

REST API_VictorSuros.

REST API_VictorSuros.....	1
1. Levanta tu servidor API REST:.....	3
2. No generes un conjunto de datos aleatorio. Crea el tuyo, de lo que quieras. En JSON.....	8
3. Realiza dos operaciones por cada método del CRUD:.....	9
4. Muestra su funcionamiento con Postman. Swagger no nos interesa en este ejercicio. PostmanCollection.....	11
5. Prueba a acceder a dos operaciones desde un cliente que se conecte al servidor API rest que has levantado(TestClient/GameApiClient). 12	
6. ¿Pueden dos clientes hacer peticiones al servidor simultáneas?.....	13

1. Levanta tu servidor API REST.:

- a. Para crear el modelo inicial de la API:

```
dotnet new webapi -n TestApi
```

- b. Para crear el controlador de la API(-name cambiado a GameController):

```
dotnet tool install -g dotnet-aspnet-codegenerator  
add package Microsoft.VisualStudio.Web.CodeGeneration.Design  
dotnet add package Microsoft.EntityFrameworkCore.Design  
dotnet aspnet-codegenerator controller -name HelloController -async -api -outDir Controllers
```

- c. Modificar GameController para gestionar las operaciones a la API:

```

// POST: Add a single game
[HttpPost("add")]
0 references
public async Task<IActionResult> AddGame([FromBody] Game game) ...

// POST: Add multiple games
[HttpPost("add-multiple")]
0 references
public async Task<IActionResult> AddGames([FromBody] List<Game> newGames) ...

// GET: Retrieve a single game by ID
[HttpGet("{id}")]
0 references
public async Task<IActionResult> GetGame(int id) ...

// GET: Retrieve all games
[HttpGet]
0 references
public async Task<IActionResult> GetAllGames() ...

// PUT: Update a game by ID
[HttpPut("{id}")]
0 references
public async Task<IActionResult> UpdateGame(int id, [FromBody] Game updatedGame) ...

// PUT: Update multiple games
[HttpPut("update-multiple")]
0 references
public async Task<IActionResult> UpdateMultipleGames([FromBody] List<Game> updatedGames) ...

// DELETE: Delete a game by ID
[HttpDelete("{id}")]
0 references
public async Task<IActionResult> DeleteGame(int id) ...

// DELETE: Delete multiple games by IDs
[HttpDelete("delete-multiple")]
0 references
public async Task<IActionResult> DeleteMultipleGames([FromBody] List<int> gameIds) ...
}

```

- d. Crear una carpeta llamada Models y crear una clase:

```

0 references
public class Game
{
    0 references
    public int Id { get; set; }
    0 references
    public string Name { get; set; }
    0 references
    public string Genre { get; set; }
    0 references
    public decimal Price { get; set; } // Game price
    0 references
    public List<DLC> Dlcs { get; set; } = new List<DLC>(); // List of DLCs
}

2 references
public class DLC
{
    0 references
    public string Name { get; set; }
    0 references
    public decimal Price { get; set; }
}

```

- e. Crear una carpeta llamada Services y crear la clase que inicia la API con los datos de test-data.json e inicializa el id al final de la lista inicial.

```

0 references
public GameService()
{
    LoadData();
    InitializeIds();
}

1 reference
private void LoadData()
{
    string filePath = "test-data.json";
    if (File.Exists(filePath))
    {
        string jsonData = File.ReadAllText(filePath);
        Games = JsonConvert.DeserializeObject<List<Game>>(jsonData) ?? new List<Game>();
    }
}

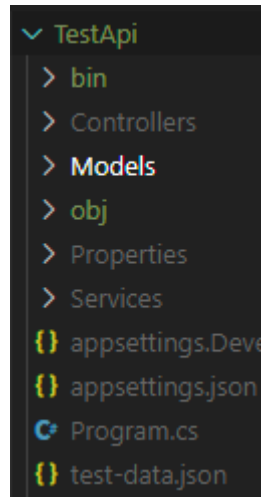
// Initialize the next available GameId
1 reference
private void InitializeIds()
{
    if (Games.Any())
    {
        _nextGameId = Games.Max(g => g.Id) + 1; // Start after the highest Game ID
    }
    else
    {
        _nextGameId = 1;
    }
}

// Method to get the next Game ID
0 references
public int GetNextGameId()
{
    return _nextGameId++;
}

```

2. No generes un conjunto de datos aleatorio. Crea el tuyo, de lo que quieras. En JSON.

La carpeta RestApi/TestApi contiene un Json : test-data.json que se inicializa con la Api.



3. Realiza dos operaciones por cada método del CRUD:

a. Create:

```
// POST: Add a single game  
[HttpPost("add")]  
0 references  
public async Task<IActionResult> AddGame([FromBody] Game game) ...  
  
// POST: Add multiple games  
[HttpPost("add-multiple")]  
0 references  
public async Task<IActionResult> AddGames([FromBody] List<Game> newGames) ...
```

b. Read:

```
// GET: Retrieve a single game by ID  
[HttpGet("{id}")]  
0 references  
public async Task<IActionResult> GetGame(int id) ...  
  
// GET: Retrieve all games  
[HttpGet]  
0 references  
public async Task<IActionResult> GetAllGames() ...
```

c. Update:

```
// PUT: Update a game by ID
[HttpPut("{id}")]
0 references
public async Task<IActionResult> UpdateGame(int id, [FromBody] Game updatedGame) ...

// PUT: Update multiple games
[HttpPut("update-multiple")]
0 references
public async Task<IActionResult> UpdateMultipleGames([FromBody] List<Game> updatedGames) ...
```

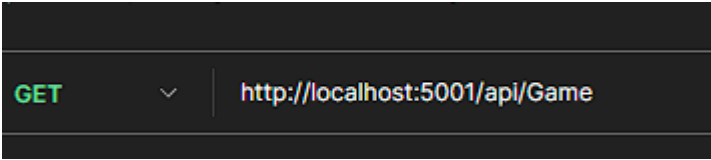
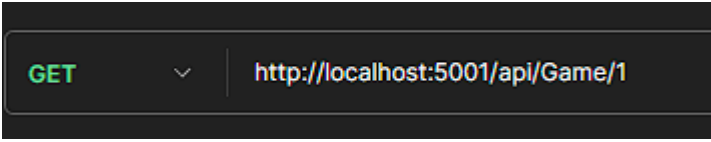
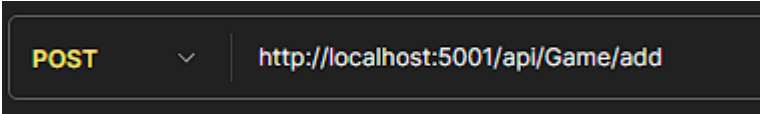
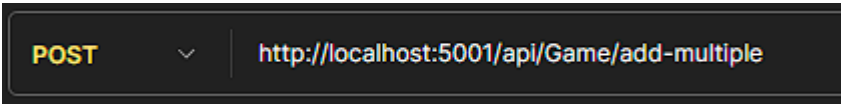
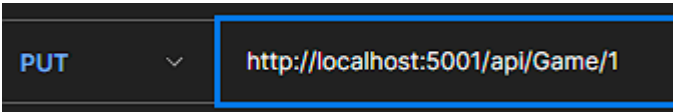
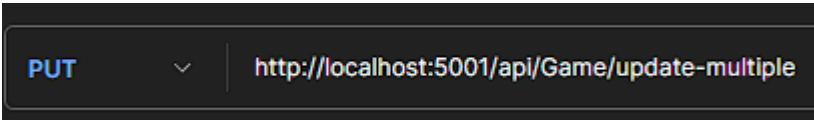
d. Delete:

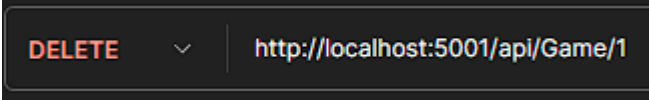
```
// DELETE: Delete a game by ID
[HttpDelete("{id}")]
0 references
public async Task<IActionResult> DeleteGame(int id) ...

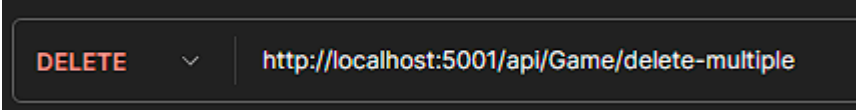
// DELETE: Delete multiple games by IDs
[HttpDelete("delete-multiple")]
0 references
public async Task<IActionResult> DeleteMultipleGames([FromBody] List<int> gameIds) ...
```

4. Muestra su funcionamiento con Postman. Swagger no nos interesa en este ejercicio.

[PostmanCollection](#)

- a.  Devuelve un Json con todos los datos guardados.
- b.  Devuelve un Json con todos los datos del objeto que coincide con el Id.
- c.  Crea un nuevo objeto mediante un Json.
- d.  Crea múltiples objetos nuevos mediante una lista en Json.
- e.  Modifica el objeto que coincide con el Id introducido, mediante un Json.
- f.  Modifica varios objetos mediante una lista en Json.

- g.  Borra el objeto que coincida con el Id introducido.

- h.  Borra los objetos cuyo Id coincida con los Ids introducidos en la lista Json.

5. Prueba a acceder a dos operaciones desde un cliente que se conecte al servidor API rest que has levantado(TestClient/GameApiClient).

- a. Una operación de cada tipo

```
1 reference
private static async Task AddGame(string apiUrl) ...

// Method to GET all games
4 references
private static async Task GetAllGames(string apiUrl) ...

// Method to PUT (Update) a game
1 reference
private static async Task UpdateGame(string apiUrl, int gameId) ...

// Method to DELETE a game
1 reference
private static async Task DeleteGame(string apiUrl, int gameId) ...
```

6. ¿Pueden dos clientes hacer peticiones al servidor simultáneas?

- a. Todos los métodos de GameController y GameApiClient son asíncronos por lo que debería de permitir la conexión simultánea de varios clientes.