



**UNIVERSIDAD ISRAEL**

**CIENCIAS DE LA INGENIERÍA**

**CARRERA DE SISTEMAS DE INFORMACIÓN**

**PLATAFORMAS DE DESARROLLO 2**

**SEMESTRE 2021 A**

**LABORATORIO S3**

**TEMA: SOAP**

**PROFESOR:** Mg. Luis Fernando Aguas Bucheli

**QUITO, 2021**

#### 1. TEMA: SOAP

#### 2. OBJETIVOS:

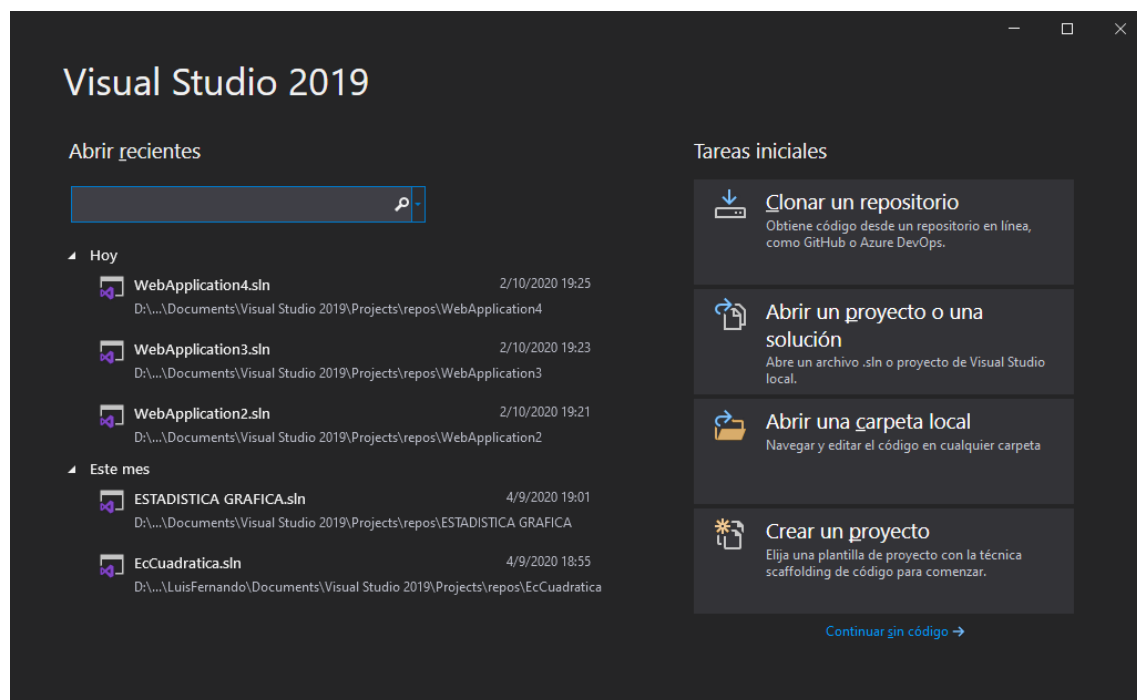
- Adquirir los conceptos básicos relacionados con SOAP
- Reconocer las características de SOAP

#### 3. INTRODUCCION:

SOAP es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros. Está actualmente bajo el auspicio de la W3C.

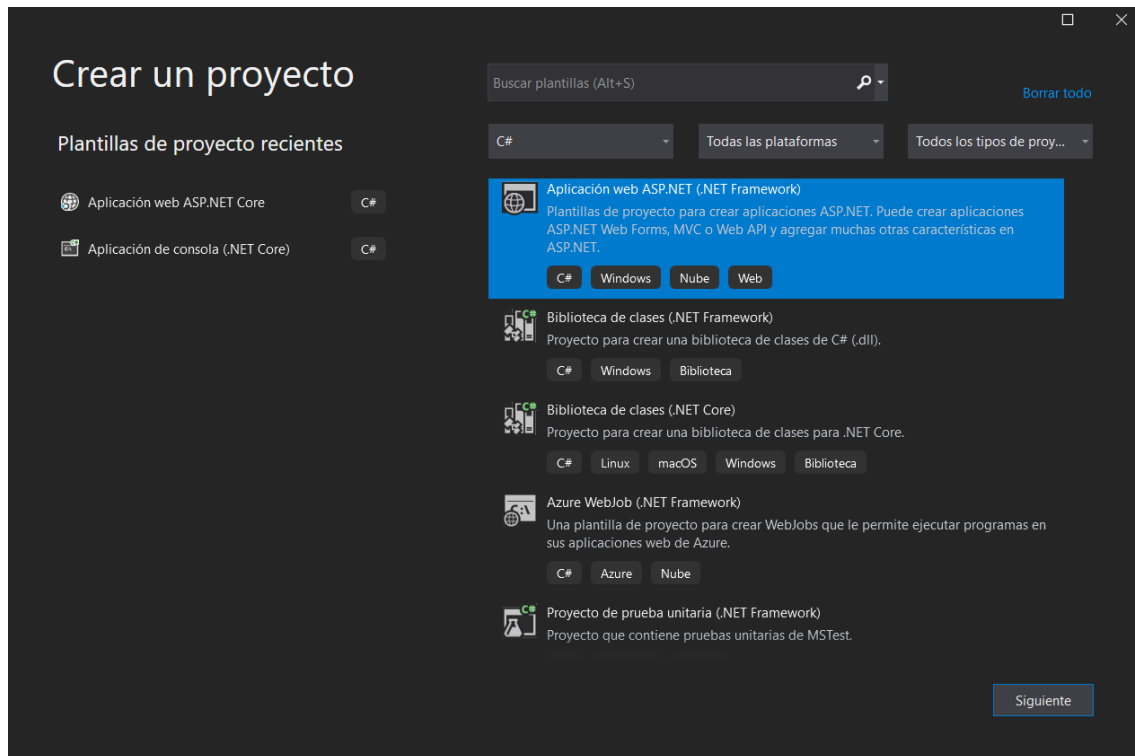
#### 4. DESARROLLO:

Ingresamos al Visual Studio

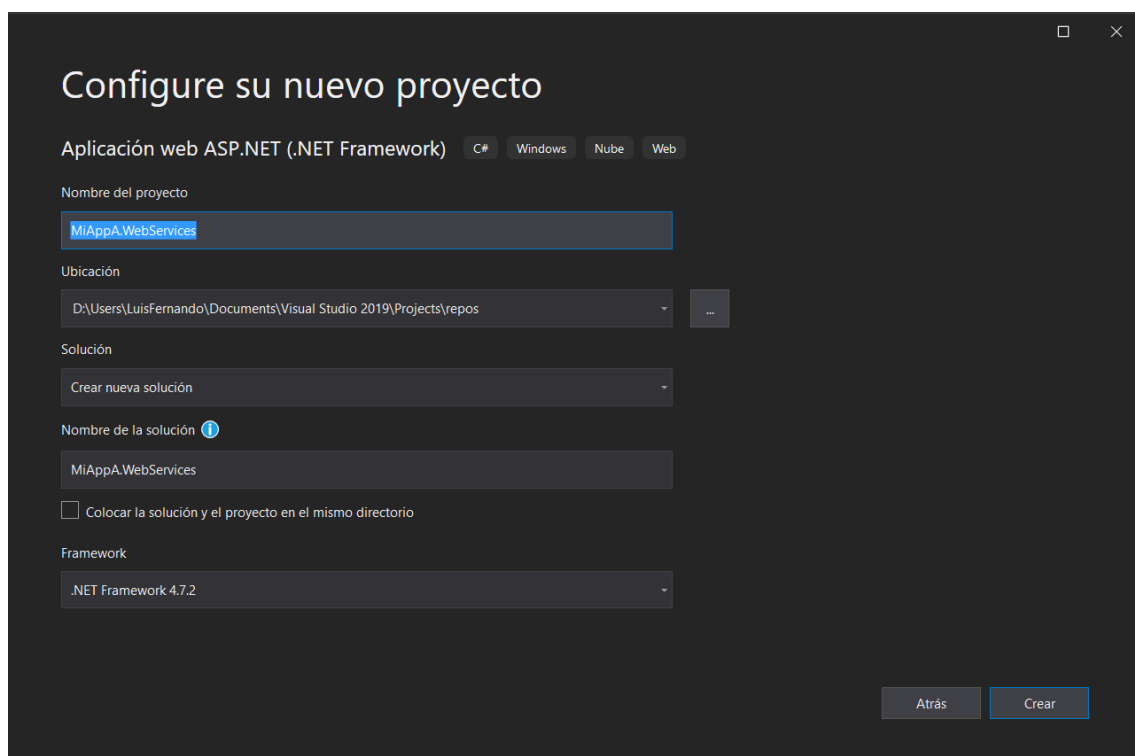


Seleccionamos crear un proyecto y escogemos

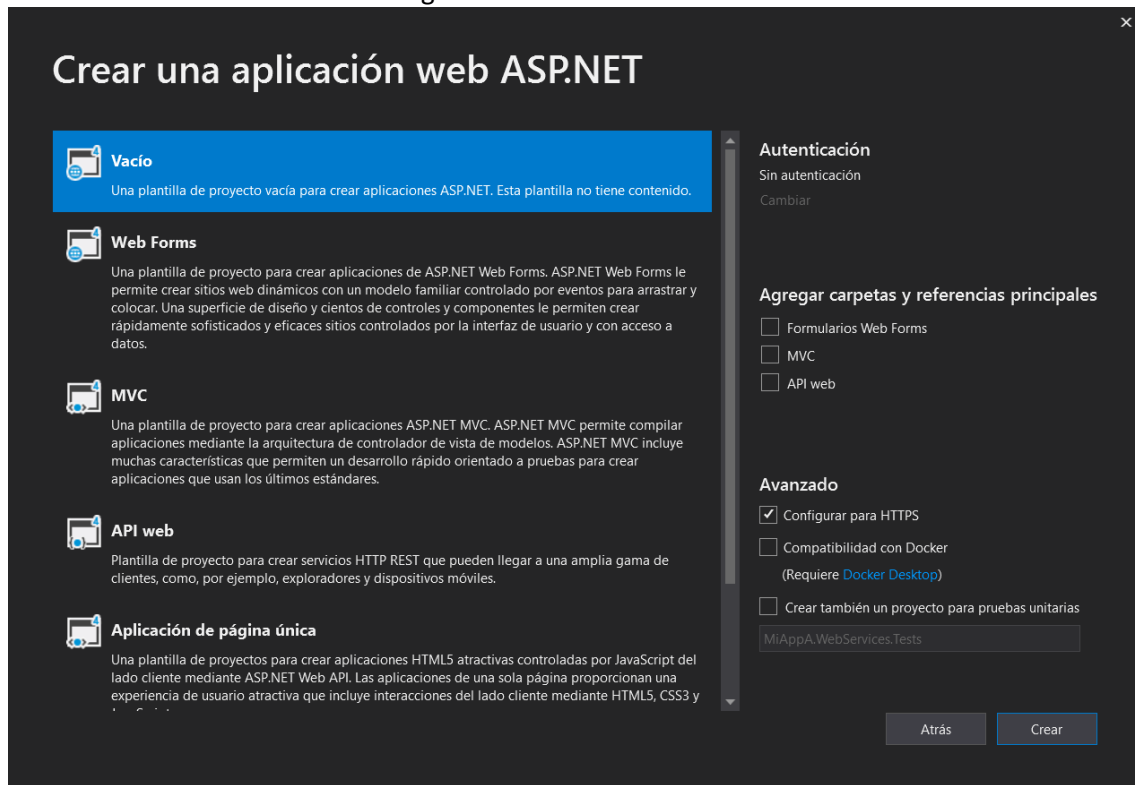




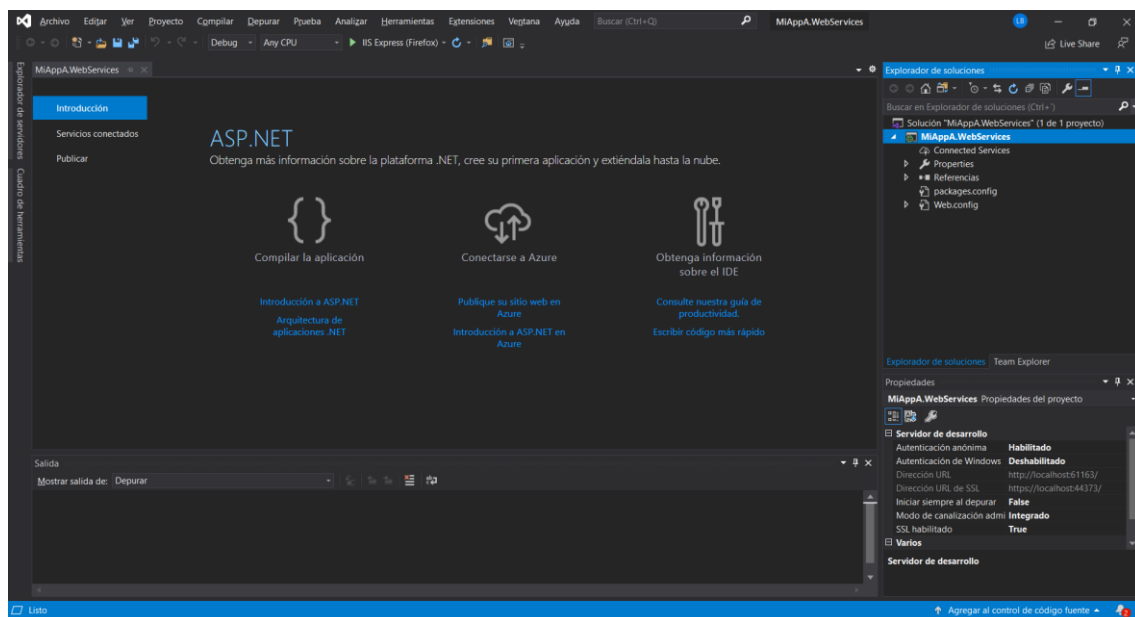
En primer lugar, crearemos un nuevo proyecto del tipo Aplicación Web ASP.NET, con el nombre MiAppA.WebServices



Seleccionamos a continuación lo siguiente

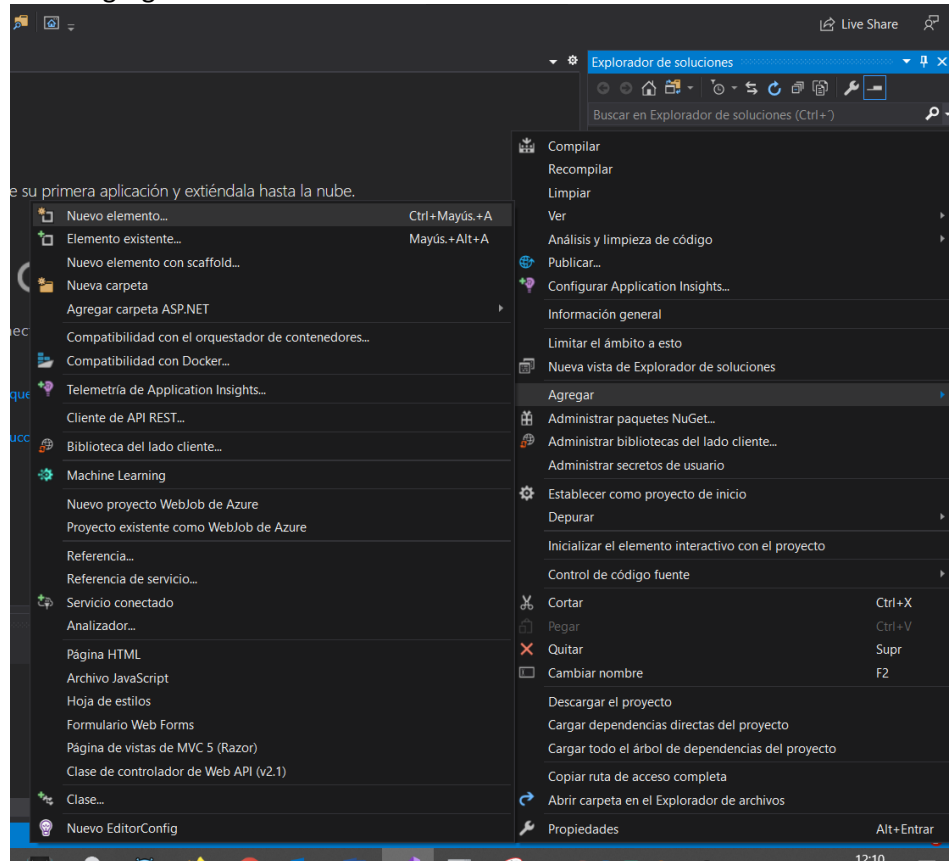


Damos clic en crear:

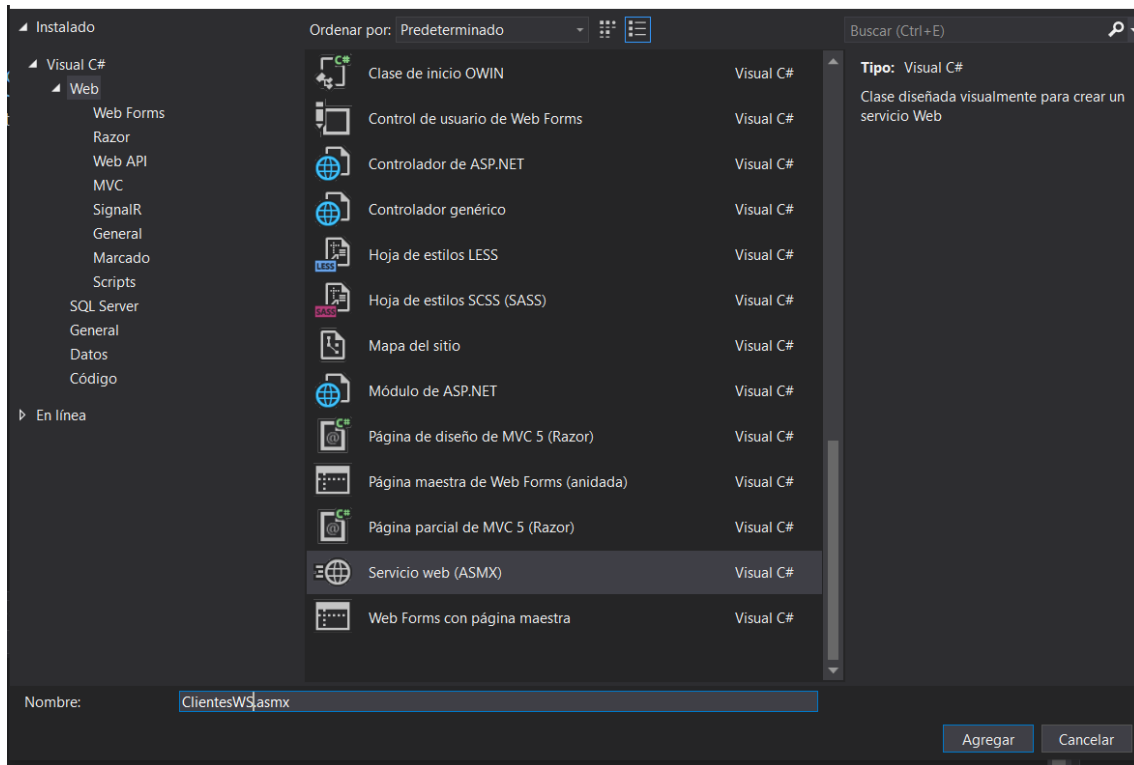


Seguidamente, agregaremos al proyecto un nuevo elemento del tipo Servicio web (ASMX) con el nombre ClientesWS.asmx. Este será el Web Service sobre el que desarrollaremos con posterioridad.

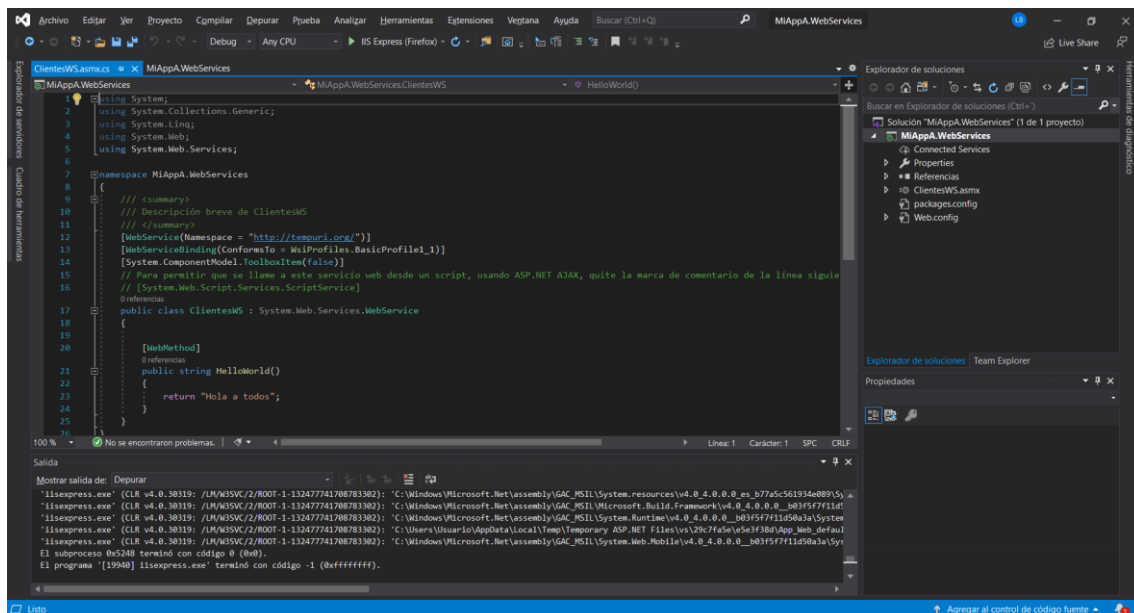
Damos clic en agregar nuevo elemento



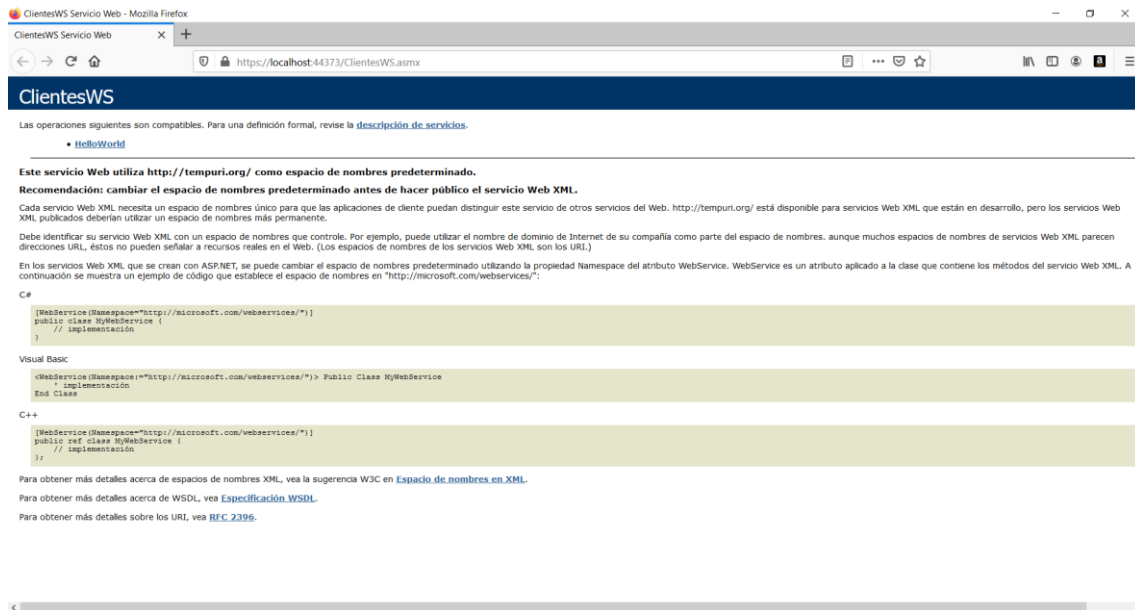
Luego seleccionamos lo siguiente y colocamos el nombre



Y luego damos clic en aceptar

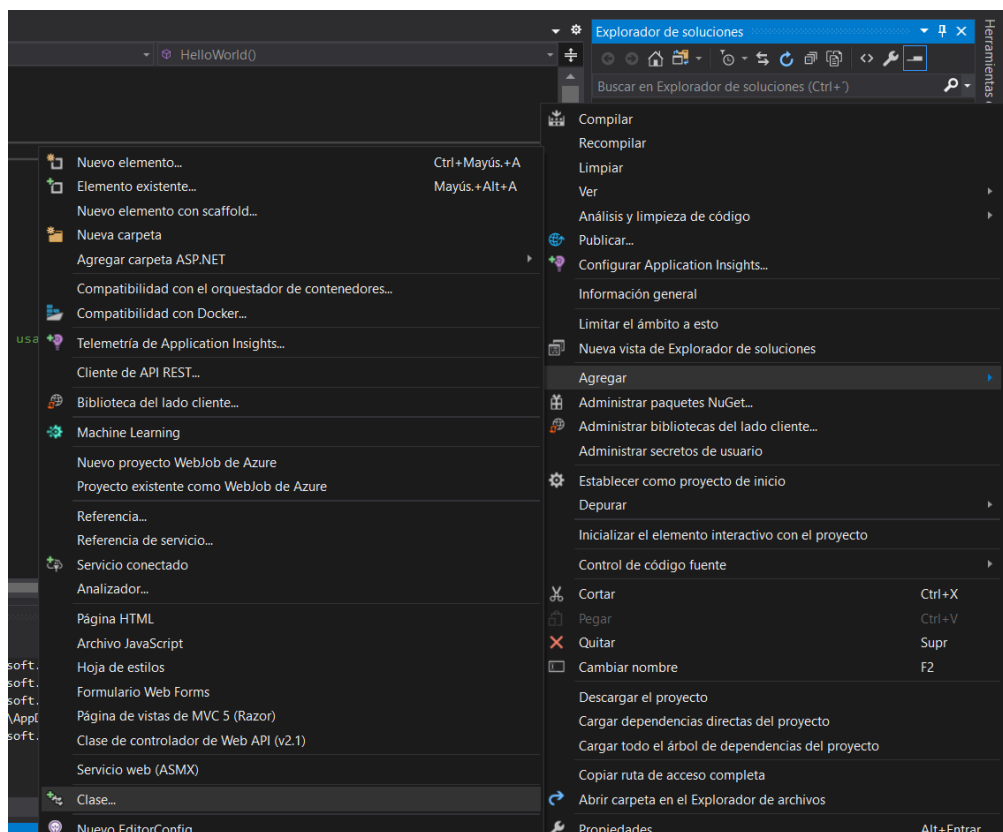


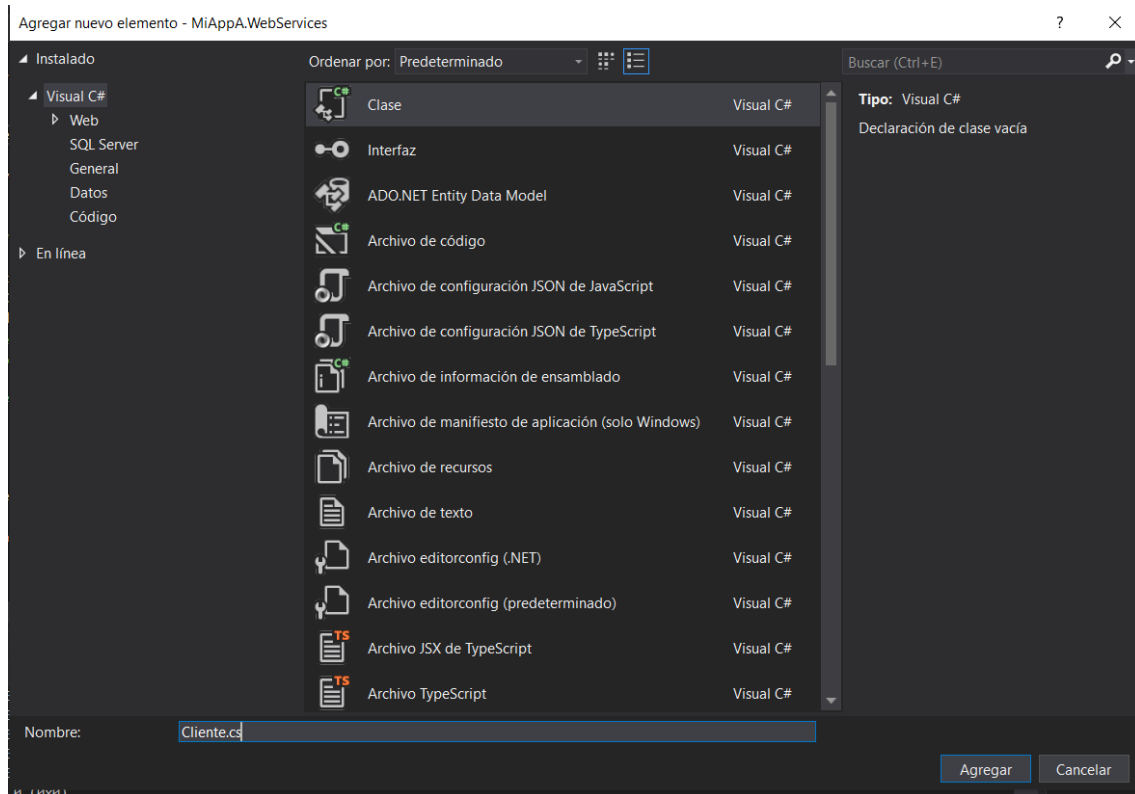
Compilamos y ejecutamos para verificar que el servidor de aplicaciones este en buen estado



Antes de comenzar a desarrollar nuestro Web Service SOAP, debemos definir los Modelos de datos sobre los que trabajaremos posteriormente.

Primero crearemos una nueva clase llamada Cliente.cs, la cual será la base para construir una lista de clientes que con posterioridad devolveremos desde el Servicio web a través del método GetClientes().

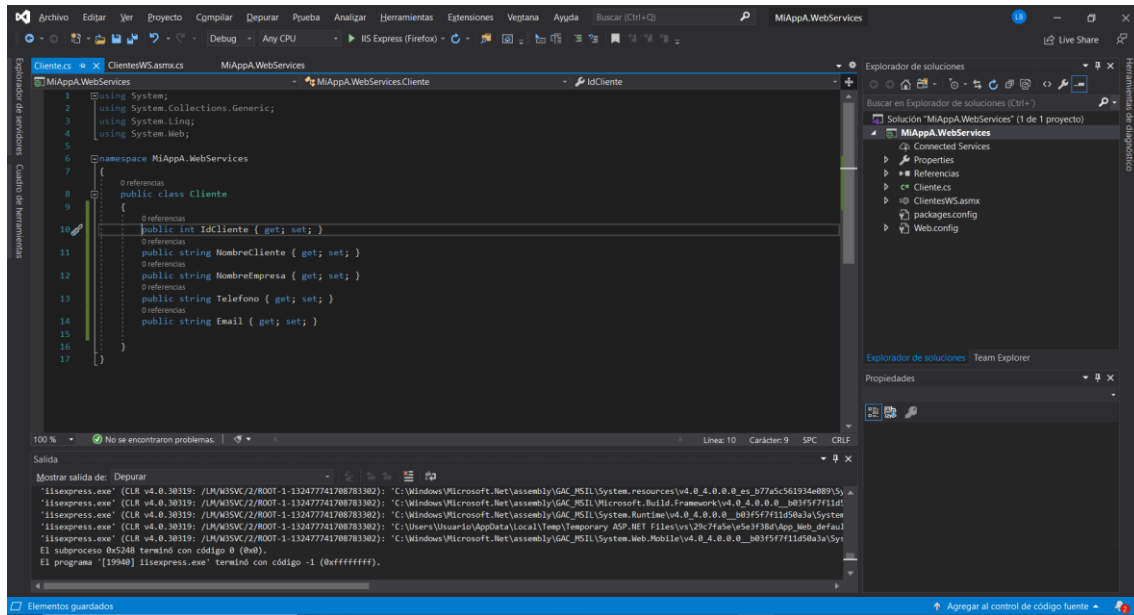




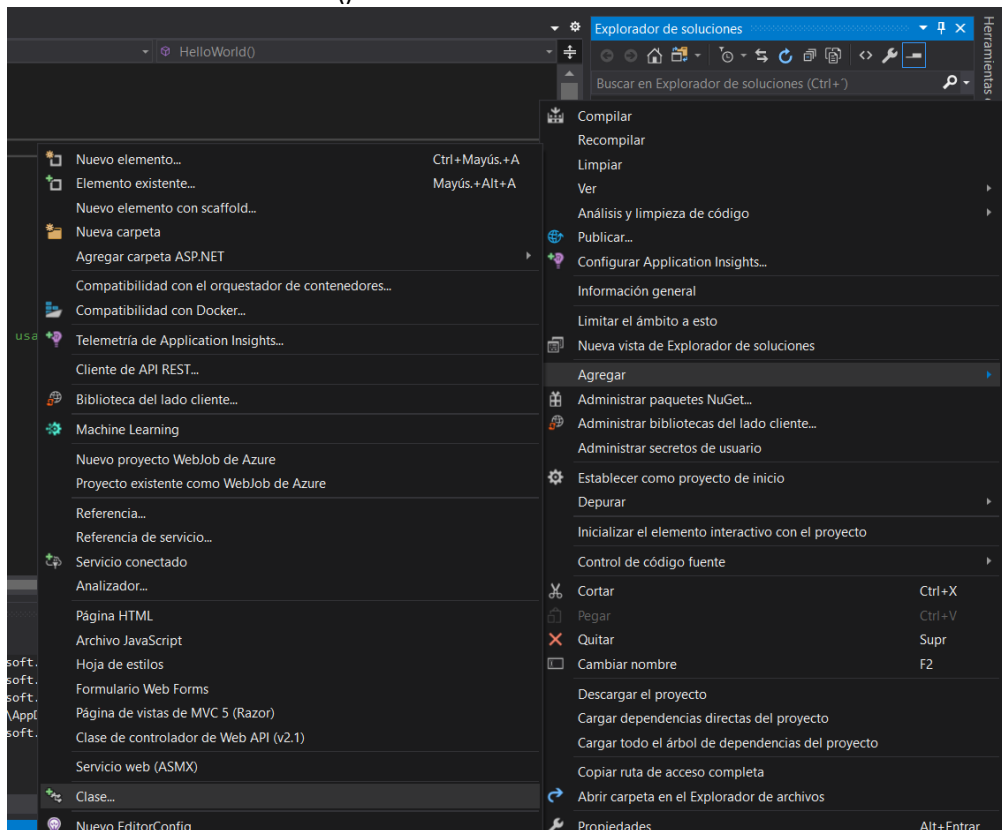
Damos clic en agregar y colocamos el siguiente código:

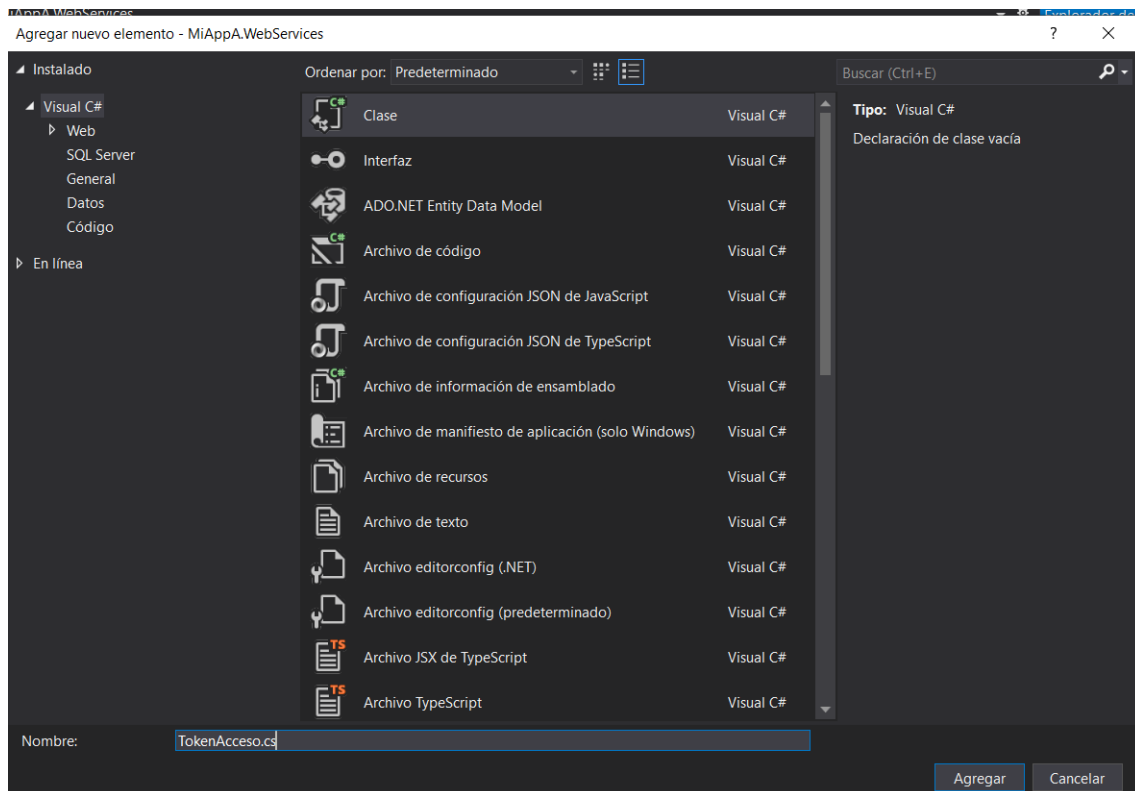
```
public class Cliente
{
    public int IdCliente { get; set; }
    public string NombreCliente { get; set; }
    public string NombreEmpresa { get; set; }
    public string Telefono { get; set; }
    public string Email { get; set; }
}
```





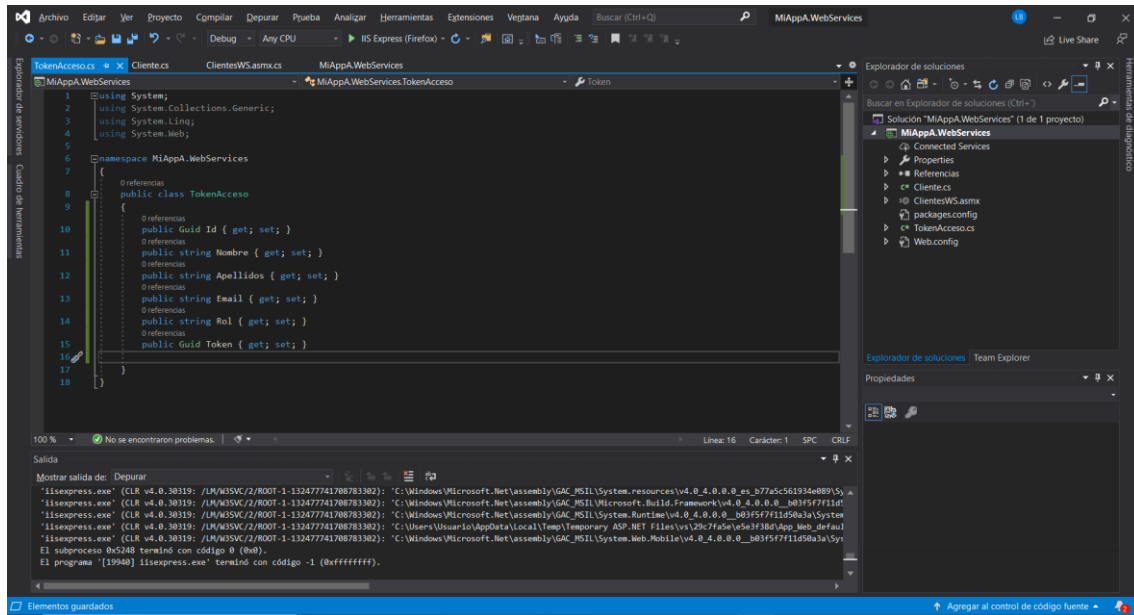
También crearemos la clase `TokenAcceso.cs`, la cual contendrá además del propio Token de acceso, los datos identificativos del usuario que lo solicita. Este modelo de datos, lo devolverá el método `GetTokenAcceso()`.





Damos clic en agregar y colocamos el siguiente código:

```
public class TokenAcceso
{
    public Guid Id { get; set; }
    public string Nombre { get; set; }
    public string Apellidos { get; set; }
    public string Email { get; set; }
    public string Rol { get; set; }
    public Guid Token { get; set; }
}
```



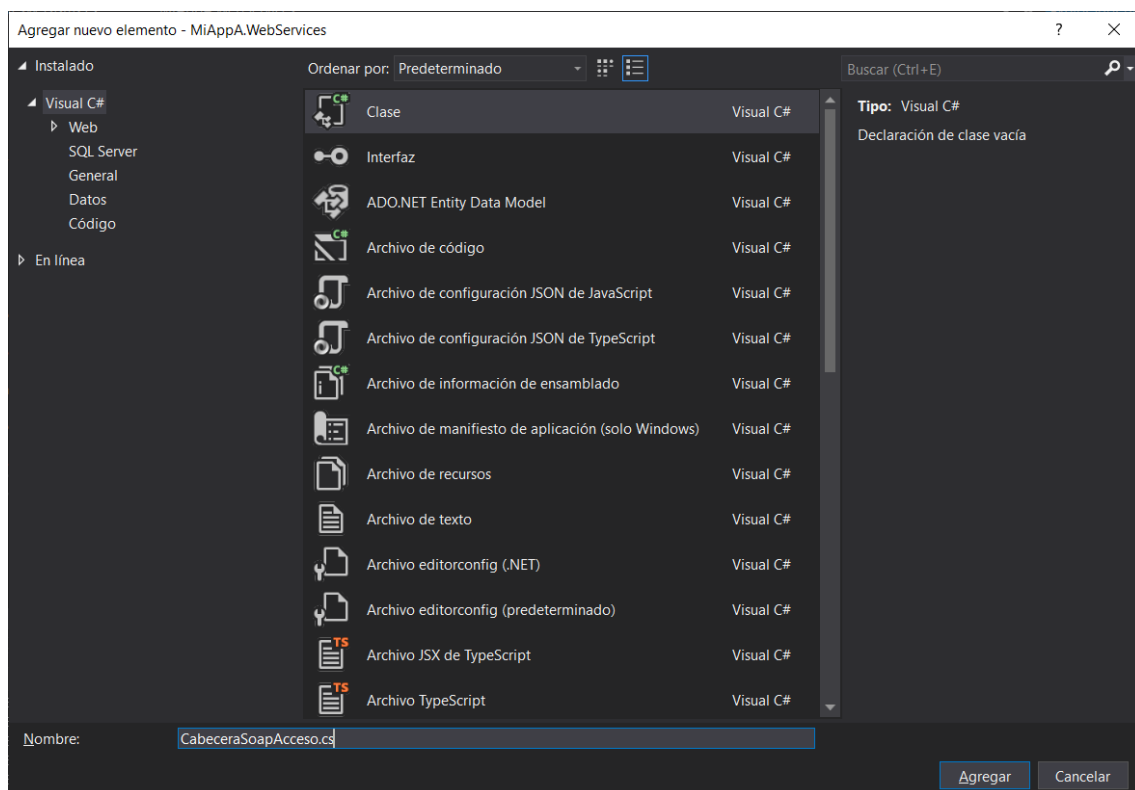
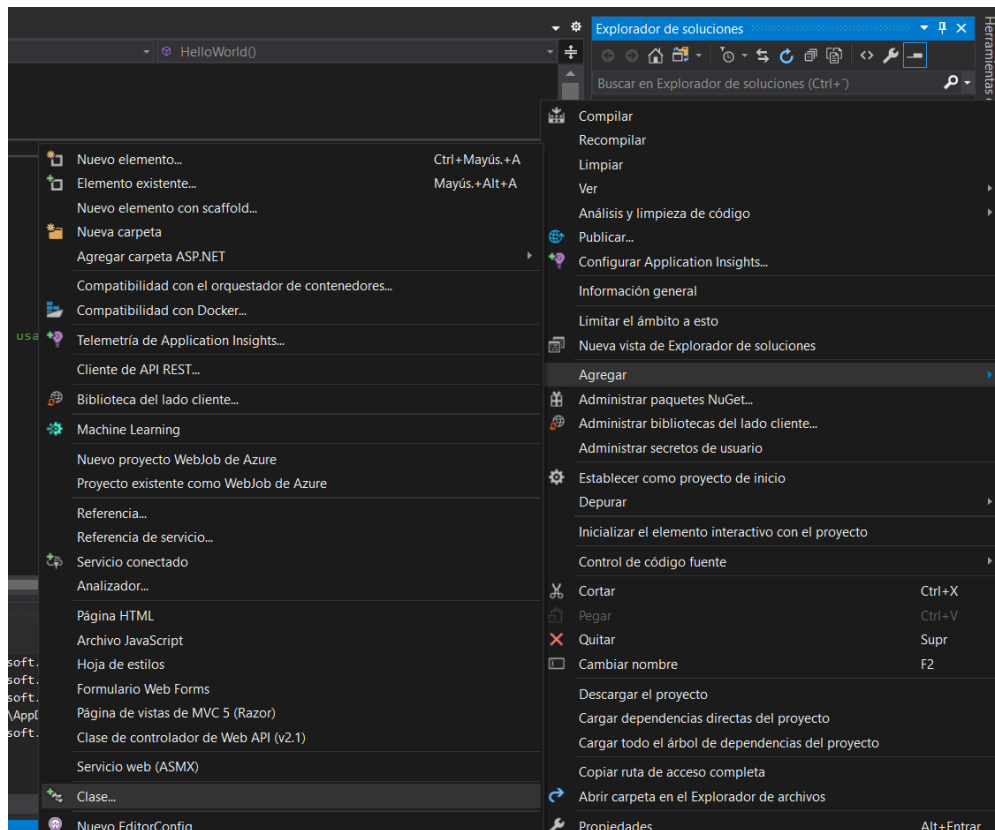
Como ya seguramente sabemos, SOAP es un estándar que nos permite definir la estructura de un documento XML que será enviado o recibido por un Servicio Web por medio del protocolo HTTP.

La estructura de este documento XML consta básicamente de un cuerpo (Body), donde irán alojados los datos tanto de ida como vuelta, y una cabecera (Header), que también permite incluir datos en las solicitudes.

Siguiendo esta estructura, utilizaremos la cabecera (Header) para enviar al Web Service todos los datos referentes a la seguridad, ya sea tanto el usuario y contraseña de autenticación como el propio Token de acceso a los servicios.

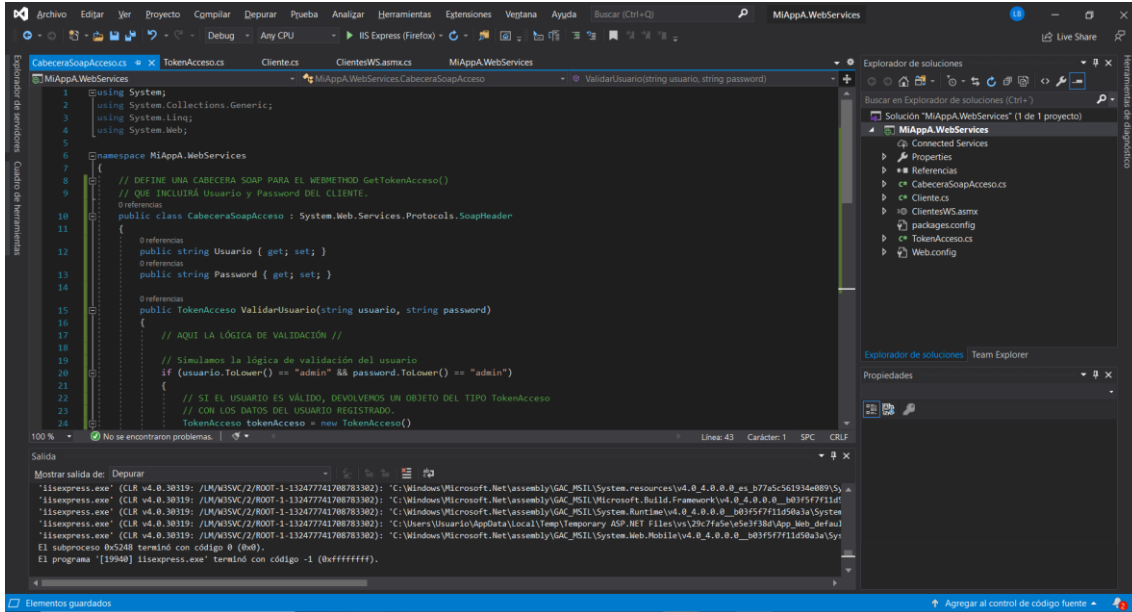
Para ello, crearemos dos nuevas clases (CabeceraSoapAcceso.cs y CabeceraSoapToken.cs) que heredarán de la clase System.Web.Services.Protocols.SoapHeader. Estas clases representarán las cabeceras SOAP que utilizaremos posteriormente para enviar al Servicio Web todos los datos relativos a la seguridad.

Creamos la clase CabeceraSoapAcceso.cs



Damos clic en agregar y colocamos este código:

```
// DEFINE UNA CABECERA SOAP PARA EL WEBMETHOD GetTokenAcceso()  
// QUE INCLUIRÁ Usuario y Password DEL CLIENTE.  
public class CabeceraSoapAcceso : System.Web.Services.Protocols.SoapHeader  
{  
    public string Usuario { get; set; }  
    public string Password { get; set; }  
  
    public TokenAcceso ValidarUsuario(string usuario, string password)  
    {  
        // AQUI LA LÓGICA DE VALIDACIÓN //  
  
        // Simulamos la lógica de validación del usuario  
        if (usuario.ToLower() == "admin" && password.ToLower() == "admin")  
        {  
            // SI EL USUARIO ES VÁLIDO, DEVOLVEMOS UN OBJETO DEL TIPO TokenAcceso  
            // CON LOS DATOS DEL USUARIO REGISTRADO.  
            TokenAcceso tokenAcceso = new TokenAcceso()  
            {  
                Id = Guid.NewGuid(),  
                Nombre = "Rafael",  
                Apellidos = "Acosta",  
                Email = "mi.email@gmail.com",  
                Rol = "Administrador"  
            };  
  
            return tokenAcceso;  
        }  
        else  
        {  
            // SI NO ES VÁLIDO EL USUARIO RETORNAMOS NULL.  
            return null;  
        }  
    }  
}
```



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5
6 namespace MIAppA.WebServices
7 {
8     // DEFINE UNA CABECERA SOAP PARA EL WEBMETHOD GetTokenAcceso()
9     // QUE INCLUIRÁ Usuario y Password DEL CLIENTE.
10
11     public class CabeceraSoapAcceso : System.Web.Services.Protocols.SoapHeader
12     {
13         public string Usuario { get; set; }
14         public string Password { get; set; }
15
16         public TokenAcceso ValidarUsuario(string usuario, string password)
17         {
18             // AQUÍ LA LÓGICA DE VALIDACIÓN //
19             // Simulamos la lógica de validación del usuario
20             if (usuario.ToLower() == "admin" && password.ToLower() == "admin")
21             {
22                 // SI EL USUARIO ES VÁLIDO, DEVOLVEMOS UN OBJETO DEL TIPO TokenAcceso
23                 // CON LOS DATOS DEL USUARIO REGISTRADO.
24                 TokenAcceso tokenAcceso = new TokenAcceso();
25             }
26         }
27     }
28 }
```

Salida

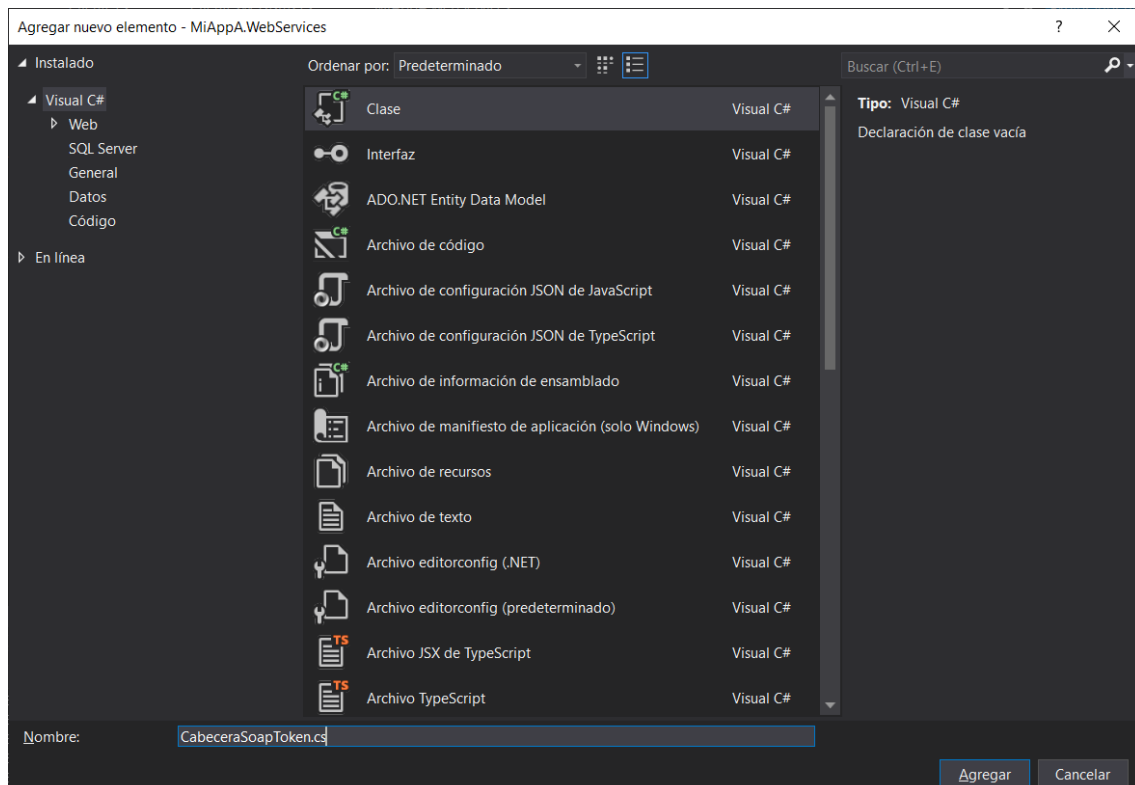
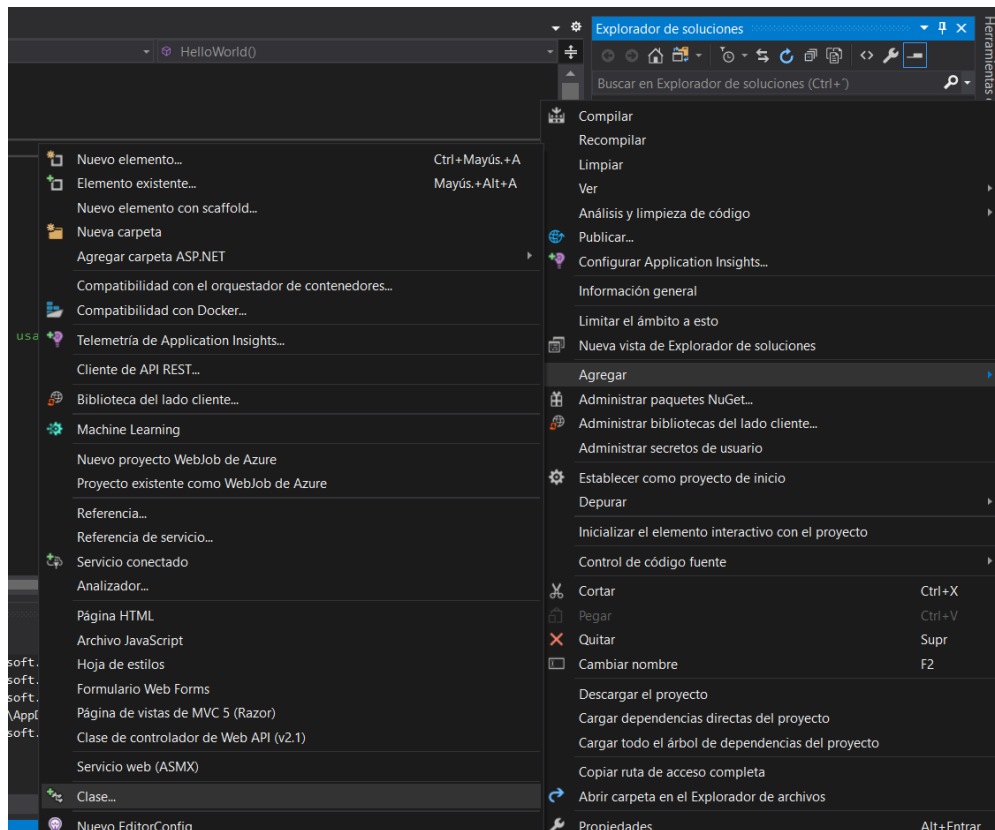
Mostrar salida de: Depurar

```
"IISexpress.exe" (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132477741708783302): "C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.resources\v4.0.4.0.es_b77a5c561934e889\System.resources.dll"
"IISexpress.exe" (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132477741708783302): "C:\Windows\Microsoft.Net\assembly\GAC_MSIL\Microsoft.Build.Framework\v4.0.4.0.es_b03f5f7f11d50a3a\System.Build.Framework.dll"
"IISexpress.exe" (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132477741708783302): "C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Runtime\v4.0.4.0.es_b03f5f7f11d50a3a\System.Runtime.dll"
"IISexpress.exe" (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132477741708783302): "C:\Users\Usuario\AppData\Local\Temp\Temporary ASP.NET Files\329c7f45a5e3f38a\MIAppA.Web.Default.aspx.dll"
El subproceso 0x5248 terminó con código 0 (0x0).
El programa "[13940] IISexpress.exe" terminó con código -1 (0xffffffff).
```

Como vemos en el código, esta cabecera SOAP consta de dos propiedades (Usuario y Password), las cuales servirán para autenticar el usuario en el sistema de información corporativo.

La autenticación la haremos mediante el método ValidarUsuario(string usuario, string password), el cual devolverá un objeto del tipo TokenAcceso en el caso de ser un usuario válido, en caso contrario devolverá null.

Creamos la clase CabeceraSoapToken.cs

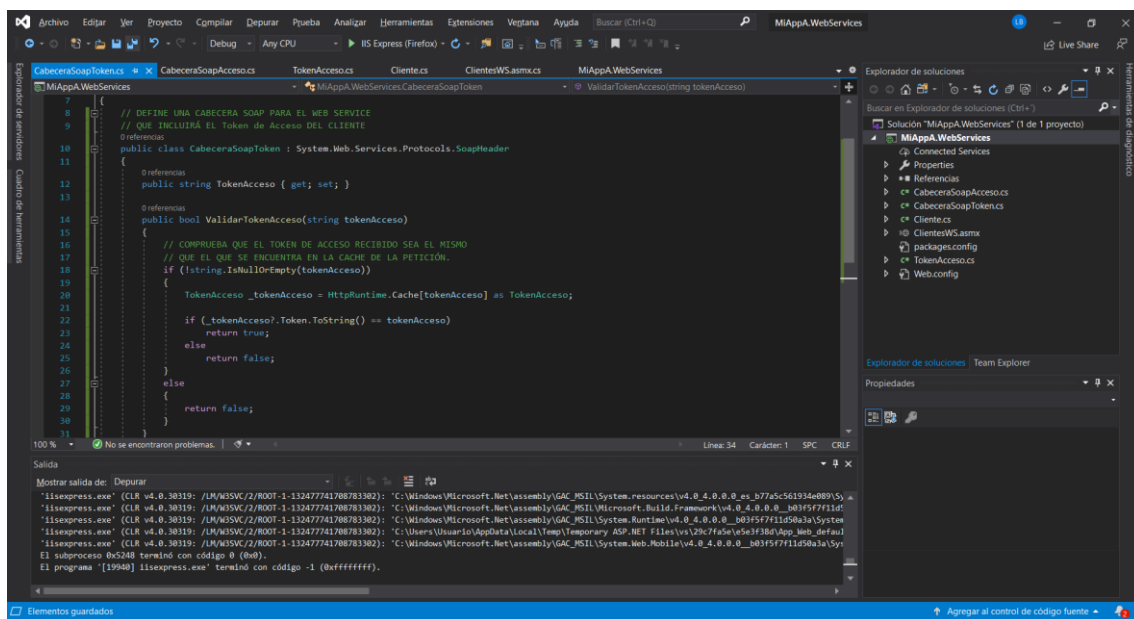


Damos clic en agregar y colocamos este código:

```
// DEFINE UNA CABECERA SOAP PARA EL WEB SERVICE
// QUE INCLUIRÁ EL Token de Acceso DEL CLIENTE
public class CabeceraSoapToken : System.Web.Services.Protocols.SoapHeader
{
    public string TokenAcceso { get; set; }

    public bool ValidarTokenAcceso (string tokenAcceso)
    {
        // COMPRUEBA QUE EL TOKEN DE ACCESO RECIBIDO SEA EL MISMO
        // QUE EL QUE SE ENCUENTRA EN LA CACHE DE LA PETICIÓN.
        if (!string.IsNullOrEmpty(tokenAcceso))
        {
            TokenAcceso _tokenAcceso = HttpRuntime.Cache[tokenAcceso] as TokenAcceso;

            if (_tokenAcceso?.Token.ToString() == tokenAcceso)
                return true;
            else
                return false;
        }
        else
        {
            return false;
        }
    }
}
```



Esta cabecera SOAP será la encargada "transportar" el Token que dará acceso a los servicios, mediante la propiedad TokenAcceso.



El método ValidarTokenAcceso (string tokenAcceso), será el encargado de comprobar que el Token recibido es un Token válido para acceder al Servicio Web.

Nota: La comprobación de la validez del Token recibido, se realizará comparándolo con el que se encuentra en la Cache Http del Servicio Web (HttpRuntime.Cache), y que ya fue anteriormente enviado al usuario cuando se validó en el sistema. Todo este proceso de generación del Token, almacenamiento en Cache y envío al usuario, lo veremos a continuación cuando construyamos el Web Service principal de la aplicación, ClientesWS.asmx.

En este punto, ya estamos en disposición de comenzar a desarrollar el Servicio Web con seguridad basada en Tokens de nuestro ejemplo.

Para ello, abriremos el archivo ClientesWS.asmx creado anteriormente, y le añadiremos el siguiente código:

```
/// <summary>
/// Descripción breve de ClientesWS
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.ComponentModel.ToolboxItem(false)]
// Para permitir que se llame a este servicio web desde un script, usando ASP.NET AJAX,
// quite la marca de comentario de la línea siguiente.
// [System.Web.Script.Services.ScriptService]
public class ClientesWS : System.Web.Services.WebService
{
    private List<Cliente> clientes;

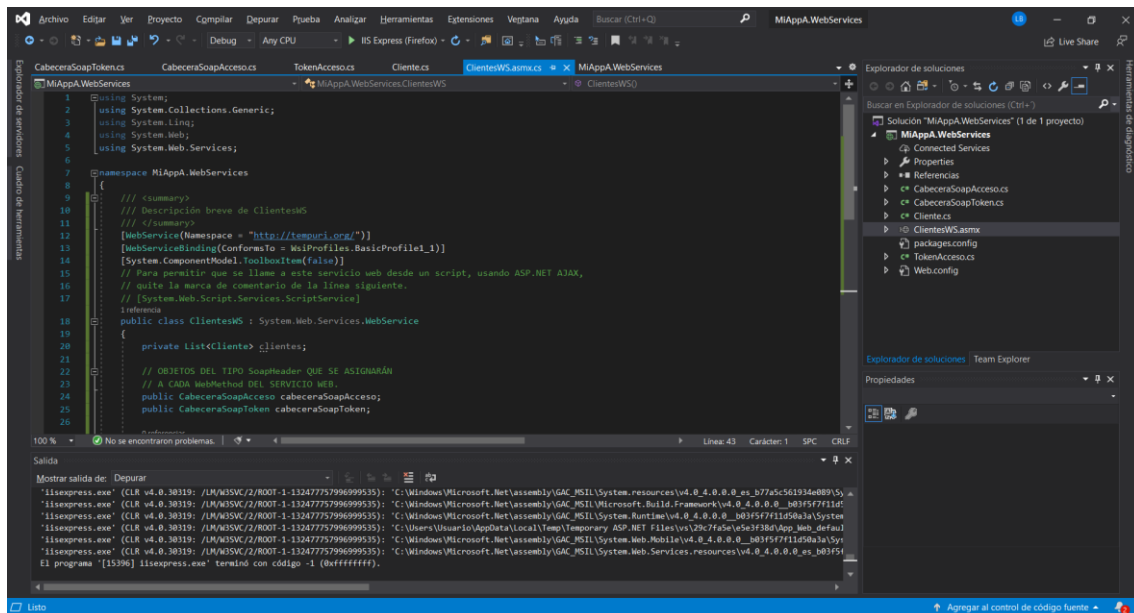
    // OBJETOS DEL TIPO SoapHeader QUE SE ASIGNARÁN
    // A CADA WebMethod DEL SERVICIO WEB.
    public CabeceraSoapAcceso cabeceraSoapAcceso;
    public CabeceraSoapToken cabeceraSoapToken;

    public ClientesWS()
    {
        // SIMULAMOS UNA LISTA DEL MODELO DE DATOS.
        clientes = new List<Cliente>()
        {
            new Cliente() { IdCliente = 1, NombreCliente = "Maria Anders", NombreEmpresa =
"Alfreds Futterkiste", Telefono = "030-0074321", Email="maria.anders@northwind.com"},
            new Cliente() { IdCliente = 2, NombreCliente = "Ana Trujillo", NombreEmpresa = "Ana
Trujillo Emparedados y helados", Telefono = "(5) 555-4729",
Email="ana.trujillo@northwind.com"},
            new Cliente() { IdCliente = 3, NombreCliente = "Antonio Moreno", NombreEmpresa =
"Antonio Moreno Taquería", Telefono = "(5) 555-3932",
Email="antonio.moreno@northwind.com"},
            new Cliente() { IdCliente = 4, NombreCliente = "Thomas Hardy", NombreEmpresa =
"Berglunds snabbkop", Telefono = "(171) 555-7788", Email="thomas.hardy@northwind.com"},
        }
    }
}
```

```

        new Cliente() { IdCliente = 5, NombreCliente = "Christina Berglund", NombreEmpresa =
"Alfreds Futterkiste", Telefono = "0921-12 34 65",
Email="christina.berglund@northwind.com1"},
        new Cliente() { IdCliente = 6, NombreCliente = "Hanna Moos", NombreEmpresa =
"Blauer See Delikatessen", Telefono = "0621-08460", Email="hanna.moos@northwind.com"},
        new Cliente() { IdCliente = 7, NombreCliente = "Martín Sommer", NombreEmpresa =
"Bóldo Comidas preparadas", Telefono = "(91) 555 22 82",
Email="martín.sommer@northwind.com"}
    };
}
}

```



Como vemos, hemos creado dos cabeceras SOAP como las que ya vimos anteriormente, y hemos simulado en el constructor de la clase, una lista del tipo List<Cliente> con datos de prueba.

Seguidamente crearemos el método ([WebMethod]) encargado de generar el Token de acceso, almacenarlo en Cache y enviarlo al usuario: GetTokenAcceso().

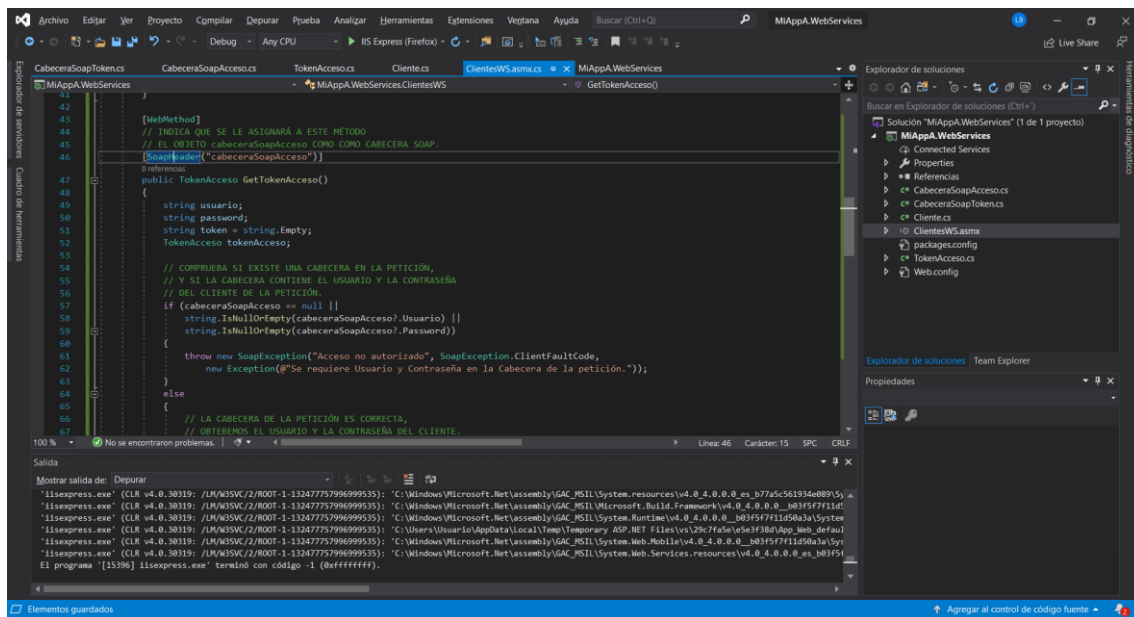
```

[WebMethod]
// INDICA QUE SE LE ASIGNARÁ A ESTE MÉTODO
// EL OBJETO cabeceraSoapAcceso COMO COMO CABECERA SOAP.
[SoapHeader("cabeceraSoapAcceso")]
public TokenAcceso GetTokenAcceso()
{
    string usuario;
    string password;
    string token = string.Empty;
    TokenAcceso tokenAcceso;

```

```
// COMPRUEBA SI EXISTE UNA CABECERA EN LA PETICIÓN,  
// Y SI LA CABECERA CONTIENE EL USUARIO Y LA CONTRASEÑA  
// DEL CLIENTE DE LA PETICIÓN.  
if (cabeceraSoapAcceso == null ||  
    string.IsNullOrEmpty(cabeceraSoapAcceso?.Usuario) ||  
    string.IsNullOrEmpty(cabeceraSoapAcceso?.Password))  
{  
    throw new SoapException("Acceso no autorizado", SoapException.ClientFaultCode,  
        new Exception(@"Se requiere Usuario y Contraseña en la Cabecera de la petición."));  
}  
else  
{  
    // LA CABECERA DE LA PETICIÓN ES CORRECTA,  
    // OBTENEMOS EL USUARIO Y LA CONTRASEÑA DEL CLIENTE.  
    usuario = cabeceraSoapAcceso.Usuario;  
    password = cabeceraSoapAcceso.Password;  
}  
  
// COMPRUEBA QUE EL USUARIO Y LA CONTRASEÑA  
// DE LA PETICIÓN SON VÁLIDOS.  
tokenAcceso = cabeceraSoapAcceso.ValidarUsuario(usuario, password);  
if (tokenAcceso != null)  
{  
    // EL USUARIO Y CONTRASEÑA SON CORRECTOS,  
    // SE CREA EL TOKEN DE ACCESO Y SE ALMACENA  
    // EN LA CACHE DE LA PETICIÓN ANTES DE DEVOLVERLO.  
    tokenAcceso.Token = Guid.NewGuid();  
  
    HttpRuntime.Cache.Add(  
        tokenAcceso.Token.ToString(),  
        tokenAcceso,  
        null,  
        System.Web.Caching.Cache.NoAbsoluteExpiration,  
        TimeSpan.FromMinutes(2), // 2 MINUTOS DE VIDA (PARA PRUEBAS)  
        System.Web.Caching.CacheItemPriority.NotRemovable,  
        null  
    );  
}  
else  
{  
    // EL USUARIO Y LA CONTRASEÑA NO SON VÁLIDOS.  
    throw new SoapException("Acceso no autorizado", SoapException.ClientFaultCode,  
        new Exception(@"EL Usuario y/o Contraseña no son válidos."));  
}  
  
// DEVOLVEMOS EL TOKEN DE ACCESO  
if (!string.IsNullOrEmpty(tokenAcceso.Token.ToString()))  
{
```

```
return tokenAcceso;  
}  
else  
{  
    throw new SoapException("Acceso no autorizado", SoapException.ClientFaultCode,  
        new Exception(@"ERROR. No se ha podido generar el Token de acceso."));  
}  
}
```



Como vemos en el código, este método requiere de una cabecera SOAP del tipo CabeceraSoapAcceso, donde irán el Usuario y Password para la autenticación.

Importante: La forma de asignar a un WebMethod una cabecera SOAP, se realiza mediante el atributo [SoapHeader("...")]. En este caso, le indicaremos que la cabecera SOAP será del tipo CabeceraSoapAcceso mediante su objeto correspondiente: [SoapHeader("cabeceraSoapAcceso")].

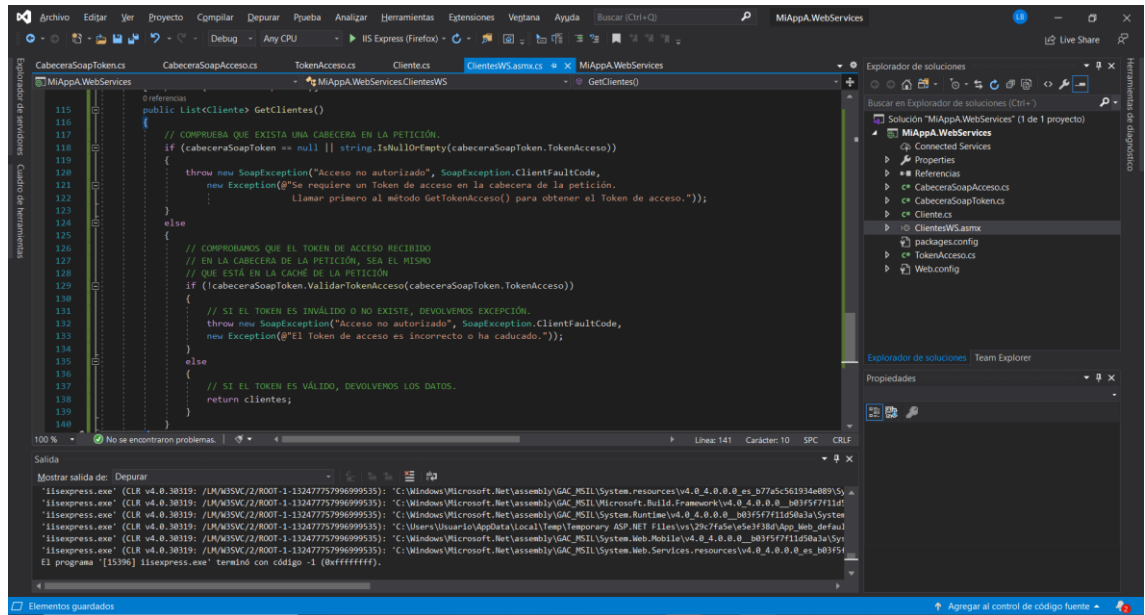
Una vez comprobada la validez del usuario mediante los datos enviados en la cabecera (Usuario y Password), generamos un Token de acceso, lo añadimos al objeto del tipo TokenAcceso, almacenamos este objeto en el HttpRuntime.Cache y lo enviamos de vuelta al usuario.

Como vemos en la imagen, lo que almacenamos en Cache (HttpRuntime.Cache) es un elemento del tipo Key/Value. El Value contendrá un objeto del tipo TokenAcceso con un tiempo de caducidad asignado, y el Key será el propio Token de acceso generado. Esto lo hacemos así, para poder mantener en Cache tantos objetos con Keys diferentes como usuarios accedan al Web Service, y cada uno de ellos con un tiempo de caducidad asignado.

En este punto, ya tenemos un Token de acceso válido al Web Service. Es el momento entonces de probar la funcionalidad de seguridad, creando un método ([WebMethod]) en el servicio, que requiera este Token para acceder a sus datos.

Crearemos entonces el método `GetClientes()`, que devolverá la lista de clientes (`List<Cliente>` clientes), con los datos de prueba que creamos anteriormente en el constructor de Web Service.

```
[WebMethod]
// INDICA QUE SE LE ASIGNARÁ A ESTE MÉTODO
// EL OBJETO cabeceraSoapToken COMO COMO CABECERA SOAP.
[SoapHeader("cabeceraSoapToken")]
public List<Cliente> GetClientes()
{
    // COMPRUEBA QUE EXISTA UNA CABECERA EN LA PETICIÓN.
    if (cabeceraSoapToken == null ||
string.IsNullOrEmpty(cabeceraSoapToken.TokenAcceso))
    {
        throw new SoapException("Acceso no autorizado",
SoapException.ClientFaultCode,
        new Exception(@"Se requiere un Token de acceso en la cabecera de la
petición. Lllamar primero al método GetTokenAcceso() para obtener el Token
de acceso."));
    }
    else
    {
        // COMPROBAMOS QUE EL TOKEN DE ACCESO RECIBIDO
        // EN LA CABECERA DE LA PETICIÓN, SEA EL MISMO
        // QUE ESTÁ EN LA CACHÉ DE LA PETICIÓN
        if
(!cabeceraSoapToken.ValidarTokenAcceso(cabeceraSoapToken.TokenAcceso))
        {
            // SI EL TOKEN ES INVÁLIDO O NO EXISTE, DEVOLVEMOS EXCEPCIÓN.
            throw new SoapException("Acceso no autorizado",
SoapException.ClientFaultCode,
            new Exception(@"El Token de acceso es incorrecto o ha caducado."));
        }
        else
        {
            // SI EL TOKEN ES VÁLIDO, DEVOLVEMOS LOS DATOS.
            return clientes;
        }
    }
}
```



```

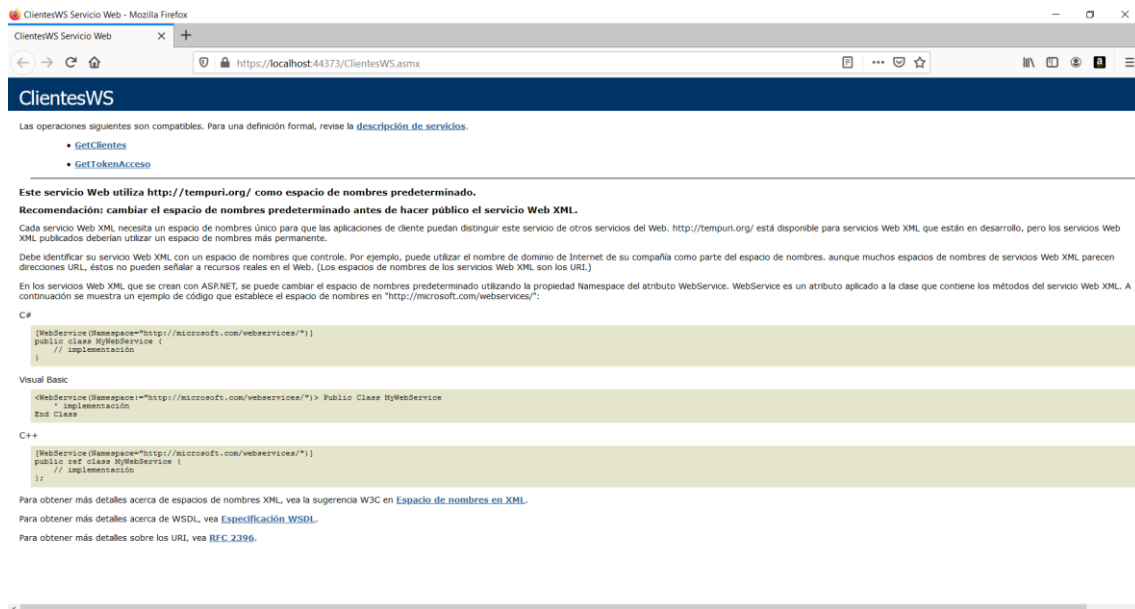
115 public List<Cliente> GetClientes()
116 {
117     // COMPROBAMOS QUE EXISTA UNA CABECERA EN LA PETICIÓN.
118     if (cabeceraSoapToken == null || string.IsNullOrEmpty(cabeceraSoapToken.TokenAcceso))
119     {
120         throw new SoapException("Acceso no autorizado", SoapException.ClientFaultCode,
121             new Exception("Se requiere un Token de acceso en la cabecera de la petición.
122             Llamar primero al método GetTokenAcceso() para obtener el Token de acceso."));
123     }
124     else
125     {
126         // COMPROBAMOS QUE EL TOKEN DE ACCESO RECIBIDO
127         // EN LA CABECERA DE LA PETICIÓN, SEA EL MISMO
128         // QUE ESTÁ EN LA CACHE DE LA PETICIÓN
129         if (!cabeceraSoapToken.ValidarTokenAcceso(cabeceraSoapToken.TokenAcceso))
130         {
131             // SI EL TOKEN ES INVÁLIDO O NO EXISTE, DEVOLVEMOS EXCEPCIÓN.
132             throw new SoapException("Acceso no autorizado", SoapException.ClientFaultCode,
133                 new Exception("El Token de acceso es incorrecto o ha caducado."));
134         }
135         else
136         {
137             // SI EL TOKEN ES VÁLIDO, DEVOLVEMOS LOS DATOS.
138             return clientes;
139         }
140     }
141 }

```

Como vemos en el código, este método requiere de una cabecera SOAP del tipo CabeceraSoapToken, donde irá el Token de acceso que permitirá acceder a los datos.

El método ValidarTokenAcceso(cabeceraSoapToken.TokenAcceso) de la propia cabecera SOAP, comprobará si el Token recibido es válido, comparándolo con el que se almacenó en su momento en la Cache del servicio (HttpRuntime.Cache)

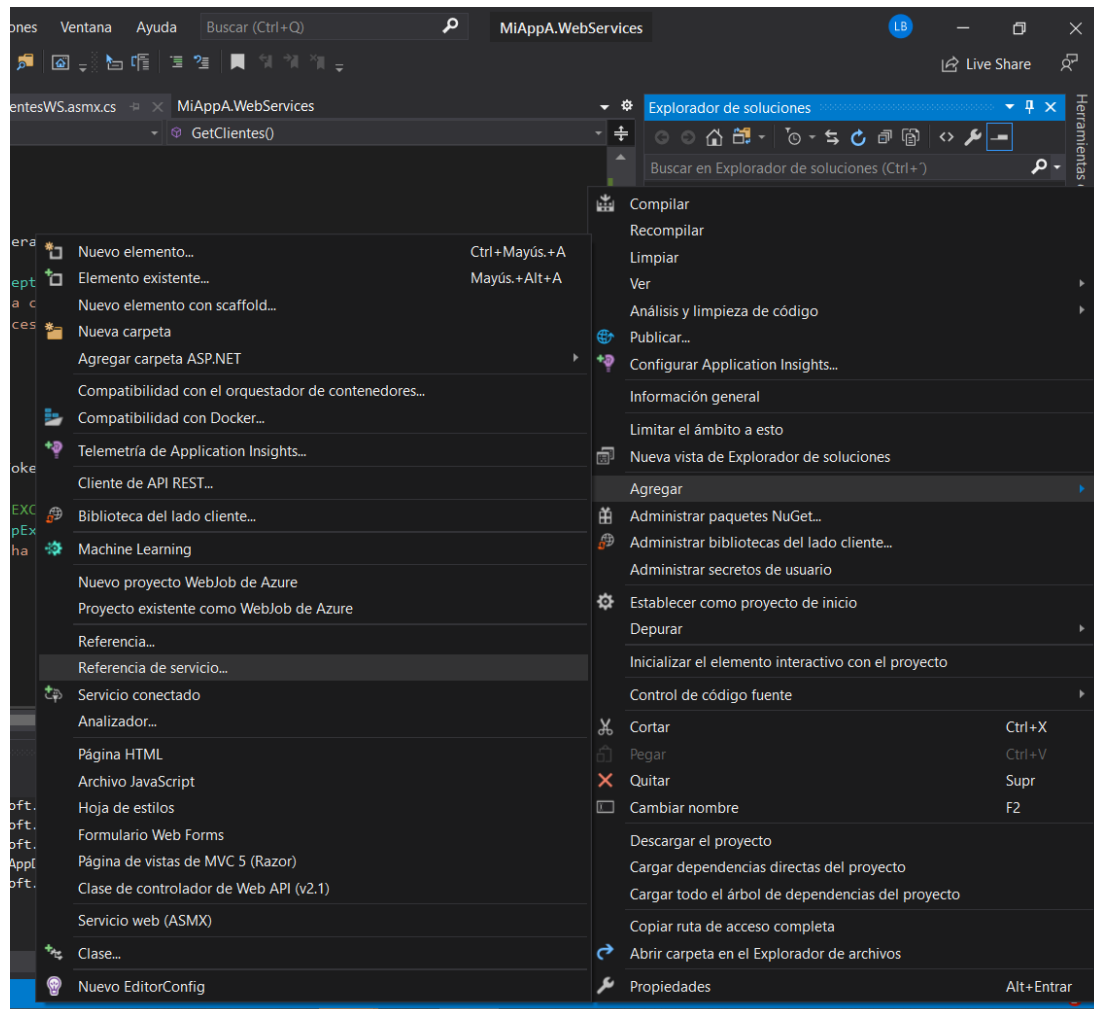
Compilamos y ejecutamos



Con ello se verifica que se creó se servicio web correctamente.

A continuación, crearemos el Cliente o Proxy que consumirá el Servicio Web (ClientesWS.asmx)

Esto lo haremos creando una nueva Referencia de servicio (Botón derecho sobre el Proyecto: Agregar > Referencia de servicio... ) al Web Service (por ejemplo: <https://localhost:44394/ClientesWS.asmx>) con el Nombre ClientesWS.



Teniendo lo siguiente:

Agregar referencia de servicio

?

×

Para ver una lista de los servicios disponibles en un servidor específico, escriba una dirección URL de servicio y haga clic en Ir. Para buscar servicios disponibles, haga clic en Detectar.

Dirección:

Ir

Detectar

Servicios:

Operaciones:

Espacio de nombres:

ServiceReference1

Ayanzadas...

Aceptar

Cancelar

Damos clic en el botón detectar



Agregar referencia de servicio ? X

Para ver una lista de los servicios disponibles en un servidor específico, escriba una dirección URL de servicio y haga clic en Ir. Para buscar servicios disponibles, haga clic en Detectar.

Dirección:

Servicios:	Operaciones:
<ul style="list-style-type: none"><li>ClientesWS.asmx<ul style="list-style-type: none"><li>ClientesWS<ul style="list-style-type: none"><li>ClientesWSSoap</li></ul></li></ul></li></ul>	<ul style="list-style-type: none"><li>GetClientes</li><li>GetTokenAcceso</li></ul>

1 servicios encontrados en la dirección 'https://localhost:44373/ClientesWS.asmx'.

Espacio de nombres:

En espacio de nombres colocamos:

Agregar referencia de servicio

?

×

Para ver una lista de los servicios disponibles en un servidor específico, escriba una dirección URL de servicio y haga clic en Ir. Para buscar servicios disponibles, haga clic en Detectar.

Dirección:

https://localhost:44373/ClientesWS.asmx

Ir

Detectar

Servicios:

Operaciones:

▲ ClientesWS.asmx

▲ ClientesWS

● ClientesWSSoap

GetClientes

GetTokenAcceso

1 servicios encontrados en la dirección 'https://localhost:44373/ClientesWS.asmx'.

Espacio de nombres:

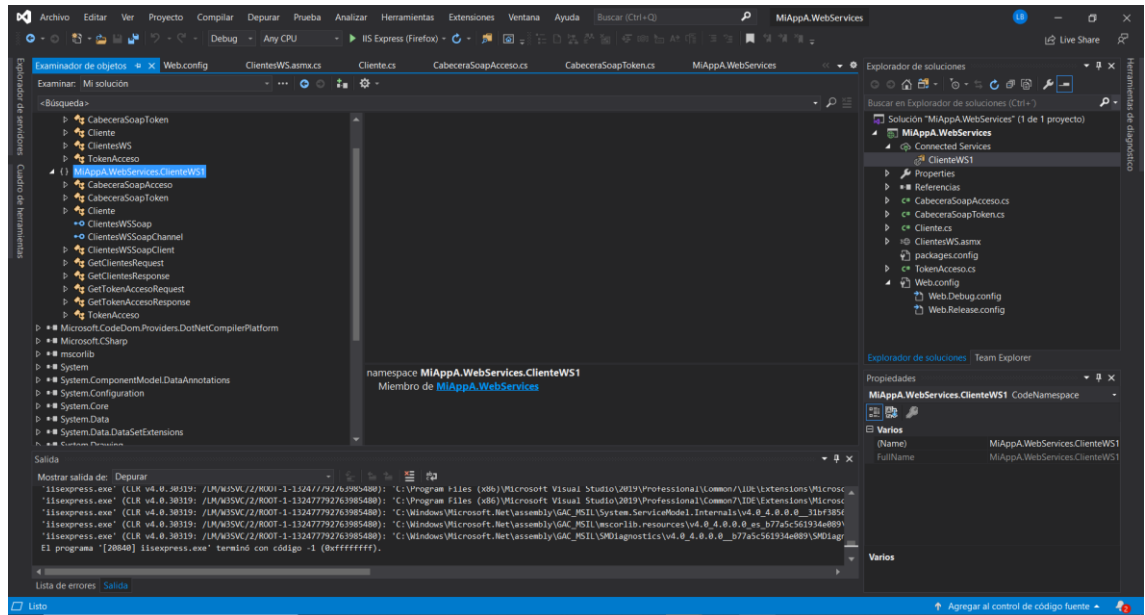
ClienteWS1

Ayanzadas...

Aceptar

Cancelar

Y damos clic en aceptar



## 5. BIBLIOGRAFIA:

TITULO	AUTOR	AÑO	EDITORIAL	URL/OBSERV
Enciclopedia de Microsoft Visual C#: interfaces gráficas y aplicaciones para Internet con Windows Forms y ASP.NET (4a. ed.).	Ceballos Sierra, F. J.	2015	RA-MA Editorial.	<a href="https://elibro.net/es/lc/uisrael/titulos/62510">https://elibro.net/es/lc/uisrael/titulos/62510</a>