

Universidad Autónoma de Aguascalientes
Centro de Ciencias Básicas



**UNIVERSIDAD AUTÓNOMA
DE AGUASCALIENTES**

Desarrollo Web

Documento Técnico

6° Semestre

Proyecto Final: TIENDA DE ENCHILADAS

Maestra: Margarita Mondragón Arellano

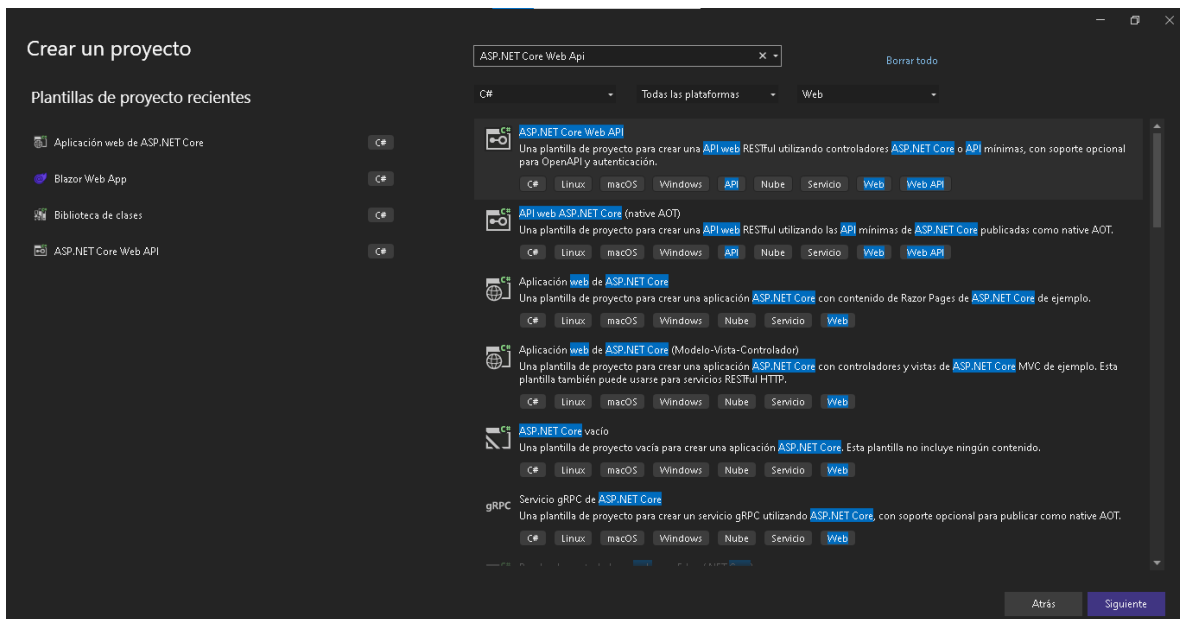
Integrantes:

Luis Antonio Jaime Vidales ID: 336273
Yair de la Riva Belmares ID: 339089

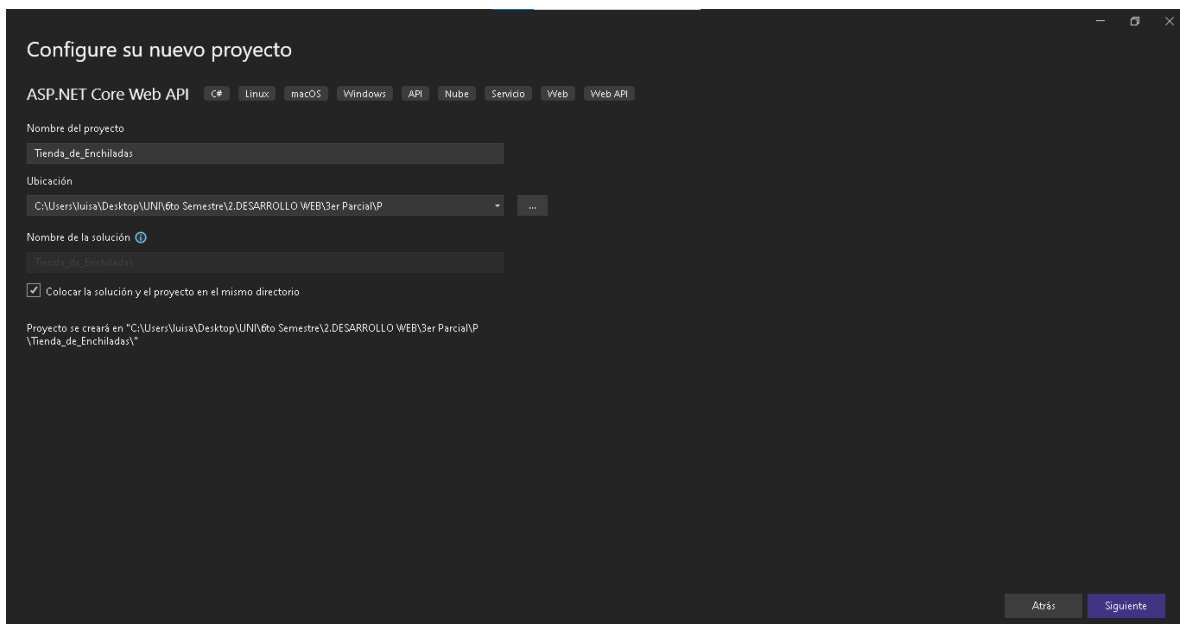
Documento Técnico

Link GitHub: <https://github.com/Luis0426/ProyectoFinalWeb>

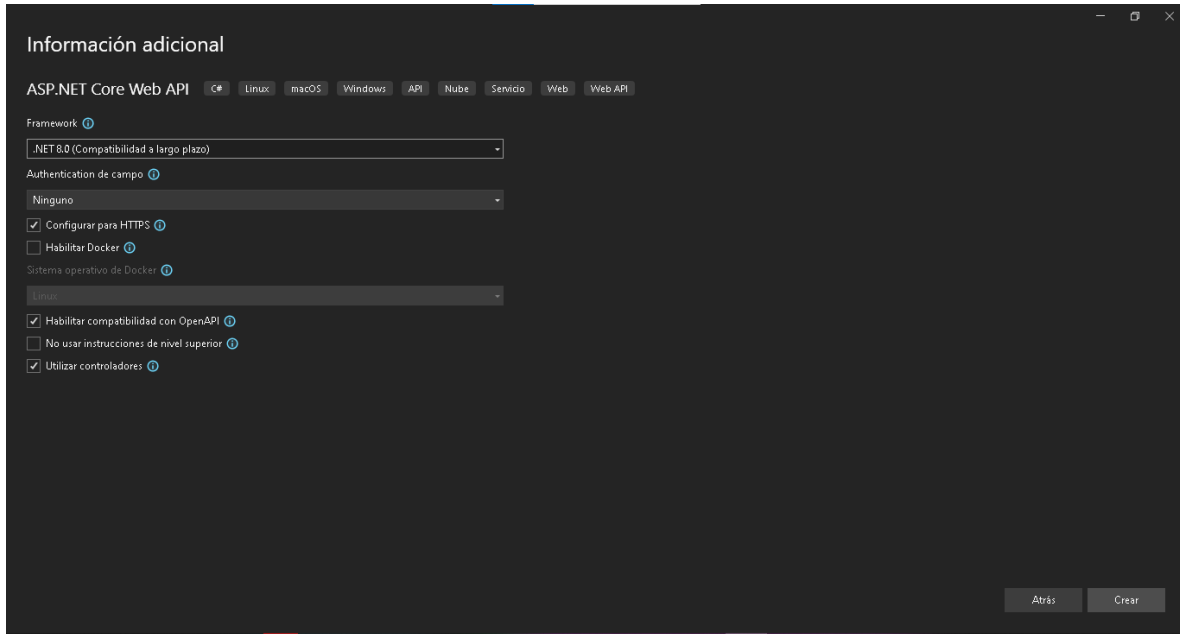
- Primero se crea un proyecto en visual studio donde se hara un proyecto de tip ASP.NET Core Web API



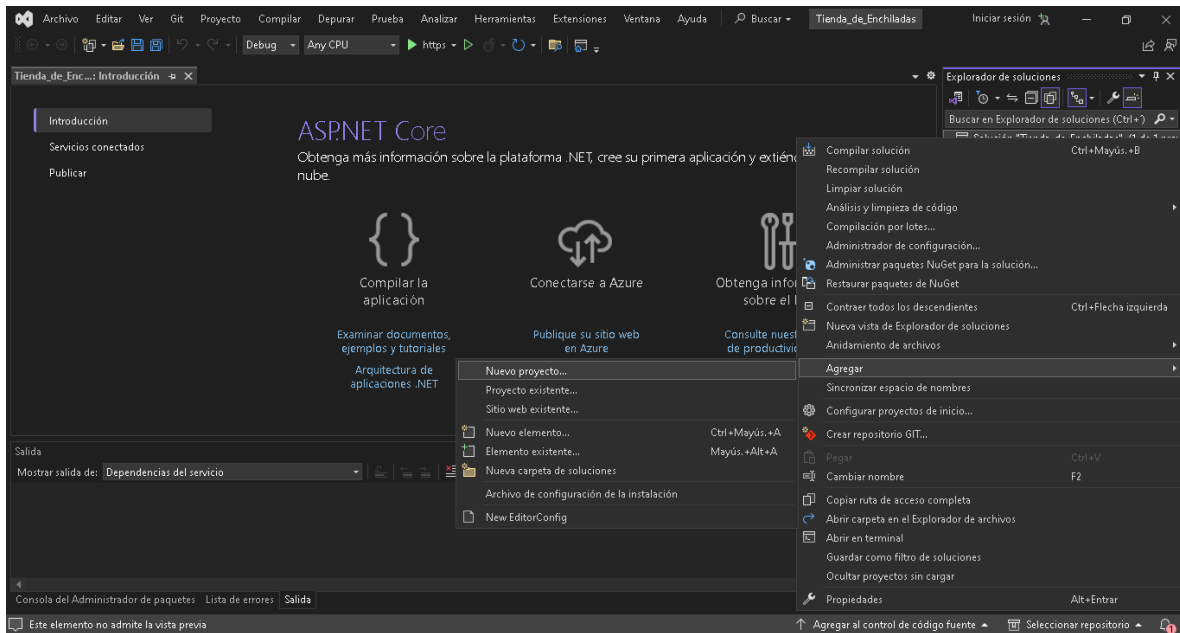
- Después de dar al botón de aceptar, se tiene que el nombre del proyecto en nuestro caso será : “Tienda de enchiladas” al asignarle el nombre damos clic en el botón de siguiente



- Al dar siguiente nos aparecerá información adicional de la configuración del proyecto que estamos creando, por recomendación dejamos por default las casillas y secciones ya configuradas y al damos al boton “crear” por consiguiente



- Al momento de crear la ventana del proyecto, nos vamos al explorador de soluciones del proyecto para crear un nuevo proyecto donde crearemos una biloteca de clases



-

- ## Configure su nuevo proyecto

Biblioteca de clases **C#** Android Linux macOS Windows Biblioteca

Nombre del proyecto

BD

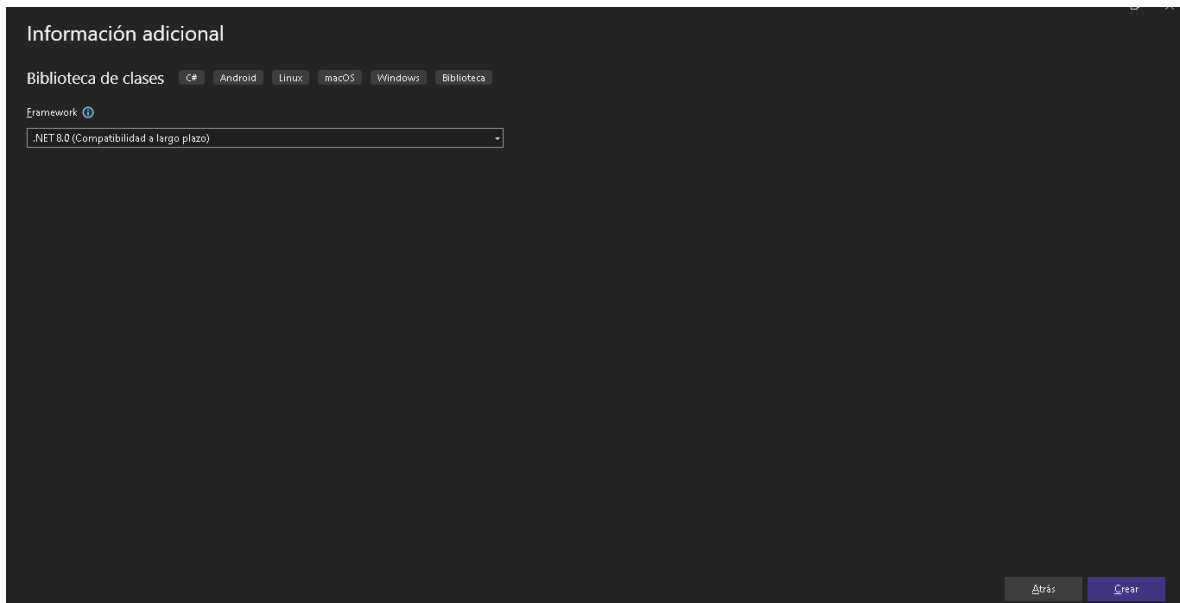
Ubicación

C:\Users\luisa\Desktop\UNIN\6to Semestre\2.DESARROLLO WEB\3er Parcial\BD

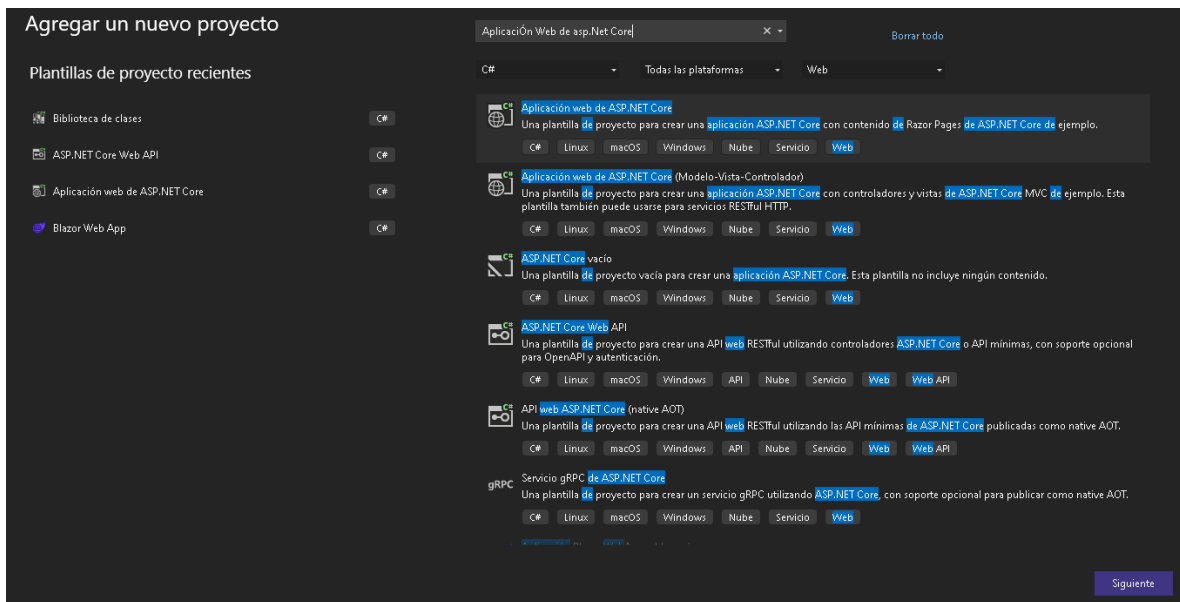
Proyecto se creará en "C:\Users\luisa\Desktop\UNIN\6to Semestre\2.DESARROLLO WEB\3er Parcial\BD"

Atrás **Siguiente**

- Al momento de crear y dar nombre a la biblioteca de clases damos en crear para crear la clase



- Después creamos otro proyecto para utilizar las páginas de razor para mostrar la interfaz gráfica de altas, bajas y cambios



- Después le asignamos el nombre del proyecto como: “Pagina_Enchiladas” y damos a crear al botón de crear proyecto

Configure su nuevo proyecto

Aplicación web de ASP.NET Core C# Linux macOS Windows Nube Servicio Web

Nombre del proyecto

Paginas Enchiladas

Ubicación

C:\Users\luisa\Desktop\UNIH\oto Semestre\2.DESARROLLO WEB\3er Parcial\IP ...

Proyecto se creará en "C:\Users\luisa\Desktop\UNIH\oto Semestre\2.DESARROLLO WEB\3er Parcial\IP\Paginas Enchiladas\"

Atrás Siguiente

- Al dar siguiente nos aparecerá información adicional de la configuración del proyecto que estamos creando, por recomendación dejamos por default las casillas y secciones ya configuradas y al damos al botón “crear” por consiguiente

Información adicional

Aplicación web de ASP.NET Core C# Linux macOS Windows Nube Servicio Web

Framework ⓘ

.NET 8.0 (Compatibilidad a largo plazo)

Authentication de campo ⓘ

Ninguno

☒ Configurar para HTTPS ⓘ

☐ Habilitar Docker ⓘ

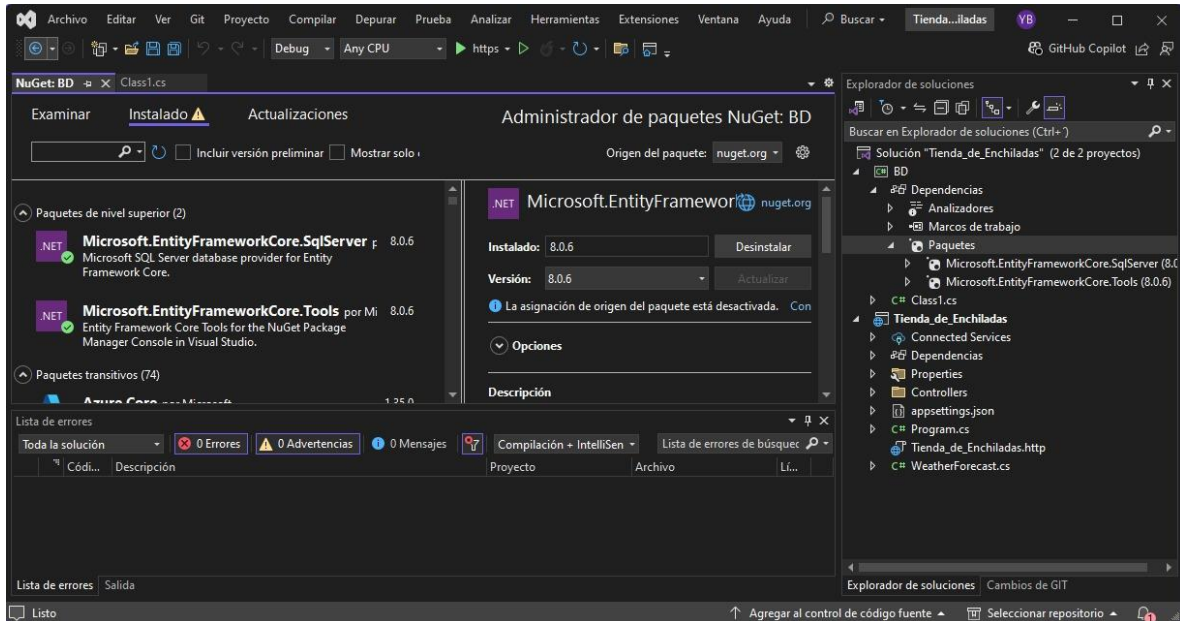
Sistema operativo de Docker ⓘ

Linux

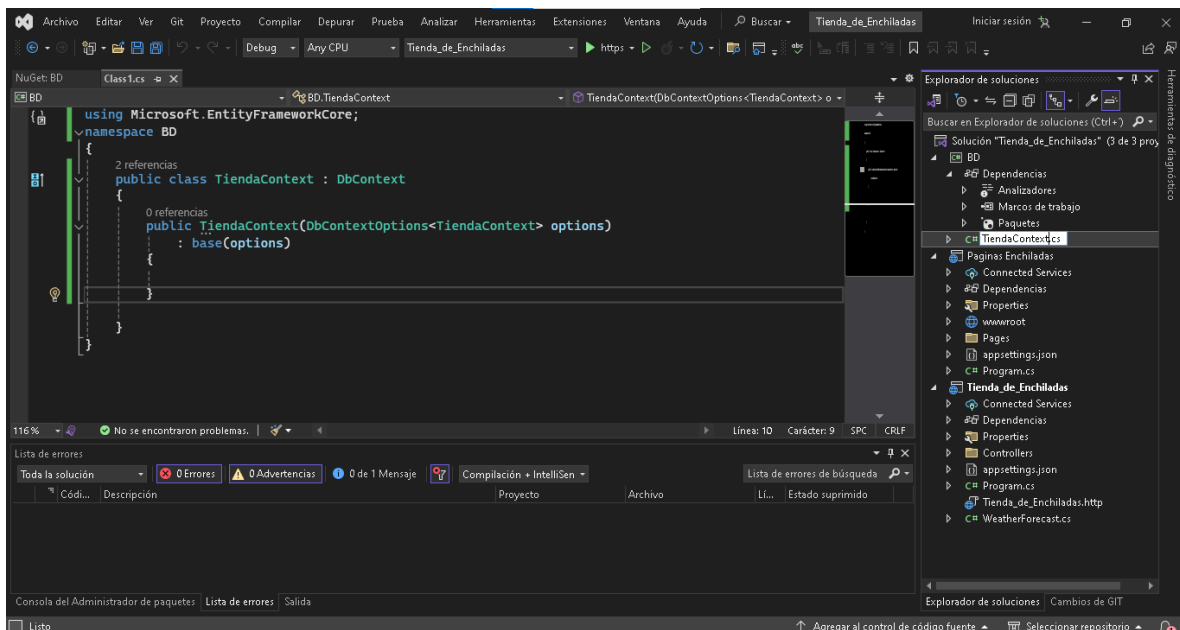
☐ No usar instrucciones de nivel superior ⓘ

Atrás Crear

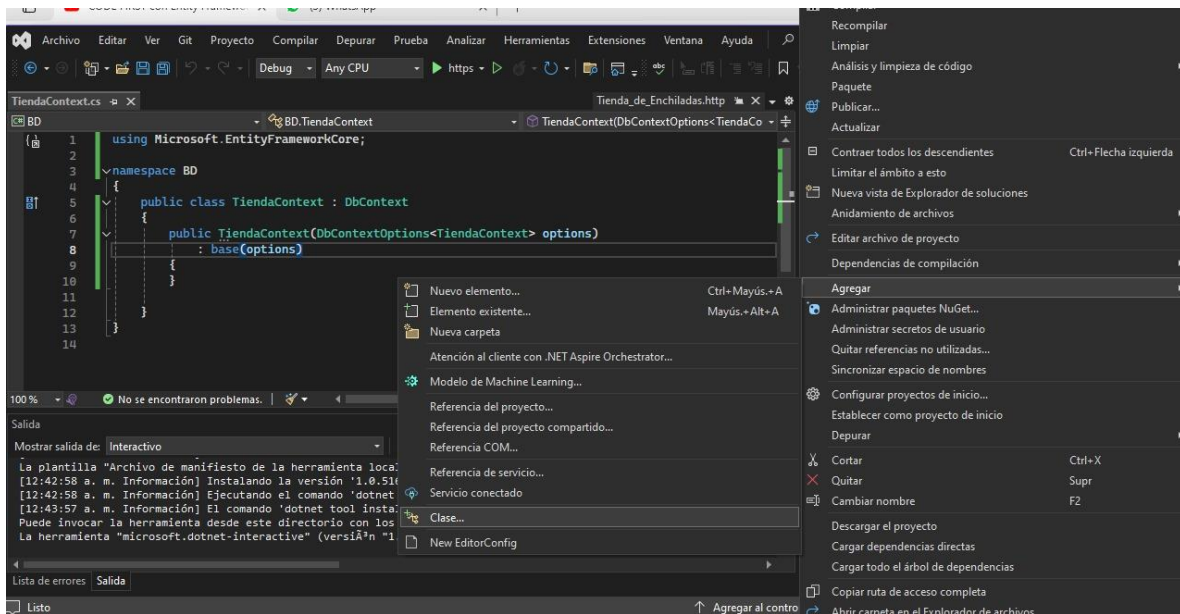
- Después nos vamos a la sección DB – dependencias damos clic derecho y damos clic en “administrador de paquetes NUGET”. Vamos a instalar los NUGETS: microsoft.EntityFrameworkCore.SqlServer y microsoft.EntityFrameworkCore.Tools esto nos servirá para poder conectarnos a la base de datos y hacer consultas y demás para gestionar la base de datos



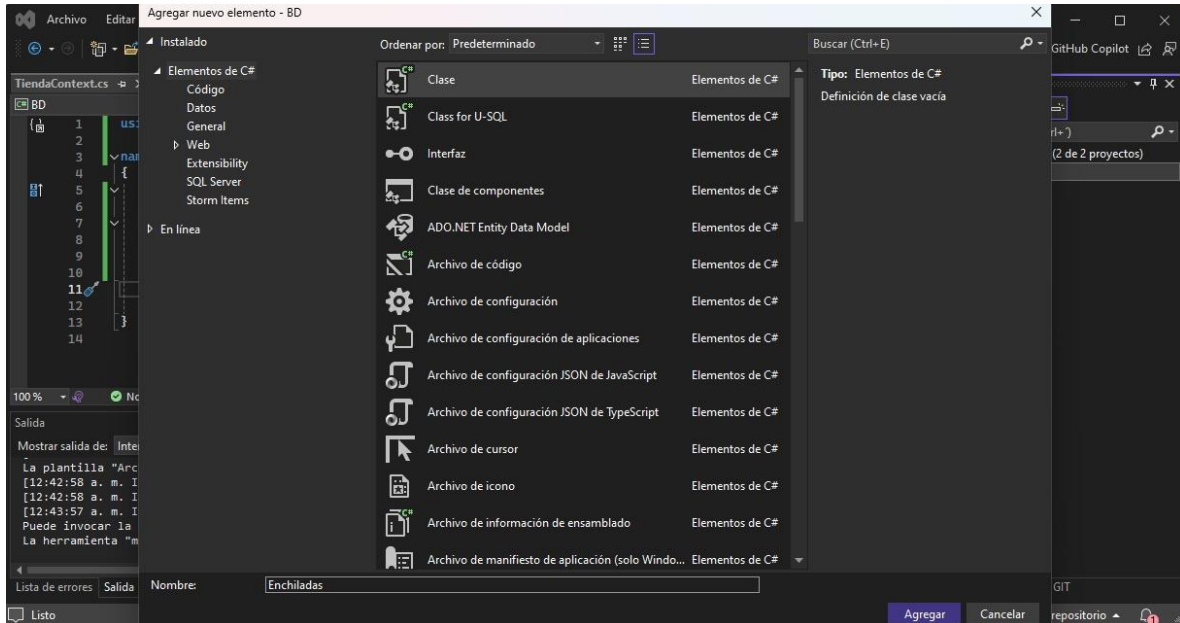
- Cuando se termine de poner la constructor se tiene que cambiar el titulo de la clase para poder enlazar con la tabla que vamos a crear



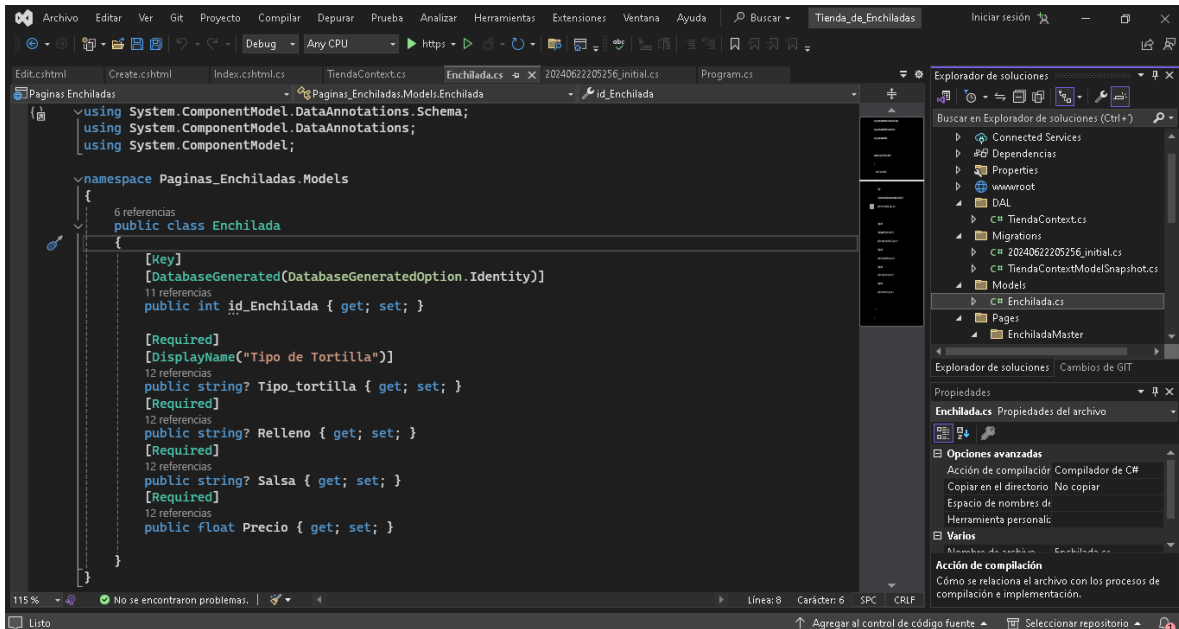
- Para poder crear una tabla a través de una clase necesitamos dar clic derecho en la clase de BD en la cual seleccionaremos el boton Agregar y despues daremos clic en el boton que dice clase



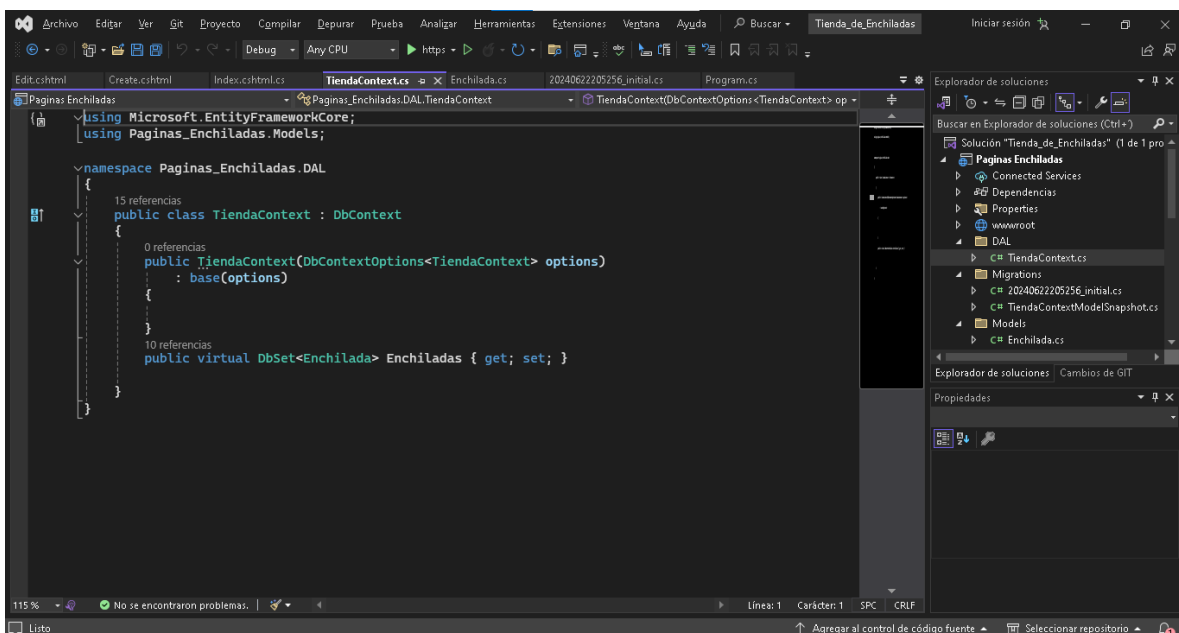
- Al mostrarnos una segunda ventana seleccionaremos “clase” y en nuestro caso le pondremos nombre “Enchilada” para simular que es la tabla



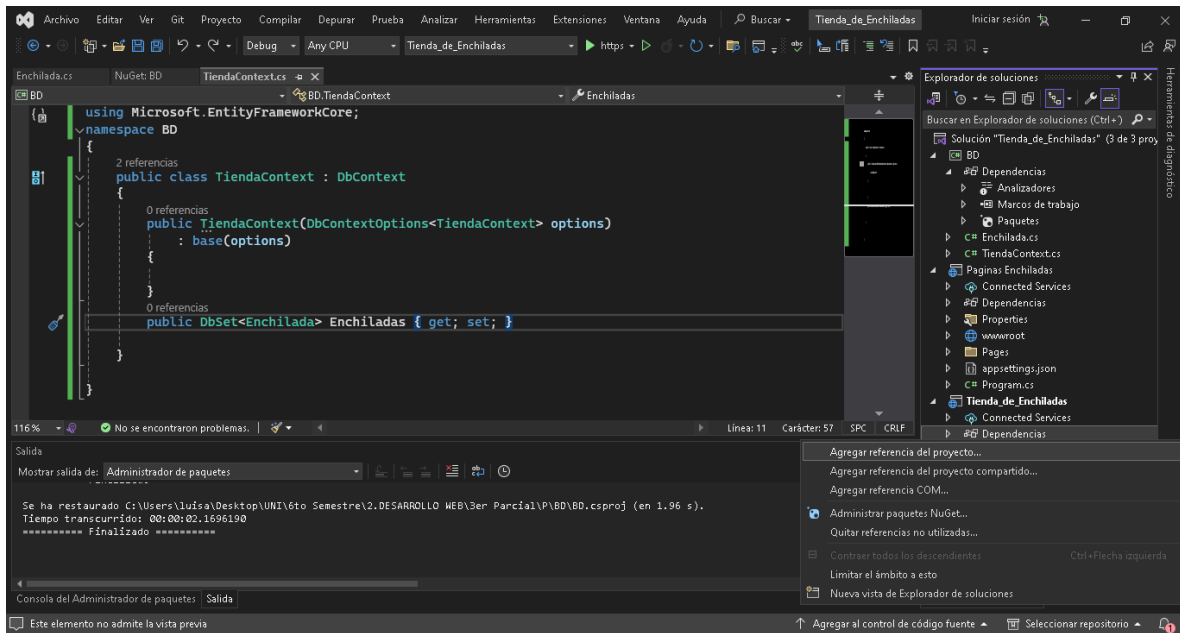
- Cuando creamos la clase, en vez de internal, la volvemos publica, para que la podamos llamar desde otra clase y después creamos los atributos de la tabla enchiladas que con la instruccion key y DatabaseGenerated damos entender que la primera sentencia de la id es un id intercrementable



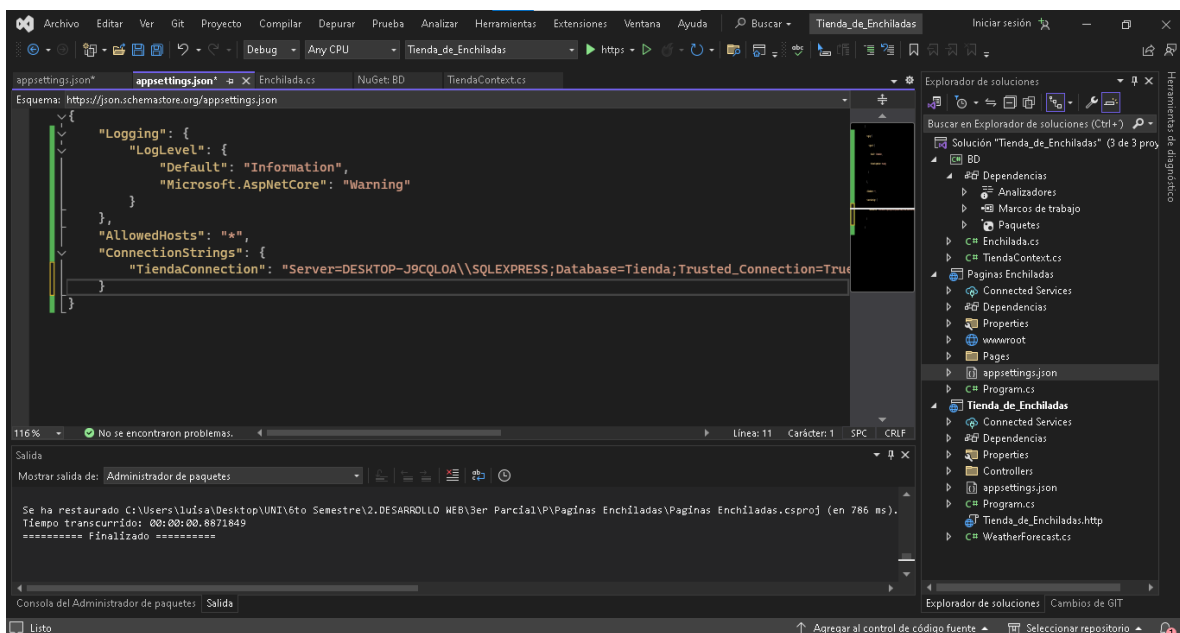
- Con la instrucción public DbSet<Enchilada> Enchiladas {get; set;} enlazamos y conectamos la clase Enchilada a nuestro nucleo de la base de datos que en este caso sería el contexto de nuestra base de datos llamada “Tienda Context” donde “dbContextOptions” le indicamos que justamente nuestra clase es el contexto



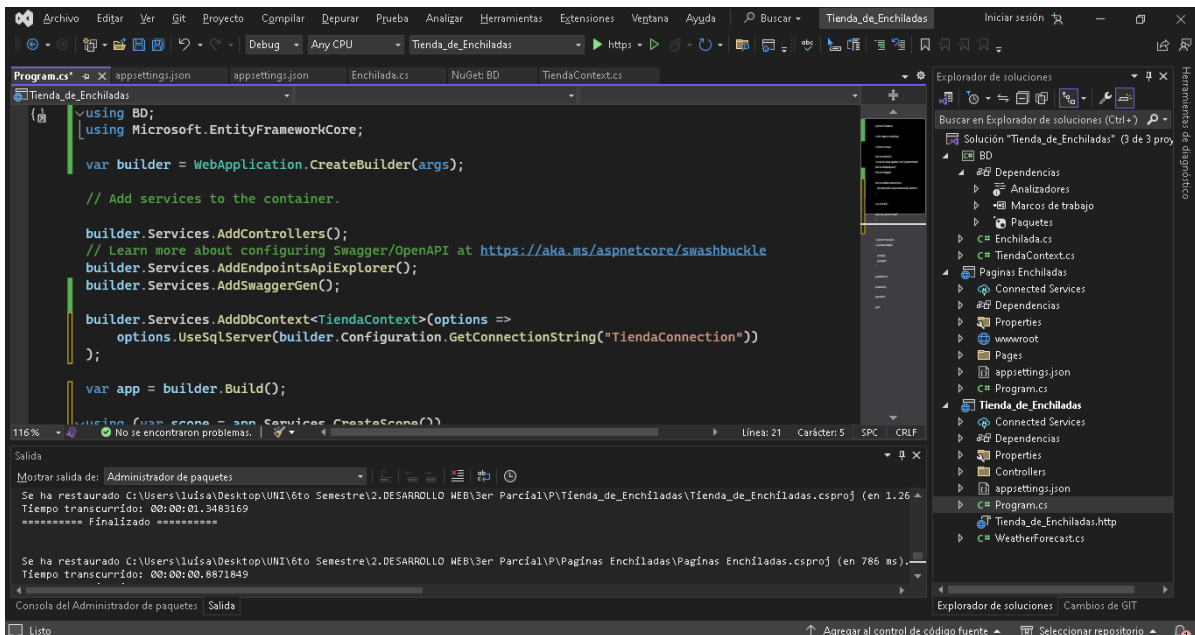
- Con la instrucción “dbset<Enchilada> Enchilada” enlazamos la clase enchilada que sería como nuestra tabla para cuando hagamos la migración puede tomar la información de esa clase también



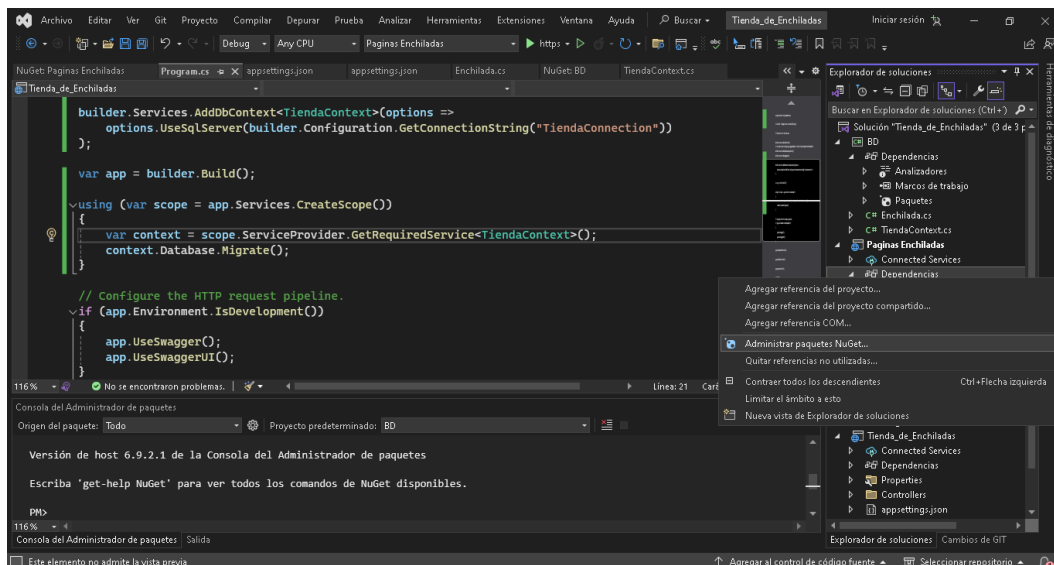
- Después pasamos a la clase “appsettings.json” que es donde pondremos nuestra conexión nuestra base de datos que gracias a “ConnectionString” podemos poner una cadena que servirá para poner el enlace de la base de datos y en la cual llamaremos a nuestro enlace de conexión “TiendaConnection”



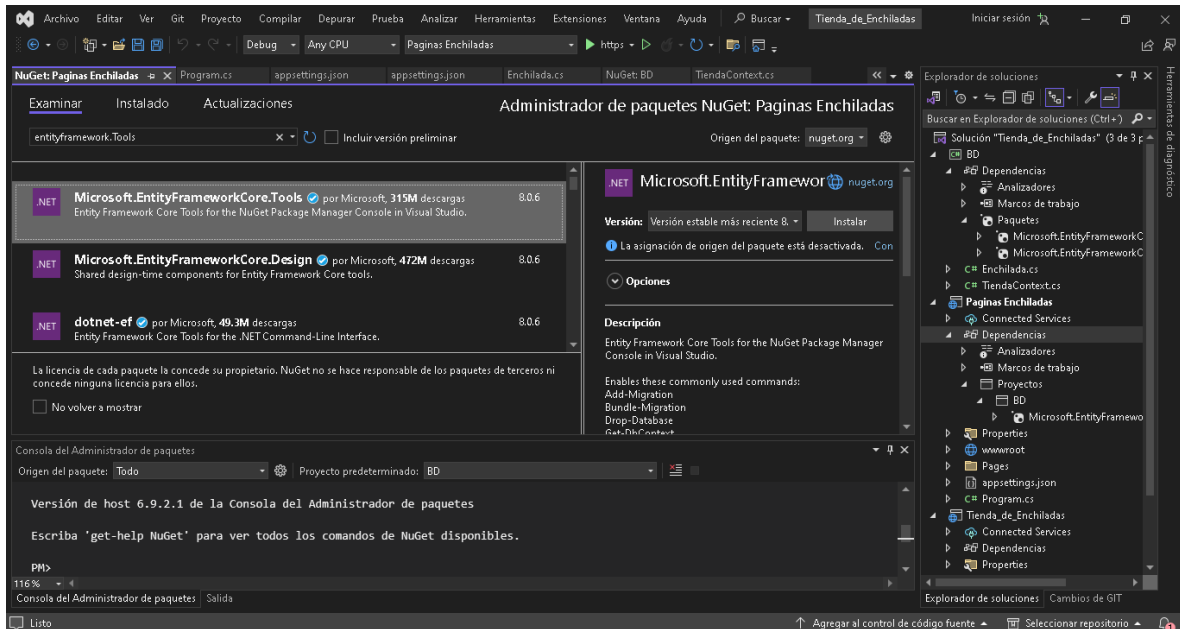
- Ahora vamos a la clase “program.cs” donde ahora haremos una inyección de dependencia que seria “builder.Services.AddDbContext<TiendaContext>(options=>options.UseSqlServer(builder.Configuration.GetConnectionString("TiendaConnection")));” esta dependencia significa que le quitara la acción de crear objetos si no que solo recibira tales objetos, y como podemos ver mandamos a llamar nuestro enlace a la base de datos y con esto hacemos que ejecute nuestra base de datos. Donde es muy importante donde poner la inyección de dependencia que es antes de la instrucción “var app = builder.Build();”



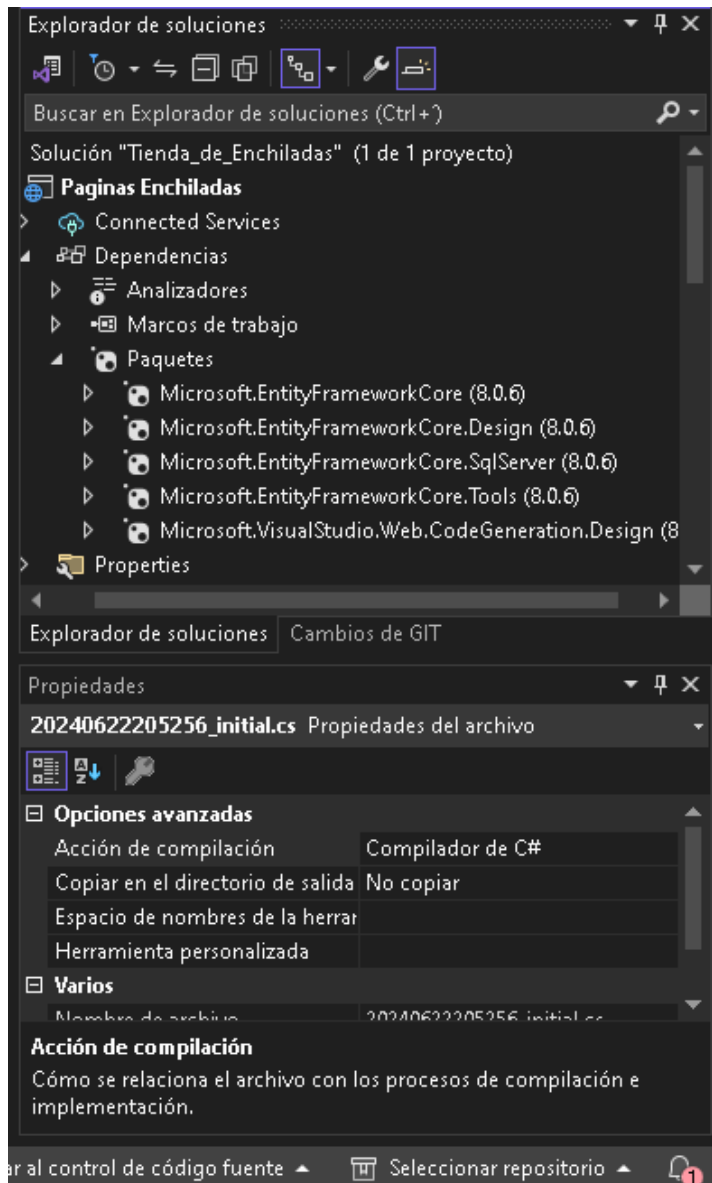
- Ahora en el apartado del explorador de soluciones buscamos nuestro proyecto de soluciones que vendria siendo “Paginas_Enchiladas” y dentro del proyecto buscamos el apartado de dependencias y seleccionamos las dependencias y hacemos clic derecho y damos clic en: “Administrador de paquetes Nuguet”



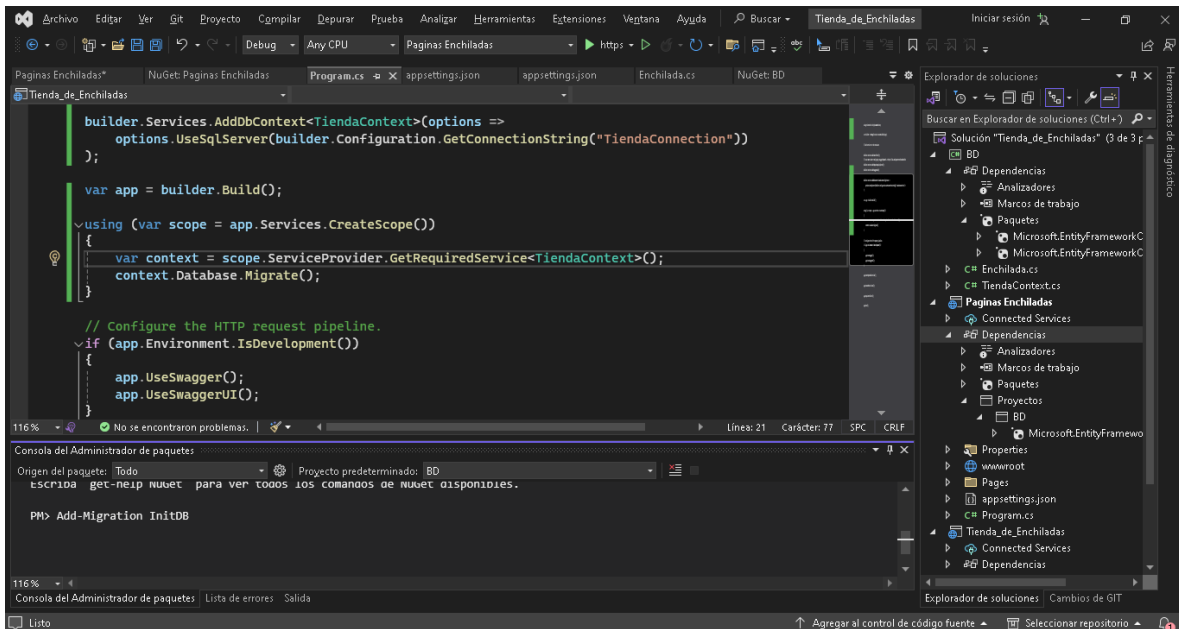
- En el administrador de paquetes nuget buscamos los paquetes de Microsoft.EntityFrameworkCore.Tools, Microsoft.EntityFrameworkCore.SqlServer, Microsoft.EntityFrameworkCore.Design, Microsoft.EntityFrameworkCore y los instalamos en las dependencias del proyecto de solución



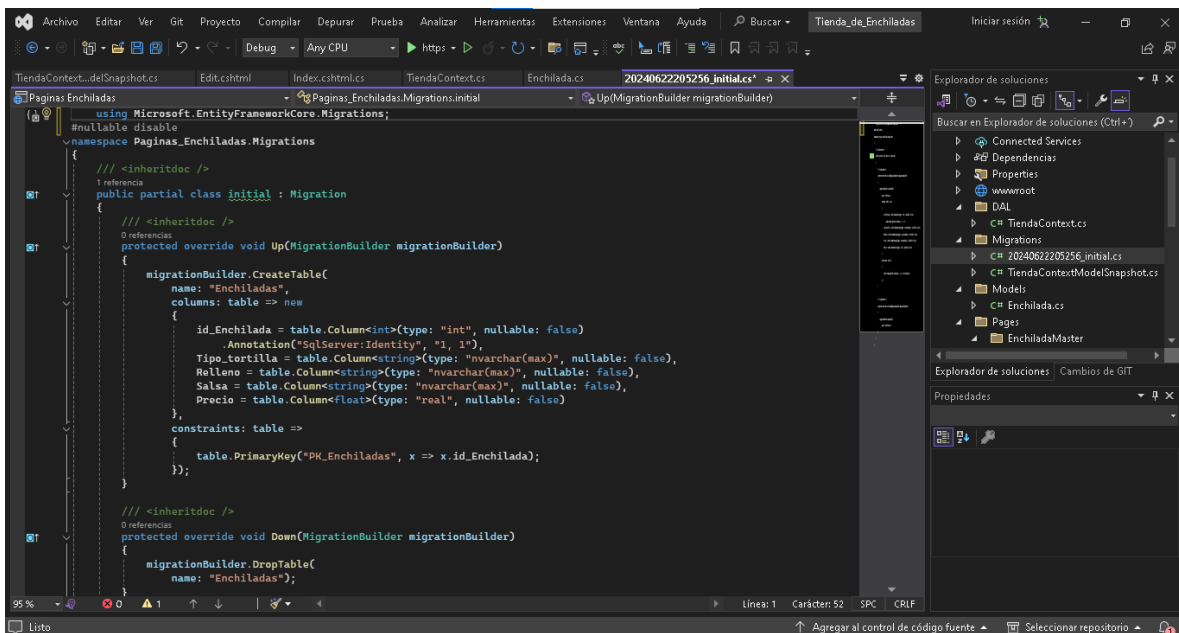
- Para checar que se allá instalado los paquetes nugets, entramos al explorador de soluciones del proyecto y en el apartado de las dependencias hacemos clic hasta desplegarlo hasta los paquetes donde veremos que todos estén instalados. El rol que tienen estos paquetes es que nos van a servir en toda la administración de la base de datos, clases y métodos al igual que las páginas que vamos a utilizar



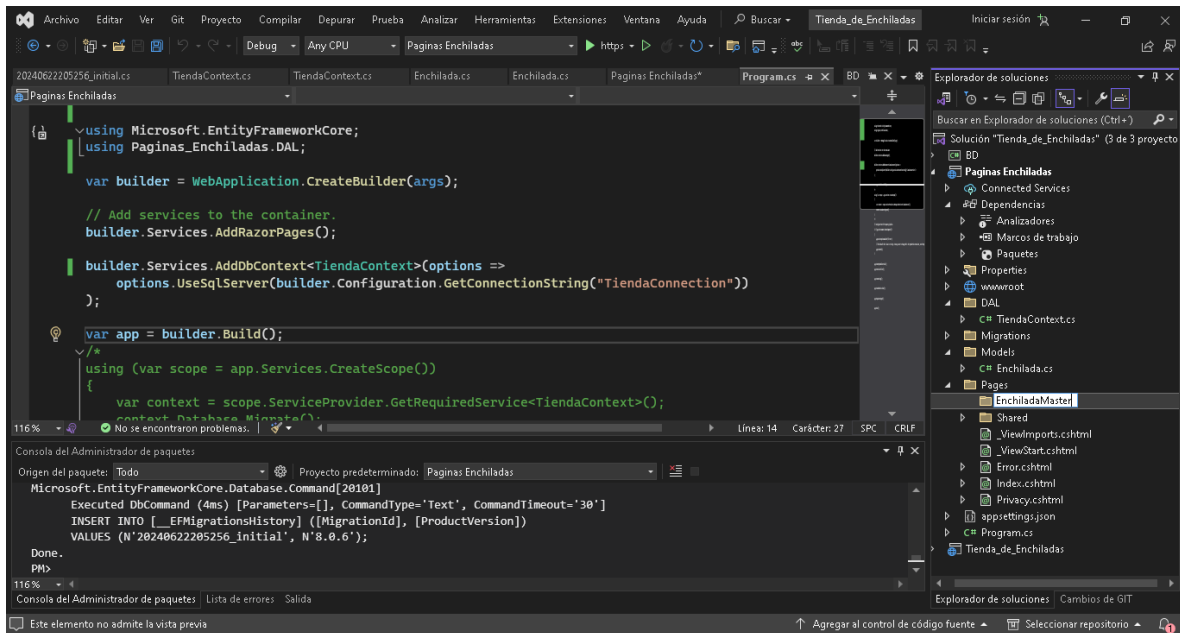
- Lo siguiente es abrir el administrador de paquetes nutget que es la terminal donde haremos la migración de la base de datos con el comando “add-Migration initial” esto genera una carpeta adicional donde se llamara “Migration”



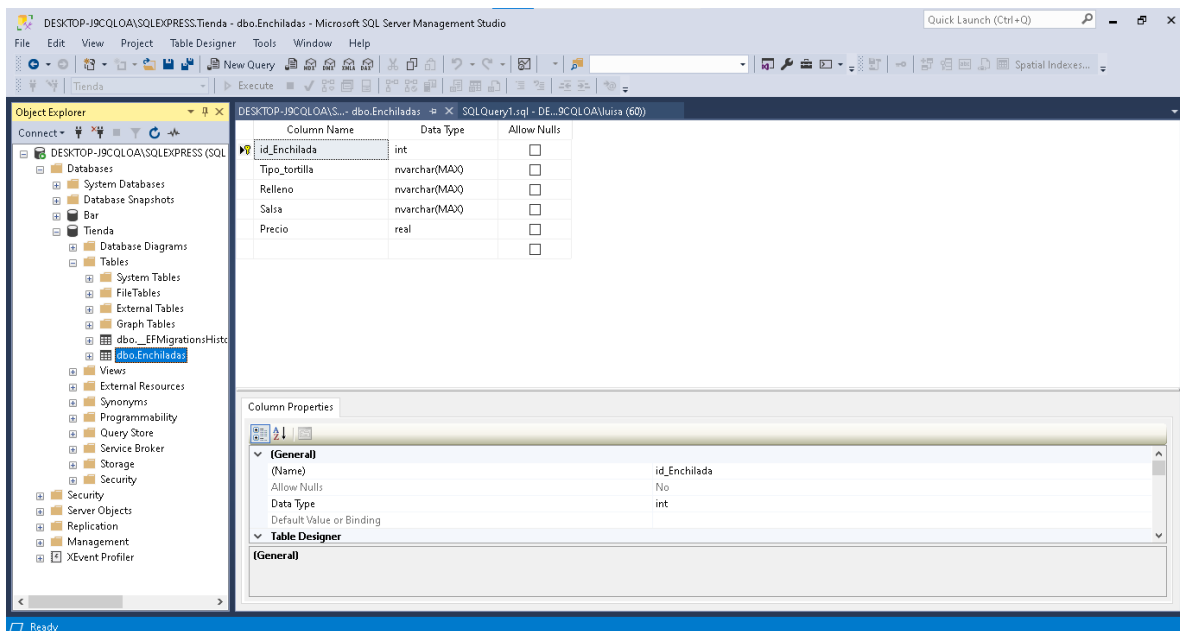
Dentro de la carpeta “Migration” se encontrarán 2 clases generados por el comando donde la clase que dice initial.cs se encuentra la estructura de la tabla donde genero columnas, llaves primarias y campos de la clase enchilada, la migración significa que estamos migrando toda la información de la clase enchilada a la base de datos



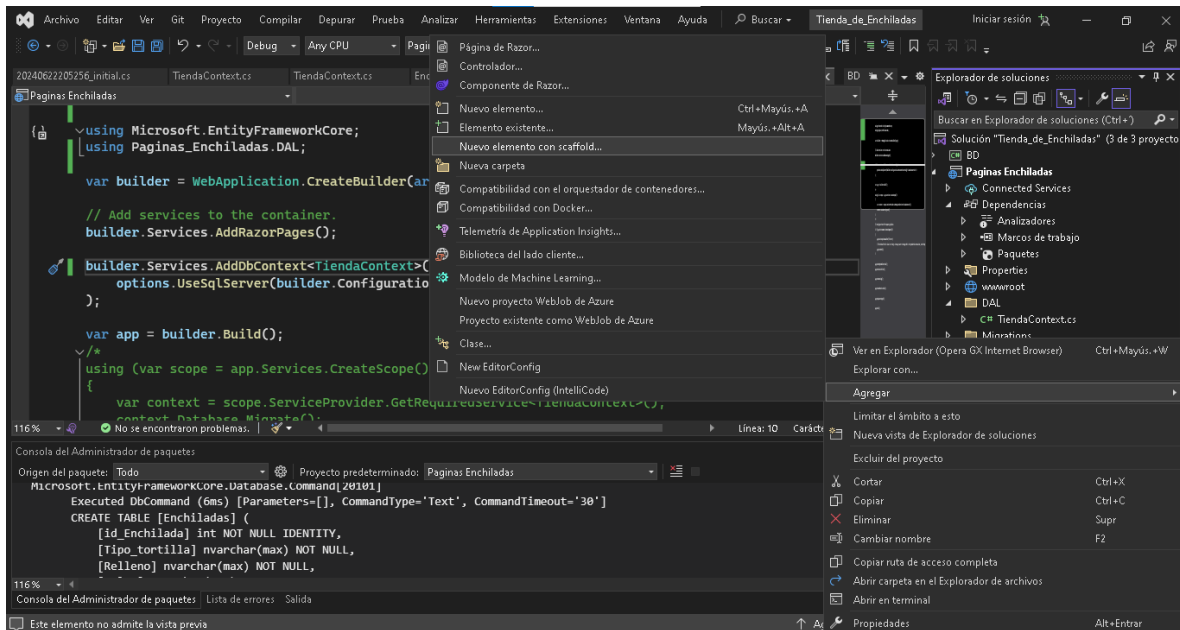
- Ahora en la misma terminal de los paquetes nuget escribimos el comando “update-database” esto genera las columnas de la tabla en la base de datos y por consiguiente la tabla tambien



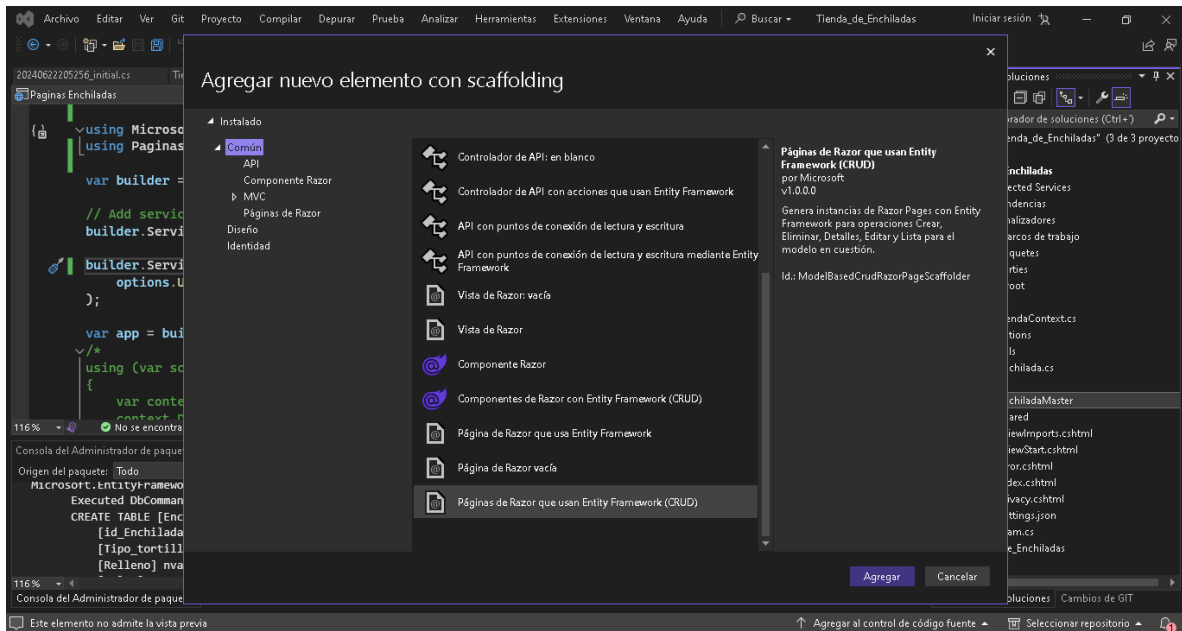
- Vamos a nuestra aplicacion de Microsoft SQL sever manassgement donde comprobaremos que se haya creado correctamente las columnas y la tabla de enchilada para asegurarnos damos clic derecho en la base de datos llamada tienda y damos clic otra vez en la opción de refrech



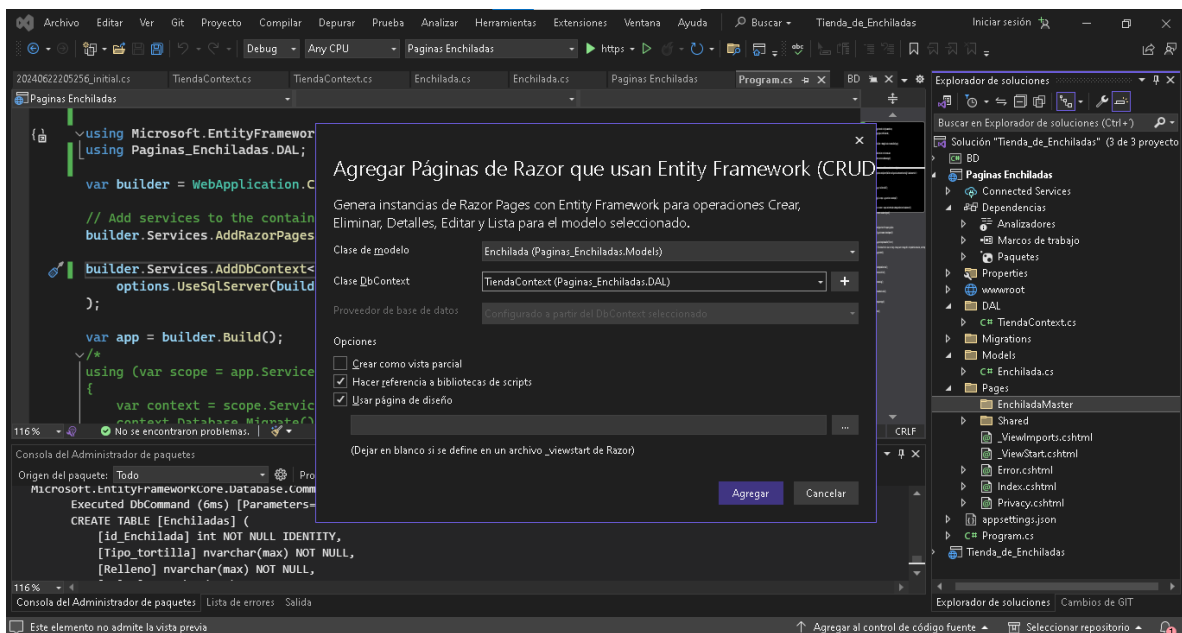
- Ya que terminamos con el contexto de la base de datos, ahora vamos con la creación de las paginas que vamos utilizar para dar altas, bajas, cambios y consulta en este caso vamos al explorado de soluciones y creamos una carpeta que se va a llamar “enchiladasMaster” donde vamos a crear todas las paginas que vamos a utilizar
- Después damos clic derecho en la carpeta creada que fue “Enchilada Master” damos clic en agregar y seleccionamos “Nuevo Elemento con Scaffold” para crear las páginas que vamos a requerir de alta, baja, cambio y consulta



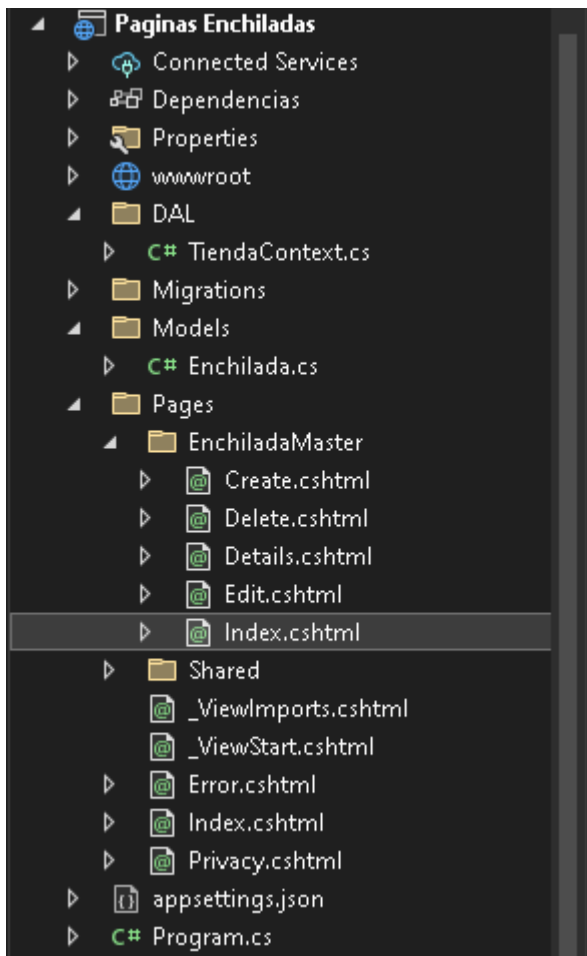
- Al dar clic nos mandará una ventana donde nos dará opciones de que elementos queremos agregar, nosotros daremos clic en el elemento “Paginas de razor que usan Entity Framework (CRUD)” y damos clic en agregar)



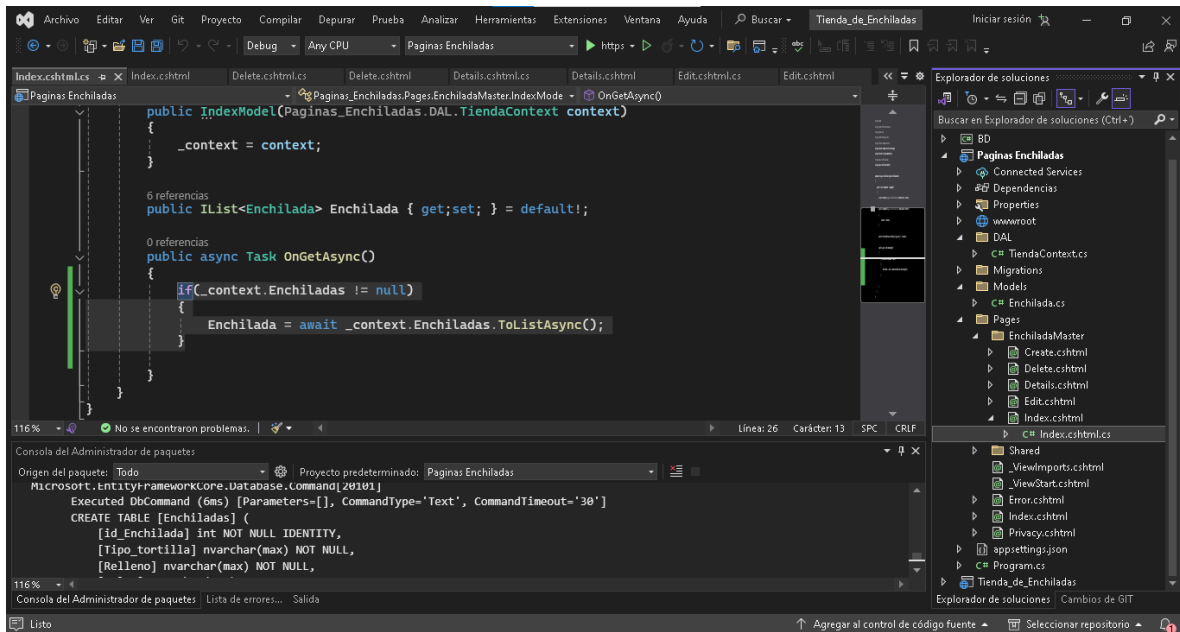
- Después se nos presentara otra pantalla donde se muestran los campos “Clase modelo” y “Clase DbContext” donde en el campo de clase modelo seleccionaremos la clase “Enchilada” y Nuestra clase DbContext será “TiendaContext” lo demas lo dejaremos en default y damos clic en agregar



- Al darle al botón “agregar” en la carpeta de “EnchiladasMaster” se crearon 5 páginas y 5 clases de tipo .cshtml donde haremos el diseño y configuración de las altas, bajas, cambios y consulta

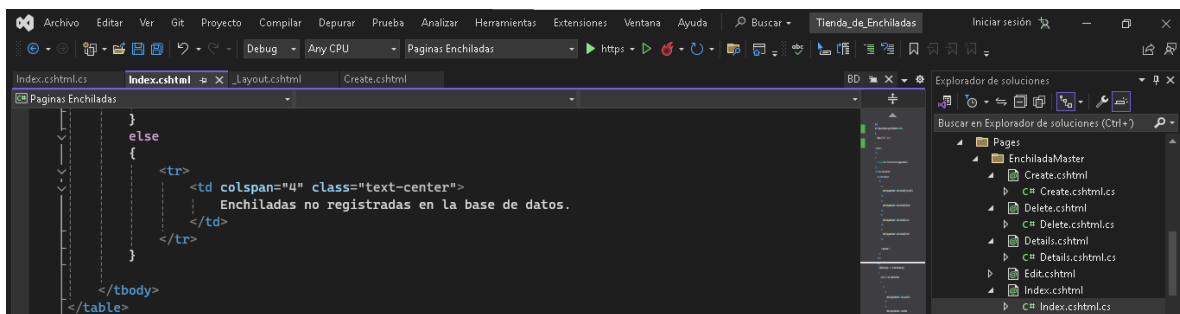


- En la primera página que es el index dentro de lo que es atrás de la pagina encontraremos la clase “.CS” de la página index donde estan todas las actividades que hace toda la base de datos, lo que hizimos modificación del código donde esta marcado color parte azul, en la parte del if, llamaremos el metodo OneGetAsync donde dentro de la condición, obtendremos la lista de enchiladas y conectaremos el contexto de nuestra base de datos y leeremos los datos de la tabla enchilada para que la presente en la parte del frontend



Index (Tabla en caso de no tener datos)

- En la condición, se especifica que si hay registros dentro de la base de datos se mostrara los datos de las enchiladas, pero si no hay datos en la tabla se mostrara el siguiente mensaje diciendo que no hay enchiladas registradas



- En la parte del layout que se encuentra en la carpeta de “shared” forma del indice del menu en el cual cambiamos y agregamos, una imagen que vendria siendo el logo de la tienda en la parte superior donde esta el encabezado que este lo pusimos en Formato .ico para que este sea el icono de nuestra pagina de inicio para despues poner en la parte del body cambiamos el fondo de un color beige para todas las paginas
- Después en la parte sub encabezado que seria el header agregamos en la etiqueta nav nuestro titulo de la pagina que vendria siendo tienda de enchiladas y eliminamos las incensarías del contenido nava que vendría siendo: home, y private
- Y agregamos por ultimo una imagen a lado del titulo de nuestra pagina que vendría siendo el icono mismo que pusimos en nuestra pagina

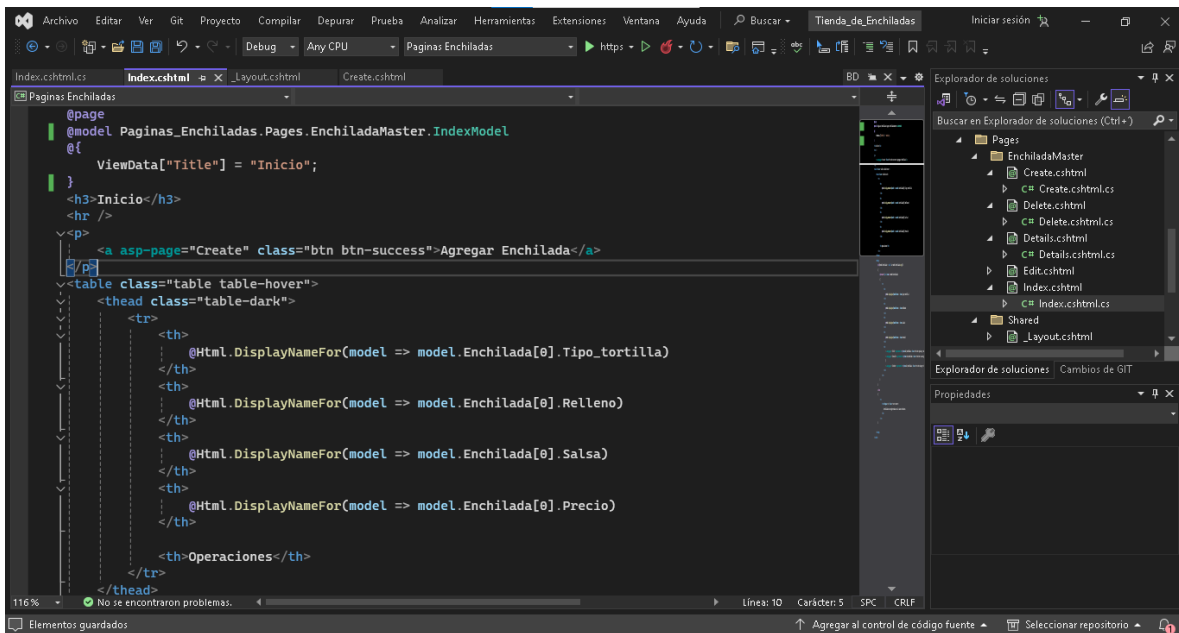
```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - Tienda de Enchiladas</title>
  <link rel="stylesheet" href="~/Lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
  <link rel="icon" href="/Imagen/enchilada.ico" type="image/x-icon">
  <link rel="stylesheet" href="~/Paginas_Enchiladas.styles.css" asp-append-version="true" />
</head>
<body class="bg-opacity-25 bg-warning">
  <header>
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light border-bottom box-shadow">
      <div class="container">
        <a class="navbar-brand" asp-area="" asp-page="/Index">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
          <ul class="navbar-nav flex-grow-1">
          </ul>
        </div>
      </div>
    </nav>
  </header>
  <div class="container">
    <main role="main" class="pb-3">

```

Index(FrontEnd- Boton Agregar- Tabla Encabezado)

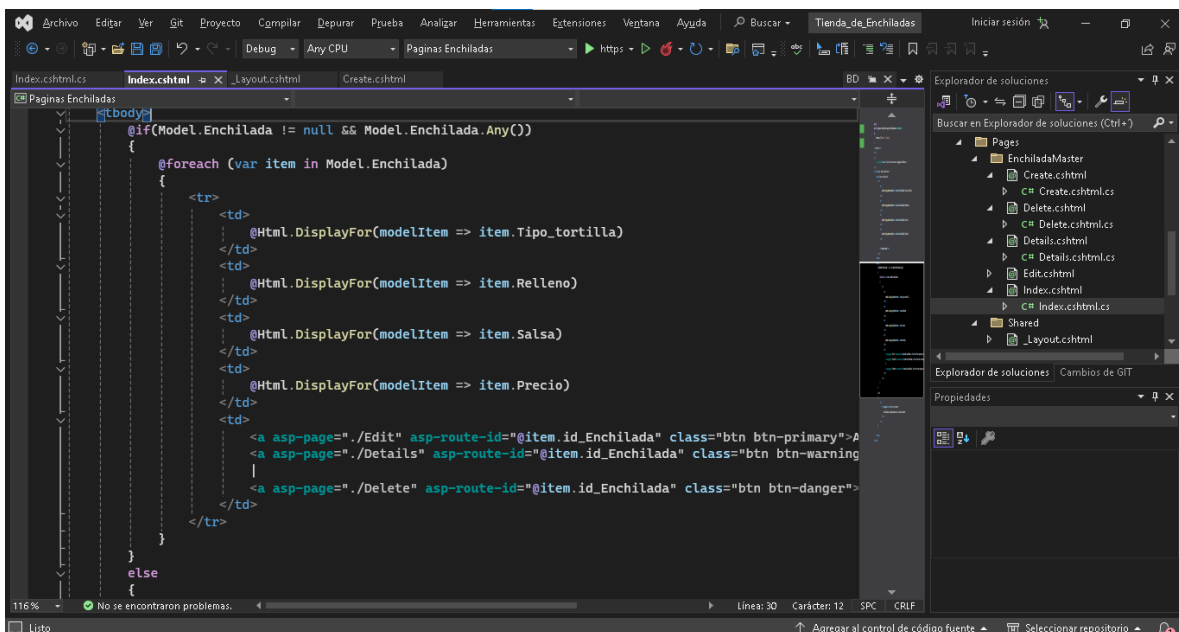
- En la parte de la pagina index modificamos y agregamos los botones que serian de agregar, modificar, consultar y eliminar para una mejor estética para la tabla y el botón de agregar que sería registrar una nueva enchilada la pusimos abajo de la tabla
- Por ultimo la tabla le agregamos un diseño de bootstrap determinado para una mejor estética en la cual al pasar el mouse en el alguno de los campos tiene un efecto de color al poner el mouse sobre el



```
@page
@model Paginas_Enchiladas.Pages.EnchiladaMaster.IndexModel
@{
    ViewData["Title"] = "Inicio";
}
<h3>Inicio</h3>
<hr />
<p>
    <a asp-page="Create" class="btn btn-success">Agregar Enchilada</a>
</p>
<table class="table table-hover">
    <thead class="table-dark">
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Enchilada[0].Tipo_tortilla)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Enchilada[0].Relleno)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Enchilada[0].Salsa)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Enchilada[0].Precio)
            </th>
            <th>Operaciones</th>
        </tr>
    </thead>
</table>
```

Index(Tabla Datos y Operaciones)

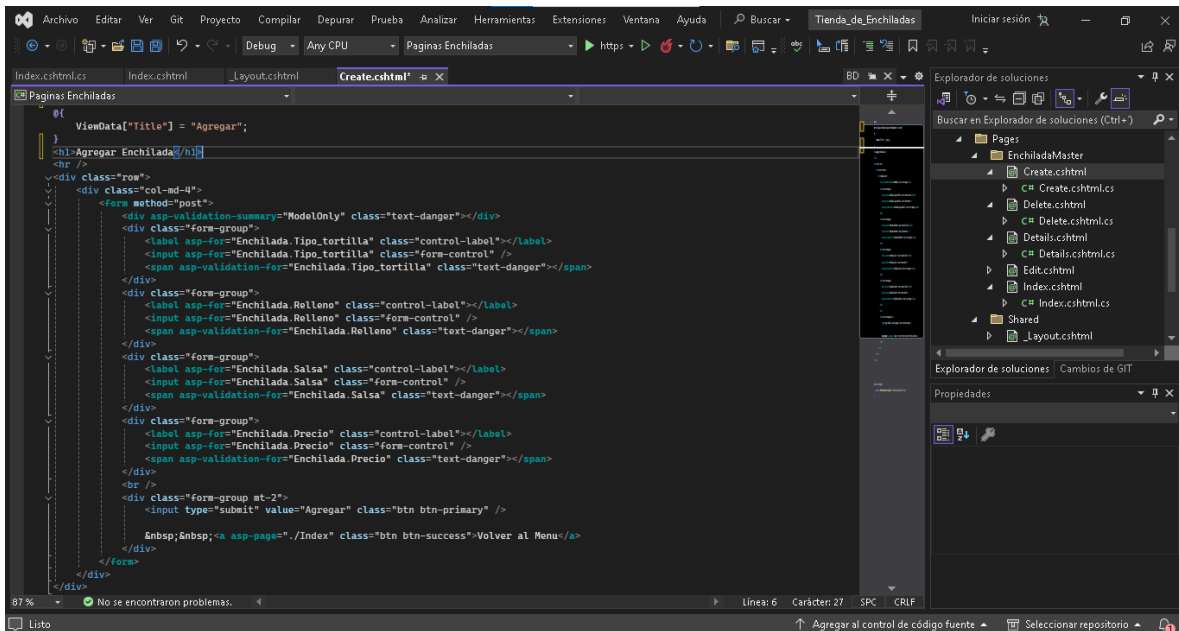
- En este básicamente muestra los datos que tenemos el tabla de enchilada



```
@body
@if(Model.Enchilada != null && Model.Enchilada.Any())
{
    @foreach (var item in Model.Enchilada)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Tipo_tortilla)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Relleno)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Salsa)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Precio)
            </td>
            <td>
                <a asp-page="Edit" asp-route-id="@item.id_Enchilada" class="btn btn-primary">A
                <a asp-page="Details" asp-route-id="@item.id_Enchilada" class="btn btn-warning"
                <a asp-page="Delete" asp-route-id="@item.id_Enchilada" class="btn btn-danger">
            </td>
        </tr>
    }
}
else
{
}
```

Create-Altas (FrontEnd)

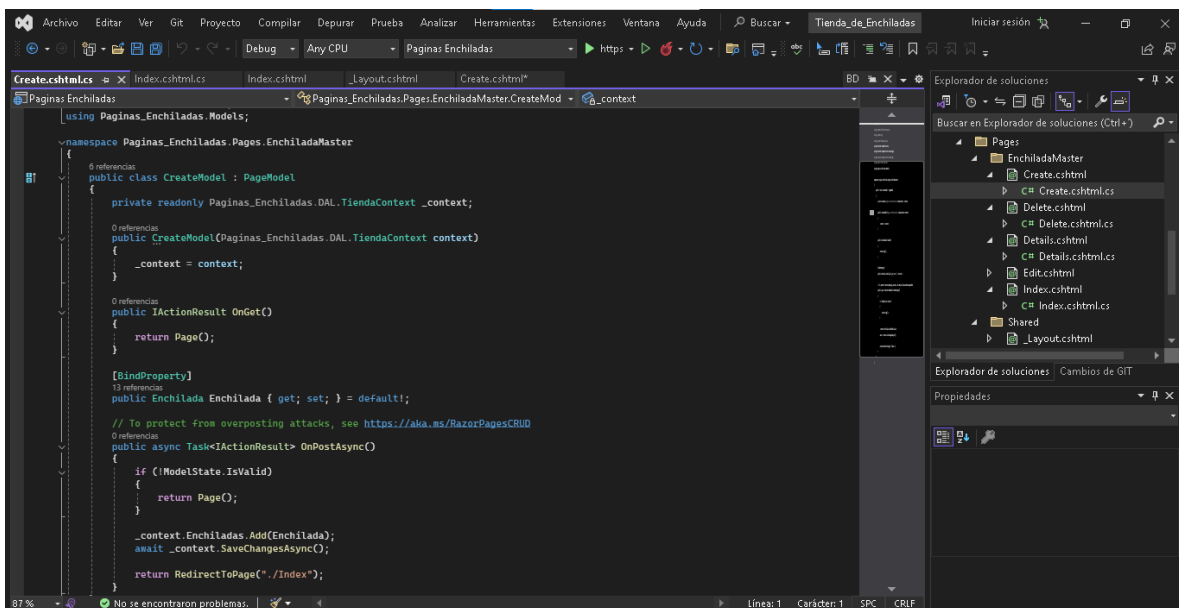
- En esta pagina podemos crear y registra una nueva enchilada en la cual pide todos de los datos de los campos que se necesitan obligatoriamente para llenar requeridos y este viene con dos botones que es una para registrar y volver al menú o la lista de los datos de la tablas enchilada



```
ViewData["Title"] = "Agregar";
<h1>Agregar Enchilada</h1>
<div class="row">
  <div class="col-md-4">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div class="form-group">
      <label asp-for="Enchilada.Tipo_tortilla" class="control-label"></label>
      <input asp-for="Enchilada.Tipo_tortilla" class="form-control" />
      <span asp-validation-for="Enchilada.Tipo_tortilla" class="text-danger"></span>
    </div>
    <div class="form-group">
      <label asp-for="Enchilada.Relleno" class="control-label"></label>
      <input asp-for="Enchilada.Relleno" class="form-control" />
      <span asp-validation-for="Enchilada.Relleno" class="text-danger"></span>
    </div>
    <div class="form-group">
      <label asp-for="Enchilada.Salsa" class="control-label"></label>
      <input asp-for="Enchilada.Salsa" class="form-control" />
      <span asp-validation-for="Enchilada.Salsa" class="text-danger"></span>
    </div>
    <div class="form-group">
      <label asp-for="Enchilada.Precio" class="control-label"></label>
      <input asp-for="Enchilada.Precio" class="form-control" />
      <span asp-validation-for="Enchilada.Precio" class="text-danger"></span>
    </div>
    <br />
    <div class="form-group mt-2">
      <input type="submit" value="Agregar" class="btn btn-primary" />
    </div>
  </div>
  <div class="col-md-4">
    <div class="text-align-right">
      <input type="button" value="Volver al Menu" class="btn btn-success" />
    </div>
  </div>
</div>
```

Create-Altas(BackEnd)

- En la parte de backend de la pagina para crear o registra un la enchilada contiene todo lo necesario para se registre en la base de datos



```
using Paginas_Enchiladas.Models;

namespace Paginas_Enchiladas.Pages.EnchiladaMaster
{
    public class CreateModel : PageModel
    {
        private readonly Paginas_Enchiladas.DAL.TiendaContext _context;

        public CreateModel(Paginas_Enchiladas.DAL.TiendaContext context)
        {
            _context = context;
        }

        public IActionResult OnGet()
        {
            return Page();
        }

        [BindProperty]
        public Enchilada Enchilada { get; set; } = default!;

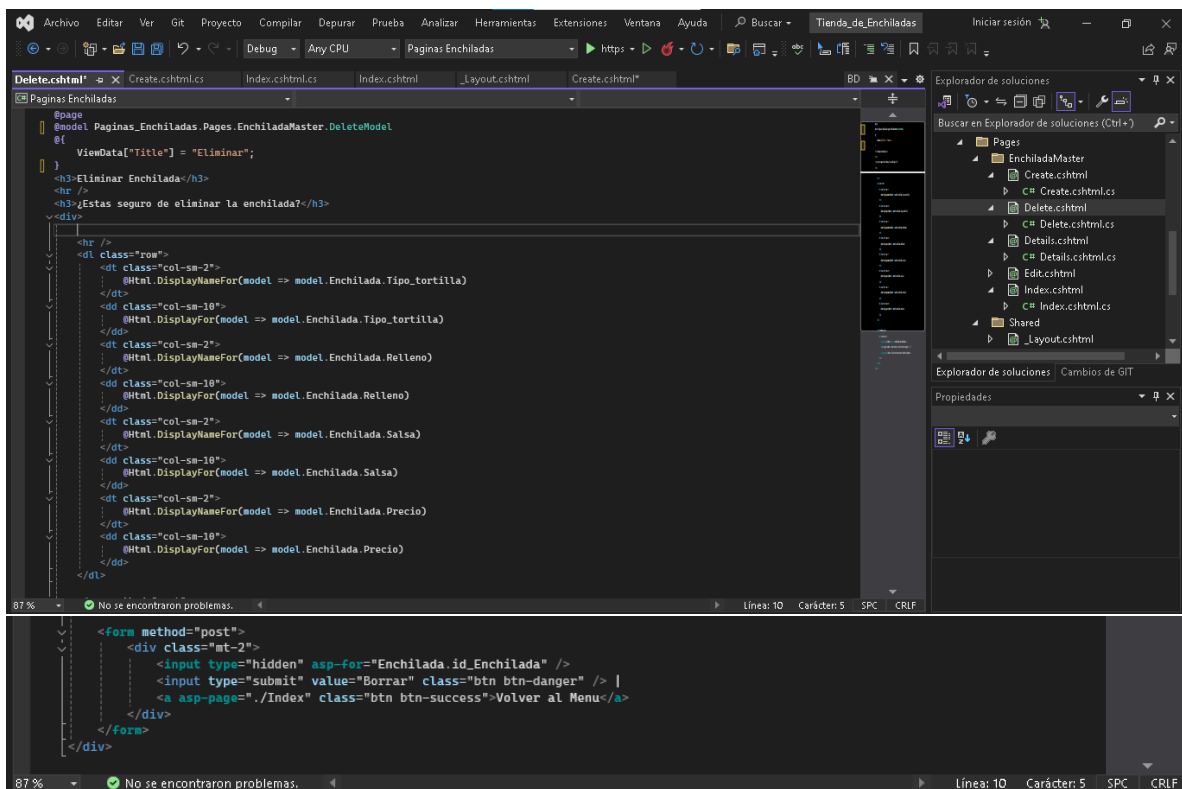
        // To protect from overposting attacks, see https://aka.ms/RazorPagesCRUD
        public async Task<IActionResult> OnPostAsync()
        {
            if (!ModelState.IsValid)
            {
                return Page();
            }

            _context.Enchiladas.Add(Enchilada);
            await _context.SaveChangesAsync();

            return RedirectToPage("./Index");
        }
    }
}
```

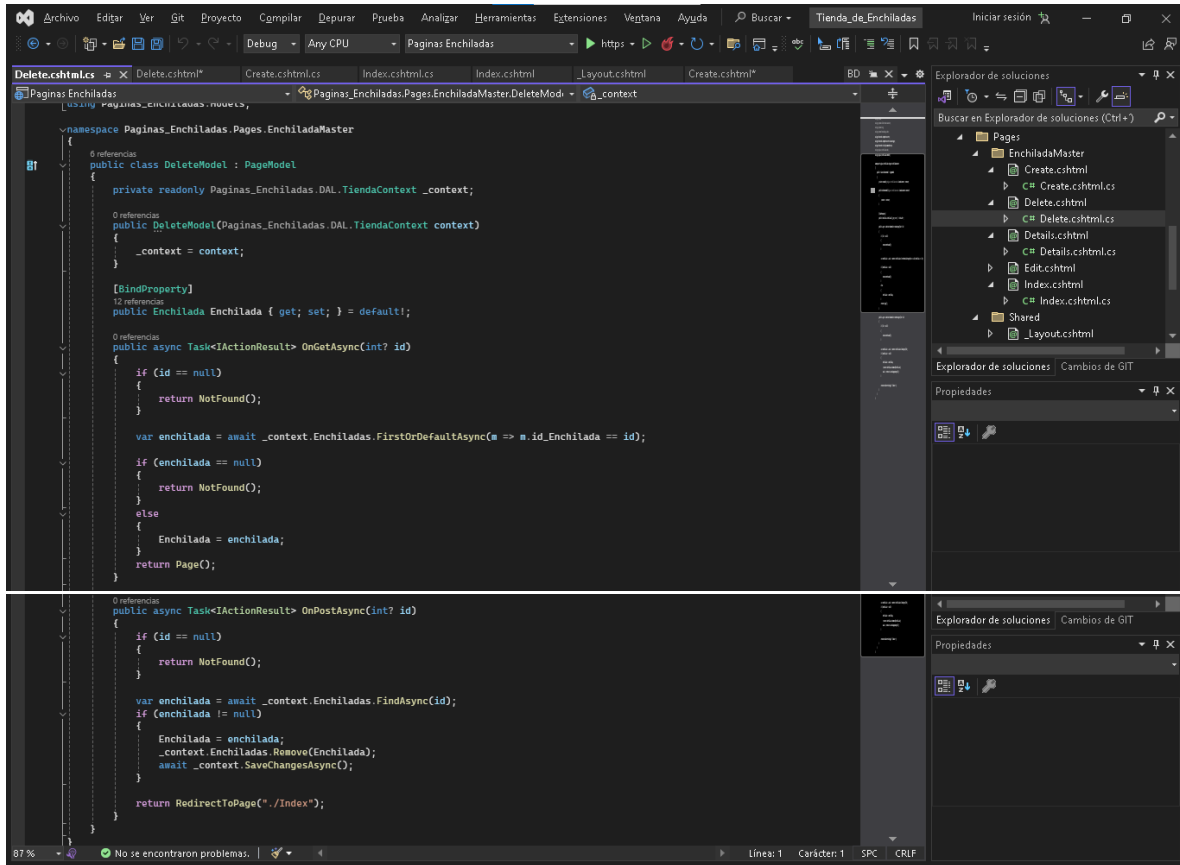
Delete (FrontEnd)

- En la pagina de eliminar un registro de la base de datos básicamente muestra todos los todos del registro de la enchilada que queremos eliminar, para eliminar este registro básicamente en el pagina del index donde viene la tabla de todos los datos registrados anteriormente en el lado inferior derecho viene el botón de eliminar donde muestra específicamente el dato o registro que quieres eliminar



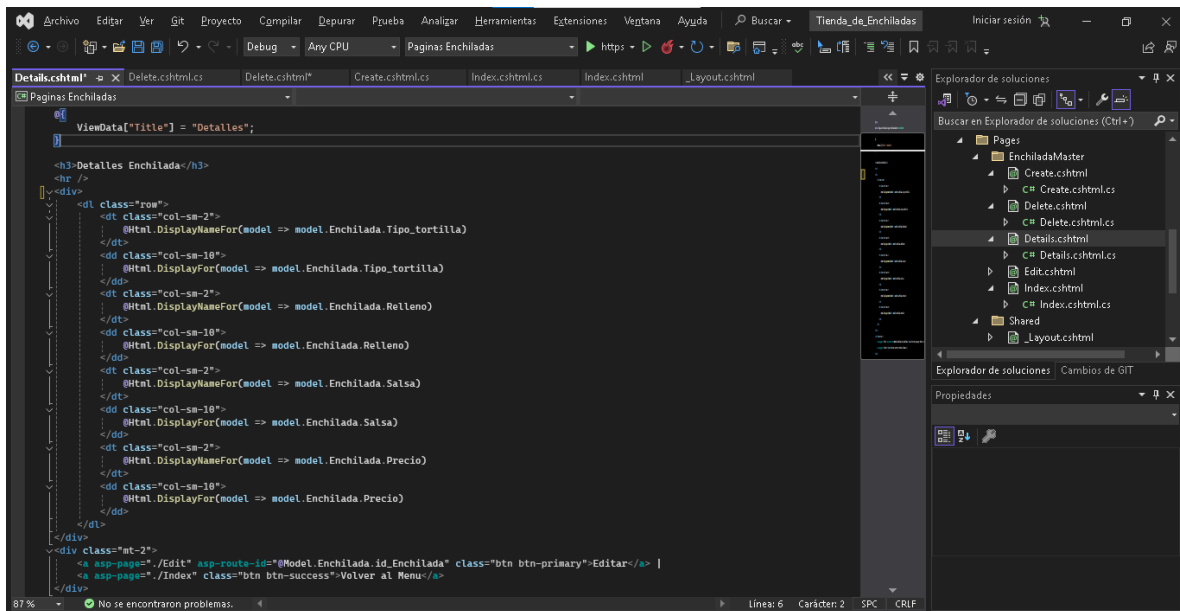
Delete (BackEnd)

- En la parte del backend de la página para eliminar un registro de la tabla enchilada básicamente muestra toda la actividad que se tiene que hacer en la base de datos para eliminar un registro de la tabla



Detalles-Consulta(FrontEnd)

- En la parte de la pagina de consulta de los datos de la tabla enchilada muestra todos los datos registrados de la enchilada ya para consultar dicho datos, al dar clic en del botón de consultar desde la pagina de index o menu redirecciona a este pagina de consulta para mostrar la información mas detallada



The screenshot shows the Visual Studio Code editor with the 'Details.cshtml' file open. The file is part of a project named 'Tienda_de_Enchiladas'. The code in the file is as follows:

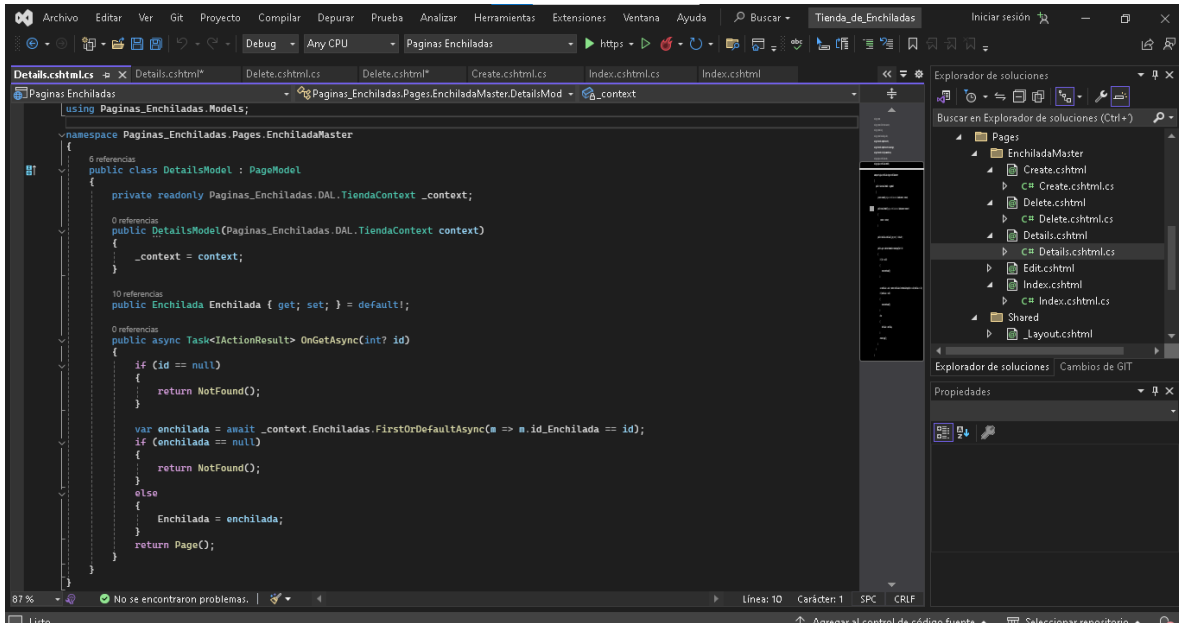
```
ViewData["Title"] = "Detalles";

<h3>Detalles Enchilada</h3>
<hr />
<div>
  <dl class="row">
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.Enchilada.Tipo_tortilla)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.Enchilada.Tipo_tortilla)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.Enchilada.Relleno)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.Enchilada.Relleno)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.Enchilada.Salsa)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.Enchilada.Salsa)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.Enchilada.Precio)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.Enchilada.Precio)
    </dd>
  </dl>
  <div class="mt-2">
    <a asp-page="/Edit" asp-route-id="@Model.Enchilada.id_Enchilada" class="btn btn-primary">Editar</a> |
    <a asp-page="/Index" class="btn btn-success">Volver al Menu</a>
  </div>
</div>
```

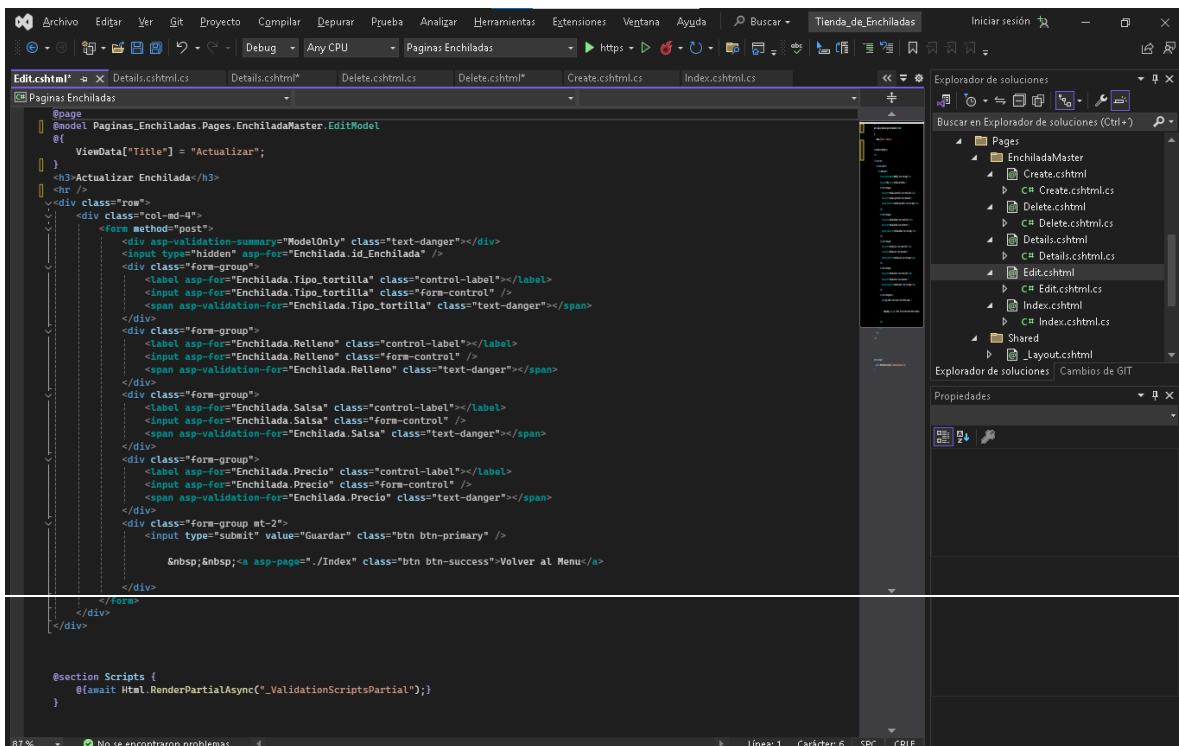
The Explorer pane on the right shows the project structure, including the 'Pages' folder with files like 'EnchiladaMaster', 'Create.cshtml', 'Delete.cshtml', 'Details.cshtml', 'Edit.cshtml', 'Index.cshtml', and 'Shared'.

Detalles-Consulta (BackEnd)

- En la parte del backend de la página para consultar un registro de la tabla enchilada básicamente muestra toda la actividad que se tiene que hacer en la base de datos para consultar un registro de la tabla enchilada

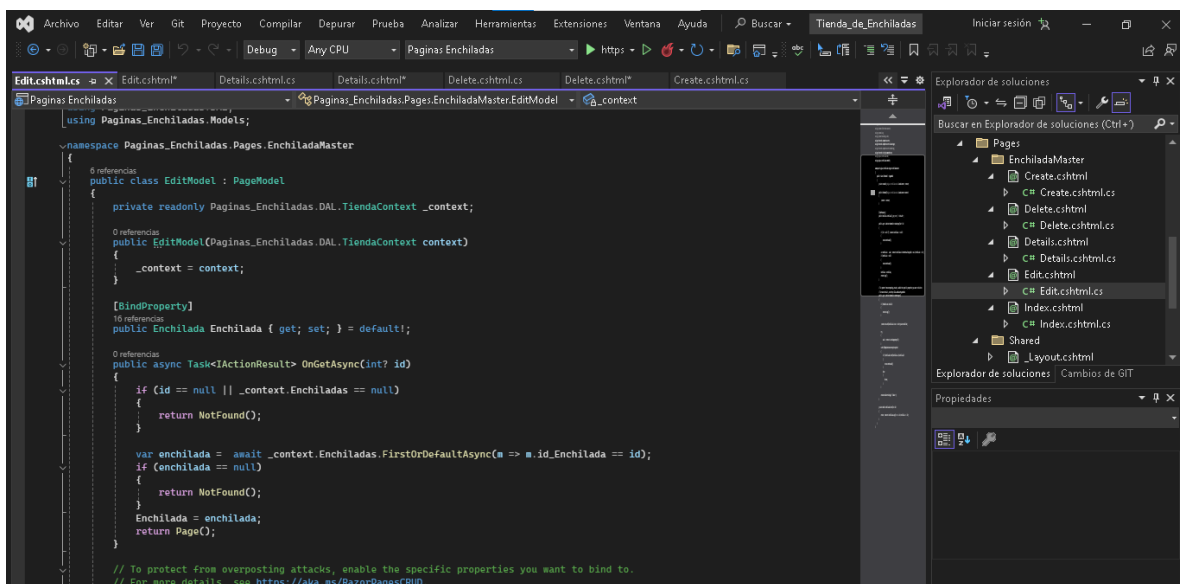


Edit-Actualizar(FronEnd)



Editar-Actualizar (BackEnd)

- Para la pagina actualizar un registro de la tabla de enchiladas, para relizar esta accion se necesita dar al boton de actualizar desde la pagina index para que selecciones el registro que quieres actualizar al momento de dar clic al boton redirecciona a este, la pagina muestra en cuandros de dialogo toda la informacion del dato o registro que ya se habia agregado anteriormente para que este puede ser reemplazado por la nueva información que vamos agregar ya que modificamos los campos que queremos modificar hasta abajo aparacera el boton de de guardar registro y este inmediatamente se regresera al indez o menu para que se muestra que se ha modificado correctamente



```
using Paginas_Enchiladas.Models;

namespace Paginas_Enchiladas.Pages.EnchiladaMaster
{
    [Route("")]
    public class EditModel : PageModel
    {
        private readonly Paginas_Enchiladas.DAL.TiendaContext _context;

        [Route("")]
        public EditModel(Paginas_Enchiladas.DAL.TiendaContext context)
        {
            _context = context;
        }

        [BindProperty]
        public Enchilada Enchilada { get; set; } = default!;

        [Route("")]
        public async Task<IActionResult> OnGetAsync(int? id)
        {
            if (id == null || _context.Enchiladas == null)
            {
                return NotFound();
            }

            var enchilada = await _context.Enchiladas.FirstOrDefaultAsync(m => m.id_Enchilada == id);
            if (enchilada == null)
            {
                return NotFound();
            }
            Enchilada = enchilada;
            return Page();
        }

        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see https://aka.ms/RazorPagesCRUD
    }
}
```

- En la parte del backend de la página para modificar o cambiar un registro de la tabla enchilada básicamente muestra toda la actividad que se tiene que hacer en la base de datos para modificar un registro de la tabla enchilada sin necesidad de que primero que metamos el id del registro

```

public async Task<ActionResult> OnPostAsync()
{
    if (!ModelState.IsValid)
    {
        return Page();
    }

    _context.Attach(Enchilada).State = EntityState.Modified;

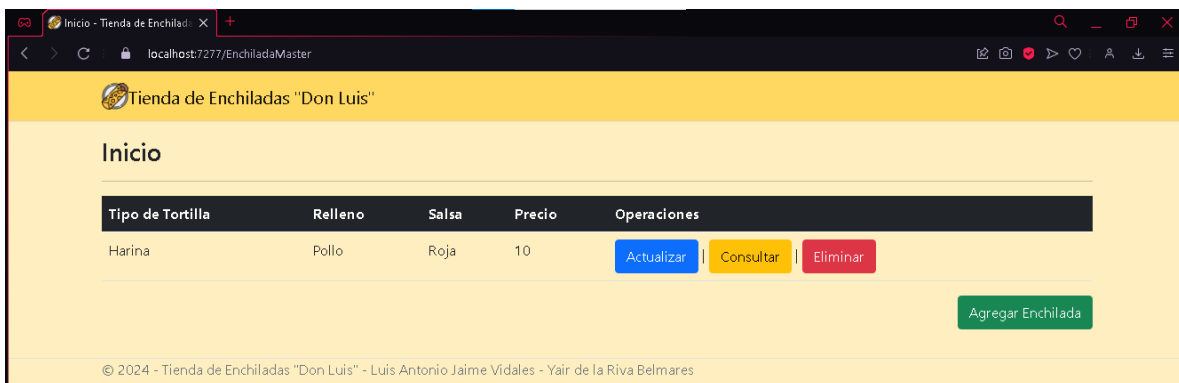
    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!EnchiladaExists(Enchilada.id_Enchilada))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return RedirectToPage("./Index");
}

1 referencia
private bool EnchiladaExists(int id)
{
    return _context.Enchiladas.Any(e => e.id_Enchilada == id);
}

```

- Así finalmente quedaría nuestra pagina y diseño de la pagina tienda de enchiladas



- Vista de diseño de para crear un registro de la tabla de enchiladas

localhost:7277/EnchiladaMaster/Create

Tienda de Enchiladas "Don Luis"

Agregar Enchilada

Tipo de Tortilla

Relleno

Salsa

Precio

[Agregar](#) [Volver al Menu](#)

© 2024 - Tienda de Enchiladas "Don Luis" - Luis Antonio Jaime Vidales - Yair de la Riva Belmares

- Vista de diseño de para modificar o actualizar un registro de la tabla de enchiladas

localhost:7277/EnchiladaMaster/Edit

Tienda de Enchiladas "Don Luis"

Actualizar Enchilada

Tipo de Tortilla

Relleno

Salsa

Precio

[Guardar](#) [Volver al Menu](#)

© 2024 - Tienda de Enchiladas "Don Luis" - Luis Antonio Jaime Vidales - Yair de la Riva Belmares

- Vista de diseño de para consultar un registro de la tabla de enchiladas

localhost:7277/EnchiladaMaster/Details

Tienda de Enchiladas "Don Luis"

Detalles Enchilada

Tipo de Tortilla	Harina
Relleno	Pollo
Salsa	Roja
Precio	10

[Editar](#) [Volver al Menu](#)

© 2024 - Tienda de Enchiladas "Don Luis" - Luis Antonio Jaime Vidales - Yair de la Riva Belmares

- Vista de diseño de para eliminar un registro de la tabla de enchiladas

