demuestra la diversidad de preferencias y el atractivo de nuestro contenido para diferentes audiencias.

Ahora, conocer aquellos 10 usuarios que pasan más tiempo dentro de la plataforma es de gran utilidad pues nos brinda información valiosa para mejorar la experiencia del usuario, optimizar la recomendación de contenido y fomentar la retención de usuarios. Esto ayuda a adaptar la plataforma a las preferencias de los usuarios más comprometidos y a impulsar el crecimiento y el éxito de la misma.

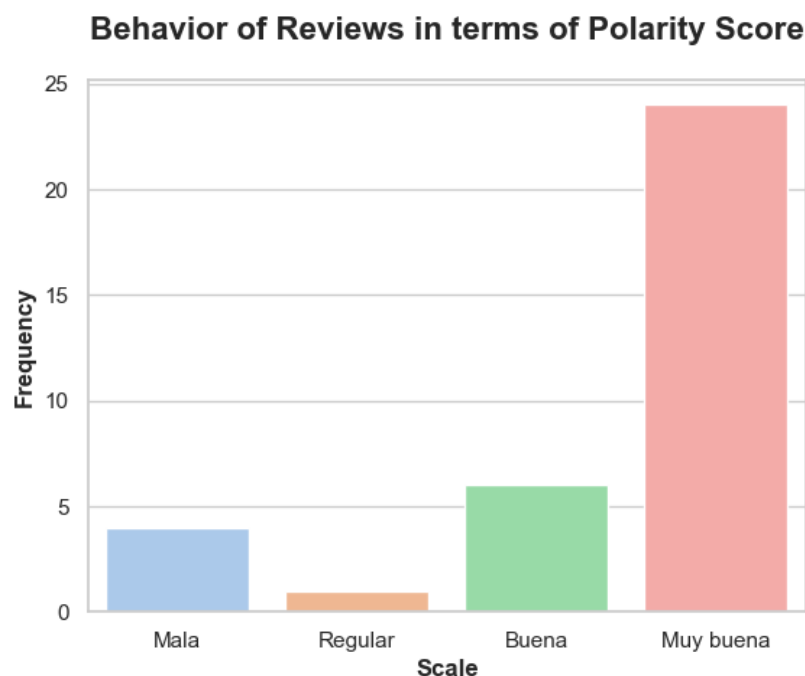


Por otro lado tenemos que saber qué películas son las más vistas te proporciona información importante para tomar decisiones sobre qué contenido agregar o promocionar en tu plataforma, pues las películas más vistas y mejor calificadas pueden servir como una referencia para mejorar el sistema de recomendación de contenido en tu plataforma. De igual manera las películas más vistas y mejor calificadas pueden ser utilizadas como una herramienta de marketing eficaz. Finalmente en materia económica, el interés y éxito de las películas populares puede ayudar a fortalecer la posición y obtener acceso a un catálogo de películas más amplio y atractivo con algunos proveedores.



Finalmente, realizaremos un análisis de sentimientos basado en las reseñas de los usuarios en tu plataforma de películas es de vital importancia para comprender el nivel de satisfacción

de tus clientes y evaluar la calidad de las películas que proyectas. Este enfoque te permite obtener información valiosa sobre las emociones y opiniones de tus usuarios, lo que te brinda una perspectiva más completa sobre su experiencia y preferencias cinematográficas. En resumen, el análisis de sentimientos basado en las reseñas de los usuarios en tu plataforma de películas te permite obtener información valiosa para comprender mejor a tus usuarios, mejorar su experiencia y tomar decisiones más estratégicas. Al integrar esta metodología, estarás en una posición sólida para brindar un servicio excepcional, satisfaciendo las necesidades y expectativas de tus usuarios y fortaleciendo tu posición en el mercado cinematográfico. Para este análisis utilizaremos las siguientes librerías de Python: **re** (que nos permite usar regex para limpiar los strings), **unidecode** (que nos ayuda a limpiar los strings y dejarlos en este formato), **deep_translator** (que nos ayuda a traducir oraciones), **NLTK** (que es la librería principal que nos ayudará a puntuar las palabras en función de su positividad, negatividad y neutralidad) e igualmente **Pandas**, **Matplotlib** y **Seaborn** (para la graficación y visualización):



Los resultados muestran una predominancia de opiniones positivas. Con un total de 24 reseñas clasificadas como "muy buena" y 6 como "buena", es evidente que la mayoría de los usuarios están altamente satisfechos con las películas que proyectas. Esta es una excelente noticia, ya que indica que se está ofreciendo contenido que cumple con las expectativas y genera una experiencia positiva para tus usuarios. En general, los resultados del análisis de sentimientos refuerzan la importancia de continuar ofreciendo películas que generen emociones positivas y una experiencia satisfactoria para tus usuarios. Al aprovechar estos conocimientos, se puede mantener la calidad de las proyecciones, adaptar la oferta a las preferencias de tus usuarios y trabajar en las áreas que necesitan mejoras. Esto fortalecerá la reputación de la plataforma, fomentará la lealtad de los usuarios existentes y atraerá a nuevos usuarios en busca de películas que satisfagan sus necesidades y expectativas.

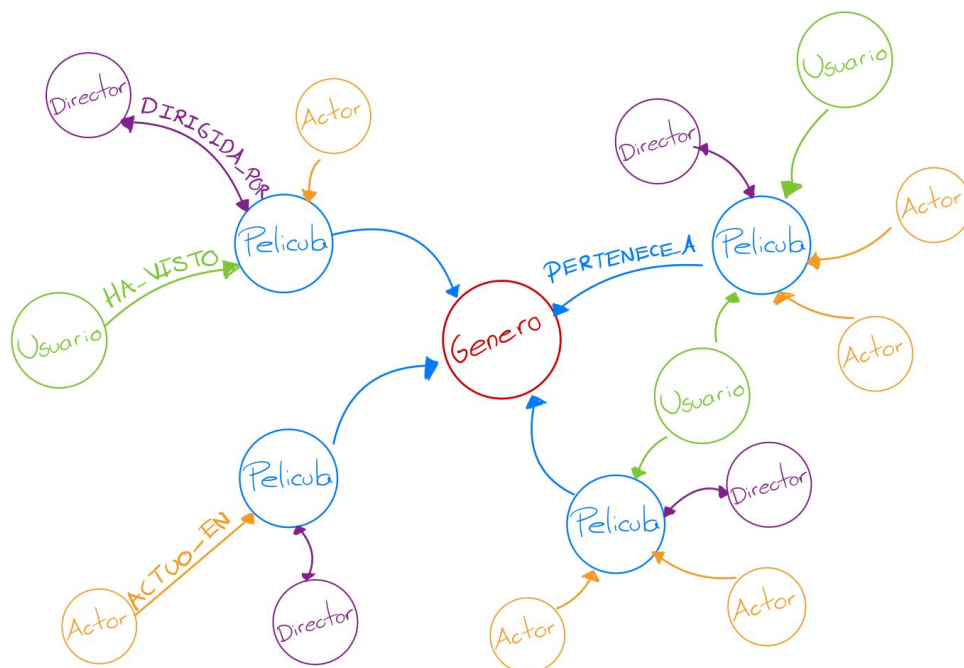
Neo4j

Guardar información sobre películas, directores, géneros de las películas y la visualización de películas por los usuarios en una base de datos orientada a grafos tiene un propósito en específico ya que lo más importante de esta información es almacenar la relación que existe entre los objetos anteriormente mencionados. Por ello es que una de las ventajas es la flexibilidad que tiene esta base para almacenar todo tipo de relaciones. Las bases de datos orientadas a grafos permiten representar de manera eficiente las relaciones complejas entre películas, directores y géneros mediante nodos y relaciones en un grafo. Esto facilita el almacenamiento y consulta de la información relacionada, lo que resulta en un modelo de datos más intuitivo y coherente.

Además, las consultas en una base de datos orientada a grafos son eficientes. Neo4j está optimizado para recorrer y consultar grafos grandes de manera rápida y eficiente. Esto permite realizar consultas complejas y obtener resultados de manera ágil al seguir las relaciones en el grafo. Por ejemplo, se pueden obtener recomendaciones de películas similares o encontrar películas relacionadas a través de los directores o géneros.

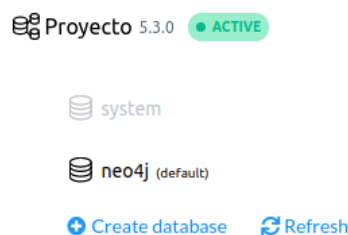
Las bases de datos orientadas a grafos son especialmente útiles para el análisis de recomendación. Al aprovechar algoritmos de recomendación basados en grafos, se pueden descubrir patrones de preferencias y relaciones entre películas, directores y géneros. Esto permite ofrecer recomendaciones personalizadas a los usuarios, mejorando su experiencia.

Gracias a las ventajas de la base de datos hemos decidido realizar el siguiente diseño de la base de datos:



El diseño anterior nos brinda la oportunidad de explorar las interconexiones entre las películas, ya sea a través del género compartido, el director común o la presencia de ciertos actores. Esta capacidad resulta crucial, ya que nos permite visualizar las características distintivas de nuestro conjunto de películas. Al comprender estas relaciones, podremos determinar qué tipo de películas podrían resultar atractivas para nuestros usuarios. Las conexiones entre las películas destacarán las interrelaciones que se presentan entre ellas, revelando valiosos patrones y permitiéndonos ofrecer recomendaciones más precisas y personalizadas las cuales nos harían destacar ante la competencia.

Para la implementación de la base de datos, creamos la base de datos en Neo4J llamada 'Proyecto', la cual nos ayudará a almacenar todas las relaciones que queremos sobre las películas.



Para la creación de las películas se siguió la siguiente estructura de creación de los nodos:

```
CREATE (p1:Pelicula {id: 'Interstellar05112014', nombre: 'Interstellar'})
CREATE (p2:Pelicula {id: 'IRobot15072004', nombre: 'I, Robot'})
```

Para la creación de los nodos de géneros:

```
CREATE (g1:Genero {nombre: 'Acción'})
CREATE (g2:Genero {nombre: 'Aventura'})
```

Para la creación de los nodos de los actores:

```
CREATE (a1:Actor {nombre: 'Matthew McConaughey'})
CREATE (a2:Actor {nombre: 'Anne Hathaway'})
```

Para la creación de los nodos de directores:

```
CREATE (d1:Director {nombre: 'Christopher Nolan'})
CREATE (d2:Director {nombre: 'Alex Proyas'})
```

Para la creación de los nodos de los usuarios:

```
CREATE (u1:Usuario {ID: 'Juan150520034421050928', nombre: 'Juan
```

```
Perez'})  
CREATE (u2:Usuario {ID: 'Maria030419885512093876', nombre: 'Maria  
Gonzalez'})
```

Ahora con respecto a las relaciones que se presentan entre los nodos, tenemos la creación de las aristas 'DIRIGIDA_POR' dada entre películas y directores:

```
CREATE (p1)-[:DIRIGIDA_POR]->(d1)  
CREATE (p2)-[:DIRIGIDA_POR]->(d2)
```

Para la creación de las aristas 'PERTENECE_A' dado entre películas y géneros:

```
CREATE (p1)-[:PERTENECE_A]->(g4)  
CREATE (p1)-[:PERTENECE_A]->(g6)
```

Para la creación de las aristas 'ACTUO_EN' dado entre actores y películas

```
CREATE (a79)-[:ACTUO_EN]->(p10)  
CREATE (a62)-[:ACTUO_EN]->(p24)
```

Para la creación de las aristas 'HA_VISTO' dado entre usuarios y películas

```
CREATE (u1)-[:HA_VISTO]->(p8)  
CREATE (u1)-[:HA_VISTO]->(p18)
```

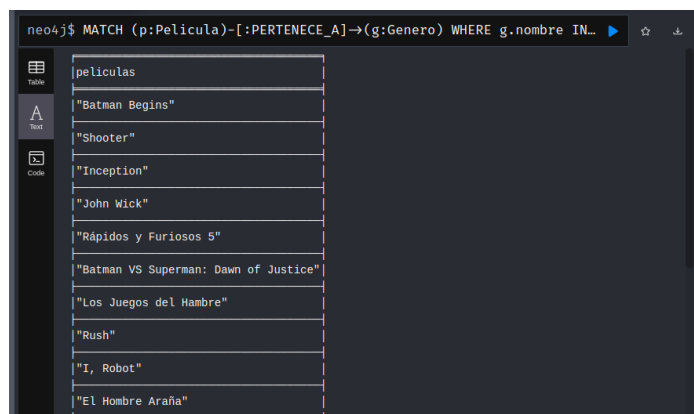
Consultas

Visualizar toda la base de datos

```
MATCH (n) - [r] -> (m) RETURN n,r,m
```

Películas que pertenezcan a los géneros Acción y Aventura.

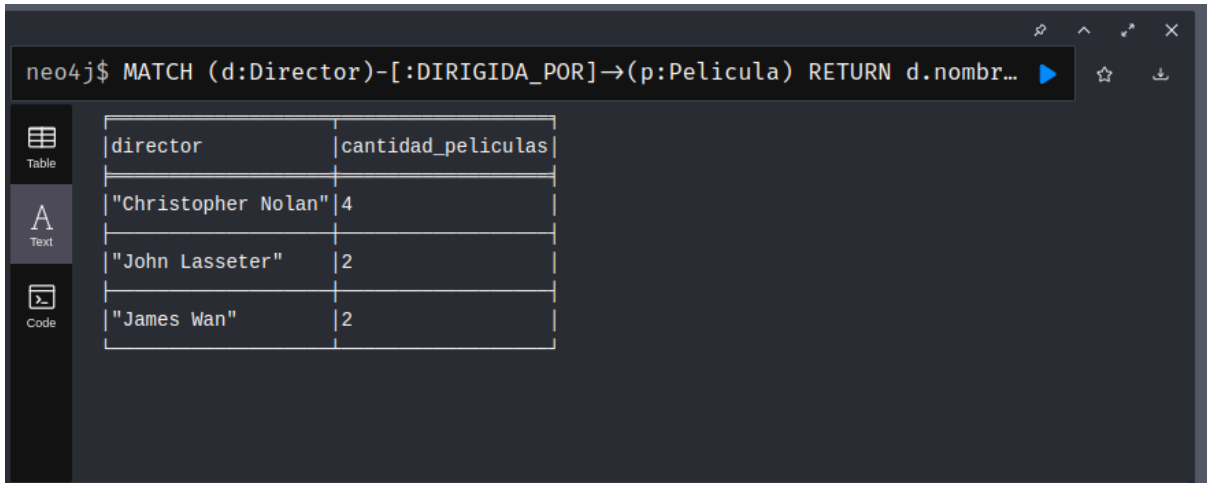
```
MATCH (p:Película)-[:PERTENECE_A]->(g:Genero)  
WHERE g.nombre IN ['Aventura', 'Acción']  
RETURN p.nombre AS peliculas
```



peliculas
"Batman Begins"
"Shooter"
"Inception"
"John Wick"
"Rápidos y Furiosos 5"
"Batman VS Superman: Dawn of Justice"
"Los Juegos del Hambre"
"Rush"
"I, Robot"
"El Hombre Araña"

- **Director con más películas dirigidas:**

```
MATCH (d:Director)-[:DIRIGIDA_POR]->(p:Pelicula)
RETURN d.nombre AS director, count(p) AS cantidad_peliculas
ORDER BY cantidad_peliculas DESC
LIMIT 3
```

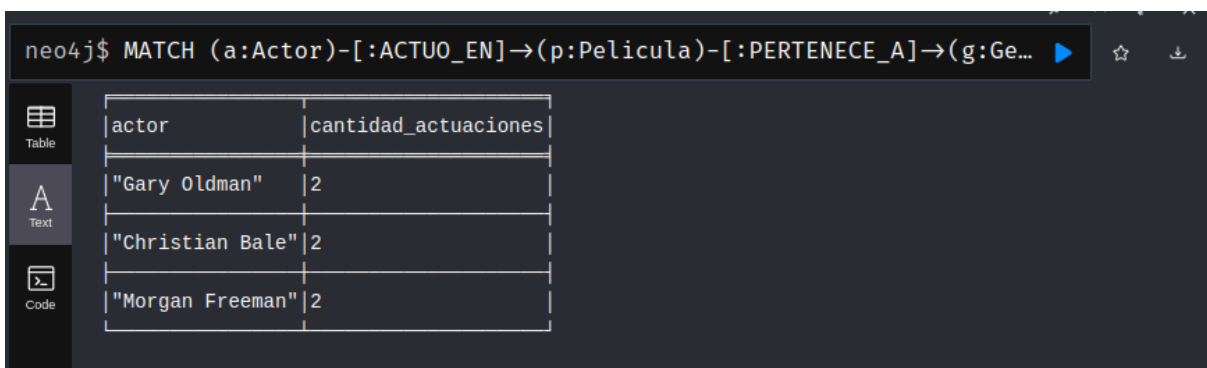


The screenshot shows the Neo4j Desktop interface. At the top, a Cypher query is entered in the command bar: `neo4j$ MATCH (d:Director)-[:DIRIGIDA_POR]->(p:Pelicula) RETURN d.nombr...`. Below the command bar, the results are displayed in a table view. The table has two columns: `director` and `cantidad_peliculas`. The results are ordered by `cantidad_peliculas` in descending order, with a limit of 3.

director	cantidad_peliculas
"Christopher Nolan"	4
"John Lasseter"	2
"James Wan"	2

- **Top 3 de actores con más apariciones en películas de acción**

```
MATCH
(a:Actor)-[:ACTUO_EN]->(p:Pelicula)-[:PERTENECE_A]->(g:Genero
{nombre: "Acción"})
RETURN a.nombre AS actor, count(p) AS cantidad_actuaciones
ORDER BY cantidad_actuaciones DESC
LIMIT 3
```



The screenshot shows the Neo4j Desktop interface. At the top, a Cypher query is entered in the command bar: `neo4j$ MATCH (a:Actor)-[:ACTUO_EN]->(p:Pelicula)-[:PERTENECE_A]->(g:Ge...`. Below the command bar, the results are displayed in a table view. The table has two columns: `actor` and `cantidad_actuaciones`. The results are ordered by `cantidad_actuaciones` in descending order, with a limit of 3.

actor	cantidad_actuaciones
"Gary Oldman"	2
"Christian Bale"	2
"Morgan Freeman"	2

- **Género con más películas y las películas que pertenecen a este género**

```

MATCH (g:Genero)<-[:PERTENECE_A]-(p:Pelicula)
RETURN g.nombre AS Genero, COUNT(p) AS NumeroPeliculas,
COLLECT(p.nombre) AS Peliculas
ORDER BY NumeroPeliculas DESC
LIMIT 1

```

neo4j\$ MATCH (g:Genero)←[:PERTENECE_A]-(p:Pelicula) RETURN g.nombre A...

Genero	NumeroPeliculas	Peliculas
"Drama"	13	["Interstellar", "Three Meters Above the Sky", "The Imitation Game", "Forest Gump", "Batman Begins", "John Wick", "La La Land", "The Dark Knight", "Quisiera ser millonario", "Cuestión de Tiempo", "El Lobo de Wall Street", "Rush", "Mulan"]

Recomendaciones

- Ahora creemos una consulta para recomendarle a una usuaria 'Luisa Martinez' otras películas que han dirigido los directores de las películas que ha visto:

```

MATCH (u:Usuario {nombre: 'Luisa Martinez'})-[:HA_VISTO]->(:Pelicula)<-[:DIRIGIDA_POR]-(d:Director)
-[:DIRIGIDA_POR]->(p:Pelicula)
WHERE NOT (u)-[:HA_VISTO]->(p)
RETURN DISTINCT p.nombre as peliculasRecomendadas

```

```

1 MATCH (u:Usuario {nombre: 'Luisa Martinez'})-[:HA_VISTO]->(:Pelicula)<-[:DIRIGIDA_POR]-(d:Director)
  [:DIRIGIDA_POR]->(p:Pelicula)
2 WHERE NOT (u)-[:HA_VISTO]->(p)
3 RETURN DISTINCT p.nombre as peliculasRecomendadas

```

peliculasRecomendadas	
1	"The Dark Knight"
2	"Batman Begins"
3	"Inception"
4	"The Hangover Part II"
5	"La Noche Del Demonio"

Vemos que, como Luisa Martínez ha visto Interstellar, algunas de las películas que se le recomiendan son también de Christopher Nolan, como es el caso de Batman e Inception.

Ahora creemos una consulta para recomendarle películas al usuario Maximiliano Romero, sugiriendo aquellas que pertenecen al género que más ha visto:

```
MATCH (u:Usuario {nombre: 'Maximiliano
Romero'})-[:HA_VISTO]->(:Pelicula)-[:PERTENECE_A]->(g:Genero)
WITH u, g, COUNT(*) AS count
ORDER BY count DESC
LIMIT 1
MATCH (u)-[:HA_VISTO]->(:Pelicula)-[:PERTENECE_A]->(g)
WITH u, g
MATCH (g)<-[:PERTENECE_A]-(p:Pelicula)
WHERE NOT EXISTS((u)-[:HA_VISTO]->(p))
RETURN p.nombre AS pelicula
```

Obtenemos 18 películas:

```
1 MATCH (u:Usuario {nombre: 'Maximiliano Romero'})-[:HA_VISTO]->(:Pelicula)-[:PERTENECE_A]->(g:Genero)
2 WITH u, g, COUNT(*) AS count
3 ORDER BY count DESC
4 LIMIT 1
5 MATCH (u)-[:HA_VISTO]->(:Pelicula)-[:PERTENECE_A]->(g)
6 WITH u, g
7 MATCH (g)<-[:PERTENECE_A]-(p:Pelicula)
8 WHERE NOT EXISTS((u)-[:HA_VISTO]->(p))
9 RETURN p.nombre AS pelicula
```

	pelicula
1	"I, Robot"
2	"Batman VS Superman: Dawn of Justice"
3	"The Dark Knight"
4	"Rush"
5	"John Wick"
6	"Los Juegos del Hambre"
7	

Recomendar una película dependiendo al usuario que tenga más coincidencia

```
MATCH (u1:Usuario {ID:
"Maria030419885512093876"})-[:HA_VISTO]->(p1:Pelicula)<-[:HA_VISTO]
-(u2:Usuario)
WHERE u1 <> u2
```

```

WITH u2, COLLECT(p1) AS PeliculasVistas
ORDER BY size(PeliculasVistas) DESC
LIMIT 1
MATCH (u2)-[:HA_VISTO]->(p2:Pelicula)
WHERE NOT p2 IN PeliculasVistas
RETURN p2.nombre AS PeliculaRecomendada

```

neo4j\$ MATCH (u1:Usuario {ID: "Maria030419885512093876"})-[:HA_VISTO] ... ▶

Table

PeliculaRecomendada

"Rápidos y Furiosos 5"

Text

Code

```

MATCH (u:Usuario {ID:
"Maria030419885512093876"})-[:HA_VISTO]->(p:Pelicula)
RETURN p.nombre AS PeliculaVista

```

neo4j\$ MATCH (u:Usuario {ID: "Maria030419885512093876"})-[:HA_VISTO]→... ▶

Table

PeliculaVista

"Shrek 2"

"The Hangover"

"The Imitation Game"

"Toy Story 2"

"The Conjuring 2"

Text

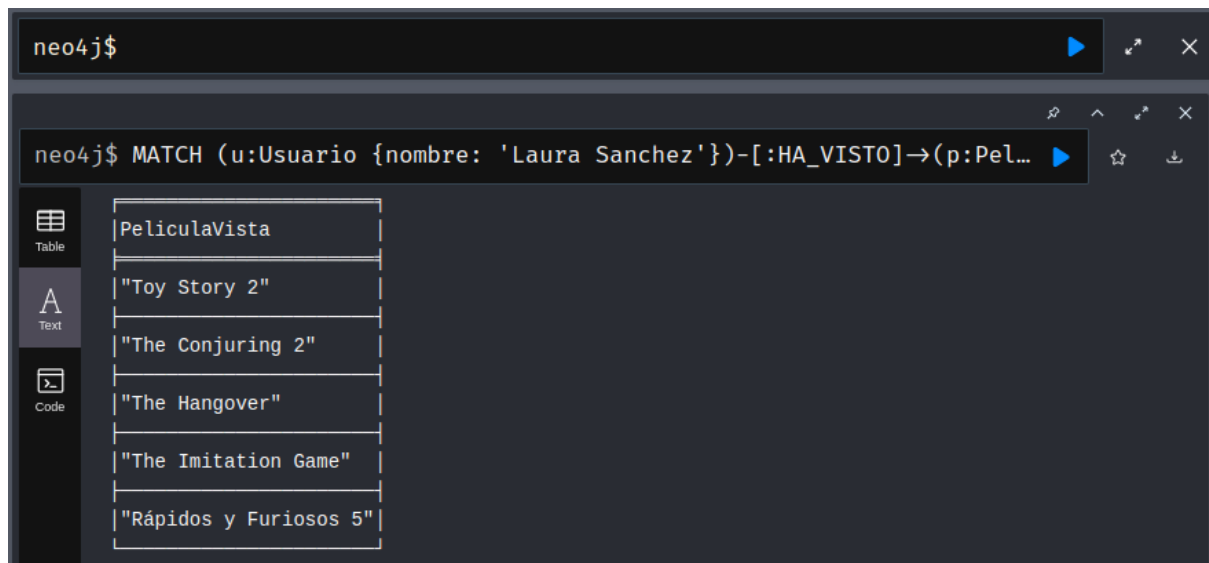
Code

Películas vistas por Laura.

```

MATCH (u:Usuario {nombre: 'Laura
Sanchez'})-[:HA_VISTO]->(p:Pelicula)
RETURN p.nombre AS PeliculaVista

```



Ahora, vamos a crear una consulta para sugerirle películas a Carolina Ruiz, recomendando aquellas que ha visto la persona con la que más tiene películas en común, pero que ella no ha visto.

```
MATCH (u1:Usuario {nombre: 'Carolina
Ruiz'})-[:HA_VISTO]->(p1:Película)
WITH u1, p1
MATCH (u2:Usuario)-[:HA_VISTO]->(p1)
WHERE u1 <> u2
WITH u1, u2, COUNT(p1) AS sharedMovies
ORDER BY sharedMovies DESC
LIMIT 1
MATCH (u2)-[:HA_VISTO]->(p2:Película)
WHERE NOT EXISTS((u1)-[:HA_VISTO]->(p2))
RETURN p2.nombre AS pelicula_sugerida
```

Obtuvimos:

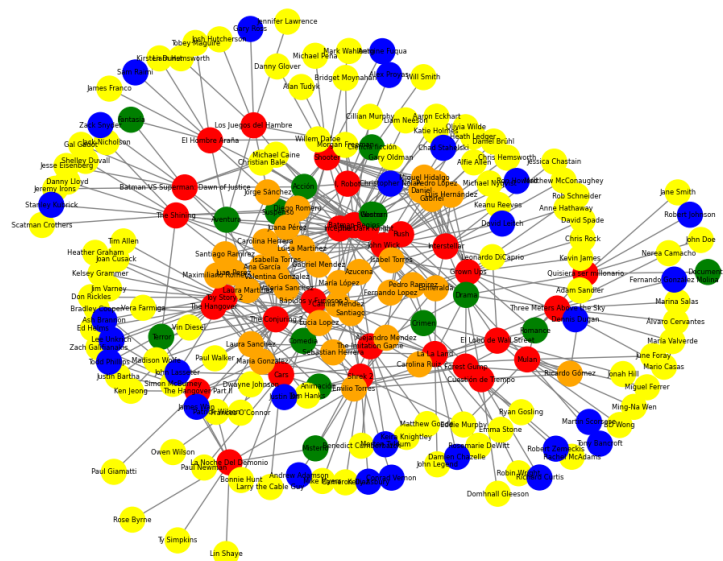
```
1 MATCH (u1:Usuario {nombre: 'Carolina Ruiz'})-[:HA_VISTO]→(p1:Pelicula)
2 WITH u1, p1
3 MATCH (u2:Usuario)-[:HA_VISTO]→(p1)
4 WHERE u1 <> u2
5 WITH u1, u2, COUNT(p1) AS sharedMovies
6 ORDER BY sharedMovies DESC
7 LIMIT 1
8 MATCH (u2)-[:HA_VISTO]→(p2:Pelicula)
9 WHERE NOT EXISTS((u1)-[:HA_VISTO]→(p2))
10 RETURN p2.nombre AS pelicula_sugerida
```

	pelicula_sugerida
1	"The Imitation Game"
2	"John Wick"
3	"The Hangover Part II"

Neo4j en Python

Conectamos la base de datos de Neo4j a Python utilizando las siguientes librerías : **py2neo**, **networkx**, **matplotlib.pyplot**. Estas librerías nos ayudarán a traer la gráfica de **Neo4j** y guardarla como un objeto de **NetworkX** para posteriormente dibujarla. La base de datos almacenada en Neo4j vista como una gráfica de NetworkX se ve de la siguiente manera :

Gráfico Neo4j en NetworkX



Realizamos esta conexión a Python para poder aplicar algoritmos sobre gráficas que serán de utilidad para la plataforma en función de estrategia ya sea de productos como de negocio. Así mismo realizaremos varios análisis ya sea de la gráfica misma, como también sociales.

Análisis de centralidad

Los algoritmos de centralidad son útiles para identificar nodos importantes en la red. Se pueden utilizar estos algoritmos para identificar los nodos con más conexiones, o la centralidad de intermediación (betweenness centrality) para identificar los nodos que conectan diferentes partes de la red. Esto nos podría ayudar a identificar los directores más influyentes, los géneros más populares, los actores más conectados, etc. El algoritmo PageRank es un algoritmo de centralidad popular creado por los fundadores de Google, que se utiliza para medir la importancia relativa de los nodos en una red. A diferencia de otros algoritmos de centralidad, PageRank se basa en la idea de que un nodo es importante si es apuntado por otros nodos importantes.

Los resultados que aparecieron primero para cada tipo de nodo son:

- Película : En el puesto # 1 fue Toy Story 2
- Género : En el puesto # 18 fue Drama
- Usuario : En el puesto # 35 fue Ricardo Gómez
- Director : En el puesto # 75 fue Christopher Nolan
- Actor : En el puesto # 76 fue Christian Bale

De un total de 204 nodos, los análisis de centralidad revelaron que ciertos nodos se destacan como los más importantes en la red. Estos nodos clave tienen un mayor número de conexiones y desempeñan un papel significativo en la estructura de la red. Esto nos proporciona información valiosa para tomar decisiones estratégicas.

Entre los nodos de tipo "Película", la película "Toy Story 2" se destaca como una de las más importantes debido a las numerosas conexiones que tiene con otros nodos. Esta información sugiere que la saga de Toy Story puede ser una franquicia relevante y exitosa dentro de la red, lo cual podría influir en el desarrollo de una estrategia para incluir más películas de esta saga y aprovechar su popularidad.

En cuanto al género, el análisis de centralidad revela que el género "Drama" ocupa un lugar destacado en la red. Esto se debe a que muchas películas en la red están asociadas con este género en particular, lo que le otorga una gran predominancia dentro de la estructura general. Este hallazgo sugiere que el género "Drama" tiene una influencia significativa en la red y puede ser un factor importante a considerar en futuras decisiones relacionadas con la selección de películas.

Al analizar los nodos de tipo "Usuario", se identificó que el usuario "Ricardo Gómez" es el más importante dentro de la red. Esto podría indicar que "Ricardo Gómez" ha visto películas que son consideradas importantes en la red, lo que lo posiciona como un usuario influyente o activo. Esta información puede ser útil para comprender las preferencias de los usuarios y

personalizar recomendaciones o estrategias de marketing dirigidas a usuarios con intereses similares.

En cuanto a los nodos de tipo "Director", el análisis de centralidad señala a "Christopher Nolan" como el director más importante en la red. Esto se debe a que varias películas en la red han sido dirigidas por él, lo que refuerza su relevancia y destaca su contribución a la estructura general de la red. Esta información puede ser valiosa al considerar colaboraciones futuras con directores destacados o al promocionar películas dirigidas por "Christopher Nolan".

Por último, entre los nodos de tipo "Actor", "Christian Bale" se destaca como el actor más importante en la red. Esto se debe a su participación en varias películas de Batman, que parecen tener una presencia significativa en la red. La popularidad de las películas de Batman y el papel protagónico de Christian Bale pueden explicar su importancia en la red y ofrecer oportunidades para aprovechar su influencia en futuros proyectos o campañas de marketing.

En resumen, el análisis de centralidad nos permite identificar los nodos más importantes en la red, como películas, géneros, usuarios, directores y actores clave. Estos resultados nos brindan información valiosa para tomar decisiones estratégicas relacionadas con la selección de películas, el desarrollo de franquicias, la personalización de recomendaciones y la colaboración con directores y actores influyentes.