



Estructura del Proyecto: Gestor de Colección de Videojuegos Retro

Basado en los requisitos del trabajo práctico final, te propongo la siguiente estructura para tu proyecto "Gestor de Colección de Videojuegos Retro":

1. Diagrama UML de Clases (Resumen)

text

Copy Download



2. Estructura de Archivos del Proyecto

text

```
/GestorColeccionRetro
├── /Models
│   ├── Usuario.cs
│   ├── Videojuego.cs
│   ├── Museo.cs
│   ├── Plataforma.cs (enum)
│   └── Genero.cs (enum)
├── /Services
│   ├── AutenticacionService.cs
│   ├── APIClientService.cs (para IGDB API)
│   ├── ColeccionService.cs
│   ├── MuseoService.cs
│   └── SerializadorService.cs
├── /Interfaces
│   ├── IAPIClient.cs
│   └── ISerializador.cs
├── Program.cs (Main)
├── coleccion.json (archivo de datos)
└── usuarios.json (archivo de datos)
```

3. Implementación Detallada

Models/Usuario.cs

csharp

 Copy  Download

```
public class Usuario
{
    public string UserId { get; set; }
    public string Username { get; set; }
    public string Password { get; set; } // En un proyecto real, debería estar encriptado
    public List<Videojuego> Coleccion { get; set; } = new List<Videojuego>();
}
```

Models/Videojuego.cs

csharp

 Copy  Download

```
public class Videojuego
{
    public string JuegoId { get; set; }
    public string Nombre { get; set; }
    public Plataforma Plataforma { get; set; }
    public int Año { get; set; }
    public Genero Genero { get; set; }
    public string Descripcion { get; set; }
    public string PortadaUrl { get; set; }
}

public enum Plataforma { NES, SNES, Genesis, Arcade, GB, GBA, PS1, Otro }
public enum Genero { Plataformas, Aventura, RPG, Lucha, Puzzle, Deportes, Estrategia, Otro }
```

Services/AutenticacionService.cs

csharp

 Copy  Download

```
public class AutenticacionService
{
    private List<Usuario> _usuarios;
    private readonly ISerializador _serializador;

    public AutenticacionService(ISerializador serializador)
    {
        _serializador = serializador;
        _usuarios = _serializador.CargarUsuarios() ?? new List<Usuario>();
    }

    public Usuario Login(string username, string password)
    {
        return _usuarios.FirstOrDefault(u => u.Username == username && u.Password == password);
    }

    public bool Registrar(Usuario nuevoUsuario)
    {
        if (_usuarios.Any(u => u.Username == nuevoUsuario.Username))
            return false;

        _usuarios.Add(nuevoUsuario);
        _serializador.GuardarUsuarios(_usuarios);
        return true;
    }
}
```

Services/APIClientService.cs

csharp

```
public class APIClientService : IAPIClient
{
    private readonly HttpClient _httpClient;
    private const string ApiUrl = "https://retro.gg/api";

    public APIClientService()
    {
        _httpClient = new HttpClient();
    }

    public async Task<Videojuego> BuscarJuego(string nombre)
    {
        // Implementación para buscar juegos en la API
        // Parsear respuesta JSON a objeto Videojuego
    }
}
```

Program.cs (Menú Principal)

csharp

 Copy  Download

```
class Program
{
    static async Task Main(string[] args)
    {
        // Configuración inicial e inyección de dependencias
        var serializador = new SerializadorService();
        var authService = new AutenticacionService(serializador);
        var apiClient = new APIClientService();
        var coleccionService = new ColeccionService(serializador);
        var museoService = new MuseoService();

        Usuario usuarioActual = null;

        while (true)
        {
            Console.Clear();
            Console.WriteLine("=== GESTOR DE COLECCIÓN RETRO ===");

            if (usuarioActual == null)
            {
                Console.WriteLine("1. Iniciar sesión");
                Console.WriteLine("2. Registrarse");
                Console.WriteLine("3. Salir");

                var opcion = Console.ReadLine();
```

```
var opcion = Console.ReadLine();

switch (opcion)
{
    case "1":
        usuarioActual = MostrarLogin(authService);
        break;
    case "2":
        MostrarRegistro(authService);
        break;
    case "^":
        return;
    }
}
else
{
    Console.WriteLine($"Bienvenido, {usuarioActual.Username}!");
    Console.WriteLine("1. Mi colección");
    Console.WriteLine("2. Buscar juego en API");
    Console.WriteLine("3. Museo Retro");
    Console.WriteLine("4. Cerrar sesión");

    var opcion = Console.ReadLine();

    switch (opcion)
    {
        case "1":
            await MostrarMenuColeccion(usuarioActual, coleccionService);
            break;
        case "2":
            await BuscarEnAPI(usuarioActual, apiClient, coleccionService);
            break;
        case "3":
            museoService.MostrarMuseo(usuarioActual.Coleccion);
            break;
        case "4":
            usuarioActual = null;
            break;
    }
}
}
```

4. Características que Cumplen los Requisitos

1. POO: Uso de clases como Usuario, Videojuego, Museo, etc.
2. Herencia/Agregación: Museo contiene una lista de Videojuegos (agregación)
3. Encapsulamiento: Todos los atributos son privados con propiedades públicas
4. Listas: List<Usuario>, List<Videojuego> para gestionar colecciones
5. Interfaces: IAPIClient, ISerializador para diferentes implementaciones
6. Enums: Plataforma y Genero como enumeraciones
7. JSON: Serialización de usuarios y colección en archivos JSON
8. API Externa: Conexión a retro.gg/api para buscar información
9. UML: Diagrama proporcionado arriba
10. Interfaz Consola: Menú interactivo con todas las funcionalidades

5. Recomendaciones Adicionales

1. Validaciones: Añadir validaciones en los inputs del usuario
2. Manejo de errores: Try-catch para operaciones con API y archivos
3. Comentarios: Documentar cada clase y método importante
4. Testing: Probar especialmente la serialización y conexión a API
5. UI Mejorada: Usar colores y ASCII art para el museo retro