~~ESTO ES UN BORRADOR BORRADOR BORRADOR~~

REQUISITOS QUE PIDE EL TP

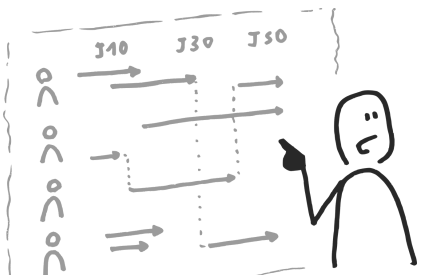
1. Programación Orientada a Objetos (POO):
Uso adecuado de clases, atributos y métodos.
2. Herencia, Agregación y Composición:
Incluir al menos una jerarquía de clases con herencia, y ejemplos claros de agregación y composición (opcionales).
3. Encapsulamiento:
Todos los atributos deben ser privados o protegidos, con acceso a través de propiedades.
4. Listas:
Utilización de listas para la gestión de colecciones de objetos.
5. Interfaces
Implementación de al menos una interfaz con distintos comportamientos definidos por las clases.
6. Enumeraciones (enum)
Uso de al menos un enum para representar un conjunto fijo de valores.
7. Serialización y deserialización en JSON:
Debe haber persistencia de información en archivos .json.
8. Conexión a una API externa:
Consumir información desde una API pública o simulada (puede ser una API real o una creada para el ejercicio).
9. Diagrama UML de clases del sistema
Entregar un diagrama UML que represente las clases del sistema, sus relaciones (herencia, agregación, composición), interfaces implementadas, atributos y métodos principales.
10. Interfaz de usuario por consola:
Menú interactivo que permita navegar las distintas funcionalidades del sistema de forma clara y ordenada.

IDEA VIDEOJUEGOS

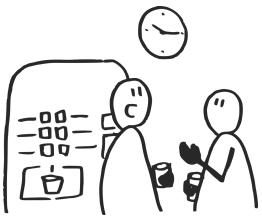
generar una colección de videojuegos

el usuario deberia poder hacer lo siguiente :

- **AGREGAR** juegos manualmente:
- Ingresando el ***nombre, su plataforma, el año de lanzamiento***
- que pueda **BUSCAR** juegos en la API, (chequear apis disponibles)
- **RECIBIR** automáticamente datos como ***nombre, plataforma, año cuando*** busque por título.
- que pueda manipular : **ELIMINAR o EDITAR** juegos:
- Modificar algun detalle del juego
- (en el menu que tenga algun detalle disntintivo / atractivo relacionado al videojuego)



~~ESTO ES UN BORRADOR BORRADOR BORRADOR~~



////// considerar hacer algo relacionado a videojuegos retros, donde es màs simple tener una colección actualizada //////

////// si es retro se podría agregar como una galeria muy muy simple tipo museo de lo retro o de los juegos que mas famosos son o los que tienen mejor valoración ò simplemente los màs iconicos de los retros //////

MENU PARA LA COLECCIÓN:

todo con un while / switch case / console. writeline / readkey

mensaje de bienvenida a la colección

opciones del menu :
Gestionar una colección personal de videojuegos retro.
BUSCAR información de juegos en la API de IGDB.
poder VER la colección en un Museo Retro interactivo.
GUARDAR los datos localmente en formato JSON.

1. AÑADIR JUEGO MANUALMENTE
2. BUSCAR JUEGO EN API
3. VER COLECCIÓN
4. MUSEO RETRO :)
5. GUARDAR Y SALIR

(buscar como poner lineas para que quede bien el recuadro del menu tipo dibujito en consola)

todo con un while / switch case / console. writeline / readkey

recordar librerias que no deberian faltar (chequear si falta algo) :

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Text;
using System.Text.Json;
```

~~ESTO ES UN BORRADOR BORRADOR BORRADOR~~

DIAGRAMA UML :

sitios web para hacer el diagrama:
smartdraw (RECOMENDADO POR CECCHI)

la colección

crear usuario y login

tiene la lista de videojuegos

tiene metodos de
agregar videojuego
guardar la colección
cargar colección

relación: colección
contiene videojuegos

nombre de la clase

videojuego

propiedades y su privacidad

– *string nombre*
– *int anio*
– *valoración (double? o decimal ??)*

metodos

+ videojuegos (string , int , plataforma)

la api que vamos a usar le da la información a los videojuegos con el http
con ella se puede BUSCAR el videojuego

metodo???

LA API NO SE AGREGA EN EL UML

la api va a tener la data de los nombres de los videojuegos, la plataforma en donde se podia jugar (lista) y la fecha de lanzamiento

~~ESTO ES UN BORRADOR BORRADOR BORRADOR~~

1. Programación Orientada a Objetos (POO):
Uso adecuado de clases, atributos y métodos.

2. Herencia, Agregación y Composición:
relacion de agregación con colección y videojuego
relación de dependencia con el api y el juego

3. Encapsulamiento:
estan encapsulados los atributos importantes

4. Listas:
la lista de videojuego de la colección

5. Interfaces
el http de la api
el metodo buscar

6. Enumeraciones (enum)
Plataformas { NES, SNES, SEGA, Arcade }

7. Serialización y deserialización en JSON:

8. Conexión a una API externa:
https://api-docs.igdb.com/#getting-started

9. Diagrama UML de clases del sistema
se usa web para uml

10. Interfaz de usuario por consola:

1. *Añadir juego manualmente*
2. *Buscar juego en API*
3. *Ver colección*
4. *Museo Retro*
5. *Guardar y salir*

SEGUN EL PROFE, CUMPLIRIA CON LO PEDIDO