Instituto Tecnológico y de Estudios Superiores de Monterrey

Programación Orientada a Objetos

Evidencia 1 | Proyecto integrador

Situación Problema: Modelado de servicio de streaming

Profesor: Ing. Mario Alberto Solano Saldaña

Alumno: Carlos Sánchez Gutiérrez

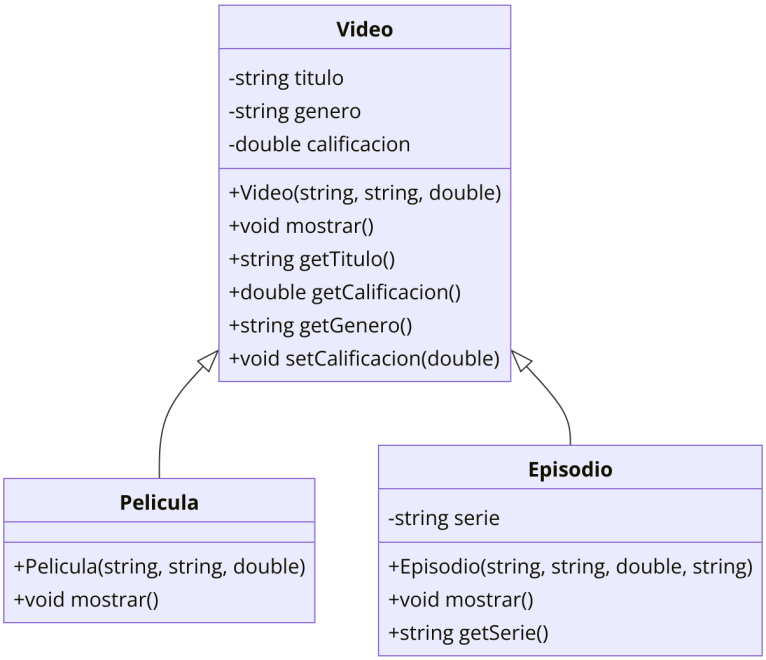
Matrícula del alumno: A01412419

Índice

Planteamiento del problema

Buscamos ofrecer una opción de un servicio de streaming, dirigido a toda la población y todo tipo de consumidores de contenido multimedia. Y es por esto, que ocupamos desarrollar un código que nos permita soportar las necesidades del proceso de solución de la situación de esta magnitud, y todas sus vertientes o cualidades que acompañan a su contexto. Con esto nos referimos a que, ocupamos utilizar la programación orientada a objetos, porque todo, prácticamente siempre es una clase, y todo siempre tiene más cosas, más especificaciones, que vamos a desarrollar, y utilizar conceptos del tema de la herencia en programación y del polimorfismo.

Diagrama UML y su argumentación



Clases y Atributos:

- **Video**: Esta es la clase base con atributos título, género y calificación. Estos atributos son protegidos (protected) y accesibles por las clases derivadas.
- **Película**: Esta clase hereda de Video y no añade nuevos atributos, pero sobrescribe el método mostrar.
- **Episodio**: Esta clase también hereda de Video y añade un nuevo atributo serie. Y sobrescribe el método mostrar y añade un método adicional get Serie.

Métodos:

- La clase Video incluye métodos para mostrar información (mostrar), obtener valores (get Titulo, get Calificación, get Género) y establecer una nueva calificación (get Calificación).
- Las clases Película y Episodio sobrescriben el método mostrar para incluir información específica de cada tipo.

Herencia:

- La relación de herencia se representa mediante flechas de la clase derivada (Película y Episodio) hacia la clase base (Video), indicando que ambas clases heredan de Video.

Argumentación del código

1. La definición de la clase base video: Facilita que a partir de esta agreguemos todo lo demás. Porque en general todo trata en sí sobre videos, sobre formatos multimedia y audiovisuales. Tratamos de llamarla o definirla de la forma o manera menos redundante posible, porque a partir de esta es de donde deriva o se expande todo lo demás.

2. Definición de las clases Película y Serie: Especifica un formato de video. Podríamos haberle agregado métodos y atributos adicionales específicos para películas, como la duración, el director, los actores etc.

3. Definición de la clase Episodio: La clase Episodio extiende Video y añade un atributo adicional para representar episodios de una serie. Con esta podemos diferenciar episodios de distintas series, y almacenar más información específica sobre cada serie. Podríamos haber agregado por ejemplo temporadas.

4. Cargar Datos desde un archivo: La función cargarDatos lee los datos de videos desde un archivo de texto (datos.txt) y crea instancias de Pelicula o Episodio. Esto facilita mucho la inicialización del programa con un conjunto predefinido de videos. El hecho de que en sí usemos ifstream permite leer datos de una mejor manera.

5. Mostrar Videos por Calificación y Género: Nos ayuda a filtrar opciones para los usuarios. Se puede agregar u omitir criterios.

6. Mostrar Episodios de una Serie por Calificación: Filtramos y mostramos todos los episodios de una serie en específico que cumpla con la calificación mínima que nos exija el usuario. Creo que al igual en mi programa y en sus comentarios mencioné,que esta me pareció una de las partes más interesantes, por el hecho de que utilizamos dynamic_cast lo cual nos permite verificar dinámicamente si un objeto Video es en realidad un Episodio. Es indispensable para no modificar la interfaz de la clase base Video.O sea, verificamos, buscamos y filtramos. Lo cual se relaciona con el polimorfismo.

7. Mostrar Películas por Calificación: Parecida al punto anterior, nos ayudará a que en un hipotético caso en el futuro, nosotros podamos agregar, o tener la posibilidad de incluir, más tipos de videos, sin la necesidad de tener que cambiar la estructura base.

8. Calificar un Video: Que el usuario pueda interactuar activamente siempre es importante. O sea, en la teoría o en POO, estamos encontrando un objeto específico dentro de un conjunto o colección por así decirlo, y modificamos sus atributos.

Identificación de casos que harían que el programa deje de funcionar

- Archivos de datos en formatos incorrectos: Si por ejemplo, de inicio, no tenemos los archivos cargados en el directorio con el que trabajamos o trabajaremos por x o y circunstancia, o de inicio está corrupto el archivo o en un formato incompatible, nuestro programa no podrá cargar los datos y reproducirse.

- Intentar cargar más archivos de los capaces de soportar nuestra plataforma (100): En mi idea, como proyecto amateur, pensé en ir cambiando el contenido, pero manteniendo 100 siempre en total al final. Pero pues, es más que nada una justificación o argumento, para no tener que emplear mucho espacio, una base de datos, o mucho contenido en general y hacer más extensos en otros casos este tipo de situaciones. Pero en general entonces, en mi programa, se puede presentar el caso de que se desborde el arreglo por así decirlo, al momento de extender la cantidad de videos de 100.

- Buscar videos que no existen: En si no es un gran error, porque puede uno simplemente poner un mensaje para el usuario e informarle que el contenido que parece que se está buscando no se encuentra disponible en nuestra plataforma.

- Entradas de Usuarios Inválidas: Poner límites, reglas, mensajes que se arrojen en caso de poner datos que de inicio no son numéricos, o no entran dentro del rango de valores aceptados. No admitir más decimales, etc.

- Que el usuario ingrese mucho texto o muchos números en la Entrada de Usuario: Hay que establecer límites en los strings por ejemplo.

- Varios usuarios entrando en la misma cuenta a la vez: Errores de sincronización, porque el programa no está pensado para manejar múltiples usuarios que estén tratando de acceder a la base de datos de forma simultánea.

Creo que, mientras más trato de buscar errores y de ver videos y pensar en qué podría salir más, voy teniendo más ideas de como debería de modificar el código en un hipotético uso real de este programa en la situación para la cual se busca en teoría implementar.

Conclusión

Creo que la programación es un área muy interesante y extremadamente importante para nuestra sociedad. Porque me puedo dar cuenta que, casi todo, sino es que todo, prácticamente está hecho en clases, en programación orientada a objetos, en capas y más capas de complejidad, con cualidades, o sea, con atributos, características, especificaciones, condiciones en donde con x o y variables se ven afectadas z, etc. Y el poder tener un paradigma más complejo y con más visión de todas las situaciones del mundo, nos hará poder ofrecer soluciones efectivas y más eficientes y de calidad, a la sociedad, en proyectos personales, en problemáticas reales, en trabajos profesionales, en empresas, negocios, etc. Creo que, sin duda alguna, una parte muy importante igual fue el pensar que podría salir mal del código, y también el hecho de tener que comprender todos los conceptos de los temas o subtemas core o principales, y cómo es que con ese grado de abstracción tenemos que aterrizar y poder hacer algo práctico.

Referencias Consultadas

(Páginas Web)

- Udemy
- Domestika
- Codigofacilito
- TodoCode Youtube
- hdelon.net Youtube
- Betta Tech Youtube