API Base de Datos Estadísticos Web Service SOAP

Manual Técnico Para Desarrollador

banco central Chile

Índice

1. Introducción

Este manual tiene como objetivo proporcionar una guía detallada para integrar y consumir la **API BDE del Banco Central de Chile** a través de su **servicio Web SOAP**, uno de los métodos disponibles para acceder a los datos económicos disponibles en su **Base de Datos Estadística (BDE).** Este servicio permite acceder a una amplia variedad de indicadores económicos, como series de datos con frecuencias diaria, mensual, trimestral y anual, fundamentales para realizar análisis financieros y económicos automatizados. A través de este servicio, los desarrolladores pueden obtener información relevante de manera eficiente y en tiempo real, integrándola directamente en sus aplicaciones.

Para comenzar a utilizar el servicio, los desarrolladores deben contar con una cuenta activa en la **BDE del Banco Central de Chile**, utilizando sus credenciales (usuario y contraseña) para autenticar las solicitudes. En caso de no contar con una cuenta, deberán dirigirse a la BDE y registrarse. Si es la primera vez utilizando la API BDE en cualquiera de sus variantes (librería bcchapi, REST o SOAP), los desarrolladores deberán dirigirse al portal oficial de la API BDE, donde en la sección "Acceso a la API" encontrarán la opción "Habilitar la API" que permitirá autenticarse y habilitar el acceso apretando el botón "Habilitar API". Una vez realizado este paso, no será necesario volver a repetirlo.

Para guiar el uso de la API BDE mediante su servicio SOAP, en este manual se presentará un ejemplo de C# en Visual Studio 2022, donde se generará un cliente y se hará uso de la API SOAP utilizando el archivo WSDL disponible (https://si3.bcentral.cl/SieteWS/sietews.asmx?wsdl). En este ejemplo, se abordarán tanto las implementaciones en .NET Framework como en .NET 5/6/7/8, mostrando cómo generar el cliente y cómo interactuar con el servicio SOAP en cada uno de estos entornos.

Al final del manual, se dejará el código fuente completo disponible para que los desarrolladores puedan replicar la implementación en su propio entorno. Además, se incluirá un ejemplo en PHP que reproduce la misma funcionalidad que los códigos en C#, permitiendo ilustrar cómo consumir el Web Service SOAP en una página web, permitiendo así ver las diferencias entre las plataformas y cómo adaptarse a distintos entornos de desarrollo.

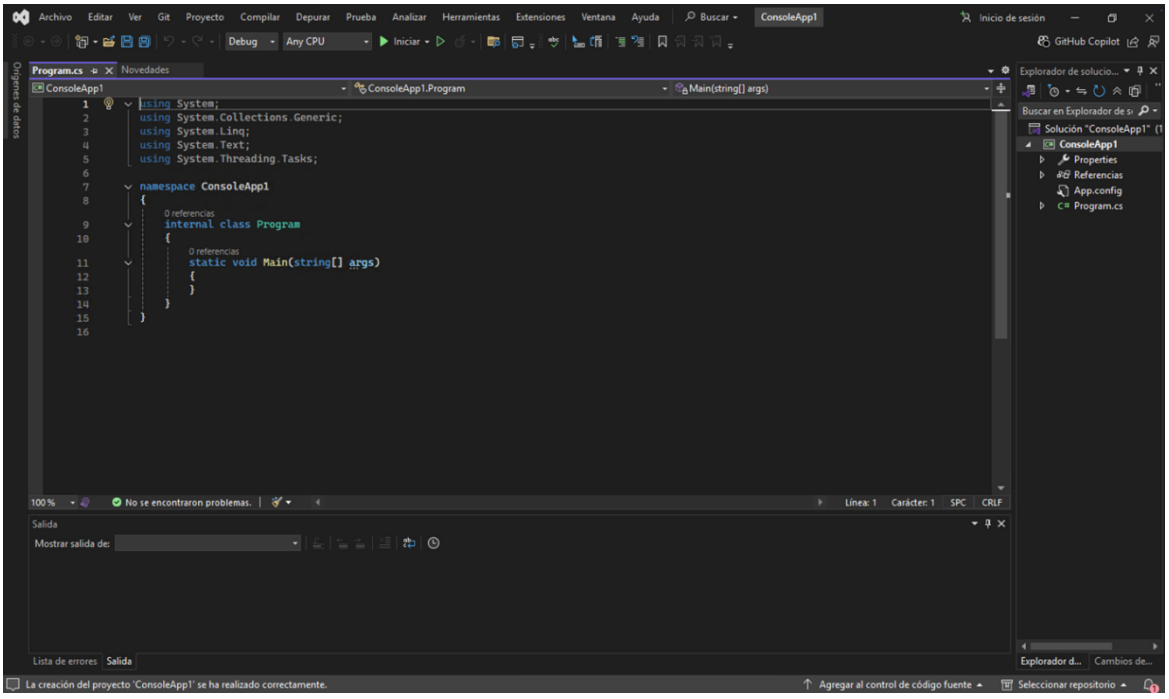
En caso de querer utilizar la API BDE en otro de sus servicios disponibles (librería bcchapi en Python o Web Service REST), consulte el portal de la API BDE para acceder a la documentación oficial.

2. ¿Cómo consumir el Web Service SOAP?

En esta sección, se explicará cómo generar el cliente y consumir el **servicio SOAP de la API BDE** utilizando **C#** en dos entornos: **.NET Framework y .NET 5/6/7/8.** En **.**NET Framework, se trabaja con la implementación tradicional de Web Services a través de Add Web Reference, mientras que en .NET moderno (5/6/7/8), el proceso se adapta mediante WCF Connected Services, lo que genera un cliente diferente que utiliza métodos asíncronos por defecto para una mayor eficiencia en aplicaciones modernas. A continuación, se detallarán los pasos para crear la aplicación, generar el cliente, realizar consultas y manejar errores en Visual Studio 2022.

2.1 Creación de Aplicación

Para comenzar a consumir el servicio SOAP de la API BDE, el primer paso es crear una aplicación en el entorno .NET que prefiera el desarrollador. Puede optar por **.NET Framework** o por una versión moderna como **.NET 5, 6, 7 u 8**, según las necesidades de su proyecto. A continuación, se presenta una aplicación de consola en .NET Framework de referencia.

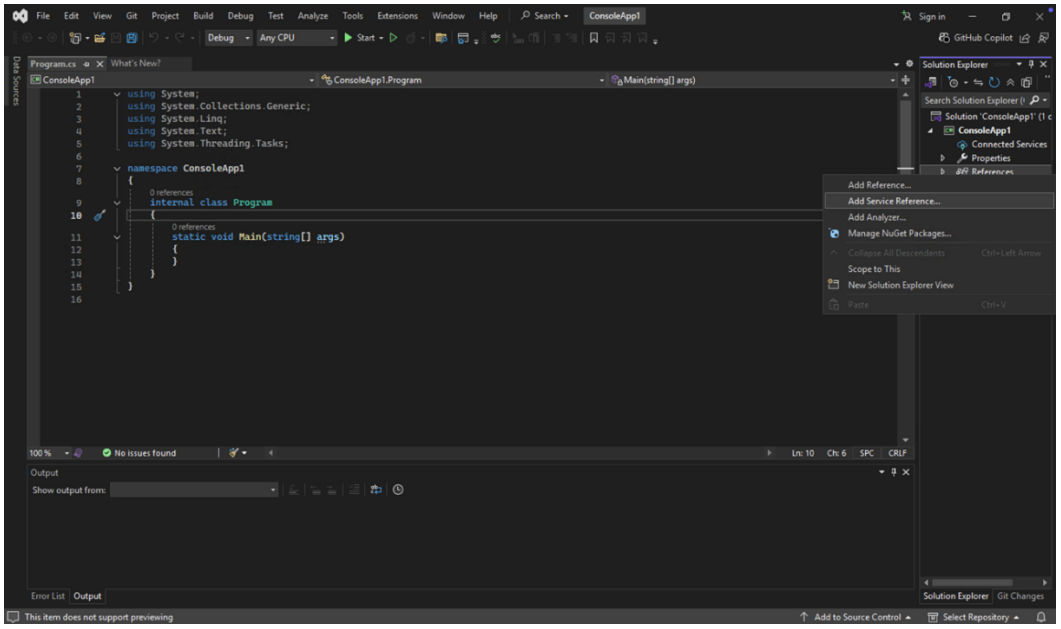


2.2 Generar Cliente SOAP con referencia al Servicio Web

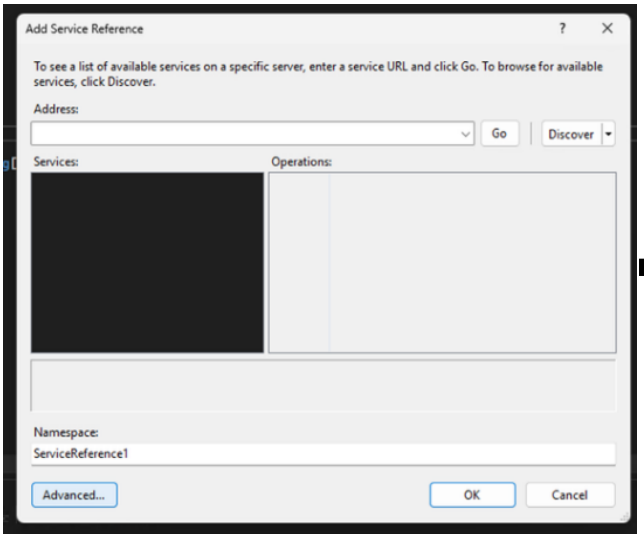
Una vez creada la aplicación en el entorno .NET deseado, el siguiente paso consiste en generar el cliente que permitirá interactuar con el servicio SOAP de la API BDE usando el WSDL del servicio Web. Este proceso varía según el tipo de aplicación utilizado, por lo que en esta sección se presentan los pasos para ambos escenarios: **.NET Framework y .NET 5/6/7/8**.

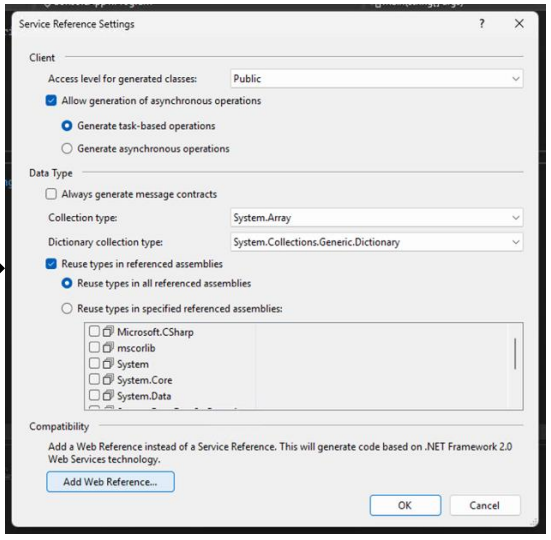
2.2.1 .NET Framework: con referencia a Web Service – Versión de compatibilidad

Paso 1: En el Explorador de soluciones, haga clic derecho sobre References y seleccione Add Service Reference.

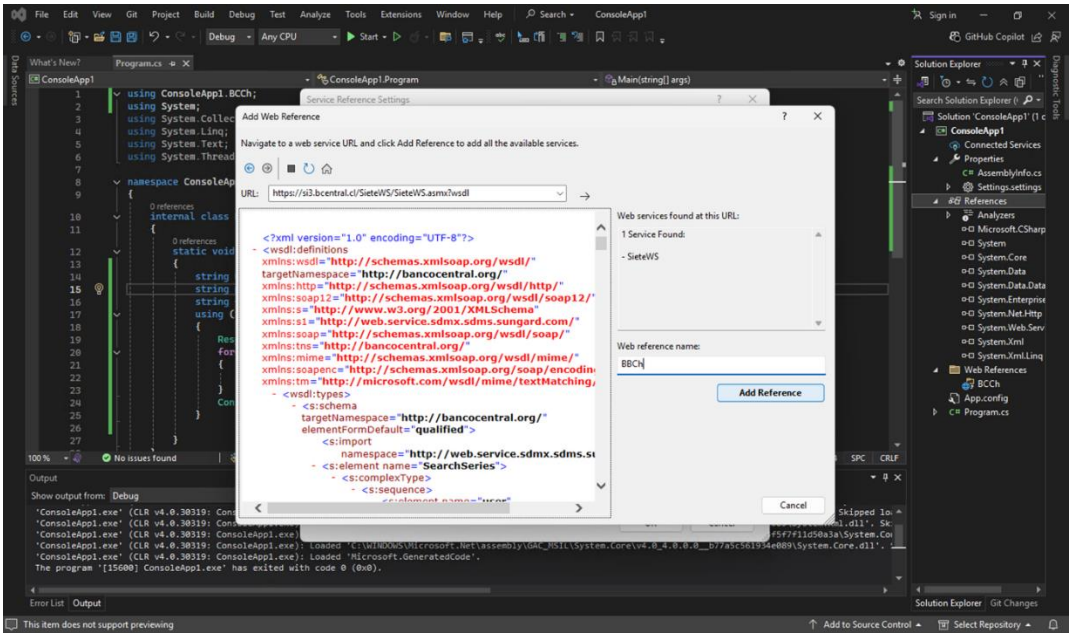


Paso 2: En la ventana emergente, seleccione Advanced → Compatibility → Add Web Reference.



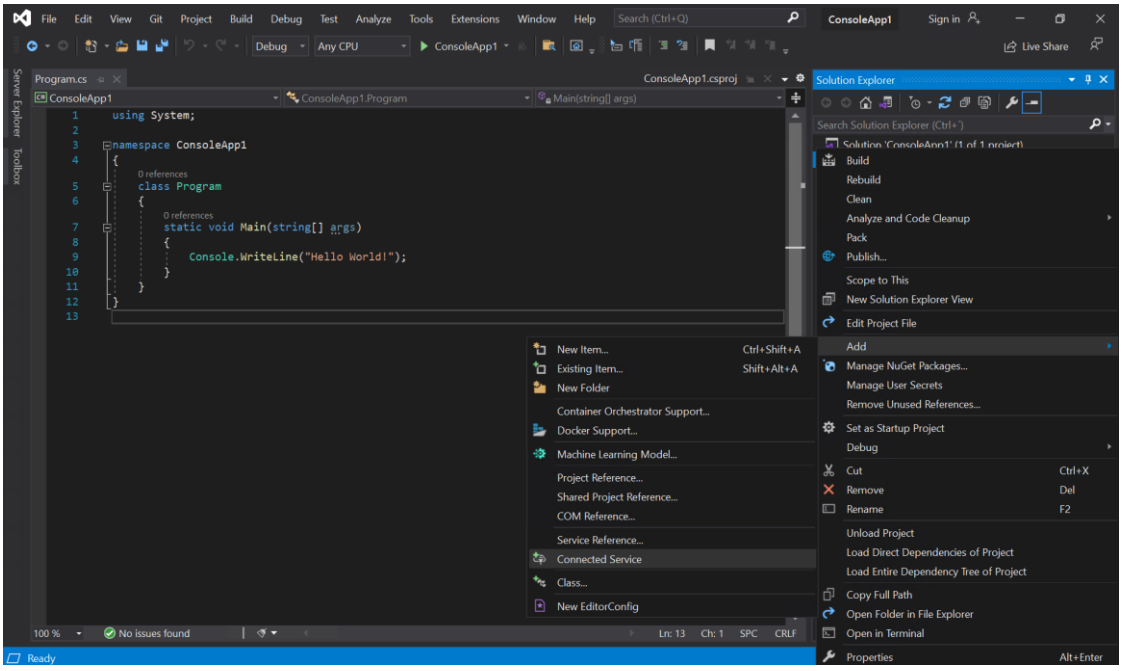


Paso 3: En el campo URL, ingrese la dirección del WSDL. Luego, presione la flecha para cargar el servicio y, una vez visible, defina un nombre para el servicio web y haga clic en Add Reference para empezar a utilizar.

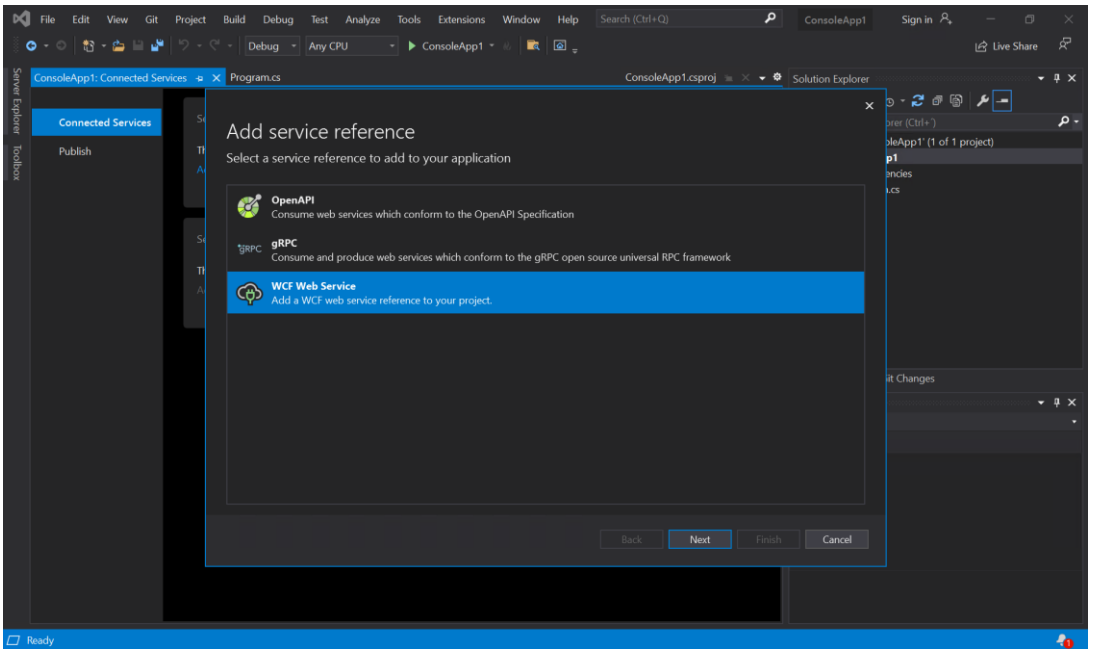


2.2.2 .NET 5/6/7/8: con WCF Connected Services

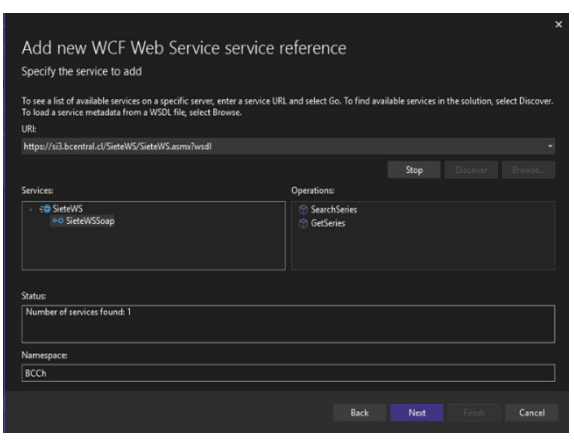
Paso 1: En el Explorador de soluciones, haga clic derecho sobre la consola y seleccione Add → Connected Services.

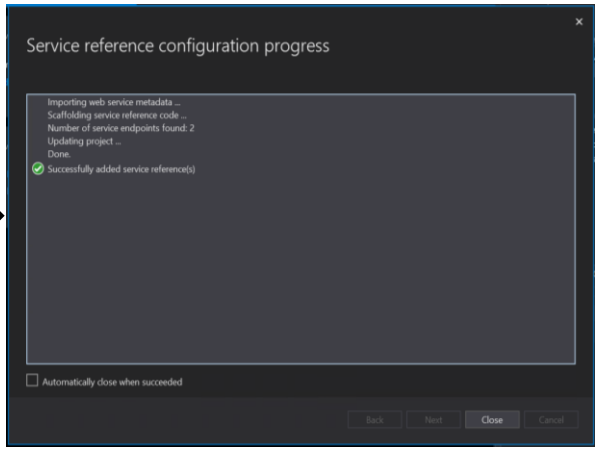


Paso 2: En la ventana de Add service reference, elija **WCF Web Service.**



Paso 3: En el campo URL, ingrese la dirección del WSD y apretar "Go" para identificar el servicio. Una vez identificado, agregue un nombre al servicio web en "Namespace". Luego apretar Next y Finish para que se configure el servicio, una vez salga la señal del éxito ya está listo para utilizar.





Una vez que Visual Studio haya terminado de agregar la referencia y creado las clases necesarias, estas podrán ser utilizadas desde la aplicación de consola.

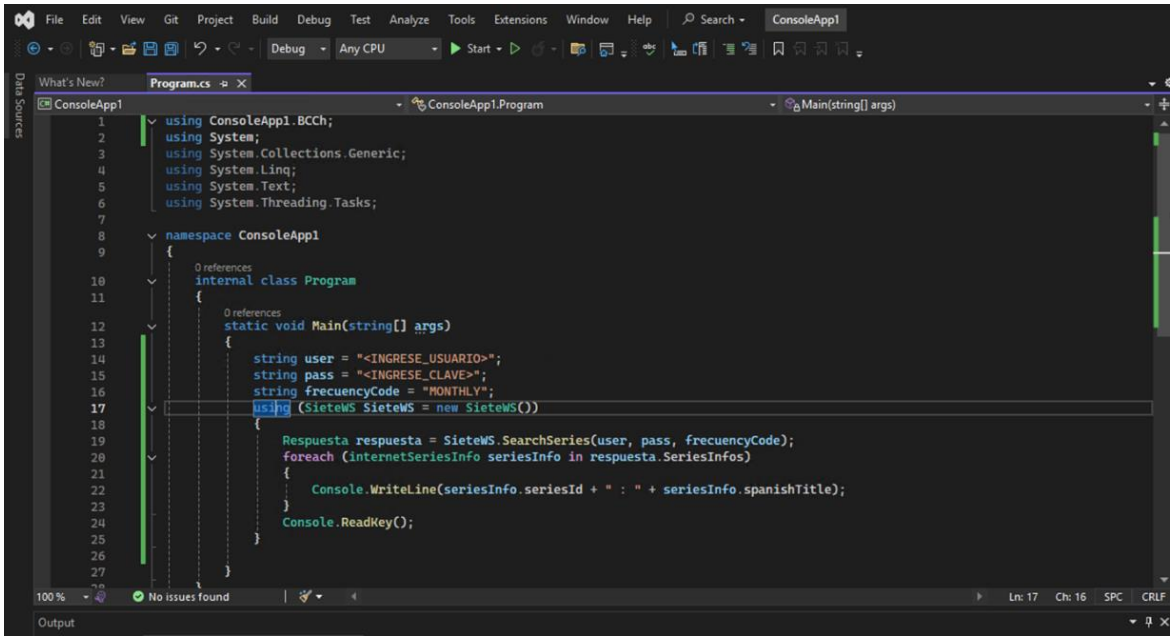
2.3 Buscar las series disponibles: Método SearchSeries

Una vez generado el cliente, ya es posible invocar los métodos disponibles. Para consultar las series disponibles en la API BDE, se utiliza el método **SearchSeries**, el cual recibe tres parámetros *string:*

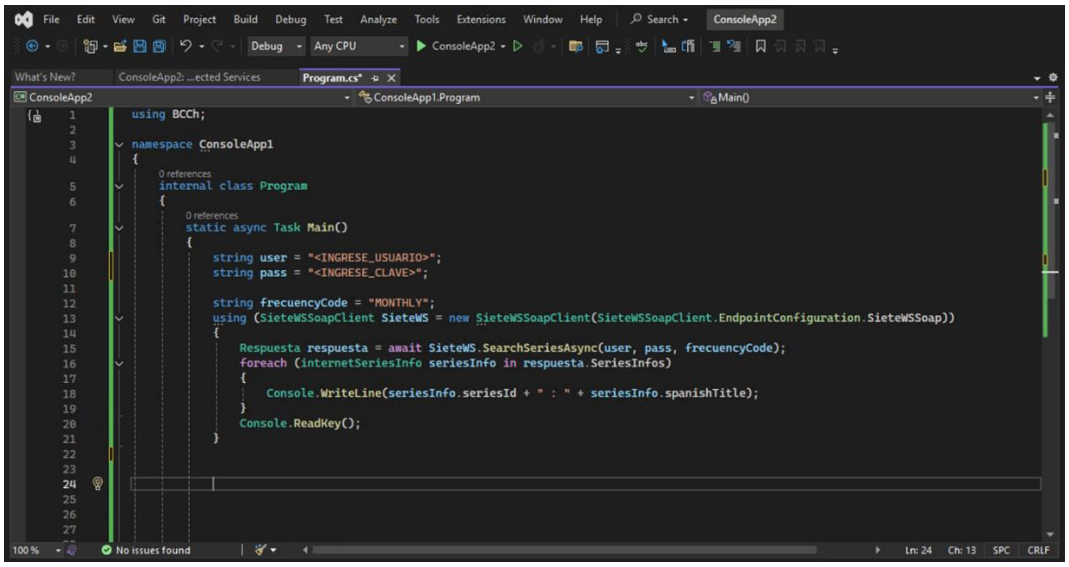
- **user:** correo electrónico registrado en la BDE.
- **password:** contraseña utilizada para ingresar a la BDE.
- **frecuencyCode**: filtro para limitar el resultado a las series con una frecuencia específica. Las opciones válidas para este parámetro son **DAILY, MONTHLY, QUARTERLY y ANNUAL.**

En las siguientes imágenes se muestra un ejemplo de código para obtener las series que tienen datos con la frecuencia MONTHLY, haciendo énfasis en las diferencias de código entre **.NET Framework y .NET 5/6/7/8**.

En **.NET Framework**, el método Main es sincrónico y la llamada al servicio se realiza directamente, donde para instanciar al cliente se ocupa: "SieteWS SieteWS = new SieteWS()" como se ve en la imagen.



En cambio, en **versiones modernas de .NET,** el método Main debe ser asíncrono y la llamada al servicio utiliza await y métodos terminados en Async, mientras que para instanciar el cliente se ocupa: "SieteWSSoapClient SieteWS = new SieteWSSoapClient (SieteWSSoapClient.EndpointConfiguration.SieteWSSoap))".

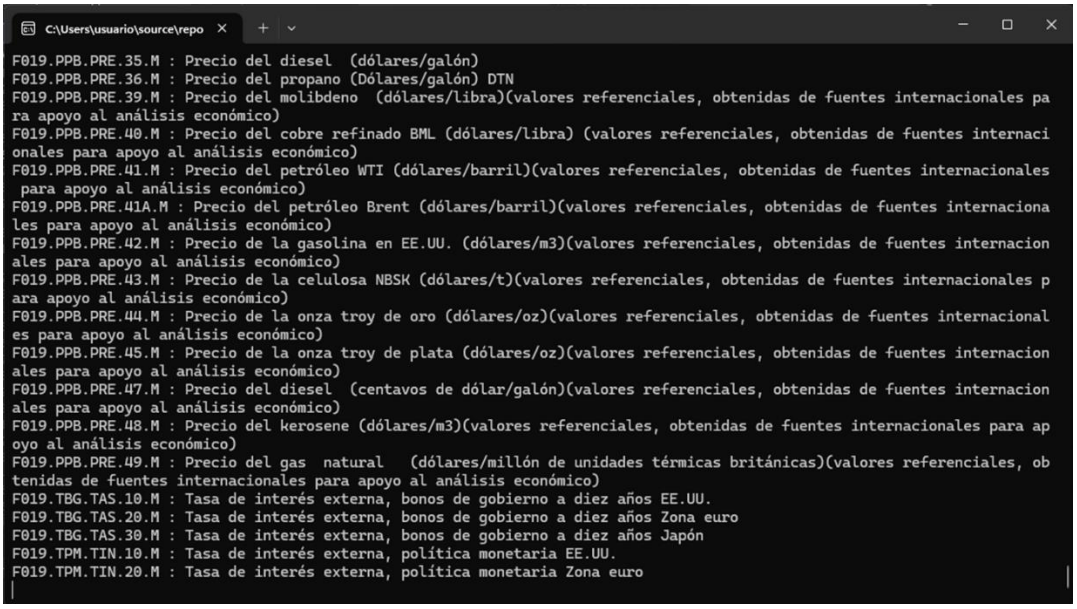


```
using BCCh;

namespace ConsoleApp1
{
    0 references
    internal class Program
    {
        0 references
        static async Task Main()
        {
            string user = "<INGRESE_USUARIO>";
            string pass = "<INGRESE_CLAVE>";

            string frecuencyCode = "MONTHLY";
            using (SieteWSSoapClient SieteWS = new SieteWSSoapClient(SieteWSSoapClient.EndpointConfiguration.SieteWSSoap))
            {
                Respuesta respuesta = await SieteWS.SearchSeriesAsync(user, pass, frecuencyCode);
                foreach (internetSeriesInfo seriesInfo in respuesta.SeriesInfos)
                {
                    Console.WriteLine(seriesInfo.seriesId + " : " + seriesInfo.spanishTitle);
                }
                Console.ReadKey();
            }
```

La ejecución de cualquiera de estos códigos despliega el listado de series que se observa a continuación, donde se despliega el ID de la serie y la descripción en español.



```
C:\Users\usuario\source\repo
F019.PPB.PRE.35.M : Precio del diesel  (dólares/galón)
F019.PPB.PRE.36.M : Precio del propano (Dólares/galón) DTN
F019.PPB.PRE.39.M : Precio del molibdeno  (dólares/libra)(valores referenciales, obtenidas de fuentes internacionales pa
ra apoyo al análisis económico)
F019.PPB.PRE.40.M : Precio del cobre refinado BML (dólares/libra) (valores referenciales, obtenidas de fuentes internaci
onales para apoyo al análisis económico)
F019.PPB.PRE.41.M : Precio del petróleo WTI (dólares/barril)(valores referenciales, obtenidas de fuentes internacionales
 para apoyo al análisis económico)
F019.PPB.PRE.41A.M : Precio del petróleo Brent (dólares/barril)(valores referenciales, obtenidas de fuentes internaciona
les para apoyo al análisis económico)
F019.PPB.PRE.42.M : Precio de la gasolina en EE.UU. (dólares/m3)(valores referenciales, obtenidas de fuentes internacion
ales para apoyo al análisis económico)
F019.PPB.PRE.43.M : Precio de la celulosa NBSK (dólares/t)(valores referenciales, obtenidas de fuentes internacionales p
ara apoyo al análisis económico)
F019.PPB.PRE.44.M : Precio de la onza troy de oro (dólares/oz)(valores referenciales, obtenidas de fuentes internacional
es para apoyo al análisis económico)
F019.PPB.PRE.45.M : Precio de la onza troy de plata (dólares/oz)(valores referenciales, obtenidas de fuentes internacion
ales para apoyo al análisis económico)
F019.PPB.PRE.47.M : Precio del diesel  (centavos de dólar/galón)(valores referenciales, obtenidas de fuentes internacion
ales para apoyo al análisis económico)
F019.PPB.PRE.48.M : Precio del kerosene (dólares/m3)(valores referenciales, obtenidas de fuentes internacionales para ap
oyo al análisis económico)
F019.PPB.PRE.49.M : Precio del gas  natural  (dólares/millón de unidades térmicas británicas)(valores referenciales, ob
tenidas de fuentes internacionales para apoyo al análisis económico)
F019.TBG.TAS.10.M : Tasa de interés externa, bonos de gobierno a diez años EE.UU.
F019.TBG.TAS.20.M : Tasa de interés externa, bonos de gobierno a diez años Zona euro
F019.TBG.TAS.30.M : Tasa de interés externa, bonos de gobierno a diez años Japón
F019.TPM.TIN.10.M : Tasa de interés externa, política monetaria EE.UU.
F019.TPM.TIN.20.M : Tasa de interés externa, política monetaria Zona euro
```

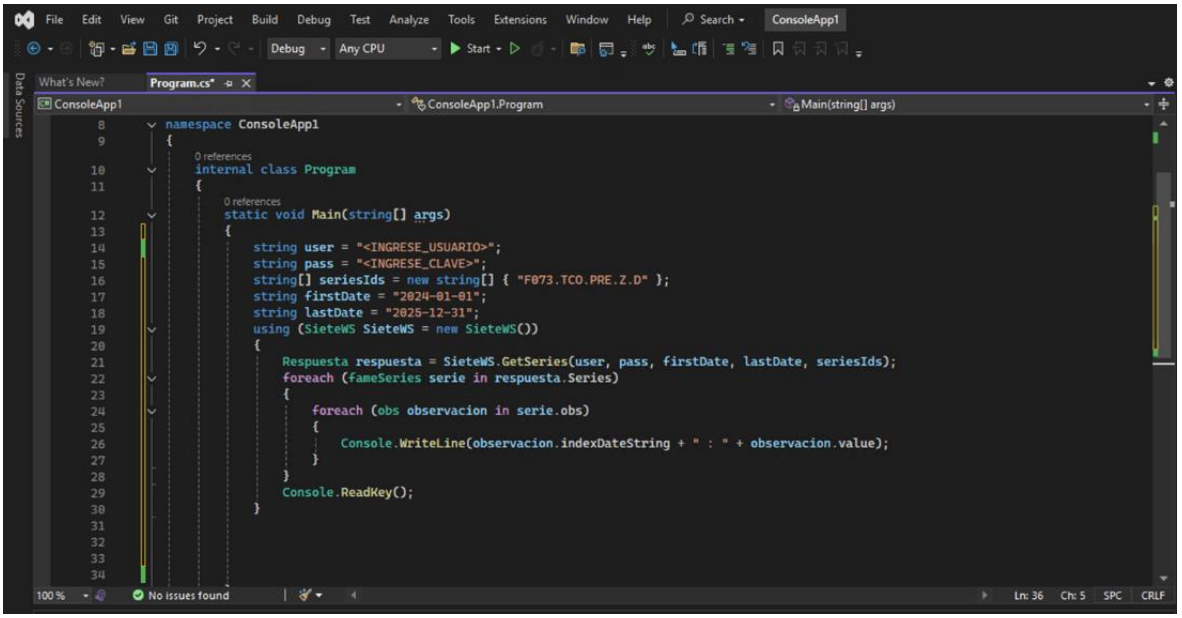
2.4 Obtener las observaciones de una serie: Método GetSeries

Para obtener las observaciones para una serie se dispone de un método llamado **GetSeries**, el cual recibe cinco parámetros *string*. Estos parámetros son:

- **user:** correo electrónico registrado en la BDE.
- **password:** contraseña utilizada para ingresar a la BDE.
- **firstDate (opcional):** filtro para acotar las observaciones a partir de una fecha de inicio. El formato debe ser YYYY-MM-DD.
- **lastDate (opcional):** filtro para acotar las observaciones hasta una fecha de término. El formato debe ser YYYY-MM-DD.
- **seriesIds:** filtro para indicar el código de la serie de la que se quieren obtener las observaciones. Si bien el parámetro es un arreglo, **solo se puede consultar una serie por cada llamada del método**.

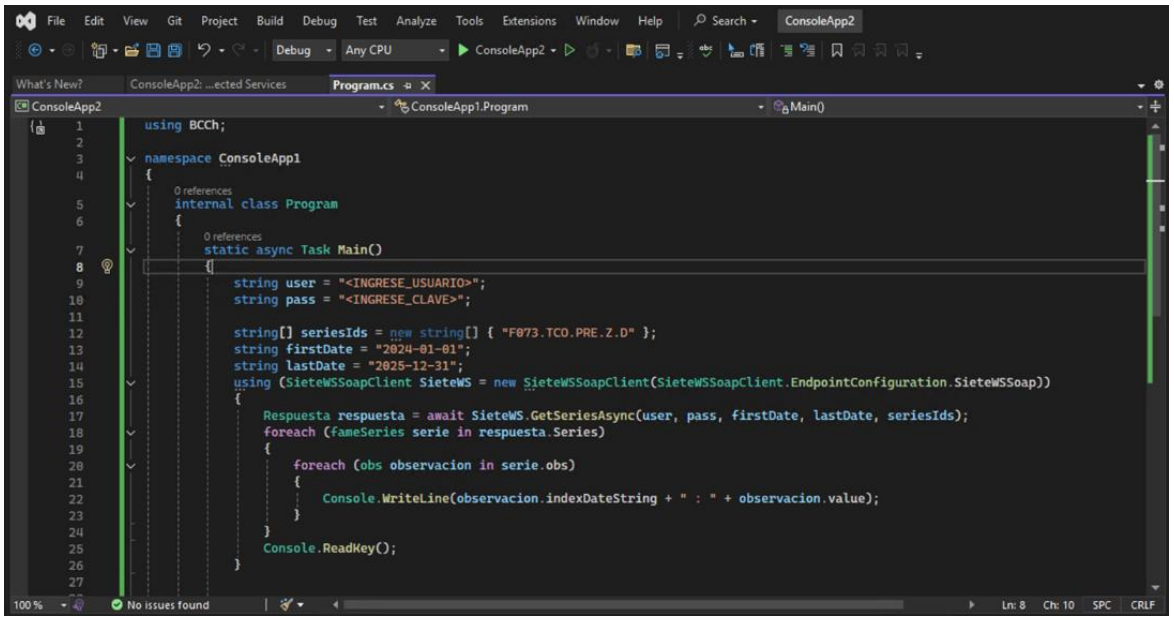
Por ejemplo, si se desean obtener las observaciones de la serie del dólar, se debe utilizar el ID de ésta, el cual corresponde a "F073.TCO.PRE.Z.D". Si se quieren obtener todas las del año 2024 y 2025, se deberá llamar el servicio web como se muestra en el código de más abajo.

En el caso de **.NET Framework**, se mantienen las mismas características de invocación sincrónica explicadas en el aparatado anterior.



```
namespace ConsoleApp1
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            string user = "<INGRESE_USUARIO>";
            string pass = "<INGRESE_CLAVE>";
            string[] seriesIds = new string[] { "F073.TCO.PRE.Z.D" };
            string firstDate = "2024-01-01";
            string lastDate = "2025-12-31";
            using (SieteWS SieteWS = new SieteWS())
            {
                Respuesta respuesta = SieteWS.GetSeries(user, pass, firstDate, lastDate, seriesIds);
                foreach (fameSeries serie in respuesta.Series)
                {
                    foreach (obs observacion in serie.obs)
                    {
                        Console.WriteLine(observacion.indexDateString + " : " + observacion.value);
                    }
                }
                Console.ReadKey();
            }
```

Por otro lado, en el caso de **.NET moderno**, la invocación es con métodos asincrónicos.



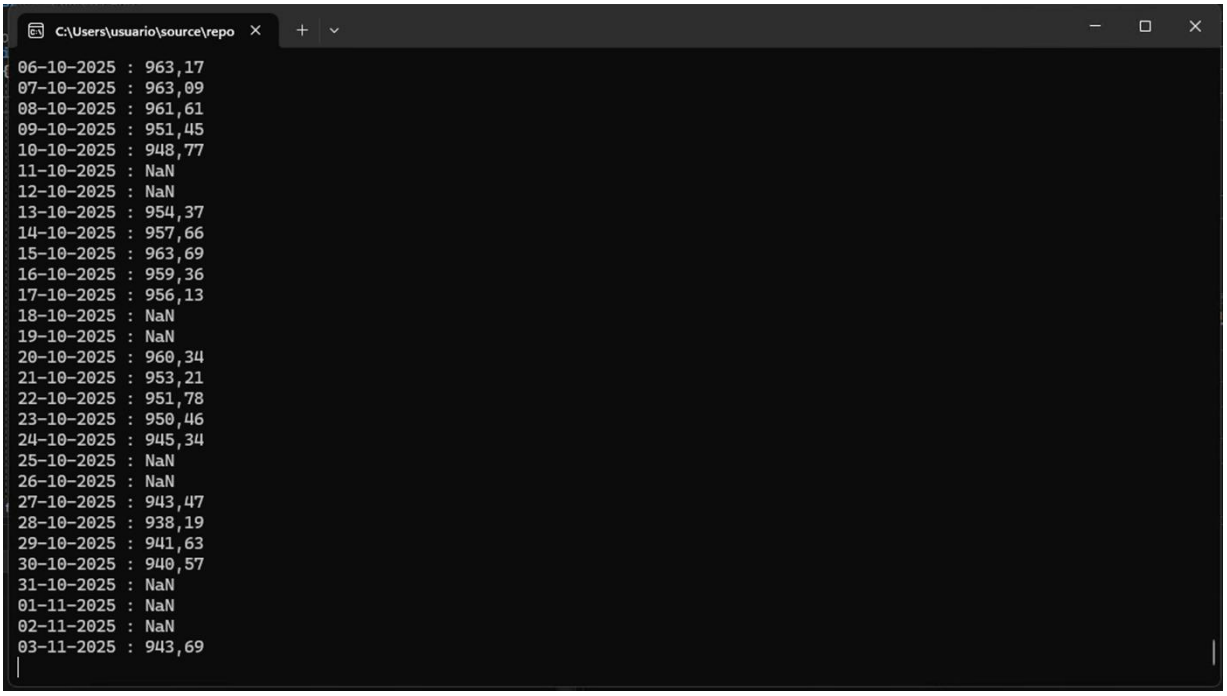
```using BCCh;

namespace ConsoleApp1
{
    0 references
    internal class Program
    {
        0 references
        static async Task Main()
        {
            string user = "<INGRESE_USUARIO>";
            string pass = "<INGRESE_CLAVE>";

            string[] seriesIds = new string[] { "F073.TCO.PRE.Z.D" };
            string firstDate = "2024-01-01";
            string lastDate = "2025-12-31";
            using (SieteWSSoapClient SieteWS = new SieteWSSoapClient(SieteWSSoapClient.EndpointConfiguration.SieteWSSoap))
            {
                Respuesta respuesta = await SieteWS.GetSeriesAsync(user, pass, firstDate, lastDate, seriesIds);
                foreach (fameSeries serie in respuesta.Series)
                {
                    foreach (obs observacion in serie.obs)
                    {
                        Console.WriteLine(observacion.indexDateString + " : " + observacion.value);
                    }
                }
                Console.ReadKey();
            }
```

En el código se observa que es necesario hacer un ciclo foreach dentro de otro ciclo foreach. El primer ciclo recorre selecciona la serie solicitada y el segundo ciclo recorre las observaciones de esta serie.

La ejecución de este código da el resultado desplegado a continuación, en donde destaca la fecha de la observación, una por día dado que es una serie diaria (DAILY), y el valor de la observación.

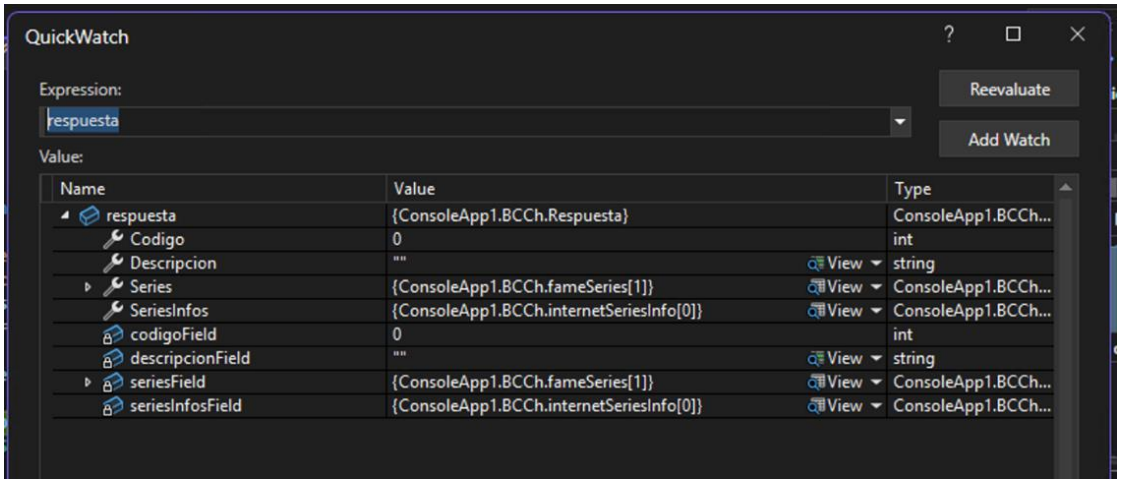


```C:\Users\usuario\source\repo
06-10-2025 : 963,17
07-10-2025 : 963,09
08-10-2025 : 961,61
09-10-2025 : 951,45
10-10-2025 : 948,77
11-10-2025 : NaN
12-10-2025 : NaN
13-10-2025 : 954,37
14-10-2025 : 957,66
15-10-2025 : 963,69
16-10-2025 : 959,36
17-10-2025 : 956,13
18-10-2025 : NaN
19-10-2025 : NaN
20-10-2025 : 960,34
21-10-2025 : 953,21
22-10-2025 : 951,78
23-10-2025 : 950,46
24-10-2025 : 945,34
25-10-2025 : NaN
26-10-2025 : NaN
27-10-2025 : 943,47
28-10-2025 : 938,19
29-10-2025 : 941,63
30-10-2025 : 940,57
31-10-2025 : NaN
01-11-2025 : NaN
02-11-2025 : NaN
03-11-2025 : 943,69
```

2.5 Control de Errores

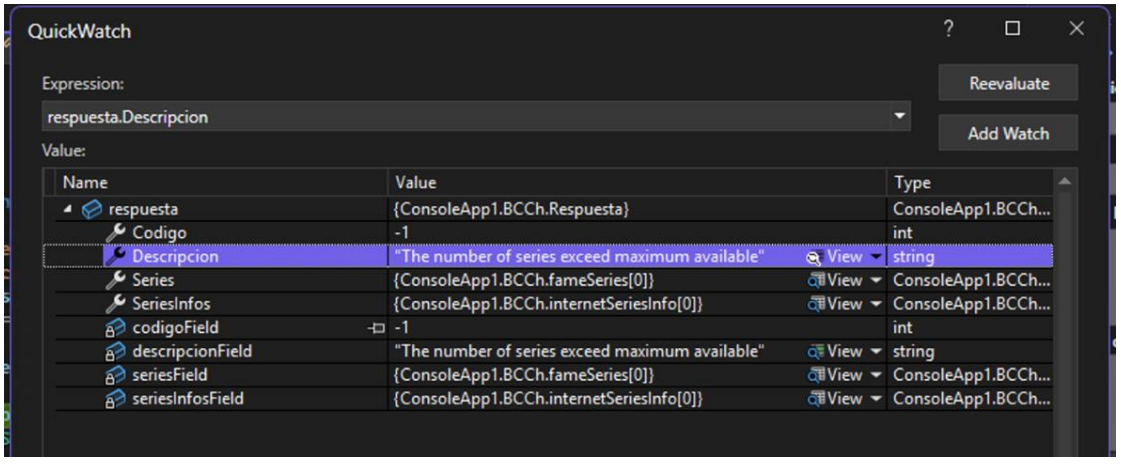
A continuación, se mostrará cómo poder inspeccionar el código en caso de presentar un error al invocar alguno de los métodos mostrados anteriormente. Para poder revisar si la operación fue realizada con éxito o no, es necesario revisar los atributos **"Codigo" y "Descripcion" del objeto "Respuesta"** que guarda el resultado retornado por los métodos.

En caso de éxito, el "Codigo" de Respuesta será 0 y "Descripcion" estará vacía, indicando que el método retornó una respuesta esperada. En la siguiente imagen se puede observar los atributos del objeto "Respuesta" en este caso.

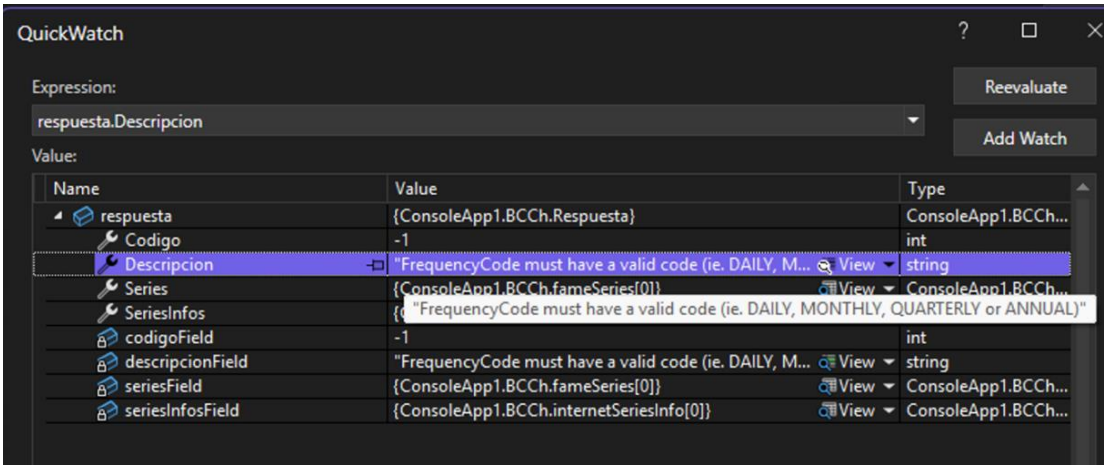


En caso de error, "Codigo" tomará valores distintos de 0 dependiendo del problema, y en "Descripción" se detallará la causa de error. A continuación, mostraremos cómo se ven los atributos del objeto "Respuesta" en ciertos errores frecuentes.

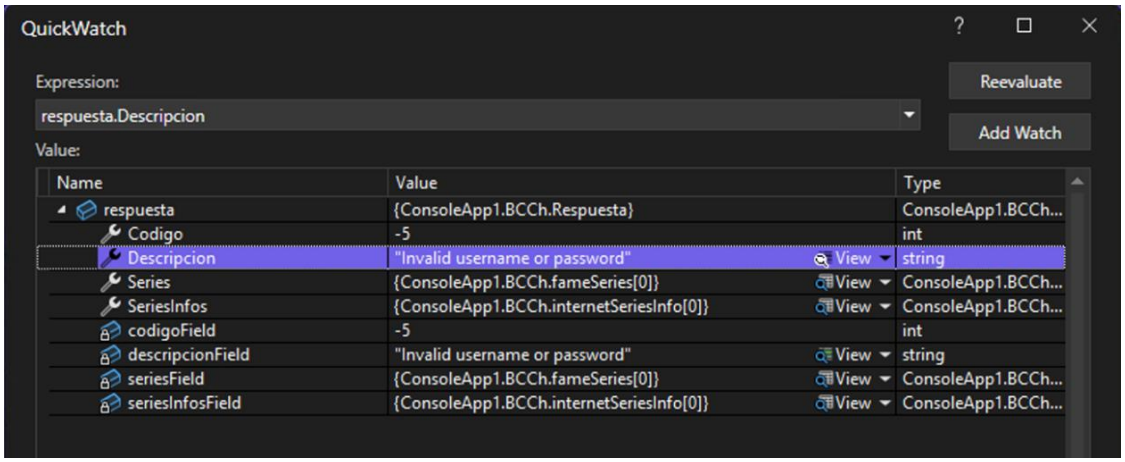
1. **Consultar más de una serie a la vez:** como se mencionó en la sección anterior, al ocupar el método GetSeries solo se puede consultar cómo máximo una serie. En caso de agregar más de un código de serie al arreglo seriesIds, "Codigo" será -1 y la "Descripcion" dirá: "The number of series exceed máximum available".



2. **Ingresar una frecuencia inválida:** el método SearchSeries solo admite las frecuencias DAILY, MONTHLY, QUATERLY y ANNUAL. En caso de ingresar algo distinto a esto en frecuencyCode, "Codigo" será -1 y la "Descripcion" dirá: "FrecuencyCode must have a valid code".



3. **Error en credenciales de acceso:** en caso de ingresar mal el usuario o contraseña, "Codigo" será -5 y la "Descripción dirá: "Invalid username or password".



3. Anexo: Código Fuente

En esta sección se presentan los ejemplos completos de código para consumir el servicio SOAP de la API BDE en los entornos .NET Framework y .NET 5/6/7/8 de C#. De manera adicional, se presenta un código en PHP que reproduce la misma funcionalidad que los ejemplos presentados en C#, para que así los desarrolladores puedan comparar como consumir el servicio SOAP en página web.

3.1 Código en C# - .NET Framework

```
using ConsoleApp1.BCCh;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string user = "<INGRESE_USUARIO>";
            string pass = "<INGRESE_CLAVE>";

            string frecuencyCode = "MONTHLY";
            using (SieteWS SieteWS = new SieteWS())
            {
                Respuesta respuesta = SieteWS.SearchSeries(user, pass, frecuencyCode);
                foreach (internetSeriesInfo seriesInfo in respuesta.SeriesInfos)
                {
                    Console.WriteLine(seriesInfo.seriesId + " : " + seriesInfo.spanishTitle);
                }
                Console.ReadKey();
            }
            string[] seriesIds = new string[] { " F073.TCO.PRE.Z.D "};
            string firstDate = "2024-01-01";
            string lastDate = "2025-12-31";
            using (SieteWS SieteWS = new SieteWS())
            {
                Respuesta respuesta = SieteWS.GetSeries(user, pass, firstDate, lastDate, seriesIds);
                foreach (fameSeries serie in respuesta.Series)
                {
                    foreach (obs observacion in serie.obs)
                    {
                        Console.WriteLine(observacion.indexDateString + " : " + observacion.value);
                    }
                }
                Console.ReadKey();
            }
        }
    }
}
```

3.2 Código en C# - .NET 5/6/7/8

```
using BCCh;

namespace ConsoleApp1
{
    internal class Program
    {
        static async Task Main()
        {
            string user = "<INGRESE_USUARIO>";
            string pass = "<INGRESE_CLAVE>";

            string frecuencyCode = "MONTHLY";
            using (SieteWSSoapClient SieteWS = new
SieteWSSoapClient(SieteWSSoapClient.EndpointConfiguration.SieteWSSoap))
            {
                Respuesta respuesta = await SieteWS.SearchSeriesAsync(user, pass, frecuencyCode);
                foreach (internetSeriesInfo seriesInfo in respuesta.SeriesInfos)
                {
                    Console.WriteLine(seriesInfo.seriesId + " : " + seriesInfo.spanishTitle);
                }
                Console.ReadKey();
            }
            string[] seriesIds = new string[] { "F073.TCO.PRE.Z.D" };
            string firstDate = "2024-01-01";
            string lastDate = "2025-12-31";
            using (SieteWSSoapClient SieteWS = new
SieteWSSoapClient(SieteWSSoapClient.EndpointConfiguration.SieteWSSoap))
            {
                Respuesta respuesta = await SieteWS.GetSeriesAsync(user, pass, firstDate, lastDate,
seriesIds);
                foreach (fameSeries serie in respuesta.Series)
                {
                    foreach (obs observacion in serie.obs)
                    {
                        Console.WriteLine(observacion.indexDateString + " : " + observacion.value);
                    }
                }
                Console.ReadKey();
            }
        }
    }
}
```

3.3 Extra: Código en PHP

```
<?
$user = '<INGRESE_USUARIO>';
$password='<INGRESE_CLAVE>';
$frequencyCode ='MONTHLY';

$wsdl="<DIRECCION_SITIO_WEB>?WSDL";

$client = new soapclient($wsdl);
$params = new stdClass;
$params->user = $user;
$params->password = $password;
$params->frequencyCode = $frequencyCode;
$result = $client->SearchSeries($params)->SearchSeriesResult;

foreach ($result->SeriesInfos->internetSeriesInfo as $serie)
{
      echo $serie->seriesId . " : " . $serie->spanishTitle . "<br/>";
}

$seriesIds = array ("F073.TCO.PRE.Z.D");
$firstDate = "2024-01-01";
$lastDate = "2025-12-31";

$client = new soapclient($wsdl);
$params = new stdClass;
$params->user = $user;
$params->password = $password;
$params->firstDate = $firstDate;
$params->lastDate = $lastDate;
$params->seriesIds = $seriesIds;

$result = $client->GetSeries($params)->GetSeriesResult;

$fameSeries =$result->Series->fameSeries;

//Cuando se solicita una sola serie, la respuesta no es interpretada como un arreglo

if (is_array($fameSeries ) != 1) $fameSeries = array($fameSeries);

foreach ($fameSeries as $serieFame)
{
      echo $serieFame->seriesKey->seriesId . " : " . $serieFame->precision . "<br/>";
      foreach ($serieFame->obs as $obs)
      {
            echo $obs->indexDateString . " : " . $obs->value . "<br/>";
      }
}
?>
```