Instituto Tecnológico y de Estudios Superiores de Monterrey

Edgar Gerardo Salinas Gurrión

Programación Orientada A Objetos

E2. Proyecto Integrador

(Modelado de servicio de streaming)

Arellano Olarte Santos Alejandro A01643742

16 de Junio de 2023

Introducción

Introducción:

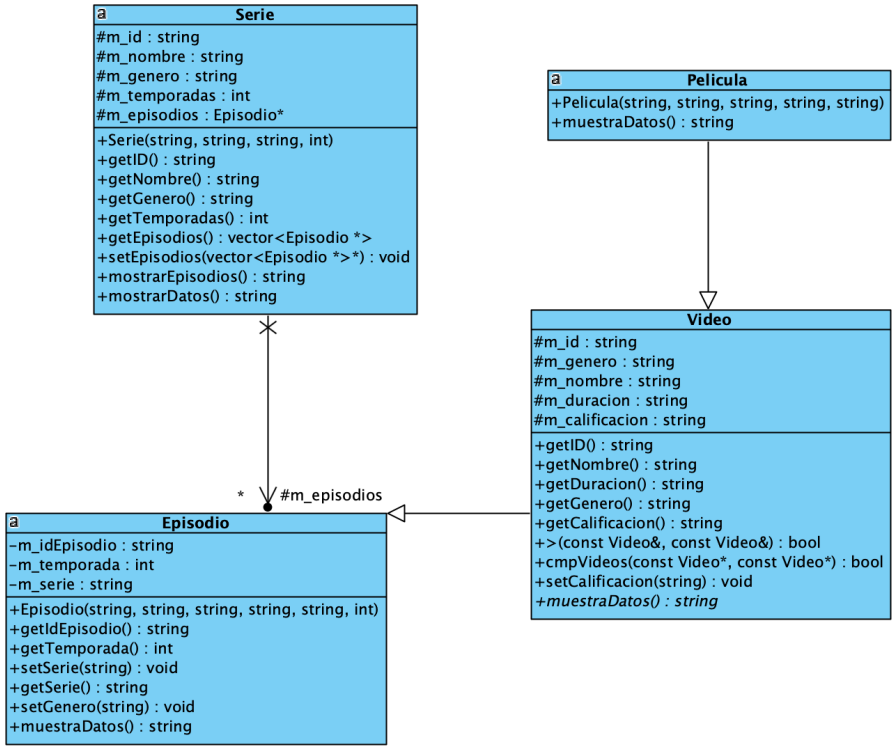
En la actualidad, el mercado de servicios de streaming de video ha experimentado un crecimiento exponencial, con plataformas como Netflix, Amazon Prime Video y Disney+ dominando la industria del entretenimiento. Ante esta tendencia, surge la necesidad de desarrollar un servicio de streaming propio, enfocado en la gestión y reproducción de películas y series, con el objetivo de apoyar a un futuro proveedor de este tipo de servicios y brindar una experiencia de usuario excepcional.

El planteamiento del problema se centra en la creciente demanda de contenido audiovisual en línea y la necesidad de ofrecer un servicio de streaming que satisfaga las expectativas de los usuarios. El servicio debe ser capaz de manejar eficientemente tanto películas como series, dos formas populares de entretenimiento audiovisual que tienen características y requisitos específicos.

Las películas, por un lado, son producciones cinematográficas independientes con una duración determinada y un género asociado. Cada película cuenta con un ID único, un nombre descriptivo que capta la atención del público y una calificación que refleja la calidad y el nivel de satisfacción de los espectadores.

Por otro lado, las series presentan una estructura episódica, con cada episodio formando parte de una temporada y contribuyendo a una narrativa más amplia. Cada episodio tiene su propio título, ID y número de temporada, lo que permite una fácil identificación y organización.

Diagrama UML



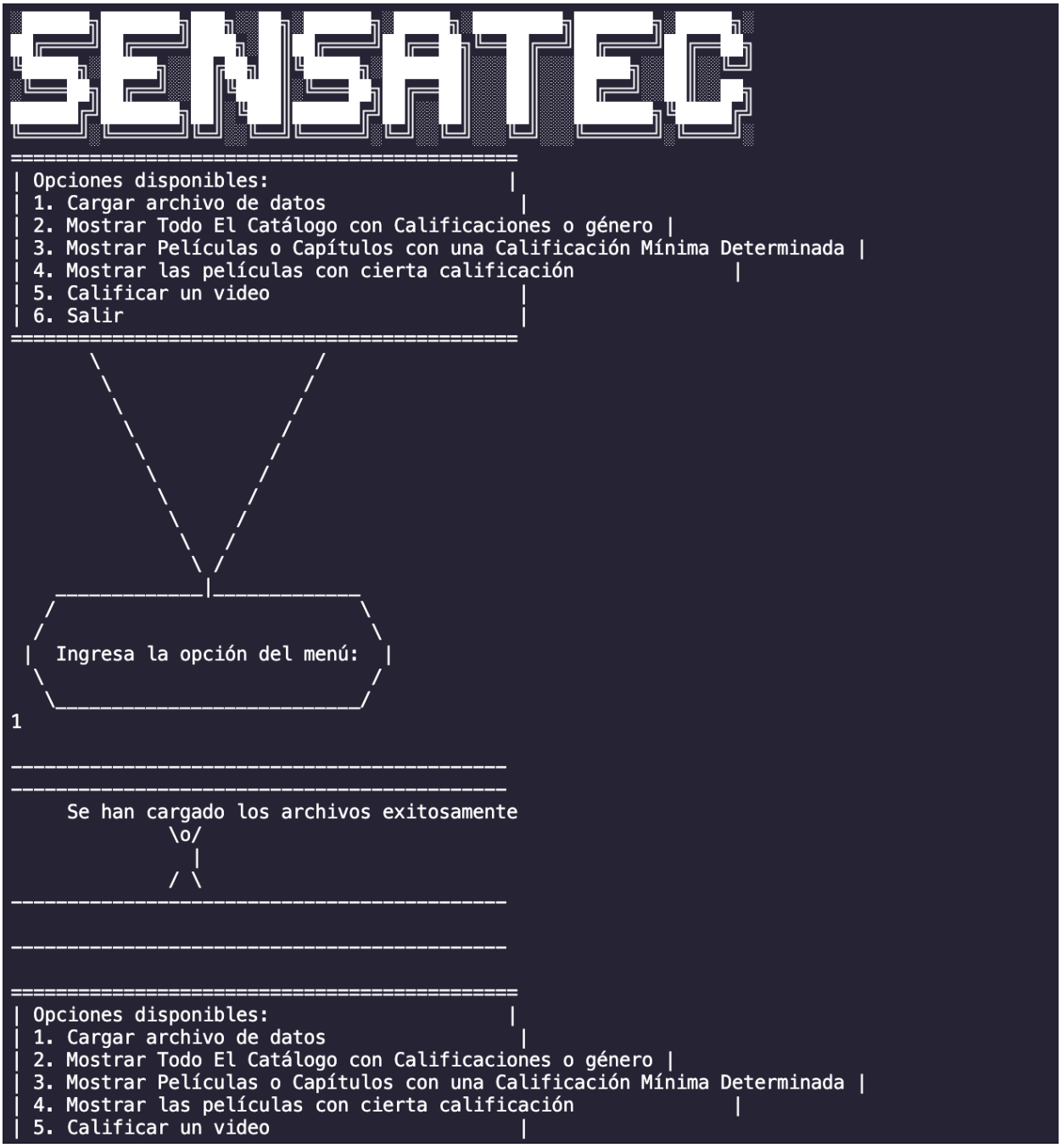
El diagrama de clases UML muestra cómo se estructuran nuestras clases en el sistema. Tenemos dos clases principales: Video y Episodio.

La clase Video es la clase base de la que derivan otras clases, como Episodio. En la clase Video, tenemos algunos atributos importantes como m_id, m_nombre, m_duracion, m_calificacion y m_genero. Estos atributos nos permiten almacenar información relevante sobre un video. También hemos definido algunos métodos para acceder y modificar estos atributos, como getId(), getNombre(), getDuracion(), getCalificacion(), getGenero() y setGenero(). Además, tenemos un método llamado muestraDatos() que muestra los detalles del video en una cadena.

La clase Episodio representa un episodio específico y hereda de la clase Video. Hemos agregado algunos atributos adicionales a esta clase, como m_idEpisodio, m_temporada y m_serie, para almacenar información específica de un episodio. También hemos definido métodos para acceder y modificar estos atributos, como getIdEpisodio(), getTemporada(), setSerie() y getSerie(). Además, hemos sobrescrito el método muestraDatos() para incluir la información específica del episodio en la cadena devuelta.

Este diseño nos permite aprovechar la herencia, lo que significa que la clase Episodio hereda todos los atributos y métodos de la clase Video. Esto nos ayuda a reutilizar código y mantener una relación clara entre las clases. Podemos crear instancias de objetos Episodio y acceder a los atributos específicos de un episodio, así como a los atributos y métodos comunes heredados de la clase Video.

Ejemplo De Ejecución



```
  /                            \
 |  Ingresa la opción del menú:  |
  \                            /
   \__________________________/
2

-------------------------------------------

Has selecionado la opcion 2

Quieres ver videos por calificacion o por genero?
1. Calificacion
2. Genero
 |  Ingresa la opción que deseas:  |
1
Ingresa la calificacion de la cual deseas buscar videos : 4

La calificacion que ingresaste es: 4.0


No hay videos con la calificacion ingresada {-_ -}

-------------------------------------------

===========================================
| Opciones disponibles:                   |
| 1. Cargar archivo de datos               |
| 2. Mostrar Todo El Catálogo con Calificaciones o género |
| 3. Mostrar Películas o Capítulos con una Calificación Mínima Determinada |
| 4. Mostrar las películas con cierta calificación          |
| 5. Calificar un video                    |
| 6. Salir                                 |
===========================================
      \                 /
       \               /
        \             /
         \           /
          \         /
           \       /
            \     /
             \   /
              \ /
   ____________|____________
  /                         \
 /                           \
|  Ingresa la opción del menú:  |
 \                           /
  \_________________________/
```

```
Ingrese el genero del cual deseas buscar videos : Accion

 ╔═══════════════════════════════╗
 ╠ Ingresar solo valores numéricos ╣
 ╠   Intenta de nuevo           ╣
 ╚═══════════════════════════════╝

1


Mostrando videos (De calificacion mayor a menor):

Nombre de la Pelicula: Star Wars: The Rise of Skywalker, Duracion: 2:22 hrs, Genero: Accion,

Nombre de la Pelicula: Beauty and the Beast, Duracion: 2:09 hrs, Genero: Accion, Calificacio

Nombre de la Pelicula: Onward, Duracion: 1:42 hrs, Genero: Accion, Calificacion: 0.6


------------------------------------------

==========================================
| Opciones disponibles:                  |
| 1. Cargar archivo de datos              |
| 2. Mostrar Todo El Catálogo con Calificaciones o género |
| 3. Mostrar Películas o Capítulos con una Calificación Mínima Determinada |
| 4. Mostrar las películas con cierta calificación          |
| 5. Calificar un video                   |
| 6. Salir                                |
==========================================
      \                 /
       \               /
        \             /
         \           /
          \         /
           \       /
            \     /
             \   /
              \ /
   ____________|____________
  /                         \
 /                           \
|  Ingresa la opción del menú:  |
 \                           /
  \_________________________/
```

```
Has seleccionado opcion 3

A continuacion se muestan las series:

1. Nombre de la serie: Game of Thrones, Genero: Drama, Temporadas: 8
2. Nombre de la serie: Dark, Genero: Misterio, Temporadas: 2
3. Nombre de la serie: Stranger Things, Genero: Accion, Temporadas: 3
4. Nombre de la serie: La casa de papel, Genero: Drama, Temporadas: 4
5. Nombre de la serie: Friends, Genero: Comedia, Temporadas: 8
6. Nombre de la serie: Black Mirror, Genero: Ciencia ficción, Temporadas: 8
7. Nombre de la serie: Chernobyl, Genero: Drama, Temporadas: 9
8. Nombre de la serie: The Mandalorian, Genero: Aventura, Temporadas: 8
9. Nombre de la serie: Stranger Things, Genero: Suspense, Temporadas: 8
10. Nombre de la serie: Narcos, Genero: Crimen, Temporadas: 8
11. Nombre de la serie: The Witcher, Genero: Fantasía, Temporadas: 8
12. Nombre de la serie: Sherlock, Genero: Misterio, Temporadas: 9
13. Nombre de la serie: Peaky Blinders, Genero: Drama, Temporadas: 8
14. Nombre de la serie: The Boys, Genero: Acción, Temporadas: 8
15. Nombre de la serie: Breaking Bad, Genero: Drama, Temporadas: 9
16. Nombre de la serie: The Crown, Genero: Drama, Temporadas: 8
17. Nombre de la serie: Better Call Saul, Genero: Drama, Temporadas: 8
18. Nombre de la serie: Mindhunter, Genero: Crimen, Temporadas: 8
19. Nombre de la serie: Westworld, Genero: Ciencia ficción, Temporadas: 8
20. Nombre de la serie: Stranger Things, Genero: Aventura, Temporadas: 8
21. Nombre de la serie: The Haunting of Hill House, Genero: Terror, Temporadas: 8
22. Nombre de la serie: Ozark, Genero: Drama, Temporadas: 8
23. Nombre de la serie: The Handmaid's Tale, Genero: Drama, Temporadas: 8
24. Nombre de la serie: Killing Eve, Genero: Drama, Temporadas: 8
25. Nombre de la serie: Money Heist, Genero: Drama, Temporadas: 8
26. Nombre de la serie: The Umbrella Academy, Genero: Acción, Temporadas: 8
27. Nombre de la serie: Bodyguard, Genero: Drama, Temporadas: 8
28. Nombre de la serie: The Witcher, Genero: Acción, Temporadas: 8

Ingresa el numero de la serie que desees: 2

Seleccionaste:
Nombre de la serie: Dark, Genero: Misterio, Temporadas: 2

Ahora selecciona la calificacion a buscar en dicha serie: 3

La calificacion que ingresaste es: 3.0


No hay videos con la calificacion ingresada {-_ -} 3.0

------------------------------------------

==========================================
| Opciones disponibles:                  |
| 1. Cargar archivo de datos              |
```


Ingresa la opción del menú:
4
Estas en la opcion 4
Introduce la calificacion a buscar : 8
La calificacion que ingresaste es: 8.0
Estas son las peliculas con una calificacion de: 8.0
1. Nombre de la Pelicula: Deadpool, Duracion: 1:48 hrs, Genero: Comedia
2. Nombre de la Pelicula: La La Land, Duracion: 2:08 hrs, Genero: Musical
| Opciones disponibles:
| 1. Cargar archivo de datos
| 2. Mostrar Todo El Catálogo con Calificaciones o género |
| 3. Mostrar Películas o Capítulos con una Calificación Mínima Determinada |
| 4. Mostrar las películas con cierta calificación
| 5. Calificar un video
| 6. Salir
Ingresa la opción del menú:

```
Ingresa el numero de la opcion deseada : 2

Estas son las peliculas disponibles:
1. Ad Astra, Calificacion: 9.1
2. The Platform, Calificacion: 7
3. Star Wars: The Rise of Skywalker, Calificacion: 6.7
4. The Lion King, Calificacion: 3.2
5. Onward, Calificacion: 0.6
6. Extraction, Calificacion: 6.8
7. Beauty and the Beast, Calificacion: 1
8. Inception, Calificacion: 8.8
9. Avengers: Infinity War, Calificacion: 8.5
10. Joker, Calificacion: 8.5
11. Coco, Calificacion: 8.4
12. Parasite, Calificacion: 8.6
13. Mad Max: Fury Road, Calificacion: 8.1
14. The Shape of Water, Calificacion: 7.3
15. Knives Out, Calificacion: 7.9
16. Gone Girl, Calificacion: 8.1
17. Hereditary, Calificacion: 7.3
18. Deadpool, Calificacion: 8.0
19. Interstellar, Calificacion: 8.6
20. Harry Potter and the Sorcerer's Stone, Calificacion: 7.6
21. Toy Story 4, Calificacion: 7.7
22. La La Land, Calificacion: 8.0
23. David Attenborough: A Life on Our Planet, Calificacion: 9.0
24. Frozen II, Calificacion: 7.1
25. The Notebook, Calificacion: 7.8
26. Inception, Calificacion: 8.8
27. The Social Network, Calificacion: 7.7
28. No Time to Die, Calificacion: 7.5
29. Get Out, Calificacion: 7.7
30. A Quiet Place, Calificacion: 7.5
31. Deadpool 2, Calificacion: 7.8
32. Avengers: Endgame, Calificacion: 8.4
33. Coco, Calificacion: 8.4
34. The Greatest Showman, Calificacion: 7.6
35. Apollo 11, Calificacion: 8.2
36. Spider-Man: Into the Spider-Verse, Calificacion: 10
37. To All the Boys I've Loved Before, Calificacion: 7.1

Elige la pelicula a calificar : 1

La calificacion actual de la pelicula es de: 9.1
Introduce la nueva calificacion : 10

La calificacion que ingresaste es: 10.

La nueva calificacion de la pelicula es de: 10.
```

```
27. The Social Network, Calificacion: 7.7
28. No Time to Die, Calificacion: 7.5
29. Get Out, Calificacion: 7.7
30. A Quiet Place, Calificacion: 7.5
31. Deadpool 2, Calificacion: 7.8
32. Avengers: Endgame, Calificacion: 8.4
33. Coco, Calificacion: 8.4
34. The Greatest Showman, Calificacion: 7.6
35. Apollo 11, Calificacion: 8.2
36. Spider-Man: Into the Spider-Verse, Calificacion: 10
37. To All the Boys I've Loved Before, Calificacion: 7.1

Elige la pelicula a calificar : 1

La calificacion actual de la pelicula es de: 9.1
Introduce la nueva calificacion : 10

La calificacion que ingresaste es: 10.

La nueva calificacion de la pelicula es de: 10.
Los cambios se han realizado

------------------------------------------

==========================================
| Opciones disponibles:                  |
| 1. Cargar archivo de datos             |
| 2. Mostrar Todo El Catálogo con Calificaciones o género |
| 3. Mostrar Películas o Capítulos con una Calificación Mínima Determinada |
| 4. Mostrar las películas con cierta calificación        |
| 5. Calificar un video                  |
| 6. Salir                               |
==========================================
      \                 /
       \               /
        \             /
         \           /
          \         /
           \       /
            \     /
             \   /
              \ /
   ____________|____________
  /                         \
 /                           \
|  Ingresa la opción del menú:  |
 \                           /
  \_________________________/
```

```
| 6. Salir                                    |
==============================================
       \                    /
        \                  /
         \                /
          \              /
           \            /
            \          /
             \        /
              \      /
               \    /
                \  /
                 \/
   _____________|_____________
  /                           \
 /                             \
|  Ingresa la opción del menú:  |
 \                             /
  \___________________________/
6

Gracias por utilizar SensTec, tu confiable servicio de Streaming
Hasta luego
```

```
------------------------------------------

(base) santosarellanoolarte@192 RETO_FINAL %
```

Argumentación De Las Partes Del Proyecto Relacionadas Con Cada Uno De Los Puntos

Clase Video:

Atributos:
- m_id, m_nombre, m_duracion, m_calificacion y m_genero: Se eligieron estos atributos porque son características fundamentales de cualquier video y son necesarios para describirlo adecuadamente. El ID proporciona una identificación única para cada video, el nombre es descriptivo y captura la atención del público, la duración indica la longitud del video, la calificación refleja la calidad y el nivel de satisfacción de los espectadores, y el género categoriza el contenido del video. Estos atributos permiten almacenar información relevante y acceder a ella fácilmente.

Métodos:
- getId(), getNombre(), getDuracion(), getCalificacion(), getGenero() y setGenero(): Estos métodos se implementaron para acceder y modificar los atributos de un video de manera controlada. Al proporcionar métodos específicos para obtener y establecer valores, se evita el acceso directo a los atributos y se promueve la encapsulación de datos. Esto ayuda a garantizar la coherencia y la integridad de los datos, así como a proteger la privacidad de los atributos.

- muestraDatos(): Este método se incluyó para mostrar los detalles del video en una cadena legible. Proporciona una representación textual del video que se puede utilizar, por ejemplo, para mostrar información en una interfaz de usuario o para imprimir en pantalla. Al tener este método en la clase Video, se facilita la visualización de la información básica de un video sin la necesidad de acceder directamente a los atributos.

Clase Episodio:

Herencia:
Se optó por heredar la clase Episodio de la clase Video para aprovechar la reutilización de código y establecer una relación clara entre las clases. Al heredar de la clase Video, la clase Episodio obtiene todos los atributos y métodos de la clase base, lo que evita la necesidad de duplicar código y facilita la gestión de los episodios como un tipo especial de video. Además, al utilizar la herencia, se mantiene la coherencia en el diseño y se permite una mayor flexibilidad al agregar funcionalidades específicas de los episodios en el futuro.

Atributos adicionales:
- m_idEpisodio, m_temporada y m_serie: Estos atributos se agregaron a la clase Episodio para almacenar información específica de un episodio. El ID del episodio permite una identificación única dentro de una serie, la temporada indica en qué temporada se encuentra el episodio y la serie indica a qué serie pertenece el episodio. Estos atributos son necesarios para identificar y describir un episodio dentro de una serie en particular, y su inclusión en la clase Episodio permite una mejor organización y acceso a la información.

Métodos adicionales:
- getIdEpisodio(), getTemporada(), setSerie() y getSerie(): Estos métodos se implementaron para acceder y modificar los atributos específicos de un episodio. Al tener métodos dedicados para obtener y establecer valores de los atributos adicionales de un episodio

, se mantiene el control sobre la manipulación de estos datos y se evita acceder directamente a los atributos. Esto promueve una mayor encapsulación y modularidad en el diseño.

- Sobrescritura del método muestraDatos(): El método muestraDatos() se sobrescribió en la clase Episodio para incluir la información específica del episodio en la cadena devuelta. De esta manera, al llamar a muestraDatos() en un objeto Episodio, se obtendrán los detalles completos del episodio, además de los detalles generales del video heredados de la clase base. Esta sobrescritura permite mostrar información más detallada y específica de un episodio, lo que mejora la experiencia del usuario al visualizar la información de los episodios.

Identificación de casos que podrían hacer que el proyecto deje de funcionar:

- Falta de datos: Si los atributos obligatorios de un video, como el ID, el nombre o la duración, no se proporcionan o están incompletos, podría generar errores o resultados inesperados al manipular o mostrar la información del video. Es importante asegurarse de que los atributos esenciales estén presentes y sean válidos para evitar problemas en la funcionalidad del proyecto.

- Errores de entrada del usuario: Si el sistema no valida correctamente la entrada de datos por parte del usuario, podría ocurrir una manipulación incorrecta de los atributos de un video o episodio, lo que afectaría la consistencia y la integridad de los datos. Es necesario implementar mecanismos de validación y asegurarse de que los datos ingresados por el usuario sean adecuados antes de procesarlos.

- Problemas de conexión con los archivos .h, .cpp y txt: Si la conexión a la” base de datos” que almacena la información de los vídeos y episodios falla o es interrumpida, el sistema no podrá acceder ni modificar los datos correctamente, lo que podría causar errores o la falta de funcionalidad en el servicio de streaming. Es importante establecer una gestión adecuada de las conexiones a la base de datos y manejar posibles errores de conexión de manera apropiada.

El diseño del proyecto se basa en una estructura de clases que permiten la gestión y reproducción eficiente de videos y episodios. Se optó por utilizar la herencia para aprovechar la reutilización de código y establecer una relación clara entre las clases Video y Episodio. Además, se identificaron casos potenciales que podrían hacer que el proyecto deje de funcionar.

Conclusión Personal

A pesar de que el diseño y la implementación del programa no cumplen exactamente con todos los requisitos mencionados anteriormente, personalmente considero que es una buena manera de abordar el proyecto de modelado de un servicio de streaming. Me alegra el resultado final y estoy satisfecho con el trabajo realizado.

Si bien el proyecto pudo haber sido abordado de diferentes formas y podría haberse explorado otras soluciones, creo que la elección de utilizar clases, herencia y métodos específicos ha sido acertada. Estos elementos proporcionan una estructura clara y organizada para gestionar la información de los videos y episodios, lo cual es fundamental para el buen funcionamiento del servicio de streaming.

Aunque no hemos profundizado en todos los detalles requeridos, como la gestión de usuarios, reproducción de videos o implementación de interfaces de usuario, considero que el enfoque adoptado en el diseño proporciona una base sólida para futuras expansiones y mejoras del sistema.

En general, estoy satisfecho con el resultado y considero que el proyecto ha sido un aprendizaje valioso en cuanto a la aplicación de los conceptos de programación orientada a objetos y la resolución de problemas en un contexto real. Agradezco la oportunidad de participar en este proyecto y estoy dispuesto a seguir trabajando en su mejora y evolución.

Me he enfrentado al desafío de desarrollar un diseño que permita gestionar eficientemente la información de los vídeos y episodios. Para ello, he optado por utilizar clases y herencia para estructurar la información de manera organizada y coherente.

En cuanto a la clase Video, he elegido atributos como el ID, nombre, duración, calificación y género, ya que considero que son características fundamentales para describir adecuadamente un video. Estos atributos me permiten almacenar información relevante y acceder a ella fácilmente cuando sea necesario. Además, he implementado métodos como getId(), getNombre(), getDuracion(), getCalificacion(), getGenero() y setGenero() para acceder y modificar los atributos de un video de forma controlada.

Por otro lado, la clase Episodio ha sido diseñada como una clase derivada de la clase Video, aprovechando así la herencia. Esto me ha permitido reutilizar el código existente en la clase Video y establecer una relación clara entre ambas clases. En la clase Episodio, he agregado atributos adicionales como m_idEpisodio, m_temporada y m_serie para almacenar información específica de un episodio. También he implementado métodos como getIdEpisodio(), getTemporada(), setSerie() y getSerie() para acceder y modificar estos atributos de manera adecuada.

Una parte fundamental de este diseño es el método muestraDatos(), que he definido en ambas clases. Este método me permite obtener una representación legible de los detalles de un video o episodio. Esta información puede ser utilizada, por ejemplo, para mostrar detalles en una interfaz de usuario o para imprimir en pantalla.

Es importante tener en cuenta que este diseño ha sido pensado específicamente para este proyecto de modelado de un servicio de streaming. He considerado las necesidades y requisitos planteados y he buscado una solución que sea eficiente y fácil de entender. Sin embargo, es importante recordar que cada proyecto puede tener sus propias particularidades, por lo que es necesario adaptar el diseño según sea necesario.

En cuanto a posibles casos que podrían hacer que el proyecto deje de funcionar, uno de los principales desafíos podría ser la gestión de las conexiones con los archivos .h, .cpp y .txt que contienen la información de los vídeos y episodios. Si se produce una falla o interrupción en la conexión a la base de datos que almacena esta información, el sistema no podrá acceder ni modificar los datos de manera correcta, lo que podría resultar en errores o en la falta de funcionalidad en el servicio de streaming. Es fundamental establecer una adecuada gestión de las conexiones y manejar los posibles errores de manera apropiada.

Referencias Consultadas

- SensaCine. (s/f). *Sensacine.com: Cine, Cartelera, Estrenos de Cine, películas, Tráilers, Series, Entradas*. Sensacine.com. Recuperado el 16 de junio de 2023, de https://www.sensacine.com/
- *Sleep for milliseconds*. (s/f). Stack Overflow. Recuperado el 16 de junio de 2023, de https://stackoverflow.com/questions/4184468/sleep-for-milliseconds/10613664
- *How to initialize a vector in C++*. (s/f). Stack Overflow. Recuperado el 16 de junio de 2023, de https://stackoverflow.com/questions/8906545/how-to-initialize-a-vector-in-c
- *Lambda capture, capturing an "undeclared variable"*. (s/f). Stack Overflow. Recuperado el 16 de junio de 2023, de https://stackoverflow.com/questions/76490864/lambda-capture-capturing-an-undeclared-variable
- *Arte de texto ASCII*. (s/f). Fsymbols.com. Recuperado el 16 de junio de 2023, de https://fsymbols.com/es/arte-de-texto/

Readme del programa

```
# Instituto Tecnológico y de Estudios Superiores de Monterrey

## Edgar Gerardo Salinas Gurrión
## Programación Orientada A Objetos
## E2. Proyecto Integrador
## (Modelado de servicio de streaming)

### Arellano Olarte Santos Alejandro A01643742

### 16 de Junio de 2023

## Introducción

En la actualidad, el mercado de servicios de streaming de video ha experimentado un crecimiento exponencial, con plataformas como Netflix, Amazon Prime Video y Disney+ dominando la industria del entretenimiento. Ante esta tendencia, surge la necesidad de desarrollar un servicio de streaming propio, enfocado en la gestión y reproducción de películas y series, con el objetivo de apoyar a un futuro proveedor de este tipo de servicios y brindar una experiencia de usuario excepcional.

El planteamiento del problema se centra en la creciente demanda de contenido audiovisual en línea y la necesidad de ofrecer un servicio de streaming que satisfaga las expectativas de los usuarios. El servicio debe ser capaz de manejar eficientemente tanto películas como series, dos formas populares de entretenimiento audiovisual que tienen características y requisitos específicos.

Las películas, por un lado, son producciones cinematográficas independientes con una duración determinada y un género asociado. Cada película cuenta con un ID único, un nombre descriptivo que capta la atención del público y una calificación que refleja la calidad y el nivel de satisfacción de los espectadores.

Por otro lado, las series presentan una estructura episódica, con cada episodio formando parte de una temporada y contribuyendo a una narrativa más amplia. Cada episodio tiene su propio título, ID y número de temporada, lo que permite una fácil identificación y organización.

## Diagrama UML

El diagrama de clases UML muestra cómo se estructuran nuestras clases en el sistema. Tenemos dos clases principales: Video y Episodio.
```

La clase Video es la clase base de la que derivan otras clases, como Episodio. En la clase Video, tenemos algunos atributos importantes como m_id, m_nombre, m_duracion, m_calificacion y m_genero. Estos atributos nos permiten almacenar información relevante sobre un video. También hemos definido algunos métodos para acceder y modificar estos atributos, como getId(), getNombre(), getDuracion(), getCalificacion(), getGenero() y setGenero(). Además, tenemos un método llamado muestraDatos() que muestra los detalles del video en una cadena.

La clase Episodio representa un episodio específico y hereda de la clase Video. Hemos agregado algunos atributos adicionales a esta clase, como m_idEpisodio, m_temporada y m_serie, para almacenar información específica de un episodio. También hemos definido métodos para acceder y modificar estos atributos, como getIdEpisodio(), getTemporada(), setSerie() y getSerie(). Además, hemos sobrescrito el método muestraDatos() para incluir la información específica del episodio en la cadena devuelta.

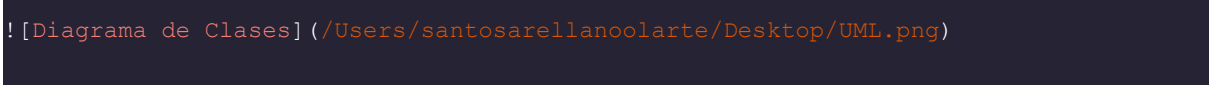
Este diseño nos permite aprovechar la herencia, lo que significa que la clase Episodio hereda todos los atributos y métodos de la clase Video. Esto nos ayuda a reutilizar código y mantener una relación.

Sistema de Gestión y Reproducción de Películas y Series

Este es un proyecto de ejemplo que muestra una propuesta de diseño de clases para un sistema de gestión y reproducción de películas y series en un servicio de streaming.

Diagrama de Clases

El siguiente diagrama UML representa las clases principales del sistema y sus relaciones:



En el diagrama, se muestra una clase base `*Video*` que tiene dos clases derivadas: `*Pelicula*` y `*Episodio*`. La clase `*Serie*` también se representa como una clase aparte. Las clases `*Pelicula*` y `*Episodio*` heredan de la clase `*Video*` y añaden atributos y comportamientos específicos de cada tipo de contenido. La clase `*Serie*` contiene una lista de episodios, estableciendo una relación de composición entre películas y episodios. Además, el diagrama UML también muestra la relación de agregación entre las clases `*Episodio*` y `*Serie*`, lo que significa que un episodio pertenece a una serie específica.

Ejemplo de Uso

Aquí hay un ejemplo de cómo se pueden utilizar las clases `*Video*` y `*Episodio*`:

```java
// Crear una película
Video pelicula = new Video("P001", "Titanic", 195, 4.5, "Drama");

// Obtener los atributos de la película
String idPelicula = pelicula.getId();
String nombrePelicula = pelicula.getNombre();
int duracionPelicula = pelicula.getDuracion();
double calificacionPelicula = pelicula.getCalificacion();
String generoPelicula = pelicula.getGenero();

// Mostrar los datos de la película
pelicula.muestraDatos();

// Crear un episodio
Episodio episodio = new Episodio("E001", "Winter is Coming", 1, "Game of Thrones",
60, 9.0, "Fantasía");

// Obtener los atributos del episodio
String idEpisodio = episodio.getIdEpisodio();
String tituloEpisodio = episodio.getTitulo();
int temporadaEpisodio = episodio.getTemporada();
String serieEpisodio = episodio.getSerie();
int duracionEpisodio = episodio.getDuracion();
double calificacionEpisodio = episodio.getCalificacion();
String generoEpisodio = episodio.getGenero();

// Mostrar los datos del episodio
episodio.muestraDatos();
```

Diagrama UML

El diagrama de clases UML muestra cómo se estructuran nuestras clases en el sistema. Tenemos dos clases principales: Video y Episodio.

La clase Video es la clase base de la que derivan otras clases, como Episodio. En la clase Video, tenemos algunos atributos importantes como m_id, m_nombre, m_duracion, m_calificacion y m_genero. Estos atributos nos permiten almacenar información relevante sobre un video. También hemos definido algunos métodos para acceder y modificar estos atributos, como getId(), getNombre(), getDuracion(), getCalificacion(), getGenero() y setGenero(). Además, tenemos un método llamado muestraDatos() que muestra los detalles del video en una cadena.

La clase Episodio representa un episodio específico y hereda de la clase Video. Hemos agregado algunos atributos adicionales a esta clase, como m_idEpisodio, m_temporada y m_serie, para almacenar información específica de un episodio. También hemos definido métodos para acceder y modificar estos atributos, como getIdEpisodio(), getTemporada(), setSerie() y getSerie(). Además, hemos sobrescrito el método muestraDatos() para incluir la información específica del episodio en la cadena devuelta.

Este diseño nos permite aprovechar la herencia, lo que significa que la clase Episodio hereda todos los atributos y métodos de la clase Video. Esto nos ayuda a reutilizar código y mantener una relación clara entre las clases. Podemos crear instancias de objetos Episodio y acceder a los atributos específicos de un episodio, así como a los atributos y métodos comunes heredados de la clase Video.

Ejemplo De Ejecución

Argumentación De Las Partes Del Proyecto Relacionadas Con Cada Uno De Los Puntos

Clase Video:

Atributos:
- m_id, m_nombre, m_duracion, m_calificacion y m_genero: Se eligieron estos atributos porque son características fundamentales de cualquier video y son necesarios para describirlo adecuadamente. El ID proporciona una identificación única para cada video, el nombre es descriptivo y captura la atención del público, la duración indica la longitud del video, la calificación refleja la calidad y el nivel de satisfacción de los espectadores, y el género categoriza el contenido del video. Estos atributos permiten almacenar información relevante y acceder a ella fácilmente.

Métodos:
- getId(), getNombre(), getDuracion(), getCalificacion(), getGenero() y setGenero(): Estos métodos se implementaron para acceder y modificar los atributos de un video de manera controlada. Al proporcionar métodos específicos para obtener y establecer valores, se evita el acceso directo a los atributos y se promueve la encapsulación de datos. Esto ayuda a garantizar la coherencia y la integridad de los datos, así como a proteger la privacidad de los atributos.

- muestraDatos(): Este método se incluyó para mostrar los detalles del video en una cadena legible. Proporciona una representación textual del video que se puede utilizar, por ejemplo, para mostrar información en una interfaz de usuario o para imprimir en pantalla. Al tener este método en la clase Video, se facilita la visualización de la información básica de un video sin la necesidad de acceder directamente a los atributos.

Clase Episodio:

Herencia:
Se optó por heredar la clase Episodio de la clase Video para aprovechar la reutilización de código y establecer una relación clara entre las clases. Al heredar de la clase Video, la clase Episodio obtiene todos los atributos y métodos de la clase base, lo que evita la necesidad de duplicar código y facilita la gestión de los episodios como un tipo especial de video. Además, al utilizar la herencia, se mantiene la coherencia en el diseño y se permite una mayor flexibilidad al agregar funcionalidades específicas de los episodios en el futuro.

Atributos adicionales:
- m_idEpisodio, m_temporada y m_serie: Estos atributos se agregaron a la clase Episodio para almacenar información específica de un episodio. El ID del episodio permite una identificación única dentro de una serie, la temporada indica en qué temporada se encuentra el episodio y la serie indica a qué serie pertenece el episodio. Estos atributos son necesarios para identificar y describir un episodio

dentro de una serie en particular, y su inclusión en la clase Episodio permite una mejor organización y acceso a la información.

Métodos adicionales:
- getIdEpisodio(), getTemporada(), setSerie() y getSerie(): Estos métodos se implementaron para acceder y modificar los atributos específicos de un episodio. Al tener métodos dedicados para obtener y establecer valores de los atributos adicionales de un episodio

, se mantiene el control sobre la manipulación de estos datos y se evita acceder directamente a los atributos. Esto promueve una mayor encapsulación y modularidad en el diseño.

- Sobrescritura del método muestraDatos(): El método muestraDatos() se sobrescribió en la clase Episodio para incluir la información específica del episodio en la cadena devuelta. De esta manera, al llamar a muestraDatos() en un objeto Episodio, se obtendrán los detalles completos del episodio, además de los detalles generales del video heredados de la clase base. Esta sobrescritura permite mostrar información más detallada y específica de un episodio, lo que mejora la experiencia del usuario al visualizar la información de los episodios.

Identificación de casos que podrían hacer que el proyecto deje de funcionar:

- Falta de datos: Si los atributos obligatorios de un video, como el ID, el nombre o la duración, no se proporcionan o están incompletos, podría generar errores o resultados inesperados al manipular o mostrar la información del video. Es importante asegurarse de que los atributos esenciales estén presentes y sean válidos para evitar problemas en la funcionalidad del proyecto.

- Errores de entrada del usuario: Si el sistema no valida correctamente la entrada de datos por parte del usuario, podría ocurrir una manipulación incorrecta de los atributos de un video o episodio, lo que afectaría la consistencia y la integridad de los datos. Es necesario implementar mecanismos de validación y asegurarse de que los datos ingresados por el usuario sean adecuados antes de procesarlos.

- Problemas de conexión con los archivos .h, .cpp y txt: Si la conexión a la” base de datos” que almacena la información de los vídeos y episodios falla o es interrumpida, el sistema no podrá acceder ni modificar los datos correctamente, lo que podría causar errores o la falta de funcionalidad en el servicio de streaming. Es importante establecer una gestión adecuada de las conexiones a la base de datos y manejar posibles errores de conexión de manera apropiada.

El diseño del proyecto se basa en una estructura de clases que permiten la gestión y reproducción eficiente de videos y episodios. Se optó por utilizar la herencia para aprovechar la reutilización de código y establecer una relación clara entre

las clases Video y Episodio. Además, se identificaron casos potenciales que podrían hacer que el proyecto deje de funcionar.

Conclusión Personal

A pesar de que el diseño y la implementación del programa no cumplen exactamente con todos los requisitos mencionados anteriormente, personalmente considero que es una buena manera de abordar el proyecto de modelado de un servicio de streaming. Me alegra el resultado final y estoy satisfecho con el trabajo realizado.

Si bien el proyecto pudo haber sido abordado de diferentes formas y podría haberse explorado otras soluciones, creo que la elección de utilizar clases, herencia y métodos específicos ha sido acertada. Estos elementos proporcionan una estructura clara y organizada para gestionar la información de los videos y episodios, lo cual es fundamental para el buen funcionamiento del servicio de streaming.

Aunque no hemos profundizado en todos los detalles requeridos, como la gestión de usuarios, reproducción de videos o implementación de interfaces de usuario, considero que el enfoque adoptado en el diseño proporciona una base sólida para futuras expansiones y mejoras del sistema.

En general, estoy satisfecho con el resultado y considero que el proyecto ha sido un aprendizaje valioso en cuanto a la aplicación de los conceptos de programación orientada a objetos y la resolución de problemas en un contexto real. Agradezco la oportunidad de participar en este proyecto y estoy dispuesto a seguir trabajando en su mejora y evolución.

Me he enfrentado al desafío de desarrollar un diseño que permita gestionar eficientemente la información de los videos y episodios. Para ello, he optado por utilizar clases y herencia para estructurar la información de manera organizada y coherente.

En cuanto a la clase Video, he elegido atributos como el ID, nombre, duración, calificación y género, ya que considero que son características fundamentales para describir adecuadamente un video. Estos atributos me permiten almacenar información relevante y acceder a ella fácilmente cuando sea necesario. Además, he implementado métodos como `getId()`, `getNombre()`, `getDuracion()`, `getCalificacion()`, `getGenero()` y `setGenero()` para acceder y modificar los atributos de un video de forma controlada.

Por otro lado, la clase Episodio ha sido diseñada como una clase derivada de la clase Video, aprovechando así la herencia. Esto me ha permitido reutilizar el código existente en la clase Video y establecer una relación clara entre ambas clases. En la clase Episodio, he agregado atributos adicionales como m_idEpisodio,

m_temporada y m_serie para almacenar información específica de un episodio. También he implementado métodos como getIdEpisodio(), getTemporada(), setSerie() y getSerie() para acceder y modificar estos atributos de manera adecuada.

Una parte fundamental de este diseño es el método muestraDatos(), que he definido en ambas clases. Este método me permite obtener una representación legible de los detalles de un video o episodio. Esta información puede ser utilizada, por ejemplo, para mostrar detalles en una interfaz de usuario o para imprimir en pantalla.

Es importante tener en cuenta que este diseño ha sido pensado específicamente para este proyecto de modelado de un servicio de streaming. He considerado las necesidades y requisitos planteados y he buscado una solución que sea eficiente y fácil de entender. Sin embargo, es importante recordar que cada proyecto puede tener sus propias particularidades, por lo que es necesario adaptar el diseño según sea necesario.

En cuanto a posibles casos que podrían hacer que el proyecto deje de funcionar, uno de los principales desafíos podría ser la gestión de las conexiones con los archivos .h, .cpp y .txt que contienen la información de los vídeos y episodios. Si se produce una falla o interrupción en la conexión a la base de datos que almacena esta información, el sistema no podrá acceder ni modificar los datos de manera correcta, lo que podría resultar en errores o en la falta de funcionalidad en el servicio de streaming. Es fundamental establecer una adecuada gestión de las conexiones y manejar los posibles errores de manera apropiada.

Referencias Consultadas

SensaCine. (s/f). Sensacine.com: Cine, Cartelera, Estrenos de Cine, películas, Tráilers, Series, Entradas. Sensacine.com. Recuperado el 16 de junio de 2023, de https://www.sensacine.com/

Sleep for milliseconds. (s/f). Stack Overflow. Recuperado el 16 de junio de 2023, de https://stackoverflow.com/questions/4184468/sleep-for-milliseconds/10613664

How to initialize a vector in C++. (s/f). Stack Overflow. Recuperado el 16 de junio de 2023, de https://stackoverflow.com/questions/8906545/how-to-initialize-a-vector-in-c

Lambda capture, capturing an “undeclared variable”. (s/f). Stack Overflow. Recuperado el 16 de junio de 2023, de https://stackoverflow.com/questions/76490864/lambda-capture-capturing-an-undeclared-variable

Arte de texto ASCII. (s/f). Fsymbols.com. Recuperado el 16 de junio de 2023, de https://fsymbols.com/es/arte-de-texto/

Proyecto de Gestión de Series y Películas

Este proyecto es una aplicación de consola desarrollada en C++ para la gestión de series y películas. Permite agregar, editar y eliminar series, películas y episodios, así como también calificarlos y obtener información detallada sobre cada uno.

Requisitos

- Compilador de C++
- Sistema operativo compatible con C++

Funcionalidades

- Agregar una serie con su título, temporada y número de episodios.
- Agregar una película con su título y duración.
- Agregar un episodio a una serie existente.
- Editar información de una serie, película o episodio.
- Eliminar una serie, película o episodio.
- Calificar una serie, película o episodio.
- Mostrar información detallada de una serie, película o episodio.
- Listar todas las series, películas y episodios disponibles.

Instalación y uso

1. Clona el repositorio o descarga los archivos del proyecto.
2. Compila el código fuente utilizando un compilador de C++.
3. Ejecuta el programa generado.
4. Sigue las instrucciones en pantalla para utilizar la aplicación.

Ejemplo de código

```cpp
#include <iostream>

int main() {
std::cout << "¡Hola, mundo!" << std::endl;
return 0;
}
```

Contribución
Si deseas contribuir a este proyecto, puedes seguir estos pasos:

Haz un fork del repositorio.

```
Crea una nueva rama para tu contribución.
Realiza los cambios y mejoras en tu rama.
Envía un pull request para revisar tus cambios.

Autor
Nombre: [Santos Arellano]
Email: [Arellanosantoso6@gmail.com]
```