MANUAL DE DESARROLLO DE LOADERS ELLIPSE 8.4

HÉCTOR J. HERNÁNDEZ R.
Ingeniero de Sistemas
Hambings S.A.S

Versión 1.1.5
26 de Abril de 2016

HÉCTOR JOSÉ HERNÁNDEZ REINOZA
hernandezrhectorj@gmail.com

TABLA DE CONTENIDO

PALABRAS CLAVES

Ellipse, VSTO, Office, Excel, AddIn, WebService, Loader, Macro, SOAP

REQUISITOS PREVIOS:

1. Versión de Visual Studio compatible con VSTO para Office 2010
2. .Net FrameWork 4.0
3. Acceo a la Red Interna

PROCEDIMIENTO

1. CREAR NUEVO PROYECTO

Desplegamos el menú principal y seleccionamos "*[1] FILE > [2] New > [3] Project …*"

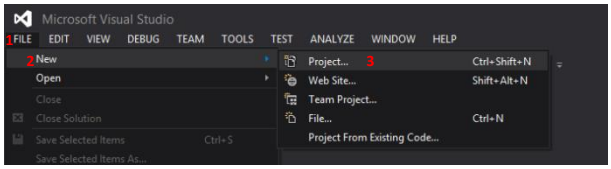

Imagen 1. Nuevo Proyecto: Creación de Nuevo proyecto

Seleccionamos la plantilla correspondiente a "*Excel 2010 AddIn*" en las opciones laterales, *"[4] Visual C# > [5] Office/SharePoint > [6] Office Add-Ins > [7] Excel 2010 AddIn".*

[8] Nos aseguramos que la versión del *.NET Framework* seleccionada sea la versión 4 para garantizar la compatibilidad con el Excel 2010 utilizado en Cerrejón.

[9] Ingresamos el Nombre del proyecto como ***EllipseProyectoExcelAddIn***, respetando la convención establecida para nombres de desarrollo.

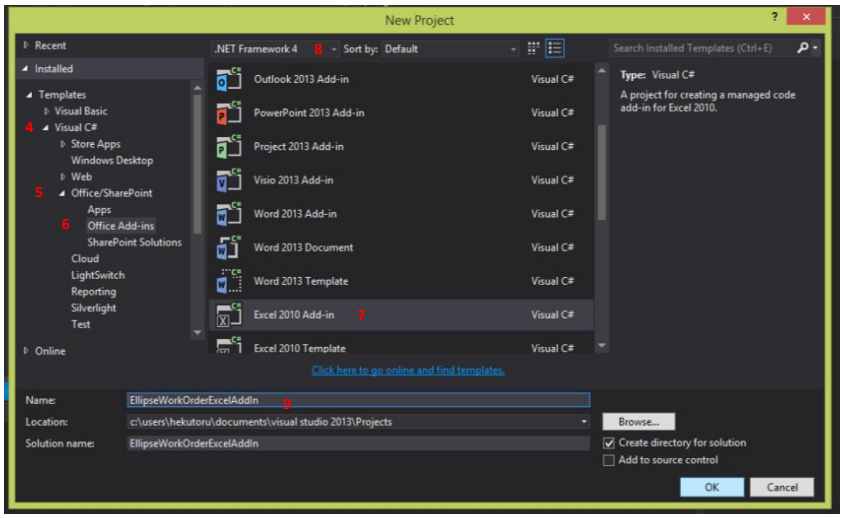

Imagen 2. Nuevo Proyecto: Selección de Tipo de Proyecto

2. AGREGAR REFERENCIAS WEB Y CUALQUIER OTRA REFERENCIA REQUERIDA

2.1 Agregar Web Service de Ellipse

[1] Desplegamos el menú de opciones del proyecto de la solución y seleccionamos *"[2] Add > [3] Service Reference …"*

HÉCTOR JOSÉ HERNÁNDEZ REINOZA
hernandezrhectorj@gmail.com

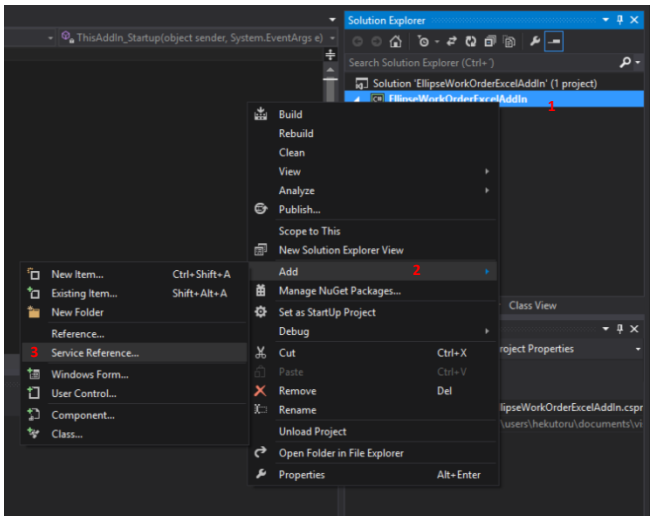

Imagen 3: Agregar Referencias Web: Agregar referencia a servicio

[4] Seleccionamos el botón *"Advanced…"*, para más opciones avanzadas y luego [5] el botón *"Add Web Reference…"*. [6] Copiamos la URL del servicio web que deseamos añadir en el cuadro de URL y [7] le damos clic en el botón de enviar. [8] Finalmente le damos nombre al servicio y seleccionamos [9] el botón de *"Add Reference"*

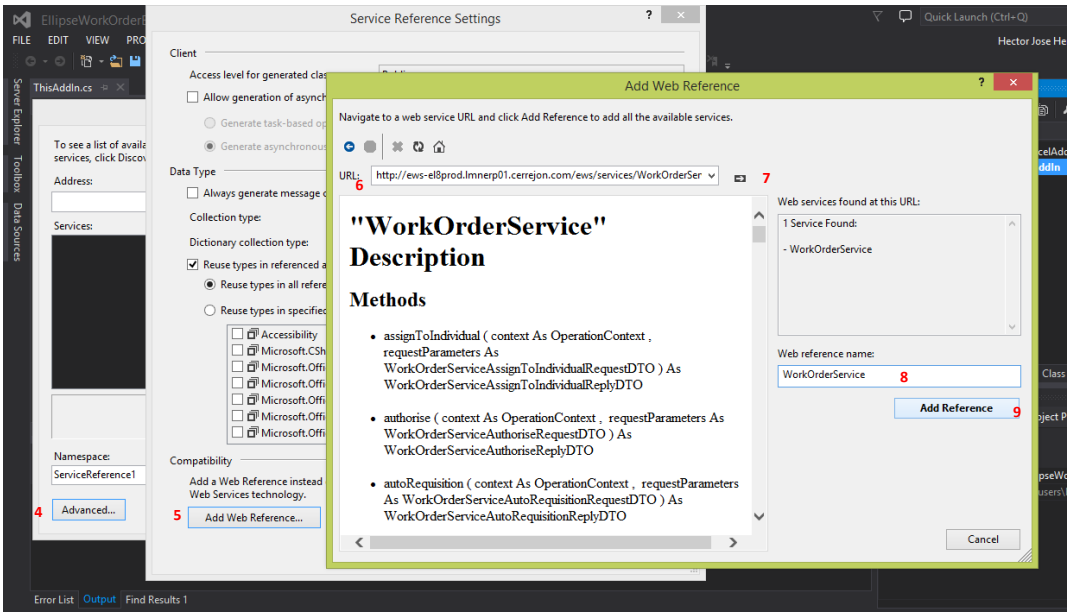

Imagen 4: Agregar Referencias Web: Especificar Web Service de Ellipse

El servicio de ScreenService no necesita ser referenciado. Si el desarrollo va a hacer uso de este servicio, puede invocarlo directamente del Proyecto EllipseCommonsClassLibrary descrito en el siguiente numeral.

Si no va a adicionar ningún servicio, deberá agregar de forma manual la referencia de System.Web.Services (este servicio se agrega automáticamente cuando se agrega cualquier servicio web)

2.2 Agregar referencia de Web Service de Ellipse en el Proyecto

Se añade el siguiente código a la clase principal del desarrollo del proyecto (ver numeral 5.4)

```
using EllipseProyectoExcelAddIn.ProyectoService;
```

3. AGREGAR REFERENCIA A PROYECTO COMMONS Y A ORACLE DATA READER

3.1 Vincular el proyecto EllipseCommonsClassLibrary

Desplegamos el [1] menú de opciones de la solución y seleccionamos *"[2] Add > [3] Existing Project…"*

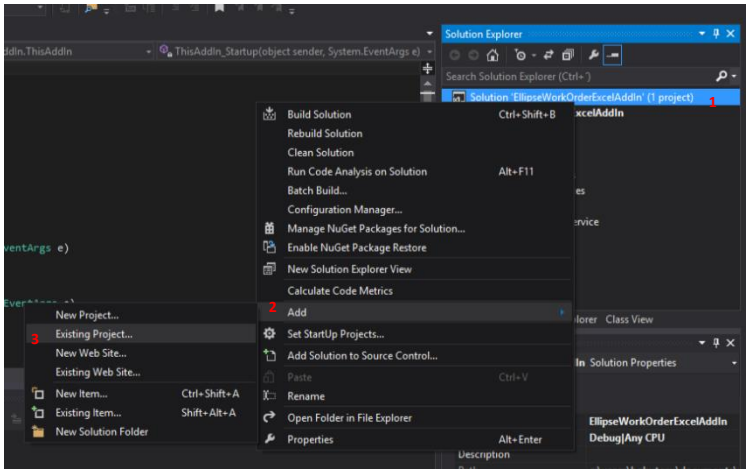

Imagen 5: Agregar EllipseCommonsClassLibrary: Agregar Proyecto EllipseCommonsClassLibrary

Buscamos y [4] seleccionamos el archivo del proyecto *"EllipseCommonsClassLibrary.csproj"* y finalmente [5] abrimos el proyecto para agregarlo a nuestro nuevo proyecto.

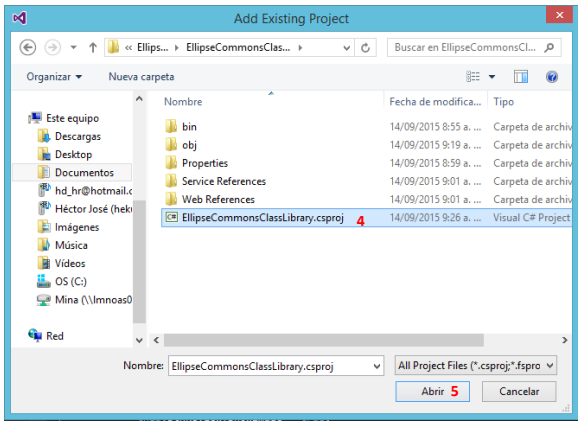

Imagen 6: Agregar *EllipseCommonsClassLibrary*: Seleccionar Proyecto *EllipseCommonsClassLibrary*

Repetir los pasos 3.1, 3.2 y 3.3 Para cada componente Commons que quiera añadir al proyecto.

3.2 Agregar Referencia del EllipseCommonsClassLibrary dentro de nuestro nuevo proyecto

Desplegamos el [1] menú de opciones de nuestro proyecto y seleccionamos *"[2] Add > [3] Reference…"*

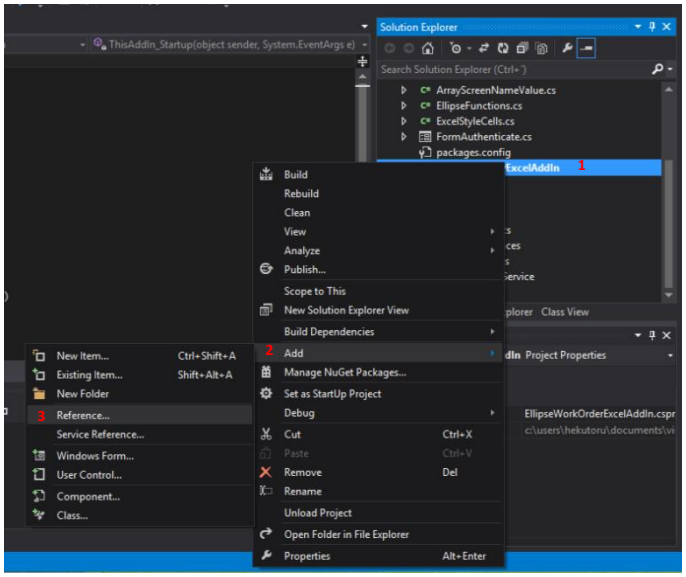

Imagen 7: Agregar EllipseCommonsClassLibrary: Agregar Referencia de EllipseCommonsClassLibrary

Buscamos en la opción de [4] Projects y [5] seleccionamos el proyecto que adicionamos en el numeral anterior y le damos al [6] botón OK

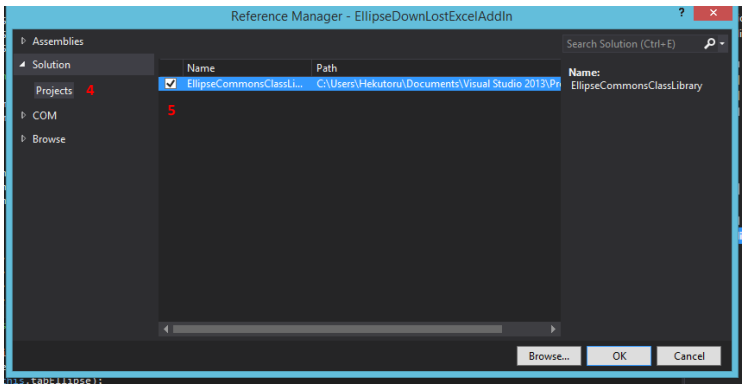

Imagen 8: Agregar EllipseCommonsClassLibrary: Seleccionar Referencia de EllipseCommonsClassLibrary

3.3 Agregar código de referencia de EllipseCommonsClassLibrary y del Screen Service

Se añade el siguiente código a la clase principal del desarrollo del proyecto (ver numeral 5.4)

```
using Screen = EllipseCommonsClassLibrary.ScreenService;
using EllipseCommonsClassLibrary;
```

3.4 Agregar la librería de uso del Oracle Data Manager

3.4.1 Instalar

Para instalar la librería Oracle ODP.NET Managed Drivers nos dirigimos al menú de *"[1] PROJECT > [2] Manage NuGet Packages…"*

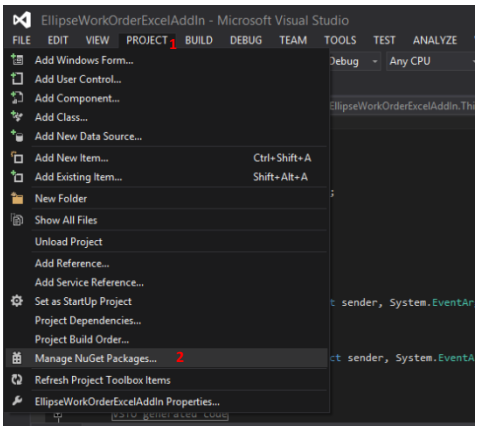

Imagen 9: Agregar Oracle Data Manager: Buscar la referencia Online

Buscamos [3] Online, y escribimos como criterio [4] *"Oracle ODP.NET"*. Seleccionamos el resultado que se muestra en la imagen (ver imagen 10) y le damos al [5] botón Instalar.

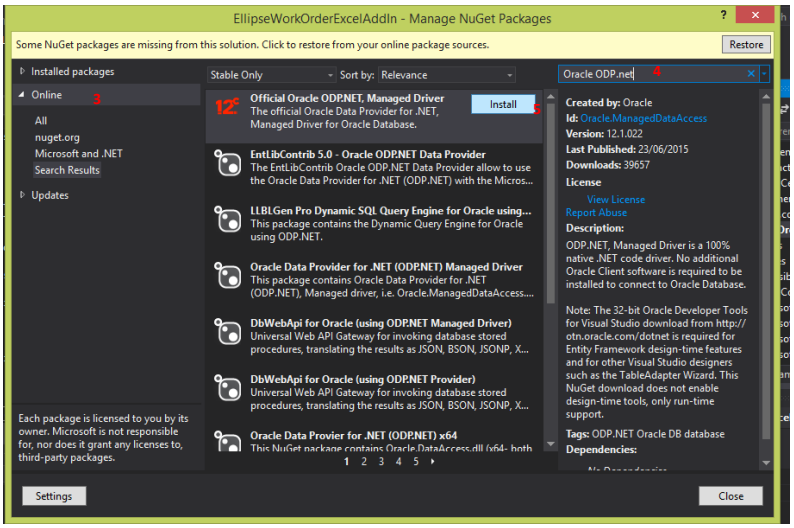

Imagen 10: Agregar Oracle Data Manager: Instalar la referencia

Finalmente, seguimos el proceso de aceptación de licencia y comprobamos al final que esté instalada la librería corroborando que el ícono verde de verificación esté sobre la librería instalada (ver imagen 11).

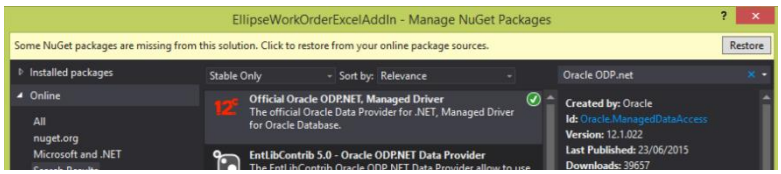

Imagen 11: Agregar Oracle Data Manager: Verificar el estado

3.4.2 Agregar código de referencia de Oracle Data Manager

Se añade el siguiente código a la clase principal del desarrollo del proyecto (ver numeral 5.4)

```
using Oracle.ManagedDataAccess.Client;
```

3.4.3 Agregar la referencia de acceso a las bases de datos TNS en app.config

Abres el archivo *app.config* dentro del proyecto

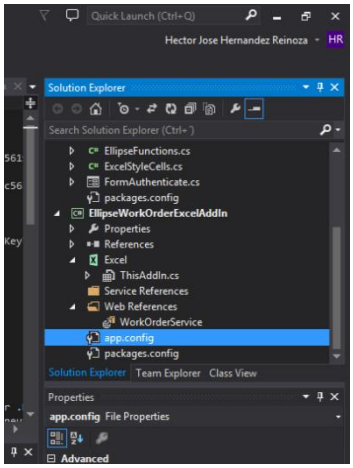

Imagen 12: Referenciar Oracle Data Manager: Abrir app.config

Y copias el código dentro de la etiqueta de <dataSources> de la información de gestión de acceso del Oracle.

```
<dataSource                                                    alias="EL8PROD"
descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=lmndbs09)(PORT=1521))(CONNECT_
DATA=(SERVICE_NAME=el8prod))) "/>
<dataSource                                                    alias="EL8TEST"
descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=lmndbs05)(PORT=1521))(CONNECT_
DATA=(SERVICE_NAME=el8test))) "/>
<dataSource                                                    alias="EL8DESA"
descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=lmndbs05)(PORT=1521))(CONNECT_
DATA=(SERVICE_NAME=el8desa))) "/>
```

En la siguiente imagen (Imagen 13) podemos observar el resultado

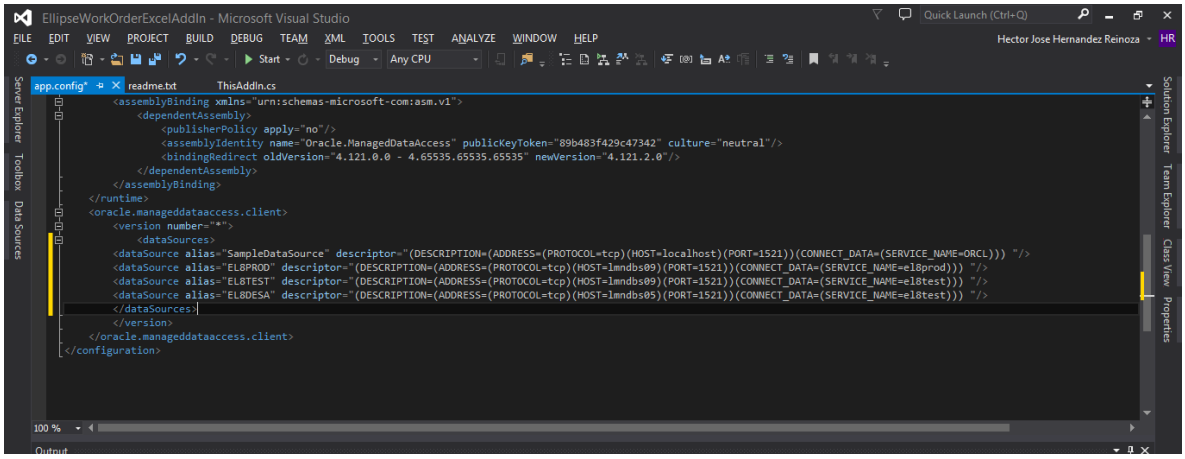

Imagen 13: Referencia Oracle Data Manager: Adicionar TNS

4. AGREGAR CABECERA DE CONSUMO SOAP

4.1 Agregar referencia a librería de consumo SOAP

Desplegamos el [1] menú de nuestro proyecto y seleccionamos la opción "*[2] Add… > Reference…"*

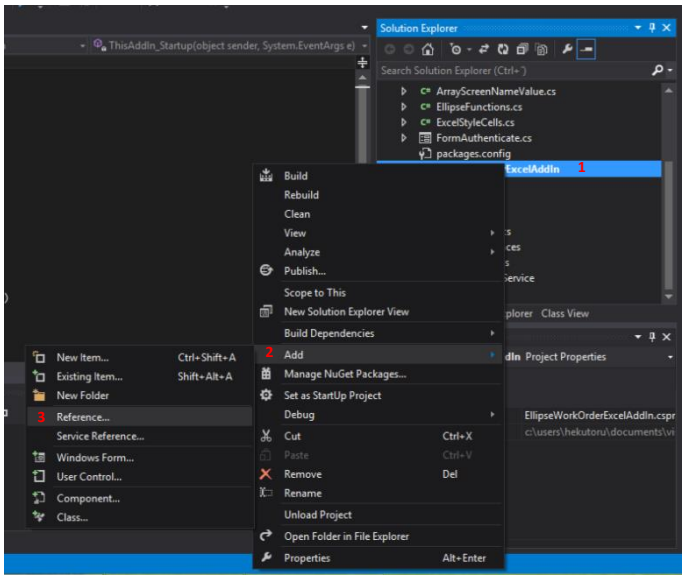

Imagen 14: Agregar Encabezado SOAP: Adicionar referencia

Buscamos la librería dentro del proyecto commons "...\EllipseCommonsExcelAddIn\Web References" y [4] seleccionamos el archivo System.Web.Services.Ellipse.dll. Finalmente le damos al [5] botón Add

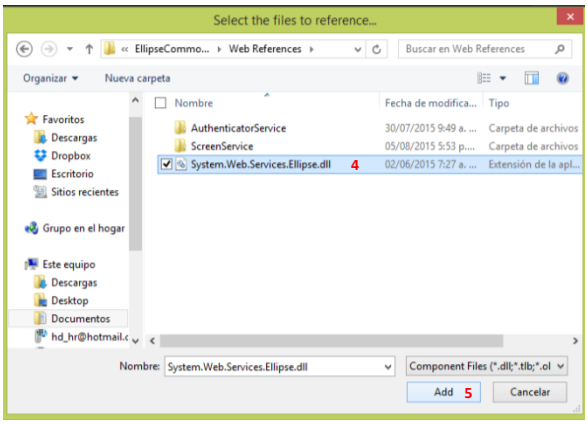

Imagen 15: Agregar Encabezado SOAP: Seleccionar librería

4.2 Agregar código de referencia de encabezado SOAP en el proyecto

Se añade el siguiente código a la clase principal del desarrollo del proyecto (ver numeral 5.4)

```
using System.Web.Services.Ellipse;
```

4.3 Adicionar referencia a app.config

Adicionamos el siguiente código al archivo app.config dentro de las etiquetas de <configuration></configuration>

```
<system.web>
        <webServices>
          <soapExtensionTypes>
               <add        type="System.Web.Services.Ellipse.AuthenticationExtension,
System.Web.Services.Ellipse" priority="1"/>
          </soapExtensionTypes>
        </webServices>
</system.web>
```

El resultado es como se muestra en la siguiente imagen (ver imagen 16)

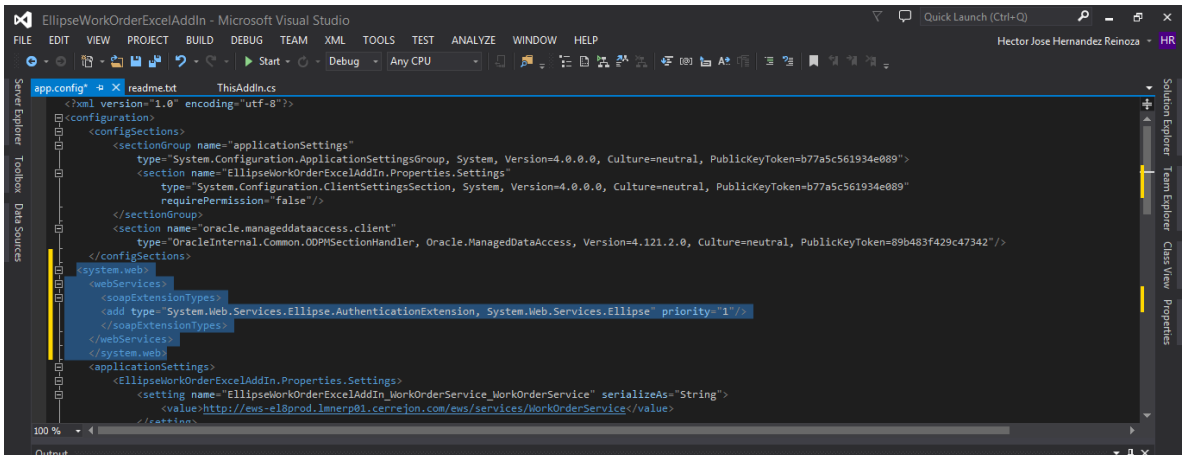

Imagen 16: Agregar Encabezado SOAP: Adicionar información a app.config

5. CREAR CLASE PRINCIPAL

5.1 Crear Ribbon

Desplegamos el [1] menú de opciones del proyecto, y seleccionamos *"[2] Add > [3] New Item…"*

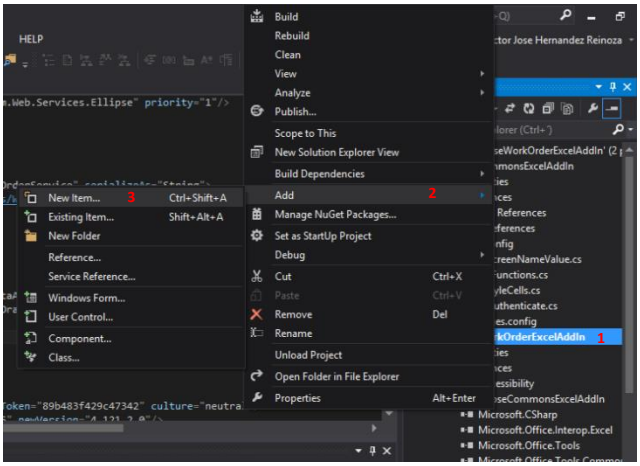

Imagen 17: Crear Ribbon: Nuevo Item

En el cuadro desplegado seleccionamos el elemento de la categoría "*[4] Visual C# Items"* y de tipo *"[5] Ribbon (Visual Designer)"*. Le damos nombre al Ribbon a adicionar [6] según la convención establecida *"RibbonEllipse.cs"* y le damos al "*[7] botón Add*"

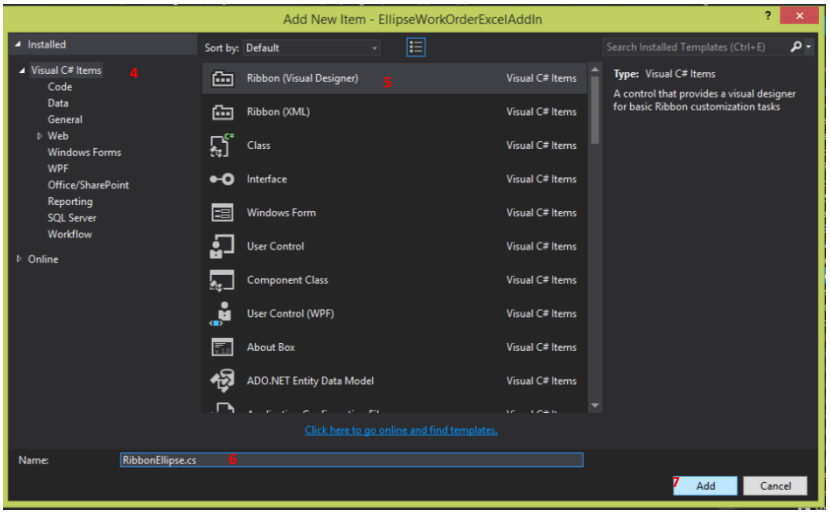

Imagen 18: Crear Ribbon: Seleccionar tipo de Item

5.2 Renombrar elementos del Ribbon

Seleccionamos la pestaña del Ribbon [1] y cambiamos el texto de la etiqueta [2] a "ELLIPSE 8" y también cambiamos el nombre [3] con el que vamos a hacer referencia en nuestro código a "tabEllipse" como se ve en la siguiente imagen (ver imagen 19)

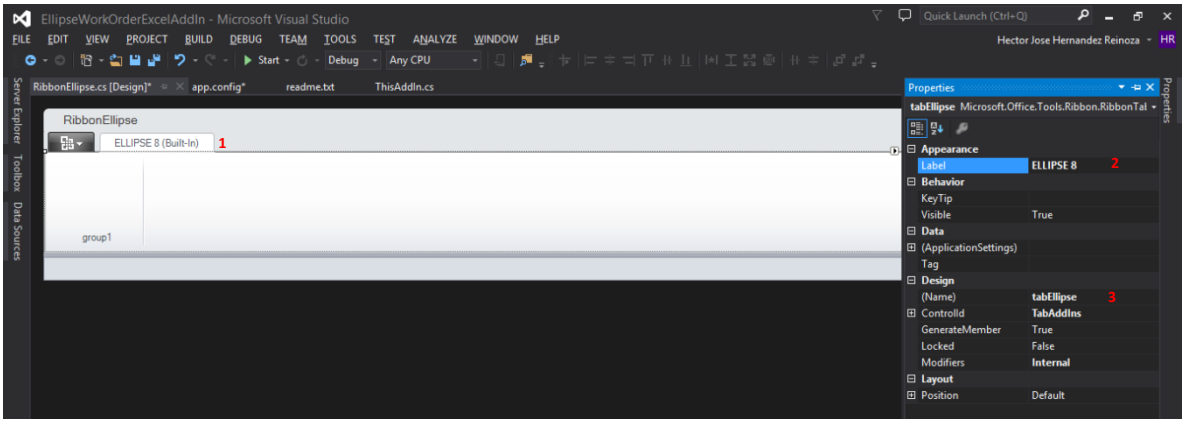

Imagen 19: Renombrar Ribbon: Renombrar Tab del Ribbon

Repetimos el mismo proceso para el grupo del Ribbon. Seleccionamos el grupo del Ribbon [4] y cambiamos el texto de la etiqueta [5] a "Proyecto v1.0.0" y también cambiamos el nombre [6] con el que vamos a hacer referencia en nuestro código a "grpProyecto" como se ve en la siguiente imagen (ver imagen 20)

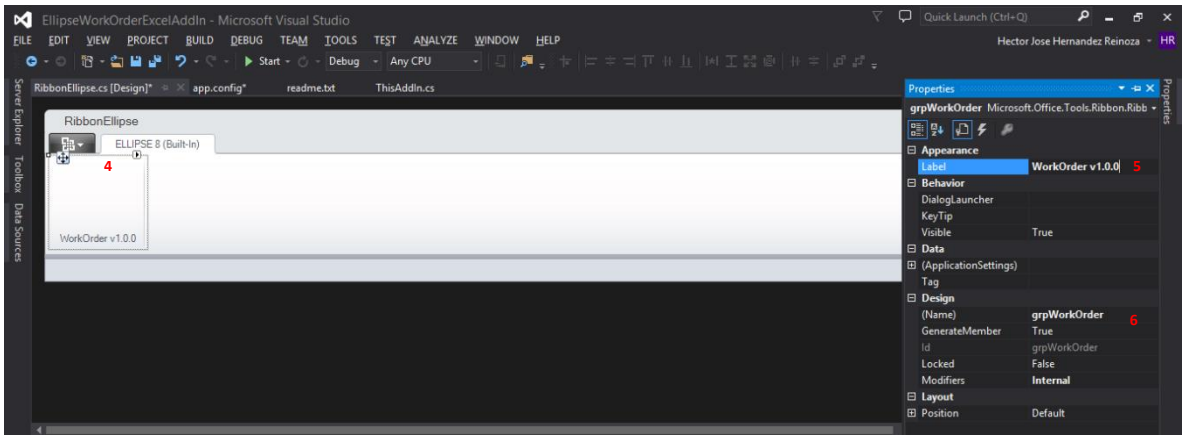

Imagen 20: Renombrar Ribbon: Renombrar Grupo del Ribbon

5.3 Agregar controles al Ribbon

Adicionamos un elemento de tipo dropdown (lista desplegable) desde la caja de herramientas [1] (*Toolbox*) en la barra lateral derecha y seleccionamos en la categoría de *Office Ribbon Controls [2]* el elemento de tipo *DropDown [3].*

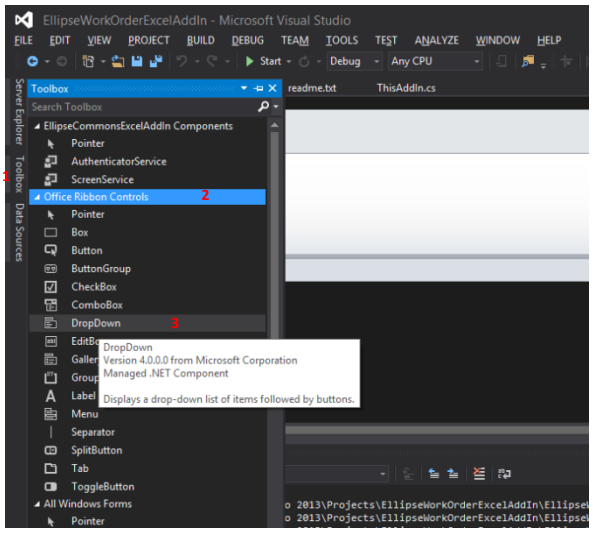

Imagen 21: Agregar controles a Ribbon: Agregar DropDown de Entorno

Adicionamos el control en el Ribbon de modo que quede como se ve en la siguiente imagen [4] (ver imagen 22).

Cambiamos el texto del control [5] a “Env.” y el nombre del control [6] “drpEnviroment”. Finalmente damos clic a las opciones de colección del control [7] para adicionar los diferentes entornos de Ellipse.

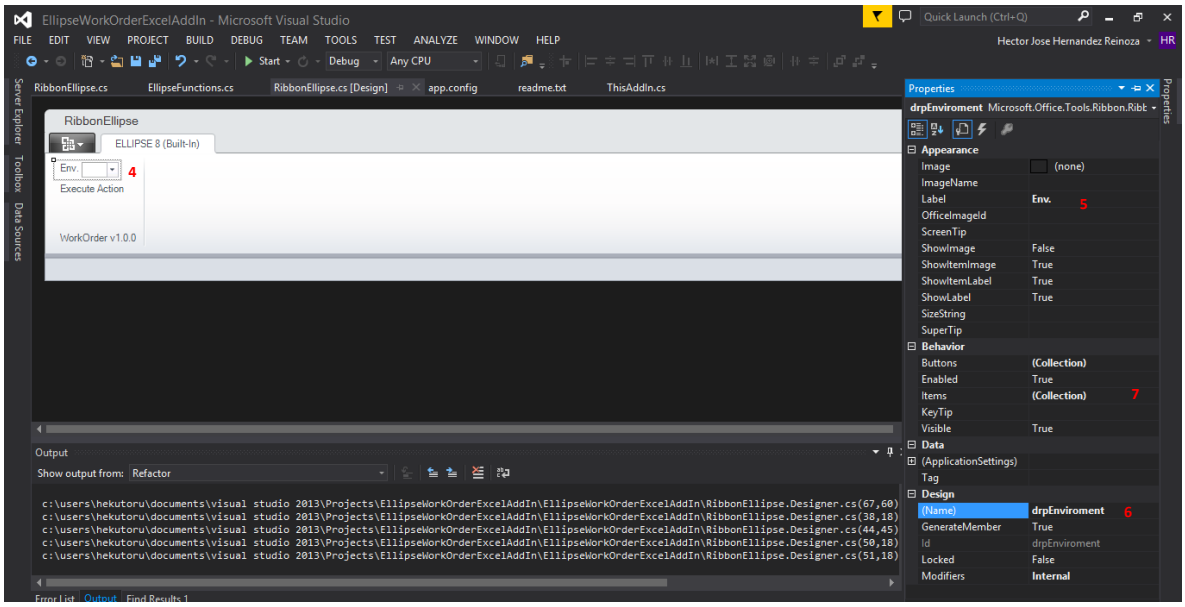

Imagen 22: Agregar controles a Ribbon: Configurar DropDown de Entorno

~~Damos clic en el botón Add [8] y ponemos el texto del entorno [9]. Repetimos este paso hasta que tengamos los entornos deseados (Productivo, Test y Desarrollo) y finalizamos dándole al botón OK [10].~~ *Los servidores se agregarán de forma automática en la función de carga del Ribbon en el paso 6.2*

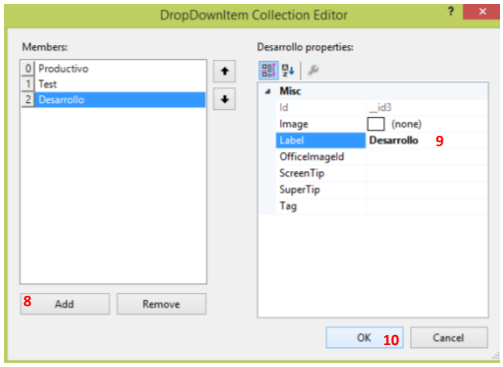

Imagen23: Agregar controles a Ribbon: Configurar Colección de DropDown de Entorno

5.4 Agregar referencias de librerías en la clase RibbonEllipse

Al ser RibbonEllipse.cs nuestra clase principal, debemos adicionar en esta todas las referencias de las librerías, clases, entre otras cosas, que estuvimos configurando durante todo el proyecto (ver imagen 24).

```
using Screen = EllipseCommonsClassLibrary.ScreenService;
using EllipseProyectoExcelAddIn.ProyectoService;
using EllipseCommonsClassLibrary;
using Oracle.ManagedDataAccess.Client;
using System.Web.Services.Ellipse;
using Excel = Microsoft.Office.Interop.Excel;
using System.Windows.Forms;
```

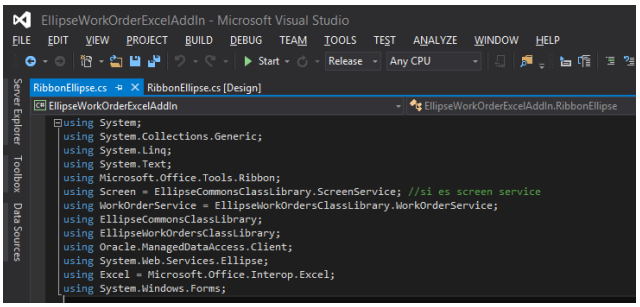

Imagen 24: Agregar referencias en encabezado de librerías

A continuación una breve descripción de cada una de las librerías adicionadas:

using Screen = EllipseCommonsClassLibrary.ScreenService;
Corresponde a la referencia para usar los ScreenService (MSO) de Ellipse. No requerida si se va a trabajar exclusivamente con los MSE.

using EllipseProyectoExcelAddIn.ProyectoService;
Corresponde a la referencia para usar el Servicio del Proyecto que creaste en el numeral 2. Hay programas que requieren el funcionamiento de más de un servicio, para estos casos tendrás varias líneas haciendo cada línea referencia al servicio especificado.

using EllipseCommonsClassLibrary;
Corresponde a la referencia para usar las clases del proyecto común compartido por todos los loaders agregado en el numeral 3.1.

using Oracle.ManagedDataAccess.Client;
Corresponde a la referencia para usar las opciones de consulta a la base de datos mediante el ODP agregado en el numeral 3.4

using System.Web.Services.Ellipse;
Corresponde a la librería de consumo SOAP para la autenticación y consumo de servicios de Ellipse agregada en el numeral 4.

using Excel = Microsoft.Office.Interop.Excel;
Corresponde a la librería de uso de opciones y acciones de Excel

using System.Windows.Forms;
Corresponde a la librería de uso de opciones y acciones de controles de Windows Form

6. AGREGAR VARIABLES DE GESTIÓN EN LA CLASE PRINCIPAL

6.1 Variables comunes de gestión

Adicionamos las siguientes variables que servirán de uso para el desarrollo del proyecto, y que dependen de la clase commons (ver imagen 25).

```
ExcelStyleCells _cells;
EllipseFunctions _eFunctions = new EllipseFunctions();
FormAuthenticate _frmAuth = new FormAuthenticate();
Excel.Application _excelApp;
string SheetName01 = "ProjectName";
```

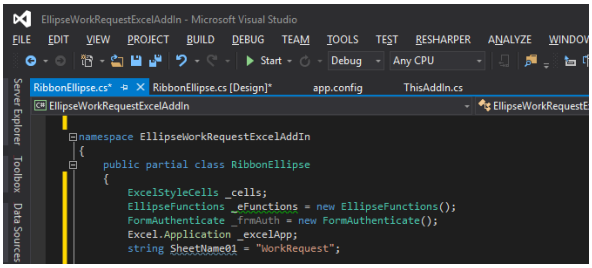

Imagen 25: Agregar variables a clase principal: Agregar variables comunes

_cells: Corresponde a la clase creada en commons para el manejo de celdas, estilos, entre otras opciones, de Excel; facilitando así el desarrollo del proyecto.

_eFunctions: Corresponde a la clase creada en commons para el manejo de funciones de Ellipse como son la ejecución de los Screen Services, solicitud de consultas en la base de datos de Ellipse, entre otras opciones.

_frmAuth: Corresponde a la clase creada en commons para el manejo de autenticación en Ellipse.

_excelApp: Corresponde a la variable que hará referencia a la instancia de Excel que se está ejecutando

SheetName01: Corresponde a la variable que hará referencia al nombre de la hoja de Excel para validación de ejecución de las acciones del Loader. Puede ser usada otra variable según se desee.

6.2 Valores iniciales de la clase principal (RibbonEllipse_Load)

Adicionamos el siguiente código dentro del método de iniciación del RibbonEllipse (ver imagen 26)

```
_excelApp = Globals.ThisAddIn.Application;

_eFunctions.DebugQueries = false;
_eFunctions.DebugErrors = false;
_eFunctions.DebugWarnings = false;
var enviroments = EnviromentConstants.GetEnviromentList();
foreach (var env in enviroments)
{
        var item = Factory.CreateRibbonDropDownItem();
        item.Label = env;
```

```
        drpEnviroment.Items.Add(item);
    }
```

```
private void RibbonEllipse_Load(object sender, RibbonUIEventArgs e)
{
    _excelApp = Globals.ThisAddIn.Application;

    _eFunctions.DebugQueries = false;
    _eFunctions.DebugErrors = false;
    _eFunctions.DebugWarnings = false;
    var enviroments = EnviromentConstants.GetEnviromentList();
    foreach (var env in enviroments)
    {
        var item = Factory.CreateRibbonDropDownItem();
        item.Label = env;
        drpEnviroment.Items.Add(item);
    }
}
```

Imagen 26: Agregar variables clase principal: Código de iniciación

~~_cells: Esta clase se puede iniciar de forma que haga siempre referencia a la hoja activa, o puede crearse especificando siempre una misma hoja (ver sobrecarga de constructor)~~

Se omite la inicialización de la clase ExcelStyleCells (_cells) en el constructor para garantizar compatibilidad con versiones de Office 2013 y superiores.

_eFunctions.DebugQueries/.DebugWarnings /.DebugErrors: Corresponden a las variables para hacer debuggin y seguimiento dentro del código.

7. OBSERVACIONES ADICIONALES DE DESARROLLO

A continuación se dejan algunas observaciones adicionales para el desarrollo

7.1 Autenticación

```
//Se realiza la autenticación
_frmAuth.StartPosition = FormStartPosition.CenterScreen;
_frmAuth.SelectedEnviroment = drpEnviroment.SelectedItem.Label;

if (_frmAuth.ShowDialog() != DialogResult.OK) return;// si no se autentica, se cancela el proceso
//se adiciona el código de la acción a realizar en el proceso
```

Ver imagen 27

7.2 Uso del servicio seleccionado

7.2.1 Screen Service

```
//Instanciar el Contexto de Operación
var opSheet = new OperationContext
{
          district = _frmAuth.EllipseDsct,
          position = _frmAuth.EllipsePost,
          maxInstances = 100,
          maxInstancesSpecified = true,
          returnWarnings = _eFunctions.DebugWarnings,
          returnWarningsSpecified = true
};

//Instanciar el SOAP
ClientConversation.authenticate(_frmAuth.EllipseUser, _frmAuth.EllipsePswd);

//Proceso del screen
var proxySheet = new Screen.ScreenService();
var requestSheet = new Screen.ScreenSubmitRequestDTO();


//Se define el ambiente del Dropdown de Enviroment
var urlService = _eFunctions.GetServicesUrl(drpEnviroment.SelectedItem.Label);
proxySheet.Url = urlService + "/ScreenService";
```

```
//ejecutamos el programa y leemos la respuesta si se necesita en el replySheet
var replySheet = proxySheet.executeScreen(opSheet, "MSO435");

//validamos el ingreso al programa
if (replySheet.mapName != "MSM435A" || _excelApp.ActiveWorkbook.ActiveSheet.Name != SheetName01)
          return "ERROR:" + "No se pudo establecer comunicación con el servicio";

//arreglo para los campos del screen
var arrayFields = new ArrayScreenNameValue();

//se adicionan los campos que se vayan a enviar
arrayFields.Add("OPTION1I", "1");
arrayFields.Add("MODEL_CODE1I", modelCode);
arrayFields.Add("STAT_DATE1I", modelDate);
arrayFields.Add("SHIFT1I", modelShift);

//se envía el screen
requestSheet.screenFields = arrayFields.ToArray();
requestSheet.screenKey = "1";

replySheet = proxySheet.submit(opSheet, requestSheet);

//manejo de opciones para el tipo de respuesta
if (replySheet != null
          && replySheet.mapName == "MSM435B")
          && !_eFunctions.CheckReplyError(replySheet)
          && !_eFunctions.CheckReplyWarning(replySheet)
          && replySheet.functionKeys = "XMIT-Confirm")
{
          //si necesitas obtener los campso del reply y trabajar con ellos
          var replyFields = new ArrayScreenNameValue(replySheet.screenFields);
          bool isEmpty = replyFields.getField("WO_PROJ1I" + i).value.Equals("");
}
```

Ver imagen 27 y 28

7.2.2 Otros servicios (MSE)

```
var opSheet = new WorkOrderService.OperationContext();
opSheet.district = _frmAuth.EllipseDsct;
opSheet.position = _frmAuth.EllipsePost;
opSheet.maxInstances = 100;
opSheet.maxInstancesSpecified = true;
opSheet.returnWarnings = _eFunctions.DebugWarnings;
opSheet.returnWarningsSpecified = true;

var proxyWo = new WorkOrderService.WorkOrderService();//ejecuta las acciones del servicio
var requestWo = new WorkOrderServiceCreateRequestDTO();

proxyWo.Url = urlService + "/WorkOrder";

//Se cargan los parámetros de  la solicitud
requestWo.districtCode = "ICOR";
requestWo.workGroup = "MTOLOC";
//se envía la acción
var replyWO = proxyWo.modify(opContext, requestWo);

//se analiza el reply según la acción
//se hacen las acciones pertinentes
```

Ver imagen 29

7.3 Ejecución de Querys

```
//EJECUCIÓN DE QUERIES
//Estableces el servidor de base de datos con alguna de las siguientes tres opciones
_eFunctions.SetDBSettings(urlService);
_eFunctions.SetDBSettings(dbname, dbuser, dbpass);
_eFunctions.SetDBSettings(dbname, dbuser, dbpass, dblink, dbreference);

var dataReader = ef.GetQueryResult(sqlQuery);
if (dataReader == null || dataReader.IsClosed || !dataReader.HasRows) return null;
while (dataReader.Read())
{
          var workOrder = dataReader["WORK_ORDER"].ToString().Trim();
}
```

Ver imagen 30

7.4 Control de Errores y Debugging

```
//CONTROL DE ERRORES Y DEBUGGING
//debugging de queries
if(_eFunctions.DebugQueries)
          _cells.getCell("L1").Value = sqlQuery;

//GESTIÓN DE EXCEPCIONES
try
{
          if (_excelApp.ActiveWorkbook.ActiveSheet.Name != SheetName01 + LabourSheetTypeConstants.Default)
                    throw new Exception("La hoja seleccionada no coincide con el modelo requerido");

          if (drpEnviroment.Label == null || drpEnviroment.Label.Equals(""))
                    throw new ArgumentException("Seleccione un ambiente válido");
}
catch (Exception ex)
{
          //Despliega una alerta de error o realiza la acción según se requiera
          MessageBox.Show(ex.Message, @"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
          //Genera un log del error. Si _eFunctions.DebugErrors es true, siempre desplegará un mensaje de error
          ErrorLogger.LogError("RibbonEllipse:startLabourCostLoad()", "\n\rMessage:" + ex.Message + "\n\rSource:" +
ex.Source + "\n\rStackTrace:" + ex.StackTrace, _eFunctions.DebugErrors);
}
```

Ver imagen 30

```
//Se valida la variable de control de acciones en las hojas de excel
if (_cells == null)
    _cells = new ExcelStyleCells(_excelApp);

//Se realiza la autenticación
_frmAuth.StartPosition = FormStartPosition.CenterScreen;
_frmAuth.SelectedEnviroment = drpEnviroment.SelectedItem.Label;

if (_frmAuth.ShowDialog() != DialogResult.OK) return;// si no se autentica, se cancela el proceso

//Instanciar el Contexto de Operación
var opSheet = new OperationContext
{
    district = _frmAuth.EllipseDsct,
    position = _frmAuth.EllipsePost,
    maxInstances = 100,
    maxInstancesSpecified = true,
    returnWarnings = _eFunctions.DebugWarnings,
    returnWarningsSpecified = true
};

//Instanciar el SOAP
ClientConversation.authenticate(_frmAuth.EllipseUser, _frmAuth.EllipsePswd);

//Proceso del screen
var proxySheet = new Screen.ScreenService();
var requestSheet = new Screen.ScreenSubmitRequestDTO();


//Se define el ambiente del Dropdown de Enviroment
var urlService = _eFunctions.GetServicesUrl(drpEnviroment.SelectedItem.Label);
proxySheet.Url = urlService + "/ScreenService";
```

Imagen 27: Observaciones Adicionales: Autenticación y uso de servicios

```
//ejecutamos el programa y leemos la respuesta si se necesita en el replySheet
var replySheet = proxySheet.executeScreen(opSheet, "MSO435");

//validamos el ingreso al programa
if (replySheet.mapName != "MSM435A" || _excelApp.ActiveWorkbook.ActiveSheet.Name != SheetName01)
    return "ERROR:" + "No se pudo establecer comunicación con el servicio";

//arreglo para los campos del screen
var arrayFields = new ArrayScreenNameValue();

//se adicionan los campos que se vayan a enviar
arrayFields.Add("OPTION1I", "1");
arrayFields.Add("MODEL_CODE1I", modelCode);
arrayFields.Add("STAT_DATE1I", modelDate);
arrayFields.Add("SHIFT1I", modelShift);

//se envia el screen
requestSheet.screenFields = arrayFields.ToArray();
requestSheet.screenKey = "1";

replySheet = proxySheet.submit(opSheet, requestSheet);

//manejo de opciones para el tipo de respuesta
if (replySheet != null
    && replySheet.mapName == "MSM435B")
    && !_eFunctions.CheckReplyError(replySheet)
    && !_eFunctions.CheckReplyWarning(replySheet)
    && replySheet.functionKeys = "XMIT-Confirm")
{
    //si necesitas obtener los campso del reply y trabajar con ellos
    var replyFields = new ArrayScreenNameValue(replySheet.screenFields);
    bool isEmpty = replyFields.getField("WO_PROJ1I" + i).value.Equals("");
}
```

Imagen 28: Observaciones Adicionales: Continuación de uso de servicios y ScreenService

```
var opSheet = new WorkOrderService.OperationContext();
opSheet.district = _frmAuth.EllipseDsct;
opSheet.position = _frmAuth.EllipsePost;
opSheet.maxInstances = 100;
opSheet.maxInstancesSpecified = true;
opSheet.returnWarnings = _eFunctions.DebugWarnings;
opSheet.returnWarningsSpecified = true;

var proxyWo = new WorkOrderService.WorkOrderService();//ejecuta las acciones del servicio
var requestWo = new WorkOrderServiceCreateRequestDTO();

proxyWo.Url = urlService + "/WorkOrder";

//Se cargan los parámetros de  la solicitud
requestWo.districtCode = "ICOR";
requestWo.workGroup = "MTOLOC";
//se envía la acción
var replyWO = proxyWo.modify(opContext, requestWo);

//se analiza el reply según la acción
//se hacen las acciones pertinentes
```

Imagen 29: Observaciones Adicionales: Uso de otros servicios y MSE

```
//EJECUCIÓN DE QUERIES
//Estableces el servidor de base de datos con alguna de las siguientes tres opciones
_eFunctions.SetDBSettings(urlService);
_eFunctions.SetDBSettings(dbname, dbuser, dbpass);
_eFunctions.SetDBSettings(dbname, dbuser, dbpass, dblink, dbreference);

var dataReader = ef.GetQueryResult(sqlQuery);
if (dataReader == null || dataReader.IsClosed || !dataReader.HasRows) return null;
while (dataReader.Read())
{
    var workOrder = dataReader["WORK_ORDER"].ToString().Trim();
}


//CONTROL DE ERRORES Y DEBUGGING
//debugging de queries
if(_eFunctions.DebugQueries)
    _cells.getCell("L1").Value = sqlQuery;

//GESTIÓN DE EXCEPCIONES
try
{
    if (_excelApp.ActiveWorkbook.ActiveSheet.Name != SheetName01 + LabourSheetTypeConstants.Default)
        throw new Exception("La hoja seleccionada no coincide con el modelo requerido");

    if (drpEnviroment.Label == null || drpEnviroment.Label.Equals(""))
        throw new ArgumentException("Seleccione un ambiente válido");
}
catch (Exception ex)
{
    //Despliega una alerta de error o realiza la acción según se requiera
    MessageBox.Show(ex.Message, @"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    //Genera un log del error. Si _eFunctions.DebugErrors es true, siempre desplegará un mensaje de error
    ErrorLogger.LogError("RibbonEllipse:startLabourCostLoad()", "\n\rMessage:" + ex.Message
        + "\n\rSource:" + ex.Source + "\n\rStackTrace:" + ex.StackTrace, _eFunctions.DebugErrors);
}
```

Imagen 30: Observaciones Adicionales: Uso, ejecución de Querys y control de errores y debugging

7.5 Validaciones de Campo

Para utilizar validaciones de campo en las hojas de Excel utilizamos una variable global con el nombre de la hoja de Excel donde vamos a tener el listado de validaciones.

```
private const string ValidationSheetName = "ValidationSheetWorkOrder";
```

Se opta por tener el listado de validaciones en una hoja de Excel porque de hacerlo dinámicamente, al cerrar y reabrir el documento se perderán los valores en la fórmula de validación.

Hacemos uso de algunas de las tres formas para establecer la validación:

1. `SetValidationList(Range targetRange, List<string> validationValues, bool showError = true)`

   Establece en las celdas especificadas los valores de validación de forma dinámica. *No crea datos en la hoja de validación, ni usa la hoja de validación*

2. `SetValidationList(Range targetRange, List<string> validationValues, string validationSheetName, int validationColumnIndex, bool showError = true)`

   Establece en las celdas especificadas los valores de validación escribiéndolos en la hoja de validación según la columna indicada. *Crea datos en la hoja de validación según la lista de datos ingresada*

3. `SetValidationList(Range targetRange, string validationSheetName, int validationColumnIndex, bool showError = true)`

   Establece en las celdas especificadas los valores de validación tomándolos de la hoja de validación según la columna indicada. *Hace uso de los datos ya existentes en la hoja de validación*

targetRange: rango de celdas objetivo que tendrán la validación de datos
validationValues: lista de datos que se utilizarán como muestra de validación
validationSheetName: nombre de la hoja de Excel donde se agregarán u obtendrán los datos de validación
validationColumnIndex: índice de la columna de la hoja de validación de Excel donde se agregarán u obtendrán los datos de validación
showError: indica si se mostrará el mensaje de alerta de error cuando se ingresen datos que no correspondan en la celda

Obtenemos el valor del campo y lo utilizamos según como necesitemos:

```
//obtengo el valor de la celda
var codeComp = _cells.GetNullIfTrimmedEmpty(_cells.GetCell(15, i).Value);
//obtengo solo el código sin la descripción ("cod - description")
if (codeComp != null && codeComp.Contains(" - "))
        codeComp = codeComp.Substring(0, codeComp.IndexOf(" - ",
StringComparison.Ordinal));
```

A continuación algunos ejemplos:

```
//EJEMPLO CON TABLA DE CÓDIGOS 010
//obtengo el listado de códigos
var itemList1 = _eFunctions.GetItemCodes("SC");
//creo la variable de lista con código - descripción
var compCodeList = itemList1.Select(item => item.code + " - " + item.description).ToList();
//asigno los datos a la hoja de validación y la validación a la celda respectiva
_cells.SetValidationList(_cells.GetCell(15, i), compCodeList, ValidationSheetName, 2);

//para hacer uso de lo seleccionado. Obtengo el valor de la celda
var codeComp = _cells.GetNullIfTrimmedEmpty(_cells.GetCell(15, i).Value);
//elimino la descripción y quedo solo con el código
if (codeComp != null && codeComp.Contains(" - "))
          codeComp = codeComp.Substring(0, codeComp.IndexOf(" - ", StringComparison.Ordinal));

//uso la variable codeComp según necesite

//EJEMPLO CON DISTRITO Y GRUPOS DE TRABAJO
//Adicionar validaciones
_cells.SetValidationList(_cells.GetCell("B3"), DistrictConstants.GetDistrictList(), ValidationSheetName, 1);
_cells.SetValidationList(_cells.GetCell("B4"), GroupConstants.GetWorkGroupList().Select(g => g.Name).ToList(),
ValidationSheetName, 2);
var district = _cells.GetNullIfTrimmedEmpty(_cells.GetCell("B3").Value);
var workGroup = _cells.GetNullIfTrimmedEmpty(_cells.GetCell("B4").Value);

//uso la variable district y workGroup según necesite
```

```
//EJEMPLO CON LISTAS CORTAS DINÁMICAS
_cells.SetValidationList(_cells.GetCell(8, i), new List<string> { "P - Project", "W - WorkOrder" }, ValidationSheetName,
1);
var referenceType = _cells.GetNullIfTrimmedEmpty(_cells.GetCell(8, i).Value);
//elimino la descripción y quedo solo con el código
if (referenceType != null && referenceType.Contains(" - "))
          referenceType = referenceType.Substring(0, referenceType.IndexOf(" - ", StringComparison.Ordinal));

//EJEMPLO CON DATOS YA EXISTENTE
_cells.SetValidationList(_cells.GetCell("B3"), DistrictConstants.GetDistrictList(), ValidationSheetName, 1);
_cells.SetValidationList(_cells.GetCell("B4"), ValidationSheetName, 1);
var district1 = _cells.GetNullIfTrimmedEmpty(_cells.GetCell("B3").Value);
var district2 = _cells.GetNullIfTrimmedEmpty(_cells.GetCell("B4").Value);
```

```
1   //EJEMPLO CON TABLA DE CÓDIGOS 010
2   //obtengo el listado de códigos
3   var itemList1 = _eFunctions.GetItemCodes("SC");
4   //creo la variable de lista con código - descripción
5   var compCodeList = itemList1.Select(item => item.code + " - " + item.description).ToList();
6   //asigno los datos a la hoja de validación y la validación a la celda respectiva
7   _cells.SetValidationList(_cells.GetCell(15, i), compCodeList, ValidationSheetName, 2);
8
9   //para hacer uso de lo seleccionado. Obtengo el valor de la celda
10  var codeComp = _cells.GetNullIfTrimmedEmpty(_cells.GetCell(15, i).Value);
11  //elimino la descripción y quedo solo con el código
12  if (codeComp != null && codeComp.Contains(" - "))
13      codeComp = codeComp.Substring(0, codeComp.IndexOf(" - ", StringComparison.Ordinal));
14
15  //uso la variable codeComp según necesite
16
17  //EJEMPLO CON DISTRITO Y GRUPOS DE TRABAJO
18  //Adicionar validaciones
19  _cells.SetValidationList(_cells.GetCell("B3"), DistrictConstants.GetDistrictList(), ValidationSheetName, 1);
20  _cells.SetValidationList(_cells.GetCell("B4"), GroupConstants.GetWorkGroupList().Select(g => g.Name).ToList(),
    ValidationSheetName, 2);
21  var district = _cells.GetNullIfTrimmedEmpty(_cells.GetCell("B3").Value);
22  var workGroup = _cells.GetNullIfTrimmedEmpty(_cells.GetCell("B4").Value);
23
24  //uso la variable district y workGroup según necesite
25
26  //EJEMPLO CON LISTAS CORTAS DINÁMICAS
27  _cells.SetValidationList(_cells.GetCell(8, i), new List<string> { "P - Project", "W - WorkOrder" },
    ValidationSheetName, 1);
28  var referenceType = _cells.GetNullIfTrimmedEmpty(_cells.GetCell(8, i).Value);
29  //elimino la descripción y quedo solo con el código
30  if (referenceType != null && referenceType.Contains(" - "))
31      referenceType = referenceType.Substring(0, referenceType.IndexOf(" - ", StringComparison.Ordinal));
32
33  //EJEMPLO CON DATOS YA EXISTENTE
34  _cells.SetValidationList(_cells.GetCell("B3"), DistrictConstants.GetDistrictList(), ValidationSheetName, 1);
35  _cells.SetValidationList(_cells.GetCell("B4"), ValidationSheetName, 1);
36  var district1 = _cells.GetNullIfTrimmedEmpty(_cells.GetCell("B3").Value);
37  var district2 = _cells.GetNullIfTrimmedEmpty(_cells.GetCell("B4").Value);
```

Imágen 31. Ejemplos de uso de validaciones

7.6 Gestión de múltiples hojas de Excel

Para la gestión las hojas de Excel se utiliza la clase *ExcelStyleCells.cs*, por lo que hay que conocer algunos elementos básicos de la misma.

La declaración de una variable de esta clase está sujeta a la ejecución de la aplicación y la hoja activa de la misma. Utiliza un apuntador que establece la hoja con la que se definió al momento de la creación. Si no se definió ninguna hoja, la hoja definida será la hoja principal.

1. Instanciar sin definir la hoja principal

```
var _cells = new ExcelStyleCells Application excelApp,
       bool alwaysActiveSheet = true);
```

Al definir la variable de esta forma, siempre que se haga referencia a *_cells* se estará haciendo referencia a la hoja activa de Excel dentro del proceso. La hoja base será la hoja que estaba activa al momento de la creación de la variable.

2. Instanciar definiendo la hoja principal

```
var _cells = ExcelStyleCells(Application excelApp, string sheetName);
```

Al definir la variable de esta forma, siempre que se haga referencia a _cells se estará haciendo referencia a la hoja base, la hoja con el nombre *sheetName* dado inicialmente, sin importar si es la hoja activa o no.

Existen otros métodos para variar este comportamiento después que la variable haya sido instanciada.

```
_cells.SetActiveSheet();
```

Establece como hoja base la hoja activa que tenga la aplicación en el momento. No modifica el comportamiento de referenciación

```
_cells.SetActiveSheet(string sheetName);
```

Establece como hoja base la hoja ingresada. No modifica el comportamiento de referenciación

```
_cells.SetAlwaysActiveSheet(bool value);
```

Establece el comportamiento de referenciación. *True* para que la referencia siempre sea la hoja activa y *False* para que sea la hoja base.

```
_cells.ToggleAlwaysActiveSheet();
```

Cambia el estado del comportamiento de referenciación. Si estaba activo lo desactiva y viceversa.

7.7 Manejo de hilos de procesos

Para el desarrollo de algunos loaders puede ser requerido el manejo de hilos de proceso con el fin de evitar que la aplicación entre en estado de *no responde* hasta que se termine el proceso solicitado.

VSTO tiene algunas restricciones en cuanto a desempeño y manejo de estos hilos de procesos, por lo que si lo requiere puede investigar más al respecto. A continuación hacemos uso de los hilos de procesos tomando en cuenta algunas recomendaciones en la implementación de los mismos para VSTO.

Definición:

Hacemos uso de la librería del sistema para los hilos del proceso.

```
using System.Threading;
```

Dentro de nuestra clase principal (RibbonEllipse) definiríamos la variable que vamos a utilizar para la gestión de los hilos del proceso. *Si requiere más de un hilo de proceso simultáneo puede crear más de una variable.*

```
private Thread _thread;
```

Iniciar un hilo de proceso:

Para hacer uso de un hilo de proceso validamos que este no haya sido iniciado previamente y que todavía esté corriendo. *De no hacer esta validación se podrían crear incontrolables números de hilos de proceso lo cual podría llevar a errores en los procesos requeridos*. Si ya hay un hilo iniciado y no se ha terminado, bien podría generar una excepción o simplemente no hacer nada.

Generar Excepción:

```
if (_thread != null && _thread.IsAlive)
      throw new Exception("Ya hay un proceso ejecución. Espere que el proceso termine e intente nuevamente");
```

No hacer nada:

```
if (_thread != null && _thread.IsAlive) return;
```

Si no hay ningún hilo activo se procede a continuar. Para esto inicializamos la variable que definimos para nuestro hilo y le asignamos el nombre del método que deseamos que lleve a cabo de nuestro hilo (en el ejemplo *CreateWoList*). La siguiente línea de ejecución corresponde a una recomendación de manejo de hilos en VSTO.

```
_thread = new Thread(CreateWoList);
_thread.SetApartmentState(ApartmentState.STA);
```

Finalmente, invocamos la ejecución del hijo donde se requiera.

```
_thread.Start();
```

Se recomiendo el uso del método de clase "_cells.SetCursorWait();" dentro del método que ejecutará el Thread para indicar al usuario mediante la animación del cursor que se está ejecutando un proceso en segundo plano. Recuerde que deberá garantizar que _cells no sea nulo.

Detener un hilo de proceso:

Es importante conocer cómo detener un hilo de proceso para que en caso que una solicitud se haya vuelto demasiado extensa podamos detenerla sin perder la información recogida hasta ese momento.

Para hacerlo validamos que el hilo esté inicializado y realmente esté en ejecución, de este modo evitamos procesos y errores innecesarios. Posteriormente llamando su método de Abort() detemos el proceso que este hilo esté ejecutando. Este proceso se hace dentro de

un try/catch para poder capturar cualquier excepción que ocurra en el proceso de forzar una detención.

```
try
{
    if (_thread != null && _thread.IsAlive)
        _thread.Abort();
}
catch (ThreadAbortException ex)
{
    MessageBox.Show(@"Se ha detenido el proceso. " + ex.Message);
}
```

Se recomiendo el uso del método de clase "_cells.SetCursorDefault ();" para indicar al usuario mediante la animación del cursor que se detuvieron los procesos en segundo plano.

A continuación dejamos un ejemplo de este proceso que está sujeto a unos controles de tipo button. En el primero iniciamos el llamado al hilo y lo ejecutamos de una vez, y en el segundo la opción de detener el proceso.

```
private void btnCreate_Click(object sender, RibbonControlEventArgs e)
{
    if (_excelApp.ActiveWorkbook.ActiveSheet.Name == SheetName01)
    {
        _frmAuth.StartPosition = FormStartPosition.CenterScreen;
        _frmAuth.SelectedEnviroment = drpEnviroment.SelectedItem.Label;
        if (_frmAuth.ShowDialog() != DialogResult.OK) return;
        //si si ya hay un thread corriendo que no se ha detenido
        if (_thread != null && _thread.IsAlive) return;
        _thread = new Thread(CreateWoList);

        _thread.SetApartmentState(ApartmentState.STA);
        _thread.Start();
    }
    else
        MessageBox.Show(@"La hoja de Excel seleccionada no tiene el formato válido para realizar la acción");
}
```

Imagen 32. Inicialización y ejecución de un método en un hilo de proceso

```
private void btnStopThread_Click(object sender, RibbonControlEventArgs e)
{
    try
    {
        if (_thread == null || !_thread.IsAlive) return;
        _thread.Abort();
        _cells.SetCursorDefault();
    }
    catch (ThreadAbortException ex)
    {
        MessageBox.Show(@"Se ha detenido el proceso. " + ex.Message);
    }
}
```

Imagen 33. Detención forzosa de un hilo de proceso

Para efectos de desarrollo de los loaders en Excel, hay que tener en cuenta que si el hilo está consultando o escribiendo información en una hoja de un libro lo haga haciendo uso de una referencia absoluta de la clase ExcelStyleCell. Esto con el fin de evitar que si el proceso está en ejecución y el usuario cambia de hoja o de libro, interfiera con el proceso del hilo al cambiar la hoja activa (ver 7.6).

8. CONTROL DE VERSIONES Y PUBLICACIÓN DEL PROYECTO

8.1 Control de versiones

Es importante la correcta gestión de versiones para que siempre se mantenga actualizado el AddIn ante cualquier cambio que se realice.

Para esto desplegamos el menú de nuestro proyecto [1] le damos *Properties [2]*

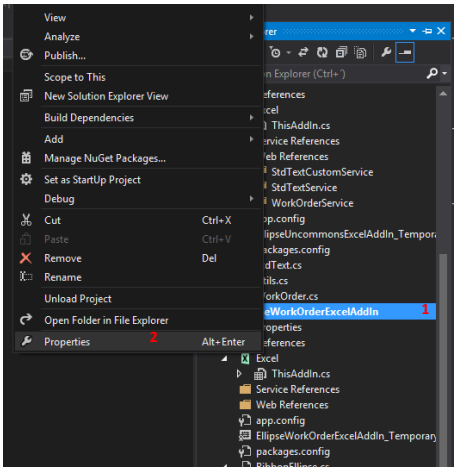

Imagen 34 Despliegue de propiedades del proyecto

Seleecionamos la opción de *Application [3]* en la barra lateral izquierda y hacemos clic en el botón *de Assembly Information [4].*

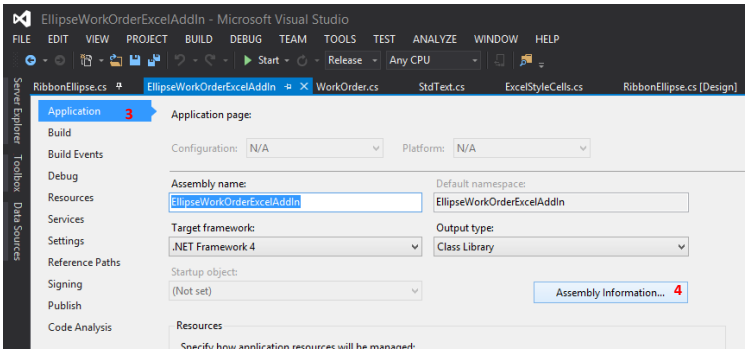

Imagen 35. Propiedades del proyecto

Gestionamos la información según deseamos teniendo en principal cuenta el número de la versión [5] y seleccionamos el botón OK [6].

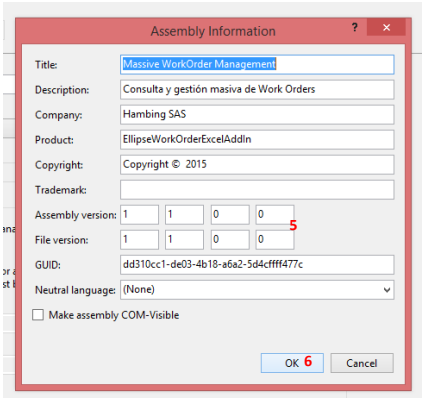

Imagen 36. Información de Ensamblado

Al mismo tiempo, seleccionamos ahora la opción de *Publish* [7] y ponemos el mismo número de versión en la sección de *Publish Version [8]*

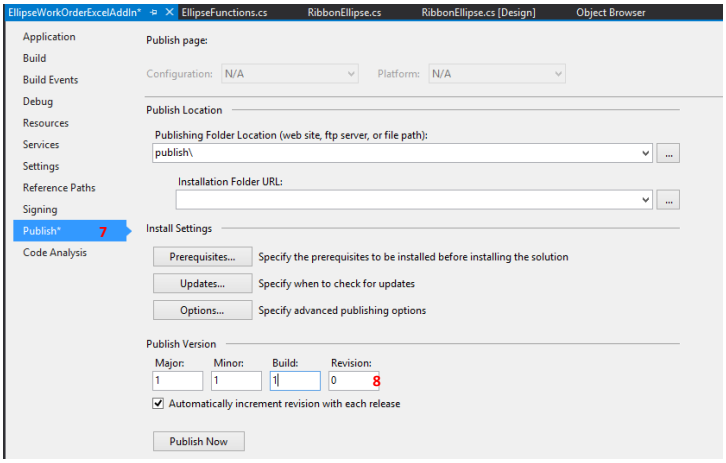

Imagen 37. Información de Publicación

Finalmente podremos ingresar en la ruta del proyecto y en la carpeta publish encontraremos los ficheros de publicación del AddIn

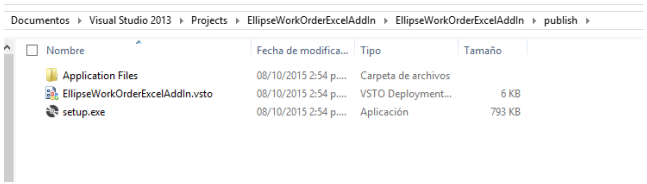

Imagen 38. Archivos de Publicación