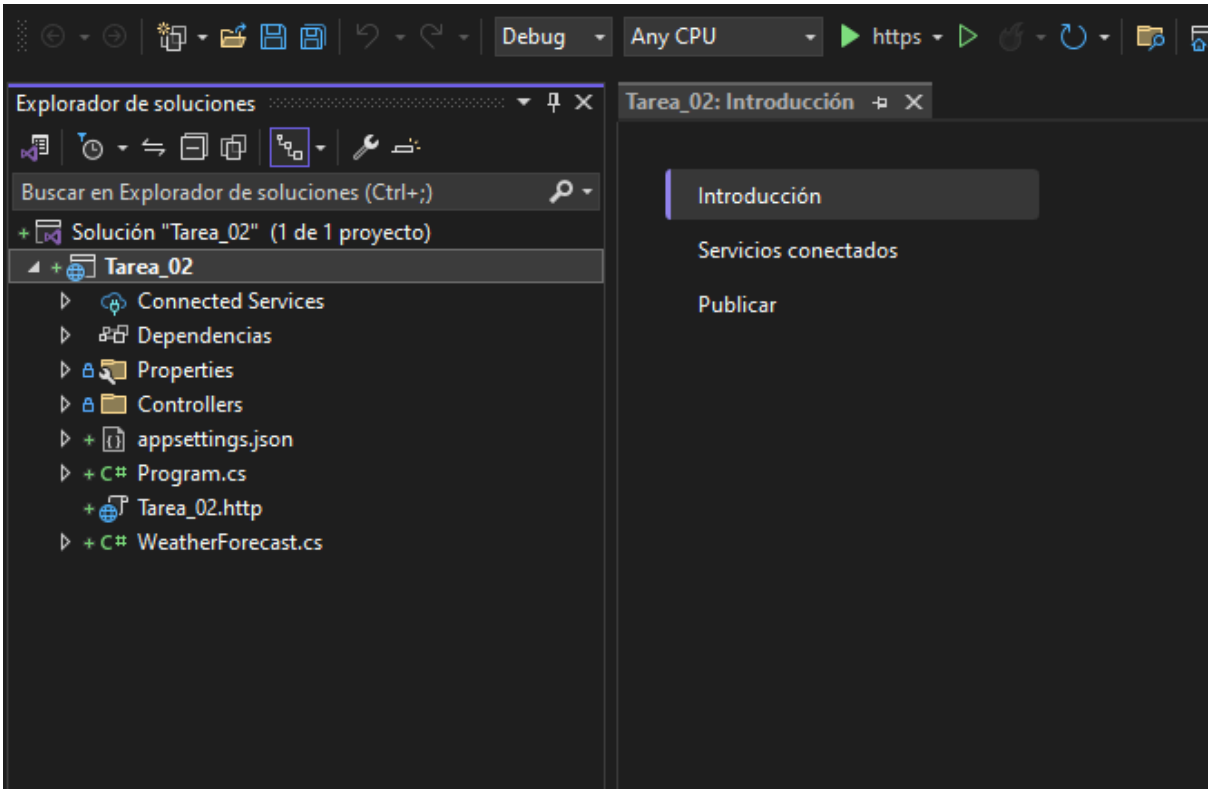
Taller .NET

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.

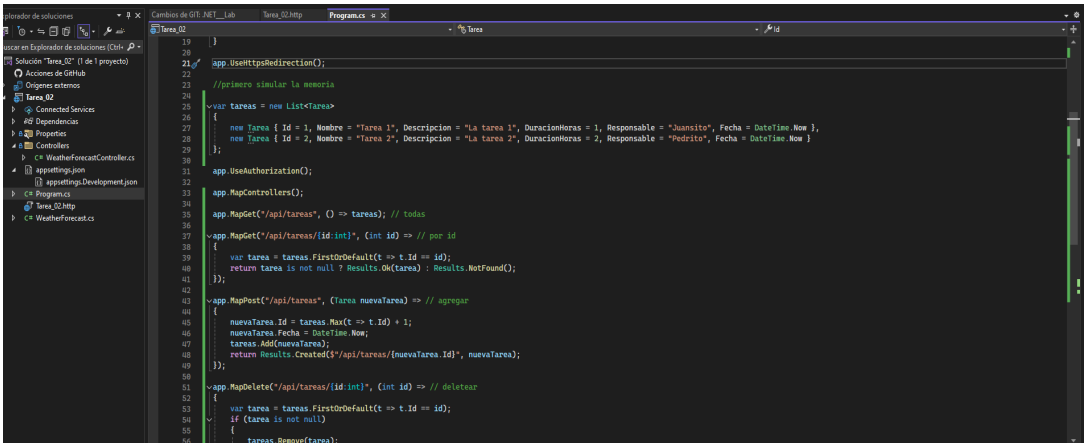


Una vez ya creamos nuestro proyecto ASP.Net Core Web Api, procedemos a replicar parte de lo que hicimos la vez pasada, donde teníamos nuestro controlador, con la arquitectura MVC, solo que esta vez, no haremos uso del controlador y nos acotamos a realizar la API con un único archivo, con lo mínimo necesario.

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

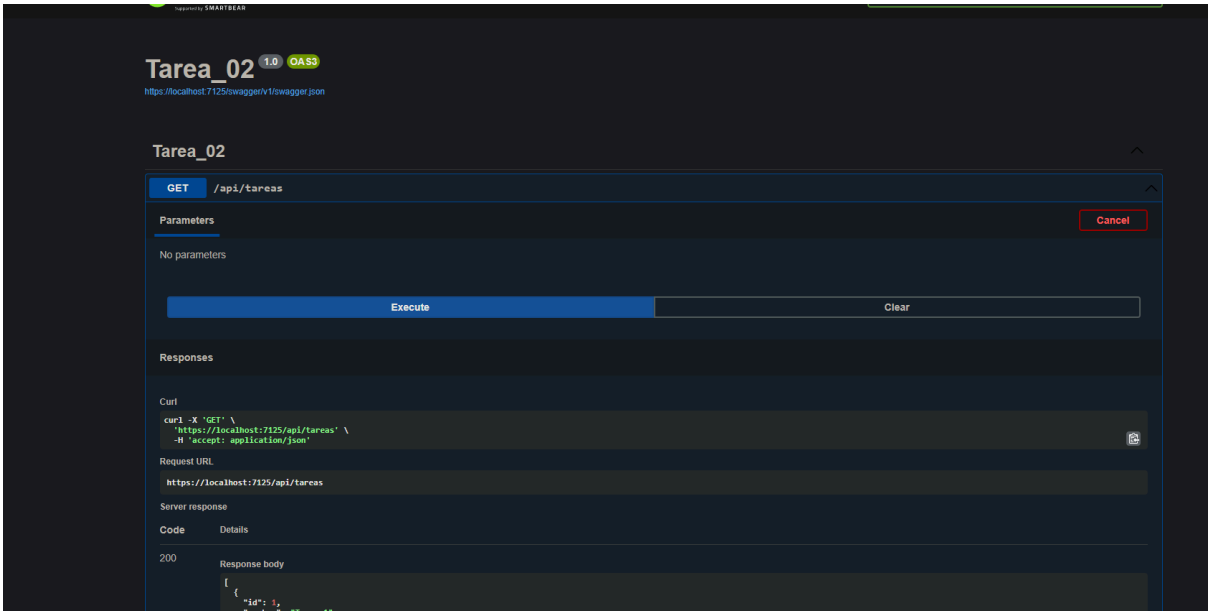
```
Program.cs
Tarea_02

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

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



```
Cambios de GIT: .NET__Lab   Tarea_02.http   Program.cs
Tarea_02                                              Tarea
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();
        4 referencias
63      record Tarea // declarar datos
64      {
            7 referencias
65          public int Id { get; set; }
            2 referencias
66          public string Nombre { get; set; }
            2 referencias
67          public string Descripcion { get; set; }
            2 referencias
68          public double DuracionHoras { get; set; }
            2 referencias
69          public string Responsable { get; set; }
            3 referencias
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