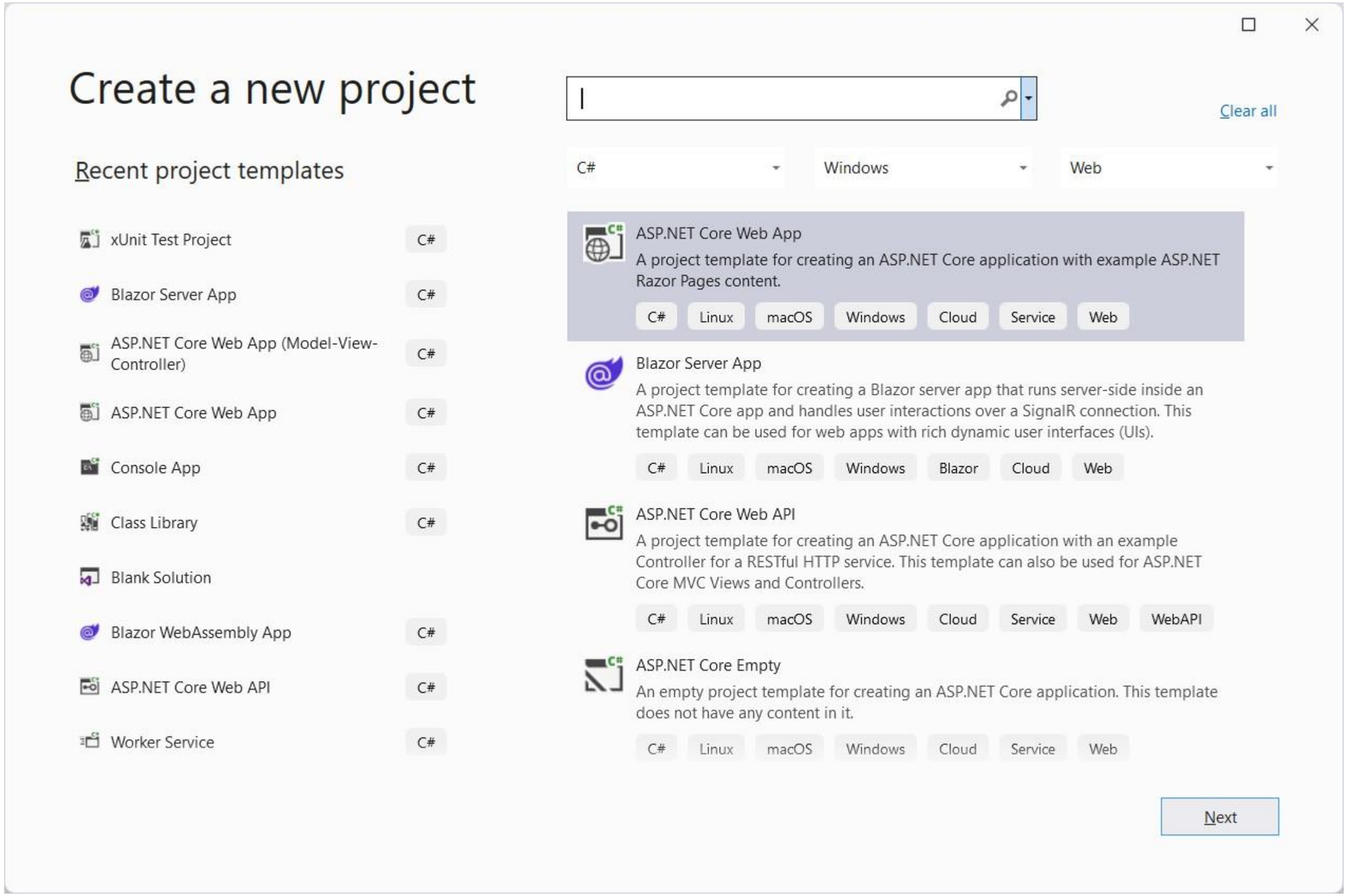
Preparar una aplicación ASP.NET Core 6 MVC

Resumen

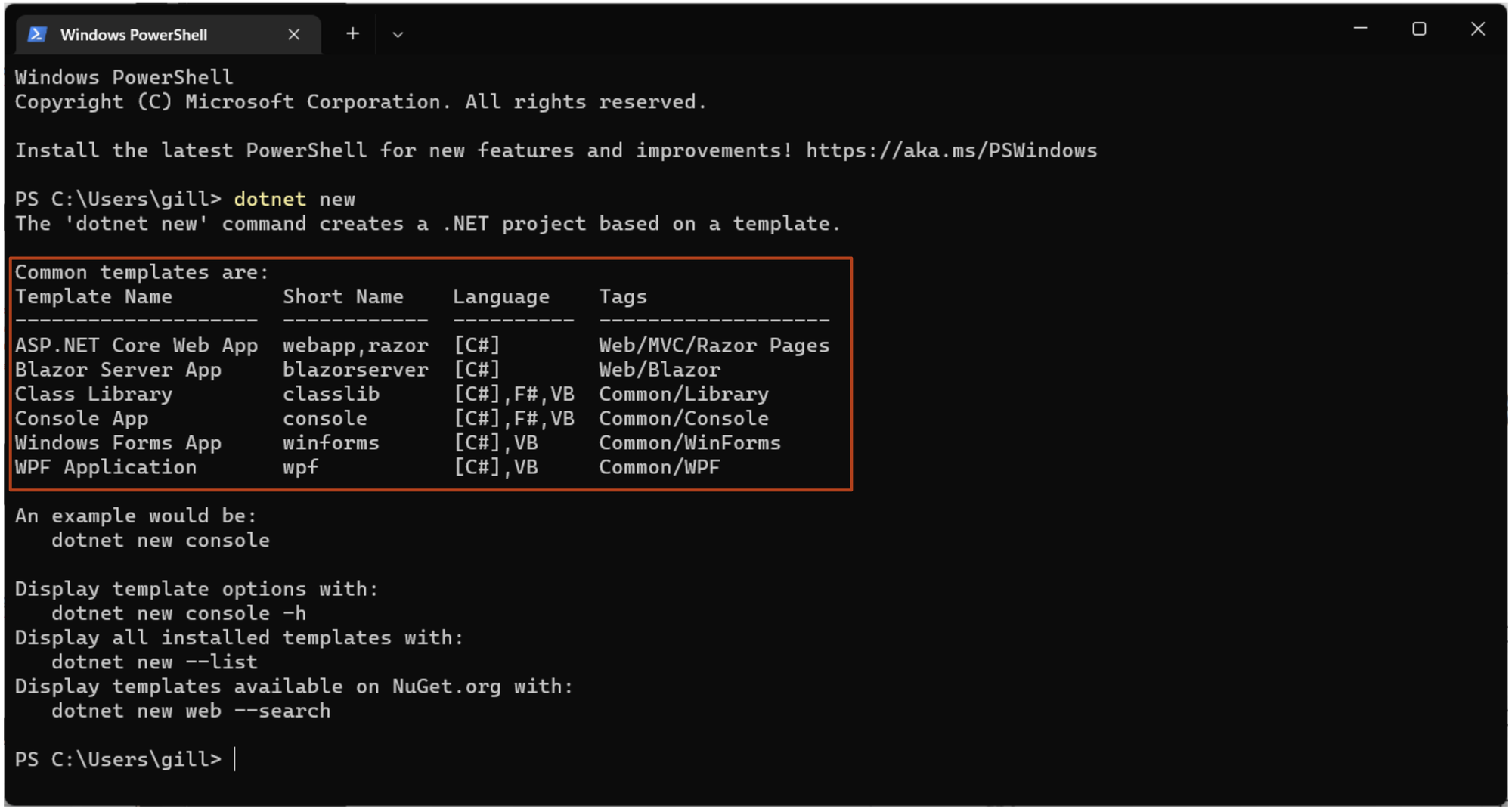
- **Crear un proyecto nuevo**
- **Examinar los archivos creados**
- **Configurar el sitio**
- **Cómo gestiona Asp.Net Core una petición**

Crear un proyecto nuevo

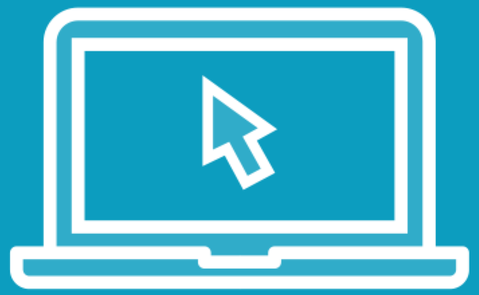
Plantillas


Create a new project
Recent project templates
xUnit Test Project C#
Blazor Server App C#
ASP.NET Core Web App (Model-View-Controller) C#
ASP.NET Core Web App C#
Console App C#
Class Library C#
Blank Solution
Blazor WebAssembly App C#
ASP.NET Core Web API C#
Worker Service C#
Clear all
C#
Windows
Web
ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
C# Linux macOS Windows Cloud Service Web
Blazor Server App
A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Blazor Cloud Web
ASP.NET Core Web API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
C# Linux macOS Windows Cloud Service Web WebAPI
ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
C# Linux macOS Windows Cloud Service Web
Next

Plantillas en el CLI de .NET

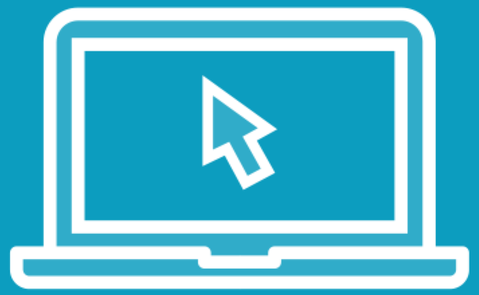


Demo



Crear un nuevo proyecto mediante una plantilla
Compilar y ejecutar la aplicación

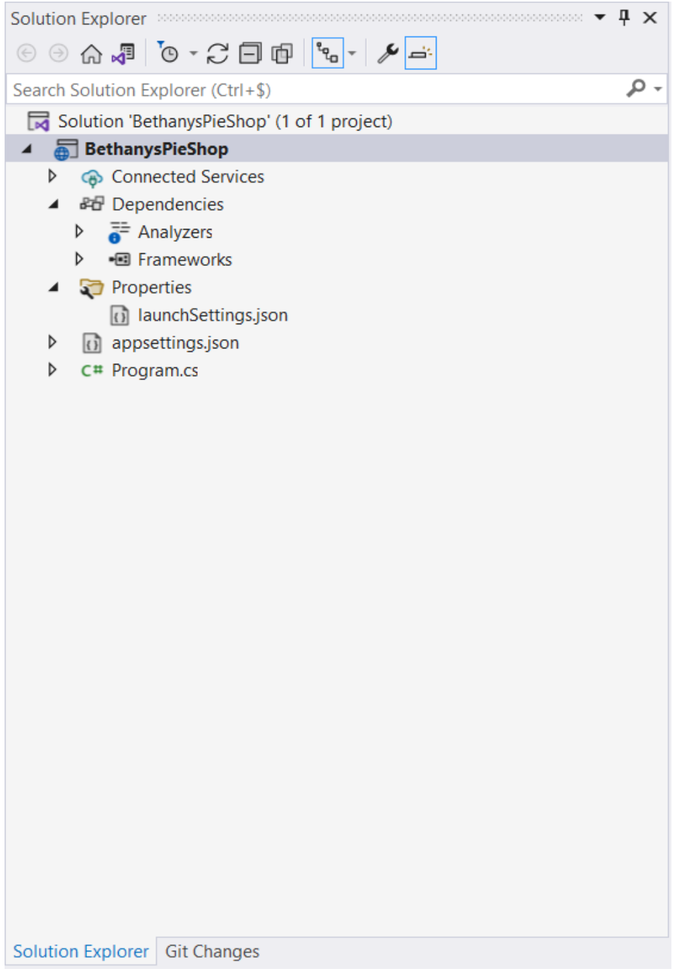
Demo



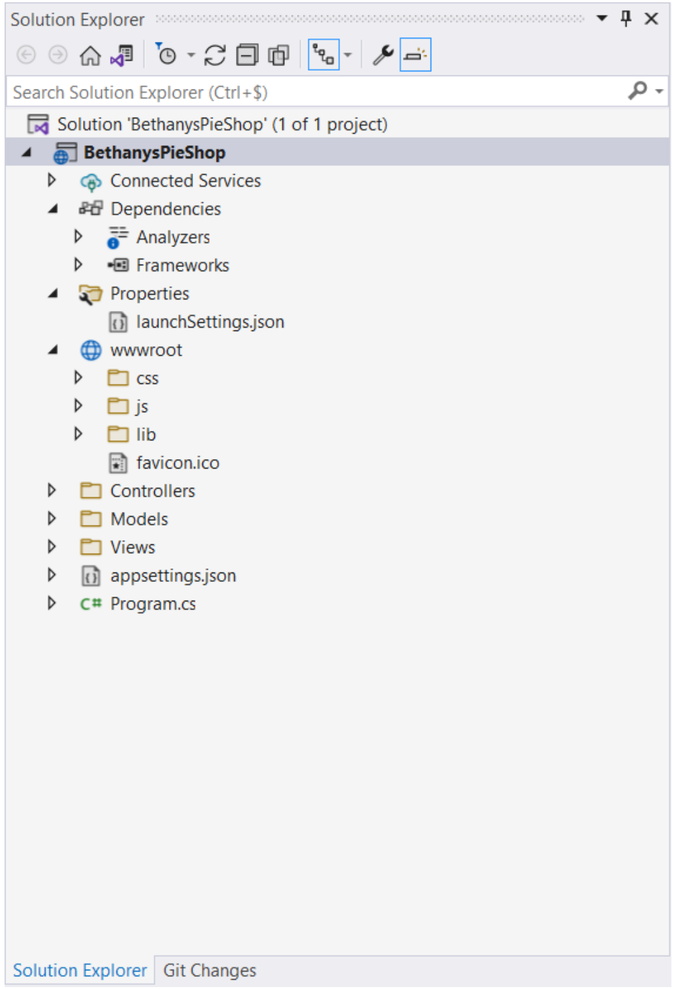
Uso del CLI para crear una nueva aplicación web

Explorando el nuevo proyecto

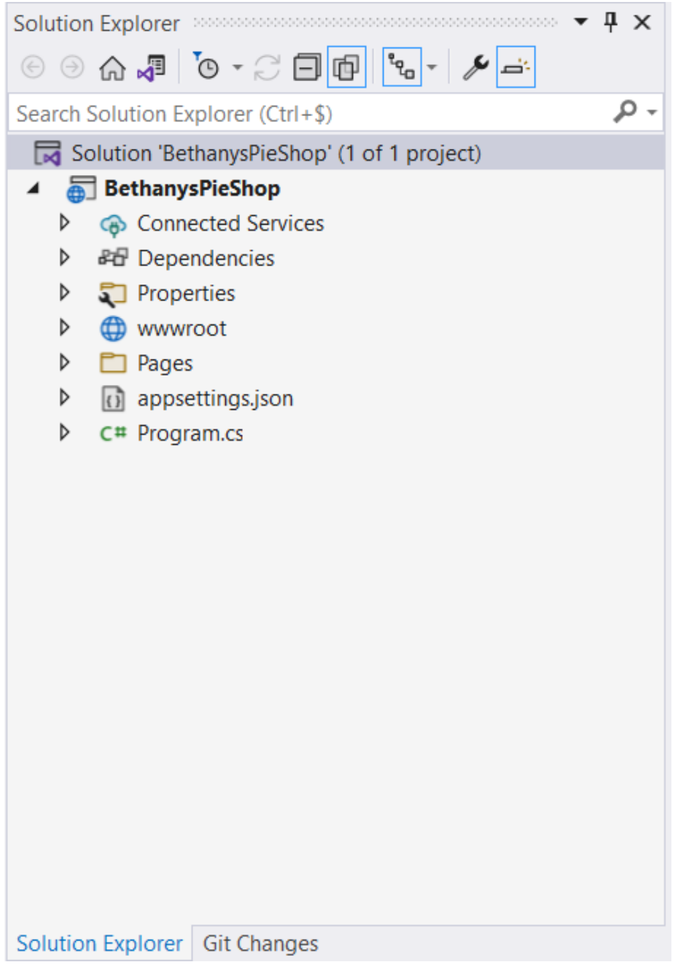
Estructura de proyecto (Aplicación web vacia)



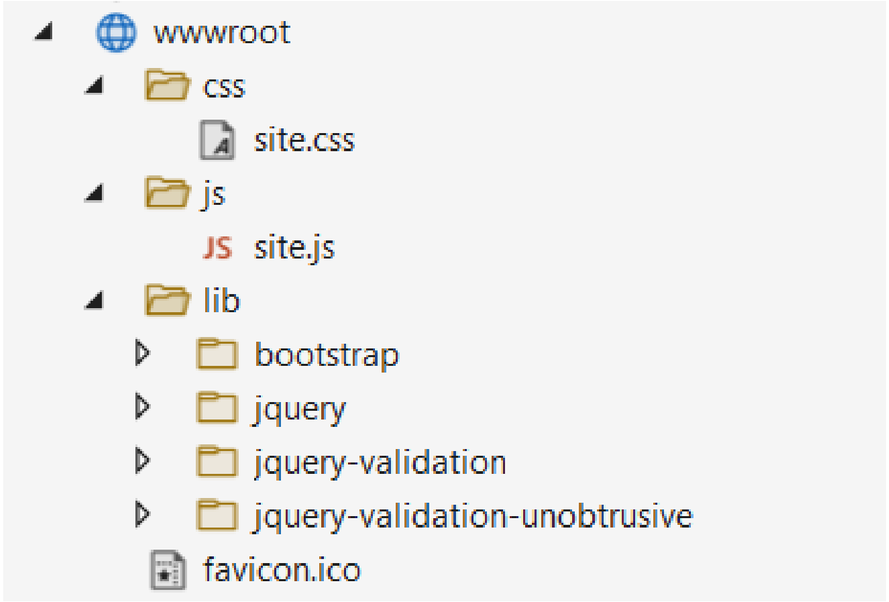
Estructura de proyecto (Aplicación Web MVC)



Estructura de proyecto (Razor Pages)



La carpeta wwwroot



wwwroot/image1.jpg

http://bethanyspieshop.com/image1.jpg

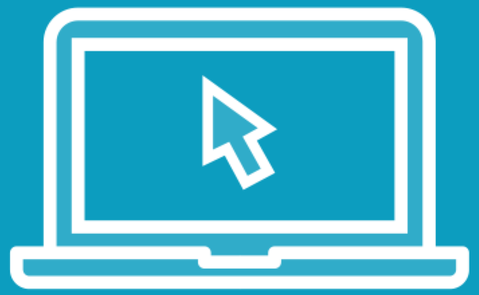
El archivo csproj

```
<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>
```

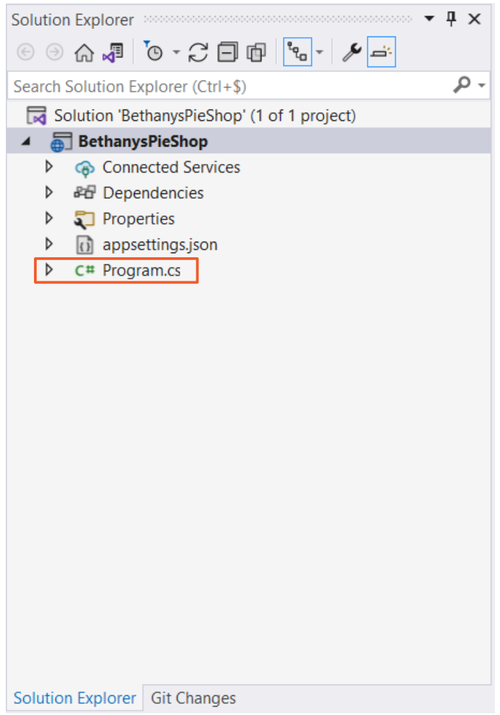
Demo



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

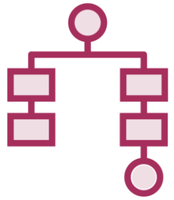
```
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.MapGet("/", () => "Hello World!");

app.Run();
```

El Program.cs predeterminado

El método CreateBuilder

Configura el servidor Kestrel

Configura la integración con IIS



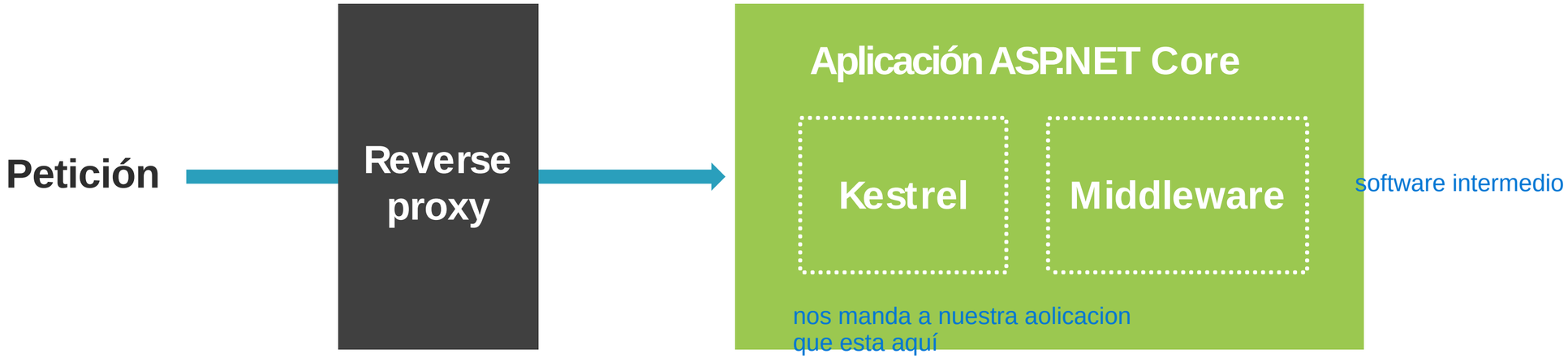
Especificar el contenido raiz

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



software intermedio

apache o cualquier otro servidor

Configurar la aplicación

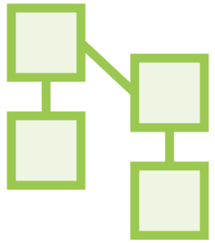
1- Add services to the container.

Registro de servicios

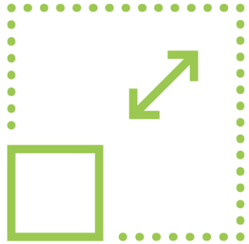
2- Configure the HTTP request pipeline.

Middleware

Registro de servicios



Todas las clases que vaya a necesitar la aplicación

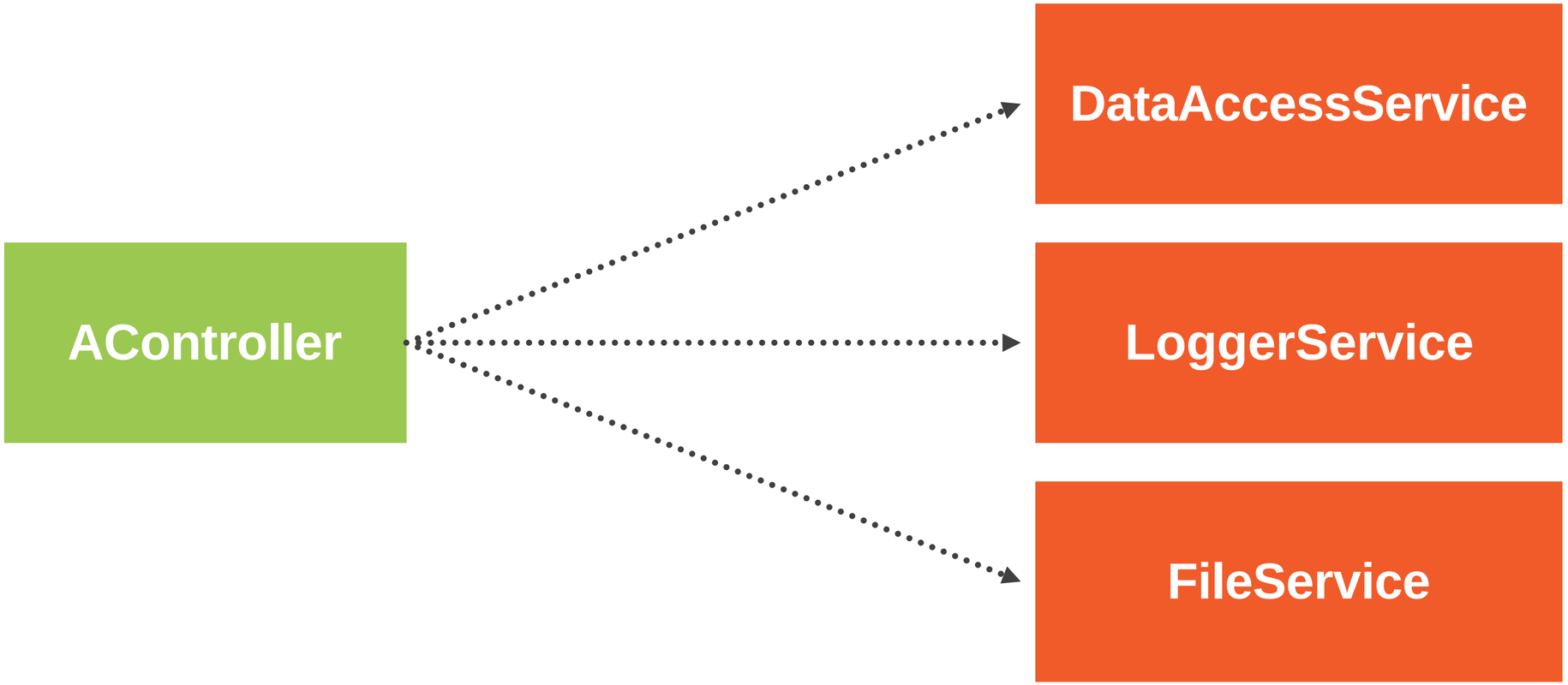


Enfoque modular

Igual es necesario inyectar mas dependencias

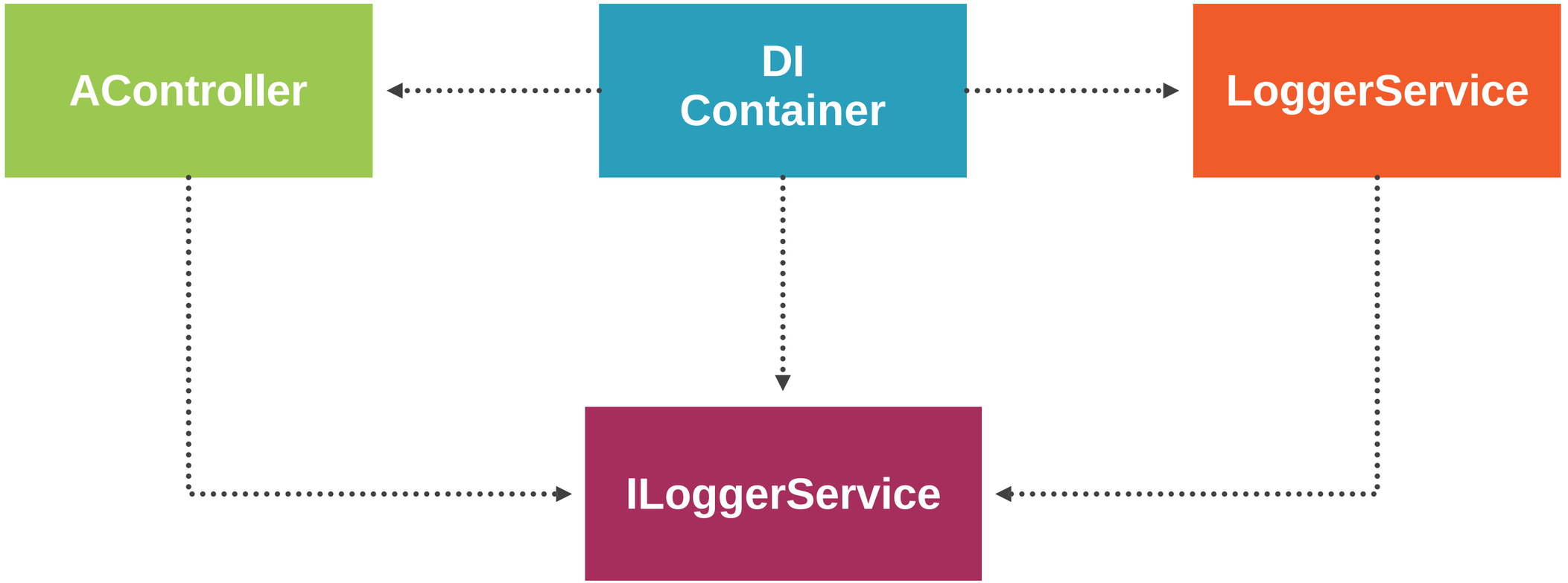
Uso de servicios

Inicialización de dependencias



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

contenedor de inyeccion de dependencias



recibo esto y decido instnciar loggerservice

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

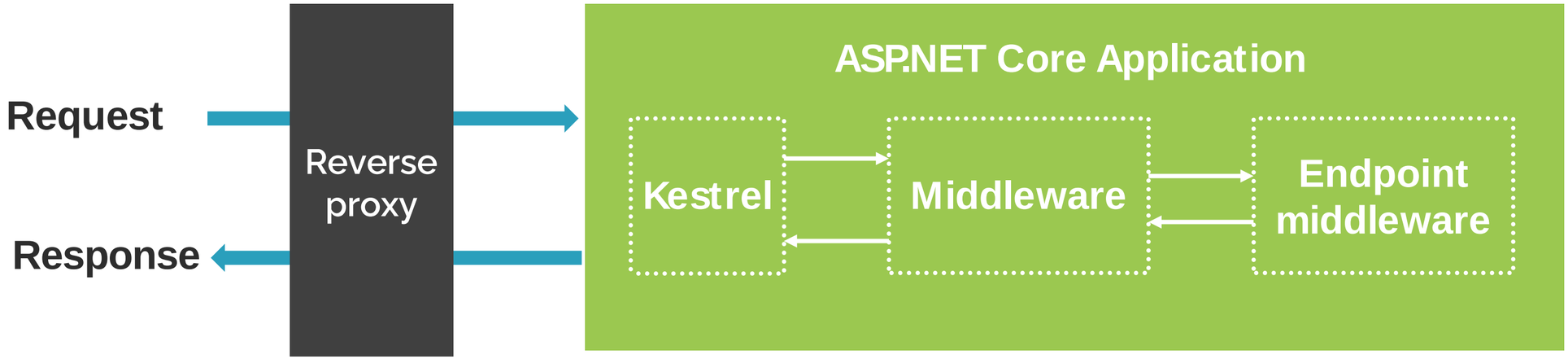
Servicios personalizados

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

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

Uso de servicios

Inyectados en el constructor

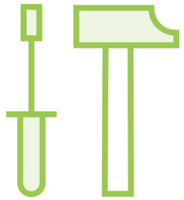
Gestión de peticiones mediante Middleware



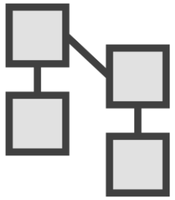
El Middleware creará una respuesta basada en la petición entrante

El Middleware Request Pipeline

El Pipeline está compuesto por una serie de componentes



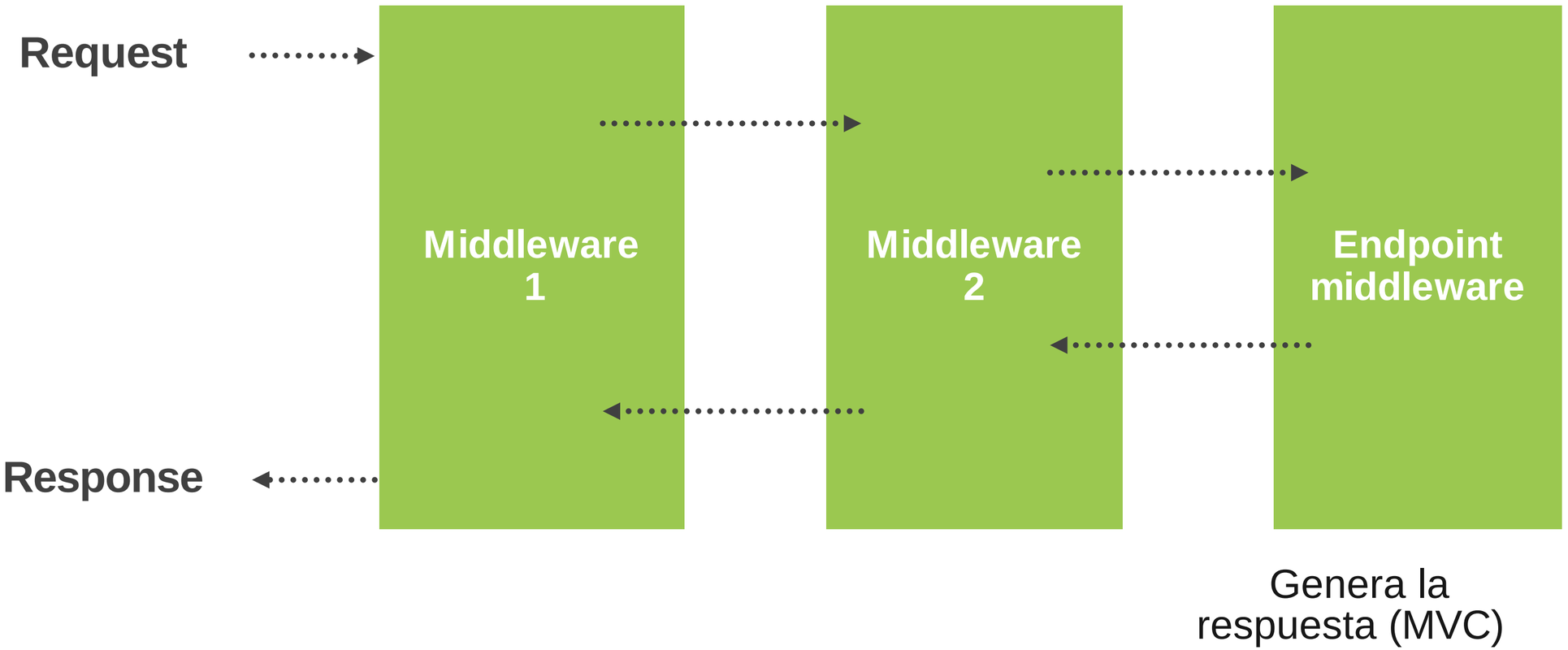
Los componentes trabajan tanto el na petición como en la respuesta



Existen muchos componentes incorporados en Asp.Net Core

Los middlewares de Endpoints están situados al final

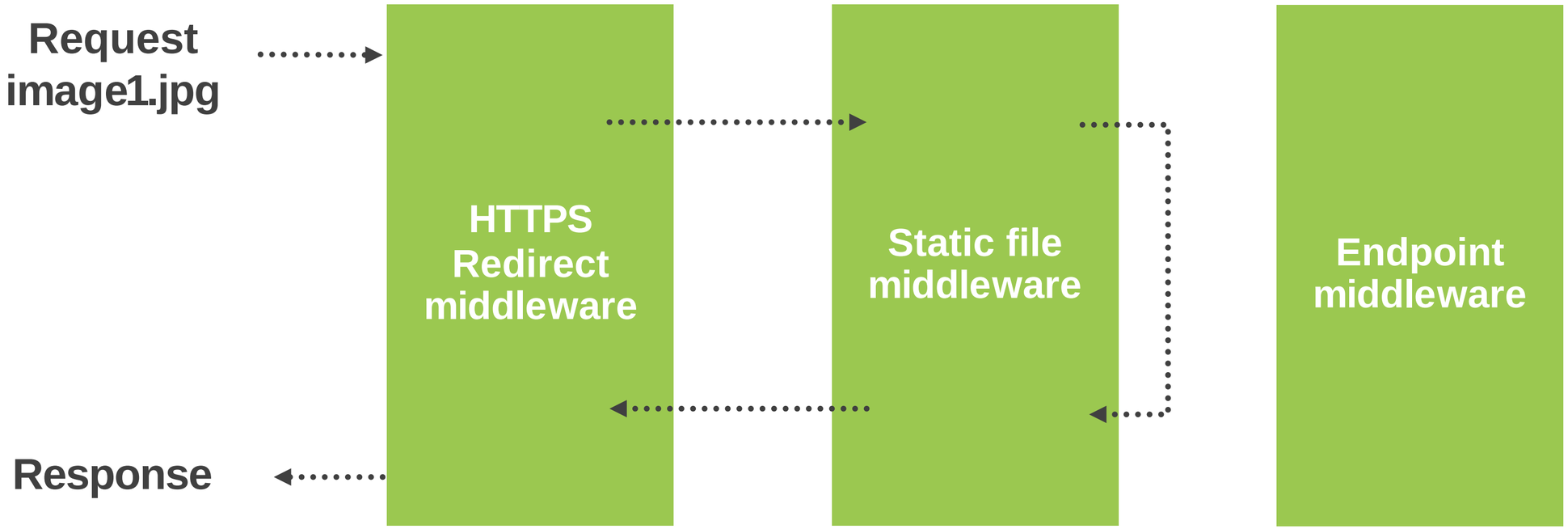
El Middleware Request Pipeline



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();

app.UseRouting();

app.UseAuthorization();

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

app.Run();
```

El middleware de archivos estáticos



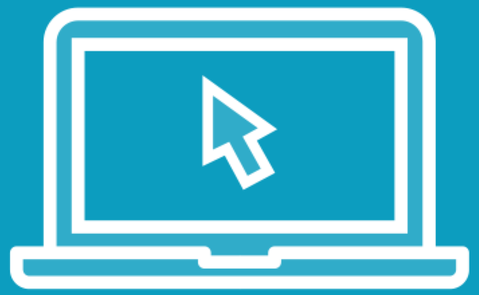
El orden en el que agregamos los componentes, será el orden de los componentes en el pipeline

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