

<p align="center">Universidad Tecnológica Nacional</p> <p align="center">Facultad Regional Avellaneda</p>	
<p align="center">Técnico Universitario en Programación - Técnico Universitario en Sistemas Informáticos</p>	
<p align="center">Materia: Laboratorio de Programación II</p>	
Apellido:	Fecha: 02-07-2024
Nombre:	Docente ⁽²⁾ : Gustavo Rivas
División:	Nota ⁽²⁾ :

Legajo: Firma⁽²⁾:

Instancia⁽¹⁾: **PP RP**
pSP X RS P FIN

(1) Las instancias validas son: 1^{er} Parcial (**PP**), Recuperatorio 1^{er} Parcial (**RPP**), 2^{do} Parcial (**SP**), Recuperatorio 2^{do} Parcial (**RSP**), Final (**FIN**). Marque con una cruz. **(2)** Campos a ser completados por el docente.

IMPORTANTE:

- **2 (dos) errores en el mismo tema anulan su puntaje.**
- La correcta documentación y reglas de estilo de la cátedra serán evaluadas.
- Colocar sus datos personales en el nombre de la carpeta principal:
Apellido.Nombre.Div. Ej: Pérez.Juan.2D. No sé corregirán proyectos que no sea identificable su autor.
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.
- Aplicar los principios de los 4 pilares de la POO.

Se realizará una aplicación para remover series de nuestro backlog

- **Partir del formulario dado.**

https://drive.google.com/file/d/1L8DZUqs4ZTRTCg9wJZ_LwZpb4o65kpri/view?usp=sharing

- **Se deberá trabajar con .NET 6 y Visual Studio 2022.** Instrucciones para la descarga:

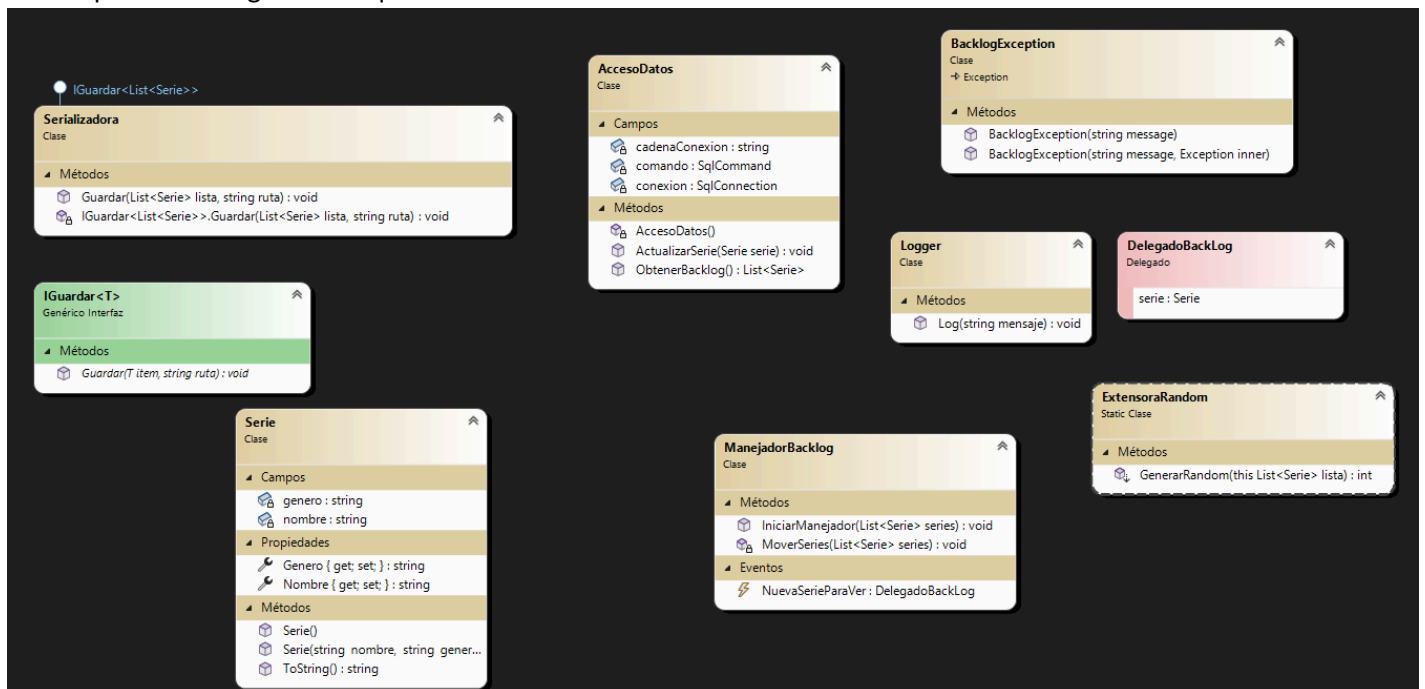
https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/docs/introduccion/entorno-trabajo#instalacion-de-net

- Crear la base de datos 20240701-SP y correr el siguiente script:

```
CREATE DATABASE [20240701-SP];
GO
USE [20240701-SP];
GO
CREATE TABLE [20240701-SP].dbo.series
(
    nombre VARCHAR(255) NOT NULL,
    genero VARCHAR(255) NOT NULL,
    alumno VARCHAR(255)
);
GO

INSERT INTO series (nombre, genero, alumno) VALUES ('The Sopranos', 'Drama criminal', NULL);
GO
INSERT INTO series (nombre, genero, alumno) VALUES ('Vinland Saga', 'Anime seinen', NULL);
GO
INSERT INTO series (nombre, genero, alumno) VALUES ('Brooklyn99', 'Comedia', NULL);
GO
INSERT INTO series (nombre, genero, alumno) VALUES ('The Boys', 'Comedia negra', NULL);
GO
INSERT INTO series (nombre, genero, alumno) VALUES ('Smiling friends', 'Animación', NULL);
GO
```

1. Para comenzar, agregar a la solución un proyecto del tipo Biblioteca de Clases y reproducir el siguiente esquema:



Serie

2. Todos los atributos de la clase van a ser privados.
3. Generar un constructor que reciba e inicialice todos los atributos.
4. Generar un constructor sin parámetros por defecto
5. Tendrá dos propiedades que van a exponer a los atributos de la clase.
6. Sobrecribir el método **ToString** retornando el nombre y el género de la serie.

BackLogException

7. Crear una excepción llamada **BackLogException**. Con dos constructores uno inicializando el mensaje de la excepción y la inner exception y el otro solo inicializando el mensaje, Esta excepción va a ser utilizada para el manejo de archivos

ManejadorBackLog

8. El método **IniciarManejador()** será público, recibirá como argumento un objeto de tipo `List<Serie>`.

- a. Deberá generar una Task parametrizada que se encargará de correr el método

MoverSeries()(Usar operadores lambda) .

ExtensoraRandom

9. Generar un método de extensión para la clase **List<Serie>** que va a recibir la lista de series y en base a su tamaño total generar un número random entre 0 y el tamaño de la lista para retornar un índice random.

ManejadorBackLog - Parte 2

- 10 El método **MoverSeries** será privado y deberá:

- a. Recibir una lista de series por parámetro.
- d. Iterar la lista mientras tenga elementos (teniendo en cuenta el metodo de **ExtensoraRandom**), utilizar el método **ActualizarSerie()** de la clase **AccesoDatos** sobre cada posición.
- i. Dormir el hilo durante 1500 milisegundos.
- ii. Si el evento **serieParaVer** tiene suscriptores, invocarlo, dicha invocación recibirá por parámetro la serie para ver.

AccesoDatos

11. Se deberá instalar el **paquete NuGet System.Data.SqlClient**. Si necesita ayuda pregunte al profesor a cargo.

12. Será la clase que represente la conexión con nuestra base de datos SQL. Dicha clase será estática, y tendrá 2 atributos de tipo **SqlCommand** y **SqlConnection** respectivamente.

13. Los atributos se deben inicializar en el constructor de la clase.

14. El método **ObtenerBacklog()**, cuyo propósito es obtener con un **SELECT** todos los datos de la tabla series para retornar una lista de tipo **Serie**.

15. El método **ActualizarSerie()**, va a recibir un objeto serie por parámetro y va a actualizar el campo alumno de la tabla series con el nombre del alumno que se encuentra dando el examen. 20. Utilizar manejo de excepciones y la clase **Log**

IGuardar<T>

16. El método Guardar() va a recibir como primer parámetro el ítem a guardar y como segundo parámetro la ruta en donde se quiere generar ese archivo.

Serializadora

17. Implementar IGuardar<T>, se va a encargar de guardar una lista de tipo **List<Serie>** en el escritorio en dos formatos distintos.

i. Implementación implícita, se va a encargar de serializar el backlog de series en formato **XML**.

ii. Implementación explícita, se va a encargar de serializar las series para ver en formato **JSON**.

iii. Implementar **BackLogException** donde sea necesario.

Logger

18. El método estático **Log()** va a recibir el mensaje de una excepción por parámetro y va a escribir en un archivo de texto en el escritorio el mensaje junto con la fecha y la hora en ese momento.

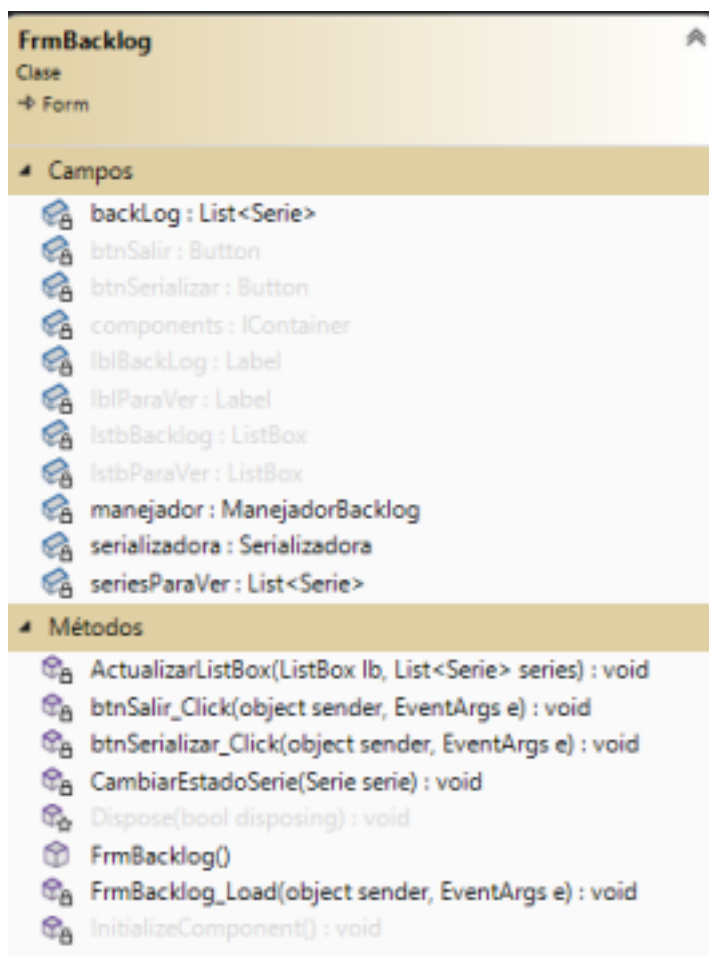
i. Implementar **BackLogException** donde sea necesario.

Test Unitarios

19. Crear un test unitario que pruebe la serialización en formato **XML** y

JSON. 20. Crear un test unitario que pruebe la excepción **BacklogException**.

Formulario



21. El formulario tendrá 4 atributos:

- backLog** de tipo **List<Serie>**.
- seriesParaVer** de tipo **List<Serie>**.
- serializadora** de tipo **Serializadora**.
- manejador** de tipo **ManejadorBacklog**.

22. El constructor deberá instanciar los atributos **seriesParaVer** , **serializadora** y **manejador** . También deberá instanciar el atributo **backLog** utilizando el método **ObtenerBacklog()** de la clase **AccesoDatos**.

23. El manejador del evento **Load** del formulario deberá:

- a. Asignar como manejador del evento **serieParaVer** de **ManejadorBacklog** al método del formulario **CambiarEstadoSerie**.
- b. Inicializar la propiedad **DataSource** del **listBox IstbBacklog** con el atributo **backlog**.
- c. Llamar al método **IniciarManejador** de **ManejadorBacklog** pasándole como argumentos la lista de backlog.

24. Agregar el método **CambiarEstadoSerie** que recibirá una serie y la agrega a la lista de **seriesParaVer** y lo removerá de la lista de **backLog** .

- a. Agregar la serie a **seriesParaVer** y removerla de **backLog** .
- b. Actualizar **IstbBacklog**. Para esto, setear la propiedad **DataSource** del **listBox** en **null** y luego reasignar la lista de **backLog** al **DataSource**. Hacer lo mismo con **IstbParaVer**.
- c. La modificación de los **listBox** se deberá realizar desde el hilo principal (en el que se crearon los controles).

25. El botón serializar será el encargado de serializar a XML la lista de backlog y en JSON la lista de series para ver

26. El botón Salir será el encargado de cerrar el formulario y controlar que los hilos se hayan cerrado correctamente

Al finalizar, colocar la carpeta de la Solución completa en un archivo comprimido que deberá tener como nombre *Apellido.Nombre.zip* y subir esta solución a github.