



**universidad
de león**



Escuela de Ingenierías
Industrial, Informática y Aeroespacial
GRADO EN INGENIERÍA INFORMÁTICA

**SISTEMAS DE INFORMACIÓN DE GESTIÓN Y BUSINESS
INTELLIGENCE**

**SISTEMA DE RECOMENDACIÓN DE PLATAFORMAS DE
VIDEOS EN STREAMING**

Autor: Hugo Pérez Rodríguez

(Febrero, 2021)

Índice de contenidos

Índice de contenidos	2
Índice de imágenes	4
1. Descripción del problema.....	5
2. Herramientas	6
2.1 Neo4j.....	6
2.2 Anaconda (Spyder).....	6
2.3 Tkinter.....	7
2.4 JustWatch.....	7
2.5 FilmAffinity.....	8
3. Aplicación	9
3.1 Base de datos	9
3.1.1 Nodos.....	9
3.1.2 Relaciones.....	10
3.1.3 Ejemplo gráfico	10
3.1.4 Base de datos inicial	11
3.1.5 Base de datos final.....	11
3.2 Código Python.....	13
4. Algoritmo de recomendación	17
5. Análisis de resultados	19
5.1 Ejemplo 1	19
5.2 Ejemplo 2	22
6. DAFO	25
6.1 Fortalezas.....	25

6.2	Debilidades	26
6.3	Oportunidades	26
6.4	Amenazas	26
7.	Líneas de futuro.....	27
8.	Lecciones aprendidas	29
9.	Bibliografía.....	30

Índice de imágenes

Imagen 2-1 Logo Neo4j (Fuente: [1])	6
Imagen 2-2 Logo Anaconda (Fuente: [3])	7
Imagen 2-3 Logo JustWatch (Fuente: [5])	7
Imagen 2-4 Logo FilmAffinity (Fuente: [7])	8
Imagen 3-1 Comando Cypher Plataforma	9
Imagen 3-2 Comando Cypher Género	9
Imagen 3-3 Comando Cypher Película	10
Imagen 3-4 Comando Cypher ESTA_EN	10
Imagen 3-5 Comando Cypher ES_DE	10
Imagen 3-6 Ejemplo gráfico	10
Imagen 3-7 Base de datos inicial	11
Imagen 3-8 Base de datos final	12
Imagen 3-9 Pantalla principal	13
Imagen 3-10 Código Python 1	14
Imagen 3-11 Código Python 2	15
Imagen 3-12 Código Python 3	16
Imagen 4-1 Llamada a la base de datos	17
Imagen 4-2 Función de contar elementos	17
Imagen 4-3 Extraer máximo	17
Imagen 4-4 Mostrar nombre y logo de plataforma	18
Imagen 5-1 Ejemplo 1 primer paso	19
Imagen 5-2 Ejemplo 1 segundo paso	20
Imagen 5-3 Ejemplo 1 tercer paso	21
Imagen 5-4 Ejemplo 2 primer paso	22
Imagen 5-5 Ejemplo 2 segundo paso	23
Imagen 5-6 Ejemplo 2 tercer paso	24

1. Descripción del problema

Hoy en día el mercado de plataformas de video bajo demanda está saturado, con un número muy elevado de opciones donde elegir y cada opción con muchas películas y series de diferentes géneros. Con esta saturación es difícil saber cuál es la que puede interesar al usuario realmente, lo ideal sería tener todas, pero el coste que esto conlleva es algo que la gran mayoría de la población no podría asumir.

Con esta premisa, se ha creado esta aplicación que recomienda al usuario qué plataforma escoger basándose en los géneros de películas en los que esté interesado.

Así, gracias a esta herramienta, el usuario sabrá qué plataforma es la ideal para él y a la que debería estar suscrito, en vez de estar gastando tanto dinero teniendo contratadas todas ellas.

Además de la recomendación de plataforma de streaming, también se le recomienda películas que se encuentren en esta y que tengan esos géneros buscados, para no tener que andar perdiendo el tiempo buscando en la propia plataforma.

En conclusión, con esta aplicación se pretende ahorrar tiempo y facilitar en gran medida la elección de la plataforma de streaming en la que el usuario pueda estar interesado, así como ayudarlo a elegir qué películas ver en dicha plataforma y por supuesto se le ayuda a ahorrar dinero en suscripciones a otras páginas en las que en realidad no hay contenido interesante para él.

2. Herramientas

Para realizar esta aplicación se han utilizado diferentes herramientas tanto para la fase de desarrollo como para la fase de documentación. Algunas de ellas son programas que necesitan ser descargados y otras son simplemente páginas web en las que se ha buscado información, en cualquier caso, se aportará un link al sitio web en el que se podrá ver la herramienta utilizada.

2.1 Neo4j

Esta herramienta se ha utilizado durante la fase de desarrollo de la base de datos de la aplicación. Es una tecnología relativamente nueva que organiza los datos de las bases de datos en grafos de conocimiento en vez de en tablas como las bases de datos comunes. Además, cuenta con su propio lenguaje de programación, que se llama Cypher. En concreto para este proyecto, se ha utilizado la herramienta de escritorio que Neo4j ofrece a cualquier usuario, ya que es de código libre. [1]



Imagen 2-1 Logo Neo4j (Fuente: [1])

2.2 Anaconda (Spyder)

Esta herramienta fue utilizada durante la fase de desarrollo de la aplicación. Anaconda es un entorno de desarrollo con diferentes softwares incluidos para diferentes lenguajes de programación, en este caso se utilizó el software de Spyder que es el específico para Python. Spyder ayuda y agiliza el proceso de programación en Python con herramientas para ejecutar el programa de una manera muy sencilla, además de mostrar información útil que ayuda a poder subsanar errores de compilación. Además, es un programa

Hugo Pérez Rodríguez

sencillo y muy intuitivo, no requiere de mucho esfuerzo para poder llegar a trabajar con él. [2]



Imagen 2-2 Logo Anaconda (Fuente: [3])

2.3 Tkinter

Esta herramienta se utilizó para el diseño de la interfaz. Tkinter es una librería de Python especializada en la creación de interfaces. Proporciona la posibilidad de programar una interfaz fácil para las aplicaciones Python, además de contener numerosas herramientas proporcionando una amplia gama de usos. [4]

2.4 JustWatch

Esta herramienta se utilizó durante la etapa de documentación. JustWatch es una página web en la que aparecen todas las películas y series que tiene cada plataforma de streaming en España. Aunque en algunos casos hay errores o no está el contenido actualizado, ayudó mucho a la hora de seleccionar películas por plataforma para después ser añadidas a la base de datos. [5]



Imagen 2-3 Logo JustWatch (Fuente: [5])

2.5 FilmAffinity

Esta herramienta se utilizó en conjunto con JustWatch durante la fase de documentación. FilmAffinity es una página web donde aparece toda la información que se pueda necesitar sobre películas y series. Esta página web fue visitada para recabar información sobre las películas que iban a ser añadidas a la base de datos. Los datos principales que se extrajeron dos: el título de la película en español y el género o géneros a los que pertenecía dicha película. [6]



Imagen 2-4 Logo FilmAffinity (Fuente: [7])

3. Aplicación

Este proyecto puede dividirse en dos partes, la parte de la base de datos y la parte del código en Python. En la primera parte se explicará cómo se creó la base de datos, así como las relaciones que la forman. En la segunda se entrará a fondo con la programación y la interfaz de la aplicación.

El algoritmo de recomendación será explicado en otro apartado, así como algún ejemplo del funcionamiento de la aplicación que también se mostrarán en un apartado distinto.

3.1 Base de datos

La base de datos está creada desde cero con la información obtenida de las dos páginas mencionadas anteriormente (JustWatch y FilmAffinity).

3.1.1 Nodos

Esta está formada por tres tipos distintos de nodo con un atributo cada uno que aparecerá entre paréntesis:

1. Plataforma (nombre): en este tipo de nodos se han introducido el nombre de las plataformas que podrán ser recomendadas para el usuario. Este es el comando en Cypher para la creación de este nodo:

```
Create (l:Plataforma{nombre:"Netflix"})
```

Imagen 3-1 Comando Cypher Plataforma

2. Género (nombre): en este tipo de nodos se han introducido el nombre de los géneros cinematográficos que podrán ser seleccionados por el usuario para la recomendación de plataformas. Este es el comando en Cypher para la creación de este nodo:

```
Create (g:Genero{nombre:"Acción"})
```

Imagen 3-2 Comando Cypher Género

3. Película (nombre): en este tipo de nodos se han introducido el nombre de las películas que están en las plataformas y son de los géneros introducidos anteriormente. Este es el comando en Cypher para la creación de este nodo:

```
Create (e: Pelicula {nombre: "Gladiator"})
```

Imagen 3-3 Comando Cypher Película

3.1.2 Relaciones

En cuanto a las relaciones entre nodos, esta base de datos cuenta con dos tipos distintos de ellas:

1. ESTA_EN: esta relación une la película con la plataforma donde se encuentra dicha película. Este es el comando en Cypher para la creación de esta relación:

```
Create (e)-[:ESTA_EN]->(l)
```

Imagen 3-4 Comando Cypher ESTA_EN

2. ES_DE: esta relación une la película con el género cinematográfico al que pertenece dicha película. Este es el comando en Cypher para la creación de esta relación:

```
Create (e)-[:ES_DE]->(g)
```

Imagen 3-5 Comando Cypher ES_DE

3.1.3 Ejemplo gráfico

Por lo tanto, la unión final de los tres nodos con las dos relaciones tendría como resultado gráfico el siguiente:

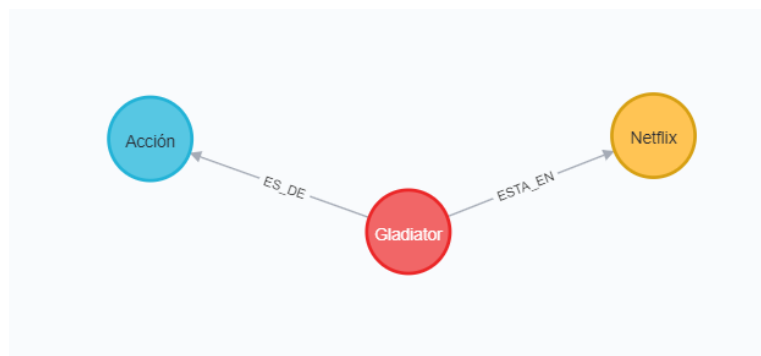


Imagen 3-6 Ejemplo gráfico

simular el catálogo real de estas plataformas, aun así, aunque se ha ampliado considerablemente la base de datos, sigue siendo ínfima si se compara con el contenido que realmente tienen dichas plataformas.

Tras la ampliación, la base de datos contiene 6 nodos Plataforma, 16 nodos Genero y 120 nodos Pelicula, llegando así a un total de 142 nodos.

En cuanto a relaciones, contiene 249 relaciones ES_DE y 120 relaciones ESTA_EN, llegando a un total de 369 relaciones.

El resultado gráfico de esta versión final de la base de datos es el siguiente:

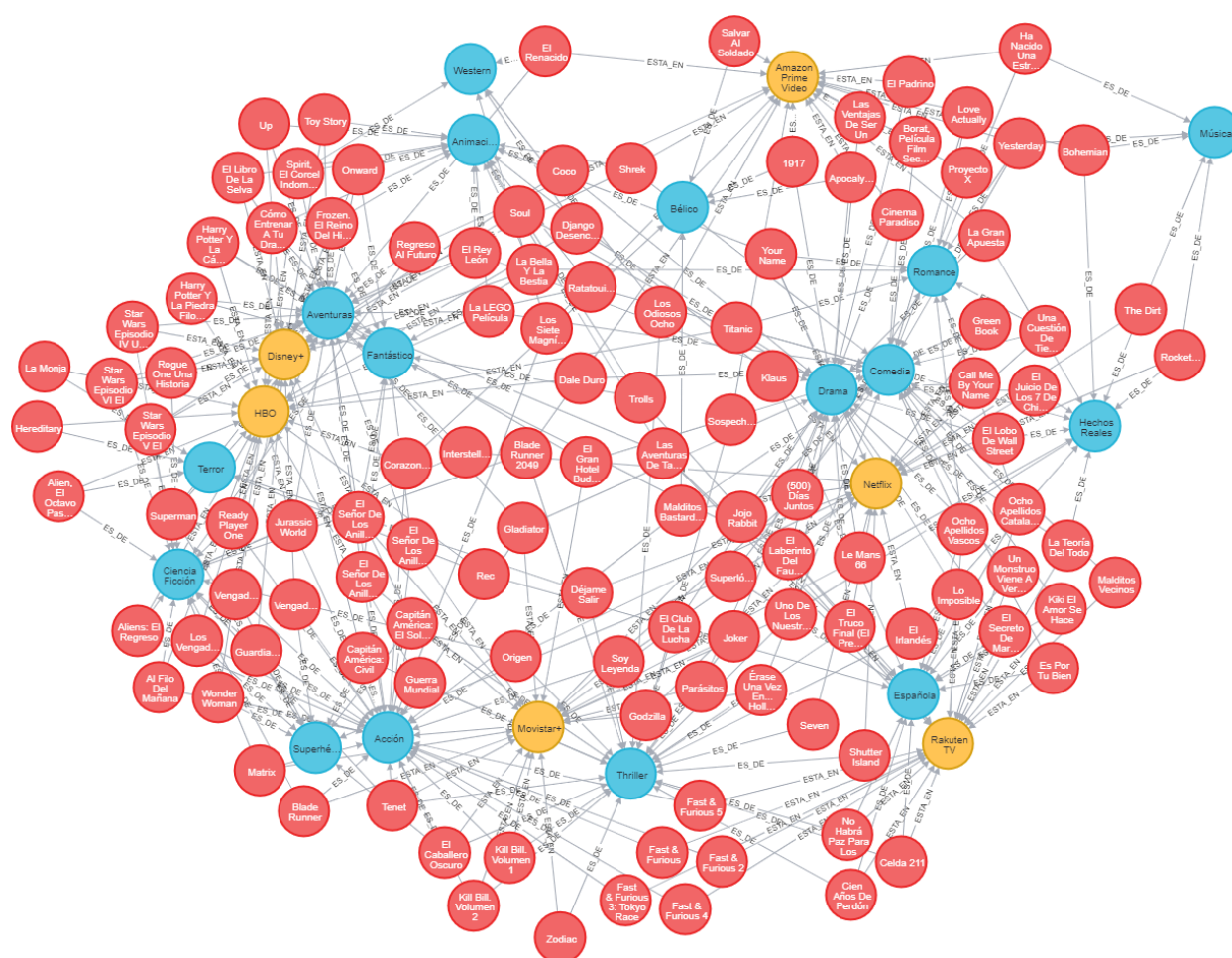


Imagen 3-8 Base de datos final

3.2 Código Python

La interfaz de esta aplicación está basada en la que aparece en este Github [8]. En cuanto al código y la funcionalidad son bastante diferentes.

La pantalla principal de la aplicación es en la cual se seleccionarán los géneros que interesan al usuario, como máximo podrán ser cinco géneros los que seleccione. La manera en que puede seleccionarlos es a través de un desplegable, en el cual, gracias a una llamada a la base de datos, aparecen todos los géneros que se han introducido en la base. Esta pantalla cuenta con cinco desplegables, uno por cada género que el usuario quiera o no seleccionar. Además de estos cinco desplegables, también cuenta con dos botones. Uno de nueva búsqueda en el que se cierra la ventana de la aplicación y se abre una nueva en la que se pueden introducir nuevos géneros para ser buscado. El otro botón es el de buscar que al darle busca en la base de datos la plataforma que mejor convenga según el algoritmo de recomendación.

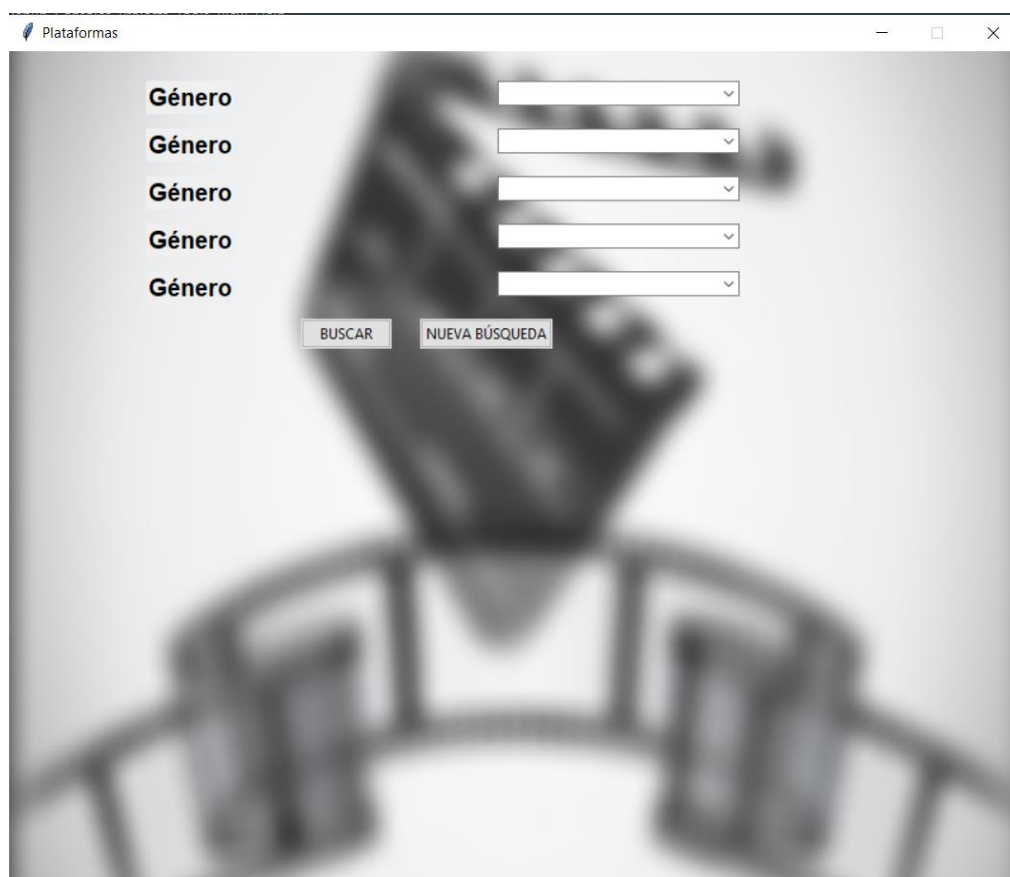


Imagen 3-9 Pantalla principal

En el código que se muestra a continuación se puede ver cómo se conecta la base de datos con la aplicación de Python. A continuación, aparecen las bibliotecas importadas y la creación de la ventana principal donde se añadirán todos los botones y desplegables, esta ventana cuenta con fondos aleatorios, que cambian al iniciarse. Por último, se ve la llamada a la base de datos para rellenar el vector que se usará en los desplegables con todos los géneros que aparecen en dicha base.

```

4  from neo4j import GraphDatabase
5  uri = "bolt://localhost:7687"
6  driver = GraphDatabase.driver(uri, auth=("Hugo", "Hugo"))
7
8  import sys
9  import os
10 from tkinter import Tk
11 from tkinter import ttk
12 from tkinter import messagebox
13 import tkinter as tk
14 from PIL import ImageTk, Image, ImageDraw, ImageFont
15 from random import randint, uniform, random
16
17 raiz = Tk()
18
19 raiz.title("Plataformas")
20 raiz.resizable(width=False, height=False)
21 raiz.geometry("850x700")
22 raiz.config(bg="grey")
23
24 aleatorio1 = randint(1, 12)
25 aleatorio2 = randint(1, 12)
26
27 imagen = Image.open("C:\\Users\\Usuario\\Desktop\\Universidad\\BusinessIntelligence\\Proyecto\\Imagenes\\aleatorio1.jpg")
28 imagen_de_fondo = ImageTk.PhotoImage(imagen)
29 fondo = tk.Label(raiz, image=imagen_de_fondo)
30 fondo.place(x=0, y=0, relwidth=1, relheight=1)
31
32
33 vecGen = []
34 final = []
35 libro = []
36
37 #Funciones
38 def generos():
39     vector = []
40     with driver.session() as session:
41         for record in session.run("MATCH (g:Genero)
42                                   RETURN g.nombre"):
43             vector.append(record["g.nombre"])
44     return vector

```

Imagen 3-10 Código Python 1

La siguiente parte del código es la referente al algoritmo de recomendación, el cual será explicado en su propio apartado, por lo que esta parte se saltará.

Así pues, el código siguiente muestra lo que ocurre una vez se presiona al botón "PELÍCULAS RECOMENDADAS" después de haberse introducido los géneros deseados y haber recomendado una plataforma.

En el código se ve cómo se crea una nueva ventana donde aparecerán las películas que se recomiendan ver para dicha plataforma, esta ventana también posee fondos aleatorios que cambian una vez se abre la ventana. Además, se ve cómo se llena el vector de los títulos de las películas que se van a recomendar mediante cinco llamadas a la base de datos. Estas llamadas buscan las películas que están en la plataforma deseada y que son de los géneros introducidos por el usuario. Para que se añada al vector no puede estar repetida, esta comprobación se hace a partir de la segunda llamada, ya que en la primera el vector está vacío y no se pueden añadir películas repetidas.

```

123
124 def infopeli():
125
126     info = tk.Toplevel(raiz)
127
128     info.title("Películas")
129     info.resizable(width=False, height=False)
130     info.geometry("850x770")
131     info.config(bg="grey")
132
133     titulo = []
134
135     aux1 = 90
136     true = 0
137
138     imagenFondo = Image.open("C:\\Users\\Usuario\\Desktop\\Universidad\\BussinessInteligenc
139     imagen_de_fondo = ImageTk.PhotoImage(imagenFondo)
140     fondo = tk.Label(info, image=imagen_de_fondo)
141     fondo.place(x=0, y=0, relwidth=1, relheight=1)
142
143     ##### COMBO1 #####
144
145     with driver.session() as session:
146         for record in session.run("MATCH (m: Pelicula)-[:ES_DE]->(g:Genero{nombre:'"+str(var
147             "MATCH (m)-[:ESTA_EN]->(l:Plataforma{nombre:'"+str(final
148             "RETURN m.nombre"):
149             titulo.append(record["m.nombre"])
150
151     ##### COMBO2 #####
152
153     with driver.session() as session:
154         for record in session.run("MATCH (m: Pelicula)-[:ES_DE]->(g:Genero{nombre:'"+str(var
155             "MATCH (m)-[:ESTA_EN]->(l:Plataforma{nombre:'"+str(final
156             "RETURN m.nombre"):
157             for i in range(len(titulo)):
158                 if titulo[i] == record["m.nombre"]:
159                     true = 1
160             if true == 0:
161                 titulo.append(record["m.nombre"])
162             true=0
163

```

Imagen 3-11 Código Python 2

Por último, la parte final del código es la que se encarga de mostrar el título de las películas que se ajustan al género o géneros y a la plataforma, además también se muestran todos los géneros a los que pertenece cada película recomendada mediante una llamada a la base de datos.


```

203 ##### FINAL #####
204
205     for i in range (len(titulo)):
206
207         generoPeli = []
208
209         with driver.session() as session:
210             for g in session.run("MATCH (m:Película)-[ES_DE]-(g:Genero)"
211                                 "WHERE m.nombre='"+str(titulo[i])+"'
212                                 "RETURN g.nombre"):
213                 generoPeli.append(g["g.nombre"])
214
215         generoPeliAux = ", ".join(generoPeli)
216
217         info.label2=ttk.Label(info)
218         info.label2.place(x=100, y= aux1 )
219         info.label2.config(text="Película: "+titulo[i]+"      Género/s: "+str(generoPeliAux))
220         info.label2.config(font="Arial 10 bold")
221         aux1 = aux1 + 34
222
223         info.label1=ttk.Label(info)
224         info.label1.place(x=325, y=30 )
225         info.label1.config(text=final[0])
226         info.label1.config(font="Arial 25 bold")
227
228         info.mainloop()
229
230     raiz.Button9 = ttk.Button(raiz, command=infopeli, text = "PELÍCULAS RECOMENDADAS")
231     raiz.Button9.place(x=455, y=335)
232
233

```

Imagen 3-12 Código Python 3

4. Algoritmo de recomendación

El algoritmo de recomendación en el que se basa esta aplicación es un algoritmo sencillo que consiste en cinco llamadas a la base de datos de las que se extrae el nombre de las plataformas que contengan las películas que son del género deseado y se añaden estos nombres a un vector. El código que se muestra a continuación es el referente a una de las cinco llamadas a la base de datos, una por cada desplegable.

```

52     var2=raiz.combo2.get()
53     with driver.session() as session:
54         for record in session.run("MATCH (m:Película)-[ES_DE]->(g:Genero{nombre: '"+str(var2)
55                                     "MATCH (m)-[ESTA_EN]->(l:Plataforma)"
56                                     "RETURN l.nombre");
57         vectorPlataforma.append(record["l.nombre"])
58 
```

Imagen 4-1 Llamada a la base de datos

Una vez que estos nombres están añadidos al vector, se procede a contar las apariciones de cada plataforma en el vector, para posteriormente extraer la plataforma con más apariciones, que es la que se está buscando.

En la primera imagen que se mostrará a continuación, aparece la función que cuenta las apariciones de cada plataforma en el vector y en la segunda imagen aparece donde se llama a dicha función y cómo se extrae la plataforma que más apariciones tiene.

Además, se muestra también el código donde se comprueba que se ha seleccionado algún género en los desplegables y si no es así, muestra una ventana de error.

```

239
240     def contarElementosLista(lista):
241
242         return {i:lista.count(i) for i in lista}
243 
```

Imagen 4-2 Función de contar elementos

```

96
97     if not vectorPlataforma:
98         messagebox.showerror("Error", "Introduce algún criterio")
99         nueva()
100
101     resultado = contarElementosLista(vectorPlataforma)
102     plataforma = max(resultado, key=resultado.get)

```

Imagen 4-3 Extraer máximo

Para acabar con el algoritmo de recomendación, se muestra en la interfaz un mensaje en el que aparece el nombre de la plataforma con más apariciones y el logotipo de dicha plataforma.

Este es el código de estas dos acciones mencionadas.

```

104     final.append(plataforma)
105
106     raiz.label=tkk.Label(raiz)
107     raiz.label.place(x=65, y=285 )
108     raiz.label.config(text="Según los criterios introducidos, la plataforma que se recomienda es: " + final[0])
109     raiz.label.config(font="Verdana 9 italic bold")
110
111
112     ##### IMAGEN PLATAFORMA #####
113
114     path = 'C:\\Users\\Usuario\\Desktop\\Universidad\\BussinessIntelligence\\Proyecto\\Imágenes\\Plataformas\\'+ str(final[0]) + ".jpg"
115     load = Image.open(path)
116     render = ImageTk.PhotoImage(load)
117     img = tkk.Label(raiz, image=render)
118     img.image = render
119     img.place(x=115 , y=315)
120

```

Imagen 4-4 Mostrar nombre y logo de plataforma

5. Análisis de resultados

En este apartado se mostrarán dos ejemplos con un pequeño análisis de cada uno de ellos para poder entender mejor el funcionamiento de la aplicación.

5.1 Ejemplo 1

Para este primer ejemplo el usuario va a seleccionar como géneros que le interesan superhéroes y ciencia ficción.

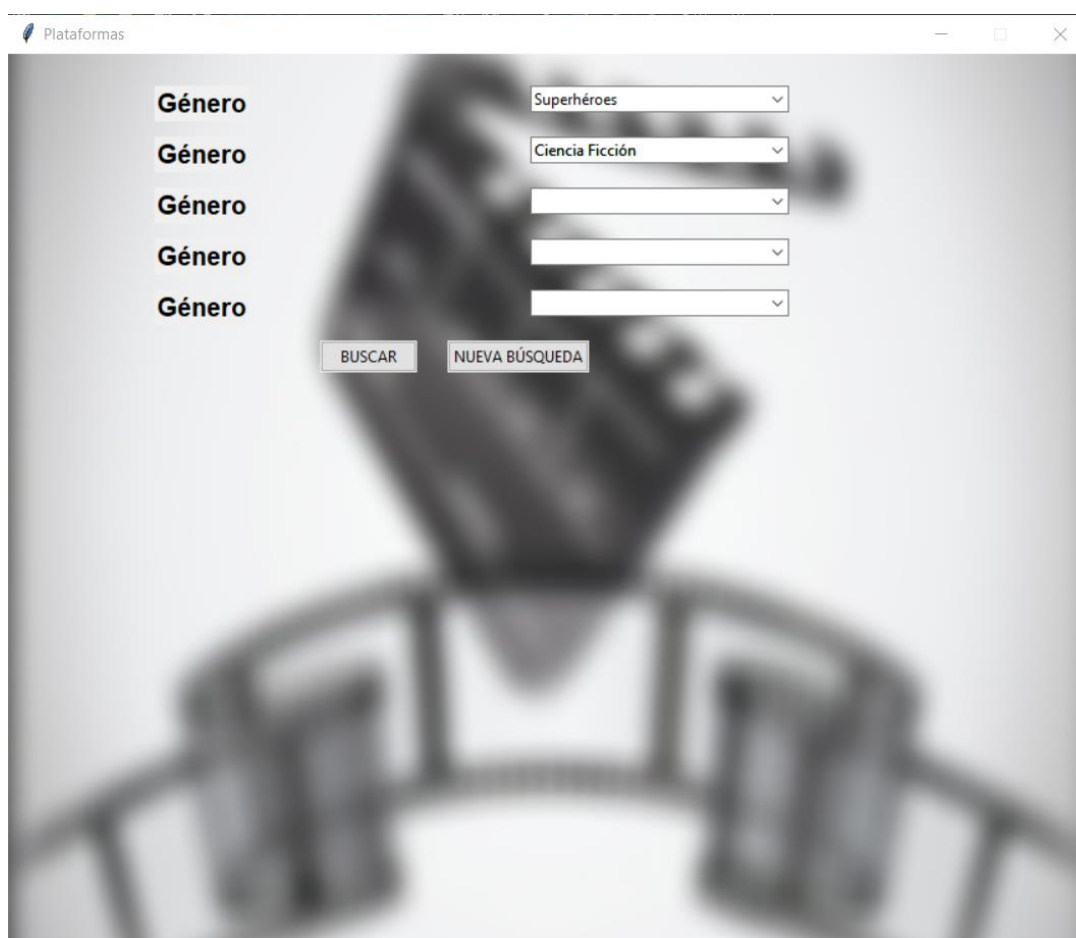
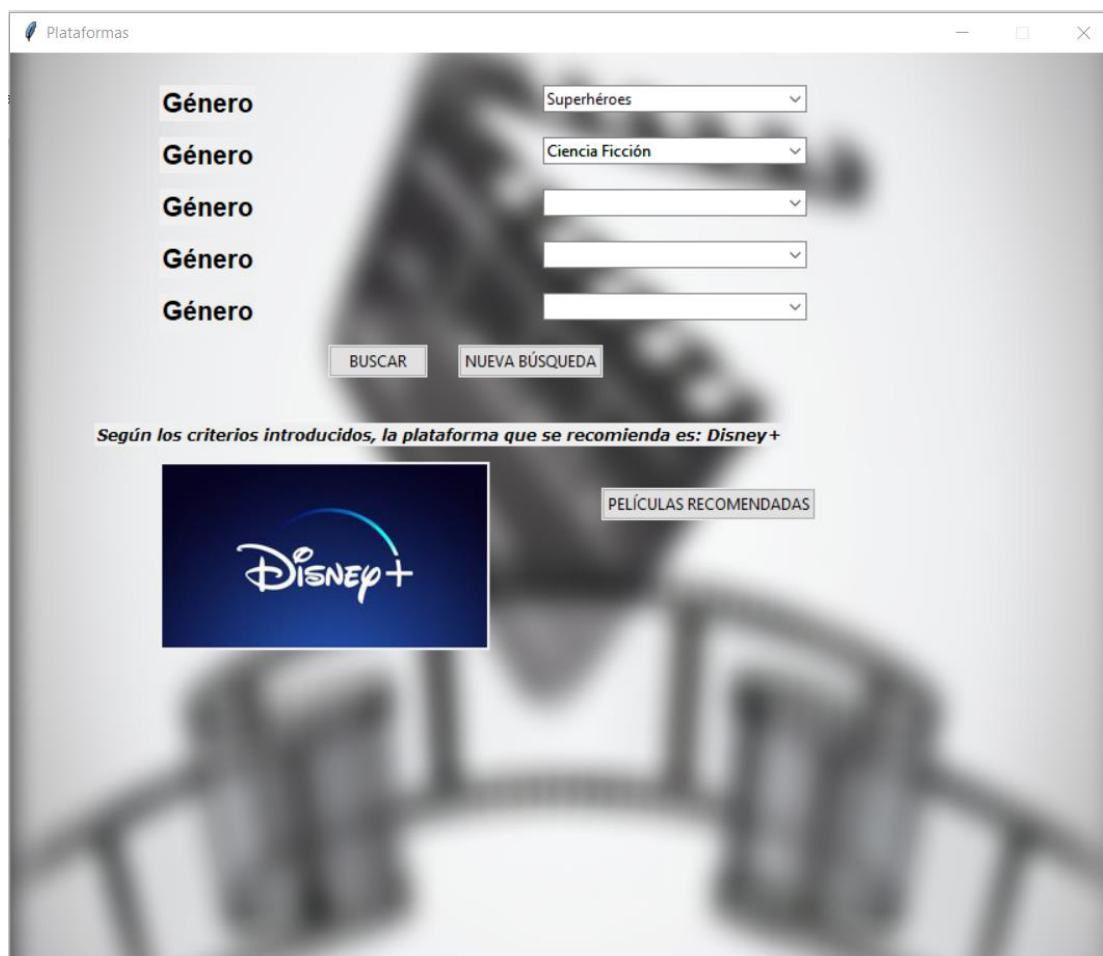
The image shows a web browser window titled 'Plataformas'. Inside, there is a search form. On the left, the word 'Género' is repeated five times in a vertical list. To the right of each 'Género' label is a dropdown menu. The first dropdown is set to 'Superhéroes' and the second to 'Ciencia Ficción'. The other three dropdowns are empty. Below the dropdowns are two buttons: 'BUSCAR' and 'NUEVA BÚSQUEDA'. The background of the form is a blurred image of a hand holding a globe.

Imagen 5-1 Ejemplo 1 primer paso

A continuación, el usuario presionará el botón “BUSCAR” para que la aplicación busque cual es la plataforma que recomienda habiendo introducido estos géneros.



Plataformas

Género Superhéroes

Género Ciencia Ficción

Género

Género

Género

BUSCAR NUEVA BÚSQUEDA

Según los criterios introducidos, la plataforma que se recomienda es: Disney+

Disney+

PELÍCULAS RECOMENDADAS

Imagen 5-2 Ejemplo 1 segundo paso

Como se ve en la imagen, la plataforma que se recomienda es Disney+, ya que según el algoritmo de recomendación y según la base de datos introducida, Disney+ aparece 11 veces en el vector de las plataformas, por 3 veces Netflix, 2 veces Amazon Prime Video, 7 veces HBO, 4 veces Movistar+ y 0 veces Rakuten TV. Así pues, la plataforma recomendada es Disney+.

A continuación, el usuario tiene la posibilidad de dar al botón “PELÍCULAS RECOMENDADAS” para que aparezcan todas las películas que pertenecen a los géneros cinematográficos introducidos y que están en esta plataforma.

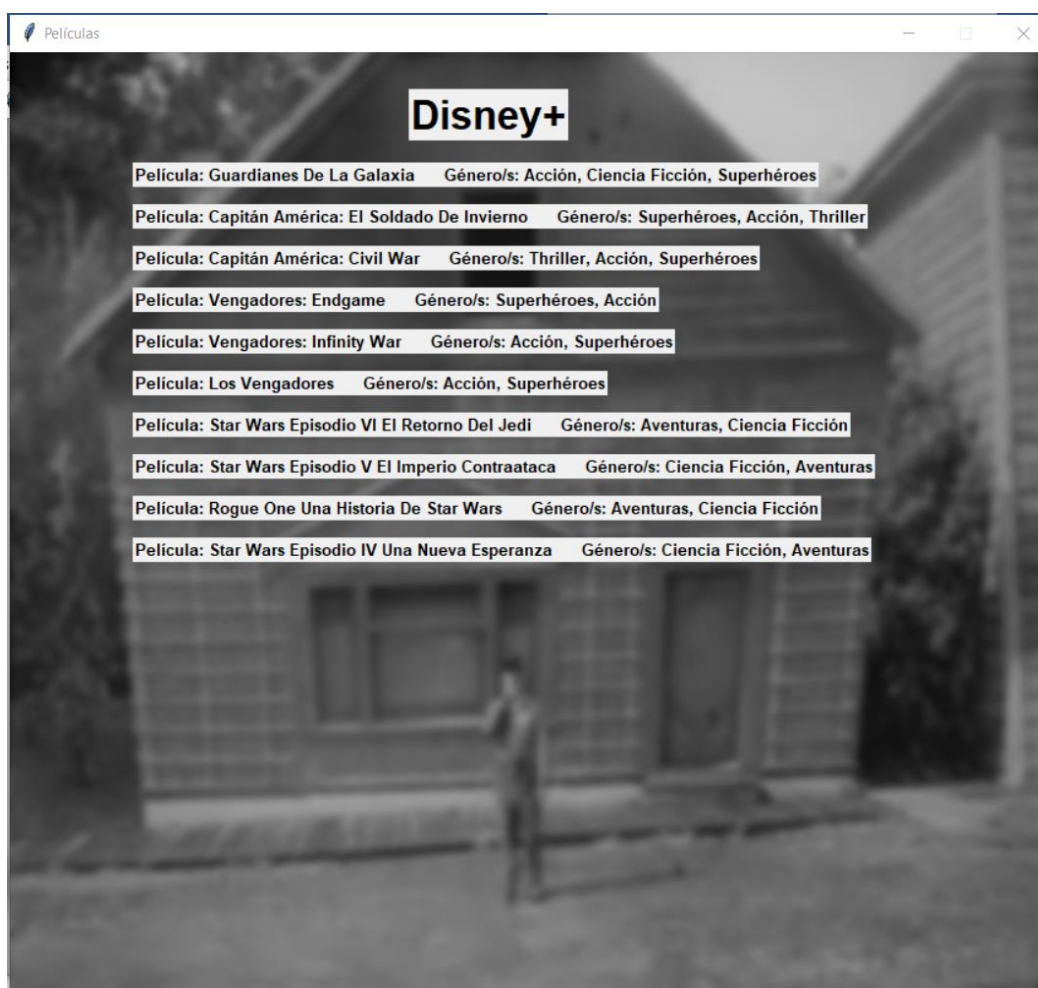


Imagen 5-3 Ejemplo 1 tercer paso

Según se muestra en la imagen aparecen 10 películas que pertenecen a los géneros introducidos y que se encuentran en la plataforma recomendada.

Se muestran 10 películas y en el conteo aparece Disney+ 11 veces porque en este caso la película de Guardianes De La Galaxia es de los dos géneros introducidos, tanto de superhéroes como de ciencia ficción, por lo que a la hora de contar se cuenta dos veces, pero para mostrar la película en la lista de películas recomendadas solo se muestra una vez.

Así pues, se puede dar por concluida la ejecución de este primer ejemplo y el usuario ha podido obtener información útil en cuanto a la plataforma a la que debería de estar suscrito.

5.2 Ejemplo 2

Para este segundo ejemplo, el usuario introducirá Thriller, Acción, Drama y Hechos Reales como los géneros cinematográficos que le interesan en las plataformas de video bajo demanda.

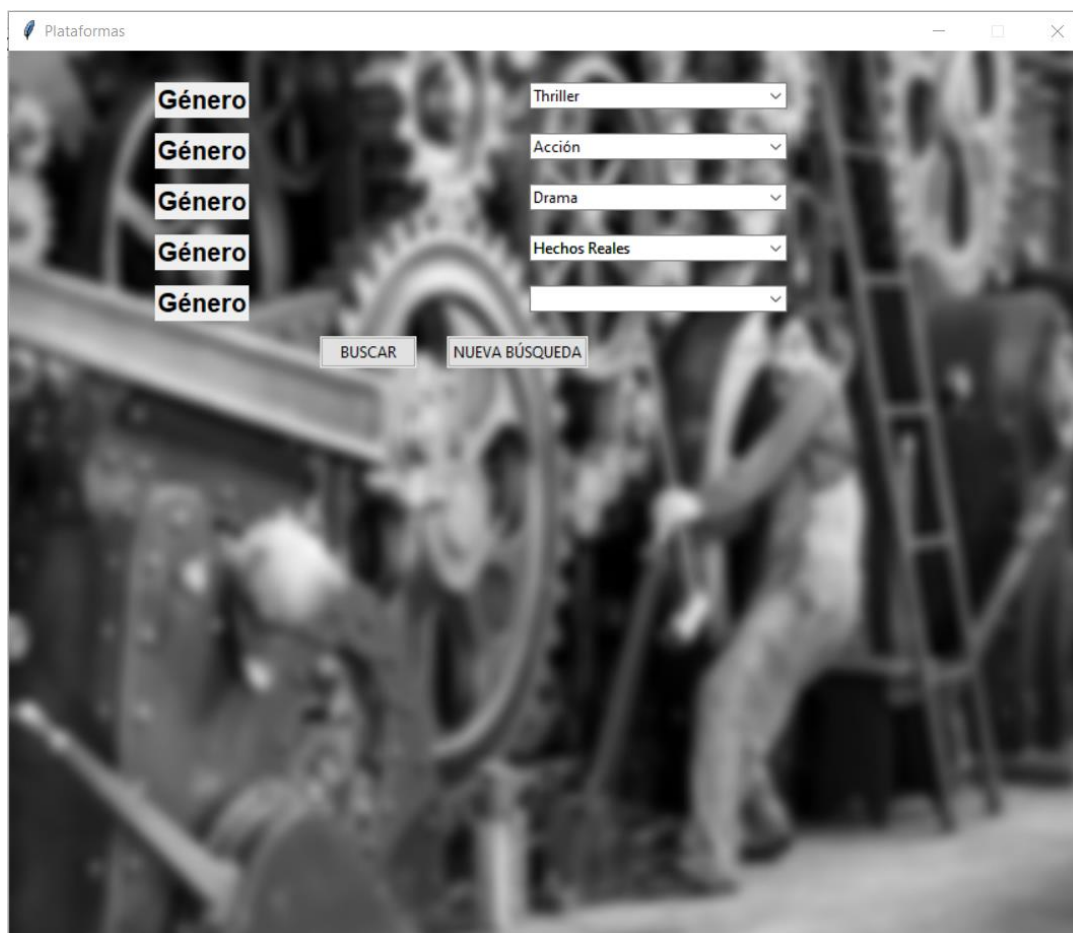


Imagen 5-4 Ejemplo 2 primer paso

A continuación, el usuario presionará el botón “BUSCAR” para que la aplicación busque cual es la plataforma que recomienda habiendo introducido estos géneros.

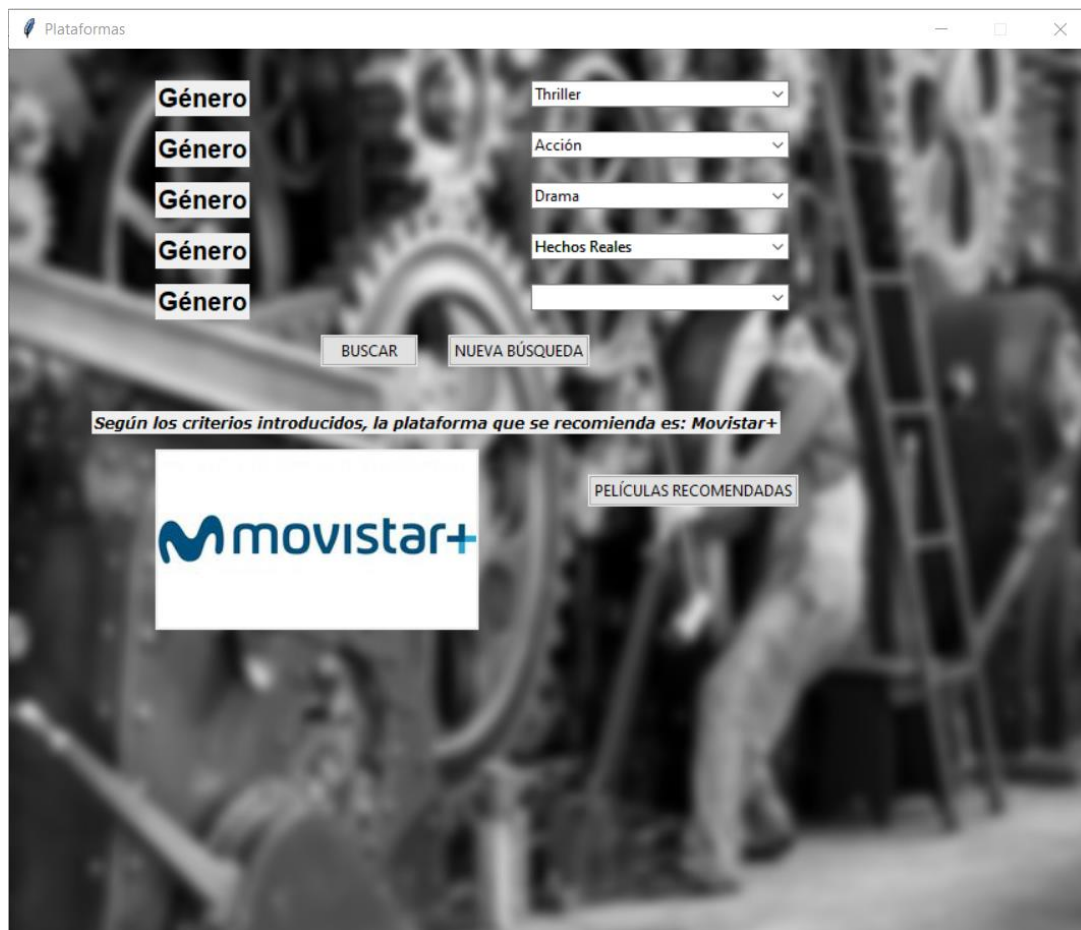


Imagen 5-5 Ejemplo 2 segundo paso

Como se ve en la imagen, la plataforma que se recomienda es Movistar+, ya que según el algoritmo de recomendación y según la base de datos introducida, Movistar+ aparece 29 veces en el vector de las plataformas, por 21 veces Netflix, 12 veces Amazon Prime Video, 10 veces HBO, 11 veces Disney+ y 14 veces Rakuten TV. Así pues, la plataforma recomendada es Movistar+.

A continuación, el usuario tiene la posibilidad de dar al botón “PELÍCULAS RECOMENDADAS” para que aparezcan todas las películas que pertenecen a los géneros cinematográficos introducidos y que están en esta plataforma.



Imagen 5-6 Ejemplo 2 tercer paso

Según se muestra en la imagen aparecen 18 películas que pertenecen a los géneros introducidos y que se encuentran en la plataforma recomendada.

Se muestran 18 películas y en el conteo aparece Movistar+ 29 veces porque hay películas que pertenecen a varios de los géneros introducidos, por lo que a la hora de contar se cuentan varias veces, pero para mostrar las películas en la lista de películas recomendadas solo se muestran una vez.

Así pues, se puede dar por concluida la ejecución de este segundo ejemplo y el usuario ha podido obtener información útil en cuanto a la plataforma a la que debería de estar suscrito.

6. DAFO

Tabla 6-1 DAFO

Fortalezas	Debilidades	Oportunidades	Amenazas
<ul style="list-style-type: none"> - Interfaz sencilla - Visualmente atractiva - Escalabilidad - Datos comprobados - Utilidad real 	<ul style="list-style-type: none"> - Base de datos escasa - Tiempo de desarrollo escaso - Recomendaciones básicas 	<ul style="list-style-type: none"> - Plataformas de streaming cada vez más usadas - Situación actual 	<ul style="list-style-type: none"> - Competencia en las páginas web

6.1 Fortalezas

La aplicación consta de una interfaz sencilla y visualmente atractiva que hace que el uso de la misma sea apto para cualquier usuario de cualquier edad y con cualquier tipo de conocimiento sobre nuevas tecnologías.

En cuanto a la escalabilidad, esta aplicación es totalmente escalable. Cambiando la base de datos o añadiendo nuevos elementos a ella, la aplicación seguiría funcionando sin ningún tipo de problema y sin necesidad de tocar nada en el código.

Aunque la actual base de datos es pequeña, los datos que han sido introducidos están totalmente comprobados y contrastados. Todas las películas se encuentran realmente en dichas plataformas en el momento en el que se está escribiendo este documento.

La fortaleza más importante es la utilidad real que tiene esta aplicación, tanto a la hora de recomendar plataforma como a la hora de recomendar películas de dicha plataforma que sean de los géneros introducidos. Esta aplicación aporta utilidad y con ella se puede ahorrar tiempo y dinero a la hora de escoger plataforma de entretenimiento.

6.2 Debilidades

Una de las debilidades más importantes es la base de datos, ya que contiene una cantidad ínfima de películas por plataforma si lo comparamos con las que realmente tienen estas plataformas. Además, las películas se están actualizando casi diariamente, por lo que, si esta base de datos no se actualiza, llegará un momento en el que la aplicación no sirva para nada.

Otra debilidad ha sido el tiempo que se ha dispuesto para el desarrollo de la aplicación, teniendo que simplificar las cosas y no poder realizar operaciones más complejas que hubieran supuesto mucho mayor tiempo de desarrollo, tiempo del que no se disponía.

Esta aplicación realiza recomendaciones básicas, lo que quiere decir que el algoritmo solo se basa en el número de películas por género y plataforma, cuando lo ideal sería que se fijase en más elementos como por ejemplo la calidad de las películas.

6.3 Oportunidades

La mayor de las oportunidades es la situación sanitaria actual, ya que la gente tiene que estar más tiempo en casa que fuera de ella, por lo que las plataformas de streaming son la mayor fuente de entretenimiento actualmente. Además de esto, también se suma la situación económica que hace que la capacidad adquisitiva de la mayoría de las personas se haya visto reducida en gran medida. Debido a estos dos factores, la gente quiere suscribirse a plataformas de streaming, pero no puede suscribirse a todas, así pues, esta aplicación da solución a ambos problemas, recomendado una plataforma de video bajo demanda según sus gustos y además les permite ahorrar, ya que no tendrán la necesidad de estar suscritos a todas las plataformas.

6.4 Amenazas

La mayor de las amenazas está en las páginas web, ya que son mucho más inmediatas que una aplicación de escritorio como esta, por ello ahí es donde está la principal competencia para esta aplicación.

7. Líneas de futuro

En cuanto al futuro de la aplicación hay varias direcciones en las que hay que mirar, el perfeccionamiento y la expansión.

En cuanto al perfeccionamiento, se pueden añadir un mayor número de atributos a las películas, en vez de solo el título. Atributos como: director, actores, duración o valoración. Estos atributos servirían para realizar búsquedas en la base de datos, ya que actualmente el algoritmo de recomendación solo busca por géneros cinematográficos. Además, uno de los atributos más importantes es la valoración. Como se ha dicho anteriormente, el algoritmo solo busca cantidad de películas, pero debería de tener en cuenta también la calidad de estas para que la recomendación sea mejor.

Por otro lado, el perfeccionamiento de la base de datos. Las plataformas de streaming actualizan sus catálogos casi diariamente, por lo que la base de datos debería ser actualizada constantemente y por supuesto ampliada.

Un punto importante que está entre el perfeccionamiento y la expansión es el añadir series a la aplicación. Actualmente, se consume muchísimo contenido audiovisual a través de las plataformas de video bajo demanda, tanto series como películas, pero esta aplicación solo tiene en cuenta para recomendar plataformas las películas, cuando el catálogo de series es igual de importante que el de películas. Por esta razón, este es un punto muy importante tanto para el perfeccionamiento como para la expansión.

Hablando solo de expansión, la funcionalidad de la aplicación debería de ser trasladada también a una página web o una aplicación móvil, ya que son mercados más amplios e inmediatos, por lo que la cantidad de usuarios que utilizarían la aplicación se ampliaría en gran medida.

Además, se debería pasar de un mercado nacional a uno global, es decir, expandirse internacionalmente. Los catálogos de las plataformas de streaming no son iguales para todo el mundo, por esta razón, otra forma de expandirse sería la de crear bases de datos

propias para cada país con sus películas por plataforma y con las plataformas disponibles en cada país. Así se los usuarios de la aplicación se ampliarían en una gran medida.

En conclusión, la aplicación tiene mucho margen de mejora con muchos puntos a tener en cuenta que en algún futuro podrían ser cumplidos y así llegar a tener un producto final realmente útil y de gran calidad.

8. Lecciones aprendidas

En lo personal, el haber desarrollado esta aplicación me ha aportado mayor soltura a la hora de programar y entender el lenguaje de Python, además de quitarle el miedo a lenguajes de programación que no he visto durante la carrera.

Asimismo, también he adquirido conocimientos en Neo4j y en Cypher. Me he dado cuenta de que las bases de datos basadas en grafos son muy útiles y que realmente son sencillas de programar. Aunque al principio llegara a ser difícil y frustrante comprender el funcionamiento de estas, al final las posibilidades que pueden llegar a tener llegan a sorprender. Para muchas aplicaciones, este tipo de bases de datos pueden ser mucho más útiles que las bases de datos basadas en tablas que siempre se usan.

En conclusión, estoy satisfecho con el trabajo realizado y con los conocimientos que me ha aportado el haber desarrollado esta aplicación, aunque podría ser mejorable en muchos aspectos, la aplicación funciona bien y es capaz de hacer recomendaciones con sentido.

9. Bibliografía

- [1] «Neo4j,» [En línea]. Available: <https://neo4j.com/>. [Último acceso: 9 Febrero 2021].
- [2] «Anaconda,» [En línea]. Available: <https://www.anaconda.com/>. [Último acceso: 9 Febrero 2021].
- [3] «File:Anaconda Logo.png,» [En línea]. Available: https://en.wikipedia.org/wiki/File:Anaconda_Logo.png. [Último acceso: 9 Febrero 2021].
- [4] «tkinter — Interface de Python para Tcl/Tk,» [En línea]. Available: <https://docs.python.org/es/3/library/tkinter.html>. [Último acceso: 9 Febrero 2021].
- [5] JustWatch, «Todos los servicios de streaming en una aplicación.,» [En línea]. Available: <https://www.justwatch.com/>. [Último acceso: 10 Febrero 2021].
- [6] «FilmAffinity España,» [En línea]. Available: <https://www.filmaffinity.com/es/main.html>. [Último acceso: 10 Febrero 2021].
- [7] elGazapo, «¿Aún no conoces FilmAffinity?,» Marzo 2013. [En línea]. Available: <https://elgazapo.wordpress.com/2013/03/13/filmaffinity/>. [Último acceso: 10 Febrero 2021].
- [8] dalonr00, «Github,» 18 Diciembre 2019. [En línea]. Available: <https://github.com/dalonr00/SIBI>. [Último acceso: 10 Febrero 2021].