Proyecto Integrador

Andrea Villaseñor Jiménez A01642088.

Jose Hugo Aguayo Mendoza A01640917.

Jared Rafael Garcia Almaguer A01640914.

Programación orientada a objetos

Grupo 316

Tania del Carmen Rodriguez Flores

Jueves 12 de junio del 2025

Proyecto Integrador - Programación orientada a objetos

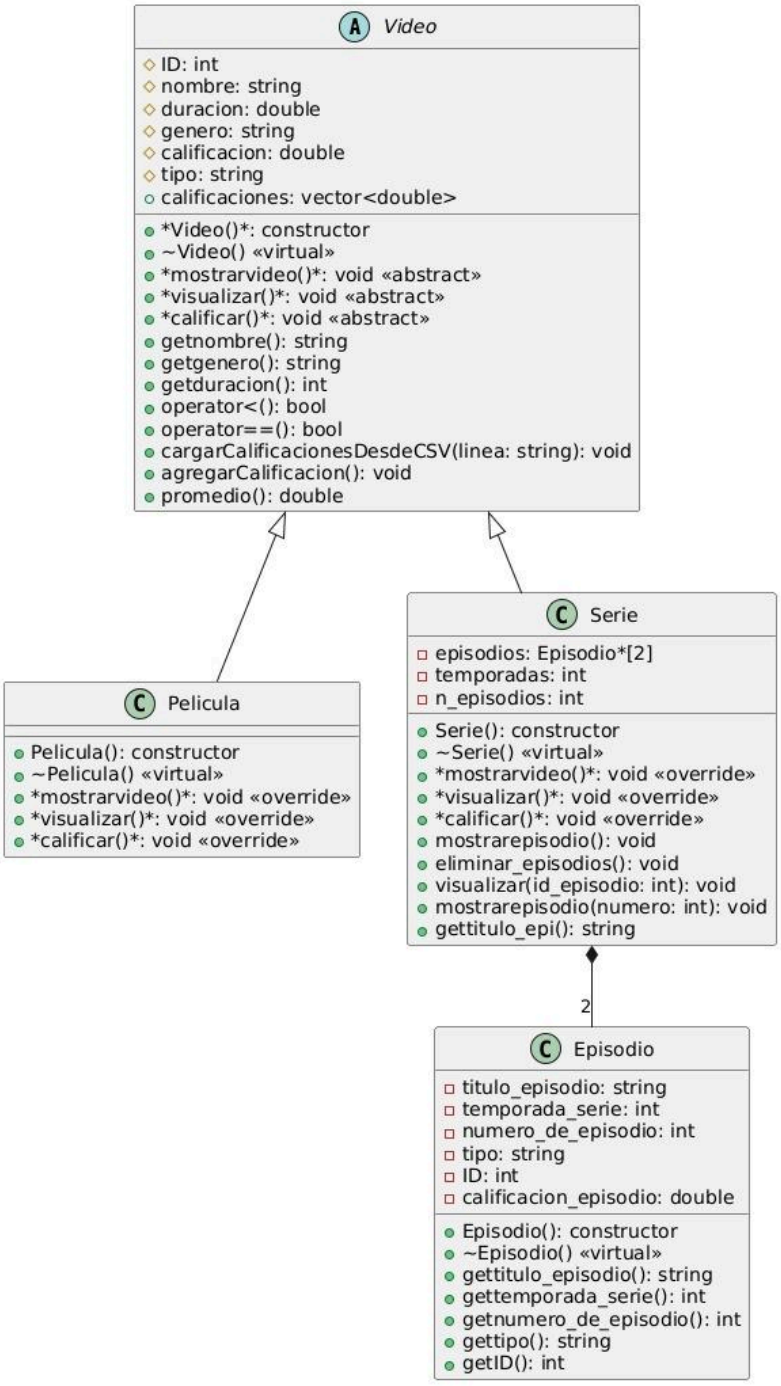
Índice

Introducción

La programación orientada a objetos (POO) es un estilo de programación basado en la idea de clases y objetos que permite organizar el código de una forma muy sencilla por su cercanía al mundo real al momento de modelar, (UNIR, 2023). Este estilo tiene una facilidad mayor para dar a entender, mantener y reutilizar código haciendo que la creación de sistemas complicados sea más simple. En lugar de pensar en funciones sueltas, en POO, organizamos el código con respecto a objetos que se relacionan entre sí dejando de usar pura lógica permitiéndonos ser más intuitivos, inspirándonos en cómo percibimos las cosas a nuestro alrededor.

Un ejemplo de implementación puede verse en la situación problema. Ver películas o series hoy en día a través de plataformas de streaming como Disney+, Netflix o HBO se ha vuelto parte cotidiana de nuestras vidas. Esta habilidad de ver el contenido desde cualquier parte requiere de un sistema que organice y muestre el contenido de la mejor forma, se fue planteando en la situación problema. Se trataba de crear un modelo de servicio de streaming para películas y series, donde el sistema tiene que ser capaz de mostrar la información como el nombre, duración, género y una calificación promedio (del 1 al 5) dada por los usuarios (que estará visible junto a la película o serie a la que pertenezca) acomodados en una lista. Además las series deben enseñar la lista de episodios y la temporada de la que son parte y tener cierta memoria de las calificaciones anteriores para así ir manteniendo su promedio dinámico.

Diagrama de clases UML



Se diseñó de esta manera por la practicidad y porque las instrucciones nos pedían unas ciertas clases con sus propiedades, y bajo las instrucciones pudimos darle nuestro estilo, aun así el molde del diseño fue otorgado por las instrucciones. Se tienen 4 clases, usamos el concepto de herencia y el concepto de composición, como se puede notar en el diagrama UML y en el código, nos intentamos apegar lo más posible a lo visto en clase y a lo visto en

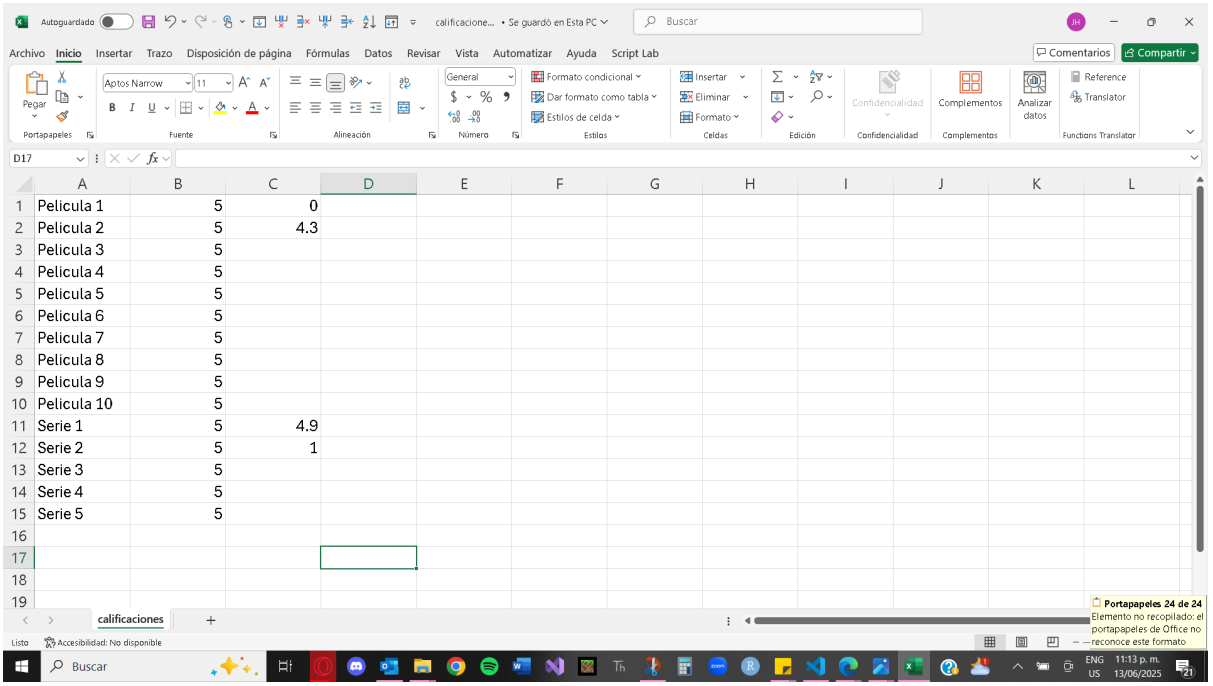
la materia previa llevada en primer semestre. Los rombos amairllos representan los atributos que tiene la clase Video, mientras los cuadrados representan los de Episodio y Serie. Los círculos verdes representan los métodos necesarios para el funcionamiento del código que son los que incluyen cada clase. Pues el diagrama UML es de clases. A su vez, se puede ver que Video es la clase padre, donde hereda Película y Serie, pero no Episodio. La clase Episodio por otro lado, tiene una composición con Serie.

Ejemplo de ejecución

Menú principal:

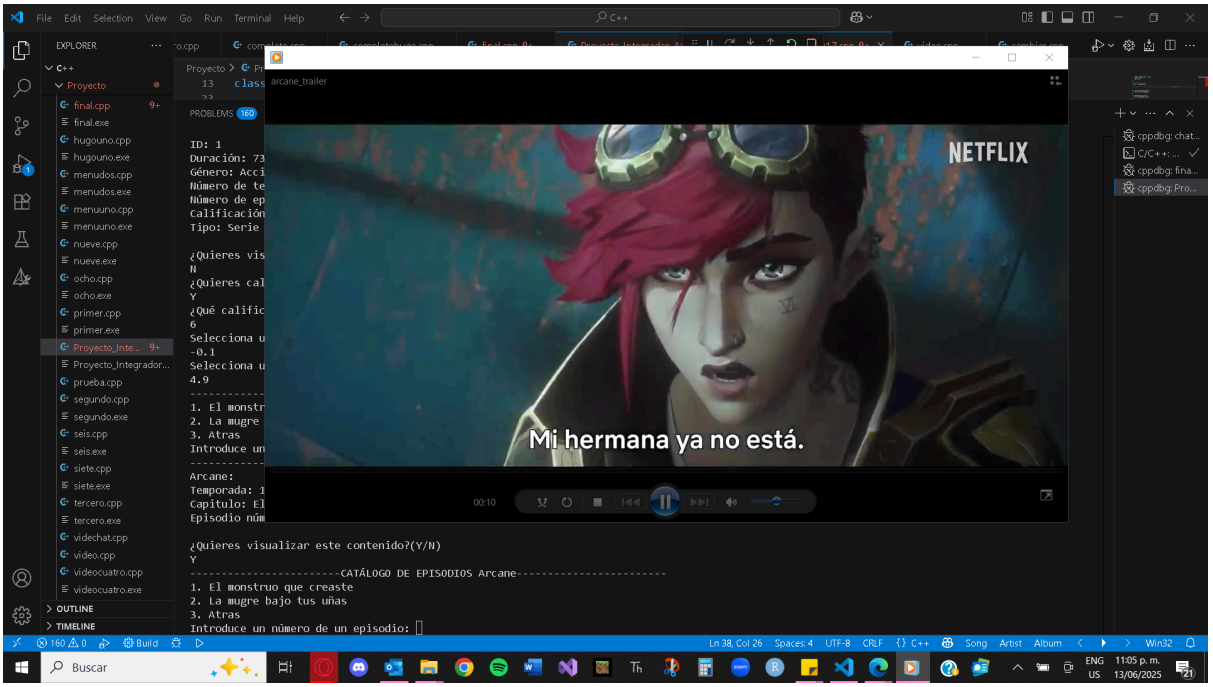
```
--------------------------------MENU---------------------------------
1. Mostrar los videos en general con una cierta calificación o de un cierto género
2. Mostrar los episodios de una determinada serie con una calificacion determinada
3. Mostrar las películas con cierta calificacion
4. Calificar un video
5. Comparar la duración de las películas.
0. Salir
Selecciona una opción del menú:
```

Vista archivo Excel:



	A	B	C
1	Pelicula 1	5	0
2	Pelicula 2	5	4.3
3	Pelicula 3	5	
4	Pelicula 4	5	
5	Pelicula 5	5	
6	Pelicula 6	5	
7	Pelicula 7	5	
8	Pelicula 8	5	
9	Pelicula 9	5	
10	Pelicula 10	5	
11	Serie 1	5	4.9
12	Serie 2	5	1
13	Serie 3	5	
14	Serie 4	5	
15	Serie 5	5	

Vista vídeos:

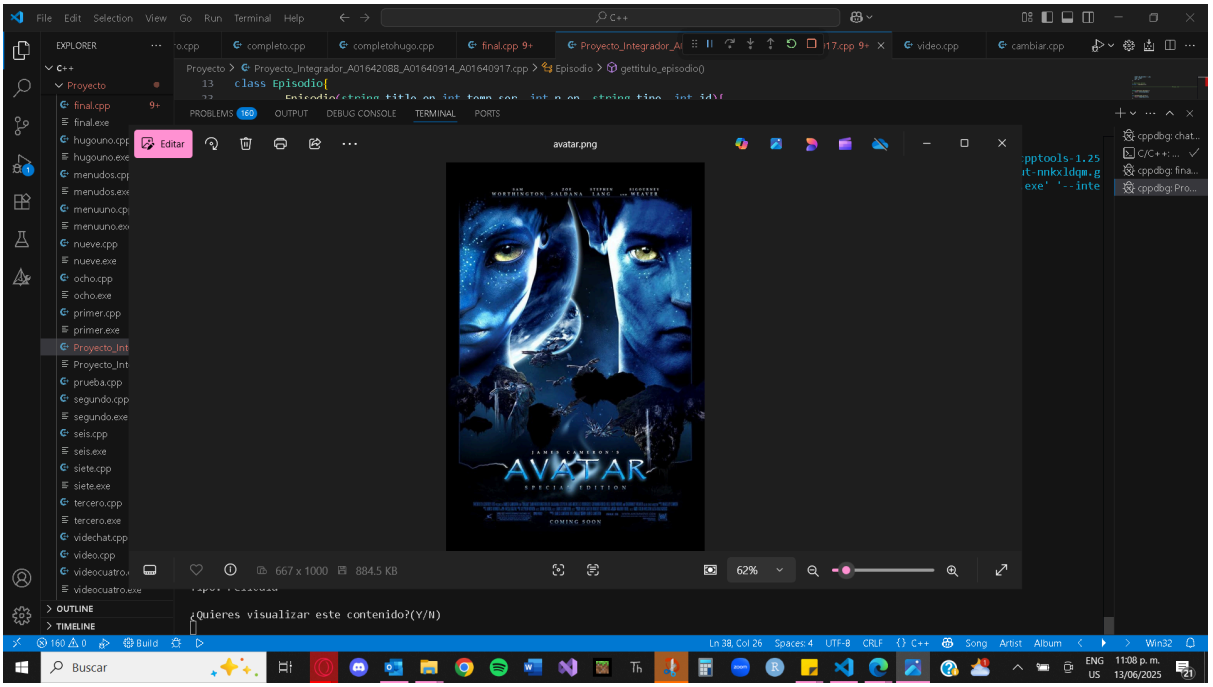


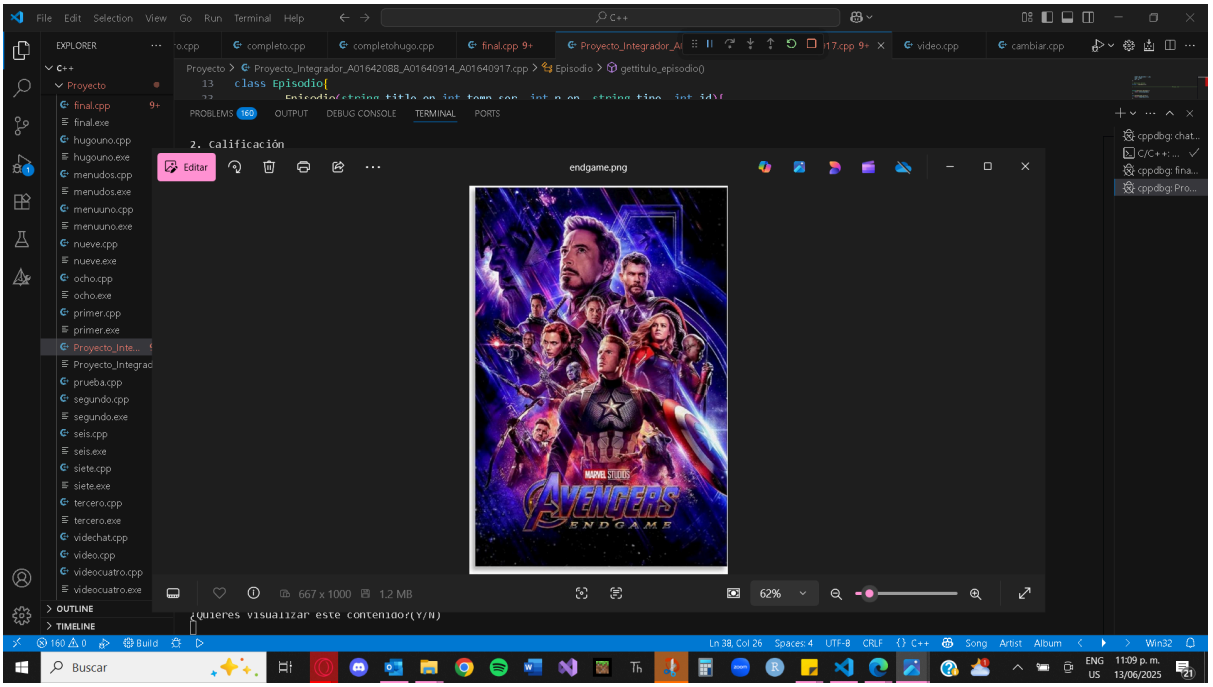

```
---------------------------------MENU---------------------------------
1. Mostrar los videos en general con una cierta calificación o de un cierto género
2. Mostrar los episodios de una determinada serie con una calificacion determinada
3. Mostrar las películas con cierta calificacion
4. Calificar un video
5. Comparar la duración de las películas.
0. Salir
Selecciona una opción del menú: 4
Calificar un video
-------------------------VIDEOS-------------------------
1. Película
2. Serie
3. Atras
¿Quieres calificar?
2
-------------------------CATÁLOGO DE SERIES-------------------------
1. Arcane
2. Game of Thrones
3. Stranger Things
4. Gravity Falls
5. New Amsterdam
6. Atrás
Introduce el número de la serie que quieres calificar: 1
```

```
------------------------SERIE-----------------------
Arcane
ID: 1
Duración: 735 minutos
Género: Acción
Número de temporadas: 2
Número de episodios: 18
Calificación: 5
Tipo: Serie

¿Quieres visualizar este contenido?(Y/N)
N
¿Quieres calificar esta serie?(Y/N)
Y
¿Qué calificación le pones a esta serie?(0-5)
6
Selecciona una opción válida ¿Qué calificación le pones a esta serie?(0-5)
-0.1
Selecciona una opción válida ¿Qué calificación le pones a esta serie?(0-5)
4.9
------------------------CATÁLOGO DE EPISODIOS Arcane-----------------------
1. El monstruo que creaste
2. La mugre bajo tus uñas
3. Atras
Introduce un número de un episodio: 1
------------------------EPISODIO 0-----------------------
Arcane:
Temporada: 1
Capitulo: El monstruo que creaste
Episodio número: 9

¿Quieres visualizar este contenido?(Y/N)
Y
```

```
------------------------CATÁLOGO DE EPISODIOS Arcane------------------------
1. El monstruo que creaste
2. La mugre bajo tus uñas
3. Atras
Introduce un número de un episodio: 3
------------------------CATÁLOGO DE SERIES------------------------
1. Arcane
2. Game of Thrones
3. Stranger Things
4. Gravity Falls
5. New Amsterdam
6. Atrás
Introduce el número de la serie que quieres calificar: 6
------------------------VIDEOS------------------------
1. Película
2. Serie
3. Atras
¿Quieres calificar?
3
-------------------------------MENU-------------------------------
1. Mostrar los videos en general con una cierta calificación o de un cierto género
2. Mostrar los episodios de una determinada serie con una calificacion determinada
3. Mostrar las películas con cierta calificacion
4. Calificar un video
5. Comparar la duración de las películas.
0. Salir
Selecciona una opción del menú: 0
Adiós, vuelva pronto
```

Vista fotos:




File Edit Selection View Go Run Terminal Help
EXPLORER
Proyecto
final.cpp
final.exe
hugouno.cpp
hugouno.exe
menudos.cpp
menudos.exe
menuuno.cpp
menuuno.exe
nueve.cpp
nueve.exe
ocho.cpp
ocho.exe
primer.cpp
primer.exe
prueba.cpp
segundo.cpp
segundo.exe
seis.cpp
seis.exe
siete.cpp
siete.exe
tercero.cpp
tercero.exe
videchat.cpp
video.cpp
videocuatro.cpp
videocuatro.exe
OUTLINE
TIMELINE
completo.cpp
completohugo.cpp
final.cpp 9+
video.cpp
cambiar.cpp
class Episodio{
PROBLEMS 160
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
2. Calificación
Editar
endgame.png
667 x 1000
1.2 MB
62%
¿Quieres visualizar este contenido?(Y/N)
Ln 38, Col 26
Spaces: 4
UTF-8
CRLF
C++
Win32
Buscar
ENG
US
11:09 p. m.
13/06/2025

```
------------------------------MENU------------------------------
1. Mostrar los videos en general con una cierta calificación o de un cierto género
2. Mostrar los episodios de una determinada serie con una calificacion determinada
3. Mostrar las películas con cierta calificacion
4. Calificar un video
5. Comparar la duración de las películas.
0. Salir
Selecciona una opción del menú: 1
----------------------------MENU FILTROS----------------------------
1. Género
2. Calificación
3. Atrás
Selecciona una opción: 1
----------------------------MENU GÉNEROS----------------------------
1. Drama
2. Acción
3. Misterio
4. Atrás
Selecciona un opción: 2
-----------------------ACCIÓN-----------------------

-----------------------PELICULA-----------------------
Avatar
ID: 2
Duración: 162 minutos
Género: Acción
Calificación: 5
Tipo: Pelicula

¿Quieres visualizar este contenido?(Y/N)
N
¿Quieres calificar esta pelicula?(Y/N)
Y
¿Qué calificación le pones a esta pelicula?(0-5)
4.3
```

```
------------------------PELICULA------------------------
Avengers: Endgame
ID: 3
Duración: 181 minutos
Género: Acción
Calificación: 5
Tipo: Pelicula

¿Quieres visualizar este contenido?(Y/N)
Y
¿Quieres calificar esta pelicula?(Y/N)
N

------------------------PELICULA------------------------
Avatar: El camino del agua
ID: 4
Duración: 192 minutos
Género: Acción
Calificación: 5
Tipo: Pelicula

¿Quieres visualizar este contenido?(Y/N)
N
¿Quieres calificar esta pelicula?(Y/N)
N
```

```
------------------------PELICULA------------------------
Star Wars: El despertar de la fuerza
ID: 6
Duración: 138 minutos
Género: Acción
Calificación: 5
Tipo: Pelicula

¿Quieres visualizar este contenido?(Y/N)
N
¿Quieres calificar esta pelicula?(Y/N)
N

------------------------PELICULA------------------------
Avengers: Infinity War
ID: 7
Duración: 149 minutos
Género: Acción
Calificación: 5
Tipo: Pelicula

¿Quieres visualizar este contenido?(Y/N)
N
¿Quieres calificar esta pelicula?(Y/N)
N

------------------------PELICULA------------------------
Los Vengadores
ID: 8
Duración: 143 minutos
Género: Acción
Calificación: 5
Tipo: Pelicula
```

```
¿Quieres visualizar este contenido?(Y/N)
N
¿Quieres calificar esta pelicula?(Y/N)
N

------------------------PELICULA------------------------
Jurassic World
ID: 9
Duración: 124 minutos
Género: Acción
Calificación: 5
Tipo: Pelicula

¿Quieres visualizar este contenido?(Y/N)
N
¿Quieres calificar esta pelicula?(Y/N)
N
------------------------SERIE------------------------
Arcane
ID: 1
Duración: 735 minutos
Género: Acción
Número de temporadas: 2
Número de episodios: 18
Calificación: 4.95
Tipo: Serie

¿Quieres visualizar este contenido?(Y/N)
N
¿Quieres calificar esta serie?(Y/N)
N
¿Quieres ver los episodios disponibles de la serie?(Y/N)
N
```

```
------------------------SERIE------------------------
Game of Thrones
ID: 2
Duración: 4170 minutos
Género: Acción
Número de temporadas: 8
Número de episodios: 73
Calificación: 3
Tipo: Serie

¿Quieres visualizar este contenido?(Y/N)
N
¿Quieres calificar esta serie?(Y/N)
N
¿Quieres ver los episodios disponibles de la serie?(Y/N)
N
------------------------------MENU GÉNEROS------------------------------
1. Drama
2. Acción
3. Misterio
4. Atrás
Selecciona un opción: 4
------------------------------MENU FILTROS------------------------------
1. Género
2. Calificación
3. Atrás
Selecciona una opción: 3
-------------------------------MENU-------------------------------
1. Mostrar los videos en general con una cierta calificación o de un cierto género
2. Mostrar los episodios de una determinada serie con una calificacion determinada
3. Mostrar las películas con cierta calificacion
4. Calificar un video
5. Comparar la duración de las películas.
0. Salir
Selecciona una opción del menú: 0
Adiós, vuelva pronto
```

Argumentación del proyecto: por qué se optó por esa solución y no por otras

a) Video, clase abstracta que representa el contenido audiovisual en general, Pelicula y Serie, clases derivadas de video. Episodio una clase que ayuda a Serie modelando los capítulos individuales de la serie. Pelicula y Serie comparten los datos generales, es por eso que lo mejor era tener una clase base en común para los 2 y están separados para no olvidarnos del principio de responsabilidad única.

b) Con la ventaja de evitar duplicar código y tener la habilidad de extender en el futuro el código, las clases Pelicula y Serie heredan de Video como mencionamos antes, ya

que comparten atributos y funciones (virtuales puras) como visualizar(), calificar() y mostrarvideo().

c) Con el objetivo de evitar consecuencias colaterales que no buscamos tener dentro de nuestro sistema protegemos de manera correcta el estado de los objetos respetando claramente el principio de encapsulamiento, Con el private protegimos atributos sensibles de las clases, como el título de los episodios o la temporada de la serie, con protected dejamos que las clases hijas tengan acceso a atributos de la clase madre como el nombre y la duración ya sea de la película o serie y por último con public exponemos la información que se necesite al usuario y al sistema sin arriesgarnos a tener modificaciones que nos queremos.

d) En el código encontramos tanto sobrecarga como sobreescritura de funciones, la sobreescritura la usamos para redefinir los métodos virtuales dentro de las clases hijas Pelicula y Serie, por otra parte, la sobrecarga de funciones la podemos encontrar en la clase hija Serie donde sobrecargamos el método visualizar() para poder ver cada episodio, lo que garantiza que el usuario tenga una mejor experiencia.

e) La capacidad de almacenar y trabajar con los objetos de Película y Serie utilizando punteros de tipo Vídeo* fue posible gracias al polimorfismo. Esto hace que usar distintos datos constantemente y ejecutar lo necesario según el tipo del objeto sea posible y sencillo. Por ejemplo, no hay necesidad de condicionales cuando se utiliza mostrarvideo() o calificar() en un puntero Video*; en su lugar, se ejecuta automáticamente la versión apropiada. Esto hace que el sistema sea más fácil de gestionar. El código cuenta con polimorfismo estático, con la sobrecarga de funciones y de operadores, al igual que con polimorfismo dinámico con herencia, funciones virtuales, punteros y el uso del new.

f) Pensando en usarla como una función abstracta. La clase video utiliza métodos virtuales puros para forzar a sus hijas a adoptar características particulares (utilizar override).

Esto garantiza que la lógica para mostrar, exhibir y clasificar material sea implementada por cualquier clase que derive de Video. Esto permite una programación más controlada desde el diseño.

g) Para comparar el tiempo de dos vídeos, sobrecargamos los operadores < y == dentro de la clase Vídeo. Esto no solo facilita la lectura del código, sino que también permite una forma sencilla de comparar las películas dentro del menú sin necesidad de usar o crear funciones adicionales que solo terminarían por complicar el código.

Casos que harían que el proyecto deje de funcionar

Es importante mencionar que hay métodos o objetos específicos que se necesitaron importar en este código. Por lo que quitar las librerías (o no incluir alguna) podría ocasionar problemas en el código. Condicionando así su funcionamiento y por ende, no pudiendo ejecutarlo en el compilador hasta corregirlo. Aunado a esto, si abres desde otra computadora el programa y no cambias la dirección tanto de los videos como de las imágenes, están no se mostrarán y puede comprometer el funcionamiento del código. Finalmente, ya que este código fue pensado para realizarse en un sistema operativo de Windows, habrá que modificar algunas cosas para que el código no truene, no pueda ejecutarse o no detecte esa librería. Al igual que introducir un carácter en una variable. int o double.

Conclusión personal

Conclusión de Andrea

Después de haber trabajado en el proyecto de programación con mi equipo, aprendí que la programación orientada a objetos puede optimizar mucho tiempo de programación, ejecución o incluso espacio en memoria. Además, esto nos permitió reescribir, reciclar y llamar varios bloques de código las veces necesarias para aquellas tareas específicas. Un

ejemplo de esto se puede ver en los archivos CSV, donde utilizamos un método para guardar este CSV; para crear (o en caso de que ya exista: reescribir) la base de datos, para que no se reinicien los datos si cerramos el programa. De manera que siempre que modificamos las puntuaciones (al calificar alguna serie o película), se recalculan los promedios, se adjunta la calificación y se guardan los CSV respectivos inmediatamente. Aunado a esto, la sobrecarga de operadores nos ayudó a comparar las duraciones entre objetos. Finalmente, me pareció un proyecto maravilloso ya que integraba todos los conceptos que vimeo de POO de una manera muy práctica simulado un sistema de streaming con el que estamos familiarizados en nuestra vida diaria.

Conclusión de Hugo

Este proyecto me resultó divertido y con un gran aprendizaje, ya que pude aplicar lo visto en el salón de clase, pero esta vez en un proyecto un poco más apegado a la realidad. Al tener un código más largo, mejoré mi manera de estructurarlos. De este proyecto me llevo 2 principales aprendizajes: el uso de los diagramas UML para tener mejor visualizado tu código, y el uso de manipular archivos, CSV, MP4 y PNG. Desde mi punto de vista, es una herramienta súper interesante y que puede servir para muchas utilidades. Como el nombre del proyecto lo dice, se logró integrar todo lo aprendido en esta materia, en lo visto en la materia de primer semestre y lo investigado.

Conclusión de Jared

Trabajar con este proyecto me ayudó a tener una mejor comprensión del apoyo tan útil para crear sistemas reales y aplicables que nos da la programación orientada a objetos, gracias al proyecto y a la ayuda de mis compañeros de equipo ahora tengo una mejor habilidad para darle estructura a mis códigos con el uso de clases, herencias, polimorfismo y muchas más con el objetivo de modelar cosas de la vida real como lo son en este caso las películas y las series, además ahora entiendo mejor la ventaja que viene con el uso de clases

para poder reutilizar código y la facilidad que agrega la sobrecarga para hacer trabajos que de otra forma serían más difíciles.

Explicación: puntos extra e investigación

Librería	Elemento proveniente de la librería	Explicación	¿Por qué lo utilizamos? ¿Dónde lo utilizamos?
#include <sstream>	1. stringstream	1. Trata a un string como "un flujo de entrada y salida. Proporciona una forma fácil de convertir entre cadenas y varios tipos de datos", (LabEx, n.d).	● Para el CSV.
#include <vector>	1. vector 2. push_back	1. Matriz que puede cambiar de dimensiones. El vector almacena solo 1 mismo tipo de datos. 2. Añade un elemento al vector. Este se irá al final.	● Para el CSV. ○ Vector de calificaciones (de series o películas) y promedios. ● A diferencia del array, en el vector sí se pueden añadir y eliminar elementos. ○ conveniente ya que desconocemos el tamaño de las posibles calificaciones de las películas/series.
#include <string>	1. .find() 2. .substr() 3. .empty() 4. npos	1. Es una función da como resultado "un iterador que apunta a la primera aparición del valor especificado en el rango de datos, o al final del rango de	● Para el CSV. ○ Para leer el CSV de promedios y de calificaciones (de series y películas).

		datos si no se encontró el valor.”, (w3schools, n.d). 2. Extraer subcadena 3. Indica si está vacío (no hay elementos) es true o false. 4. Se usa con find e indica si no encuentra alguna coincidencia con la cadena.	
#include <fstream>	1. ofstream 2. ifstream 3. .close() 4. fout 5. ifstream::is_open	1. Es el "output file stream"; crea archivos y es una clase de <fstream>. 2. Es el "input file stream"; lee archivos y es una clase de <fstream>. 3. Cerrar el archivo. 4. Objeto que llamamos así para hacer referencia al CSV. 5. “Indica si la secuencia está actualmente asociada a un archivo.”, (Cplusplus, s.f).	● Para generar el CSV. ○ Calificaciones ○ Promedio ● Para leer los archivos CSV.
#include <iostream>	1. using namespace std;	1. “Se pueden usar nombres sin especificar std::nada antes”, (Built In, s.f).	● Para el código en general. ○ no usar std::
#include <windows.h>	1. ShellExecute	1. API Windows	● Videos ○ pop-ups ● Imágenes ○ pop-ups
#include <chrono>	1. chrono:: duraction <double>()	1. Es para el tiempo (timer).	● Videos ● Episodios

#include <thread>	1. this_thread:: sleep_for()	1. Para episodios: al terminar su duración exacta, se ejecuta el segundo.	• Videos • Episodios

Puntos extras: reproducción de videos y visualización de imágenes

```
ShellExecute(
    HWND hwnd,        // NULL → No hay ventana padre (usualmente está bien para apps de consola
    LPCSTR lpOperation, // "open" → Indica que se quiere abrir el archivo
    LPCSTR lpFile,       // Ruta completa del archivo a abrir
    LPCSTR lpParameters, // NULL → No se pasan parámetros adicionales
    LPCSTR lpDirectory,  // NULL → Usa el directorio actual por defecto
    INT nShowCmd         // SW_SHOWNORMAL → Muestra la ventana normalmente
);
```

Syntaxis/Explicación de ShellExecute

Aquí podemos ver cómo usamos ShellExecute en los comentarios, esto lo utilizamos igual en cuanto a parámetros tanto para imágenes como videos. Sin embargo cada uno con diferente dirección en base a cómo estaban guardadas en la computadora de Hugo.

En el código

```
409
410         // Sobrecarga con parámetro extra (no override)
411         void visualizar(int id_episodio){
412             int otra = 0;
413             int swicth = 1;
414             string visualiza;
415             while(swicth){
416                 if (otra){
417                     cout<<"Selecciona una opción válida ¿Quieres visualizar este contenido?(Y/N) "<<endl;
418                 }
419                 else{
420                     cout<<"¿Quieres visualizar este contenido?(Y/N) "<<endl;
421                 }
422                 cin>>visualiza;
423                 if (visualiza == "Y"){
424                     if(ID == 1){
425                         if (id_episodio == 1){
426                             ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Videos_C\\arcane_trailer.mp4", NULL, NULL,SW_SHOWNORMAL);
427                         }
428                         else{
429                             ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Videos_C\\arcane_escena.mp4", NULL, NULL,SW_SHOWNORMAL);
430                         }
431                     }
432                     else if(ID == 2){
433                         if (id_episodio == 1){
434                             ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Videos_C\\got_trailer.mp4", NULL, NULL,SW_SHOWNORMAL);
435                         }
436                         else{
437                             ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Videos_C\\got_escena.mp4", NULL, NULL,SW_SHOWNORMAL);
438                         }
439                     }
```

Para videos

```
146  // Clase Heredada
147  class Pelicula:public Video{
148      public:
149          // Constructor
150          Pelicula(string name,int time,string gen,string tipe,int ine):Video(name, time, gen, tipe, ine){
151          };
152          //Destrcutor
153          ~Pelicula(){};
154          // ------------------------------------------------------------Funciones virtuales puras (Override)-----------------------------------------------------------
155          void mostrarvideo() override{
156              calificacion = promedio();
157              cout<<endl;
158              cout<<"----------------------PELICULA----------------------"<<endl;
159              if(ID == 1){
160                  ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Imagenes_C\\lo_que_el_viento_se_llevo.png", NULL, NULL,SW_SHOWNORMAL);
161              }
162              else if(ID == 2){
163                  ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Imagenes_C\\avatar.png", NULL, NULL,SW_SHOWNORMAL);
164              }
165              else if(ID == 3){
166                  ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Imagenes_C\\endgame.png", NULL, NULL,SW_SHOWNORMAL);
167              }
168              else if(ID == 4){
169                  ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Imagenes_C\\avatar2.png", NULL, NULL,SW_SHOWNORMAL);
170              }
```

Para imágenes

Syntaxis/Explicación de sleep_for() y chrono::duration

```
this_thread::sleep_for(chrono::duration<double>(132.6));
```

this_thread::sleep_for(chrono::duration<tipo de dato>(segundos en el tipo de dato especificado));

En el código

```
    cout<<"¿Quieres visualizar este contenido?(Y/N) "<<endl;
}
cin>>visualiza;
if (visualiza == "Y"){
    if(ID == 1){
        ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Videos_C\\arcane_trailer.mp4", NULL, NULL,SW_SHOWNORMAL);
        this_thread::sleep_for(chrono::duration<double>(132.6));
        ShellExecute(NULL, "open", "C:\\Users\\hugoa\\OneDrive\\Escritorio\\Videos_C\\arcane_escena.mp4", NULL, NULL,SW_SHOWNORMAL);
    }
```

Para episodios o trailers

Investigación: Archivo CSV

Syntaxis/Explicación del archivo (apertura)

Aquí podemos ver que se está mandando un puntero de Películas y Series (que son un arreglo de punteros a objetos Video). Donde Peliculas[]: de punteros a videos que son películas y Series[]: de punteros a videos que son series. Además, de la línea 722 a la 726 se usa algo a lo que le llamamos “fin”. Este es un objeto que viene representando a el archivo “calificaciones.csv”. Por ende, en el if estamos viendo si este no se abre correctamente (porque tiene la negación “!”), despliega el mensaje “Error al abrir calificaciones.csv”. No retorna nada, porque es un void.

En el código

```
719  // ------------------------------------------------------------EXCEL-----------------------------------------------------------------
720  // Función para leer el archivo CSV y cargar las calificaciones a los objetos correspondientes
721  void cargarCalificacionesDesdeCSV(Video* Peliculas[], Video* Series[]) {
722      ifstream fin("calificaciones.csv"); // Se abre el archivo (Leer)
723      if (!fin.is_open()) { // Si el archivo se abrió correctamente.
724          cout << "Error al abrir calificaciones.csv\n";
725          return;
726      }
```

Apertura del CSV

Syntaxis/Explicación del archivo (lectura)

getline(objeto, variable de cadena)

Aquí podemos ver que se genera una variable lineal (tipo string) que usaremos en el while con el getline utilizando el archivo y esta misma variable. Esto es para que lea línea por línea. De manera que si hay un hueco (línea vacía) se la salta (porque no estamos asumiendo que el archivo está limpio siempre. Ahora, busca la primera coma que aparece en la línea (ya que el archivo CSV se separa por comas y así podemos ver qué elementos hay. Entonces, en el caso del if, lo que hace es que si no se encuentra ninguna coma (con npos), no haga nada, pero si se encuentra una, extrae la primera coma. Y todo esto está en el while por lo que se repite varias veces.

En el código

```
727      string linea;
728      while (getline(fin, linea)) { //Lee línea por línea el archivo hasta que termine.
729          if (linea.empty()) continue; //Si la línea está vacía, la salta y continúa con la siguiente.
730
731          size_t coma = linea.find(','); // Busca la primera coma en la línea.
732          if (coma == string::npos) continue;
733
734          string nombre = linea.substr(0, coma); // Extrae el texto antes de la primera coma
```

Leer archivo CSV

Syntaxis/Explicación del archivo (extracción)

Ahora, nombre tiene la información que queremos del CSV. Busca la película si está ahí y con substr nos ayuda a extraer su datos numéricos (el 9 es porque se salta los carácteres que dicen Pelicula). Con el stoi convierte nuevamente la cadena a un entero y el -1 es porque los IDs inician en 1 pero en los arreglos es 0. El of de la línea 738 verifica que el puntero no

se abuelo y que el ID esté en rango. Al llamar a cargarCalificacionesDesdeCSV() (método del objeto), pasa la línea completa del CSV (datos). Sin embargo, el “else if” de la 742 es para hacer un proceso similar si se trata de una serie y no una película. Finalmente cierras el archivo con fin.close().

En el código

```
736         if (nombre.find("Pelicula") != string::npos) {  // Para peliculas
737             int id = stoi(nombre.substr(9)) - 1;  // Extrae la parte numérica
738             if (id >= 0 && id < 10 && Peliculas[id] != nullptr) {
739                 Peliculas[id]->cargarCalificacionesDesdeCSV(linea); // Linea es toda la fila del csv
740             }
741         }
742         else if (nombre.find("Serie") != string::npos) { // Para series
743             int id = stoi(nombre.substr(6)) - 1;  // Extrae la parte numérica
744             if (id >= 0 && id < 5 && Series[id] != nullptr) {
745                 Series[id]->cargarCalificacionesDesdeCSV(linea); // Linea es toda la fila del csv
746             }
747         }
748     }
749     fin.close();
```

Extracción de datos CSV

Syntaxis/Explicación del archivo (vector)

Se creó un vector de tipo double llamado calificaciones (más conveniente que una matriz o algún otro arreglo porque no sabemos qué dimensión pudiera llegar a tomar y no todos los vectores serán del mismo tamaño). En el void cargarCalificacionesDesdeCSV() borra los datos antiguos con clear(), mientras que el stringstream nos ayuda a simular celdas (para separar). De manera que ignora la primera celda (Pelicula) y recorre el resto hasta el final para convertirlos a double y añadirlos al vector con un push_back(). El push_back() lo vemos en el siguiente void, que es básicamente como un append en python (lo manda al final). Finalmente, se calcula el promedio lo retorna. si está vacío va a dar 0.0 (naturalmente).

En el código

```
        vector<double> calificaciones; //almacena muchas calificaciones.

        void cargarCalificacionesDesdeCSV(const string& linea) { // Se define una función que carga calificaciones desde una línea de texto CSV
            calificaciones.clear(); // Limpia el vector calificaciones para borrar cualquier dato anterior antes de cargar los nuevos.
            stringstream ss(linea); // <sstream> // Esto permite leer palabra por palabra (o celda por celda) separadas por comas
            string celda;
            getline(ss, celda, ',');  // Ignorar primera celda (nombre)
            while (getline(ss, celda, ',')) { // Bucle que sigue leyendo las celdas
                if (!celda.empty()) { // Se verifica que la celda no esté vacía
                    calificaciones.push_back(stod(celda));  // (stod)Convierte la celda de texto en un double
                }
            }
        }

        void agregarCalificacion() {
            calificaciones.push_back(calificacion); //<vector>
        }

        double promedio() {
            if (calificaciones.empty()) { // <string> /Se verifica que la calificaciones <vector> no esté vacía
                return 0.0;
            }
            double suma = 0;
            for (double c : calificaciones) suma += c; // Bucle que recorre cada calificación, basicamenete se hace la formula
            double prom = suma / calificaciones.size(); // Calcular el promedio
            return prom;
        }
    };

```

Vector para la base de datos (CSV)

Syntaxis/Explicación del archivo (reescribir archivo/actualizar)

Aquí volvemos a abrir el archivo de salida (fout) que representa a "calificaciones.csv". Despliega un error en texto en la consola en caso de haber alguna falla realizando su apertura. Además, existen 2 ciclos "for": uno para películas y otro para series. Esencialmente cumplen lo mismo para sus respectivas necesidades. Explicaré solo el primer "for" por lo mismo: el de películas. Aquí recorre el arreglo de punteros a películas. Siempre y cuando no esté vació (no se aposición nula), escribe "Película num"; donde num representa al número ID correspondiente. Es ahí donde, para pasarle los datos debe escribir cada calificación correspondiente separadas por comas (le está dando el formato que necesita el CSV). Y entonces da un salto de línea para generar la siguiente película hasta llegar a la última.

Después, en "fout.close()" cierra el archivo CSV. Es entonces que se concluye que en realidad no se está actualizando el archivo, sino que se elimina la información anterior, se agrega lo nuevo al vector y se reescribe el archivo. Por lo que necesitamos limpiar los datos (quitando comas) y volviéndose a poner ya que hayamos extraído y añadido datos. Para

poderlo guardar en un CSV con el mismo nombre (se reescribe). Permitiéndonos tener una "base de datos" para que tenga cierta memoria incluso tras haber cerrado la simulación. Al volver a abrir el código, seguirán teniendo esa información.

En código

```
// Guardar todas las calificaciones en CSV sin borrar lo anterior (agregando calificaciones)
void guardarCalificacionesCSV(Video* Peliculas[], Video* Series[]) {
    ofstream fout("calificaciones.csv"); // Se abre el archivo (Escribir)
    if (!fout.is_open()) {
        cout << "Error al abrir calificaciones.csv para guardar\n";
        return;
    }
    for (int i = 0; i < 10; i++) {
        if (Peliculas[i] != nullptr) {
            fout << "Pelicula " << (i + 1);
            for (double c : Peliculas[i]->calificaciones) {
                fout << "," << c;
            }
            fout << "\n";
        }
    }
    for (int i = 0; i < 5; i++) {
        if (Series[i] != nullptr) {
            fout << "Serie " << (i + 1);
            for (double c : Series[i]->calificaciones) {
                fout << "," << c;
            }
            fout << "\n";
        }
    }
    fout.close();
}
```

Guardar calificaciones (en el CSV)

Tabla de contribución individual al trabajo grupal

Integrante del Equipo	Tareas realizadas en el código	Tareas realizadas en el documento
Hugo	● Corregir código ● Ideas de cómo aplicar estructuras ● Menú ● Esqueleto del proyecto ● Código de imágenes ● Conexión o acciones de las clases con los inputs o outputs (con métodos get, set, etc…) ● Optimización del código ● Videos (conseguir los archivos)	● Casos que harían que el proyecto deje de funcionar ● Archivo UML ● Ejecuciones ● Referencias en APA 7 ● Revisión ortográfica
Jared	● Corregir código ● Ideas de cómo aplicar estructuras ● Código de videos e imágenes ● Imágenes (conseguir los pngs)	● Introducción ● Argumentación del proyecto ● Explicación de investigación ● Explicación de puntos extras ● Referencias en APA 7 ● Revisión ortográfica
Andrea	● CSV ○ Vector ○ Crear archivo ○ Leer archivo ○ Reescribir archivo ● Promedios en base a CSV ● Corregir código ● Ideas de cómo aplicar estructuras ● Conexión del CSV con las clases ● Código de imágenes	● Explicación de puntos extras ● Explicación de investigación ● Corrección de redacción ● Casos que harían que el proyecto deje de funcionar ● Referencias en APA 7 ● Índice y esqueleto del documento

Referencias consultadas

Boucher, A. (s.f.). UML Diagram Expert [Herramienta de inteligencia artificial]. https://www.plantuml.com/plantuml/uml/nLKzRyCW4DtzAr2xE8czPBIYA5BI8LkbI5cgAYBcQY7PO07sq8__NWo6OXn5QfTkvkuTt7iFpZkfi51TKoS9laebSAbGMMCfqPaIuEWZGUWH7Gy5eaoPRyQRgu02QJ1bbGcHJjTHpWf4U7UjmGGhO23u15VYchxHSevLj0sHgmaIP85wA1KN6rkoDP2bQJ1TB0jKSgPhkm5YKmqVg8ZU8tfE2Tfi7CNjk4LFPQTFUiVY7ierSmzKWRB2f8i94HstGipZJhKXOyKTU8iAY_s4_G4aWVtfdDQK0NQxB6mdfWfN0gAoDCotWZT0gEbXLFqhIQpDpr3JigknSNhbboabfgiiyZkRcPr5JYz-uGVlGGWTlEz7ZBhfHm3ot4uWA1XYEOAMIgurq9VfQVnSlgnVJKv1qtA12PREYHonIr3Xx34R_bsQB3xf_xKQGEE9Zk_awiGvrZM6a6tUmBpSJjTHSKLLLtClOt3z0yKlSf0qaDtiVo4GrBbSD2NoO2pbauaHrFeXi_BjfRQn6wOuMtyuPfD3CpHQVyCt_Q0ZmlCN7x4E7_umFsRK8y3GodpJuy71j15sQ6y-yzo_v31cxczYdyLIhn_M3rwM9Da18_fFy0q0

Built In. (s.f). Understanding “Using Namespace STD;” in C++ and Better Alternatives. Retomado de https://builtin.com/articles/using-namespace-std

Cómo usar stringstream en C++ | LabEx. (2017). LabEx. https://labex.io/es/tutorials/cpp-how-to-use-stringstream-in-c-425236

Cplusplus. (2023). std::ifstream::is_open. Cplusplus.com.

GeeksforGeeks. (2018). CSV File Management Using C++. GeeksforGeeks. https://www.geeksforgeeks.org/cpp/csv-file-management-using-c/

GeeksforGeeks. (2021). string::npos in C++ with Examples. GeeksforGeeks. https://www.geeksforgeeks.org/cpp/stringnpos-in-c-with-examples/

https://cplusplus.com/reference/fstream/ifstream/is_open/

OpenAI. (2024). ChatGPT (GPT-4o) [Modelo de lenguaje grande]. https://chat.openai.com/

StackOverflow. (2011). ¿Necesito cerrar un std::fstream? [duplicado]. Retomado de https://stackoverflow.com/questions/4802494/do-i-need-to-close-a-stdfstream

StackOverflow. (2015). std thread sleep_for doesnt work with some chrono::duration. Retomado de https://stackoverflow.com/questions/28704958/std-thread-sleep-for-doesnt-work-with-some-chronoduration

Vive UNIR. (2023). ¿Qué es la programación orientada a objetos? UNIR FP; UNIR. https://unirfp.unir.net/revista/ingenieria-y-tecnologia/programacion-orientada-objetos/

W3Schools.com. (2024). C++ ofstream Class W3schools.com. https://www.w3schools.com/cpp/ref_fstream_ofstream.asp

W3Schools.com. (2025). C++ algorithm find() function W3schools.com. https://www.w3schools.com/cpp/ref_algorithm_find.asp#:~:text=The%20find()%20function%20returns,data%20is%20specified%20by%20iterators

W3Schools.com. (2025). C++ vector push_back() function W3schools.com. https://www.w3schools.com/cpp/ref_vector_push_back.asp

W3Schools.com. (2025). C++Vectors W3schools.com. https://www.w3schools.com/cpp/cpp_vectors.asp#gsc.tab=0&gsc.q=push_back

Windows App Development. (2022). Launching Applications (ShellExecute, ShellExecuteEx, SHELLEXECUTEINFO). Retoamdo de https://learn.microsoft.com/en-us/windows/win32/shell/launch