

Crear la primera página

Resumen



Introducción al patrón MVC

Creación del modelo y del repositorio

Creación del controlador

Añadir la vista

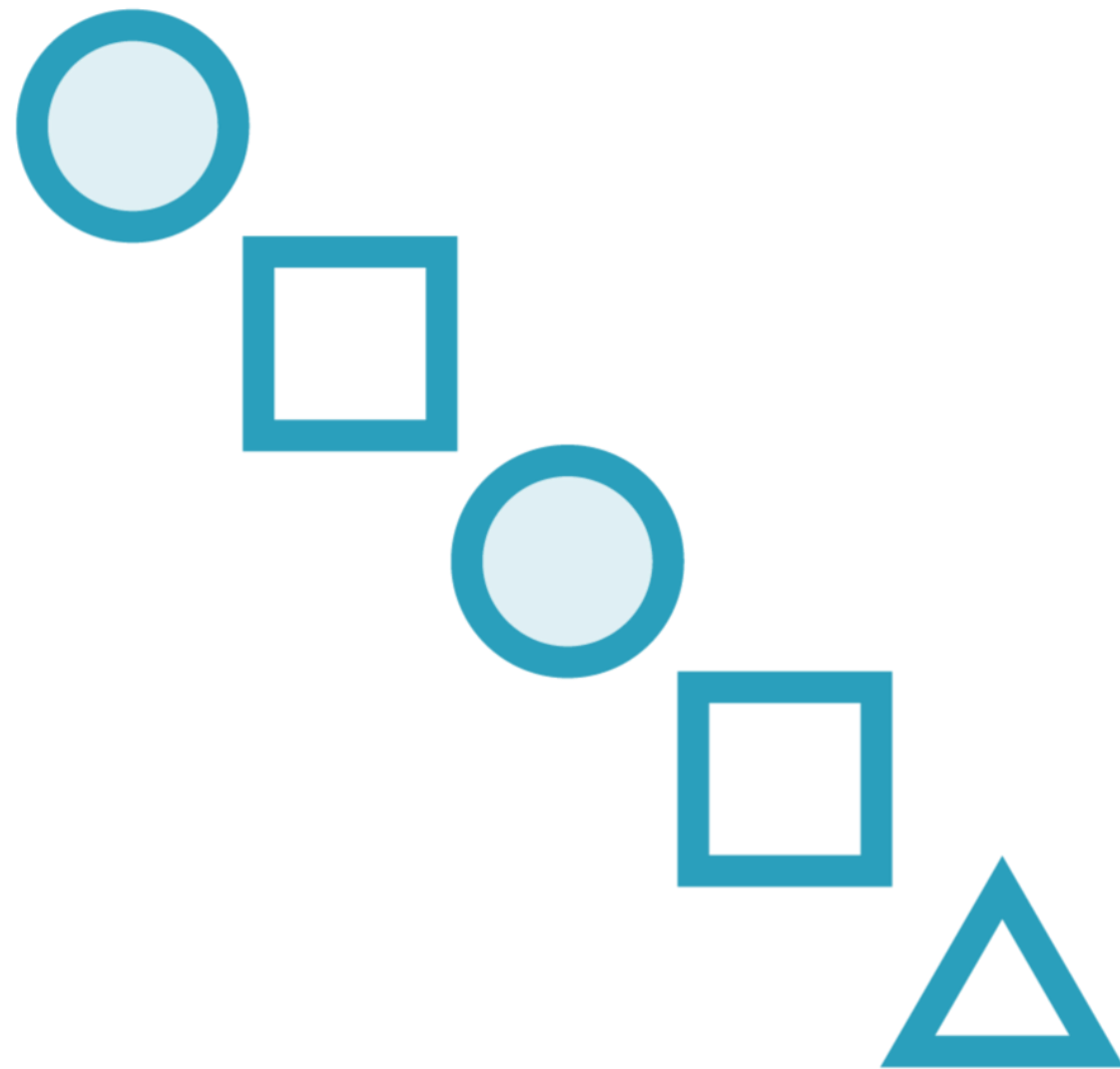
Agregar estilos a la vista

Demo



Viendo la página que vamos a crear

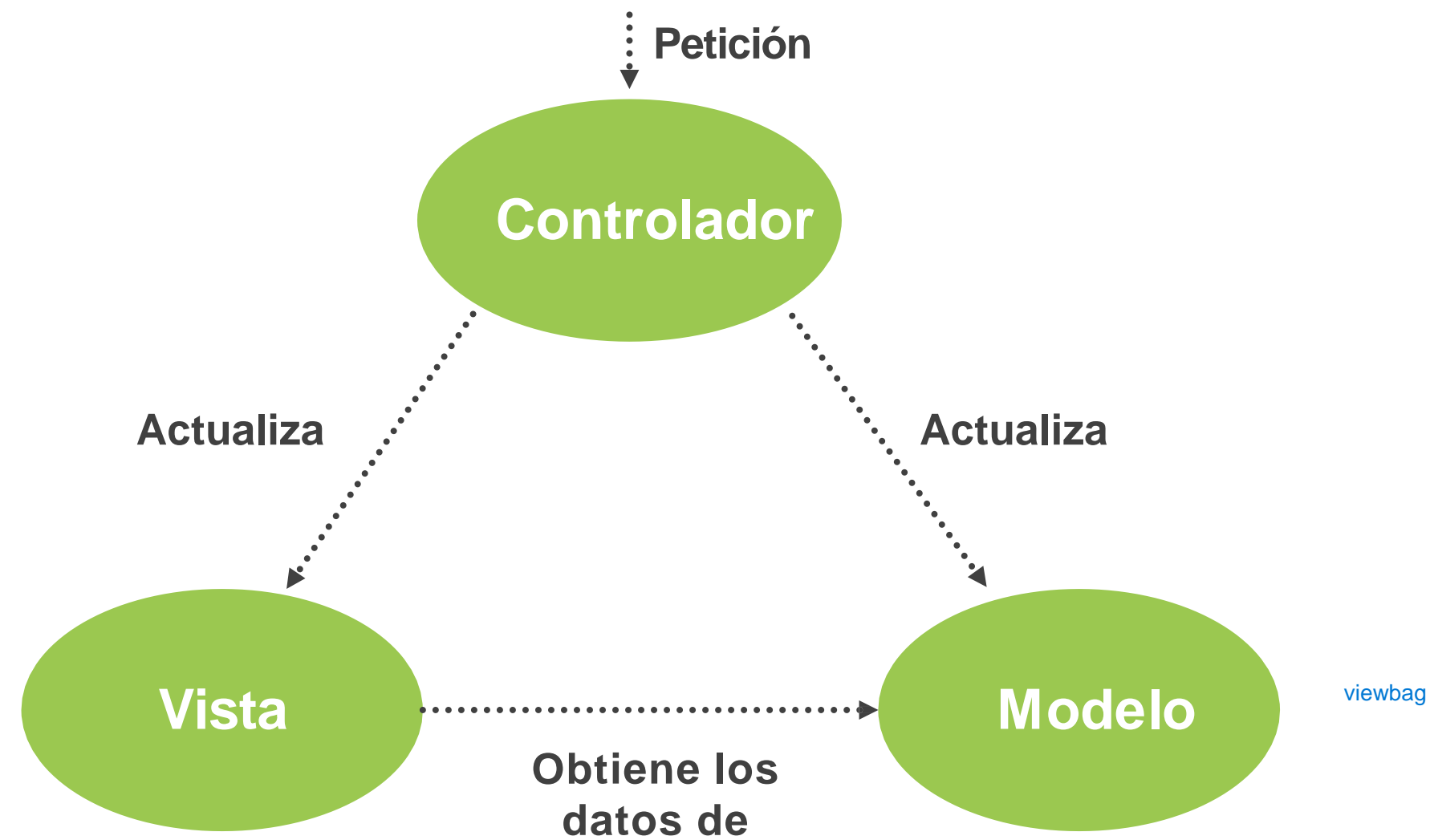
Introducción al patrón MVC



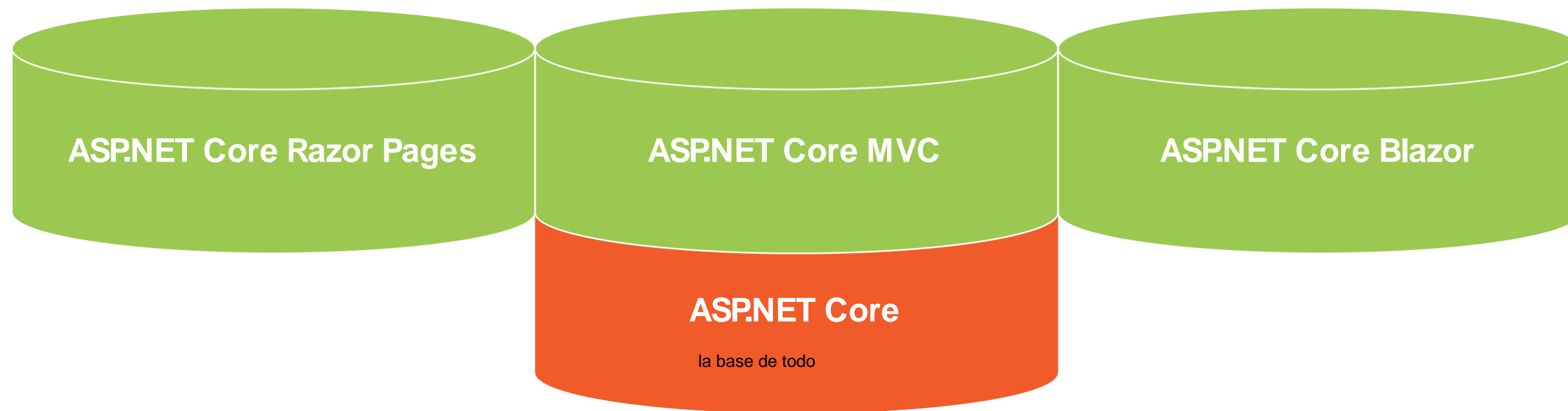
Modelo – Vista – Controlador

- Patrón de diseño muy común
- Separación de responsabilidades
- Menos dependencias
- Fomenta la posibilidad de realizar pruebas y el mantenimiento

El patrón MVC

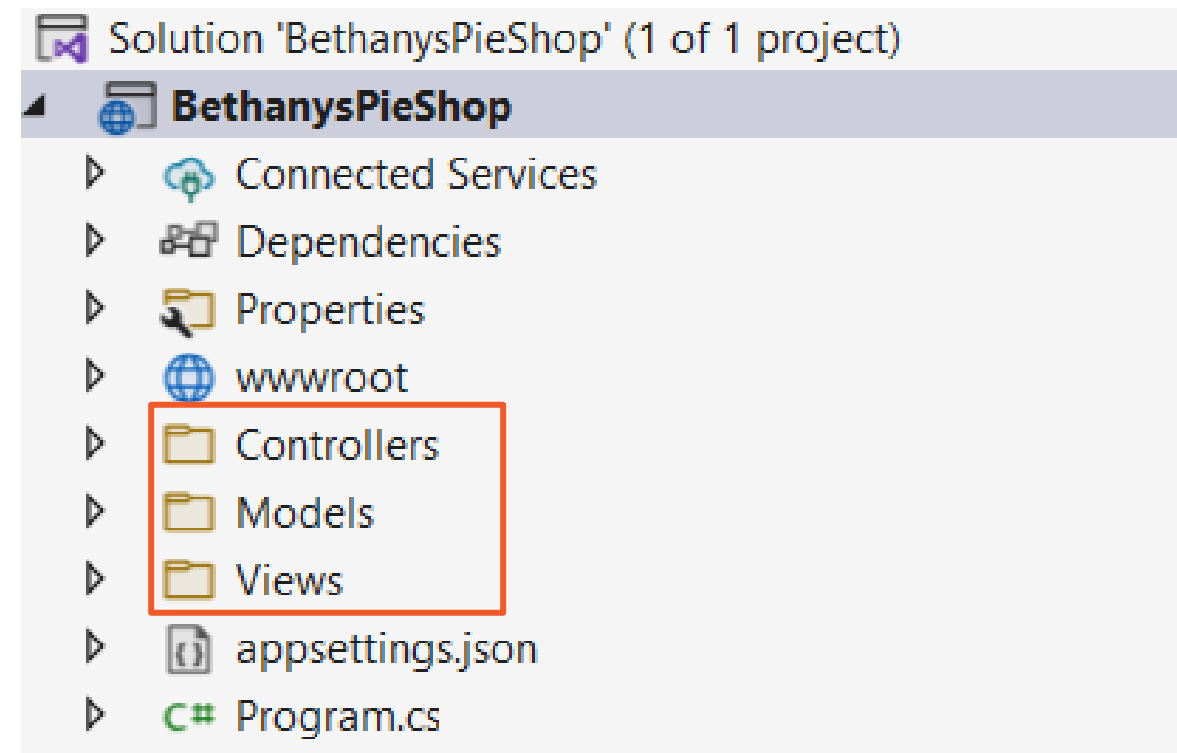


El framework ASP.NET Core MVC

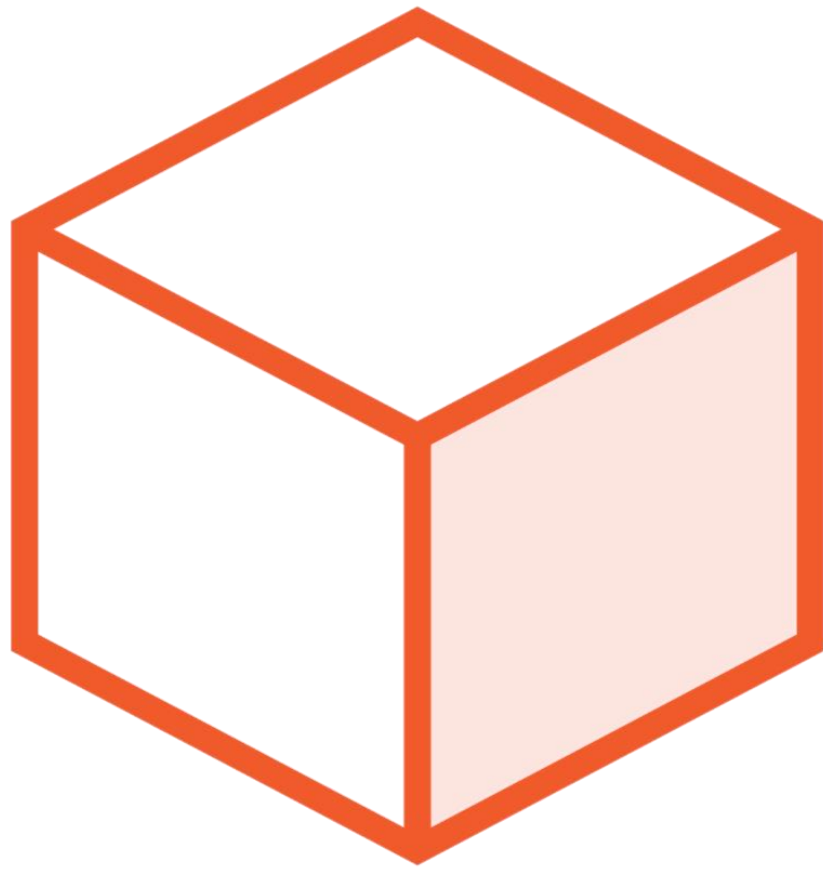


Creación del modelo y del repositorio

Carpetas basadas en convenciones



El Modelo



Datos de dominio y lógica para gestionar los datos

API sencilla

Oculto los detalles de la gestión de los datos

Ejemplo de clase de Modelo

Utilizando Nullable

```
public class Pie
{
    public int PieId { get; set; }
    public string Name { get; set; }
    public string? ShortDescription { get; set; }
    public string? LongDescription { get; set; }
    public string? AllergyInformation { get; set; }
    public decimal Price { get; set; }
    public string? ImageUrl { get; set; }
    public string? ImageThumbnailUrl { get; set; }
    public bool IsPieOfTheWeek { get; set; }
    public bool InStock { get; set; }
    public int CategoryId { get; set; }
    public Category Category { get; set; }
}
```



El repositorio permite a nuestro código utilizar objetos sin saber cómo se persisten.

Crearemos una interfaz y su implementación.



```
public interface IPieRepository
{
    IEnumerable<Pie> AllPies { get; }
    Pie GetPieById(int pieId);
}
```

El interfaz IPieRepository

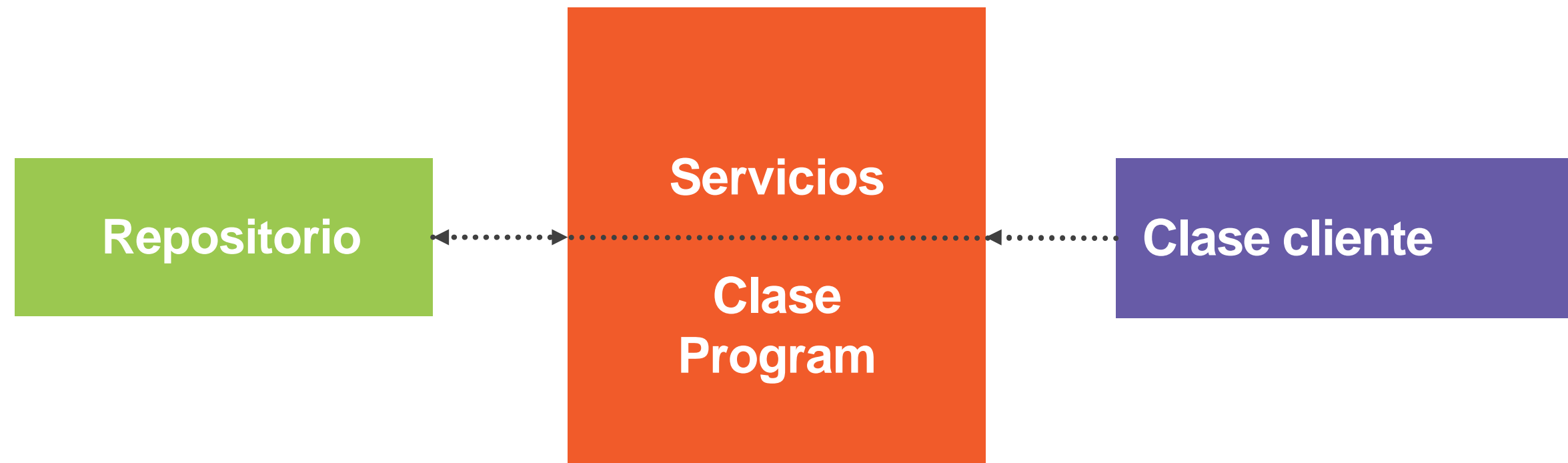
Creando la implementación

```
public class MockPieRepository : IPieRepository
{
    public IEnumerable<Pie> AllPies
    {
        get { ... }
    }

    public Pie GetPieById(int pieId)
    { ... }
}
```



Registrando el repositorio



Registrando el repositorio

Program.cs

```
var builder = WebApplication.CreateBuilder(args);  
  
builder.Services.AddScoped<IPieRepository, MockPieRepository>();  
var app = builder.Build();
```


Registro de servicios

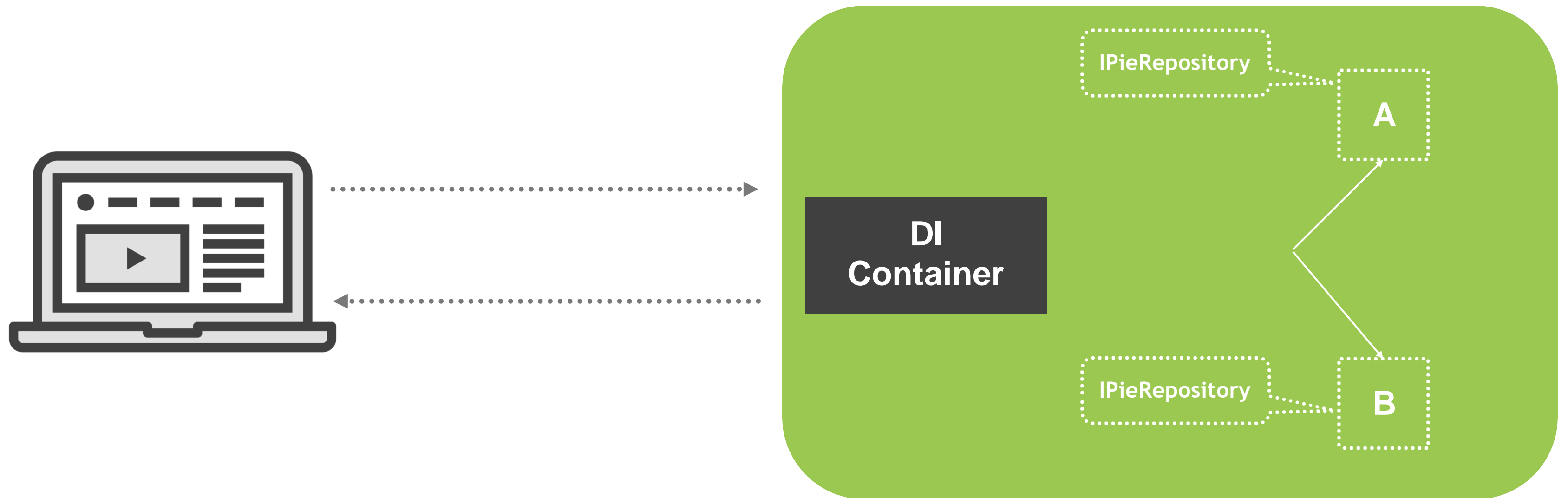
AddTransient

AddSingleton

casi nunca

AddScoped

Entender AddScoped



Demo



Creación del dominio

Añadir el repositorio

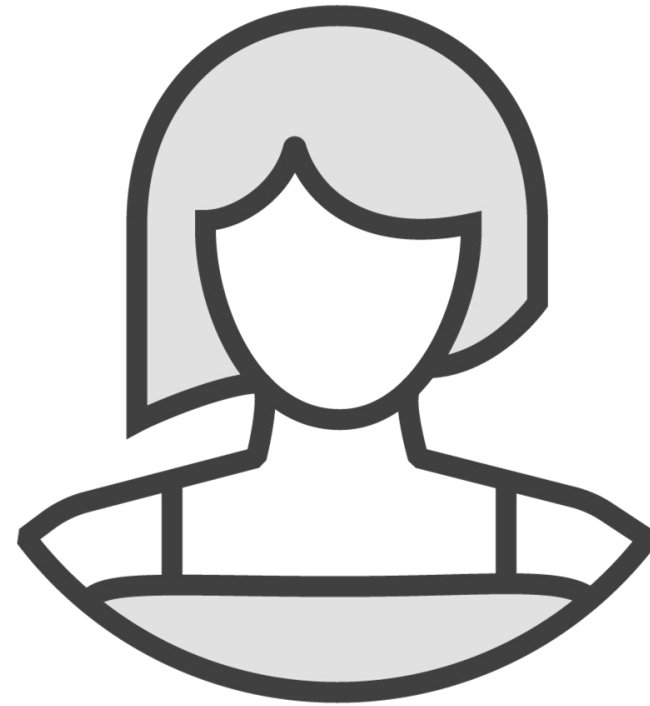
Registro en la colección de servicios

Crear el controlador

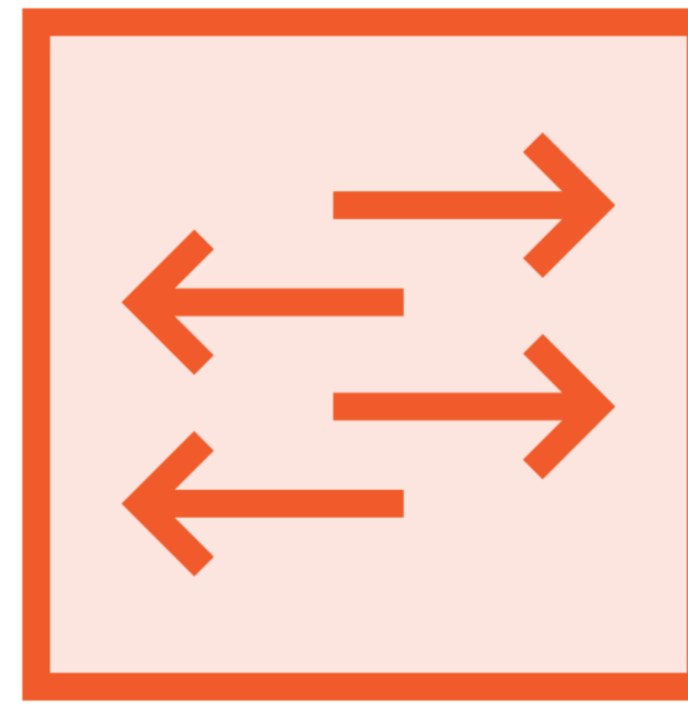
El controlador



Rol Central



**Responde a la
interacción
del usuario**



**Interactúa
con el
modelo y
selecciona
la vista**



**Sin
conocimiento
sobre la
persistencia de
datos**

Un controlador básico

```
public class PieController : Controller
{
    public ActionResult Index() ←..... Acción
    {
        return View(); ←..... Vista a mostrar
    }
}
```

Un controlador real

```
public class PieController : Controller
{
    private readonly IPieRepository _pieRepository;

    public PieController(IPieRepository pieRepository)
    {
        _pieRepository = pieRepository;
    }

    public ViewResult List()
    {
        return View(_pieRepository.Pies);
    }
}
```

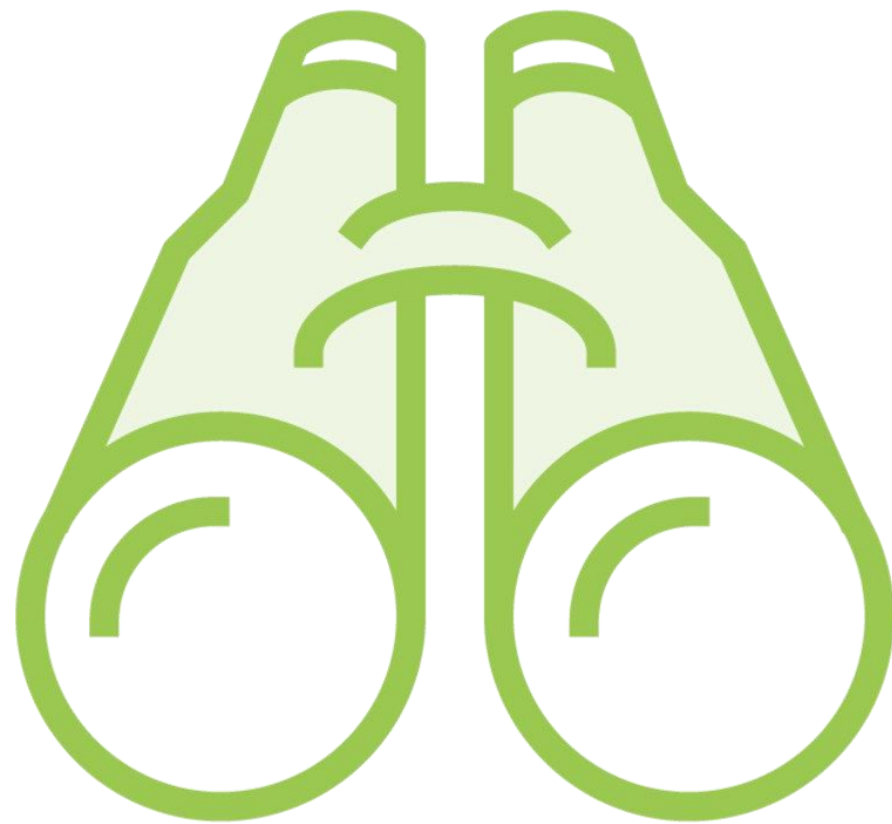
Demo



Agregar el controlador

Agregar la vista

La vista



Plantilla HTML (*.cshtml)

NO ES HTML!!!!!!!!!!!!!!!!!!!!!!

Utiliza Razor

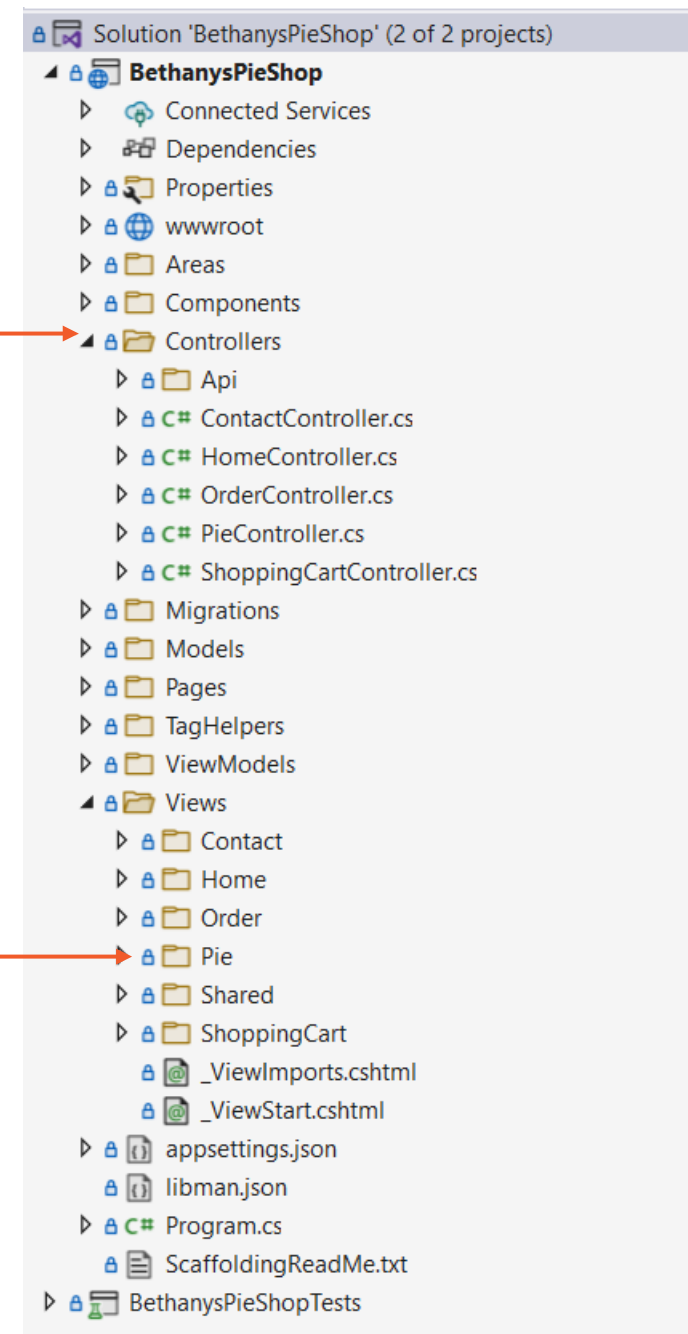
(Casi) sin lógica

- Condiciones, bucles
- Añadir ayudantes de etiquetas y componentes de vista

Correspondencia entre el controlador y sus vistas

PieController

Carpeta Pie



Correspondencia de la acción con la vista

Enfoque basado en convención

```
public class PieController : Controller
{
    public ActionResult Index() ..... Acción
    {
        return View(); ..... Vista a mostrar: Index.cshtml
    }
}
```

Vista normal

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Index</title>
```

```
  </head>
```

```
  <body>
```

```
    <div>
```

```
      Welcome to Bethany's Pie Shop
```

```
    </div>
```

```
  </body>
```

```
</html>
```

Uso de ViewBag procedente del controlador

```
public class PieController : Controller
{
    public ActionResult Index()
    {
        ViewBag.Message = "Welcome to Bethany's Pie Shop";
        return View();
    }
}
```

Contenido dinámico mediante ViewBag

```
<!DOCTYPE html>

<html>
  <head>
    <title>Index</title>
  </head>
  <body>
    <div>
      @ViewBag.Message
    </div>
  </body>
</html>
```

Razor es una sintaxis de
marcado que nos permite incluir
funcionalidad C# en nuestras
páginas web.


```
@ViewBag.Message
```

```
<p>@DateTime.Now</p>
```

```
@{
```

```
    var message = "Welcome to Bethany's  
    Pie Shop";
```

```
}
```

```
<h3>@message</h3>
```

◀ **Uso de ViewBag en Razor**

◀ **Dmostrando una fecha en código Razor**

◀ **Uso de un bloque de código**

```
public class PieController : Controller
{
    public ViewResult List()
    {
        return View(_pieRepository.AllPies);
    }
}
```

Llamada a una vista fuertemente tipada

Una vista fuertemente tipada

```
@model IEnumerable<Pie>

<html>
  <body>
    <div>
      @foreach (var pie in Model)
      {
        <div>
          <h2>@pie.Name</h2>
          <h3>@pie.Price.ToString("c")</h3>
          <h4>@pie.Category.CategoryName</h4>
        </div>
      }
    </div>
  </body>
</html>
```

Demo



Creación de la primera vista

```
public class PieListViewModel
{
    public IEnumerable<Pie>? Pies { get; set; }
    public string? CurrentCategory { get; set; }
}
```

Presentación del modelo de vista (ViewModel)

Demo



Utilizando un View Model

Uso de _Layout.cshtml

Plantilla

**Situada en la
carpeta Shared**
**Buscada de manera
predeterminada**

Una o más
**La vista puede
especificar**

_Layout.cshtml

```
<!DOCTYPE html>
<html>
  <head>
    <title>Bethany's Pie Shop</title>
  </head>
  <body>
    <div>
      @RenderBody() ..... Sustituido por la vista
    </div>
  </body>
</html>
```



```
@{  
    Layout = "_Layout";  
}
```

```
@model PieListViewModel
```

La vista puede especificar el diseño

```
@{  
    Layout = "_Layout";  
}
```

__ViewStart.cshtml

Contains logic shared by set of views
Executed automatically

```
@using BethanysPieShop.Models
```

View Imports

Group commonly used using statements

Demo



Añadir una plantilla de diseño
Creación del archivo ViewStart
Añadir el archivo ViewImports

Dar estilo a la vista

Dónde queremos llegar

BETHANY'S

PIE SHOP


SHOP

CONTACT

Q


REGISTER

LOGIN



Pies of the week


Enjoy a weekly selection of our favorite pies



+ ADD TO CART

Apple Pie


\$12.95



+ ADD TO CART

Pumpkin Pie

\$12.95



+ ADD TO CART

Rhubarb Pie

\$15.95

Añadir bibliotecas del lado del cliente

```
{  
  "version": "1.0",  
  "defaultProvider": "cdnjs",  
  "libraries": [  
    {  
      "library": "bootstrap@5.1.3",  
      "destination": "wwwroot/lib/bootstrap/"  
    }  
  ]  
}
```

Add Client-Side Library

Provider:

cdnjs

Library:

bootstrap@5.1.3

☒ Include all library files

☐ Choose specific files:

☒

Files:

☒

css

☒

bootstrap-grid.css

☒

bootstrap-grid.css.map

☒

bootstrap-grid.min.css

☒

bootstrap-grid.min.css.map

Target Location:

wwwroot/lib/bootstrap/

Install

Cancel

Demo



Añadir Bootstrap a través del Gestor de Bibliotecas

Introducción al aislamiento CSS



Estilos específicos para una página



Ayuda a evitar conflictos entre nombres de estilos



.cshtml.css

Demo



Añadir aislamiento CSS

Resumen



MVC garantiza una buena separación de responsabilidades

- M
- V
- C

Los modelos encapsulan los datos

Las vistas son plantillas que muestran los datos del modelo

Los controladores dirigen el flujo



A continuación:

Acceder a los datos de una
base de datos