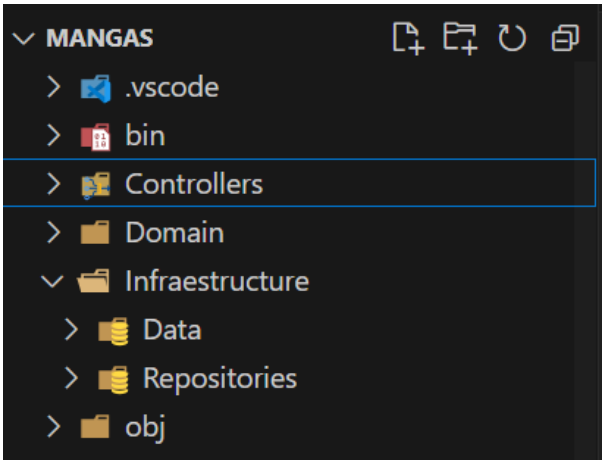
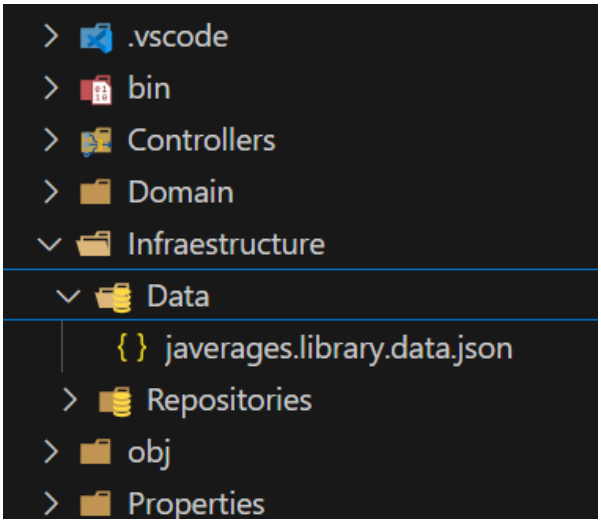
My First Access Data Class

¡Vamos a incrementar la funcionalidad de tu API!

1. **Incorporación de la Capa de Infraestructura**
    - Abre tu proyecto y en la raíz del proyecto crea una carpeta con el nombre Infrastructure.
    - Dentro de la carpeta Infrastructure crea una subcarpeta con el nombre Repositories.
    - Dentro de la carpeta Infrastructures crea una subcarpeta con el nombre Data.



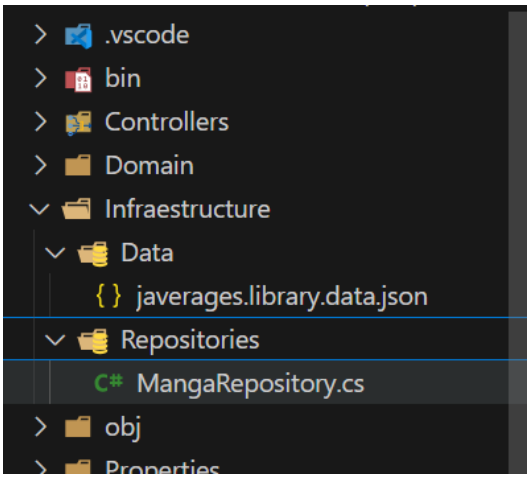
2. Configura la capa de infraestructura para gestionar el acceso a datos.
    - Dentro de la carpeta Data copia el archivo javerages.library.data.json.



En el archivo appsettings.json agrega una nueva entrada con el nombre dataBank, en esta entrada configuraremos la dirección donde hayamos guardado nuestro archivo javerages.library.data.json; para este caso será "*Infrastructure/Data*":

```
{ } appsettings.json > ...
1   {
2     "Logging": {
3       "LogLevel": {
4         "Default": "Information",
5         "Microsoft.AspNetCore": "Warning"
6       }
7     },
8     "AllowedHosts": "*",
9     "dataBank": "Infraestructure\\Data\\javerages.library.data.json"
10  }
11
```

Ahora dentro de la carpeta Repositories, crea una nueva clase con el nombre MangaRepository.cs.



Para poder acceder a la entrada dataBank que acabamos de configurar en el archivo appsettings.json es necesario inyectar la configuración en el constructor de la clase MangaRepository.cs.

Dentro de la clase MangaRepository.cs configura la extracción de los datos almacenados en el archivo javerages.library.data.json. Para ello crea un método con el nombre LoadData e invócalo en el constructor de la clase.

Dentro de la clase MangaRepository.cs crea un nuevo método con el nombre GetAll, este método debe retornar un listado de objetos de tipo Manga.cs.

Dentro de la clase MangaRepository.cs crea un nuevo método con el nombre GetById, este método debe retornar un objeto de tipo Manga.cs.

Dentro de la clase MangaRepository.cs crea un nuevo método con el nombre Add, este método no debe retornar nada.

Dentro de la clase MangaRepository.cs crea un nuevo método con el nombre Update, este método no debe retornar nada.

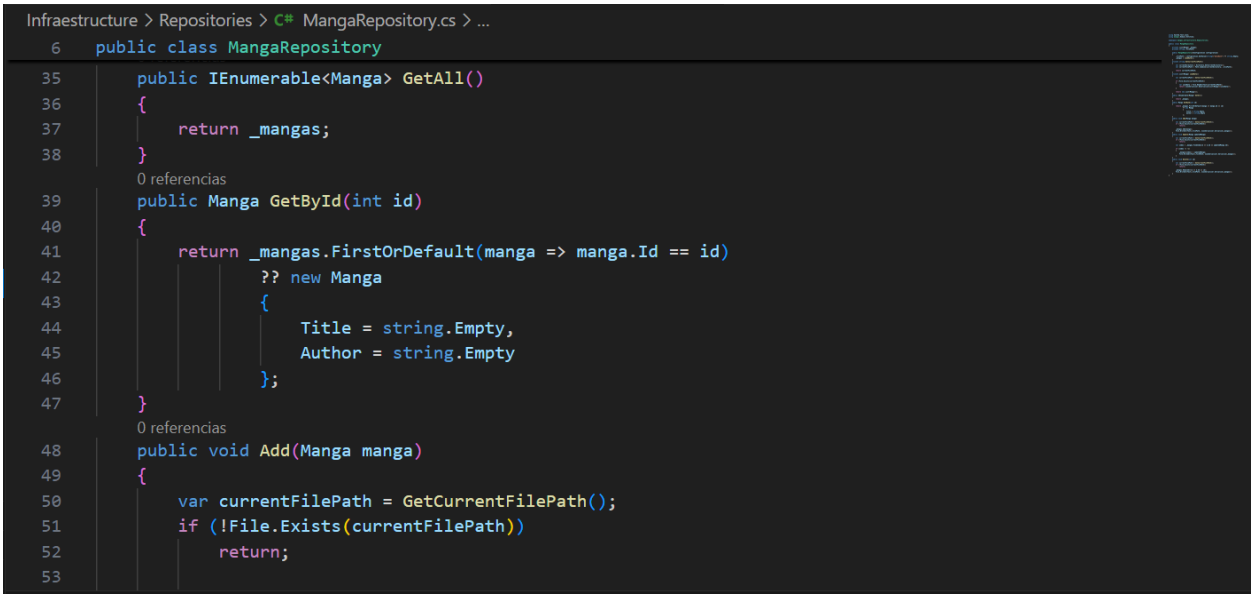
Dentro de la clase MangaRepository.cs crea un nuevo método con el nombre Delete, este método no debe retornar nada.

Infraestructure > Repositories > C# MangaRepository.cs > ...

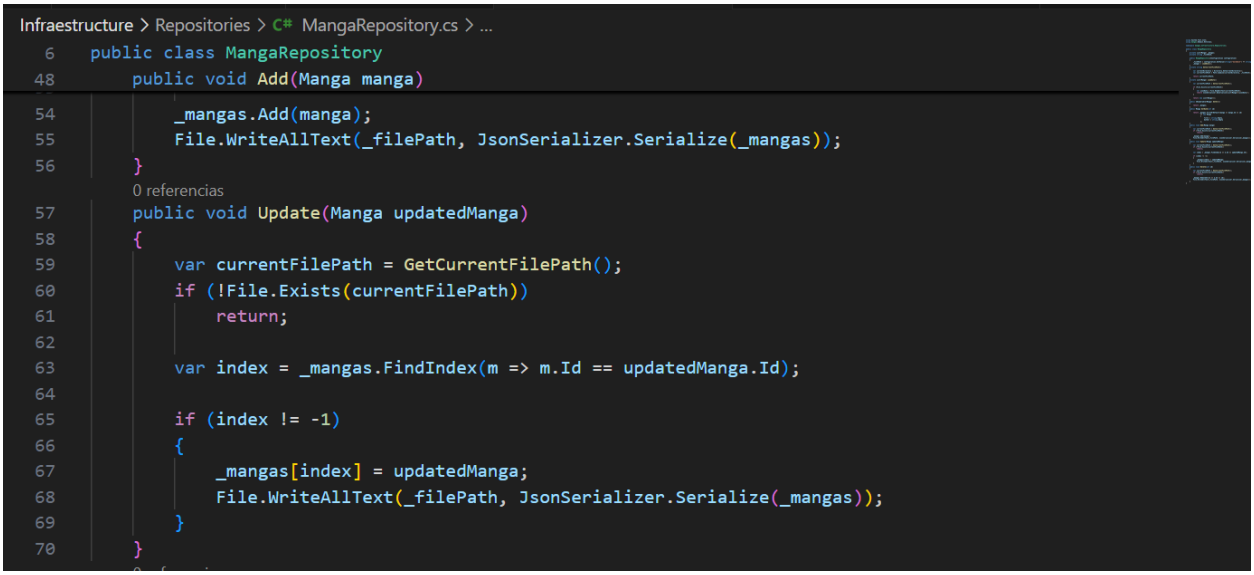
```
using System.Text.Json;
using mangas.Domain.Entities;

namespace mangas.Infrastructure.Repositories;

1 referencia
public class MangaRepository
{
    10 referencias
    private List<Manga> _mangas;
    5 referencias
    private string _filePath;

    0 referencias
    public MangaRepository(IConfiguration configuration)
    {
        _filePath = configuration.GetValue<string>("dataBank") ?? string.Empty;
        _mangas = LoadData();
    }
    4 referencias
    private string GetCurrentFilePath()
    {
        var currentDirectory = Directory.GetCurrentDirectory();
```

Infraestructure > Repositories > C# MangaRepository.cs > ...

```
public class MangaRepository
    private string GetCurrentFilePath()
        var currentDirectory = Directory.GetCurrentDirectory();
        var currentFilePath = Path.Combine(currentDirectory, _filePath);

        return currentFilePath;
    }
    1 referencia
    private List<Manga> LoadData()
    {
        var currentFilePath = GetCurrentFilePath();

        if (File.Exists(currentFilePath))
        {
            var jsonData = File.ReadAllText(currentFilePath);
            return JsonSerializer.Deserialize<List<Manga>>(jsonData)!;
        }

        return new List<Manga>();
    }
    0 referencias
```



```
Infraestructure > Repositories > C# MangaRepository.cs > ...
  6   public class MangaRepository
 35       public IEnumerable<Manga> GetAll()
 36       {
 37           return _mangas;
 38       }
          0 referencias
 39       public Manga GetById(int id)
 40       {
 41           return _mangas.FirstOrDefault(manga => manga.Id == id)
 42                   ?? new Manga
 43                   {
 44                       Title = string.Empty,
 45                       Author = string.Empty
 46                   };
 47       }
          0 referencias
 48       public void Add(Manga manga)
 49       {
 50           var currentFilePath = GetCurrentFilePath();
 51           if (!File.Exists(currentFilePath))
 52               return;
 53
```



```
Infraestructure > Repositories > C# MangaRepository.cs > ...
  6   public class MangaRepository
 48       public void Add(Manga manga)
 54           _mangas.Add(manga);
 55           File.WriteAllText(_filePath, JsonSerializer.Serialize(_mangas));
 56       }
          0 referencias
 57       public void Update(Manga updatedManga)
 58       {
 59           var currentFilePath = GetCurrentFilePath();
 60           if (!File.Exists(currentFilePath))
 61               return;
 62
 63           var index = _mangas.FindIndex(m => m.Id == updatedManga.Id);
 64
 65           if (index != -1)
 66           {
 67               _mangas[index] = updatedManga;
 68               File.WriteAllText(_filePath, JsonSerializer.Serialize(_mangas));
 69           }
 70       }
```

```
Infraestructure > Repositories > C# MangaRepository.cs > ...
  6   public class MangaRepository
          0 referencias
 71       public void Delete(int id)
 72       {
 73           var currentFilePath = GetCurrentFilePath();
 74           if (!File.Exists(currentFilePath))
 75               return;
 76
 77           _mangas.RemoveAll(m => m.Id == id);
 78           File.WriteAllText(_filePath, JsonSerializer.Serialize(_mangas));
 79       }
 80   }
```

3. **Configura el repositorio dentro del contenedor de inyección de dependencias**
    - En la clase Program.cs modifica el tipo de alcance a scoped para la clase MangaService.cs.

Ahora configura el contenedor de inyección de dependencias para incluir el registro a la clase MangaRepository.cs para que pueda ser inyectado en otras clases.

```
C# Program.cs
1  using mangas.Services.Features.Mangas;
2  using mangas.Infrastructure.Repositories;
3
4  var builder = WebApplication.CreateBuilder(args);
5
6  // Add services to the container.
7  builder.Services.AddScoped<MangaService>();
8
9  // Add services to the container.
10 //builder.Services.AddSingleton<MangaService>();
11 builder.Services.AddScoped<MangaService>();
12 builder.Services.AddTransient<MangaRepository>();
13
14 builder.Services.AddControllers();
15 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
16 builder.Services.AddEndpointsApiExplorer();
17 builder.Services.AddSwaggerGen();
18
19 var app = builder.Build();
```

4. Modifica las llamadas en la clase de servicio.
    - Dentro de la clase MangaService debes inyectar una instancia de la clase MangaRepository, también debes modificar las llamadas a los métodos para que invoquen las acción correspondiente en el repositorio.

```
Services > Features > Mangas > C# MangaService.cs > MangaService > _mangaRepository
1  using mangas.Domain.Entities;
2  using mangas.Infrastructure.Repositories;
3
4
5  namespace mangas.Services.Features.Mangas;
6
   1 referencia
7  public class MangaService
8  {
       6 referencias
9      private readonly MangaRepository _mangaRepository;
10
       0 referencias
11     public MangaService(MangaRepository mangaRepository)
12     {
13         this._mangaRepository = mangaRepository;
14     }
15
       0 referencias
16     public IEnumerable<Manga> GetAll()
17     {
18         return _mangaRepository.GetAll();
19     }
```

```
7   public class MangaService
20
        2 referencias
21      public Manga GetById(int id)
22      {
23          return _mangaRepository.GetById(id);
24      }
25
        0 referencias
26      public void Add(Manga manga)
27      {
28          _mangaRepository.Add(manga);
29      }
30
        0 referencias
31      public void Update(Manga mangaToUpdate)
32      {
33          var manga = GetById(mangaToUpdate.Id);
34
35          if (manga.Id > 0)
36              _mangaRepository.Update(mangaToUpdate);
37      }
```

```
38
        0 referencias
39      public void Delete(int id)
40      {
41          var manga = GetById(id);
42          if (manga.Id > 0)
43              _mangaRepository.Delete(id);
44      }
```

5. **Prueba tu aplicación**
    - Abre la terminal de comandos de Visual Studio Code y lanza la aplicación utilizando el comando dotnet run.

Reto: explica que fue lo que sucedió o que cambio detectas con respecto a la versión anterior de tu aplicación elaborada anteriormente.