

# **MANUAL DE DESARROLLO DE LOADERS ELLIPSE 8.4**

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

**Versión 1.3.2**  
**23 de Diciembre de 2019**

## TABLA DE CONTENIDO

PALABRAS CLAVES .....	3
REQUISITOS PREVIOS:.....	3
PROCEDIMIENTO .....	4
1. CREAR NUEVO PROYECTO .....	4
2. CONSTRUCCIÓN DE LA ESTRUCTURA GENERAL DEL PROYECTO.....	5
2.1 Crear Cinta de Opciones (Ribbon).....	5
2.2 Renombrar Elementos de la Cinta de Opciones (Ribbon).....	7
2.3 Adicionar controles a la Cinta de Opciones (Ribbon).....	8
3. AGREGAR REFERENCIA A ORACLE DATA PROVIDER.....	11
3.1 Instalación .....	11
3.1.1 Instalar por NuGet del ODP.Net .....	11
3.1.2 Instalación Manual .....	12
Agregar referencia a la librería.....	12
Agregar archivo de configuración app.config .....	14
3.2 Agregar referencias TNS/DataSources de acceso a bases de datos .....	16
3.3 Agregar código de referencia de Oracle Data Manager .....	18
4. AGREGAR REFERENCIAS A LIBRERÍAS COMUNES.....	19
4.1 Vincular el proyecto EllipseCommonsClassLibrary .....	19
4.2 Agregar código de referencias de Librerías Commons .....	21
5. AGREGAR CABECERA DE CONSUMO SOAP .....	24
5.1 Agregar referencia a librería de consumo SOAP.....	24
5.2 Adicionar referencia a app.config .....	25
5.3 Agregar código de referencia de encabezado SOAP en el proyecto .....	26
6. REFERENCIAR Y CONSUMIR SERVICIOS DE ELLIPSE.....	27
6.1 Agregar Web Service de Ellipse.....	27
6.2 Uso de un Web Service .....	29
6.2.1 Servicio de Autenticación:.....	30
6.2.2 Screen Service (MSO) .....	31
6.2.3 Web Service General (MSE) .....	34
7. OBSERVACIONES ADICIONALES DE DESARROLLO .....	37

7.1	Ejecución de Querys.....	37
7.3.	Gestión de Códigos con Descripción (Tabla 010) .....	38
7.4	Validaciones de Campo.....	39
7.5	Gestión de múltiples hojas de Excel.....	40
7.6	Manejo de hilos de procesos .....	42
7.7	Control de Errores, Debugging y Configuración.....	44
8.	CONTROL DE VERSIONES Y PUBLICACIÓN DEL PROYECTO .....	46
8.1	Acerca de.....	46
8.2	Control de versiones .....	47

## **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. Microsoft Office 2010 o superior
3. .Net Framework 4.0
4. Acceso a la Red Interna

# PROCEDIMIENTO

## 1. CREAR NUEVO PROYECTO

Desplegamos el menú principal y seleccionamos “[1] FILE > [2] New > [3] Project ...”

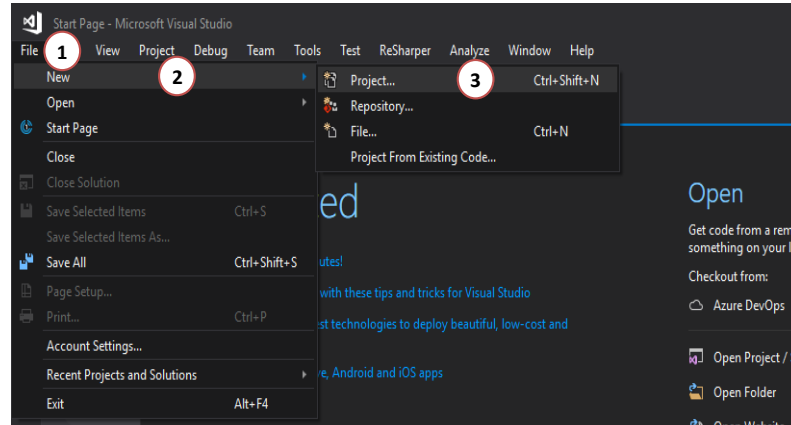


Imagen 1. Nuevo Proyecto: Creación de Nuevo proyecto

Seleccionamos la plantilla correspondiente a “Excel 2010 AddIn” en las opciones laterales, “[4] Visual C# > [5] Office/SharePoint > [6] VSTO 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.

[10] Establecemos las opciones del repositorio y de la jerarquía de directorios según se requiera y presionamos el botón de OK.

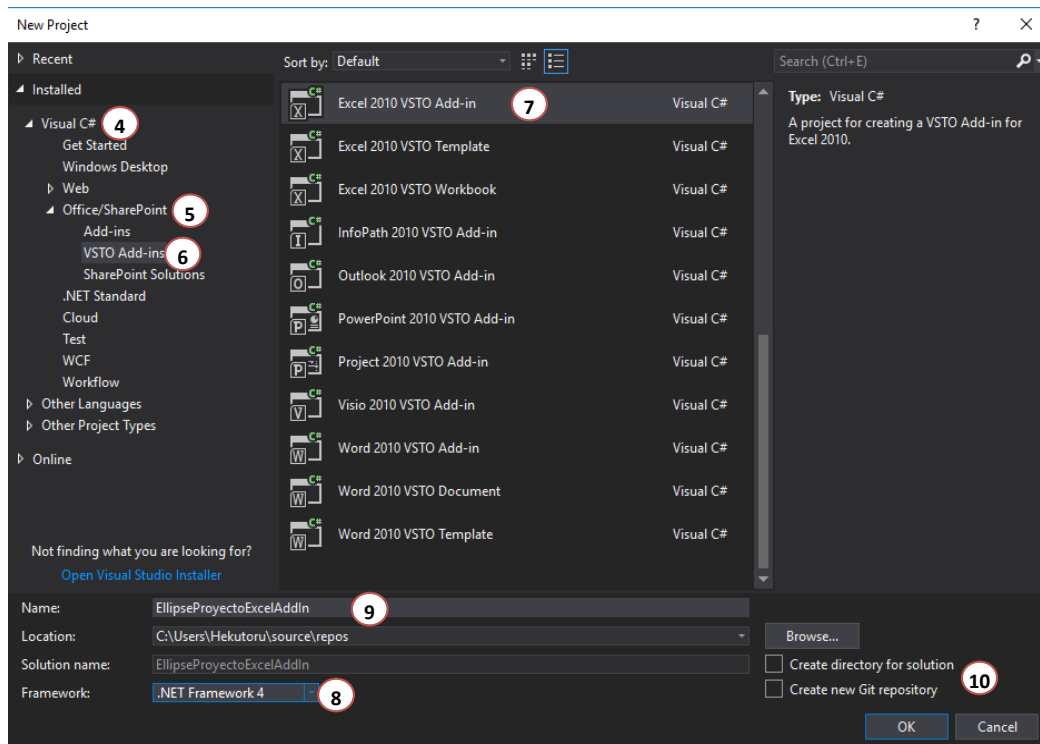


Imagen 2. Nuevo Proyecto: Selección de Tipo de Proyecto

## 2. CONSTRUCCIÓN DE LA ESTRUCTURA GENERAL DEL PROYECTO

### 2.1 Crear Cinta de Opciones (Ribbon)

Desplegamos el [1] menú de opciones del proyecto sea desde el menú principal o haciendo clic derecho en el explorador de soluciones, y seleccionamos “[2] Add > [3] New Item...”

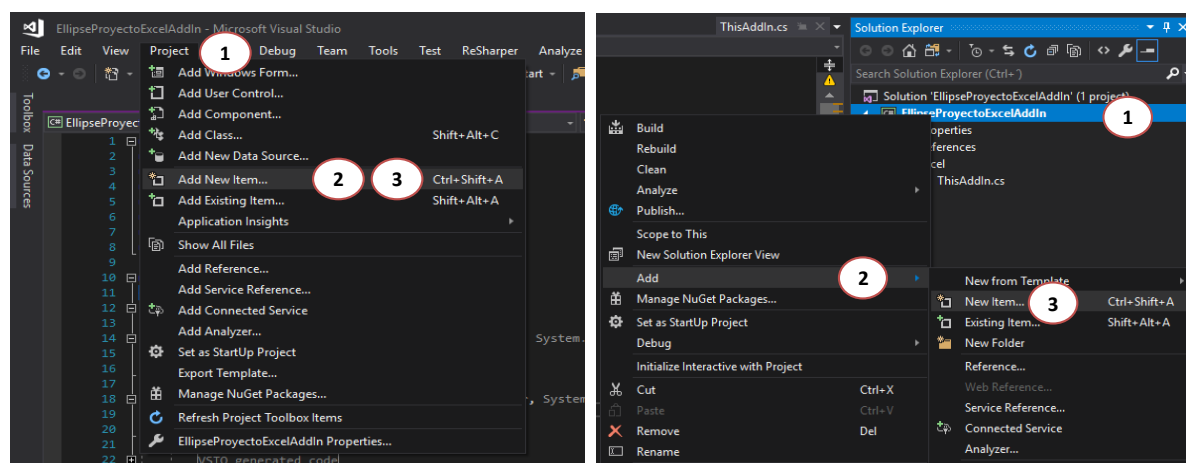


Imagen 3. Crear Cinta de Opciones (Ribbon): Nuevo Ítem

En el cuadro desplegado seleccionamos el elemento de la categoría “[4] Visual C# Items” y “[5] Office/SharePoint” el elemento “[6] Ribbon (Visual Designer)”. Le damos nombre al Ribbon a adicionar [7] según la convención establecida “RibbonEllipse.cs” y le damos al “[8] botón Add”

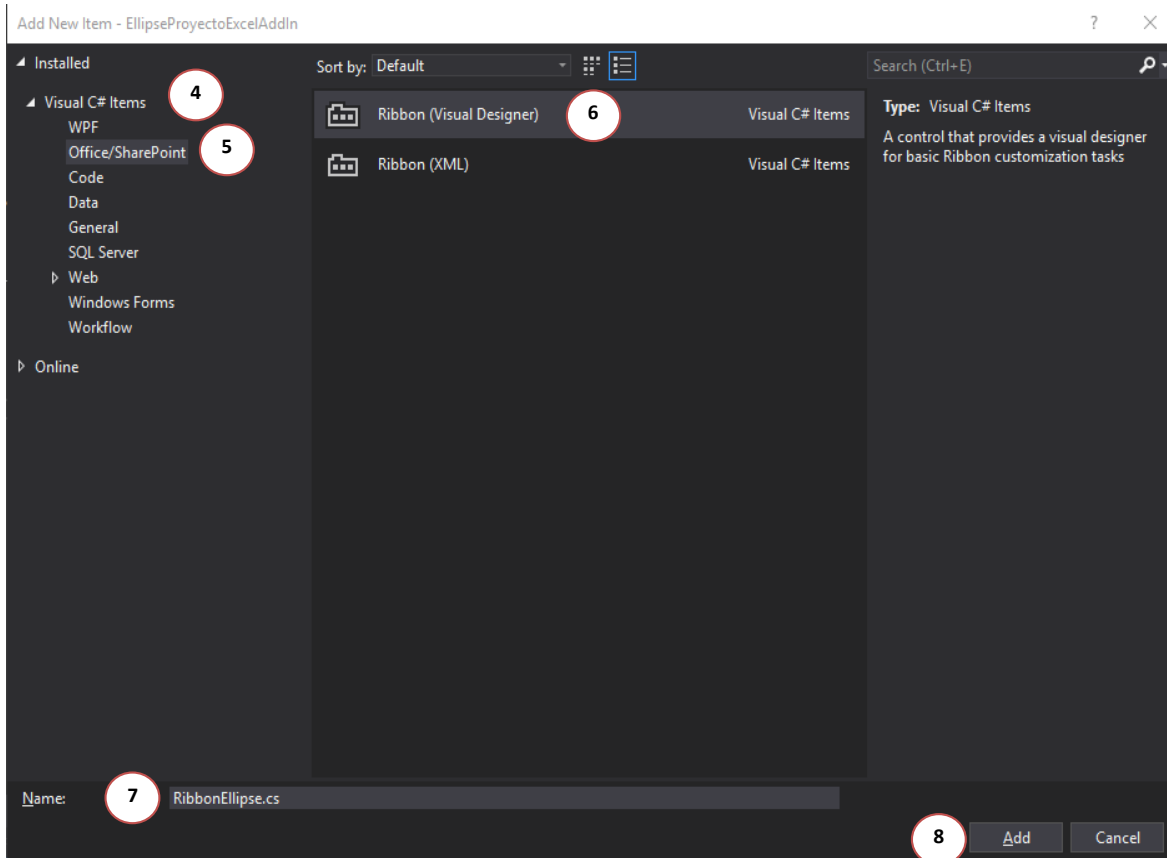


Imagen 4. Crear Cinta de Opciones (Ribbon): Seleccionar tipo de Item

## 2.2 Renombrar Elementos de la Cinta de Opciones (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. Esta acción permitirá agrupar todos los AddIns que se desarrollen, bajo la misma pestaña de opciones.

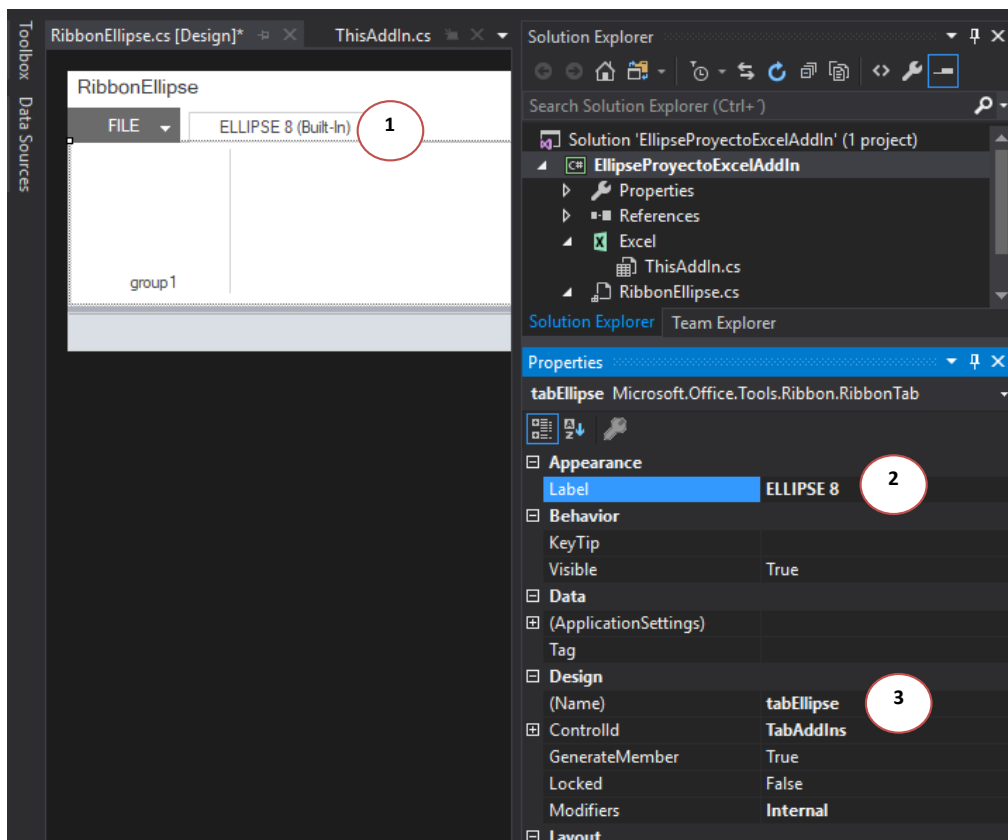


Imagen 5. Renombrar Cinta de Opciones (Ribbon): Renombrar Pestaña

Repetimos el mismo proceso para el grupo del Ribbon. Seleccionamos el grupo del Ribbon [4] y cambiamos el texto de la etiqueta al nombre del proyecto [5] “Proyecto” y también cambiamos el nombre [6] con el que vamos a hacer referencia en nuestro código a “grpProyecto”.

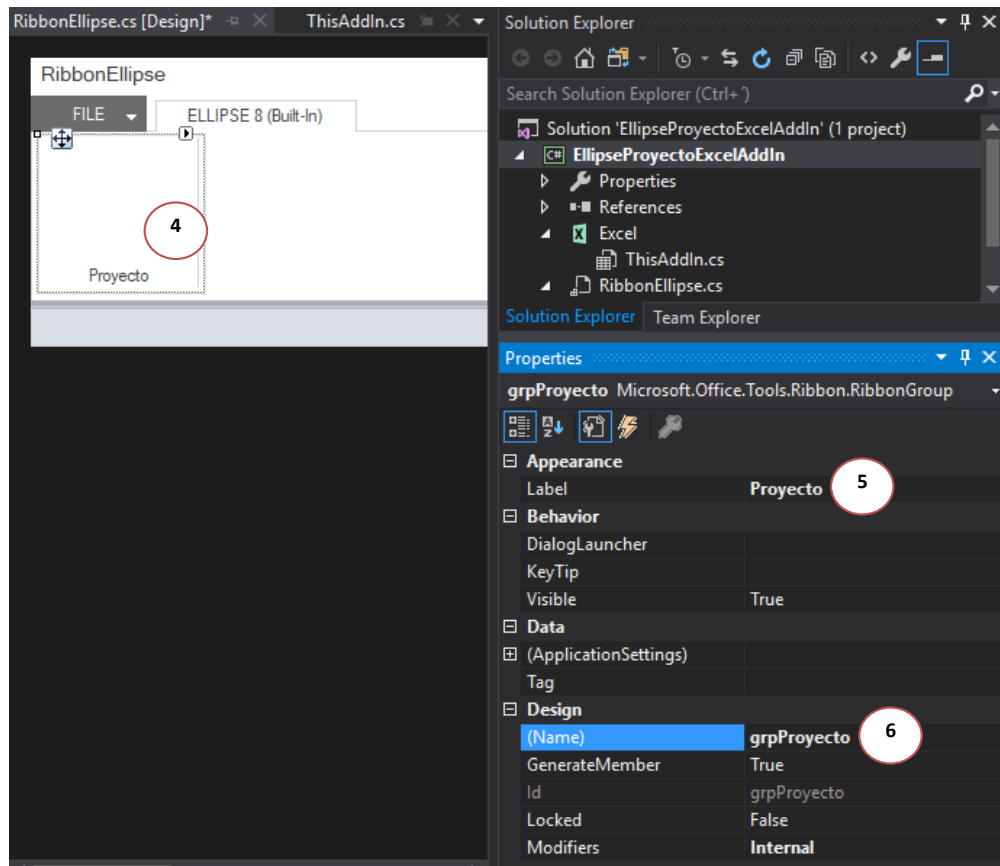


Imagen 6. Renombrar Elementos de la Cinta de Opciones (Ribbon)

### 2.3 Adicionar controles a la Cinta de Opciones (Ribbon)

Desde la barra lateral izquierda, desplegamos las opciones de la “[1] Caja de Herramientas (Toolbox)” y del menú “[2] Office Ribbon Controls” añadiremos los siguientes elementos según como se ve en la imagen:

- Box
- Button (Dentro del elemento Box “button1”)
- Button (Dentro del elemento Box “button2”)
- DropDown
- Button (“button3”)

Nota: Según la cantidad de opciones y/o acciones que se requiera, el botón 3 deberá ser reemplazado por un menú de opciones, y dentro de este se adicionarán los botones respectivos para cada opción/acción.



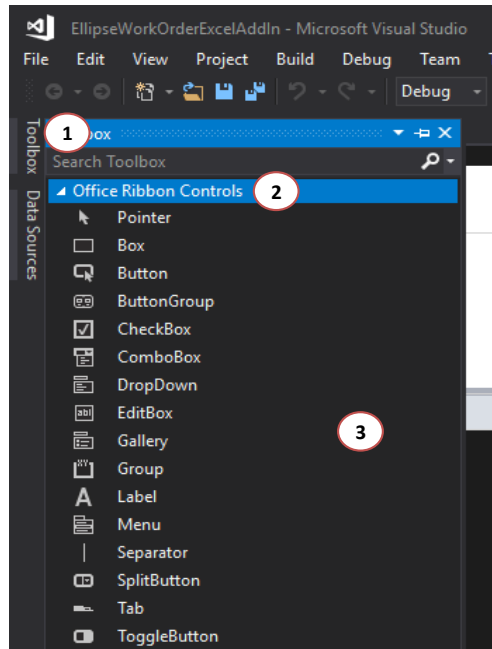


Imagen 7. Adicionar controles a la Cinta de Opciones (Ribbon): Selección de Controles

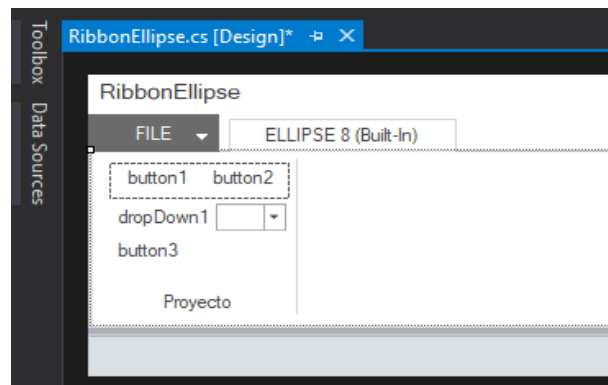


Imagen 8. Adicionar controles a la Cinta de Opciones (Ribbon): Controles Añadidos

Seleccionamos cada uno de los elementos y modificamos los valores en la barra de propiedades, según como lo vemos en la siguiente tabla.

Elemento (Item)	Etiqueta (Label)	Nombre (Name)
<b>button1</b>	Formatear	btnFormat
<b>button2</b>	?	btnAbout
<b>dropDown1</b>	Env.	drpEnvironment
<b>button3</b>	Ejecutar	btnExecution

Tabla 1: Renombrar valores de Controles de Cinta de Opciones (Ribbon)

Nota: Para asignar un atajo de tecla a algún elemento, es suficiente con añadir el símbolo ampersand "&" antes de la letra del atajo en los valores de Label.

El resultado esperado debe verse como lo muestra la siguiente imagen.

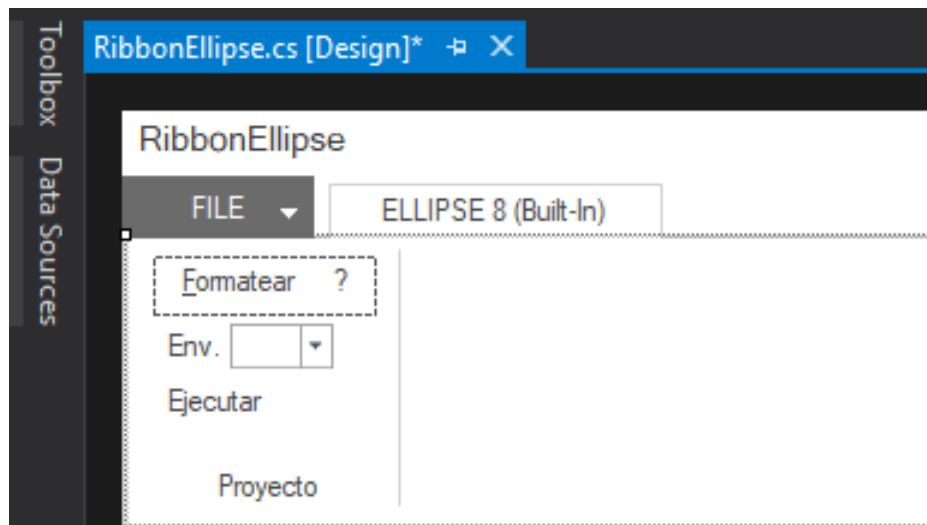


Imagen 9. Renombrar Controles de la Cinta de Opciones (Ribbon)

Nota: La lista de los servidores para la lista desplegable de Entorno se agregará de forma automática en la función de carga del Ribbon.

### 3. AGREGAR REFERENCIA A ORACLE DATA PROVIDER

La librería de Oracle Data Provider corresponde a la librería para acceder a las bases de datos tipo Oracle. Esta librería puede ser añadida directamente desde el repositorio NuGet para instalar la última versión o puede añadirse de forma manual la versión deseada.

#### 3.1 Instalación

##### 3.1.1 Instalar por NuGet del ODP.Net

Nota: Si va a hacer uso de la librería `EllipseCommonsClassLibrary` se recomienda omitir este paso y seguir con la instalación manual.

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

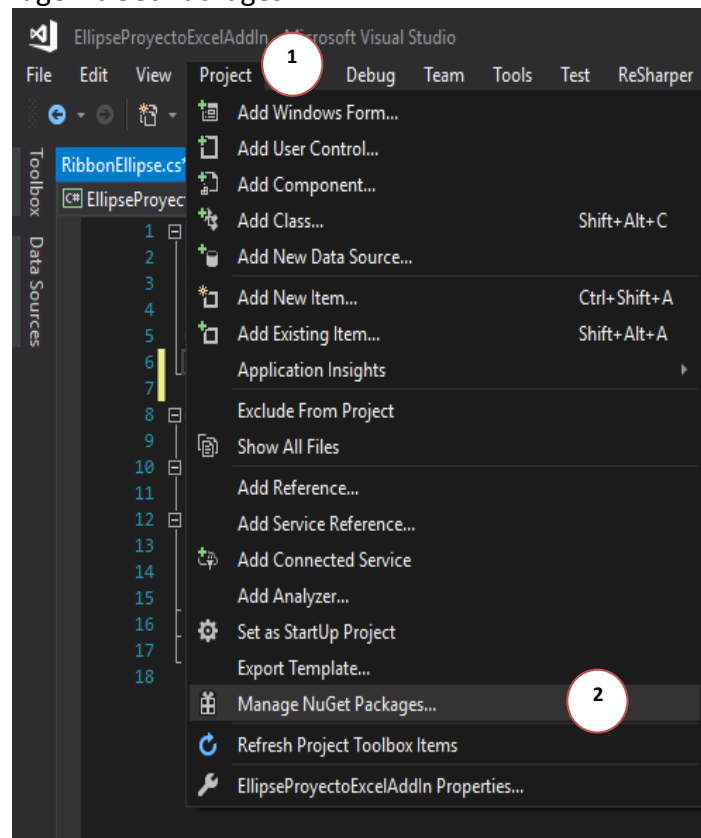


Imagen 10. Agregar Oracle Data Provider

Buscamos [3] Online, y escribimos como criterio [4] “Oracle ODP”. Seleccionamos el resultado como se muestra en la imagen y le damos al [5] botón Instalar.

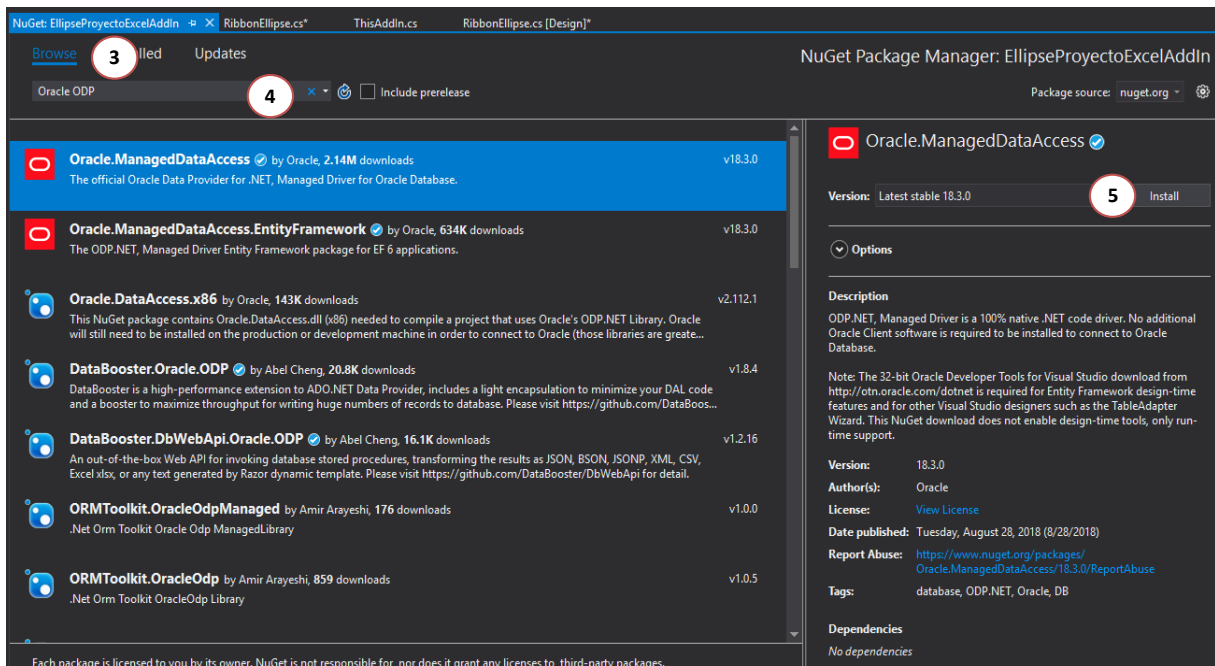


Imagen 11. Agregar Oracle Data Provider: Instalar la referencia

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

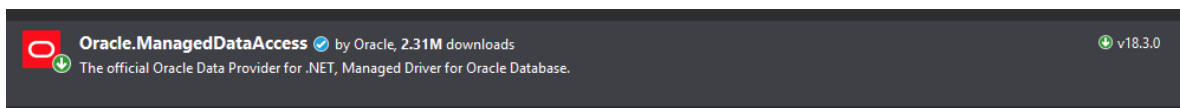


Imagen 12. Agregar Oracle Data Provider: Verificar el estado

### 3.1.2 Instalación Manual

Nota: Si realizó la instalación por el paquete NuGet, puede omitir este paso.

#### Agregar referencia a la librería

Desplegamos el [1] menú de opciones del proyecto sea desde el menú principal o haciendo clic derecho en el explorador de soluciones, y seleccionamos "[2] Agregar (Add) > [3] Referencia (Reference)"

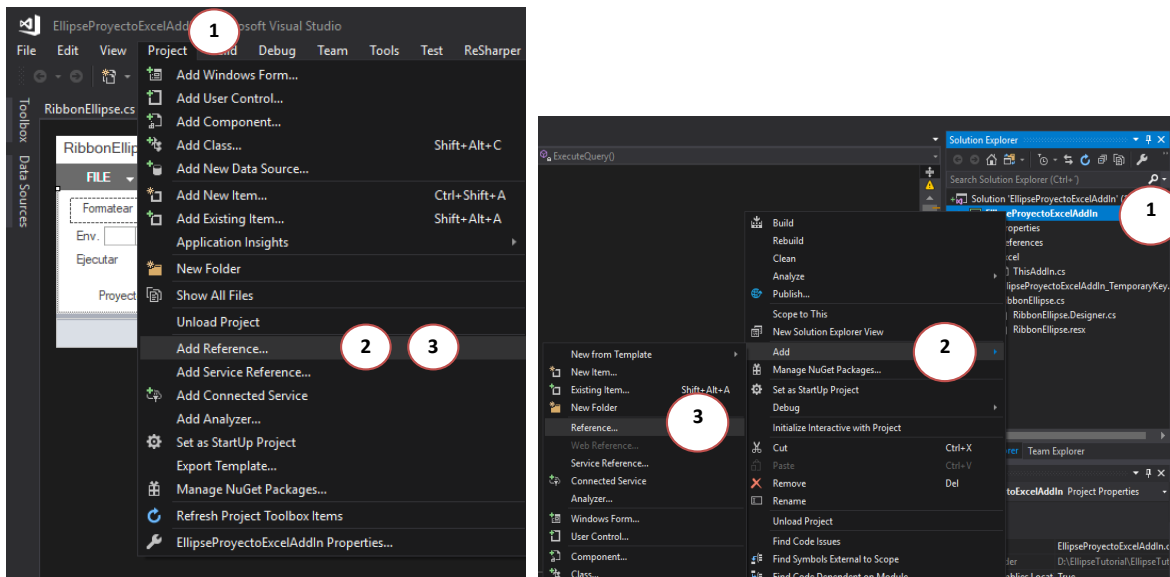


Imagen 13. Adicionar librería ODP: Nueva referencia

En la barra lateral izquierda seleccionamos la opción de [4] Búsqueda (Browse) y nuevamente en el botón de [5] Búsqueda (Browse). Ubicamos la ruta donde se encuentra el archivo de librería con la versión correspondiente, seleccionamos [6] Oracle.ManagedDataAccess.dll y hacemos clic en el botón de [7] Añadir (Add).

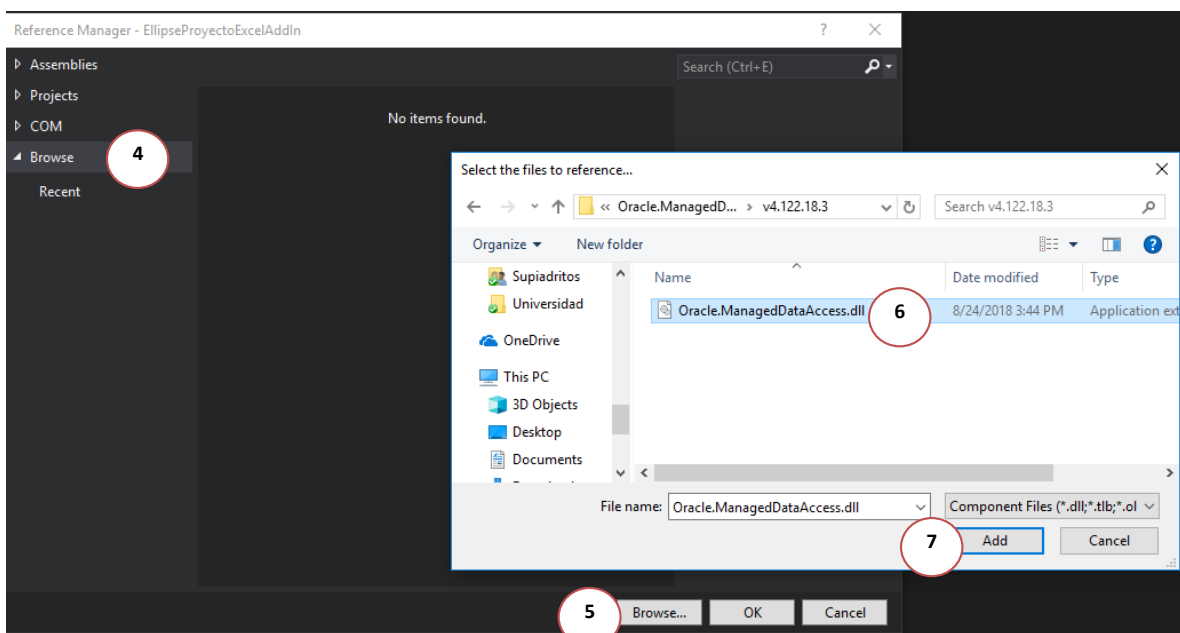


Imagen 14. Adicionar librería ODP: Seleccionar ruta

Nos aseguramos que el cuadro de selección [8] esté activo en el listado y finalizamos haciendo clic en el botón de [9] OK

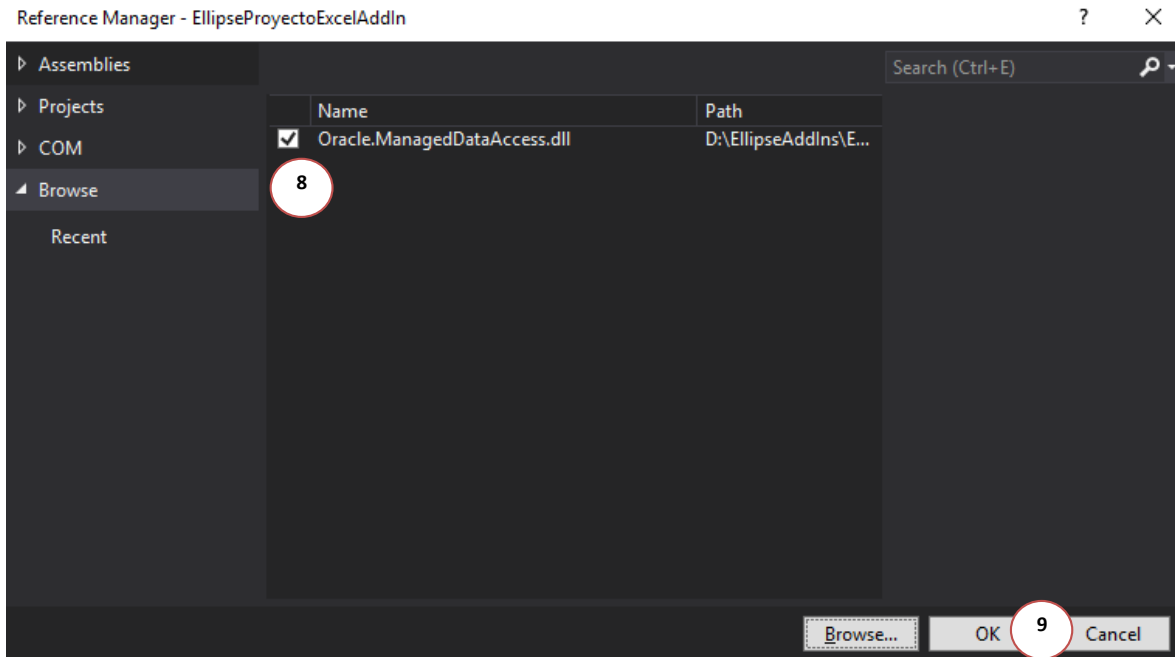


Imagen 15. Adicionar librería ODP: Seleccionar librería

### Agregar archivo de configuración app.config

Desplegamos el menú de [1] Proyecto (Project) en el menú principal o haciendo clic derecho en nuestro proyecto, seleccionamos la opción de [2] Agregar (Add) y [3] Nuevo Ítem (New Item)

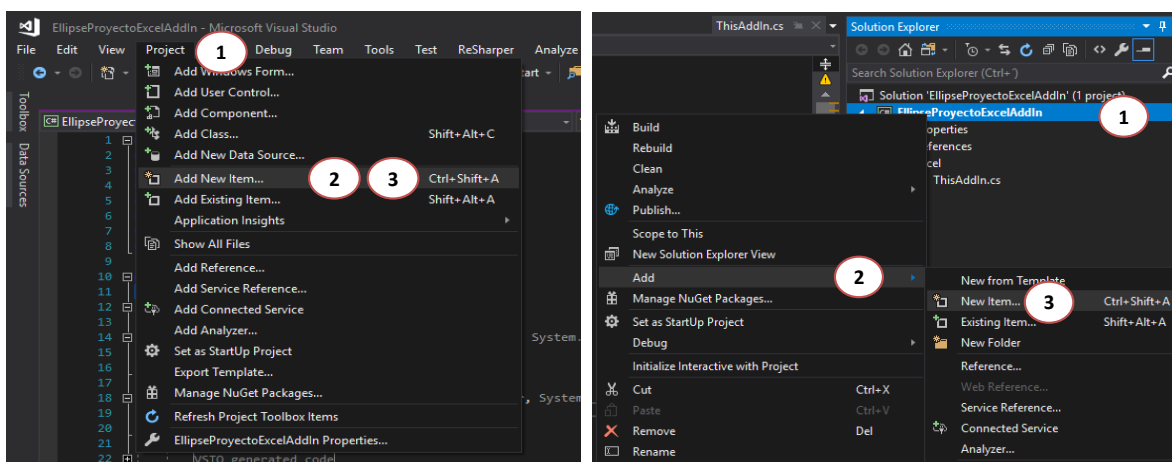


Imagen 16. Crear archivo de configuración app.config: agregar nuevo

Seleccionamos en la barra lateral izquierda la opción de [4] Visual C#Items -> General y como tipo de archivo [5] Archivo de texto (Text File). Finalmente ingresamos el nombre [6] app.config y hacemos clic en el botón de [7] Agregar (Add)

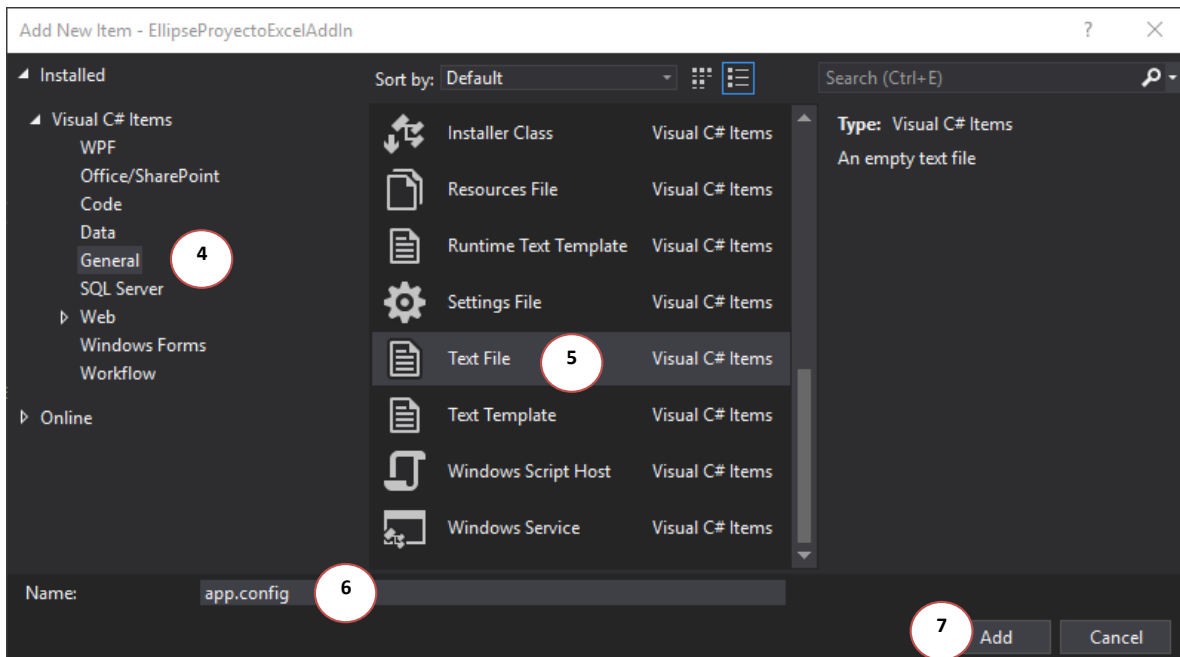


Imagen 17. Crear archivo de configuración app.config: crear nuevo

En el explorador de soluciones, veremos el nuevo archivo “app.config” creado. Damos doble clic en este para abrirlo y agregamos el contenido mostrado a continuación.

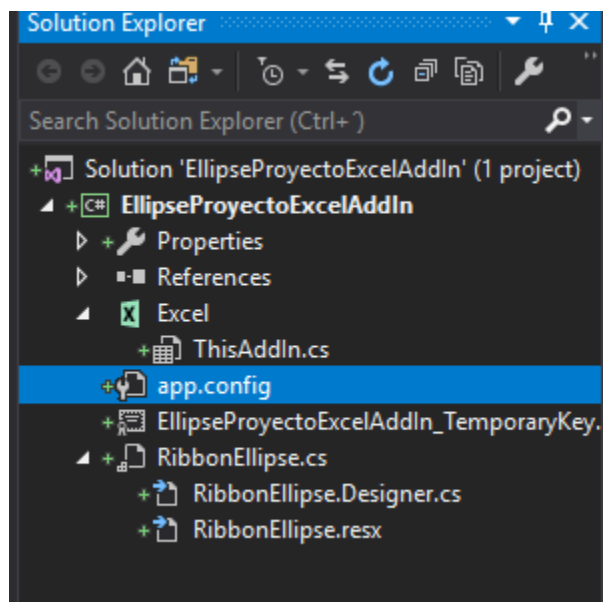


Imagen 18. Crear archivo de configuración app.config: agregar contenido

Tenga en cuenta los lugares resaltados en amarillo que corresponden a la versión que se ha adicionado al proyecto. Si instala una versión diferente deberá especificarla en estos lugares.

Imagen 19. Crear archivo de configuración app.config: detalle de contenido

Nota: Junto a este documento encontrará unos archivos anexos con este archivo creado completamente (app.config.initial.txt). Podrá copiar su contenido y omitir la creación manual del archivo.

### 3.2 Agregar referencias TNS/DataSources de acceso a bases de datos

De forma predeterminada, la librería ODP hace uso de las fuentes de datos ingresadas manualmente en el archivo de configuración app.config creado al instalar la librería ODP.net. Puede omitir este paso si hará uso de la librería EllipseCommonsClassLibrary.

Abres el archivo *app.config* dentro del proyecto



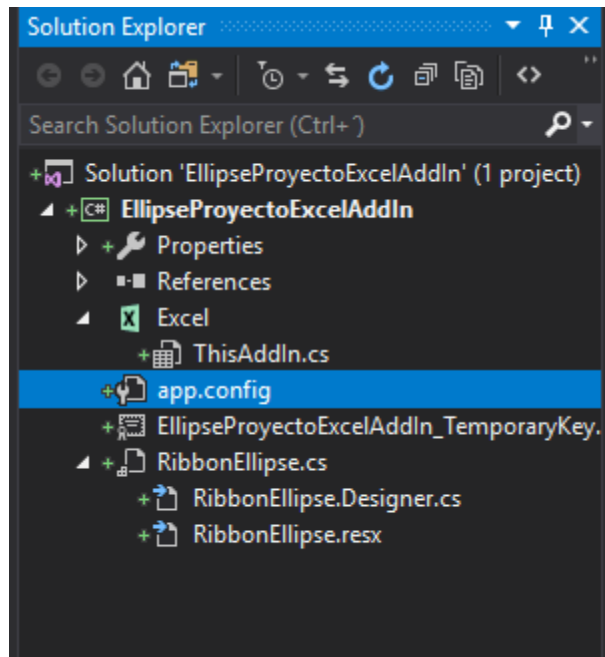


Imagen 20. Referenciar DataSources: Abrir app.config

Y copias el código dentro de la etiqueta de <oracle.manageddataaccess.client> la información de gestión de acceso del Oracle según la forma de trabajo deseada.

Para TNS:

```
<settings>
  <setting name="TNS_ADMIN" value="C:\oracle\product\11.2.0\client\network\ADMIN"/>
</settings>
```

```
</system.data>
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <publisherPolicy apply="no" />
      <assemblyIdentity name="Oracle.ManagedDataAccess" publicKeyToken="89b483f429c47342" culture="neutral" />
      <bindingRedirect oldVersion="4.121.0.0 - 4.65535.65535.65535" newVersion="4.122.18.3"/>
    </dependentAssembly>
  </assemblyBinding>
</runtime>
<oracle.manageddataaccess.client>
  <version number="*">
    <settings>
      <setting name="TNS_ADMIN" value="C:\oracle\product\11.2.0\client\network\ADMIN"/>
    </settings>
    <dataSources>
    </dataSources>
  </version>
</oracle.manageddataaccess.client>
</configuration>
```

Imagen 21. Referenciar Bases de Datos: Definiendo la ruta del tnsnames.ora

Para DataSources:

```
<dataSources>
  <dataSource alias="EL8PROD"
descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=1mndbs09)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=el8prod)))" />
  <dataSource alias="EL8TEST"
descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=1mndbs05)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=el8test)))" />
</dataSources>
```

```

        <dataSource alias="EL8DESA"
descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=lmndbs05)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=el8desa)))" />
    </dataSources>

```

```

</system.data>
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <publisherPolicy apply="no" />
      <assemblyIdentity name="Oracle.ManagedDataAccess" publicKeyToken="89b483f429c47342" culture="neutral" />
      <bindingRedirect oldVersion="4.121.0.0 - 4.65535.65535.65535" newVersion="4.122.18.3" />
    </dependentAssembly>
  </assemblyBinding>
</runtime>
<oracle.manageddataaccess.client>
  <version number="*">
    <dataSources>
      <dataSource alias="EL8PROD" descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=lmndbs09)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=e18prod)))" />
      <dataSource alias="EL8TEST" descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=lmndbs05)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=e18test)))" />
      <dataSource alias="EL8DESA" descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=lmndbs05)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=e18desa)))" />
    </dataSources>
  </version>
</oracle.manageddataaccess.client>
</configuration>

```

Imagen 22. Referenciar Bases de Datos: Definiendo por DataSources

Nota:

Hay que tener en cuenta que cada sistema de Oracle tiene su propia configuración de datasources (Ejemplo: El fichero TNSNAMES.ora que suele estar localizado en la ruta indicada en la variable de entorno del equipo según la versión y el sistema de bases de datos instalado). Por lo tanto, para acceder a las bases de datos deberá hacer los ajustes de configuración necesarios según la necesidad de su sistema. Para el ejercicio asumiremos que se utilizarán los DataSources especificados en el archivo app.config.

La librería `EllipseCommonsClassLibrary` hace uso de su propia configuración para acceder a los DataSources, por lo que si se usa esta librería no es necesario adicionar los DataSources al app.config. Esta configuración crea una variable local de entorno para referenciar la ruta del archivo tnsnames.ora. De forma predeterminada, la ruta corresponde a la establecida en las políticas de TI de Cerrejón (C:\ORACLE\PRODUCT\11.2.0\CLIENT\NETWORK).

### 3.3 Agregar código de referencia de Oracle Data Manager

Para hacer uso de la librería se deberá adicionar el llamado en los namespaces de la clase que hará uso de ella.

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

Nota: Puede copiar el código del archivo `ExecuteQuery.txt` adjunto a este Manual para comprobar los resultados del proceso. Tenga presente que probablemente deba agregar referencias en el namespace adicional según lo indique el archivo adjunto.

## 4. AGREGAR REFERENCIAS A LIBRERÍAS COMUNES

Las librerías comunes son herramientas desarrolladas por diferentes grupos de desarrolladores, entre ellos la contratista de Hambings SAS, y sirven para facilitar el desarrollo de los AddIns al optimizar y automatizar muchas de las funciones comunes de gestión de hojas de cálculo de Excel, conexión a bases de datos, y acceso a los servicios de Ellipse.

Entre las clases más representativas de la librería común tenemos las siguientes:

**ExcelStyleCells:** con diversos métodos para manejar la lectura y escritura de las hojas de cálculo, manejo de estilo, creación dinámica de tablas, validación de datos de celda, entre otras opciones.

**EllipseFunctions:** con diversos métodos para manejar la conexión a las bases de datos y diferentes entornos de Ellipse, consultar queries, y acceso al servicio de screen service de Ellipse.

**Autenticación de Ellipse:** con los cuadros de diálogos requeridos para solicitar una autenticación con Ellipse y hacer los consumos SOAP respectivos.

**Otras Clases:** la librería cuenta con diferentes clases para el manejo de respuesta de los servicios SOAP, códigos de la tabla 010, superintendencias, grupos de trabajo, entre otros.

### 4.1 Vincular el proyecto EllipseCommonsClassLibrary

Desplegamos el [1] menú de opciones del proyecto sea desde el menú principal o haciendo clic derecho en el explorador de soluciones, y seleccionamos "[2] Agregar (Add) > [3] Referencia (Reference)"

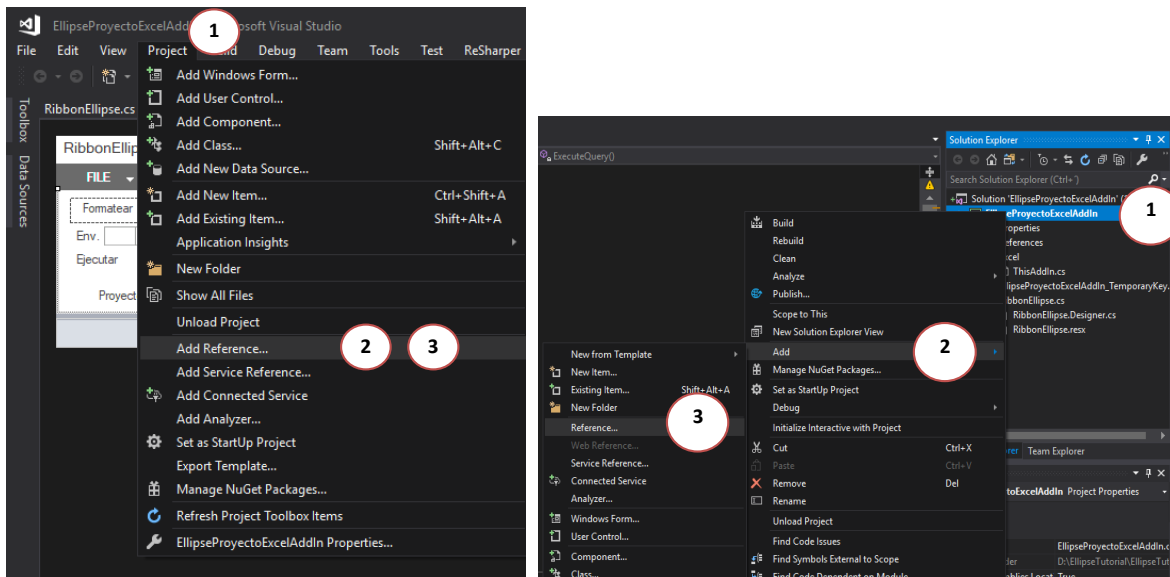


Imagen 23. Adicionar Librerías Comunes: Agregar Nueva Librería

En la barra lateral izquierda seleccionamos la opción de [4] Búsqueda (Browse) y nuevamente en el botón de [5] Búsqueda (Browse). Ubicamos la ruta donde se encuentra el archivo de librería con la versión correspondiente, seleccionamos [6] `EllipseCommonsClassLibrary.dll` y hacemos clic en el botón de [7] Añadir (Add).

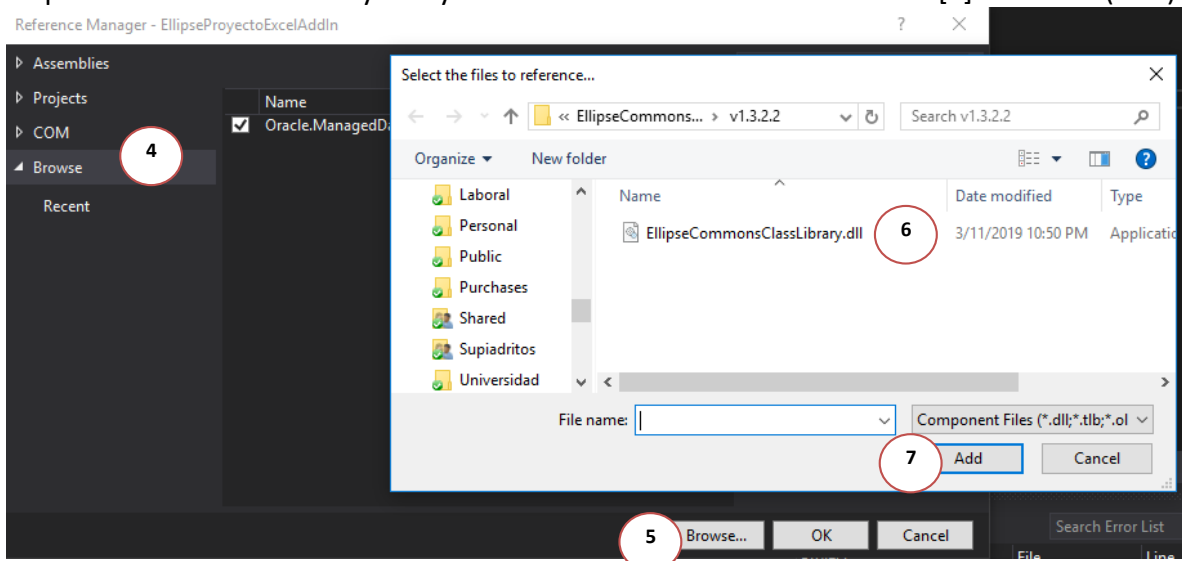


Imagen 24. Adicionar Librerías Comunes: Agregar `EllipseCommonsClassLibrary`

Nos aseguramos que el cuadro de selección [8] esté activo en el listado y finalizamos haciendo clic en el botón de [9] OK

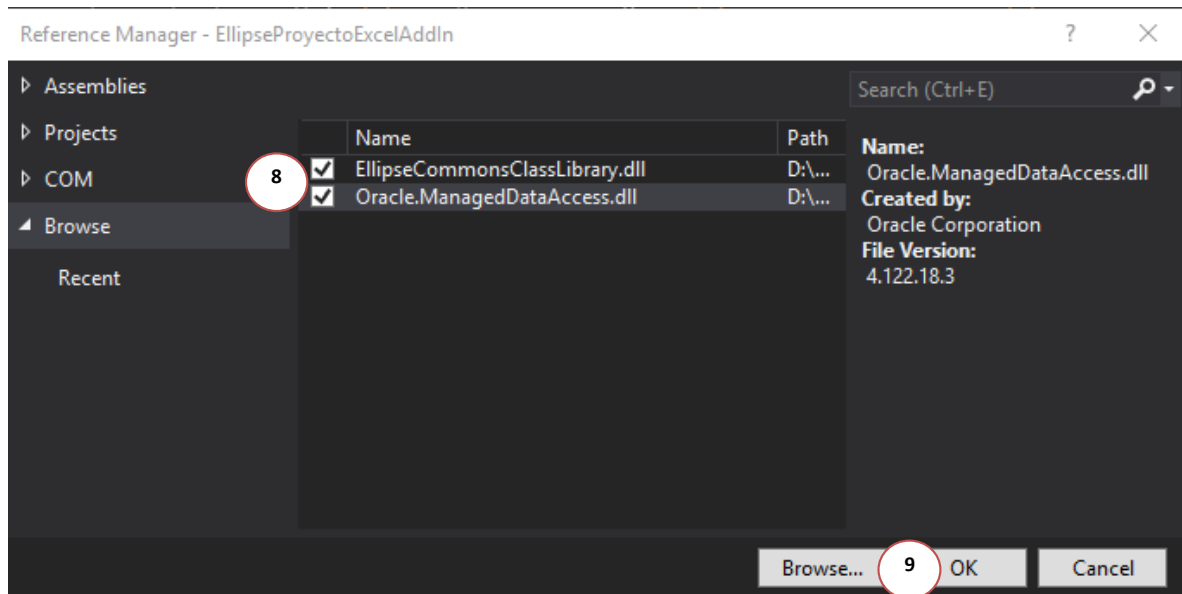


Imagen 25. Adicionar Librerías Comunes: Confirmar Librería Común

*Repetir el numeral para cada una de las librerías comunes que desee añadir al proyecto.*

## 4.2 Agregar código de referencias de Librerías Commons

Para hacer uso de la librería se deberá adicionar las siguientes sentencias.

### Namespaces:

```
using EllipseCommonsClassLibrary;
using EllipseCommonsClassLibrary.Classes;
using EllipseCommonsClassLibrary.Connections;
using Excel = Microsoft.Office.Interop.Excel;
```

### Definición de Variables:

```
ExcelStyleCells _cells;
EllipseFunctions _eFunctions = new EllipseFunctions();
private Excel.Application _excelApp;
```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using Microsoft.Office.Tools.Ribbon;
6  using EllipseCommonsClassLibrary;
7  using EllipseCommonsClassLibrary.Classes;
8  using EllipseCommonsClassLibrary.Connections;
9  using Excel = Microsoft.Office.Interop.Excel;
10
11 namespace EllipseProyectoExcelAddIn
12 {
13     public partial class RibbonEllipse
14     {
15         ExcelStyleCells _cells;
16         EllipseFunctions _eFunctions = new EllipseFunctions();
17         private Excel.Application _excelApp;
18
19         private void RibbonEllipse_Load(object sender, RibbonUIEventArgs e)
20         {
21         }
22     }
23 }
24
25

```

Imagen 26. Agregar código de referencia a Librería Commons

Adicionamos el siguiente código al método de iniciación de la Cinta de Opciones (Ribbon). Estas sentencias poblarán la información de la lista desplegable de entornos disponibles.

```

_excelApp = Globals.ThisAddIn.Application;

var environments = Environments.GetEnvironmentList();
foreach (var env in environments)
{
    var item = Factory.CreateRibbonDropDownItem();
    item.Label = env;
    drpEnviroment.Items.Add(item);
}

```

Nota: Según la configuración de la librería se hará referencia al archivo de entornos XML donde está disponible la información de servicios de Ellipse.

Ruta Predeterminada: [\\lmnoas02\SideLine\EllipsePopups\Ellipse8\](\\lmnoas02\SideLine\EllipsePopups\Ellipse8)

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

    var environments = Environments.GetEnvironmentList();
    foreach (var env in environments)
    {
        var item = Factory.CreateRibbonDropDownItem();
        item.Label = env;
        drpEnviroment.Items.Add(item);
    }
}
```

Imagen 27 Agregar código de referencia a Librería Commons: Código de Iniciación

Nota: Puede copiar el código del archivo ExecuteQueryCommons.txt adjunto a este Manual para comprobar los resultados del proceso. Tenga presente que probablemente deba agregar referencias en el namespace adicional según lo indique el archivo adjunto.

Nota: Si al agregar el código del punto anterior la librería *Microsoft.Office.Interop.Excel* no es encontrada deberá adicionarla manualmente.

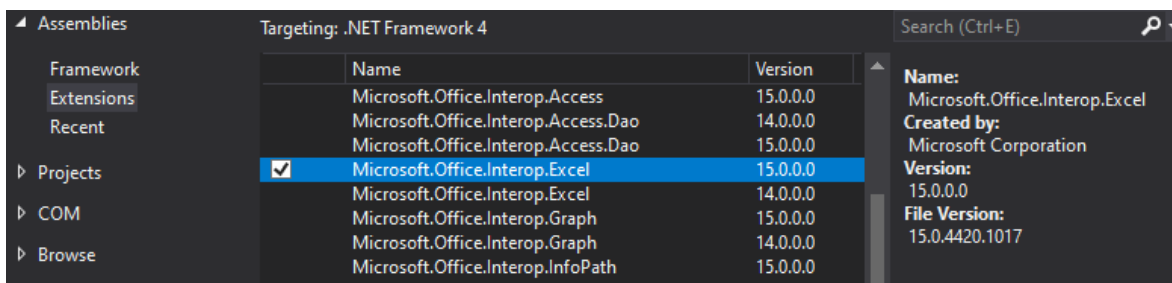


Imagen 28 Agregar referencia a librería Interop de Excel si esta no existe

## 5. AGREGAR CABECERA DE CONSUMO SOAP

### 5.1 Agregar referencia a librería de consumo SOAP

Desplegamos el [1] menú de opciones del proyecto sea desde el menú principal o haciendo clic derecho en el explorador de soluciones, y seleccionamos “[2] Agregar (Add) > [3] Referencia (Reference)”

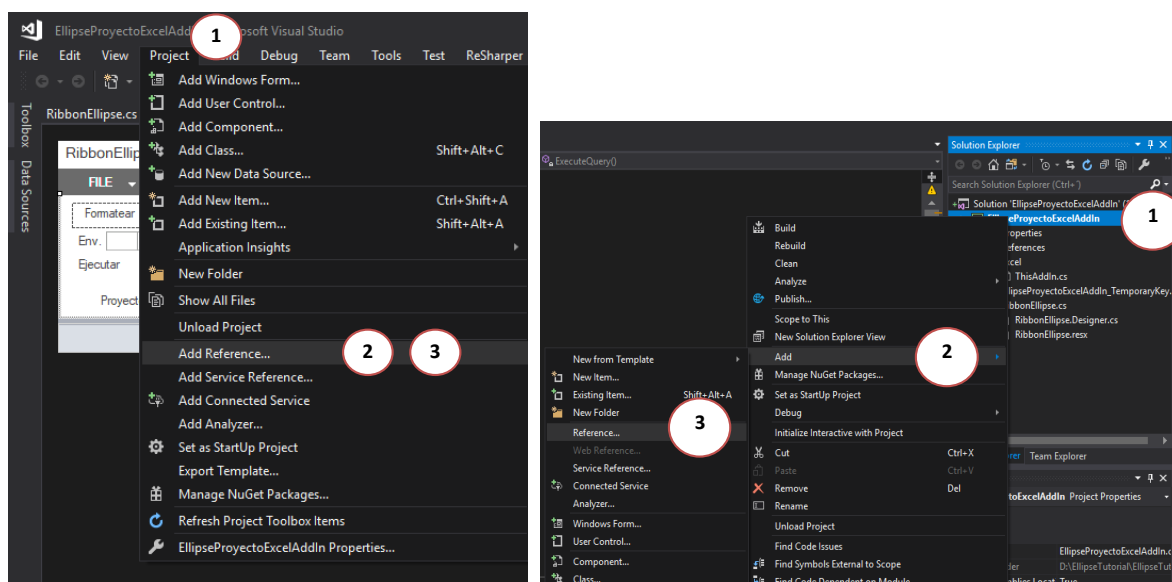


Imagen 29. Agregar Encabezado SOAP: Adicionar Referencia

En la barra lateral izquierda seleccionamos la opción de [4] Búsqueda (Browse) y nuevamente en el botón de [5] Búsqueda (Browse). Ubicamos la ruta donde se encuentra el archivo de librería con la versión correspondiente, seleccionamos [6] System.Web.Services.Ellipse.dll y hacemos clic en el botón de [7] Añadir (Add).



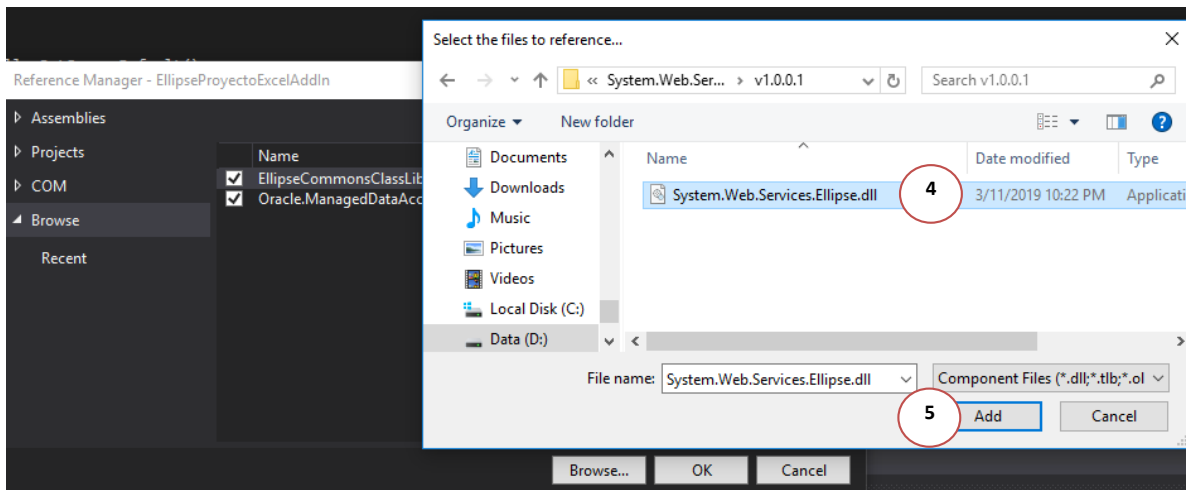


Imagen 30. Agregar Encabezado SOAP: Seleccionar librería

Nos aseguramos que el cuadro de selección [8] esté activo en el listado y finalizamos haciendo clic en el botón de [9] OK

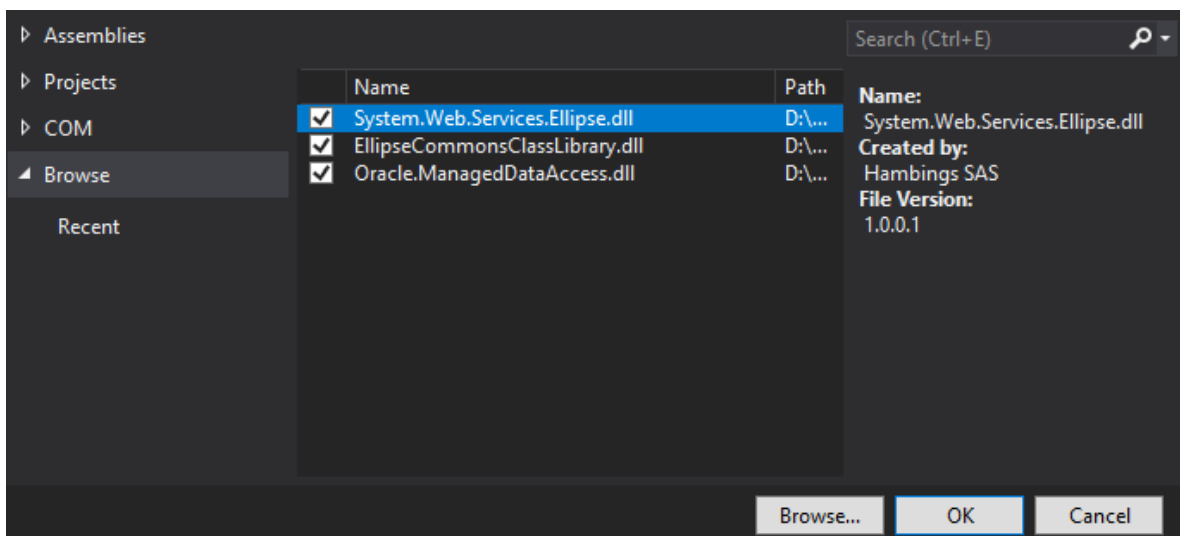


Imagen 31. Agregar Encabezado SOAP: Confirmar librería

## 5.2 Adicionar referencia a app.config

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

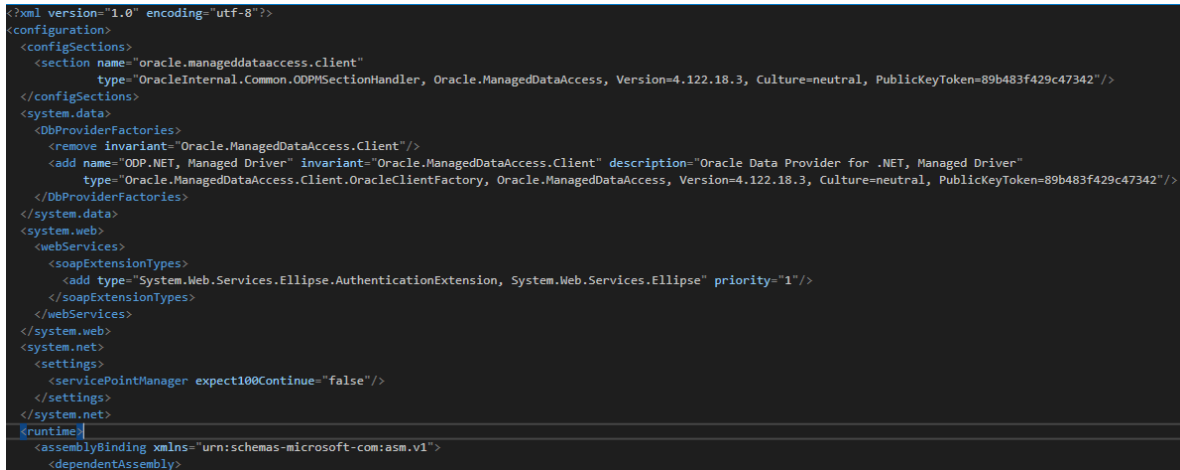
```
<configuration>
  <system.web>
    <webServices>
      <soapExtensionTypes>
        <add type="System.Web.Services.Ellipse.AuthenticationExtension,
System.Web.Services.Ellipse" priority="1"/>
      </soapExtensionTypes>
    </webServices>
  </system.web>
</configuration>
```

```

    <system.net>
      <settings>
        <servicePointManager expect100Continue="false"/>
      </settings>
    </system.net>
  </configuration>

```

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



```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="oracle.manageddataaccess.client"
      type="OracleInternal.Common.ODPMSectionHandler, Oracle.ManagedDataAccess, Version=4.122.18.3, Culture=neutral, PublicKeyToken=89b483f429c47342"/>
    </configSections>
  <system.data>
    <DbProviderFactories>
      <remove invariant="Oracle.ManagedDataAccess.Client"/>
      <add name="ODP.NET, Managed Driver" invariant="Oracle.ManagedDataAccess.Client" description="Oracle Data Provider for .NET, Managed Driver"
        type="Oracle.ManagedDataAccess.Client.OracleClientFactory, Oracle.ManagedDataAccess, Version=4.122.18.3, Culture=neutral, PublicKeyToken=89b483f429c47342"/>
    </DbProviderFactories>
  </system.data>
  <system.web>
    <webServices>
      <soapExtensionTypes>
        <add type="System.Web.Services.Ellipse.AuthenticationExtension, System.Web.Services.Ellipse" priority="1"/>
      </soapExtensionTypes>
    </webServices>
  </system.web>
  <system.net>
    <settings>
      <servicePointManager expect100Continue="false"/>
    </settings>
  </system.net>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>

```

Imagen 32. Agregar Encabezado SOAP: Adicionar información a app.config

### 5.3 Agregar código de referencia de encabezado SOAP en el proyecto

Se añade el siguiente código a la clase principal del desarrollo del proyecto en los namespaces

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

Antes de consumir cualquier servicio SOAP de Ellipse, deberá hacer uso de la siguiente instrucción, especificando el nombre de usuario, contraseña, distrito y posición, respectivamente. Este encabezado es enviado con todas las solicitudes que se realicen.

```
ClientConversation.authenticate(EllipseUser, EllipsePswd, EllipseDsct, EllipsePost);
```

Si desea hacer uso de las opciones de autenticación para login deberá utilizar el servicio provisto por Ellipse de AuthenticatorService. Este servicio se consume de la misma forma que cualquier otro servicio, lo cual veremos más adelante. La librería común tiene ya incorporada esta funcionalidad, y para usarla solo requiere las siguientes instrucciones.

En la definición de variables: `FormAuthenticate _frmAuth = new FormAuthenticate();`

Para las sentencias de ejecución:

```

_frmAuth.StartPosition = FormStartPosition.CenterScreen;
_frmAuth.SelectedEnvironment = drpEnviroment.SelectedItem.Label;
if (_frmAuth.ShowDialog() != DialogResult.OK) return;

```

Nota: Añada las instrucciones dadas a continuación para continuar la construcción del proyecto con las librerías commons y compare las opciones de desarrollo.

## 6. REFERENCIAR Y CONSUMIR SERVICIOS DE ELLIPSE

El listado de servicios disponibles para consumo por parte de Ellipse puede encontrarlos en la siguiente dirección web:

<http://ews-el8prod.lmnerp01.cerrejon.com/ews/services>

### 6.1 Agregar Web Service de Ellipse

Desplegamos el [1] menú de opciones del proyecto sea desde el menú principal o haciendo clic derecho en el explorador de soluciones, y seleccionamos "[2] Agregar (Add) > [3] Referencia de Servicio (Service Reference)"

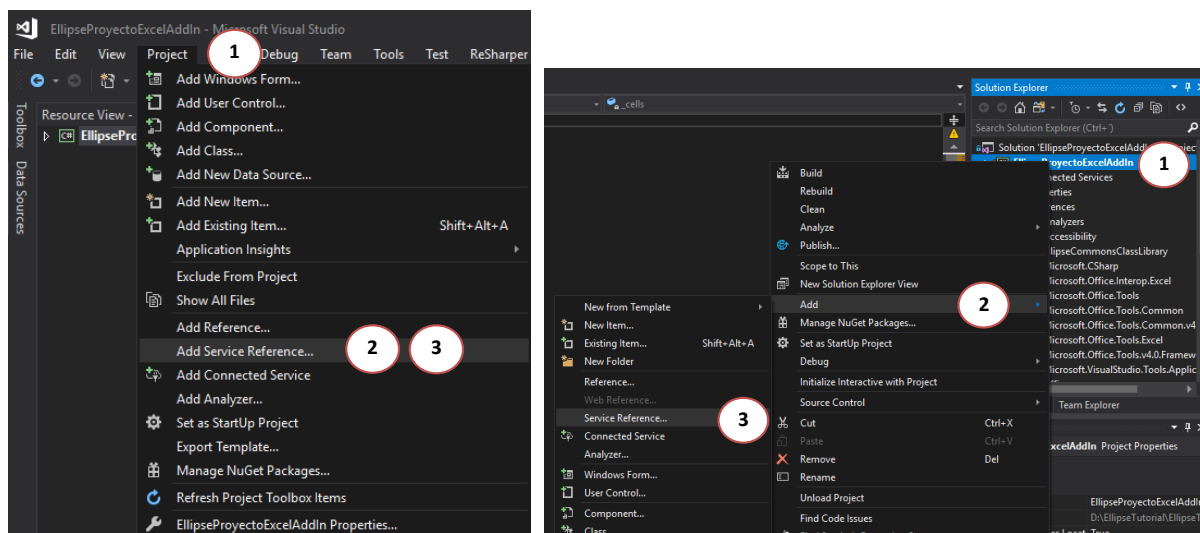


Imagen 3: Agregar Referencias Web: Agregar referencia a servicio

[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"

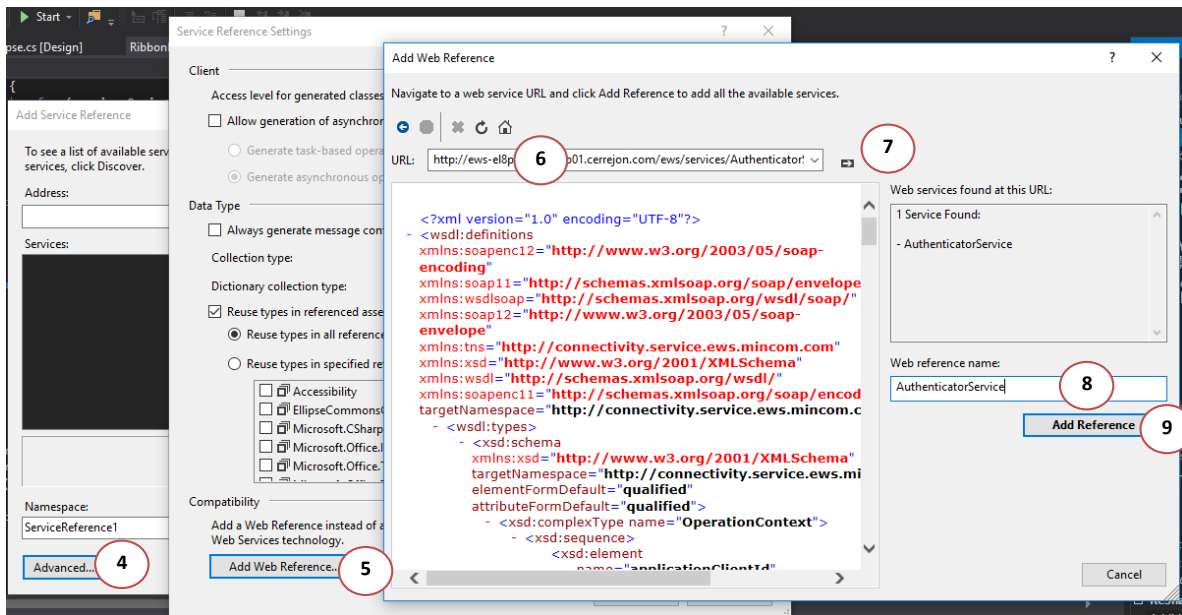


Imagen 33. Agregar Referencia a Servicio Web: Especificar Web Service de Ellipse

Una vez agregado el servicio, podemos confirmarlo en el listado de servicios referenciados del proyecto, en el explorador de soluciones.

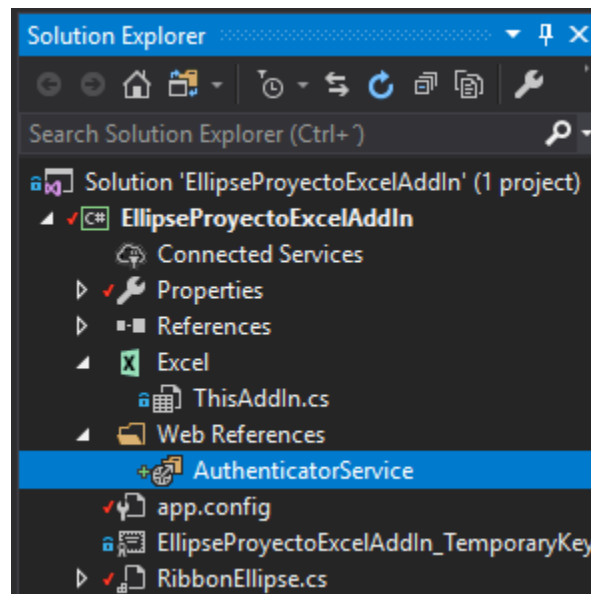


Imagen 34. Agregar Referencia a Servicio Web: Comprobar Servicio

Nota: Si no va a adicionar ningún servicio, puede que deba agregar de forma manual la referencia de System.Web.Services (este servicio se agrega automáticamente cuando se agrega cualquier servicio web) por Proyecto (Project) > Agregar (Add) > Referencia (Reference) > Assemblies > System.Web.Services.

Después de añadir el servicio deberá referenciarlo en los namespaces del proyecto. Puede hacerlo directamente o mediante la asignación de un alias que facilite el nombre de referencia para sus clases. Para nuestro ejemplo hicimos uso del `AuthenticatorService`.

Con alias:

```
using Authenticator = EllipseProyectoExcelAddIn.AuthenticatorService;
```

Sin alias:

```
using EllipseProyectoExcelAddIn.AuthenticatorService;
```

Importante: Esto es recomendado porque si va a trabajar con más de un servicio al tiempo, puede encontrar coincidencias de clases entre estos. El caso más común es con el contexto de operación. Cada servicio tiene una clase de `OperationContext`, y podrá referenciar e instanciar el correspondiente para cada servicio con mayor claridad si asigna un alias a cada servicio.

Repita este proceso para cada uno de los servicios que desea añadir a su proyecto.

Nota: Tenga presente que al referenciar el servicio lo que se está haciendo es una copia a disco del wsdl expuesto por Ellipse. Este wsdl tiene la información de la estructura, métodos y atributos requeridos para cada una de las operaciones del servicio. Durante las actualizaciones del sistema de Ellipse esta estructura puede cambiar. Por lo que si observa un problema con el AddIn después de una actualización, actualice la información del servicio (eliminando el servicio y añadiéndolo nuevamente) y verifique que no haya errores en el código producto de los cambios.

## 6.2 Uso de un Web Service

Los Servicios Web de Ellipse tienen cuatro componentes esenciales que deben ser entendidos claramente:

- Servicio (service): corresponde al objeto que tiene la estructura principal del servicio. En este objeto encontramos los métodos para las operaciones que requiere el servicio.
- Contexto de Operación (operation context): se usa para acceder a los canales de devolución de llamada en servicios duales, para almacenar datos de estado adicionales en partes de las operaciones, y para acceder a encabezados de mensajes entrantes y propiedades, así como para agregar encabezados de mensajes salientes y propiedades. Simplificando, podría entenderse como una sesión que garantiza la comunicación entre el AddIn y Ellipse.
- Solicitud (request): usualmente es el objeto o los objetos que tiene todos los parámetros y valores necesarios que deben ser enviados por algún método del servicio. Algunas operaciones no requieren directamente un objeto de solicitud.
- Respuesta (reply): es el objeto con la respuesta de la solicitud enviada. Este puede contener los valores de respuesta, mensajes de estado, errores, entre otros. Algunas operaciones no devuelven una respuesta.

Nota: Para el manejo de los objetos de las solicitudes y respuestas de Ellipse hay que tener cuidado en la información que se envía para cada ítem. Ellipse tiene una diferenciación en cómo tratar un ítem nulo, vacío, con valor, decimal y booleano. Estos comportamientos son debido a la naturaleza de la tipología de los datos y de las aplicaciones que los utilizan.

- Nulo: si un ítem es enviado como nulo en la solicitud, el servicio lo ignorará. Es decir, el servicio no realizará ninguna modificación al valor que se tenga del servicio.
- Vacío: hay que tener especial cuidado al momento de la lectura de celdas vacías en Excel. Una celda vacía será leída como un valor nulo a menos que se haga explícita la asignación en el código de un texto vacío ("").
- Booleanos: algunos valores booleanos van acompañados de una variable adicional que especifica si este valor debe leerse (Specified), por lo que para su asignación se debe indicar la variable specified como true y al valor a asignar respectivamente en el nombre de la variable.
- Cantidades: todos los valores de cantidades que trabaja Ellipse corresponde a una tipología decimal. Estos valores van acompañados de su respectiva variable específica (Specified), por lo que para su asignación se debe indicar la variable specified como true y al valor a asignar respectivamente en el nombre de la variable. Tenga especial cuidado que los valores decimales no pueden ser vacíos ni nulos, por lo que su eliminación debe ser una asignación a cero.

### 6.2.1 Servicio de Autenticación:

A continuación presentamos un ejemplo de uso del servicio de autenticación. Si no se produce un error, la autenticación se habrá realizado exitosamente. Si va a utilizar la librería commons, no necesita referenciar el servicio porque este ya está dentro de la librería. Si aún no ha agregado el servicio siga el proceso del numeral Agregar Web Service de Ellipse para el servicio de Authentication Service.

```
//creación del servicio
var service = new Authenticator.AuthenticatorService();
service.Url = @"http://ews-el8prod.lmnerp01.cerrejon.com/ews/services/AuthenticatorService";

//creación del contexto de operación
var opContext = new Authenticator.OperationContext
{
    district = "ICOR",
    position = ""
};

//Encabezado de consumo
ClientConversation.authenticate("NombreUsuario", "Password", "Distrito", "Posición");
//Recuerde que el encabezado SOAP es enviado con todas las solicitudes
//Envío de Solicitud
service.authenticate(opContext);
```

Como se puede observar en el código, la solicitud solo requiere el contexto de operación y la información enviada en el encabezado para realizarse. Además, no retorno ningún objeto de respuesta. Su comportamiento es que de no recibir respuesta se produce la autenticación y de recibirse un error de excepción es porque hay un problema con la autenticación.

Con la librería commons solo necesitará hacer uso de las siguientes tres instrucciones.

```
_frmAuth.StartPosition = FormStartPosition.CenterScreen;
_frmAuth.SelectedEnvironment = drpEnvironment.SelectedItem.Label;
if (_frmAuth.ShowDialog() != DialogResult.OK) return;
```

Nota: Puede encontrar un código un poco más elaborado en los archivos adjuntos a este manual para ambos casos.

### 6.2.2 Screen Service (MSO)

Todos los programas de Ellipse que su nombre inicia con MSO hacen uso del servicio de pantalla Screen Service. Por lo que para comunicarse con ellos se debe referenciar este servicio (siga el proceso del numeral Agregar Web Service de Ellipse para el servicio de Screen Service). Si va a utilizar la librería commons, no necesita referenciar el servicio porque este ya está dentro de la librería, solo adicionar su alias. Utilice el alias correspondiente según haga uso de la librería commons o no.

Sin Commons:

```
using Screen = EllipseProyectoExcelAddIn.ScreenService;
```

Con Commons:

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

Para el proceso descrito a continuación se asume que ya realizó la autenticación requerida y solo necesita enviar el encabezado.

Se crea el servicio instanciando el Screen Service y se define la URL de referencia del servicio expuesto por Ellipse.

```
//Proceso del Servicio Screen
var service = new Screen.ScreenService();
var urlService = _eFunctions.GetServicesUrl(drpEnvironment.SelectedItem.Label);
service.Url = urlService + "/ScreenService";
```

Se crea el contexto de operación utilizando el alias dado al servicio.

```
//Instanciar el Contexto de Operación
var opContext = new Screen.OperationContext
{
    district = _frmAuth.EllipseDscst,
    position = _frmAuth.EllipsePost
};
```

Se crea el encabezado de consumo y se envía la solicitud de la pantalla que se desea utilizar (Ej. MSO435). La respuesta de esta acción se almacena en la variable reply.

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

//Solicitud 1 y Respuesta 1
var reply = service.executeScreen(opContext, "MS0435");
```

```
//Proceso del Servicio Screen
var service = new Screen.ScreenService();
var urlService = _eFunctions.GetServicesUrl(drpEnviroment.SelectedItem.Label);
service.Url = urlService + "/ScreenService";

//Instanciar el Contexto de Operación
var opContext = new Screen.OperationContext
{
    district = _frmAuth.EllipseDsct,
    position = _frmAuth.EllipsePost
};

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

//Solicitud 1 y Respuesta 1
var reply = service.executeScreen(opContext, "MS0435");
```

Imagen 35. Uso del Screen Service: Solicitud de Pantalla

Se analiza el objeto de respuesta y se determina que la respuesta es la deseada. Si no es la deseada se deberá realizar alguna acción pertinente que indique esto. (Ej. Validamos que esté en el programa y pantalla que deseamos (Ej. MSM435A que es la pantalla inicial cuando se ejecuta el programa)).

```
//validamos el ingreso al programa
if (reply.mapName != "MSM435A")
    throw new Exception ("ERROR:" + "No se pudo establecer comunicación con el servicio");
```

Como la respuesta es satisfactoria, empezamos a preparar nuestra nueva solicitud. Las pantallas hacen uso de un tipo de objeto de vector con los nombres y valores de los campos que se desean enviar. Si un campo no se envía, la pantalla asumirá el valor predeterminado (previo mostrado en la respuesta de la pantalla) por lo que solo es necesario enviar los valores que se desean modificar.

La librería commons hace uso de una clase ArrayScreenNameValue para facilitar el envío de estos datos. Para el ejemplo enviamos los campos OPTION1I, MODEL\_CODE1I, STAT\_DATE1I y SHIFT1I con sus respectivos valores.

```
//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", "CÓDIGO MODELO");
```



```
arrayFields.Add("STAT_DATE1I", "FECHA MODELO");
arrayFields.Add("SHIFT1I", "TURNO MODELO");
```

Completamos nuestro objeto de solicitud (request) y enviamos la acción mediante el método de submit, adjuntando nuestro objeto de solicitud.

```
//Solicitud 2
var request = new Screen.ScreenSubmitRequestDTO();
request.screenFields = arrayFields.ToArray();
request.screenKey = "1";
```

```
//Respuesta 2
reply = service.submit
```

```
//validamos el ingreso al programa
if (reply.mapName != "MSM435A")
    throw new Exception ("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", "CÓDIGO MODELO");
arrayFields.Add("STAT_DATE1I", "FECHA MODELO");
arrayFields.Add("SHIFT1I", "TURNO MODELO");

//Solicitud 2
var request = new Screen.ScreenSubmitRequestDTO();
request.screenFields = arrayFields.ToArray();
request.screenKey = "1";

//Respuesta 2
reply = service.submit(opContext, request);
```

Imagen 36. Uso de Screen Service: Envío de solicitud

Deberá repetir este proceso para tantas pantallas como tenga el programa MSO. Después de cada solicitud deberá analizar los resultados del objeto de respuesta y generar una acción correspondiente a lo deseado.

Cada pantalla tiene su propia forma de enviar la respuesta de errores, advertencias, confirmación, entre otros. Por lo que dependerá del programa y el análisis realizado el saber cómo gestionar las acciones para este. Algunas de las repuestas comunes contienen los siguientes comportamientos:

Análisis por existencia de respuesta y nombre del programa que envía la respuesta.

```
if (reply == null || reply.mapName == "MSM435B")
    throw new Exception("ERROR:" + "Se ha producido un error al intentar enviar la solicitud");
```

La respuesta tiene un error o una advertencia

```
if(_eFunctions.CheckReplyError(reply) || _eFunctions.CheckReplyWarning(reply))
    throw new Exception("ERROR:" + "Se ha producido un error al intentar enviar la solicitud");
```

La respuesta pide confirmación

```
if(reply.functionKeys == "XMIT-Confirm")
    reply = service.submit(opContext, request);
```

```
//manejo de opciones para el tipo de respuesta
if (reply == null || reply.mapName == "MSM435B")
    throw new Exception("ERROR:" + "Se ha producido un error al intentar enviar la solicitud");

if(_eFunctions.CheckReplyError(reply) || _eFunctions.CheckReplyWarning(reply))
    throw new Exception("ERROR:" + "Se ha producido un error al intentar enviar la solicitud");

//Confirmación
if(reply.functionKeys == "XMIT-Confirm")
    reply = service.submit(opContext, request);
```

Imagen 37. Uso de Screen Service: Manejo de Respuestas

Para obtener o leer algún elemento de la pantalla puede usar las siguientes opciones de la librería commons. De hacerlo directamente deberá leer el arreglo del objeto de respuesta (reply.screenFields) y determinar el objeto con el que desea trabajar.

```
//si necesitas obtener los campos del reply y trabajar con ellos
var replyFields = new ArrayScreenNameValue(reply.screenFields);
var woProject = replyFields.GetField("WO_PROJ111").value.Equals("");
```

```
//si necesitas obtener los campos del reply y trabajar con ellos
var replyFields = new ArrayScreenNameValue(reply.screenFields);
var woProject = replyFields.GetField("WO_PROJ111").value.Equals("");
```

Imagen 38. Uso de Screen Service: Manejo de Ítems

### 6.2.3 Web Service General (MSE)

Todos los programas de Eclipse que su nombre inicia con MSE (antiguos MSQ) hacen uso de su propio servicio de acuerdo a su nombre. Por lo que para comunicarse con ellos se debe referenciar este servicio (siga el proceso del numeral Agregar Web Service de Eclipse para el servicio correspondiente).

Tenga presente que hay programas MSE que usan más de un servicio, por lo que deberá identificar cuáles son estos servicios y referenciarlos para el desarrollo.

Para el ejemplo, haremos uso del Servicio de Ordenes de Trabajo WorkOrderService. Para el proceso descrito a continuación se asume que ya realizó la autenticación requerida y solo necesita enviar el encabezado.

Se crea el servicio instanciando el servicio y se define la URL de referencia del servicio expuesto por Eclipse.

```
//Creación del Servicio
var service = new WorkOrderService.WorkOrderService();
var urlService = _eFunctions.GetServicesUrl(drpEnvironment.SelectedItem.Label);
```

```
service.Url = urlService + "/WorkOrderService";
```

Se crea el contexto de operación para las órdenes de trabajo y el encabezado de consumo

```
//Instanciar el Contexto de Operación
var opContext = new WorkOrderService.OperationContext
{
    district = _frmAuth.EllipseDsct,
    position = _frmAuth.EllipsePost
};
```

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

```
//Creación del Servicio
var service = new WorkOrderService.WorkOrderService();
var urlService = _eFunctions.GetServicesUrl(drpEnviroment.SelectedItem.Label);
service.Url = urlService + "/WorkOrderService";

//Instanciar el Contexto de Operación
var opContext = new WorkOrderService.OperationContext
{
    district = _frmAuth.EllipseDsct,
    position = _frmAuth.EllipsePost
};

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

Imagen 39. Uso del Servicio de Órdenes: Creación

Para los servicios generales se deben crear objetos de tipo DTO (Data Transfer Object) que tienen la información pertinente sea de una solicitud o de una respuesta. Cada método del servicio requiere de un tipo de objeto específico, por lo que se debe analizar el método primero antes de definir el objeto a utilizar. Para la solicitud de creación haremos uso del objeto WorkOrderServiceCreateRequestDTO.

```
//Se cargan los parámetros de la solicitud
var request = new WorkOrderServiceCreateRequestDTO();
request.districtCode = "ICOR";
request.workGroup = "MTOLOC";
request.workOrderDesc = "ORDEN DE PRUEBA";
request.workOrderType = "CO";
request.maintenanceType = "CO";
request.equipmentNo = "1000016";
```

Finalmente enviamos la solicitud y recibiremos una respuesta con un tipo de objeto DTO de respuesta. Del objeto recibido obtendremos el número de la nueva orden de trabajo que se ha creado, la cual escribiremos en la celda inicial.

```
//se envía la acción
var reply = service.create(opContext, request);

//se analiza la respuesta y se hacen las acciones pertinentes
```

```
_cells.GetCell(1, 1).Value2 = reply.workOrder.prefix + reply.workOrder.no;
```

```
//Se cargan los parámetros de la solicitud
var request = new WorkOrderServiceCreateRequestDTO();
request.districtCode = "ICOR";
request.workGroup = "MTOLOC";
request.workOrderDesc = "ORDEN DE PRUEBA";
request.workOrderType = "CO";
request.maintenanceType = "CO";
request.equipmentNo = "1000016";

//se envía la acción
var reply = service.create(opContext, request);

//se analiza la respuesta y se hacen las acciones pertinentes
_cells.GetCell(1, 1).Value2 = reply.workOrder.prefix + reply.workOrder.no;
```

Imagen 40. Uso del Servicio de Órdenes: Envío de solicitud

## 7. OBSERVACIONES ADICIONALES DE DESARROLLO

### 7.1 Ejecución de Querys

Sin hacer uso de la librería de Commons

```
OracleConnection sqlOracleConn = null;
try
{
    _excelApp.Cursor = Excel.XlMousePointer.xlWait;
    var excelSheet = (Excel.Worksheet)_excelApp.ActiveWorkbook.ActiveSheet;

    var sqlQuery = @"SELECT WORK_ORDER FROM ELLIPSE.MSF620 WO WHERE" +
        @" WO.RAISED_DATE = '20190124' AND WO.WORK_GROUP = 'MTOLOC'";

    var connectionString = "Data Source=EL8PROD;User ID=SIGCON;Password=ventyx; Connection Timeout=30; Pooling=true";

    sqlOracleConn = new OracleConnection(connectionString);
    var sqlOracleComm = new OracleCommand();

    if (sqlOracleConn.State != ConnectionState.Open)
        sqlOracleConn.Open();
    sqlOracleComm.Connection = sqlOracleConn;
    sqlOracleComm.CommandText = sqlQuery;

    var dataReader = sqlOracleComm.ExecuteReader();

    if (dataReader == null)
        return;

    //Cargo el encabezado de la tabla y doy formato
    for (var i = 0; i < dataReader.FieldCount; i++)
    {
        var cell = (Excel.Range)excelSheet.Cells[1, i + 1];
        cell.Value2 = "" + dataReader.GetName(i);
    }

    //carga los datos
    if (dataReader.IsClosed || !dataReader.HasRows) return;

    var currentRow = 2;
    while (dataReader.Read())
    {
        for (var i = 0; i < dataReader.FieldCount; i++)
        {
            var cell = (Excel.Range)excelSheet.Cells[currentRow, i + 1];
            cell.Value2 = "" + dataReader[i].ToString().Trim();
        }

        currentRow++;
    }
}
catch (Exception ex)
{
    MessageBox.Show(@"Se ha producido un error. " + ex.Message);
}
finally
{
    if (sqlOracleConn != null && sqlOracleConn.State != ConnectionState.Closed)
        sqlOracleConn.Close();
    _excelApp.Cursor = Excel.XlMousePointer.xlDefault;
}
```

Imagen 41. Ejecutar Query sin Librería Commons

Haciendo uso de la librería `EllipseCommonsClassLibrary`

```
try
{
    if (_cells == null)
        _cells = new ExcelStyleCells(_excelApp);
    _cells.SetCursorWait();

    var titleRow = 1;
    var sqlQuery = @"SELECT WORK_ORDER FROM ELLIPSE.MSF620 WO" +
        @" WHERE WO.RAISED_DATE = '20190124' AND WO.WORK_GROUP = 'MTOLOC'";
    var tableName = "table";
    _eFunctions.SetDBSettings(drpEnviroment.SelectedItem.Label);
    var dataReader = _eFunctions.GetQueryResult(sqlQuery);

    if (dataReader == null)
        return;

    //Cargo el encabezado de la tabla y doy formato
    for (var i = 0; i < dataReader.FieldCount; i++)
        _cells.GetCell(i + 1, titleRow).Value2 = "" + dataReader.GetName(i);

    _cells.FormatAsTable(_cells.GetRange(1, titleRow, dataReader.FieldCount, titleRow + 1), tableName);

    //cargo los datos
    if (dataReader.IsClosed || !dataReader.HasRows) return;

    var currentRow = titleRow + 1;
    while (dataReader.Read())
    {
        for (var i = 0; i < dataReader.FieldCount; i++)
            _cells.GetCell(i + 1, currentRow).Value2 = "" + dataReader[i].ToString().Trim();
        currentRow++;
    }
}
catch (Exception ex)
{
    Debugger.LogError("RibbonEllipse:GetQueryResult()", "\n\rMessage:" + ex.Message +
        "\n\rSource:" + ex.Source + "\n\rStackTrace:" + ex.StackTrace);
    MessageBox.Show(@"Se ha producido un error. " + ex.Message);
}
finally
{
    if (_cells != null) _cells.SetCursorDefault();
    _eFunctions.CloseConnection();
}
```

Imagen 42. Ejecutar Query con librería Commons

### 7.3. Gestión de Códigos con Descripción (Tabla 010)

Para la gestión de los códigos de la tabla 010 podrá simplemente ejecutar la instrucción y obtendrá el listado de códigos especificados según el tipo de tabla. La primera opción traerá solamente los códigos activos, mientras que la segunda traerá todos los códigos disponibles.

```
var itemListActiveOnly = _eFunctions.GetItemCodes("SC");
var itemListCompleterList = _eFunctions.GetItemCodes("SC", false);
```

Imagen 43. Gestión de Códigos con Descripción: Tabla 010

Cada ítem de este listado de códigos tiene los atributos propios de la tabla 010, es decir: código (code), descripción (description), tipo de tabla (table\_type), registro asociado (assoc\_rec) y si está activo (active).

También podrá hacer uso de la clase estática de utilidades `EllipseCommonsClassLibrary.Utilities` que le permitirá una rápida gestión para obtener los códigos cuando estos tienen la descripción añadida.

```
var code = MyUtilities.GetCodeKey("XX - Descripción");
var value = MyUtilities.GetCodeValue("XX - Descripción");
var codeValuePair = MyUtilities.GetCodeKeyValue("XX - Descripción");
var codeKey = codeValuePair.Key;
var codeValue = codeValuePair.Value;
```

Imagen 44. Gestión de Códigos con Descripción: Opciones

Para cada uno de estos casos obtendremos los siguientes resultados.

- `code = "XX"`
- `value = "Descripción"`
- `codeKey = "XX"`
- `codeValue = "Descripción"`

## 7.4 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";
```

Nota: 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 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*

Los parámetros de ingreso corresponden a la siguiente información:

- 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

A continuación algunos ejemplos:

- Ejemplo con listas de códigos de la tabla 010

```
//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 columna 2 de la hoja de validación y la validación a la celda respectiva (B15)
_cells.SetValidationList(_cells.GetCell(15, 2), compCodeList, ValidationSheetName, 2);
```

Imagen 45. Validación de Celda con Códigos de Tabla 010

- Ejemplo con las constantes de grupos de trabajo y distritos

```
//EJEMPLO CON DISTRITO Y GRUPOS DE TRABAJO
//asigno los datos a la columna 1 de la hoja de validación y la validación a la celda respectiva (B3)
_cells.SetValidationList(_cells.GetCell("B3"), Districts.GetDistrictList(), ValidationSheetName, 1);
//asigno los datos a la columna 2 de la hoja de validación y la validación a la celda respectiva (B4)
_cells.SetValidationList(_cells.GetCell("B4"), Groups.GetWorkGrouplist().Select(g => g.Name).ToList(),
    ValidationSheetName, 2);
```

Imagen 46. Validación de Celda con Códigos de Distrito y Grupos de Trabajo

- Ejemplo con listas dinámicas

```
//EJEMPLO CON LISTAS CORTAS DINÁMICAS
_cells.SetValidationList(_cells.GetCell(8, 2), new List<string> { "P - Project", "W - WorkOrder" },
    ValidationSheetName, 1);
```

Imagen 47. Validación de Celda con Listas Dinámicas

- Ejemplo con datos ya existentes en la hoja de validación

```
//EJEMPLO CON DATOS YA EXISTENTE
//asigno los datos a la columna 1 de la hoja de validación y la validación a la celda respectiva (B3)
_cells.SetValidationList(_cells.GetCell("B3"), Districts.GetDistrictList(), ValidationSheetName, 1);
//hago uso de los datos de la columna 1 de la hoja de validación y para la celda respectiva (B3)
_cells.SetValidationList(_cells.GetCell("B4"), ValidationSheetName, 1);
```

Imagen 48. Validación de Celda con Datos Existentes

## 7.5 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.

Si hace uso de la instancia global, deberá corroborar que haya sido inicializada en cada método que use. Puede hacer uso de cualquiera de las opciones de inicialización más adelante descritas.

```
if (_cells == null)
    _cells = new ExcelStyleCells(_excelApp);
```

- 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.

- 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.

## 7.6 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 incontables 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);
```

Si el método tiene parámetros puede iniciarlo de la siguiente forma

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

Finalmente, invocamos la ejecución del hijo donde se requiera.

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

Nota: Se recomienda 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);  
}
```

Nota: Se recomienda 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.SelectedEnvironment = drpEnvironment.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 49. 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 50. Detención forzosa de un hilo de proceso

Nota: 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

## 7.7 Control de Errores, Debugging y Configuración

Para el control de errores lo mejor es organizar los procesos mediante uso de excepciones cuando se presente algún error, y posteriormente controlar la excepción generada. Tenga presente que puede generar una excepción de forma manual después de que se presenta alguna condición con la siguiente instrucción.

```

if (condicion)
    throw new Exception ("ERROR: No se pudo establecer comunicación con el servicio");

```

El tipo de excepción puede ser especificada, desde `NullReferenceException`, `DivideByZeroException`, `AccessViolationException`, entre otras. Esto le permitirá hacer un control detallado según se requiera de las excepciones.

Para el debuggin, la librería de Commons viene con una clase de Debugging que ayuda con la gestión de los logs. De forma predeterminada los logs que se generen por esta clase serán escritos en el directorio C:\Ellipse.

La clase viene con los valores DebugErros, DebugWarnings y DebugQueries desactivados. Esto solo significa que cuando se produzca un error o una advertencia no se producirá ningún mensaje de error y que dependerá de cómo el desarrollador controle la excepción.

Mediante la instrucción de `Debugger.LogError`, `Debugger.LogWarning` y `Debugger.LogQuery` el desarrollador podrá escribir el log respectivo.

Ejemplo: Después de una excepción, se controla el error y se escribe el log.

```
catch (Exception ex)
{
    _cells.GetCell(1, i).Style = StyleConstants.Error;
    _cells.GetCell(ResultColumn02, i).Style = StyleConstants.Error;
    _cells.GetCell(ResultColumn02, i).Value = "ERROR: " + ex.Message;
    Debugger.LogError("RibbonEllipse.cs:ExecuteTaskActions()", ex.Message);
}
```

Para acceder a los ajustes de debugging y otras configuraciones, ejecute la ventana de diálogo acerca de haciendo clic en el ícono de [?] en el AddIn. Posteriormente, haga clic múltiples veces en el título del AddIn

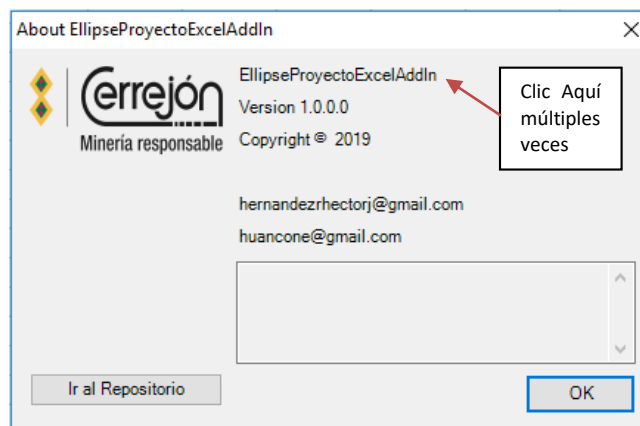


Imagen 51. Ventana de Diálogo Acerca de

Nota: Para estos ajustes deberá tener la ejecución de la instrucción de la ventana Acerca de en algún control. Se recomienda usar el estándar del botón con interrogación.

Esto desplegará la ventana con las opciones de configuración. Desde esta ventana podrá cambiar las opciones de la clase Debugger en tiempo de ejecución sin recurrir al código de desarrollo.

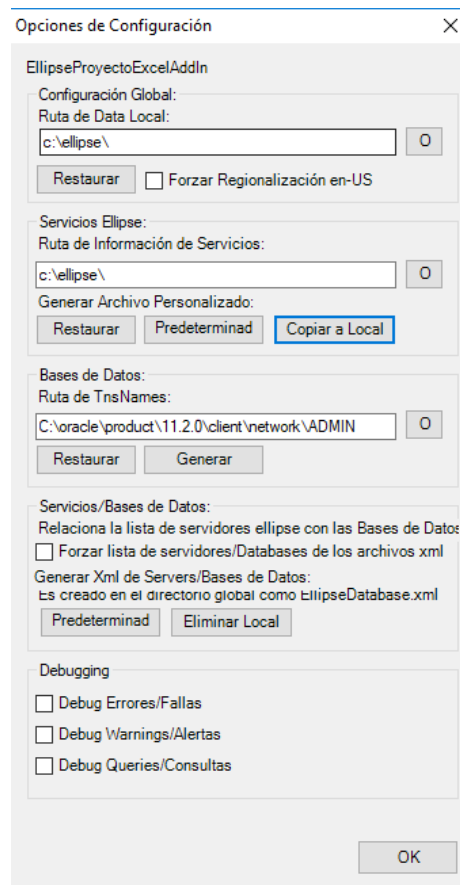


Imagen 52. Ventana de Configuración del AddIn

## 8. CONTROL DE VERSIONES Y PUBLICACIÓN DEL PROYECTO

### 8.1 Acerca de

La librería commons viene con una ventana de Acerca de con la información de desarrollo que se gestione en las opciones del proyecto. Para invocarla desde cualquier control es suficiente con ejecutar la siguiente instrucción.

```
new AboutBoxExcelAddIn().ShowDialog();
```

Esta instrucción tiene una sobrecarga para indicar el nombre de hasta dos desarrolladores del proyecto. Si no se usa la sobrecarga, los nombres mostrados corresponderán a los desarrolladores de la librería Commons.

```
new AboutBoxExcelAddIn("Desarrollador 1", "Desarrollador 2").ShowDialog();
```

## 8.2 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 [1] Proyecto (Project) y seleccionamos la opción de [2] Propiedades (Properties) o haciendo clic derecho sobre nuestro proyecto y la opción de propiedades.

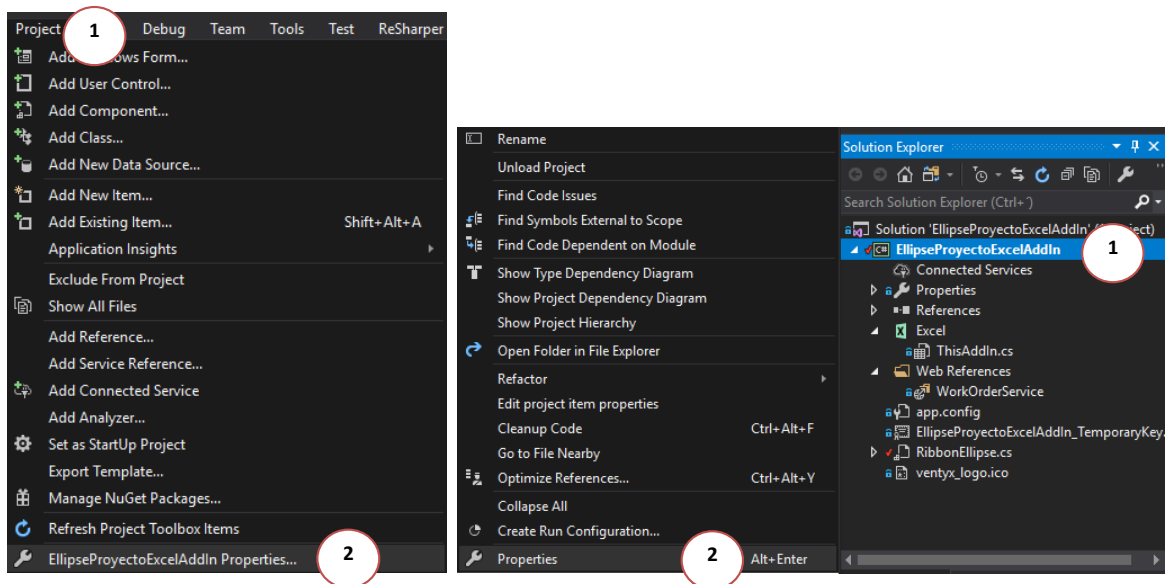


Imagen 53. Despliegue de propiedades del proyecto

Seleccionamos la opción de [3] Aplicación (Application) en la barra lateral izquierda y hacemos clic en el botón de [4] Información de Ensamblado (Assembly Information)

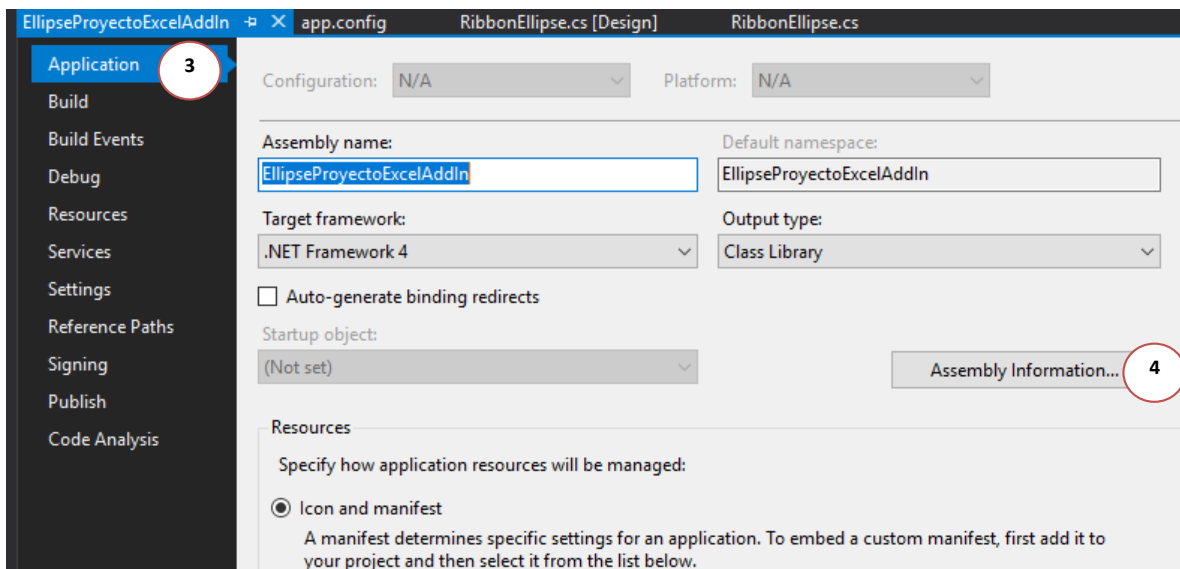


Imagen 54. Propiedades del proyecto

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

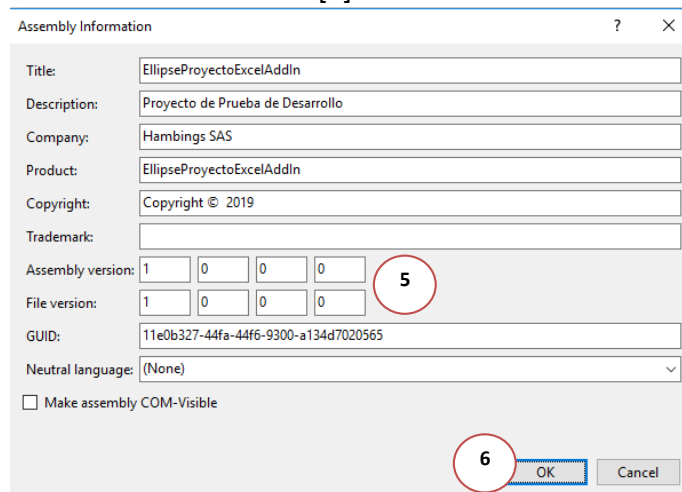


Imagen 55. Información de Ensamblado

Al mismo tiempo, seleccionamos ahora la opción de Publicación (Publish) [7] y ponemos el mismo número de versión en la sección de Versión de Publicación (Publish Version) [8] y hacemos clic en el botón de [9] Publicar Ahora (Publish Now)



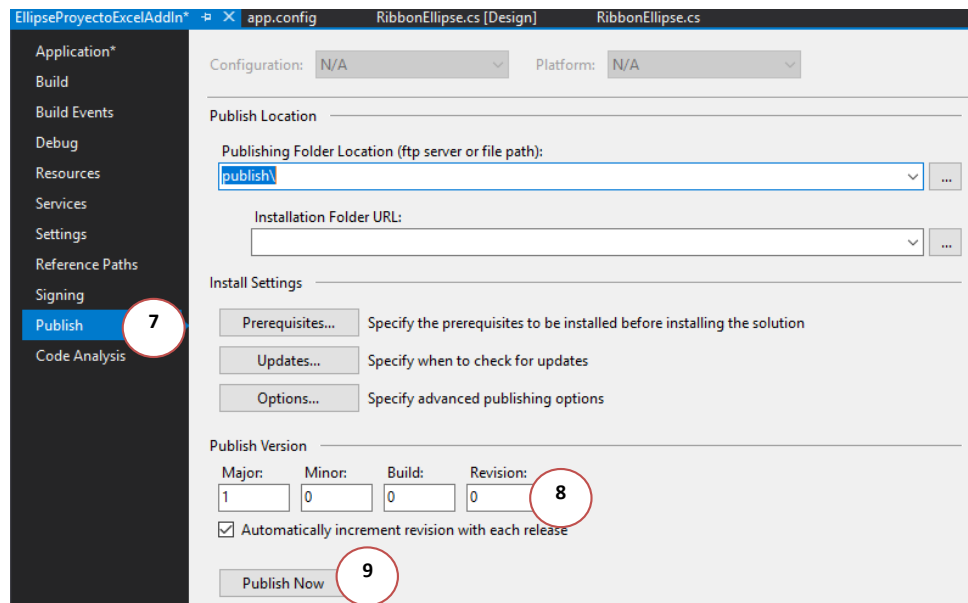


Imagen 56. Información de Publicación

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

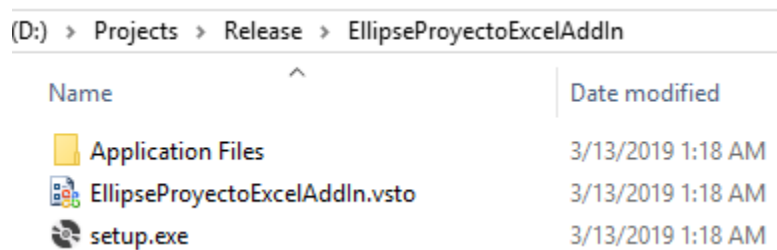


Imagen 57. Archivos de Publicación