

[MinimalAPIsDemo.zip](#)

Analizaremos las API mínimas en .NET Core 6, su propósito y su implementación paso a paso.

Agenda

- Introducción
- Implementación paso a paso usando .NET Core 6

Requisitos previos

- SDK de .NET Core 6
- Estudio visual 2022
- servidor SQL

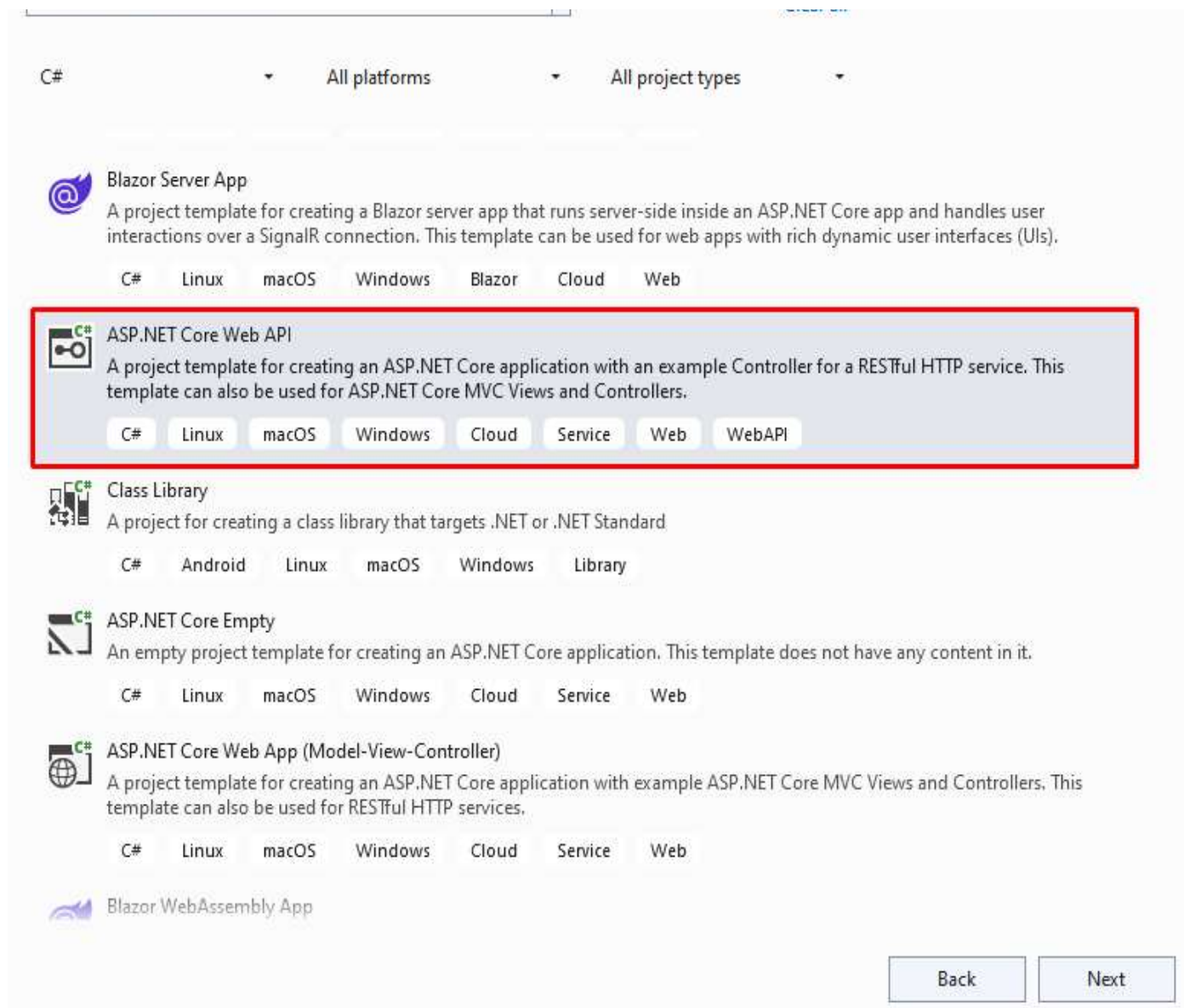
Introducción

- Las API mínimas se utilizan para crear API HTTP con dependencias y configuración mínimas.
- Se utiliza principalmente en microservicios que tienen menos archivos y funcionalidades dentro de un solo archivo.
- Pero hay algunas cosas que no son compatibles con API mínimas, como filtros de acción y validación integrada, además, algunas más que aún están en progreso y que el equipo .NET obtendrá en el futuro.

Implementación mínima de API utilizando .NET Core 6

Paso 1

Cree una nueva API web de .NET Core



C# All platforms All project types

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

Class Library
A project for creating a class library that targets .NET or .NET Standard
C# Android Linux macOS Windows Library

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

ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
C# Linux macOS Windows Cloud Service Web

Blazor WebAssembly App

Back Next

Paso 2

Configura tu proyecto

ASP.NET Core Web API C# Linux macOS Windows Cloud Service Web WebAPI

Project name

MinimalAPIsDemo

Location

D:\

Solution name ⓘ

MinimalAPIsDemo

☐ Place solution and project in the same directory

Paso 3

Proporcione información adicional como se muestra a continuación.

Additional information

ASP.NET Core Web API C# Linux macOS Windows Cloud Service Web WebAPI

Framework ⓘ

.NET 6.0 (Long-term support)

Authentication type ⓘ

None

☐ Configure for HTTPS ⓘ

☐ Enable Docker ⓘ

Docker OS ⓘ

Linux

☐ Use controllers (uncheck to use minimal APIs) ⓘ ←

☒ Enable OpenAPI support ⓘ

☐ Do not use top-level statements ⓘ

Etapas 4

Instale los siguientes paquetes NuGet

```

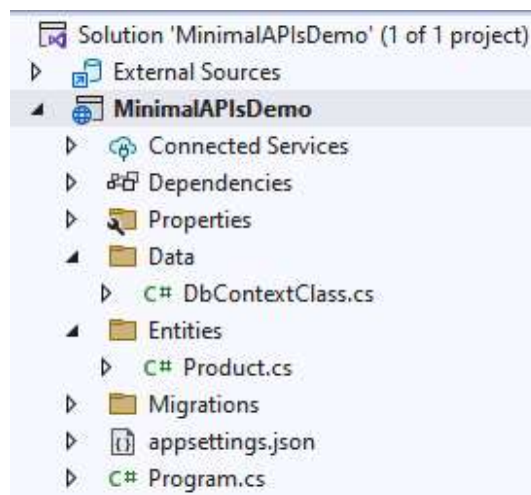
</PropertyGroup>
  <TargetFramework>net6.0</TargetFramework>
  <Nullable>disable</Nullable>
  <ImplicitUsings>enable</ImplicitUsings>
</PropertyGroup>

<ItemGroup>
  <PackageReference Include="Swashbuckle.AspNetCore" Version="6.2.3" />
  <PackageReference Include="Microsoft.EntityFrameworkCore" Version="6.0.7" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="6.0.7">
    <PrivateAssets>all</PrivateAssets>
    <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
  </PackageReference>
  <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="6.0.7" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="6.0.7">
    <PrivateAssets>all</PrivateAssets>
    <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
  </PackageReference>
</ItemGroup>

</Project>

```

Estructura del proyecto



Paso 5

Cree una clase de Producto dentro de la carpeta de entidades

```

1 namespace MinimalAPIsDemo.Entities
2 {
3     public class Product
4     {
5         public int ProductId { get; set; }
6         public string ProductName { get; set; }
7         public string ProductDescription { get; set; }
8         public int ProductPrice { get; set; }
9         public int ProductStock { get; set; }
10    }

```

Paso 6

A continuación, cree DbContextClass dentro de la carpeta Datos

```
using Microsoft.EntityFrameworkCore;
using MinimalAPIDemo.Entities;

namespace MinimalAPIDemo.Data
{
    public class DbContextClass : DbContext
    {
        protected readonly IConfiguration Configuration;

        public DbContextClass(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        protected override void OnConfiguring(DbContextOptionsBuilder options)
        {
            options.UseSqlServer(Configuration.GetConnectionString("DefaultCon

        }

        public DbSet<Product> Product { get; set; }
    }
}
```

Paso 7

Registre el servicio Db Context en el contenedor DI dentro de la clase Programa que es el punto de entrada de nuestra aplicación.

```
1 // Add services to the container.
2 builder.Services.AddDbContext<DbContextClass>();
```

Paso 8

Agregue una cadena de conexión a la base de datos dentro del archivo de configuración de la aplicación

```
{
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft.AspNetCore": "Warning"
```

```
8     "AllowedHosts": "*",
9     "ConnectionStrings": {
10         "DefaultConnection": "Data Source=DESKTOP;Initial Catalog=MinimalAPID
11     }
12 }
```

Paso 9

Más adelante, agregue diferentes puntos finales de API dentro de la clase Programa con la ayuda de Mapa y el patrón de enrutamiento especificado como se muestra a continuación.

```
using Microsoft.EntityFrameworkCore;
using MinimalAPIDemo.Data;
using MinimalAPIDemo.Entities;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddDbContext<DbContextClass>();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

//get the list of product
app.MapGet("/productlist", async (DbContextClass dbContext) =>
{
    var products = await dbContext.Product.ToListAsync();
    if (products == null)
    {
        return Results.NoContent();
    }
    return Results.Ok(products);
});
```

```
app.MapGet("/getproductbyid", async (int id, DbContextClass dbContext) =>
{
    var product = await dbContext.Product.FindAsync(id);
    if (product == null)
    {
        return Results.NotFound();
    }
    return Results.Ok(product);
});

//create a new product
app.MapPost("/createproduct", async (Product product, DbContextClass dbContext) =>
{
    var result = dbContext.Product.Add(product);
    await dbContext.SaveChangesAsync();
    return Results.Ok(result.Entity);
});

//update the product
app.MapPut("/updateproduct", async (Product product, DbContextClass dbContext) =>
{
    var productDetail = await dbContext.Product.FindAsync(product.ProductId);
    if (productDetail == null)
    {
        return Results.NotFound();
    }
    productDetail.ProductName = product.ProductName;
    productDetail.ProductDescription = product.ProductDescription;
    productDetail.ProductPrice = product.ProductPrice;
    productDetail.ProductStock = product.ProductStock;

    await dbContext.SaveChangesAsync();
    return Results.Ok(productDetail);
});

//delete the product by id
app.MapDelete("/deleteproduct/{id}", async (int id, DbContextClass dbContext) =>
{
    var product = await dbContext.Product.FindAsync(id);
    if (product == null)
    {
        return Results.NoContent();
    }
}
```



```
80     return Results.Ok();  
81 });  
82  
83 app.Run();
```

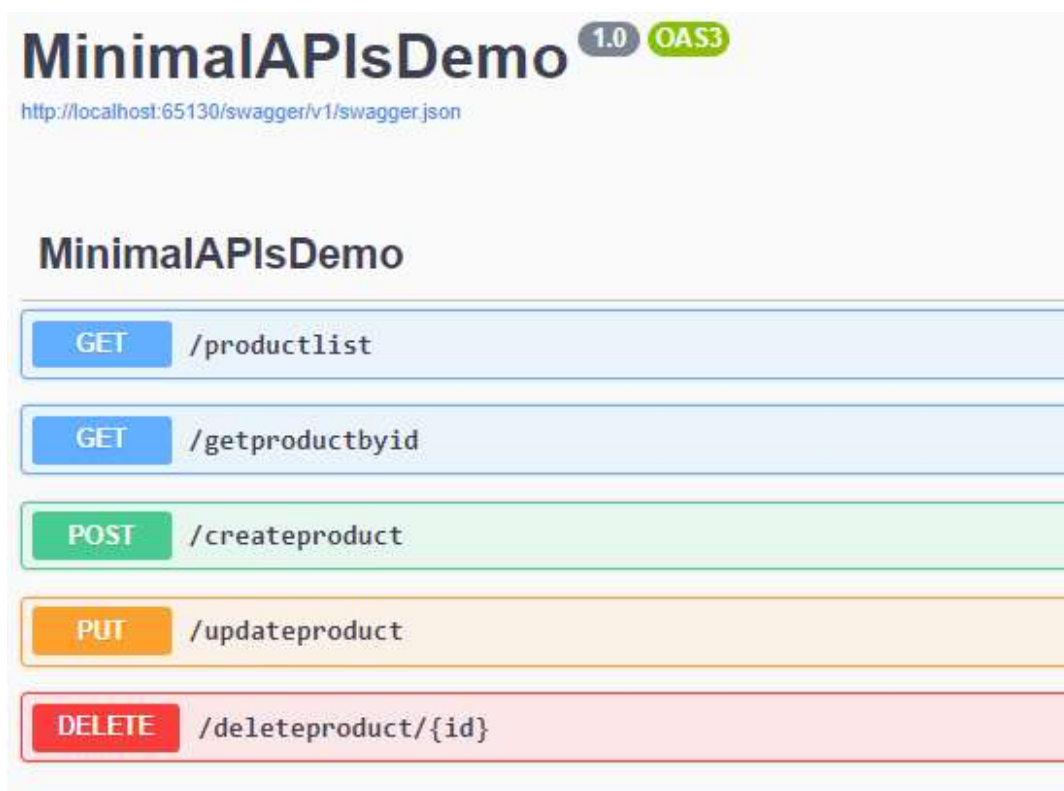
Paso 10

Ejecute el siguiente comando de entidad marco para crear la migración y actualizar la base de datos

```
1 add-migration "initial"  
2 update-database
```

Paso 11

Finalmente, ejecute su aplicación.



URL de GITHUB

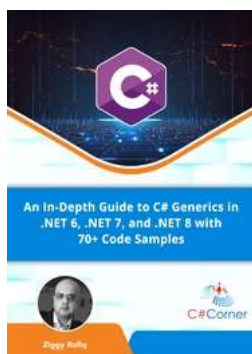
<https://github.com/Jaydeep-007/MinimalAPIsDemo>



Aquí analizamos las API mínimas y aspectos relacionados con el uso de la aplicación web .NET Core 6 y Entity Framework con la ayuda de SQL Server.

¡Feliz aprendizaje!

LIBRO ELECTRÓNICO GRATUITO RECOMENDADO



Una guía detallada sobre genéricos de C# en .NET 6, .NET 7 y .NET 8 con más de 70 ejemplos de código

[¡Descargar ahora!](#)

ARTÍCULOS SIMILARES

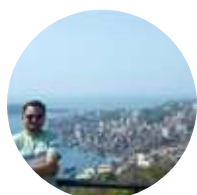
[Registro de API web de .NET Core utilizando NLog en RabbitMQ](#)

[API web ASP.NET Core 5.0](#)

[API web mínima con .Net 8](#)

[Introducción a la API mínima en .NET 7](#)

[Revelando la elegancia y eficiencia de las API mínimas](#)



Jaydeep Patil *TOP 500*

MVP de la esquina de C# | Desarrollador Fullstack con más de 2,5 años de experiencia en .NET Core API, Angular 8+, SQL Server, Docker, Kubernetes y Azure

106

1,8 m

2

[Ver todos los comentarios](#)

1



Type your comment here and press Enter Key (Minimum 10 characters)

[Sobre nosotros](#) [Contáctenos](#) [política de privacidad](#) [Términos](#) [Kit de medios](#) [Mapa del sitio](#)
[Reportar un error](#) [Preguntas más frecuentes](#) [Socios](#)

[Tutoriales de C#](#) [Preguntas comunes de la entrevista](#) [Cuentos](#) [Consultores](#) [Ideas](#) [Certificaciones](#)
[Televisión nítida](#)

[Universo Web3](#) [Construir con JavaScript](#) [Reaccionemos](#) [Charlas DB](#) [Impulsar la cadena de bloques](#)
[Entrevistas.ayuda](#)

©2024 C# Esquina. Todos los contenidos son propiedad intelectual de sus autores.