



CORS

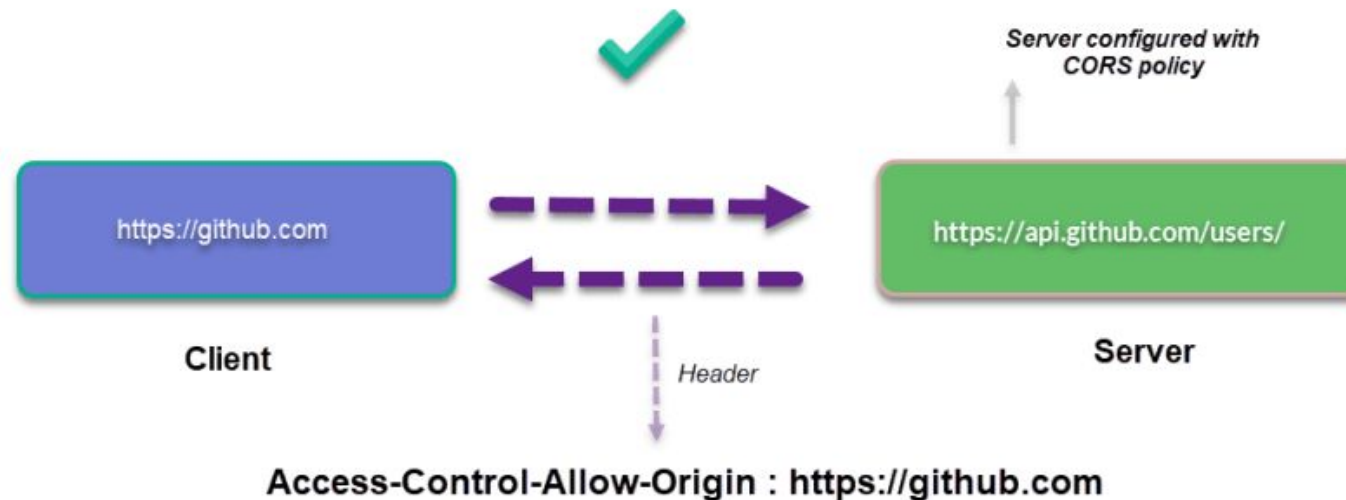
Intercambio de Recursos de Origen Cruzado

¿Qué es CORS?

CORS (Cross Origin Resource Sharing, o bien en español Intercambio de Recursos de Origen Cruzado) es un mecanismo que permite solicitar recursos restringidos desde una página web de un dominio determinado a otro recurso web de otro dominio. De esta manera, CORS define una manera en la que el navegador y el servidor puedan interactuar para determinar si la petición de origen cruzado es segura.

¿Cómo funciona CORS?

CORS trabaja agregando cabeceras HTTP que permiten a los servidores describir un conjunto de orígenes (dominios) que tienen permisos para obtener una información usando el navegador. Adicionalmente la especificación de estos, sugiere que los navegadores verifiquen la solicitud solicitando métodos soportados desde el servidor con un método de solicitud HTTP OPTIONS, luego con la aprobación del servidor se envía la verdadera solicitud con el método HTTP.



¿Cómo habilitar CORS?

1- Agregar el paquete NuGet CORS. Para ello, ir a **Herramientas -> Administrar paquetes de Nuget -> Consola del Administrador de Paquetes.**

Luego en la consola escribir el siguiente comando: **Install-Package**

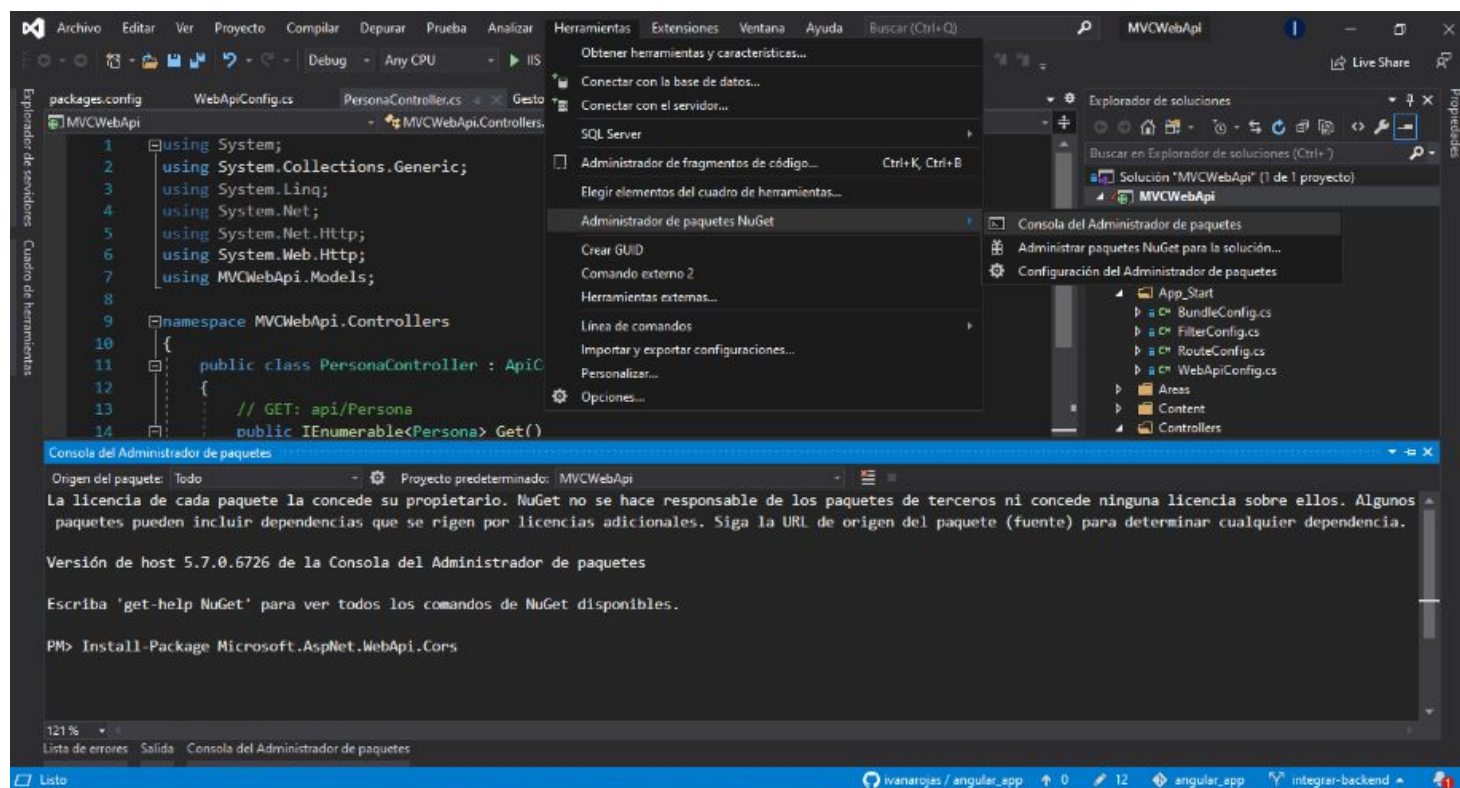
Microsoft.AspNet.WebApi.Cors

Al finalizar, se habrán actualizado los archivos:

-package.config

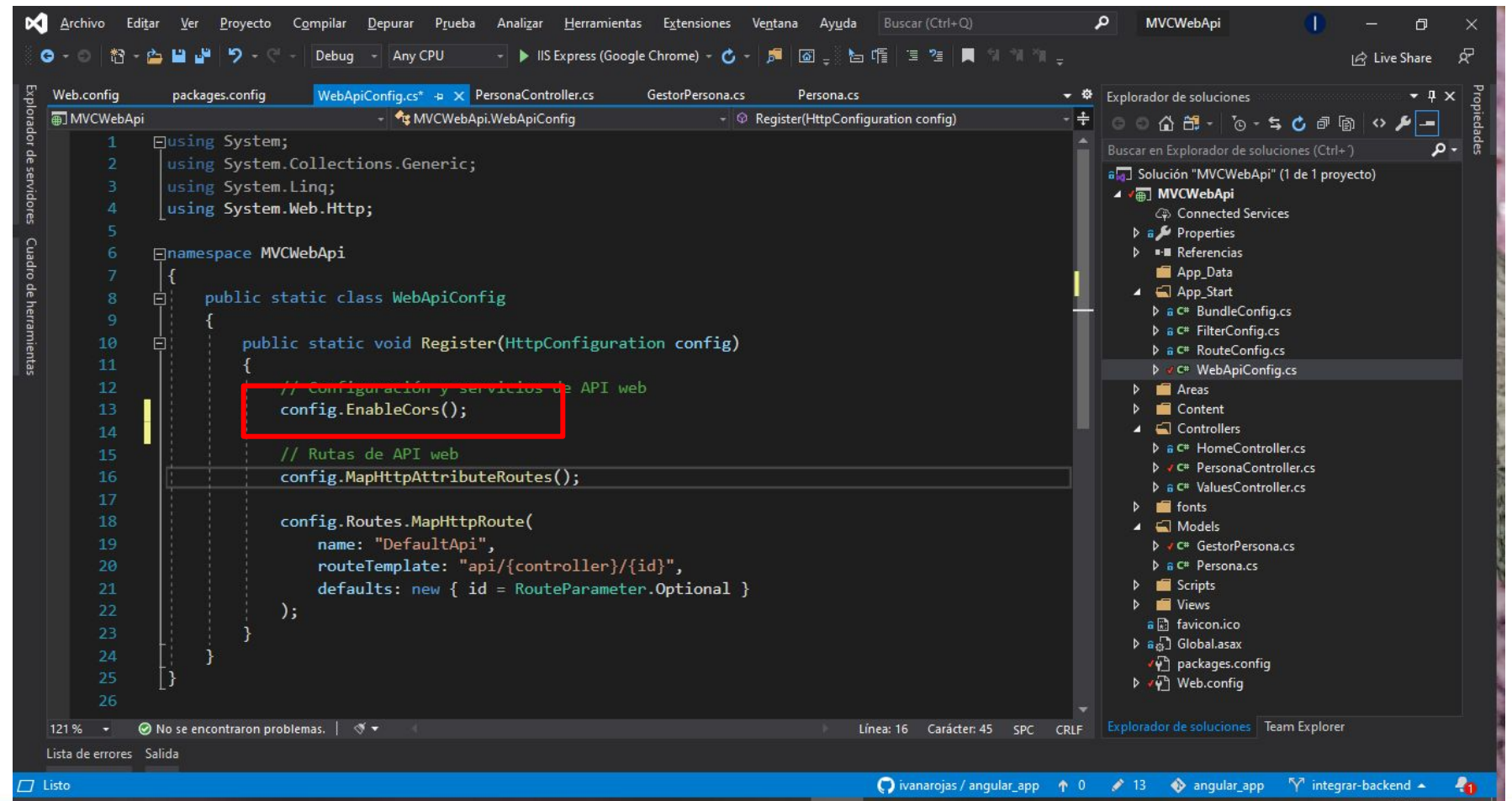
-web.config

y la solución de visual studio.



¿Cómo habilitar CORS?

2- Abrir el archivo **App_Start/WebApiConfig.cs** y agregar el siguiente código al método **WebApiConfig.Register**:



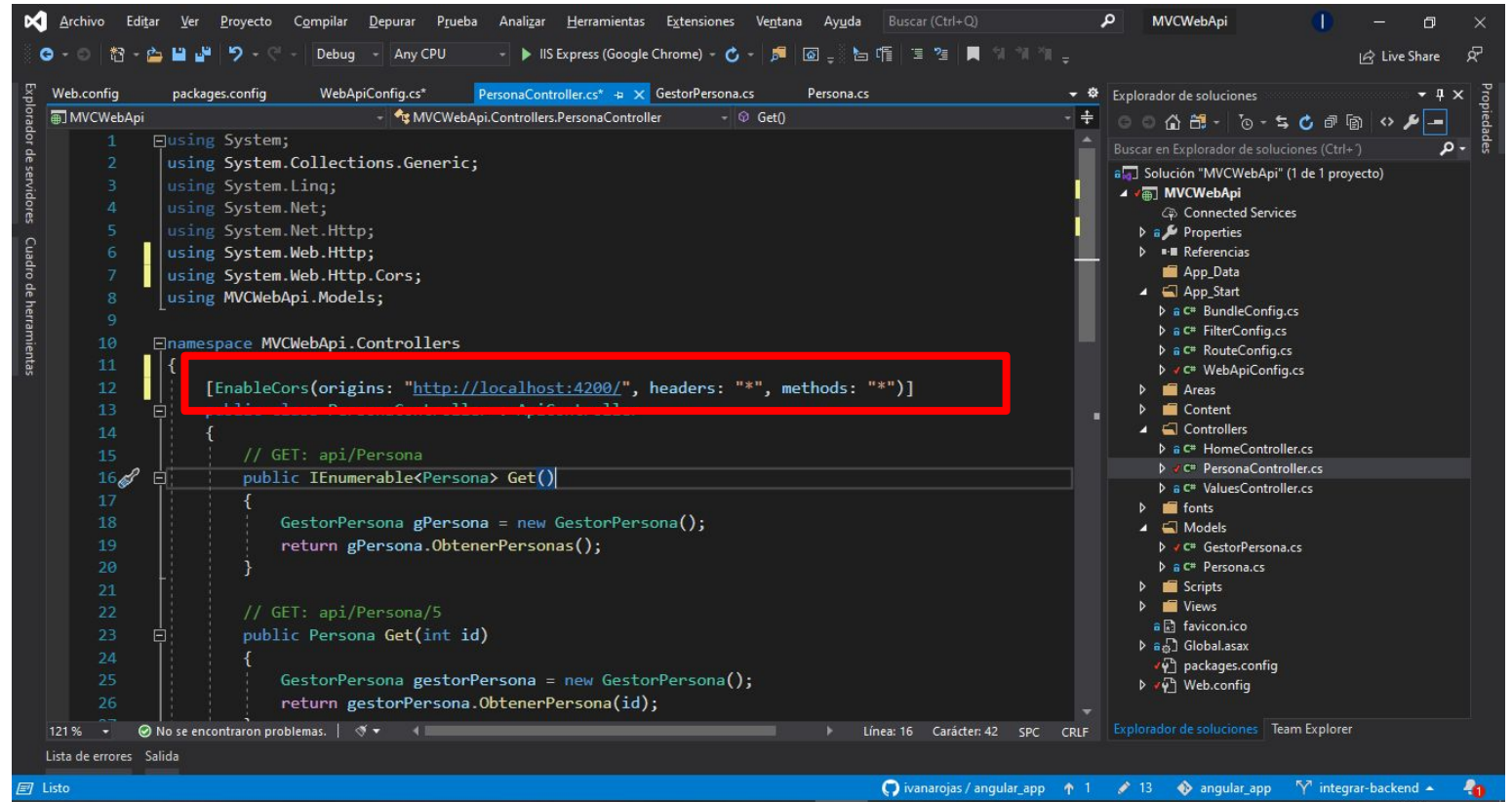
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web.Http;
5
6 namespace MVCWebApi
7 {
8     public static class WebApiConfig
9     {
10         public static void Register(HttpConfiguration config)
11         {
12             // Configuración y servicios de API web
13             config.EnableCors();
14
15             // Rutas de API web
16             config.MapHttpAttributeRoutes();
17
18             config.Routes.MapHttpRoute(
19                 name: "DefaultApi",
20                 routeTemplate: "api/{controller}/{id}",
21                 defaults: new { id = RouteParameter.Optional }
22             );
23         }
24     }
25 }
```

The screenshot shows the Visual Studio IDE with the **WebApiConfig.cs** file open. The **Register** method is visible, and the line **config.EnableCors();** is highlighted with a red rectangle. The Solution Explorer on the right shows the project structure, including the **App_Start** folder where **WebApiConfig.cs** is located. The status bar at the bottom indicates "No se encontraron problemas." (No problems found).

¿Cómo habilitar CORS?

3- Finalmente, agrega el atributo **[EnableCors]** a la clase controller.

Los parámetros del atributo **[EnableCors]** especifican cuáles métodos HTTP están permitidos para acceder al recurso. Para permitir todos los métodos usar "*".



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Net;
5  using System.Net.Http;
6  using System.Web.Http;
7  using System.Web.Http.Cors;
8  using MVCWebApi.Models;
9
10 namespace MVCWebApi.Controllers
11 {
12     [EnableCors(origins: "http://localhost:4200/", headers: "*", methods: "*")]
13     public class PersonaController : ApiController
14     {
15         // GET: api/Persona
16         public IEnumerable<Persona> Get()
17         {
18             GestorPersona gPersona = new GestorPersona();
19             return gPersona.ObtenerPersonas();
20         }
21
22         // GET: api/Persona/5
23         public Persona Get(int id)
24         {
25             GestorPersona gestorPersona = new GestorPersona();
26             return gestorPersona.ObtenerPersona(id);
27         }
28     }
29 }
```



Referencias:

<https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/enabling-cross-origin-requests-in-web-api>