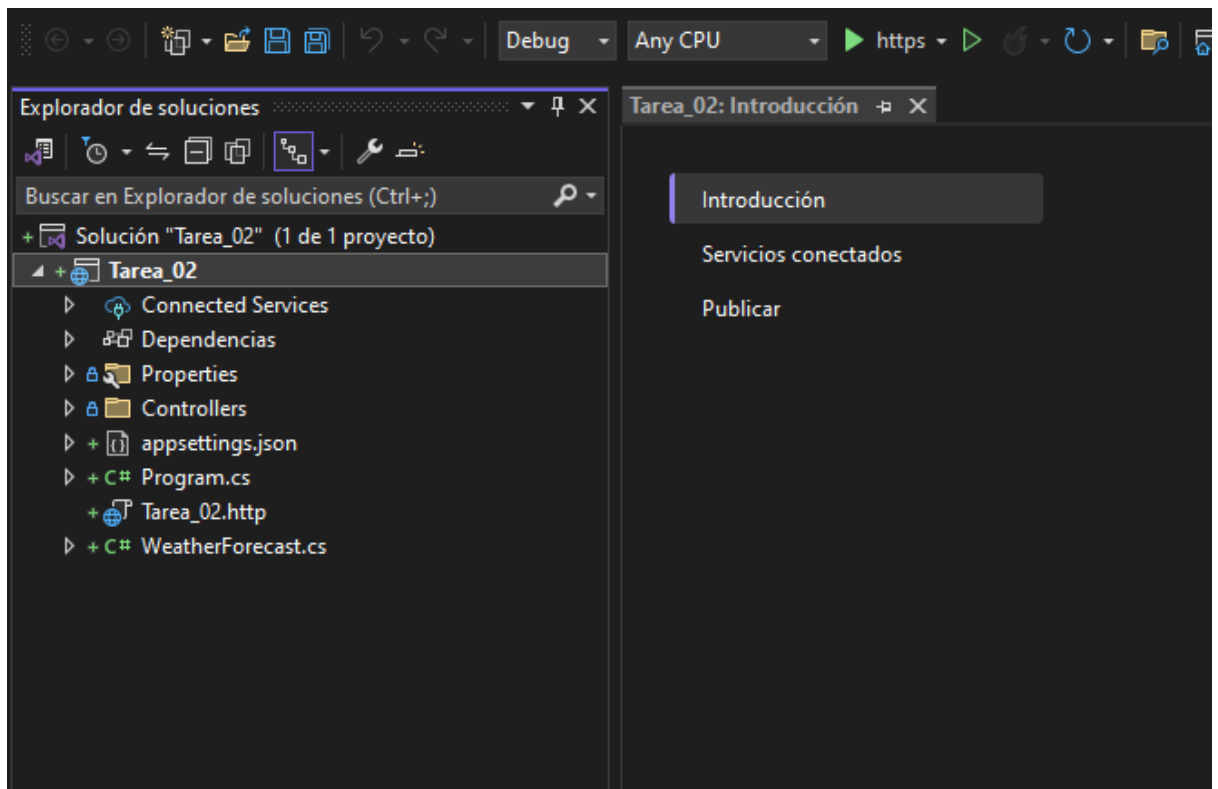


Taller .NET

Tarea_02

- 1) Para la siguiente tarea, podemos volver a crear el proyecto de la misma manera que lo hicimos la vez pasada en Tarea_01.



Una vez ya creamos nuestro proyecto ASP.Net Core Web Api, procedemos a replicar parte de lo que hicimos la vez pasada, donde teníamos nuestro controlador, con la arquitectura MVC, solo que esta vez, no haremos uso del controlador y nos acotamos a realizar la API con un único archivo, con lo mínimo necesario.

- 2) Ahora debemos empezar a pasar parte de lo que hicimos en el controlador , hacia nuestro archivo "Program.cs", donde se ejecutará todo nuestro código de la API.

Una vez en nuestro archivo

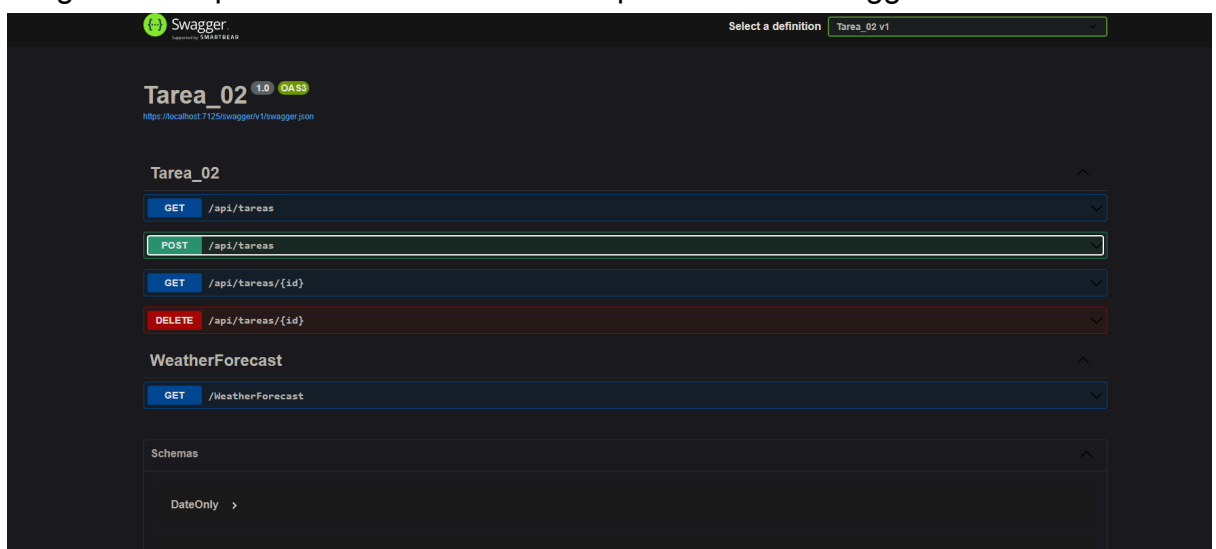
```
Program.cs
Tarea_02
{
1    var builder = WebApplication.CreateBuilder(args);
2
3    // Add services to the container.
4
5    builder.Services.AddControllers();
6    // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
7    builder.Services.AddEndpointsApiExplorer();
8    builder.Services.AddSwaggerGen();
9
10   var app = builder.Build();
11
12   // Configure the HTTP request pipeline.
13   if (app.Environment.IsDevelopment())
14   {
15       app.UseSwagger();
16       app.UseSwaggerUI();
17   }
18
19   app.UseHttpsRedirection();
20
21   app.UseAuthorization();
22
23   app.MapControllers();
24
25   app.Run();
26
}
```

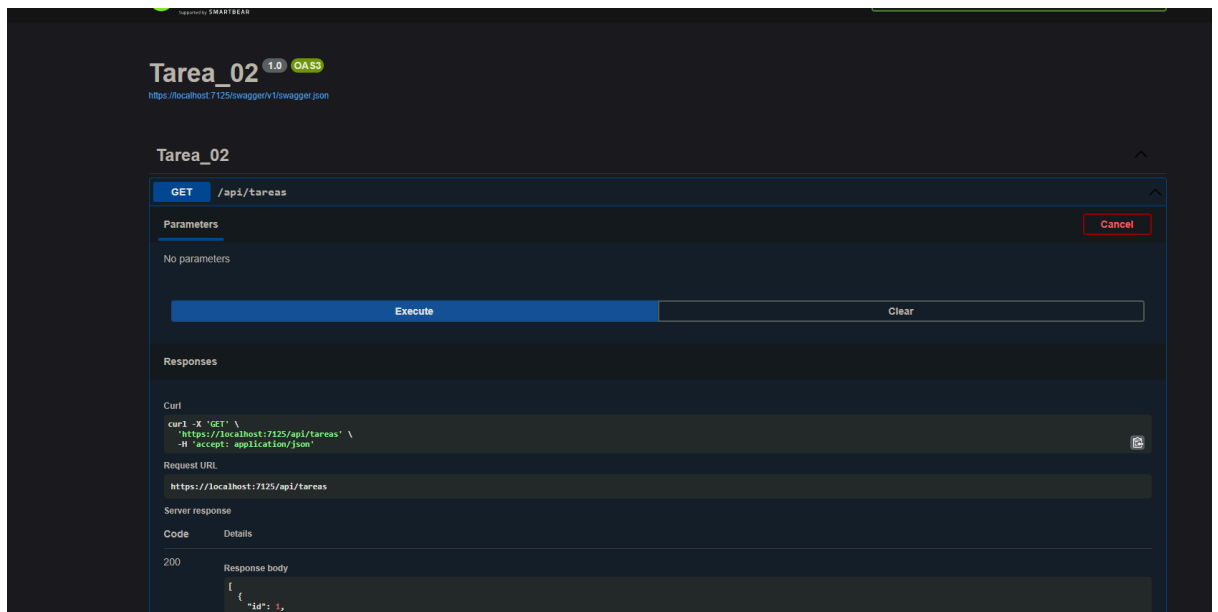
Una vez todo lo mínimo fue implementado en Program.cs, debería quedar de la siguiente manera, muy similar a antes.

```
Program.cs
Tarea_02
{
19   }
20
21   app.UseHttpsRedirection();
22
23   //primero simular la memoria
24
25   var tareas = new List<Tarea>
26   {
27       new Tarea { Id = 1, Nombre = "Tarea 1", Descripcion = "La tarea 1", DuracionHoras = 1, Responsable = "Juanito", Fecha = DateTime.Now },
28       new Tarea { Id = 2, Nombre = "Tarea 2", Descripcion = "La tarea 2", DuracionHoras = 2, Responsable = "Pedrito", Fecha = DateTime.Now }
29   };
30
31   app.UseAuthorization();
32
33   app.MapControllers();
34
35   app.MapGet("/api/tareas", () => tareas); // todas
36
37   >app.MapGet("/api/tareas/{id:int}", (int id) => // por id
38   {
39       var tarea = tareas.FirstOrDefault(t => t.Id == id);
40       return tarea is not null ? Results.Ok(tarea) : Results.NotFound();
41   });
42
43   >app.MapPost("/api/tareas", (Tarea nuevaTarea) => // agregar
44   {
45       nuevaTarea.Id = tareas.Max(t => t.Id) + 1;
46       nuevaTarea.Fecha = DateTime.Now;
47       tareas.Add(nuevaTarea);
48       return Results.Created($"api/tareas/{nuevaTarea.Id}", nuevaTarea);
49   });
50
51   >app.MapDelete("/api/tareas/{id:int}", (int id) => // deletar
52   {
53       var tarea = tareas.FirstOrDefault(t => t.Id == id);
54       if (tarea is not null)
55       {
56           tareas.Remove(tarea);
57       }
58   });
59
60 }
```

```
Cambios de GIT: .NET__Lab  Tarea_02.http  Program.cs  X
Tarea_02
49  });
50
51  app.MapDelete("/api/tareas/{id:int}", (int id) => // deletear
52  {
53      var tarea = tareas.FirstOrDefault(t => t.Id == id);
54      if (tarea is not null)
55      {
56          tareas.Remove(tarea);
57          return Results.NoContent();
58      }
59      return Results.NotFound();
60  });
61
62  app.Run();
63  record Tarea // declarar datos
64  {
65      public int Id { get; set; }
66      public string Nombre { get; set; }
67      public string Descripcion { get; set; }
68      public double DuracionHoras { get; set; }
69      public string Responsable { get; set; }
70      public DateTime Fecha { get; set; }
71  }
```

Luego de esto procedemos una vez mas a probarlo con Swagger.





Una vez más, podemos ver que funcionan como antes, pero con el enfoque minimal API.

Repo de GitHub con el trabajo:

https://github.com/JuanmaPilon/dotNet_Practices