

Preparar una aplicación ASP.NET Core 6 MVC

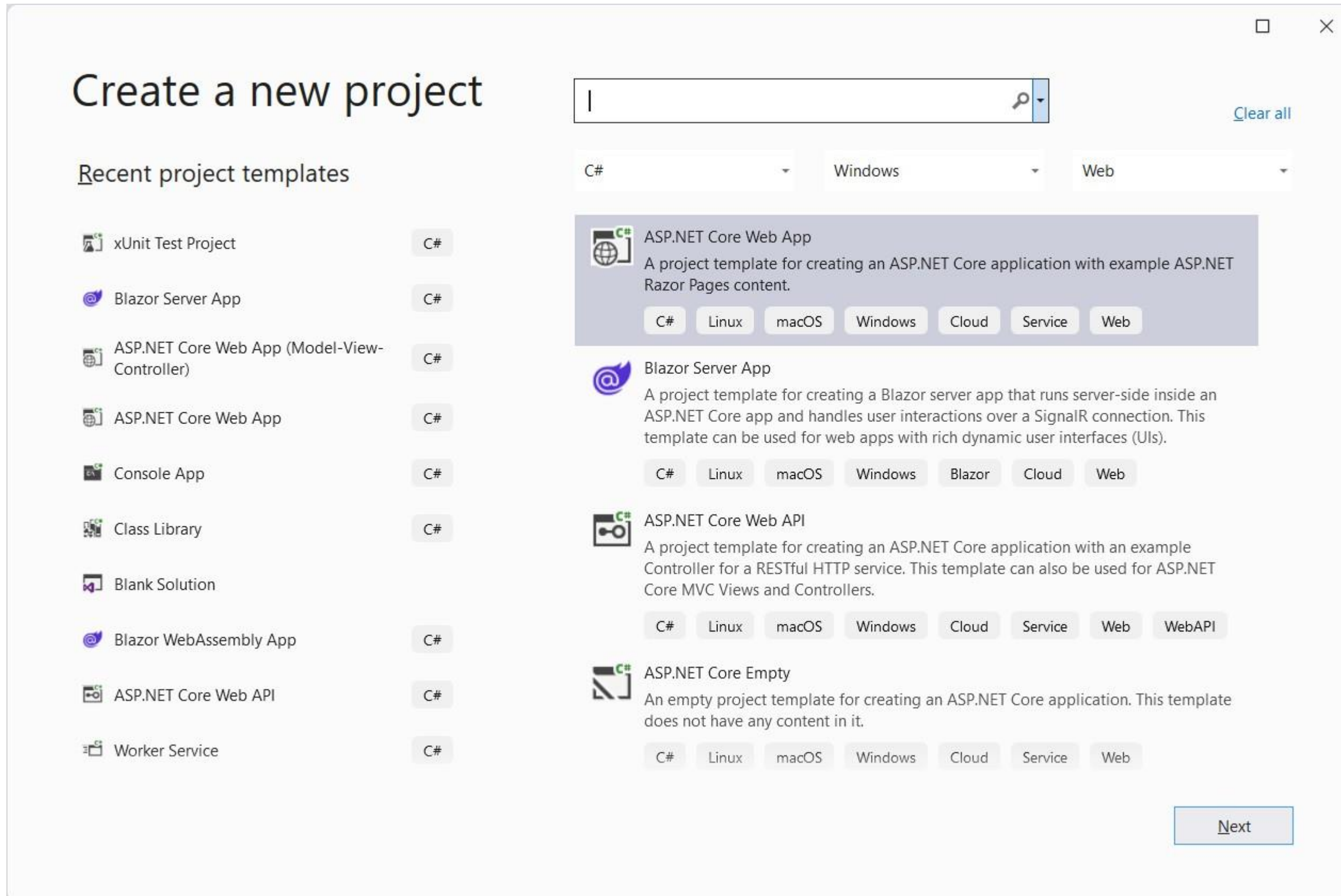
Resumen



- **Crear un proyecto nuevo**
- **Examinar los archivos creados**
- **Configurar el sitio**
- **Cómo gestiona Asp.Net Core una petición**

Crear un proyecto nuevo

Plantillas



Plantillas en el CLI de .NET

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\gill> dotnet new
The 'dotnet new' command creates a .NET project based on a template.

Common templates are:
Template Name      Short Name      Language      Tags
-----
ASP.NET Core Web App webapp, razor  [C#]          Web/MVC/Razor Pages
Blazor Server App  blazorserver   [C#]          Web/Blazor
Class Library      classlib       [C#], F#, VB  Common/Library
Console App        console        [C#], F#, VB  Common/Console
Windows Forms App  winforms       [C#], VB      Common/WinForms
WPF Application    wpf            [C#], VB      Common/WPF

An example would be:
dotnet new console

Display template options with:
dotnet new console -h
Display all installed templates with:
dotnet new --list
Display templates available on NuGet.org with:
dotnet new web --search

PS C:\Users\gill> |
```

Demo



Crear un nuevo proyecto mediante una plantilla

Compilar y ejecutar la aplicación

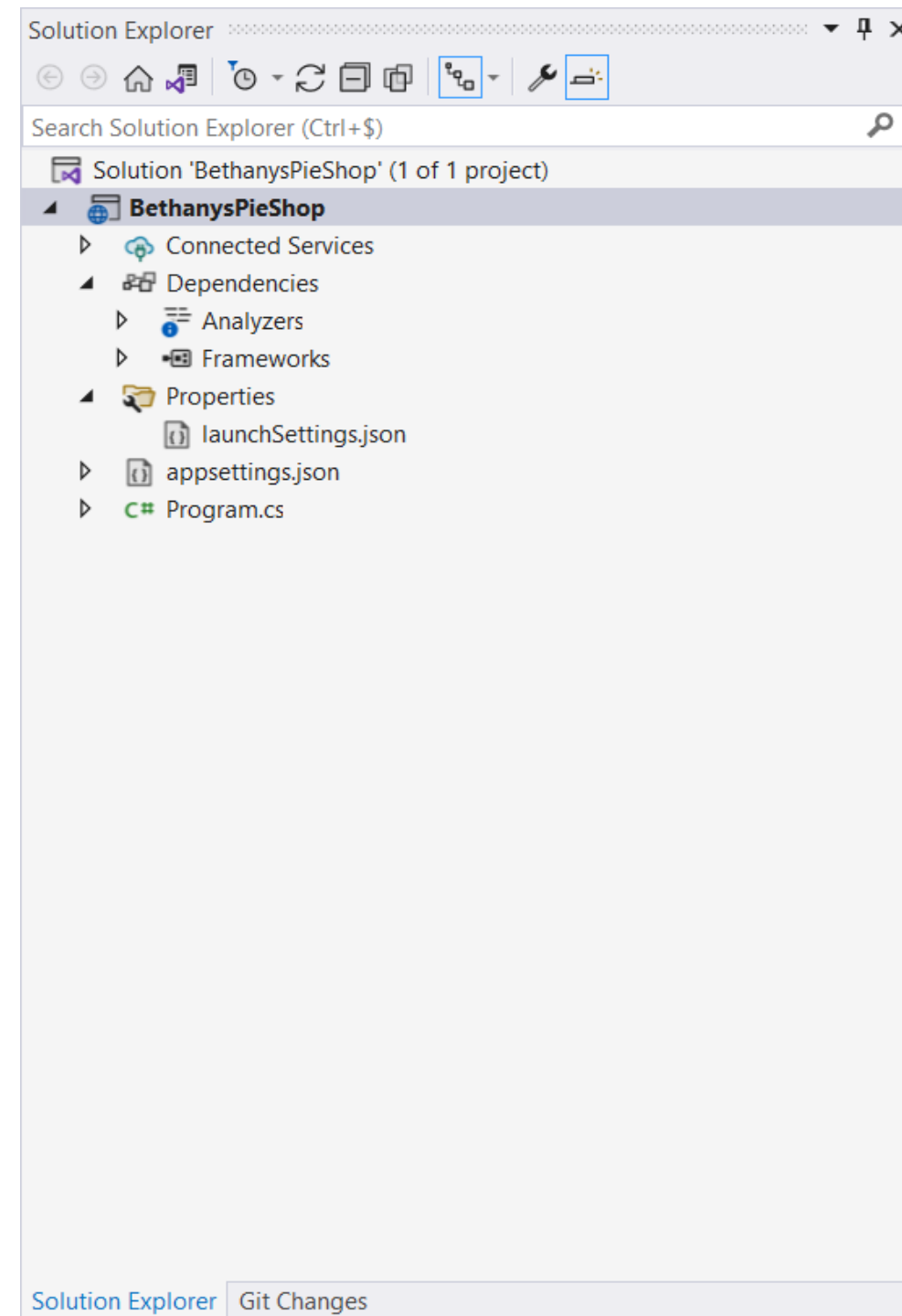
Demo



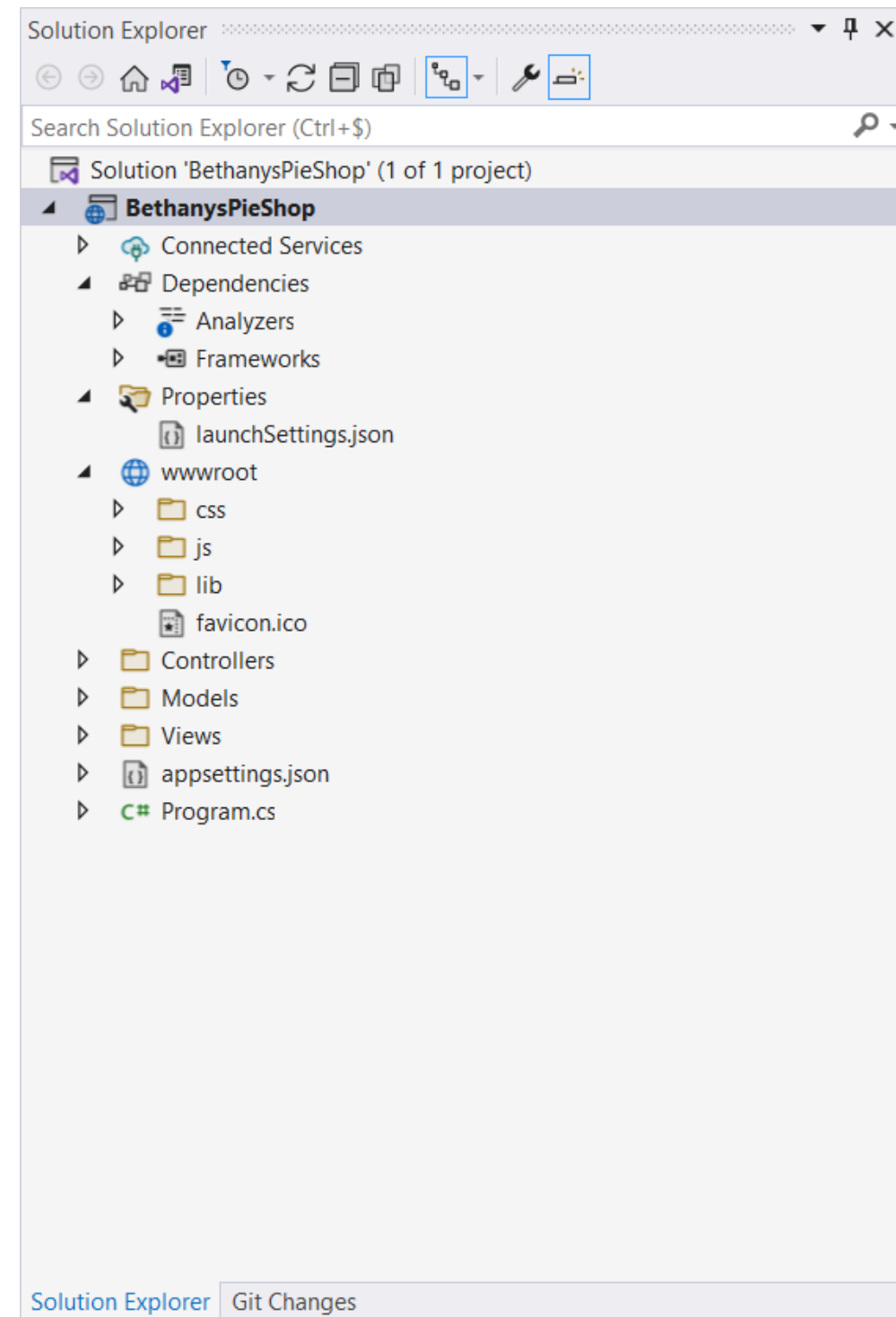
**Uso del CLI para crear una
nueva aplicación web**

Explorando el nuevo proyecto

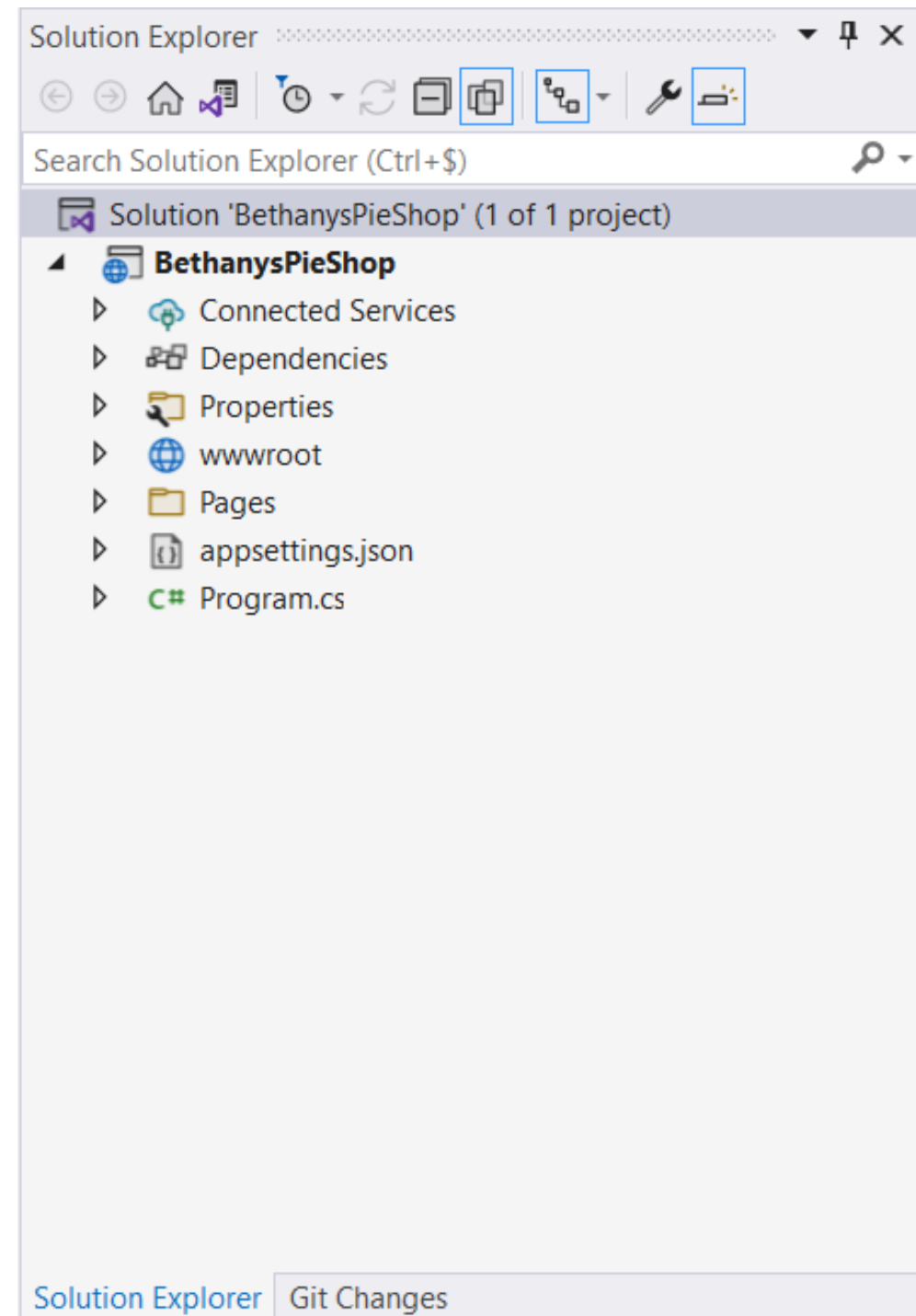
Estructura de proyecto (Aplicación web vacía)



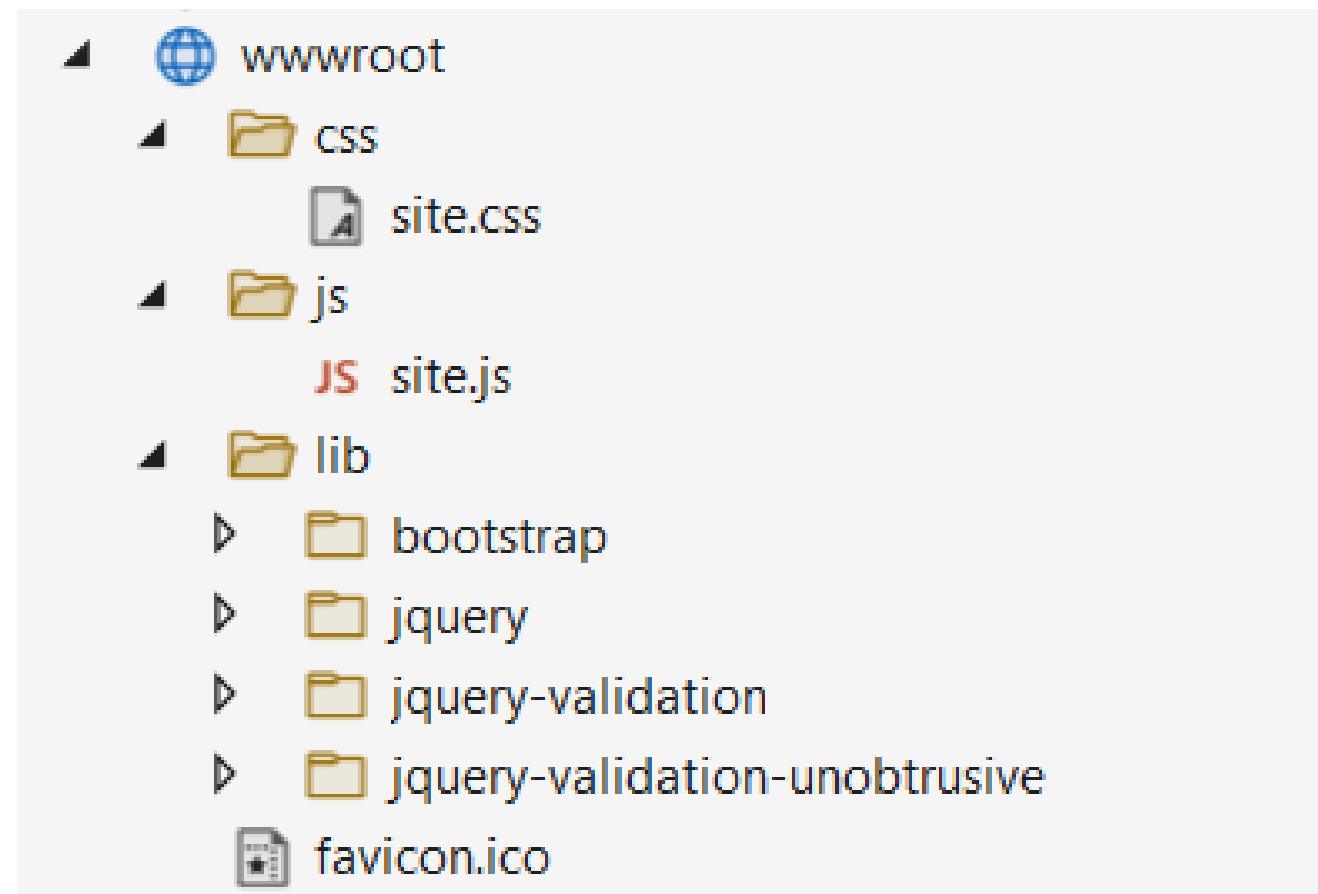
Estructura de proyecto (Aplicación Web MVC)



Estructura de proyecto (Razor Pages)



La carpeta wwwroot



[wwwroot/image1.jpg](#)

<http://bethanyspieshop.com/image1.jpg>

El archivo csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

</Project>
```

Agregando dependencias

```
<Project Sdk="Microsoft.NET.Sdk.Web">
```

```
  <PropertyGroup>
```

```
    <TargetFramework>net6.0</TargetFramework>
```

```
    <Nullable>enable</Nullable>
```

```
    <ImplicitUsings>enable</ImplicitUsings>
```

```
  </PropertyGroup>
```

```
  <ItemGroup>
```

```
    <PackageReference Include="Newtonsoft.Json" Version="13.0.1" />
```

```
  </ItemGroup>
```

```
</Project>
```

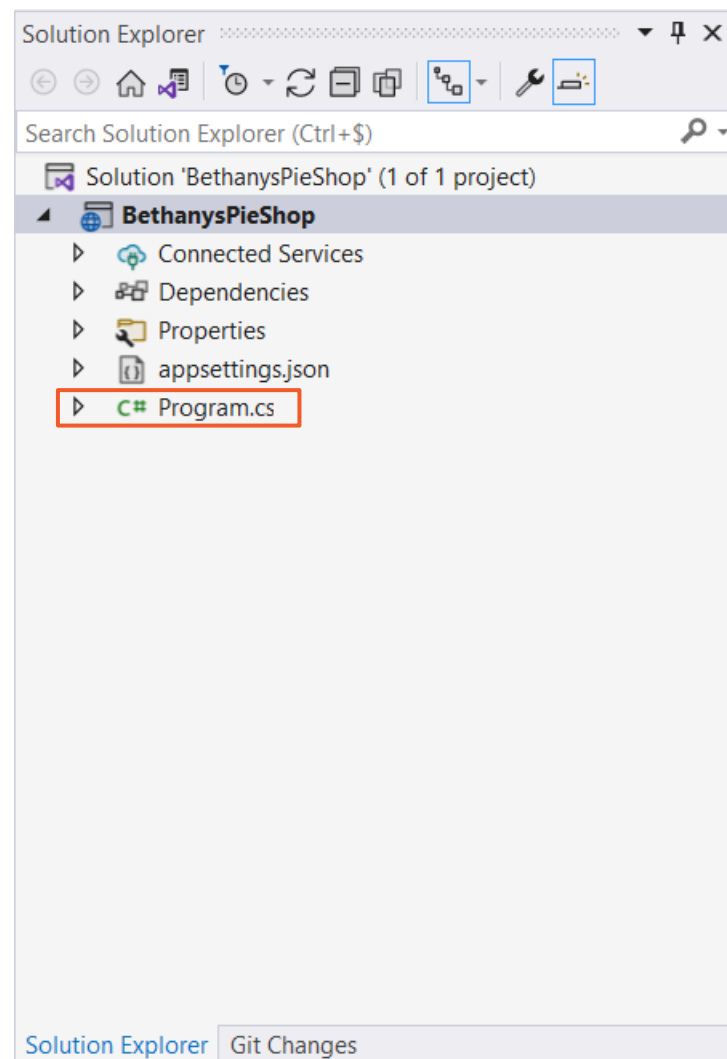
Demo



Explorando los archivos generados
Un vistazo a launchSettings.json

Configurar el sitio web

La clase Program



Las aplicaciones ASP.NET Core applications se ejecutan como las aplicaciones de consola

- static void main
- “Reemplazado” en .NET 6 & C# 10 con instrucciones de alto nivel

Contiene la lógica para iniciar el servidor e iniciar la escucha de peticiones además de configurar la aplicación

```
var builder = WebApplication.CreateBuilder(args);  
var app = builder.Build();  
  
app.MapGet("/", () => "Hello World!");  
  
app.Run();
```

El Program.cs predeterminado

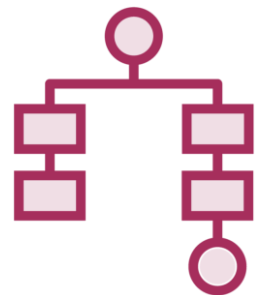
El método CreateBuilder



Configura el servidor Kestrel



Configura la integración con IIS



Especificar el contenido raiz

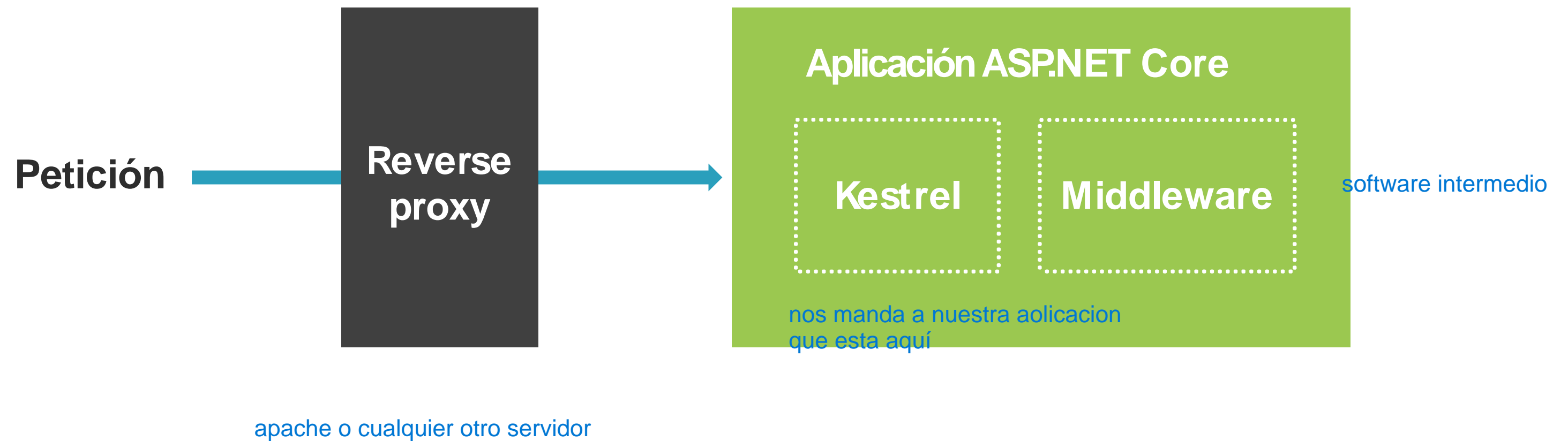


Lee la configuración de la aplicación

```
var builder = WebApplication.CreateBuilder(args);  
var app = builder.Build();  
  
app.MapGet("/", () => "Hello World!");  
  
app.Run();
```

El Program.cs predeterminado

Notas: Ejecutar un servidor mediante Kestrel



Configurar la aplicación

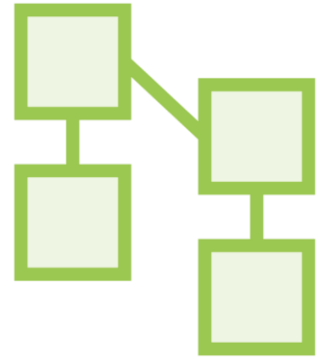
1- Add services to the container.

Registro de servicios

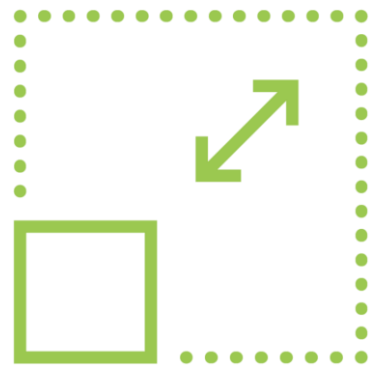
2- Configure the HTTP request pipeline.

Middleware

Registro de servicios



Todas las clases que vaya a necesitar la aplicación



Enfoque modular

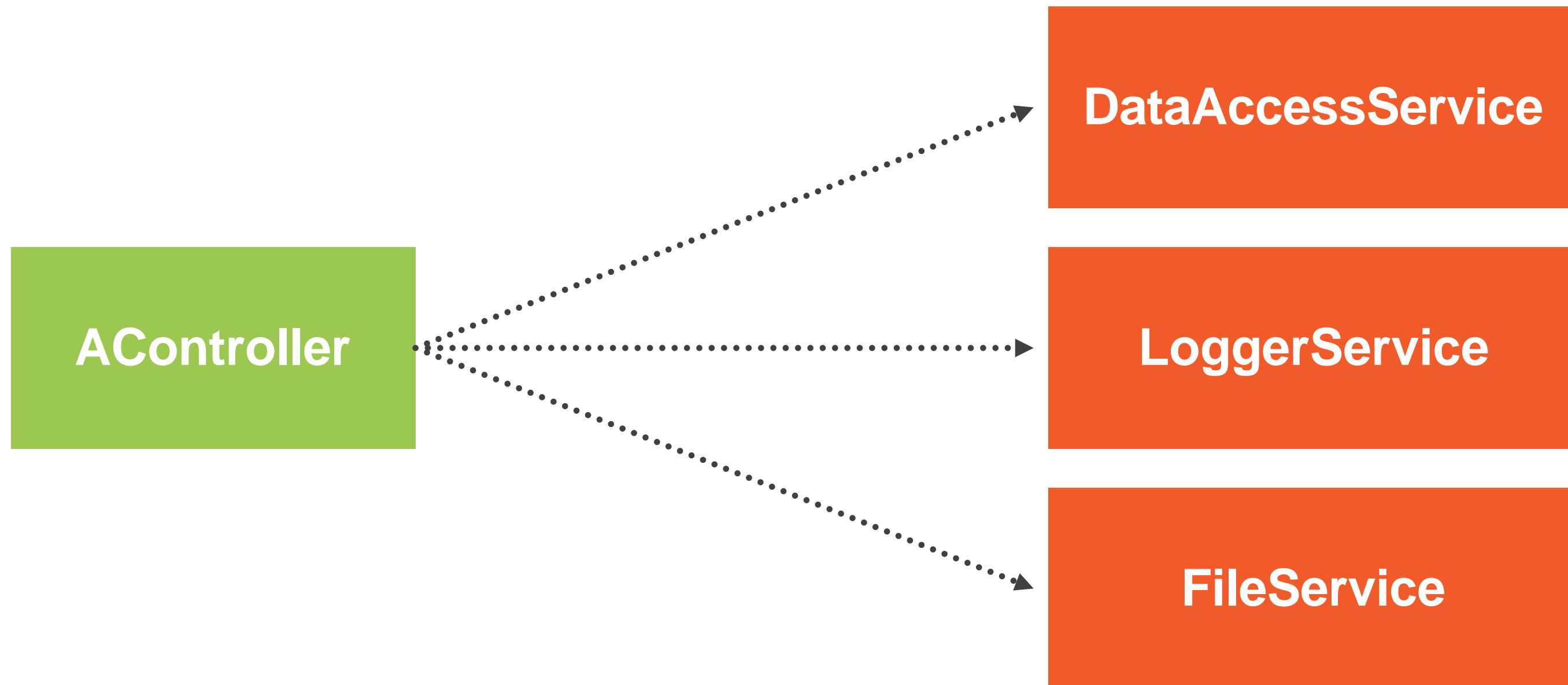


Igual es necesario inyectar mas dependencias

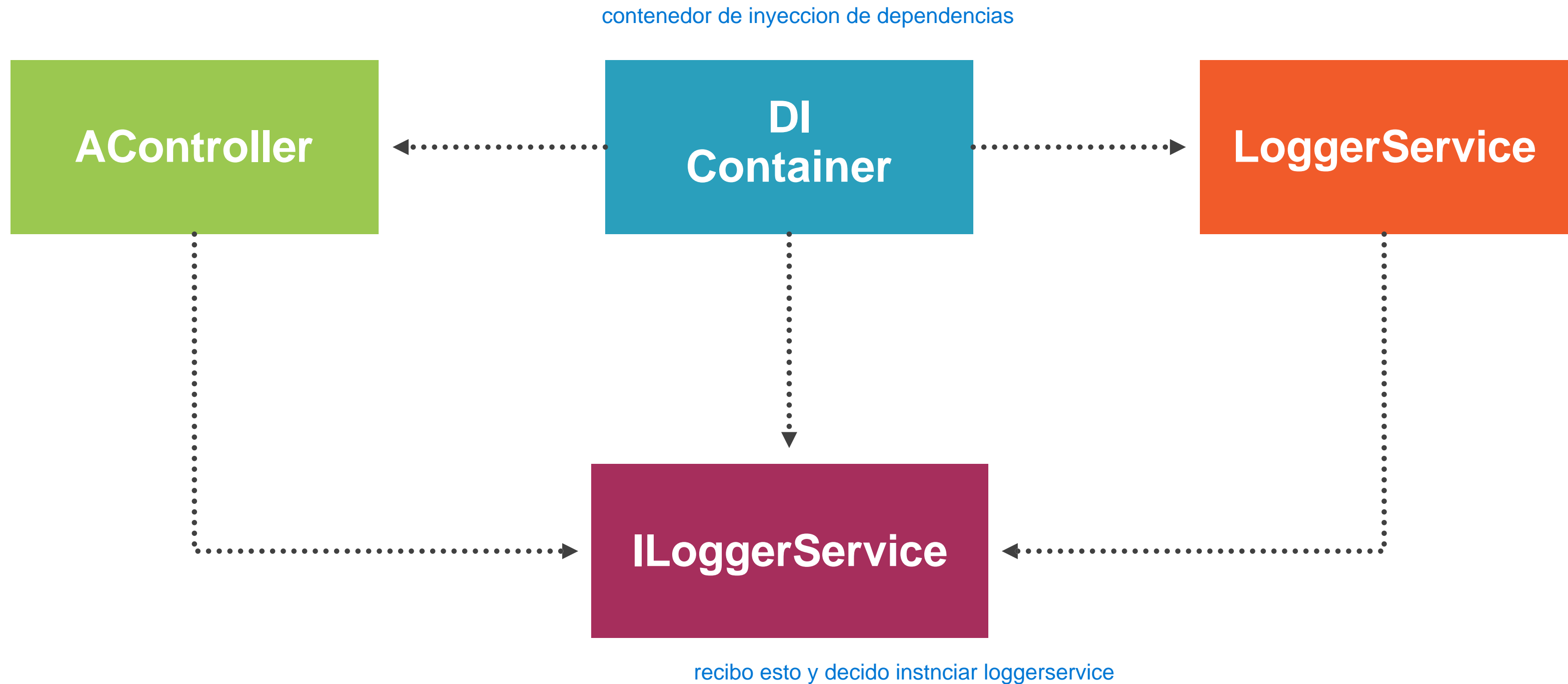
Uso de servicios



Inicialización de dependencias



Introducción a la inyección de dependencias (Dependency Injection - DI)



Registro de servicios

```
var builder = WebApplication.CreateBuilder(args);
```

```
// Add services to the container.
```

```
builder.Services.AddControllersWithViews();
```

```
builder.Services.AddScoped<ILoggerService, LoggerService>();
```

```
var app = builder.Build();
```

```
...
```

```
app.Run();
```

Servicios de framework

**Servicios
personalizados**

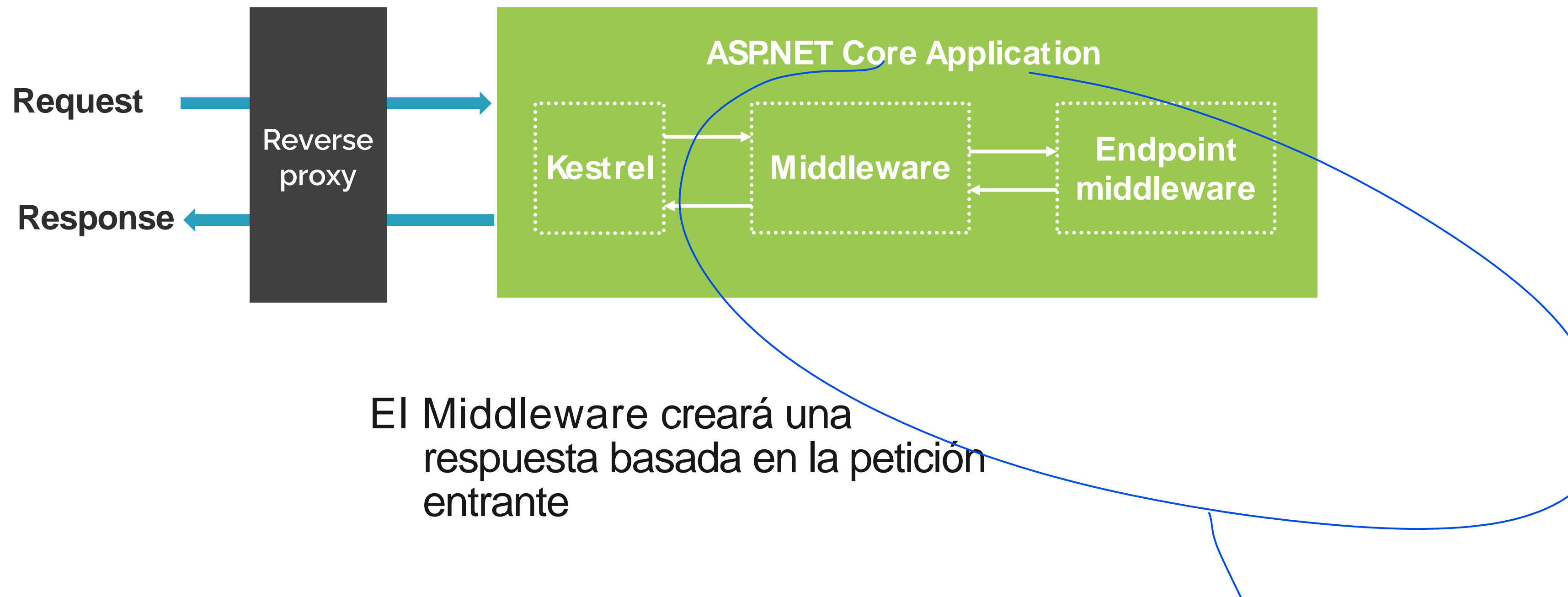
```
public class OrderController : Controller
{
    private readonly ILoggerService _loggerService;

    public OrderController(ILoggerService loggerService)
    {
        _loggerService = loggerService;
    }
}
```

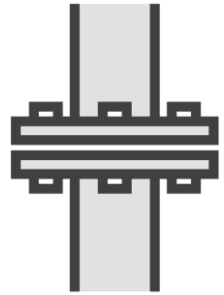
Uso de servicios

Inyectados en el constructor

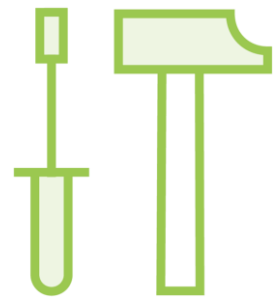
Gestión de peticiones mediante Middleware



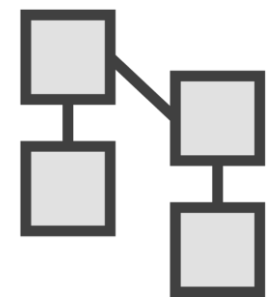
El Middleware Request Pipeline



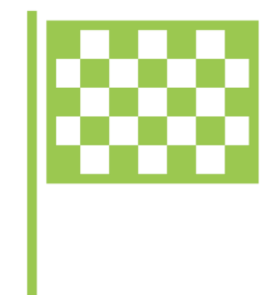
El Pipeline está compuesto por una serie de componentes



Los componentes trabajan tanto en la petición como en la respuesta

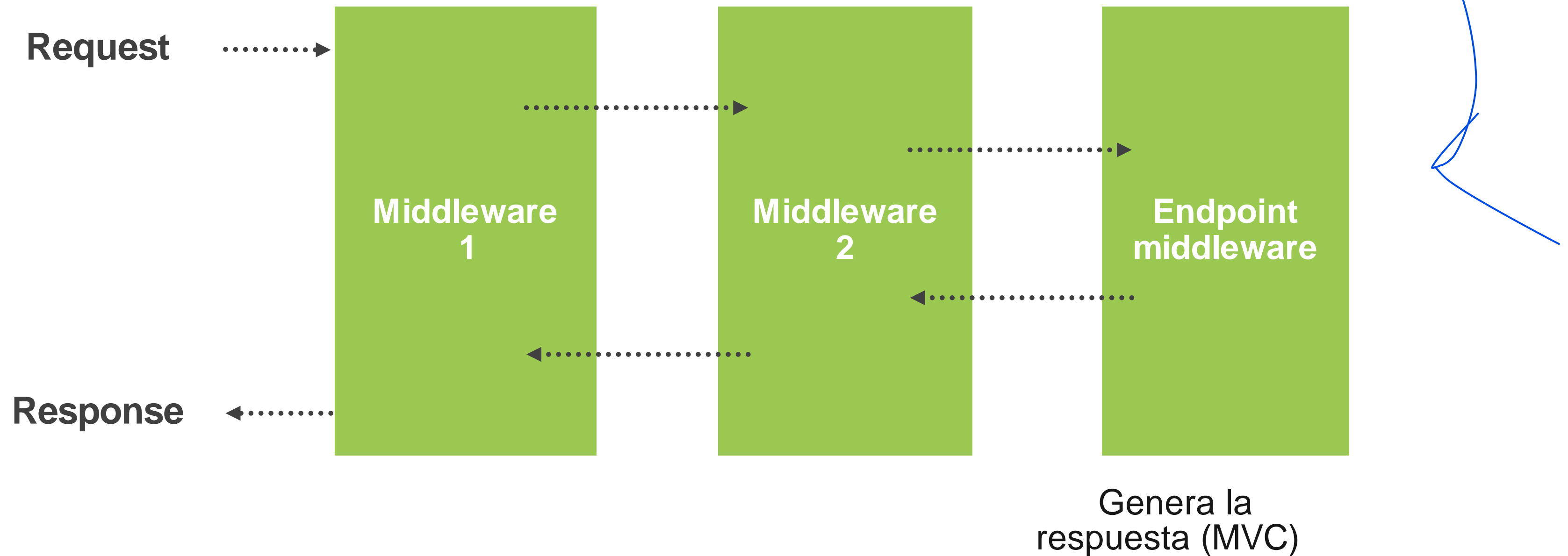


Existen muchos componentes incorporados en Asp.Net Core



Los middlewares de Endpoints están situados al final

El Middleware Request Pipeline



Middleware Request Pipeline

```
var builder = WebApplication.CreateBuilder(args);
```

```
// Add services to the container.  
builder.Services.AddControllersWithViews();
```

```
var app = builder.Build();
```

```
// Configure the HTTP request pipeline.  
if (!app.Environment.IsDevelopment())  
{  
    app.UseExceptionHandler("/Home/Error");  
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see  
    https://aka.ms/aspnetcore-hsts.  
    app.UseHsts();  
}
```

```
app.UseHttpsRedirection();  
app.UseStaticFiles();
```

todo esto son los middleware

```
app.UseRouting();
```

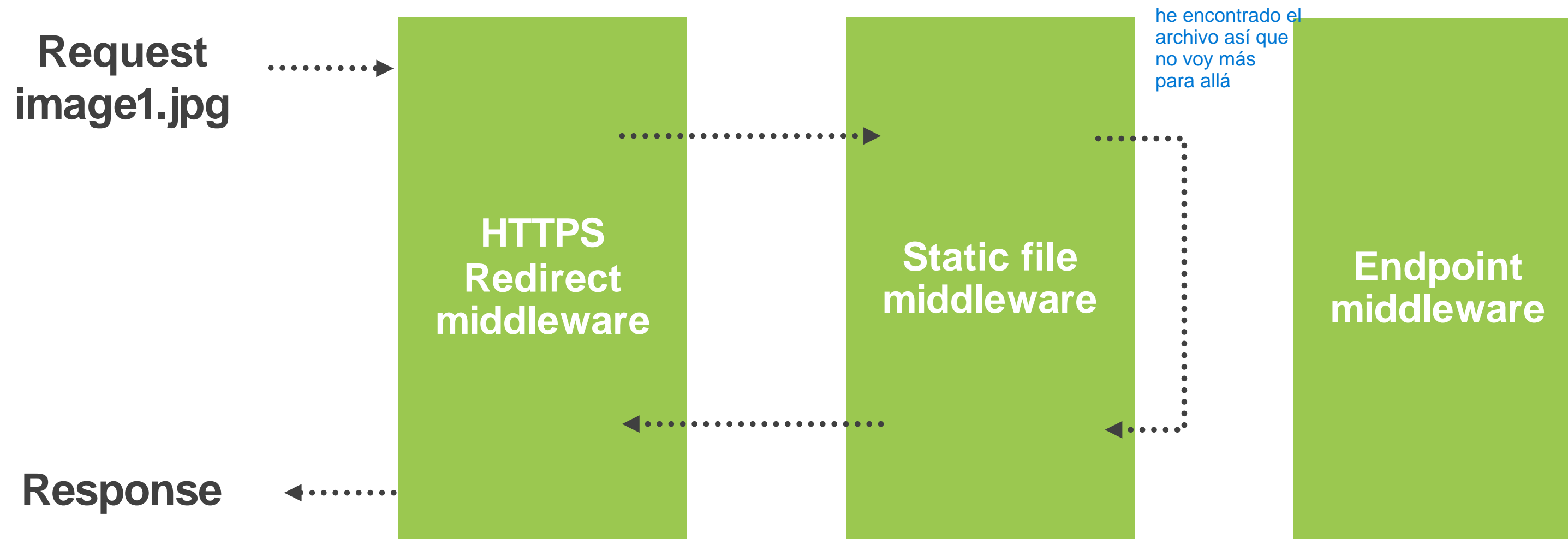
esto tenemos en una aplicacion web

```
app.UseAuthorization();
```

```
app.MapControllerRoute(  
    name: "default",  
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

```
app.Run();
```


El middleware de archivos estáticos



El orden en el que agregamos los componentes, será el orden de los componentes en el pipeline

Program.cs

Registro de servicios

Middleware

Demo



Configurar la aplicación

Resumen



La clase Program configura y arranca la aplicación

ASP.NET Core tiene un servidor incorporado (Kestrel)

Se utiliza la inyección de dependencias de forma predeterminada

La gestión de peticiones se hace mediante middleware (software intermedio)



A continuación:
Crear la primera página