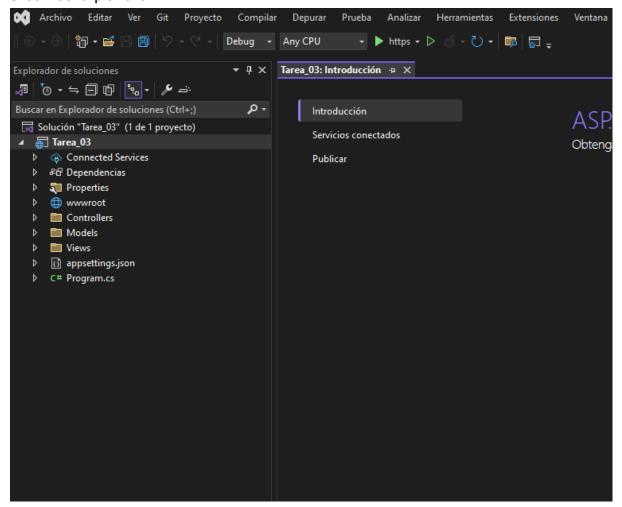
## Taller .NET

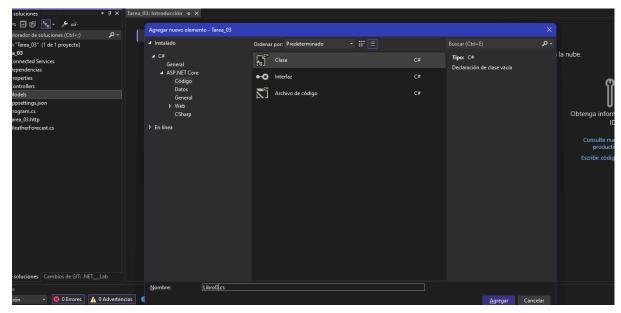
## Tarea\_03

 Para la siguiente tarea, podemos volver a crear el proyecto de la misma manera que lo hicimos la vez pasada en Tarea\_01 y Tarea\_02. Solo que esta vez usaremos la plantilla Aplicación Web de ASP.NET (Modelo-Vista-Controlador)

Creamos la plantilla.



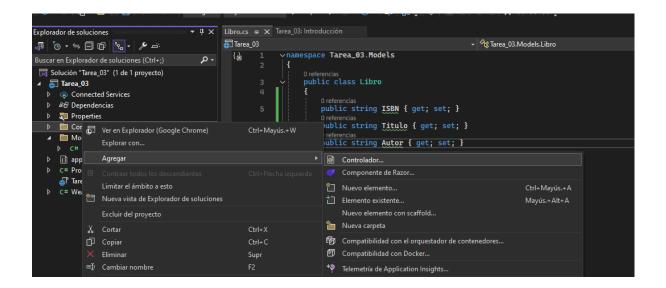
Una vez creada la plantilla procedemos a crear una carpeta para models. Creamos también una clase libro.cs.



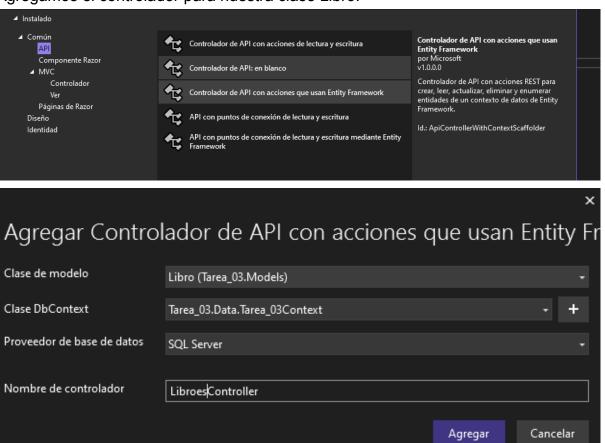
Luego a la clase Libro le agregamos sus atributos y sus respectivos setters y getters.

```
| Tarea_03 | Tarea_03: Introducción | Parea_03: Introducción | Parea_03
```

Una vez hecho esto podemos proceder a usar Scaffolding para generar el controlador.



## Agregamos el controlador para nuestra clase Libro.



```
`o • ≒ 🗐 🗗 🛼 • 🎤 ≕
                                                                     ∨namespace Tarea_03.Models
r en Explorador de soluciones (Ctrl+;)
 lución "Tarea 03" (1 de 1 provecto)
Tarea_03
  Connected Services
                                                                                  O referencias public string ISBN { get; set; }
 Properties
                                                                                  public string Titulo { get; set; }
 Controllers
Models
                                                                                  public string Autor { get; set; }
    C# Libro.cs
 appsettings.json
                                                                                   Oreferencias
public DateTime FechaPublicacion { get; set; }
 C# WeatherForecast.cs
                                                                                                       Microsoft Visual Studio
                                                                                                       Instalando los paquetes NuGet. Esta operación puede tardar unos minutos Microsoft. Visual Studio. Web. Code Generation. Design
```

Seleccionamos todo lo relacionado a nuestra clase libro y le damos add.

```
Microsoft Visual Studio

Scaffolding...

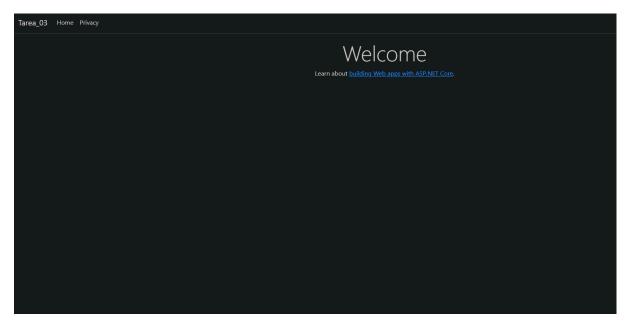
Instalando los paquetes NuGet. Esta operación puede tardar unos minutos.

Microsoft.EntityFrameworkCore.SqlServer
```

Vemos como automáticamente va a empezar a agregar el NuGet package manager.

Una vez creado podemos ver que nos creó toda la estructura con Scaffolding. También de ahí podemos probar deployando el servidor.

Ponemos a correr el servidor y vemos que tenemos nuestro proyecto andando.



Ahí vemos que tenemos nuestra página template creada correctamente. Luego procedemos a modificar nuestro controlador y crear las vistas de Razor. Para nuestro controlador debemos agregar que retorne cada view que queremos.

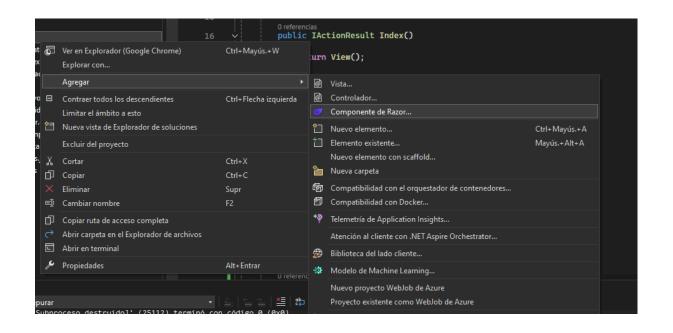
Vamos a nuestro controlador y cambiamos las view.

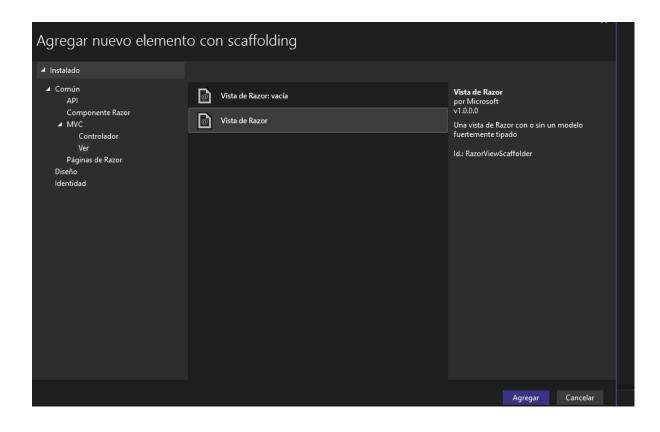
```
▼ Tarea_03.Controllers.HomeController

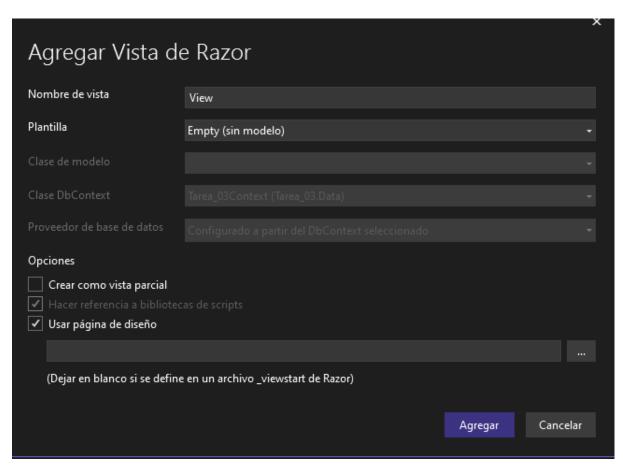
                                             Tarea_03
 · ≒ 🗐 📵 🔪 - 🎤 📑
                                                             vusing Microsoft.AspNetCore.Mvc;
using System.Diagnostics;
using Tarea_03.Models;
                                               { ja
Explorador de soluciones (Ctrl+;)
ución "Tarea_03" (1 de 1 proyecto)
Tarea 03
Connected Services
                                                             ∨namespace Tarea_03.Controllers
Pa Dependencias
Properties
Controllers
                                                                        private readonly ILogger<HomeController> _logger;
 C# HomeController.cs
 C# LibroController.cs
                                                                        Oreferencias public HomeController(ILogger<HomeController> logger)
Data
Models
                                                                             _logger = logger;
  C# ErrorViewModel.cs
  C# Libro.cs
                                                                        Oreferencias
public IActionResult Index()
  Home
     Create.cshtml
                                                                             return View();
     Privacy.cshtml
                                                                        O referencias
public IActionResult Privacy()
    return View();
     Error.cshtml
     _ViewImports.cshtml
  ViewStart.cshtml
                                                                        O referencias

public IActionResult Create()
appsettings.json
                                                                             return View();
                                                                        0 referencias
public IActionResult Delete()
                                                                                  return View();
```

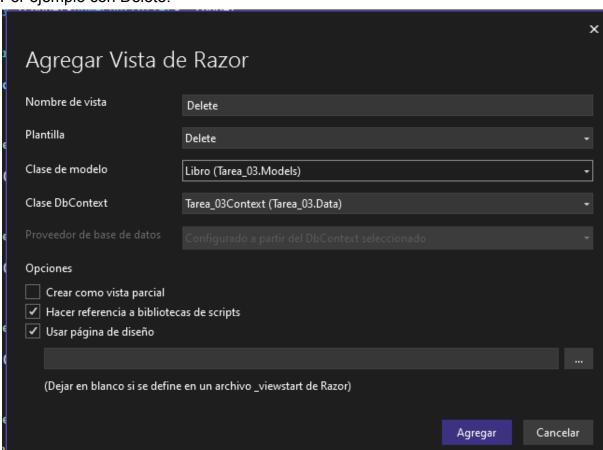
De esta manera tenemos que retorne cada view para cada método. Luego procedemos a crear una view con Razor.







## Por ejemplo con Delete.



Quedó la view hecha con Delete.

Repetimos para los demas métodos del CRUD.

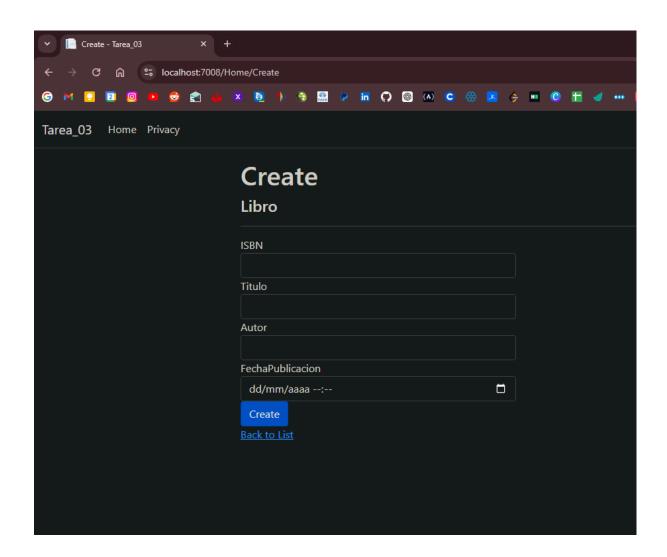
```
C# Tarea_03
                                                         @model Tarea_03.Models.Libro
 car en Explorador de soluciones (Ctrl+;)
Solución "Tarea_03" (1 de 1 proyecto)

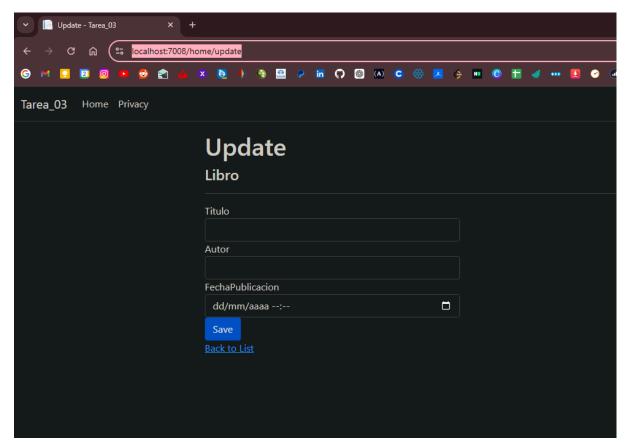
Tarea_03
                                                             ViewData["Title"] = "Update";
 Properties
                                                         <h1>Update</h1>
                                                         <h4>Libro</h4>
                                                       C# HomeController.cs
   C# LibroController.cs
                                                                  Data
Models
   C# ErrorViewModel.cs
   C# Libro.cs

■ Views
         Create.cshtml
Delete.cshtml

<
          List.cshtml
         Privacy.cshtml
Update.cshtml
                                                                     ☐ _ViewImports.cshtml
                                                                      ViewStart.cshtml
   appsettings.json
```

Una vez que tenemos todas las view podemos acceder a ellas levantando el servidor.





Ahí algunos ejemplos de como queda funcional con Razor pages.

Repo de GitHub con el trabajo:

https://github.com/JuanmaPilon/dotNet Practices