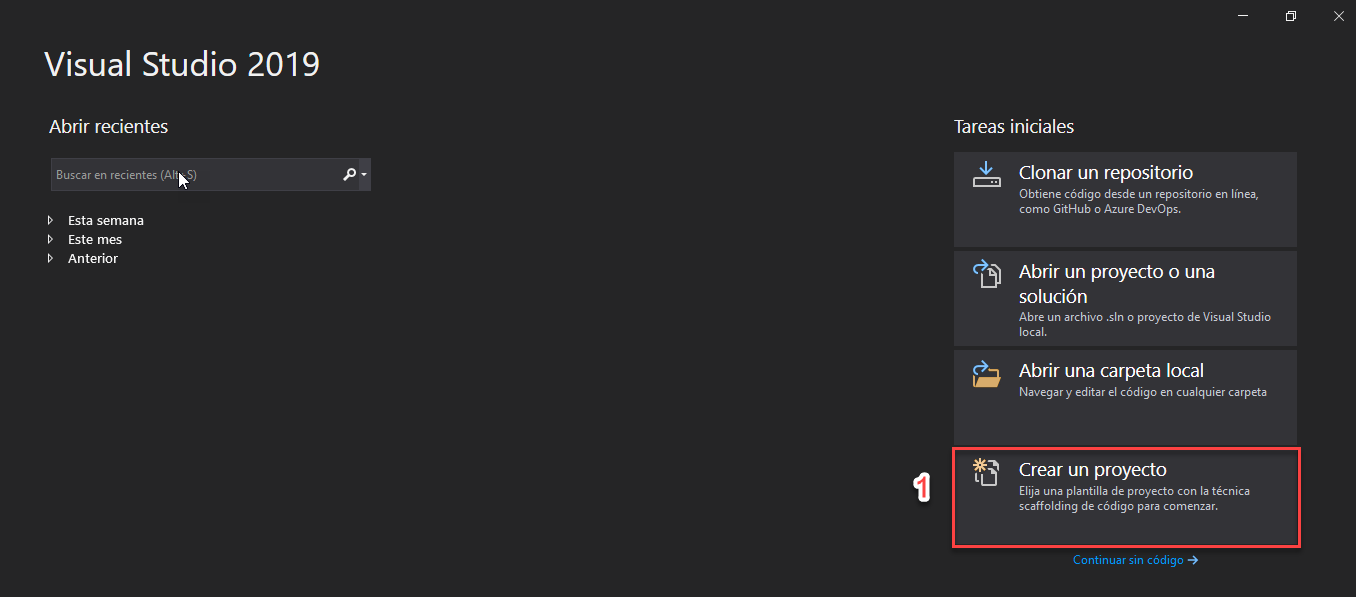
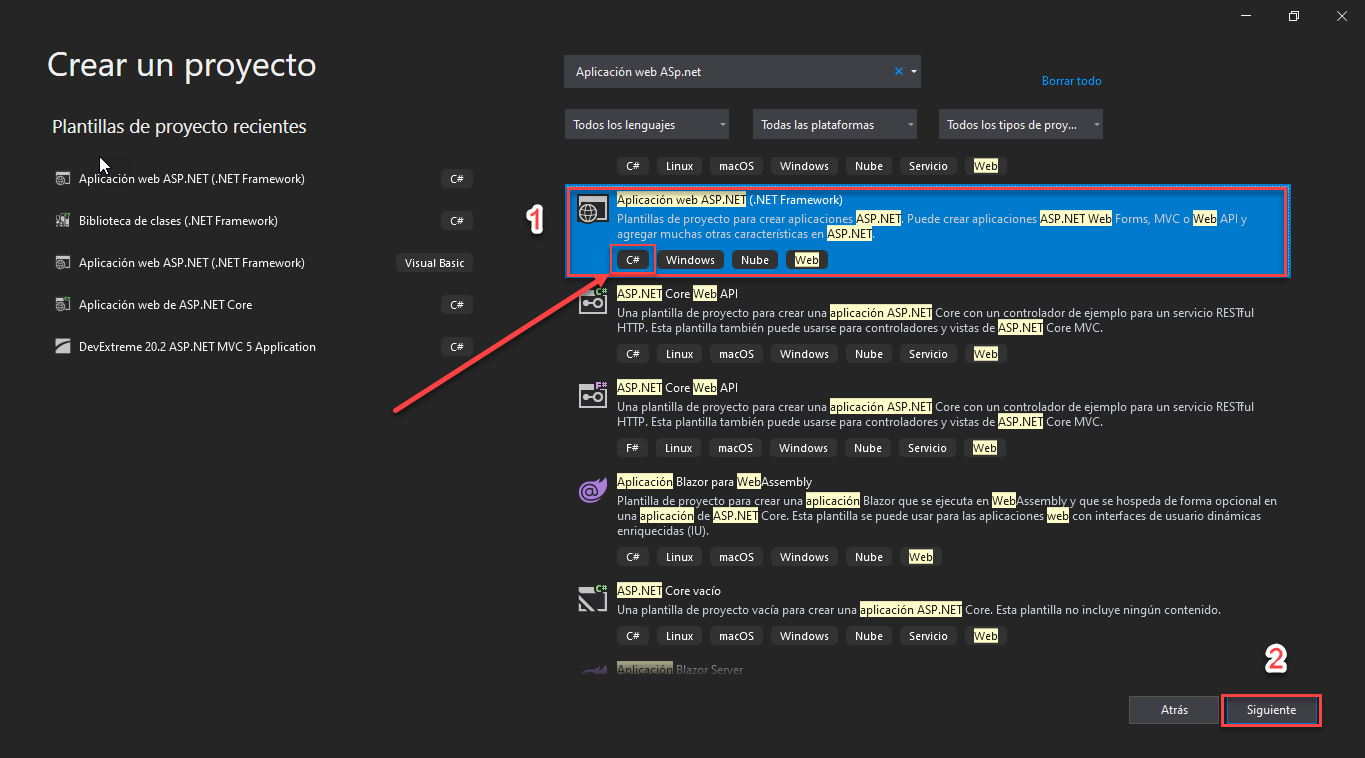
¿Qué es una API Rest?

Creando nuestro proyecto Web

Lo primero es crear un nuevo proyecto en Visual Studio, como se muestra en la siguiente imagen.

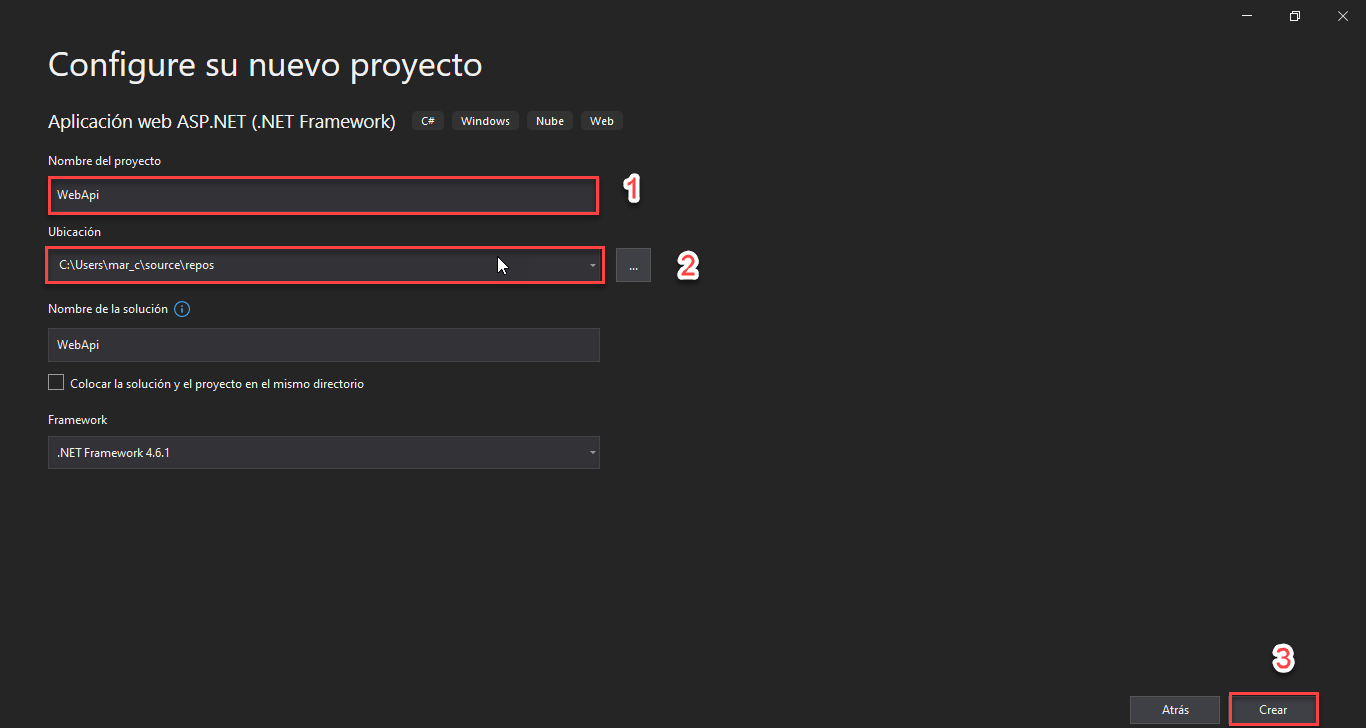


A continuación vamos a seleccionar **Aplicacion web ASP.NET (.NET Framework)**, como se muestra en la imagen siguiente:

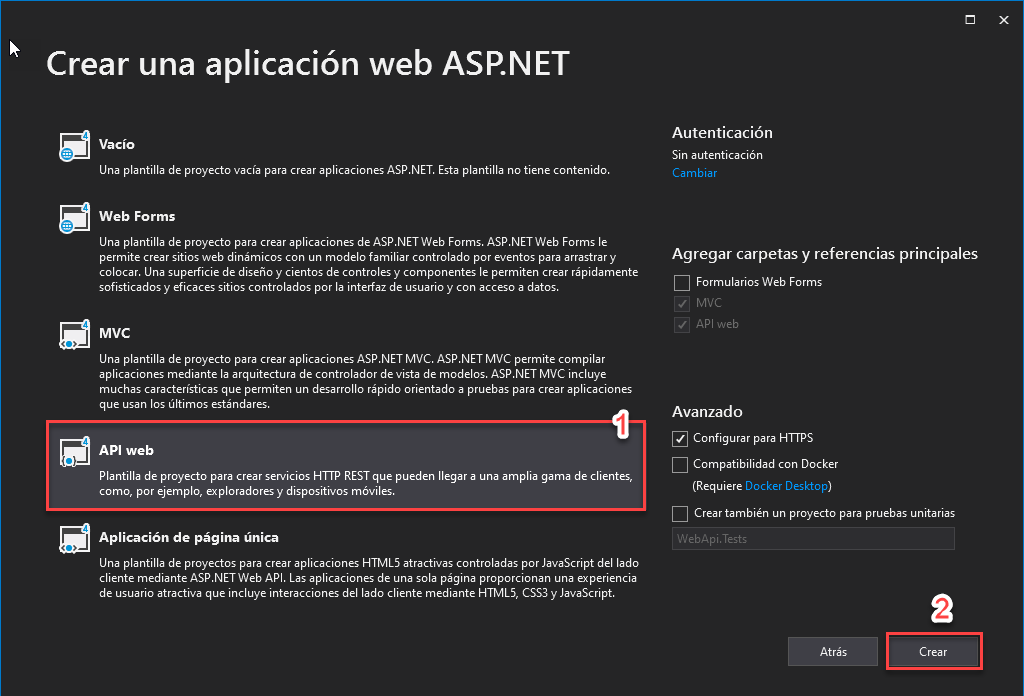


*Nota: Asegurate se que el proyecto seleccionado sea del tipo C#, por que puedes confundirte con un proyecto del tipo VisualBasic*

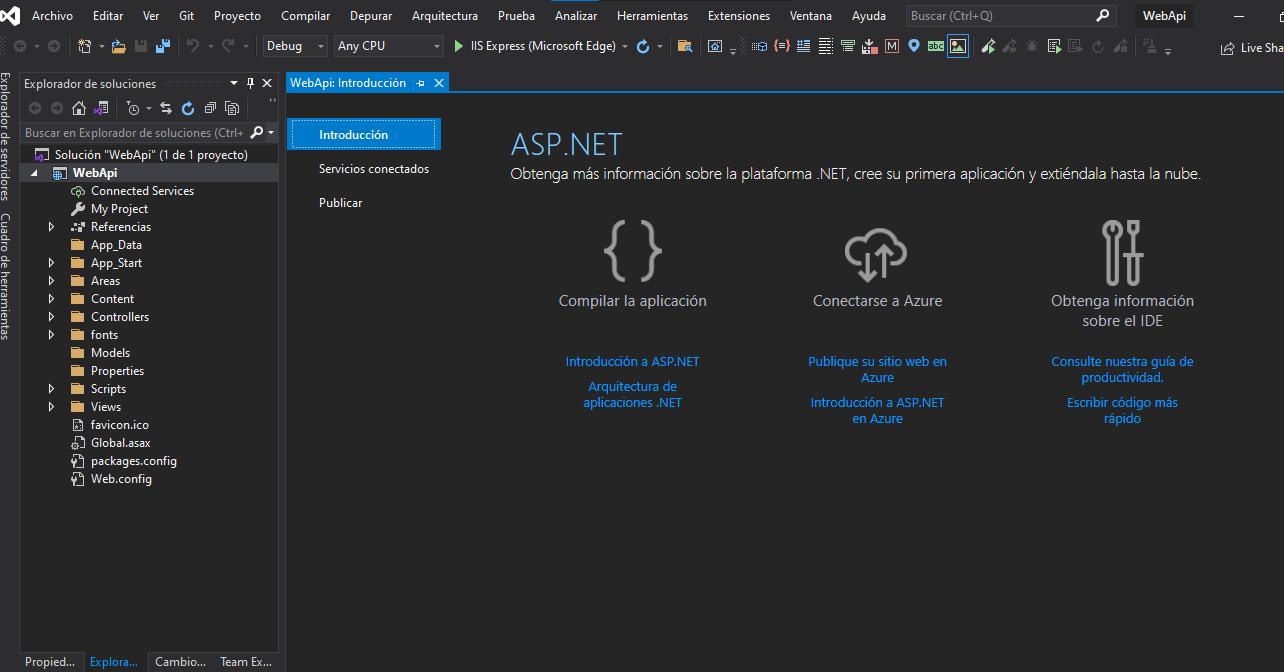
Configuramos el nombre de nuestro proyecto, asi como la ubicación del mismo.



A continuación seleccionaremos el template o plantilla API Web.

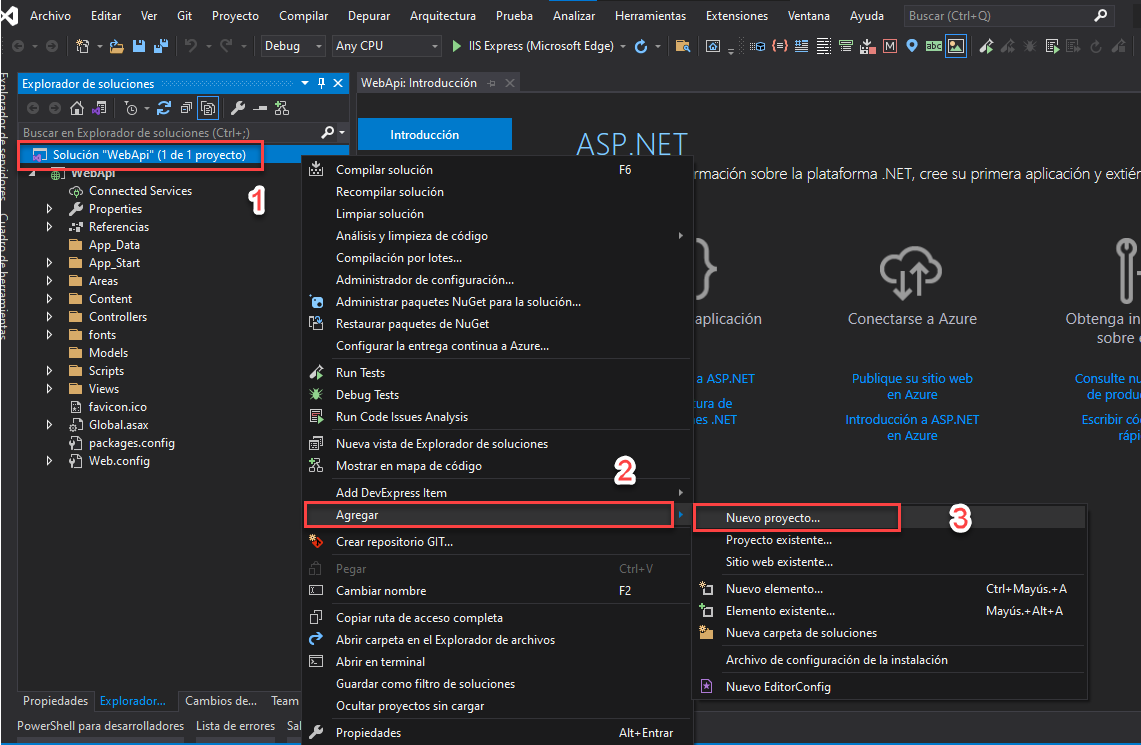


Una vez terminado el proceso, se mostrara una pantalla de información a ASP.NET.

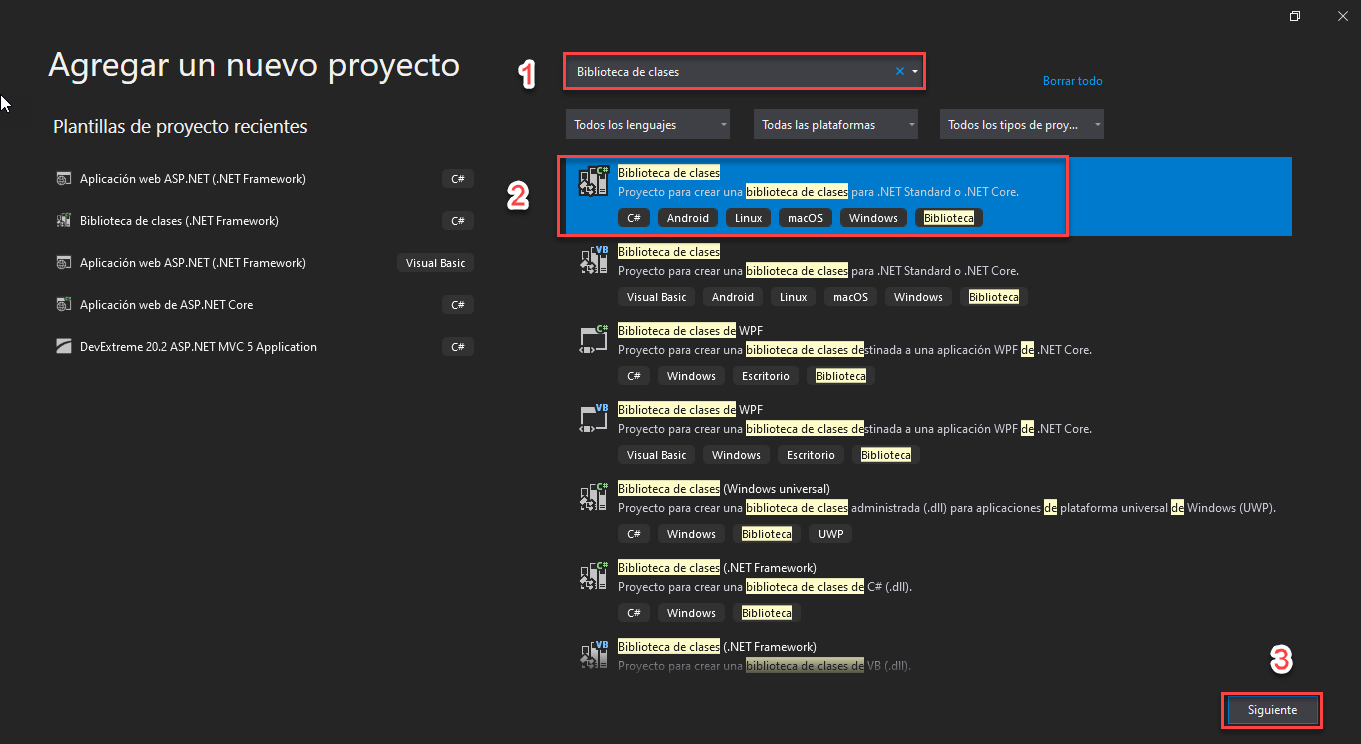


Agregando proyecto de Acceso a Datos

Agregamos un nuevo proyecto donde realizaremos la conexion a la Base de Datos como se muestra a continuación: **Seleccionamos el nombre de la Solución > Click derecho sobre el proyecto > Agregar > Nuevo proyecto.**

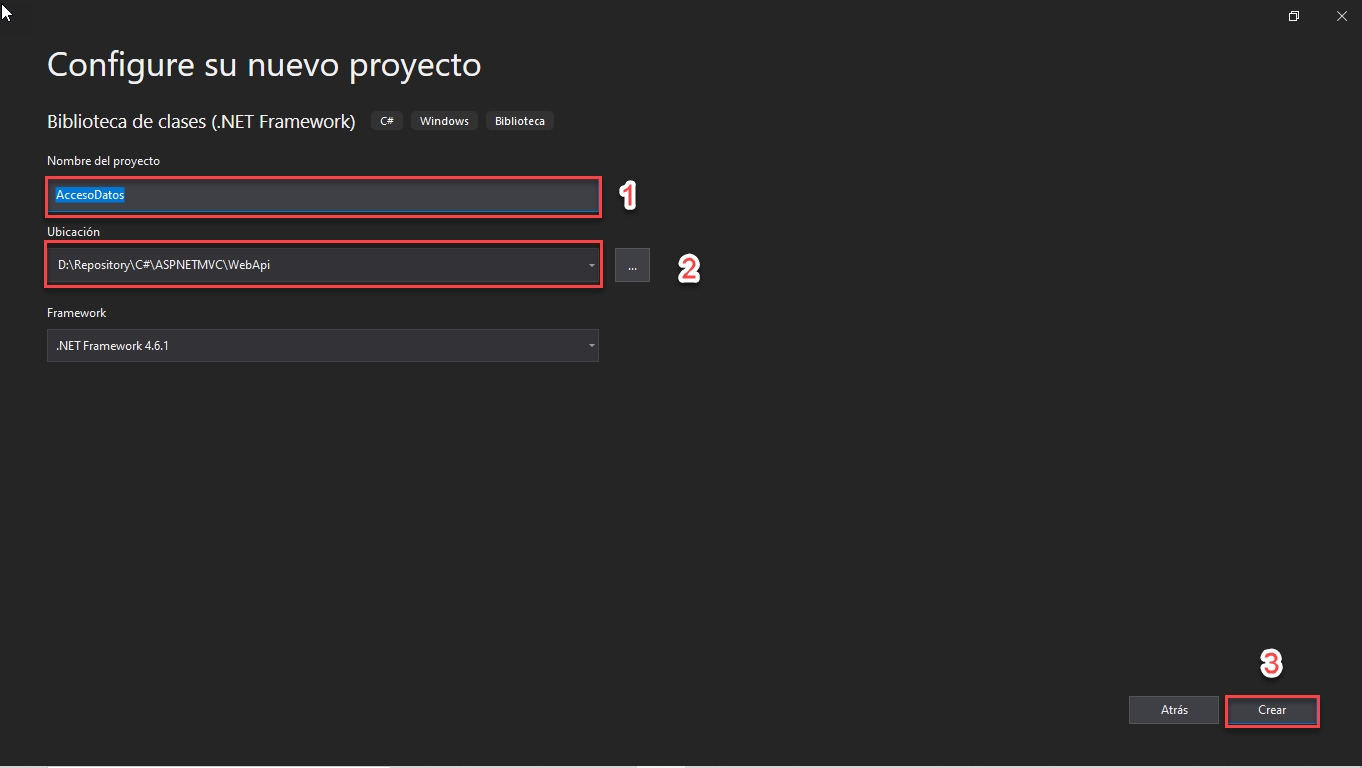


El nuevo proyecto será del tipo biblioteca de clases, para esto podemos hacer uso del control de busqueda de plantilla teclear **"Biblioteca de Clases"** en cual nos mostrara el proyecto que vamos a ocupar.

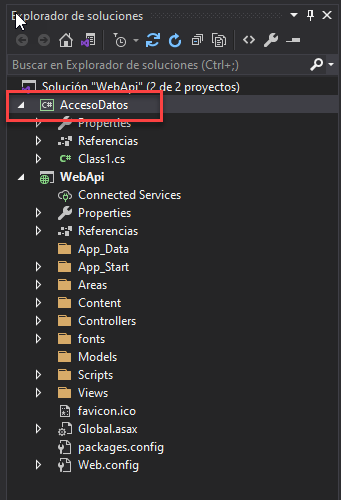


*Nota: Asegurate se que el proyecto seleccionado sea del tipo C#, por que puedes confundirte con un proyecto del tipo VisualBasic*

Configuramos el nombre de nuestro proyecto en este caso será **AccesoDatos**, por defecto dejaremos la misma ubicación del proyecto.

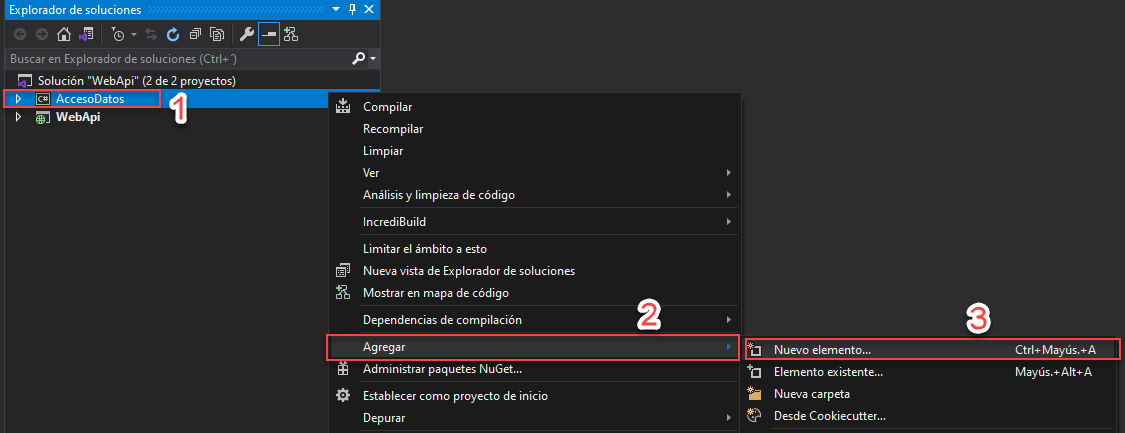


Una vez que se ha creado nuestro segundo proyecto podemos ver los dos proyectos en la misma solución como se muestra en la siguiente imagen.

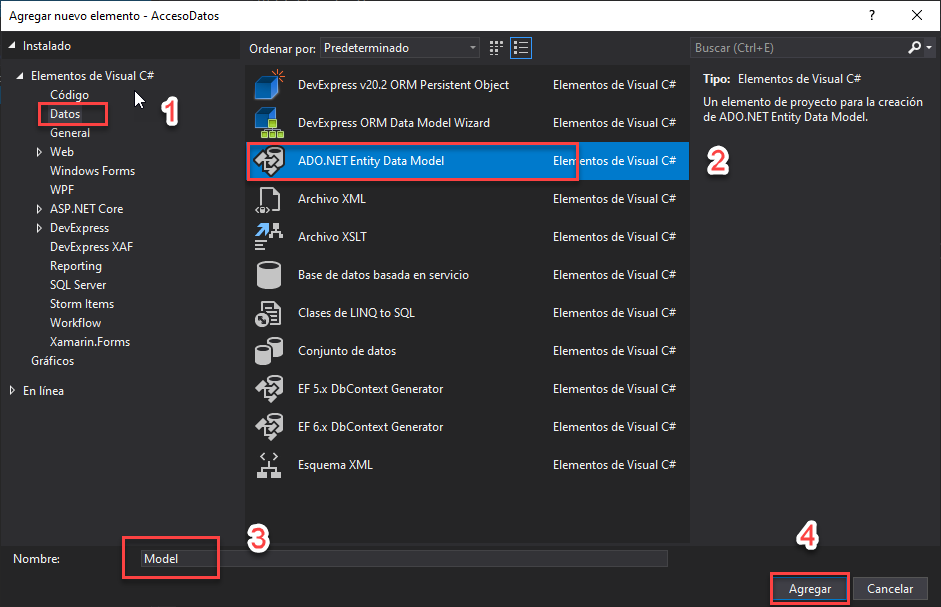


*Nota: Podemos eliminar la Class1.cs ya que no la necesitamos para este ejemplo.*

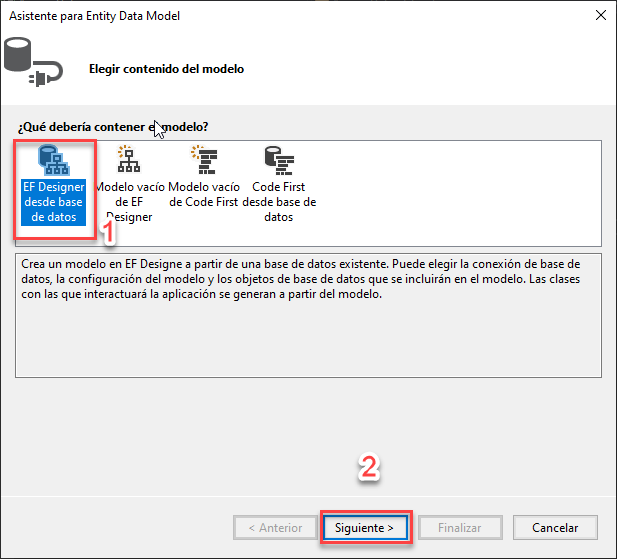
A continuación procedemos a agregar un nuevo elemento como semuestra a continuación: **Seleccionamos el nombre del proyecto "AccesoDatos" > Click derecho sobre el proyecto > Agregar > Nuevo elemento.**



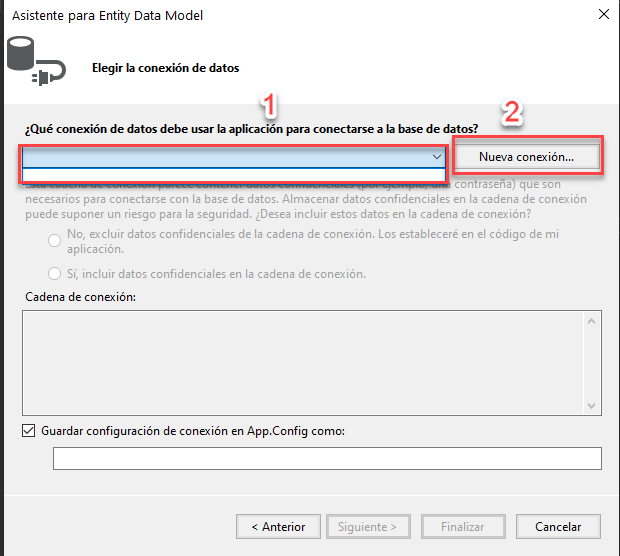
El elemento que vamos a agregar será **Visual C# > Datos >ADO.NET Entity Data Model** le cambiamos el nobre del Modelo y le damos agregar.



En el asistente para Entity Data Model escogeremos el primero **EF Designer desde base de datos**

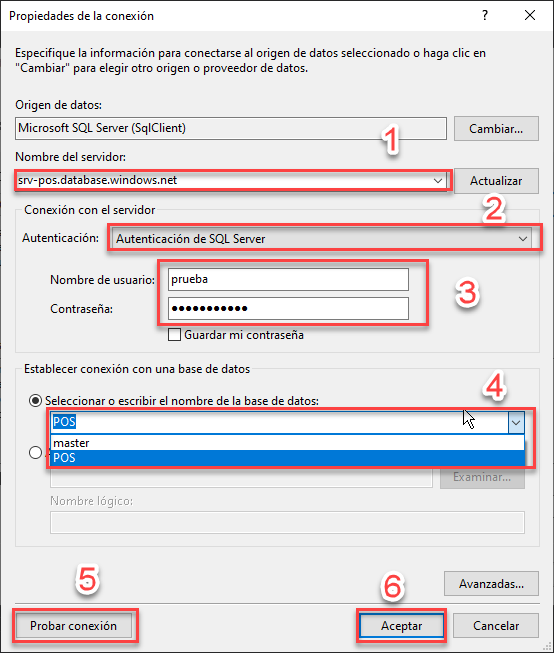


A continuación debemos de seleccionar la conexión de datos que vamos a ocupar, en este caso como es un proyecto nuevo no tenemos ninguna por lo que deberemos de crearla, para ello vamos a darle click al boton nueva conexión.



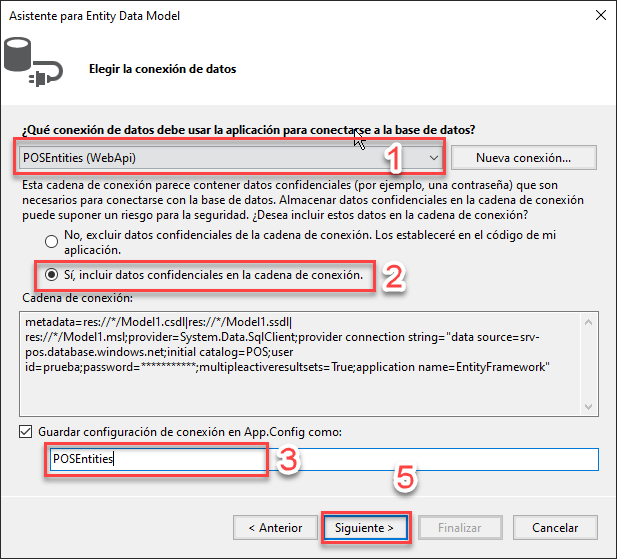
Se nos mostrara la pantalla de propiedades de la conexión, donde nos pide:

* Nombre Servidor
* Tipo de Autenticación
* Usuario
* Contraseña
* Nombre Base de Datos

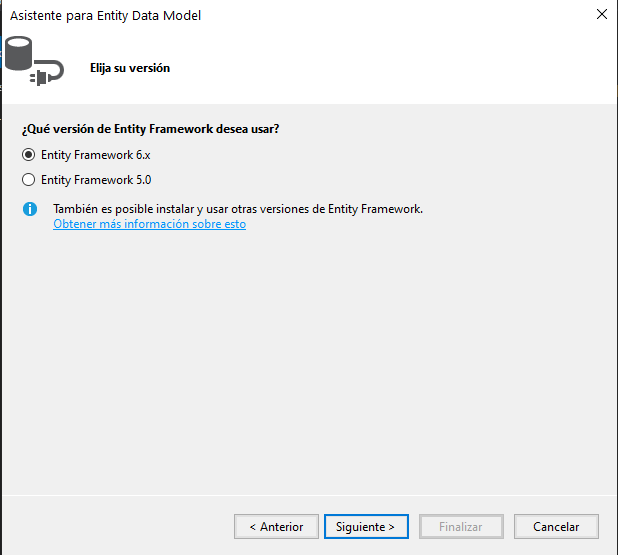


*Nota: En caso de querer ocupar otro tipo de origen de datos debes de darle click al boton cambiar y seleccionar el origen de datos que deseas. Una vez que haz llenado todos los campos requeridos con los datos que te conectaran con tu base de datos puedes procedes a probar conexión y aceptar.*

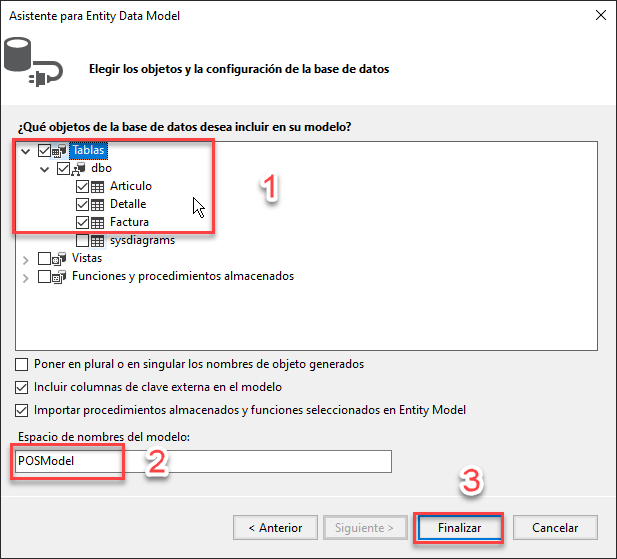
Si nuestra cadena de conexión se ha creado correctamente el asistente nos regresara a la pantalla previa y llenara los datos que necesitamos, en este punto como este es un ejemplo seleccionaremos incluir datos confidenciales de nuestra cadena de conexión, pero lo ideal es que esta información no permitieramos para luego pasarlo a través de nuestro código, pero ese es tema para otro blog.



Luego seleccionamos la versión del Entity Framework en este caso la version 6.x.

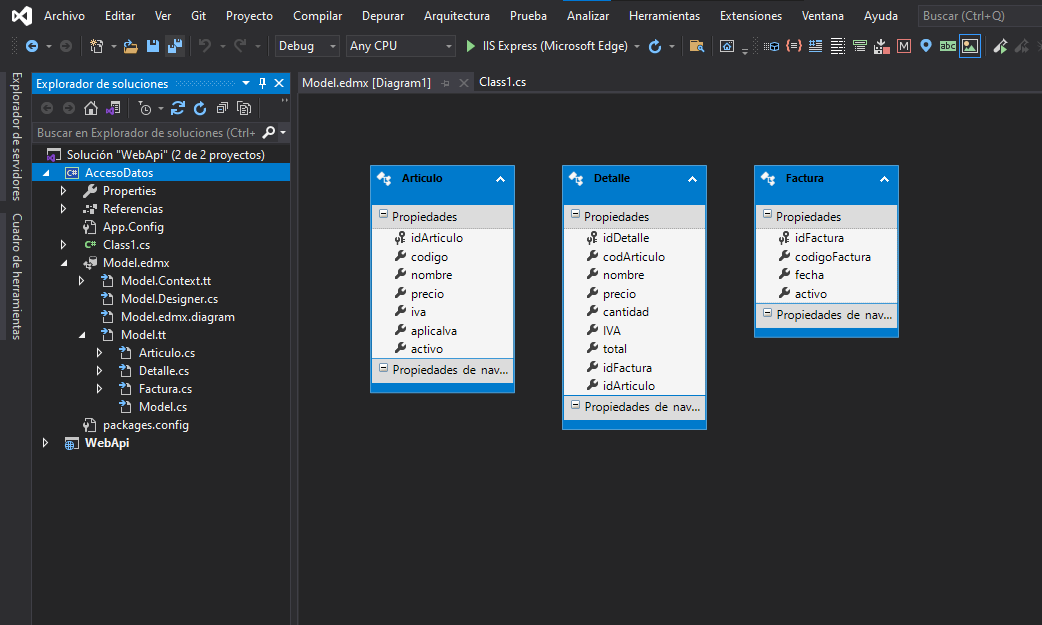


En la siguiente pantalla del asistente seleccionamos la base de datos y las tablas que vamos a utilizar, por ultimo tecleamos el nombre de nuestro modelo, en este caso **POSmodel** y haga clic en finalizar.



*Nota: 1) Puedes Pluralizar o singularizar el nombre de objeto generado: básicamente, agregar o quitar****"s"****o es en los nombres de objeto. 2) Incluir una columna de clave externa en el modelo: esta propiedad se utiliza para incluir relaciones entre tablas.*

Una vez que lo haga, obtendrá una buena vista de sus tablas (detrás de escena).

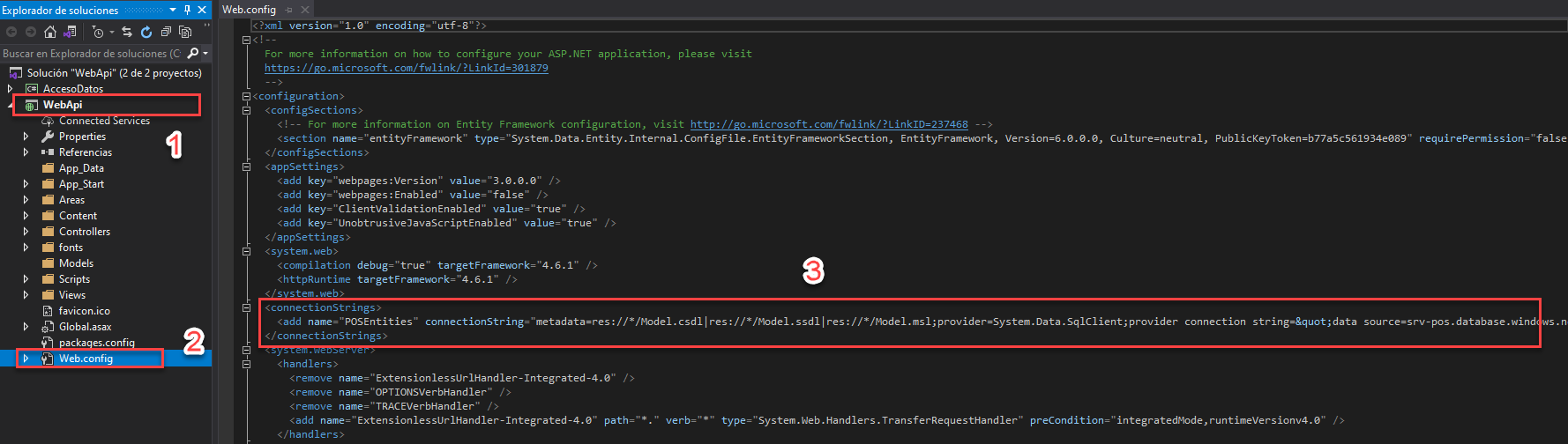


*Nota: Asegurate en este momento de compilar tu proyecto, para que no te de problemas más adelante.*

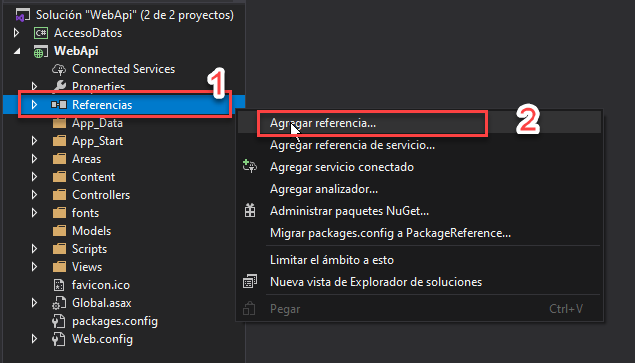
En nuestro proyecto AccesoDatos vamos a buscar el archivo app.Config, donde seleccionaremos y copiaremos la cadena de conexión a la base de datos.



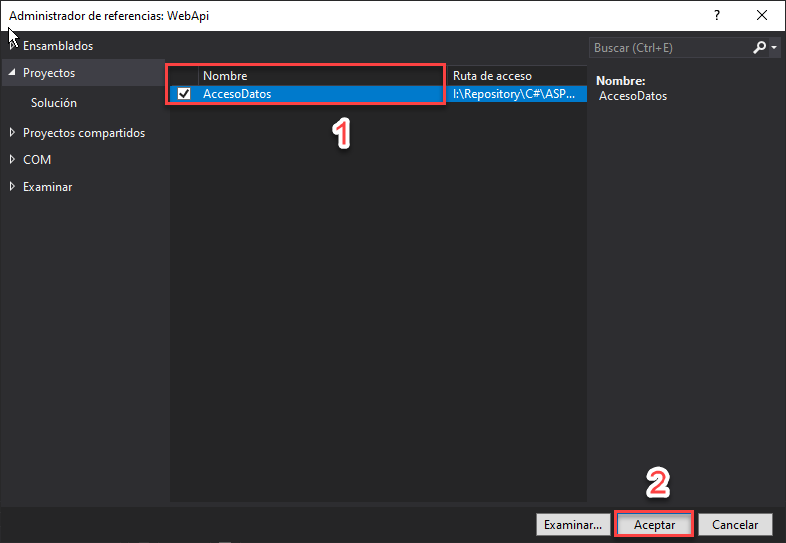
En nuestro proyecto principal **WebApi** vamos a buscar el archivo **Web.config** y pegar la cadena de conexión copiada en el paso anterior.



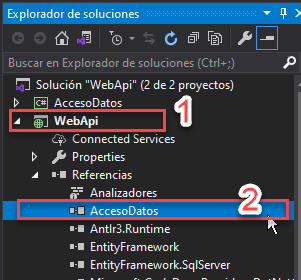
A continuación agregaremos una referencia.



En la opción proyectos seleccionaremos el nombre del proyecto donde se encuentra la configuración de la base de datos, en este caso **AccesoDatos** .

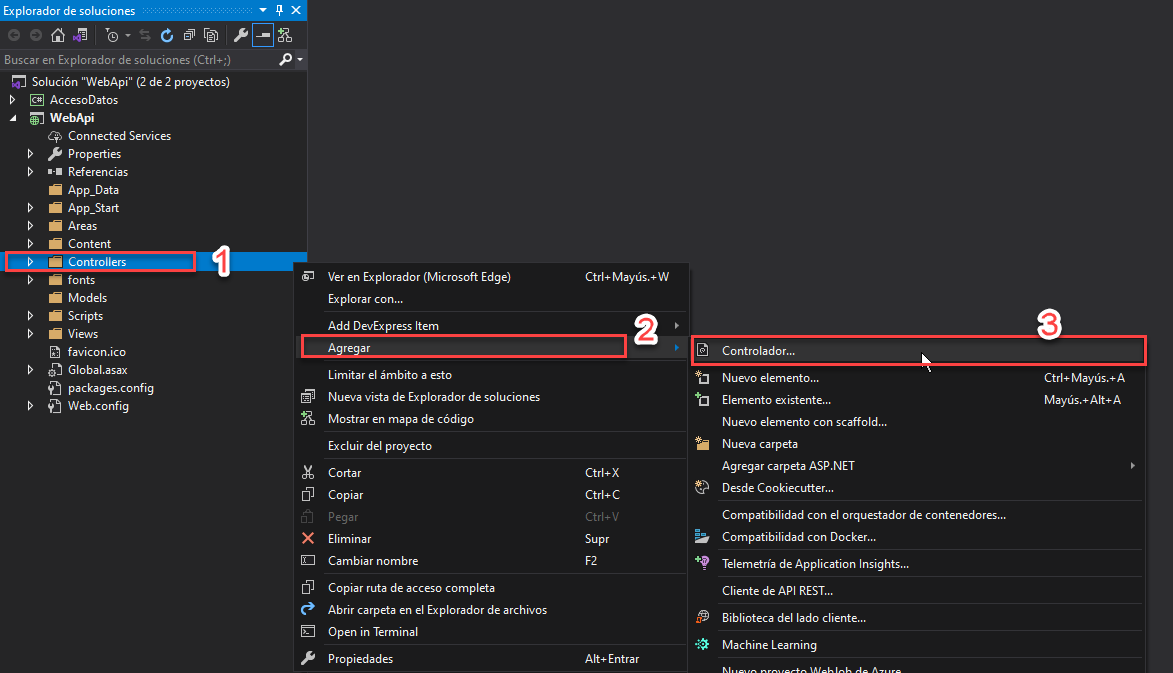


Puedes ver la referencia del proyecto en el apartado **Referencias**.

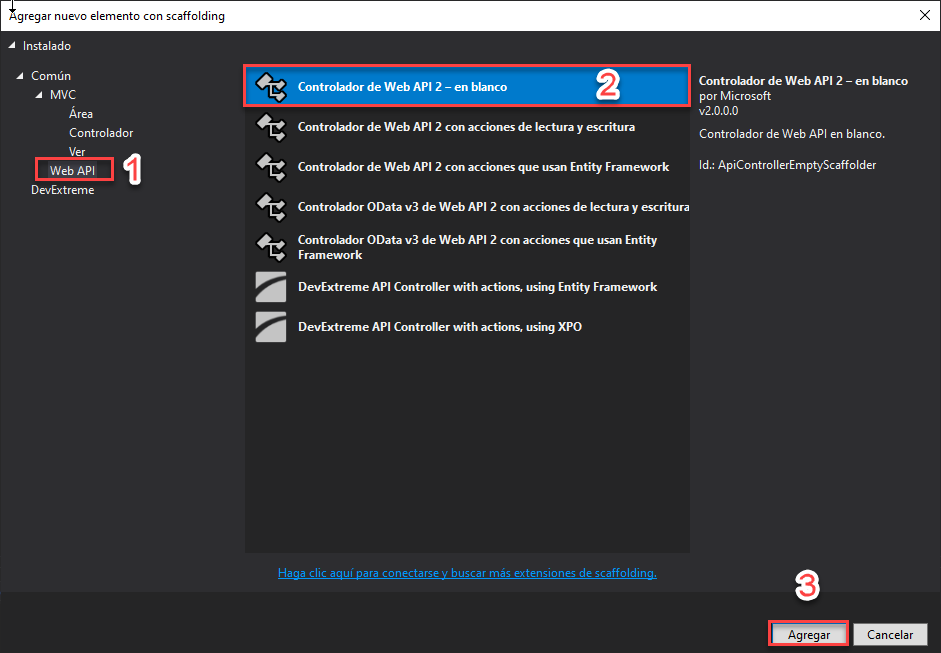


Agregando un Controlador

Siempre en nuestro proyecto principal agregaremos un Controller como semuestra a continuación: **Seleccionamos la carpeta "Controller" > Click derecho sobre el proyecto > Agregar > Controlador.**

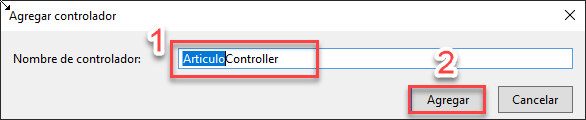


En la pantalla del asistente seleccionaremos **Web API** el tipo **Controlador Web Api 2 - Empty**.



*Nota: En caso de que te salga un error incicando que no puede encontrar el archivo AccesoDatos.dll puedes seleccionar el proyecto AccesoDatos -> Build -> Rebuild Solution.*

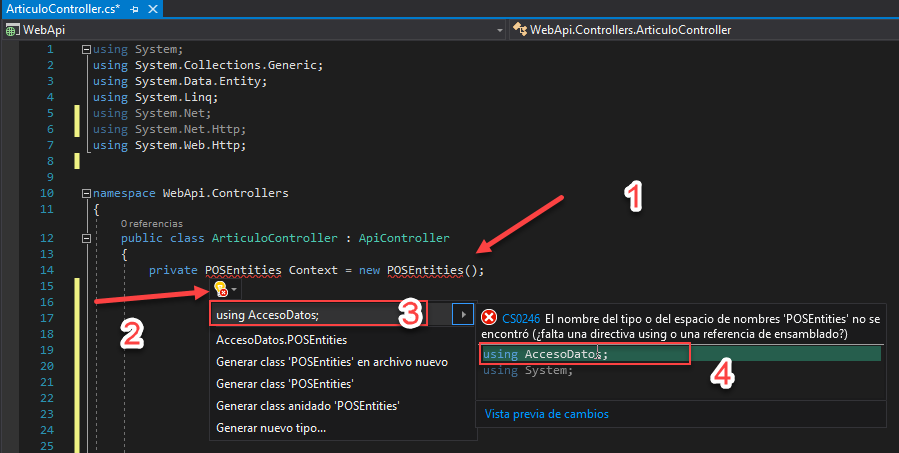
Si no te dio problemas puedes proceder a editar el nombre del controlador, en este caso vamos a ocupar **ArticuloController**.

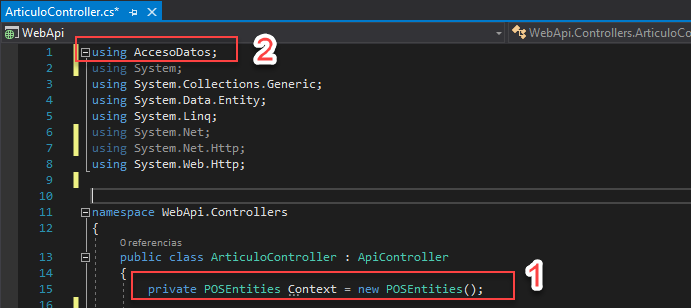


En el controllador que se acaba de crear agregaremos la siguiente línea **"private POSEntities Context = new POSEntities()"** que es la referencia de la conexión de la base datos.

Al agregar esta linea se nos presente un error por que no detecta o no reconoce la referencia de POSEntities, para solucionarlo tenemos que agregar la referencia del proyecto.

Una solución sencilla es posicionarlos en POSEntities damos click al icono icono y este nos desplegara un menu donde seleccionaremos **"using AccesoDatos;"**

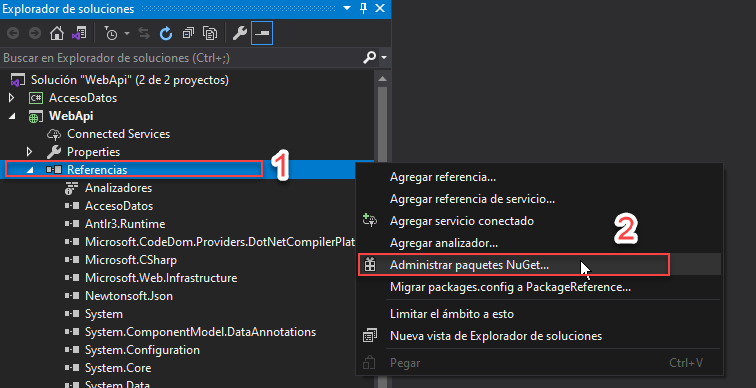


Si hemos realizado los pasos correctamente nuestra controlador deberia de verse a como se muestra a continuación. 

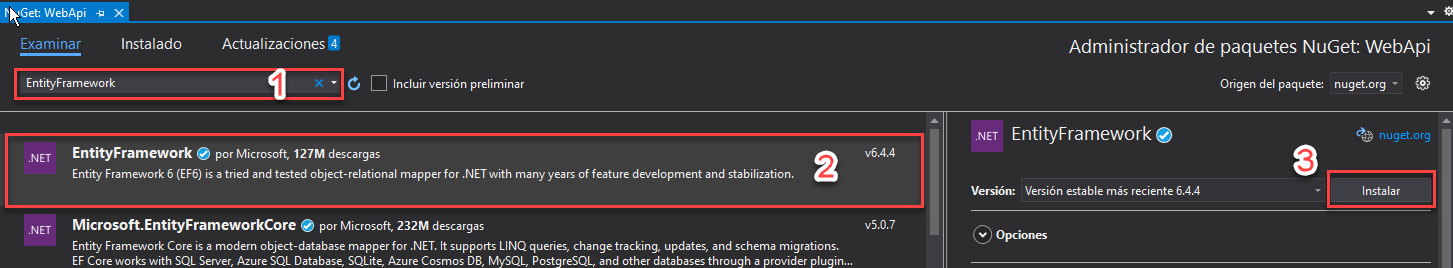
Administrar paquetes NuGet

Para poder continuar con este ejemplo necesitamos algunos paquetes o archivos DLL necesarios, para ellos vamos a hacer uso del Administrador de paquetes Nuget que nos ofrece Visual Studio.

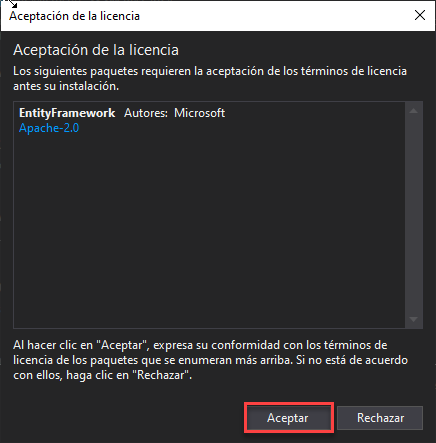
En nuestro proyecto WebAPI vamos a seleccionar Referencias > Administrar paquetes NuGet...



En el administrador de paquetes en el apartado Examinar vamos a buscar el paquete **EntityFramework** y le damos instalar.

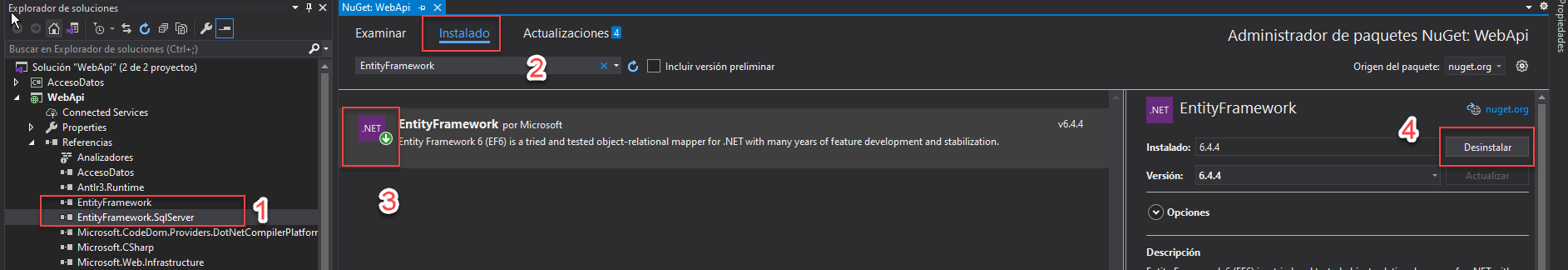


Al momento de la instalación se mostraran una ventana de Aceptación de Lincencia, le damos Aceptar para poder continuar.

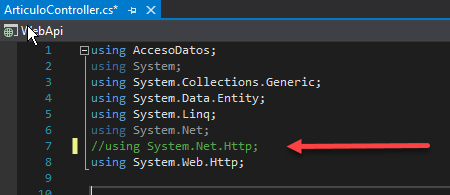


Se agregaran 2 referencia **EntityFramework** y **EntityFramework.SqlServer**. Puedes revisarlo en:

1. Las referencias del proyecto.
2. Apartado instalado del Administrador de paquetes Nuget.
3. El paquete instalado muestra un icono en color verde.
4. Al seleccionar el paquete en la parte derecha nos muestra la opción de desinstalar.



En la sección de referencias del controlador vamos a eliminar la referencia **using System.Net.Http**.



Agregando CRUD

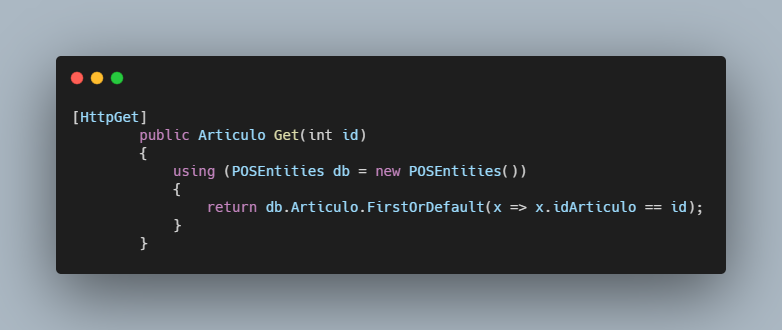
En este punto vamos a agregar el CRUD que ocupara nuestra API-REST.

La primera acción que vamos a crear es la de **Consultar** donde ocuparemos el método **GET**, para ello agregaremos el siguiente código que realizara la consulta de todos los registros de la tabla.

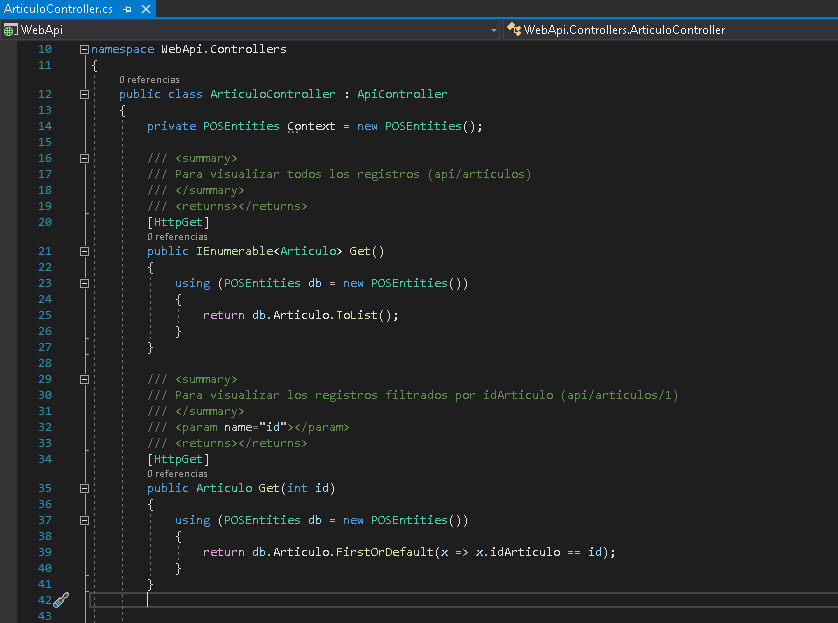


*Nota: El nombre de la tabla Articulo que se ocupa en****IEnumerable<Articulo>****la podemos obtener consultando la referencia de nuestra conexión a la base de datos.*

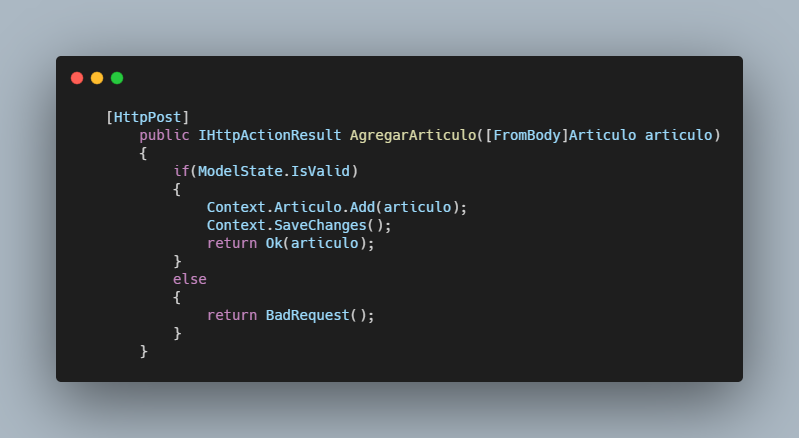
Para retornar los valores filtrados por id deberiamos agregar el siguiente código.



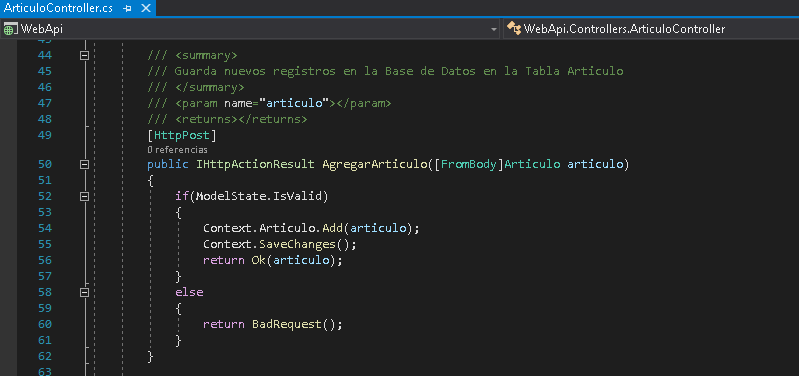
Tu código debería quedar de la siguiente forma.



La siguiente acción a realizar es para **Crear** un nuevo registro, en este caso vamos utilizar el método **POST**.



Tu código debería quedar de la siguiente forma.

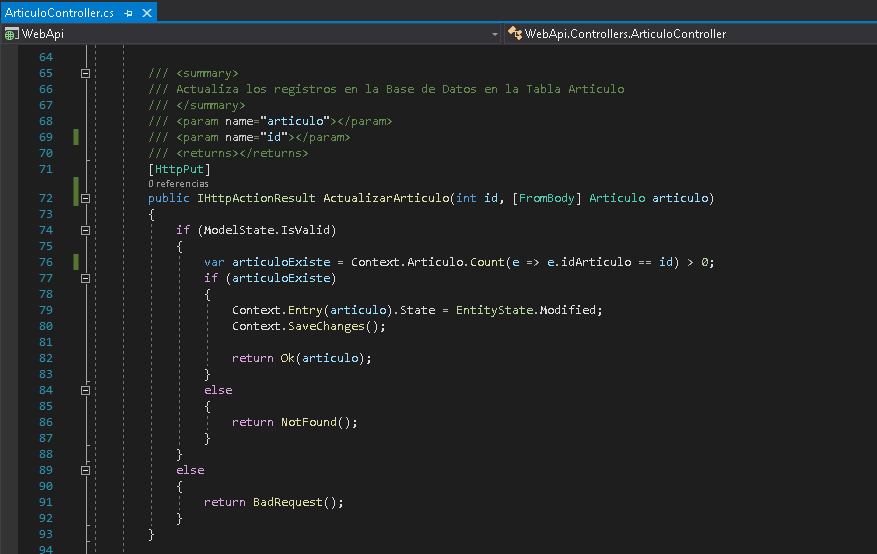


La siguiente acción a realizar nos permitira **Actualizar** nuestros registros, en este caso vamos utilizar el método **PUT**.



*Nota: En caso de que te salga un error asegurate de tener agregada la referencia****System.Data.Entity****.*

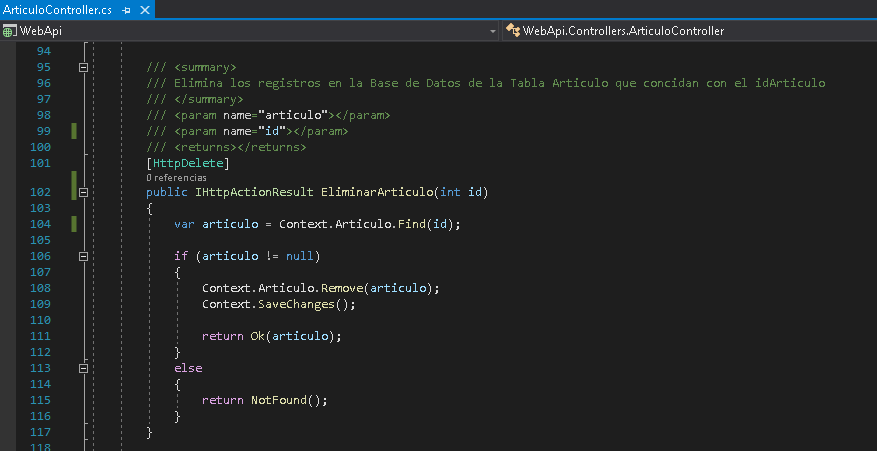
Tu código debería quedar de la siguiente forma.



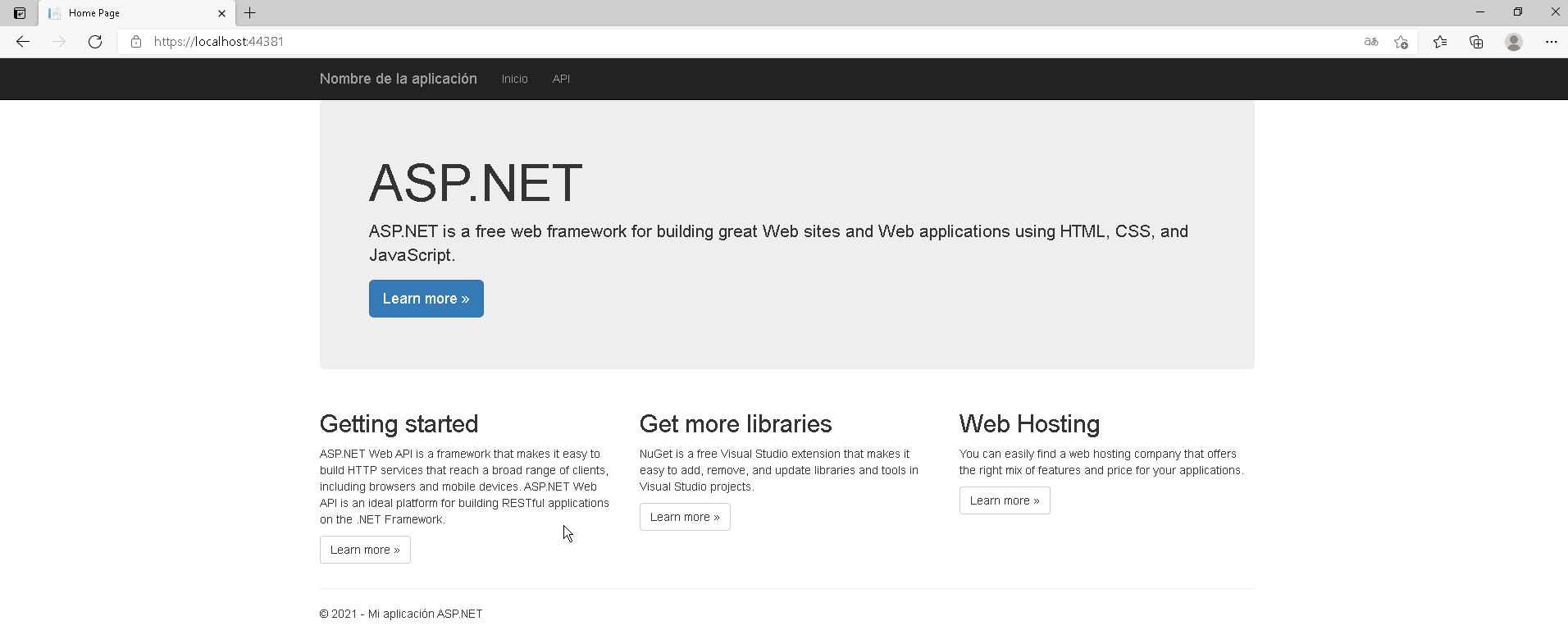
La siguiente acción a realizar nos permitira **Eliminar** nuestros registros, en este caso vamos utilizar el método **Delete**.



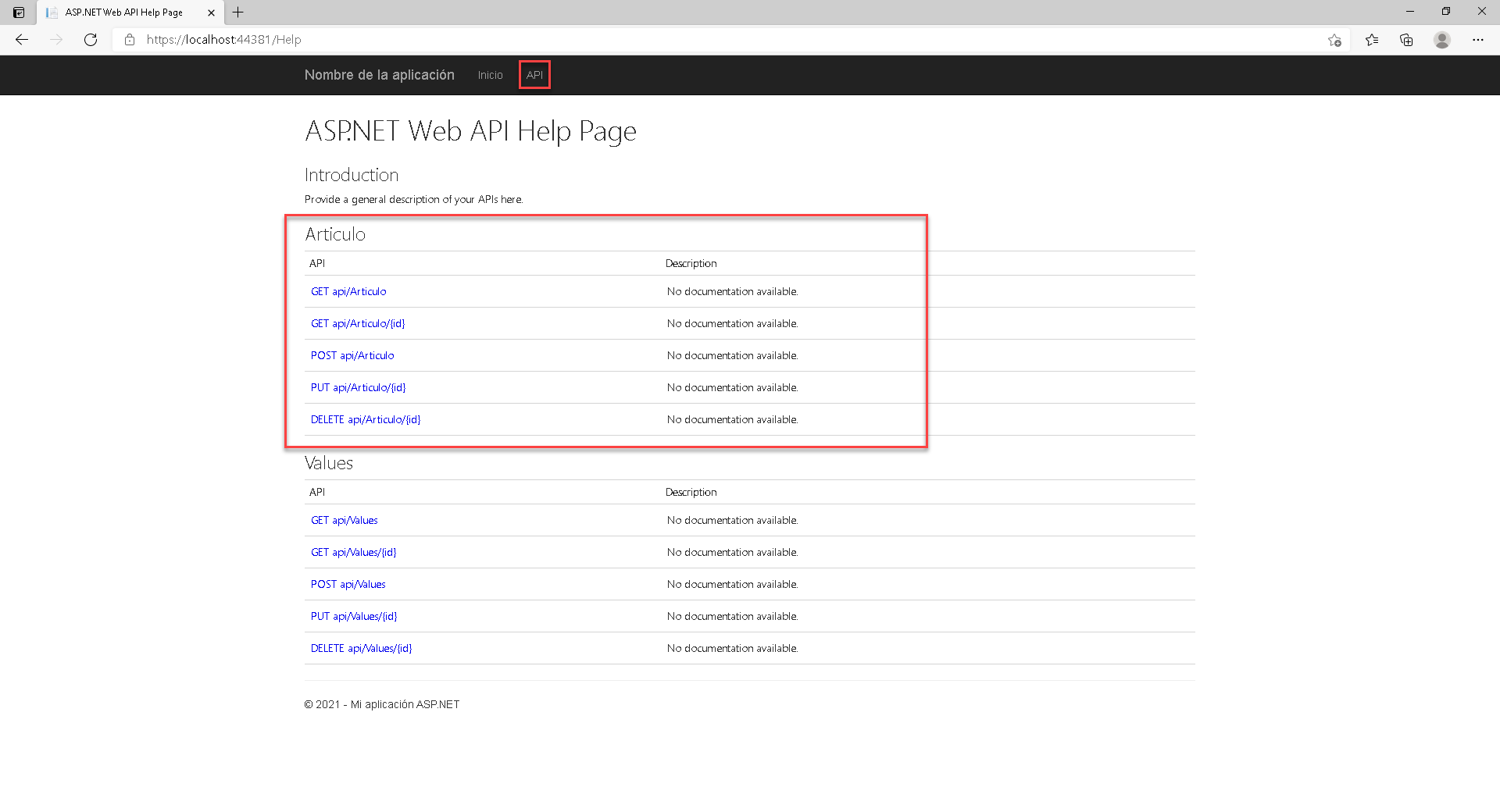
Tu código debería quedar de la siguiente forma.



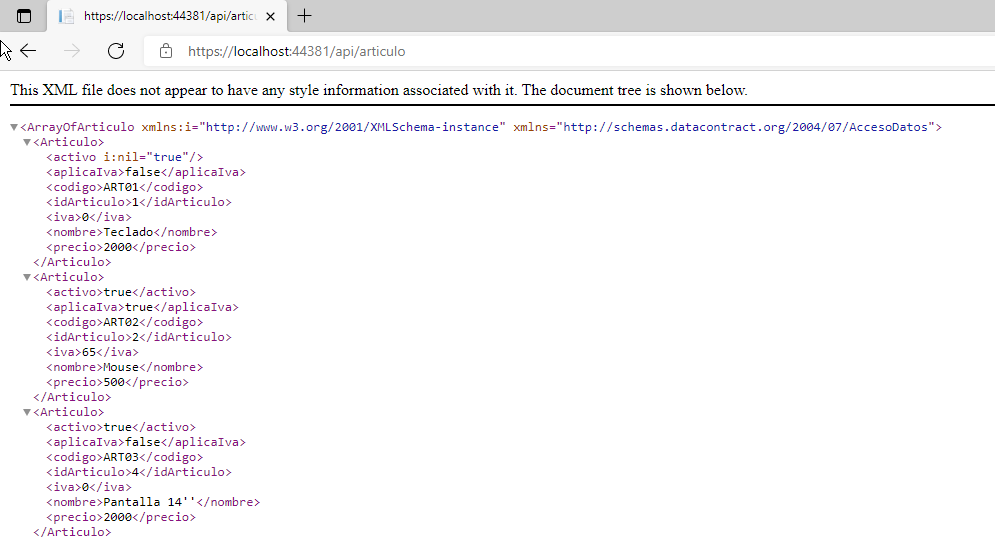
A continución ejecutaremos nuestra aplicación la cual nos abrira nuestro navegador con la dirección de Localhost que nos mostrara la página principal de nuestra API-REST.



En la página que se nos ha cargado, seleccionaremos el apartado API, aquí podemos visualizar los métodos activos en este caso el primero que se nos muestra es el método GET.



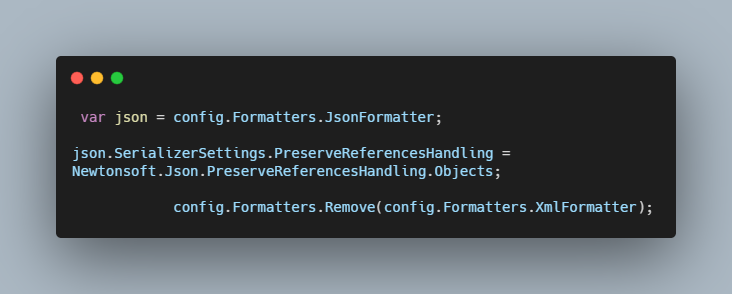
Para poder probar que lo que hemos hecho esta funcionando vamos consultar todos los registros de la tabla, en nuestro navegador debemos teclear despues de **localhost:puerto/** **api/articulo**, quedando de la siguiente manera **localhost/api/usuarios**, si todo esta bien se visualizaran todos los registros de la tabla en formato XML.



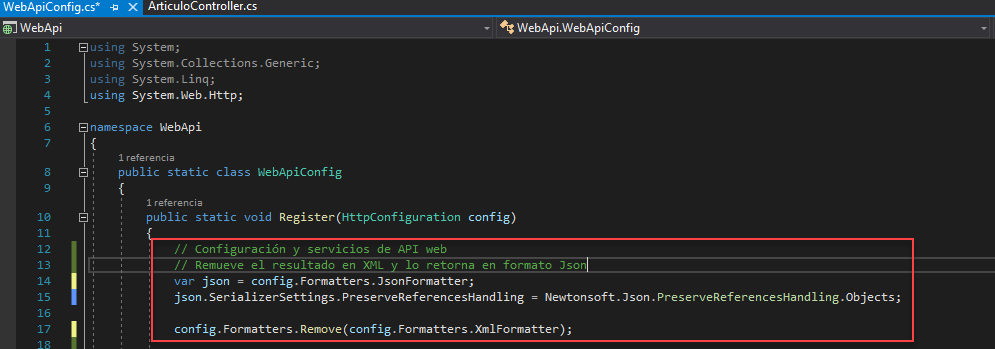
*Nota: En este ejemplo estamos ocupando la tabla Articulo tu tendras que cambiarla por las tablas que vas a ocupar ejemplo: api/usuarios, api/facturas, api/clientes, etc*

La salida de la información esta en formato XML para generarla en formato Json debemos editar el archivo **WebApiConfig.cs** que se ubica en el proyecto principal WebAPI.

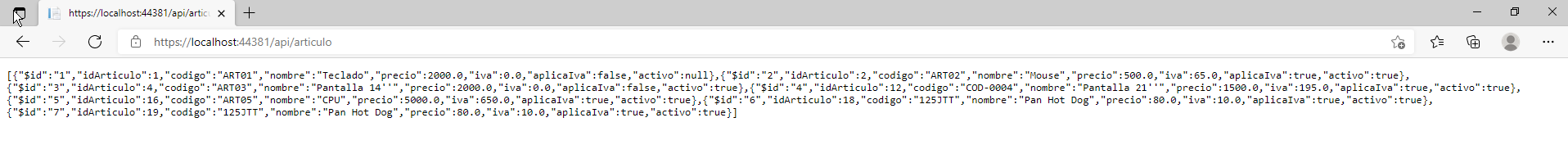
Teclearemos el siguiente código que realizar la salida en formato Json y remueve la salida en formato XML.



Tu código debería quedar de la siguiente forma.



Si volvemos a realizar la consulta de todos los registros veremos que la salida será en formato Json.



Probando API-REST con POSTMAN

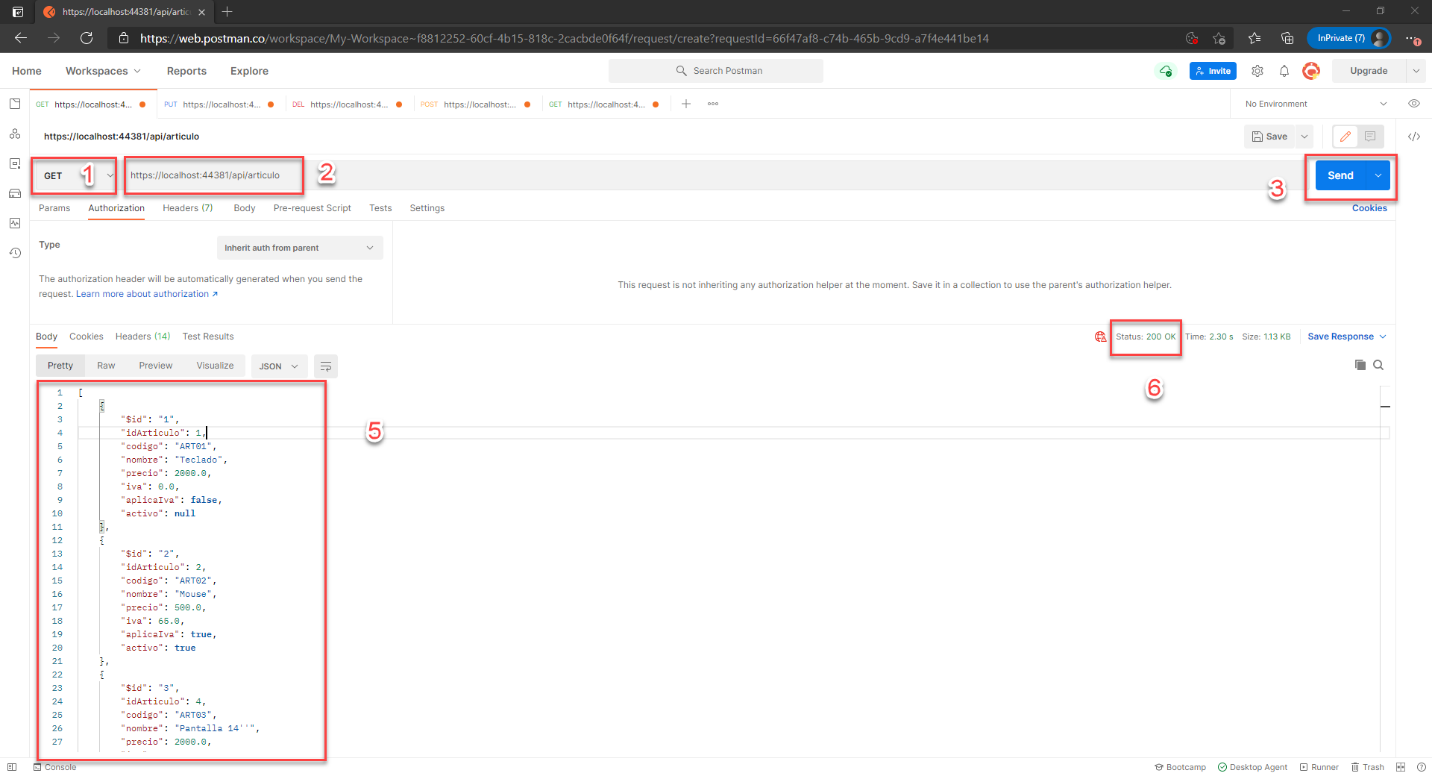
Llegamos al final de esta publicación y en esta sección vamos a realizar la prueba final de nuestra API-REST.

Lo primero que debemos realizar es ejecutar POSTMAN que previamente deberias tener instalado, si aun no lo tienes instalado te dejo el [Link](https://www.postman.com/) para que te registres y descargues.

MÉTODO GET (Todos los registros)

Para probar nuestro método GET con POSTMAN deberiamos de realizar los siguientes pasos:

1. Seleccionamos el método **GET**.
2. Tecleamos la **URL** de nuestra API, en mi caso sería: (https://localhost:44381/api/articulo).
3. Presionamos el botón **SEND**.
4. En la parte inferior **(Body)** se visualizan todos los registros de la tabla.
5. Se nos indica el stado de nuestra consulta **Status: 200 ok**.

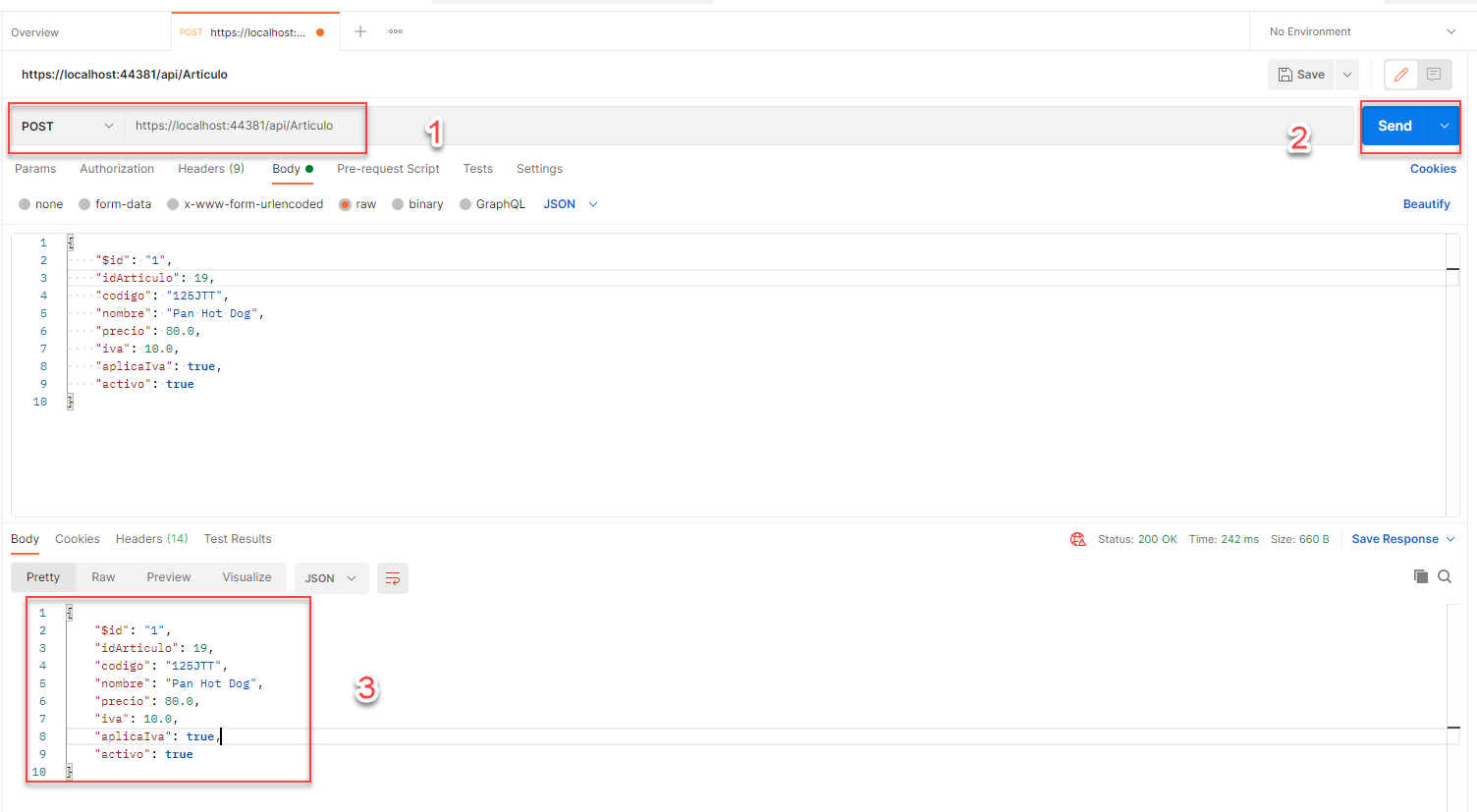


MÉTODO POST (Agregar un nuevo registro)

Para agregar nuevos registros en nuestra base de datos ocuparemos el método POST.

Siguiendo los pasos anteriores nos quedaria de la siguiente manera:

1. Seleccionamos el método **POST**.
2. Tecleamos la **URL** de nuestra API, en mi caso sería: (https://localhost:44381/api/articulo).
3. En los parametros seleccionaremos **Body > raw > Formato Json**.
4. Tecleremos los nuevos datos a agregar en formato Json
5. Presionamos el botón **SEND**.
6. En la parte inferior **(Body)** se visualizara el nuevo registros agregado a nuestra base de datos.
7. Se nos indica el stado de nuestra consulta **Status: 200 ok**.

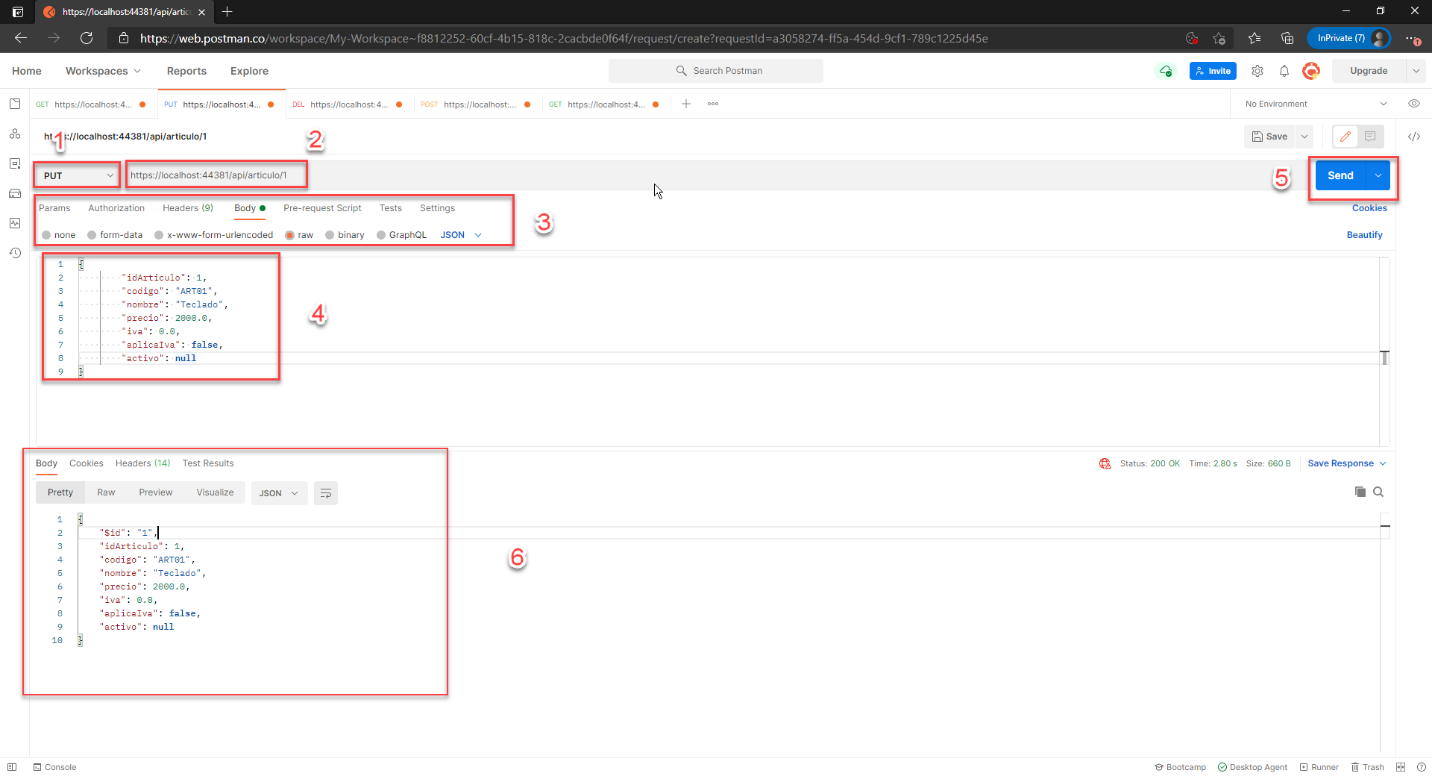


MÉTODO PUT (Modificando un registro)

Para modificar o actualizar nuestros registro en base de datos ocuparemos el método PUT.

Siguiendo los pasos anteriores para actualizar un registro desde Postman realizaremos los siguientes pasos:

1. Seleccionamos el método **PUT**.
2. Tecleamos la **URL** de nuestra API, en mi caso sería: (https://localhost:44381/api/articulo/1).
3. En los parametros seleccionaremos **Body > raw > Formato Json**.
4. Tecleremos los datos a cambiar en formato Json
5. Presionamos el botón **SEND**.
6. En la parte inferior **(Body)** se visualizan los registros ya actualizados.
7. Se nos indica el stado de nuestra consulta **Status: 200 ok**.

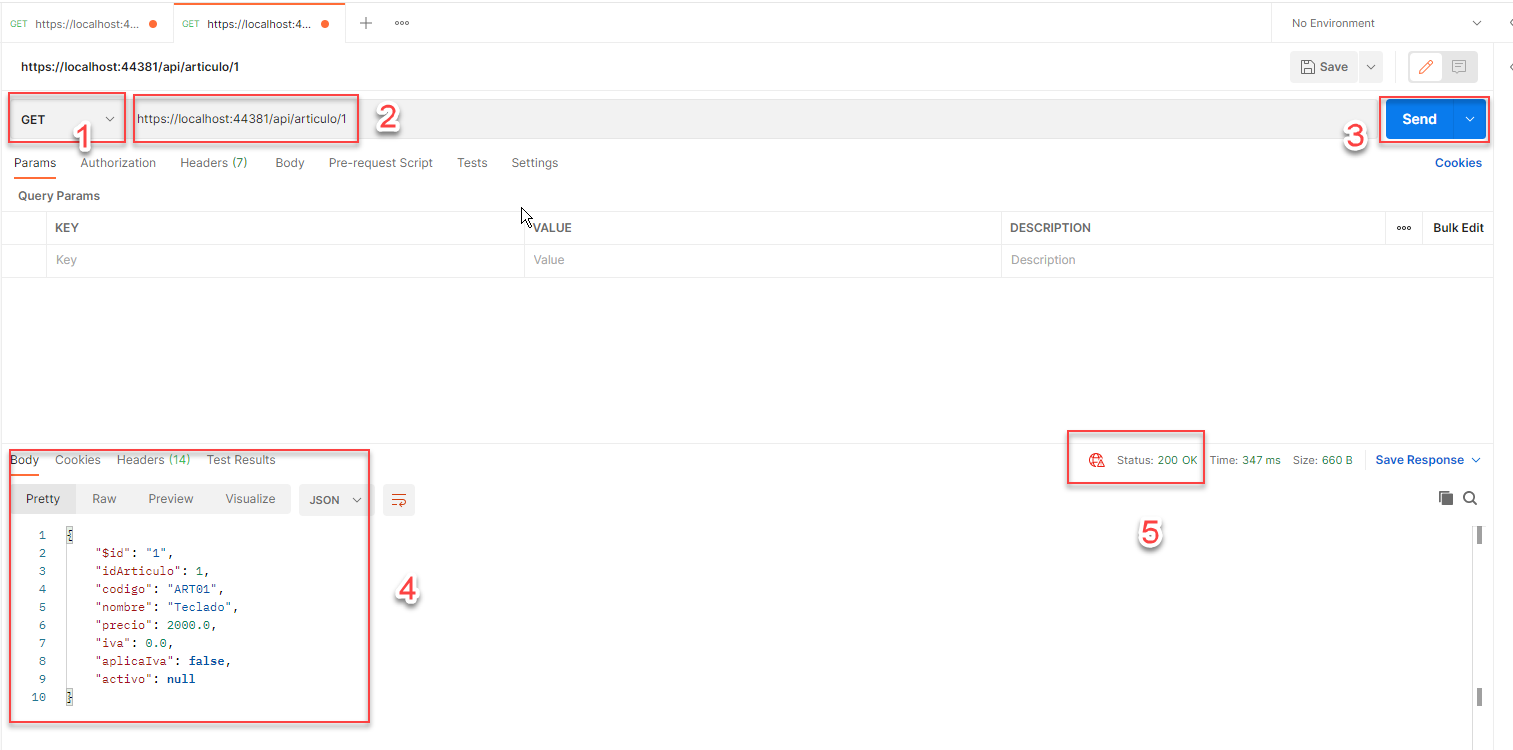


MÉTODO GET (Obtener registros filtrado por id)

Tambien se puede consultar los cambios volviendo a ejecutar un método GET ya sea el de obtener todo o bien el método de filtrado por id.

Para ello podemos realizar los siguientes pasos:

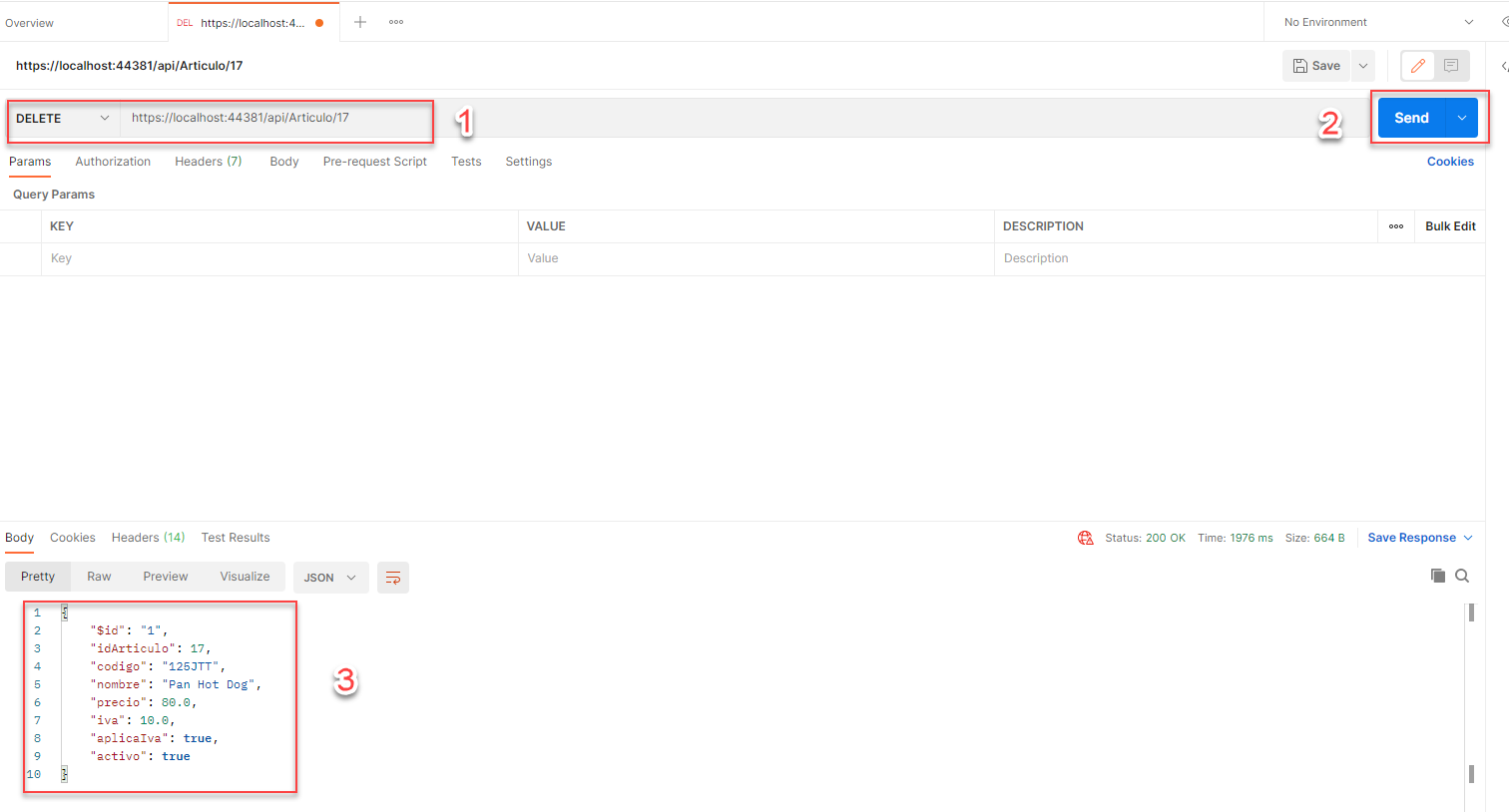
1. Seleccionamos el método **GET**.
2. Tecleamos la **URL** de nuestra API y le agregamos al final el id de nuestro articulo, en mi caso sería articulo/1: (https://localhost:44381/api/articulo/1).
3. Presionamos el botón **SEND**.
4. En la parte inferior **(Body)** se visualizara el registros de que seleccionado.
5. Se nos indica el stado de nuestra consulta **Status: 200 ok**.



MÉTODO DELETE (Eliminar registros)

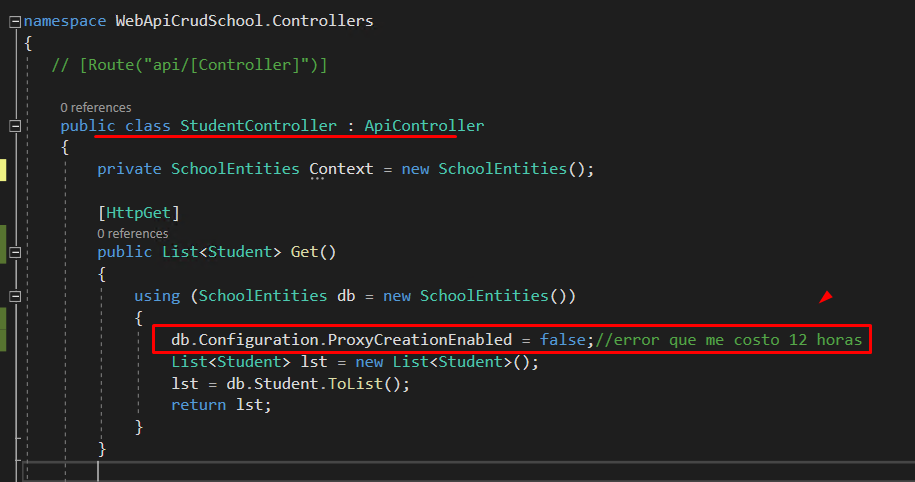
Para eliminar los registros de nuestra base de datos ocuparemos el método Delete. Siguiendo los pasos anteriores nos quedaria de la siguiente manera:

1. Seleccionamos el método **DELETE**.
2. Tecleamos la **URL** de nuestra API y le agregamos al final el id de nuestro articulo que deseamos eliminar, en mi caso sería articulo/17: (https://localhost:44381/api/articulo/17).
3. Presionamos el botón **SEND**.
4. En la parte inferior **(Body)** se visualizara el registros eliminados.
5. Se nos indica el stado de nuestra consulta **Status: 200 ok**.



*Nota: Recuerda que despues de utilizar los métodos POST, DELETE, PUT puedes ocupar el método GET (Obtener Todos) ó el método GET(id) para consultar los cambios realizados.*

ERRORES RELACIONES CON LOS DATOS DEL GET



db.Configuration.ProxyCreationEnabled = false;//error que me costo 12 horas

Captura de pantalla de computadora

Descripción generada automáticamente

var json = config.Formatters.JsonFormatter;

json.SerializerSettings.PreserveReferencesHandling = Newtonsoft.Json.PreserveReferencesHandling.Objects;

config.Formatters.Remove(config.Formatters.XmlFormatter);

json.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore;

Captura de pantalla de computadora

Descripción generada automáticamente

GlobalConfiguration.Configuration.Formatters.JsonFormatter.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore;

GlobalConfiguration.Configuration.Formatters.Remove(GlobalConfiguration.Configuration.Formatters.XmlFormatter);