Integración de proyectos ASP.Net Core 2.0 y Angular 4 usando CLI y Visual Studio Code - Tutorial para principiantes

Contenido

Introducción

Los framework más recientes tienden a ser más modulares. ASP.Net Core es una reescritura de ASP.Net pero mucho más modular con Angular 4 (2) de AngularJS.

Estos framework modernos ofrecen herramientas CLI (interfaz de línea de comandos) que ofrecen a los desarrolladores más opciones para desarrollar aplicaciones de próxima generación. Las herramientas CLI brindan a los desarrolladores la libertad de trabajar desde cualquier sistema operativo. Es evidente que las herramientas GUI no funcionan en todos los principales sistemas operativos. El mejor ejemplo de esto es Visual Studio, que funciona solo en Windows y recientemente en Mac.

Una de las formas de crear un proyecto SPA (Aplicación de página única) hecha de Angular y ASP.NET Core Web API es usar Visual Studio Template. Aunque este enfoque es conveniente, usted depende de la plantilla que se crea y se actualiza por algunas personas que pueden no ser parte de los desarrolladores del framework.

Este manual es una guía de inicio para el proyecto que tenemos que realizar y para desarrolladores que escucharon y sintieron curiosidad acerca de la creciente popularidad de Angular y ASP.NET Core, pero no saben cómo comenzar.

Lo que vas a aprender

1. Cómo crear un proyecto angular usando CLI angular
2. Cómo crear una API ASP.Net Core Wep utilizando Dotnet Core CLI
3. Cómo integrar ambos proyectos
4. Cómo instalar, actualizar y usar las últimas CLI

Medio ambiente

Este tutorial es para Microsoft Windows pero puede usarse como guía para otros sistemas operativos, excepto para la parte de instalación.

- Microsoft Windows 10 versión 1703, 64 bit (creado en esta versión)

Pasos clave

Estos son los pasos clave a seguir:

1. Descargue e instale SDK
2. Descargue e instale Visual Studio Code
3. Actualizar npm
4. Instalar CLI angular
5. Verificar versiones
6. Crear proyecto angular (aquí es donde comienza el trabajo real)
7. Crear proyecto de API web ASP.Net Core
8. Integrar proyectos

Paso 1: descargue e instale SDK

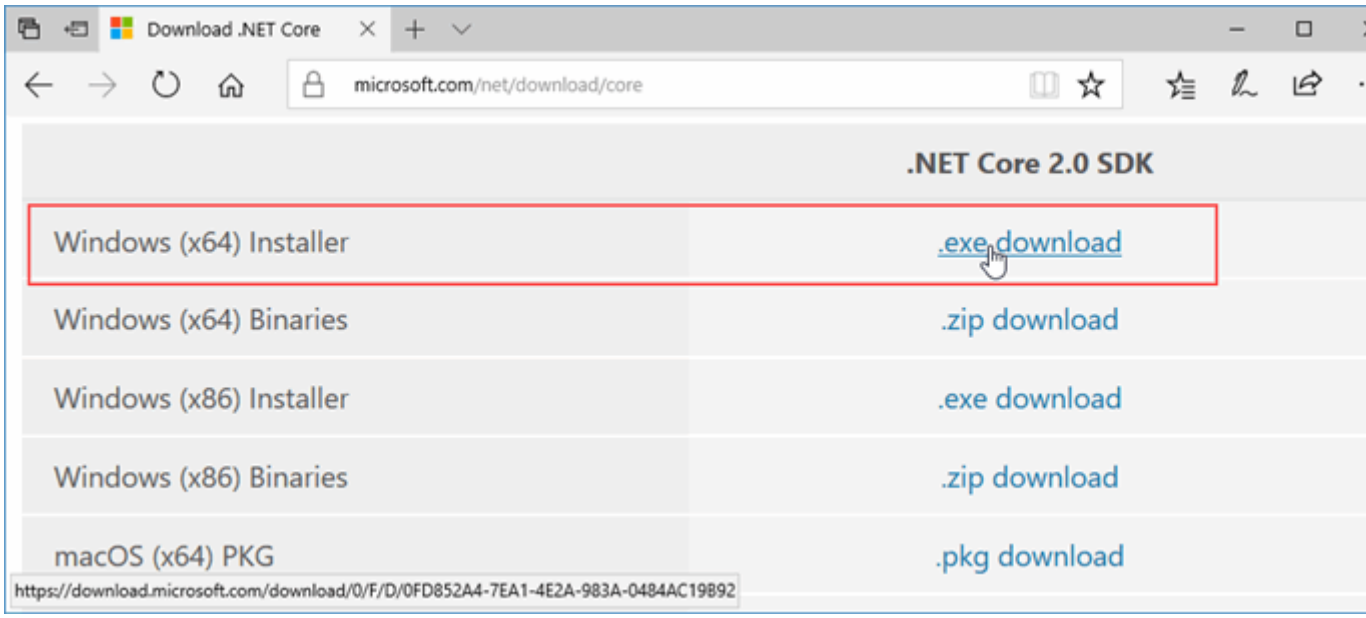
Tenga en cuenta que los procesos de instalación son diferentes en cada sistema operativo. La instalación paso a paso del SDK no está cubierta en este tutorial. Las capturas de pantalla son la primera ventana después de ejecutar el instalador para Windows.

Descargue e instale SDK:

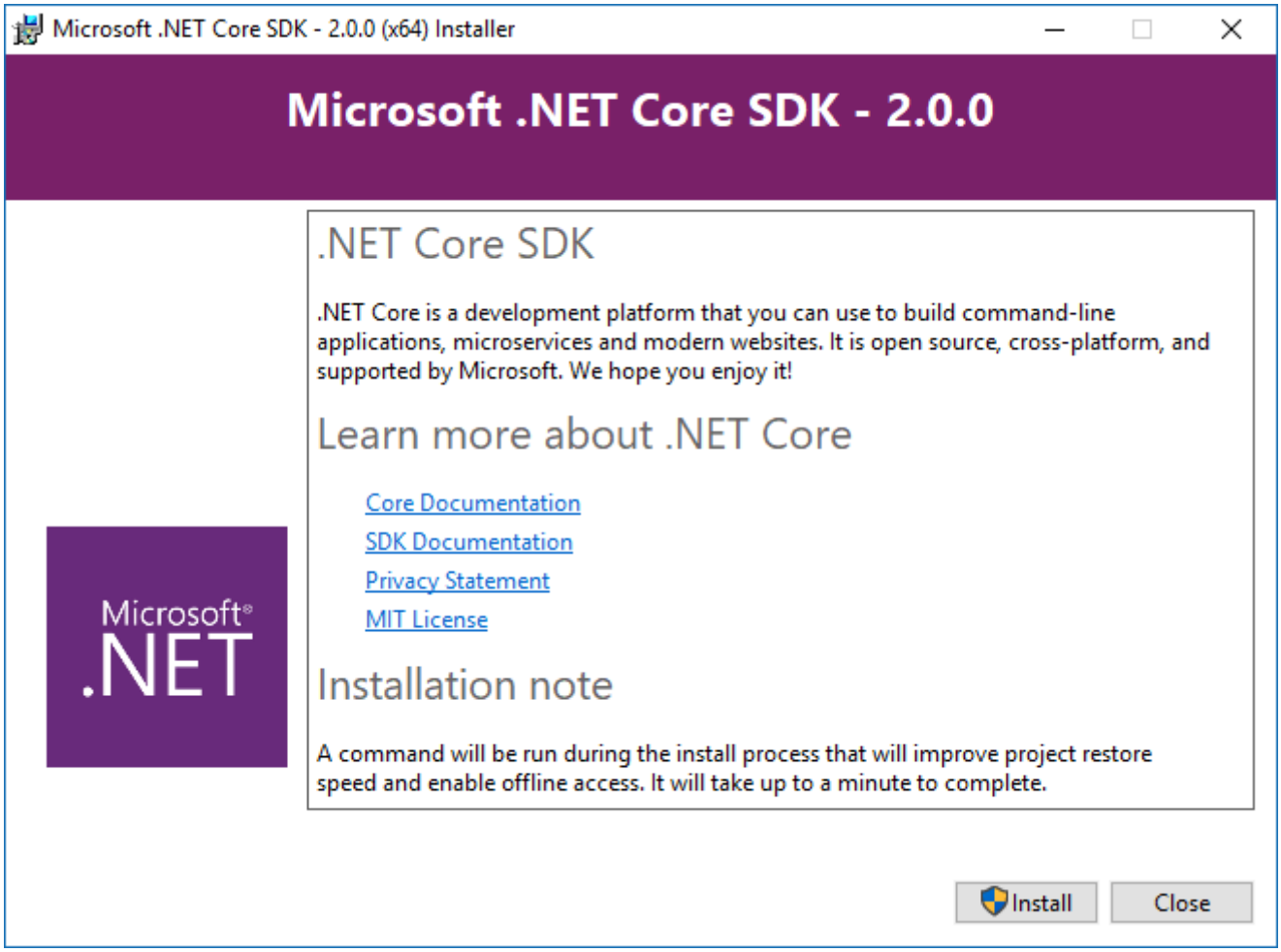
1. .NET Core 2.0 SDK
2. Node.js SDK

- **.Net Core 2.0**

  Descargue el instalador del SDK de .NET Core 2.0 .



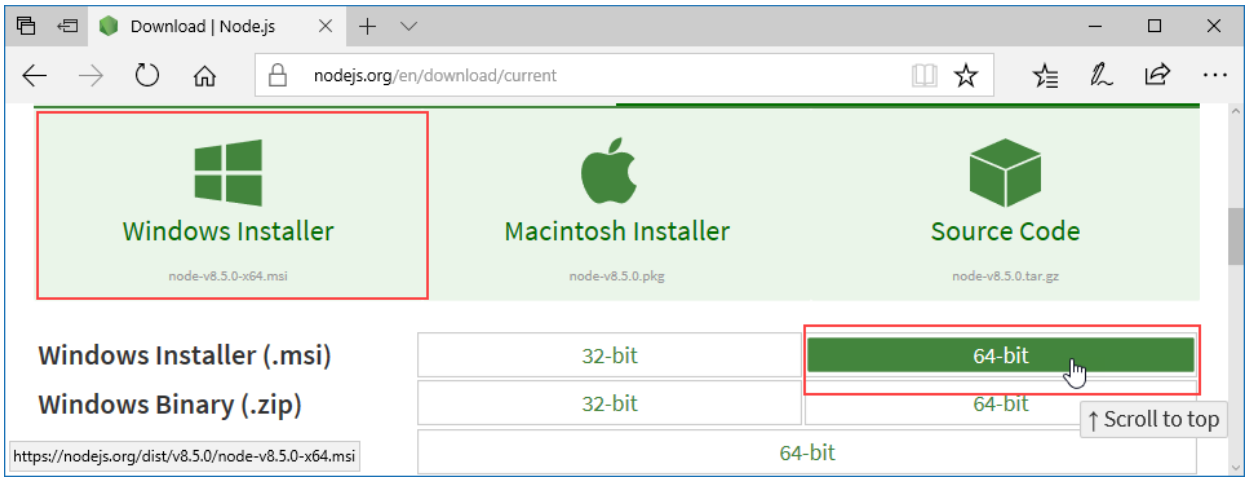
Instala el instalador descargado.



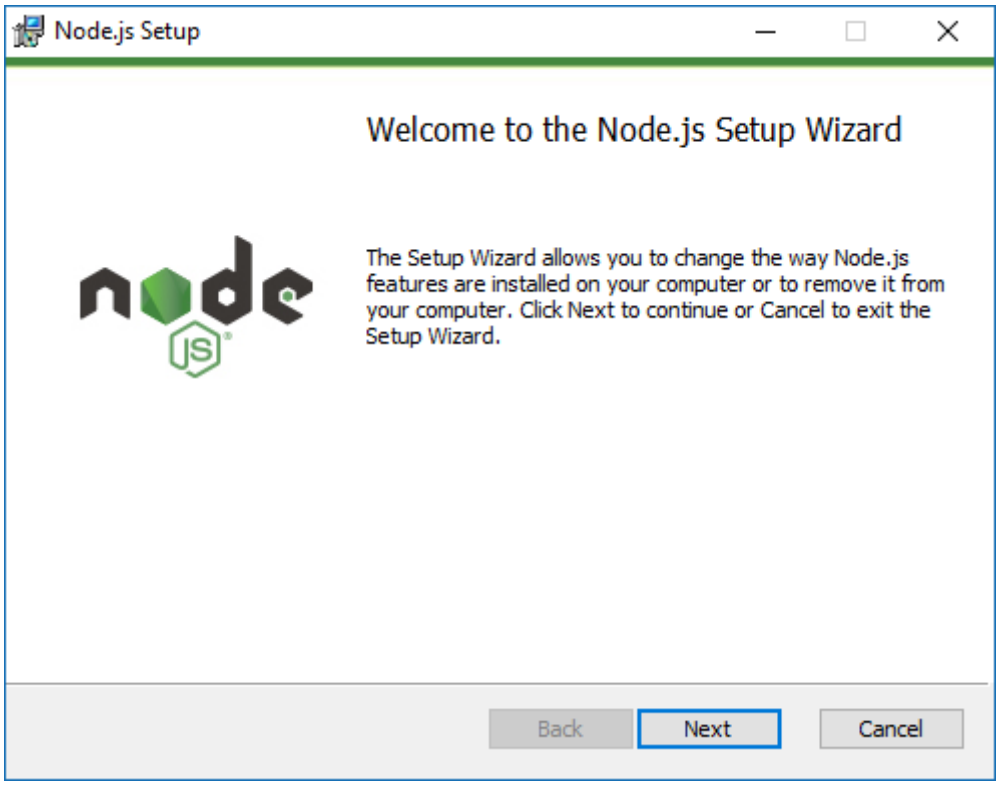
- **Node.js**

  Descargar Node.js SDK instalador .

Descargue la actual para conocer las últimas funciones.



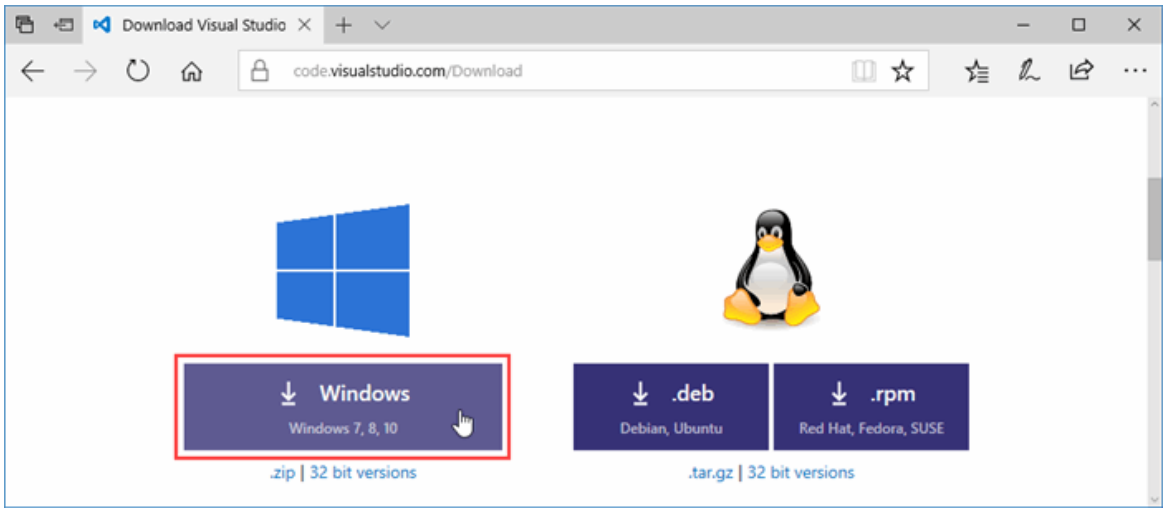
Instala el instalador descargado.



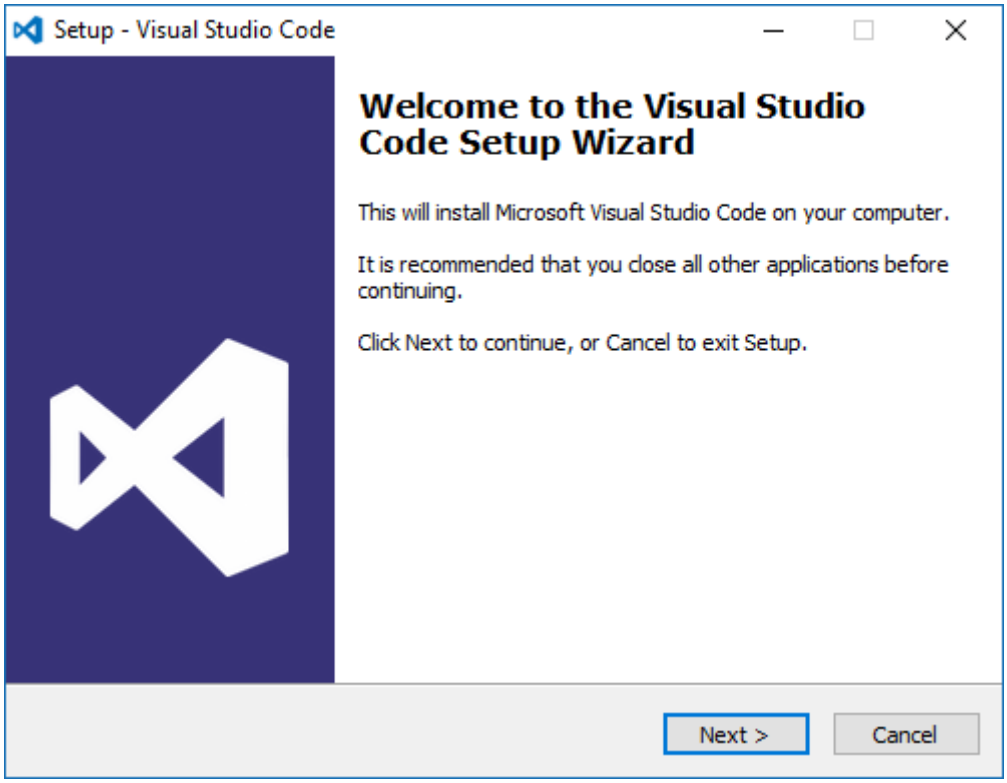
Paso 2: descargue e instale Visual Studio Code

Use Microsoft Visual Studio Code (VS Code) como herramienta GUI, IDE (Integrated Development Environment) para el desarrollo.

Descargue el instalador de Microsoft Visual Studio Code .



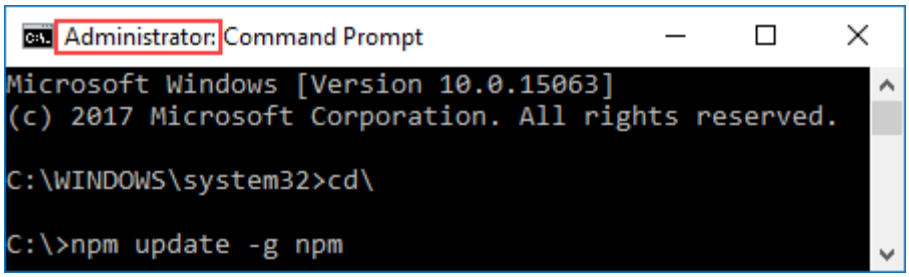
Instalar . Instala el instalador descargado.



Paso 3. Actualiza npm

El instalador de Node.js instala npm de forma predeterminada, pero tal vez no sea la última.

Actualice npm desde la línea de comandos como administrador. Siga la guía aquí sobre cómo ejecutar la línea de comandos (cmd.exe) como administrador.



No cierre la línea de comando, ya que se utilizará en los pasos siguientes.

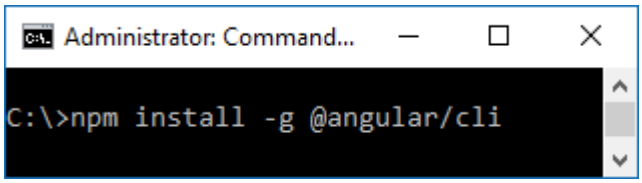
Explicación:

cd \
    - cambia el directorio de la ruta actual a la ruta raíz. En las capturas de pantalla anteriores, C: es la ruta raíz.

npm update -g npm
  : actualiza el Node Package Manager (NPM) instalado globalmente (-g)

Paso 4. Instalar CLI angular

Instale cli angular desde la línea de comando.



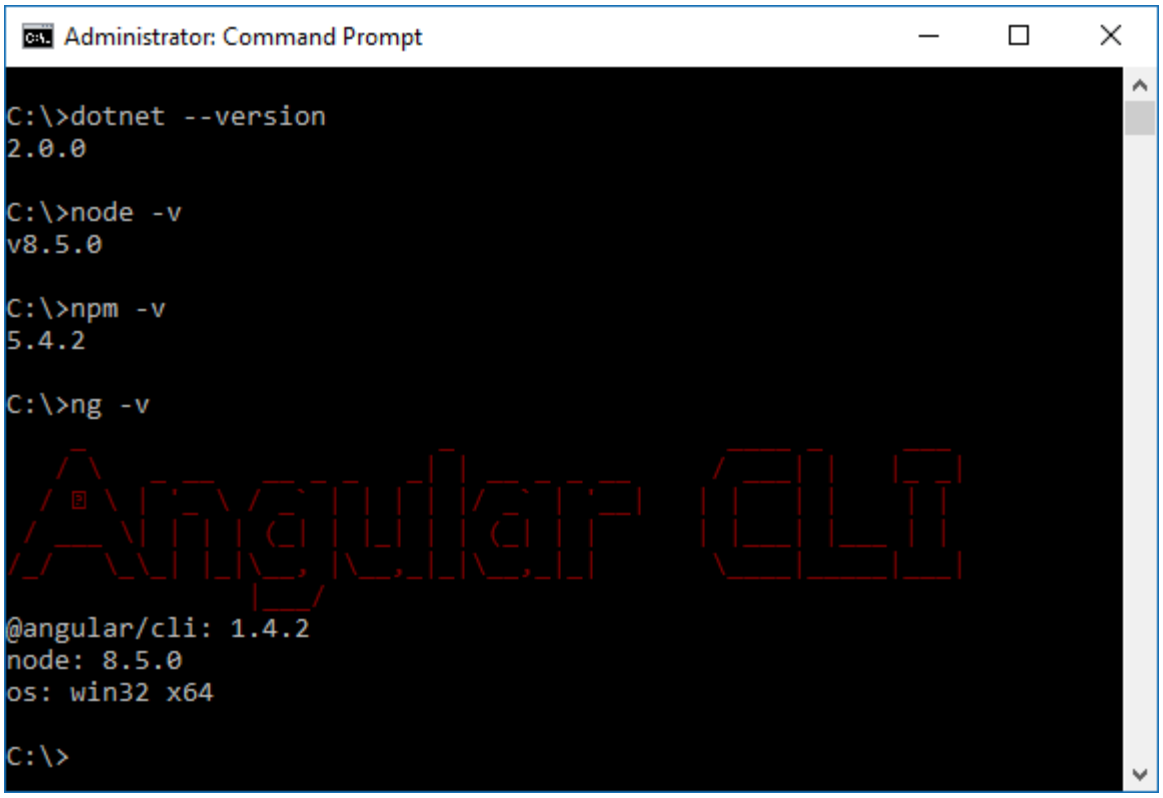
Explicación:

npm install -g @ angular / cli

  - usando npm, angular cli se instala globalmente.

Paso 5. Verifique las versiones (esto es opcional)

En este punto, verifique las versiones de SDK / CLI instaladas.



Explicación:

dotnet --version
- muestra la versión .NET Core SDK

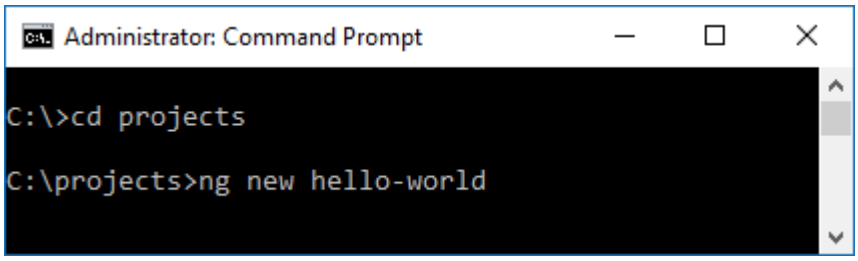
nodo -v
: muestra la versión Node.js

npm -v
: muestra la versión NPM

ng -v
: muestra la versión de CLI angular

Paso 6. Cree un proyecto angular (aquí es donde comienza el trabajo real)

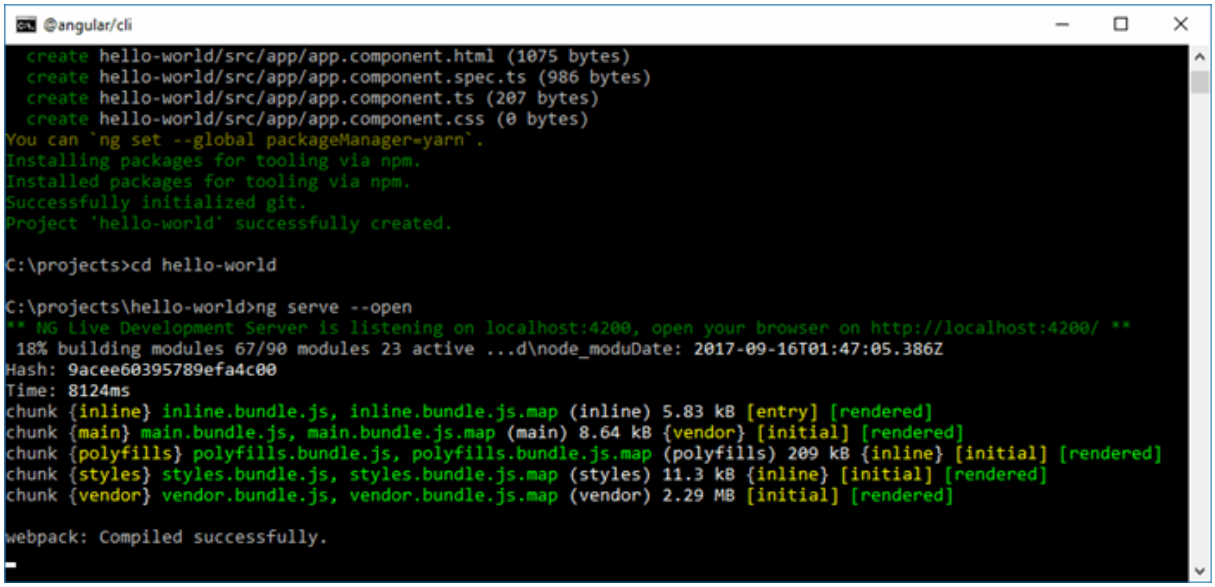
Cree un proyecto angular utilizando CLI angular (ng).



Explicación:

proyectos de cd
: cambie la ruta actual a la carpeta "projects" debajo de la ruta actual. Esto es para organizar proyectos en una carpeta
ng new hello-world
- usando Angular (ng) CLI, crear una carpeta "hello-world" bajo la ruta actual y agregar un nuevo proyecto angular con el mismo nombre
- emitiendo "ng new hello-word & cd new hello-world "desde la línea de comandos cambiará inmediatamente la ruta actual a" hello-world "después de crear el proyecto

Ejecute el proyecto angular recién creado.

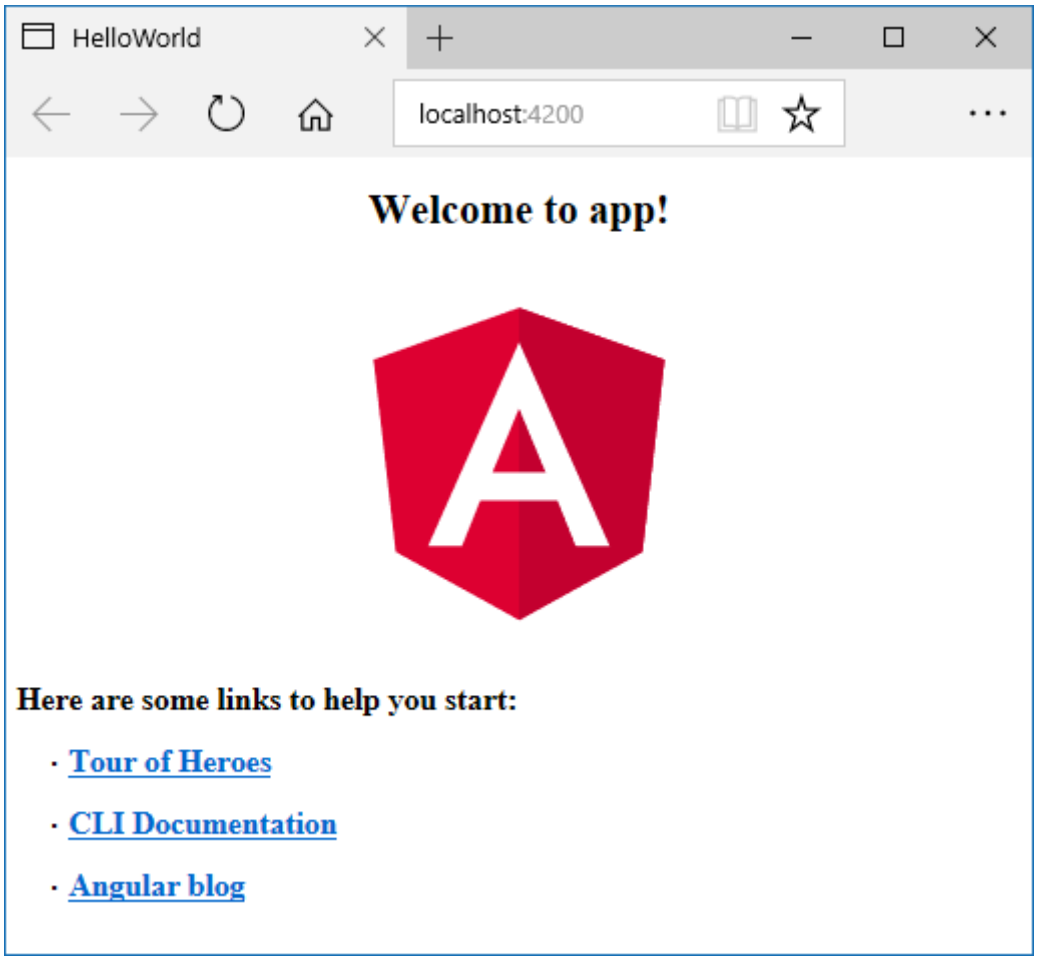


Explicación:

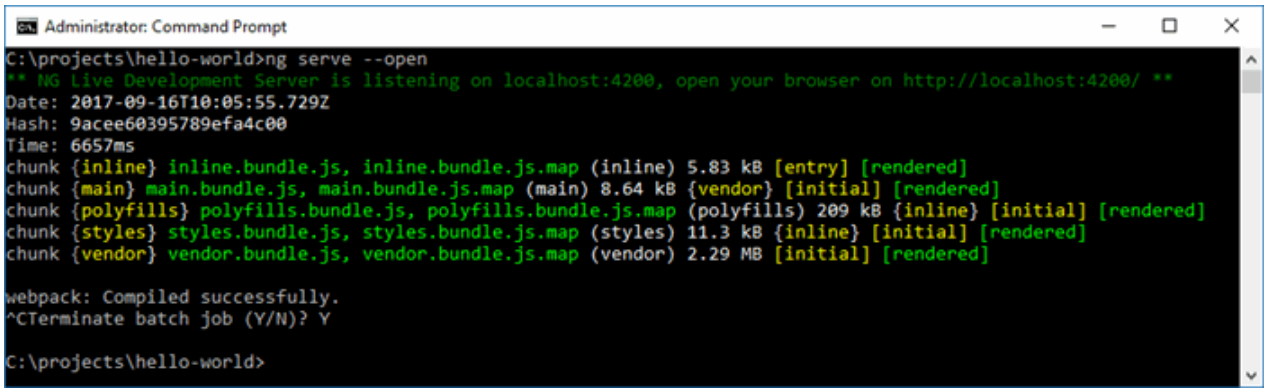
cd hello-world
- cambia la ruta actual a la carpeta "hello-world" debajo de la ruta actual

ng server --open
- "*ng serve*" construye la aplicación e inicia un servidor web
- "*--open*" abre el navegador web después de que la compilación y el servidor web se inician correctamente. Para otras opciones, consulte aquí . Sin usar la opción "*--open*" , puede ver el resultado abriendo el navegador web en "*http: // localhost: 4200*" como url.

La página web creada se mostrará en el navegador web.



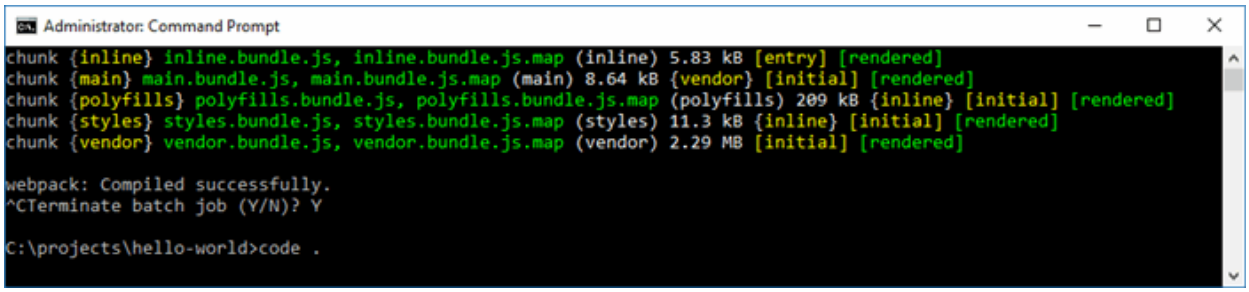
Termine el servidor web emitiendo "ctrt + c", luego presione "Y" y luego "Enter".



Paso 7. Crear un proyecto ASP.Net Core Web API

Aunque los procesos subsiguientes todavía se pueden ejecutar usando la línea de comando, esta vez se usará VS Code. De hecho, todos los comandos angulares realizados previamente en la línea de comandos también se pueden ejecutar dentro de VS Code. Prefiero trabajar dentro de VS Code para evitar cambiar entre ventanas abiertas.

Abra el proyecto en VS Code usando la línea de comando (1st Way)



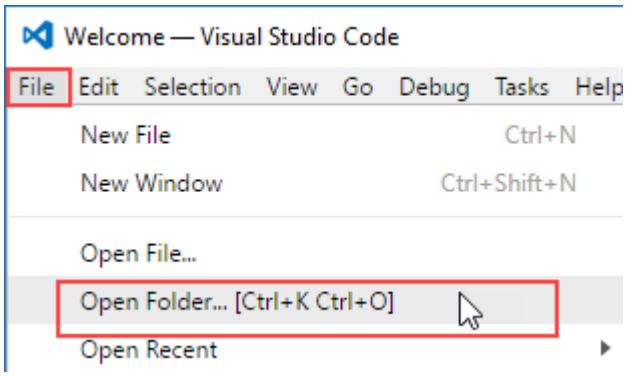
Explicación:

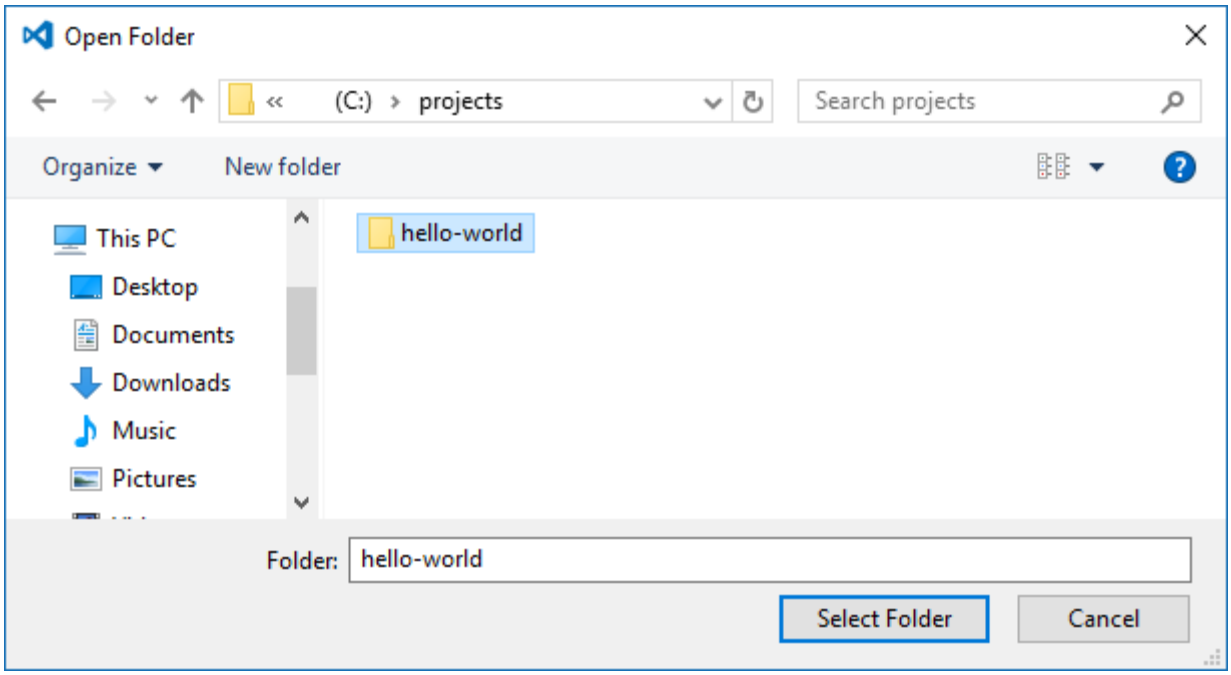
código

- abre la carpeta actual en VS Code

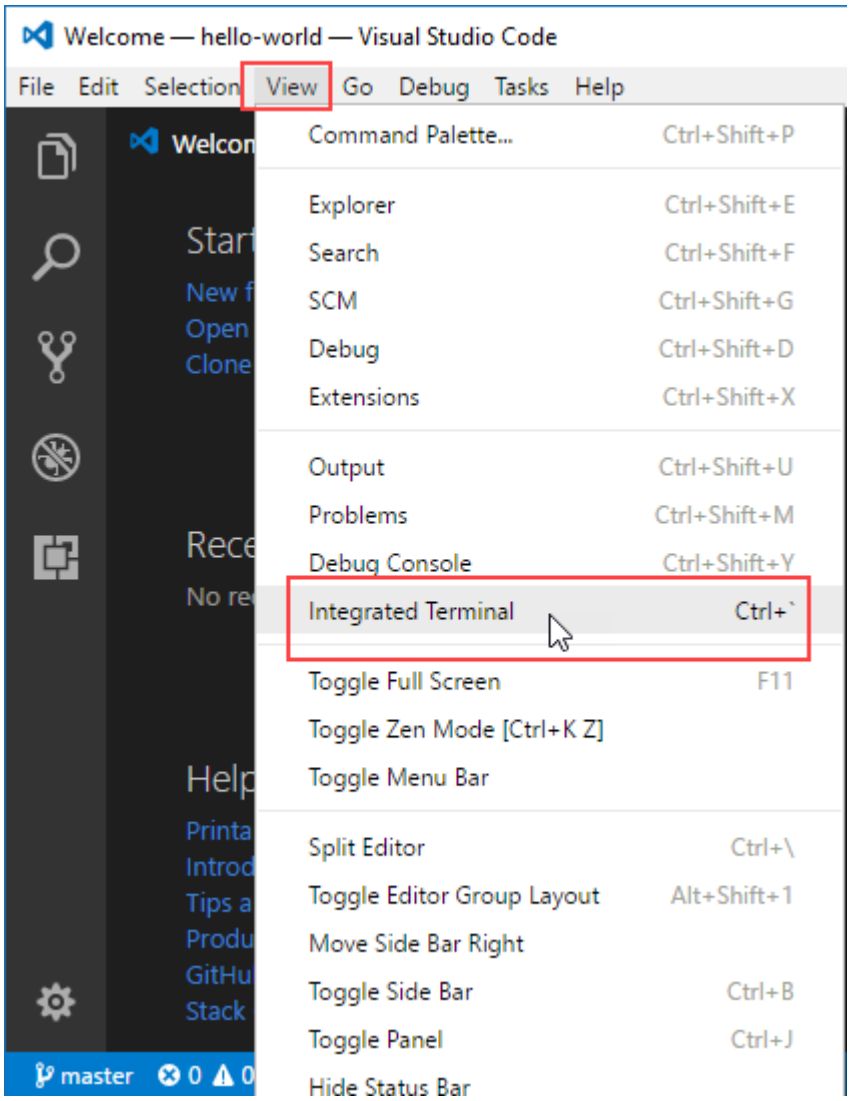
Proyecto abierto en VS Code (2nd Way)

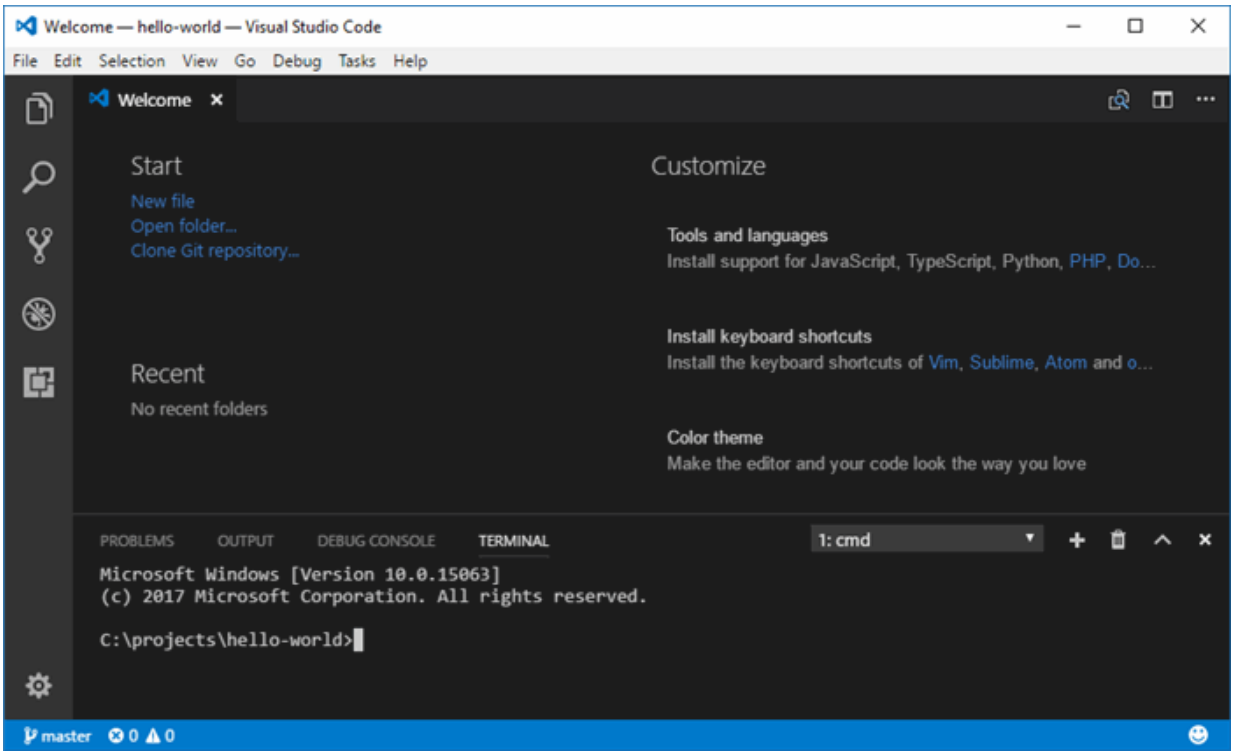
Después de ejecutar VS Code, seleccione la carpeta del proyecto haciendo clic en *"Archivo-Abrir carpeta ..."* en el menú.



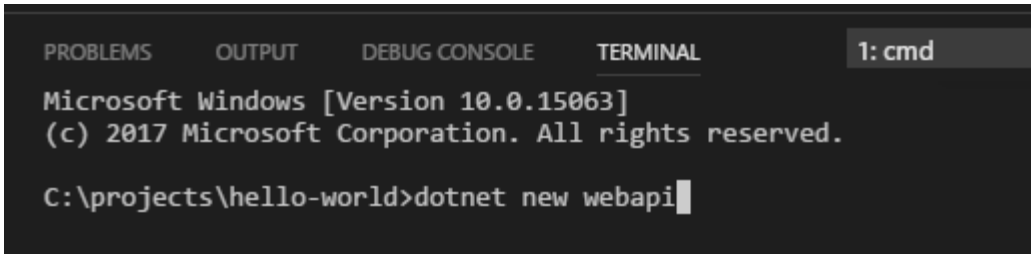


Muestre la terminal integrada (línea de comando) en el Código VS haciendo clic en "*Ver-Terminal integrada*" en el menú o "*Ctrl* + ~" como método abreviado de teclado. Esto explica por qué todo se puede ejecutar dentro de VS Code.


Welcome — hello-world — Visual Studio Code
File Edit Selection View Go Debug Tasks Help
Command Palette... Ctrl+Shift+P
Explorer Ctrl+Shift+E
Search Ctrl+Shift+F
SCM Ctrl+Shift+G
Debug Ctrl+Shift+D
Extensions Ctrl+Shift+X
Output Ctrl+Shift+U
Problems Ctrl+Shift+M
Debug Console Ctrl+Shift+Y
Integrated Terminal Ctrl+`
Toggle Full Screen F11
Toggle Zen Mode [Ctrl+K Z]
Toggle Menu Bar
Split Editor Ctrl+\
Toggle Editor Group Layout Alt+Shift+1
Move Side Bar Right
Toggle Side Bar Ctrl+B
Toggle Panel Ctrl+J
Hide Status Bar
master 0 0



Desde la terminal (línea de comando integrada), cree un nuevo proyecto ASP.NET Core Web API.

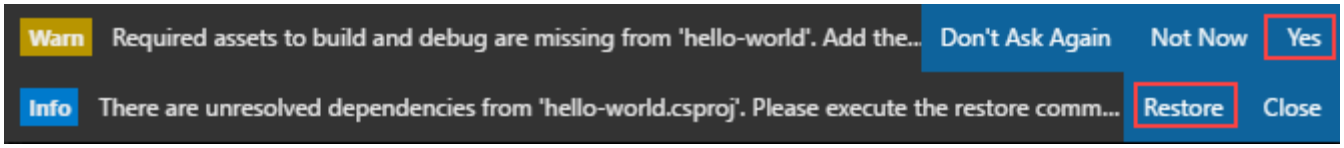


Explicación:

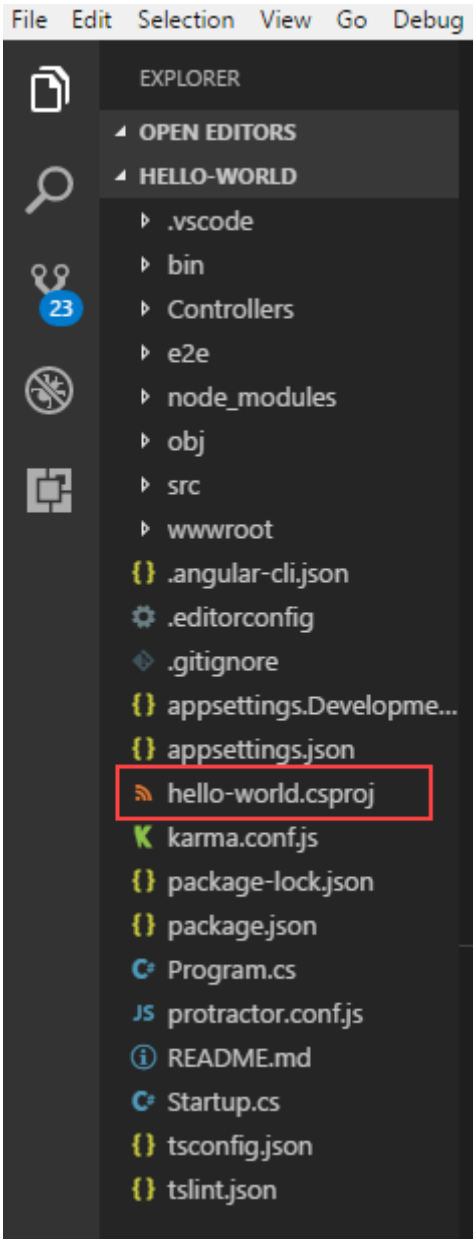
dotnet new webapi

- utilizando DotNet CLI, cree un nuevo proyecto de API web. Sin especificar opciones, Dotnet CLI creará un proyecto con un nombre igual a la carpeta donde se emite el comando "new". Para obtener información adicional, consulte este enlace .

Seleccione "*Sí*" y "*Restaurar*" en los mensajes mostrados por VS Code. La configuración se agregará para habilitar la depuración después de hacer clic en "*Sí*". Solo tenga en cuenta que la depuración se incluye en este tutorial, por lo que no importa la opción que elija. Al hacer clic en "*Restaurar*" se restaurarán las dependencias o bibliotecas necesarias para ASP.NET Core Web API.

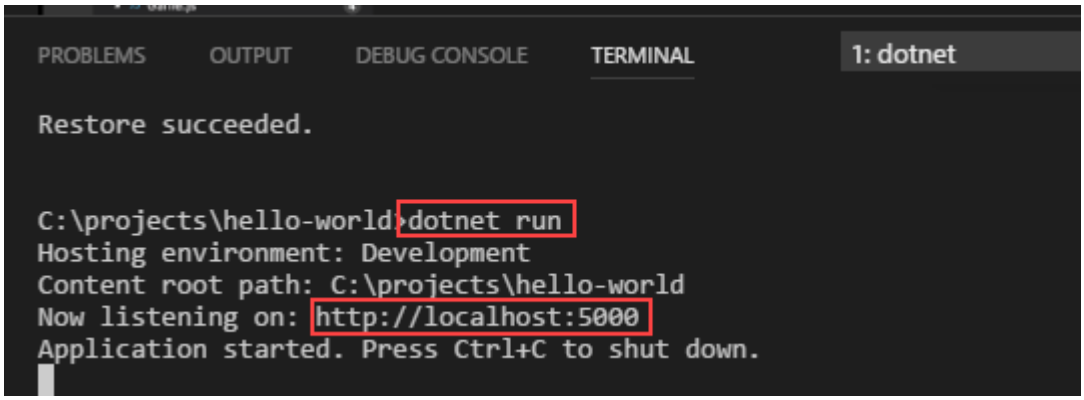


Se ha agregado el aviso en la barra lateral "*hello-world.csproj*" . Ese es el nuevo proyecto de API web ASP.NET Core. Si no se muestra la barra lateral, haga clic en "*Ver-alternar barra lateral*" del menú para mostrarla / ocultarla o "Ctrl + B" si prefiere el atajo de teclado.

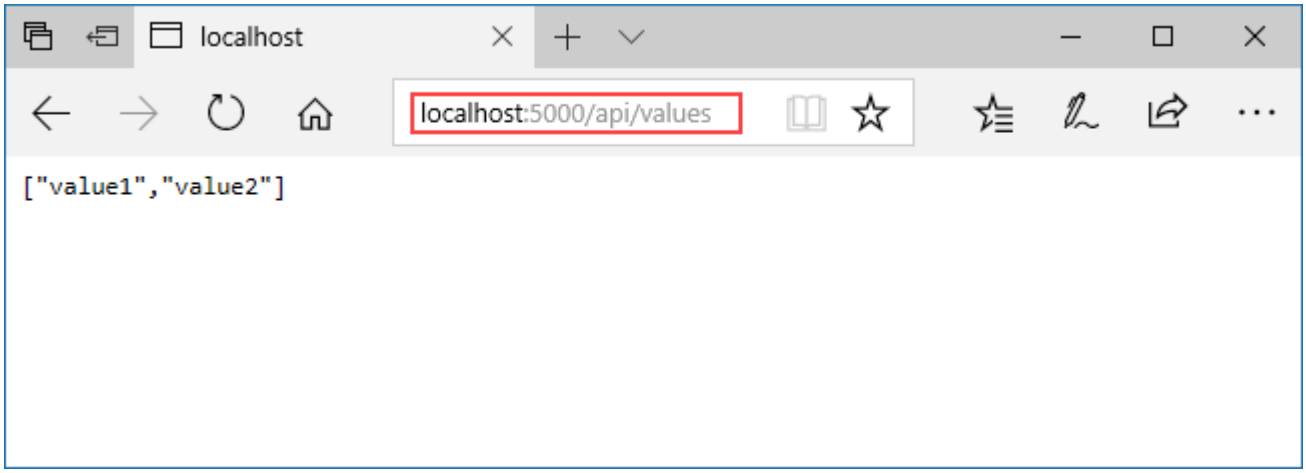


En este punto, existen dos proyectos: el proyecto Angular existente y el proyecto ASP.NET Core Web API recientemente agregado. Aunque los proyectos están en la misma carpeta, no tiene relación en este momento.

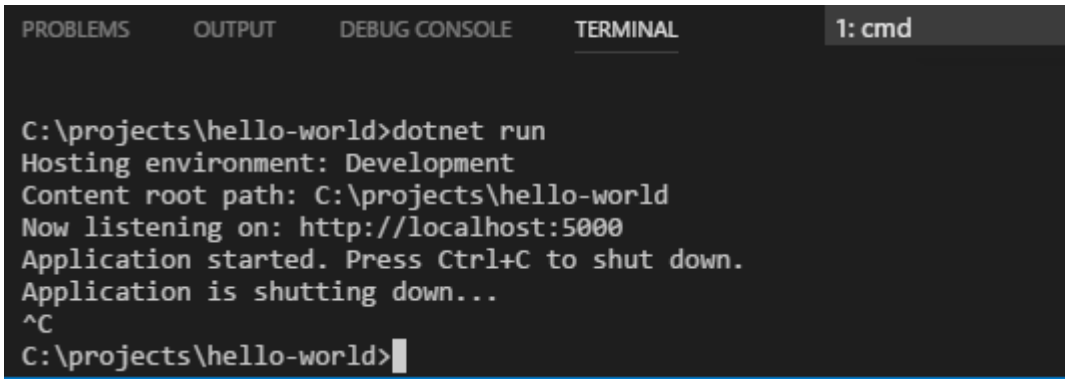
Compile y ejecute el proyecto ASP.NET Core Web API recién creado.



Abra el navegador web en *"http: // localhost: 5000* ***/ api / values*** " como url. Asegúrese de agregar **"/ api / values"** a la url porque esa es la ruta predeterminada para la API web recién creada.

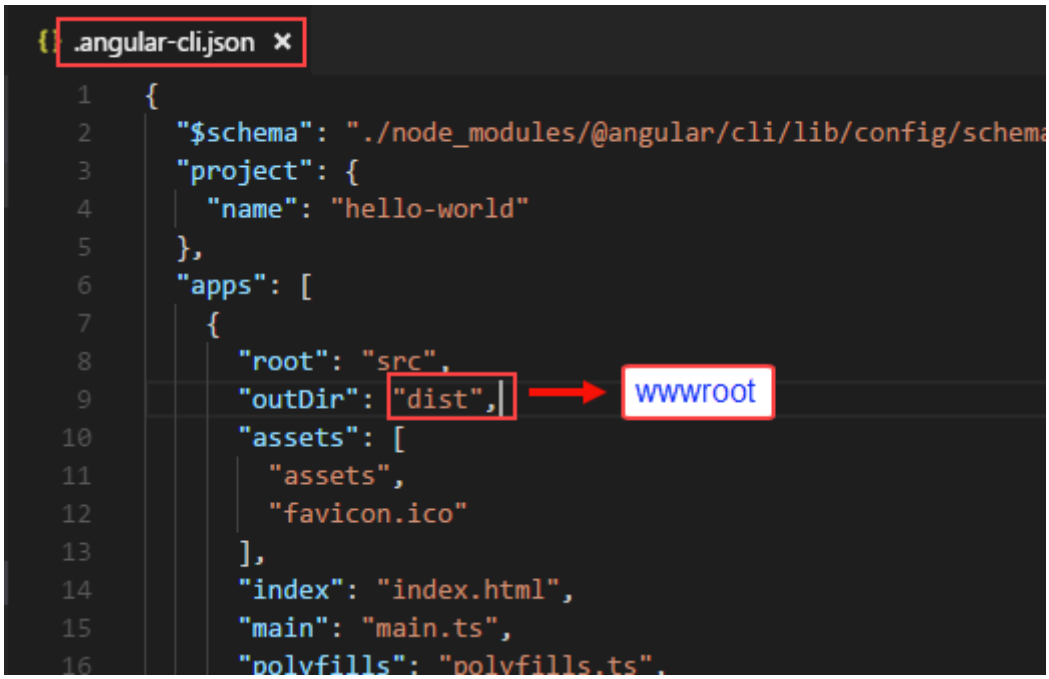


Termine el servidor web presionando "Ctrl + C" en el terminal integrado.

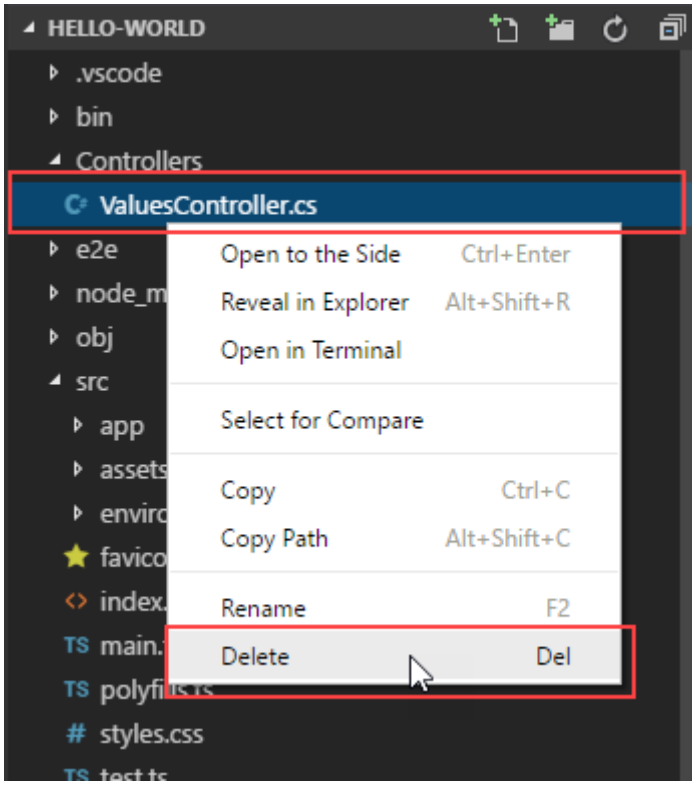


Paso 8. Integrar proyectos

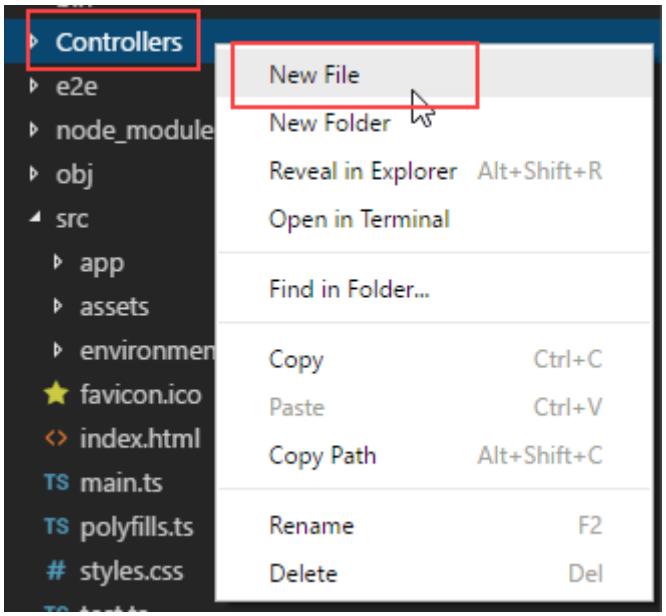
Abra *".angular-cli.json"* y cambie en *"outDir"* que esta el la sección "apps" a *"wwwroot"* . Esto le dice a Angular CLI que guarde los archivos de salida de la compilación en la misma carpeta donde está saliendo la CLI de DotNet.

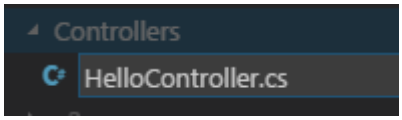


Elimine "*ValuesController.cs*" de la carpeta "*Controllers*" y *reemplácelo* con una API web simple que solo devuelva un simple saludo "hola".



Cree "*HelloController.cs*" dentro de la carpeta "*Controllers*" . Haga clic derecho en la carpeta "*Controlers*" y haga clic en "*Nuevo archivo*" . Nombre el archivo a "*HelloContoller.cs*".



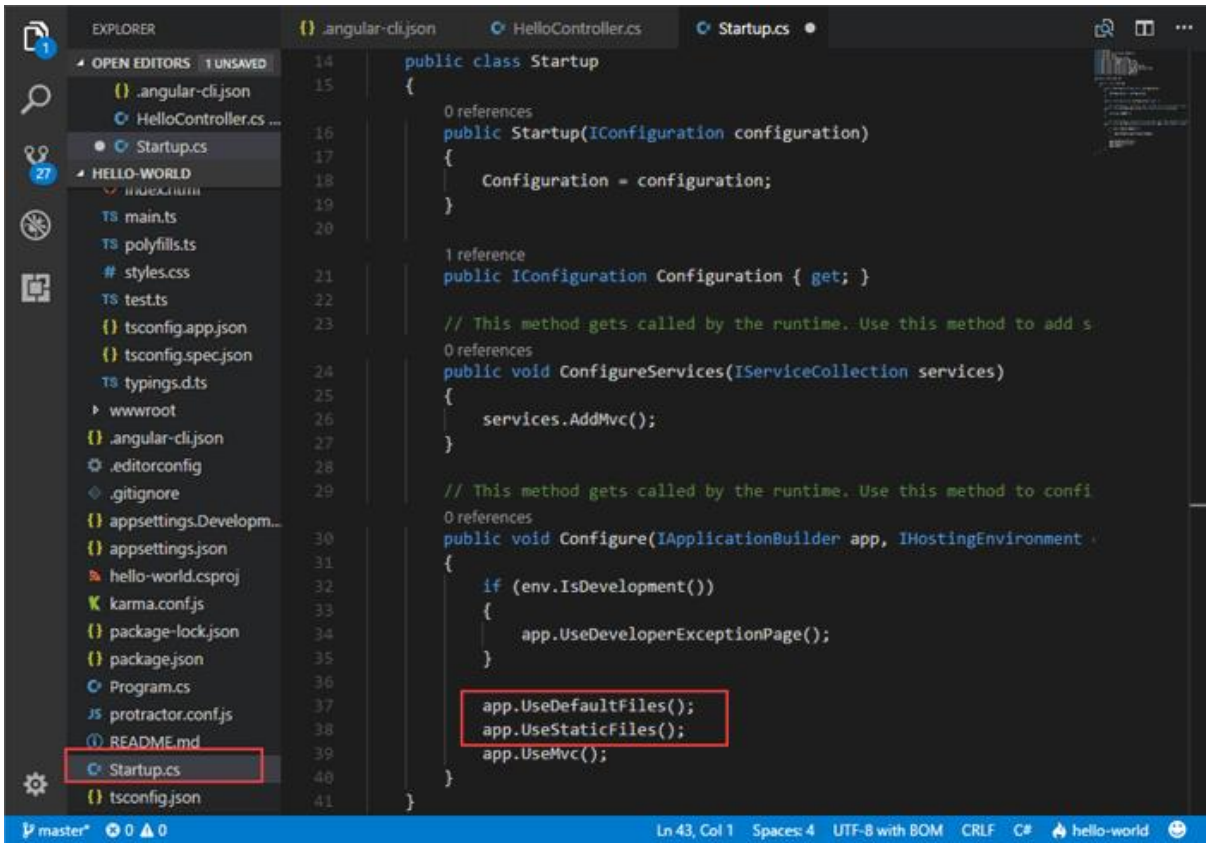


Pega el código a continuación.

[Controllers \ HelloController.cs]

```
1. using System;
2. using Microsoft.AspNetCore.Mvc;
3.
4. namespace hello_world
5. {
6.     [Route("api/[Controller]")]
7.     public class HelloController : Controller
8.     {
9.         [HttpGet]
10.             public IActionResult Greetings() {
11.                 return Ok("Hello from ASP.NET Core Web API.");
12.             }
13.         }
14.     }
```

Modifique Startup.cs para incluir "*UseDefaultFiles*" *y* "*UseStaticFiles*" en la canalización de solicitudes HTTP.



Con "*UseDefaultFiles*" , las solicitudes a una carpeta buscarán:

- default.htm
- default.html
- index.htm
- index.html

"*UseStaticFiles*" hace que los archivos en la raíz web (*wwwroot* por defecto) sean servibles.

Para obtener información adicional sobre *UseDefaultFiles "y" UseStaticFiles* " , consulte este enlace .

En este momento, nuestro trabajo en el proyecto de API web se ha completado. Volvamos a Angular y consumamos API web de Angular.

Cree un nuevo archivo "*app.service.ts*" dentro de la carpeta "*src \ app*" y pegue el código a continuación.

[src \ app \ app.service.ts]

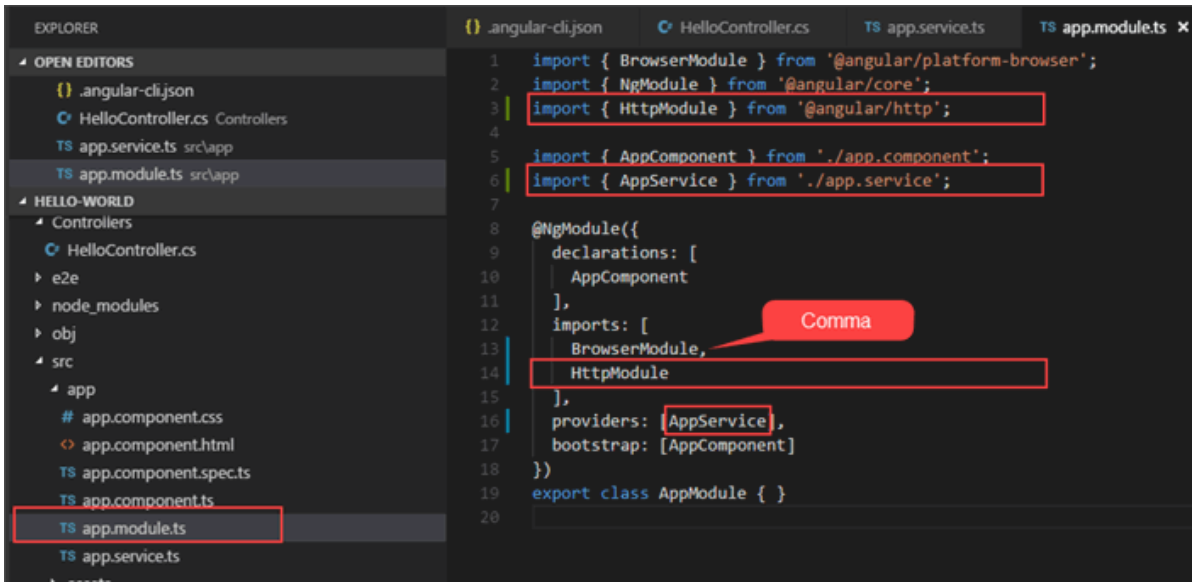
```
1. import { Injectable } from '@angular/core';
2. import { Http, Response, Headers, RequestOptions } from
   '@angular/http';
```

```
3. import { Observable } from 'rxjs/Rx';
4.
5. // Import RxJs required methods
6. import 'rxjs/add/operator/map';
7. import 'rxjs/add/operator/catch';
8.
9. @Injectable()
10.      export class AppService {
11.          private greetUrl = 'api/Hello';
12.
13.          // Resolve HTTP using the constructor
14.          constructor(private _http: Http) { }
15.
16.          sayHello(): Observable<any> {
17.              return
   this._http.get(this.greetUrl).map((response: Response) => {
18.                  return response.text();
19.              });
20.          }
21.      }
```

Modifique "*app.module.ts*" y agregue "*AppService*" como proveedores. Luego importe "*HttpModule*" .

[src \ app \ app.module.ts]

```
1. import { BrowserModule } from '@angular/platform-browser';
2. import { NgModule } from '@angular/core';
3. import { HttpModule } from '@angular/http';
4.
5. import { AppComponent } from './app.component';
6. import { AppService } from './app.service';
7.
8. @NgModule({
9.   declarations: [
10.          AppComponent
11.        ],
12.        imports: [
13.          BrowserModule,
14.          HttpModule
15.        ],
16.        providers: [AppService],
17.        bootstrap: [AppComponent]
18.      })
19.      export class AppModule { }
```



Modifica el "*app.component.ts*" y agrega el servicio "*AppService*".

[src\app\app.component.ts]

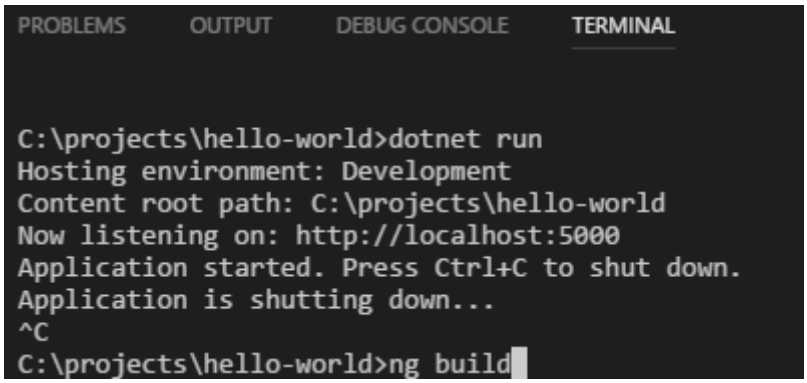
```
1. import { Component, OnInit } from '@angular/core';
2. import { Http, Response } from '@angular/http';
3.
4. import { AppService } from './app.service';
5.
6. @Component({
7.   selector: 'app-root',
8.   templateUrl: './app.component.html',
9.   styleUrls: ['./app.component.css']
10.      })
11.      export class AppComponent implements OnInit {
12.        greetings = '';
13.
14.        constructor(private _appService: AppService) { }
15.
16.        ngOnInit(): void {
17.          this._appService.sayHello()
18.            .subscribe(
19.            result => {
20.              this.greetings = result;
21.            }
22.            );
23.        }
24.      }
```

Cambia en contenido de "*app.component.html*".

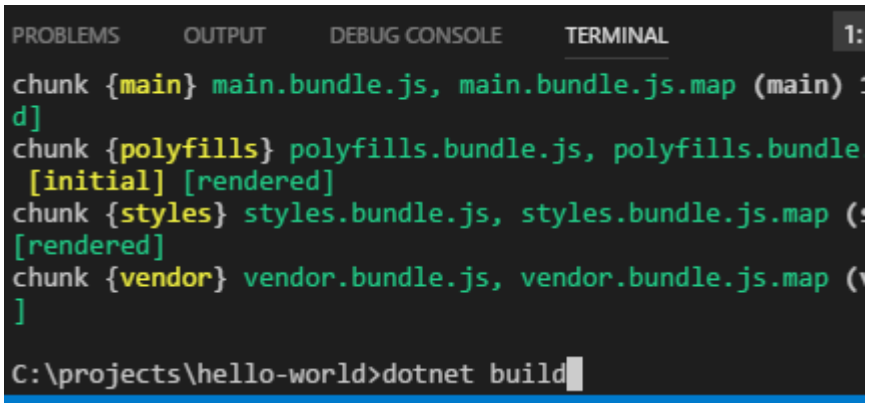
[src\app\app.component.html]

```
<h2 class='panel-heading'>
  {{greetings}}
</h2>
<hr/>
```

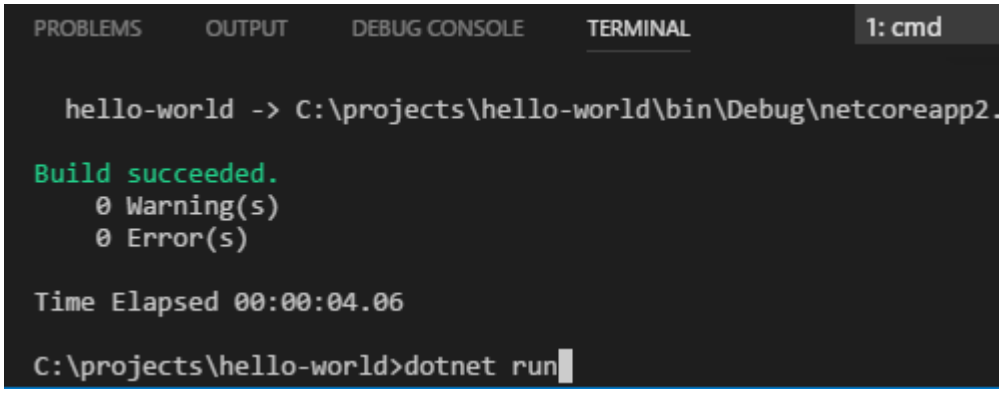
Contruye Angular el .NET



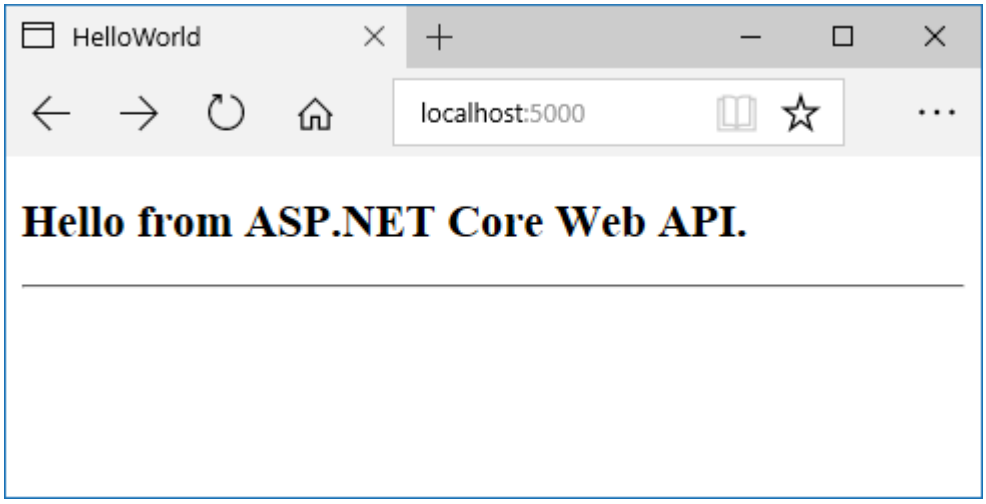
Construye DotNet.



Ejecuta DotNet.



Abra el navegador web en "*http: // localhost: 5000 /*" . Tenga en cuenta que "*api / **" no está incluido en la url. Cuando se carga la página, Angular se carga primero y luego se consume la API web dentro de Angular, mostrando así los saludos.



Configurado nuestro proyecto.