ITESM

CAMPUS CEM

PROYECTO INTEGRADOR

RETO DE PROGRAMACIÓN ORIENTADA A OBJJETOS

José Alonso Segura De Lucio: A01747872

Elizabeth Saldaña Andriano A01747979

ENTREGA: 18 DE JUNIO DEL 2022

ÍNDICE

Introducción

Hoy en día las plataformas de streaming son la forma más sencilla para consumir medios audiovisuales de todo tipo, desde Netflix que contiene un poco de muchos estudios, hasta gigantes corporativos como Disney Plus que colocan todo su catálogo al alcance de unos cuantos clics. Desde hace algunos años, las personas han ido modificando sus patrones de consumo de entretenimiento, donde antes se mantenía la televisión como el principal medio, actualmente se ha superado el número de horas que alguien pasa viendo contenido en algunas de estas plataformas a comparación de la televisión abierta o por cable (Molla, 2018), lo cual ha vuelto a estos negocios demasiado rentables.

Entre algunas de las razones de ello, se encuentra que el contenido de streaming esta listo para ser visualizado cuando el usuario quiere hacerlo, bastará tener el servicio contratado, una conexión a internet y seleccionar la serie o película a ver, en contraste con la televisión, donde el usuario debe adaptarse al horario y oferta programada. Por lo tanto, este tipo de servicios se han convertido en productos de alta demanda y algo que garantiza si este tendrá o no éxito en el mercado es la funcionalidad de la plataforma en la web.

A continuación, se muestra ordenadamente lo incluido y realizado para darle la solución al reto de POO, como lo es el diagrama de clases, su justificación, la redacción de cómo funciona el programa, los casos de prueba, es decir, las capturas de la corrida del código y la demostración de que se cumple lo solicitado en la rúbrica.

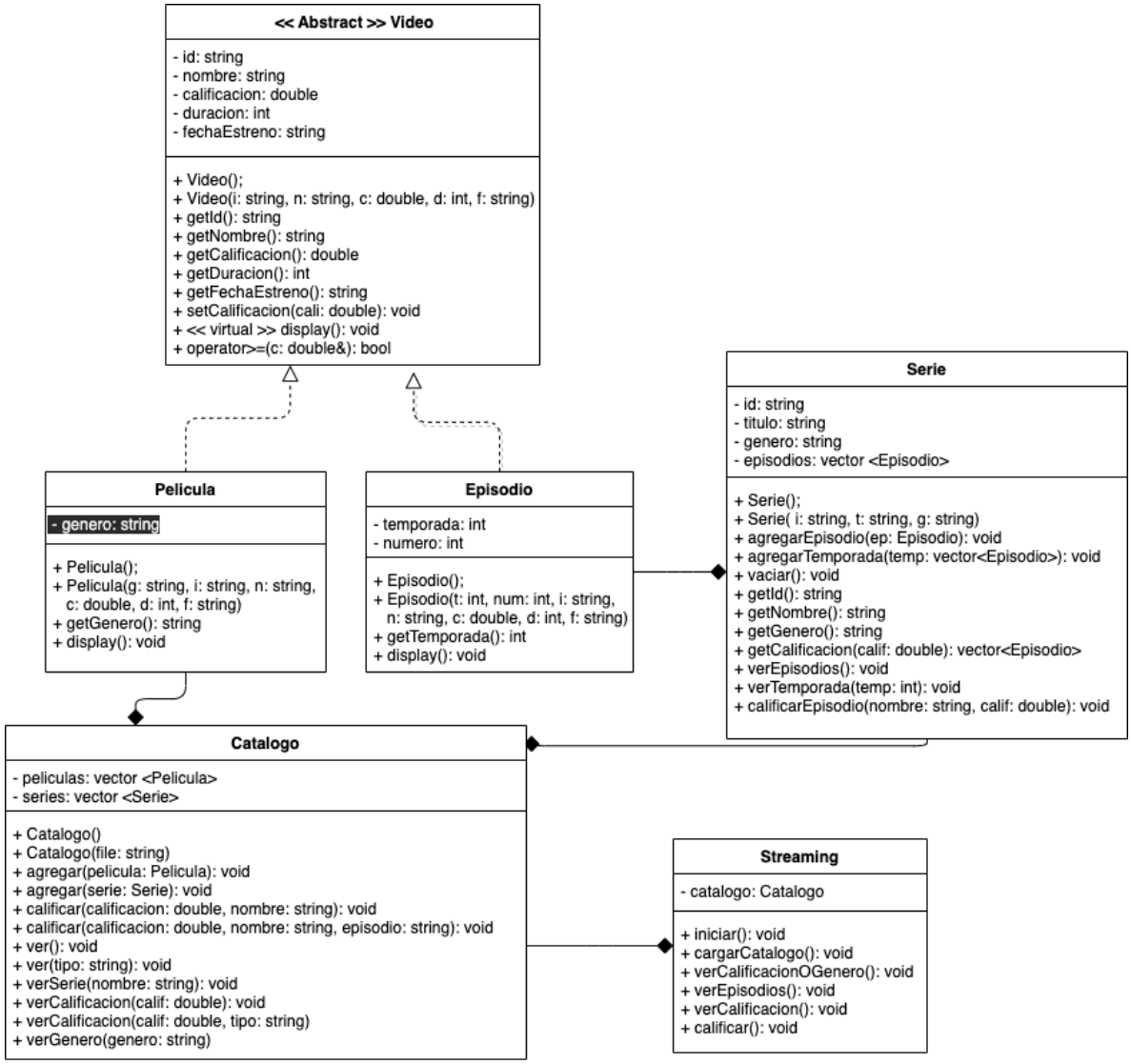
El problema planteado es: Moldear una plataforma en la que se quiere trabajar con dos tipos de videos: películas y series. Todo video tiene un ID, un nombre, una duración, una calificación y un género.

Las series tienen episodios y cada episodio tiene un título y temporada a la que pertenece de esta misma serie. Nos interesa conocer la calificación de todos los videos al igual que sus demás datos por medio de un filtro, para así poder seleccionar el que deseemos ver. La calificación mencionada está en escala de 1 a 10 donde 10 es la mejor calificación.

La plataforma debe ser capaz de:

- Mostrar los videos en general con sus datos
- Mostrar los videos de una determinada serie o película con sus datos
- Cambiar la calificación del video seleccionado y reproducir los contenidos multimedia.

Diagrama UML



Argumentación diagrama

Se optó por este diagrama puesto que es la mejor forma de representar un sistema de streaming de películas y series en forma de código. Video sería una clase abstracta que solo sirve para aquellas clases derivadas de esta. Es decir que Video no hace nada por sí sola, pero es la encargada de darle polimorfismo al programa (esto con la función virtual pura de display(). Las dos clases que son videos son Película y Episodio, pues ambas comparten propiedades y el método de display(). Asimismo Serie contiene un vector de Episodios y Catálogo contiene dos vectores, uno de Películas y otro de Series. La clase Catalogo es la encargada de leer el archivo de datos, y de filtrar las películas, series y episodios en general. La clase Streaming es la encargada de mostrar el menú y pedir datos al usuario, igualmente se encarga de manejar excepciones por si el usuario ingresa algún dato incorrecto. Finalmente en el main solo se crea un objeto del tipo Streaming y se llama a su método de iniciar().

Capturas de la corrida

```
Menu:
1. Cargar archivo de datos
2. Ver videos con calificacion o genero especificos
3. Ver episodios de una serie
4. Ver peliculas o episodios con cierta calificacion
5. Calificar un video
0. Salir

Ingrese el numero de la opcion elegida: 1
Escribe el nombre del archivo (ej. base.csv):
BasePeliculas.csv

**********    Catalogo    **********

Pelicula tt0107290
Nombre: Jurassic Park
Calificacion: 8.1
Duracion: 127
Fecha De Estreno: 11/06/1993
Genero: Accion/Aventura/Sci-Fi/Suspenso

Pelicula tt0796366
Nombre: Star Trek
Calificacion: 7.9
Duracion: 127
Fecha De Estreno: 08/05/2009
Genero: Accion/Aventura/Sci-Fi

Pelicula tt0120762
Nombre: Mulan
Calificacion: 7.6
Duracion: 88
Fecha De Estreno: 6/19/1998
Genero: Animacion/Aventura/Familiar/Fantasy/Musical/Guerra

Pelicula tt0120131
Nombre: The Lion King 2: Simba Pride
Calificacion: 6.5
Duracion: 81
Fecha De Estreno: 10/27/1998
Genero: Animacion/Aventura/Drama/Familiar/Musical/Romance

Pelicula tt0078346
Nombre: Superman
Calificacion: 7.3
Duracion: 143
Fecha De Estreno: 12/15/1978
Genero: Accion/Aventura/Drama/Sci-Fi
```

```
Ingrese el numero de la opcion elegida: 2
Ingresa 1 para ver calificaciones o 2 para ver generos: 2
Ingresa el genero: Animacion

********************    Animacion     ********************

Pelicula tt0120762
Nombre: Mulan
Calificacion: 7.6
Duracion: 88
Fecha De Estreno: 6/19/1998
Genero: Animacion/Aventura/Familiar/Fantasy/Musical/Guerra

Pelicula tt0120131
Nombre: The Lion King 2: Simba Pride
Calificacion: 6.5
Duracion: 81
Fecha De Estreno: 10/27/1998
Genero: Animacion/Aventura/Drama/Familiar/Musical/Romance

Pelicula tt4116284
Nombre: The Lego Batman Movie
Calificacion: 7.3
Duracion: 104
Fecha De Estreno: 10/02/2017
Genero: Animacion/Accion/Comedia/Familiar
```

```
Ingrese el numero de la opcion elegida: 3
Ingresa el nombre de la serie: Modern Familiar

Serie tt1442437
Titulo: Modern Familiar
Genero: Comedia/Drama/Romance

   Episodio tt1444504
   Nombre: Pilot
   Calificacion: 8.4
   Duracion: 22
   Fecha De Estreno: 9/23/2009
   Temporada: 1
   Numero: 1

   Episodio tt1492032
   Nombre: The Bicycle Thief
   Calificacion: 8.2
   Duracion: 22
   Fecha De Estreno: 9/23/2009
   Temporada: 1
   Numero: 2

   Episodio tt1520618
   Nombre: Come Fly With Me
   Calificacion: 8
   Duracion: 22
   Fecha De Estreno: 07/10/2009
   Temporada: 1
   Numero: 3
```

```
Ingrese el numero de la opcion elegida: 4
Ingresa 1 para ver peliculas o 2 para ver series: 1

Ingresa el numero de calificacion: 8.5

**********    Calificacion Peliculas 8.5    **********

Pelicula tt0088763
Nombre: Back To The Future
Calificacion: 8.5
Duracion: 116
Fecha De Estreno: 03/07/1985
Genero: Aventura/Comedia/Sci-Fi

Pelicula tt0900488
Nombre: The Prestige
Calificacion: 8.5
Duracion: 130
Fecha De Estreno: 10/20/2006
Genero: Drama/Misterio/Sci-Fi/Suspenso

Pelicula tt0110912
Nombre: Pulp Fiction
Calificacion: 8.9
Duracion: 154
Fecha De Estreno: 10/14/1994
Genero: Crimen/Drama

Pelicula tt0468569
Nombre: The Dark Knight
Calificacion: 9
Duracion: 152
Fecha De Estreno: 7/18/2008
Genero: Accion/Crimen/Drama/Suspenso

**************************************************
```

```
Ingrese el numero de la opcion elegida: 5
Ingresa 1 para calificar una pelicula o 2 para calificar una serie: 1

Ingresa el nombre de la película: Jurassic Park

Ingresa la nueva calificacion: 9

Pelicula tt0107290
Nombre: Jurassic Park
Calificacion: 9
Duracion: 127
Fecha De Estreno: 11/06/1993
Genero: Accion/Aventura/Sci-Fi/Suspenso
```

```
Ingrese el numero de la opcion elegida: 1
Escribe el nombre del archivo (ej. base.csv):
NoExisto.csv

Error el archivo NoExisto.csv no existe
```

Argumentación proyecto integrador

Clases

Para definir las clases se usó como base lo planteado en la situación problema acerca de las series y películas, donde ambos son un video. No obstante, se determinó que las películas si son videos, pero que las series son conjuntos de videos llamados episodios. A partir de esto se tuvo que existiría una clase abstracta conocida como video que contiene la base tanto para dos subclases conocidas como película y episodio. Como la serie es un conjunto de episodios, esta se puede considerar como un vector de episodios, pero como también contienen otra información por aparte, esto las convierte en otro objeto que pasa a ser la clase serie. Las series y películas se agrupan en un mismo espacio que es el conjunto total de ambas, a esto se le conoce como catálogo y es una clase que tiene un vector para series y otro para películas. Finalmente, el usuario no tiene una interacción directa con el catálogo, sino con la plataforma de streaming que está usando, por lo que esta es una clase que contiene un catálogo.

Herencia y modificadores de acceso

Dado que en el análisis del proyecto se identificaron dos clases con características similares, se optó por realizar una herencia jerárquica que agrupara las características en común de las películas y los episodios. El estándar de programación que se usó en todo el programa fue escribir las variables privadas y los métodos públicos, los cuales serían después utilizados por la clase que contenía a objetos de otra clase, por lo que era posible usar modificadores de acceso públicos que garantizan la seguridad de las variables privadas sin sacrificar la utilidad de los métodos públicos de cada clase.

Sobrecarga y sobreescritura de métodos

Como se usó una clase abstracta como base para las películas y episodios, se declaró una función virtual pura (display), esta se tuvo que sobreescribir para cada una de las clases secundarias con instrucciones diferentes de acuerdo al contenido de cada una tanto como un requerimiento como para aprovechar las ventajas de la herencia. Por otro lado, debido a la implementación de polimorfismo en las diversas clases, una forma sencilla de facilitar muchos métodos que hacían lo mismo pero con diferente información, fue sobrecargar los métodos para que hiciera cada quien el manejo correcto de los datos sin la necesidad de que el programador tenga que aprender de memoria cada uno de los nombres de los métodos específicos, sino métodos generales con parámetros diferentes.

Polimorfismo

Algunas de las instrucciones que se pedían aplicaban tanto para todo el catálogo como para una sola categoría de videos, por lo que era mucho más sencillo usar el mismo nombre del método si es que su funcionalidad era prácticamente idéntica, lo cual facilita el manejo de la clase. Por este motivo, existen algunos métodos (principalmente en catálogo) que usan polimorfismo para realizar el mismo trabajo sin importar si se trata de una serie, una película o las dos. El ejemplo más claro es la función agregar dentro de catálogo; en este se

utiliza polimorfismo para que en la implementación de la clase no sea necesario investigar cuál es el método que puede agregar un tipo de video en especifico al catálogo, dado que si la función recibe como parámetro una película, la función que contiene ese parámetro la agregará al vector de películas y funciona de la misma forma para las series.

Clases abstractas

Las películas y episodios comparten muchas características en común que son agrupadas en la categoría de videos y luego pasan a ser heredadas. Sin embargo, no puede existir un video en sí mismo, puesto que todo video es una serie o una película, por lo que no debe ser posible crear un objeto del tipo video. En consecuencia, se usaron funciones virtuales puras en la construcción de la clase para garantizar que no puedan existir objetos del tipo video, así como para que las clases que heredan tengan obligatoriamente implementados algunos métodos funcionales para el manejo de la plataforma como el display.

Sobrecarga de operador

Puesto que el programa necesita un sistema de filtración conforme la calificación de los videos, se optó por sobrecargar el operador >= (mayor o igual que) en la clase abstracta Video. Esta sobrecarga regresa un booleano y recibe un float. Regresa True o False dependiendo de si el video tiene una calificación mayor o igual que el float recibido. Esto se hizo para aumentar la legibilidad del código, pues otra alternativa era simplemente llamar al getter de calificación del video y compararlo con el float requerido. Sin embargo para fines prácticos se tomó la elección de sobrecargar dicho operador para que solo se pudiera poner de la siguiente forma en el código: episodio>calificación.

Casos en los que no funcionaría el código

El código fue hecho especialmente para manejar errores por parte del usuario. Esto quiere decir que el programa no se detendrá incluso si el usuario mete valores erróneos, simplemente mostrará el error pero seguirá corriendo. Los casos específicos en donde ocurren este tipo de errores es cuando al usuario se le pide ingresar un número e ingresa letras. Otro caso específico es cuando el usuario introduce un nombre de archivo que no existe.

Conclusión del reto

En conclusión, se logró demostrar el uso correcto de los conceptos solicitados, que son polimorfismo, herencia, las excepciones y, por último, la sobrecarga.

En las capturas de pantalla se observó la ejemplificación de la corrida de la plataforma creada, indicando que funciona perfectamente.

Cuando se implementan conceptos de programación orientada a objetos como el polimorfismo o herencia nos damos cuenta de que hay diversas formas de llegar al mismo resultado, pero justamente en este proceso de abstracción es en donde se tiene que analizar cuál sería el sistema más adecuado para plasmarlo a código.

El polimorfismo sin duda es una herramienta que nos sirve para poder plasmar sistemas de la vida real en programas, tal como lo fue en este proyecto. Cada aspecto de POO nos sirve para plasmar aquello que vemos en la vida real en sistemas que puedan ser usados de manera eficiente.

Este tipo de sistemas nos permite automatizar y realizar de forma más rápida aquellas acciones en las que manualmente nos tardaríamos muchísimo más. Tal es el caso de clasificar películas y episodios conforme su género y calificación. Manualmente sería una acción demasiada tediosa y repetitiva, pero con ayuda de programación orientada a objetos podemos hacer que todo quede automatizado por medio de un programa.

Alonso Segura: Me gustó este proyecto puesto que me ayudó a ver en qué fallaba y qué partes dominaba bien, además de que ahora sé un poco más acerca de este paradigma. Fue bastante retador, pero eso mismo hizo que mi aprendizaje fuera bastante. Ahora domino más los temas de herencia y polimorfismo.

Elizabeth Andriano: Este proyecto considero fue una forma efectiva de poner en práctica los conocimientos de todo el semestre, principalmente de polimorfismo, dado que mediante su uso se podía facilitar el manejo de métodos de cada clase. Finalmente, me gustó que es similar al proyecto del semestre pasado, ya que después de agregar los nuevos conceptos de herencia y polimorfismo, pude identificar el avance dentro de la programación al ver que el código era mucho más legible, contaba con menos errores y es mucho más eficiente.

Referencias

AVG. (s.f.). *What Is Streaming and How Does It Work?* https://www.avg.com/en/signal/what-is-streaming

Cloudflare. (s.f.). *What is streaming? | How video streaming works*. https://www.cloudflare.com/es-es/learning/video/what-is-streaming/

Molla, R. (2018, junio 8). *Next year, people will spend more time online than they will watching TV. That's a first.* https://www.vox.com/2018/6/8/17441288/internet-time-spent-tv-zenith-data-media