UNT - FRSN

TRABAJO PRÁCTICO FINAL

PROGRAMACIÓN 2

Integrantes del grupo:

Maria del Pilar Sanchez
Maria Candela Caraballo
Maria Luz Opazo

Docentes a Cargo:

David Cecchi
Osvaldo Torrez

CONSIGNAS DEL TP GRUPAL

Lenguaje: C# – Aplicaciones de Consola
Modalidad: Trabajo en grupos de hasta 3 integrantes
Objetivo: Aplicar todos los conceptos aprendidos a lo largo de la materia en el desarrollo de una aplicación funcional, original y completa.

Cada grupo deberá desarrollar una aplicación de consola en C# que simule un sistema real de su elección,
integrando los siguientes conceptos trabajados durante el cuatrimestre:

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..

CONSIGNAS DEL TP GRUPAL

Restricciones

• Prohibido reutilizar proyectos realizados durante las clases.
La aplicación debe ser completamente original.
• Temas únicos:
No se permitirá que dos trabajos sean sistemas con temáticas similares.
Ejemplo: no puede haber dos sistemas de gestión de empleados, dos sistemas de librería, etc.

Entrega
• Fecha límite: 17/06/2025
• Forma de entrega: subir la solución completa (código fuente) comprimida en .zip a la plataforma (no se recibirán soluciones por correo).
También se deberá subir a la plataforma el diagrama UML.

El proyecto será evaluado en función de:

• Aplicación correcta de los temas obligatorios.
• Originalidad del sistema.
• Buenas prácticas de programación (nombres, estructura, comentarios).
• Funcionalidad completa y correcta ejecución.
• Defensa oral del proyecto (explicación del código)

DESARROLLO DE LA APLICACIÓN

"GESTOR DE JUEGOS"

Nombre COMPLETO del Sistema: Gestor Integral de Colección de Juegos

Descripción General:

El Gestor de Juegos es una aplicación de consola desarrollada en C# que permite a los usuarios organizar y gestionar su colección personal de juegos (de mesa, consola y realidad virtual)
con la opción de revisar su historia, y modificar el estado del juego en la sala de juego personal.

Características Técnicas Destacadas (según requisitos de la consigna)

- **POO Avanzada:**

Uso de clases abstractas (Juego) y métodos polimórficos (mostrarDetalles() implementado de forma única en cada subclase).
Encapsulamiento estricto: todos los atributos son privados/protegidos con acceso mediante propiedades.

- **Relaciones entre Objetos:**

Agregación: Un Usuario contiene una colección de objetos Juego.
Composición: Cada Usuario posee una SalaJuego exclusiva que no existe independientemente.
nterfaz: IChequeoEstado implementada por SalaJuego para gestionar estados de juego (cumpliendo con el requisito de interfaces).

DESARROLLO DE LA APLICACIÓN

"GESTOR DE JUEGOS"

Funcionalidades Clave:

- **Autenticación de Usuarios:**

Registro e inicio de sesión con validación de credenciales.

- **Gestión de Colección:**

Añadir/eliminar juegos desde un catálogo inicial cargado via API.
Visualización detallada por categorías.

- **Sistema de Historial:**

Registro de cambios de estado con comentarios.

- **Sala de Juegos Virtual:**

Simulación de sesiones de juego con gestión de estados (iniciar/pausar/finalizar)

DESARROLLO DE LA APLICACIÓN

"GESTOR DE JUEGOS"

Diagrama UML Incluido:

Representación completa de:
Jerarquía de herencia de clases de juegos.
Relaciones de agregación/composición.
Implementación de interfaces.
Enumeraciones utilizadas.

Desiciones clave del trabajo que tomamos en grupo:

- utilizamos un REPOSITORIO grupal en GITHUB para llevar adelante el seguimiento de proyecto, y para llevar tambien un DIARIO DE AVANCES en donde documentamos cada avance del proyecto

LINK DEL REPOSITORIO

- Debido a que encontramos Apis PAGAS, tomamos la desición de poder **SIMULAR - personalizar** segun las necesidades del trabajo:

se utilizó la herramienta ***online Mocky*** (**LINK**) para simular una API REST. Esta herramienta nos permitió probar y consumir datos desde una URL como si provinieran de un servidor real, sin necesidad de contar con un backend funcional en esta etapa.

Pasos que se generaron paa trabajar con Online Mocky

- Creamos un JSON con datos simulados
- Pegamos ese contenido en la plataforma de Mocky.
- Mocky generó una URL pública que responde con ese JSON.
- Usamos esa URL en nuestras pruebas, como si fuera una API real.


UTN
San Nicolás