Tecnológico de Monterrey

Proyecto Servicio de Streaming

Jorge Alejandro González Díaz

A00344893

Programación Orientada a Objetos

Grupo 319

M.C. Ana Karina Ríos Araujo

Tecnológico de Monterrey Campus Guadalajara

Proyecto - Servicio de calificación de videos

Descripción del proyecto

El programa realizado consiste en un servicio de calificación de películas y series con funcionalidades implementadas para buscar la facilidad de uso del sistema y la interactividad con el usuario. Las principales implementaciones con las que el usuario puede interactuar con el sistema se rigen por un menú que permite al usuario seleccionar acciones intuitivamente.

```
std::cout << "\nMENU DEL PROGRAMA" << std::endl;
std::cout << "-----------------------------------------------------" << std::endl;
std::cout << "OPCION 1 = Imprimir catalogo de videos" << std::endl;
std::cout << "OPCION 2 = Calificar un video" << std::endl;
std::cout << "OPCION 3 = Imprimir videos por genero" << std::endl;
std::cout << "OPCION 4 = Ver mejores calificados" << std::endl;
std::cout << "OPCION 5 = Mostrar informacion de video especifico" << std::endl;
std::cout << "OPCION 6 = Evaluar episodios de una serie" << std::endl;
std::cout << "OPCION 7 = Salir" << std::endl;
std::cout << "Introduce numero de opcion: ";
```

La implementación de este código y su estructuración fue posible a través de la organización en clases de los diferentes tipos de videos y el desarrollo de herencia para el reuso y la conservación de orden en el código.

Archivos

A continuación se muestran los archivos del programa con sus respectivos contenidos.

Video.hpp (declaración de clase Video)

```
#ifndef VIDEO_H
#define VIDEO_H
#include <iostream>
#include <vector>

class Video{
    public:
      Video(); //Constructor default
      Video(std::string, char, int, int);
```

```
      void mostrarInfo();   //Mostrar la informacion de un video
        void calificar();       //Funcion para que el usuario introduzca su reseña
       void updateRating(int calificacion);   //Funcion para actualizar el promedio de calificaciones
       bool friend operator < (Video vid1, Video vid2); //Sobrecarga de operadores para ver cual tiene mejor rating
      //Getters
      std::string getTitulo();
      char getCategoria();
      float getRating();
    protected:
      std::string titulo;   //Titulo del video
        char genero;              //Clasificacion de la pelicula
      int duracion;          //Duracion en minutos
      int fecha;             //Fecha de lanzamiento
               float  calificacionAvg;  //Promedio de calificiaciones
         std::vector<int> calificaciones; //Vector de calificaciones
};

#endif //VIDEO_H
```

Video.cpp (implementación de clase video)

```
//Archivo para implementacion de clase Video
#include <iostream>
#include "../.include/Video.hpp"

Video::Video(){}
```

```
Video::Video(std::string title, char genre, int duracion_min, int estreno){
       titulo = title;   //Asignacion de atributos en inicializacion
    genero = genre;
    duracion = duracion_min;
    fecha = estreno;
       std::cout << "Video " << titulo << " creado efectivamente" << std::endl;
}

std::string detectarCategoria(char cat){   //Funcion usada en mostrarInfo para imprimir nombre completo de genero
    int index;
       std::string generos[] = {"Accion", "Comedia", "Drama", "Terror/suspenso", "Infantil"};
    char charsCategorias[] = {'a', 'c', 'd', 't', 'i'};
            int arraySize = sizeof(generos) / sizeof(generos[0]); //Calcular el numero de elementos en el array
    for(int i=0;i<arraySize;i++){
        if(cat == charsCategorias[i]){
              index = i;    //Encontrar el indice de la categoria (en char) para traducir a string
            break;
        }
    }
    return generos[index];  //Mapeo a string
}

void Video::mostrarInfo(){   //Funcion para imprimir la informacion relacionada a un video
```

```
        std::cout << "\nINFORMACION DEL VIDEO" <<
std::endl;
                                    std::cout        <<
"-------------------------------------------------
---------" << std::endl;
    std::cout << "TITULO: " << titulo << std::endl;
                std::cout    <<    "GENERO:    "    <<
detectarCategoria(genero) << std::endl;
    std::cout << "DURACION: " << duracion << std::endl;
       std::cout << "FECHA ESTRENO: " << fecha <<
std::endl;
        std::cout << "CALIFICACION PROMEDIO: " <<
calificacionAvg << std::endl;
                                    std::cout        <<
"-------------------------------------------------
---------\n" << std::endl;
}

void Video::calificar(){ //Funcion para que usuario
introduzca calificacion de un video
    int calificacion = 0;
    std::cout << "\nCALIFICAR VIDEO" << std::endl;
                                    std::cout        <<
"-------------------------------------------------
---------" << std::endl;
    std::cout << titulo << std::endl;
     std::cout << "Introduzca calificacion para el video
-> ";
    std::cin >> calificacion;
       while(calificacion < 1 || calificacion > 5){
//Mientras entrada no sea valida
        std::cout << "Numero no valido, introducir del
1-5" << std::endl;
        std::cin >> calificacion;
```

```
    }
        updateRating(calificacion);    //Guardar nueva
calificacion y actualizar promedio de ratings
       std::cout << "Rating " << calificacion << "
guardado exitosamente." << std::endl;
       std::cout << "Numero de calificaciones: " <<
calificaciones.size() << std::endl;
       std::cout << "Rating actual del video: " <<
calificacionAvg << std::endl;
}

//Llamar cada vez que se agrega una calificacion
void Video::updateRating(int calificacion){ //Funcion
para calcular el promedio de ratings actualizado
                calificaciones.push_back(calificacion);
//Actualizar el vector de calificaciones
    int total = 0;
    int counter = 0;
    for(int num:calificaciones){
        total += num; //Sumar calificaciones
        counter++;
    }
         calificacionAvg = float(total) / counter;
//Obtencion del rating promedio
}

std::string Video::getTitulo(){
    return titulo;
}

float Video::getRating(){
    return calificacionAvg;
}
```

```
char Video::getCategoria(){
    return genero; //Utilizamos el atributo de la clase
para pasar como argumento a la funcion local
}

bool operator < (Video vid1, Video vid2){
    if(vid1.getRating() < vid2.getRating()){
        return true;
    }else{
        return false;
    }
}
```

Pelicula.hpp (declaración de clase película)

```
#ifndef PELICULA_H
#define PELICULA_H
#include <iostream>
#include "./Video.hpp"

class Pelicula : public Video{
    public:
        Pelicula(std::string, char, int, int);
//Constructor de la pelicula
};
#endif //PELICULA_H
```

Pelicula.cpp (implementación de clase Película)

```
#include <iostream>
#include "../.include/Pelicula.hpp"

Pelicula::Pelicula(std::string title, char genre, int
minutos, int estreno){
```

```
    titulo = title;
    genero = genre;
    duracion = minutos;
    fecha = estreno;
}
```

Serie.hpp (declaración de clase Serie)

```
#ifndef SERIE_H
#define SERIE_H
#include <iostream>
#include <vector>
#include "./Episodio.hpp"

class Serie{
    public:
      Serie(std::string title, char genre, int temps,
int num_ep, std::vector<Episodio> episodios);
      void mostrarInfo(); //Mostrar la info de la serie
      void rate();        //Calificar la serie
(general) con input del usuario
      void updateRate(int calificacion);  //Funcion que
se llama dentro de rate() para actualizar vector
ratings
      void calificarEps(int); //Para calificar los
episodios en el vector
      bool friend operator < (Serie, Serie); //Comparar
calificaciones de series
      //Getters
      std::string getTitulo();
      char getCategoria();
      float getRating();
      std::vector<Episodio>& getEpisodios(); //Pasar
referencia a direccion de memoria y modificar
directamente
```

```
    protected:
      std::string titulo; //Titulo de la serie
      char genero;        //genero de la serie
      int temporadas;     //Numero de temporadas
      int num_episodios;  //Numero de episodios
      std::vector<Episodio> episodios; //Episodios de
la serie
      //Rating promedio
      int num_ratings; //numero de calificaciones
totales (para calcular promedio)
      float calificacionAvg; //calificacion promedio de
la serie
      std::vector <int> ratings; //vector que almacena
calificaciones
};

#endif //SERIE_H
```

Serie.cpp (implementación de clase Serie)

```
#include <iostream>
#include <vector>
#include "../.include/Serie.hpp"

Serie::Serie(std::string title, char genre, int num_ep,
int temps, std::vector<Episodio> epis){
    titulo = title;
    genero = genre;
    temporadas = temps;
    num_episodios = num_ep;
    episodios = epis;
}

void Serie::calificarEps(int index){
    episodios[index].calificar();
```

```
}

std::string detectCategoria(char cat){ //Funcion usada en mostrarInfo para imprimir nombre completo de genero
    int index;
    std::string generos[]  = {"Accion", "Comedia", "Drama", "Terror/suspenso", "Infantil"};
    char charsCategorias[] = {'a', 'c', 'd', 't', 'i'};
    int arraySize = sizeof(generos) / sizeof(generos[0]); //Calcular el numero de elementos en el array
    for(int i=0;i<arraySize;i++){
        if(cat == charsCategorias[i]){
            index = i;   //Encontrar el indice de la categoria (en char) para traducir a string
            break;
        }
    }
    return generos[index];  //Mapeo a string
}

void Serie::mostrarInfo(){
    int epMejor = 0; //para guardar el indice del episodio mejor evaluado
    for(int i=0; i < episodios.size() - 1; i++){
        if(episodios[i] < episodios[i+1]){
            epMejor = i + 1;
        }else if (episodios[i+1] < episodios[i]){
            epMejor = i;
        }
    }
    std::cout << "INFORMACION DE LA SERIE" << std::endl;
```

```
    std::cout <<
"-------------------------------------------------
---------" << std::endl;
    std::cout << "TITULO: " << titulo << std::endl;
    std::cout << "GENERO: " << detectCategoria(genero)
<< std::endl;
    std::cout << "TEMPORADAS: " << temporadas <<
std::endl;
    std::cout << "EPISODIOS: " << num_episodios <<
std::endl;
    if(episodios[epMejor].getRating() != 0){
        std::cout << "EPISODIO MEJOR EVALUADO: " <<
episodios[epMejor].getTitulo() << "_ RATING = " <<
episodios[epMejor].getRating() << std::endl;
    }else{
        std::cout << "EPISODIO MEJOR EVALUADO: NO HAY "
<< std::endl;
    }
    std::cout << std::endl;
    for(int i=0; i < episodios.size(); i++){
        std::cout << episodios[i].getTitulo() << " -
Rating = " << episodios[i].getRating() << std::endl;
    }
    std::cout <<
"-------------------------------------------------
---------" << std::endl;
}

void Serie::rate(){ //Funcion para que usuario
introduzca calificacion de un video
    int calificacion = 0;
    std::cout << "\nCALIFICAR VIDEO" << std::endl;
```

```
    std::cout <<
"-------------------------------------------------
---------" << std::endl;
    std::cout << titulo << std::endl;
    std::cout << "Introduzca calificacion para el video
-> ";
    std::cin >> calificacion;
    while(calificacion < 1 || calificacion > 5){
//Mientras entrada no sea valida
        std::cout << "Numero no valido, introducir del
1-5" << std::endl;
        std::cin >> calificacion;
    }
    updateRate(calificacion);  //Guardar nueva
calificacion y actualizar promedio de ratings
    std::cout << "Rating " << calificacion << "
guardado exitosamente." << std::endl;
    std::cout << "Numero de calificaciones: " <<
ratings.size() << std::endl;
    std::cout << "Rating actual del video: " <<
calificacionAvg << std::endl;
}

//Llamar cada vez que se agrega una calificacion
void Serie::updateRate(int calificacion){ //Funcion
para calcular el promedio de ratings actualizado
    ratings.push_back(calificacion); //Actualizar el
vector de calificaciones
    int total = 0;
    int counter = 0;
    for(int num:ratings){
        total += num; //Sumar calificaciones
        counter++;
    }
```

```
    calificacionAvg = float(total) / counter;
//Obtencion del rating promedio
}

bool operator < (Serie ser1, Serie ser2){
    if(ser1.calificacionAvg < ser2.calificacionAvg){
        return true;
    }else{
        return false;
    }
}

std::string Serie::getTitulo(){
    return titulo;
}

float Serie::getRating(){
    return calificacionAvg;
}

char Serie::getCategoria(){
    return genero;
}

std::vector<Episodio>& Serie::getEpisodios(){
    return episodios;
}
```

Episodio.hpp (declaración de clase Episodio)

```
#ifndef EPISODIO_H
#define EPISODIO_H
#include <iostream>
#include "./Video.hpp"
```

```
class Episodio : public Video{
    public:
      Episodio(std::string, char, int, int, int);
//Constructor del episodio
      std::string getTitulo();  //getter para el titulo
    private:
      int temporada; //Temporada a la que pertenece
};

#endif //EPISODIO_H
```

Episodio.cpp (implementación de clase Episodio)

```
#include <iostream>
#include "../.include/Episodio.hpp"

Episodio::Episodio(std::string title, char genre, int
minutos, int estreno, int temp){
    titulo = title;
    genero = genre;
    duracion = minutos;
    fecha = estreno;
    temporada = temp;
}

std::string Episodio::getTitulo(){
    return titulo;
}
```

Código principal (main.cpp)

```
#include <iostream>
#include <vector>
#include "../.include/Video.hpp"
```

```cpp
#include "../.include/Pelicula.hpp"
#include "../.include/Serie.hpp"
#include "../.include/Episodio.hpp"

//Biblioteca de peliculas
Pelicula shrek2("Shrek 2", 'i', 120, 2008);
Pelicula ironman("Ironman: El hombre de hierro", 'a',
130, 2008);
Pelicula terminator("Terminator", 'a', 125, 1984);
Pelicula matrix("Matrix", 'a', 130, 1999);
Pelicula fastf("Rapidos y furiosos 7", 'a', 140, 2015);
Pelicula rush("Una pareja explosiva", 'c', 98, 1998);

//Series (declaracion de objetos)
Episodio ep1("Big Bang Ep1", 'c', 60, 2008, 1);
Episodio ep2("Big Bang Ep2", 'c', 55, 2009, 1);
Episodio ep3("Big Bang Ep3", 'c', 40, 2010, 2);
Episodio ep4("Big Bang Ep4", 'c', 68, 2011, 2);
std::vector<Episodio> bigBangEps = {ep1, ep2, ep3,
ep4};
Serie bigbang("The big bang theory", 'c', 4, 2,
bigBangEps);

Episodio epd1("Good doctor Ep1", 'c', 80, 2019, 1);
Episodio epd2("Good doctor Ep2", 'c', 85, 2020, 1);
Episodio epd3("Good doctor Ep3", 'c', 75, 2021, 2);
Episodio epd4("Good doctor Ep4", 'c', 70, 2022, 2);
std::vector<Episodio> goodDocEps = {epd1, epd2, epd3,
epd4};
Serie goodDoctor("The good doctor", 'd', 4, 2,
goodDocEps);

//Agrupacion de series y peliculas
```

```
Pelicula peliculas[] = {shrek2, ironman, terminator,
matrix, fastf, rush};
Serie series[] = {bigbang, goodDoctor};

void mostrarCatalogo(){  //Codigo para mostrar catalogo
    std::cout << "\nCATALOGO DE PELICULAS & SERIES" <<
std::endl;
    std::cout <<
"-------------------------------------------------
-" << std::endl;
    std::cout << "SERIES --> " << sizeof(series) /
sizeof(series[0]) << std::endl;  //Calcular cantidad de
series
    for(Serie ser : series){
//Iterar en cada serie del array de series disponibles
        std::cout << ser.getTitulo() << std::endl;
//Imprimir titulos
    }
    std::cout << "\nPELICULAS --> " <<
sizeof(peliculas) / sizeof(peliculas[0]) << std::endl;
//Calcular cantidad de pelis
    for(Pelicula peli : peliculas){
//Iterar en cada pelicula del array global de peliculas
        std::cout << peli.getTitulo() << std::endl;
//Imprimir titulos
    }
}

void calificarVideo(){ //funcion para calificar
pelicula o serie
    int sizePeli =
sizeof(peliculas)/sizeof(peliculas[0]);
    int sizeSerie = sizeof(series)/sizeof(series[0]);
    std::cout << "CALIFICACION DE VIDEOS" << std::endl;
```

```
    std::cout <<
"-------------------------------------------------
-" << std::endl;
    std::cout << "INDICE" << std::endl;
    std::cout << "PELICULAS" << std::endl;
    //Imprimir primero todas las peliculas y despues
todas las series
    for(int i = 0; i < sizePeli; i++){
        std::cout << peliculas[i].getTitulo() << " - "
<< i + 1 << std::endl;
    }
    std::cout << "\nSERIES" << std::endl;
    for(int i=sizePeli; i < sizeSerie + sizePeli; i++){
        std::cout << series[i - sizePeli].getTitulo() <<
" - "  << i + 1 << std::endl;  //Normalizar el indice
    }
    //Eleccion de video
    int index = -1;
    std::cout << "Introduce indice para elegir video :
";
    std::cin >> index;
    while(index < 1 || index > sizePeli + sizeSerie){
//CHECAR ESTA LINEA
        std::cout << "Introduce un indice valido: ";
        std::cin >> index;
    }
    //Enfoque en video elegido
    if(index <= sizePeli){
        std::cout << "PELICULA ELEGIDA: " <<
peliculas[index-1].getTitulo() << std::endl;
        peliculas[index - 1].calificar();
    }else{
        std::cout << "SERIE ELEGIDA: " <<
series[(index-sizePeli)-1].getTitulo() << std::endl;
```

```
        series[(index-sizePeli)-1].rate();
    }
}

void imprimirGenero(){ //Funcion para checar las series
y peliculas de un determinado genero e imprimirlas
   char opciones[] = {'a','c', 'd', 't', 'i'};
   std::string opcionString[] = {"Accion", "Comedia",
"Drama", "Suspenso/terror", "Infantil"};
   int index = 0;
   std::cout << "EXPLORAR POR GENERO" << std::endl;
   std::cout <<
"-----------------------------------------------------
--" << std::endl;
   std::cout << "INDICE :" << std::endl;
   std::cout << "OPCION 1 - ACCION" << std::endl;
   std::cout << "OPCION 2 - COMEDIA" << std::endl;
   std::cout << "OPCION 3 - DRAMA" << std::endl;
   std::cout << "OPCION 4 - SUSPENSO/TERROR" <<
std::endl;
   std::cout << "OPCION 5 - INFANTIL" << std::endl;
   std::cout << "Elegir indice de opcion: ";
   std::cin >> index;

   while(index <= 0 ||
sizeof(opciones)/sizeof(opciones[0]) < index){
      std::cout << "Elegir indice de opcion valido: ";
      std::cin >> index;
   }
   char catElegida = opciones[index - 1];
   //ITERAR PARA ENCONTRAR CATEORIAS E IR GUARDANDO
INDICES
   std::vector <int> clasificadosIndex;
   int counter = 0;  //CONTADOR PARA MAPEAR A ENTEROS
```

```
   for(Pelicula peli : peliculas){
      char cat = peli.getCategoria();
      if(cat == catElegida){    //Si la categoria del
video actual es la misma que la elegida, agregar
         clasificadosIndex.push_back(counter);
      }
      counter++;
   }
   for(Serie ser : series){
      char cat = ser.getCategoria(); //Obtener
categoria de cada serie en formato char
      if(cat == catElegida){         //Si la categoria
del video actual es la misma que la elegida, agregar
         clasificadosIndex.push_back(counter);
      }
      counter ++;
   }
   std::cout << "\nCategoria elegida: " <<
opcionString[index-1] << std::endl;
   std::cout << "Mostrando videos de " <<
opcionString[index-1] << std::endl;
   for(int i : clasificadosIndex){ //Iterar en indices
de videos clasificados para mapear a peliculas/series
      if(i < sizeof(peliculas)/sizeof(peliculas[0])){
//El indice es de una pelicula
         std::cout << peliculas[i].getTitulo() << " -
Tipo: Pelicula" << std::endl;
      }else{
         std::cout << series[i -
sizeof(peliculas)/sizeof(peliculas[0])].getTitulo() <<
" - Tipo: Serie" << std::endl; //Si el indice es mayor
al size de peliculas, entonces es serie
      }
   }
```

```
}

void mejorCalificado(){
   std::cout << "\nVIDEOS CON MEJOR CALIFICACION" <<
std::endl;
   std::cout <<
"-------------------------------------------------
--" << std::endl;
   std::cout << "Opcion 1: Pelicula mejor calificada"
<< std::endl;
   std::cout << "Opcion 2: Serie mejor calificada" <<
std::endl;
   std::cout << "Elegir opcion: ";
   int opcion;
   std::cin >> opcion; //Entrada de opcion para usuario
   while(opcion < 1 || opcion > 2){
      std::cout << "Elegir opcion valida: ";
      std::cin >> opcion; //Entrada de opcion para
usuario
   }
   int indexMejor = -1; //variable para guardar el
mejor calificado
   if(opcion == 1){ //si usuario elige categoria
pelicula
      for(int i = 0; i <
(sizeof(peliculas)/sizeof(peliculas[0])-1); i++){
//iterar en todas las peliculas
         if(peliculas[i] < peliculas[i + 1]){
//SOBRECARGA DE OPERADORES
            indexMejor = i + 1;  //siguiente indice si
el actual es menor al siguiente
         }
      }
```

```
      std::cout << "Pelicula mejor calificada: " <<
peliculas[indexMejor].getTitulo()
                << " con calificacion: " <<
peliculas[indexMejor].getRating(); //Imprimir peli
mejor calificada
   }else{ //ver series mejor calificadas
      for(int i = 0; i <
(sizeof(series)/sizeof(series[0])-1); i++){
         if(series[i] < series[i + 1]){  //SOBRECARGA
DE OPERADORES
            indexMejor = i + 1;
         }
      }
      std::cout << "Serie mejor calificada: " <<
series[indexMejor].getTitulo()
                << " con calificacion: " <<
series[indexMejor].getRating(); //Imprimir serie mejor
calificada
   }
}

void mostrarParticular(){  //Mostrar info de un video
en particular
   std::cout << "\nINFORMACION DE VIDEO PARTICULAR" <<
std::endl;
   std::cout <<
"-----------------------------------------------------
--" << std::endl;
   std::cout << "OPCION 1: Pelicula " << std::endl;
   std::cout << "OPCION 2: Serie " << std::endl;
   std::cout << "Elegir tipo de video: ";
   int option;
   std::cin >> option;
   while(option != 1 && option != 2){ //Validar input
```

```
        std::cout << "Elegir tipo de video valido: ";
        std::cin >> option;
    }
    //Casos de eleccion
    int index = 0;
    if(option == 1){ //Si eligio tipo pelicula
        for(int i = 0; i < sizeof(peliculas) /
sizeof(peliculas[0]); i++){
            std::cout << i + 1 << " - " <<
peliculas[i].getTitulo() << std::endl; //imprimir
titulos
        }
        std::cout << "Elegir indice de pelicula: ";
        std::cin >> index; //tomar indice como input
        while(index < 1 || index > sizeof(peliculas) /
sizeof(peliculas[0])){ //Mientras entrada sea no valida
            std::cout << "Elegir indice de pelicula
valido: ";
            std::cin >> index;
        }
        peliculas[index-1].mostrarInfo(); //Mostrar
informacion de la pelicula con indice especificado
    }else{    //Si usuario elige serie
        for(int i=0; i < sizeof(series) /
sizeof(series[0]); i++){
            std::cout << i + 1 << " - " <<
series[i].getTitulo() << std::endl;
        }
        std::cout << "Elegir indice de serie: ";
        std::cin >> index; //tomar indice como input
        while(index < 1 || index > sizeof(series) /
sizeof(series[0])){ //Mientras entrada sea no valida
            std::cout << "Elegir indice de serie valido:
";
```

```
            std::cin >> index;
        }
        series[index-1].mostrarInfo(); //Mostrar
informacion de la serie con indice especificado
    }
}

void calificarEpisodio(){ //funcion para calificar
episodios individuales de una serie
    std::cout << "\nCALIFICAR EPISODIOS DE UNA SERIE" <<
std::endl;
    std::cout <<
"-----------------------------------------------------
--" << std::endl;
    for(int i=0; i < sizeof(series) / sizeof(series[0]);
i++){
        std::cout << "OPCION " << i + 1 << " - " <<
series[i].getTitulo() << std::endl;
    }
    int opcion;
    std::cin >> opcion;
    while(opcion < 1 || opcion > sizeof(series) /
sizeof(series[0])){ //Mientras entrada sea no valida
        std::cout << "Elegir indice de serie valido: ";
        std::cin >> opcion;
    }
    //Listar episodios
    std::cout << "EPISODIOS DE: " << series[opcion -
1].getTitulo() << std::endl;
    std::vector<Episodio> eps = series[opcion -
1].getEpisodios();
    for(int i=0; i < eps.size(); i++){ //Iterar en
vector de episodios por indice
```

```
        std::cout << "OPCION " << i + 1 << " - " <<
eps[i].getTitulo() << std::endl; //Mostrar titulos de
episodios
    }
    //Elegir opcion de episodio
    int opcionEp = 0;
    std::cout << "Elegir episodio: " << std::endl;
    std::cin >> opcionEp;
    while(opcionEp < 1 || opcionEp > eps.size()){
//verificar entrada valida
        std::cout << "Elegir indice de episodio valido:
";
        std::cin >> opcionEp;
    }
    std::cout << "Serie: " << series[opcion -
1].getTitulo() << std::endl;
    std::cout << "Episodio elegido: " << eps[opcionEp -
1].getTitulo() << std::endl;
    //Rating
    series[opcion - 1].getEpisodios()[opcionEp -
1].calificar();
}

void menu(){
    bool run = true;
    while(run){
        int option = 0;
        std::cout << "\nMENU DEL PROGRAMA" << std::endl;
        std::cout <<
"----------------------------------------------------
--" << std::endl;
        std::cout << "OPCION 1 = Imprimir catalogo de
videos" << std::endl;
```

```
      std::cout << "OPCION 2 = Calificar un video" <<
std::endl;
      std::cout << "OPCION 3 = Imprimir videos por
genero" << std::endl;
      std::cout << "OPCION 4 = Ver mejores calificados"
<< std::endl;
      std::cout << "OPCION 5 = Mostrar informacion de
video especifico" << std::endl;
      std::cout << "OPCION 6 = Evaluar episodios de una
serie" << std::endl;
      std::cout << "OPCION 7 = Salir" << std::endl;
      std::cout << "Introduce numero de opcion: ";
      std::cin >> option;
      //Checar opcion valida
        while(option <= 0 || option > 7){
         std::cout << "Introduce numero de opcion
valido: ";
         std::cin >> option;
      }
      switch(option){
         case 1:
            mostrarCatalogo(); //Mostar el catalogo de
videos disponibles
            break;
         case 2:
            calificarVideo(); //Calificar un video
            break;
         case 3:
            imprimirGenero(); //Mostrar videos de un
genero
             break;
         case 4:
            mejorCalificado(); //Ver pelicula o serie
mejor calificados
```

```
            break;
         case 5:
            mostrarParticular(); //Mostrar info de un
video en particular
            break;
         case 6:
            calificarEpisodio(); //calificar un
episodio de una serie
            break;
         case 7:
            std::cout << "\nQue tenga un excelente dia"
<< std::endl; //Salir del programa
            run = false;
            break;
      }
   }
}

int main(){
    menu();
    return EXIT_SUCCESS;
}
```

Casos de prueba

- **Menú**

```
MENU DEL PROGRAMA
-----------------------------------------------------
OPCION 1 = Imprimir catalogo de videos
OPCION 2 = Calificar un video
OPCION 3 = Imprimir videos por genero
OPCION 4 = Ver mejores calificados
OPCION 5 = Mostrar informacion de video especifico
OPCION 6 = Evaluar episodios de una serie
OPCION 7 = Salir
Introduce numero de opcion:
```

- **Opción 1 - Imprimir catálogo**

```
MENU DEL PROGRAMA
-----------------------------------------------------------
OPCION 1 = Imprimir catalogo de videos
OPCION 2 = Calificar un video
OPCION 3 = Imprimir videos por genero
OPCION 4 = Ver mejores calificados
OPCION 5 = Mostrar informacion de video especifico
OPCION 6 = Evaluar episodios de una serie
OPCION 7 = Salir
Introduce numero de opcion: 1

CATALOGO DE PELICULAS & SERIES
-----------------------------------------------------------
SERIES --> 2
The big bang theory
The good doctor

PELICULAS --> 6
Shrek 2
Ironman: El hombre de hierro
Terminator
Matrix
Rapidos y furiosos 7
Una pareja explosiva
```

- **Opción 2: Calificar un video**

```
Introduce numero de opcion: 2
CALIFICACION DE VIDEOS
-----------------------------------------------------
INDICE
PELICULAS
Shrek 2 - 1
Ironman: El hombre de hierro - 2
Terminator - 3
Matrix - 4
Rapidos y furiosos 7 - 5
Una pareja explosiva - 6

SERIES
The big bang theory - 7
The good doctor - 8
Introduce indice para elegir video : 2
PELICULA ELEGIDA: Ironman: El hombre de hierro

CALIFICAR VIDEO
-------------------------------------------------------------
Ironman: El hombre de hierro
Introduzca calificacion para el video -> 5
Rating 5 guardado exitosamente.
Numero de calificaciones: 1
Rating actual del video: 5
```

- **Opción 3 (extra) - Imprimir videos por género**

```
Introduce numero de opcion: 3
EXPLORAR POR GENERO
------------------------------------------------------
INDICE :
OPCION 1 - ACCION
OPCION 2 - COMEDIA
OPCION 3 - DRAMA
OPCION 4 - SUSPENSO/TERROR
OPCION 5 - INFANTIL
Elegir indice de opcion: 1

Categoria elegida: Accion
Mostrando videos de Accion
Ironman: El hombre de hierro - Tipo: Pelicula
Terminator - Tipo: Pelicula
Matrix - Tipo: Pelicula
Rapidos y furiosos 7 - Tipo: Pelicula
```

-Opción 4 - Ver mejores calificados

```
Introduce numero de opcion: 4

VIDEOS CON MEJOR CALIFICACION
------------------------------------------------------
Opcion 1: Pelicula mejor calificada
Opcion 2: Serie mejor calificada
Elegir opcion: 2
Serie mejor calificada: The good doctor con calificacion: 5
MENU DEL PROGRAMA
------------------------------------------------------
```

-Opción 5 - Mostrar info de video en especifico

```
Introduce numero de opcion: 5

INFORMACION DE VIDEO PARTICULAR
--------------------------------------------------------
OPCION 1: Pelicula
OPCION 2: Serie
Elegir tipo de video: 2
1 - The big bang theory
2 - The good doctor
Elegir indice de serie: 1
INFORMACION DE LA SERIE
-----------------------------------------------------------------
TITULO: The big bang theory
GENERO: Comedia
TEMPORADAS: 2
EPISODIOS: 4
EPISODIO MEJOR EVALUADO: Big Bang Ep1_ RATING = 5

Big Bang Ep1 - Rating = 5
Big Bang Ep2 - Rating = 0
Big Bang Ep3 - Rating = 0
Big Bang Ep4 - Rating = 0
-----------------------------------------------------------------
```

- **Opción 6 - Evaluar episodios de una serie**

```
Introduce numero de opcion: 6

CALIFICAR EPISODIOS DE UNA SERIE
--------------------------------------------------------
OPCION 1 - The big bang theory
OPCION 2 - The good doctor
1
EPISODIOS DE: The big bang theory
OPCION 1 - Big Bang Ep1
OPCION 2 - Big Bang Ep2
OPCION 3 - Big Bang Ep3
OPCION 4 - Big Bang Ep4
Elegir episodio:
2
Serie: The big bang theory
Episodio elegido: Big Bang Ep2

CALIFICAR VIDEO
-----------------------------------------------------------------
Big Bang Ep2
Introduzca calificacion para el video -> 4
Rating 4 guardado exitosamente.
Numero de calificaciones: 1
Rating actual del video: 4
```

- **Opción 7 - Salir**

```
MENU DEL PROGRAMA
------------------------------------------------------------
OPCION 1 = Imprimir catalogo de videos
OPCION 2 = Calificar un video
OPCION 3 = Imprimir videos por genero
OPCION 4 = Ver mejores calificados
OPCION 5 = Mostrar informacion de video especifico
OPCION 6 = Evaluar episodios de una serie
OPCION 7 = Salir
Introduce numero de opcion: 7

Que tenga un excelente dia
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP\.src>
```

<u>INSTRUCCIONES</u>: Para correr código (en powershell)

1. Situarse en carpeta .src

```
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP> cd .\.src\
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP\.src>
```

2. Compilar

```
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP> cd .\.src\
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP\.src> g++ main.cpp Pelicula.cpp Serie.cpp Episodio.cpp Video.cpp -o stream
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP\.src>
```

Comando
g++ main.cpp Pelicula.cpp Serie.cpp Episodio.cpp Video.cpp -o stream

3. Correr código

```
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP> cd .\.src\
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP\.src> g++ main.cpp Pelicula.cpp Serie.cpp Episodio.cpp Video.cpp -o stream
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP\.src> g++ main.cpp Pelicula.cpp Serie.cpp Episodio.cpp Video.cpp -o stream^C
PS C:\Users\jorgl\OneDrive\Escritorio\POO-SP\.src> ./stream.exe

MENU DEL PROGRAMA
------------------------------------------------------------
OPCION 1 = Imprimir catalogo de videos
OPCION 2 = Calificar un video
OPCION 3 = Imprimir videos por genero
OPCION 4 = Ver mejores calificados
OPCION 5 = Mostrar informacion de video especifico
OPCION 6 = Evaluar episodios de una serie
OPCION 7 = Salir
Introduce numero de opcion:
```

Comando ./stream.exe

Posibles limitaciones

El programa es capaz de filtrar entradas que no son válidas volviendo a pedir al usuario que introduzca una opción correcta, sin embargo, esto sólo es posible cuando el dato que se introduce es el mismo tipo que el esperado; cuando esto no ocurre, el programa puede generar errores en la ejecución. Es posible resolver esto mediante el manejo de excepciones, no obstante, en el código actual este concepto no fue implementado.