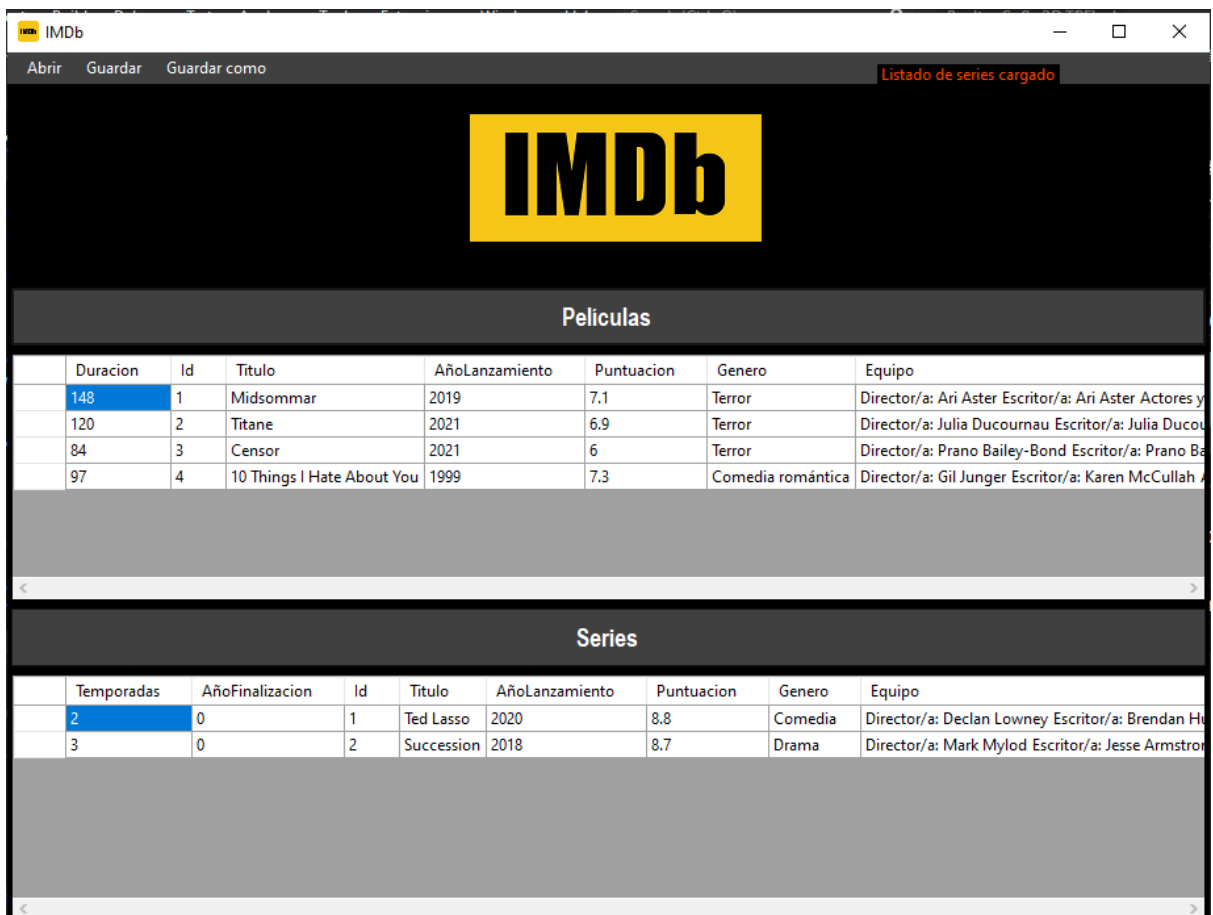


TP 3 Laboratorio II

Aplicación de escritorio para gestionar la base de datos de IMDb.

Internet Movie Database (IMDb) es una base de datos en línea que almacena datos de películas y contenido audiovisual de distintos tipos.

Esta aplicación permite abrir, crear y guardar listas con películas o series permitiendo la creación de nuevos elementos o la modificación de elementos existentes. Además, permite filtrar las listas por distintos criterios mostrando la cantidad de elementos que cumplen con dichos criterios.



The screenshot shows a desktop application window titled 'IMDb'. At the top, there is a menu bar with 'Abrir', 'Guardar', and 'Guardar como'. A status bar at the top right indicates 'Listado de series cargado'. The main content area features the IMDb logo and two sections: 'Películas' and 'Series'. Each section contains a table with movie or series data.

	Duracion	Id	Titulo	AñoLanzamiento	Puntuacion	Genero	Equipo
	148	1	Midsommar	2019	7.1	Terror	Director/a: Ari Aster Escritor/a: Ari Aster Actores y
	120	2	Titane	2021	6.9	Terror	Director/a: Julia Ducournau Escritor/a: Julia Ducou
	84	3	Censor	2021	6	Terror	Director/a: Prano Bailey-Bond Escritor/a: Prano Ba
	97	4	10 Things I Hate About You	1999	7.3	Comedia romántica	Director/a: Gil Junger Escritor/a: Karen McCullah

	Temporadas	AñoFinalizacion	Id	Titulo	AñoLanzamiento	Puntuacion	Genero	Equipo
	2	0	1	Ted Lasso	2020	8.8	Comedia	Director/a: Declan Lowney Escritor/a: Brendan Hu
	3	0	2	Succession	2018	8.7	Drama	Director/a: Mark Mylod Escritor/a: Jesse Armstron

El formulario principal de la aplicación muestra dos DataGridView con la información de la lista de películas por un lado y de series por el otro.

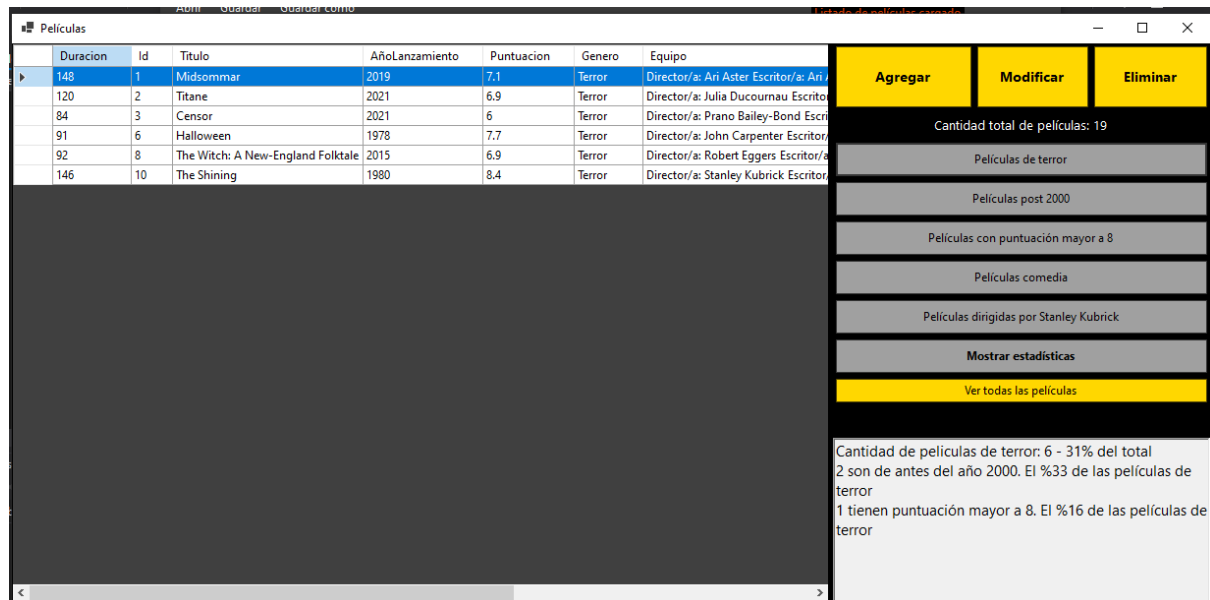


En el menú que se encuentra en la parte superior del formulario hay tres opciones desplegables que permiten abrir archivos permitiendo elegir si se quieren cargar películas o

series, guardar los datos en un archivo o guardarlos eligiendo en donde se quieren guardar y con qué nombre.

En la carpeta TP3 hay dos archivos .json para probar la aplicación con algunas películas y series almacenadas.

Al presionar el botón *Películas* o *Series* en el formulario principal se abre un nuevo formulario que permite agregar, modificar, eliminar series o películas según corresponda. Además permite ver distintas estadísticas.



Duración	Id	Título	AñoLanzamiento	Puntuación	Genero	Equipo
148	1	Midsommar	2019	7.1	Terror	Director/a: Ari Aster Escritor/a: Ari Aster
120	2	Titane	2021	6.9	Terror	Director/a: Julia Ducournau Escritor/a: Julia Ducournau
84	3	Censor	2021	6	Terror	Director/a: Prano Bailey-Bond Escritor/a: Prano Bailey-Bond
91	6	Halloween	1978	7.7	Terror	Director/a: John Carpenter Escritor/a: John Carpenter
92	8	The Witch: A New-England Folktales	2015	6.9	Terror	Director/a: Robert Eggers Escritor/a: Robert Eggers
146	10	The Shining	1980	8.4	Terror	Director/a: Stanley Kubrick Escritor/a: Stanley Kubrick

Agregar **Modificar** **Eliminar**

Cantidad total de películas: 19

Películas de terror

Películas post 2000

Películas con puntuación mayor a 8

Películas comedia

Películas dirigidas por Stanley Kubrick

Mostrar estadísticas

Ver todas las películas

Cantidad de películas de terror: 6 - 31% del total
2 son de antes del año 2000. El 33% de las películas de terror
1 tienen puntuación mayor a 8. El 16% de las películas de terror

En este nuevo formulario se ven nuevamente los datos de las películas o series pero en este caso se puede filtrar la lista utilizando los botones de la derecha. Al presionar alguno de los botones se muestran las estadísticas correspondientes en el recuadro abajo a la derecha y solo se muestran los ítems que cumplan con la condición del filtrado.

Si se presiona el botón *Mostrar estadísticas* se muestra un mensaje con todos los datos recolectados.

Al presionar el botón *Agregar* se abre un nuevo formulario que permite agregar un nuevo elemento a la lista.



Agregar película

Película

Título

Año

Año

Género

Duración

Duración en minutos

Puntuación

Puntuación

Director/a

Director/a

Escritor/a

Escritor/a

Actores/actrices

Actor/actriz 1

Actor/actriz 2

Actor/actriz 3

Agregar

Cancelar

El usuario de la aplicación puede cargar manualmente los campos de una película o serie. Al presionar el botón *Agregar*, si los datos son correctos, se crea el nuevo objeto y se agrega a la lista correspondiente.

Si se selecciona un ítem de la lista y se presiona el botón *Modificar* se abre el mismo formulario que se utiliza para crear un elemento pero en este caso muestra los datos del elemento seleccionado y permite que el usuario los modifique.

Para eliminar un elemento de la lista es necesario seleccionarlo y presionar el botón *Eliminar*.

Al **salir de la aplicación**, si se han cargado o agregado películas, se muestra un mensaje que permite al usuario elegir si guardar los datos de las películas en una base de datos. La base de datos debe ser creada previamente desde el script que se encuentra en la carpeta del TP4. [scriptTPimdb.sql](#)

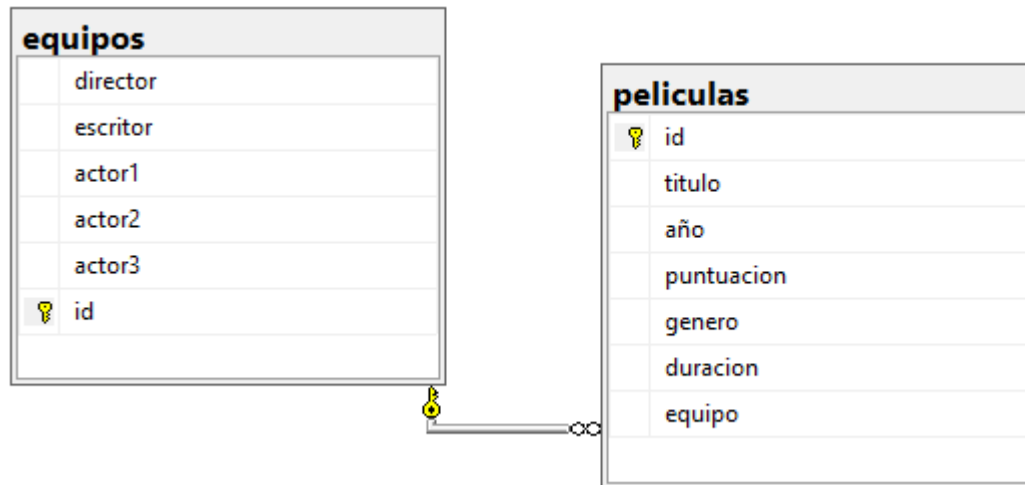
Test de consola de la aplicación: Al iniciar el programa carga los datos del archivo json [basePelículas.json](#) que debe estar en el escritorio del usuario. Muestra todas las películas que contiene y luego permite agregar una manualmente. Vuelve a mostrar las películas y las guarda en un nuevo archivo, también en el escritorio bajo el nombre [basePelículasActualizada.json](#).

Luego muestra las estadísticas de las películas.

Implementación de los temas vistos

- **Clase 10. Excepciones:** Utilicé excepciones en los métodos que abren archivos. En la clase `ArchivoJson` los métodos lanzan la excepción `ArchivoInvalidoException` si el archivo no existe o la extensión no es la correcta y muestran el mensaje correspondiente. Estas excepciones son atrapadas en el formulario principal en los métodos que abren los archivos.
- **Clase 11. Pruebas unitarias:** Desarrollé el proyecto de tests `UnitTest` y dentro de este una clase para testear métodos relacionados al manejo de archivos con la extensión `.Json`. Otra clase de Tests es `TestPelículas` que testea la sobrecarga del operador `==`.
- **Clase 12. Tipos genéricos:** Implementé los tipos genéricos en la interface `IArchivo` para que se puedan guardar y leer datos de distinto tipo.
- **Clase 13. Interfaces:** Creé la interfaz `IArchivo` para declarar los métodos de lectura y guardado de archivos. Implementé los métodos de la interfaz en la clase `ArchivoJson` que a su vez son utilizados luego en el formulario principal para el manejo de datos y archivos.

- **Clase 14. Archivos y serialización:** Utilicé la clase StreamReader en la clase ArchivoJson para leer los datos de un archivo en el método Leer(). También utilicé la clase StreamWriter en el método Serializar() que es llamado tanto en Guardar() como en GuardarComo() para escribir contenido en un archivo. Utilicé la clase JsonSerializer en ArchivoJson y sus métodos Serialize() y Deserialize() para serializar y deserializar en formato json.
- **Clase 15. SQL:** Con el archivo [scriptTPimdb.sql](#) se puede crear la base de datos con dos tablas: películas y equipos.



La clave primaria de equipos es la clave foránea equipo en la tabla películas.

- **Clase 16. Conexión a base de datos:** La clase AccesoDB permiten la conexión con la base de datos y guardar datos en la misma. El método Guardar() se utiliza en el FRmPrincipal en el evento FrmPrincipal_FormClosing() para guardar cada película de la lista de películas en la base de datos correspondiente.
- **Clase 17. Delegados y expresiones lambda:** En el FrmLista utilicé el delegado Action en el método ActualizarContadorPelículas() para encapsular la referencia al método que modifica el control Text.

Además declaré el delegado

```
public delegate void DelegadoMensajeModificacion(string msg); para utilizar como
intermediario con el evento public event DelegadoMensajeModificacion
EventoModificacion; en la clase IMDb.
```

Usé una expresión lambda como argumento del método .Sort() en la clase IMDb en el método MostrarPelículasEstrenadasDespuesDel2000();

```
(Película p1, Película p2) => p1.Puntuacion.CompararSiEsMenor(p2.Puntuacion));
```

- **Clase 18. Hilos:** En el FrmLista abrí un nuevo hilo Task.Run(ComprobarCambios, cancelationToken); para poder ejecutar el método ComprobarCambios() sin que se paralice el formulario. Este método itera y se fija si el contador de películas del form refleja la cantidad de películas de la lista, si difieren llama a ActualizarContadorPelículas();

- **Clase 19. Eventos:** En la clase IMDb declaré el evento
`public event DelegadoMensajeModificacion EventoModificacion;`
El evento es generado en el método `ModificarPelicula()` cada vez que cambia el valor de un atributo. En el `FrmLista` el método que escucha al evento es
`public void NotificarModificacion(string mensaje).` El método y el evento están asociados en el load del formulario.
`imdb.EventoModificacion += NotificarModificacion;`
- **Clase 20. Métodos de extensión:** En la clase `ExtensionFloat` el método
`public static int CompararSiEsMenor(this float numero, float numero2);` es una extensión de `float`. Se utiliza en la clase `IMDb` en el método `.Sort()`
`(Pelicula p1, Pelicula p2) => p1.Puntuacion.CompararSiEsMenor(p2.Puntuacion));`