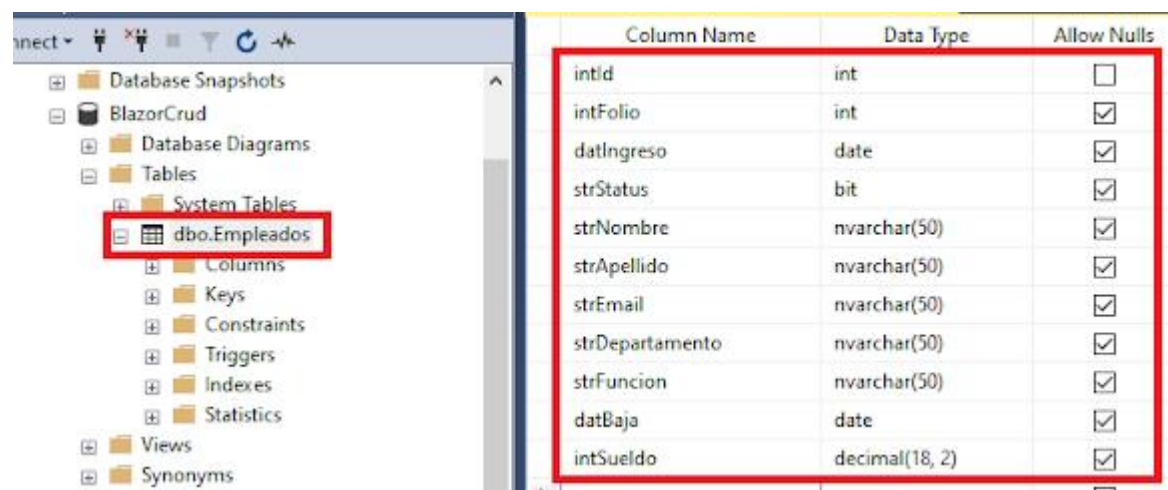


Para la creación del CRUD no crearemos la base de datos desde el código (**Code First**), llamare una base de datos ya creada con **Scaffold (Database First)**.

- **Create** (crear) utilizar el método **HTTP POST**.
- **Read** (leer) utilizar el método **HTTP GET**.
- **Update** (actualizar) utilizar el método **HTTP PUT**.
- **Delete** (eliminar) utilizar el método **HTTP DELETE**.

Ingresaremos a **SQL Server** y crearemos la base de datos llama **BlazorCrud (O como gusten)** solo con la tabla **Empleados**, la estructura de la tabla es la siguiente, solo el campo **intId** es autoincremental.



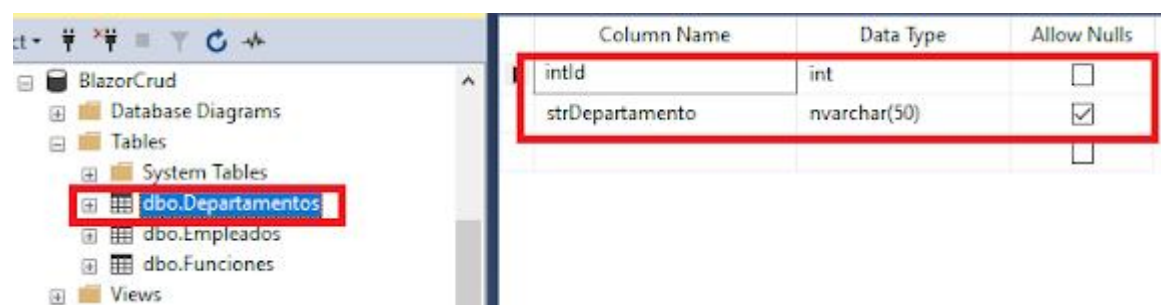
Column Name	Data Type	Allow Nulls
intId	int	<input type="checkbox"/>
intFolio	int	<input checked="" type="checkbox"/>
dateTimeIngreso	date	<input checked="" type="checkbox"/>
strStatus	bit	<input checked="" type="checkbox"/>
strNombre	nvarchar(50)	<input checked="" type="checkbox"/>
strApellido	nvarchar(50)	<input checked="" type="checkbox"/>
strEmail	nvarchar(50)	<input checked="" type="checkbox"/>
strDepartamento	nvarchar(50)	<input checked="" type="checkbox"/>
strFuncion	nvarchar(50)	<input checked="" type="checkbox"/>
dateTimeBaja	date	<input checked="" type="checkbox"/>
intSueldo	decimal(18, 2)	<input checked="" type="checkbox"/>

Ingresaremos 15 registros manualmente que se visualizarán en la tabla.

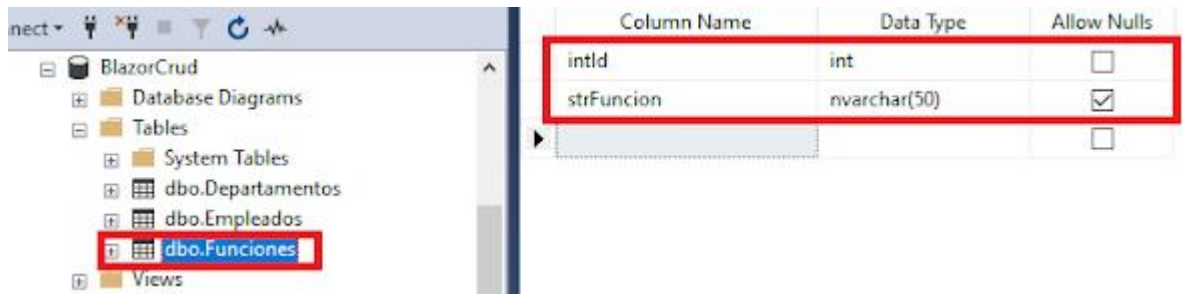


intId	intFolio	dateTimeIngreso	strStatus	strNombre	strApellido	strEmail	strDepartamen...	strFuncion	dateTimeBaja
1	1	2022-01-22	True	Juan	Perez	jperez@email.com	Tesoreria	Contador	NULL
2	2	2022-01-04	True	Pedro	Sola	psola@email.com	Finanzas	Supervisor	NULL
3	3	2022-01-07	False	Victor	Pedroza	vpedroza@email.com	Tesoreria	Supervisor	NULL
4	4	2022-01-27	True	Mario	Gonzalez	mgonzalez@email.com	Cobranza	Ejecutivo	NULL
5	5	2022-01-28	True	Teresa	Jimenez	tjimenez@email.com	Producción	Productor	NULL
6	6	2022-01-15	False	Lorena	Itama	litaama@email.com	Producción	Asistente	NULL
7	7	2022-01-29	True	María	Becerra	mbecerra@email.com	Cuentas	Ejecutivo	NULL
8	8	2022-01-31	True	Louises	Campos	lcampo@email.com	Cuentas	Ejecutivo	NULL
9	9	2022-01-31	True	Karla	Cacho	kcacho@email.com	Cuentas	Supervisor	NULL
10	10	2022-02-01	True	Gabriela	Mendez	gmendez@email.com	Finanzas	Contador	NULL
11	11	2022-02-01	False	Raquel	Gómez	rgomez@email.com	Tesoreria	Asistente	NULL
12	12	2022-02-02	True	Ramón	Caballero	rcaballero@email.com	Cobranza	Coordinador	NULL
13	13	2022-02-02	True	Hector	Hernandez	hhernandez@email.com	CuentasPagar	Supervisor	NULL
14	14	2022-02-02	True	Ricardo	Garcia	rgarcia@email.com	Tesoreria	Contador	NULL
15	15	2022-02-02	True	Edoardo	Santillan	esantillan@email.com	CuentasPagar	Asistente	NULL

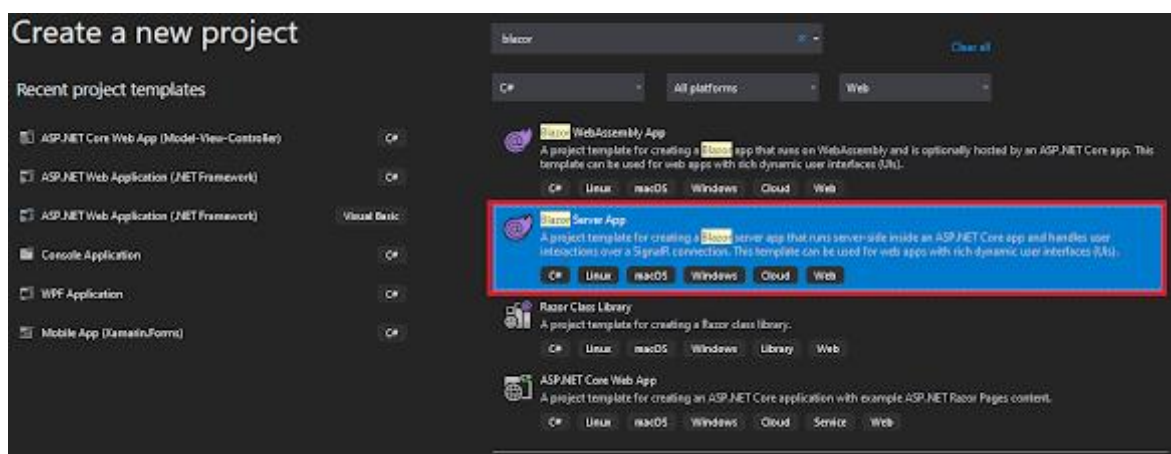
También crearemos 2 tablas más, **Departamentos** y **Funciones**.



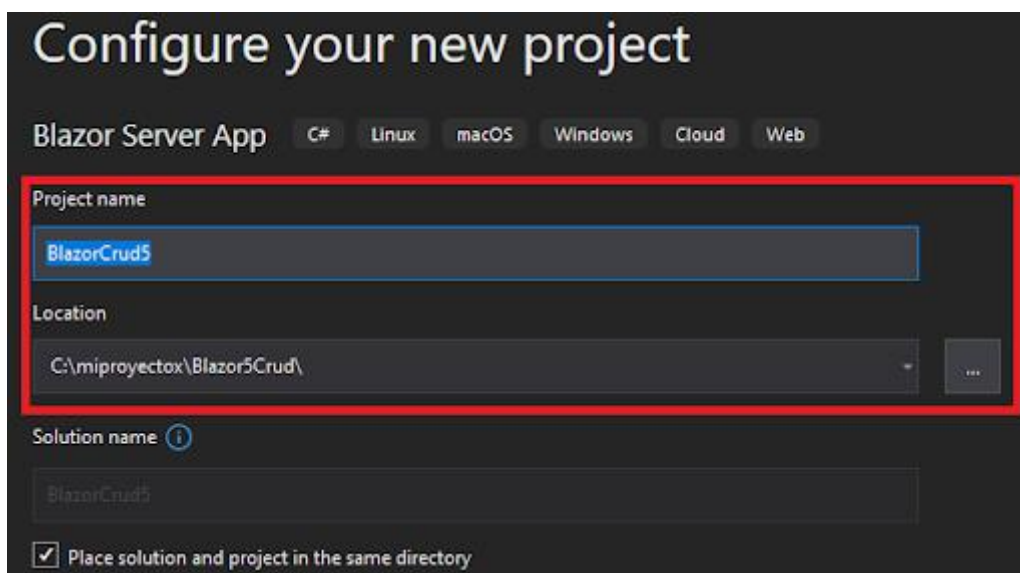
Column Name	Data Type	Allow Nulls
intId	int	<input type="checkbox"/>
strDepartamento	nvarchar(50)	<input checked="" type="checkbox"/>



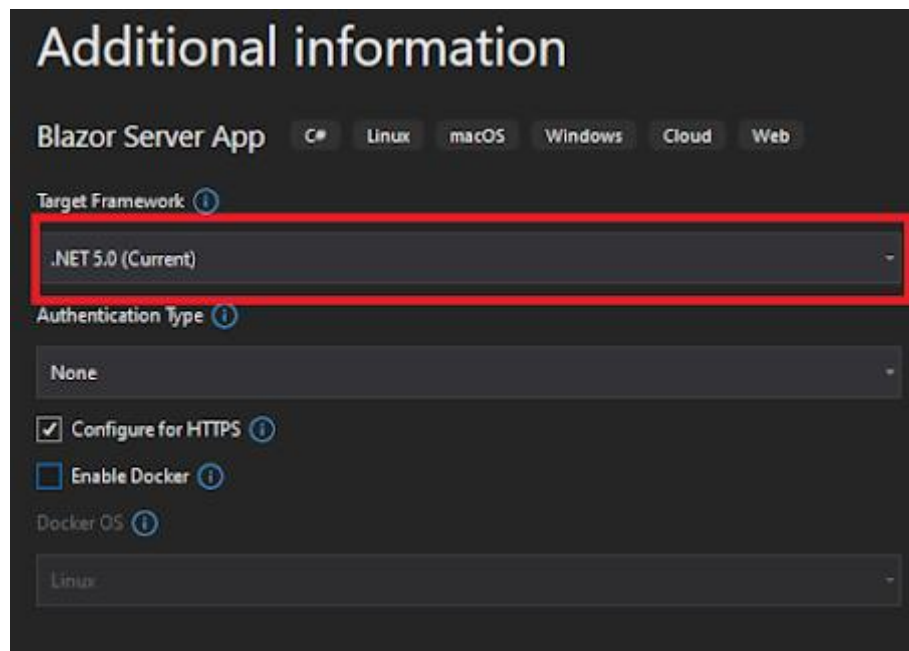
Crearemos un proyecto **Blazor** en **VS2019** o **VS2022**, seleccionaremos **Blazor Server App**, ya que la aplicación se ejecuta desde el servidor y se enviarán los datos al cliente por medio de una conexión en tiempo real de **SignalR**, mientras que **Blazor WebAssembly App** se descarga directamente todos los archivos Assembly al navegador del cliente, el código sirve para ambos casos.



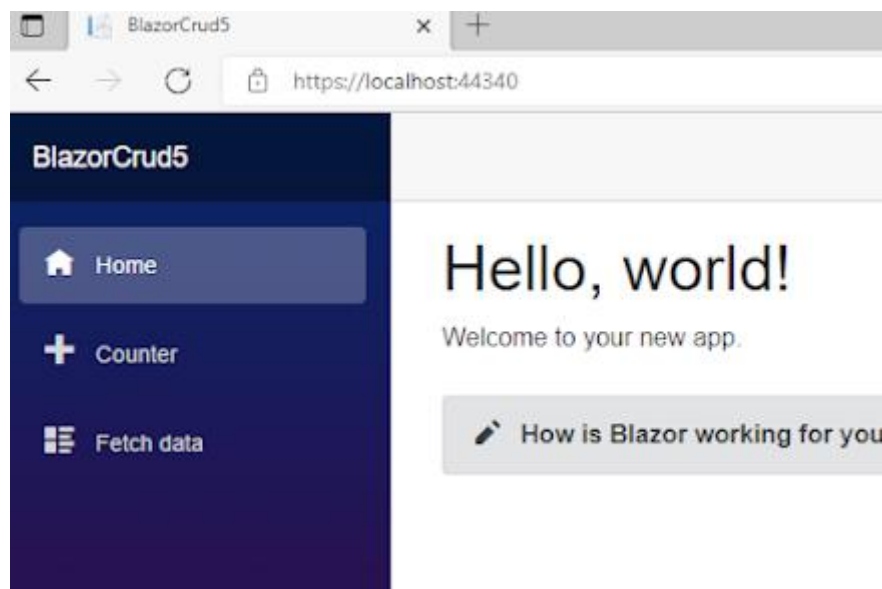
Teclaremos el nombre de nuestro proyecto así como la ruta donde será guardado.



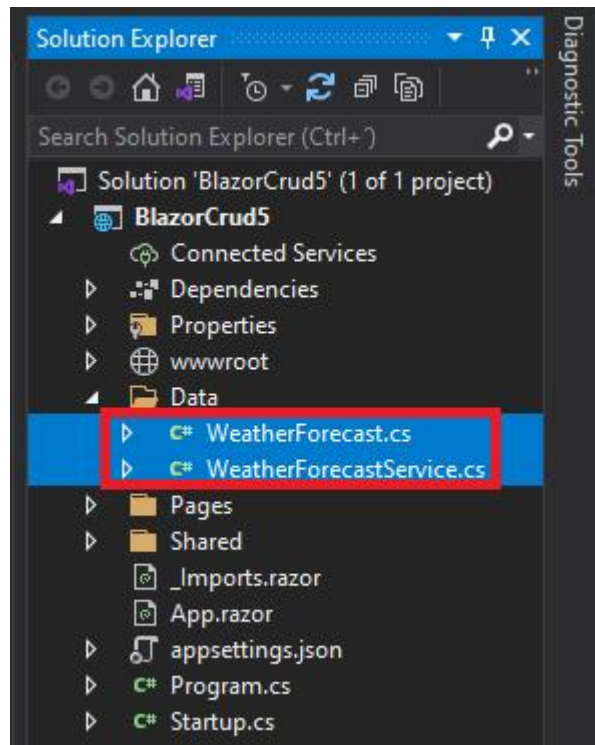
Utilizaremos la versión **.Net 5** o **6** segun marque su **VS2019** o **VS2022**



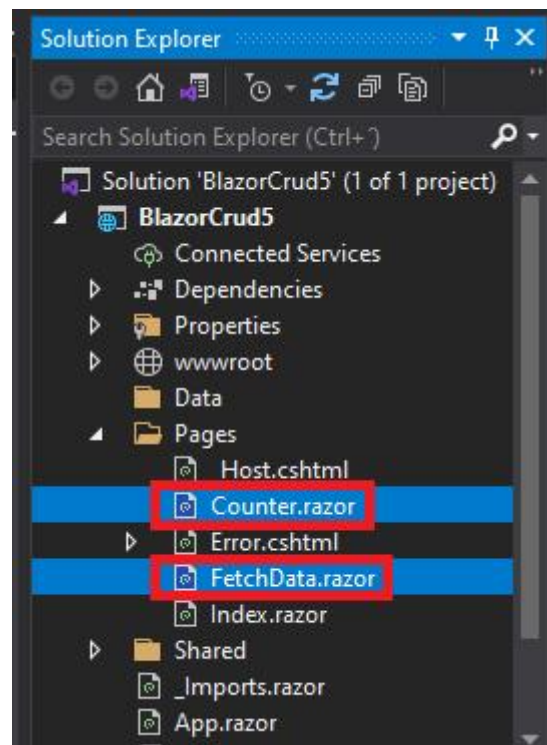
Una vez creado el proyecto podemos ejecutarlo para verificar que todo este correcto.



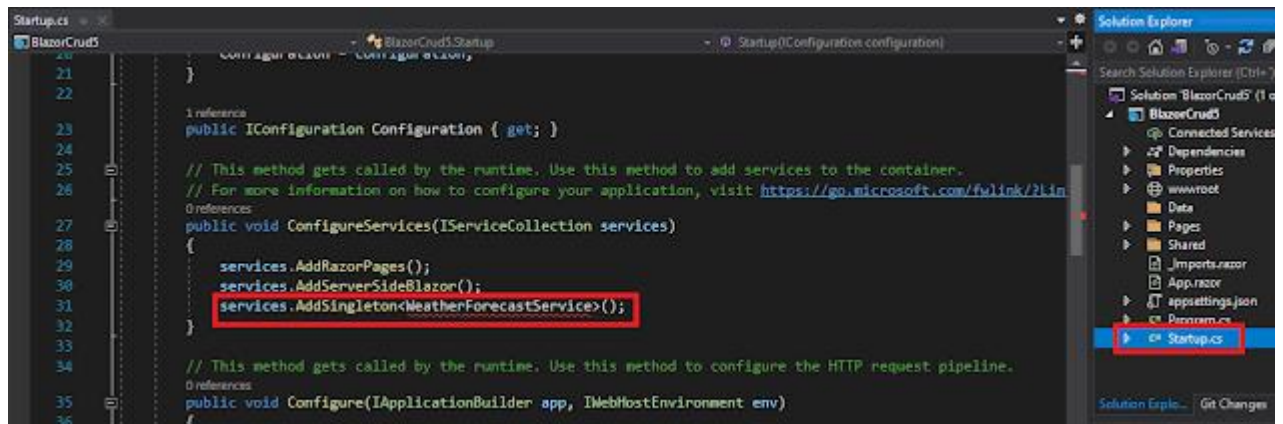
Ahora debemos depurar el proyecto eliminando los archivo que no utilizaremos, eliminar los archivos de la carpeta **Data**.



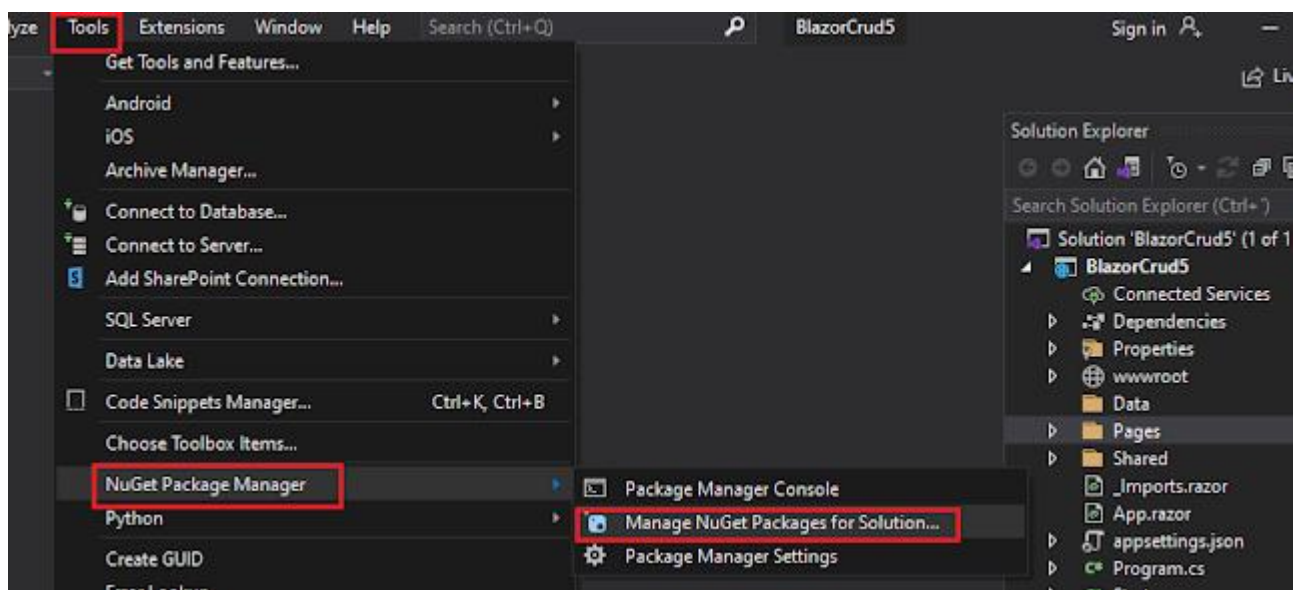
Eliminar de la carpeta **Pages** los archivos **Counter.razor** y **FetchData.razor**.



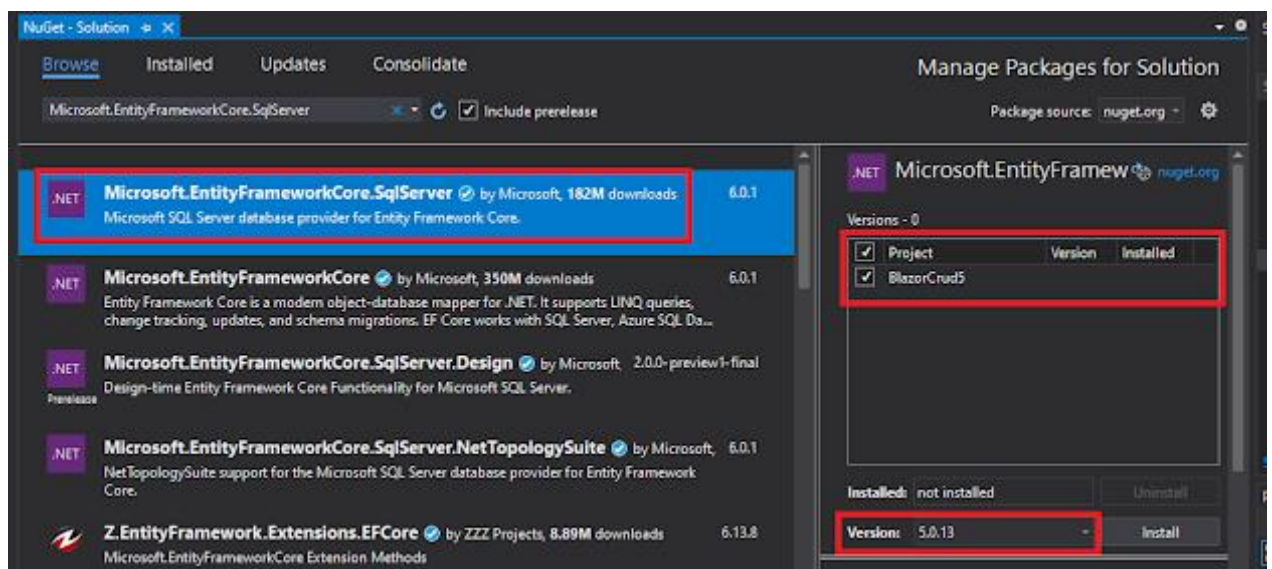
En el archivo **Startup.cs** eliminar la siguiente línea donde se llama **WeatherForecastService**.



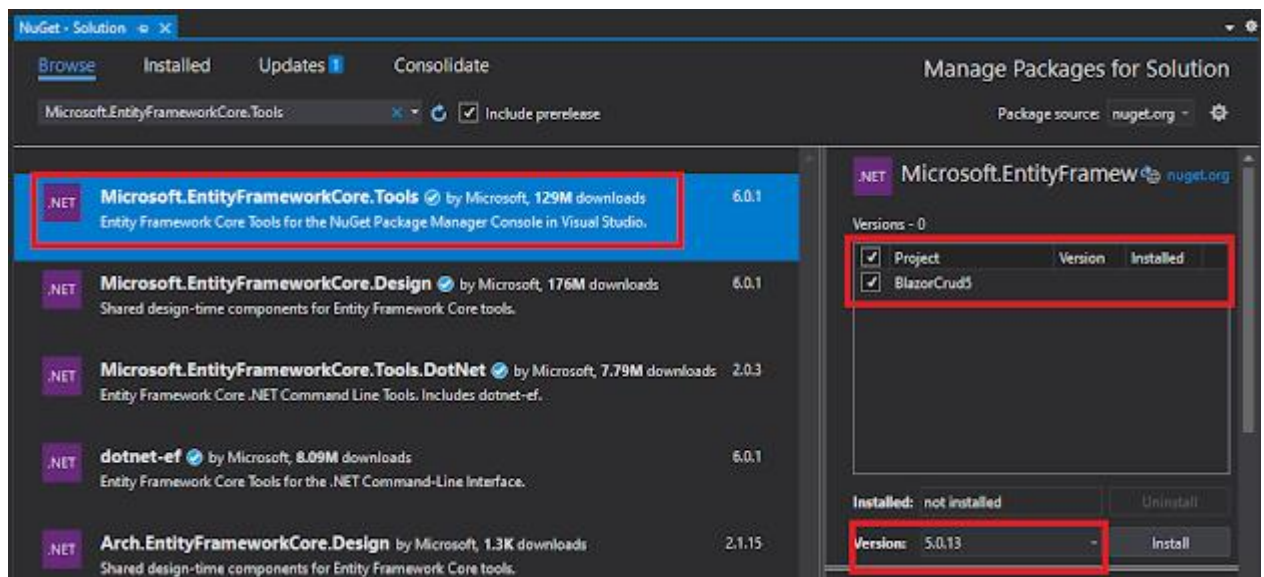
Instalaremos los siguientes paquetes, seleccionamos **Tools/NuGet Packages Manager for Solution...** solo se deberá verificar que se instale la **versión 5**.



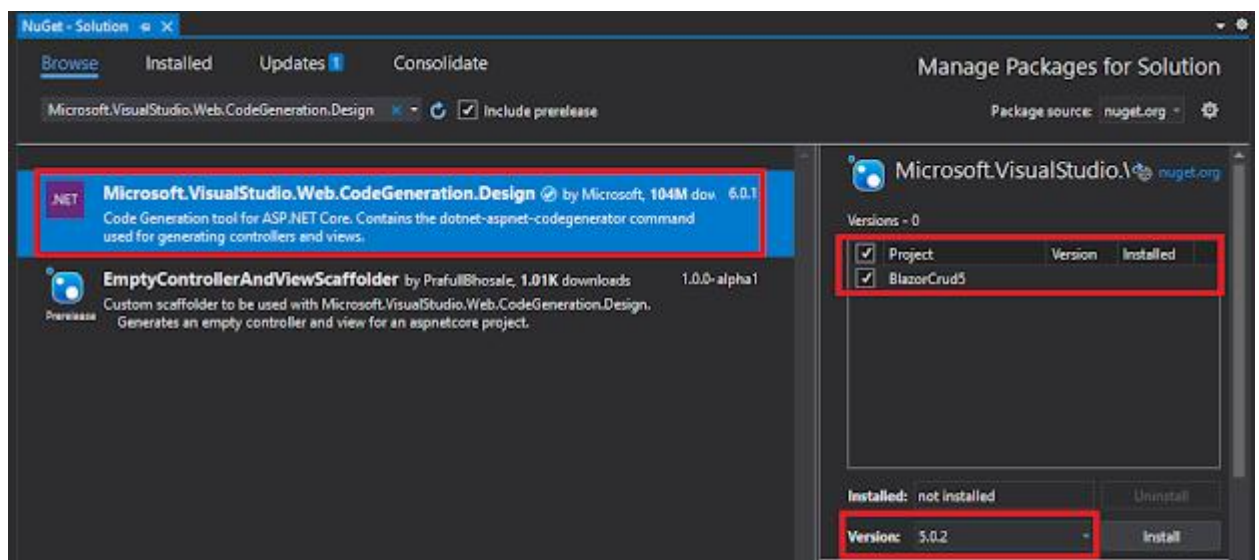
Microsoft.EntityFrameworkCore.SqlServer. Realizara la conexión con el proveedor de datos de SQL Server.



Microsoft.EntityFrameworkCore.Tools. Se encarga de crear el "modelo de datos" en la aplicación MVC.



Microsoft.VisualStudio.Web.CodeGeneration.Design. Por último, instalaremos el paquete de "diseño", este nos ayudará a crear los Controladores y las Pages de nuestra aplicación (**Scaffolding**), en función del "**Modelo**" y "**Contexto**" de datos generados automáticamente por Entity Framework.

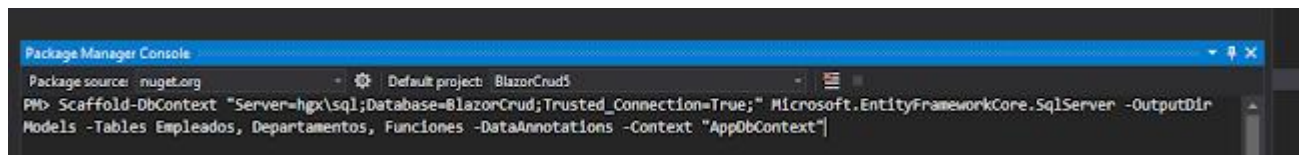


En la consola teclearemos la siguiente instrucción:

```
PM> Scaffold-DbContext  
"Server=hgx\sql;Database=BlazorCrud;Trusted_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Tables Empleados, Departamentos, Funciones -DataAnnotations -Context "AppDbContext"
```

Indicamos el servidor y la base de datos, realiza la integración de paquete Sql Server, indicamos que creé una carpeta llamada **Models** donde creara las clases de las

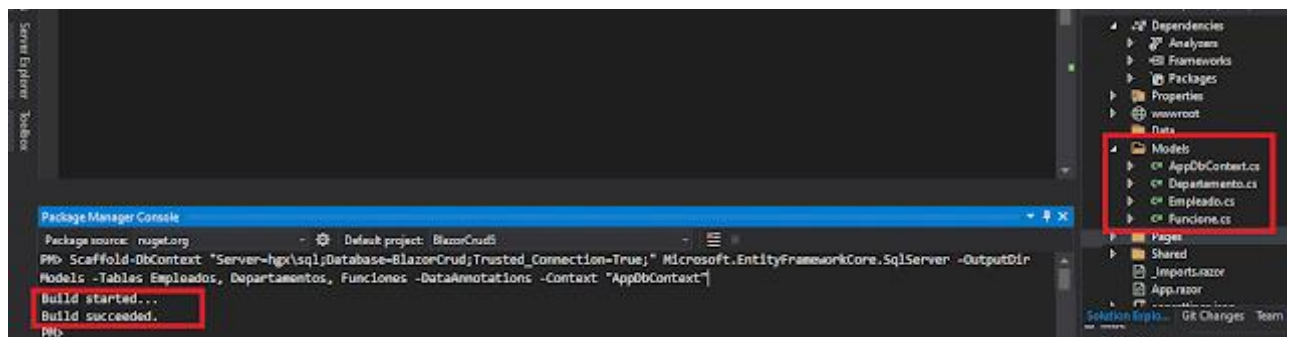
tablas **Empleados**, **Departamentos** y **Funciones**, por último se creará el archivo **AppDbContext** que contendrá los **DbSet** de las tablas que indicamos en la importación.



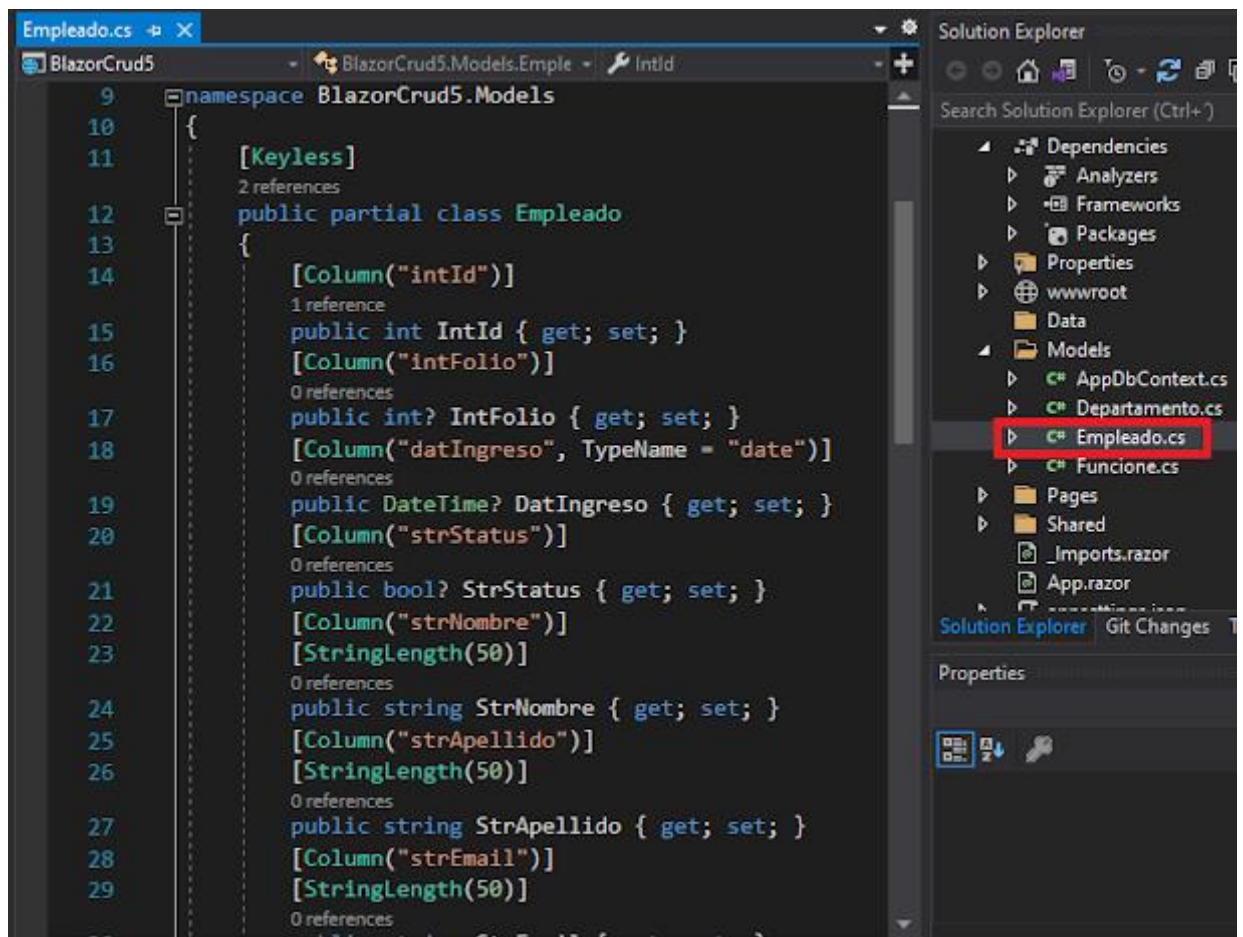
```
Package Manager Console
Package source: nuget.org
Default project: BlazorCrud5
PM> Scaffold-DbContext "Server=hgx\sql;Database=BlazorCrud;Trusted_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Tables Empleados, Departamentos, Funciones -DataAnnotations -Context "AppDbContext"
```

Una vez ejecutado se podrá visualizar una nueva carpeta llamada **Models** donde se crearán las clases de las 3 tablas, también se creó el archivo **AppDbContext.cs** que contiene la conexión a la base de datos.

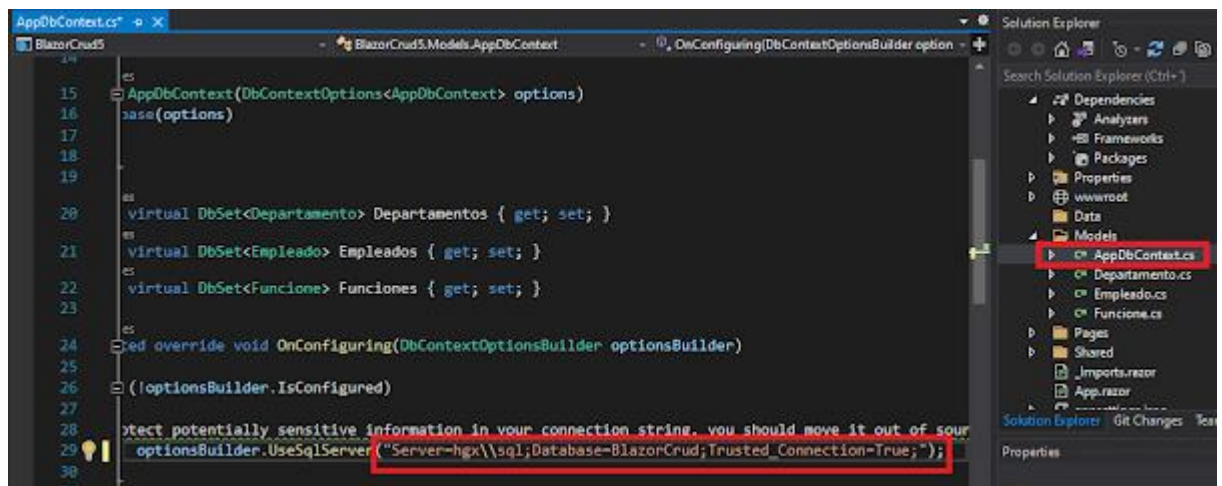
La importación con Scaffold solo se puede realizar una vez, posteriormente en caso de querer agregar más tablas o campos a las tablas existente se debe utilizar Add-Migration.



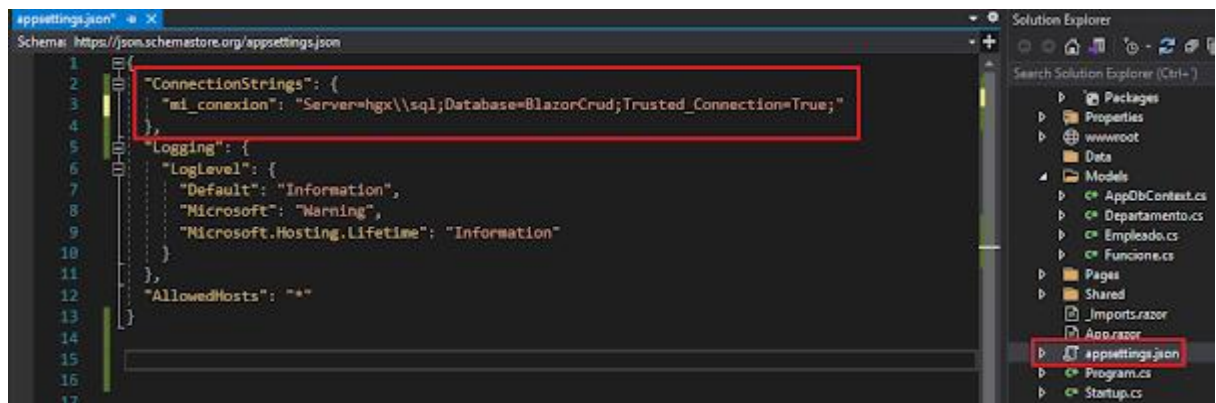
Cada clase creada en **Models** contiene los campos de las tablas.



En el archivo **AppDbContext.cs** se visualiza la cadena de conexión a la base de datos, se recomienda quitarlo y agregarlo en el archivo **appsetting.json**.

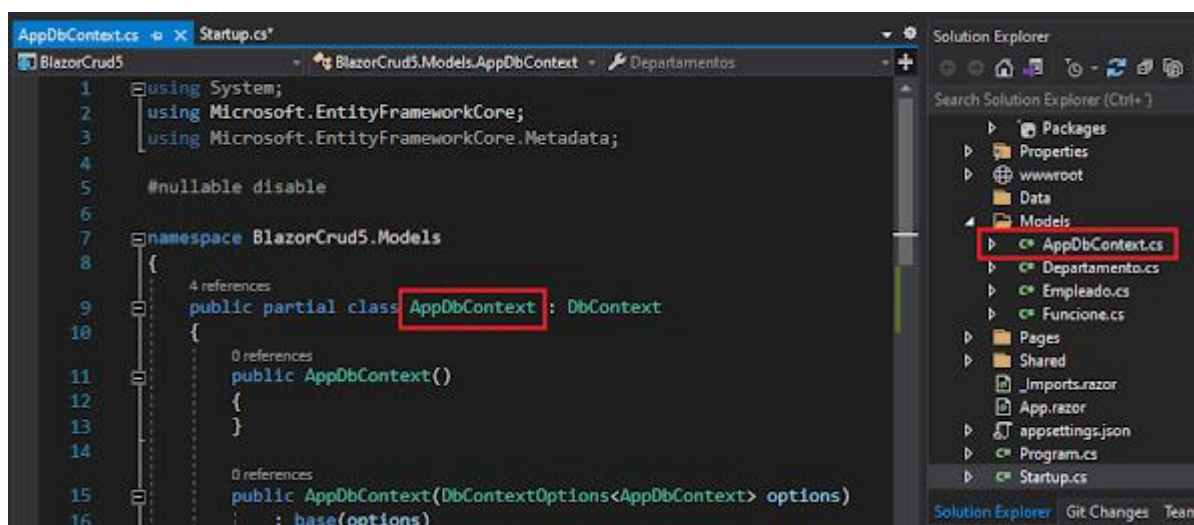
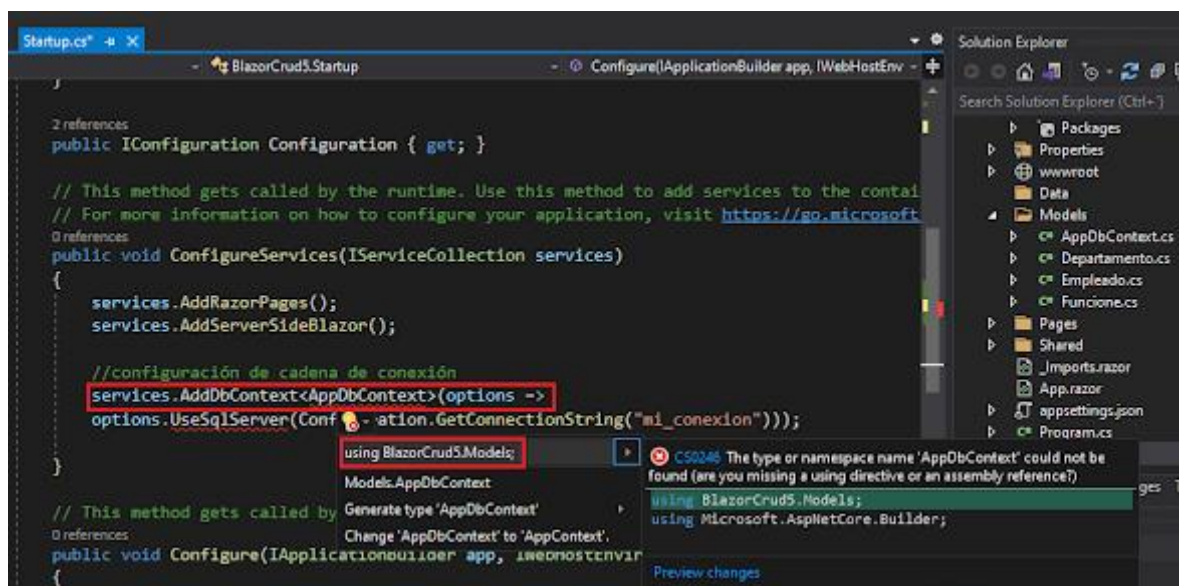


Dentro de archivo **appsetting.json** se creara el nombre de la cadena de conexión.

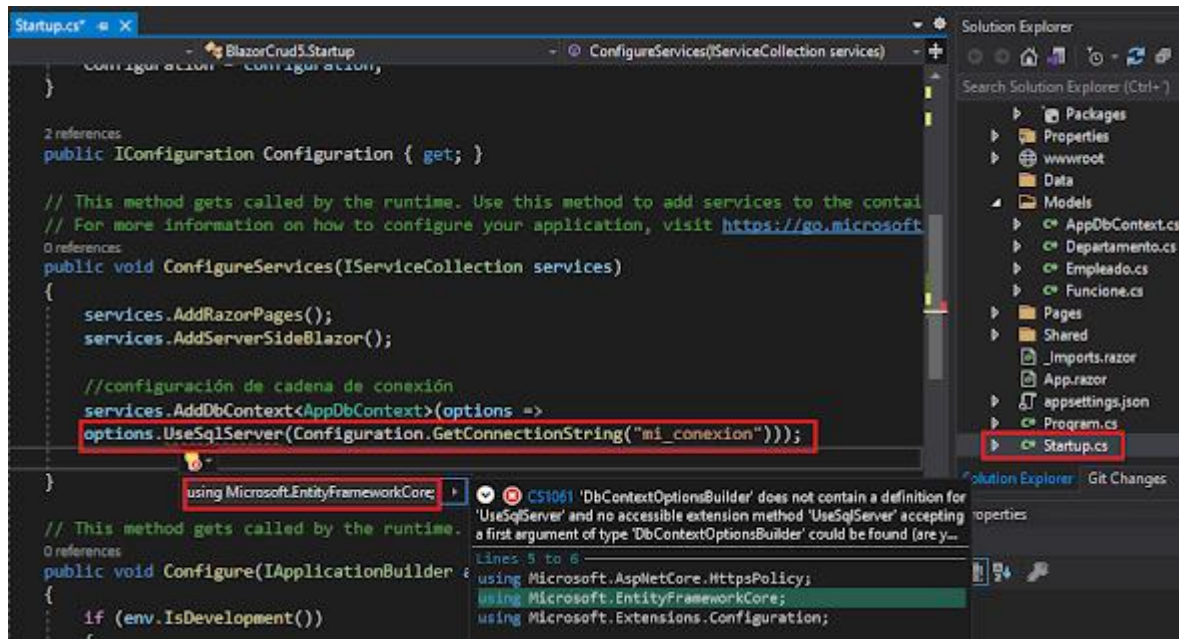


Ahora llamaremos la conexión a la base de datos en el archivo **Startup.cs** dentro del método **ConfigureServices**.

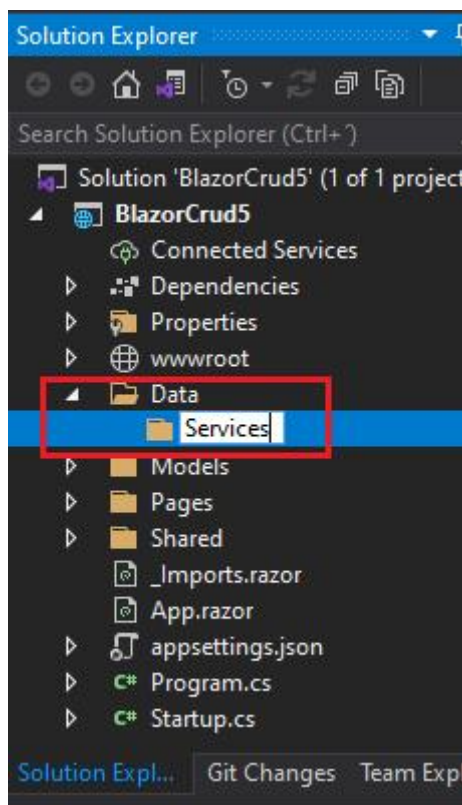
Nos aparecerán 2 errores, en el primero debemos agregar la referencia **using BlazorCrud5.Models**, ya que **AppDbContext** se creó en el archivo **AppDbContext.cs**.



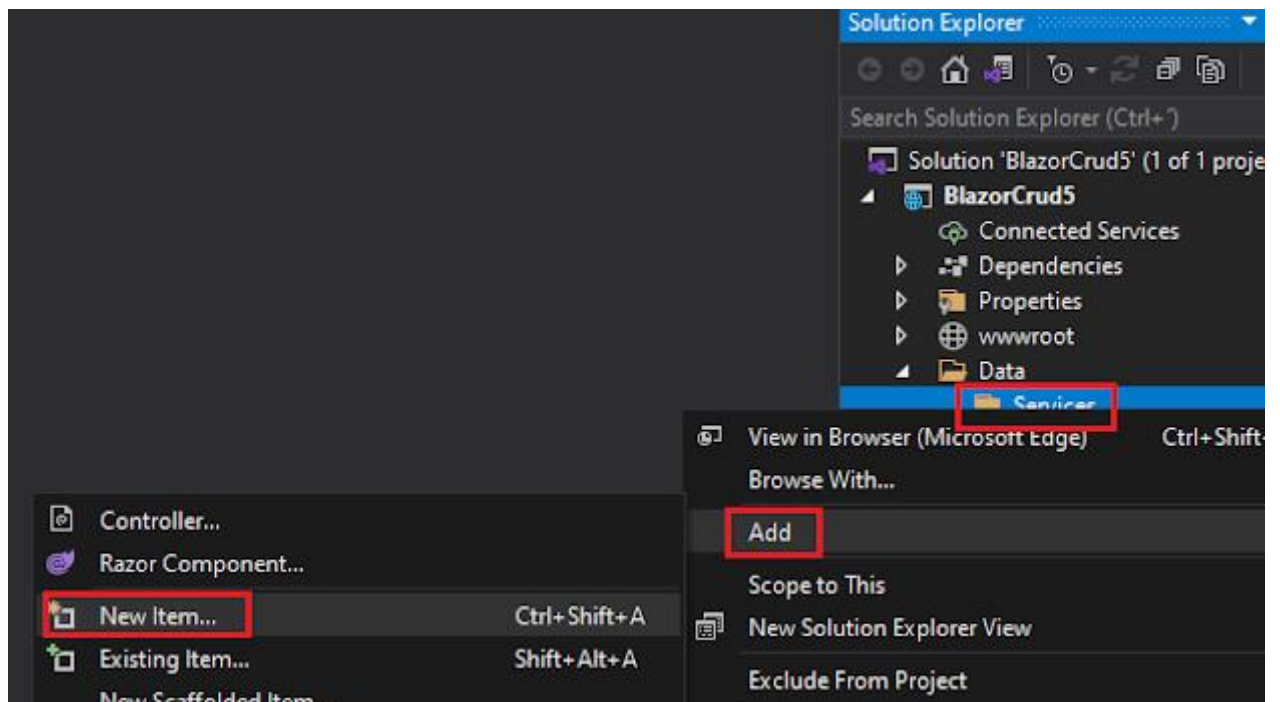
En el segundo error se debe agregar la referencia **using Microsoft.EntityFrameworkCore**.



Ahora debemos crear un repositorio de datos, es una clase que contiene toda la lógica de acceso a los datos, para esto creamos la carpeta llama **Services** en **Data**.



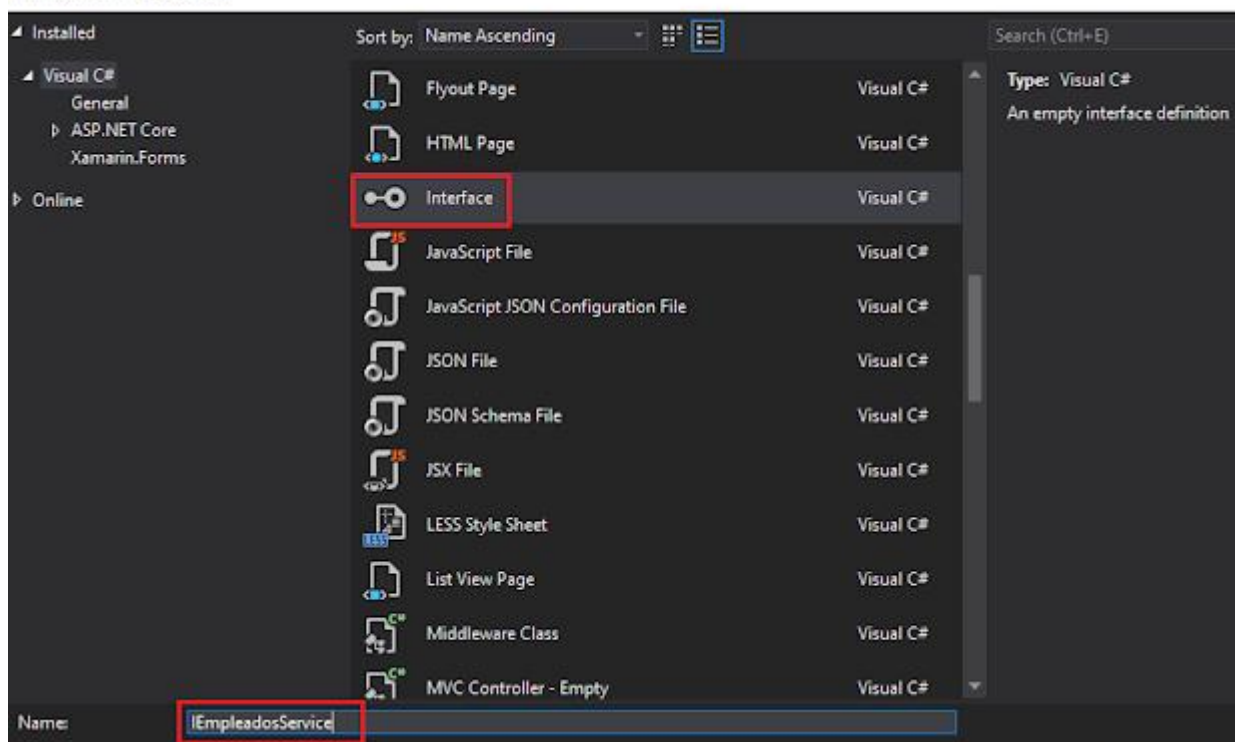
Creamos una interface que realizara la firma de los métodos a utilizar, sobre la carpeta **Services** botón derecho **Add/New Item**



Seleccionamos **Interface** y tecleamos el nombre de la interface, por standard debe tener una "I" al inicio del nombre.

Al utilizar una interface estas condicionado a utilizar todos los métodos que contengan la interface, es por ello que lo llaman "**un contrato**" donde debes de cumplir todas las cláusulas, si posteriormente ingresar un nuevo métodos esté lo deberás utilizar en la clase donde implementes la interface.

Add New Item - BlazorCrud5



Hacemos la interface pública.


```
EmpleadoService.cs* X
BlazorCrud5
1 using System;
2     using System.Collections.Generic;
3     using System.Linq;
4     using System.Threading.Tasks;
5
6     namespace BlazorCrud5.Data.Services
7     {
8         public interface IEmpleadoService
9         {
10         }
11     }
```

Ahora teclaremos los métodos para realizar el **CRUD**, todos los métodos deberán llevar al inicio **Task** ya que utilizaremos **programación asincrona**.

En caso de visualizar algunos errores solo deberás verificar las posibles soluciones.

```
EmpleadoService.cs* X
BlazorCrud5
1 using System;
2     using System.Collections.Generic;
3     using System.Linq;
4     using System.Threading.Tasks;
5
6     namespace BlazorCrud5.Data.Services
7     {
8         public interface IEmpleadoService
9         {
10             //Devuelve una lista de empleados
11             public Task<List<Empleado>> ObtenerTodos();
12             //Recibe el id de empleado para obtener un empleado
13             public Task<Empleado> ObtenerPorId(int id);
14             //Recibe el objeto Empleado para crear un nuevo empleado
15             public Task<Empleado> Crear(Empleado empleado);
16             //Recibe el id de empleado para actualizar un empleado
17             public Task<Empleado> Actualizar(Empleado empleado);
18             //Recibe el id de empleado para eliminar un empleado
19             public Task<Empleado> Eliminar(int id);
20         }
21     }
```

CS0246 The type or namespace name 'Empleado' could not be found (are you missing a using directive or an assembly reference?)

using BlazorCrud5.Models;

Los métodos para el **CRUD** son los siguientes.


```
EmpleadoService.cs* x
BlazorCrud5 - BlazorCrud5.Data.Services.EmpleadoService

1 using BlazorCrud5.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6
7 namespace BlazorCrud5.Data.Services
8 {
9     0 references
10     public interface IEmpleadoService
11     {
12         //Devuelve una lista de empleados
13         0 references
14         public Task<List<Empleado>> ObtenerTodos();
15         //Recibe el id del empleado para obtener un empleado
16         0 references
17         public Task<Empleado> ObtenerId(int id);
18         //Recibe el objeto de tipo empleado que se insertara
19         0 references
20         public Task<Empleado> Crear(Empleado empleadocrear);
21         //Recibe el id y el objeto de tipo empleado que trae los datos a actualizar
22         0 references
23         public Task<Empleado> Actualizar(int id, Empleado empleadoactualizar);
24         //Recibe el id del empleado a eliminar
25         0 references
26         public Task<Empleado> Eliminar(int id);
27     }
28 }
```

Ahora debemos crear una clase que implemente la interface, crearemos la clase **EmpleadoService.cs** dentro de **Data/Services**.

```
EmpleadoService.cs x IEmpleadoService.cs
BlazorCrud5 - BlazorCrud5.Data.S

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5
6 namespace BlazorCrud5.Data.Services
7 {
8     0 references
9     public class EmpleadoService
10     {
11     }
12 }
```

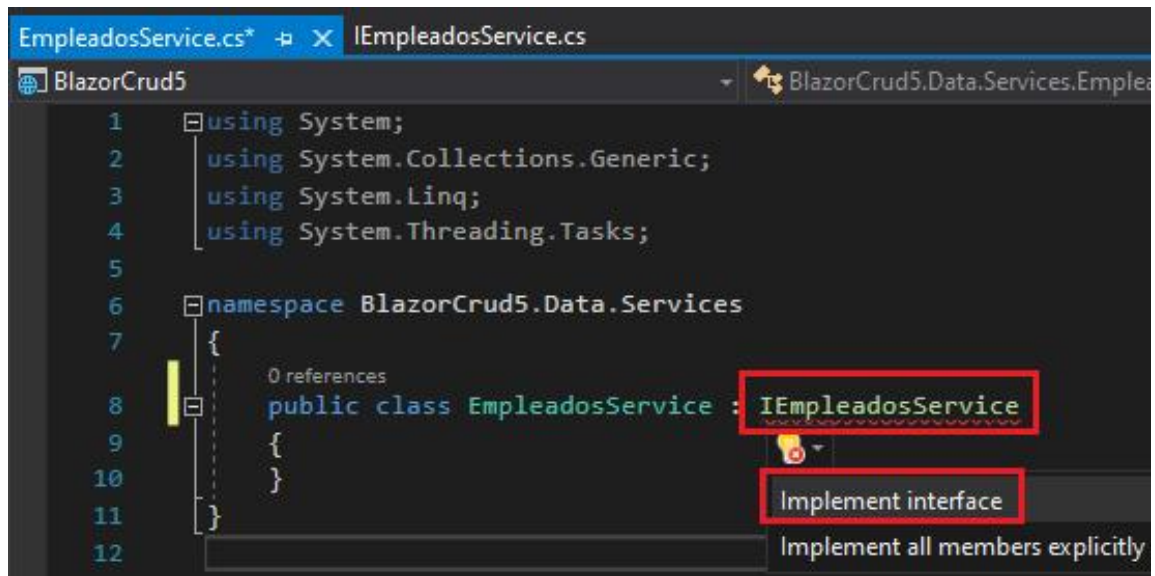
Solution Explorer

Search Solution Explorer (Ctrl+)

Solution 'BlazorCrud5' (1 of 1 project)

- BlazorCrud5
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - Data
 - Services
 - EmpleadoService.cs
 - IEmpleadoService.cs
 - Models
 - Pages
 - Shared
 - _Imports.razor
 - App.razor
 - appsettings.json

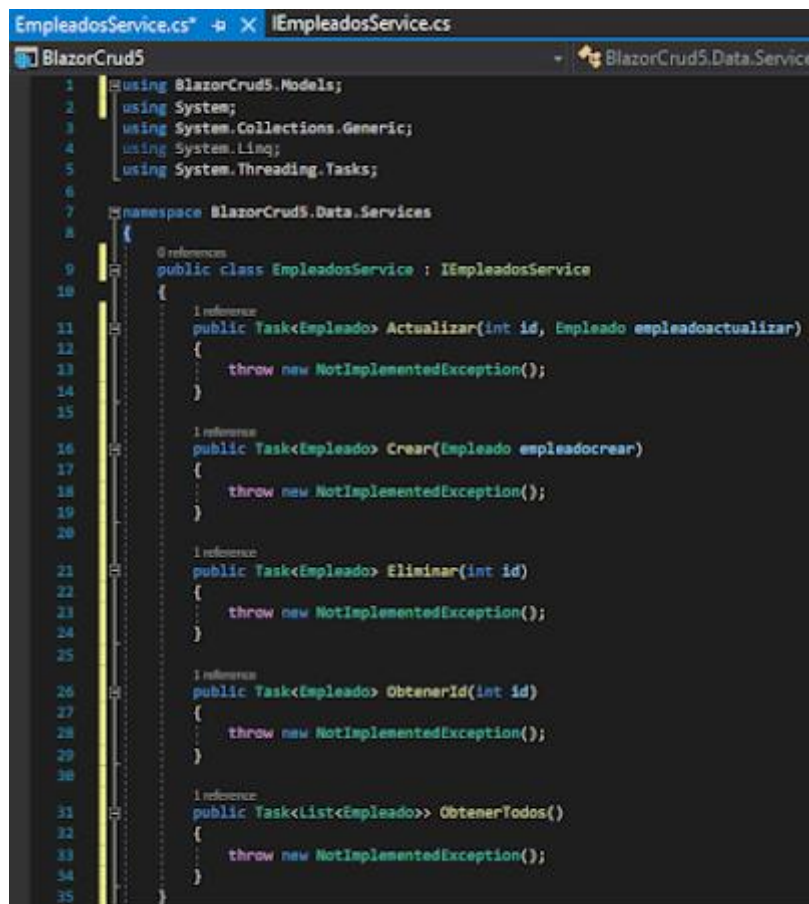
Implementaremos la interface agregando ": IEmpleadoService", para llamar todos los método nos posicionamos sobre la interface y seleccionamos **Implement interface**.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5
6 namespace BlazorCrud5.Data.Services
7 {
8     public class EmpleadosService : IEmpleadosService
9     {
10     }
11 }
12
```

Implement interface
Implement all members explicitly

Aparecera todos los métodos de la interface.



```
1 using BlazorCrud5.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6
7 namespace BlazorCrud5.Data.Services
8 {
9     public class EmpleadosService : IEmpleadosService
10     {
11         public Task<Empleado> Actualizar(int id, Empleado empleadoactualizar)
12         {
13             throw new NotImplementedException();
14         }
15
16         public Task<Empleado> Crear(Empleado empleadocrear)
17         {
18             throw new NotImplementedException();
19         }
20
21         public Task<Empleado> Eliminar(int id)
22         {
23             throw new NotImplementedException();
24         }
25
26         public Task<Empleado> ObtenerId(int id)
27         {
28             throw new NotImplementedException();
29         }
30
31         public Task<List<Empleado>> ObtenerTodos()
32         {
33             throw new NotImplementedException();
34         }
35     }
36 }
```

Configuramos el contexto de datos para conectarnos a la base de datos que esta declarado en **Startup.cs**.

```
EmpleadosService.cs*  X  IEmpleadosService.cs
BlazorCrud5.Data.Services.EmpleadosService

using BlazorCrud5.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace BlazorCrud5.Data.Services
{
    1 reference
    public class EmpleadosService : IEmpleadosService
    {
        //Configuramos el contexto de datos para conectarnos a la base de datos
        //llamado el contexto que esta configurado en Startup.cs
        private readonly AppDbContext _context;
        0 references
        public EmpleadosService(AppDbContext context)
        {
            _context = context;
        }
        1 reference
        public Task<Empleado> Actualizar(int id, Empleado empleadoactualizar)...
```

Por ahora solo utilizaremos el método **ObtenerTodos**, enviaremos todos los registros de la tabla **Empleados** a una lista con **ToList**, pero como es mejor realizar las llamadas asíncronas agregamos **Async** al final, **Async** al Task y un **await** en el return.

```
EmpleadosService.cs*  X  IEmpleadosService.cs
BlazorCrud5

12 {
13     1 reference
14     public class EmpleadosService : IEmpleadosService
15     {
16         //Configuramos el contexto de datos para conectarnos a la base de datos
17         //que esta declarado en Startup.cs.
18         private readonly AppDbContext _context;
19         0 references
20         public EmpleadosService(AppDbContext context)
21         {
22             _context = context;
23         }
24         1 reference
25         public Task<Empleado> Actualizar(int id, Empleado empleadoactualizar)...
```

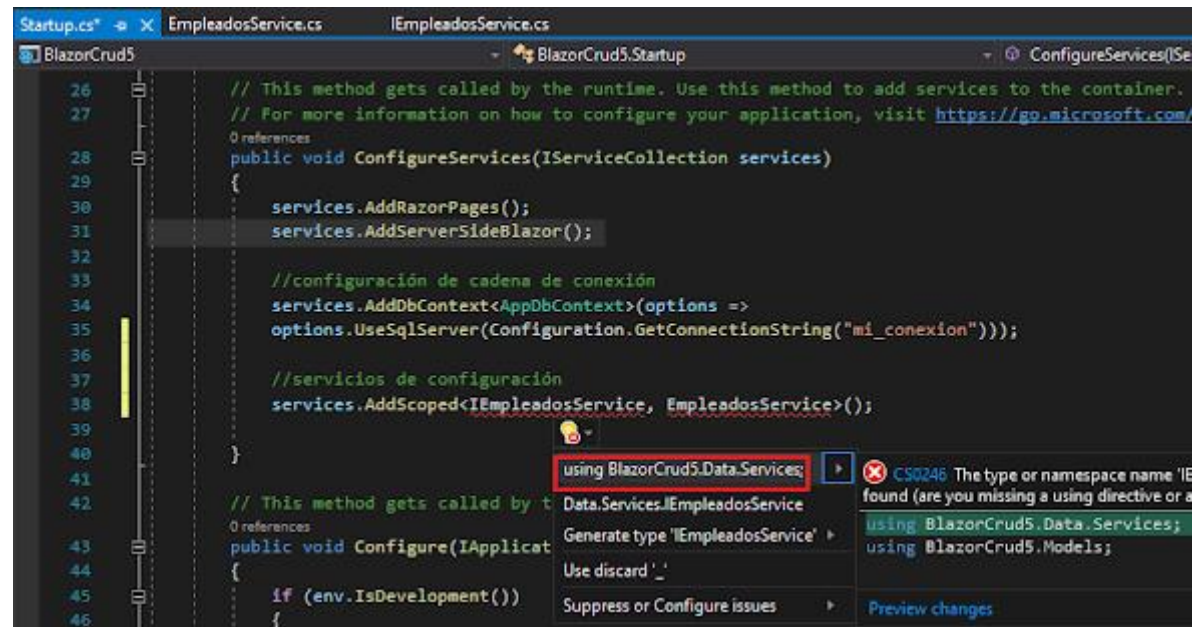
```
26
27         1 reference
28         public Task<Empleado> Crear(Empleado empleadocrear)...
```

```
29
30         1 reference
31         public Task<Empleado> Eliminar(int id)...
```

```
32
33         1 reference
34         public Task<Empleado> ObtenerId(int id)...
```

```
35
36
37         1 reference
38         public async Task<List<Empleado>> ObtenerTodos()
39         {
40             return await _context.Empleados.ToListAsync();
41         }
42
43
44
45
46
```


Ahora debemos registrar el servicio en el contenedor de inyección de dependencias para poder utilizarlo, para ellos, ingresamos a **Startup.cs**, tecleamos en **ConfigureServices** lo siguiente: **services.AddScoped<IEmpleadosService, EmpleadosService>()**; primero se indica la interface después la clase que lo implementa y por último la ruta donde se ubican los servicios.



```
26 // This method gets called by the runtime. Use this method to add services to the container.
27 // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398109
28 public void ConfigureServices(IServiceCollection services)
29 {
30     services.AddRazorPages();
31     services.AddServerSideBlazor();
32
33     //configuración de cadena de conexión
34     services.AddDbContext<AppDbContext>(options =>
35     options.UseSqlServer(Configuration.GetConnectionString("mi_conexion")));
36
37     //servicios de configuración
38     services.AddScoped<IEmpleadosService, EmpleadosService>();
39 }
40
41 // This method gets called by the runtime. Use this method to add services to the container.
42 // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398109
43 public void Configure(IApplicationBuilder app)
44 {
45     if (env.IsDevelopment())
46     {
47         app.UseDeveloperExceptionPage();
48     }
49     else
50     {
51         app.UseExceptionHandler("/Error");
52         // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts
53         app.UseHsts();
54     }
55     app.UseStaticFiles();
56     app.UseRouting();
57     app.UseAuthorization();
58     app.UseEndpoints(endpoints =>
59     {
60         endpoints.MapRazorPages();
61         endpoints.MapBlazorRoute();
62     });
63 }
```

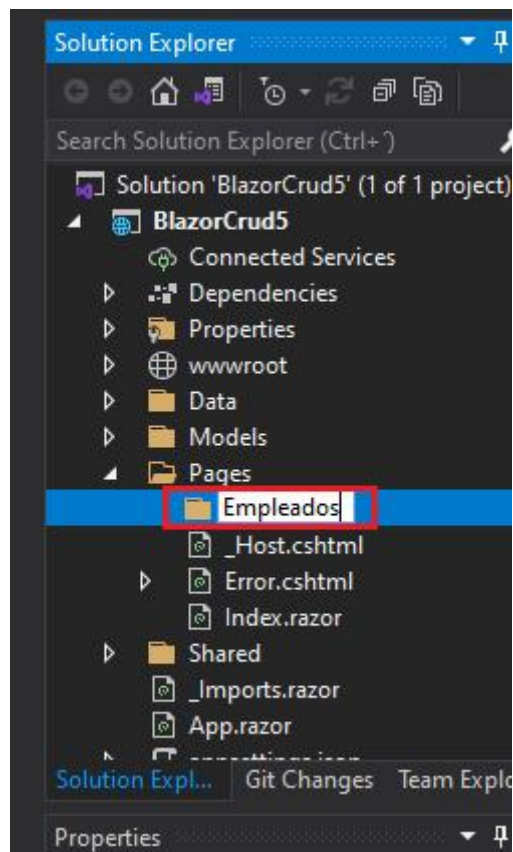
Code completion menu options:

- using BlazorCrud5.Data.Services;
- Data.Services.IEmpleadosService
- Generate type 'IEmpleadosService'
- Use discard '_'
- Suppress or Configure issues

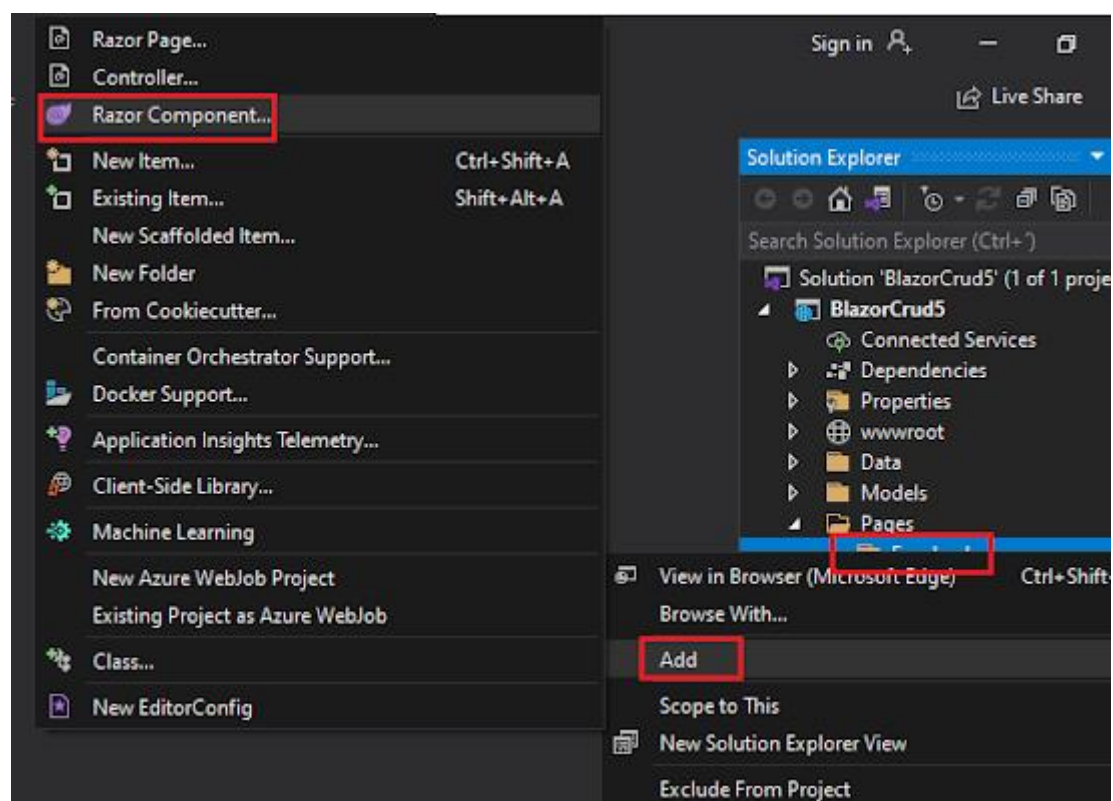
Error message: CS0246 The type or namespace name 'IEmpleadosService' could not be found (are you missing a using directive or an assembly reference?)

Suggested fix: using BlazorCrud5.Data.Services;

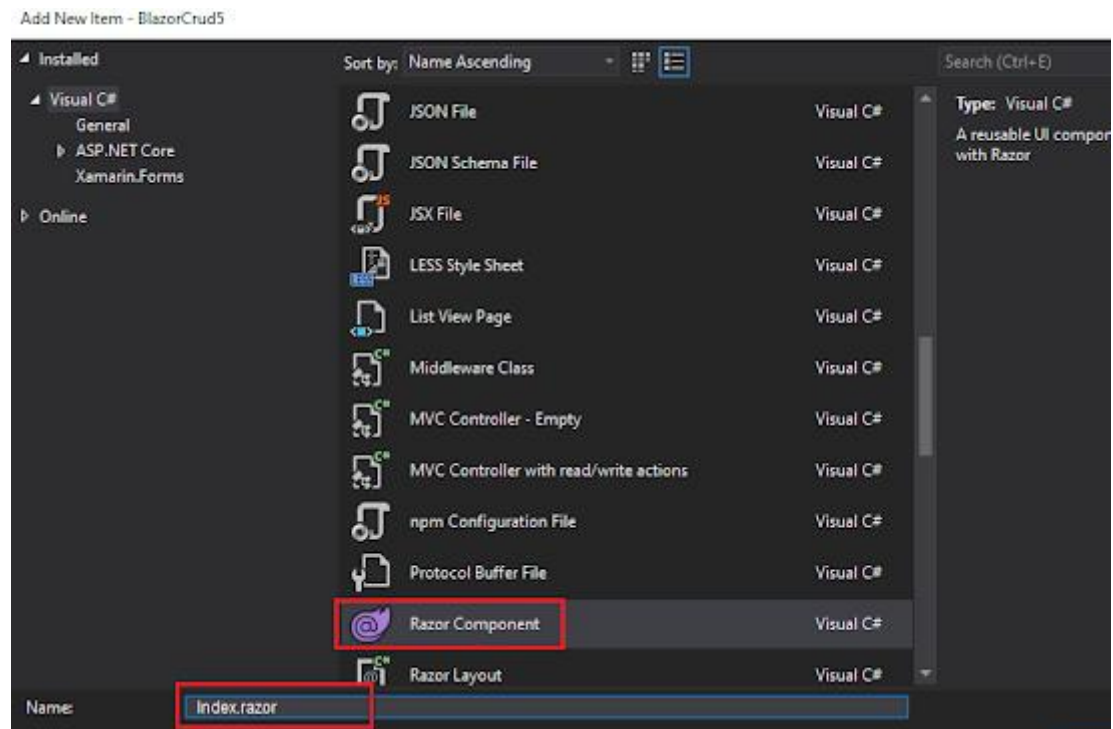
Lo siguiente será crear la **Page** donde se visualizaran los registros, para ellos crearemos una carpeta en **Pages** con el nombre **Empleados**, en ella almacenaremos todas las páginas que utilizaremos para visualizar, crear, actualizar o eliminar los empleados.



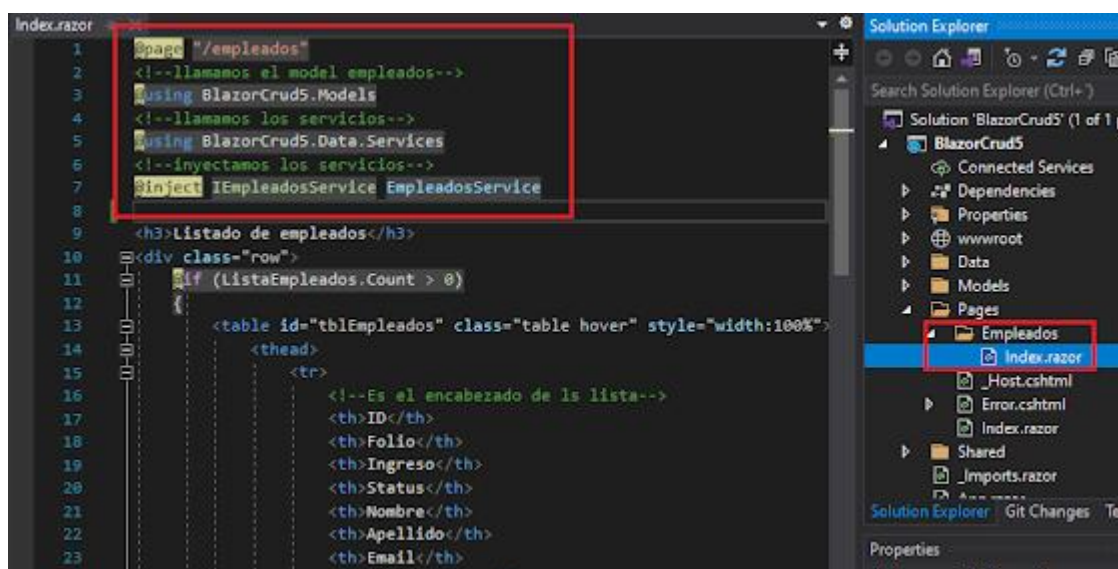
Crearemos un archivo `.razor` para visualizar los registros de los empleados, botón derecho sobre la carpeta **Empleados** **Add/Razor Component...**



Teclamos el nombre de nuestro archivo `.razor`.



Editamos **Index.razor** y tecleamos el siguiente código, llamamos el **Model**, los servicios donde de ubican las **Interfaces** y la **inyección** de los servicios.



Tecleamos los titulo que apatecen en la tabla.

```
Index.razor X
3 using BlazorCrud5.Models
4 <!-- llamamos los servicios -->
5 using BlazorCrud5.Data.Services
6 <!-- inyectamos los servicios -->
7 @inject IEmpleadosService EmpleadosService
8
9 <h3>Listado de empleados</h3>
10 <div class="row">
11     @if (ListaEmpleados.Count > 0)
12     {
13         <table id="tblEmpleados" class="table hover" style="width:100%">
14             <thead>
15                 <tr>
16                     <!-- Es el encabezado de la lista -->
17                     <th>ID</th>
18                     <th>Folio</th>
19                     <th>Ingreso</th>
20                     <th>Status</th>
21                     <th>Nombre</th>
22                     <th>Apellido</th>
23                     <th>Email</th>
24                     <th>Departamento</th>
25                     <th>Función</th>
26                     <th>Baja</th>
27                     <th>Sueldo</th>
28                     <th></th>
29                 </tr>
30             </thead>
31             <tbody>
32                 @foreach (var item in ListaEmpleados)
33                 {
```

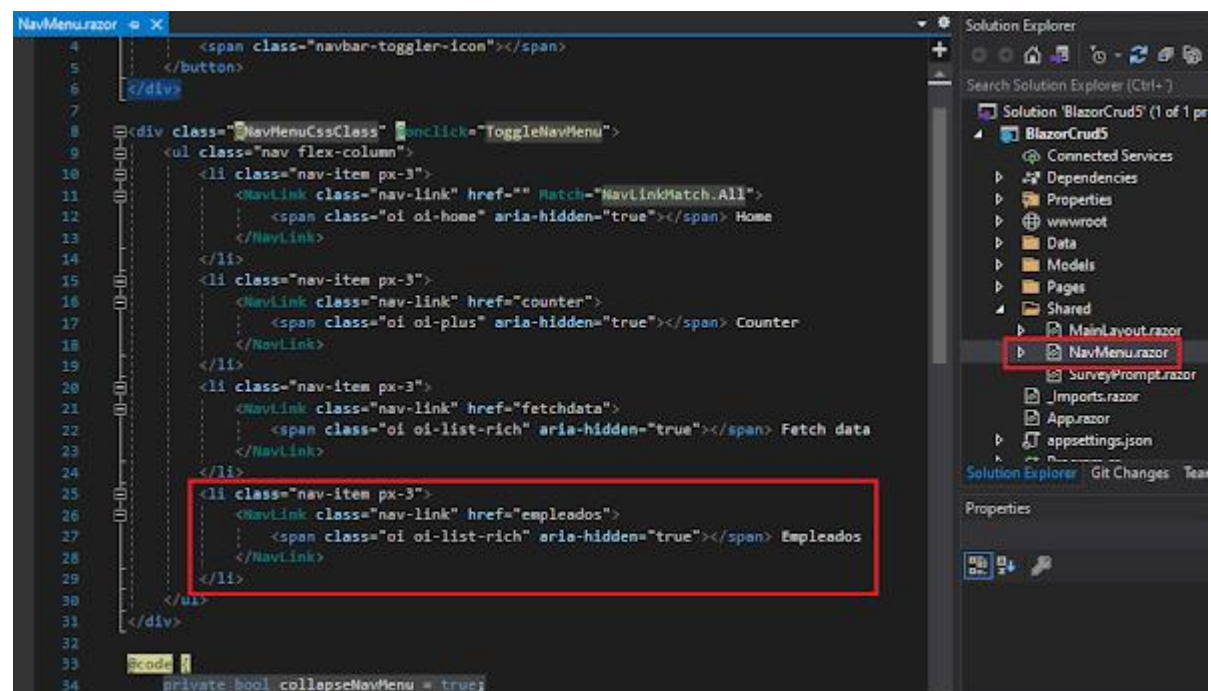
Con un **@foreach** recorreremos la tabla de la base de datos para llamar los registros y agregamos 2 botones para editar y eliminar los registros.

```
Index.razor  X
27      <th>Sueldo</th>
28      <th></th>
29  </tr>
30  </thead>
31  <tbody>
32      @foreach (var item in ListaEmpleados)
33      {
34          <tr>
35              <td>@item.IntId</td>
36              <td>@item.IntFolio</td>
37              <td>@item.DatIngreso</td>
38              <td>@item.StrStatus</td>
39              <td>@item.StrNombre</td>
40              <td>@item.StrApellido</td>
41              <td>@item.StrEmail</td>
42              <td>@item.StrDepartamento</td>
43              <td>@item.StrFuncion</td>
44              <td>@item.DatBaja</td>
45              <td>@item.IntSueldo</td>
46              <td>
47                  <a class="btn btn-primary mr-2" href="#">Editar</a>
48                  <a class="btn btn-danger mr-2" href="#">Borrar</a>
49              </td>
50          </tr>
51      }
52  </tbody>
53  </table>
```

Por último llamaremos los datos del repositorio **Services** utilizamos el método **OnInitializedAsync** que se ejecuta en cuanto se ingrese al componente, le asignamos **async**.

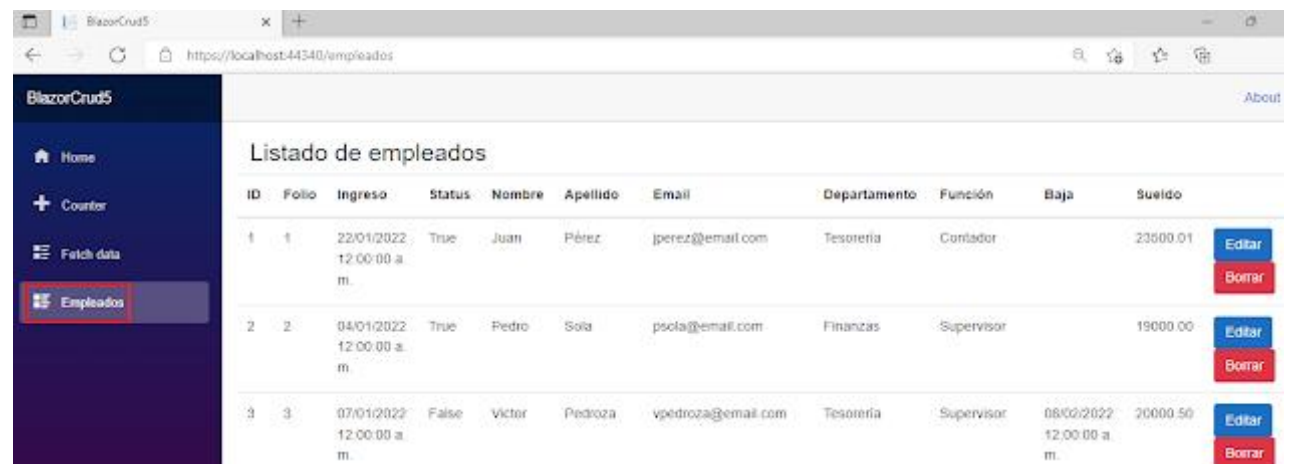
```
Index.razor  X
51      }
52  </tbody>
53  </table>
54  }
55  else
56  {
57      <div class="text-center">
58          <h3 class="alert-danger"> No existen registro</h3>
59      </div>
60  }
61  </div>
62
63  @code
64  //se recibe el listado de empleados y le podremos el nombre ListaEmpleados
65  private List<Empleado> ListaEmpleados = new List<Empleado>();
66
67  //para obtener los datos del repositorio services utilizamos el método
68  //OnInitializedAsync que se ejecuta en cuanto se entra al componente
69  //lo volvemos async
70  protected override async Task OnInitializedAsync()
71  {
72      //la ListaEmpleado obtiene los datos del método ObtenerTodos del
73      //repositorio EmpleadosService
74      ListaEmpleados = await EmpleadosService.ObtenerTodos();
75  }
76
77  }
```


Ahora editamos **Shared/NavMenu.razor** para agregar una opción en el menu principal que llamará el módulo de empleados.



```
4 <span class="navbar-toggler-icon"></span>
5 </button>
6 </div>
7
8 <div class="NavMenuCssClass" onclick="ToggleNavMenu">
9   <ul class="nav flex-column">
10    <li class="nav-item px-3">
11      <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
12        <span class="oi oi-home" aria-hidden="true"></span> Home
13      </NavLink>
14    </li>
15    <li class="nav-item px-3">
16      <NavLink class="nav-link" href="counter">
17        <span class="oi oi-plus" aria-hidden="true"></span> Counter
18      </NavLink>
19    </li>
20    <li class="nav-item px-3">
21      <NavLink class="nav-link" href="fetchdata">
22        <span class="oi oi-list-rich" aria-hidden="true"></span> Fetch data
23      </NavLink>
24    </li>
25    <li class="nav-item px-3">
26      <NavLink class="nav-link" href="empleados">
27        <span class="oi oi-list-rich" aria-hidden="true"></span> Empleados
28      </NavLink>
29    </li>
30  </ul>
31 </div>
32
33 @code {
34   private bool collapseNavMenu = true;
```

Ejecutamos la app y podemos ver el módulo empleados que mostrado los registros agregados manualmente.

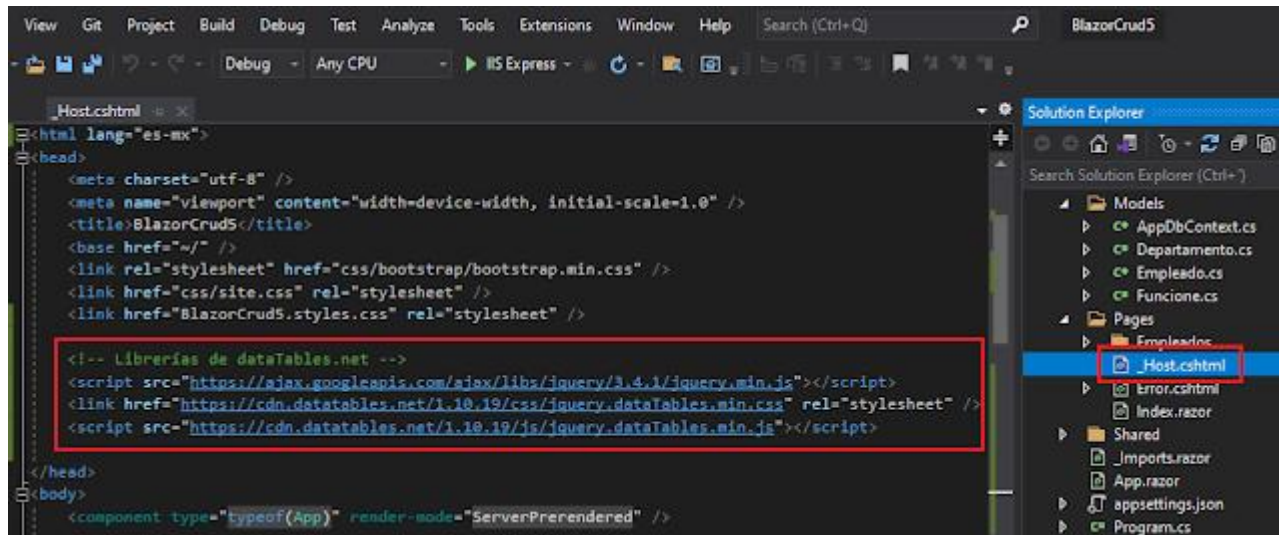


ID	Folio	Ingreso	Status	Nombre	Apellido	Email	Departamento	Función	Baja	Sueldo	
1	1	22/01/2022 12:00:00 a. m.	True	Juan	Pérez	jperez@email.com	Tesoreria	Contador		23500.01	Editar Borrar
2	2	04/01/2022 12:00:00 a. m.	True	Pedro	Sola	psola@email.com	Finanzas	Supervisor		19000.00	Editar Borrar
3	3	07/01/2022 12:00:00 a. m.	False	Victor	Pedroza	vpedroza@email.com	Tesoreria	Supervisor	08/02/2022 12:00:00 a. m.	20000.50	Editar Borrar

LIBRERIA DATATABLES.NET

Si queremos ver un poco más profesional nuestra lista de empleados podemos utilizar la librería **datatables.net**.

Para esto editamos el archivo **_Host.cshtml** y tecleamos las referencias de la librería.

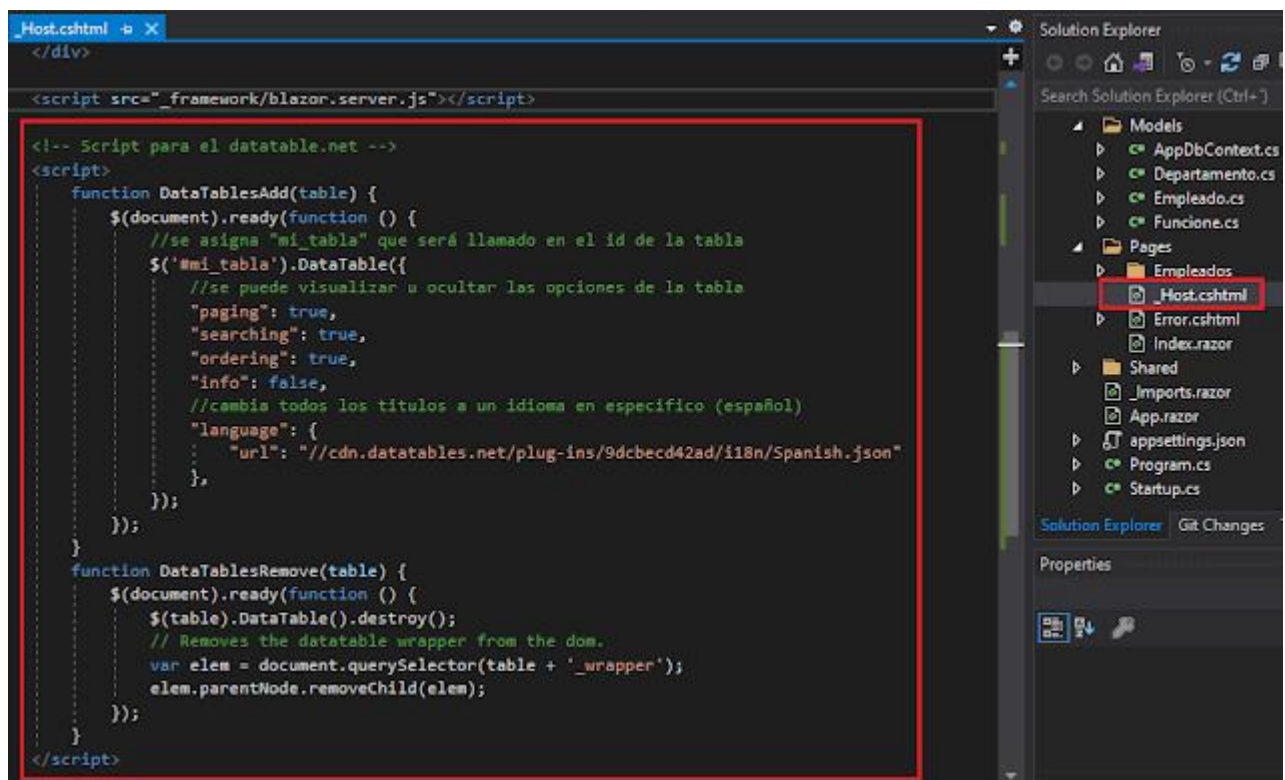


```
<html lang="es-mx">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>BlazorCrud5</title>
  <base href="/" />
  <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
  <link href="css/site.css" rel="stylesheet" />
  <link href="BlazorCrud5.styles.css" rel="stylesheet" />

  <!-- Librerías de dataTables.net -->
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <link href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" rel="stylesheet" />
  <script src="https://cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

</head>
<body>
  <component type="typeof(App)" render-mode="ServerPrerendered" />
</body>
```

También agregamos los **script** donde indicaremos los parámetros de la librería, el **id** que será llamado en la tabla del archivo **.razor** es **"mi_tabla"**..



```
</div>

<script src="_framework/blazor.server.js"></script>

<!-- Script para el datatable.net -->
<script>
  function DataTablesAdd(table) {
    $(document).ready(function () {
      //se asigna "mi_tabla" que será llamado en el id de la tabla
      $('#mi_tabla').DataTable({
        //se puede visualizar u ocultar las opciones de la tabla
        "paging": true,
        "searching": true,
        "ordering": true,
        "info": false,
        //cambia todos los títulos a un idioma en específico (español)
        "language": {
          "url": "https://cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"
        },
      });
    });
  }

  function DataTablesRemove(table) {
    $(document).ready(function () {
      $(table).DataTable().destroy();
      // Removes the datatable wrapper from the dom.
      var elem = document.querySelector(table + '_wrapper');
      elem.parentNode.removeChild(elem);
    });
  }
</script>
```

En el archivo **Index.razor**, agregamos las referencias y agregamos el **id="mi_tabla"**.

```
1 @page "/empleados"
2 <!-- llamamos el model empleados -->
3 using BlazorCrud5.Models
4 <!-- llamamos los servicios -->
5 using BlazorCrud5.Data.Services
6 <!-- inyectamos los servicios -->
7 @inject IEmpleadosService EmpleadosService
8
9 <!-- llama funciones JS desde Blazor -->
10 @inject JSRuntime JSRuntime
11 <!-- Proporciona la liberación de recursos-->
12 implements IDisposable
13
14 <h3>Listado de empleados</h3>
15 <div class="row">
16     @if (ListaEmpleados.Count > 0)
17     {
18         <table id="mi_tabla" class="table hover" style="width:100%">
19             <thead>
20                 <tr>
21                     <!-- Es el encabezado de la lista-->
22                     <th>ID</th>
23                     <th>Folio</th>
24                     <th>Ingreso</th>
25                     <th>Status</th>
26                     <th>Nombre</th>
27                     <th>Apellido</th>
28                     <th>Email</th>
```

Agregamos 2 métodos más, cuando los componentes terminan de renderizar se ejecuta **OnAfterRenderAsync**, **IDisposable.Dispose()** libera todos los recursos que ha adquirido pero que aún no libera del **Document Object Model (DOM)**.

```
67
68
69 //se recibe el listado de empleados y le podremos el nombre ListaEmpleados
70 private List<Empleado> ListaEmpleados = new List<Empleado>();
71
72 //para obtener los datos del repositorio services utilizamos el método
73 //OnInitializedAsync que se ejecuta en cuanto se entra al componente
74 //lo volvemos async
75 protected override async Task OnInitializedAsync()
76 {
77     //la listaEmpleado obtiene los datos del método ObtenerTodos del
78     //repositorio EmpleadosService
79     ListaEmpleados = await EmpleadosService.ObtenerTodos();
80 }
81
82 //Métodos para el datatable.net
83 //Cuando los componentes terminan de renderizar se ejecuta OnAfterRenderAsync
84 protected override async Task OnAfterRenderAsync(bool firstRender) {
85     await JSRuntime.InvokeAsync<object>("DataTablesAdd", "#mi_tabla");
86 }
87
88 //Libera todos los recursos que ha adquirido pero que aún no libera
89 void IDisposable.Dispose()
90 {
91     JSRuntime.InvokeAsync<object>("DataTablesRemove", "#mi_tabla");
92 }
93
94
```

Ejecutamos la app y revisamos las mejoras.

BlazorCrud5

Home

Counter

Fetch data

Empleados

Listado de empleados

Mostrar 10 registros

Buscar

ID	Ingreso	Status	Nombre	Apellido	Email	Departamento	Función	Baja	Salario	
15	02/02/2022 12:00:00 a. m.	True	Eduardo	Santillan	esantillan@email.com	Cuentas/Pagar	Asistente		15000.00	<div>Editar</div> <div>Borrar</div>
14	02/02/2022 12:00:00 a. m.	True	Ricardo	Gerda	rgarcia@email.com	Tesoreria	Contador		30000.00	<div>Editar</div> <div>Borrar</div>
13	02/02/2022 12:00:00 a. m.	True	Hector	Hernandez	hhernandez@email.com	Cuentas/Pagar	Supervisor		17000.00	<div>Editar</div> <div>Borrar</div>
12	02/02/2022 12:00:00 a. m.	True	Ramón	Caballero	rcaballero@email.com	Cobranza	Coordinador		18000.00	<div>Editar</div> <div>Borrar</div>
11	01/02/2022 12:00:00 a. m.	False	Raquel	Gómez	rgomez@email.com	Tesoreria	Asistente	06/02/2022 12:00:00 a. m.	12000.00	<div>Editar</div> <div>Borrar</div>
10	01/02/2022 12:00:00 a. m.	True	Gabriel	Mendez	gmendez@email.com	Finanzas	Contador		22000.00	<div>Editar</div> <div>Borrar</div>
9	31/01/2022 12:00:00 a. m.	True	Karla	Cacho	kcacho@email.com	Cuentas	Supervisor		20000.00	<div>Editar</div> <div>Borrar</div>
8	31/01/2022 12:00:00 a. m.	True	Lourdes	Campos	lcampos@email.com	Cuentas	Ejecutivo		16000.00	<div>Editar</div> <div>Borrar</div>
7	29/01/2022 12:00:00 a. m.	True	Maria	Becerra	mbecerra@email.com	Cuentas	Ejecutivo		15000.00	<div>Editar</div> <div>Borrar</div>
6	15/01/2022 12:00:00 a. m.	False	Lorena	Ibarrá	libarra@email.com	Producción	Asistente	06/02/2022 12:00:00 a. m.	11000.00	<div>Editar</div> <div>Borrar</div>

Anterior

1

2

Siguiente