

MEMORIA TRABAJO FINAL



Daniel Alonso Rodríguez

18-12-2019

Índice

Índice	0
Introducción.....	2
Descripción del trabajo: Tema y problemas.	2
Objetivos.....	3
Herramientas y metodología.....	4
Trabajo.....	7
Casos y Sensibilidad	28
Análisis crítico	31
DAFO.....	31
Línea de futuro	33
Bibliografía.....	34
Anexo: Cuaderno de Trabajo.....	35

Introducción

Con la llegada del mundo globalizado y de las nuevas tecnologías, muchas industrias han sufrido cambios radicales en sus aspectos más tradicionales. Algunas de ellas han ido desapareciendo o perdiendo protagonismo en favor de otras iguales pero conectadas a la red. Una de las industrias que más protagonismo ha perdido ha sido la del cine, en favor de las plataformas online de distribución y streaming online de películas. En el último lustro, páginas webs como Netflix, HBO, Rakuten TV, Amazon Prime Video ofrecen servicio de visualización de series y películas online sin necesidad de descargas a cambio del pago de una mensualidad relativamente asequible. El gran éxito de todas estas plataformas, sobretodo de Netflix, ha sido ofrecer un sistema de recomendación muy exacto y preciso para los gustos de cada cliente.

Asimismo, con la proliferación de Internet y de la interacción entre usuarios en un mundo cada vez más y más conectado, numerosas páginas webs relativas al cine y a películas como IMBD, FilmAffinity etc. han aparecido y se han expandido. En ellas, los usuarios de las citadas páginas, pueden dejar una valoración y una opinión. En función de estas notas de los consumidores, y de las críticas que deja un mismo usuario, además de las características propias de cada película, estas webs brindan no solo información relativa a las películas si no también recomiendan otros films en función de lo explicado anteriormente. Esto ha introducido una nueva manera de interactuar entre usuarios y de búsqueda y recomendación de películas.

Descripción del trabajo: Tema y problemas.

Después de analizar todas las posibilidades de las páginas webs y el sistema de recomendación que utilizaban las plataformas dichas anteriormente, encontré una posibilidad nueva. Una variante de todo lo visto anteriormente y que, a mí, personalmente, me parecía muy interesante y útil. La idea era realizar un recomendador de películas, pero a través de parámetros introducidos por el usuario de la aplicación. Asimismo, también me parecía atractivo el hecho de poder recomendar libros, a través de películas, y viceversa, ya que muchas películas que se han filmado, y cada vez más de las que se filman actualmente, están basadas en libros. Esto me ofrecía un abanico nuevo de posibilidades bastante amplio. En muchas ocasiones, me ha surgido el problema de querer ver una película de un director concreto, o un actor, y no saber exactamente qué ver. De esta manera, proporcionando al usuario la posibilidad de controlar en todo momento su recomendación, se podrá ofrecer algo mucho más acorde a lo que quiere ver en un momento preciso.

El principal problema que he tenido a lo largo de la creación de la aplicación ha sido el tiempo. Buscar información, conocer la aplicación de Neo4J, familiarizarse con su entorno y comprender el lenguaje de programación de Cypher, conocer Python y sus distintas posibilidades, y, por último, desarrollar el software solucionando sus fallos, me ha ocupado un tiempo muy valioso y ha sido complicado terminar todo lo que yo pretendía en un periodo tan corto, que han sido 4 meses. Este ha sido el principal escollo que superar, y por supuesto, no he podido completar todo lo que tenía pensado en un principio, aunque creo que el resultado es más que satisfactorio y positivo.

Además del tiempo, he tenido otros problemas, la mayoría inherentes al desarrollo del programa que soporta la aplicación. El principal inconveniente que encontré aquí, fue que

algunos de los datos que había introducido en la base de datos, eran difícilmente exportables a Python, y por tanto trabajar con ellos era imposible. Para solventar esto, tuve que realizar auténticos jeribeques con el código, para ir cambiando, desde Python, los datos introducidos, sin modificar su contenido propio.

También, otro problema que encontré, fue el de usar Python, un lenguaje con numerosas debilidades y muchísimas restricciones, tanto en sus posibilidades de entorno gráfico como en sus posibilidades de desarrollo de código.

Sin embargo, a pesar de esto, he conseguido llevar a cabo una aplicación bastante interesante bajo mi punto de vista.

Objetivos

Los objetivos de este trabajo son simples: conocer y comprender las bases de datos de grafos de conocimiento basado en nodos, especialmente el entorno de Neo4J, entender la plataforma y su manera de trabajo y funcionamiento. Además de esto, al acabar todo el proceso de aprendizaje quiero poder realizar bases de datos de nodos de conocimiento de forma fluida y sin problemas, teniendo un dominio alto de Neo4J, así como de Cypher y su programación y programa de consultas.

Como no conozco nada de Python ni de su funcionamiento más básico, el objetivo en ese sentido es ser capaz de programar en Python y adquirir la sabiduría necesaria para llevar a cabo cualquier tipo de programa e interacción con este lenguaje, que al principio del trabajo es nulo ya que nunca he trabajado con ello.

Sin embargo, aunque todo esto anterior está muy bien, no debemos olvidar cuál es el propósito primario del trabajo: realizar una aplicación que recomiende películas. Por ello, también existen objetivos claros en cuanto al desarrollo del trabajo. Quiero llevar a cabo una aplicación buena, que cumpla los requisitos personales que he expuesto, que recomiende de forma precisa y que su único problema sea la escalabilidad, cosa que podría solventarse de forma natural, añadiendo más y más datos a la base de datos de Neo4J. Si bien el objetivo principal es que funcione correctamente, no debo descuidar la parte gráfica. Todos podemos llegar a la conclusión de que una interfaz atractiva, intuitiva y sencilla es mucho más interesante y “seductora” para un potencial usuario. Cuando entramos en una página web que a simple vista nos resulta desordenada, con la información poco precisa, nos provoca cierto rechazo y de primeras la repercusión, la primera impresión, es negativa. Por todo ello, me parecía que el hecho de que la parte gráfica del programa fuese estéticamente bonito constituía uno de los objetivos secundarios más importantes. Específico secundarios porque ante todo está el funcionamiento real del software. Si éste no funciona, independientemente de cómo sea el resto, carece de cualquier sentido. Por lo tanto, en orden de prioridades, podemos establecer que el aprendizaje mediante el desarrollo del software es lo primero, y lo segundo realizar una interfaz atractiva para el usuario.

Herramientas y metodología

Dentro de este apartado, incluiremos los distintos programas y aplicaciones, así como posibles páginas webs que se hayan usado dentro del desarrollo del trabajo. También incluiremos cómo hemos llevado a cabo el trabajo desde el sentido de la metodología.

La lista de herramientas utilizadas es la siguiente:

- **Neo4J Desktop:** la principal herramienta dentro del desarrollo de la base de datos. Neo4J Desktop es el entorno gráfico en el cual se basa nuestra base de datos. Es desde esta aplicación desde la cual vamos construyendo nuestro grafo. Neo utiliza un lenguaje propio, llamado Cypher muy intuitivo y bastante sencillo. La principal diferencia con el resto de programas que gestionan bases de datos, es que Neo almacena datos estructurados en grafos, en vez de en tablas. Desde ahí, hemos construido nuestro grafo, mediante la programación de Cypher.
- **Sublime Text:** Sublime Text es un procesador de texto y editor de código fuente para Python, entre otros lenguajes. Al principio comencé con este editor de texto, pero rápidamente deseché la idea de continuar con ese procesador de texto. Aunque me gustaba el entorno y cómo estaba presentado, el método de ejecución y demás no era excesivamente bueno. Por ello, busqué una opción distinta.
- **Anaconda (Spyder):** Anaconda es un software de distribución libre para editar código en lenguaje Python. Tiene varios softwares dentro del propio software principal, y uno de ellos es Spyder, el editor de texto de Python. Dentro de él, existen todas las funciones necesarias para desarrollar una aplicación de la manera que me gustaba. Entre estas funciones incluía una consola intuitiva siempre activa y un visualizador de variables.
- **Internet:** Por muy evidente que parezca, Internet fue una de las herramientas más relevantes que utilicé para llevar a cabo la aplicación. Principalmente lo usé para reunir la información de todas las películas y los libros, evidentemente. Sin embargo, no fue el único propósito para lo que usé la Web. Al ser la primera vez que programaba en Python, al principio, mi desconocimiento del lenguaje al ser un primerizo, me dio numerosos problemas que fui solucionando a lo largo del tiempo mediante búsquedas en la red. Esto fue algo recurrente a lo largo del desarrollo del software.
- **TKinter:** TKinter es una herramienta propia de Python, una librería de la biblioteca gráfica del lenguaje de programación Python. Se considera un estándar para la interfaz gráfica de usuario (GUI) para este lenguaje. Es la librería que utilicé para crear toda la interfaz gráfica de mi aplicación.
- **Microsoft Office Excel:** Excel es una hoja de cálculo desarrollada por Microsoft para Windows, entre otros sistemas operativos. Aunque cuenta con cálculo, herramientas gráficas, tablas dinámicas y un lenguaje propio de programación. No obstante, a pesar de tener tantas herramientas tan útiles, la única funcionalidad que he usado es simplemente la de almacenar información en las tablas, ya que lo muestra de manera ordenada y muy visual.

En cuanto a la metodología que hemos utilizado, vamos a realizar un rápido resumen de la manera en la cual hemos llevado a cabo poco a poco el trabajo. La técnica empleada fue ir de lo general a lo particular, en orden, esto fue lo más importante, y de forma pausada pero continuada. Vamos a establecer un orden cronológico dentro de esta metodología:

- 1. Establecer un método de trabajo ordenado:** organizar el trabajo de forma que cuando se realice una tarea, se encuentren todos los datos necesarios y demandados para llevar a cabo dicha actividad sin perder el tiempo requerido en recopilar estos datos que nos faltan. Cuando hablamos de datos que nos faltan, englobamos tanto información estadística más sencilla, como pueden ser datos de películas o directores, como la información necesaria para comprender, entender y saber utilizar todas las herramientas antes de comenzar con el trabajo.
- 2. Comprender y saber trabajar con las herramientas antes citadas:** en relación con lo explicado anteriormente, el paso siguiente fue llevar a cabo el proceso de entendimiento de los programas que iba a usar. Unos cuantos de ellos ya eran conocidos para mí, como Excel, Sublime Text, o las herramientas proporcionadas por Internet ya eran vicisitudes que controlaba de forma más que aceptable. Sin embargo, el entorno Anaconda y sobretodo Python, el lenguaje en sí, y su librería TKinter y las opciones que ésta daba, y también Neo4J Desktop, su funcionamiento y dinámicas, además de su lenguaje de programación: Cypher. Dentro de este último apartado, también fue muy necesario tener un conocimiento relativamente alto de Cypher y de su manera de trabajo para introducir todos los datos necesarios para construir la base de datos y su posterior utilización.
- 3. Recopilación de información y datos:** Una vez entendidas todas las herramientas, damos paso a la parte de análisis y recolección de datos sobre las películas y libros, así como su almacenamiento, con el objetivo de lograr una visualización de los mismos, mucho más rápida, efectiva y eficaz. Todo ello con el objetivo de conseguir introducirlos también de la manera más cómoda y mejor posible.
- 4. Introducción de datos y formación de la base de datos:** Ya con los datos necesarios adquiridos, pasamos a componer poco a poco la base de datos. Utilizando el lenguaje de Cypher y sus sentencias y consultas, pasamos a meter y estructurar toda la base de datos.
- 5. Organizar y ordenar el método de programación:** Antes de comenzar con el proceso de desarrollo de software, debemos construir un esquema mental de cómo vamos a querer programar nuestro código, como va a estar estructurado y qué queremos hacer con el software.
- 6. Organizar y ordenar la interfaz gráfica:** Lo siguiente antes de iniciar el software es plantear cómo queremos que sea nuestra interfaz, las ventanas que van a estar por delante de nuestro código fuente. Estructurar las ventanas y, ahora que ya conozco las posibilidades, pensar en cómo queremos que sea la aplicación.
- 7. Desarrollo del software:** Por fin llegamos a la etapa importante, la parte de programación. Aquí, seguimos los pasos que hemos construido antes en nuestra cabeza y en el papel, y vamos programando de manera ordenada y correcta.
- 8. Corrección de errores:** Una vez nuestro código está programado, debemos probarlo y considerar dónde se están produciendo los errores que, en un primer momento, han pasado desapercibidos, y lógicamente, ir solucionando cada uno de ellos.

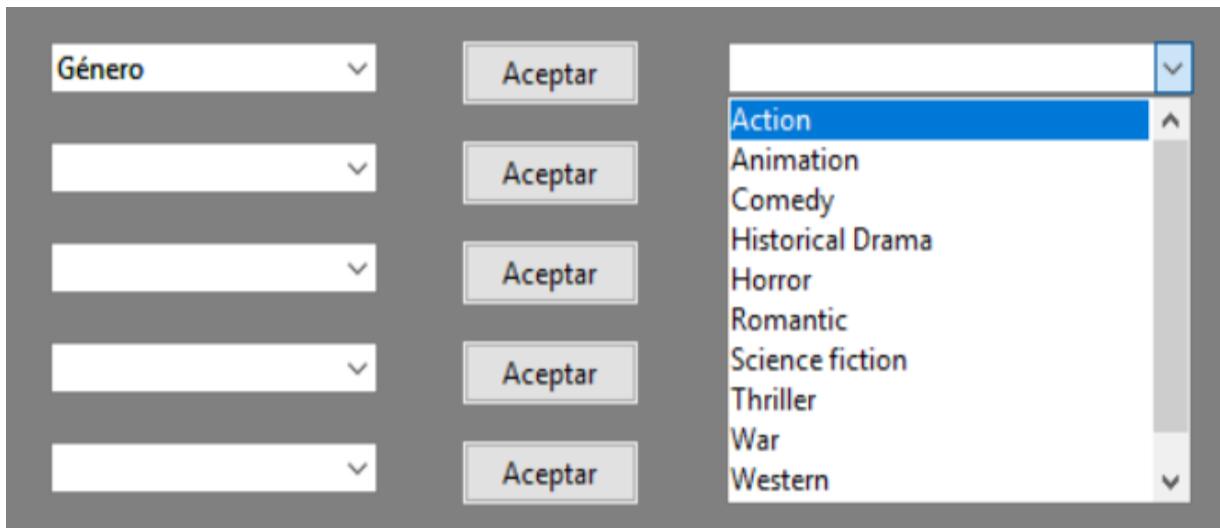
9. **Ampliación del programa:** Muy a menudo, a pesar de la estructuración inicial de los pasos 5 y 6, explicados anteriormente, nos ocurre que, a medida que va avanzando el desarrollo del programa, vamos observando y se nos van ocurriendo nuevas funcionalidades que se podrían añadir al programa para hacerlo más completo, por lo que, si el tiempo y las condiciones lo permiten, ya que a veces la programación realizada anteriormente impide ciertos cambios en el código. Lo imposibilita porque a veces, para las nuevas implementaciones, es necesario reformar gran parte del código fuente hecho anteriormente. Por ello, si es posible, realizamos las nuevas implementaciones de código nuevo con las nuevas funciones que se nos ocurren.
10. **Nueva corrección definitiva de errores:** Volvemos a realizar el paso 8, ya que, si se realizan nuevas aplicaciones en el programa, debemos volver a chequear que sigue sin haber inconvenientes en el código de forma definitiva.
11. **Versión final:** Reajuste sólo de la parte gráfica, cambios en el emplazamiento de los elementos, pruebas del software y comprobaciones finales. Una vez todo esto realizado, lanzamos la versión final de la aplicación.

Trabajo

En este apartado valoraremos todos los aspectos más importantes dentro del trabajo y sólo exclusivamente del trabajo final, y esto es en lo que se diferencia del cuaderno de campo. Es recomendable matizar esto, ya que como el cuaderno de campo ha sido redactado y puesto en orden, ambos apartados pueden parecerse considerablemente. Una vez especificado este punto que me parece sumamente relevante, pasamos a elaborar un resumen amplio de nuestro trabajo. De lo que hace nuestro software y de cómo lo hace. Además, incluiremos un caso práctico para poder ver de forma visual y gráfica, cómo funciona y lo que aporta, y realizaremos una variación sobre este caso, para ilustrar, si cabe, todavía más, nuestro programa y su funcionamiento.

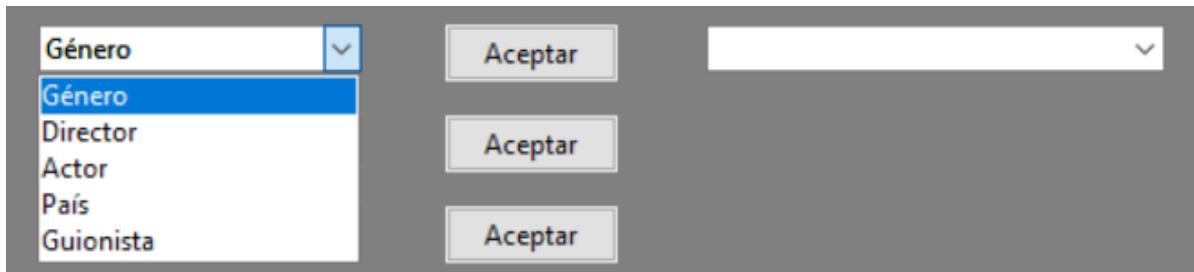
El trabajo consiste en un recomendador de películas y libros, una aplicación, con entorno gráfico, que recomiende películas basadas en libros, y sus respectivos libros. La diferencia con el resto de recomendadores, es que, en mi programa, la aplicación no recomienda nada en base a su conocimiento, sino que, basará su recomendación en parámetros que el propio usuario introducirá. La idea consiste en ofrecer al usuario una serie de características comunes a todas las películas. Estas son el director, los actores protagonistas y principales, el género, el país y el guionista. Los dos últimos apartados parecen nimios y poco relevantes, aunque hay varios aspectos que pueden explicar el porqué de la existencia de estas dos opciones. Empecemos por lo que parece menos importante, el país. A pesar de parecer poco importante, este es un factor que las personas más aficionadas al cine tienen muy en cuenta a la hora de visualizar una película. Las filmaciones de una misma nacionalidad suelen tener rasgos comunes entre ellas: mismos planos, mismas maneras de rodar, misma fotografía, debido a que los escenarios de grabación son muy parecidos y de las mismas ciudades, la temática es parecida en muchos aspectos, por lo tanto, el país de origen de la película, es para las personas más cinéfilas, un criterio importante en ciertas ocasiones. Por otro lado, existe el guionista. El guionista es una de las personas más imprescindibles en el desarrollo de una película, el guion puede ser lo que lance al estrellato una película, y es uno de los aspectos más relevantes y distinguibles en una filmación. Incluso en algunas ocasiones, hay particularidades destacadas entre películas de un mismo guionista, ya que ciertos directores trabajan siempre con el mismo guionista. Además de todo esto, en películas basadas en libros, que es de lo que trata mi aplicación, los guionistas y adaptadores, que, en algunas ocasiones, son dos cargos que ocupa una misma persona, toman una relevancia aún más grande si cabe, ya que muchas veces hemos visto grandes libros realmente mal adaptados a la gran pantalla. Por todo ello, podemos finalmente concluir que el guionista es una pieza clave en las películas, y aún más en aquellas que son adaptaciones de libros, que es lo que mi recomendador contempla.

Finalizado este debate, podemos avanzar con lo siguiente. El usuario podrá elegir entre estos cinco parámetros explicados anteriormente. Después de elegir uno de ellos, el usuario podrá seleccionar nuevamente entre las distintas opciones que se generan dentro del primer grupo. Este concepto puede resultar difuso sin un ejemplo que lo precise de mejor forma, así que pondremos uno relativo al género de la película. Si el usuario decide elegir el género como criterio para realizar la primera filtración para buscar películas, se desplegará luego otro abanico de opciones entre los cuales encontraremos todos los tipos de género que hayamos valorado introducir, y elegirá uno de los géneros. Vamos a dejarlo aún más claro con una imagen de la aplicación.



Como vemos, en este caso, el usuario ha querido seleccionar el género de la película, como su primera opción para que la película cumpla estos requisitos. Y, por ende, en la parte de la derecha, la persona que está utilizando la aplicación, puede seleccionar entre todos los géneros que valoramos que estén. Esto ocurriría con todos los criterios hablados antes. Tanto con el director, el guionista, el país... Siempre que se seleccione uno de estos parámetros en la parte de la izquierda, se desplegará una lista de los mismos en la parte de la derecha.

De esta manera, con la imagen adjuntada antes, podemos observar un poco, a grandes rasgos, de qué manera se estructura la aplicación. El usuario selecciona en las cajas de la izquierda uno de los parámetros que venimos repitiendo, y en la derecha el parámetro definitivo.



Este es el funcionamiento principal y más básico del programa. El usuario elige parámetros en las cajas de la izquierda, se despliegan opciones relativas a ellas en las cajas de la derecha y el recomendador busca y filtra según lo introducido.

Sin embargo, el funcionamiento es mucho más complejo. Como pretendemos realizar un recomendador lo más preciso y conciso posible, debemos conseguir filtrar de manera exacta todas las películas. Para conseguir esto, hemos establecido varias cajas distintas en la zona de la izquierda, para que el usuario pueda establecer numerosos criterios distintos, para, con ello, llegar a sacar la película que mejor se adapta a lo que el usuario requiere.



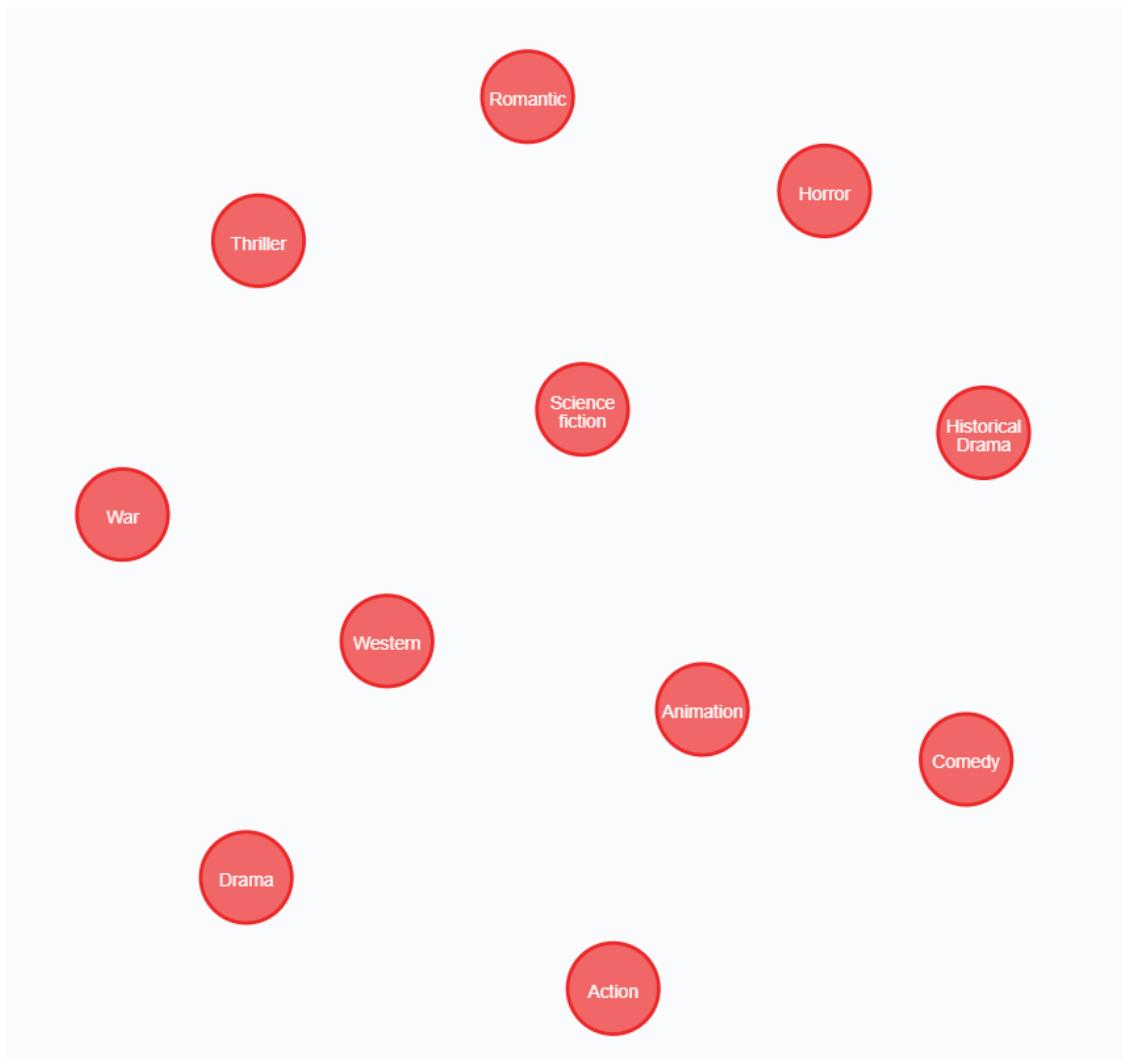
Hay hasta cinco cajas para que el usuario lleve a cabo su búsqueda. Hemos decidido que cinco era el número ideal porque así la persona que utiliza la aplicación podrá poner en cada caja los cinco parámetros distintos que se ofrecen.

En todas las imágenes hemos podido apreciar que hay varios botones. La mayoría de ellos contienen el texto “Aceptar”, aunque en esta última imagen, vemos otros dos nuevos: “Buscar” y “Nueva Búsqueda”. Continuando con la explicación detallada del programa, es necesario realizar una ilustración detallada del funcionamiento y del propósito que cumplen todos los elementos que vemos en la aplicación. Comencemos por los botones “Aceptar”.

Si bien antes hemos comprobado que es necesario ir seleccionando criterios. Pero no todo es tan sencillo. Cuando elegimos un parámetro de la izquierda, siempre debemos apretar el botón aceptar en cada una de las franjas en las que impongamos una característica. Es decir, por cada opción seleccionada en la caja de la izquierda, debemos apretar el botón aceptar que está a la derecha de la caja en donde queremos introducir un criterio. Así, si introducimos una característica en la primera caja, debemos apretar el primer botón aceptar, si elegimos en la segunda caja, apretamos el segundo botón aceptar, y así sucesivamente. Cuando apretamos el botón aceptar, antes hemos visto que aparece una caja a la derecha del botón apretado, donde salen las opciones. Aquí aparece una parte importante dentro del desarrollo del software junto con el desarrollo de la base de datos. Cada vez que elegimos una opción a la izquierda, y pulsamos su respectivo botón aceptar, el código realiza una consulta a la base de datos de Cypher que anteriormente hemos hecho. Por ejemplo, si cogemos el primer ejemplo, en el cual elegimos el género, en cuanto apretamos aceptar, se realiza una consulta a todos los géneros que hemos introducido en la base de datos. Esto sucederá en todas las cajas y cada vez que apremos aceptar.

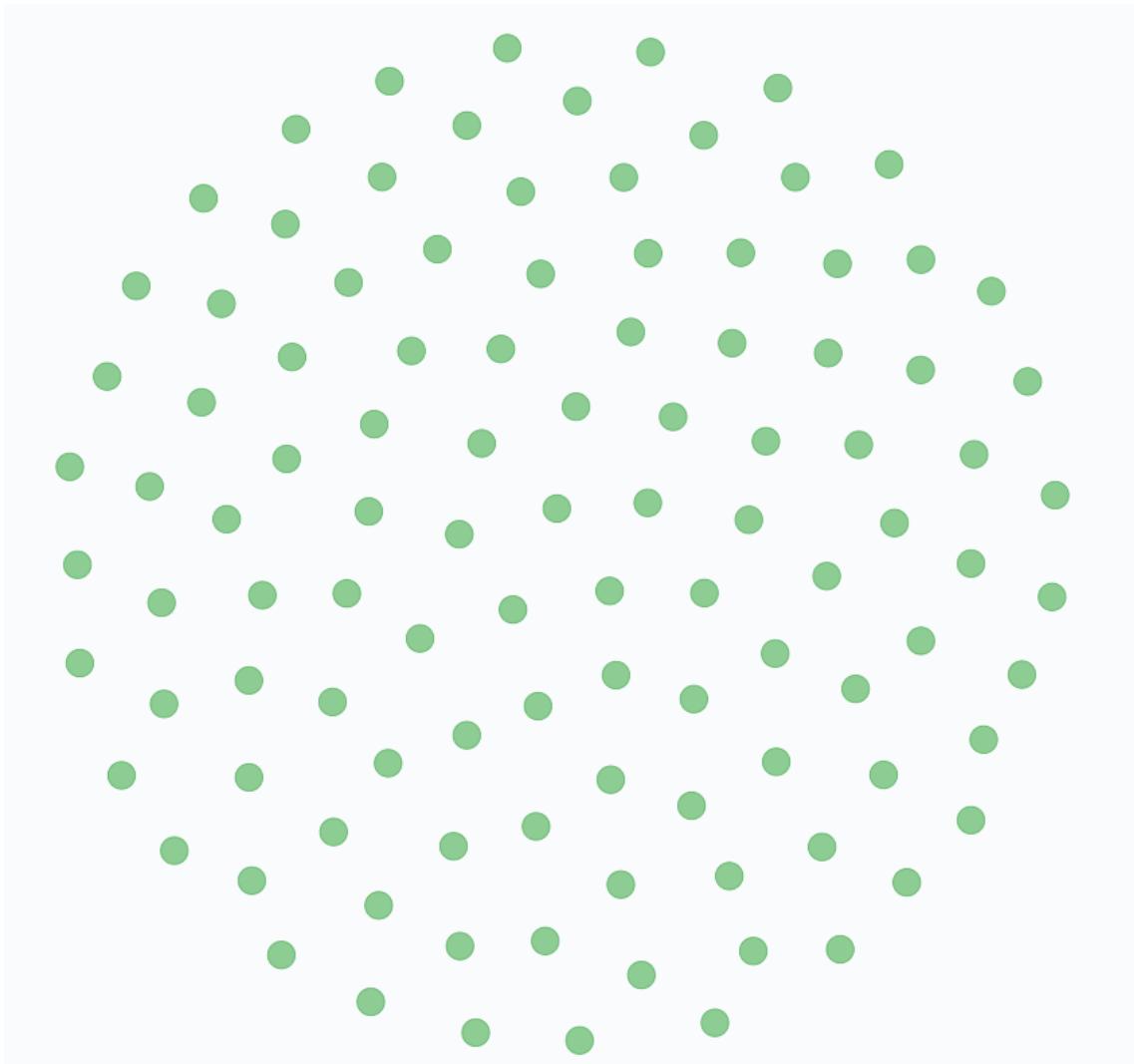
Previamente en Neo4J, mediante Cypher hemos creado una base de datos de nodos de conocimiento. En ella, podemos distinguir tres tipos distintos de nodos. Unos pertenecen a los géneros, con los cuales podemos englobar todas las películas y realizar relaciones muy fácilmente entre película y género, libros, las películas están basadas en libros, y por tanto

también se puede realizar una relación sencilla entre libros y películas, y, por último, las películas conforman el tercer tipo de nodo.



Aquí vemos todos los géneros que introduce mediante Cypher y aquí debajo el modo de introducir todos los géneros, es decir, el trabajo detrás de ello y habría que extrapolar esta sentencia de Cypher a todos los géneros existentes.

```
1 CREATE (g:Genre {type="War"})  
2  
3
```



Estos son todos los nodos relativos a las películas. La manera de crearlos es exactamente la misma que en los géneros.

```
1 CREATE (m:Movie{title="El Padrino"})
2
```

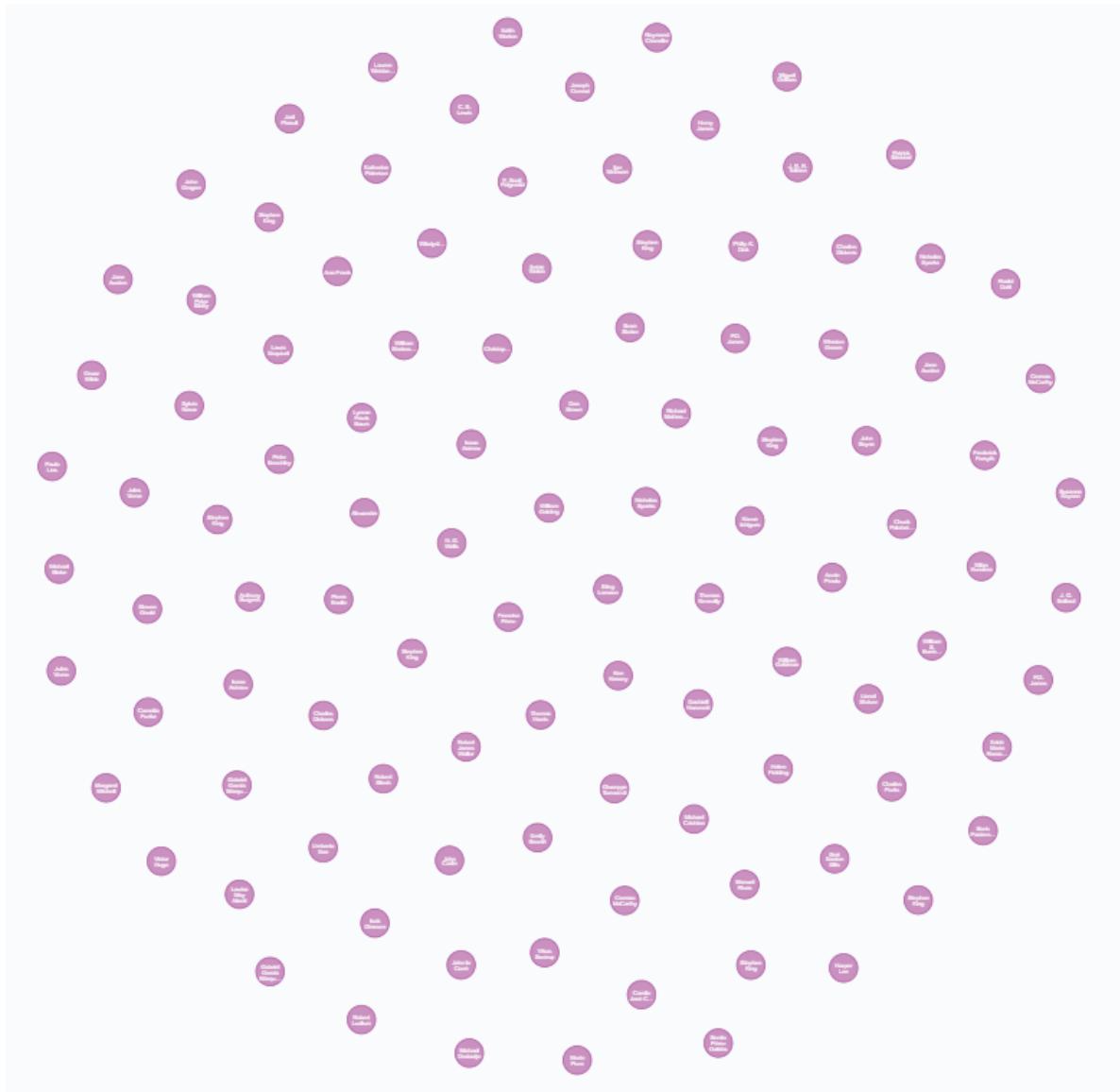
Cada una de ellas contiene unos cuantos atributos propios de cada una.

actores: Sam Neill, Samuel L. Jackson, Richard Attenborough, Laura Dern **critica:** 8'1, 8, 7

director: Steven Spielberg **guion:** Michael Crichton, David Koepp **pais:** EE.UU. **title:** Parque Jurásico

Que son introducidos de la siguiente forma:

```
1 MATCH (m:Movie)
2 WHERE m.title="El Padrino"
3 SET m.actores="", m.director="", m.guion="", m.pais="", m.critica=""
4
```



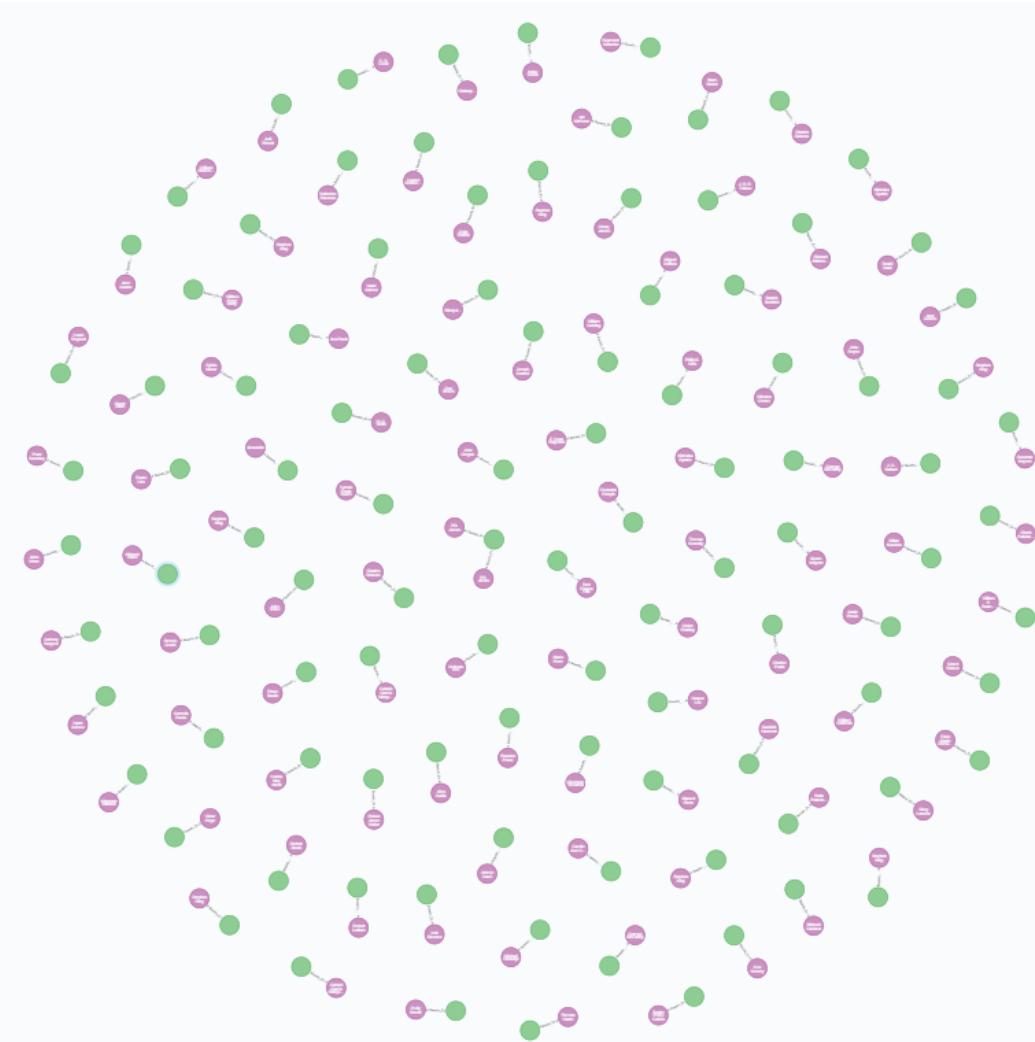
Por último, estos son los nodos pertenecientes a los libros. Al igual que todos los otros nodos, los creamos igual.

```
CREATE (l:Libro{titulo="El Padrino"})
```

Y estas son las propiedades que tiene cada nodo del tipo Libro.

escritor: Ken Kesey **país:** EE.UU. **título:** Alguien voló sobre el nido del cuco

Una vez que completé la creación de todos los nodos necesarios, llegaba la hora de trazar las relaciones, dentro de Neo4J, mediante Cypher, entre los tipos de nodos que existen. Las relaciones son sencillas, debemos unir los libros con las películas, cada libro introducido, con la película que se basa en él, y cada película con su género o géneros correspondientes.

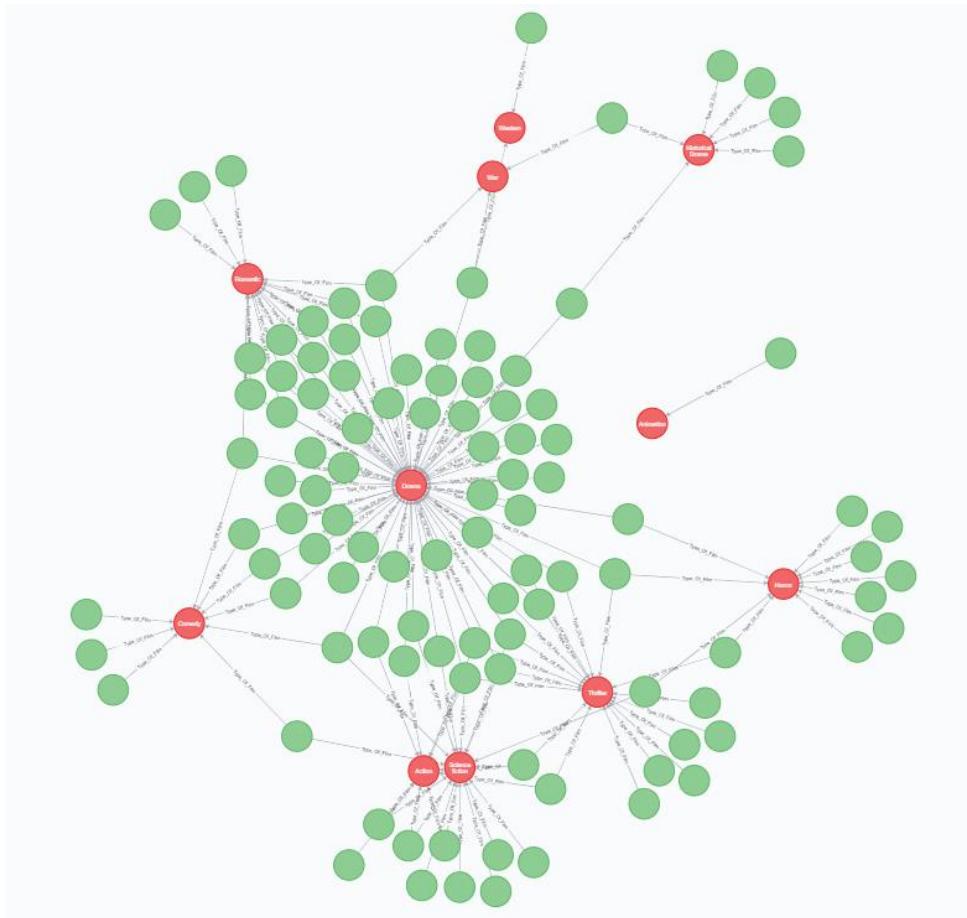


Aquí podemos ver, teniendo en cuenta el código de color visto anteriormente en los nodos, la relación entre películas y libros. En verde las películas, en morado los libros. El mismo color que antes en las imágenes en las cuales sólo veíamos los nodos, para no dar lugar a equivocaciones.

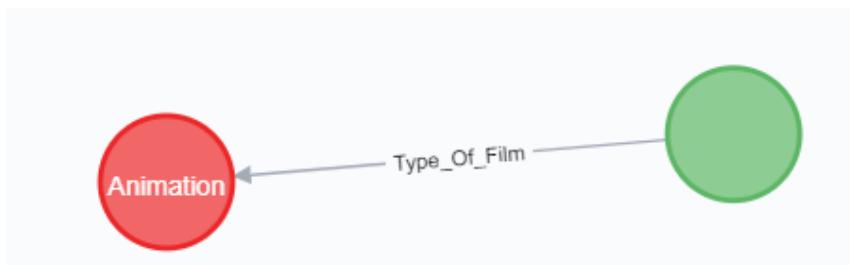
```
$ CREATE (m:Movie)-[:Basado_En]→(l:Libro)
```

Y ésta es la sentencia necesaria de Cypher para ir creando las relaciones. Faltaría especificar en otras dos líneas de código cuál es la película y cuál el libro, pero la creación en sí de la relación, se resume básicamente a eso. Observamos que todas las relaciones entre Movie, nodos referidos a las películas, y Libro, nodos referidos a los libros, se llamará Basado_En.





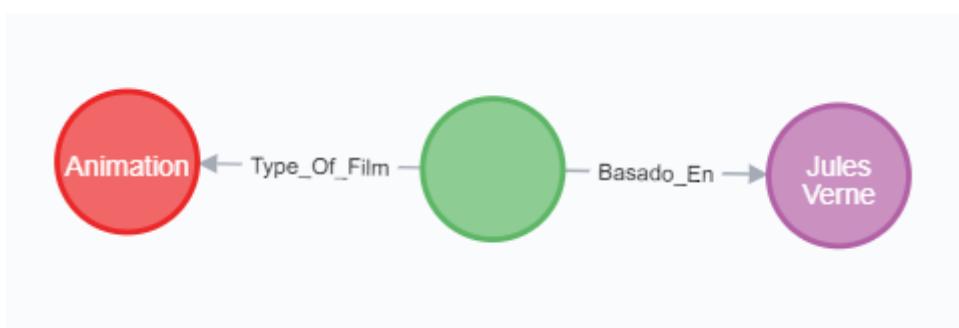
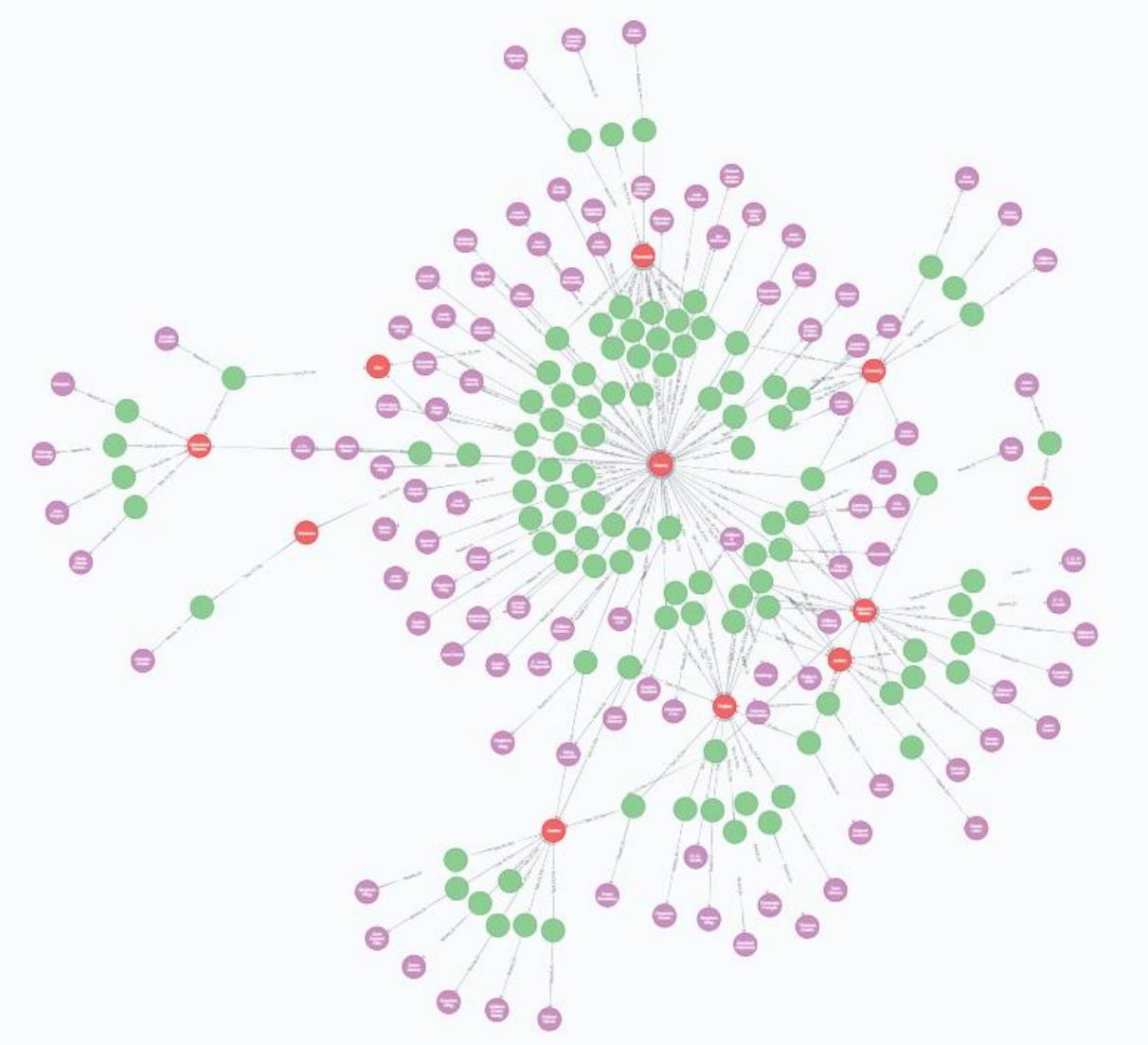
En esta imagen observamos el grafo de relaciones entre películas y géneros. Al igual que en la imagen anterior, hemos guardado el mismo color para los nodos que los que teníamos antes, resultando el rojo para los géneros y el verde para las películas.



```
$ CREATE (m:Movie)-[b:Type_of_Film]→(g:Genre )
```

Y aquí, la sentencia que crea las relaciones entre géneros y películas, que tomaran el nombre de Type_of_Film.

Finalmente, ya estaría completada toda la base de datos, con todos los nodos, películas, libros y géneros y sus correspondientes atributos. Esta es la apariencia general que adquiere el grafo, una vez se encuentra ya del todo completo.



En la segunda imagen, vemos un extracto más cercano de lo que sería la base de datos de todas las películas. Películas, géneros y libros, y relaciones entre libro y película y entre género y película.

Como hemos visto antes, en los nodos correspondientes a las películas y a los libros, hay atributos propios en cada nodo, referentes a aspectos cinematográficos. Estos serán luego los parámetros que podremos elegir en la aplicación y que hemos explicado antes. El proceso de introducción de estas características fue el más tedioso prácticamente de todo el trabajo. Se dividió en dos partes: una primera de búsqueda y almacenamiento de información, y otra de pasar estos datos al grafo mediante Cypher. Adjunto una imagen del Excel que utilicé y donde fui gestionando y estructurando los datos, para las películas:

Película	Director	Actor Principal	Valoración	Guionista	País
El Padrino	Francis Ford Coppola	Marlon Brando, Al Pacino	10, 9'8, 9	Mario Puzo, Francis Ford Coppola	EE.UU.
El silencio de los corderos	Jonathan Demme	Anthony Hopkins, Jodie Foster	8'6, 9'6, 8'2	Ted Tally	EE.UU.
La Colmena	Mario Camus	Victoria Abril, Ana Belén	7'2, 7'1, 7	José Luis Dibildos	España
La Carretera	John Hillcoat	Viggo Mortensen, Emile Hirsch	7'2, 8'1, 6'6	Joe Penhall	EE.UU.
Nazarín	Luis Buñuel	Francisco Rabal, Marisa Paredes	7'9, 7'8, 7'7	Luis Buñuel, Julio Alejandro	México
El Gatopardo	Luchino Visconti	Burt Lancaster, Alain Delon	8'0, 8'7, 7'8	Suso Cecchi d'Amico	Italia
Carrie	Brian de Palma	Sissy Spacek, Piper Laurie	7'4, 7'9, 7	Lawrence D. Cohen	EE.UU.
La lengua de las mariposas	José Luis Cuerda	Fernando Fernández Gómez	7'6, 7'6, 7'5	José Luis Cuerda	España
Alguien voló sobre el nido del cuco	Milos Forman	Jack Nicholson, Louise Fazenda	8'7, 8'3, 8'4	Bo Goldman, Lawrence Gordon	EE.UU.
To Kill a Mockingbird	Robert Mulligan	Gregory Peck, Mary Badham	8'3, 7, 8'3	Horton Foote	EE.UU.
Parque Jurásico	Steven Spielberg	Sam Neill, Samuel L. Jackson	8'1, 8, 7	Michael Crichton	EE.UU.
American Psycho	Mary Harron	Christian Bale, Will Ferrell	7'6, 7'4, 6'6	Mary Harron, Guillermo del Toro	EE.UU.
El Halcón Maltés	John Huston	Humphrey Bogart, Marlene Dietrich	8'8, 8'8, 8	John Huston	EE.UU.
Cadena Perpetua	Frank Darabont	Tim Robbins, Morgan Freeman	9'3, 8'6, 8'6	Frank Darabont	EE.UU.
Los hombres que no amaban a las mujeres	David Fincher	Daniel Craig, Rooney Mara	7'8, 7'8, 7	Steven Zaillian	EE.UU.
Doctor Zhivago	David Lean	Omar Sharif, Alec Guinness	8'4, 7'4, 7'9	Robert Bolt	EE.UU.
El diario de Bridget Jones	Sharon Maguire	Renée Zellweger	6'7, 6'6, 6'3	Helen Fielding, Anna Karenina	UK
Valor de ley	Joel Coen, Ethan Coen	Hailee Steinfeld, Matt Damon	7'6, 6'4, 7'1	Joel Coen, Ethan Coen	EE.UU.
La princesa prometida	Rob Reiner	Peter Falk, Robin Wright	8'1, 6'6, 7'4	William Goldman	EE.UU.
Sin novedad en el frente	Lewis Milestone	Louis Wolheim, Lew Ayres	8, 7'4, 7'8	Maxwell Anderson	EE.UU.
Tenemos que hablar de Kevin	Lynne Ramsay	Tilda Swinton, John C. Reilly	7'5, 7'2, 7'1	Lynne Ramsay	UK
Brokeback Mountain	Ang Lee	Anne Hathaway, Jake Gyllenhaal	7'7, 8, 7	Dianna Ossana, Ang Lee	EE.UU.
Hijos de los hombres	Alfonso Cuarón	Clive Owen, Julianne Moore	7'9, 7, 7'1	Alfonso Cuarón	UK
Grandes esperanzas	Alfonso Cuarón	Ethan Hawke, Gwyneth Paltrow	6'9, 8, 6'4	Mitch Glazer	EE.UU.
Blade Runner	Ridley Scott	Harrison Ford, Rutger Hauer	8'1, 9, 8'1	Hampton Fancher	EE.UU.
El Señor de los Anillos	Peter Jackson	Viggo Mortensen, Elijah Wood	8'9, 8'8, 8'2	Philippa Boyens	Nueva Zelanda
Drácula de Bram Stoker	Francis Ford Coppola	Gary Oldman, Winona Ryder	7'4, 7, 7'6	James V. Hart	EE.UU.
La lista de Schindler	Steven Spielberg	Liam Neeson, Ralph Fiennes	8'9, 9, 8'7	Steven Zaillian	EE.UU.
El almuerzo desnudo	David Cronenberg	Peter Weller, Judd Nelson	7'1, 8, 6'6	David Cronenberg	Canada

La insoportable levedad del ser	Philip Kaufman	Juliette Binoche, Daniel Day-Lewis	7'3, 6'3, 6'6	Jean-Claude Carrière	EE.UU.
El Imperio del Sol	Steven Spielberg	Christian Bale, John C. Reilly	7'8, 7'8, 7'4	Tom Stoppard	EE.UU.
Los santos inocentes	Mario Camus	Alfredo Landa, Terelu Campos	8'2, 7'4, 8'1	Mario Camus, María Casas	España
El Señor de las Moscas	Peter Brook	James Aubrey, Tom Stoppard	6'9, 6'2, 6'8	Peter Brook	UK
El Club de la Lucha	David Fincher	Brad Pitt, Edward Norton	8'8, 9, 8'1	Jim Uhls	EE.UU.
Chacal	Fred Zinnemann	Edward Fox	7'8, 7, 7'5	Kenneth Ross	UK
La heredera	William Wyler	Olivia de Havilland, Ingrid Bergman	8'2, 9, 8'2	Ruth Goetz, August Wilson	EE.UU.
Expiación	Joe Wright	James McAvoy, Keira Knightley	7'8, 8, 6'9	Christopher Hampton	UK
El sueño eterno	Howard Hawks	Humphrey Bogart, Lauren Bacall	7'9, 8, 8'1	William Faulkner	EE.UU.
Cuenta conmigo	Rob Reiner	Will Wheaton, River Phoenix	8'1, 7, 7'3	Bruce A. Evans	EE.UU.
Apocalypse Now	Francis Ford Coppola	Marlon Brando, Martin Sheen	8'4, 9'5, 8'3	John Milius, Francis Ford Coppola	EE.UU.
Lo que queda del día	James Ivory	Anthony Hopkins, Emily Watson	7'8, 8, 7'6	Ruth Prawer Jhabvala	UK
Trainspotting	Danny Boyle	Ewan McGregor, Ewan Bremner	8'1, 8, 8'1	John Hodge	UK
Inocencia interrumpida	James Mangold	Winona Ryder, Angela Petrelli	7'3, 5, 6'6	Susanna Kaysen	Alemania
La edad de la inocencia	Martin Scorsese	Daniel Day-Lewis, Michael Stuhlbarg	7'2, 8'6, 7	Jay Cocks, Martin Scorsese	EE.UU.
El truco final	Christopher Nolan	Christian Bale, Hugh Jackman	8'5, 8, 7'4	Christopher Nolan	EE.UU.
El resplandor	Stanley Kubrick	Jack Nicholson, Shelley Duvall	8'4, 8'8, 8'2	Stanley Kubrick	EE.UU.
No es país para viejos	Joel Coen, Ethan Coen	Javier Bardem, Tommy Lee Jones	8'1, 9, 7'1	Joel Coen, Ethan Coen	EE.UU.
El niño con el pijama de rayas	Mark Herman	Asa Butterfield, Jack P Shepherd	7'8, 8, 6'9	Mark Herman	UK
Orgullo y prejuicio	Joe Wright	Keira Knightley, Matthew Macfadyen	7'8, 8, 7	Deborah Moggach	UK
Las crónicas de Narnia	Andrew Adamson	William Moseley, Anna Popplewell	6'9, 5, 5'8	Ann Peacock, Christopher Tolkien	UK
El Diario de Noah	Nick Cassavetes	Ryan Gosling, Rachel Weisz	7'8, 7, 7'3	Jeremy Leven	EE.UU.
Charlie y la Fábrica de Chocolate	Tim Burton	Johnny Depp, Freddie Highmore	6'6, 6, 6'7	John August	EE.UU.
El curioso caso de Benjamin Button	David Fincher	Brad Pitt, Cate Blanchett	7'8, 8, 7'2	Eric Roth	EE.UU.
Forrest Gump	Robert Zemeckis	Tom Hanks, Robin Wright	8'8, 9, 8'2	Eric Roth	EE.UU.
Un paseo para recordar	Adam Shankman	Mandy Moore, Shailene Woodley	7'4, 4, 6'6	Karen Janszen	EE.UU.
Soy leyenda	Francis Lawrence	Will Smith, Alice Braga	7'2, 7, 6'5	Arkiva Goldsmith	EE.UU.
El perfume	Tom Tykwer	Ben Wishaw, Dustin Hoffman	7'5, 7, 6'7	Andrew Birkin	Alemania
El diablo viste de Prada	David Frankel	Meryl Streep, Anne Hathaway	6'9, 7, 5'7	Aline Brosh McKenna	EE.UU.
El Pianista	Roman Polanski	Adrien Brody	8'5, 8'8, 8'3	Ronald Harwood	UK

Y para los libros:

Libro	Escritor	País
El Padrino	Mario Puzo	Italia
El silencio de los c	Thomas Harris	EE.UU.
La Colmena	Camilo José Cela	España
La Carretera	Cormac McCarthy	EE.UU.
Nazarín	Benito Pérez Gald	España
El gatopardo	Giuseppe Tomasi	Italia
Carrie	Stephen King	EE.UU.
¿Qué me quieres,	Manuel Rivas	España
Alguien voló sobr	Ken Kesey	EE.UU.
Matar a un ruisel	Harper Lee	EE.UU.
Parque Jurásico	Michael Crichton	EE.UU.
American Psycho	Bret Easton Ellis	EE.UU.
El halcón maltés	Dashiell Hammett	EE.UU.
Rita Hayworth y l	Stephen King	EE.UU.
Millenium: Los he	Stieg Larsson	Suecia
Doctor Zhivago	Boris Pasternak	URSS
El diario de Bridg	Helen Fielding	UK
True Grit	Charles Portis	EE.UU.
La princesa prom	William Goldman	EE.UU.
Sin novedad en e	Erich Maria Remar	Alemania
Tenemos que hal	Lionel Shriver	EE.UU.
Brokeback Moun	Annie Proulx	EE.UU.
The Children of M	P.D. James	UK
Grandes esperanz	Charles Dickens	UK
¿Sueñan los andr	Philip K. Dick	EE.UU.
El Señor de los Ar	J. R. R. Tolkien	UK
Drácula	Bram Stoker	UK
El arca de Schindl	Thomas Keneally	Australia
El almuerzo desn	William S. Burrou	Francia

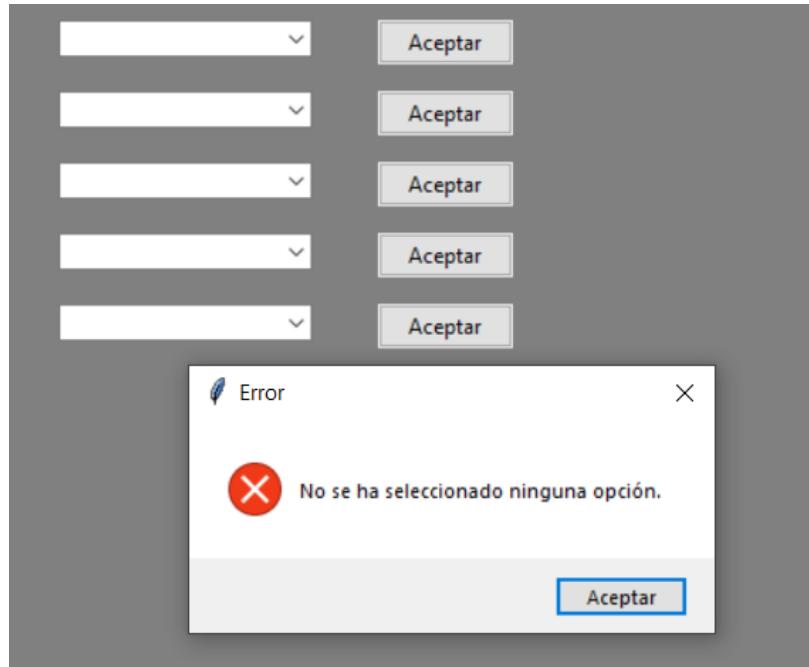
La insoportable le	Milan Kundera	Francia
El imperio del sol	J. G. Ballard	UK
Los santos inocen	Miguel Delibes	España
El Señor de las M	William Golding	UK
El Club de la Luch	Chuck Palahniuk	EE.UU.
El día del Chacal	Frederick Forsyth	UK
Washington Squa	Henry James	EE.UU.
Expiación	Ian McEwan	UK
El sueño eterno	Raymond Chand	EE.UU.
El cuerpo	Stephen King	EE.UU.
El corazón de las	Joseph Conrad	UK
Lo que queda del	Kazuo Ishiguro	UK
Trainspotting	Irvine Welsh	Escocia
Inocencia interru	Susanna Kaysen	EE.UU.
La edad de la ino	Edith Warton	EE.UU.
El truco final	Christopher Pries	UK
El resplandor	Stephen King	EE.UU.
No es país para vi	Cormac McCarthy	EE.UU.
El niño con el pija	John Boyne	Irlanda
Orgullo y prejuici	Jane Austen	UK
Las crónicas de N	C. S. Lewis	UK
The Notebook	Nicholas Sparks	EE.UU.
Charlie y la Fábric	Roald Dahl	UK
El curioso caso de	F. Scott Fitzgerald	EE.UU.
Forrest Gump	Winston Groom	EE.UU.
Un paseo para re	Nicholas Sparks	EE.UU.
Soy leyenda	Richard Matheson	EE.UU.
El perfume	Patrick Süskind	Alemania
El diablo viste de	Lauren Weisberg	EE.UU.
El pianista del eur	Wladyslaw Szpil	Polonia

Un puente hacia	Katherine Paterson	EE.UU.
El Código Da Vinc	Dan Brown	EE.UU.
La decisión de An	Jodi Picoult	EE.UU.
Romeo y Julieta	William Shakespeare	UK
Marley & Me: Lif	John Grogan	EE.UU.
Yo, Robot	Isaac Asimov	EE.UU.
El pasillo de la mu	Stephen King	EE.UU.
¿Quiere ser millor	Vikas Swarup	India
El diario de Ana F	Ana Frank	Países Bajos
La naranja mecán	Anthony Burgess	UK
La vuelta al mund	Jules Verne	Francia
El hombre biceps	Isaac Asimov	EE.UU.
El exorcista	William Peter Bla	EE.UU.
Jumper	Steven Gould	EE.UU.
La guerra de los m	H. G. Wells	UK
Sentido y sensibil	Jane Austen	UK
Corazón de tinta	Cornelia Funke	Alemania
El planeta de los s	Pierre Boulle	Francia
Lo que el viento se	Margaret Mitchell	EE.UU.
El amor en los tie	Gabriel García M	Colombia
Oliver Twist	Charles Dickens	UK
Como agua para ce	Laura Esquivel	México
Una mente marav	Sylvia Nasar	EE.UU.
El maravilloso ma	Lyman Frank Bau	EE.UU.
Los miserables	Victor Hugo	Francia
Ventana secreta,	Stephen King	EE.UU.
Mujercitas	Louisa May Alcott	EE.UU.
El nombre de la ri	Umberto Eco	Italia
Psicosis	Robert Bloch	EE.UU.
El retrato de Dori	Oscar Wilde	UK

Una vez explicado claramente de dónde sale la base de datos, y cómo ha sido el trabajo detrás del programa, podemos continuar con la explicación de la aplicación. Habíamos dejado la exposición en qué pasaba cuando apretábamos el botón “Aceptar”. Veíamos que se realizaba una consulta a la base de datos antes vista. Retomando la explicación, ahora que ya conocemos a dónde se realiza la consulta, ya podemos entender mucho mejor qué es lo que se está pidiendo. Cuando se activa el botón “aceptar”, dependiendo cuál sea la opción escogida en las cajas de la izquierda, se realiza una consulta. Si se elige director, se solicita información de todos los directores de todos los nodos de las películas. Si se elige guionista, se requiere información a la base de datos de todos los guionistas que tiene cada película. Si se hace del género, directamente se solicitan todos los nodos de tipo Genre. Una vez exigido a la base de datos esta información, se guarda en una lista todos los directores, guionistas, actores, géneros o países, dependiendo de lo que se haya seleccionado en la caja de la izquierda. Sin embargo, no todo fue tan simple, ya que, en algunos casos, hay actores, directores, guionistas o países, que los comparten muchas películas, por lo que fue necesario controlar que no se repitieran los elementos en las listas en dónde lo guardaba. Una vez controlado este detalle, ya están en funcionamiento todas las cajas de la derecha.



Además, me pareció interesante también controlar el hecho de que no se hubiese seleccionado ninguna opción. En caso de que el usuario no elija ninguna opción en la caja de la izquierda de algún botón “aceptar”, se le notificará que se está produciendo un fallo, y que no tiene ninguna característica seleccionada. Vemos un ejemplo.



Observamos que no se ha escogido ningún parámetro en la izquierda, por lo tanto, el sistema expone que hay un error del usuario, y que debe elegir una opción, y permite continuar con el programa.

Aquí llegamos al punto más importante del programa y el más relevante de explicar. ¿Cómo se llega a recomendar una película? Bien. Cada parámetro que introducimos, tendrá asignado un coeficiente, de manera que, la caja de la parte más superior tendrá establecido coeficiente 1. Iremos descendiendo en orden, y cada una de las cajas tendrá un coeficiente cada vez más pequeño. La segunda caja, empezando por arriba, obtendrá coeficiente 0.8, la tercera caja, 0.6, la cuarta 0.4 y la última, la de la parte más inferior, 0.2. Iremos filtrando las películas, asignando un coeficiente y se irá acumulando una suma.

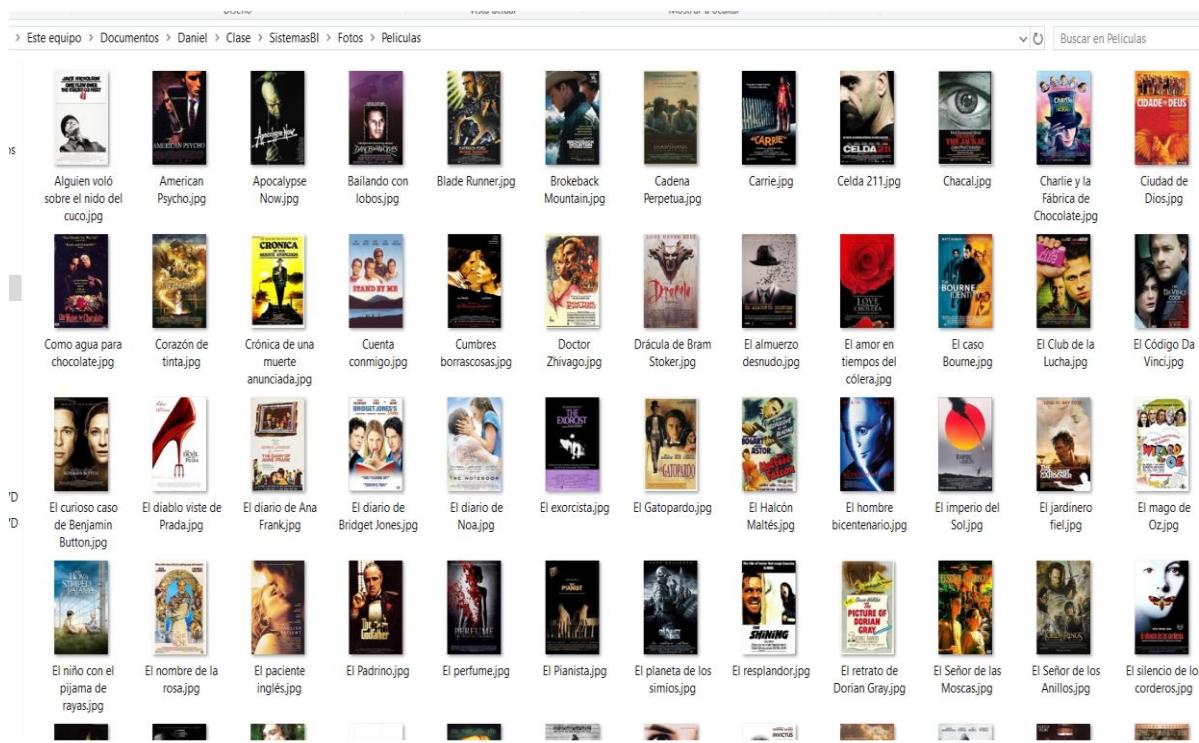
Género	Aceptar	Thriller
Director	Aceptar	Martin Scorsese
Actor	Aceptar	Morgan Freeman
País	Aceptar	EE.UU.
Guionista	Aceptar	Steven Zaillian
		BUSCAR
		NUEVA BÚSQUEDA

Vamos a realizar un análisis de cómo sería, con un ejemplo simple, el funcionamiento del programa. En este caso, el usuario ha escogido el género Thriller como primera característica. El programa realiza una consulta a la base de datos y solicita que le devuelva

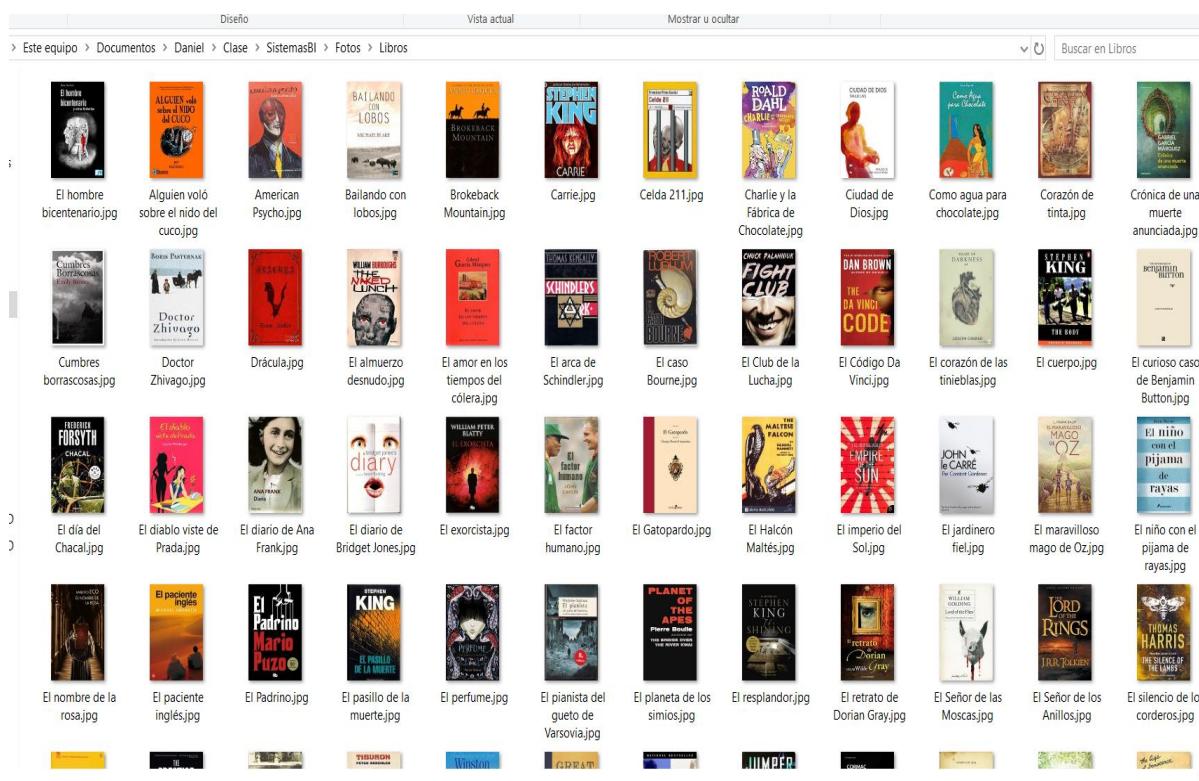
todas las películas que sean del género Thriller, y les asigna el coeficiente 1, por ser la primera. Luego volverá a pedirle a la base de datos todas las películas que se encuentren en la base de datos que haya dirigido Martin Scorsese, y les pondrá un 0.8. En caso de que alguna de las películas ya haya sido seleccionada dentro del primer apartado, y por ello, tenga ya un 1 asignado, se suma $1 + 0.8$, resultando 1.8. A continuación, seguirá adelante solicitando recibir todas las películas en las cuales Morgan Freeman, haya actuado. Todas las películas en las que esté este actor, recibirá un 0.6, acumulándose a la suma que ya antes tengan las películas anteriores devueltas. En este caso hay tres opciones distintas. Que sea una película nueva y entonces tendrá un 0.6; que reúna los tres requisitos, y entonces será $1+0.8+0.6$, resultado 2.4; y el último caso, que sólo tenga dos requerimientos, que entonces será, o bien $1+0.8$, si son los dos primeros, o $0.8+0.6$, si son los dos segundos. Después, y exactamente igual que en los ejemplos anteriores, la base de datos le da la lista completa de todas las películas de origen Estados Unidos, y asigna un 0.4, y repetimos el procedimiento de ir sumando en caso de que ya tengan cierta puntuación. Finalmente, se devolverá la lista de películas con Steven Zaillian de guionista y se determina un 0.2 y lo suma al resultado que se tenga. Por lo tanto, la máxima puntuación que pueda obtener una película será de $1+0.8+0.6+0.4+0.2$, es decir, 3 puntos. El sistema está realizado con esos coeficientes, con el objetivo de que se pueda siempre compensar una elección alta con elecciones más bajas. Por ejemplo, si una película es devuelta en la segunda y la tercera elección del usuario, pero no en la primera, y esta primera no es devuelta en la segunda y la tercera elección, la suma será mayor en el primer caso. Es decir, en el primer caso la suma será 1.4, y en el otro caso 1. Los coeficientes compensan, esto lo hace un programa justo y afina la búsqueda y la recomendación, y lo hace acorde a lo que el usuario desea ver.

Una vez que el usuario ya ha hecho su elección, y ha decidido cuáles son los criterios que quiere elegir para ver la película, debe pulsar el botón “Buscar”. Una vez que el botón “Buscar” es activado, es cuando se llevan a cabo todas las consultas y la asignación de coeficientes. Cuando todo este cálculo está finalizado se pasa al momento del cual se selecciona la película final. De todas las películas a las cuales se les ha asignado un coeficiente, es decir, tienen una suma, se busca la que tenga la mayor suma. Para ello, buscamos entre todas las valoraciones, obtenemos la más alta, y, a partir de ahí devolvemos la película. Una vez que la película con más puntuación está localizada, pasamos al momento en el cual imprimimos toda la información. Entre toda esta información, se especifica el título, la valoración media de la película, y el libro en el cual está basada, además del cartel de la película, y la portada del libro. Como tenemos ya la película, dar su título es sencillo. Sin embargo, para mostrar el resto, debemos realizar ciertos cálculos y ciertas operaciones con el código. Para la valoración general, hemos introducido antes en la base de datos, tres notas. Una de FilmAffinity, otra de SensaCine y otra de IMBD. El programa consulta en la base de datos las tres valoraciones de la película elegida, y realiza una media aritmética de las tres. Una vez que tiene la media, la devuelve como la valoración general de la película. Además, realiza otra consulta, a partir de la película ya seleccionada, el programa pide a Neo4J que le devuelva el libro en el cual se basa y, una vez recibida, imprime el título del libro. Llegados a este punto, consideramos el hecho de que ya tenemos el título de la película y del libro. Con ello ya podemos imprimir también las imágenes del cartel y de la portada. Para ello, hemos tenido que buscar todos los carteles de las películas y las portadas en dos carpetas distintas, para desde el código, copiar la ruta, con el nombre de la película y del libro, de forma separada y respectivamente, que ya teníamos antes.

Aquí podemos ver la carpeta de los carteles de las películas:



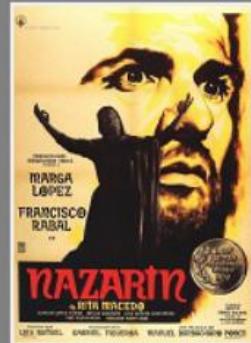
Y la de las portadas de los libros:



Y aquí como se muestra en la aplicación:

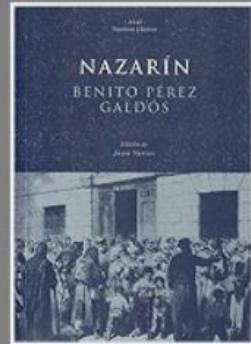
BUSCAR **NUEVA BÚSQUEDA**

Según los criterios introducidos, la película que recomendamos es: Nazarín

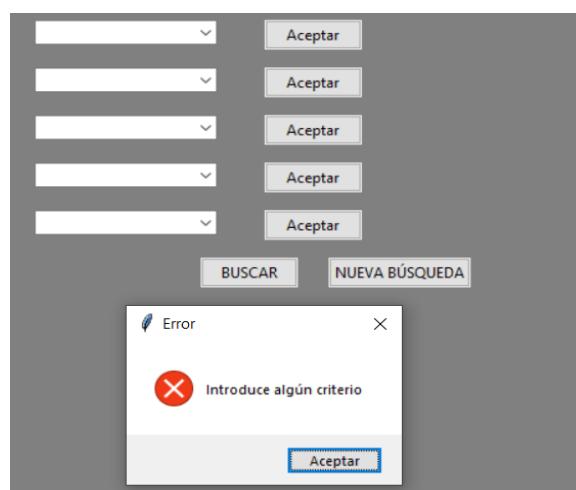
**INFORMACIÓN**

La valoración media de esta película es: 7.8

La película está basada en el libro Nazarín de Benito Pérez Galdós

**INFORMACIÓN**

Si cuando apretamos el botón “Buscar” no se ha introducido ningún criterio, el programa nos muestra un mensaje de error, y la aplicación se vuelve a lanzar. Esto es exactamente lo que hace el botón “Nueva Búsqueda”. Cuando el usuario finaliza la búsqueda, no podrá realizar otra hasta que pulse el botón “Nueva Búsqueda”. Lo único que hará este botón será cerrar la ventana actual y el programa, por tanto, y ejecutar de nuevo la aplicación, reiniciarla.

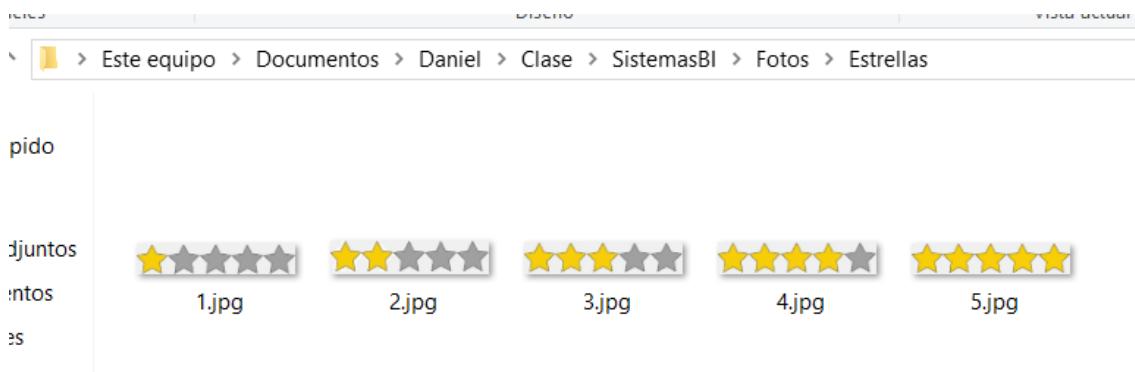


Observamos que aparecen dos nuevos botones, después de realizar la búsqueda. Dos botones de Información. El cometido de estos dos botones es muy simple, muestra, en una nueva ventana, toda la información relativa a la película recomendada y al libro en el que está basado, respectivamente. Cuando se pulsa cualquiera de estos dos botones, se abre una ventana nueva, se consultan todos los datos de la película y el libro a la base de datos y se muestran todos ellos. En las nuevas ventanas además también se podrá ver el cartel y la portada de la película y el libro, respectivamente en cada ventana.

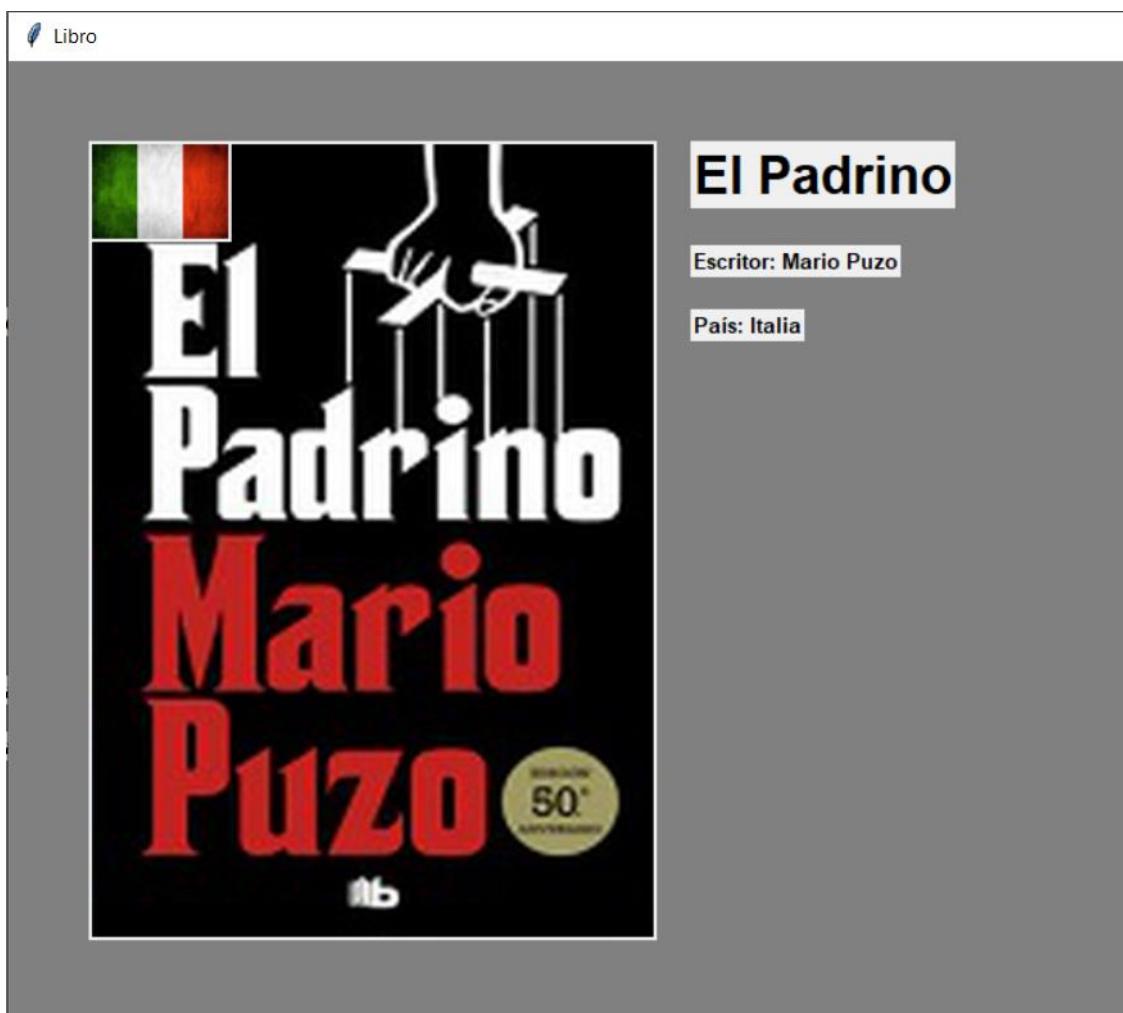
Esta sería la ventana de información de la película:



En ella vemos ciertos aspectos nuevos. Detrás de la valoración general, vemos una valoración de estrellas. Dependiendo de la nota media se mostrarán más estrellas o más. Como esto no era una posibilidad que ofreciese Tkinter ni Python, se me ocurrió hacerlo con imágenes. Según la valoración se pedirá una imagen u otra de estrellas, previamente guardadas en una carpeta.



Y la del libro:



Hay un elemento común a ambas ventanas que es la bandera del país. En la esquina superior izquierda del cartel y la portada del libro, situamos la bandera del país de procedencia. El procedimiento es el mismo que para el resto de imágenes introducidas dentro de la aplicación. Guardamos imágenes en una carpeta y, conociendo, después de la consulta, el país del libro y de la película, buscamos en la carpeta contenedora de las imágenes, la imagen correspondiente.



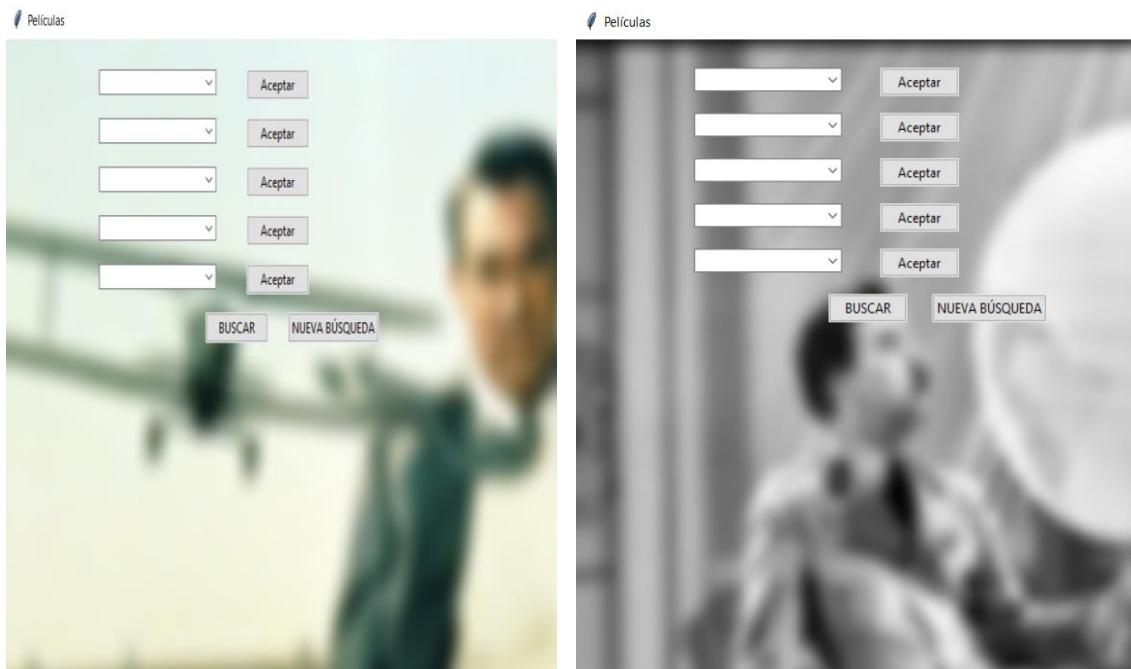
El método para encontrar las imágenes dentro de las carpetas es nombrar las imágenes dentro de las carpetas de la misma manera que están guardadas en la base de datos. De esta manera, sólo con introducir el nombre del título de la película, del país o del libro, tal y como viene en la base de datos, y colocar el nombre correcto de la ruta en la cual se encuentra la carpeta, será suficiente.

Finalmente, hemos colocado una imagen de fondo, la misma para todas las ventanas. Pero la diferencia esta vez es que hemos querido hacerlo de manera aleatoria, es decir, que se eligiese cada vez una nueva foto de fondo dentro de unas cuantas existentes en una carpeta. En una carpeta introducimos numerosas imágenes, en este caso hemos seleccionado imágenes relativas al cine. Como queremos colocar una imagen distinta cada vez que se ejecute la aplicación, generamos un número aleatorio del 1 al número total de imágenes que hay, cada vez que el programa se inicia. En la carpeta las imágenes están nombradas con números del 1 en adelante. De esta manera si el programa saca el número 3, el fondo será la imagen con el nombre 3.

He aquí la carpeta:



Y aquí las imágenes finales de la aplicación con distintas imágenes de fondo:



Finalmente, vemos las tres ventanas de la aplicación:

Ventana principal:

Películas

Director	Aceptar	Francis Ford Coppola
Actor	Aceptar	Al Pacino
	Aceptar	
	Aceptar	
	Aceptar	
	BUSCAR	NUEVA BÚSQUEDA

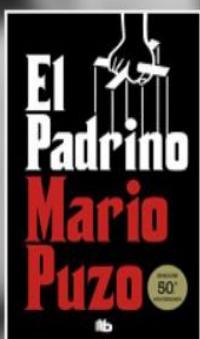
Según los criterios introducidos, la película que recomendamos es: El Padrino



INFORMACIÓN

La valoración media de esta película es: 9.6

La película está basada en el libro El Padrino de Mario Puzo



INFORMACIÓN

Ventana de información de película:



Ventana de información de libro:



Casos y Sensibilidad

En este apartado valoraremos un caso práctico, así como una variante del caso propuesto, para demostrar la valía del programa y su correcto funcionamiento. Para ello lo ejecutaremos e iremos explicando, con imágenes de la aplicación, cómo cambian los resultados.

Según los criterios introducidos, la película que recomendamos es: *Los hombres que no amaban a las mujeres*

INFORMACIÓN

La valoración media de esta película es: 7.53

La película está basada en el libro *Millenium Los hombres que no amaban a las mujeres* de Stieg Larsson

INFORMACIÓN

Observamos que hemos seleccionado tres criterios distintos, en este caso el género de la película, el actor principal y el director. La película que nos recomienda es “*Los hombres que no amaban a las mujeres*”. En este caso esta película coincide perfectamente con lo pedido, la suma de la película sería $1+0.8+0.6$, es decir 2.4, la puntuación más alta con tres requisitos solo. La aplicación hasta el momento cumple perfectamente los requisitos. Vamos a realizar un pequeño cambio en los parámetros pedidos. En este caso tan sólo cambiaremos el actor, y ya se producirá el primer cambio.

Género Aceptar Drama

Actor Aceptar Brad Pitt

Director Aceptar David Fincher

Aceptar

Aceptar

BUSCAR **NUEVA BÚSQUEDA**

Según los criterios introducidos, la película que recomendamos es: El Club de la Lucha



INFORMACIÓN

La valoración media de esta película es: 8.63

La película está basada en el libro El Club de la Lucha de Chuck Palahniuk



INFORMACIÓN

Comprobamos que tan solo con un cambio, la recomendación cambia totalmente y se ofrece una nueva película más acorde. En este caso, también la película recomendada tiene todos los puntos posibles, ya que cumple con todas las características.

A pesar de ello, vamos a poner un ejemplo en el cual los dos segundos parámetros recomiendan una película que no cumpla el primero. Con ello vamos a demostrar que también el sistema es justo y recomienda de forma correcta y precisa.

Como vemos en el ejemplo de debajo, la película que sale recomendada no tiene nada que ver con Drama Histórico. La película es una comedia romántica, y el hecho de aportar en la segunda y tercera opción dos parámetros comunes entre ellos, supera el coeficiente de la primera caja.

RECHAZAR

Género	Aceptar	Historical Drama
Actor	Aceptar	Ryan Gosling
Director	Aceptar	Nick Cassavetes
	Aceptar	
	Aceptar	
	Aceptar	

BUSCAR **NUEVA BÚSQUEDA**

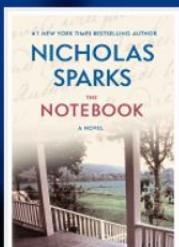
Según los criterios introducidos, la película que recomendamos es: El diario de Noa



INFORMACIÓN

La valoración media de esta película es: 7.37

La película está basada en el libro The Notebook de Nicholas Sparks



INFORMACIÓN

Con esto concluimos finalmente la demostración del caso y de las variaciones, que demuestran que el sistema es correcto, y funciona perfectamente tal y como lo habíamos diseñado.

Análisis crítico

Para llevar a cabo un análisis crítico de la aplicación y del programa usaremos un tipo de análisis muy común, llamado DAFO. El análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) también conocido como análisis FODA, es una herramienta de estudio de la situación de una empresa o un proyecto, de manera que analiza las características tanto internas (Debilidades y Fortalezas) como la situación externa (Amenazas y Oportunidades) en una matriz cuadrada. Con ella podremos ver y conocer la situación real en el que se encuentra un proyecto.

DAFO

Debilidades: Elementos de la aplicación que determinan cierto atraso. Entre ellas encontramos:

- Aplicación muy selectiva.
- Ideada para un segmento de personas muy concreto.
- Base de datos escasa.
- No hay posibilidad de ver todas las películas de la base de datos.
- Falta de implementación debido al tiempo.

Amenazas: Elementos externos que constituyen una amenaza para el proyecto:

- Ya hay otras aplicaciones muy parecidas.
- La recomendación no se hace a partir de conocimiento adquirido.
- El sistema de recomendación de otras plataformas es más personal.
- La competencia no sólo es fuerte si no que es muy conocida.
- La competencia está en la web, más accesible y más rápida.

Fortalezas: Aquellos elementos que constituyen la parte más fuerte de la aplicación:

- Es una aplicación muy útil para las personas más aficionadas al cine.
- La interfaz es sencilla, intuitiva y atractiva.
- Datos muy concretos y corroborados.
- Posibilidad de filtrar y buscar por elementos distintos a los que suelen proponerse.

Oportunidades: Factores positivos que se generan en el entorno y que pueden ser aprovechados:

- Sistemas de recomendaciones cada vez más al alza.
- Plataformas digitales cada vez más requeridas.
- Sistemas de recomendación muy distintos al propuesto, para personas que busquen ciertos elementos cinéfilos más concretos.
- Las páginas webs en las cuales se pueden buscar datos cinematográficos, no tienen implementado un sistema de recomendación realmente eficaz y orientado a personas cinéfilas.

D	A	F	O
DEBILIDADES	AMENAZAS	FORTALEZAS	OPORTUNIDADES
<ul style="list-style-type: none"> - Aplicación muy selectiva - Segmento muy concreto de personas - Base de datos escasa - Tiempo escaso para implementar todo 	<ul style="list-style-type: none"> - Aplicaciones existentes muy parecidas - Competencia fuerte, conocida, más accesible y más rápida - Recomendación no adquirida a través de conocimiento adquirido - Sistema de recomendación más personal en otras plataformas 	<ul style="list-style-type: none"> - Aplicación muy útil para cinéfilos - Interfaz sencilla, intuitiva y atractiva - Datos concretos y corroborados - Posibilidad de filtrar por elementos innovadores no comunes y no disponibles en otras aplicaciones 	<ul style="list-style-type: none"> - Sistemas de recomendación y plataformas digitales de cine cada vez más al alza - Webs cinematográficas sin sistema de recomendación eficaz - Sistema innovador y distinto a los ya conocidos

Línea de futuro

Como hemos explicado en el DAFO, nuestra principal limitación y que no nos permitió alcanzar cotas de mayor perfección en la aplicación, resultó ser el tiempo, escaso y acotado de 3, 4 meses que tuvimos para realizar todo el trabajo. Sin embargo, estoy muy contento con el trabajo tanto en la parte de la base de datos como de la parte de código fuente. A pesar de ello, han quedado ciertas implementaciones sin hacer que me hubiese gustado tener. En principio, la línea de futuro que trazaría de forma más inmediata sería la corrección de algunos pequeños errores que todavía persisten y que no he podido solventar, aunque no repercuten en el funcionamiento del programa. Como la fuerza de la aplicación reside en la base de datos, potente y actualizada, podemos asegurar al 100% que la aplicación es escalable, si lo es la base de datos, y una base de datos, es por definición escalable. Esto constituye la segunda línea de futuro más inmediata. La aplicación carece de precisión absoluta debido al número escaso de películas de la base de datos, pero esto es fácilmente solucionable. La precisión aumentaría muy considerablemente cuanto más grande hiciésemos la base de datos, cuantas más películas se introdujesen, más libros, más relaciones etc. Esto parece un detalle banal, pero no lo es, ya que, los resultados podrían cambiar muy considerablemente si en vez de comparar y filtrar entre cien películas, lo hiciésemos entre mil. Más películas por director, por guionista, por género...crearían una “lucha” real entre películas para salir. Además, también me resultaría interesante implementar una ventana nueva, con la cual pudiésemos ver la lista entera de películas y libros que están introducidos en la base de datos. Programar esto no es excesivamente complicado, pero la falta de tiempo me impidió hacerlo. Como viene intuyéndose en este apartado del trabajo, podemos ver una ligera tendencia a despegarme de tener una base de datos de películas sólo basadas en libros, para introducirme en el mundo de las películas no basadas en libros y en cine de autor.

Por último, me interesaría poder plasmar todo este trabajo al entorno gráfico de una aplicación web. Poder introducir el programa en una página web presenta un desafío bastante complicado debido a que debemos pasar todo el programa en Python, el lenguaje en el cual está programado, a HTML, CSS y JSON, los lenguajes que se utilizan mayoritariamente en aplicaciones webs. El motivo de este interés es el hecho de poder acceder de forma más rápida y sencilla a la aplicación programada. Siempre es mucho más sencillo acceder a una página web, que a un programa en el cual debemos ejecutar Neo4J, Anaconda...etc.

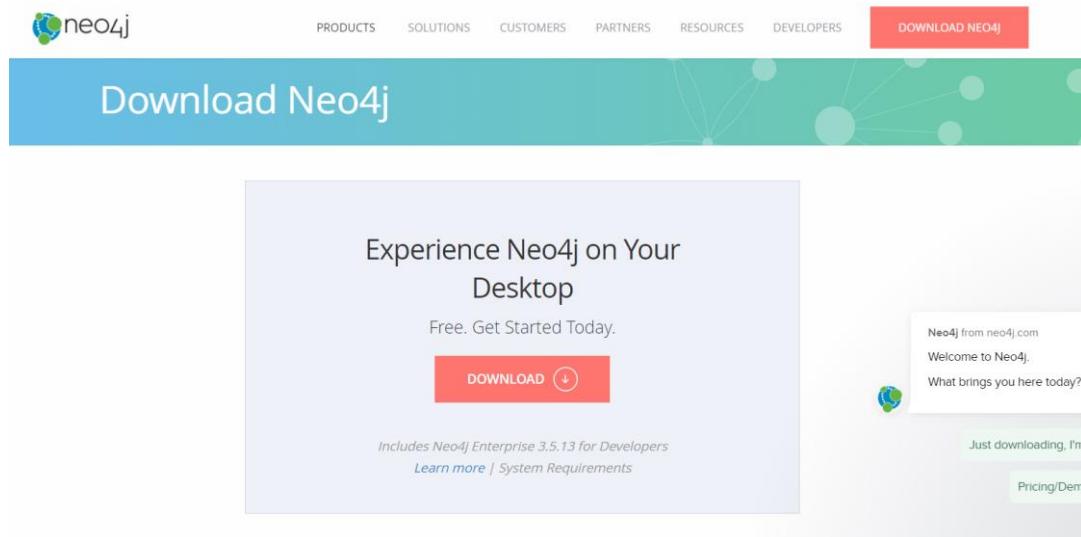
Bibliografía

- Información de películas: todo buscado en distintas webs citadas aquí debajo:
 - <https://www.imdb.com/>
 - <http://www.sensacine.com/>
 - <https://www.filmaffinity.com/es/main.html>
 - <https://es.wikipedia.org/>
- Información de los libros:
 - <https://es.wikipedia.org/>
- Imágenes de libros y películas:
 - <https://www.google.es/imghp?hl=es> – Imágenes de Google
- Información sobre Cypher y Python:
 - <https://www.tutorialspoint.com/python>
 - <http://xurxodeveloper.blogspot.com/2014/04/cypher-como-crear-datos.html>
 - <https://neo4j.com/>
 - <https://es.stackoverflow.com/>
 - <https://pythonbasics.org/>
 - <https://www.python-course.eu/>
- Información sobre Neo4J:
 - https://www.youtube.com/watch?v=k3h_y9w_7I4
 - <https://www.youtube.com/watch?v=U8ZGVx1NmQg>
 - <https://www.youtube.com/watch?v=Go3P73-KV30>
 - https://www.youtube.com/watch?v=H2yC_UiHopc
 - <https://www.youtube.com/watch?v=fiQzHVY-kRQ>
 - https://www.youtube.com/watch?v=i_suTdfQJ8U
 - <https://www.youtube.com/watch?v=pMjwgKqMzi8>
 - <https://es.wikipedia.org/wiki/Neo4j>
 - <https://neo4j.com/graphacademy/>
 - <https://neo4j.com/graphacademy/online-training/introduction-to-neo4j/part-0/>
 - <https://neo4j.com/graphacademy/online-training/introduction-to-neo4j/part-1/>
 - <https://neo4j.com/graphacademy/online-training/introduction-to-neo4j/part-2/>
 - <https://neo4j.com/graphacademy/online-training/introduction-to-neo4j/part-3/>
 - <https://neo4j.com/graphacademy/online-training/introduction-to-neo4j/part-4/>
 - <https://neo4j.com/graphacademy/online-training/introduction-to-neo4j/part-5/>
 - <https://neo4j.com/graphacademy/online-training/introduction-to-neo4j/part-6/>
 - <https://neo4j.com/graphacademy/online-training/introduction-to-neo4j/part-7/>
 - <https://neo4j.com/graphacademy/online-training/introduction-to-neo4j/part-8/>

Anexo: Cuaderno de Trabajo

En este anexo presentaremos el cuaderno de trabajo de la asignatura, una especie de cuaderno de bitácora en el cual he ido reflejando de forma cronológica todo el proceso de aprendizaje primero y posterior estructuración del trabajo que he llevado a cabo.

En primer lugar, fue necesario entrar en contacto con el entorno de Neo4J, la aplicación que iba a usar, así como ponerme al día de las bases de datos con grafos de conocimiento. Para ello, comencé por instalar el entorno Neo4J



Desde la siguiente pantalla accedí a la solicitud de registro de Neo, donde tuve que llenar la siguiente información:

Get Started Now

Please fill out this form to begin your download

* First Name

* Last Name

* Business Email

* Company Name

* Country

By downloading you agree to the [Neo4j License Agreement for Neo4j Desktop Software](#).

The information you provide will be used in accordance with the terms of our [privacy policy](#).

Tras esto descargué un ejecutable con el cual pude iniciar la instalación de Neo. El proceso fue sencillo y bastante intuitivo, con el cual pude rápidamente empezar a usar Neo.

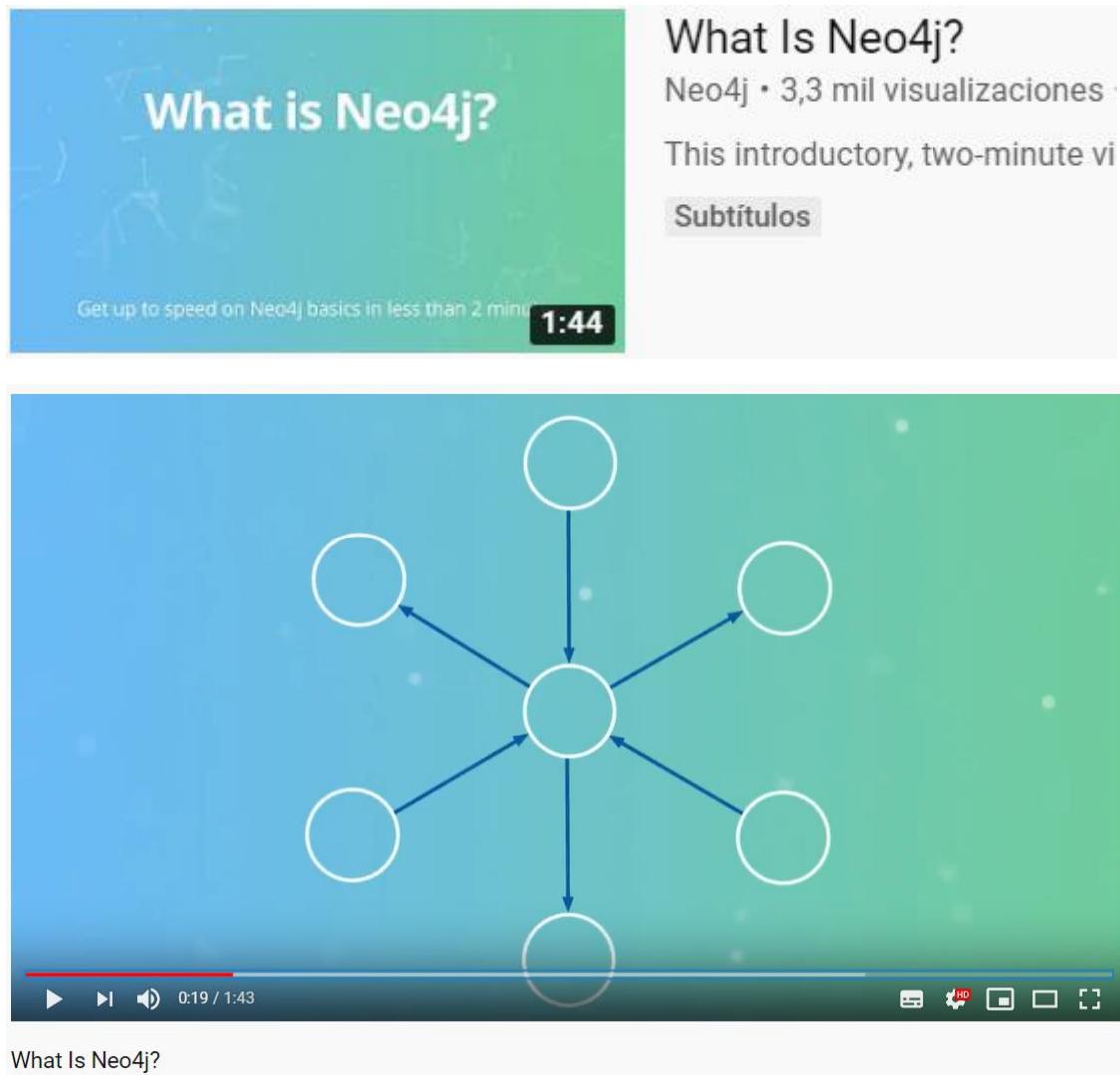
Como el sistema de grafos de conocimiento mediante nodos, así como el entorno visual y las propiedades de Neo4J eran totalmente ajenas a mi conocimiento, comencé a buscar información antes de meterme de lleno con ello. El primer lugar donde investigué fue en YouTube. Allí encontré muchos videos y entre otros canales, el oficial de Neo4J, con incesante información acerca de su propia plataforma. Sin embargo, ahora me encontraba en la disyuntiva inversa a la anterior. Si bien antes no tenía excesiva información y todo era nuevo para mí, tras realizar esta búsqueda, encontré demasiada información. Horas y horas de videos dentro del canal oficial, así que me dispuse a filtrar toda esa información para encontrar sólo la realmente útil en el punto en el cual me encontraba.

Primero necesitaba saber qué era eso de Neo4J, y después comenzar a usarlo. Para ello, uno de los vídeos más aclaratorios que encontré fue el siguiente:



En él, se hacía una pequeña introducción rápida a las bases de datos de grafos de conocimiento con nodos, y como bien indica, de Neo4J. Esto me sirvió de ejemplo inicial para entender de qué se trataba Neo4J y cómo iba a poder trabajar con él. Así mismo, pude construir una idea inicial en mi cabeza de cuáles eran las posibilidades de que se ofrecían ante

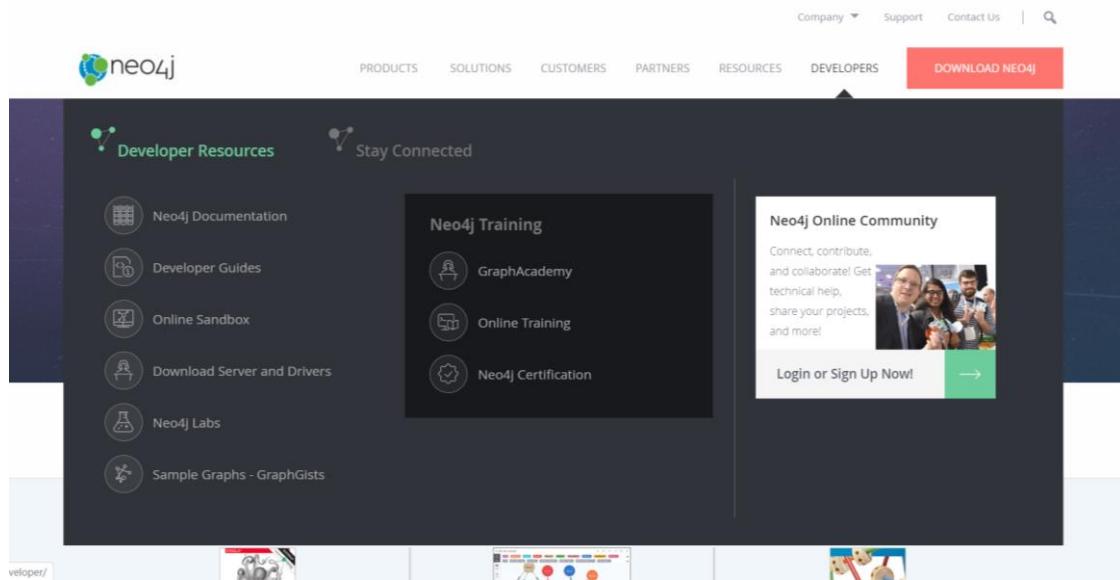
mí de cara al trabajo final. Además, para terminar de tener claros todos los aspectos, visualicé el video introductorio del propio canal de Neo4J. Un video muy cortito de 2 minutos en el cual se establecen los principios más básicos explicado por los propios creadores de Neo.



What Is Neo4j?

Una vez que el concepto de bases de datos de grafos de conocimiento basado en nodos estaba conciso en mi cabeza, comenzó el proceso de hacer que el programa fuese familiar para mí, comprender cómo funcionaba y cómo debía utilizarlo.

Para ello, dentro de la propia página de Neo4J, se encuentran una serie de tutoriales, posteriormente certificados con los cuales podía poner en marcha esa parte.



En ese menú desplegable, opté por la opción GraphAcademy, donde se encuentran los tutoriales previamente explicados.

Free Online Training & Certification

[Learn All About Neo4j](#) and how to [Launch Neo4j Into Production](#) when you take these Free online courses.

After completing the Introduction to Neo4j course, take the [Free 1-hour Neo4j Certification exam](#).

Introduction to Neo4j

Get started quickly – learn about Graph Databases, Neo4j, and Cypher – the Graph Query Language.

[START TRAINING](#)

Neo4j Administration

Learn deployment best practices and how to successfully launch Neo4j in a production environment.

[START TRAINING](#)

Data Science with Neo4j

Learn how to use Neo4j as part of your Data Science and Machine Learning workflows using an aminer.org dataset.

[START TRAINING](#)

Applied Graph Algorithms

Learn how to use graph algorithms in Neo4j, and also how to apply them to enhance web application functionality.

[START TRAINING](#)

Una vez llegados a este punto, decidí comenzar lógicamente por la introducción. Con ella aprendí rápidamente los conceptos más importantes de Neo4J. Aunque con los videos introductorios ya preparé el camino para comprender mejor estos cursillos rápidos, llevar a cabo la parte introductoria de Neo, fue vital para saber manejar todo lo necesario.

En los dos primeros apartados del tutorial se introducen las bases de datos de grafos y Neo4J, los dos aspectos en los que me había documentado antes en otras plataformas.

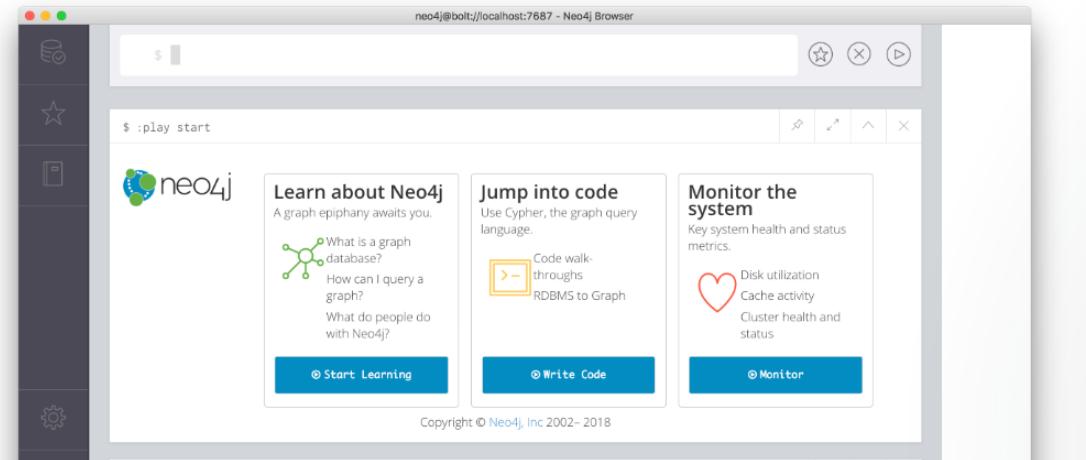
Posteriormente, se me explicaba el entorno gráfico de Neo4J Desktop, la aplicación de escritorio de Neo.

Using Neo4j Browser

Neo4j Browser is a tool that enables you to access a Neo4j Database by executing Cypher statements to create a graph to return data. The data returned is typically visualized as nodes and relationships in a graph, but can also be used to execute a number of system calls that are related to the database being retrieved. To retrieve the list of queries that are currently running in the server.

There are two ways that you can use Neo4j Browser functionality:

- Open the Neo4j Browser application from Neo4j Desktop (database is local)
- Use the Neo4j Browser Web interface by specifying a URL in a Web browser using port 7474 (database is remote)



Una vez entendidos todos los aspectos que son necesarios para comprender cómo trabaja la interfaz gráfica de Neo, llegamos a la parte más importante para mí, Cypher. Este fue el apartado más relevante en cuanto al aprendizaje de cómo usar Neo. Se divide en tres partes:

1. Introducción a Cypher: En ella podremos aprender a:

- Encontrar nodos del grafo
- Filtrar nodos encontrados usando etiquetas y propiedades de los nodos
- Encontrar atributos de los nodos dentro del grafo
- Filtrar nodos encontrados utilizando relaciones

Introduction to Cypher

About this module

Cypher is the query language you use to retrieve data from the Neo4j Database, as well as create and update the data.

At the end of this module, you should be able to write Cypher statements to:

- Retrieve nodes from the graph.
- Filter nodes retrieved using labels and property values of nodes.
- Retrieve property values from nodes in the graph.
- Filter nodes retrieved using relationships.

Throughout this training, you should refer to:

- [Neo4j Cypher Manual](#)
- [Cypher Reference card](#)

2. Consultas:

- Filtrar consultas utilizando WHERE
- Conocer el procesamiento de consultas
- Conocer y comprobar los resultados obtenidos
- Trabajar con datos, fechas y listas de Cypher

Getting More Out of Queries

About this module

You have learned how to query nodes and relationships in a graph using simple patterns. You learned how to use node labels, relationship types, and properties to filter your queries. Cypher provides a rich set of `MATCH` clauses and keywords you can use to get more out of your queries.

At the end of this module, you should be able to write Cypher statements to:

- Filter queries using the `WHERE` clause
- Control query processing
- Control what results are returned
- Work with Cypher lists and dates

3. Creación de datos:

- ***Crear un nodo:***
 - Añadir y eliminar etiquetas de los nodos
 - Añadir y eliminar propiedades de los nodos
 - Actualizar propiedades de los nodos
- ***Crear una relación:***
 - Añadir y eliminar propiedades de una relación
- ***Eliminar un nodo***
- ***Eliminar una relación***
- ***Unir datos de un grafo***
 - Crear nodos
 - Crear relaciones

About this module

You have learned how to query a graph using a number of Cypher clauses and keywords that start with the `MATCH` clause. You learned how to retrieve data based upon label values, property key values, and relationship types, and how to perform some useful intermediate processing during a query to control what data is returned.

At the end of this module, you should be able to write Cypher statements to:

- Create a node
 - Add and remove node labels
 - Add and remove node properties
 - Update properties
- Create a relationship
 - Add and remove properties for a relationship
- Delete a node
- Delete a relationship
- Merge data in a graph
 - Creating nodes
 - Creating relationships

Finalmente, tenemos una última pieza, que nos enseña:

- Usar parámetros en las declaraciones de Cypher
- Analizar cómo se ejecuta Cypher
- Consultas al monitor
- Manejar y gestionar restricciones y claves de los nodos para el gráfico
- Importar datos de un grafo en archivos CSV
- Manejar y gestionar índices de un grafo
- Acceder a los recursos adicionales de Neo4J

Getting More Out of Neo4j

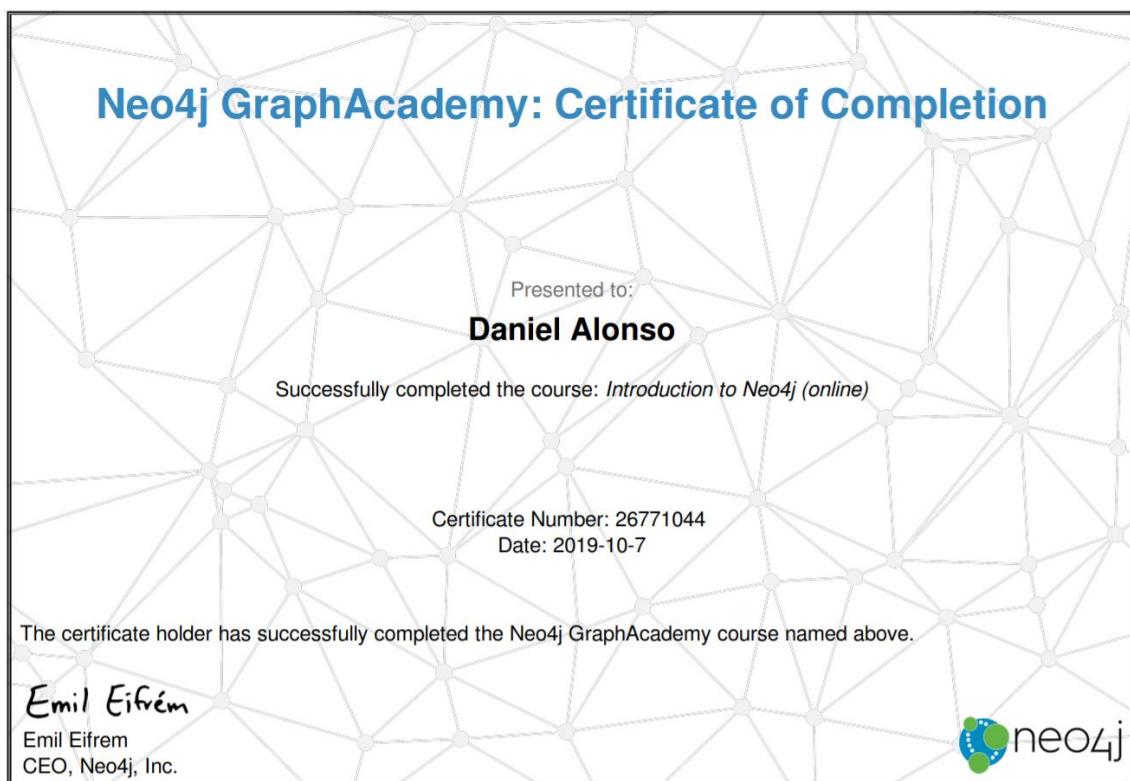
About this module

You have learned how to set up your development environment for accessing a Neo4j graph and how to write basic Cypher statements for querying the graph modifying the graph.

At the end of this module, you should be able to:

- Use parameters in your Cypher statements.
- Analyze Cypher execution.
- Monitor queries.
- Manage constraints and node keys for the graph.
- Import data into a graph from CSV files.
- Manage indexes for the graph.
- Access Neo4j resources.

Al final de cada una de las partes, hay una serie de preguntas, las cuales tenemos que responder correctamente para continuar adelante y para obtener el certificado final que acredita que hemos completado la introducción de Cypher. Después de solicitarlo, podemos descargar el diploma que certifica que hemos consumido esta parte de la GraphAcademy.

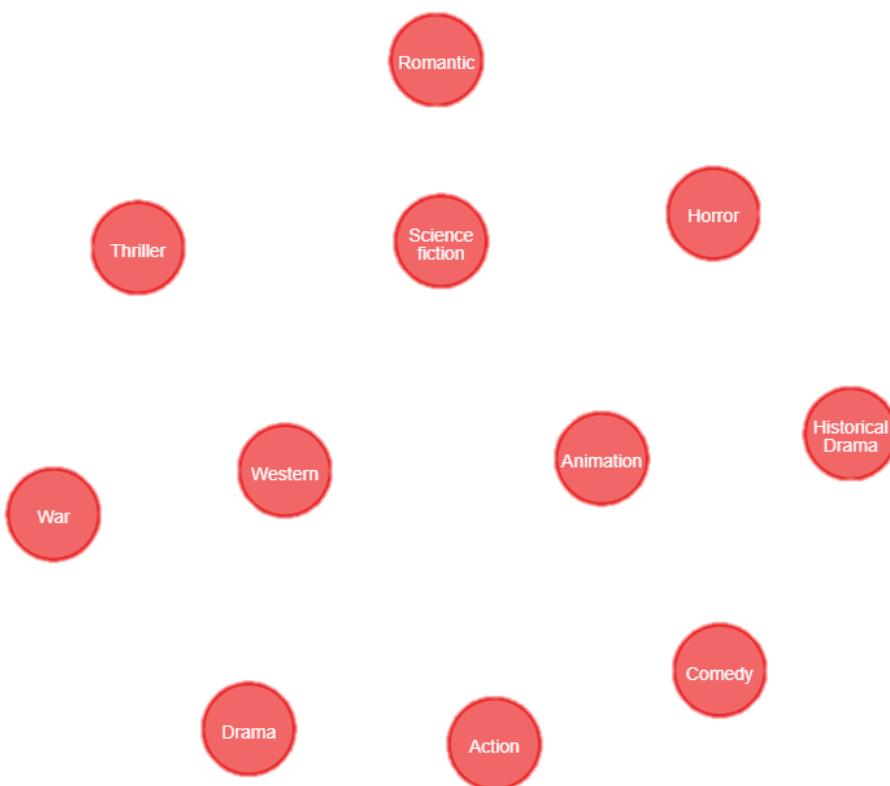


Ahora que ya conocemos los aspectos necesarios para llevar a cabo una base de datos de grafos de conocimiento, así que llegados a este punto era necesario elegir un tema en el cual centrar mi trabajo final. Después de valorar numerosas opciones, decidí decantarme por realizar una base de datos de grafos orientada a películas y libros, sobretodo películas basadas en libros. Mi finalidad era construir un sistema de recomendación de películas y libros a partir de datos que introdujese el usuario.

Para comenzar, empecé organizando los datos que me interesaba que estuviesen en la base de datos. Llegué a la conclusión de las propiedades que consideraba ineludibles eran:

- **Películas:** Género, Título, Director, Actores, Guionista, Valoración y País
- **Libros:** Título, Escritor y País.

Una vez establecidos estos puntos, me di cuenta que era muy lógico agrupar las películas por géneros, que me parecía que era la mejor manera de juntarlas, por tanto, creé nodos por cada género que iba a recoger.



```
1 CREATE (g:Genre {type="War"})
```

```
2
```

```
3
```

De esta manera fui creando todos los nodos que se ven en la primera imagen. Así ya tengo la primera forma de agrupar las películas que posteriormente introduce en la base de datos. Decidí que cada género fuese un nodo y no una propiedad de las películas, ya que era un criterio muy general, y varias películas podían tener géneros distintos.

Luego de esto, fui creando nodos de películas y añadiéndoles sus respectivas propiedades. Esta fue la parte más larga y tediosa del trabajo ya que se dividía en varias partes. La primera de ellas supuso buscar información de películas que estuviesen basadas en un libro. Para ello realicé numerosas búsquedas en la web, y rápidamente encontré una serie de páginas web que recogían esta información. Debido a que había empezado por establecer los distintos géneros que había, fui seleccionando las películas en función de esto último, para que hubiese una distribución más o menos equitativa. Una vez elegidas las 100, 110 películas que iban a conformar la base de datos, pasé a buscar la información representativa de cada una de ellas. Fui metiendo todo en un Excel, en el cual iba almacenando todos los aspectos que antes he citado.

Película	Director	Actor Principal	Valoración	Guionista	País
El Padrino	Francis Ford Coppola	Marlon Brando, Al Pacino	10, 9'8, 9	Mario Puzo, Francis Ford Coppola	EE.UU.
El silencio de los corderos	Jonathan Demme	Anthony Hopkins, Joaquin Phoenix	8'6, 9'6, 8'2	Ted Tally	EE.UU.
La Colmena	Mario Camus	Victoria Abril, Ana Belén	7'2, 7'1, 7	José Luis Dibildos	España
La Carretera	John Hillcoat	Viggo Mortensen, Christopher Meloni	7'2, 8'1, 6'6	Joe Penhall	EE.UU.
Nazarín	Luis Buñuel	Francisco Rabal, Marisol, Antonio Gómez	7'9, 7'8, 7'7	Luis Buñuel, Julio Alejandro	Méjico
El Gato Pardo	Luchino Visconti	Burt Lancaster, Alain Delon	8'0, 8, 7, 7'8	Suso Cecchi d'Amico	Italia
Carrie	Brian de Palma	Sissy Spacek, Piper Laurie	7'4, 7'9, 7	Lawrence D. Cohen	EE.UU.
La lengua de las mariposas	José Luis Cuerda	Fernando Fernán Gómez, Antonio de la Torre	7'6, 7'6, 7'5	José Luis Cuerda	España
Alguien voló sobre el nido del cuco	Milos Forman	Jack Nicholson, Louis Cukierman	8'7, 8'3, 8'4	Bo Goldman, Lawrence Gordon	EE.UU.
To Kill a Mockingbird	Robert Mulligan	Gregory Peck, Mary Badham	8'3, 7, 8'3	Horton Foote	EE.UU.
Parque Jurásico	Steven Spielberg	Sam Neill, Jeff Goldblum	8'1, 8, 7	Michael Crichton	EE.UU.
American Psycho	Mary Harron	Christian Bale, Will Ferrell	7'6, 7'4, 6'6	Mary Harron, Guillermina Valdés	EE.UU.
El Halcón Maltés	John Huston	Humphrey Bogart, Edward G. Robinson	8, 8'8, 8	John Huston	EE.UU.
Cadena Perpetua	Frank Darabont	Tim Robbins, Morgan Freeman	9'3, 8'6, 8'6	Frank Darabont	EE.UU.
Los hombres que no amaban a las mujeres	David Fincher	Daniel Craig, Rooney Mara	7'8, 7'8, 7	Steven Zaillian	EE.UU.
Doctor Zhivago	David Lean	Omar Sharif, Alec Guinness	8, 7'4, 7'9	Robert Bolt	EE.UU.
El diario de Bridget Jones	Sharon Maguire	Renée Zellweger	6'7, 6'6, 6'3	Helen Fielding, Anna Maxwell Martin	UK
Valor de ley	Joel Coen, Ethan Coen	Hailee Steinfeld, Matt Damon	7'6, 6'4, 7'1	Joel Coen, Ethan Coen	EE.UU.
La princesa prometida	Rob Reiner	Peter Falk, Robin Wright	8'1, 6'6, 7'4	William Goldman	EE.UU.
Sin novedad en el frente	Lewis Milestone	Louis Wolheim, Lew Ayres	8, 7'4, 7'8	Maxwell Anderson	EE.UU.
Tenemos que hablar de Kevin	Lynne Ramsay	Tilda Swinton, John C. Reilly	7'5, 7'2, 7'1	Lynne Ramsay	UK
Brokeback Mountain	Ang Lee	Anne Hathaway, Jake Gyllenhaal	7'7, 8, 7	Dianna Ossana, Ang Lee	EE.UU.
Hijos de los hombres	Alfonso Cuarón	Clive Owen, Julianne Moore	7'9, 7, 7'1	Alfonso Cuarón	UK
Grandes esperanzas	Alfonso Cuarón	Ethan Hawke, Gwyneth Paltrow	6'9, 8, 6'4	Mitch Glazer	EE.UU.
Blade Runner	Ridley Scott	Harrison Ford, Rutger Hauer	8'1, 9, 8'1	Hampton Fancher	EE.UU.
El Señor de los Anillos	Peter Jackson	Viggo Mortensen, Elijah Wood	8'9, 8'8, 8'2	Philippa Boyens, Peter Jackson	Nueva Zelanda
Drácula de Bram Stoker	Francis Ford Coppola	Gary Oldman, Winona Ryder	7'4, 7, 7'6	James V. Hart	EE.UU.
La lista de Schindler	Steven Spielberg	Liam Neeson, Ralph Fiennes	8'9, 9, 8'7	Steven Zaillian	EE.UU.
El almuerzo desnudo	David Cronenberg	Peter Weller, Judd Nelson	7'1, 8, 6'6	David Cronenberg	Canada

Después de acabar de recopilar todos los datos necesarios era hora de ir metiéndolos en la base de datos de Neo4J de la siguiente manera:

```

1 MATCH (m:Movie)
2 WHERE m.title="El Padrino"
3 SET m.actores="", m.director="", m.guion="", m.pais="", m.critica=""
4

```

En la línea del “SET” actores representa a los actores principales, director al director, guion al guionista de la película, y el país al país del que pertenece la película. También tenemos la propiedad critica, en la cual aparecen tres notas distintas de IMBD, SensaCine y FilmAffinity, respectivamente. Las tres webs de cine más importantes y con más tráfico de usuarios. Poco a poco, durante varios días, fui completando las etiquetas de cada película. Este proceso duró unos tres o cuatro días.

Una vez finalizada esta tarea, cree los nodos pertenecientes a los libros, y los completé, de la misma manera que las películas, con las propiedades título, escritor y país. Para esto, volví a buscar información de todos los libros en los cuales se basaban las películas que había metido anteriormente y los reuní en un Excel antes de pasarlo a Cypher. Este proceso duró más o menos otros dos días de trabajo.

Libro	Escritor	País
El Padrino	Mario Puzo	Italia
El silencio de los c	Thomas Harris	EE.UU.
La Colmena	Camilo José Cela	España
La Carretera	Cormac McCarthy	EE.UU.
Nazarín	Benito Pérez Galdós	España
El gatopardo	Giuseppe Tomasi	Italia
Carrie	Stephen King	EE.UU.
¿Qué me quieres,	Manuel Rivas	España
Alguien voló sobr	Ken Kesey	EE.UU.
Matar a un ruiser	Harper Lee	EE.UU.
Parque Jurásico	Michael Crichton	EE.UU.
American Psycho	Bret Easton Ellis	EE.UU.
El halcón maltés	Dashiell Hammett	EE.UU.
Rita Hayworth y l	Stephen King	EE.UU.
Millenium: Los ho	Stieg Larsson	Suecia
Doctor Zhivago	Boris Pasternak	URSS
El diario de Bridg	Helen Fielding	UK
True Grit	Charles Portis	EE.UU.
La princesa prom	William Goldman	EE.UU.
Sin novedad en e	Erich Maria Remarque	Alemania
Tenemos que hac	Lionel Shriver	EE.UU.
Brokeback Moun	Annie Proulx	EE.UU.
The Children of N	P.D. James	UK
Grandes esperan	Charles Dickens	UK
¿Sueñan los andr	Philip K. Dick	EE.UU.
El Señor de los Ar	J. R. R. Tolkien	UK
Drácula	Bram Stoker	UK
El arca de Schind	Thomas Keneally	Australia
El almuerzo desn	William S. Burroughs	Francia

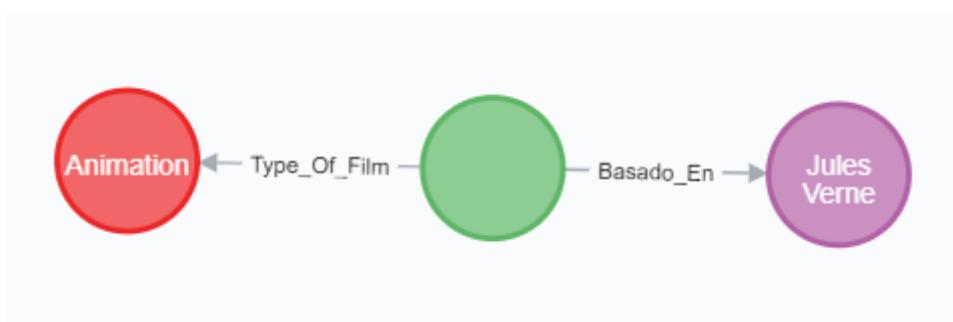
Con todos los nodos ya listos, creados y con sus propiedades, era hora de juntarlos y crear las relaciones entre ellos. Evidente y lógicamente los géneros deben estar relacionados con las películas:

```
$ CREATE (m:Movie)-[b:Type_Of_Film]→(g:Genre )
```

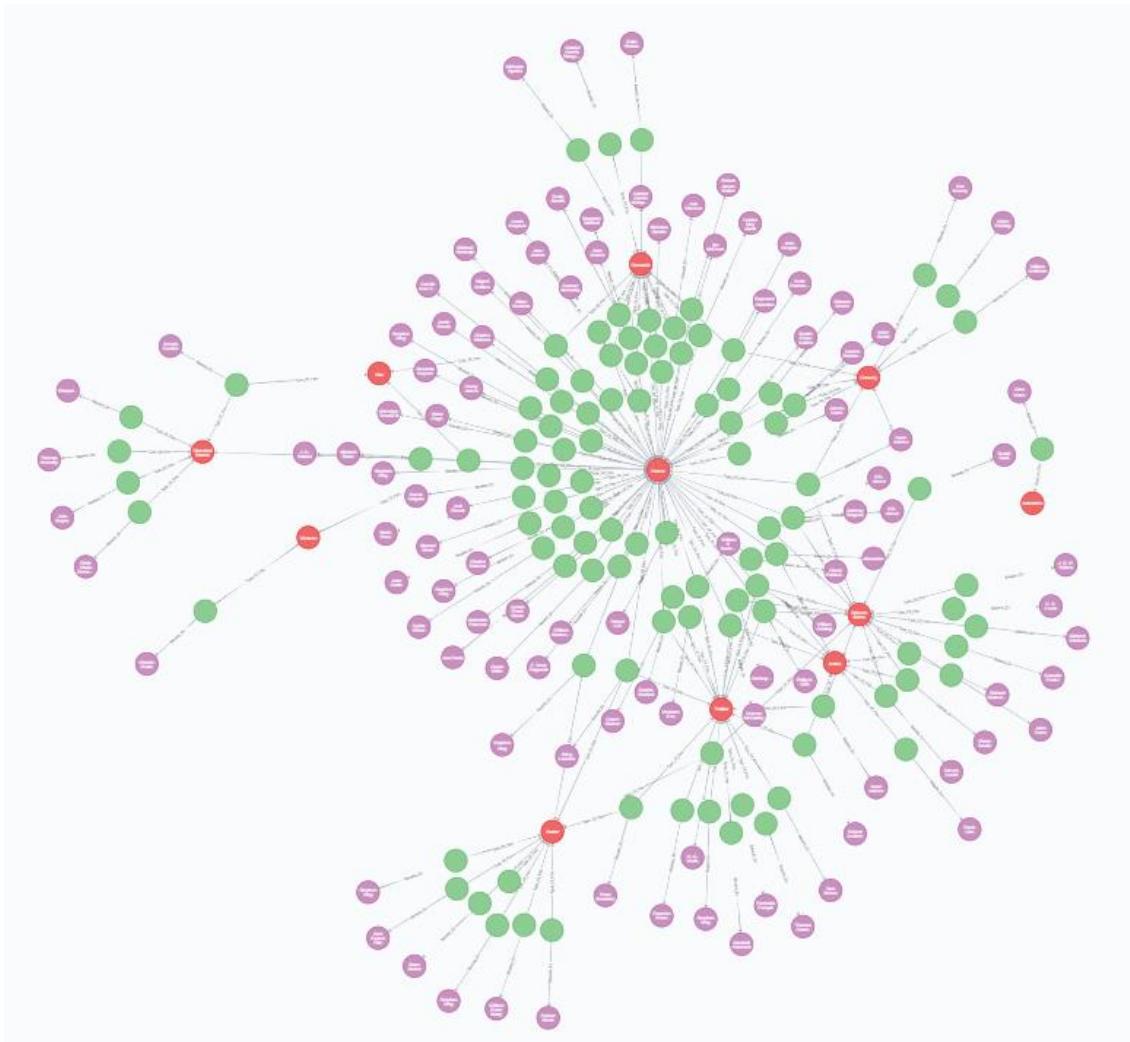
y luego con los libros y las películas:

```
$ CREATE (m:Movie)-[:Basado_En]→(l:Libro)
```

De esta manera y tras varios días realizando esta tarea, finalmente tenía completado el grafo de conocimiento que formaba mi base de datos y con la cual ya podía ponerme a realizar el algoritmo de recomendación.



El nodo rojo corresponde a los géneros, los verdes a las películas y los morados a los libros. Aquí podemos ver el grafo entero después de todo este trabajo.



Finalizado el trabajo con Cypher, tocaba comenzar a montar el algoritmo mediante el cual se pudiese realizar el recomendador. Por recomendación del profesor, decidí usar Python y como era un lenguaje que no conocía y con el cual no había trabajado nunca, tuve que estudiar las diferentes posibilidades que este lenguaje me podía proporcionar. Como ya tenía claro que quería investigué directamente a tiro fijo. Mi intención era crear un entorno gráfico donde se pudiese usar el algoritmo que fuese a implementar, a modo de aplicación, que el usuario eligiese los criterios de búsqueda y que la aplicación buscase en la base de datos, hiciese una consulta y devolviera una película acorde a los criterios pedidos. Más o menos ya tenía una idea fija de lo que me apetecía hacer y de cómo quería que fuese la aplicación, pero necesitaba conocer qué posibilidades tenía Python. Después de buscar un tiempo, vi que podía realizar un entorno gráfico mediante una librería de Python llamada TKinter.

Lo primero del código debía ser conectar la base de datos de Cypher de Neo4J.

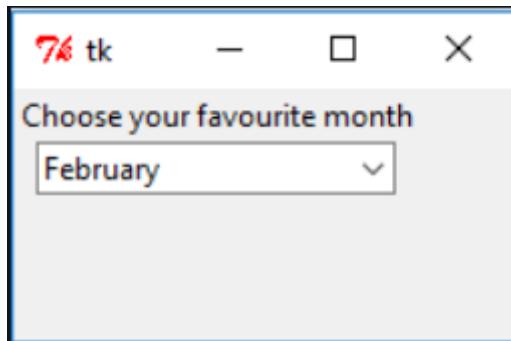
```
#Consultar a la base de datos
from neo4j import GraphDatabase
uri = "bolt://localhost:7687"
driver = GraphDatabase.driver(uri, auth=("neo4j", "dani"))
```

Tras comprobar que el programa estaba bien conectado con la base de datos, empezaba la fase de comenzar el algoritmo.

```
raiz = Tk()

raiz.title("Películas")
raiz.resizable(width=False, height=False)
raiz.geometry("850x800")
raiz.config(bg="grey")
```

De esta manera creaba la ventana principal de la aplicación y los aspectos principales de la misma. Lo siguiente fue crear los ComboBox. Un ComboBox es una caja en la cual se introducen distintas opciones seleccionables que el desarrollador proporciona. Adjunto una imagen de un ComboBox.



Como iban a haber 5 factores a tener en cuenta a la hora de elegir película, he creado 5 ComboBox distintos.

```
#####
# COMBOBOX #####
raiz.combo.place(x=115, y=25)
raiz.combo[ "values"]=[ "Género", "Director", "Actor", "País", "Guionista"]

raiz.combo5.place(x=115, y=65)
raiz.combo5[ "values"]=[ "Género", "Director", "Actor", "País", "Guionista"]

raiz.combo7.place(x=115, y=105)
raiz.combo7[ "values"]=[ "Género", "Director", "Actor", "País", "Guionista"]

raiz.combo9.place(x=115, y=145)
raiz.combo9[ "values"]=[ "Género", "Director", "Actor", "País", "Guionista"]

raiz.combo11.place(x=115, y=185)
raiz.combo11[ "values"]=[ "Género", "Director", "Actor", "País", "Guionista"]
```

En cada uno de ellos tendremos la posibilidad de seleccionar uno de los elementos que nos ofrece la lista entre Género, Director, Actor, País y Guionista. Además, por cada ComboBox, tendremos un botón "Aceptar", es decir, otros 5 botones.

```

#####
##### BOTONES #####
#####

raiz.Button = ttk.Button(raiz, command=aceptar, text = "Aceptar")
raiz.Button.place(x=295, y=25)

raiz.Button2 = ttk.Button(raiz, command=aceptar2, text = "Aceptar")
raiz.Button2.place(x=295, y=65)

raiz.Button3 = ttk.Button(raiz, command=aceptar3, text = "Aceptar")
raiz.Button3.place(x=295, y=105)

raiz.Button4 = ttk.Button(raiz, command=aceptar4, text = "Aceptar")
raiz.Button4.place(x=295, y=145)

raiz.Button5 = ttk.Button(raiz, command=aceptar5, text = "Aceptar")
raiz.Button5.place(x=295, y=185)

```

Como podemos observar, en la propiedad `command` del `Button` de TKinter, cada botón aceptar nos lleva a una función distinta del código. Esto se debe a que por cada `ComboBox` llamamos a un método distinto ya que, si no, se producía un error. Sin embargo, es curioso el hecho de que las funciones son casi exactamente iguales así que sólo pondré un ejemplo y lo explicaré. Cogeré para exemplificar el funcionamiento de las 5 funciones sólo la primera: aceptar. Lo primero y más importante es declarar los vectores en los cuales almacenaremos la información elegida por el usuario.

```

51
52 vecGen = []
53 vecDir = []
54 vecAct = []
55 vecPais = []
56 vecGuion = []

```

En cada uno de ellos almacenaremos los datos referentes al nombre que llevan, y los utilizaremos en todas las funciones del tipo “aceptar”.

Lo primero que observamos es que valoramos la opción de que no se haya seleccionado nada en el `ComboBox`. Más adelante veremos por qué queremos controlar esta situación. Luego, dependiendo de cuál sea el criterio establecido en el `ComboBox`, vamos valorando opciones. Por ejemplo, si el usuario introduce director, introducimos en el vector de directores una llamada a la función director. Aquí sucede una de los momentos más importantes ya que en esta segunda función es donde realizamos la consulta a la base de datos, de la siguiente manera:

```

50 def director():
51     vector2 = []
52     vector = []
53     with driver.session() as session:
54         i = 0
55         for record in session.run("MATCH (m:Movie)"
56                                     "RETURN m.director"):
57
58             vector.append(record[ "m.director"])
59             vector2.append(str(vector[i]).split(", "))
60             i=i+1
61
62     return vector2

```

De este modo se realiza la consulta, pidiendo los datos de todos los directores que tenemos en la base de datos y los almacena en “vector”, previamente creado al inicio de la función. Como en algunos casos, hay películas que tiene varios directores, partimos los directores donde encontramos una coma, que es como se había introducido en la base de datos y acumula los directores, esta vez ya sí, separados del todo, en “vector2”.

Una vez los directores devueltos al vector general “vecDir”, creamos el ComboBox, de la misma manera que se ha hecho el de antes, entonces aparece al apretar el botón de “aceptar” que hemos explicado anteriormente. Dentro del ComboBox, introducimos todo el contenido del vector “directores” que es donde introducimos todos los directores previamente controlando que no estén repetidos. Para ello vamos recorriendo el vector principal y si ya está introducido el mismo dato, se obvia y se pasa al siguiente. Este procedimiento se repite para los géneros, los actores, los guionistas y el país, con una función de consulta para cada una de las posibles elecciones del usuario. Asimismo, esta función aceptar es exactamente igual copiada para cada botón “aceptar”. Esto tiene una fácil explicación: cada vez que apretamos al botón “aceptar”, quería que apareciese el segundo ComboBox a la derecha del primero, y esta fue la única manera que encontré para realizarlo.

```

12 def aceptar():
13
14     if raiz.combo.get()!="":
15         if raiz.combo.get()=="Género":
16             vecGen = generos()
17             raiz.combo2.place(x=410, y=25)
18             raiz.combo2["values"] = vecGen
19
20         elif raiz.combo.get() == "Director":
21             vecDir = director()
22             raiz.combo2.place(x=410, y=25)
23             directores = []
24
25             for i in range(len(vecDir)):
26                 for j in range(len(vecDir[i])):
27                     repe = 0
28                     for z in range(len(directores)):
29                         if(directores[z]==vecDir[i][j]):
30                             repe=1
31                     if repe==0:
32                         directores.append(vecDir[i][j])
33
34
35             raiz.combo2["values"] = directores
36
37
38         elif raiz.combo.get() == "Actor":
39             vecAct = actor()
40             raiz.combo2.place(x=410, y=25)
41             actores = []
42
43             for i in range(len(vecAct)):
44                 for j in range(len(vecAct[i])):
45                     repe = 0
46                     for z in range(len(actores)):
47                         if(actores[z]==vecAct[i][j]):
48                             repe=1
49                     if repe==0:
50                         actores.append(vecAct[i][j])
51

```

Finalmente, por último, para terminar de explicar esta parte del desarrollo del software, cabe señalar, volviendo al primer párrafo de la explicación, que se controla en cada ComboBox de la izquierda, que se ha introducido algún dato, antes de activar el botón “aceptar”. Si esto no ocurre y el usuario, por equivocación no selecciona ningún parámetro de búsqueda, emitirá el sistema un mensaje de error.

```
else:  
    messagebox.showerror("Error", "No se ha seleccionado ninguna opción.")
```

El siguiente paso en el desarrollo del sistema fue pasar a seleccionar una película para mostrar, es decir, implementar el recomendador en sí. Me surgieron varias dudas con cómo llevarlo a cabo. Finalmente decidí establecer un sistema de coeficientes de manera que, al primer parámetro introducido se asigna el coeficiente 1, al segundo 0.8, el tercero 0.6, 0.4, y 0.2, respectivamente. Lo primero de todo fue implementar el botón “buscar”.

```
1289 raiz.Button7 = ttk.Button(raiz, command=buscar, text = "BUSCAR")  
1290 raiz.Button7.place(x=245, y=225)
```

Como podemos observar, al igual que hemos explicado antes, en la propiedad command del Button, llamamos a la función buscar, que procedemos a enseñar ahora.

Una vez llamada la función comienza el desarrollo del método. Como podemos apreciar en la imagen adjunta, debajo de este párrafo, dependiendo del atributo que el usuario selecciona, vamos aplicando a todas las películas con esa característica, el coeficiente marcado antes. Por ejemplo, en caso de elegir el género War (guerra) como característica primaria, lo primero consiste en guardar War en una variable, para posteriormente ser usada para requerirla en la consulta. Una vez que se realiza la consulta a la base de datos, se buscan todas las películas con ese género, y se les asigna el coeficiente 1, a todas ellas. Este procedimiento se repetiría para todas las características introducidas yendo poco a poco sumando coeficientes. Las películas irían almacenando y sumando coeficientes de manera que cada película terminaría obteniendo una suma concreta de puntos. Por un lado, se guardarían las películas en un orden, en un vector, y, por otro lado, se archivarían los coeficientes de cada una de ellas en el mismo orden. De esta manera, puedo saber en todo momento cual es la puntuación que tiene una película solo con conocer en qué posición se encuentra dentro del vector. De este mismo modo, lo realizamos con los géneros, directores, actores, guionistas y el país, y también para todos los ComboBox situados en la izquierda, que son quienes dan la posibilidad de introducir parámetros.

```

#####
##### COMBO2 #####
#####

if raiz.combo5.get() == "Género":
    var=raiz.combo6.get()
    with driver.session() as session:
        for record in session.run("MATCH (m:Movie)-[Type_Of_Film]-(g:Genre{type:'"+str(var)+"'})"
                                  "RETURN m.title"):
            for i in range (len(vector)):
                if vector[i]==record["m.title"]:
                    coeficientes[i]=coeficientes[i]+0.8
                    true=1
                else:
                    true=0
            if true==0:
                vector.append(record["m.title"])
                coeficientes.append(0.8)
    true=0

if raiz.combo5.get() == "Director":
    var=raiz.combo6.get()
    with driver.session() as session:
        for record in session.run("MATCH (m:Movie)"
                                  "WHERE m.director='"+str(var)+"'"
                                  "RETURN m.title"):
            for i in range (len(vector)):
                if vector[i]==record["m.title"]:
                    coeficientes[i]=coeficientes[i]+0.8
                    true=1
                else:
                    true=0
            if true==0:
                vector.append(record["m.title"])
                coeficientes.append(0.8)
    true=0

if raiz.combo5.get() == "Actor":
    var=raiz.combo6.get()
    with driver.session() as session:
        for record in session.run("MATCH (m:Movie)"
                                  "WHERE m.actores CONTAINS '"+str(var)+"'"
                                  "RETURN m.title"):
```

Una vez finalizado este proceso, ya tendríamos el recomendador prácticamente finalizado. Tan solo quedaría buscar dentro de nuestros coeficientes, la suma mayor que hayamos obtenido, encontrar la posición en la cual se halla la suma más grande, y buscar esa misma posición en la lista de películas para obtener la película que el sistema recomienda.

```

#####
##### FINAL #####
#####

for i in range(len(vector)):
    if(orden<coeficientes[i]):
        orden=coeficientes[i]
        pelicula[0]=vector[i]

final.append(pelicula[0])
```

Como vemos, nuestra película final la guardamos en el vector “final”. Una vez se ha hecho el proceso de recomendación, pasamos a la fase en la cual terminamos de confeccionar la aplicación, escribimos los mensajes que van a aparecer en la interfaz, y establecemos los últimos detalles. Una idea que se me ocurrió fue mostrar la valoración de la película, atributo que había introducido en Cypher pero que hasta ahora no había utilizado. En un principio, la idea era simple, obtener mediante una consulta a Neo4J los tres números de la crítica de las tres páginas web que seleccioné en el proceso de introducción de datos. Sin embargo, rápidamente surgieron las primeras complicaciones. Python no admitía el formato en el cual había introducido los datos anteriormente. Las notas de las películas estaban metidas de la manera “X’Y” y el apóstrofe daba muchísimos problemas. Aquí surgía el primer contratiempo real, aparte de los obstáculos potenciales que representa el desarrollo de software en sí mismo. Para solucionar esto, tenía dos opciones: o bien cambiar toda la base de datos y modificar las valoraciones de cada película, cosa que llevaría un trabajo enorme, o bien buscar una manera de reemplazar y cambiar el apóstrofe por un punto, para poder operar con cada nota. Finalmente decidí optar por la segunda opción y por ello esto fue lo que he debido hacer.

En esta primera captura de pantalla observamos cómo hago la petición a la base de datos, introduciendo los valores dados en el vector “valoraciones” primero, y en “valoraciones2” luego, pero con las notas divididas ya de las tres primeras iniciales con las que partimos. Como se ve al final, llamamos a un método llamado cambio que transforma de forma recursiva el apóstrofe en un punto.

```
valoraciones = []
valoraciones2 = []

with driver.session() as session:
    i = 0
    for record in session.run("MATCH (m:Movie)"
                               "WHERE m.title = '"+str(final[0])+"'"
                               "RETURN m.critica"):

        valoraciones.append(record["m.critica"])
        valoraciones2.append(str(valoraciones[i]).split(", "))
        i=i+1

valoraciones3 = []
valoraciones3.append(cambio(valoraciones2, 0, valoraciones3))
```

```
def cambio(valoraciones2, z, valoraciones3):
    auxiliar=[]
    if(z<3):
        auxiliar=valoraciones2[0][z].replace("'", ".")
        valoraciones3.append(auxiliar)
        z=z+1
        cambio(valoraciones2, z, valoraciones3)
```

Si bien es cierto que la mayor parte del trabajo estaba hecha, todavía no había llegado a mi propósito inicial, que era lograr sacar la nota media de las tres valoraciones, aunque esto ya fue la parte más fácil del proceso relativo a las notas y valoraciones.

```
critica = 0.0

for i in range(3):
    critica=float(valoraciones3[i])
critica=round(critica/3, 2)
```

Llegados a este punto, tan sólo restaban los últimos detalles de mi interfaz: generar los mensajes de salida y controlar que se haya introducido algún criterio cuando se apriete el botón “buscar”. Para hacer un control de esto último realicé algo extremadamente parecido a un procedimiento anterior:

```
if(final[0]== ""):
    messagebox.showerror("Error", "Introduce algún criterio")
    nueva()
```

Vemos algo nuevo en la última línea y es la llamada a la función “nueva” que explicaremos más tarde. Por último, añadimos los mensajes que saldrán por pantalla mediante Labels.

```
raiz.label=ttk.Label(raiz)
raiz.label.place(x=65, y=285 )
raiz.label.config(text="Según los criterios introducidos, la película que recomendamos es: " + final[0])
raiz.label.config(font="Verdana 9 italic bold")

raiz.label2=ttk.Label(raiz)
raiz.label2.place(x=65, y=515)
raiz.label2.config(text="La valoración media de esta película es: " + str(critica))
raiz.label2.config(font="Verdana 9 italic bold")

raiz.combo.config(state="disabled")
raiz.combo5.config(state="disabled")
raiz.combo7.config(state="disabled")
raiz.combo9.config(state="disabled")
raiz.combo11.config(state="disabled")
```

Aquí vemos los dos mensajes que mostramos, así como su configuración. En las últimas líneas cambiamos los estados de los ComboBox de la izquierda a disabled. De esta manera, ya no pueden ser usados y evitamos que se realice una nueva búsqueda por encima de la otra.

Luego de finalizar esto decidí que iba a ser interesante poder ver el cartel de la película seleccionada como “ganadora” después de la recomendación. Aquí empezó un proceso de trabajo un poco tedioso y largo ya que debía crear una carpeta e introducir en ella imágenes de todas las películas que estaban metidas en la base de datos, además de cambiar el nombre a cada una de ellas y su tamaño, aunque esto último por equivocación ya que posteriormente supe que podía cambiar y reajustar el tamaño de la imagen al que yo quisiera desde el código. Después de buscar cómo introducir una foto y de varios intentos fallidos, di con la clave de cómo hacerlo:

```
#####
##### IMAGEN PELICULA #####
path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Peliculas\\\\'+ str(final[0]) + ".jpg"
load = Image.open(path)
render = ImageTk.PhotoImage(load)
img = ttk.Label(raiz, image=render)
img.image = render
img.place(x=295 , y=315)
```

En el path especificamos la ruta en la cual se encuentra la carpeta que contiene las imágenes de los carteles, y añadimos jpg, ya que todas ellas tienen ese formato. De esta manera, especificamos el lugar donde queremos que salga con img. place y la foto se mostrará.

Después de acabar con la película en la ventana principal, tocaba ponerse con el libro. Este proceso fue muy sencillo. Una vez teniendo la película sólo tenemos que realizar una consulta a la base de datos y obtener el libro asociado a la película recomendada, así como su escritor. Una vez que tenemos todo esto, sacamos mediante un Label, como antes, la información del libro en el cual se basa la película recomendada.

```
escritor = []
with driver.session() as session:
    for record in session.run("MATCH (m:Movie)-[Basado_En]-(l:Libro)"
                               "WHERE m.title='"+str(final[0])+"'"
                               "RETURN l.titulo"):
        libro.append(record["l.titulo"])

with driver.session() as session:
    for record in session.run("MATCH (m:Movie)-[Basado_En]-(l:Libro)"
                               "WHERE m.title='"+str(final[0])+"'"
                               "RETURN l.autor"):
        escritor.append(record["l.autor"])

raiz.label3=ttk.Label(raiz)
raiz.label3.place(x=65, y=550)
raiz.label3.config(text="La película está basada en el Libro " + libro[0] + " de " + escritor[0])
raiz.label3.config(font="Verdiana 9 italic bold")
```

Finalmente, al igual que hicimos con la película, mostramos la imagen de la portada del libro que previamente hemos introducido todas ellas en otra carpeta distinta:

```
#####
##### IMAGEN LIBRO #####
path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Libros\\\\'+ str(libro[0]) + ".jpg"
load2 = Image.open(path)
render = ImageTk.PhotoImage(load2)
img = ttk.Label(raiz, image=render)
img.image = render
img.place(x=295 , y=585)
```

Quedaban ya solo pues, los últimos retoques de la aplicación. Cómo no se podía realizar una búsqueda sobre otra ya que me daba numerosos fallos y no conseguía solucionarlos, decidí implementar un botón, llamado “nueva búsqueda”, que reiniciaba la aplicación sin necesidad que el usuario hiciera nada más que apretar en él.

```
raiz.Button8 = ttk.Button(raiz, command=nueva, text = "NUEVA BÚSQUEDA")
raiz.Button8.place(x=345, y=225)
```

Al igual que en el resto de botones, el funcionamiento en este tampoco cambia, en la propiedad command llamamos a la función “nueva”. Anteriormente esta función ya ha sido llamada, en caso de pulsar sobre “buscar” y no haber introducido ningún criterio. Dentro de este documento, más arriba, viene explicado. El único propósito de esta función es reiniciar el programa.

```
def nueva():
    python=sys.executable
    os.execl(python, python, * sys.argv)
```

Una vez finalizado esto, la aplicación ya está en total y correcto funcionamiento y podría presentarse ya. Sin embargo, bajo mi criterio la parte gráfica de la misma estaba bastante pobre. Estéticamente resultaba realmente poco atractiva así que pensé en varias maneras para solventar esto. Tras sopesar varias posibilidades, me decanté finalmente por colocar un fondo de mi aplicación con imágenes icónicas del cine. Además de esto, guardé varias imágenes e ideé una manera de que saliesen distintas imágenes cada vez que se ejecutase la aplicación. El sistema es sencillo: las fotos guardadas en una carpeta están numeradas del 1 en adelante. Después de esto, creé una variable que guardase un número aleatorio entre 1 hasta el número de fotos guardadas. De esta manera, cada vez que la aplicación se lanza, la imagen de fondo se seleccionará aleatoriamente.

```
aleatorio = randint(1, 12)

imagen = Image.open("C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Fondos\\\\"+str(aleatorio)+".jpg")
imagen_de_fondo = ImageTk.PhotoImage(imagen)
fondo = tk.Label(raiz, image=imagen_de_fondo)
fondo.place(x=0, y=0, relwidth=1, relheight=1)
```

Unos días después de finalizar todo esto, se me ocurrió la idea de introducir la posibilidad al usuario de consultar la información total de la película recomendada, y su libro asociado. El modo es absolutamente idéntico para el libro como para la película. Lo primero fue crear un botón llamado “información”, y rápidamente pensé en que se abriese una ventana nueva donde apareciese toda la información requerida.

```
raiz.Button9 = ttk.Button(raiz, command=infopeli, text = "INFORMACIÓN")
raiz.Button9.place(x=455, y=335)
```

Como vemos, llamamos a la función infopeli. Ahí, lo primero que se hace es crear la nueva ventana e introducirle sus parámetros básicos:

```
info = tk.Toplevel(raiz)

info.title("Libro")
info.resizable(width=True, height=True)
info.geometry("950x600")
info.config(bg="grey")
```

Aplicamos a la nueva ventana la misma imagen de fondo que en la ventana principal:

```
imagenFondo = Image.open("C:\\Users\\dalon\\Documents\\Daniel\\Clase\\SistemasBI\\Fotos\\Fondos\\"+str(aleatorio))
imagenFondo = imagenFondo.resize((950, 600), Image.ANTIALIAS)
imagen_de_fondo = ImageTk.PhotoImage(imagenFondo)
fondo = tk.Label(info, image=imagen_de_fondo)
fondo.place(x=0, y=0, relwidth=1, relheight=1)
```

Hacemos una consulta a la base de datos, donde le pedimos toda la información que está almacenada acerca de la película recomendada:

```
with driver.session() as session:
    for record in session.run("MATCH (m:Movie)"
                               "WHERE m.title='"+str(final[0])+"'"
                               "RETURN m.title, m.actores, m.critica, m.director, m.guion, m.pais"):
        titulo.append(record["m.title"])
        prota.append(record["m.actores"])
        crit.append(record["m.critica"])
        direc.append(record["m.director"])
        guion.append(record["m.guion"])
        pais.append(record["m.pais"])

with driver.session() as session:
    for g in session.run("MATCH (m:Movie)-[Type_of_Film]-(g:Genre)"
                           "WHERE m.title='"+str(final[0])+"'"
                           "RETURN g.type"):
        gen.append(g["g.type"])
```

Y mediante Labels, vamos escribiendo la información extraída en la nueva ventana:

```
info.label1=ttk.Label(info)
info.label1.place(x=425, y=50 )
info.label1.config(text=final[0])
info.label1.config(font="Arial 25 bold")

info.label2=ttk.Label(info)
info.label2.place(x=425, y=115 )
info.label2.config(text="Director: "+direc[0])
info.label2.config(font="Arial 10 bold")

info.label3=ttk.Label(info)
info.label3.place(x=425, y=155 )
info.label3.config(text="País: "+pais[0])
info.label3.config(font="Arial 10 bold")

info.label4=ttk.Label(info)
info.label4.place(x=425, y=195 )
info.label4.config(text="Actores principales: "+prota[0])
info.label4.config(font="Arial 10 bold")

info.label5=ttk.Label(info)
info.label5.place(x=425, y=235 )
info.label5.config(text="Valoraciones: "+crit[0])
info.label5.config(font="Arial 10 bold")

info.label6=ttk.Label(info)
info.label6.place(x=425, y=275 )
info.label6.config(text="Valoracion media: "+str(critica)+" ")
info.label6.config(font="Arial 10 bold")

info.label7=ttk.Label(info)
info.label7.place(x=425, y=355 )
info.label7.config(text="Genero: "+gen[0])
info.label7.config(font="Arial 10 bold")
```

Aquí, se produce el primer detalle interesante de esta función. En la ventana principal, anteriormente, hemos explicado cómo hemos obtenido la nota media de las tres valoraciones que buscamos en la web. Aquí utilizamos esa nota media para, en función de cuál sea, mostrar, mediante imágenes de estrellitas de valoración almacenadas en una carpeta previamente, dicha nota obtenida.

```
if(critica<=2):
    path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Estrellas\\\\1.jpg'
    load = Image.open(path)
    load = load.resize((84, 16), Image.ANTIALIAS)
    render = ImageTk.PhotoImage(load)
    img = ttk.Label(info, image=render)
    img.image = render
    img.place(x=560 , y=275)

elif(critica>2 and critica<=4):
    path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Estrellas\\\\2.jpg'
    load = Image.open(path)
    load = load.resize((84, 16), Image.ANTIALIAS)
    render = ImageTk.PhotoImage(load)
    img = ttk.Label(info, image=render)
    img.image = render
    img.place(x=560 , y=275)

elif(critica>4 and critica<=6):
    path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Estrellas\\\\3.jpg'
    load = Image.open(path)
    load = load.resize((84, 16), Image.ANTIALIAS)
    render = ImageTk.PhotoImage(load)
    img = ttk.Label(info, image=render)
    img.image = render
    img.place(x=560 , y=275)

elif(critica>6 and critica<=8):
    path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Estrellas\\\\4.jpg'
    load = Image.open(path)
    load = load.resize((84, 16), Image.ANTIALIAS)
    render = ImageTk.PhotoImage(load)
    img = ttk.Label(info, image=render)
    img.image = render
    img.place(x=560 , y=275)

else:
    path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Estrellas\\\\5.jpg'
    load = Image.open(path)
    load = load.resize((84, 16), Image.ANTIALIAS)
    render = ImageTk.PhotoImage(load)
```

Una vez hecho esto, mostramos nuevamente el cartel de la película, esta vez en más grande, y añadimos una bandera del país de procedencia en la esquina superior izquierda del cartel.

```
path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Peliculas\\\\'+ str(final[0]) + ".jpg"
load = Image.open(path)
load = load.resize((350, 495), Image.ANTIALIAS)
render = ImageTk.PhotoImage(load)
img = ttk.Label(info, image=render)
img.image = render
img.place(x=50 , y=50)

path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Banderas\\\\'+ str(pais[0]) + ".jpg"
load = Image.open(path)
load = load.resize((85, 59), Image.ANTIALIAS)
render = ImageTk.PhotoImage(load)
img = ttk.Label(info, image=render)
img.image = render
img.place(x=50 , y=50)
```

Con el libro hacemos exactamente igual. Creamos un botón “información” para el libro:

```
raiz.Button9 = ttk.Button(raiz, command=infolibro, text = "INFORMACIÓN")
raiz.Button9.place(x=455, y=615)
```

Dentro, vemos la llamada a infolibro, una función, que, lo primero que realiza es crear una nueva ventana:

```
info = tk.Toplevel(raiz)

info.title("Libro")
info.resizable(width=True, height=True)
info.geometry("950x600")
info.config(bg="grey")
```

Establecemos la imagen de fondo, la misma que la anterior:

```
imagenFondo = Image.open("C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Fondos\\\\"+str(al
imagenFondo = imagenFondo.resize((950, 600), Image.ANTIALIAS)
imagen_de_fondo = ImageTk.PhotoImage(imagenFondo)
fondo = tk.Label(info, image=imagen_de_fondo)
fondo.place(x=0, y=0, relwidth=1, relheight=1)
```

Realizamos la consulta para obtener el escritor y el país y los mostramos en la aplicación mediante Labels:

```

with driver.session() as session:
    for record in session.run("MATCH (L:Libro)"
                               "WHERE L.titulo='"+str(final[0])+"'"
                               "RETURN L.titulo, L.autor, L.pais"):
        titulo.append(record["L.titulo"])
        autor.append(record["L.autor"])
        pais.append(record["L.pais"])

info.label=ttk.Label(info)
info.label.place(x=425, y=50 )
info.label.config(text=libro[0])
info.label.config(font="Arial 25 bold")

info.label=ttk.Label(info)
info.label.place(x=425, y=115 )
info.label.config(text="Escritor: "+autor[0])
info.label.config(font="Arial 10 bold")

info.label=ttk.Label(info)
info.label.place(x=425, y=155 )
info.label.config(text="País: "+pais[0])
info.label.config(font="Arial 10 bold")

```

Para finalizar, colocamos la imagen de la portada del libro, aunque en más grande y colocamos la bandera del país de origen del libro en la esquina superior izquierda.

```

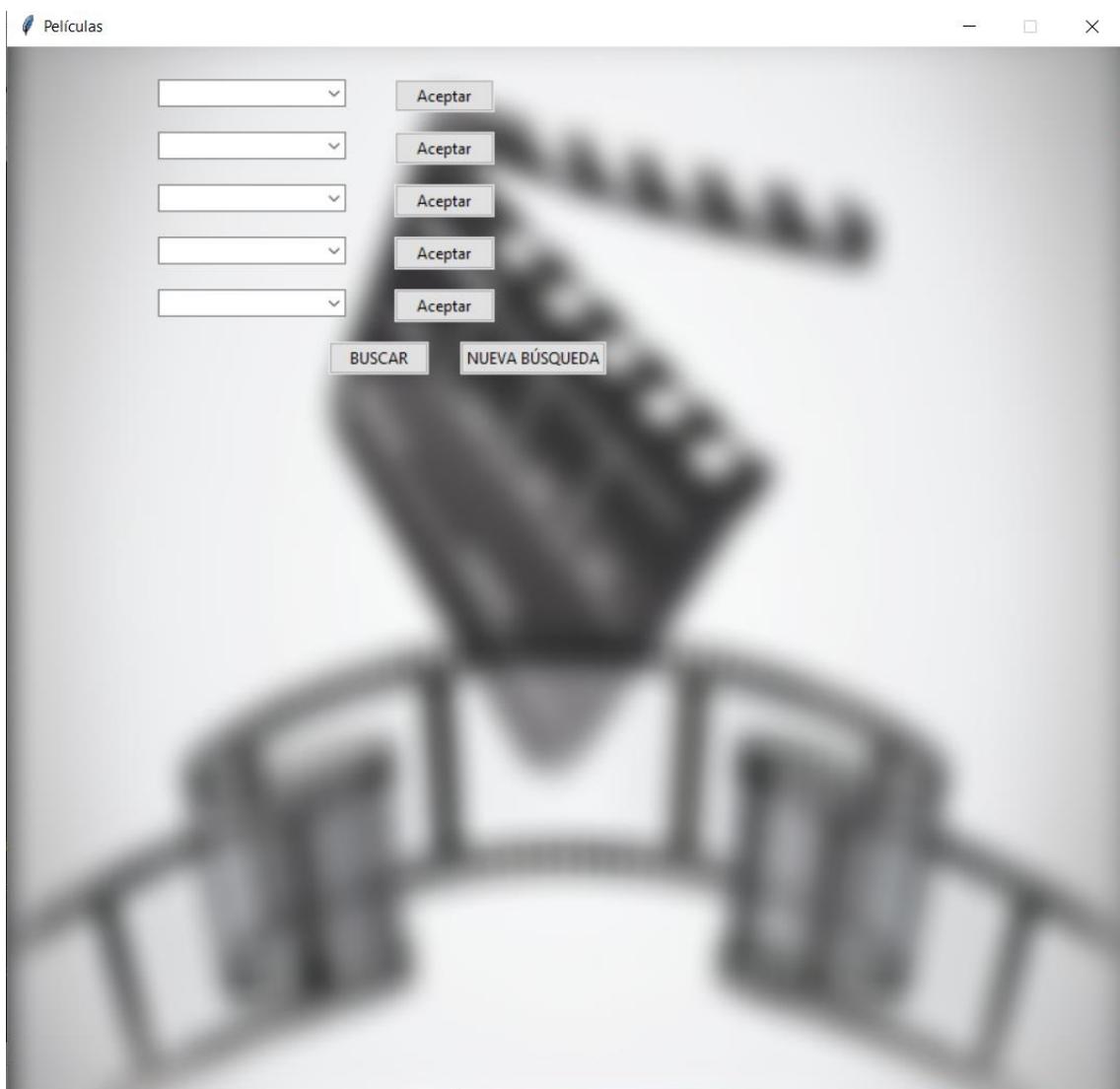
path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Libros\\\\' + str(final[0]) + ".jpg"
load = Image.open(path)
load = load.resize((350, 495), Image.ANTIALIAS)
render = ImageTk.PhotoImage(load)
img = ttk.Label(info, image=render)
img.image = render
img.place(x=50 , y=50)

path = 'C:\\\\Users\\\\dalon\\\\Documents\\\\Daniel\\\\Clase\\\\SistemasBI\\\\Fotos\\\\Banderas\\\\' + str(pais[0]) + ".jpg"
load = Image.open(path)
load = load.resize((85, 59), Image.ANTIALIAS)
render = ImageTk.PhotoImage(load)
img = ttk.Label(info, image=render)
img.image = render
img.place(x=50 , y=50)

```

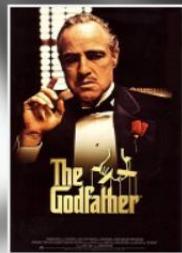
Una vez terminado esto, la aplicación estaría finalizada. Ha sido un proceso largo, de varios días, tedioso en algunos momentos, aunque estoy bastante satisfecho con el resultado final de la aplicación. A continuación, voy a enseñar cómo han quedado las tres ventanas. La principal, en la cual se encuentra el recomendador, la ventana de la información de la película, y la ventana de la información del libro.

VENTANA PRINCIPAL



Director	Aceptar	Francis Ford Coppola
	Aceptar	
	BUSCAR	NUEVA BÚSQUEDA

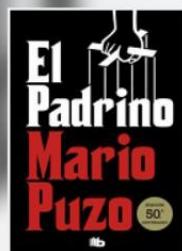
Según los criterios introducidos, la película que recomendamos es: *El Padrino*



INFORMACIÓN

La valoración media de esta película es: 9.6

La película está basada en el libro *El Padrino* de Mario Puzo



INFORMACIÓN

VENTANA INFORMACIÓN PELÍCULA



VENTANA INFORMACIÓN LIBRO

