

## Índice

Arquitectura SIZ.....	3
Vista General.....	3
Vista de componentes.....	3
PeticiónZMX, PeticiónAPI.....	5
Uso en el Cliente.....	6
Uso en el Server.....	6
Problemas comunes.....	7
ResultadoZMX.....	8
Encriptación.....	9
NOTAS IMPORTANTES.....	9
Backend .NET.....	10
Modelo utilizando Entity Framework 6 MODELO_DATOS_ORACLE (solo para soporte).....	10
Plantilla T4.....	10
NOTA IMPORTANTE.....	12
ClaseBaseOracle.....	12
ManejadorGuardadoOracle.....	13
Cadena conexión.....	13
Generación del modelo con la herramienta de EF6, paso a paso.....	16
Modelo utilizando ADO (versión con aplicación generadora de clases).....	21
Proyecto ACCESO_DATOS.....	21
Proyecto MODELO_DATOS_ADO.....	22
Anotaciones de clase y propiedad.....	22
Anotaciones para obtener navegaciones mediante propiedades (Método Carga).....	23
Servidor Backend IIS.....	24
Cuenta Servidor.....	24
Distribuir el instalador a los clientes.....	25
Configuración de cadena de conexión en WebAPI para ambos modelos.....	27
MODELO_DATOS_ORACLE.....	27
MODELO_DATOS_ADO.....	27
Generar cadena conexión MODELO_DATOS_ADO.....	27
Publicación de WebApi, Solución ApiContainer.....	27
Como debuguear un WebApi de ApiContainer en el servidor IIS.....	29
Configuración de Remote Debugger en el servidor IIS.....	31
Frontend WPF.....	32
Instalación del cliente WPF.....	32
MG_UserControls.....	32
Propiedad SessionERP.....	32
Propiedad PantallaSinPromptCuentasContables.....	32
Propiedad ControlesAgregadosASecuencia.....	33
Boolean HayRequeridosSinCapturar().....	33

Secuencia de controles.....	33
Problemas Comunes.....	34
La secuencia de controles no esta incluyendo un control.....	34
El Borde o el asterisco del Requerido se pinta de forma incorrecta.....	35
UserControls.....	35
Estructura.....	35
Toolbar SIZ.....	35
Botón de toolbar.....	36
Iconos (Iconos usados por el sistema, <FontAwesome>).....	36
Tooltip de toolbar.....	39
Menú principal.....	39
Ubicación de controles de interfaz y estructura.....	40
Login.....	40
Interfaz Principal.....	40
Controles que componen la interfaz principal.....	40
Controles que componen el popup del menú.....	40
Pendientes para su despliegue en producción.....	40
Esquema de actualizaciones de SIZ.....	40
Etapa 1: Actualización con ClickOnce.....	40
Etapa 2: Actualización de DLL Base con el splashscreen.....	41
Etapa 3: Actualización de DLL sobre demanda, formas de funcionalidades del negocio.....	41
Actualizador de DLL (administrador de actualizaciones).....	42
Base de datos.....	43
Tablas.....	44
Actualizar DLL Base.....	44
NOTA IMPORTANTE.....	45
Actualizar DLL de funcionalidades con dependencias.....	45
NOTA IMPORTANTE.....	46
APIS SIZ (ApiContainer).....	46
ApiCatZMX.....	47
ConfigController.....	47
ModeloController.....	48
Generador de código.....	48
Proceso para generar una clase.....	48
Proceso para integrar la clase al MODELO_DATOS_ADO.....	50
Ambientes SIZ.....	50
Archivo config en WebApi para enrutar a los distintos ambientes según la sesión.....	50
Archivo config de aplicación escritorio para apuntar al WebApi.....	53
ZucarmexIniUpd (Configurar ClickOnce).....	53
ZucarmexIniApp (Configuración del proyecto SIZ).....	56

## Arquitectura SIZ

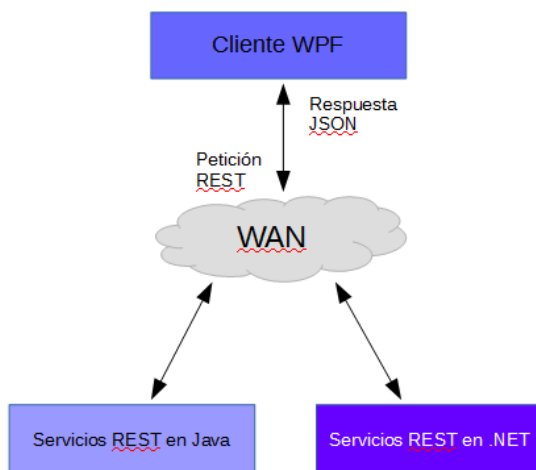
La arquitectura utilizada en el SIZ esta diseñada para soportar la interacción de múltiples tecnologías y proporcionar la seguridad necesaria para exponer EndPoints que provean información del negocio.

Estas tecnologías están basadas en una arquitectura de tipo Cliente-Servidor, donde el servidor proporciona Servicios REST, que atienden las peticiones de los clientes de acuerdo a dos criterios fundamentales, **si es una transacción o un catálogo**.

Con el fin de asegurar una mejor ejecución de los procesos de negocio, las transacciones son ejecutadas sobre servicios construidos en Java a diferencia de los catálogos que son sostenidos por .NET.

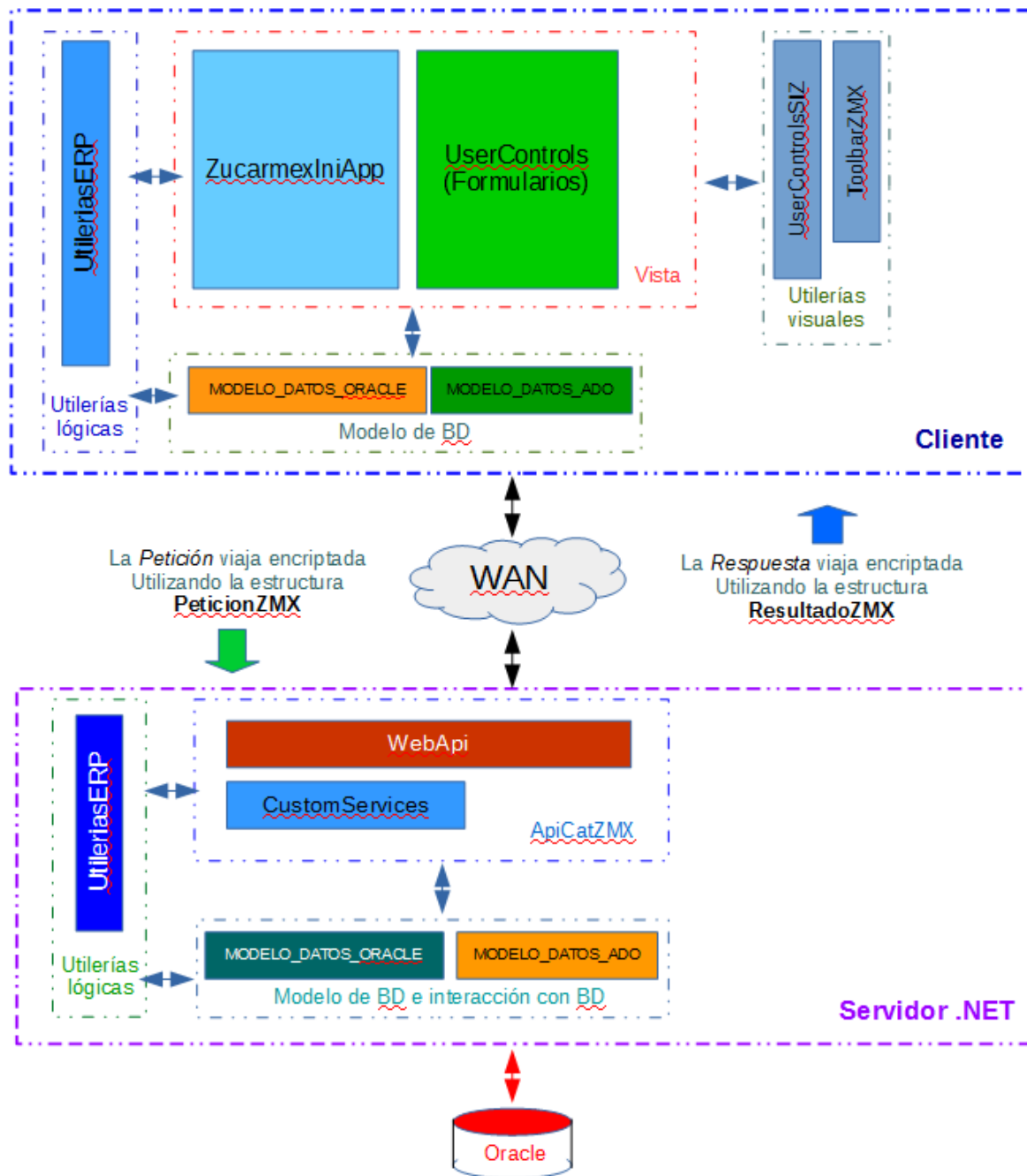
### Vista General

La siguiente vista de la arquitectura, presenta tres componentes principales operando en el SIZ, se encuentra el cliente WPF de la plataforma .NET, el cual es la vista e interacción con el usuario, por otro lado, en la parte de Endpoint tenemos Servicios REST construidos en Java y en .NET, los cuales distribuyen la carga que llega de los diferentes clientes dependiendo la información del negocio tratada.



### Vista de componentes

En la siguiente imagen se puede visualizar la interacción entre los componentes principales de la arquitectura SIZ, por lo que se puede entender a nivel general, la utilidad y dependencia de las funcionalidades específicas del sistema.



En la imagen anterior, la comunicación entre el cliente y el server, se realiza mediante mensajes estructurados en PeticionZMX y ResultadoZMX, la razón para ello, es que al enviar

o recibir información, junto con ella viajan datos de auditoría para identificar y controlar quien realiza dichas peticiones, a su vez, dichas estructuras se encapsulan con un proceso de serialización y encriptado que se describirá en un tema posterior ([#Encriptación](#)).

## PeticionZMX, PeticionAPI

Proyecto=MG\_UTILERIASGRALES, Clase=UTILERIASERP.Utilerias.API.PeticionZMX

La clase utilizada para realizar peticiones al servidor es PeticionZMX y recientemente PeticionAPI (Importante actualizar los esquemas anteriores a este, debido a su mejor rendimiento), a continuación se explica la estructura:

```
namespace UTILERIASERP.Utilerias.API
{
    51 references | Aaron Baez, 229 days ago | 1 author, 1 change
    public class PeticionAPI
    {
        2 references | Aaron Baez, 229 days ago | 1 author, 1 change
        public ERPSession Sesion { get; set; }
        2 references | Aaron Baez, 229 days ago | 1 author, 1 change
        public ParametrosControlZMX ParametrosControl { get; set; }
        1 reference | Aaron Baez, 229 days ago | 1 author, 1 change
        public object Datos { get; set; }
        public Dictionary<string, object> Parametros = new Dictionary<string, object>();

        22 references | Aaron Baez, 229 days ago | 1 author, 1 change
        public static PeticionAPI Nueva()
        {
            return new PeticionAPI();
        }

        64 references | Aaron Baez, 229 days ago | 1 author, 1 change
        public void Parametro(string nombre, object valor)
        {
            Parametros.Add(nombre, valor);
        }

        18 references | Aaron Baez, 229 days ago | 1 author, 1 change
        public ResultadoZMX Ejecutar(string ruta)
        {
            return ClienteAPI.Ejecutar(ruta, this);
        }
        5 references | Aaron Baez, 229 days ago | 1 author, 1 change
        public async Task<ResultadoZMX> EjecutarAsync(string ruta)
        {
            return await ClienteAPI.EjecutarAsync(ruta, this);
        }
    }
}
```

En el recuadro rojo, tenemos la posibilidad de agregar parámetros que son leídos posteriormente en el WebApi del servidor, y posterior a ello, se invoca el método ejecutar ya

sea de forma síncrona o asíncrona.

Esto lo podemos ver en un ejemplo de una consulta desde un cliente y como lo recibe el WebApi en el server.

## Uso en el Cliente

```
PeticionAPI peticionapi = PeticionAPI.Nueva();
peticionapi.Parametro("LOCALIDAD_ID_ORIGEN", localidadIdOrigen);
peticionapi.Parametro("LOCALIDAD_ID_DESTINO", localidadIdDestino);
peticionapi.Parametro("VIGENCIA", vigencia);

ERPToolBar.ERPisBusy = true;
ResultadoZMX resultado = await peticionapi.EjecutarAsync("locttarifasfletesautorizadas/eliminar");

if (resultado.Exitoso)
{
    MessageBoxERP.Show(Properties.Resources.textEliminar, "", MessageBoxButtons.OK, MessageBoxIcon.Information);
    if (tabControl.SelectedIndex == 1)
        Nuevo_click();
    else
        LlenaBuzon();
}
else
{
    ERPToolBar.EnabledEliminar = false;
    MessageBoxERP.Show(resultado.MensajeFormado, "", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

## Uso en el Server

```
[ConfigureDBConnectionFactory]
[Route("locttarifasfletesautorizadas/eliminar")]
0 references | Abraham Mendez, 25 days ago | 1 author, 4 changes
public string eliminar([FromBody]string value)
{
    ResultadoZMX resultado = null;
    ManejadorInstanciasBD mibd = new ManejadorInstanciasBD();
    try
    {
        PeticionAPI petition = Seguridad.ValidarPeticion<PeticionAPI>(value);
        MG_LOCALIDADES localidad = new MG_LOCALIDADES();

        if (!petition.Parametros.ContainsKey("LOCALIDAD_ID_ORIGEN"))
        {
            throw new ArgumentNullException("LOCALIDAD_ID_ORIGEN");
        }
        if (!petition.Parametros.ContainsKey("LOCALIDAD_ID_DESTINO"))
        {
            throw new ArgumentNullException("LOCALIDAD_ID_DESTINO");
        }
        if (!petition.Parametros.ContainsKey("VIGENCIA"))
        {
            throw new ArgumentNullException("VIGENCIA");
        }
    }
}
```

Como se puede apreciar en la imagen anterior, la variable `Parámetros` al ser un diccionario, se pregunta si el nombre de la variable esta contenida en él, para conocer si ésta viene en la petición y con ello hacer uso para la lógica de negocio que venga después de ello. El ejemplo mostrado anteriormente, se encuentra en la solución `ApiContainer`, es ahí donde están ubicados los proyectos backend de los catálogos que están con el esquema del ejemplo.

### Problemas comunes

#### El API no se encuentra

Cuando sucede este error, se puede deber a dos escenarios, uno de ellos es cuando la ruta configurada en la petición está mal escrita con respecto la asignada en el WebApi ó cuando esta ruta se repite más de una vez en el la publicación del WebApi del IIS. Es importante prestar atención cuando se crea un API nuevo, ya que en ocasiones es común caer en el error de “Copiar-Pegar” y repetir así dichas rutas en diferentes controladores. En las imágenes siguientes se hace la comparación entre el API configurada en la petición y la Ruta asignada en el WebApi.

En el cliente marcado en rojo:

```
PeticionAPI peticionapi = PeticionAPI.Nueva();
peticionapi.Parametro("LOCALIDAD_ID_ORIGEN", localidadIdOrigen);
peticionapi.Parametro("LOCALIDAD_ID_DESTINO", localidadIdDestino);
peticionapi.Parametro("VIGENCIA", vigencia);

ERPToolbar.ERPIsBusy = true;
ResultadoZMX resultado = await peticionapi.EjecutarAsync("locttarifasfletesautorizadas/eliminar");
```

En el server marcado en rojo:

```
[ConfigureDBConnectionFilter]
[Route("locttarifasfletesautorizadas/eliminar")]
0 references | Abraham Méndez, 25 days ago | 1 author, 4 changes
public string eliminar([FromBody]string value)
{
    ResultadoZMX resultado = null;
    ManejadorInstanciasBD mibd = new ManejadorInstanciasBD();
    try
```

## Error interno en el API

Este error suele suceder cuando dentro del API se hace referencia a una clase que no esta actualizada en la publicación IIS, el indicio para detectar este error, es que marca la excepción de **Error interno** y **es imposible Debuguear el API**, entonces es seguro que la solución es verificar que las clases (ensamblados a los que el controller hace referencia, generalmente el modelo) en el servidor estén en su última versión.

## ResultadoZMX

Esta estructura contiene la respuesta que genera el servidor hacia los clientes, cuando el caso es exitoso, se crea la instancia y se retorna de la siguiente forma (siguiente imagen):

```
resultado = new ResultadoZMX();
resultado.Exitoso = true;
resultado.Datos = localidad;
```

En caso de excepción, se anexa los detalles del error y se indica que la petición no tuvo éxito (Siguiendo imagen).



```
catch (Exception ex)
{
    mibd.ZMX_DeshacerTransaccion();
    mibd.ZMX_CerrarConexion();
    resultado = new ResultadoZMX();
    resultado.Exitoso = false;
    resultado.Mensajes = new Dictionary<int, string> { { 0, ex.Message } };
    resultado.DatosDebug = GetFullExceptionDetail(ex);
}
```

## Encriptación

Proyecto=MG\_UTILERIASGRALES, Clase=UTILERIASERP.Utilerias.API.Seguridad

Por medidas de seguridad, las peticiones y respuestas, se encriptan, por lo que cuando se crea la petición, internamente la instancia pasa a través de un proceso de serialización y encriptado antes de ser enviado al servidor.

El proceso es el siguiente:

- Se crea un diccionario con dos llaves, “**mensaje**” y “**checksum**”
- Serializar el objeto con JsonConvert de la librería Newtonsoft y lo almacenamos en una variable
- Con la instancia serializada anterior, se genera su checksum con MD5, el cual posteriormente servirá para comprobar la integridad del mensaje en el servidor.
- En este punto, ya tenemos el checksum y la instancia serializada, por lo que en el diccionario en su llave “mensaje”, asignamos la instancia serializada y en su llave “checksum” asignamos el checksum que generamos.
- El siguiente paso de la encriptación, es tomar el diccionario y serializarlo con la utilidad de Newtonsoft
- Por último, el mensaje es encriptado y enviado al servidor.

## NOTAS IMPORTANTES

NOTA: la encriptación (último paso) aún está pendiente, por lo que cuando se integre dicho paso, se necesitara agregar en el servidor su desencriptado.

NOTA 2: Para desencriptar en el servidor, se realizan los mismos pasos antes descritos en orden inverso.

## Backend .NET

### Modelo utilizando Entity Framework 6 MODELO\_DATOS\_ORACLE (solo para soporte)

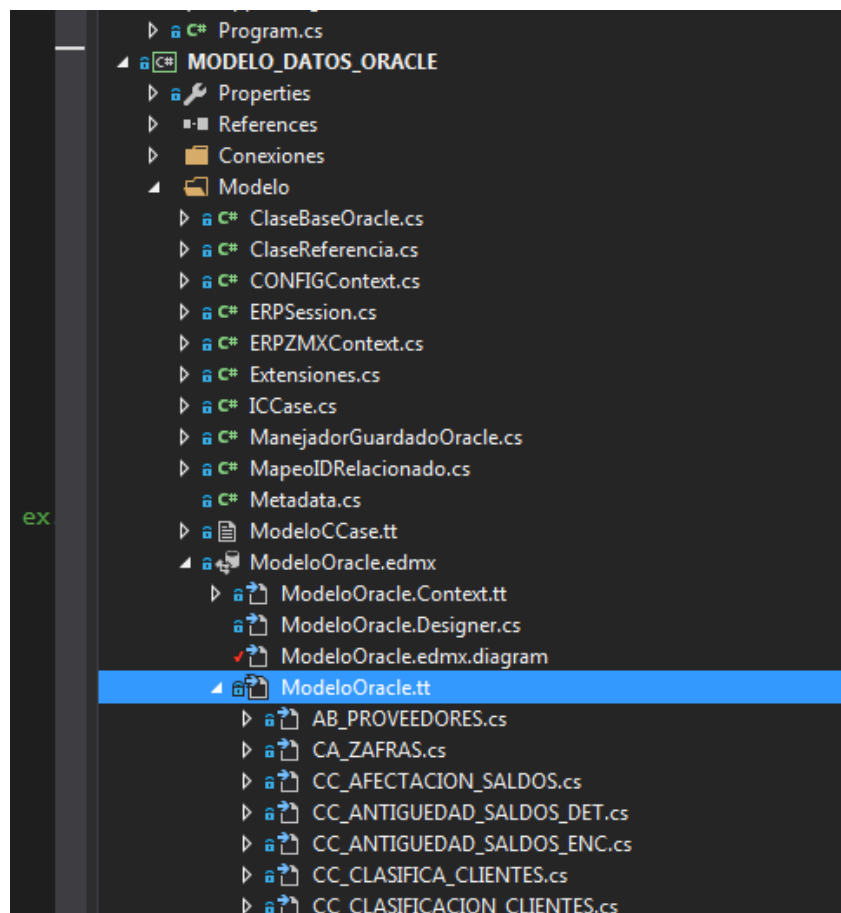
En las primeras etapas del proyecto, se utilizó como base Entity Framework 6 (EF6) como capa ORM para el uso de entidades y abstracción de la base de datos, en la siguiente versión, hubo un cambio debido a que el generar el modelo de clases para una cantidad elevada de tablas, **se hizo demasiado lento y complejo**, por lo que se generó una nueva capa de mapeo que viniera a reemplazar a EF6, esta vez utilizando ADO como motor de consultas y Reflection como herramienta tecnológica para realizar los mapeos entre los objetos y los datos puros obtenidos de la base de datos.

### Plantilla T4

La importancia de conocer las plantillas T4, radica en la posibilidad de en algún momento ofrecer soporte a la lógica existente para generar las clases del modelo para EF6.

El modelo obtenido a partir de EF6, además de generar las clases, también las adorna, debido a que utiliza ciertas validaciones automáticas (usando reflection) que obtiene de la información incrustada en las anotaciones o adornos adicionales que contiene la clase.

Algunos ejemplos de dichos adornos, los podemos encontrar en las clases ubicadas en el proyecto **MODELO\_DATOS\_ORACLE** tal como se muestra en la siguiente imagen:

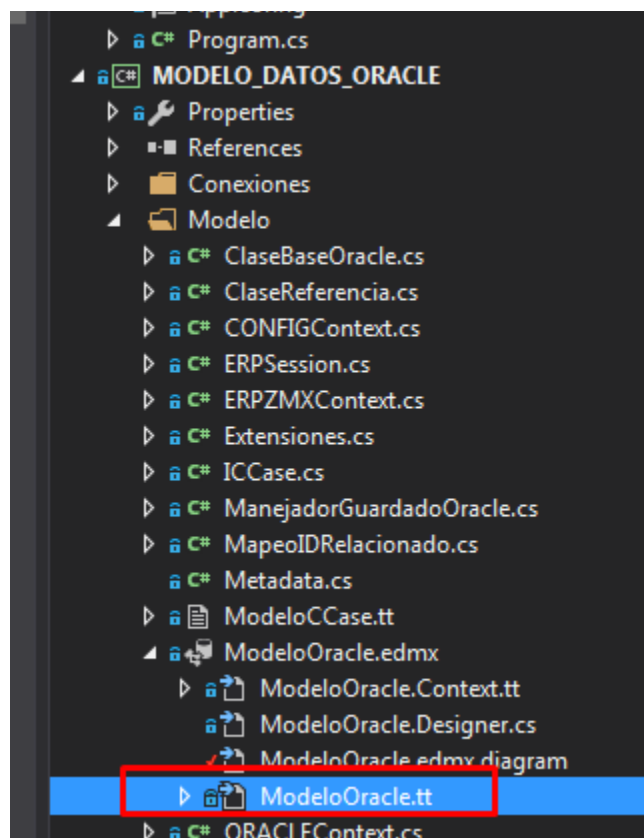


de Junio del 2017

Ejemplo de anotación:

```
///</summary>  
[MapeoIDRelacionado(AtributoID = "_EMPRESA_ID_FILIAL")]  
[Column("MG_EMPRESAS")]  
0 references | Jose Carlos Guerrero Karass, 252 days ago | 1 author, 1 change  
public virtual MG_EMPRESAS MG_EMPRESAS { get { return this._MG_EMPRESAS;} s  
private MG_EMPRESAS _MG_EMPRESAS;  
///</summary>
```

Para que el generador escriba anotaciones en las clases, es necesario realizar modificaciones a la **plantilla T4**, estas modificaciones se hacen en el archivo:



Podemos rápidamente encontrar un ejemplo de la modificación hacia la plantilla en la siguiente imagen:

```
#>
public Boolean ES_MODIFICADO { get; set; }
<#

    foreach (var edmProperty in simpleProperties)
    {

        // begin max length attribute
        string nombre = edmProperty.Name.Replace("_", " ");
        if (edmProperty.TypeUsage.Facets["Nullable"].Value.ToString() == "False" && edmProperty.Name != "ULTIMA")
        {
            var summary = string.Join("\n", "///<summary>, \t\t// Valor Requerido, \t\t///</summary>".Split(','));
            summary#>
```

### NOTA IMPORTANTE

Es necesario conocer como se hicieron las modificaciones a las clases del modelo, por si se necesita realizar un mantenimiento a dicho proceso.

### ClaseBaseOracle

Las clases del modelo, se generan extendiendo de ClaseBaseOracle, esto con el fin de acceder a funcionalidades que apoyan en el manejo de las instancias consultadas de la base de datos.

```
using System.Data.Entity.Core.Metadata.Edm;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations.Schema;

[Table("AB_PROVEEDORES")]
30 references | jguerrerok, 96 days ago | 2 authors, 4 changes
public partial class AB_PROVEEDORES : ClaseBaseOracle, INotifyPropertyChanged,
{
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214", Justification = "0 references | jguerrerok, 96 days ago | 2 authors, 3 changes")]
    public AB_PROVEEDORES()
    {
        this.FA_CONFIGURACION_INGENIOS = new HashSet<FA_CONFIGURACION_INGENIOS>();
        this.IE_TRANSPORTES = new HashSet<IE_TRANSPORTES>();
        this.IN_MOVIMIENTOS_ENC = new HashSet<IN_MOVIMIENTOS_ENC>();
        this.IN_ALMACENES = new HashSet<IN_ALMACENES>();
    }
}
```

La clase está ubicada en MODELO\_DATOS\_ORACLE.Modelo.ClaseBaseOracle, del

proyecto MODELO\_DATOS\_ORACLE.

### ManejadorGuardadoOracle

Proyecto=MODELO\_DATOS\_ORACLE,  
Clase=MODELO\_DATOS\_ORACLE.Modelo.ManejadorGuardadoOracle

En esta clase está encapsulada la funcionalidad necesaria para **acceder a la base de datos mediante EF6**, dicha funcionalidad incluye desde el guardado, validaciones previas al guardado y consultas de listas u objetos únicos, utilizando para ello las expresiones lambda.

Un ejemplo de consulta seria:

(Ejemplo tomado de: pcctSeriesController dentro del proyecto ApiContainer)

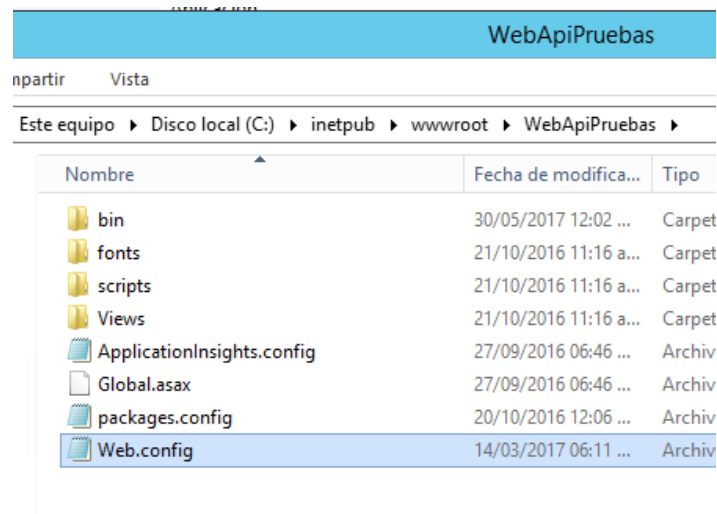
```
seriecodigo = Convert.ToInt16(peticion.Parametros["seriecodigo"].ToString());  
documentoid = Convert.ToInt16(peticion.Parametros["documentoid"].ToString());  
empresaid = Convert.ToInt16(peticion.Parametros["empresaid"].ToString());  
mg.ZMX_ConsultaUnico<MG_SERIES>(x => x.SERIE_CODIGO == seriecodigo && x.DOCUMENTO_ID == documentoid && x.EMPRESA_ID == empresaid, x => x.MG_SERIES_RELACION);  
  
throw Exception("Parámetros insuficientes");
```

La primer parte (recuadro rojo) es la sección del Where, donde mediante Expresiones lambda se realiza el filtrado de los objetos en el EF6, en el segundo parámetro (recuadro azul), colocamos las navegaciones necesarias utilizadas para realizar la consulta, en este caso en particular, además de los objetos consultados de la tabla MG\_SERIES, también se incluye en sus propiedades cargadas, la propiedad MG\_SERIES\_RELACION.

### Cadena conexión

Por la naturaleza del Backend Web, la cadena de conexión se encuentra en el archivo Web.Config de la publicación del sitio en el IIS, hasta este momento, se encuentran los valores de conexión expuestos, debido a que el cambio del modelo a ADO.

En el sitio de IIS, encontramos en cada ambiente el archivo Web.Config donde se puede ver explícitamente la cadena de conexión, por ejemplo:



Y dentro del archivo vemos lo siguiente:

### **-Sección de enrutamiento**

Aquí tenemos en esta área los distintos ambientes, la funcionalidad es la de enrutar al ambiente adecuado cuando se realiza la petición por los clientes, por ejemplo, si se realiza una petición al ambiente de Pruebas, con el ambiente configurado en Desarrollo, el API redireccionará dicha petición al ambiente correspondiente. En la siguiente imagen se muestra esta sección:

```
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />

  <add key="restapimodelo0" value="http://192.168.0.109:90/api/Modelo/" />
  <add key="restapimodelo1" value="http://192.168.0.109:88/api/Modelo/" />
  <add key="restapimodelo2" value="http://192.168.0.109:86/api/Modelo/" />
  <add key="restapimodelo3" value="http://IM-ABAEZ:86/api/Modelo/" />
  <add key="restapimodelo4" value="http://192.168.0.109/api/Modelo/" />
  <add key="restapimodelo9" value="http://IM-ABAEZ:85/api/Modelo/" />
  <add key="restapimodelo10" value="http://localhost:56426/api/Modelo/" />

  <add key="restapimenu0" value="http://192.168.0.109:90/api/Menu/" />
  <add key="restapimenu1" value="http://192.168.0.109:88/api/Menu/" />
  <add key="restapimenu2" value="http://192.168.0.109:86/api/Menu/" />
  <add key="restapimenu3" value="http://IM-ABAEZ:86/api/Menu/" />
  <add key="restapimenu4" value="http://192.168.0.109/api/Menu/" />
  <add key="restapimenu9" value="http://IM-ABAEZ:85/api/Menu/" />
  <add key="restapimenu10" value="http://localhost:56426/api/Menu/" />

  <add key="restapiapr0" value="http://192.168.0.109:90/api/apr/" />
  <add key="restapiapr1" value="http://192.168.0.109:88/api/apr/" />
  <add key="restapiapr2" value="http://192.168.0.109:86/api/apr/" />
  <add key="restapiapr3" value="http://IM-ABAEZ:86/api/apr/" />
  <add key="restapiapr4" value="http://192.168.0.109/api/apr/" />
  <add key="restapiapr9" value="http://IM-ABAEZ:85/api/apr/" />
```

### -Cadena de conexión encriptada para Modelo ADO

La cadena de conexión de ADO se encuentra encriptada, ya que utiliza un driver de conexión que ofrece .NET, en el Web.Config se puede ver con los siguientes parámetros:

```
<add key="csmodel_ado" value="VEQzxtILGVPKv+HpU+qayGH/AoUeL8XJ3G1RIZdfrEfufSsbcXYKtrIBKtOYkmieI" />
<add key="csmodel_ambiente_ado" value="pruebas" />
<add key="modo" value="pruebas" />
```

Donde el área marcada en rojo tiene como valor la cadena de conexión encriptada y el cuadro marcado en azul tiene configurado el ambiente al que se dirige.

### -Cadena de conexión explícita para Modelo con EF6 (Pendiente encriptar)

Hasta el momento, la cadena de conexión la encontramos expuesta, por lo que se deberá de realizar un mecanismo de encriptado que pueda ocultar dichos datos.

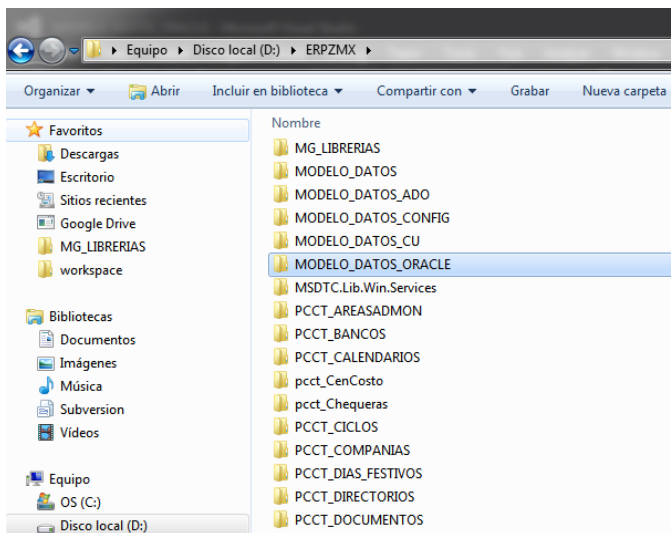
Aquí una imagen del dato en el archivo Web.Config:

```
<connectionStrings>
  <!--DESARROLLO-->
  <add name="cs_config" connectionString="DATA SOURCE=server-bdora3:1521/xe;PASSWORD=CONFIG;USER ID=CONFIG" />
  <!--DESARROLLO-->
  <add name="cs_erpzmx" connectionString="metadata=res://*/Modelo.ModeloOracle.csdl|res://*/Modelo.ModeloOracle.s
  <!--<add name="cs_erpzmx" connectionString="metadata=res://*/Modelo.ModeloOracle.csdl|res://*/Modelo.ModeloOrac
  <!--PRUEBAS-->
  <!--<add name="cs_erpzmx" connectionString="metadata=res://*/Modelo.ModeloOracle.csdl|res://*/Modelo.ModeloOrac
  <!--PREPRODUCCION-->
  <!--<add name="cs_erpzmx" connectionString="metadata=res://*/Modelo.ModeloOracle.csdl|res://*/Modelo.ModeloOrac
  <!--PRODUCCION-->
  <!--<add name="cs_erpzmx" connectionString="metadata=res://*/Modelo.ModeloOracle.csdl|res://*/Modelo.ModeloOrac
  <!--<add name="OracleDbContext" providerName="Oracle.ManagedDataAccess.Client" connectionString="User Id=oracle
</connectionStrings>
```

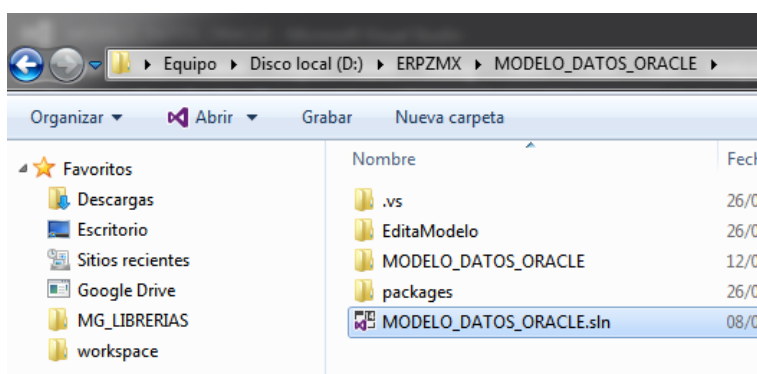
## Generación del modelo con la herramienta de EF6, paso a paso

Para generar el modelo de datos (ensamblado MODELO\_DATOS\_ORACLE.dll), los pasos son los siguientes:

1. Ubicamos la carpeta que contiene la solución del modelo de datos dentro de nuestro repositorio de proyectos, para este ejemplo trabajaremos con el directorio:  
D:\ERPZMX\MODELO\_DATOS\_ORACLE



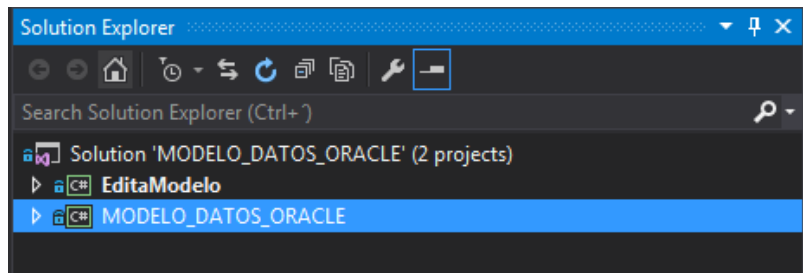
2. Una vez ubicada la carpeta, ingresamos y procedemos a abrir la solución



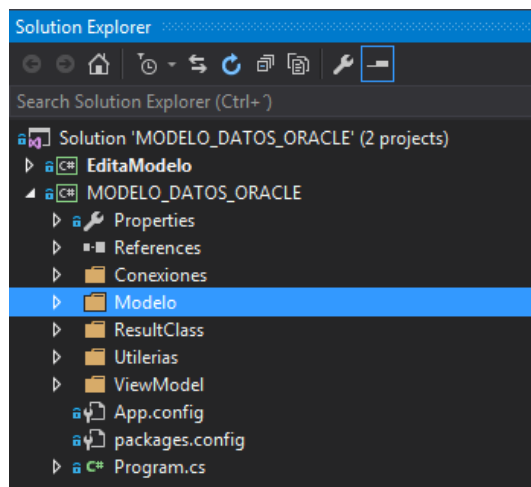
Fecha de elaboración: 2 de Junio del 2017



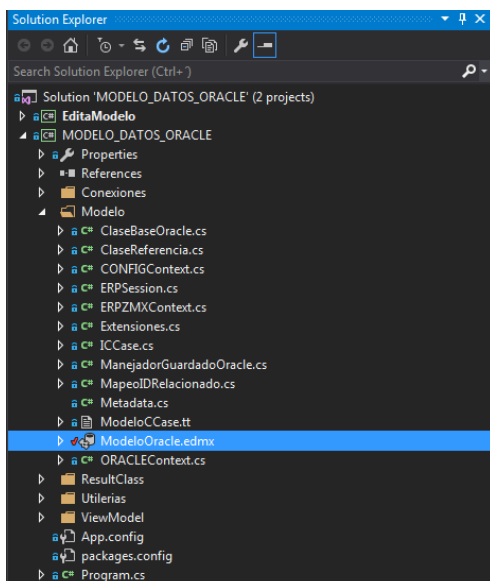
3. Lo siguiente que haremos será, ubicar el proyecto MODELO\_DATOS\_ORACLE dentro del Solution Explorer.



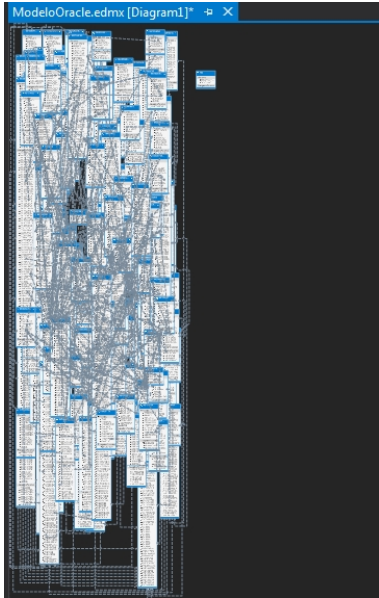
4. Ya localizado el proyecto tendremos que ubicar dentro de este, la carpeta llamada Modelo



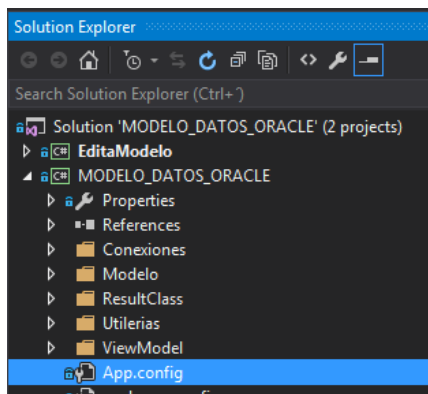
5. Una vez ubicada dicha carpeta, dentro buscaremos el elemento ModeloOracle.edmx



- Una vez abierto, nos tiene que mostrar el diagrama del Modelo de Datos, así como se muestra en la siguiente imagen:



- Procedemos a ubicar el archivo de configuración App.config dentro del proyecto MODELO\_DATOS\_ORACLE, y procederemos a abrirlo.



- Una vez abierto, tenemos que ubicar la sección de ConnectionStrings como se muestra en la siguiente imagen:

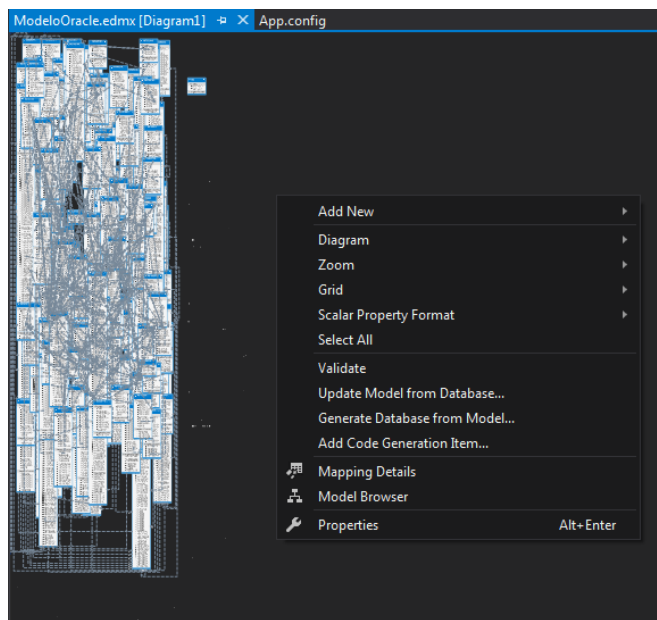
```
<connectionStrings>
  <!-- DESARROLLO -->
  <add name="ORACLEContext" connectionString="metadata=res
  <!-- PRUEBAS -->
  <!--<add name="ORACLEContext" connectionString="metadata
  <!-- PREPRODUCCION -->
  <!--<add name="ORACLEContext" connectionString="metadata
  <!-- APR -->
  <!--<add name="ORACLEContext" connectionString="metadata
</connectionStrings>
```

En esta sección encontraremos 3 líneas de conexión, una para cada ambiente que son:

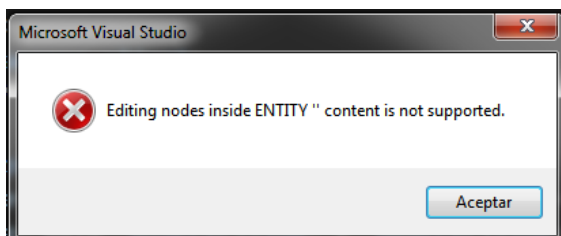
- Desarrollo.
- Pruebas
- Producción

En esta parte, al generar el modelo, **ÚNICAMENTE** debe de estar descomentada la línea de la cadena de conexión del ambiente para el cual queremos generar el modelo de datos.

9. Una vez ubicada esta sección, regresamos al modelo de datos (ModeloOracle.edmx), damos clic derecho, y seleccionamos la opción Seleccionar Todo, y por último presionamos la tecla Supr, y esperamos un momento a que termine de procesar.



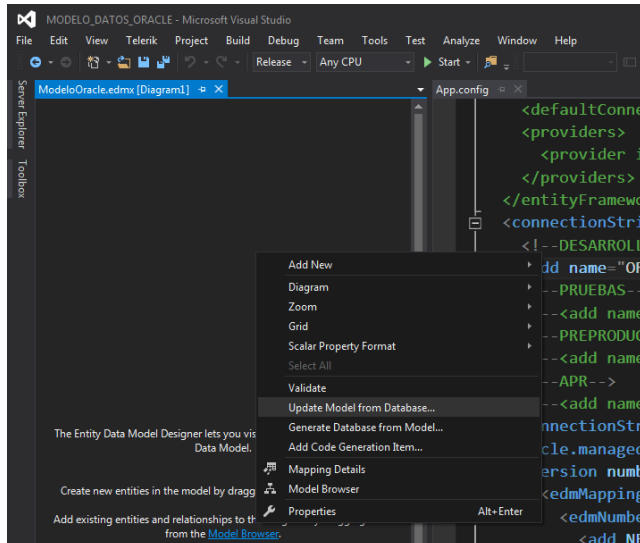
10. Una vez que termine, aparecerá el siguiente mensaje, y damos click en Aceptar y guardamos.



11. Luego regresamos al archivo de configuración App.config y nos aseguramos que este descomentada la cadena de conexión para el ambiente que queremos generar el modelo, de lo contrario, editamos el archivo y guardamos.
12. Después nos aseguramos de haber borrado el diagrama del modelo de datos y hacer los cambios en el archivo de configuración, utilizamos la opción guardar todo de Visual Studio o la

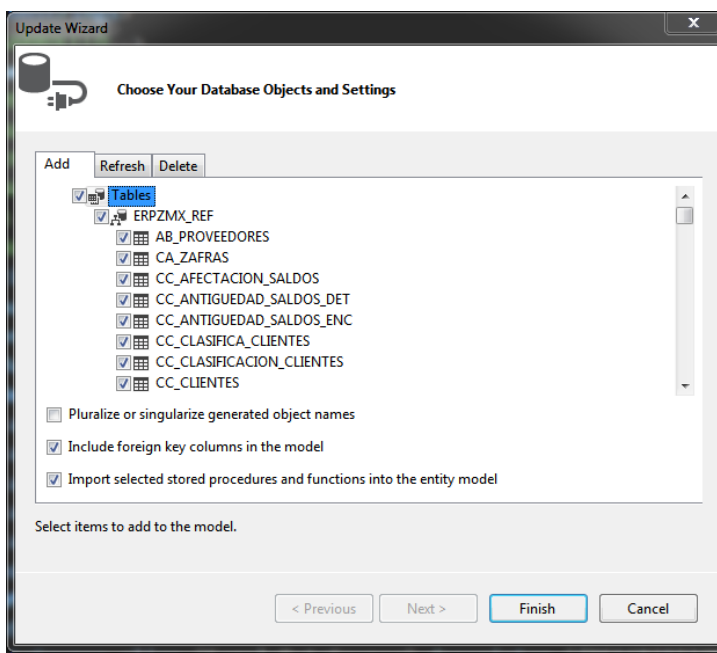
combinación Ctrl+Shift+S, una vez hecho esto, cerramos la solución y volvemos a abrirla.

13. Una vez abierta la solución vamos al ModeloOracle.edmx, damos click derecho en el espacio donde solia estar el diagrama y seleccionamos la opción: Actualizar Modelo Desde base de Datos.



14. Habiendo seleccionado la opción, nos aparecerá la siguiente ventana donde seleccionaremos la opción: Tablas y nos aseguramos de que estemos en el ambiente correcto (Archivo App.config), esto lo sabemos por el esquema que nos muestra, la correspondencia Esquema-Ambiente es la siguiente:

- ERPZMX\_REF → Desarrollo/Debug
- ERPZMX\_TEST- → Pruebas
- ERPZMX → Preproducción



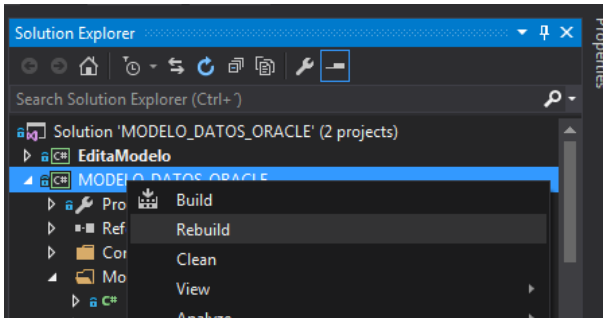
Una vez que verifiquemos la información, damos click en el botón Finalizar y esperamos a que termine de procesar.

15. Una vez que termine de procesar, guardamos.

16. Por último, nos vamos al

Fecha de elaboración: 2 de Junio del 2017

explorador de soluciones, damos click derecho sobre el proyecto MODELO\_DATOS\_ORACLE y seleccionamos la opción recompilar y esperamos a que termine de procesar, si el proceso fue exitoso se generó el ensamblado MODELO\_DATOS\_ORACLE.dll en la carpeta MG\_LIBRERIAS del repositorio.



17. En caso de que falle el proceso, la causa mas común es por que existen diferencias en las tablas de la base de datos, para lo cual se creó un script que nos facilita ver dichas diferencias tales como longitud de tipos Numéricos, diferencia en tipos de datos, diferencia de longitud de varchar, etc., dicho script se encuentra en la carpeta MG\_LIBRERIAS/Utilerias del repositorio.

## Modelo utilizando ADO (versión con aplicación generadora de clases)

Proyecto=ACCESO\_DATOS, MODELO\_DATOS\_ADO

Con el crecimiento del proyecto, se fue haciendo evidente que el modelo de datos utilizando Entity Framework quedaba lento y deficiente al momento de crear el modelo y establecer todas las relaciones que existían en base de datos, además, un problema común que se estuvo presentando, fue el de al cambiar los tipos de variable, se hacia necesaria una recompilación para resolver el problema.

### Proyecto ACCESO\_DATOS

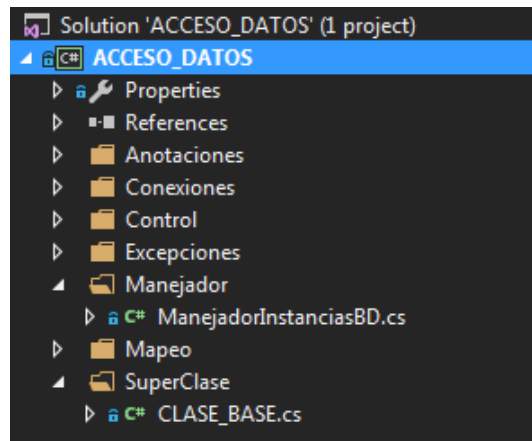
En este proyecto está contenida la lógica para interactuar con la base de datos a través del MODELO\_DATOS\_ADO, por lo que es importante tener cuidado al hacer modificaciones a dicho proyecto.

La idea tras este proyecto se basa en utilizar Reflection y ADO para realizar las consultas a la base de datos y dichos datos asignarlos a los objetos del modelo que los va a utilizar.

Al igual que con el modelo en Entity Framework, cuenta con un Manejador propio y una clase base propia, donde se encuentran los puntos de acceso hacia la lógica de mapeo y consultas.

La estructura del proyecto encapsula y organiza las partes lógicas necesarias de acuerdo a su función, como punto de partida para entender dicha lógica, se puede empezar

examinando la clase **ACCESO\_DATOS.Manejador.ManejadorInstanciasBD**.



## Proyecto MODELO\_DATOS\_ADO

En este proyecto encontramos las clases del modelo que representan las tablas de la base de datos, por lo que es importante conservar una clasificación adecuada de dichas clases de acuerdo al modulo que representan.

Para poder realizar un Mapeo exitoso, se usa como herramienta una serie de anotaciones incrustadas en la clase, que dan información de la equivalencia que existe entre la clase, la tabla, las propiedades y las columnas.

### Anotaciones de clase y propiedad

Las anotaciones de clase, las podemos encontrar en el proyecto de ACCESO\_DATOS, están ubicadas en la carpeta **ACCESO\_DATOS.Anotaciones.Clase** e indican la tabla hacia la que se mapea, la composición de la clave única y las propiedades que conforman la creación de un ID compuesto por distintos datos.

```
namespace MODELO_DATOS_ADO.FA_FACTURACION

[ANC_Tabla(Nombre = "FA_CONSIGNATARIOS")]
[ANC_Identificador(Columna = "CONSIGNA_CODIGO", Propiedad = "CONSIGNA_CODIGO")]
[ANC_Identificador(Columna = "CONSIGNA_ID", Propiedad = "CONSIGNA_ID")]
[ANC_Identificador(Columna = "GRUPO_ID", Propiedad = "GRUPO_ID")]
[ANC_Identificador(Columna = "CLIENTE_ID", Propiedad = "CLIENTE_ID")]
[ANC_IDFormado(Columna = "CONSIGNA_ID", Propiedad = "CONSIGNA_ID")]
[ANC_IDFormadoParte(Columna = "CLIENTE_ID", Digitos = 10, Orden = 1, Propiedad = "CLIENTE_ID")]
[ANC_IDFormadoParte(Columna = "CONSIGNA_CODIGO", Digitos = 4, Orden = 2, Propiedad = "CONSIGNA_CODIGO")]

1 reference | - changes | -authors, -changes
public class FA_CONSIGNATARIOS : CLASE_BASE
```

**En el recuadro azul** están las anotaciones necesarias para identificar un objeto del resto y realizar lógica necesaria en el proceso de mapeo.

**En el recuadro rojo** se indican las anotaciones necesarias para indicar que la propiedad CONSIGNA\_ID se compone de la unión de CLIENTE\_ID y CONSIGNA\_CODIGO con la cantidad de dígitos y orden especificados.

Por otro lado, tenemos las anotaciones de Propiedad, al igual que las anotaciones de clase, están ubicadas en ACCESO\_DATOS en la ruta **ACCESO\_DATOS.Anotaciones.Propiedad**.

El sentido que tienen las anotaciones de propiedad, es indicar configuraciones especiales de las propiedades, por ejemplo, la precisión numérica que debe tener un tipo Double, o si la propiedad es nullable o no. En la siguiente imagen vemos un ejemplo de ello:

```
[ANP_Configuracion(Nullable = false)]  
[ANP_PrecisionNumerica(Enteros = 14)]  
[ANP_Columna(Columna = "LOCALIDAD_ID")]  
0 references | JOSE MAGDIEL MARTINEZ, 2 days ago | 1 author, 1 change  
public Int64? LOCALIDAD_ID { get; set; }
```

#### *Anotaciones para obtener navegaciones mediante propiedades (Método Carga)*

También es posible configurar el Query que nos puede traer una navegación utilizando las propiedades de la entidad que la posee (la entidad base) o enviar dichos parámetros al realizar la carga de la entidad base.

Aquí un ejemplo de la configuración:

```
[ANP_QueryPropiedad(Query = @"SELECT * FROM MG_LOCALIDADES WHERE LOCALIDAD_ID = :LOCALIDAD_ID_ORIGEN")]  
0 references | Abraham Mendez, 43 days ago | 1 author, 1 change  
public MG_LOCALIDADES LOCALIDAD_ORIGEN_ENTIDAD { get; set; }
```

y acá la forma de hacer la carga de la navegación:

```
obj  
.Carga(() => obj.LOCALIDAD_DESTINO_ENTIDAD, mibd)  
.Carga(() => obj.LOCALIDAD_DESTINO_ENTIDAD.MUNICIPIO_ENTIDAD, mibd)  
.Carga(() => obj.LOCALIDAD_DESTINO_ENTIDAD.MUNICIPIO_ENTIDAD.ESTADO_ENTIDAD, mibd)  
.Carga(() => obj.LOCALIDAD_DESTINO_ENTIDAD.MUNICIPIO_ENTIDAD.ESTADO_ENTIDAD.PAIS_ENTIDAD, mibd)  
;
```

En este ejemplo se carga la navegación utilizando parámetros:

```
obj  
.Carga(() => obj.TARIFAS_FLETES_AUTORIZADAS_DETALLE_VIGENCIA, mibd, new Dictionary<string, object>() { { "TARIFAFLE_VIGENCIA", tarifaVigencia } })  
;
```

Solo se agrega un diccionario como parámetro:

```
new Dictionary<string, object>() { { "TARIFAFLE_VIGENCIA", tarifaVigencia } }}
```

## Servidor Backend IIS

Hasta el momento de la elaboración de este documento, el servidor IIS donde se están realizando las publicaciones de las aplicaciones web, es el siguiente:

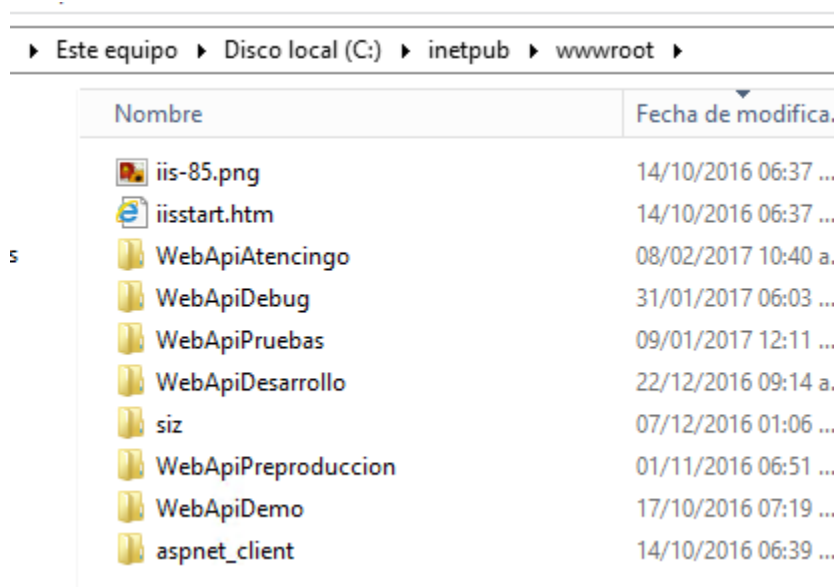
### Cuenta Servidor

IP=192.168.0.109

Usuario(Windows)=servidorcotest

Contraseña(Windows)=SrvCotest16

La estructura física de los archivos es la siguiente:



Nombre	Fecha de modifica.
iis-85.png	14/10/2016 06:37 ...
iisstart.htm	14/10/2016 06:37 ...
WebApiAtencingo	08/02/2017 10:40 a.
WebApiDebug	31/01/2017 06:03 ...
WebApiPruebas	09/01/2017 12:11 ...
WebApiDesarrollo	22/12/2016 09:14 a.
siz	07/12/2016 01:06 ...
WebApiPreproduccion	01/11/2016 06:51 ...
WebApiDemo	17/10/2016 07:19 ...
aspnet_client	14/10/2016 06:39 ...

El contenido de cada sitio web, es el siguiente:

Las carpetas que tienen como prefijo “WebApi”, por ejemplo “WebApiDesarrollo”, contiene una publicación de ApiCatZMX, con la colección de DLL (Compilados) de los proyectos WebApi que están en la solución ApiContainer, entendiendo que la palabra siguiente, **representa el ambiente donde se esta realizando la publicación**, en este caso “Desarrollo”.

Dentro de la carpeta siz, está la publicación del instalador del ClickOnce y si entramos a la carpeta, veremos una carpeta por cada **ambiente** que tengamos hasta el momento, tal como se muestra en la siguiente imagen:



Vista	
te equipo ▶ Disco local (C:) ▶ inetpub ▶ wwwroot ▶ siz ▶	
Nombre	Fecha de mo
demo	15/10/2016 1:
preproduccion	19/12/2016 0:
pruebas	08/12/2016 0:

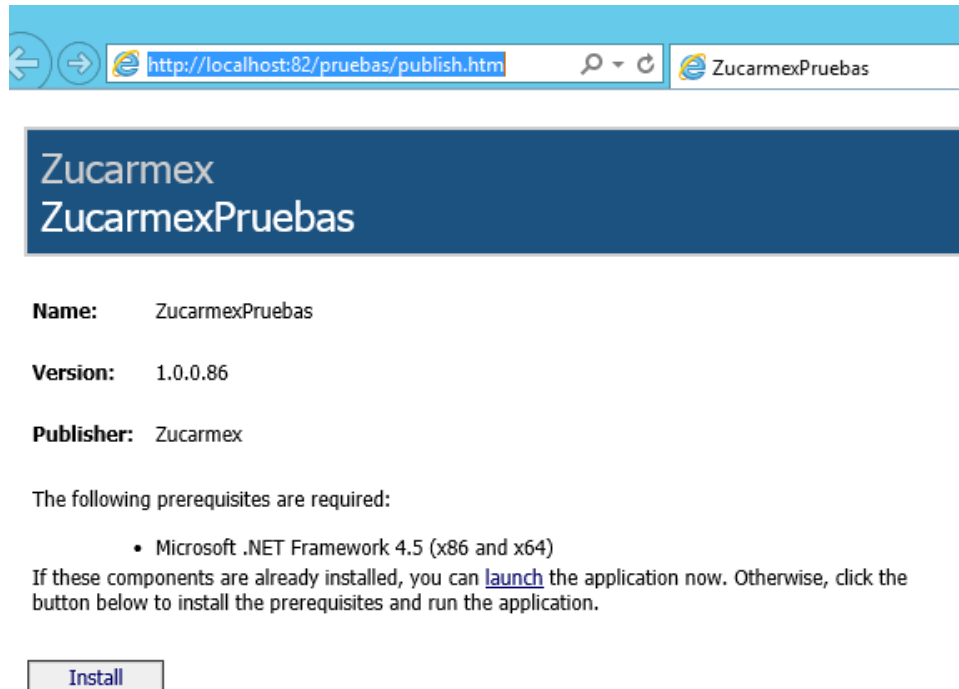
Y el contenido interior es el siguiente:

equipo ▶ Disco local (C:) ▶ inetpub ▶ wwwroot ▶ siz ▶ pruebas ▶		
Nombre	Fecha de modifica...	Ti
Application Files	08/12/2016 06:56 ...	C
publish.htm	08/12/2016 06:55 ...	D
setup.exe	08/12/2016 06:55 ...	A
ZucarmexIniPruebas.application	07/12/2016 01:33 ...	A
ZucarmexIniUpd.application	08/12/2016 06:55 ...	A

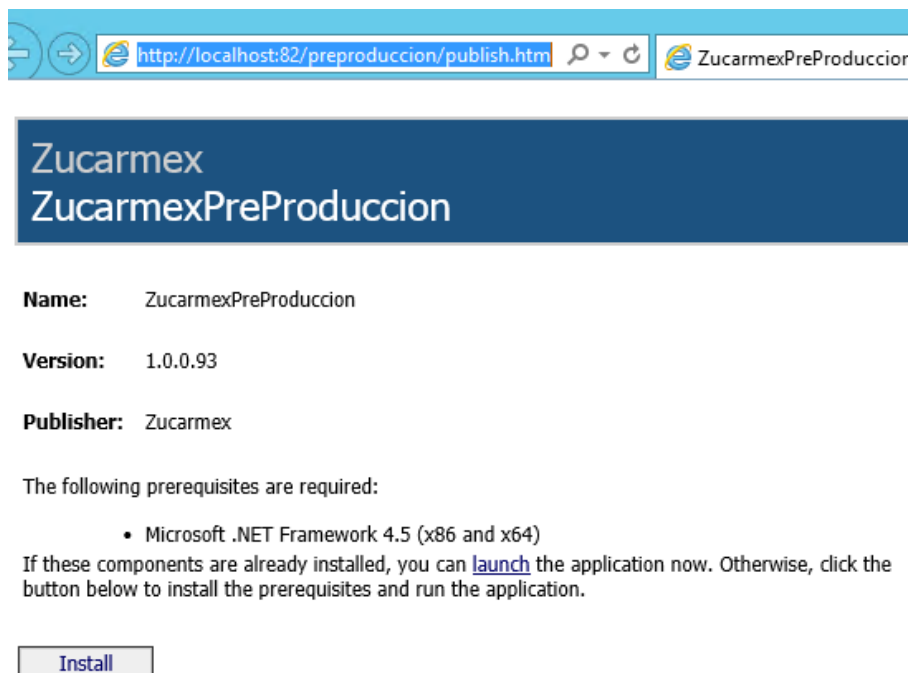
### Distribuir el instalador a los clientes

Por lo que al realizar la distribución de la aplicación, lo que necesitamos hacer es enviar la url que apunta al sitio que necesitamos instalar, por ejemplo:

URL del ambiente de pruebas = <http://localhost:82/pruebas/publish.htm> (donde localhost es la ruta del servidor del IIS)



URL del ambiente de preproducción = <http://localhost:82/preproduccion/publish.htm>



## Configuración de cadena de conexión en WebAPI para ambos modelos

### MODELO\_DATOS\_ORACLE

La configuración de la cadena de conexión para que dicho modelo funcione de forma correcta, se realiza en el archivo Web.Config de las publicaciones en el servidor de IIS para cada uno de sus ambientes (tal como se vió en un capítulo anterior), por lo que lo único que se necesita es crear la cadena de conexión con este formato:

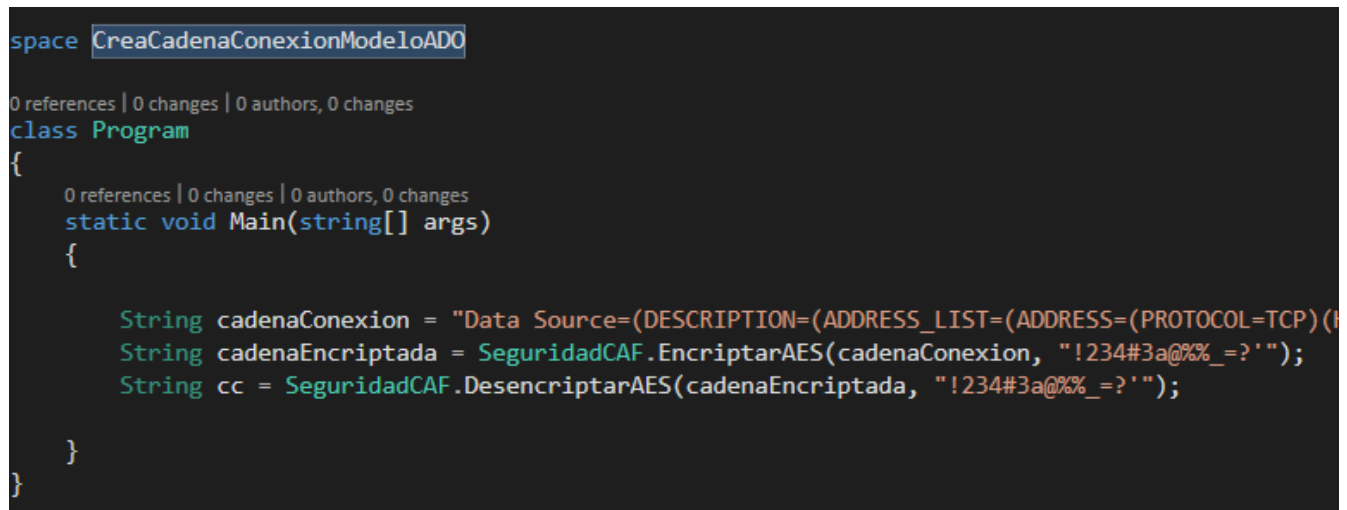
```
<add name="cs_erpzmx" connectionString="metadata=res:/**/*.Modelo.ModeloOracle.csdl|  
res:/**/*.Modelo.ModeloOracle.ssdl|  
res:/**/*.Modelo.ModeloOracle.msl;provider=Oracle.ManagedDataAccess.Client;provider  
connection string=&quot;DATA SOURCE=server-  
bdora3:1521/xe;PASSWORD=erpzmx_test;USER ID=erpzmx_test&quot;;" />
```

### MODELO\_DATOS\_ADO

Como se mencionó anteriormente, la cadena en este caso se encuentra encriptada y oculta del debugger, con esto se agrega seguridad para ocultar datos sensibles al desarrollador.

#### Generar cadena conexión MODELO\_DATOS\_ADO

Para generar la cadena encriptada, está un proyecto consola dentro de la solución ApiContainer del repositorio llamado “CreaCadenaConexionModeloADO”, dentro del código está un ejemplo de como generar dicha cadena, una vez generada dicha cadena, se agrega al archivo Web.Config de cada ambiente del servidor IIS.

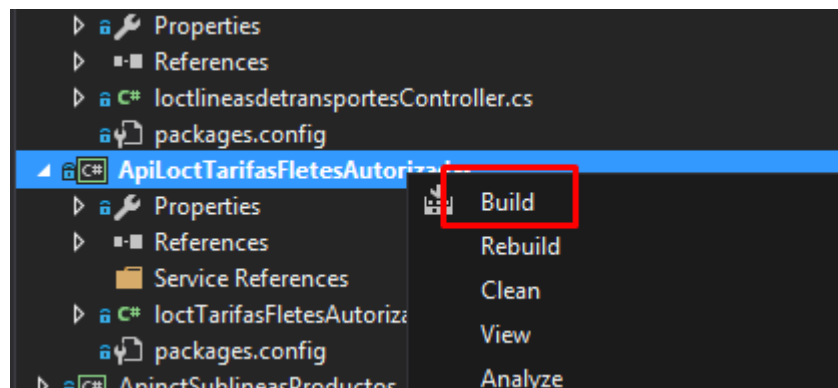


```
space CreaCadenaConexionModeloADO  
0 references | 0 changes | 0 authors, 0 changes  
class Program  
{  
    0 references | 0 changes | 0 authors, 0 changes  
    static void Main(string[] args)  
    {  
  
        String cadenaConexion = "Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(I  
        String cadenaEncriptada = SeguridadCAF.EncriptarAES(cadenaConexion, "!234#3a@%%_=?'");  
        String cc = SeguridadCAF.DesencriptarAES(cadenaEncriptada, "!234#3a@%%_=?'");  
  
    }  
}
```

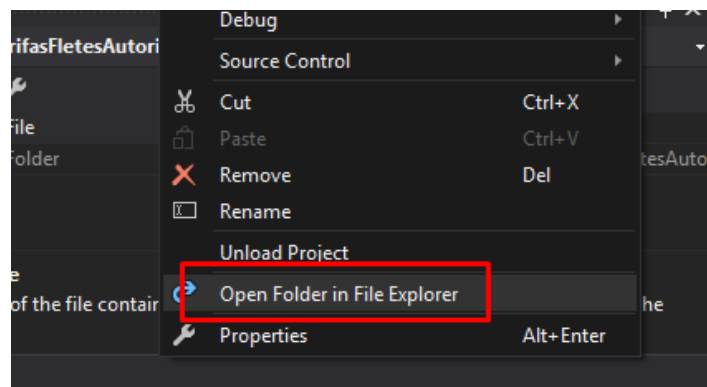
## Publicación de WebApi, Solución ApiContainer

El procedimiento para realizar una publicación de un WebApi hacia el servidor es el siguiente tomando como ejemplo el proyecto “ApiLoctTarifasFletesAutorizadas”.

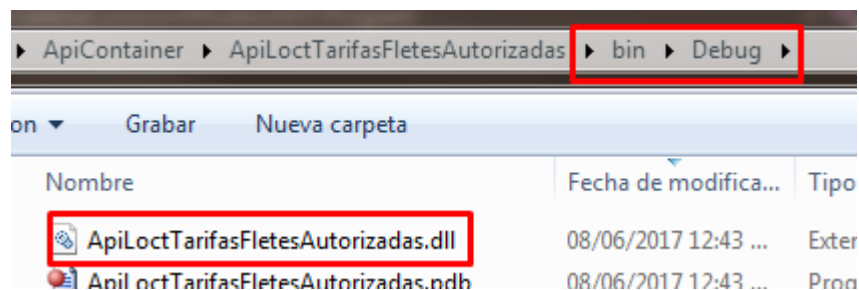
**-Se compila el proyecto con la opción build**



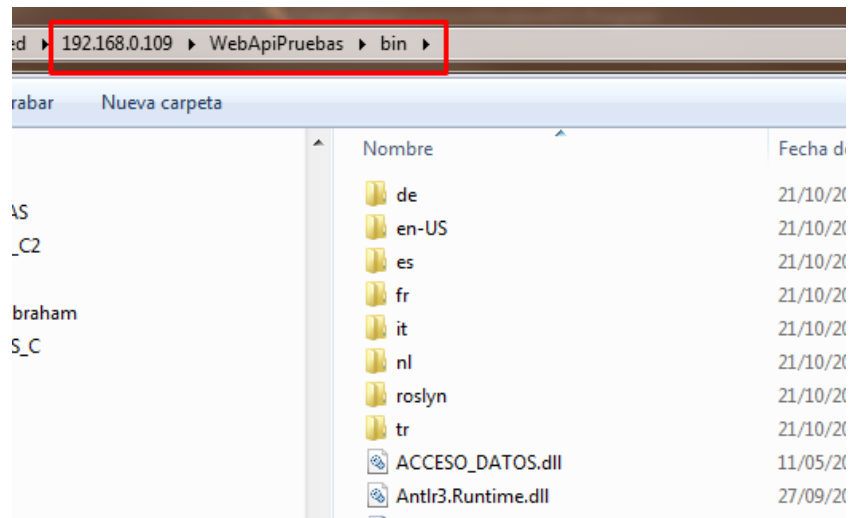
**-Vamos a la carpeta de compilación**



**-Accedemos a la carpeta Bin/Debug y copiamos la DLL generada**



**-Por ultimo la pegamos en el ambiente correspondiente**



## Como debuguear un WebApi de ApiContainer en el servidor IIS

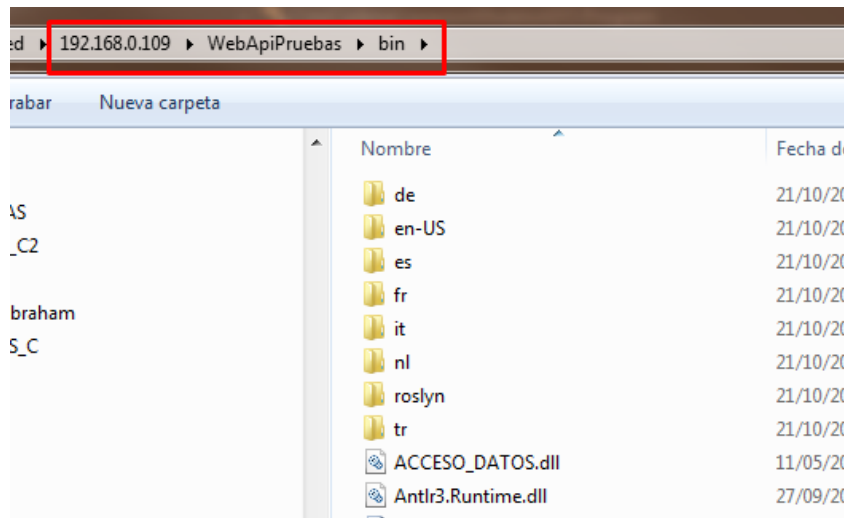
Para debuguear un WebApi, se tienen que hacer correctamente los pasos de publicación de WebApi vistos anteriormente, teniendo especial cuidado en copiar todas las DLL que tengan como fecha la misma hora de la DLL del WebApi, nuevamente tomaremos como ejemplo “ApiLoctTarifasFletesAutorizadas”.

### -Copiamos las DLL con la ultima fecha

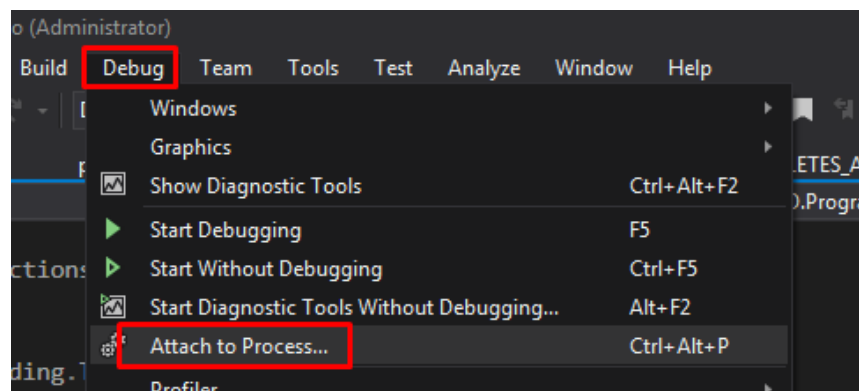
Nombre	Fecha de modifica...	Tipo	Tamaño
ApiLoctTarifasFletesAutorizadas.dll	08/06/2017 12:43 ...	Extensión de la apl...	58 KB
ApiLoctTarifasFletesAutorizadas.pdb	08/06/2017 12:43 ...	Program Debug D...	48 KB
MODELO_DATOS_ADO.dll	08/06/2017 12:43 ...	Extensión de la apl...	56 KB
MODELO_DATOS_ADO.pdb	08/06/2017 12:43 ...	Program Debug D...	136 KB
CustomFiltersMVC.dll	08/06/2017 12:35 ...	Extensión de la apl...	13 KB
CustomFiltersMVC.pdb	08/06/2017 12:35 ...	Program Debug D...	26 KB
UTILERIASERP.dll	08/06/2017 10:42 a...	Extensión de la apl...	1,172 KB
UTILERIASERP.pdb	06/06/2017 10:11 a...	Program Debug D...	544 KB
UserControlsSIZ.dll	26/05/2017 8:53 a....	Extensión de la apl...	1,577 KB
UserControlsSIZ.pdb	26/05/2017 8:53 a....	Program Debug D...	470 KB
UTILERIASERP.xml	23/05/2017 10:32 a...	Documento XML	13 KB

En este caso de la imagen anterior, aparece el MODELO\_DATOS\_ADO con la ultima fecha igual que el ensamblado de nuestro controller, esto es porque ese proyecto estaba adjunto al momento de la compilación, es importante copiar estas DLL junto con la dll de la WebApi, para evitar complicaciones.

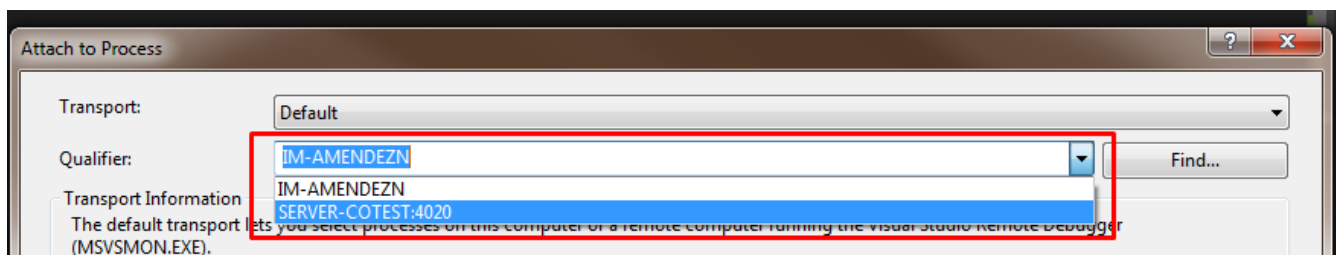
-Una vez copiadas las pegamos en la carpeta bin del ambiente que queremos debuguear en el servidor IIS.



-En el Visual Studio, teniendo abierta la solución ApiContainer, se accede a la opción del menú “Debug” y posteriormente a “Attach to Process”

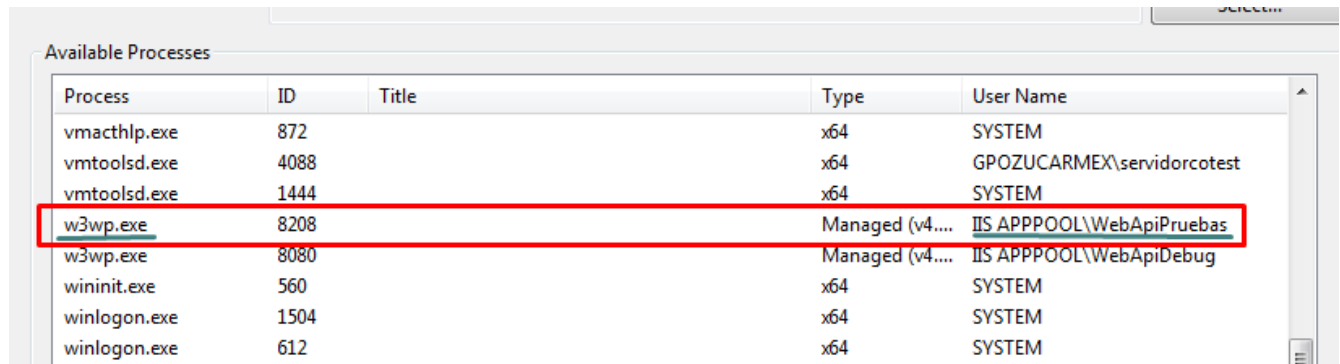


-Seleccionamos el servidor IIS donde esta publicado el Servicio Web

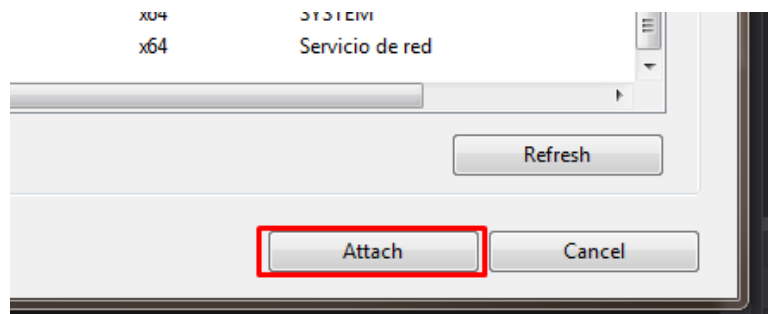


En este caso, SERVER-COTEST es lo mismo que 192.168.0.109

-Seleccionamos el proceso que tenga como nombre el ambiente que ocupamos debuggear



-Por ultimo, seleccionamos el botón Attach

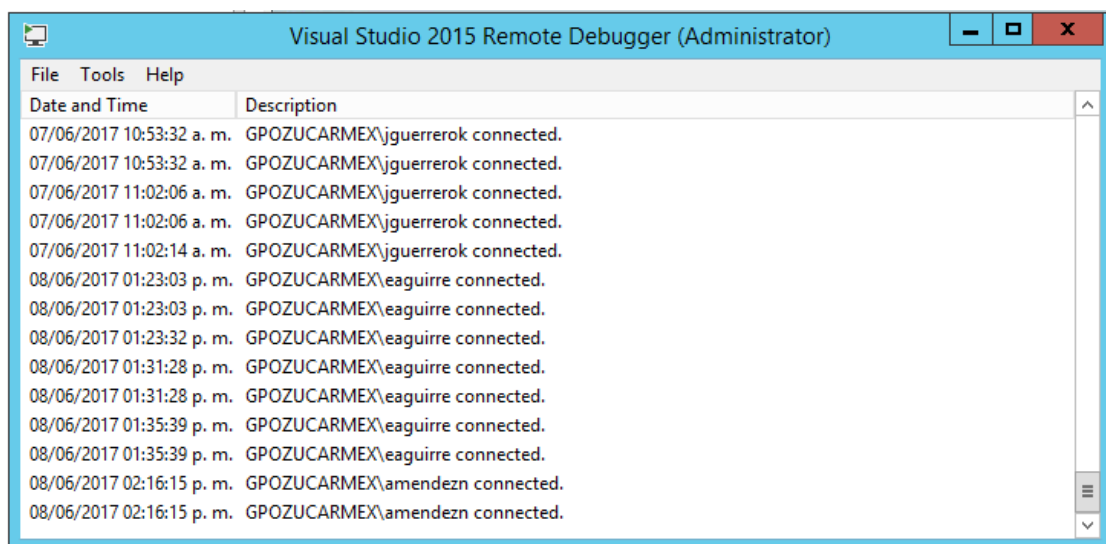


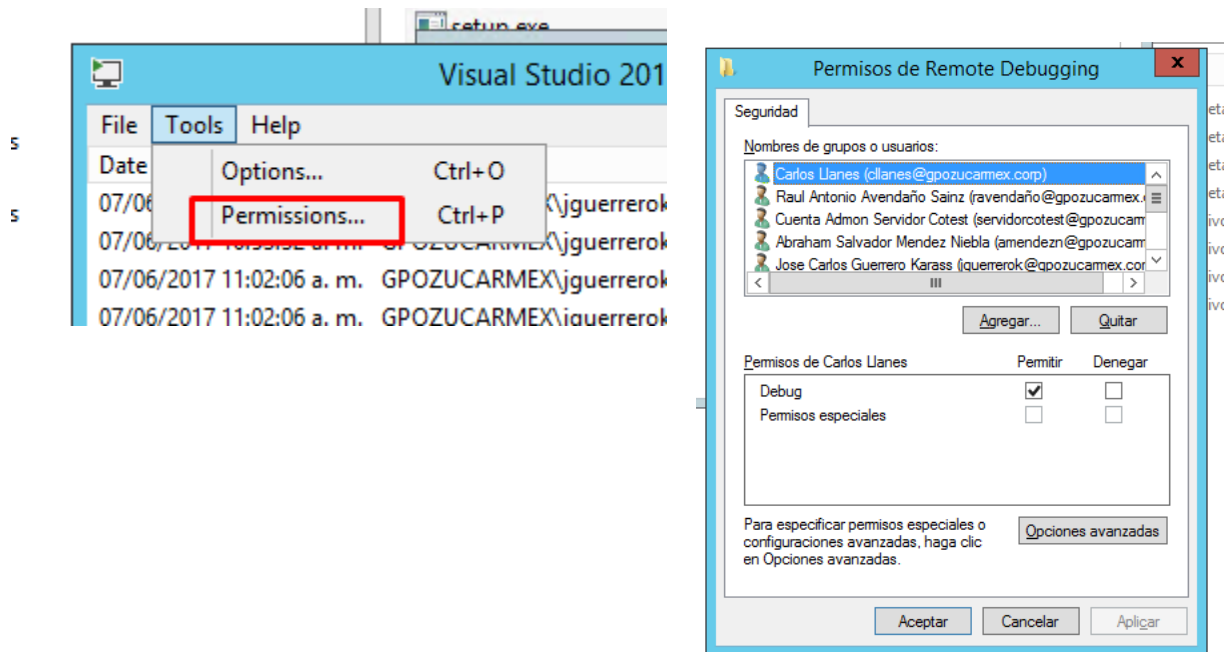
## NOTA IMPORTANTE

Si el ambiente no aparece en el listado de procesos, realizar una petición antes con cualquier otro proyecto que apunte al mismo ambiente y así este va a aparecer en la lista después de hacer un refresh.

## Configuración de Remote Debugger en el servidor IIS

Para poder acceder al debugger remoto, es necesario realizar la instalación y configuración de la herramienta **Remote Debugger** de Microsoft, también realizar la configuración de permisos de usuarios del Dominio que tienen acceso al servidor.





## Instalación del cliente WPF

[#Ir a Instalador de cliente](#)

## MG\_UserControls

Proyecto=MG\_UTILERIASGRALES, Clase=UTILERIASERP.Utilerias.MG\_UserControl

Todos los formularios (UserControls) heredan de esta clase, por lo que es importante saber que en dicha clase se encuentran los mecanismos necesarios para que **funcione la secuencia**, así como otras funcionalidades donde se cargan configuraciones.

Podemos encontrar:

## Propiedad SessionERP

Propiedad donde se encuentra la sesión que el menú principal asigna a una pantalla antes de abrirse.



### Propiedad PantallaSinPromptCuentasContables

Se activa cuando no existen prompts de cuentas contables en el formulario y así evitar una consulta hacia las configuraciones para saber la longitud máxima de las cuentas contables con las que el control de prompt hace validaciones al escribir.

### Propiedad ControlesAgregadosASecuencia

Contiene un listado de los controles agregados a la secuencia, en el cual se pueden verificar las propiedades que los controles tienen en la secuencia y así poder dar seguimiento en caso de falla o error.

### Boolean HayRequeridosSinCapturar()

Marca los controles requeridos sin capturar, a su vez retorna un valor booleano que indica si hay requeridos sin capturar.

### Secuencia de controles

Proyecto=MG\_UTILERIASGRALES, Clase=UTILERIASERP.Utilerias.MG\_UserControl

Se encuentra en MG\_UserControls, es un conjunto de métodos y eventos que controlan sistemáticamente el comportamiento de captura de los distintos controles y de acuerdo a eventos, se valida su estado para mover el foco de control a control. Cada control es tratado de forma distinta, por lo que se utilizan la propiedad adecuada de cada control y los eventos que disparan cuando un valor se actualiza o no para dicho control.

Su funcionamiento se basa en si el control es requerido o no y solo considera aquellos controles que están validados para la secuencia que tienen un valor en TabIndex, por lo que este valor no debe repetirse. Se recomienda asignar valores en sus TabIndex con valores Saltados, por ej: (3,5,7,9...).

En caso de dar clic sobre un control de la secuencia y algún control anterior aún no es capturado, indica mediante un tooltip el ultimo control que es requerido de capturar, enviando el foco hacia dicho control.

Documento Serie Folio Fecha

3-FACTURAS | Valor Requerido 06/06/2017

Cliente \*

Consignatario

Concepto \*

Cuenta de Flujo

Plazo de Pago \*

1 Clic sobre este control

2 Muestra el tooltip y ubica el foco sobre él.

Concepto  Serie  Folio  Fecha

Valor Requerido

Descripción

Signatario

Plazo de Pago

Monto de Flujo

Botones: +, ✎, -

Partida	Concepto Facturable	Unidad	Cantidad	Precio Unitario	Importe	Impuesto 1	Impuesto 2	Retención
---------	---------------------	--------	----------	-----------------	---------	------------	------------	-----------

Para lograr dicho efecto, es necesario en el evento del botón, invocar el siguiente método:

```
if (HayRequeridosSinCapturar())  
    return;
```

A su vez, este método es automáticamente invocado cuando se presiona el botón Guardar de la toolbar, lo que impide que el usuario pueda guardar cuando aún no termina su captura.

## Problemas Comunes

### *La secuencia de controles no esta incluyendo un control.*

Este problema suele ocurrir normalmente cuando en alguna pantalla existe un control que no se ha usado anteriormente.

Secuencia prefijo **ZMX\_** (donde se incluyen los UserControls, usar exclusivamente esta versión)

Dentro del metodo **ZMX\_BuscarControlesParaSecuencia** hay una validación donde se valida si el control iterado (controles dentro de la pantalla) corresponde a un control manejado por la secuencia. El metodo es **ZMX\_EsControlParaSecuencia**, dentro del if, agregar el nuevo control a administrar con la secuencia.

### *El Borde o el asterisco del Requerido se pinta de forma incorrecta.*

Para revisar este problema, en el método **ZMX\_SetControlRequeridoTemplate** dentro de la clase **MG\_UserControls** en el proyecto **MG\_UTILERIASGRALES**, se puede verificar la creación de los controles que se utilizan para dibujar el borde, en este caso, algunos controles podrían distorsionar el dibujo de dicho contorno, ya que se realiza un calculo de las dimensiones del control para ello.

## UserControls

Proyecto=UserControls\_SIZ

En este proyecto, se encuentran controles de usuario importantes que se utilizan en la interfaz principal de WPF, por ejemplo: el control de usuario de Toolbar. El control está ubicado en **UserControlsSIZ.Toolbar.ToolbarSIZ**.

## Estructura

A continuación se definen los principales controles del SIZ

### *Toolbar SIZ*

El control de usuario de Toolbar que contienen todas las ventanas del SIZ, esta compartido en dos ubicaciones, debido a que la primer etapa del proyecto su apariencia era con otro estilo, por lo que para conservar las referencias, se conservo la clase ubicada en:

Proyecto= ToolBarZMX, Clase=ToolBar.xaml

La clase anterior (ToolBar.xaml), es utilizada como clase fachada, en donde se encuentra ubicado el control **ToolbarSIZ** (Versión mejorada de toolbar), lo estrategia utilizada, fue

redirigir eventos que tenía la anterior toolbar hacia las funcionalidades de la versión mejorada.

Ubicación: UserControlSIZ.Toolbar.ToolbarSIZ

Con la intención de mejorar el rendimiento del control y de la pantalla, esta hace consultas hacia diferentes locaciones, una de ellas es obtener el nombre de grupos y empresas contenidas en la sesión y así mostrar los nombres completos dentro de sí misma.

Para habilitar o deshabilitar permisos en el uso de los botones, existen propiedades con las cuales activarse, por ejemplo:

EnabledEditar, EnabledEjecutar ← ésto activa los botones específicos

PermisoEscritura, PermisoImpresion ← ésto permite usar un conjunto u otro de botones

### *Botón de toolbar*

Ubicación: UserControlSIZ.Toolbar.ToolbarBtn

### *Iconos (Iconos usados por el sistema, <FontAwesome>)*

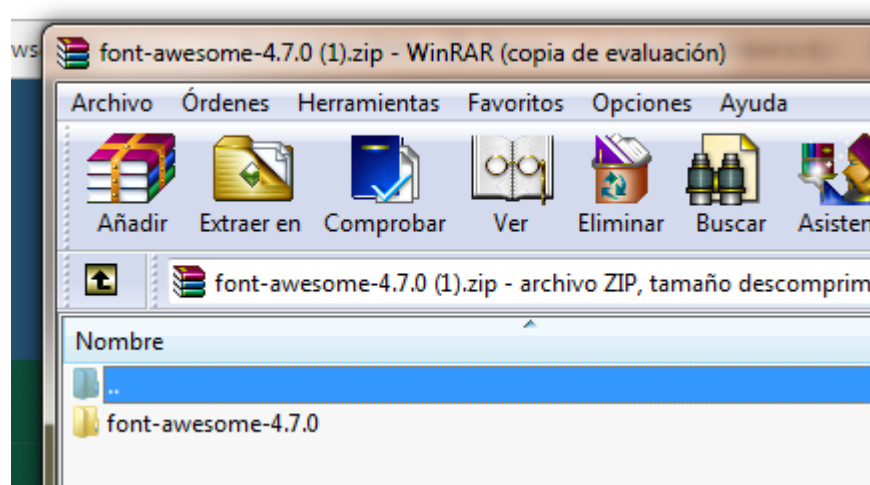
Ubicación: UserControlSIZ.Iconos.IconAwesomeSIZ

Este control es muy importante, ya que ayuda en el aspecto visual de todo el sistema, se basa en utilizar el archivo TTF (Fuente) del sitio FontAwesome (<http://fontawesome.io/>), en caso de necesitar actualizar la fuente, es necesario actualizar los arreglos de propiedades que hacen posible utilizar los iconos de forma mas sencilla y no por su código alfanumérico.

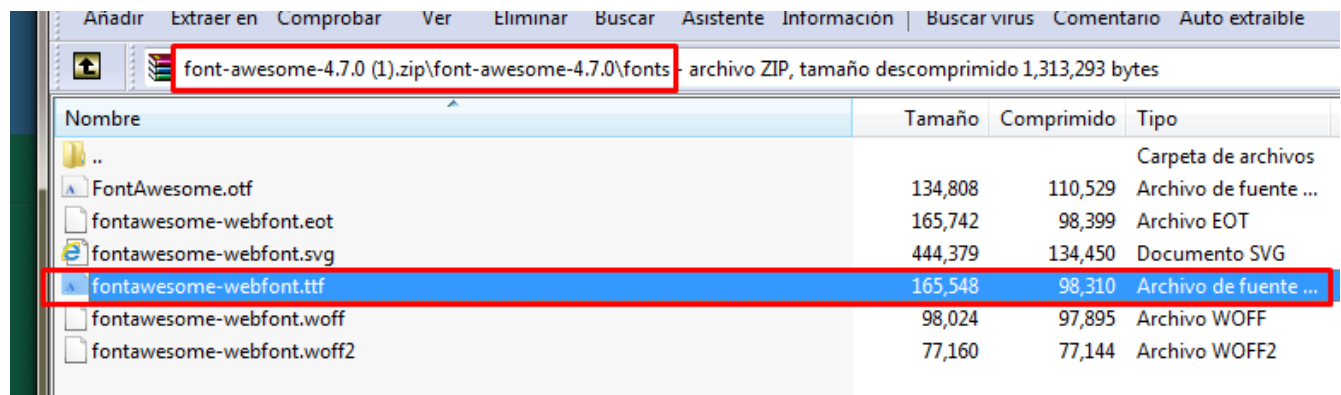
Para ello, descargamos el archivo del sitio web de la siguiente manera:



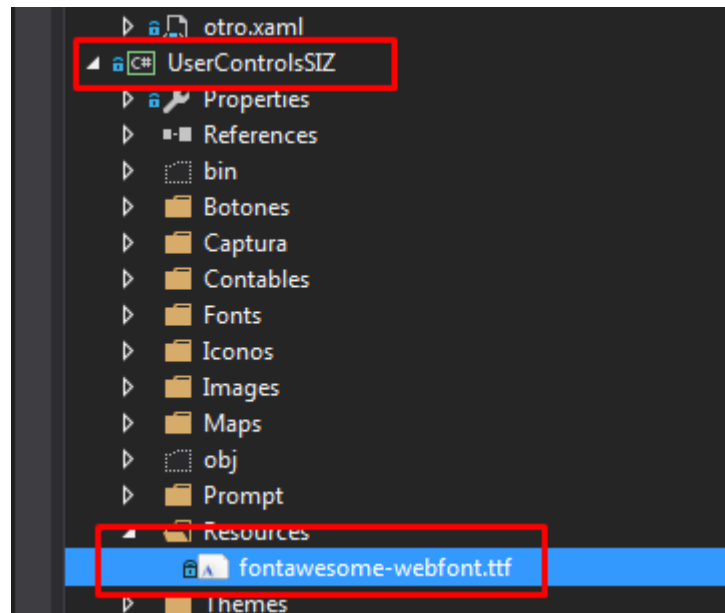
Al descargar el archivo se muestra de la siguiente forma:



El archivo a importar al proyecto como recurso, es el siguiente:

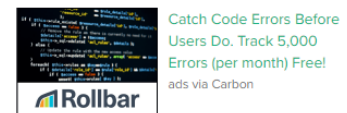


Por ultimo en el proyecto lo agregamos como **recurso**:



Es necesario ir al sitio en su apartado de cheatsheet (<http://fontawesome.io/cheatsheet/>), el cual muestra algo similar a lo siguiente:


Print this page to PDF for the complete set of vectors. Or to use on the desktop, install FontAwesome.otf, set it as the font in your application, and copy and paste the icons (not the unicode) directly from this page into your designs.



## Every Font Awesome 4.7.0 Icon, CSS Class, & Unicode

 fa-500px [ <a href="#">&amp;#xf26e;</a> ]	4.4	 fa-address-book [ <a href="#">&amp;#xf2b9;</a> ]	4.7	 fa-address-book-o [ <a href="#">&amp;#xf2ba;</a> ]	4.7	 fa-address-card [ <a href="#">&amp;#xf2bb;</a> ]	4.7
 fa-address-card-o [ <a href="#">&amp;#xf2bc;</a> ]	4.7	 fa-adjust [ <a href="#">&amp;#xf042;</a> ]		 fa-adn [ <a href="#">&amp;#xf170;</a> ]		 fa-align-center [ <a href="#">&amp;#xf037;</a> ]	4.4
 fa-align-justify [ <a href="#">&amp;#xf039;</a> ]		 fa-align-left [ <a href="#">&amp;#xf036;</a> ]		 fa-align-right [ <a href="#">&amp;#xf038;</a> ]		 fa-amazon [ <a href="#">&amp;#xf270;</a> ]	
 fa-ambulance [ <a href="#">&amp;#xf099;</a> ]		 fa-american-sign-language-interpreting [ <a href="#">&amp;#xf2a3;</a> ]	4.6	 fa-anchor [ <a href="#">&amp;#xf13d;</a> ]		 fa-android [ <a href="#">&amp;#xf17b;</a> ]	
 fa-angle-double-left [ <a href="#">&amp;#xf100;</a> ]		 fa-angle-double-right [ <a href="#">&amp;#xf101;</a> ]		 fa-angellist [ <a href="#">&amp;#xf209;</a> ]	4.2	 fa-angle-double-down [ <a href="#">&amp;#xf103;</a> ]	
 fa-angle-left [ <a href="#">&amp;#xf104;</a> ]		 fa-angle-right [ <a href="#">&amp;#xf105;</a> ]		 fa-angle-double-up [ <a href="#">&amp;#xf102;</a> ]		 fa-angle-down [ <a href="#">&amp;#xf107;</a> ]	
 fa-archive [ <a href="#">&amp;#xf187;</a> ]		 fa-area-chart [ <a href="#">&amp;#xf1fe;</a> ]	4.2	 fa-angle-up [ <a href="#">&amp;#xf106;</a> ]		 fa-apple [ <a href="#">&amp;#xf179;</a> ]	
 fa-arrow-circle-o-down [ <a href="#">&amp;#xf01a;</a> ]		 fa-area-chart [ <a href="#">&amp;#xf1fe;</a> ]	4.0	 fa-arrow-circle-down [ <a href="#">&amp;#xf0ab;</a> ]		 fa-arrow-circle-left [ <a href="#">&amp;#xf0a8;</a> ]	
 fa-arrow-circle-right [ <a href="#">&amp;#xf0a9;</a> ]		 fa-arrow-circle-o-left [ <a href="#">&amp;#xf190;</a> ]		 fa-arrow-circle-o-right [ <a href="#">&amp;#xf18e;</a> ]	4.0	 fa-arrow-circle-o-up [ <a href="#">&amp;#xf01b;</a> ]	
 fa-arrow-right [ <a href="#">&amp;#xf061;</a> ]		 fa-arrow-circle-up [ <a href="#">&amp;#xf0aa;</a> ]		 fa-arrow-down [ <a href="#">&amp;#xf063;</a> ]		 fa-arrow-left [ <a href="#">&amp;#xf060;</a> ]	
 fa-arrows-h [ <a href="#">&amp;#xf07e;</a> ]		 fa-arrow-up [ <a href="#">&amp;#xf062;</a> ]		 fa-arrows [ <a href="#">&amp;#xf047;</a> ]		 fa-arrows-alt [ <a href="#">&amp;#xf0b2;</a> ]	
 fa-asterisk [ <a href="#">&amp;#xf00a;</a> ]		 fa-arrows-v [ <a href="#">&amp;#xf07d;</a> ]		 fa-asl-interpreting [ <a href="#">&amp;#xf2a3;</a> ]	4.6	 fa-assistive-listening-systems [ <a href="#">&amp;#xf2a2;</a> ]	4.6
 fa-at [ <a href="#">&amp;#xf1fa;</a> ]	4.2	 fa-audio-description [ <a href="#">&amp;#xf29a;</a> ]	4.6	 fa-automobile [ <a href="#">&amp;#xf1b0;</a> ]	4.1		

Cada icono se conforma por su nombre y su código:

 fa-500px [[&#xf26e;](#)]

Seleccionamos **todos** los códigos de la siguiente manera:

Fecha de elaboración: 2 de Junio del 2017

## Every Font Awesome 4.7.0 Icon, CSS Class, & Unicode

<a href="#">fa-500px [&amp;#xf26e;]</a> 4.4	<a href="#">fa-address-book [&amp;#xf2b9;]</a> 4.7	<a href="#">fa-address-book-o [&amp;#xf2ba;]</a> 4.7	<a href="#">fa-address-card [&amp;#xf2bb;]</a> 4.7
<a href="#">fa-address-card-o [&amp;#xf2bc;]</a> 4.7	<a href="#">fa-adjust [&amp;#xf042;]</a>	<a href="#">fa-adn [&amp;#xf170;]</a>	<a href="#">fa-align-center [&amp;#xf037;]</a>
<a href="#">fa-align-justify [&amp;#xf039;]</a>	<a href="#">fa-align-left [&amp;#xf036;]</a>	<a href="#">fa-align-right [&amp;#xf038;]</a>	<a href="#">fa-amazon [&amp;#xf270;]</a> 4.4
<a href="#">fa-ambulance [&amp;#xf099;]</a>	<a href="#">fa-american-sign-language-interpreting [&amp;#xf2a3;]</a> 4.6	<a href="#">fa-anchor [&amp;#xf13d;]</a>	<a href="#">fa-android [&amp;#xf17b;]</a>
<a href="#">fa-angle-double-left [&amp;#xf100;]</a>	<a href="#">fa-angle-double-right [&amp;#xf101;]</a>	<a href="#">fa-angellist [&amp;#xf209;]</a> 4.2	<a href="#">fa-angle-double-down [&amp;#xf103;]</a>
<a href="#">fa-angle-left [&amp;#xf104;]</a>	<a href="#">fa-angle-right [&amp;#xf105;]</a>	<a href="#">fa-angle-double-up [&amp;#xf102;]</a>	<a href="#">fa-angle-down [&amp;#xf107;]</a>
<a href="#">fa-archive [&amp;#xf187;]</a>	<a href="#">fa-area-chart [&amp;#xf1fe;]</a> 4.2	<a href="#">fa-angle-up [&amp;#xf106;]</a>	<a href="#">fa-apple [&amp;#xf179;]</a>
<a href="#">fa-arrow-circle-o-down [&amp;#xf01a;]</a>	<a href="#">fa-arrow-circle-o-left [&amp;#xf190;]</a> 4.0	<a href="#">fa-arrow-circle-down [&amp;#xf0ab;]</a>	<a href="#">fa-arrow-circle-left [&amp;#xf0a8;]</a>
<a href="#">fa-arrow-circle-right [&amp;#xf0a9;]</a>	<a href="#">fa-arrow-circle-up [&amp;#xf0aa;]</a>	<a href="#">fa-arrow-circle-o-right [&amp;#xf18e;]</a> 4.0	<a href="#">fa-arrow-circle-up [&amp;#xf01b;]</a>
<a href="#">fa-arrow-right [&amp;#xf061;]</a>	<a href="#">fa-arrow-up [&amp;#xf062;]</a>	<a href="#">fa-arrow-down [&amp;#xf063;]</a>	<a href="#">fa-arrow-left [&amp;#xf060;]</a>
<a href="#">fa-arrows-h [&amp;#xf07e;]</a>	<a href="#">fa-arrows-v [&amp;#xf07d;]</a>	<a href="#">fa-arrows [&amp;#xf047;]</a>	<a href="#">fa-arrows-alt [&amp;#xf0b2;]</a>
		<a href="#">fa-asl-interpreting (alias) [&amp;#xf2a3;]</a> 4.6	<a href="#">fa-assistive-listening-systems [&amp;#xf2a2;]</a>

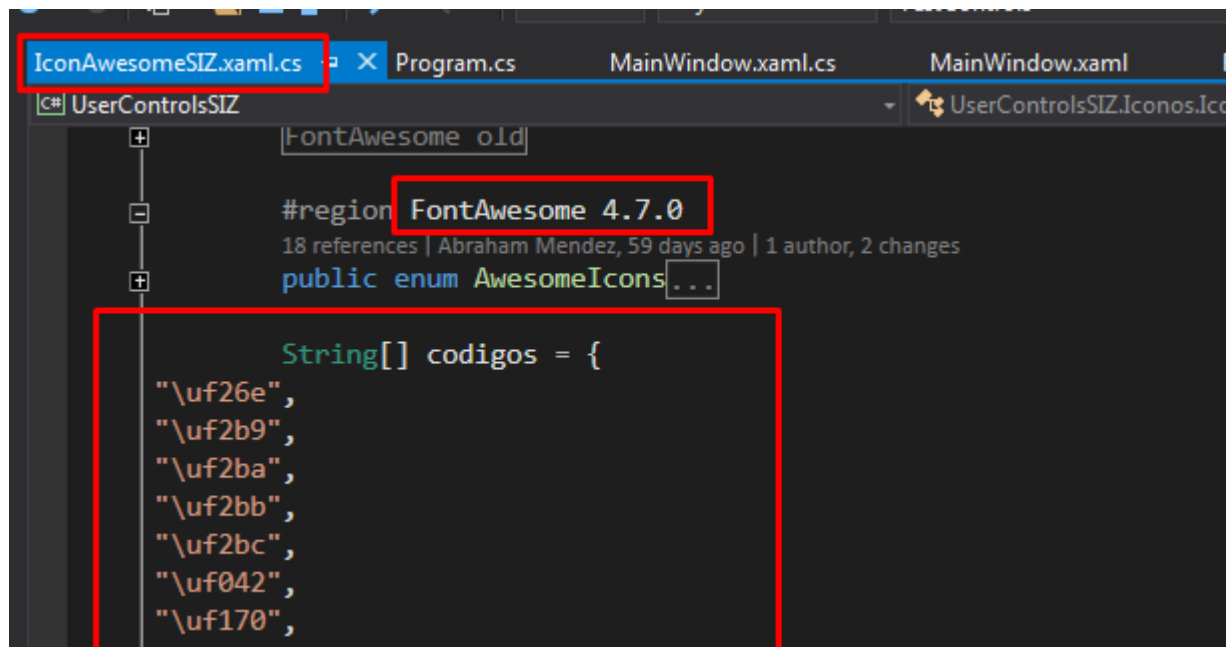
Copiamos en la papelera y vamos al proyecto:

### ConsoleApplication1 del proyecto UserControls\_SIZ

En este proyecto se encuentra dentro del Program.cs un generador donde pegamos lo copiado en la variable fontawesome:

```
[STAThread]
0 references | Abraham Mendez, 59 days ago | 1 author, 2 changes
static void Main(string[] args)
{
    //String fontAwesome = @"fa-500px [&#xf26e;] fa-adjust [&#xf042;] fa-adn [&#xf170;] fa-align-center [&#xf037;] fa-align-ju
    string fontAwesome = @"fa-500px [&#xf26e;] 4.7fa-address-book [&#xf2b9;] 4.7fa-address-book-o [&#xf2ba;] 4.7fa-address-card
    List<String> nombresIconos = new List<string>();
    List<String> codigosNombres = new List<string>();
    String[] fonts = fontAwesome.Split(' ');
```

Al compilar el proyecto, genera código necesario para obtener las propiedades y arreglos necesarios para ser usados desde los proyectos. Dicho código, lo pegamos en la clase IconAwesomeSIZ.xaml.cs:



Si todo se hace de la forma correcta, tendremos actualizados nuestra librería de iconos.

#### [Tooltip de toolbar](#)

Ubicación: UserControlsSIZ.Toolbar.ToolbarTooltip

### **Menú principal**

Proyecto=ZucarmexIniApp

#### [Ubicación de controles de interfaz y estructura](#)

La ubicación de los controles se encuentran en la siguientes rutas:

#### [Login](#)

Ubicación: ZucarmexIniApp.Login

#### [Interfaz Principal](#)

Ubicación: ZucarmexIniApp.PrincipalReducido

#### [Controles que componen la interfaz principal](#)

Ubicación: ZucarmexIniApp.UserControls



### *Controles que componen el popup del menú*

Ubicación: ZucarmexIniApp.UserControlMenuFlotante

### Pendientes para su despliegue en producción

Agregar el mecanismo necesario para que el proyecto (ejecutable) tenga la información necesaria para saber a que servidor Java o .NET apuntar (Backend) en el escenario de un sistema distribuido.

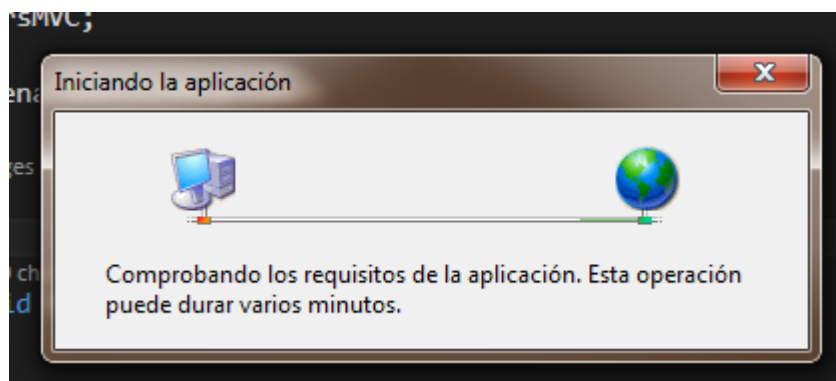
## Esquema de actualizaciones de SIZ

La actualización del cliente WPF del SIZ esta dividida en varias etapas, la primera es actualizar el instalador remoto (ClickOnce), en este caso solo se necesitaría actualizarlo si cambiara la URL a la que apunta el servidor IIS donde esta ubicado el sistema o el instalador ClickOnce, la segunda etapa comprende al actualizador de DLL base, estas son aquellas DLL incluyendo el archivo exe del ZucarmexIniApp que se suben con la herramienta del Actualizador, el cual se explicara en el tema siguiente. Por último la descarga o actualización de DLL sobre demanda, esto significa las DLL de los formularios, las cuales solo se actualizan cuando el usuario abre la opción desde el menú principal.

### Etapa 1: Actualización con ClickOnce

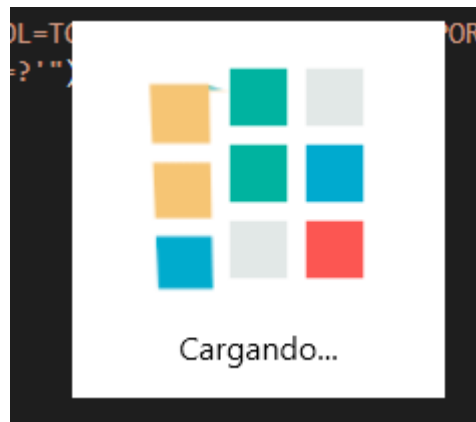
El único escenario donde es necesario actualizar el ClickOnce, es si se necesitara cambiar la aplicación del actualizador, debido a que el ClickOnce solo contiene este archivo exe.

Una vez descargado este archivo exe, se ejecuta y se encarga de descargar el ejecutable del SIZ.



### Etapa 2: Actualización de DLL Base con el splashscreen

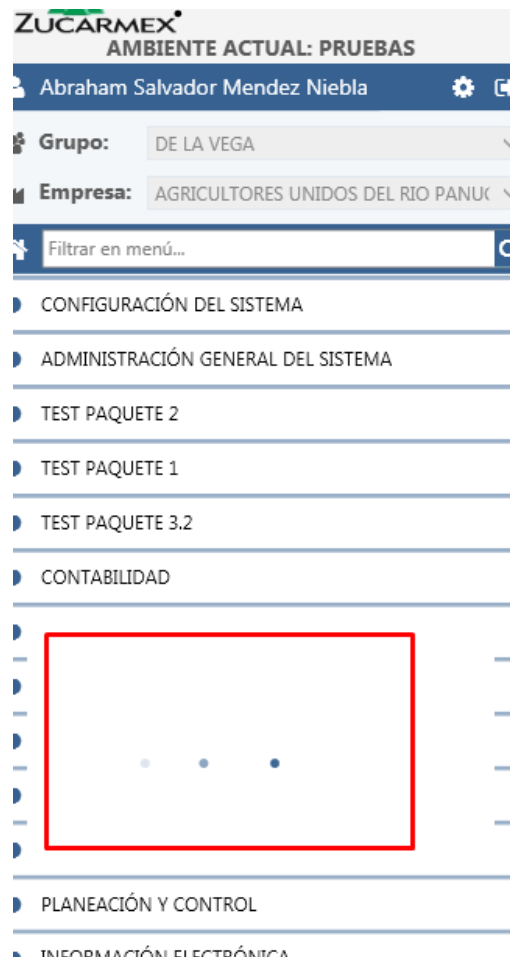
Al ejecutar el actualizador (ZucarmexIniUpd.exe), éste descarga las DLL Base o el ejecutable del sistema (ZucarmexIniApp.exe) si tuvieron algún cambio, por lo que al abrir el sistema, este arranca con las utilerias, el modelo, entre otras siempre en la ultima versión. En el capitulo donde se explica el actualizador de DLL se dan mas detalles de lo mencionado.



### Etapa 3: Actualización de DLL sobre demanda, formas de funcionalidades del negocio

Una vez abierto el menú principal, al dar click sobre una opción del Popup del menú, este inicia un proceso de consultar en el servidor si existe una actualización, esto se hace comparando el checksum de la DLL en la base de datos, con la DLL en la carpeta local del cliente, si existe algún cambio es reemplazada, además si contiene una DLL dependiente, se descarga igualmente, esto se usa en el caso de los catálogos dependientes, por ejemplo Países, Estados, Municipios, Localidades.

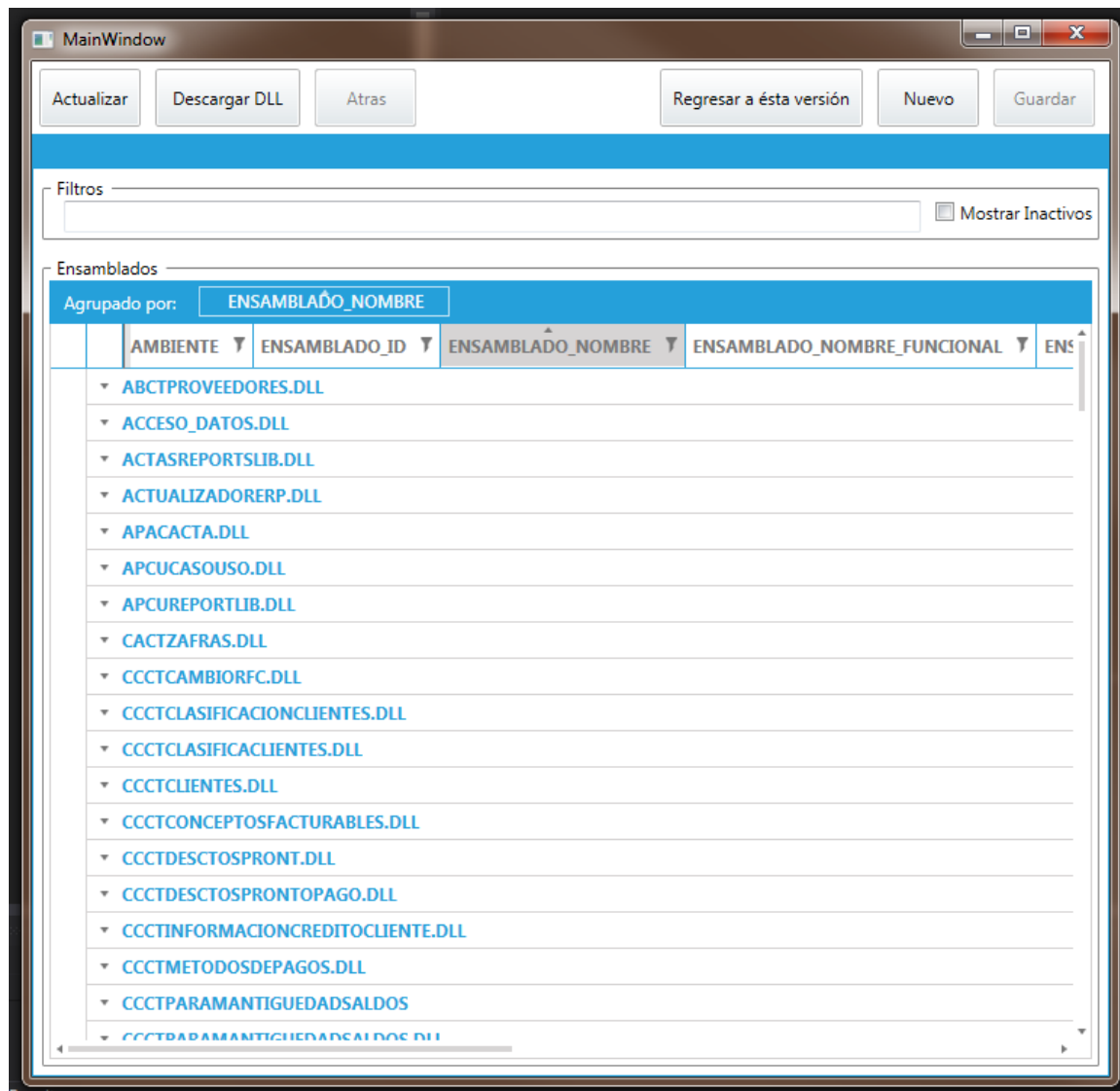
Como se muestra en la siguiente imagen, se muestra la animación que representa el proceso de comparación y descarga de la DLL de la funcionalidad a ejecutar.



### Actualizador de DLL (administrador de actualizaciones)

Solución=ActualizadorConsole, Proyecto=ActualizadorWPF

El actualizador de DLL se creó para cumplir con el requerimiento de actualizar los requerimientos conforme demanda, teniendo la posibilidad de administrar dichas actualizaciones según se necesitara realizar liberaciones.



## Base de datos

El actualizador se apoya de la siguiente base de datos:

**Usuario**=CONFIG

**Contraseña**=CONFIG

**Host**=162.168.0.81

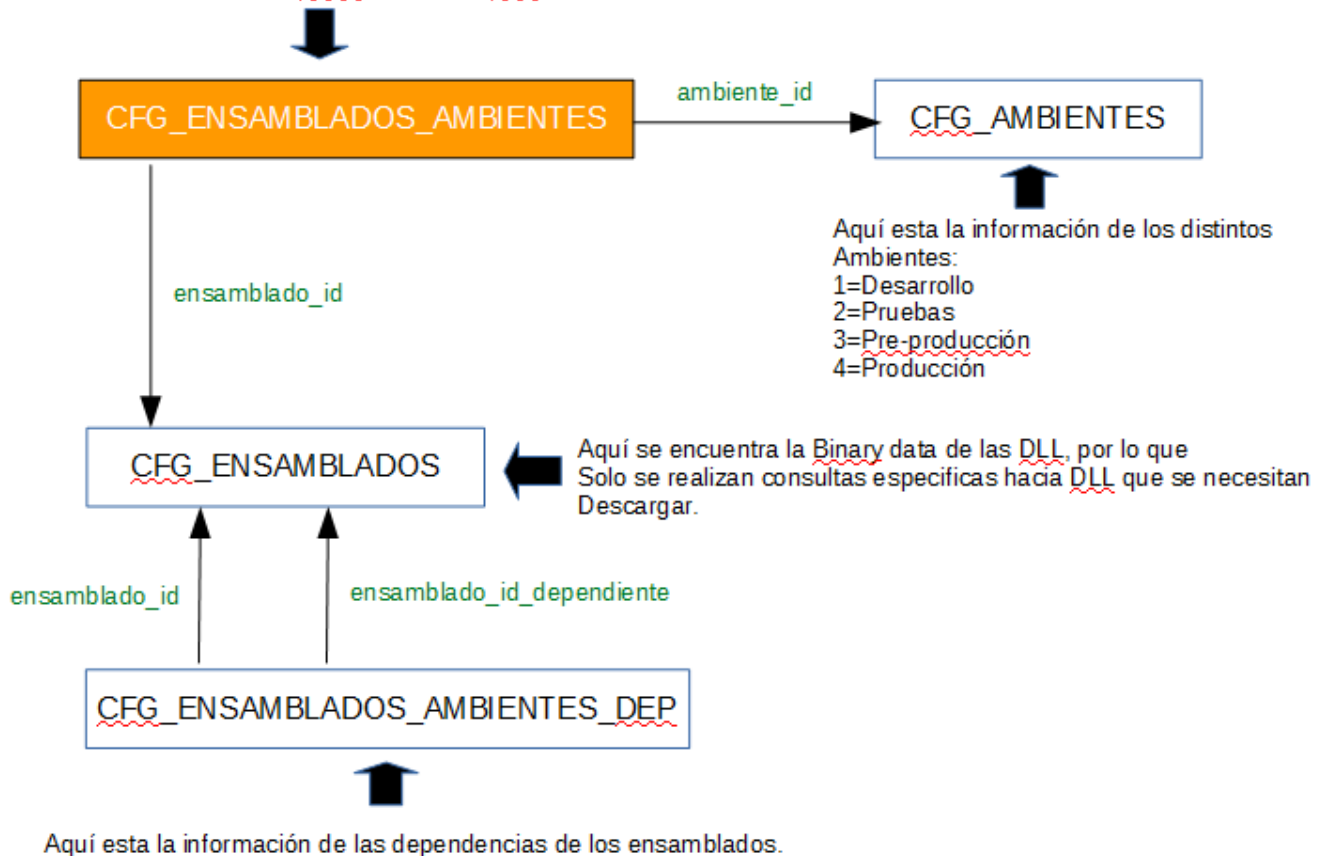
**Puerto**=1521

**SID**=xe

## Tablas

Las tablas utilizadas son:

En esta tabla se realiza el chequeo de las versiones,  
De necesitar actualizar, se accede a CFG\_ENSAMBLADOS  
Donde se accede a la binary data de la DLL.



## Actualizar DLL Base

Las DLL base son aquellas que necesitan ser actualizadas antes de ejecutar el ZucarmexIniApp, estas son aquellas que el sistema utiliza para funcionar, por ejemplo: Utilerías, Controles, Modelo de datos, etc, incluyendo en estas cualquier DLL que necesite estar disponible para el Sistema antes de que éste se abra.

Para indicar que es una DLL base, solo es necesario activar el Check al momento de actualizar la DLL, tal como se muestra en la siguiente imagen:

MainWindow

Actualizar Descargar DLL Atras Regresar a ésta versión Nuevo Guardar

**Ensamblado**

Nombre: ABCTPROVEEDORES.DLL Nombre funcional: abctProveedores.dll

Archivo: ABCTPROVEEDORES.DLL Seleccionar DLL Cargar Recursos Checksum: chv4oD2KDv4dFr9Vo9omQDD8gQy8fzgPkMPNSztjXQ4=

Comentario: abctProveedores.dll Ambiente: PRUEBAS Responsable:

☒ Es ensamblado base

Arrastrar un encabezado de columna y soltarlo aquí al lado del grupo de esa columna

SELECCIONADO	AMBIENTE	ENSAMBLADO_ID	ENSAMBLADO_NOMBRE	ENSAMBLADO_NOMBRE_FUNCIONAL
<input type="checkbox"/>	PRUEBAS	2031	ABCTPROVEEDORES.DLL	abctProveedores.dll
<input type="checkbox"/>	PRUEBAS	2314	ACCESO_DATOS.DLL	ACCESO_DATOS.DLL
<input type="checkbox"/>	PRUEBAS	694	ACTASREPORTSLIB.DLL	ACTASREPORTSLIB.DLL
<input type="checkbox"/>	PRUEBAS	1736	ACTUALIZADORERP.DLL	ACTUALIZADORERP.DLL
<input type="checkbox"/>	PRUEBAS	693	APACACTA.DLL	APACACTA.DLL
<input type="checkbox"/>	PRUEBAS	361	APCUCASOUSO.DLL	APCUCASOUSO.DLL
<input type="checkbox"/>	PRUEBAS	232	APCUREPORTLIB.DLL	APCUREPORTLIB.DLL
<input type="checkbox"/>	PRUEBAS	1704	CACTZAFRAS.DLL	cactZafras.dll
<input type="checkbox"/>	PRUEBAS	2276	CCCTCAMBIORFC.DLL	CCCTCAMBIORFC.DLL
<input type="checkbox"/>	PRUEBAS	860	CCCTCLASIFICACIONCLIENTES.DLL	CCCTCLASIFICACIONCLIENTES.DLL
<input type="checkbox"/>	PRUEBAS	1697	CCCTCLASIFICACIONCLIENTES.DLL	CCCTCLASIFICACIONCLIENTES.DLL
<input type="checkbox"/>	PRUEBAS	2379	CCCTCLIENTES.DLL	CCCTCLIENTES.DLL
<input type="checkbox"/>	PRUEBAS	2225	CCCTCONCEPTOSEACTIVABLES.DLL	CCCTCONCEPTOSEACTIVABLES.DLL

### NOTA IMPORTANTE

Para actualizar las DLL base, es estrictamente necesario cerrar el sistema y volverlo a abrir.

### Actualizar DLL de funcionalidades con dependencias

Las dependencias se dan cuando una pantalla o forma **debe abrir otras formas que dependen de ella**, por ejemplo en el caso de las pantallas de Países, Estados, Municipios y Localidades.

Es importante marcar la dependencia, por que solo de esta forma, el actualizador verificará que estén todas las pantallas en su última versión al abrir la primer forma.

Para indicar la dependencia, es necesario marcar la casilla de SELECCIONADO que se encuentra en el Grid que esta abajo de los detalles de actualización, podemos ver eso en la siguiente imagen:

MainWindow

Actualizar Descargar DLL Atras Regresar a ésta versión Nuevo Guardar

Ensamblado

Nombre: ABCTPROVEEDORES.DLL Nombre funcional: abctProveedores.dll

Archivo: ABCTPROVEEDORES.DLL Seleccionar DLL Cargar Recursos Checksum: chv4oD2KDv4dFr9Vo9omQDD8gQy8fzgPkMPNSztjXQ4=

Comentario: abctProveedores.dll Ambiente: PRUEBAS Responsable:

☐ Es ensamblado base

Arrastrar un encabezado de columna y soltarlo aquí al lado del grupo de esa columna

SELECCIONADO	AMBIENTE	ENSAMBLADO_ID	ENSAMBLADO_NOMBRE	ENSAMBLADO_NOMBRE_FUNCIONAL
<input checked="" type="checkbox"/>	PRUEBAS	2031	ABCTPROVEEDORES.DLL	abctProveedores.dll
<input type="checkbox"/>	PRUEBAS	2314	ACCESO_DATOS.DLL	ACCESO_DATOS.DLL
<input type="checkbox"/>	PRUEBAS	694	ACTASREPORTSLIB.DLL	ACTASREPORTSLIB.DLL
<input type="checkbox"/>	PRUEBAS	1736	ACTUALIZADORERP.DLL	ACTUALIZADORERP.DLL
<input type="checkbox"/>	PRUEBAS	693	APACACTA.DLL	APACACTA.DLL
<input type="checkbox"/>	PRUEBAS	361	APCUCASOUSO.DLL	APCUCASOUSO.DLL
<input type="checkbox"/>	PRUEBAS	232	APCUREPORTLIB.DLL	APCUREPORTLIB.DLL
<input type="checkbox"/>	PRUEBAS	1704	CACTAZFRAS.DLL	cactZafras.dll
<input type="checkbox"/>	PRUEBAS	2276	CCCTCAMBIORFC.DLL	CCCTCAMBIORFC.DLL
<input type="checkbox"/>	PRUEBAS	860	CCCTCLASIFICACIONCLIENTES.DLL	CCCTCLASIFICACIONCLIENTES.DLL
<input type="checkbox"/>	PRUEBAS	1697	CCCTCLASIFICACIONCLIENTES.DLL	CCCTCLASIFICACIONCLIENTES.DLL
<input type="checkbox"/>	PRUEBAS	2379	CCCTCLIENTES.DLL	CCCTCLIENTES.DLL
<input type="checkbox"/>	PRUEBAS	2225	CCCTCONCERTOSEACTIVABLES.DLL	CCCTCONCERTOSEACTIVABLES.DLL

### NOTA IMPORTANTE

Para actualizar las DLL de formas, solo es necesario cerrar el formulario a actualizar, si dicho formulario necesitara de actualización de DLL Base, entonces es estrictamente reiniciar el sistema en el cliente.

### APIS SIZ (ApiContainer)

Como vimos anteriormente, dentro de la estructura de proyectos existe una solución llamada

ApiContainer, la cual se encarga de tener en un solo sitio todas los proyectos donde se codifican los Controllers o controladores que están al servicio de las peticiones de las pantallas.

Como dato importante, hay que tener en cuenta de que dichos proyectos por si solos no son capaces de funcionar al publicarlos en un servidor IIS, por lo que el proyecto al compilarlo, requiere de un proyecto Web que pueda montarlos y ponerlos al servicio de las peticiones, Dicho proyecto es llamado ApiCatZMX.

### ApiCatZMX

En este proyecto encontramos la estructura necesaria para funcionar como un WebApi, por lo que en caso de necesitar modificar el funcionamiento básico, dichos cambios se realizan en él, aquí encontramos los archivos de configuración y los controladores necesarios para un funcionamiento previo de las WebApi.

El funcionamiento anteriormente mencionado, trata de emular las consultas lambda que se hacen a Entity Framework tomando como base una expresión en String, la cual se utilizó en algunos catálogos que ya están aprobados hasta la fecha de elaboración del documento, es importante recalcar que es necesario cambiar este esquema al nuevo donde por cada catálogo se elabora un controller que le sirva.

Dentro de el proyecto ApiCatZMX, se encuentra el controller que necesitan las peticiones para redirigirse de acuerdo al ambiente, en caso de algún problema o cambio, dicho ajuste debe hacerse en este controller.

### ConfigController

Como ya se mencionó hay un controller donde se redireccionan las peticiones, este es el ConfigController, aquí se tienen diversos endpoints que acceden al archivo de configuración Web.Config para retornar al cliente que realiza la petición, la ruta correcta según el ambiente solicitado.

Aquí una imagen detallando el acceso a este redireccionamiento:



```
[HttpGet]
[Route("api/Config/ObtenerUrl")]
0 references | Jose Carlos Guerrero Karass, 257 days ago | 1 author, 1 change
public HttpResponseMessage ObtenerUrl(int ambiente, int empresa)
{
    string resultado = "";

    if (WebConfigurationManager.AppSettings["modo"] == "debug")
        resultado = WebConfigurationManager.AppSettings["debug"];
    else
        resultado = WebConfigurationManager.AppSettings["restapimodelo" + ambiente];

    var response = Request.CreateResponse(HttpStatusCode.OK, resultado);

    response.Content.Headers.Expires = DateTime.Now.AddSeconds(2);
    response.Content.Headers.LastModified = DateTime.Now;
    response.Headers.CacheControl = new System.Net.Http.Headers.CacheControlHeaderValue()
    {
        Public = true,
        MaxAge = new TimeSpan(0, 5, 0)
    };

    return response;
}
```

### ModeloController

En este controller encontramos la funcionalidad necesaria para realizar consultas utilizando el lambda por string, es importante recordar que esta forma de realizar peticiones ya es obsoleta y debe cambiarse a la versión que utiliza un controller por catálogo.

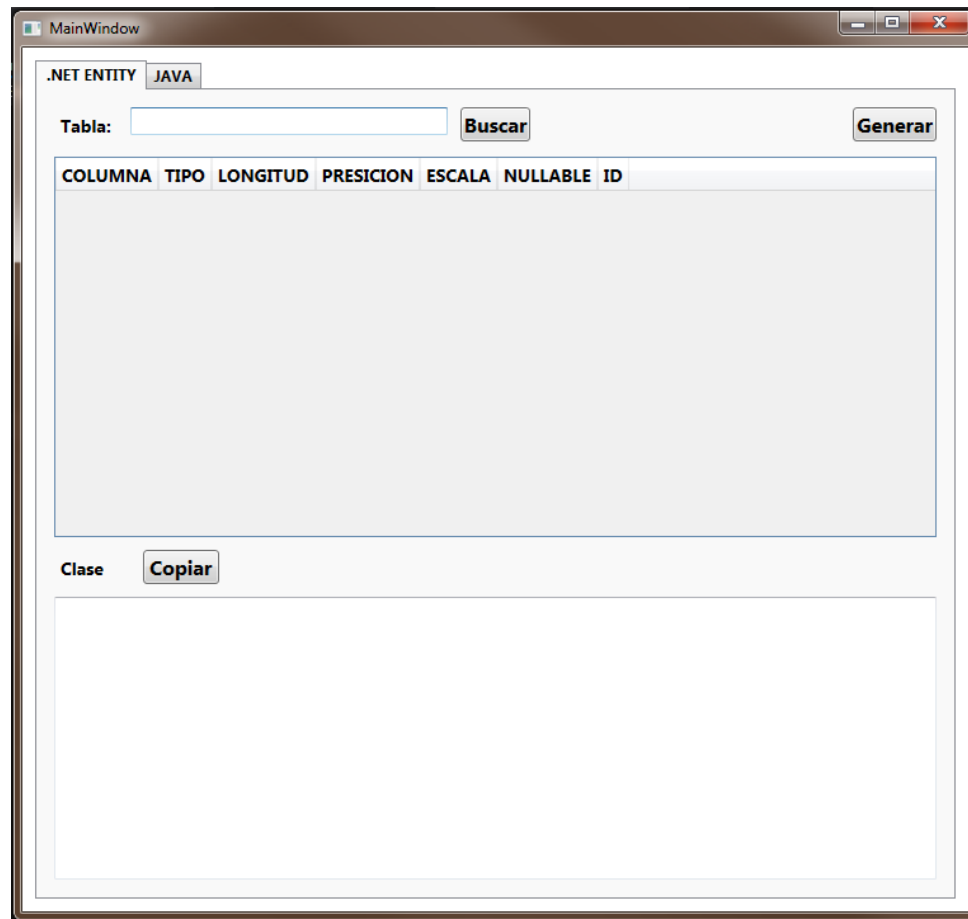
Solo en caso de necesitar hacer mantenimiento, aquí es donde se encuentran las funciones necesarias para realizar las peticiones y consultas a la base de datos, pero cuidando que las modificaciones sean completamente probadas antes de hacer alguna liberación, dadas las formas (catálogos) que quedaron funcionando con esta versión.

### Generador de código

El generador de código surgió ante la necesidad de poder crear las clases del modelo de forma automática, necesitando solamente especificar cuales columnas son de identificación y como formar el ID en caso de necesitarlo (estos detalles se explicaron anteriormente en [#Formación de ID](#)).

### Proceso para generar una clase

El proyecto a utilizar para esta tarea es el GENERADOR\_ENTIDADES, al abrirlo se mostrará una ventana como esta:



Escribimos el nombre de la tabla de la base de datos que deseamos obtener su clase posteriormente presionamos el botón Buscar (flecha roja) y nos presenta un resumen de las columnas con las que cuenta (flecha azul):

Tabla:

MG\_GRUPOS

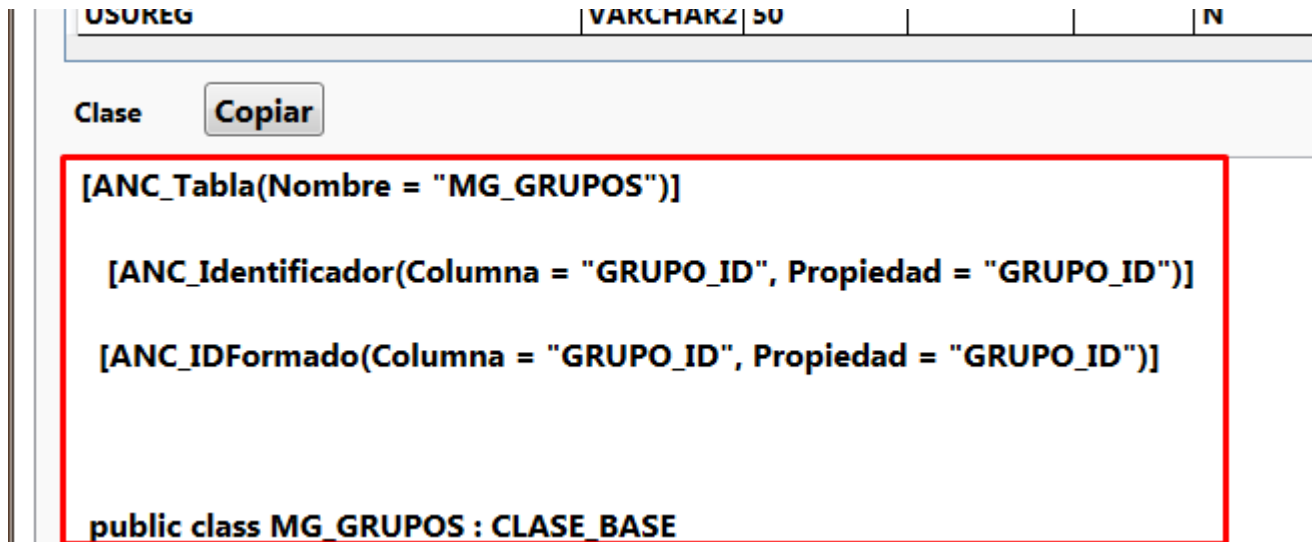
Buscar

Generar

COLUMNA	TIPO	LONGITUD	PRESICION	ESCALA	NULLABLE	ID
GRUPO_VENTAS_COMPANIAS	NUMBER	22	1	0	Y	11
GRUPO_CODIGO	NUMBER	22	3	0	N	12
GRUPO_ID	NUMBER	22	3	0	N	1
ESTATUS_ID	NUMBER	22	3	0	N	2
GRUPO_NOMBRE	VARCHAR2	320			N	3
GRUPO_NIVEL_ESTRUCT_VENDEDOR	NUMBER	22	3	0	Y	4
GRUPO_IMPTO_FLETE_COMBUSTIBLE	NUMBER	22	1	0	Y	5
ULTIMA_ACT	NUMBER	22	4	0	N	6
FECHA_CREACION	DATE	7			N	7
FECHA_MODIFICACION	DATE	7			N	8
USUARIO_CREACION	VARCHAR2	50			N	9
USUREG	VARCHAR2	50			N	10

Presionamos el botón Generar y nos muestra en la sección de abajo el código necesario

para mapear la tabla con una instancia de la clase recién generada.



### Proceso para integrar la clase al MODELO\_DATOS\_ADO

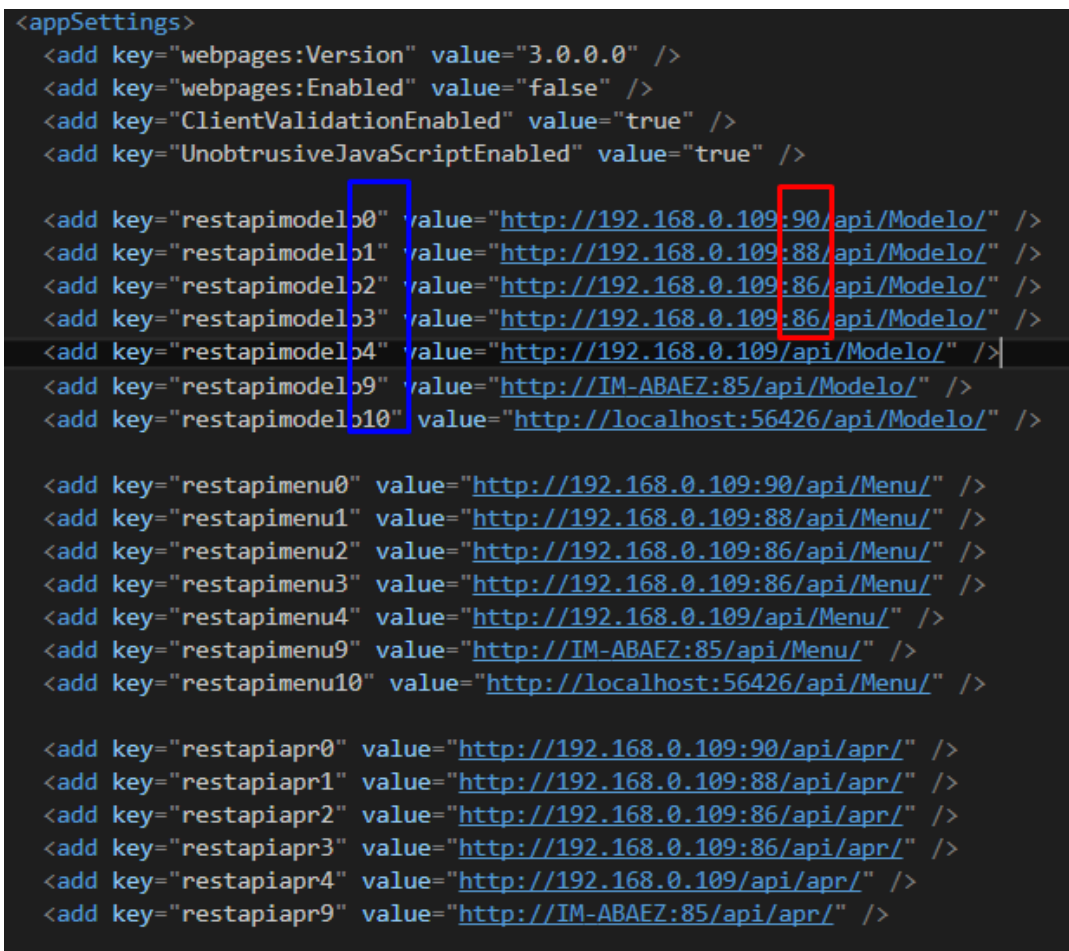
Una vez generada la clase, el siguiente paso es copiar el código generado anteriormente y pegarlo en una nueva clase dentro del proyecto MODELO\_DATOS\_ADO, teniendo cuidado con la siguiente secuencia de pasos.

- Crear la clase dentro de la carpeta que corresponda según su módulo.
- Pegar la clase y configurar (si es que tiene) su llave primaria o identificación (explicado anteriormente en [#Formación de ID](#))
- Compilar y generar la DLL
- Copiar dicha DLL en MG\_LIBRERIAS y en el WebApi correspondiente para tener disponible la nueva clase para todos los desarrolladores.
- Verificar la existencia de esta clase con el proyecto donde se necesita.

### Ambientes SIZ

#### Archivo config en WebApi para enrutar a los distintos ambientes según la sesión

Dentro del proyecto ApiCatZMX se encuentra el archivo Web.Config que contiene las URL que apuntan a los servidores IIS que contienen las WebApi para cada ambiente, en este caso por ser ambientes de desarrollo, pruebas y preproducción, todas las aplicaciones se encuentran alojadas en el mismo servidor IIS, por lo que lo único que varía es el puerto. Esto se puede ver en la siguiente imagen:



```
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />

  <add key="restapimodelo0" value="http://192.168.0.109:90/api/Modelo/" />
  <add key="restapimodelo1" value="http://192.168.0.109:88/api/Modelo/" />
  <add key="restapimodelo2" value="http://192.168.0.109:86/api/Modelo/" />
  <add key="restapimodelo3" value="http://192.168.0.109:86/api/Modelo/" />
  <add key="restapimodelo4" value="http://192.168.0.109/api/Modelo/" />
  <add key="restapimodelo9" value="http://IM-ABAEZ:85/api/Modelo/" />
  <add key="restapimodelo10" value="http://localhost:56426/api/Modelo/" />

  <add key="restapimenu0" value="http://192.168.0.109:90/api/Menu/" />
  <add key="restapimenu1" value="http://192.168.0.109:88/api/Menu/" />
  <add key="restapimenu2" value="http://192.168.0.109:86/api/Menu/" />
  <add key="restapimenu3" value="http://192.168.0.109:86/api/Menu/" />
  <add key="restapimenu4" value="http://192.168.0.109/api/Menu/" />
  <add key="restapimenu9" value="http://IM-ABAEZ:85/api/Menu/" />
  <add key="restapimenu10" value="http://localhost:56426/api/Menu/" />

  <add key="restapiapr0" value="http://192.168.0.109:90/api/apr/" />
  <add key="restapiapr1" value="http://192.168.0.109:88/api/apr/" />
  <add key="restapiapr2" value="http://192.168.0.109:86/api/apr/" />
  <add key="restapiapr3" value="http://192.168.0.109:86/api/apr/" />
  <add key="restapiapr4" value="http://192.168.0.109/api/apr/" />
  <add key="restapiapr9" value="http://IM-ABAEZ:85/api/apr/" />
</appSettings>
```

Como se ve en la imagen, en el recuadro rojo están los puertos que indican a que ambiente entrar y en el recuadro azul los distintos ambientes donde:

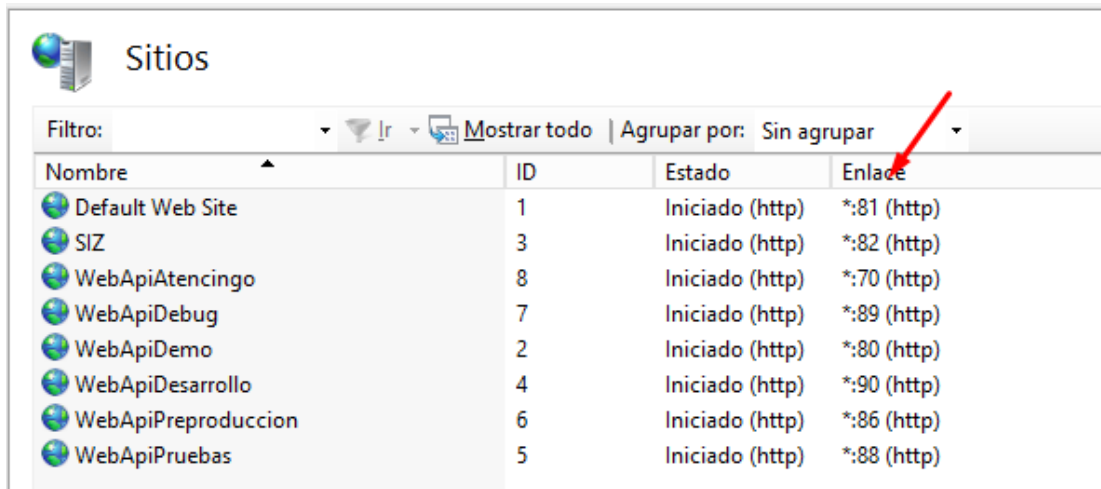
0=desarrollo

1=pruebas

2=preproducción

3=producción

Lo anteriormente explicado, lo podemos presenciar en el Servidor IIS la configuración de los puertos para cada ambiente:



Nombre	ID	Estado	Enlace
Default Web Site	1	Iniciado (http)	*:81 (http)
SIZ	3	Iniciado (http)	*:82 (http)
WebApiAtencingo	8	Iniciado (http)	*:70 (http)
WebApiDebug	7	Iniciado (http)	*:89 (http)
WebApiDemo	2	Iniciado (http)	*:80 (http)
WebApiDesarrollo	4	Iniciado (http)	*:90 (http)
WebApiPreproduccion	6	Iniciado (http)	*:86 (http)
WebApiPruebas	5	Iniciado (http)	*:88 (http)

Esta configuración posteriormente es recuperada por el ConfigController (explicado anteriormente) y el cliente es redireccionado a dicha ruta.

Aquí recordamos como el controller toma del web.Config la ruta utilizando el prefijo “restapimodelo” seguido del ambiente:

```
[HttpGet]
[Route("api/Config/ObtenerUrl")]
0 references | Jose Carlos Guerrero Karass, 257 days ago | 1 author, 1 change
public HttpResponseMessage ObtenerUrl(int ambiente, int empresa)
{
    string resultado = "";

    if (WebConfigurationManager.AppSettings["modo"] == "debug")
        resultado = WebConfigurationManager.AppSettings["debug"];
    else
        resultado = WebConfigurationManager.AppSettings["restapimodelo" + ambiente];

    var response = Request.CreateResponse(HttpStatusCode.OK, resultado);

    response.Content.Headers.Expires = DateTime.Now.AddSeconds(2);
    response.Content.Headers.LastModified = DateTime.Now;
    response.Headers.CacheControl = new System.Net.Http.Headers.CacheControlHeaderValue()
    {
        Public = true,
        MaxAge = new TimeSpan(0, 5, 0)
    };

    return response;
}
```

Seguido a esto y suponiendo que el ambiente = 0, del Web.Config recuperamos el valor siguiente enrutandonos al ambiente de desarrollo tomando el valor que se muestra en la siguiente imagen:

```
<add key="restapimodelo0" value="http://192.168.0.109:90/api/Modelo/" />
```

## Archivo config de aplicación escritorio para apuntar al WebApi

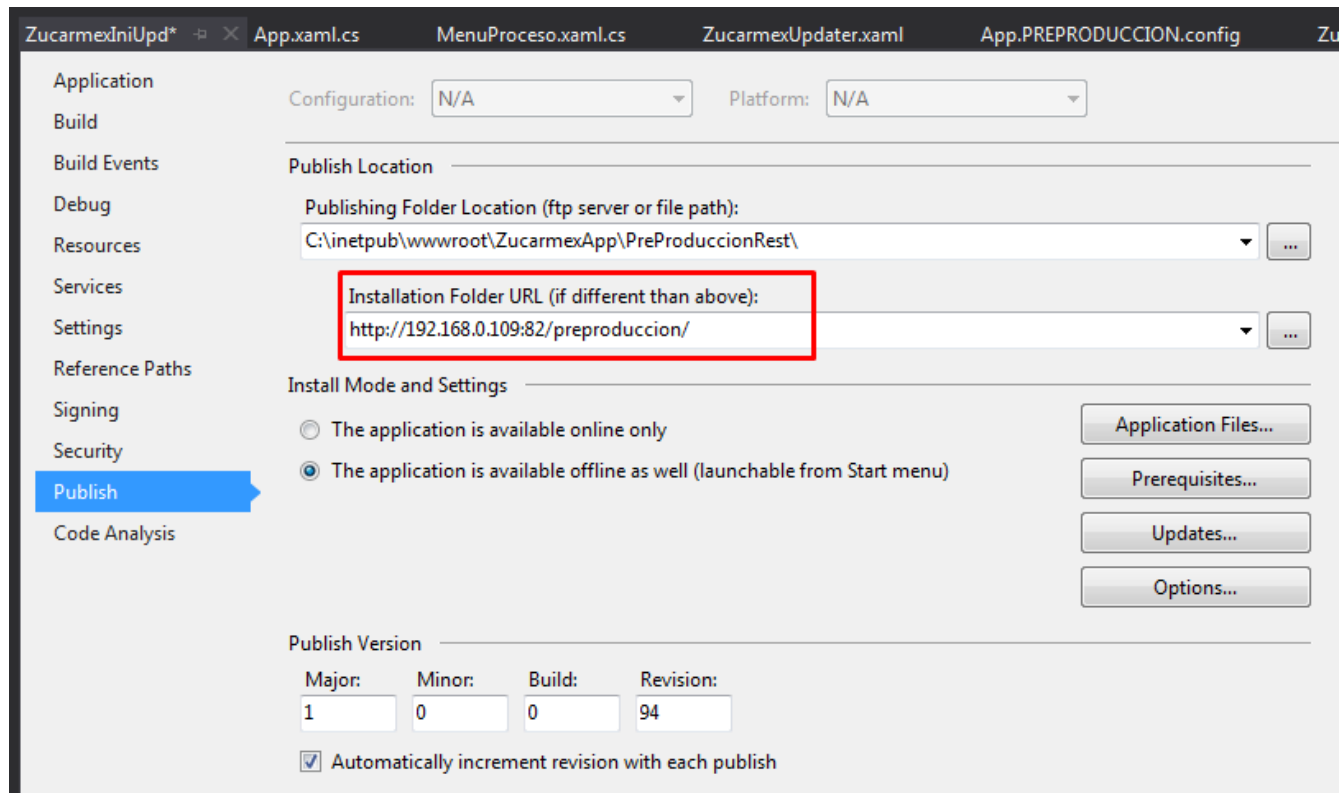
Solución=ZucarmexIniApp

### ZucarmexIniUpd (Configurar ClickOnce)

Proyecto=ZucarmexIniUpd

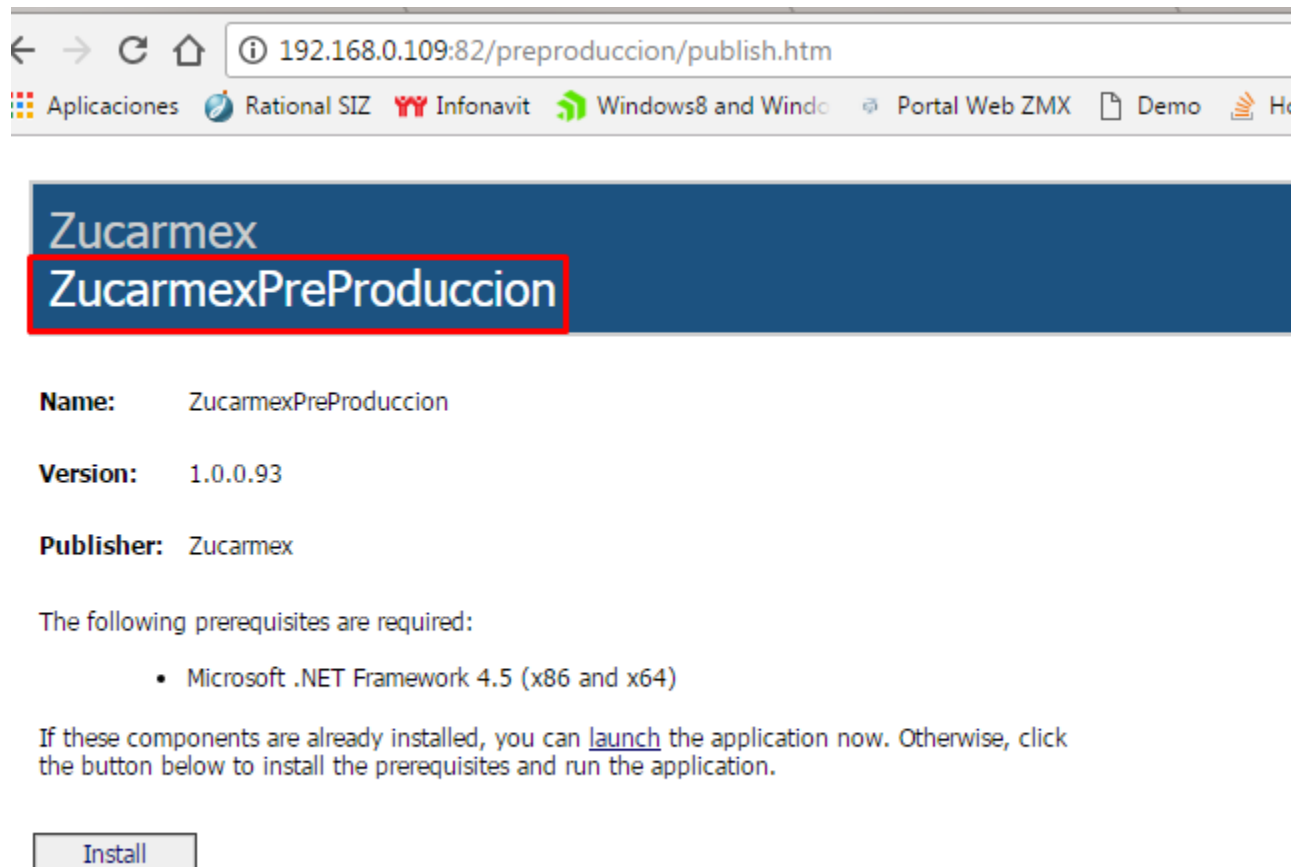
En el arranque de la aplicación, como se explico anteriormente, lo primero que se ejecuta es el ClickOnce para verificar nuevas versiones del Actualizador, dicha configuración ClickOnce se encuentra en el proyecto ZucarmexIniUpd en las propiedades del proyecto, pestaña Publish.

En caso de ser necesario y necesitar cambiar la ruta a la que apunta el ClickOnce, dentro de la sección de publish es necesario actualizar la ruta de instalación y colocar la URL que apunte a nuestra publicación ClickOnce de alojada en nuestro servidor IIS. Para hacer esto, en la siguiente imagen se muestran los detalles de donde cambiar la información:

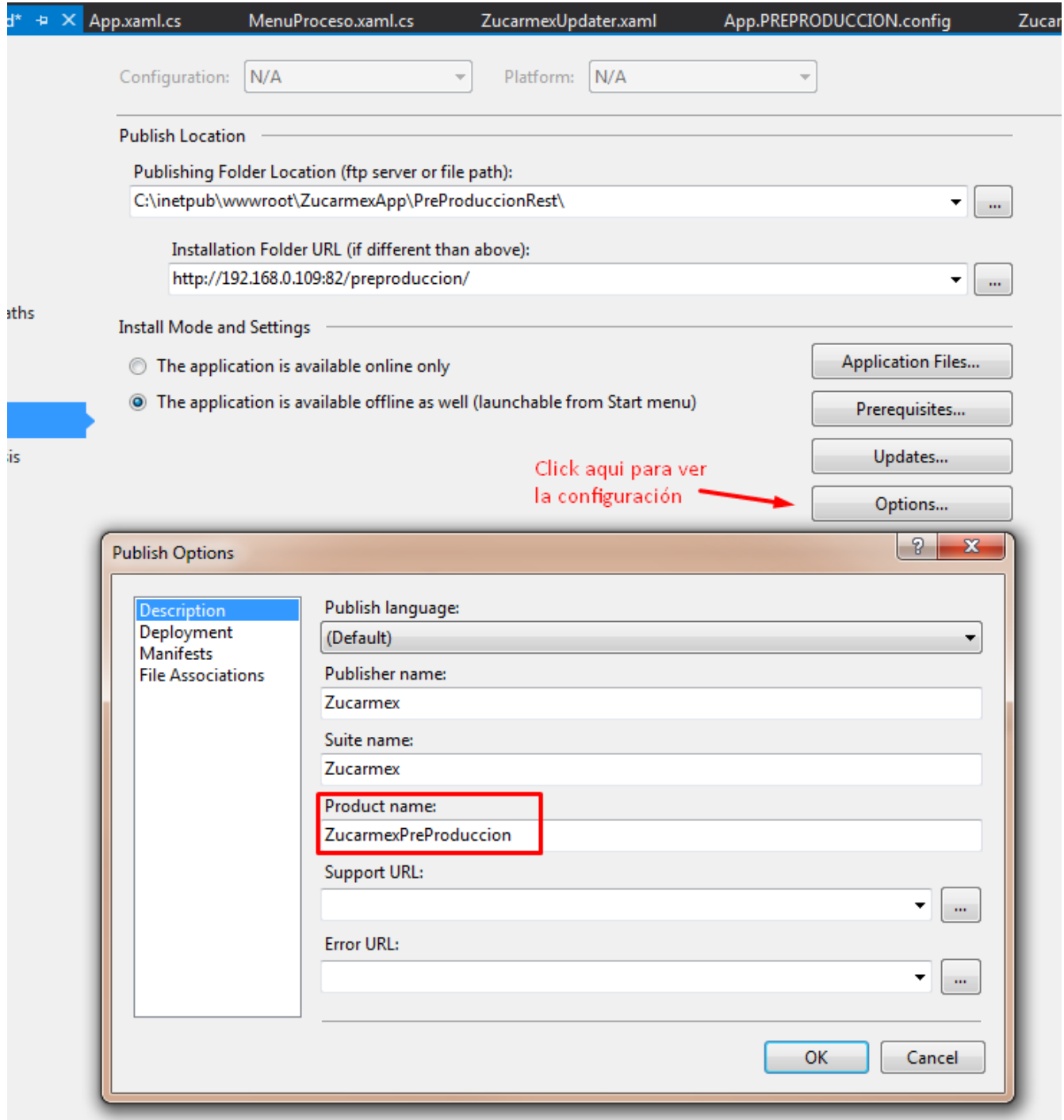


## DATO IMPORTANTE

Al realizar la publicación de la actualización del ClickOnce, es obligatorio que el nombre del producto sea idéntico al que ya se esta manejando, este nombre lo encontramos en la ruta de instalación cuando instalamos el cliente por primera vez, por ejemplo:



Debe coincidir exactamente con el nombre que pongamos en la siguiente configuración:

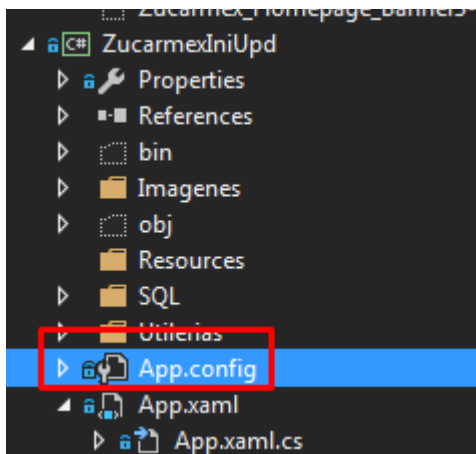


En caso de no respetar lo anterior descrito, cuando el cliente actualice, se le instalará nuevamente la aplicación, mostrando dos ejecutables en su escritorio, por lo que se puede confundir al usuario.



Una vez actualizado el Updater (SplashScreen que abre al inicio ver [#Updater](#)) este se conecta al servidor configurado en el App.Config que tenga el proyecto, éste puede ser modificable desde el proyecto ZucarmexIniUpd, en su archivo App.Config.

Un ejemplo es lo siguiente:



Y aquí una muestra de la configuración en el archivo:

```
</assemblyBinding>
</runtime>

<apiRest Ruta="http://192.168.0.109:86/api/Config/" Ambiente="2"></apiRest>

</configuration>
```

### [ZucarmexIniApp \(Configuración del proyecto SIZ\)](#)

Proyecto=ZucarmexIniApp

Por otro lado, para configurar el Exe que abre como tal el sistema SIZ, su archivo de configuración puede ser modificado desde el Actualizador (Herramienta para subir las dll al cliente ver [#Actualizador](#)), una muestra de tal archivo se ve en la siguiente imagen:

Filtros

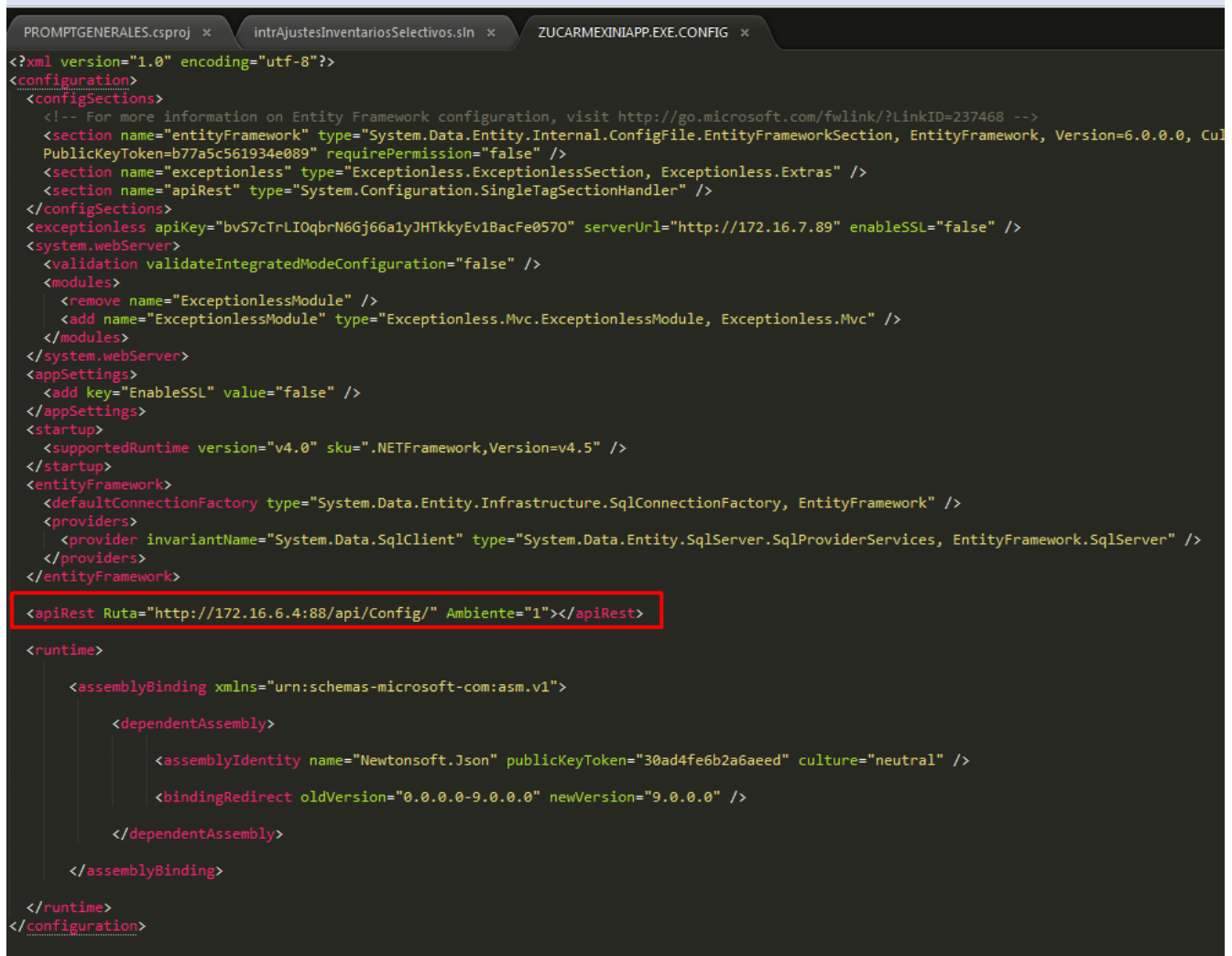
confi

Ensamblados

Agrupado por: ENSAMBLADO\_NOMBRE

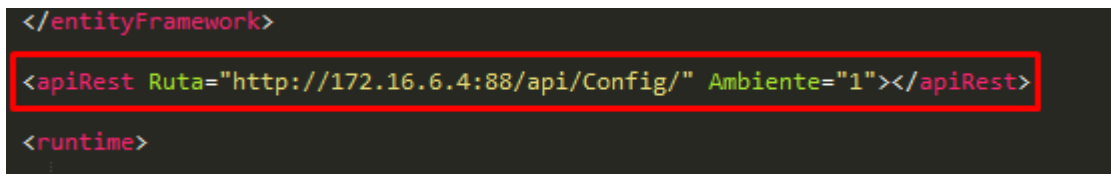
	AMBIENTE	ENSAMBLADO_ID	ENSAMBLADO_NOMBRE	ENSAMBLA
▲	FACTCONFIGACTURACIONINGENIOS.DLL			
▶	PRUEBAS	2414	FACTCONFIGACTURACIONINGENIOS.DLL	FACTCONFIC
▼	ZUCARMEXINIAPP.EXE.CONFIG			

y al descargarlo y examinarlo vemos lo siguiente:



```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
    <section name="exceptionless" type="Exceptionless.ExceptionlessSection, Exceptionless.Extras" />
    <section name="apiRest" type="System.Configuration.SingleTagSectionHandler" />
  </configSections>
  <exceptionless apiKey="bv57cTrLI0qbrN6Gj66a1yJHTkkyEv1BacFe0570" serverUrl="http://172.16.7.89" enableSSL="false" />
  <system.webServer>
    <validation validateIntegratedModeConfiguration="false" />
    <modules>
      <remove name="ExceptionlessModule" />
      <add name="ExceptionlessModule" type="Exceptionless.Mvc.ExceptionlessModule, Exceptionless.Mvc" />
    </modules>
  </system.webServer>
  <appSettings>
    <add key="EnableSSL" value="false" />
  </appSettings>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />
    <providers>
      <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
  <apiRest Ruta="http://172.16.6.4:88/api/Config/" Ambiente="1"></apiRest>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-9.0.0.0" newVersion="9.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

Aquí una vista mas cerca al recuadro en rojo:



```
</entityFramework>
<apiRest Ruta="http://172.16.6.4:88/api/Config/" Ambiente="1"></apiRest>
<runtime>
```

En caso de necesitar cambiar la ruta del servidor de las WebApi, esto se realiza en el archivo que se muestra en las imágenes anteriores y se vuelve a subir con el actualizador ([#Actualizador](#)) para hacer dicho cambio.