Curso Formiik

Manual Versión 1.1

Autores:	Pablo Monroy Sánchez
Fecha de Creación:	Junio 18, 2013
Fecha de Ultima Modificación	Junio 20, 2017

Contenido

Manual de Implementación

Introducción

El presente manual es un intento por facilitar el acceso al equipo de desarrollo de nuestros nuevos clientes a los servicios que ofrece Formiik.

El manual guía paso a paso en el proceso de implementación de los servicios necesarios para comenzar a operar sus procesos de negocio.

Todo el código presentado en este manual puede ser descargado desde la página de soporte de Formiik

Agradecemos cualquier comentario a: soporte@formiik.com

Explicación del proceso de negocio en Formiik

¿Qué es Formiik?

Es una aplicación móvil que le permite mejorar la productividad y el control de sus operaciones en campo. Ayuda a su personal a consultar, capturar y enviar la información que su negocio requiere, desde cualquier lugar y en cualquier momento.

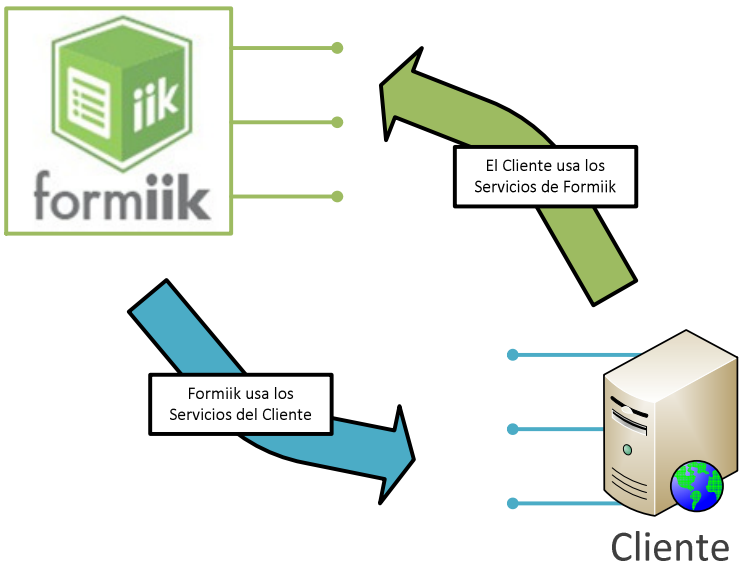
Formiik está formado por dos componentes:

1. El componente móvil que es una aplicación desarrollada para teléfonos móviles con sistema operativo Android, iOS y Windows Mobile.
2. El componente web que es una aplicación Web para el control y supervisión del trabajo de los operadores.

El área técnica de su empresa, para poder trabajar con Formiik, debe conocer y utilizar los servicios web que ofrece Formiik.

Formiik expone sus servicios web para que las empresas puedan desplegar las órdenes de trabajo para sus operadores. Los operadores recaban la información usada en su proceso de negocio en un teléfono celular con la aplicación de Formiik.

Formiik solicita a que su empresa construya sus propios servicios web. Estos servicios web tendrán la tarea de recibir las respuestas que los operadores recabaron desde su teléfono celular.

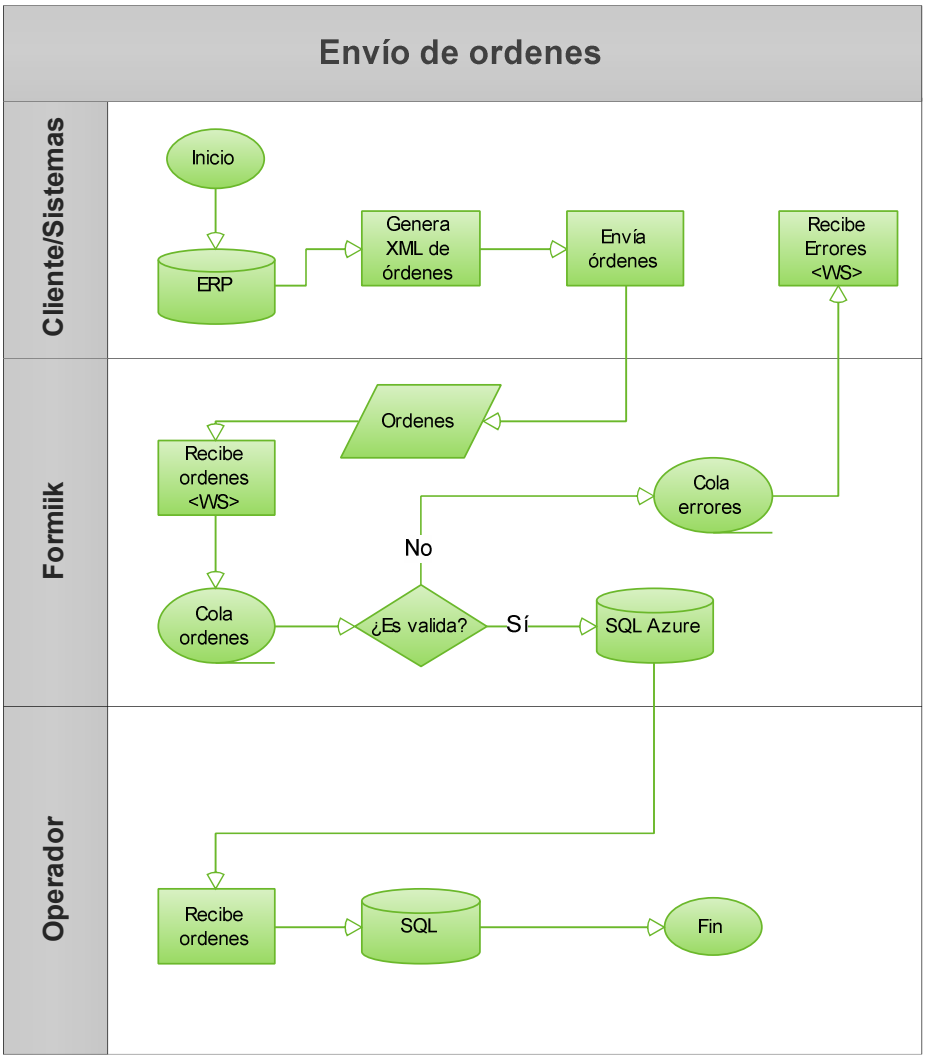


Utilizar Formiik

Para poder usar Formiik es necesario que el equipo de desarrollo del cliente (Ud.) construya los servicios que se utilizarán principalmente para el envío de las órdenes de trabajo a los operadores y la recepción de respuestas que también capturan los operadores.

Servicio de Envío de Órdenes

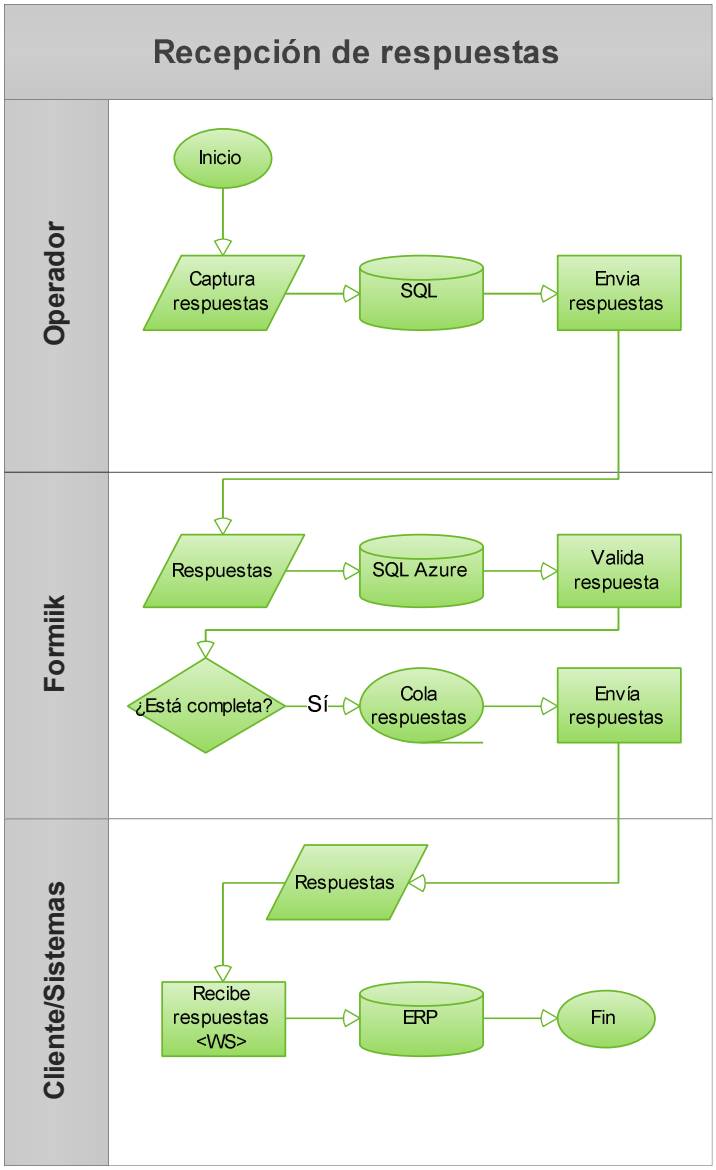
El servicio de envío de órdenes se puede desarrollar en cualquier plataforma y lenguaje de programación que sea capaz de utilizar servicios web de tipo SOAP.



	Actor	Descripción del proceso
1	Cliente/Sistemas	Extrae información de sus sistemas Ej. ERP, y con esa información genera una orden de trabajo con los datos de su propio cliente y genera un XML que se envía a Formiik
2	Formiik	Formiik a través de su servicio web recibe las órdenes, las valida y si todo está correcto guarda las órdenes en espera de que el operador sincronice su teléfono celular.
3	Operador	El operador sincroniza su teléfono y descarga las órdenes de trabajo que le corresponden.

Servicio de Recepción de Órdenes

El servicio de recepción de órdenes se puede desarrollar en cualquier plataforma y lenguaje de programación que sea capaz de crear servicios web de tipo REST.



	Actor	Descripción del proceso
1	Operador	Captura las respuestas en su teléfono celular y envía las respuestas a Formiik
2	Formiik	Formiik recibe las respuestas, guarda las respuestas junto con sus archivos adjuntos (fotos, documentos, etc.) Cuando la respuesta está completa envía las respuestas al cliente.
3	Cliente/Sistemas	El cliente con su servicio web recibe la respuesta junto con las urls de los archivos adjuntos.

Servicios SOAP

Utilizar los servicios de Formiik para el envío de órdenes de trabajo

El código para los servicios SOAP puede ser descargado de:
https://www.dropbox.com/sh/9m5khfbsddb0hf1/AADP3-AglJewxkjd2Nowt186a?dl=0

Requisitos

En el presente manual para desarrollar se utilizará:

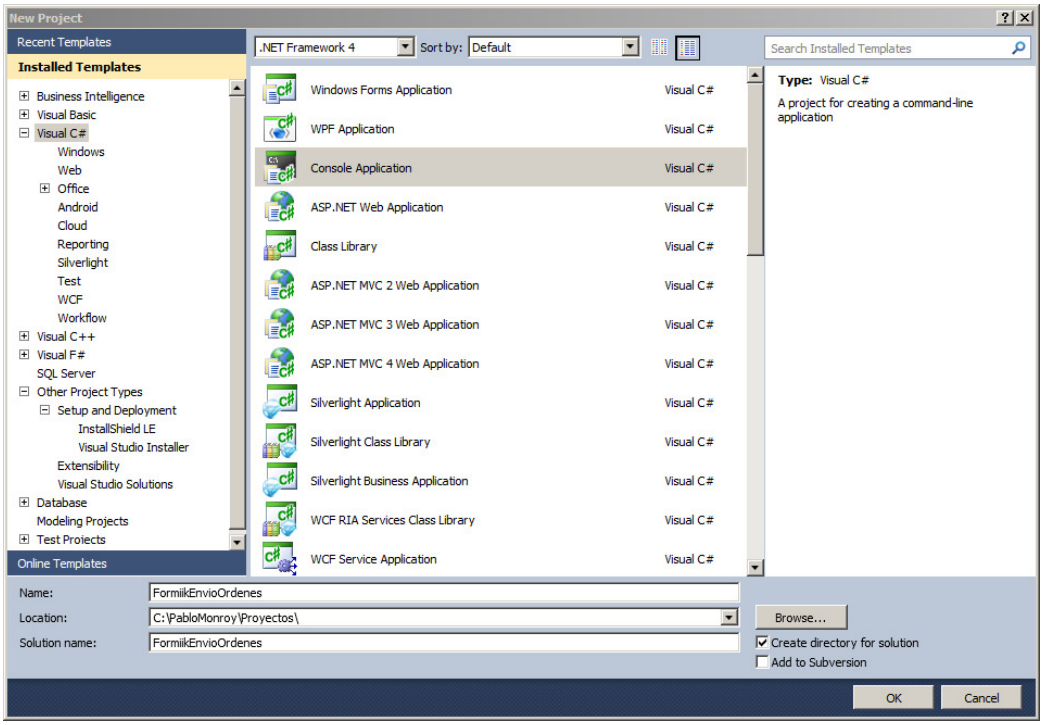
- Visual Studio 2010 o posterior
- Lenguaje C#
- .Net Framework 4.0
- IIS 7

Urls servicios SOAP

Sin Certificado: http://services.formiik.com:8081/BackEnd.svc?wsdl
Con Certificado: https://services.formiik.com:8084/BackEnd.svc?wsdl

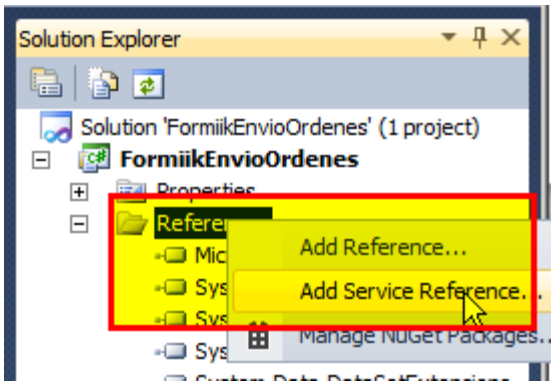
Instrucciones

1. Abrir Visual Studio
2. Crear un nuevo proyecto – para fines de este ejemplo se utilizará un proyecto de tipo consola.
3. Utilizar el .Net Framework 4
4. Escribir nombre de FormiikEnvioOrdenes



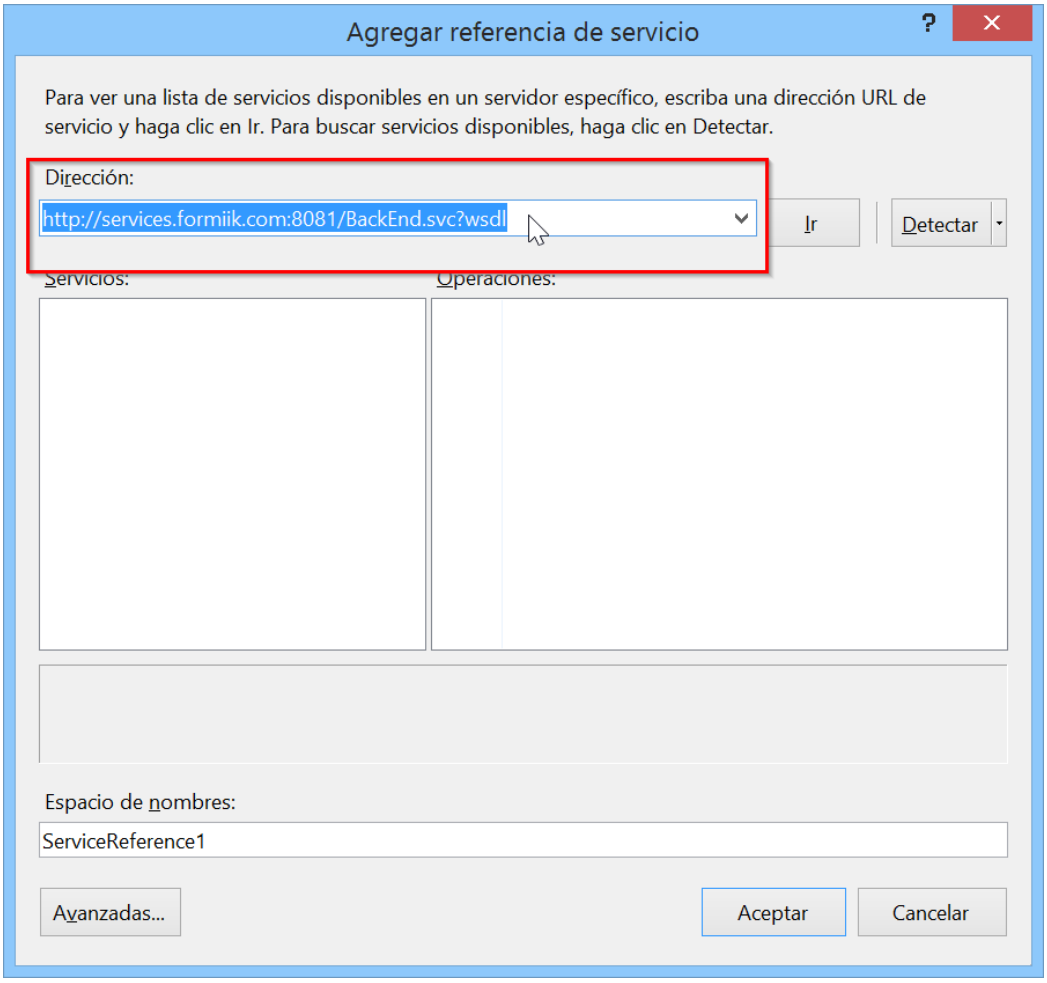
5. Seleccionar OK

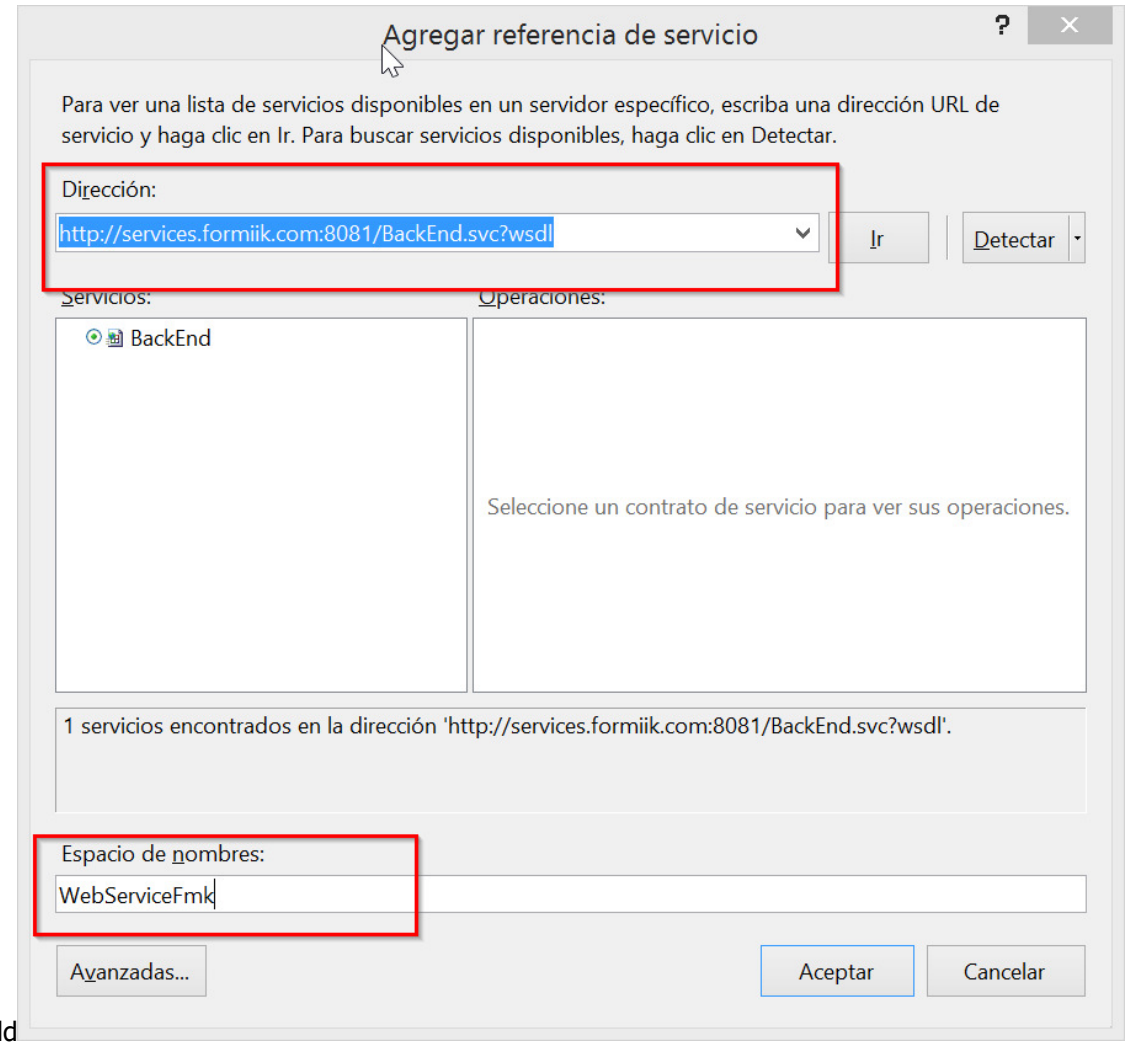
6. Agregar la referencia a los servicios web de Formiik con el botón derecho del ratón sobre el folder de References seleccionar Add Service Reference



7. En Address escribir http://services.formiik.com:8081/BackEnd.svc?wsdl

8. Oprimir Go


Agregar referencia de servicio
Para ver una lista de 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:
http://services.formiik.com:8081/BackEnd.svc?wsdl
Ir
Detectar
Servicios:
Operaciones:
Espacio de nombres:
ServiceReference1
Avanzadas...
Aceptar
Cancelar



add

9. Escribir el siguiente código para enviar respuestas

```
using System;
using System.Data;
using System.IO;
using System.IO.Compression;
using System.ServiceModel;
using System.Web.Services.Protocols;
using System.Xml;
using FormiikEnvioOrdenes.WebServiceFmk;
using System.Collections.Generic;
using System.Xml.Serialization;
namespace FormiikEnvioOrdenes
{
    class Program
    {
        static void Main(string[] args)
        {
            //AddWorkOrdersXML
            //Este servicio envía un conjunto de ordenes.
            //-----------------------------------------
            //Parametros
            //clientId      Identificador con el cual será reconocido el cliente
            //productId     Identificador con el que se reconocerá la aplicación del cliente
            //workOrders    XML con N número de ordenes para agregar.
```

```
            Coleccion coleccion = new Coleccion();
            List<NewOrder> ListaNewOrder = new List<NewOrder>();

            NewOrder orden = new NewOrder();
            orden.Id = "1";
            orden.Type = "Mantenimiento";

            ListaNewOrder.Add(orden);
            XmlSerializer serializer = new XmlSerializer(typeof(List<NewOrder>));

            serializer.Serialize(Console.Out, ListaNewOrder);

            string IdClient  = "BB99DB2A-FDB6-4104-B629-7A15A1B0E123";
            string IdProduct = "AB3B220D-000F-4C30-A96F-EB8A73631234";

            string[] filePaths = Directory.GetFiles(@"C:\Asigna\");

            //Si ya se tienen generados los archivoXML
            //Envía los XML
            foreach (string archivo in filePaths)
            {
                string IdPath = archivo;


                StreamReader objReader;

                objReader = new StreamReader(IdPath); //<-ubicación del archivo XML

                string strXML = objReader.ReadToEnd();

                XmlDocument xmldoc = new XmlDocument();

                xmldoc.LoadXml(strXML);

                //Se mandan las Ordenes
                try
                {
                    BackEndClient bkclient = new BackEndClient("BasicHttpBinding_IBackEnd1");

                    string idDeEnvio = bkclient.AddWorkOrdersXMLId(IdClient, IdProduct, strXML);

                    Console.WriteLine(idDeEnvio);
                }
                catch (FaultException ex)
                {
                    Console.WriteLine("Error {0}", ex.Message);
                    Console.WriteLine("Presione cualquier tecla para continuar..");
                    Console.ReadKey();
                }

                catch (SoapException ex)
                {
                    Console.WriteLine("Error {0}", ex.Message);
                }

                catch (Exception ex)
                {
                    Console.WriteLine("Error {0}", ex.Message);
                }
            }
            Console.ReadKey();
        }
    }
}
```

Para más detalles sobre el método AddWorkOrders

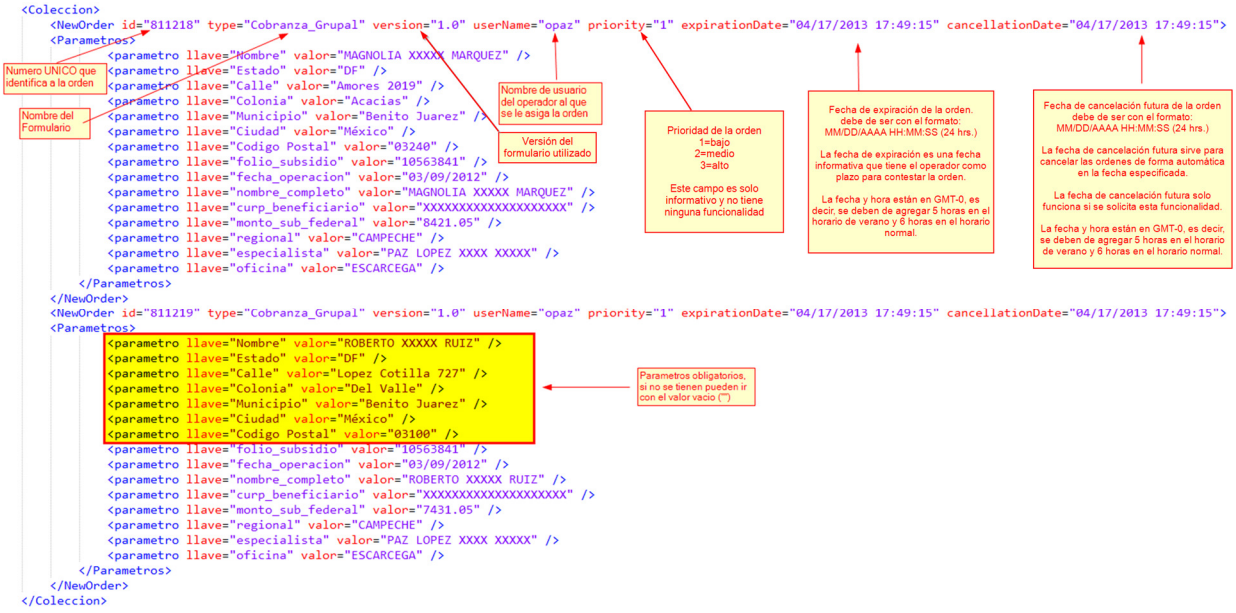
Creación del documento o cadena XML

Cada orden debe de llevar un id UNICO (similar a un folio) para que se pueda identificar cada una de las órdenes, también debe de llevar el userName del operador. Las fechas que se envían expirationDate y cancelationDate tienen que estar con el formato MM/DD/AAAA HH:MM:SS (24 hrs)

Los campos Nombre, Estado, Calle, Colonia, Municipio, Ciudad, Código Postal son obligatorios, si no se tienen pueden ir con el valor vacío (“”).

Es libre de definir los parámetros necesarios para su negocio, estos deben de llevar la estructura <parámetro llave = “nombre_campo” valor=”valor”/>

El siguiente es un ejemplo de un archivo o cadena XML con dos órdenes listas para ser enviadas.




```
<Coleccion>
    <NewOrder id="811218" type="Cobranza_Grupal" version="1.0" userName="opaz" priority="1" expirationDate="04/17/2013 17:49:15" cancellationDate="04/17/2013 17:49:15">
    <Parametros>
            <parametro llave="Nombre" valor="MAGNOLIA XXXXX MARQUEZ" />
            <parametro llave="Estado" valor="DF" />
            <parametro llave="Calle" valor="Amores 2019" />
            <parametro llave="Colonia" valor="Acacias" />
            <parametro llave="Municipio" valor="Benito Juarez" />
            <parametro llave="Ciudad" valor="México" />
            <parametro llave="Codigo Postal" valor="03240" />
            <parametro llave="folio_subsidio" valor="10563841" />
            <parametro llave="fecha_operacion" valor="03/09/2012" />
            <parametro llave="nombre_completo" valor="MAGNOLIA XXXXX MARQUEZ" />
            <parametro llave="curp_beneficiario" valor="XXXXXXXXXXXXXXXXXX" />
            <parametro llave="monto_sub_federal" valor="8421.05" />
            <parametro llave="regional" valor="CAMPECHE" />
            <parametro llave="especialista" valor="PAZ LOPEZ XXXX XXXXX" />
            <parametro llave="oficina" valor="ESCARCEGA" />
        </Parametros>
    </NewOrder>
    <NewOrder id="811219" type="Cobranza_Grupal" version="1.0" userName="opaz" priority="1" expirationDate="04/17/2013 17:49:15" cancellationDate="04/17/2013 17:49:15">
    <Parametros>
            <parametro llave="Nombre" valor="ROBERTO XXXXX RUIZ" />
            <parametro llave="Estado" valor="DF" />
            <parametro llave="Calle" valor="Lopez Cotilla 727" />
            <parametro llave="Colonia" valor="Del Valle" />
            <parametro llave="Municipio" valor="Benito Juarez" />
            <parametro llave="Ciudad" valor="México" />
            <parametro llave="Codigo Postal" valor="03100" />
            <parametro llave="folio_subsidio" valor="10563841" />
            <parametro llave="fecha_operacion" valor="03/09/2012" />
            <parametro llave="nombre_completo" valor="ROBERTO XXXXX RUIZ" />
            <parametro llave="curp_beneficiario" valor="XXXXXXXXXXXXXXXXXX" />
            <parametro llave="monto_sub_federal" valor="7431.05" />
            <parametro llave="regional" valor="CAMPECHE" />
            <parametro llave="especialista" valor="PAZ LOPEZ XXXX XXXXX" />
            <parametro llave="oficina" valor="ESCARCEGA" />
        </Parametros>
    </NewOrder>
</Coleccion>
```

Si tiene problemas para crear una cadena XML le proporcionamos una función que puede servirle.

```
        static string GenerarXmlOrdenes(
           DataTable dtOrdenes
           , string type
           , string version
           , string priority
           , string expirationDate
           , string cancellationDate
           )
       {

            XmlDocument xmlOrdenes = new XmlDocument();
            XmlElement root = xmlOrdenes.CreateElement("Coleccion");
            xmlOrdenes.AppendChild(root);

            string stringXMLOrdenes = xmlOrdenes.OuterXml;

            string[] camposEncabezado = {"id","userName"};

            foreach (DataRow row in dtOrdenes.Rows)
```

```
                {
                    XmlElement nNewOrder = xmlOrdenes.CreateElement("NewOrder");
                    nNewOrder.SetAttribute("id", row["externalid"].ToString());
                    nNewOrder.SetAttribute("type", type);
                    nNewOrder.SetAttribute("version", version);
                    nNewOrder.SetAttribute("userName", row["username"].ToString());
                    nNewOrder.SetAttribute("priority", priority);
                    nNewOrder.SetAttribute("expirationDate", expirationDate);
                    nNewOrder.SetAttribute("cancellationDate", cancellationDate);

                    XmlElement nParametros = xmlOrdenes.CreateElement("Parametros");
                    root.AppendChild(nNewOrder);
                    nNewOrder.AppendChild(nParametros);

                    for (int i = 0; i <= dtOrdenes.Columns.Count-1; i++)
                    {
                        Array results = Array.FindAll(camposEncabezado, s =>
s.Equals(dtOrdenes.Columns[i].ColumnName));

                        if (results.Length < 1)
                        {
                            XmlElement nparametro = xmlOrdenes.CreateElement("parametro");
                            nparametro.SetAttribute("llave", dtOrdenes.Columns[i].ColumnName);
                            nparametro.SetAttribute("valor",
row[dtOrdenes.Columns[i].ColumnName].ToString());
                            nParametros.AppendChild(nparametro);
                        }
                    }
                }

            string cadenaXML = xmlOrdenes.OuterXml;
            return cadenaXML;
        }
```

Utilizar los servicios de Formiik para cancelar órdenes de trabajo

1. Abrir Visual Studio
2. Crear un nuevo proyecto – para fines de este ejemplo se utilizará un proyecto de tipo consola.
3. Utilizar el .Net Framework 4
4. Escribir nombre de FormiikCancelaOrdenes
5. Seleccionar **OK**
6. Agregar la referencia a los servicios web de Formiik con el botón derecho del ratón sobre el folder de **References** seleccionar **Add Service Reference**
7. En **Address** escribir http://services.formiik.com:8081/BackEnd.svc?wsdl
8. Oprimir **Go**

9. Escribir el siguiente código para cancelar respuestas

```
using System;
using System.Data;
using System.IO;
using System.Text;
using System.Xml;
using FormiikCancelaOrdenes.WebServiceFmk;
namespace FormiikCancelaOrdenes
{
    class Program
    {
        static void Main(string[] args)
        {
            string IdClient = "BB99DB2A-FDB6-4104-B629-7A15A1B0E339";
            string IdProduct = "AB3B220D-000F-4C30-A96F-EB8A73637781";
```

```
            StreamReader objReader;
            objReader = new StreamReader("C:\\cancelaciones\\cancelaciones.xml");
            string strXML = objReader.ReadToEnd();

            //CancelWorkOrdersXML
            //Este servicio sirve para cancelar una conjunto de ordenes y borrarlas del
            //dispositivo móvil.
            //-----------------------
            //clientId          Identificador con el cual será reconocido el cliente
            //productId         Identificador con el que se reconocerá la aplicación del cliente
            //cancelOrders      XML con N número de ordenes a cancelar.

            try
            {
                BackEndClient bkclient = new BackEndClient("BasicHttpBinding_IBackEnd");
                string idFormiik = bkclient.CancelWorkOrdersXMLId(IdClient, IdProduct, strXML);

                Console.WriteLine("Se cancelaron las ordenes con exito!");
                Console.WriteLine("Presione cualquier tecla para continuar..");
                Console.ReadKey();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error {0}", ex.Message);
                Console.WriteLine("Presione cualquier tecla para continuar..");
                Console.ReadKey();
            }
        }
    }
}
```

El siguiente es un ejemplo de un archivo o cadena XML con dos órdenes listas para ser canceladas.

```
<Coleccion>
   <CancelOrder id="756458" username="rlopez" />
   <CancelOrder id="756459" username="rlopez" />
   <CancelOrder id="756460" username="rlopez" />
   <CancelOrder id="756461" username="rlopez" />
   <CancelOrder id="756462" username="rlopez" />
   <CancelOrder id="756463" username="lgarcia" />
   <CancelOrder id="756464" username="lgarcia" />
   <CancelOrder id="756465" username="lgarcia" />
   <CancelOrder id="756466" username="lgarcia" />
</Coleccion>
```

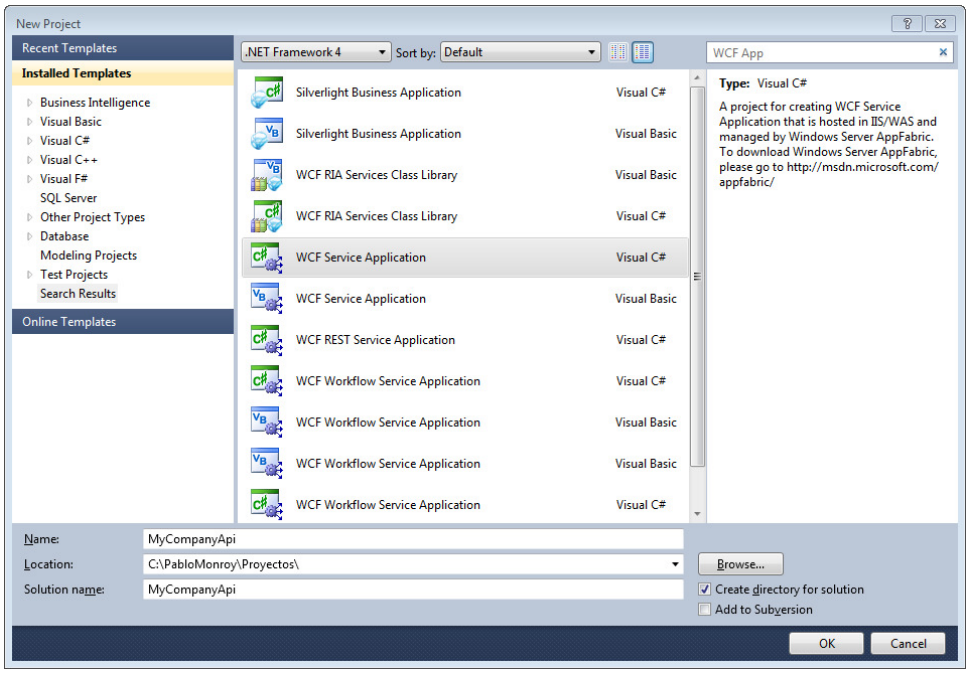
Servicios REST

El código de los servicios REST se encuentra también disponible en:
https://www.dropbox.com/s/0tn0nppxr9pr881/MyCompanyApi.zip?dl=0

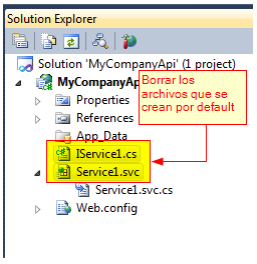
Crear los servicios web para validar usuarios

- ValidateUserSimpleReturnGroup
- ValidateUserForDeviceReturnGroup

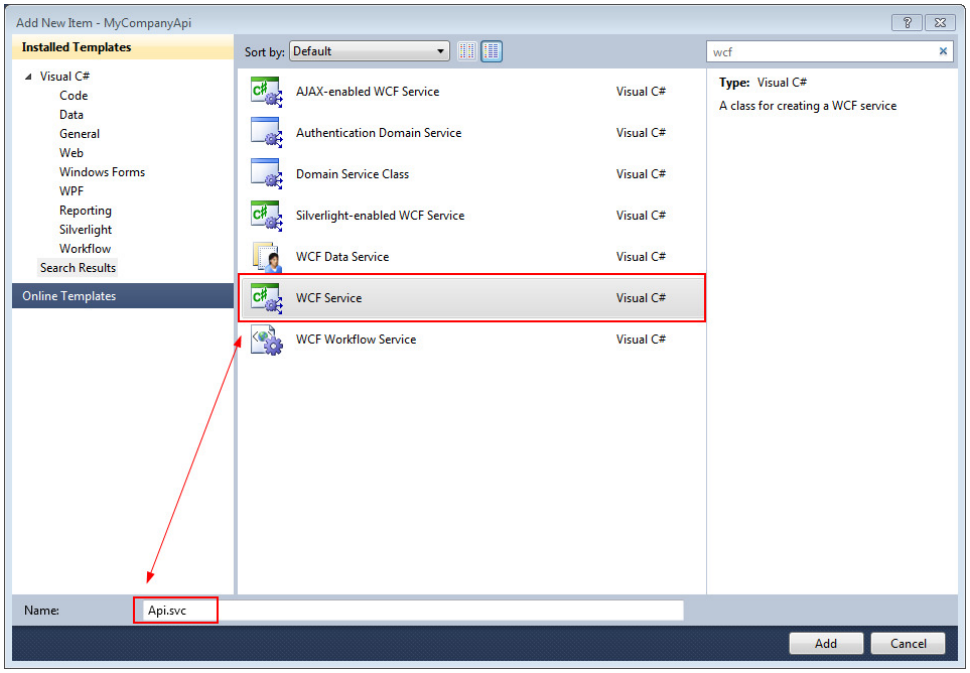
1. En primer lugar, abrir Visual Studio 2010.
2. Haga clic en Archivo-> Nuevo-> Proyecto
3. Seleccione **WCF Service Application**.



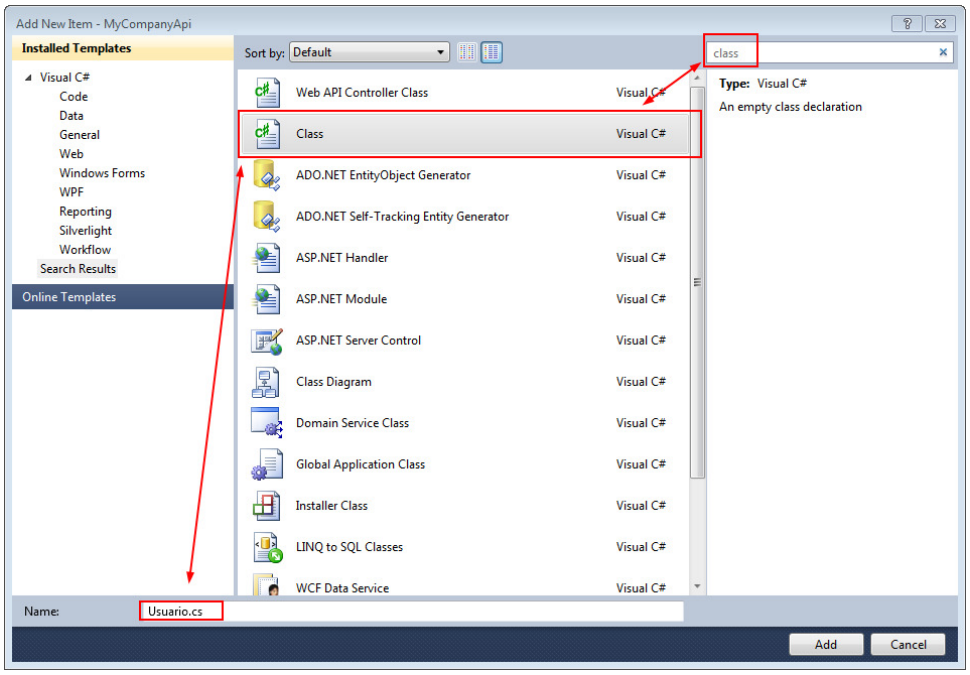
4. Una vez creado el proyecto, eliminar los archivos credos por defecto y ya que vamos a crear nuestro propio de interfaz y nuestro archivo de servicio.



5. Ahora haga clic derecho en el proyecto MyCompanyApi y seleccione Add -> New Item..
6. Seleccionar WCF Service y nombrarlo "Api.svc".



7. Ahora haga clic derecho en la solución y crear un nuevo archivo de clase y le damos el nombre de Usuario.cs



Vamos a estar escribiendo una API que puede devolver datos en formato XML, aquí está la interfaz para eso.

8. En **IApi.svc**, agregue el siguiente código (el código a continuación es para los servicios más utilizados que debe desarrollar para Formiik):

```
using System.IO;
using System.ServiceModel;
using System.ServiceModel.Web;
using MyCompanyApi.FlexibleEntity;
namespace MyCompanyApi
{
    [ServiceContract]
    public interface IApi
    {
        //Expone ValidateUserSimpleReturnGroup
        [OperationContract]
        [WebInvoke(Method = "POST",
            RequestFormat = WebMessageFormat.Xml,
            ResponseFormat = WebMessageFormat.Xml,
            UriTemplate = "ValidateUserSimpleReturnGroup"
            )]
        Stream ValidateUserSimpleReturnGroup(Stream usuario);

        //Expone ValidateUserForDeviceReturnGroup
        [OperationContract]
        [WebInvoke(Method = "POST",
            RequestFormat = WebMessageFormat.Xml,
            ResponseFormat = WebMessageFormat.Xml,
            UriTemplate = "ValidateUserForDeviceReturnGroup"
            )]
        Stream ValidateUserForDeviceReturnGroup(Stream usuario);
      }
}
```

9. En el archivo **Api.svc** agregue el siguiente código

```
using System;
using System.IO;
using System.Text;
using System.Web.Script.Serialization;
using System.Xml;
using MyCompanyApi.FlexibleEntity;
using System.DirectoryServices;

namespace MyCompanyApi
{
    public class Api : IApi
    {
        //Implementación ValidateUserForDeviceReturnGroup
        public Stream ValidateUserForDeviceReturnGroup(Stream streamUserDevice)
        {
            StreamReader reader = new StreamReader(streamUserDevice);
            string xmlUser = reader.ReadToEnd();
            XmlDocument xmlUserDoc = new XmlDocument();
            //Carga la cadena string a un XmlDoc
            xmlUserDoc.LoadXml(xmlUser);
            Usuario usuario = new Usuario();
            usuario.UserName = xmlUserDoc.GetElementsByTagName("username").Item(0).InnerText.ToString();
            usuario.Password = xmlUserDoc.GetElementsByTagName("password").Item(0).InnerText.ToString();
            usuario.SerialNumber =
xmlUserDoc.GetElementsByTagName("serialnumber").Item(0).InnerText.ToString();

            //1.- BUscar en la Base de Datos por el usuario, password y imei
            /*
             Código del cliente
            */

            //2.- BUscar en la Directorio Activo por el usuario, password y imei

            bool esValidoa = IsAuthenticated("puerto", usuario.UserName, usuario.Password);
```

```
            string respuesta = @"<Authentication>
                <Processes>
                    <ProcessId>9A6EEDE6-D51F-4B98-8A8B-83E5E266A100</ProcessId>
                </Processes>
                <GroupExternalId>VER</GroupExternalId>
                <RoleId>A1903AA949FB4CE19F7EFEC793898DE2</RoleId>
            </Authentication>";

            if (esValidoa)
            {

                return new MemoryStream(Encoding.UTF8.GetBytes(respuesta));
            }
            else
            {
                return new MemoryStream(Encoding.UTF8.GetBytes("Usuario o password inválido"));
            }
        }

        //ValidateUserSimpleReturnGroup
        public Stream ValidateUserSimpleReturnGroup(Stream streamUserDevice)
        {
            StreamReader reader = new StreamReader(streamUserDevice);
            string xmlUser = reader.ReadToEnd();
            XmlDocument xmlUserDoc = new XmlDocument();
            //Carga la cadena string a un XmlDoc
            xmlUserDoc.LoadXml(xmlUser);
            Usuario usuario = new Usuario();
            usuario.UserName = xmlUserDoc.GetElementsByTagName("username").Item(0).InnerText.ToString();
            usuario.Password = xmlUserDoc.GetElementsByTagName("password").Item(0).InnerText.ToString();


            //1.- BUscar en la Base de Datos por el usuario, password y imei
            /*
             Código del cliente
             */

            //2.- BUscar en la Directorio Activo por el usuario y password si busca grupo

            bool esValidoa = IsAuthenticated("puerto",usuario.UserName, usuario.Password);

            string respuesta = @"<Authentication>
                <Processes>
                    <ProcessId>9A6EEDE6-D51F-4B98-8A8B-83E5E266A100</ProcessId>
                </Processes>
                <GroupExternalId>VER</GroupExternalId>
                <RoleId>A1903AA949FB4CE19F7EFEC793898DE2</RoleId>
            </Authentication>";

            if (esValidoa)
            {

                return new MemoryStream(Encoding.UTF8.GetBytes(respuesta));
            }
            else
            {
                return new MemoryStream(Encoding.UTF8.GetBytes("Usuario o password inválido"));
            }
        }
    }
}
```

10. En el archivo **Usuario.cs** agregar el siguiente código.

```
using System.Runtime.Serialization;

namespace MyCompanyApi
{
    [DataContract(Namespace = "")]
    public class Usuario
    {
        [DataMember]
        public string UserName { get; set; }

        [DataMember]
        public string Password { get; set; }

        [DataMember]
        public string SerialNumber { get; set; }

        public string Login()
        {
            //Conectarse a su base
            //Hacer tu consulta la base de datos
            //Si lo encuentra regresar true de lo contrario false
            bool existe = false;

            if (existe)
            {
                return "";
            }
            else
            {
                return "Usuario/Password no valido";
            }
        }
    }
}
```

11. El archivo **Web.config** debe de quedar como se muestra

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.0" />
  </system.web>
  <system.serviceModel>
    <services>
      <service name="MyCompanyApi.Api"
               behaviorConfiguration="myServiceBehavior">
        <!-- Service Endpoints -->
        <!-- Unless fully qualified, address is relative to base address supplied above -->
        <endpoint name ="webHttpBinding"
                  address =""
                  binding="webHttpBinding"
                  contract="MyCompanyApi.IApi"
                  behaviorConfiguration="webHttp"/>
        <endpoint name="mexHttpBinding"
                  address="mex"
                  binding="mexHttpBinding"
                  contract="IMetadataExchange"
                  />
      </service>
    </services>
    <bindings>
      <basicHttpBinding>
        <binding maxReceivedMessageSize="2147483647" maxBufferSize="2147483647"
maxBufferPoolSize="2147483647" closeTimeout="00:05:00" openTimeout="00:05:00" receiveTimeout="00:10:00"
sendTimeout="00:05:00">
          <readerQuotas maxDepth="2147483647" maxStringContentLength="2147483647"
```

```
maxArrayLength="2147483647" maxBytesPerRead="2147483647"  maxNameTableCharCount="2147483647" />
        </binding>
      </basicHttpBinding>
      <webHttpBinding>
        <binding maxBufferSize="2147483647" maxReceivedMessageSize="2147483647" />
      </webHttpBinding>
    </bindings>
    <behaviors>
      <serviceBehaviors>
        <behavior name="myServiceBehavior">
          <!-- To avoid disclosing metadata information, set the value below to false and remove the
metadata endpoint above before deployment -->
          <serviceMetadata httpGetEnabled="true"/>
          <!-- To receive exception details in faults for debugging purposes, set the value below to
true.  Set to false before
          deployment to avoid disclosing exception information -->
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
      <endpointBehaviors>
        <behavior name="webHttp">
          <webHttp/>
        </behavior>
      </endpointBehaviors>
    </behaviors>
    <serviceHostingEnvironment multipleSiteBindingsEnabled="true" />
  </system.serviceModel>
  <system.webServer>
    <modules runAllManagedModulesForAllRequests="true"/>
  </system.webServer>
</configuration>
```

Ya que tenemos todo listo para probar nuestro servicio tipo REST llamado **ValidateUserSimpleReturnGroup**

Probar el Servicio ValidateUserSimpleReturnGroup

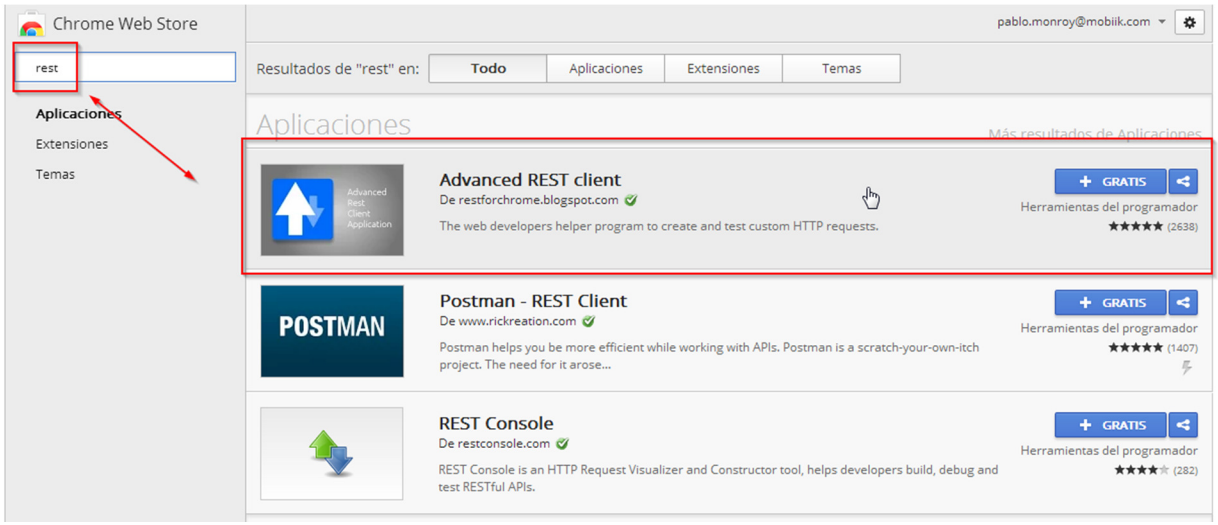
(Para más detalles, la firma del método se encuentra en el documento ***ServiciosFormiikSOAP_REST_2.docx*** o ***ServiciosFormiikSOAP_REST_2.pdf***)

Para probar el servicio se puede hacer de dos formas:

- Creando un cliente REST con C# y Visual Studio 2010
- Instalando un cliente REST gratuito

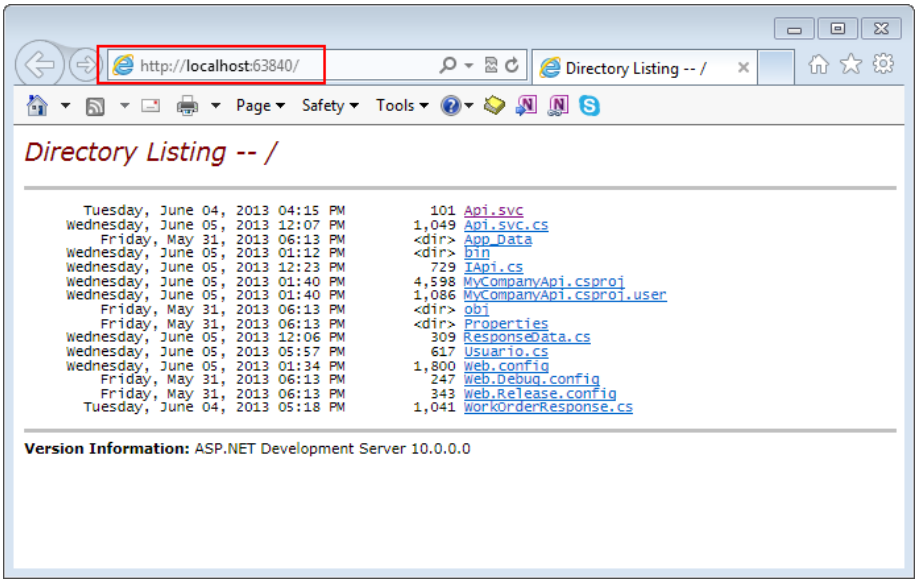
Para probarlo nosotros vamos a usar la segunda opción instalar una extensión en Google Chrome

1. Entrar a la página de Google Chrome Store http://goo.gl/uCDqf
2. Buscamos REST e instalamos la extensión **Advanced – REST Client**



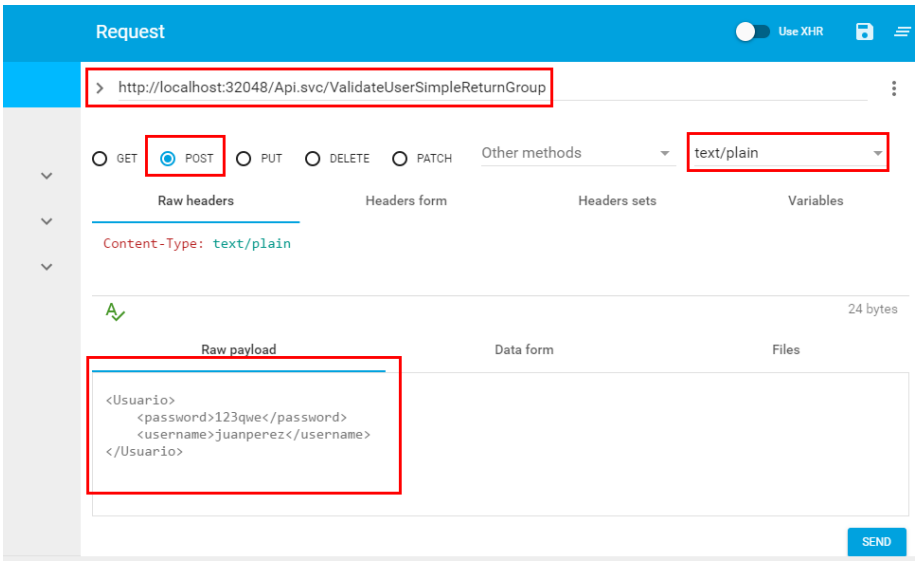
Una vez que lo tenemos instalado regresamos a Visual Studio 2010

3. Ejecutamos la aplicación **MyCompanyApi**
4. Se muestra IE



Lo dejamos tal como está y abrimos o nos cambiamos a Google Chrome.

5. En Google Chrome ejecutamos el cliente **Advanced REST Client** que recientemente instalamos.



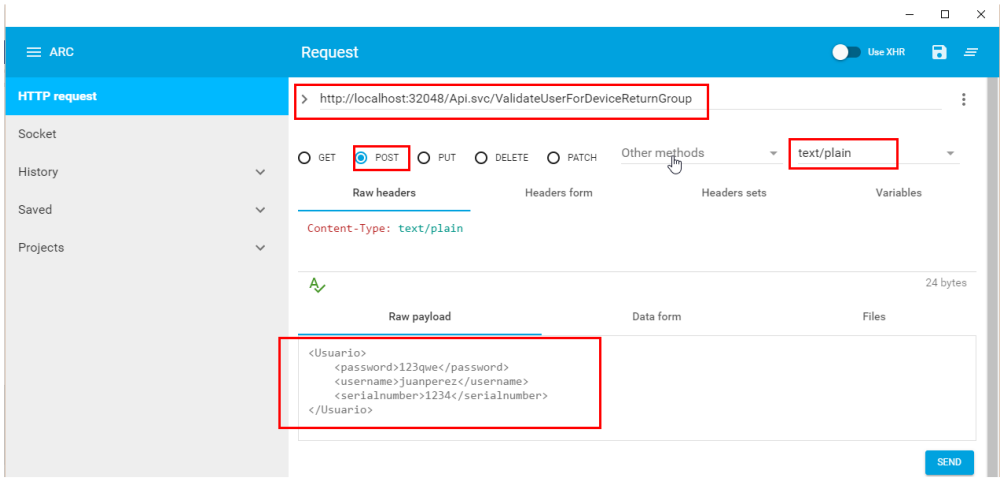
6. Capturamos la url del servicio que nos muestra en el navegador y la complementamos con el nombre del servicio y el método de ValidateUserSimple
7. Seccionamos POST
8. Seleccionamos Raw payload
9. Escribimos los parámetros que vamos a enviar al método
10. Seleccionamos que los datos enviados serán de tipo text/plain (si no existe, darlo de alta)
11. Enviamos los parámetros
12. Recibimos una respuesta del método.

En caso de ser exitosa deberá arrojar un XML con el grupo al que perrtenece el usuario y los procesos donde puede trabajar.

```
<Authentication>
    <Processes>
        <ProcessId>98E2A552-86FC-40B2-909F-7C92E840FE3E </ProcessId>
        <ProcessId> E2883B32-58F0-4614-970F-FAF67BFCBF65</ProcessId>
        </Processes>
    <GroupExternalId>001</GroupExternalId>
    <RoleId>A1903AA9-49FB-4CE1-9F7E-FEC793898DE2</RoleId>
</Authentication>
```

Probar el Servicio ValidateUserForDeviceReturnGroup

1. Como lo hicimos con el servicio anterior, vamos a probar este nuevo servicio.



En caso de ser exitosa deberá arrojar un XML con el grupo al que perrtenece el usuario y los procesos donde puede trabajar.

```
<Authentication>
    <Processes>
        <ProcessId>98E2A552-86FC-40B2-909F-7C92E840FE3E </ProcessId>
        <ProcessId> E2883B32-58F0-4614-970F-FAF67BFCBF65</ProcessId>
        </Processes>
    <GroupExternalId>001</GroupExternalId>
    <RoleId>A1903AA9-49FB-4CE1-9F7E-FEC793898DE2</RoleId>
</Authentication>
```

Crear el servicio web para recibir respuestas

1. En primer lugar abra su proyecto MyCompanyApi
2. Agregar el código señalado en la pantalla en el archivo IApi.cs

De esta manera creamos la interface para el método SendWorkOrderToClient

```
        //Expone SendWorkOrderToClient
        [OperationContract]
        [WebInvoke(Method = "POST",
            RequestFormat = WebMessageFormat.Xml,
            ResponseFormat = WebMessageFormat.Xml,
            UriTemplate = "SendWorkOrderToClient"
            )]
        Stream SendWorkOrderToClient(Stream respuesta);
```

3. Agregar un nuevo archivo de clase, como lo vimos en el método anterior, llamado **WorkOrderResponse.cs** y escribir el siguiente código

```
using System;
using System.Data;
using System.IO;
using System.Runtime.Serialization;
using System.Text;
using System.Xml;

namespace MyCompanyApi
{
    [DataContract(Namespace = "")]
    public class WorkOrderResponse
    {
        public string ProductId { get; set; }
        public string ExternalId { get; set; }
        public string ExternalType { get; set; }
        public string AssignedTo { get; set; }
        public string InitialDate { get; set; }
        public string FinalDate { get; set; }
        public string ResponseDate { get; set; }
        public string InitialLatitude { get; set; }
        public string FinalLatitude { get; set; }
        public string InitialLongitude { get; set; }
        public string FinalLongitude { get; set; }
        public string FormiikResponseSource { get; set; }
        public string XmlResponse { get; set; }
        public string XmlFullResponse { get; set; }

        public void Load(Stream strXML)
        {
            StreamReader reader = new StreamReader(strXML);
            string text = reader.ReadToEnd();

            XmlDocument xmlworkorderresponse = new XmlDocument(); //xmldoc Respuesta Completa
            XmlDocument xmlresponse = new XmlDocument();          //xmldoc Respuesta Detalle

            //Carga la cadena string a un XmlDoc
            xmlworkorderresponse.LoadXml(text);

            XmlElement root = xmlworkorderresponse.DocumentElement;
            XmlNode nodeXmlResponse = root.FirstChild;

            XmlNode newNode = xmlworkorderresponse.ImportNode(nodeXmlResponse, true);

            xmlresponse.LoadXml(newNode.OuterXml);
```

```
            this.ProductId = root.Attributes["ProductId"].Value;
            this.ExternalId = root.Attributes["ExternalId"].Value;
            this.ExternalType = root.Attributes["ExternalType"].Value;
            this.AssignedTo = root.Attributes["AssignedTo"].Value;
            this.InitialDate = root.Attributes["InitialDate"].Value;
            this.FinalDate = root.Attributes["FinalDate"].Value;
            this.ResponseDate = root.Attributes["ResponseDate"].Value;
            this.InitialLatitude = root.Attributes["InitialLatitude"].Value;
            this.FinalLatitude = root.Attributes["FinalLatitude"].Value;
            this.InitialLongitude = root.Attributes["InitialLongitude"].Value;
            this.FinalLongitude = root.Attributes["FinalLongitude"].Value;
            this.FormiikResponseSource = root.Attributes["FormiikResponseSource"].Value;
            this.XmlResponse = xmlresponse.InnerXml.ToString();
            this.XmlFullResponse = xmlworkorderresponse.InnerXml.ToString();
        }

        public Stream Save()
        {
            string fileLogName = string.Format("C:\\MyOrderResponses\\ordenes-{0:yyyy-MM-dd}.csv",
DateTime.Now);
            StreamWriter log;
            try
            {
                if (!File.Exists(fileLogName))
                {
                    log = new StreamWriter(fileLogName);
                }
                else
                {
                    log = File.AppendText(fileLogName);
                }

                log.WriteLine(this.ToString());
                log.Close();
                return new MemoryStream(Encoding.UTF8.GetBytes(String.Empty));

            }
            catch (Exception ex)
            {
                return new MemoryStream(Encoding.UTF8.GetBytes(ex.Message));
            }
        }
    }
}
```

Este código es solo de ejemplo. El equipo de desarrollo deberá implementar su propia lógica de negocio.

4. Abrir el archivo **Api.svc.cs** y agregar el siguiente código

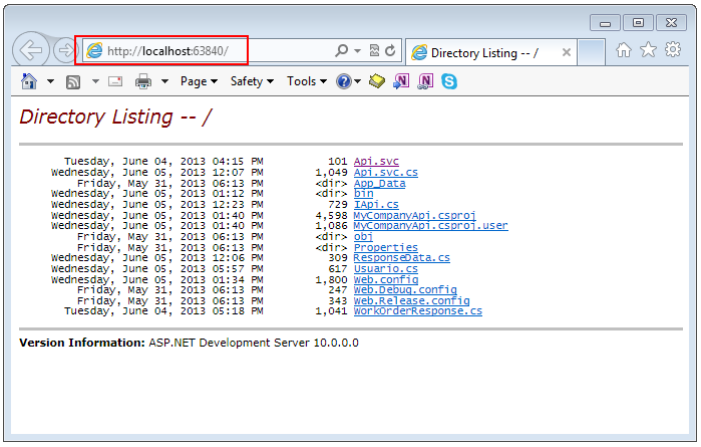
```
        //Implementación SendWorkOrderToClient
        public Stream SendWorkOrderToClient(Stream respuesta)
        {
            //Recibe las respuestas de Formiik en sus sistemas
            //Devuelve string vacío si recibió la repuesta
            WorkOrderResponse workorderresponse = new WorkOrderResponse();

            //Carga un documento XML enviado por formiik con todas las propiedades
            //de una respuesta
            //(ver objeto WorkOrderResponse
            workorderresponse.Load(respuesta);
            //A partir de este momento Ud. puede implementar algun metodo
            //para guardar las respuestas. El método que se muestra es solo un demo
```

```
            //return workorderresponse.Save();
            return workorderresponse.SaveFull();
        }
```

5. Crear una carpeta **C:\MyOrders\Responses** en el cual se guardaran las respuestas de las órdenes. El nombre del archivo se creará automáticamente y su nombre será la fecha en que llegarán las órdenes Ej. **ordenes-2017-06-07.csv**.

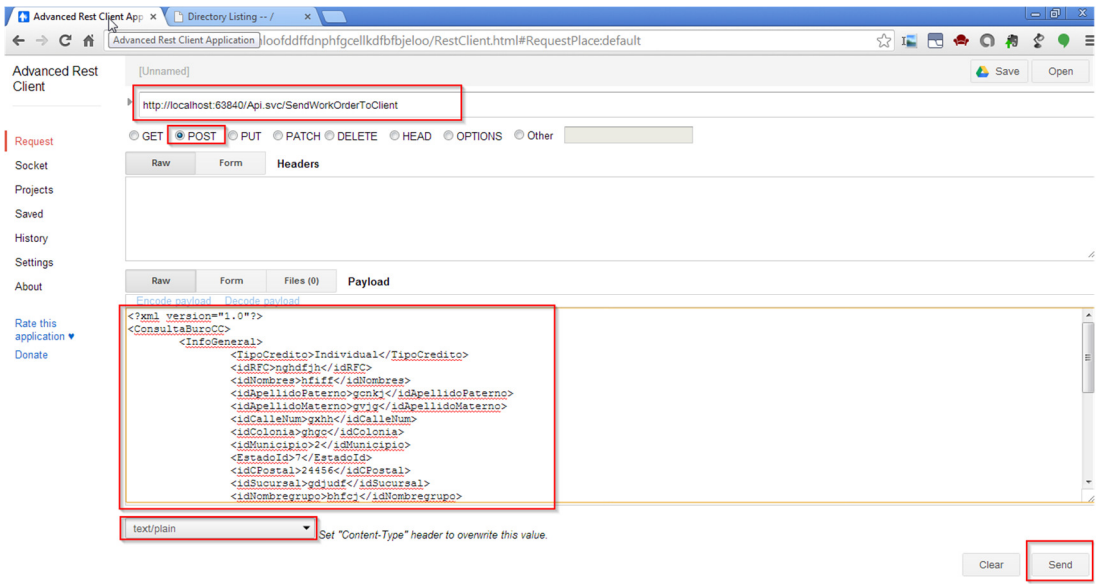
Probar el Servicio SendWorkOrderToClient

1. Ejecutamos la aplicación **MyCompanyApi**
2. Se muestra IE



Lo dejamos tal como está y abrimos o nos cambiamos a Google Chrome.

3. En Google Chrome instalamos el plugin **Advanced Test Client**.



4. Capturamos la url del servicio que nos muestra en IE y la complementamos con el nombre del servicio y el método de SendWorkOrderToClient
5. Seccionamos POST
6. Seleccionamos raw
7. Seleccionamos que los datos enviados serán de tipo text/plain
8. Escribimos los parámetros que vamos a enviar al método. Para este caso es la respuesta de una orden de trabajo. La respuesta puede ser tomada del portal o puede ser proporcionada por el equipo de Implementación y Soporte Tecnico de Formiik.
9. Enviamos los parámetros
10. Recibimos una respuesta del método.

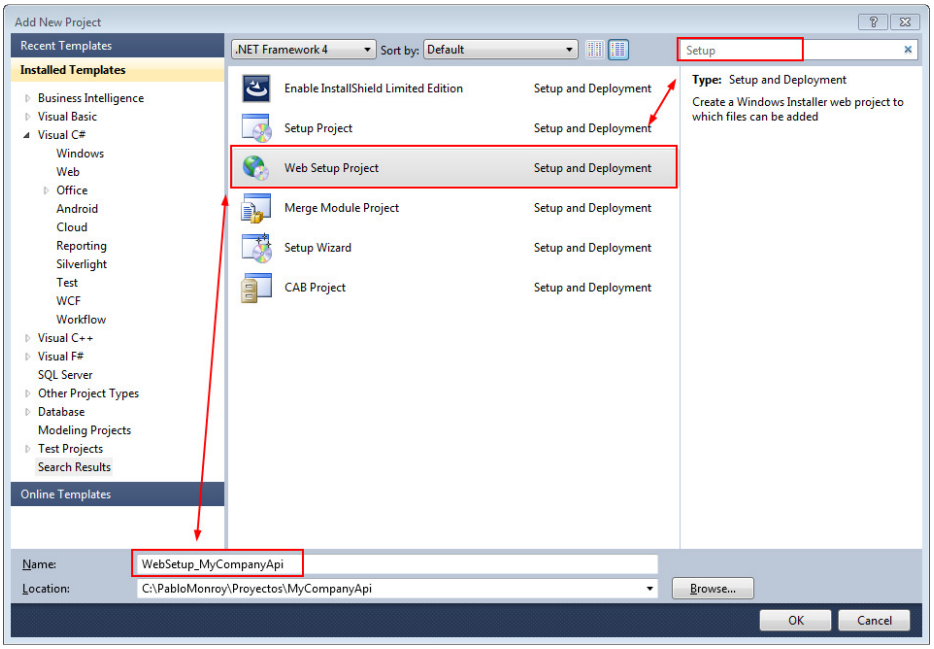
Instalación de los servicios web para validar usuarios y recibir respuestas

Una vez que se tienen desarrollados los principales servicios (ValidateUserSimple y SendWorkOrderToClient) ya podemos instalar los servicios web desarrollados en nuestro servidor.

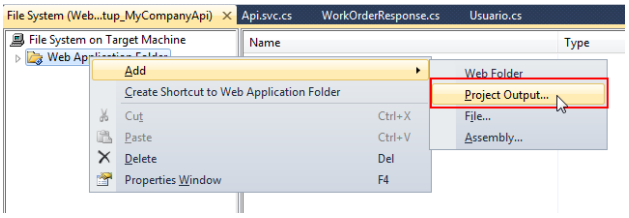
Crear Instalador de los Servicios Web

Para instalar nuestro web tenemos que crear un instalador.

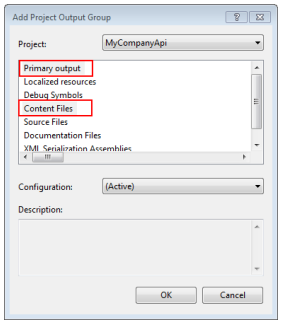
1. En el nombre de nuestra solución damos clic derecho con el mouse y seleccionamos Add->New Proyect...->Web Setup Project y le damos el nombre de **WebSetUp_MyCompanyApi**



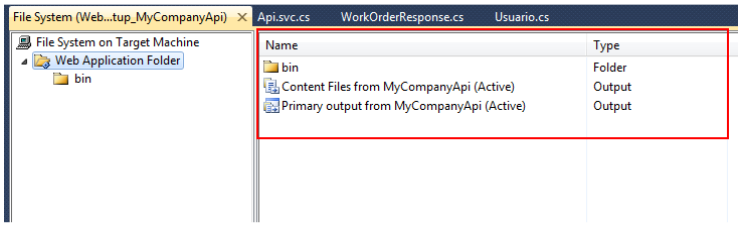
2. Nos mostrará un pantalla con las carpetas de la aplicación
3. Con el botón derecho del mouse sobre la carpeta **Web Application Folder** seleccionar Add-> Project Output



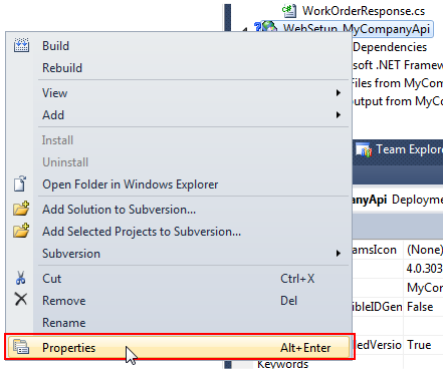
4. Seleccionamos **Primary output** y **Content Files** y damos OK



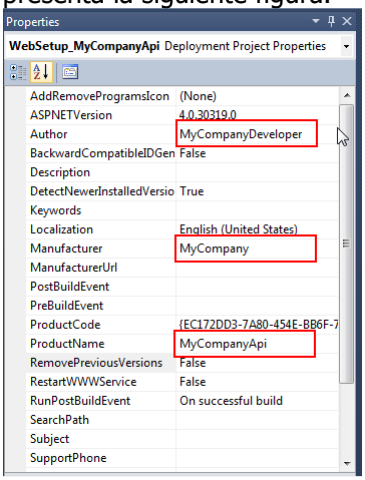
5. Nuestra aplicación muestra el nuevo contenido



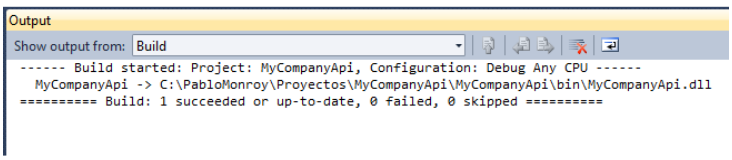
6. Opcionalmente con el botón derecho del mouse seleccionamos **Properties**



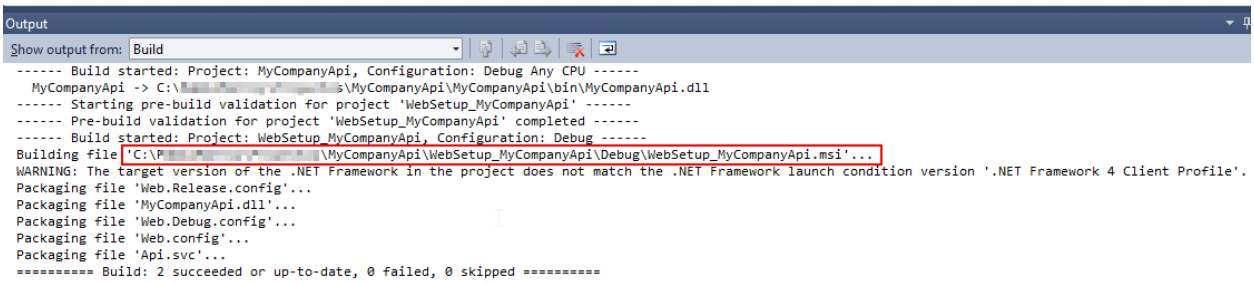
7. Cambiamos las opciones que presenta la siguiente figura.



8. Compilar ambos proyectos MyCompanyApi y WebSetup_MyCompanyApi

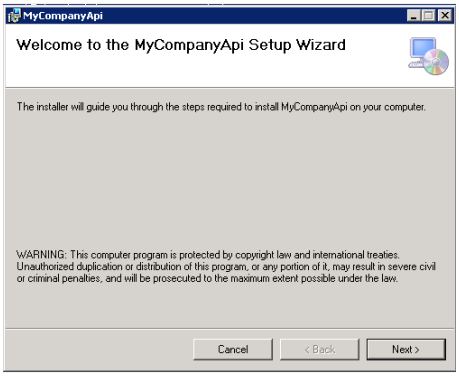


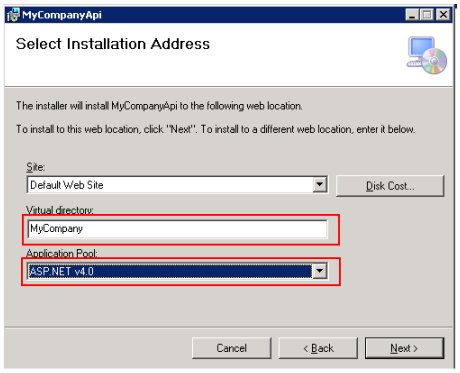
El proyecto del Setup crea un instalador en la ruta de nuestro proyecto como se muestra en la imagen.

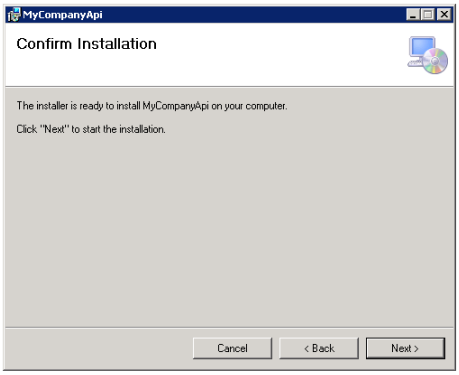


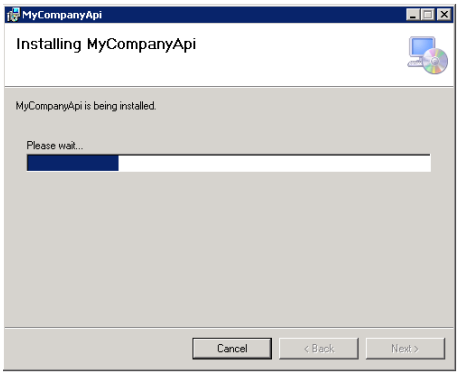
Instalar los Servicios Web en Internet

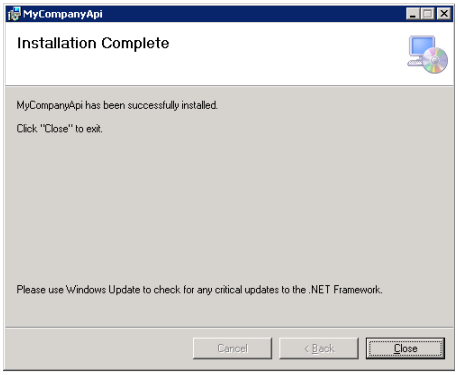
1. En un servidor **que es visible en internet** ejecutar **WebSetUp_MyCompanyApi.msi**





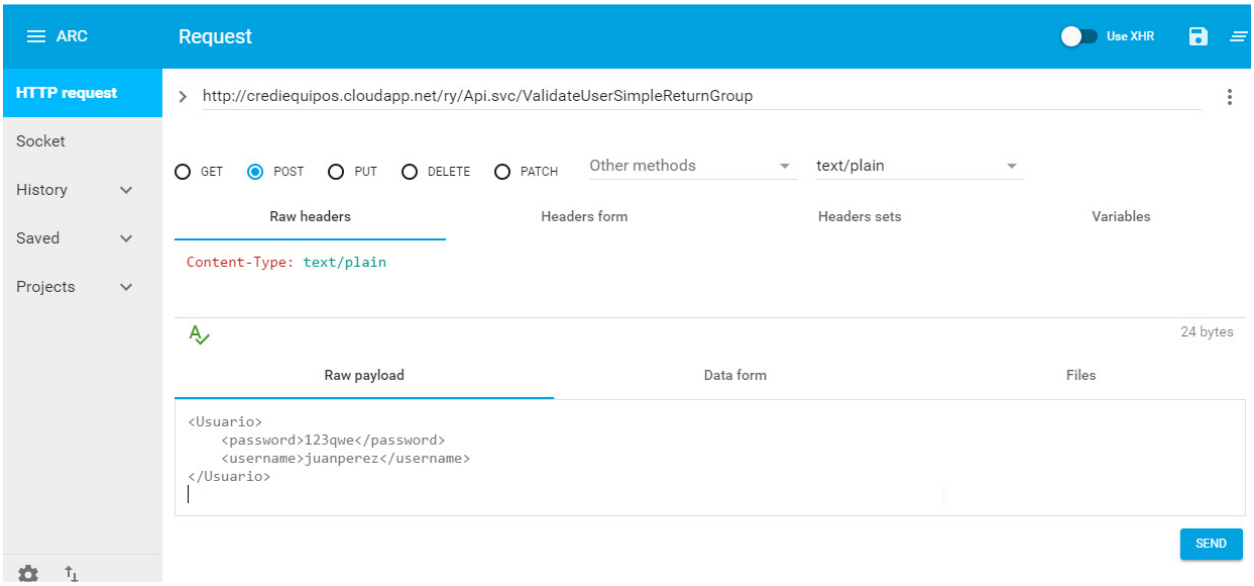




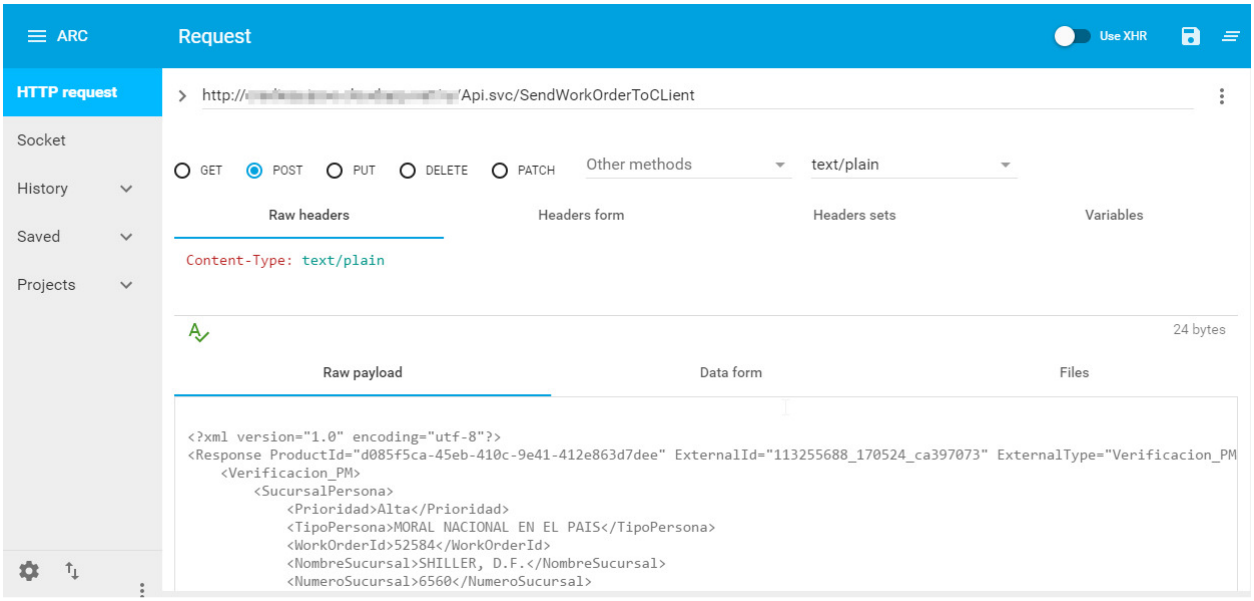


Probar los servicios web en el servidor visible en Internet

1. En Google Chrome ejecutamos el cliente **Advanced REST Client**.



2. Capturamos la url del servidor **que es visible en internet** y en el que instalamos el Servicio Web además del método **ValidateUserSimpleReturnGroup** o **SendWorkOrderToClient** como ya lo habíamos hecho en nuestro equipo local.



3. Seccionamos POST
4. Seleccionamos raw
5. Seleccionamos que los datos enviados serán de tipo text/plain
6. Escribimos los parámetros que vamos a enviar al método
7. Enviamos los parámetros
8. Recibimos una respuesta del método.

Alta de los servicios web del cliente

Una vez que se tienen instalados y probados los servicios web para validar usuarios y recibir respuestas es necesario:

1. Enviar un correo electrónico al área de **Soporte e Implementación de Formiik** al correo **soporte@formiik.com** solicitando el alta de su servicio en Formiik especificando el url de los servicios. Ej.
    a. **http://miempresa.com/servicios/api.svc/**
    b. **http://192.168.0.121/servicios/api.svc/**

2. Enviar por correo electrónico a la misma área un listado de los usuarios que comenzarán a trabajar con la prueba piloto.

Utilizar servicios adicionales de Formiik

Envío de mensajes

Formiik pone a disposición de los administradores la posibilidad de enviar mensajes a los dispositivos de los operadores.

De la misma forma que usamos los servicios de Formiik para enviar y cancelar órdenes. Puede crear un proyecto siguiendo los mismos pasos.

Use como referencia la siguiente tabla

Mensajes a Operadores

Nombre del servicio	Parámetros	Tipo de Dato	¿Requerido?	Descripción Dato	Valor de retorno	Descripción servicio
QueueMessageToUserName	clientId	string	SI	Identificador con el cual será reconocido el cliente	string	Este servicio envía mensajes a un usuario en específico. Devuelve string vacío si no hay error, si existe error devuelve una excepción.
	productId	string	SI	Identificador con el que se reconocerá la aplicación del cliente		
	userName	string	SI	Usuario al que se le enviará el mensaje		
	Sender	string	SI	Texto que describe quién es el remitente		
	Content	string	SI	Texto del contenido de mensajes		
	isImportant	string	SI	Valores permitidos: si ó no		
QueueMessageToUserNames	clientId	string	SI	Identificador con el cual será reconocido el cliente	string	Este servicio envía mensajes a una lista de usuarios. Devuelve string vacío si no hay error, si existe error devuelve una excepción.
	productId	string	SI	Identificador con el que se reconocerá la aplicación del cliente		
	userNames	string[]	SI	Arreglo de strings con los usuarios a los que se les enviará el mensaje		
	Sender	string	SI	Texto que describe quién es el remitente		
	Content	string	SI	Texto del contenido de mensajes		
	isImportant	string	SI	Valores permitidos: si ó no		

Y en su programa puede usar el siguiente código para envio individual.

```
using System;
using FormiikEnvioDeMensajes.WebServiceFmk;

namespace FormiikEnvioDeMensajes
{
    class Program
    {
        static void Main(string[] args)
        {
            string IdClient  = "6A419340-38EE-4CBB-878E-1067DDA5533C";
            string IdProduct = "3D9D2638-3FAB-4DC5-B019-CEDBA760F39D";
            string UserName = "martavm";
            string UserSender = "supervisor";
            string Mensaje = "Favor de actualizar sus datos con Recursos Humanos";
            bool EsImportante = true;

            //QueueMessageToUserName
            //Este servicio envía un conjunto de mensajes.
            //--------------------------------------------
            //Parametros
            //clientId      Identificador con el cual será reconocido el cliente
            //productId     Identificador con el que se reconocerá la aplicación del cliente
            //UserName      Destinatario.
            //UserSender    Rem itente.
            //Mensaje       Mensaje
            //EsImportante  True si es imprtante, False si no es importante
```

```
            try
            {
                BackEndClient bkclient = new BackEndClient("BasicHttpBinding_IBackEnd");

bkclient.QueueMessageToUserName(IdClient,IdProduct,UserName,UserSender,Mensaje,EsImportante);
                Console.WriteLine ("Se envió el mensaje con exito!");
                Console.WriteLine("Presione cualquier tecla para continuar..");
                Console.ReadKey();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error {0}",ex.Message);
                Console.WriteLine("Presione cualquier tecla para continuar..");
                Console.ReadKey();
            }
        }
    }
}
```

Y en su programa puede usar el siguiente código para envio grupal.

```
using System;
using FormiikEnvioDeMensajes.WebServiceFmk;

namespace FormiikEnvioDeMensajes
{
    class Program
    {
        static void Main(string[] args)
        {
            string IdClient  = "6A419340-38EE-4CBB-878E-1067DDA5533C";
            string IdProduct = "3D9D2638-3FAB-4DC5-B019-CEDBA760F39D";
            //string UserName = "martavm";
            string[] usuarios = new string[] {"pablo","luis","jaime"}
            string UserSender = "supervisor";
            string Mensaje = "Favor de actualizar sus datos con Recursos Humanos";
            bool EsImportante = true;

            //QueueMessageToUserName
            //Este servicio envía un conjunto de mensajes.
            //-------------------------------------------
            //Parametros
            //clientId      Identificador con el cual será reconocido el cliente
            //productId     Identificador con el que se reconocerá la aplicación del cliente
            //UserName      Destinatario.
            //UserSender    Rem itente.
            //Mensaje       Mensaje
            //EsImportante  True si es imprtante, False si no es importante

            try
            {
                BackEndClient bkclient = new BackEndClient("BasicHttpBinding_IBackEnd");

//bkclient.QueueMessageToUserName(IdClient,IdProduct,UserName,UserSender,Mensaje,EsImportante);

bkclient.QueueMessageToUserNames(IdClient,IdProduct,usuarios,UserSender,Mensaje,EsImportante);
                Console.WriteLine ("Se envió el mensaje con exito!");
                Console.WriteLine("Presione cualquier tecla para continuar..");
                Console.ReadKey();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error {0}",ex.Message);
                Console.WriteLine("Presione cualquier tecla para continuar..");
                Console.ReadKey();
            }
```

```
        }
    }
}
```

Crear servicios web adicionales

Actualización de catálogos.

Los operadores en sus dispositivos, al contestar órdenes siempre utilizan catálogos: Estado, Municipio, Tipo de Producto, etc. De acuerdo al tipo de negocio.

Use como referencia la siguiente tabla

Nombre del servicio	Parámetros XML	Parámetros	Tipo de Dato	¿Requerido?	Descripción Dato	Valor de retorno	Descripción servicio
GetUserCatalog	<UsuarioDevice> <password>123qwe</password> <username>pmonroy</username> </UsuarioDevice>	username	string	SI	Usuario por el que se limitará la búsqueda	String	Devuelve los catálogos al dispositivo en una cadena de texto que contiene el XML. Este XML contiene los catálogos para ese usuario, por ejemplo puede implicar código postal y colonias de la zona a la que está asignado el usuario.
		password	string	SI	Contraseña con la que se firmará el usuario		

Este XML contiene los catálogos para el usuario (Valor de retorno)

```
<UserCatalog>
    <Catalog Name="Status">
        <registry>
            <valor>BPE - Bonificacion pendiente</valor>
            <clave>BPE</clave>
        </registry>
        <registry>
            <valor>CFA - Cliente fallecio</valor>
            <clave>CFA</clave>
        </registry>
        <registry>
            <valor>DCS - Desacuerdo con saldo</valor>
            <clave>DCS</clave>
        </registry>
        <registry>
            <valor>DPA - Devolución pendiente x aplicar</valor>
            <clave>DPA</clave>
        </registry>
        <registry>
            <valor>NRC - No reconoce compra</valor>
            <clave>NRC</clave>
        </registry>
        <registry>
            <valor>MDE - Mercancia defectuosa</valor>
            <clave>MDE</clave>
        </registry>
    </Catalog>
    <Catalog Name="Turnos">
        <registry>
            <valor>BPE - Bonificacion pendiente</valor>
            <clave>BPE</clave>
        </registry>
        <registry>
            <valor>BPE - Bonificacion pendiente</valor>
            <clave>BPE</clave>
        </registry>
        <registry>
            <valor>BPE - Bonificacion pendiente</valor>
            <clave>BPE</clave>
        </registry>
    </Catalog>
</UserCatalog>
```

En el archivo **IApi.cs** agregue el siguiente código:

```
        //Expone GetUserCatalog
        [OperationContract]
        [WebInvoke(Method = "POST",
            RequestFormat = WebMessageFormat.Xml,
            ResponseFormat = WebMessageFormat.Xml,
            UriTemplate = "GetUserCatalog"
            )]
                            Stream GetUserCatalog(Stream usuario);
```

En el archivo **Api.cs** agregue el siguiente código:

```
        public Stream GetUserCatalog(Stream streamUser)
        {
            //Permite a Formiik recuperar los catalogos del usuario
            //Ud. debe de formar una cadena XML con los catalogos
            //y retonarla a formiik
            StreamReader reader = new StreamReader(streamUser);
            string xmlUser = reader.ReadToEnd();
            XmlDocument xmlUserDoc = new XmlDocument();
            //Carga la cadena string a un XmlDoc
            xmlUserDoc.LoadXml(xmlUser);
            Usuario usuario = new Usuario();
            usuario.UserName = xmlUserDoc.GetElementsByTagName("username").Item(0).InnerText.ToString();
            usuario.Password = xmlUserDoc.GetElementsByTagName("password").Item(0).InnerText.ToString();
            //El siguinete es solo un demo que toma los catalogos de un archivo
            //En su caso buscará los catalogos correspondientes del usuario
            StreamReader objReader;
            objReader = new StreamReader("C:\\Folder\\catalogos.xml");
            string strXML = objReader.ReadToEnd();
            return new MemoryStream(Encoding.UTF8.GetBytes((strXML.ToString())));
        }
```

Ejemplo de archivo de catálogos

```
<UserCatalog>
    <Catalog Name="Status">
        <registry>
            <valor>BPE - Bonificacion pendiente</valor>
            <clave>BPE</clave>
        </registry>
        <registry>
            <valor>CFA - Cliente fallecio</valor>
            <clave>CFA</clave>
        </registry>
        <registry>
            <valor>DCS - Desacuerdo con saldo</valor>
            <clave>DCS</clave>
        </registry>
        <registry>
            <valor>DPA - Devolución pendiente x aplicar</valor>
            <clave>DPA</clave>
        </registry>
        <registry>
            <valor>NRC - No reconoce compra</valor>
            <clave>NRC</clave>
        </registry>
        <registry>
            <valor>MDE - Mercancia defectuosa</valor>
            <clave>MDE</clave>
        </registry>
    </Catalog>
    <Catalog Name="Turnos">
        <registry>
            <valor>BPE - Bonificacion pendiente</valor>
            <clave>BPE</clave>
        </registry>
        <registry>
            <valor>BPE - Bonificacion pendiente</valor>
            <clave>BPE</clave>
        </registry>
    </Catalog>
</UserCatalog>
```

Informe de errores

Formiik informa si se han producido errores al mandar órdenes de trabajo de manera asíncrona, así que debemos estar preparados para recibir esos errores.

Use como referencia la siguiente tabla

Nombre del servicio	Parámetros	Tipo de Dato	¿Requerido?	Descripción Dato	Valor de retorno	Descripción servicio
SetErrors	xmlErrors	string	SI	Errores que se generaron.	string	Recibe string que contiene un XML con los errores que se hayan generado en los métodos AddWorkOrdersXML, CancelWorkOrdersXML y UpdateWorkOrdersXML. Devuelve cadena vacía si no hubo error o la excepción del error.

Se reciben los datos de los errores (Parámetros)

```
<reporte_fallas>
   <error>
     <clave_usuario>cnunes</clave_usuario>
     <fecha_evento>2013-04-17 12:11:37.117</fecha_evento>
     <id_orden>209023</id_orden>
     <descipcion_error>2: Work Order 209023 is duplicated</descipcion_error>
   </error>
   <error>
     <clave_usuario>Msilva</clave_usuario>
     <fecha_evento>2013-04-17 10:30:24.110</fecha_evento>
     <id_orden>208855</id_orden>
     <descipcion_error>2: Work Order 208855 is duplicated</descipcion_error>
   </error>
</reporte_fallas>
```

En el archivo **IApi.cs** agregue el siguiente código

```
        //Expone SendErrors
        [OperationContract]
        [WebInvoke(Method = "POST",
            RequestFormat = WebMessageFormat.Xml,
            ResponseFormat = WebMessageFormat.Xml,
            UriTemplate = "SendErrors"
            )]
        void SendErrors(Stream errores);
```

En el archivo **Api.svc.cs** agregue el siguiente código.

```
//Implementación SendErrors
        public void SendErrors(Stream errores)
        {
            //Recibe string que contiene un XML con los errores que se hayan generado en los métodos
            //AddWorkOrdersXML, CancelWorkOrdersXML y UpdateWorkOrdersXML.
            try
            {
                StreamReader reader = new StreamReader(errores);
                string text = reader.ReadToEnd();
                //Documento XML que púede usarse para guardar los errores.
                XmlDocument xmlerrors = new XmlDocument();
                xmlerrors.LoadXml(text);
                //Una vez que tiene el reporte de errores en un documento XML
                //puede guardar implenter un proceso para guardarlos en un
                //log o una base de datos
                string fileLogName = string.Format("C:\\MyErrors\\errores-de-formiik-{0:yyyy-MM-dd}.csv",
DateTime.Now);
                StreamWriter log;
                if (!File.Exists(fileLogName))
                {
                    log = new StreamWriter(fileLogName);
                }
                else
                {
                    log = File.AppendText(fileLogName);
```

```
                }
                log.WriteLine(text);
                log.Close();
            }
            catch (Exception ex)
            {
                string fileLogName = string.Format("C:\\MyErrors\\errores-de-mi servicio{0:yyyy-MM-
dd}.csv", DateTime.Now);
                StreamWriter log;
                if (!File.Exists(fileLogName))
                {
                    log = new StreamWriter(fileLogName);
                }
                else
                {
                    log = File.AppendText(fileLogName);
                }
                log.WriteLine(ex.Message);
                log.Close();
            }
        }
```

Actualización Flexible

Formiik a través de un formulario puede hacer consultas y actualizaciones del formulario en línea, así que debemos desarrollar un servicio para responder con la información necesaria para los operadores.

Nombre del servicio	Parámetros	Tipo de Dato	¿Requerido?	Descripción Dato	Valor de retorno	Descripción servicio
FlexibleUpdateWorkOrder	jsonUpdateOrder	string	SI	Datos de búsqueda.	string	Se reciben los datos de búsqueda, estos datos deben de coincidir con los datos marcados en el formulario como de búsqueda, ej. TxtNombre, TxtApaterno, TxtMaterno, TxtRFC. Además de estos datos se recibirá el tipo de formulario para que el servicio sepa que regresar. Este servicio de regresará un string que contiene un json con datos con los que se pre-llenara la solicitud, los tags del json deben de coincidir con las que se definieron en el formulario, entre estos datos regresará el número de solicitud asignado.

```
//Cadena Json que llega desde el dispositivo al servicio
{
    "IdWorkOrderFormType" : "128da336-84b5-49be-b743-7e9cfaf45be0",
    "IdWorkOrder" : "b4af203e-cb1e-4dd6-ac82-24c5e2cb379d",
    "ExternalId" : "JG-319",
    "Action" : "",
    "InputFields" : {
        "Ape_Materno" : "Campos",
        "Ape_Paterno" : "Mendez",
        "Nombres" : "Jhovany",
        "RFC" : "MECJ830414",
        "ExternalType" : "BP"
    },
    "Username" : "jgm",
    "WorkOrderType" : "BP"
}
```

IdWorkOrderFormType	Id interno del formulario de formiik
IdWorkOrder	Id interno de la orden de formiik
ExternalId	Id externo de la orden en formiik
Username	Usuario/operador del dispositivo en formiik
WorkOrderType	Nombre externo del formulario en formiik
InputFields	Campos con la información solicitada para la consulta de buró de crédito

Regresará un string que contiene un Json con datos con los que se pre-llenara la solicitud (Valor de retorno)

```
//Cadena que tiene que regresar el servicio
{
  "UpdateFieldsValues": {
    "Mensaje": "Aceptado",
    "NumeroAutorizacion": "12345"
  },
```

UpdateFieldsValues	Campos en el formulario que serán actualizados
AfectedFields	Campos que se afectarán sus propiedades

```
  "AfectedFields": [
    {
      "Name": "ResultadoBuro",
      "Settings": {
        "ReadOnly": "True",
        "Requested": "False",
        "Visible": "True"
      }
    },
    {
      "Name": "Profesion",
      "Settings": {
        "ReadOnly": "True",
        "Requested": "False",
        "Visible": "True"
      }
    }
  ],
  "FormiikReservedWords": [
    {
      "ReservedWord": "AlertMessage",
      "Value": "Consulta exitosa"
    },
    {
      "ReservedWord": "ClientError",
      "Value": "False"
    }
  ]
}
```

FormiikReservedWords

Palabras reservadas de Formiik para modificar el comportamiento del formulario

ForceSave	Forza a guardar una orden después de ejecutar un update, sin importar que falten campos requeridos.	Si está en "True" se guarda automáticamente la orden y se pasa al tab de terminadas Si está en "False" o no tiene la palabra reservada sigue el comportamiento normal.
ExternalId	Nuevo Id de la orden (external id) **ÚNICAMENTE FUNCIONA CAMBIARLA MEDIANTE SERVICIO, Y PARA EL CAMBIO DE NOMBRE DE UNA ORDEN DESDE UN APK SÓLO SE SOPORTA CUANDO LAS ORDENES SON ORIGINADAS.**	
ClientError	Valor (true o false). Disponible Android 4.5 Si el valor es True se considera que en el cliente hubo un error, esto servirá para saber si incrementar el número de intentos del widget o no	
ChangeExpirationDate	Valor para cambiar la fecha de expiración de la órden, DEBE venir en el formato: c# = yyyy/MM/dd HH:mm:ss.fff java= yyyy/MM/dd HH:mm:ss.SSS	

			y hora UTC, si no es así se ignorará el valor que llegué. Disponible Android 4.5 **ÚNICAMENTE FUNCIONA CAMBIARLA MEDIANTE SERVICIO, NO FUNCIONA MEDIANTE APK.**	
		ChangeCancelationDate	Valor para cambiar la fecha de cancelación de la órden, DEBE venir en el formato: c# = yyyy/MM/dd HH:mm:ss.fff java= yyyy/MM/dd HH:mm:ss.SSS y hora UTC, si no es así se ignorará el valor que llegué. Se comprará la hora actual y la hora de cancelación que llegué, en caso de que la fecha de cancelación sea menor o igual Entonces se mandará un mensaje al usuario "Órden cancelada", se cerrará la órden y se borrará. Disponible Android 4.5 **ÚNICAMENTE FUNCIONA CAMBIARLA MEDIANTE SERVICIO, NO FUNCIONA MEDIANTE APK.**	
		AllowDeletePartial	(True/False). Altera si se	

			permite o no borrar los datos guardados parcialmente en la orden que se está actualizando.. "False" para no permitir el borrado del guardado parcial. Reenvío y resignación de AllowDeletePartial Cuando la resignación o re-envío de una orden se realiza de forma tradicional o sin palabras reservadas se consideran las propiedades del formato para poder o no realizarse. Sin embargo cuando en el formato existe un UpdateEdit que invoca la palabra reservada AllowDeletePartial en "true" Indica que después de hacer el update ya no podrá borrarse los datos parciales y tampoco la orden si es que fue generada en dispositivo. Si regresa "False" no permitirá el re-envio o re-asignación de la misma orden.	
		AlertMessage	Mensaje que se muestra en Android una vez que el formulario se ha actualizado.	

En el archivo **IApi.cs** agregue el siguiente código

```
        //Expone FlexibleUpdateWorkOrder
        [OperationContract]
        [WebInvoke(Method = "POST",
            RequestFormat = WebMessageFormat.Json,
            ResponseFormat = WebMessageFormat.Json,
            UriTemplate = "FlexibleUpdateWorkOrder"
            )]
        FlexibleUpdateResponse FlexibleUpdateWorkOrder(Stream updateorder);
```

En el archivo **Api.svc.cs** agregue el siguiente código.

```
        //Implementación FlexibleUpdateWorkOrder
        public FlexibleUpdateResponse FlexibleUpdateWorkOrder(Stream updateorder)
        {
            FlexibleUpdateResponse respuestaJson = new FlexibleUpdateResponse();

            try
            {

                StreamReader reader = new StreamReader(updateorder);
                string text = reader.ReadToEnd();

                string fileLogName = string.Format("C:\\MyFlexibleOrder\\ordenes-flexibles{0:yyyy-MM-dd}.csv", DateTime.Now);
                StreamWriter log;
                if (!File.Exists(fileLogName))
                {
                    log = new StreamWriter(fileLogName);
                }
                else
                {
                    log = File.AppendText(fileLogName);
                }
                log.WriteLine(text);
                log.Close();



                JavaScriptSerializer serializer = new JavaScriptSerializer();
                FlexibleUpdateRequest peticionJson = serializer.Deserialize<FlexibleUpdateRequest>(text);

                if (peticionJson.Action == "ConsltaBUro")
                {
                    peticionJson.InputFields["RFC"].ToString();
                    peticionJson.InputFields["Apellido"].ToString();
                }

                //Despues de Deserealizar el Json Input
                //La consulta a tus propios sistemas
```

```
                //Comenzar a construir la respuesta
                //Comenzar a cambiar el comportamiento de los campos

                //Comenzar a cambiar el valor de los campos
                respuestaJson.UpdateFieldsValues.Add("Score", "Aceptado");
                respuestaJson.UpdateFieldsValues.Add("Puesto", "valor");

                respuestaJson.UpdateFieldsValues.Add("tablaconultaburo", "Se autorizó su crédito");

                respuestaJson.AfectedFields.Add(new FlexibleField("Score", true, false, false));


                //Si es un campo tabla ys se puede enviar el código html de forma normal, ver ejemplo
                //respuestaJson.UpdateFieldsValues.Add("Tabla", "<html><body><a href='http//www.ff.com/documento.pdf'/>><br /><table><td>Plazo</td><td>Credito a
otorgar</td><td>Gastos por apertura</td><td>Contarias con</td><td>Monto mensual de
pago</td><tr><td>12</td><td>$6,819.32</td><td>$0.00</td><td>$6,819.32</td><td>$626.83</td></tr><tr><td>18</td><td>$9,376.57</td><td>$0.00</td><td>$9,376.57</td><td
>$631.23</td></tr><tr><td>24</td><td>$11,720.72</td><td>$0.00</td><td>$11,720.72</td><td>$632.21</td></tr><tr><td>30</td><td>$14,277.96</td><td>$0.00</td><td>$14,2
77.96</td><td>$631.79</td></tr></table></body><style>#table, th, td, tr {border: 1.5px solid black; border-collapse: collapse;width:auto;font-size:12px;font-
family:Calibri;padding:5px;}</style></html>");

                FlexibleUpdateReservedWords error = new FlexibleUpdateReservedWords();
                error.ReservedWord = "ClientError";
                error.Value = "True";
                respuestaJson.FormiikReservedWords.Add(error);

                FlexibleUpdateReservedWords mensaje = new FlexibleUpdateReservedWords();
                mensaje.ReservedWord = "AlertMessage";
                mensaje.Value = "Consulta exitosa";
                respuestaJson.FormiikReservedWords.Add(mensaje);
            }
            catch (Exception e)
            {
                FlexibleUpdateReservedWords mensaje = new FlexibleUpdateReservedWords();
                mensaje.ReservedWord = "AlertMessage";
                mensaje.Value = e.Message;
                respuestaJson.FormiikReservedWords.Add(mensaje);//Mensaje de Error

                FlexibleUpdateReservedWords error = new FlexibleUpdateReservedWords();
                error.ReservedWord = "ClientError";
                error.Value = "False";
                respuestaJson.FormiikReservedWords.Add(error);

            }
            return respuestaJson;
        }
```

Este método usa unas clases adicionales que pueden descargarse de la siguiente liga:

https://www.dropbox.com/s/0tn0nppxr9pr881/MyCompanyApi.zip?dl=0

Una vez que terminamos de agregar los métodos necesarios para nuestra API, compilamos nuestro proyecto y basta con copiar la(s) dlls generadas en nuestro servidor.