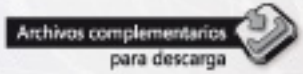
Recursosinformáticos

VBA Excel 2016

Programación en Excel: Macros y lenguaje VBA

Michèle AMELOT



VBA Excel 2016

Programación en Excel: Macros y lenguaje VBA

Completo y, a la vez, simple y práctico, este libro está dirigido a **usuarios de Excel** y a **desarrolladores** que deseen crear aplicaciones amigables, fiables y potentes.

Además de los **elementos básicos del lenguaje VBA** (estructura del lenguaje y conceptos de programación orientada a objetos) que permitirán automatizar el manejo de sus datos, aprenderá a crear **tablas dinámicas** y **gráficos**, a **diseñar formularios**, a **personalizar la interfaz de Excel,** especialmente la **cinta de opciones**, a **comunicarse con las otras aplicaciones de Office**, a importar o **publicar páginas web** y archivos **XML** y a aprovechar las funciones **API de Windows**.

Cada capítulo incluye numerosos ejemplos. El libro concluye con un **ejercicio integrador** que lo guiará a través de la creación completa de una aplicación Excel.

Los ejemplos incluidos en este libro pueden **descargarse** en esta página.

Los capítulos del libro:
Prólogo – Presentación – El lenguaje VBA – La programación de objetos en Excel – Objetos de Excel – Tablas dinámicas y gráficos – Cuadros de diálogo – Formularios – Mejoras en la interfaz de usuario – Administración de eventos – Depuración y administración de errores – Comunicación con las aplicaciones Office 2016 – Internet – Programación Windows – Código de una miniaplicación – Anexos

---

Michèle AMELOT

Formadora y, a la vez, especialista en el desarrollo de aplicaciones ofimáticas, **Michèle AMELOT** ayuda desde hace más de quince años a las empresas en sus proyectos informáticos. Es este profundo conocimiento de las necesidades de los usuarios y de los desarrolladores, lo que le permite, a través de sus libros (VBA Excel y VBA Access, todas las versiones), facilitar el aprendizaje y la práctica en la programación en lenguaje VBA.

Introducción

Este libro está dirigido a usuarios de Excel o a programadores que deseen automatizar el manejo de datos bajo Excel o desarrollar aplicaciones de gestión, reporte o análisis aprovechando las capacidades de la hoja Excel.

El objetivo del libro es presentar, de una manera accesible, todo el abanico de posibilidades del lenguaje VBA. Los diferentes ejemplos incluidos, que se pueden descargar desde la página Información, le permitirán llevar a la práctica, fácilmente y de manera progresiva, los conocimientos adquiridos.

El único requisito necesario para aprovechar el contenido de este libro es un buen conocimiento de la interfaz de Excel. Nociones de programación o el conocimiento de algún lenguaje le ayudarán a manejar el lenguaje aunque no es imprescindible para aprovechar este libro.

En primer lugar, se familiarizará con el código VBA y aprenderá a usarlo para manejar los objetos de Excel (tales como libros, hojas de cálculo, rangos de celdas, tablas dinámicas, gráficos...), especialmente en la automatización de ciertos procedimientos.

En una segunda etapa, descubrirá progresivamente cómo crear aplicaciones profesionales en Excel a través de los siguientes conceptos:

- Uso de los cuadros de diálogo predefinidos para interactuar con el usuario.
- Creación de formularios personalizados para la introducción y presentación de datos de Excel o cualquier otro tipo de datos.
- Mejora de la interfaz de Excel, especialmente en la personalización de la cinta de opciones de Microsoft Office.
- Optimización de la confiabilidad del código VBA gracias a la integración de opciones para el tratamiento de errores.
- Manejo de otras aplicaciones de Microsoft Office 2016: Word, Access y Outlook.
- Acceso a las capacidades de Internet: importar y publicar páginas web y archivos XML.
- Control del sistema operativo a través de la llamada a funciones API de Windows (*Application Programming Interface*).

Finalmente, el último capítulo lo guiará a través de la creación completa de una aplicación de Excel. Esta aplicación permite manejar presupuestos a través de las siguientes características: creación de presupuestos en formato Excel a partir de datos tomados de una base de datos de Access, registro de los presupuestos en carpetas especificadas según el cliente, búsqueda de presupuestos según diversos criterios y otras opciones.

Al terminar el libro, usted quedará convencido de que el lenguaje VBA no se limita a automatizar tareas en Excel, sino que también le permite realizar aplicaciones profesionales dotadas de una interfaz amigable.

Presentación del lenguaje VBA

Visual Basic para Aplicaciones (VBA) es el **lenguaje de programación** común a todas las aplicaciones del paquete Microsoft Office 2016 (Word, Access, Excel, Outlook y PowerPoint).

1. Objetivos del lenguaje VBA

Trabajando en Excel, el lenguaje VBA permite:

- **Automatizar acciones repetitivas**: con VBA puede realizar en una única operación todo un grupo de comandos de Excel.
- **Interactuar sobre los libros de Excel**: el contenido y la presentación de todos los elementos incluidos en un libro (hojas, celdas, gráficos, etc.) se pueden modificar a través de código VBA.
- **Crear formularios personalizados**: los formularios son los cuadros de diálogo compuestos por controles ActiveX (cuadros de texto, listas desplegables, etc.), a los que se les puede asociar código VBA. Los formularios permiten crear interfaces amigables para la entrada o la salida de información.
- **Generar automáticamente tablas dinámicas y gráficos**: así podrá automatizar la creación de estadísticas a partir de sus datos de Excel (o de los datos de la empresa exportados a Excel).
- **Personalizar la interfaz de Excel**: la cinta de opciones de Office 2016 es totalmente personalizable y se pueden asociar macros creadas en lenguaje VBA a los comandos de la cinta o a la barra de herramientas de acceso rápido.
- **Modificar las opciones de Excel**: a cada opción de Excel le corresponde una propiedad de un objeto VBA. Por ejemplo, puede modificar el tipo de fuente por defecto a partir de las propiedades StandardFont (fuente) y StandardFontsize (tamaño de fuente) del objeto Application.

Ejemplo:

Application.StandardFont = «Arial»

Application.StandardFontSize = «10»

- **Comunicar Excel con otras aplicaciones de Microsoft Office**: VBA permite intercambiar información entre las aplicaciones de Office usando objetos específicos propios de cada uno. Por ejemplo, usted puede insertar una tabla o un gráfico de Excel en un archivo de Word, crear mensajes de Outlook con un archivo de Excel adjunto, etc.

2. Algunas definiciones

Proyecto

Cada libro abierto en Excel tiene asociado un proyecto que contiene todos los módulos de código VBA agrupados en categorías.

Módulo

Los módulos contienen las macros grabadas y sus propios procedimientos y funciones escritos en VBA. Los módulos se pueden exportar como archivos independientes para luego ser importados en otros libros.

Procedimiento

Los procedimientos son subprogramas escritos en VBA. Cada macro grabada genera un procedimiento

con el mismo nombre de la macro. De la misma manera, puede crear procedimientos usando la instrucción **Sub**.

Función

Las funciones son procedimientos que devuelven un valor. Para crear una función, se debe utilizar la instrucción **Function**.

3. Escritura de código VBA

Hay dos maneras de crear un procedimiento VBA:

- Generar automáticamente el código a partir de **la grabación de macros**.
- Escribir directamente el código del procedimiento en el **Editor de Visual Basic** (o entorno VBE).

La primera solución es más sencilla, pero mucho más limitada que la segunda. Los procedimientos generados automáticamente solo permiten automatizar acciones repetitivas realizadas con Excel (formato de celdas, ordenar datos, etc.).

Si desea efectuar operaciones específicas, como algoritmos de cálculo, intercambio de mensajes y de información con el usuario, controlar la coherencia de datos en un libro o cualquier otra operación que haga uso de estructuras repetitivas o condicionales, debe crear sus propios procedimientos en el editor de VBA.

Las macros de Excel

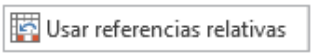
1. Mostrar la pestaña Desarrollador en la cinta de opciones

Para escribir macros, ejecutar macros grabadas o crear aplicaciones de Excel, debe mostrar la pestaña **Desarrollador** de la siguiente manera:

- Haga clic en la pestaña **Archivo** y luego en **Opciones**.
- Seleccione la categoría **Personalizar cinta de opciones**.
- Dentro de **Personalizar cinta de opciones**, en la lista **Pestañas principales**, marque la opción **Desarrollador**.
- Haga clic en el botón **Aceptar**: la pestaña **Desarrollador** se añadirá a la cinta de opciones de Excel, a la derecha de la pestaña **Vista**.

2. Descripción de la pestaña Desarrollador

a. Grupo Código

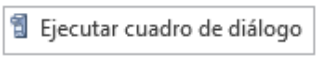
Nombre del botón	Descripción
	Abre el entorno de desarrollo. El método abreviado de teclado es [Alt] [F11].
	Muestra la lista de macros. El método abreviado de teclado es [Alt][F8].
	Comienza la grabación de una macro.
	Permite usar referencias relativas a la primera celda seleccionada.
	Personaliza la configuración de seguridad de las macros.

b. Grupo Complementos

Nombre del botón	Descripción

	Permite descargar complementos de Office.
	Permite seleccionar macros grabadas como complementos.
	Permite seleccionar complementos COM (librerías de funciones complementarias).

c. Grupo Controles

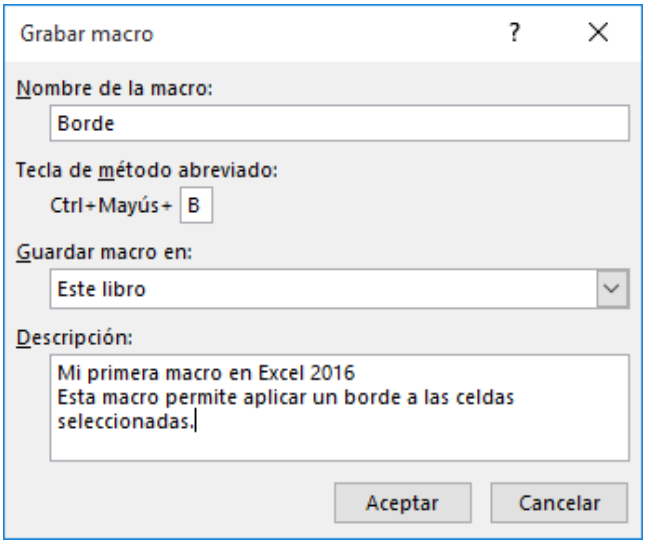
Nombre del botón	Descripción
	Permite insertar controles (formularios o ActiveX) en Excel.
	Activa o desactiva el modo Diseño. En el modo Diseño se pueden seleccionar y modificar los controles ActiveX, pero no se pueden ejecutar.
	Muestra las propiedades del objeto de Excel seleccionado (hoja o control).
	Permite acceder directamente al código asociado al control seleccionado.
	Ejecuta un cuadro de diálogo personalizado.

3. Grabar una macro

a. Grabar la primera macro

Veamos cómo crear una macro que aplique un borde y un relleno en las celdas seleccionadas.

- Seleccione una rango de celdas.
- Haga clic en el botón  en la pestaña **Desarrollador** o en la barra de estado.
- En el cuadro de diálogo que aparece, escriba el nombre de la macro, su descripción e indique, si lo desea, el método abreviado de teclas asociado.



- Haga clic en el botón **Aceptar** para iniciar la grabación.
- Realice en Excel las operaciones que desee grabar. Por ejemplo, aplique un relleno y un borde al rango actualmente seleccionado.
- Haga clic en el botón 

de la pestaña **Desarrollador** para terminar la grabación (también puede usar el mismo botón en la barra de estado).

b. Ejecutar una macro

Para ejecutar una macro desde Excel:

- Haga clic en el botón 

de la pestaña **Desarrollador** o pulse el método abreviado de teclas [Alt] [F8] y luego haga doble clic en el nombre de la macro que desea ejecutar.
- O pulse la combinación de teclas asociada a la macro.

Para visualizar la información relativa a una macro (método abreviado de teclado y descripción), seleccione la macro en la lista y haga clic en el botón **Opciones**.

Para detener la ejecución de una macro, pulse [Escape] o [Ctrl][Pausa].

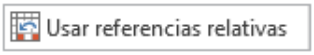
c. Grabar una macro con referencias relativas

Si graba una macro en modo de **referencias absolutas** (modo por defecto), los rangos de celdas referenciados en las operaciones de selección, desplazamientos... serán fijos. Por ejemplo: Range("A2") designa la celda A2.

Si graba una macro en modo de **referencias relativas**, los rangos de celdas serán expresados en relación con la posición de la primera celda activa. Por ejemplo: ActiveCell.range("A2") designa la celda ubicada bajo la celda activa, ActiveCell.range("B1") designa la celda ubicada a la derecha de la celda activa.

ActiveCell.range("A1") siempre hace referencia a la primera celda activa. A1 se puede considerar la referencia relativa a la primera celda activa.

Para grabar una macro con referencias relativas:

- Haga clic en el botón 

de la pestaña **Desarrollador**: el botón quedará activo (se verá destacado).

  Si vuelve a hacer clic en el botón 

, este quedará desactivado y las macros se grabarán con referencias absolutas.

Ejemplo

La misma secuencia de operaciones se ha registrado en dos macros: la primera (RefRelativa) se grabó con la opción referencias relativas; la segunda (RefAbsoluta), con la opción referencias absolutas.

La secuencia de operaciones es la siguiente:

- *Seleccionar un rango de celdas.*
- *Desplazar el rango dos filas hacia abajo y una columna a la derecha.*

```
Sub RefRelativa()
'  Referencias relativas
   ActiveCell.Range("A1:B7").Select
   Selection.Cut Destination:=ActiveCell.Offset(2, 1).Range("A1:B7")
   ActiveCell.Offset(2, 1).Range("A1:B7").Select
End Sub
```

```
Sub RefAbsoluta()
'  Referencias absolutas
   Range("B2:C8").Select
   Selection.Cut Destination:=Range("C4:D10")
   Range("C4:D10").Select
End Sub
```

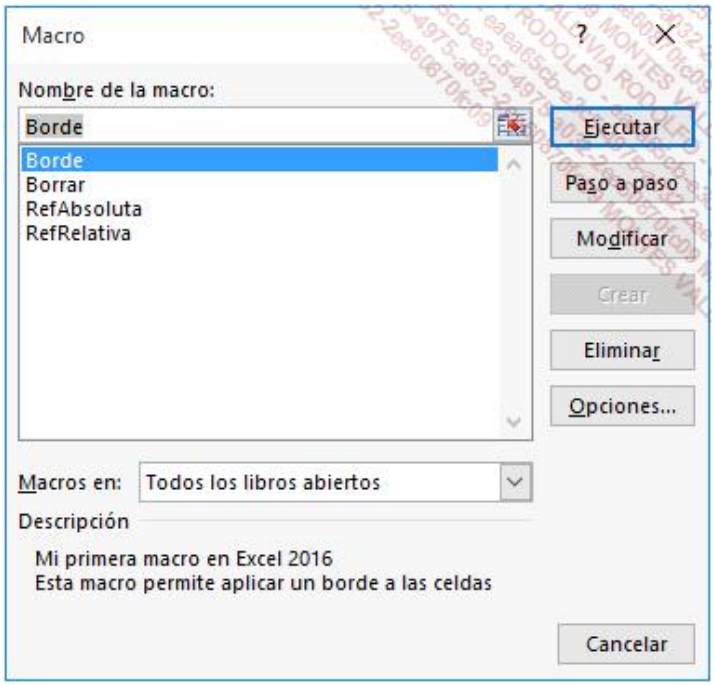
d. Definir el lugar de almacenamiento de una nueva macro

Para definir el lugar de almacenamiento de una nueva macro:

- Haga clic en el botón 

de la pestaña **Desarrollador** o pulse el método abreviado [Alt][F8].

- Abra la lista **Macros en** y seleccione el libro en el que desea crear la macro.



Si elige **Libro de macros personal**, la macro se grabará en el libro personal.xlsb; la macro será accesible desde todos los libros Excel.

e. Eliminar una macro

Para eliminar una macro:

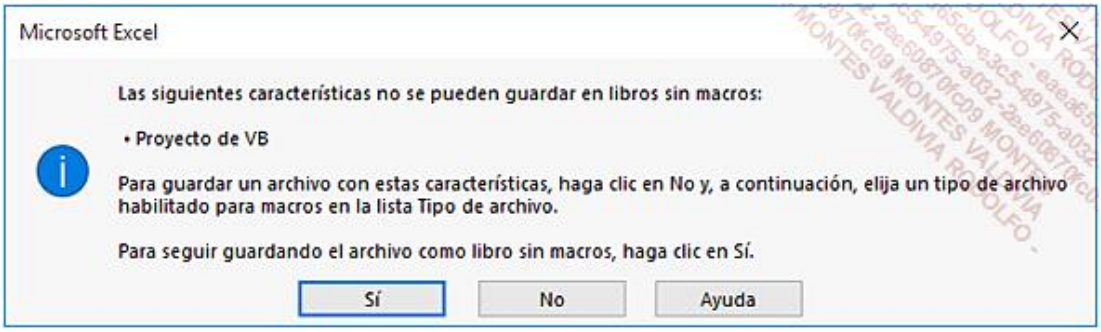
- Haga clic en el botón 

de la pestaña **Desarrollador** o pulse el método abreviado [Alt][F8].

- Seleccione la macro que desea eliminar y haga clic en el botón **Eliminar**. Haga clic en el botón **Sí** para confirmar la eliminación.

f. Guardar un libro con macros

Si ha creado macros en un libro y lo guarda por primera vez, aparecerá el siguiente mensaje:

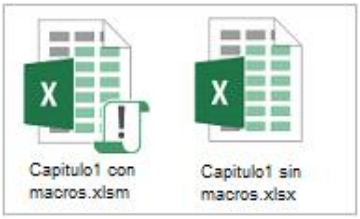


- Haga clic en **No** para no guardar el libro sin las macros.

- En el cuadro **Guardar como**, abra la lista **Tipo** y seleccione **Libro de Excel habilitado para macros**

(*.xlsm).

Los libros que contienen macros tienen la extensión xlsm (en vez de xslx) y su icono se distingue por un signo de exclamación.



También puede grabar un nuevo libro con macros mediante las siguientes operaciones:

- Haga clic en la pestaña **Archivo** y luego en **Guardar como**.
- En el cuadro **Guardar como**, abra la lista **Tipo** y seleccione la opción **Libro de Excel habilitado para macros (*.xlsm)**.

4. Las macros y la seguridad

La configuración de seguridad de las macros permite controlar lo que ocurre al abrir un libro que contiene macros.

Las modificaciones de la configuración de seguridad de macros rigen solamente en Excel y no afectan al resto de las aplicaciones de Microsoft Office.

a. Modificar la configuración de seguridad

- Haga clic en el botón 

de la pestaña **Desarrollador**.
- Dentro de **Configuración de macros**, seleccione la opción deseada (vea en la siguiente sección: Descripción de las diferentes opciones de seguridad).
- Haga clic en **Aceptar** para confirmar su elección.

Si se cambia la configuración de seguridad, las nuevas opciones se aplicarán a todos los libros, excepto a los libros actualmente abiertos. Para aplicarlas a estos libros, debe cerrarlos y abrirlos nuevamente.

b. Descripción de las diferentes opciones de seguridad

Deshabilitar todas las macros sin notificación

Todas las macros y todas las advertencias de seguridad serán deshabilitadas.

Deshabilitar todas las macros con notificación

Es la opción por defecto. Las macros serán deshabilitadas pero aparecerá una advertencia de seguridad en la barra de mensajes (bajo la cinta de opciones) para los libros que contienen macros.

Deshabilitar todas las macros excepto las firmadas digitalmente

Si las macros de un libro tienen firma digital de un origen aprobado, las macros se podrán ejecutar. Si el origen no está autorizado, aparecerá una notificación: en este caso es posible habilitar las macros firmadas o aprobar el origen. Las macros sin firma digital no se podrán habilitar.

Habilitar todas las macros

Si selecciona esta opción, todas las macros serán habilitadas. Se recomienda no usar esta opción de forma permanente.

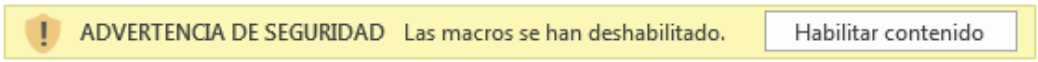
Cualquiera que sea la opción elegida, si instala un programa antivirus compatible con Microsoft Office System 2016, los libros que contengan macros serán analizados antes de abrirse.

Confiar en el acceso al modelo de objetos de proyectos de VBA

Este parámetro es para los desarrolladores y sirve para controlar o autorizar el acceso por programa al modelo de objetos VBA (formularios, módulos y módulos de clase). La manipulación de los objetos se hace a través de la propiedad **VBComponents**, que devuelve la colección de componentes de un proyecto.

c. Habilitar las macros cuando aparece la advertencia de seguridad

Si ha seleccionado la opción **Deshabilitar todas las macros con notificación**, aparecerá la siguiente advertencia bajo la cinta de opciones cuando abra un libro que contiene macros:

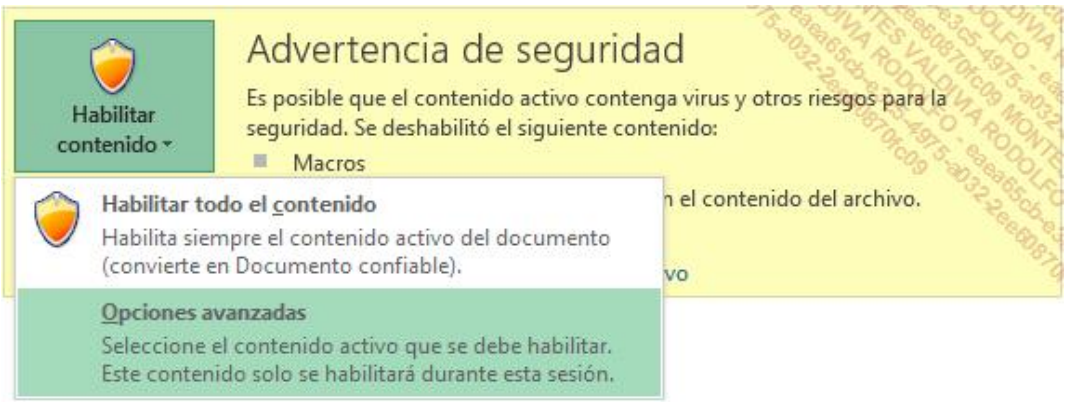


Para habilitar las macros del libro, haga clic en el botón **Habilitar contenido**. A partir de ahora, el libro pasará a ser un **documento confiable** y la advertencia de seguridad no aparecerá la próxima vez que lo abra.

Un documento se considera confiable para una determinada ubicación: si cambia la ubicación de un documento aprobado, volverá a aparecer la advertencia de seguridad la próxima vez que lo abra.

Las macros se pueden habilitar solo para una sesión (es decir, hasta que cierre el libro que las contiene):

- Al aparecer la advertencia de seguridad, haga clic en la pestaña **Archivo** y seleccione la sección **Información**.
- Dentro de **Advertencia de seguridad**, haga clic en **Habilitar contenido** y luego en **Opciones avanzadas**.



- En el cuadro **Opciones de seguridad de Microsoft Office**, seleccione la opción **Habilitar contenido para esta sesión**.
- Desaparecerá la advertencia de seguridad, aunque volverá a aparecer la próxima vez que abra el libro.

d. Activar las macros en una ubicación dada

Puede definir una lista de ubicaciones de confianza para que los libros que se encuentren en ella se consideren confiables:

- Haga clic en el botón 

de la pestaña **Desarrollador**.
- En el menú de la izquierda, seleccione la categoría **Ubicaciones de confianza**.
- Haga clic en el botón **Agregar nueva ubicación**.
- Haga clic en el botón **Examinar**. Localice y seleccione la carpeta deseada y luego haga clic en **Aceptar**: la ruta seleccionada se agregará a la lista de ubicaciones de confianza.

e. Firmas electrónicas de macros

Para firmar digitalmente un proyecto macro, debe:

- Obtener un certificado digital de parte de una autoridad de certificación comercial, como Verisign (www.verisign.es) o Thawte (www.thawte.com), e instalarlo.
- Firmar digitalmente su proyecto. Para ello, pase al entorno VBE (accesible desde el comando 

de la pestaña **Desarrollador**) y seleccione la opción **Firma digital** del menú **Herramientas**. Haga clic en el botón **Elegir** para seleccionar su certificado.

Para probar sus proyectos macros en su ordenador, puede crear su propio certificado de autofirma con la ayuda de Selfcert.exe (programa distribuido con Microsoft Office 2016 y con los ejemplos de este libro).

5. Modificar el código de una macro

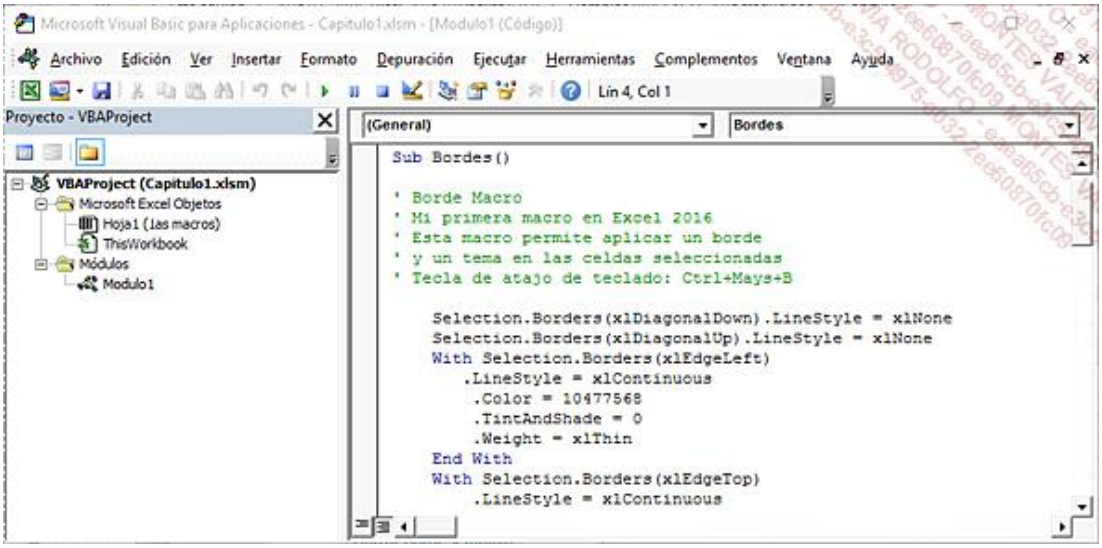
Una macro es un **procedimiento** escrito en lenguaje VBA. Un procedimiento VBA es una secuencia de instrucciones agrupadas en un bloque de código que comienza con **Sub** y termina con **End Sub**.

Para acceder al código de una macro:

- Haga clic en el botón 

de la pestaña **Desarrollador** o pulse el método abreviado [Alt][F8].
- Seleccione la macro que desea modificar y haga clic en el botón **Modificar**.

El código de la macro aparecerá en una ventana del entorno Microsoft Visual Basic (ver El entorno de desarrollo VBE, en este capítulo).



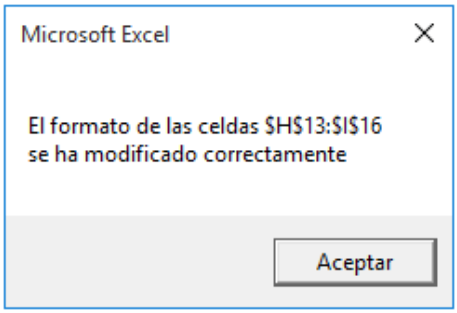
Puede modificar o completar el código dentro de este procedimiento.

Ejemplo

Inserte el siguiente código al final del procedimiento:

```
...
    MsgBox "El formato de celdas " & Selection.Address _
            & vbCr & "se ha modificado correctamente"
End Sub
```

Para probar su procedimiento, haga clic en el icono  o use la tecla [F5]. Aparecerá el siguiente mensaje:



Asignar una macro

Una macro se puede asociar a los siguientes elementos:

- Comandos personalizados en la cinta de opciones.
- Botones de comando en la barra de herramientas de acceso rápido.
- Controles u objetos insertados en las hojas de Excel (botones de comando, imágenes, etc.).

1. Acceso a una macro desde la cinta de opciones de Office 2016

Microsoft Office 2016 ofrece la posibilidad de personalizar la cinta de opciones: puede agregar, eliminar o reasignar las pestañas, grupos y comandos de la cinta para disponer de aplicaciones personalizadas.

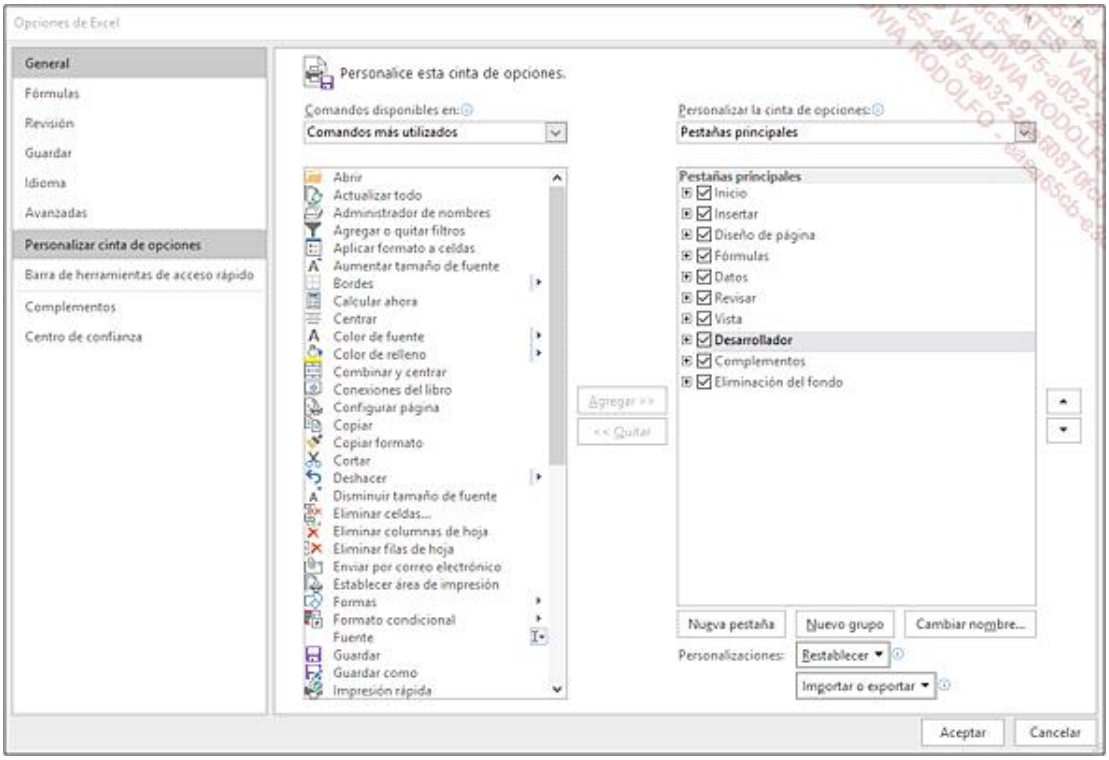
Para personalizar la cinta de opciones:

- Haga clic en el botón derecho del ratón sobre la cinta y seleccione la opción **Personalizar la cinta de opciones**.

  O

- Haga clic en la pestaña **Archivo**, luego en **Opciones** y seleccione la categoría **Personalizar cinta de opciones**.

Aparecerá el siguiente cuadro:



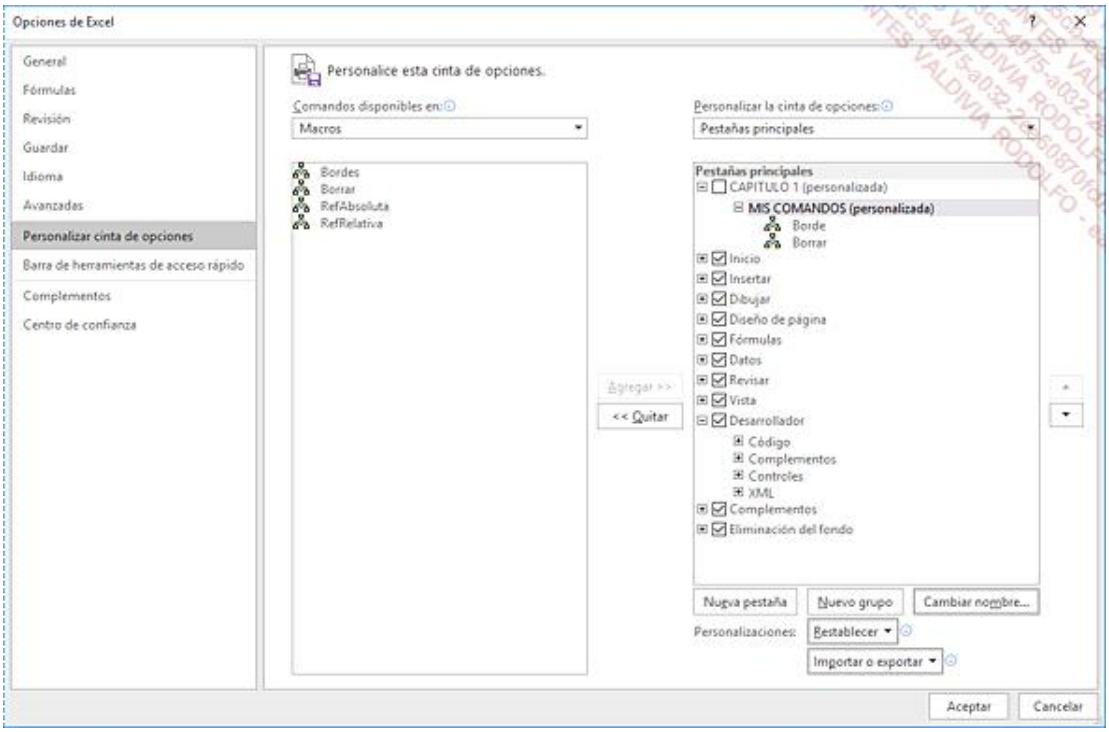
Para insertar macros en una nueva pestaña:

- Haga clic en el botón **Nueva pestaña**. Se agregará una pestaña y un grupo a la lista de pestañas principales.

- Para cambiar el nombre de la pestaña y del grupo que acaba de crear, haga clic en el botón **Cambiar nombre**. El menú contextual también le permite agregar pestañas o grupos.
- Despliegue las opciones dentro de **Comandos disponibles en:** (a la izquierda y arriba) y seleccione **Macros**. Aparecerá la lista de macros disponibles.
- Seleccione la macro que desea agregar al grupo y haga clic en **Agregar**.
- Para modificar el nombre y el icono de la macro haga clic en el botón **Cambiar nombre**.

  También puede cambiar el orden de las pestaña, grupos y comandos con las flechas situadas a la derecha de la lista.

Ejemplo de personalización:



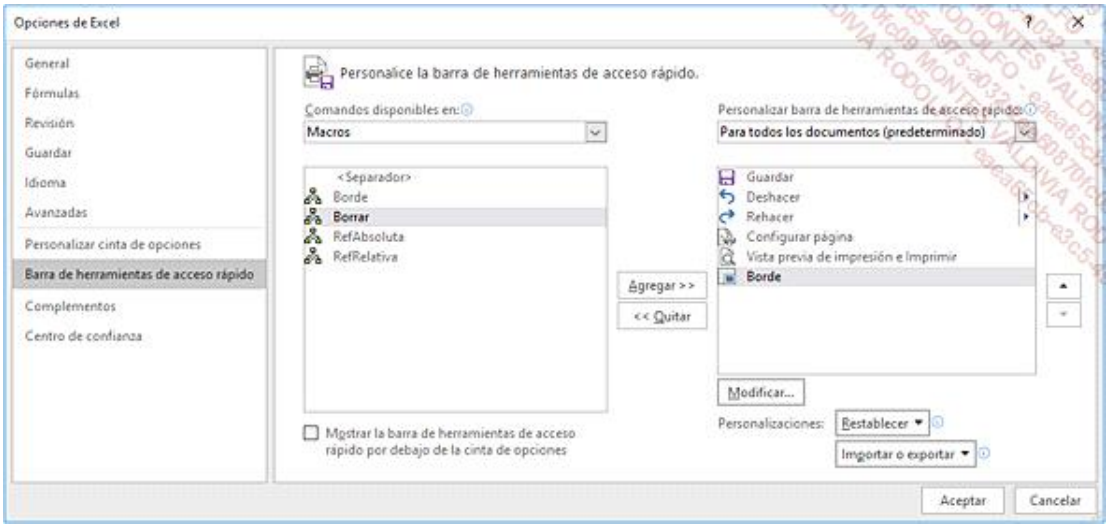
- Para confirmar esta personalización, haga clic en **Aceptar**. La nueva pestaña aparecerá en la cinta de opciones de Excel 2016:

Usted puede restablecer la cinta de opciones en cualquier momento desde el cuadro de diálogo para personalizar la cinta, haciendo clic en el botón **Restablecer** (puede restablecer toda la cinta o solamente una pestaña). También puede importar una cinta personalizada haciendo clic en el botón **Importar o exportar**.

2. Asociar una macro a un icono de la barra de herramientas de acceso rápido

También puede insertar un comando para ejecutar una macro en la barra de herramientas de acceso rápido:

- Haga clic en el botón derecho del ratón sobre la cinta de opciones y seleccione la opción **Personalizar barra de herramientas de acceso rápido**.
- En la lista desplegable **Comandos disponibles en** situada arriba y a la izquierda, seleccione **Macros**. Aparecerá la lista de macros disponibles.
- Seleccione la macro que desee y haga clic en el botón **Agregar**. La macro pasará a la lista de la derecha.



- Haga clic en el botón **Modificar...** para seleccionar un nuevo icono para la macro.
- Puede modificar el orden de los comandos con las flechas ubicadas a la derecha de la lista.
- Puede seleccionar un libro en la lista desplegable **Personalizar barra de herramientas de acceso rápido** situada arriba y a la derecha. En ese caso las modificaciones solamente afectarán al libro indicado.
- Haga clic en **Aceptar**.

  Aparecerá un nuevo comando en la barra de acceso rápido para ejecutar la macro **Bordes**.

3. Asociar una macro a un botón de comando

- Haga clic en el botón  de la pestaña **Desarrollador**.

- Dibuje el botón con la herramienta  (parte superior izquierda de la barra de **Controles de formulario**).

  Al soltar el botón del ratón, aparecerá el cuadro de diálogo **Asignar macro**.
- Seleccione el **Nombre de la macro** en la lista y haga clic en **Aceptar**.

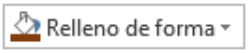
4. Asignar una macro a una imagen

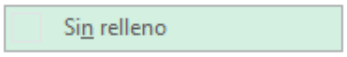
- Inserte una imagen haciendo clic en el botón 

de la pestaña **Insertar**.
- Haga clic en el botón derecho del ratón y seleccione la opción **Asignar macro**.
- Seleccione el **Nombre de la macro** en la lista y haga clic en el botón **Aceptar**.

5. Asociar una macro a una zona de un objeto gráfico

- Inserte un objeto gráfico (imagen, imagen prediseñada, forma o SmartArt, accesibles desde el grupo **Ilustraciones** de la pestaña **Insertar**).
- Para definir en el objeto una zona que permita ejecutar una macro, haga clic en 

(en la pestaña **Insertar**), seleccione la forma deseada y dibuje la forma dentro del objeto que acaba de crear.
- Haga clic en el botón derecho del ratón y seleccione la opción **Asignar macro**.
- Seleccione el **Nombre de la macro** en la lista y confirme haciendo clic en el botón **Aceptar**.
- Seleccione la forma insertada. Aparecerá la sección **Herramientas de dibujo**; haga clic en la pestaña **Formato**.
- En el grupo **Estilos de forma** de la pestaña **Formato**, haga clic en la lista 

y luego en 

.
- En el mismo grupo **Estilo de forma** de la pestaña **Formato**, haga clic en la lista 

, luego en 

.
- Ahora, al hacer clic en la forma, se ejecutará la macro asociada.

6. Asociar una macro a una imagen Control ActiveX

- Active el modo **Diseño** (si no está activo) haciendo clic en el botón 

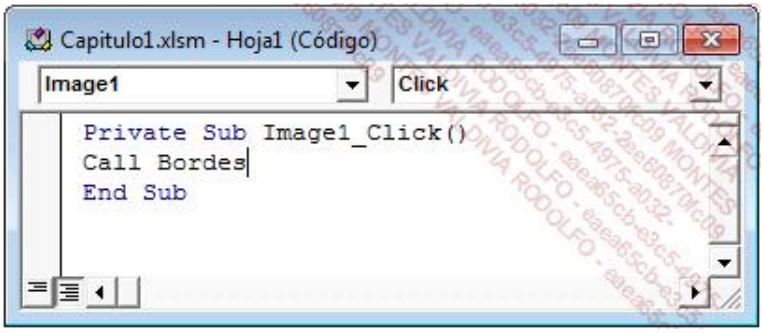
de la pestaña **Desarrollador**.
- Haga clic en el botón 

de la pestaña **Desarrollador**.
- Dibuje una imagen con la herramienta  (dentro de los controles ActiveX) y suelte el botón del ratón.
- Ahora puede mostrar y modificar las propiedades de la imagen (en particular, asociar una imagen por

medio de la propiedad **Picture**), haciendo clic en el botón 

de la pestaña **Desarrollador**.

- Haga doble clic en la imagen.
- Dentro del procedimiento VBA asociado a la imagen, escriba la instrucción **Call** seguida del nombre de la macro que desea ejecutar.



- Desactive el modo Diseño haciendo clic en el botón 

de la pestaña **Desarrollador**.
- Haga clic en la imagen: se ejecutará la macro Bordes.

El entorno de desarrollo VBE

VBE (*Visual Basic Editor*) es el entorno en el que puede escribir, modificar y probar su código VBA. Este entorno se llama también IDE (*Integrated Development Environment*) o editor de VBA.

El entorno VBE pone a su disposición numerosas herramientas para facilitar la programación y la puesta a punto de su código VBA: herramientas de depuración, introducción asistida, explorador de objetos, etc.

1. Acceso al entorno VBE

- Para acceder al entorno VBE desde Excel, haga clic en el botón 

de la pestaña **Desarrollador** o

  pulse la combinación [Alt][F11].

2. Cerrar el entorno VBE

Para cerrar el entorno VBE y volver a Excel:

- Haga clic en el aspa roja ubicada arriba a la derecha de la ventana principal del entorno VBE.

  o

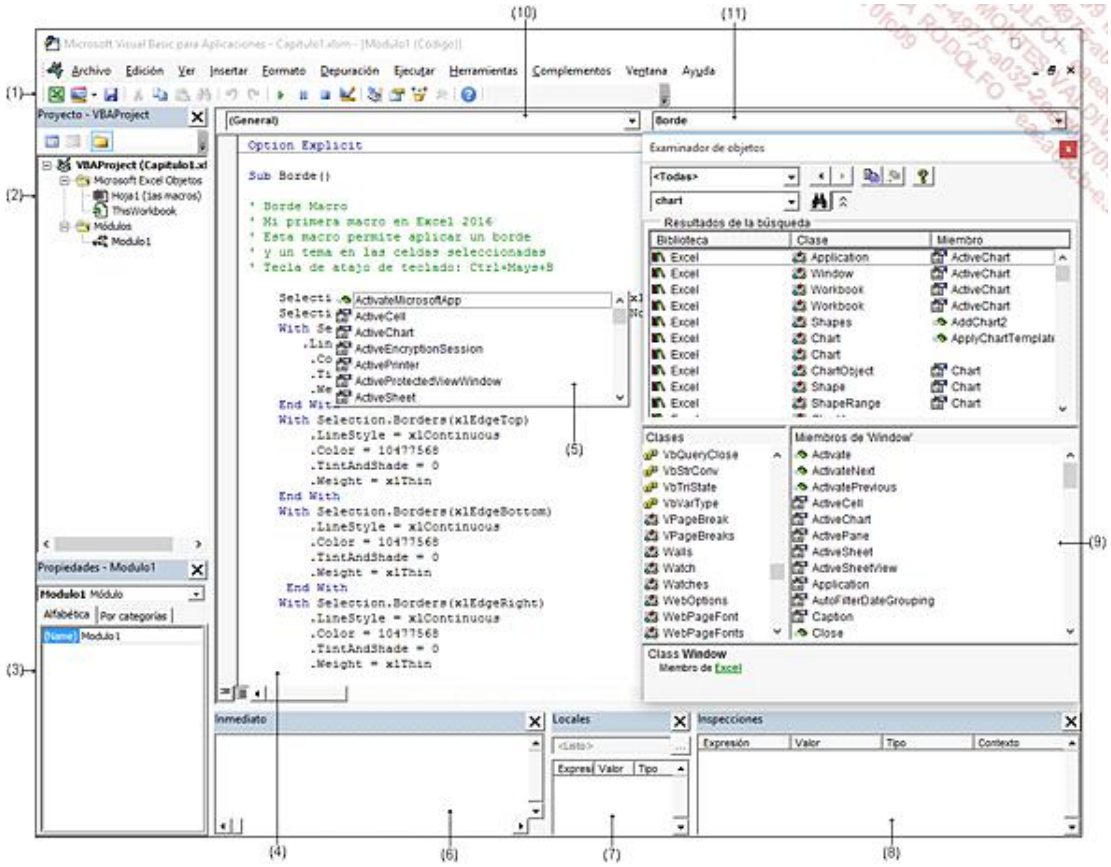
- Haga clic en la opción **Cerrar y volver a Microsoft Excel** del menú **Archivo**.

3. Volver a Excel

Para volver a Excel sin cerrar el entorno VBE, use cualquiera de estas dos posibilidades:

- Haga clic en el icono  de la barra de herramientas **Estándar**.
- Pulse el método abreviado [Alt][F11].

4. Descripción del entorno VBE



Todas las ventanas del entorno VBE se pueden visualizar desde el menú **Ver**.

(1) La barra de herramientas Estándar

1 Ver a Microsoft Excel
2 Insertar UserForm
3 Guardar libro
4 Cortar
5 Copiar
6 Pegar
7 Buscar
8 Deshacer
9 Rehacer
10 Ejecutar Sub
11 Interrumpir
12 Restablecer
13 Modo de diseño
14 Explorador de proyectos
15 Ventana de Propiedades
16 Examinador de objetos
17 Cuadro de herramientas
18 Ayuda de Microsoft Visual Basic
19 Posición actual en la ventana de código

(2) El explorador de proyectos

Cada libro abierto en Excel tiene asociado un proyecto. El explorador de proyectos permite ver todos los proyectos y todos los módulos de cada proyecto según una estructura en árbol. Los módulos se agrupan en cuatro categorías:

- Módulos asociados a objetos de Excel (libro y hojas).
- Módulos asociados a formularios.

- Módulos estándares.
- Módulos de clase.

Cada módulo puede contener muchos procedimientos.

(3) La ventana Propiedades

Muestra las propiedades relativas al libro, a las hojas de cálculo, a las hojas gráficas y a los formularios.

(4) La ventana Código

En esta ventana aparecen dos zonas con listas desplegables:

- La zona objeto **(10)** muestra la lista de los objetos del módulo.
- La zona procedimiento **(11)** muestra los procedimientos o los eventos del objeto seleccionado en la zona objeto. Los eventos ya usados aparecen en negrita.

(5) Instrucciones que se completan automáticamente

Cuando se escribe el nombre de un objeto seguido de un punto, aparece automáticamente una lista desplegable con los métodos, propiedades y constantes disponibles para ese objeto.

Si la lista no aparece, seleccione **Opciones** en el menú **Herramientas** y marque la casilla **Lista de miembros automática** de la pestaña **Editor**.

(6) Ventana Inmediato

Contiene todos los valores de las variables que hayan sido definidas previamente como expresiones de inspección.

(7) Ventana Locales

Permite mostrar los valores de las variables, modificarlas, y ejecutar las instrucciones.

(8) Ventana Inspecciones

Contiene todos los valores de las variables del procedimiento en curso.

Las ventanas **Inmediato**, **Locales** e **Inspecciones** se usan principalmente para la depuración de aplicaciones (ver capítulo Depuración y administración de errores).

(9) El Examinador de objetos

Permite visualizar, para cada objeto, sus propiedades, métodos y constantes.

5. Elegir las ventanas que hay que mostrar

Nombre de la ventana que hay que mostrar	Menú	Barra de herramientas	Teclado
Propiedades	**Ver - Ventana Propiedades**		[F4]

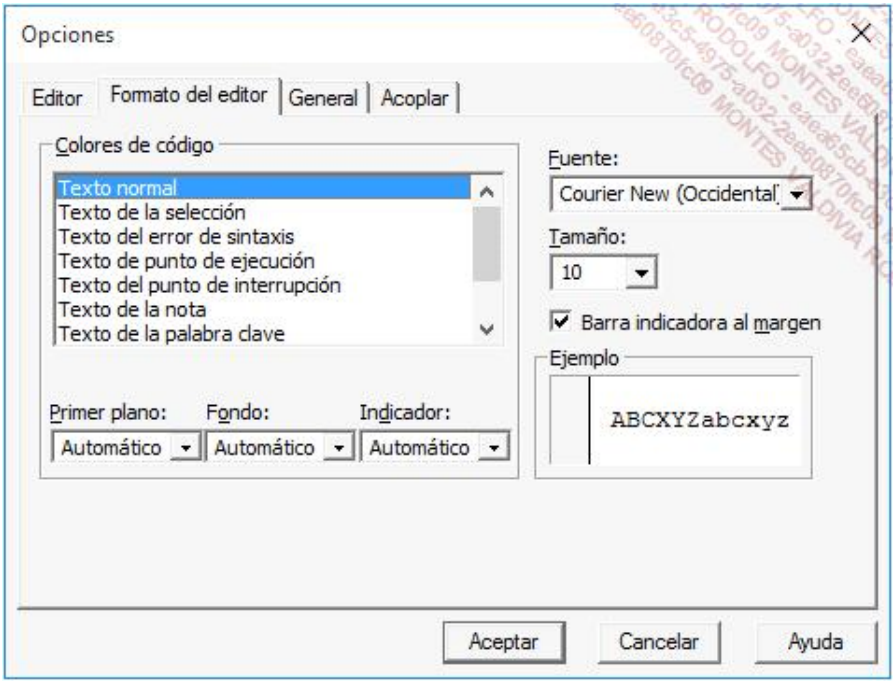
Proyectos	**Ver - Explorador de proyectos**		[Ctrl] R
Inmediato	**Ver - Ventana Inmediato**		[Ctrl] G
Inspecciones	**Ver - Ventana Inspección**		
Ventana Locales	**Ver - Ventana Locales**		
Explorador de objetos	**Ver - Examinador de objetos**		[F2]
Módulo	**Ver - Código**		[F7]

Configuración del editor VBA

1. Configuración de la tipografía

Las palabras clave, las funciones y las instrucciones VBA aparecen en azul; los objetos, métodos y propiedades, en negro, y los comentarios, en verde. Las instrucciones que contienen errores se destacan en rojo.

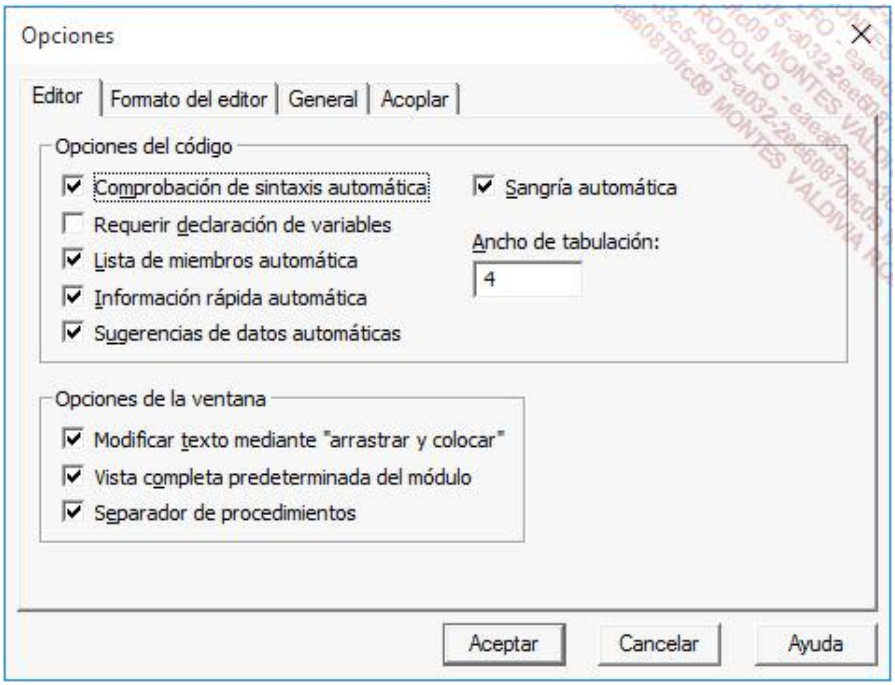
- Para modificar el estilo (color, tipo de letra, tamaño) de los diferentes tipos de código, seleccione **Opciones** en el menú **Herramientas** y haga clic en la pestaña **Formato del editor**.



2. Configuración de la introducción de código

Existen diferentes herramientas que facilitan la introducción y la actualización del código VBA: por ejemplo, la comprobación automática de la sintaxis, la declaración obligatoria de las variables, las instrucciones que se completan automáticamente, etc.

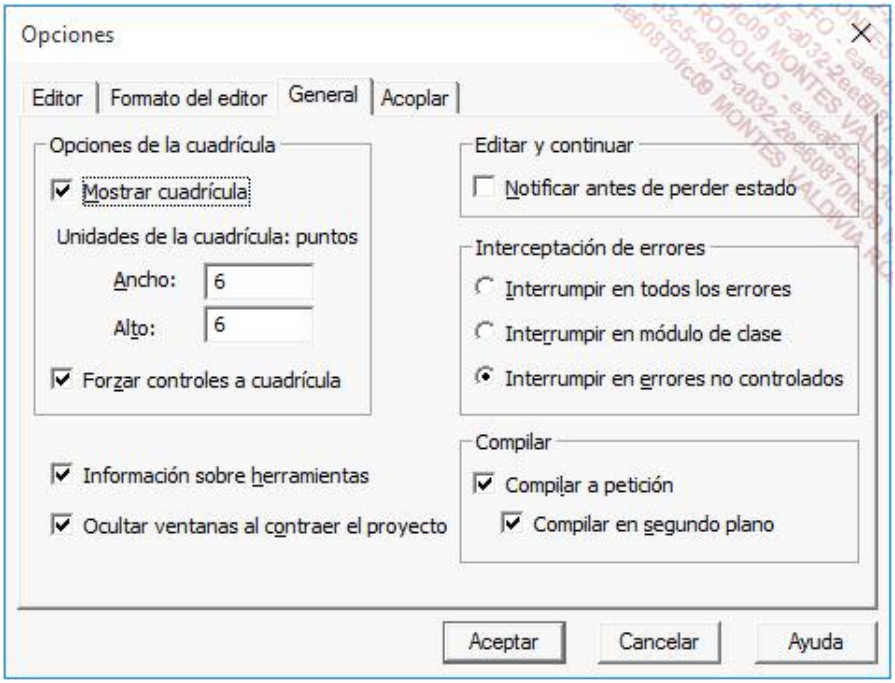
- Para activar estas opciones, seleccione **Opciones** en el menú **Herramientas** y haga clic en la pestaña **Editor**.



3. Manejo de errores

Las opciones de interceptación de errores permiten establecer si la ejecución del código se interrumpirá al ocurrir un error de ejecución.

- Para activar esta opción, seleccione **Opciones** en el menú **Herramientas** y haga clic en la pestaña **General**.



- Si su código VBA incluye una gestión de errores, seleccione la opción **Interrumpir en errores no controlados** o no se tendrán en cuenta las instrucciones de gestión de errores.

Ciertos ejemplos de este libro incluyen una gestión de errores; es importante que active esta opción para que funcionen correctamente.

La gestión de errores se explica en el capítulo Depuración y administración de errores.

4. Acoplar una ventana

Una ventana acoplada se coloca automáticamente cuando la mueve. Una ventana no está acoplada si puede ubicarse en cualquier lugar de la pantalla y permanecer ahí.

- Para definir las ventanas que desea acoplar, seleccione **Opciones** en el menú **Herramientas** y haga clic en la pestaña **Acoplar**.
- Active las ventanas que desea acoplar y desactive las otras. Luego haga clic en **Aceptar**.

Módulos

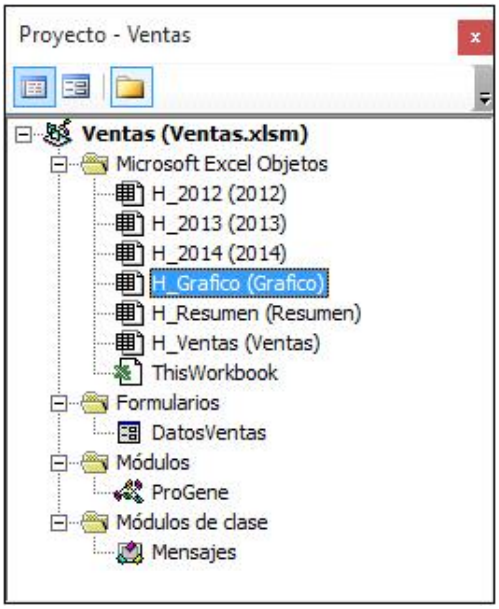
1. Presentación

El código VBA asociado a un libro está agrupado en un proyecto que contiene varias carpetas:

La carpeta **Microsoft Excel Objetos**	Contiene un módulo de clase asociado al libro del proyecto (llamado por defecto ThisWorkbook) y un módulo de clase por cada una de las hojas de cálculo u hojas de gráfico del libro. En particular, en estos módulos de clase se encuentran los procedimientos de eventos asociados al libro y a las hojas.
La carpeta **Formularios**	Contiene los formularios (UserForm) del proyecto y el código VBA asociado.
La carpeta **Módulos**	Agrupa los diferentes módulos estándares (compuestos por uno o más procedimientos) que pueden ser llamados desde cualquier procedimiento del proyecto.
La carpeta **Módulos de clase**	Contiene los módulos de clase usados para la creación de nuevas clases de objetos. Los módulos de clase se utilizan especialmente para la escritura de los procedimientos de eventos asociados a los objetos **Application** y **Chart** (ver capítulo Administración de eventos).

La lista de todos los módulos aparece en forma jerárquica en el Explorador de proyectos del entorno VBE.

Si el explorador de proyectos no está visible, elija la opción **Explorador de proyectos** del menú **Ver** o pulse el método abreviado [Ctrl] R.

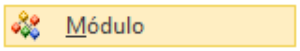


Para ver el código asociado a un módulo, haga doble clic en el nombre del módulo.

Los elementos del lenguaje VBA descritos en este capítulo se pueden usar en los diferentes módulos.

2. Acceso a los módulos

Para insertar un nuevo módulo en el entorno VBE, use la opción **Módulo** del menú **Insertar**, o haga clic en el icono

 de la barra de herramientas **Estándar** y seleccione la opción 

.

Si la ventana **Módulo** está maximizada, el nombre del módulo aparece sobre la barra de título de **Microsoft Visual Basic**.

- Para pasar de un módulo a otro, en la ventana **Proyecto**, haga doble clic en el nombre del módulo que desea activar.
- Para eliminar un módulo, en la ventana **Proyecto**, haga clic derecho en el nombre del módulo que desea eliminar, elija la opción **Quitar Módulo** e indique si desea exportar el módulo o no (ver Importar y exportar código VBA).
- Para dar nombre a un módulo, active el módulo y cambie la propiedad **Nombre** en la ventana de **Propiedades**.

3. Importar y exportar código VBA

Los módulos y formularios pueden exportarse a un archivo para luego importarlo a otro proyecto de Excel.

- Para exportar un archivo, haga clic en el nombre del archivo en el explorador de proyectos, luego seleccione la opción **Exportar archivo** del menú **Archivo** (o del menú contextual) o pulse la combinación de teclas [Ctrl] E.
- Para importar un archivo, haga clic en el nombre del archivo en el explorador de proyectos, luego seleccione la opción **Importar archivo** del menú **Archivo** (o del menú contextual) o pulse la combinación de teclas [Ctrl] M.

La extensión del archivo creado depende del tipo de archivo exportado:

- Los **módulos de clase** (módulos asociados al libro y a las hojas y módulos de clase independientes) tienen la extensión **.cls**.
- Los **formularios** tienen la extensión **.frm**.
- Los **módulos** estándar tienen la extensión **.bas**.

Procedimientos

1. Definiciones

Los procedimientos son **subprogramas** que permiten descomponer una tarea de programación compleja en un conjunto de tareas más breves y simples. Permiten organizar el código dentro de módulos para obtener un código de mantenimiento más simple y fácilmente reutilizable.

En VBA Excel, se distinguen tres tipos de procedimientos:

- Los procedimientos **Sub** (por subrutina) se llaman subprogramas o procedimientos Sub.
- Los procedimientos **Function** se llaman funciones.
- Los procedimientos **Property** se llaman procedimientos de propiedad.

En este capítulo, solamente nos interesan los dos primeros, que son los más utilizados.

- Puntos comunes entre procedimientos Sub y funciones:
  - Ambos contienen instrucciones o métodos VBA.
  - Ambos aceptan argumentos.
  - Ambos se pueden llamar desde otras funciones o procedimientos Sub.
- Características específicas de las funciones:
  - Devuelven un valor.
  - Se pueden utilizar desde Excel como cualquier función.

2. Acceso a los procedimientos

- Para acceder a un procedimiento desde la ventana de código de un módulo, abra la segunda lista de la ventana del módulo, haga clic en el nombre del procedimiento al que desea acceder o recorra los procedimientos con [Ctrl][Flecha arriba] y [Ctrl][Flecha abajo].
- Para seleccionar una palabra, haga doble clic en la palabra.
- Para seleccionar una línea, sitúe el puntero del ratón a la izquierda de la línea y haga clic cuando el puntero se convierta en una flecha.
- Para seleccionar un grupo de caracteres, use la técnica de arrastrar y soltar o haga [Mayús] clic.
- Para seleccionar un procedimiento completo, sitúe el puntero del ratón a la izquierda de cualquier línea del procedimiento. Cuando el puntero se transforme en una flecha, haga doble clic.
- Para ejecutar un procedimiento, haga clic en el procedimiento que desea ejecutar y pulse [F5] o  .
- Para eliminar un procedimiento, seleccione todo el procedimiento y pulse [Supr].

3. Procedimientos Sub

Hay dos tipos de procedimientos Sub:

- Los procedimientos Sub generales,
- Los procedimientos Sub asociados a eventos.

Un **procedimiento general** es un procedimiento declarado en un módulo (generalmente un módulo estándar). La llamada a este tipo de procedimiento se define explícitamente en el código.

Un **procedimiento asociado a un evento** es un procedimiento que se ejecuta automáticamente ante ciertos eventos de un objeto. Su nombre se forma con el nombre del objeto, seguido del guión bajo "_" y del nombre del evento (ej.: Workbook_Open). La llamada a estos procedimientos es implícita, es decir, el procedimiento se ejecuta automáticamente cuando se produce el evento asociado.

Ejemplo

*El siguiente **procedimiento general** pide al usuario que confirme su deseo de abandonar la aplicación, y sale de Excel si el usuario responde que Sí. Este código se puede ejecutar con un botón de comando o una opción de menú que permita abandonar la aplicación.*

```
Private Sub Terminar()
   If MsgBox("¿Desea salir del programa?", _
      vbQuestion + vbYesNo) = vbYes Then
      Application.Quit
    End If
End Sub
```

*El siguiente **procedimiento asociado a un evento** abre automáticamente el libro Ventas.xlsx cuando se abre el libro Resumen.xlsx. Este procedimiento está asociado al evento **Open** del objeto **Workbook** y se encuentra en el módulo **ThisWorkbook** del libro Resumen.xlsx.*

```
Private Sub Workbook_Open()
'   Abre el libro Ventas.xlsx
   Workbooks.Open Filename:="C:\VENTAS\VENTAS.xlsx"
'   Activa el libro Resumen
   Windows("RESUMEN.xlsx").Activate
End Sub
```

4. Procedimientos Function

Los procedimientos Function, llamados comúnmente **funciones**, devuelven un valor resultado de un cálculo. El valor se devuelve a través del nombre de la función.

El lenguaje Visual Basic incluye numerosas funciones integradas, tales como las usadas en cálculos con fechas (day, week, year, format, etc.).

Además de estas funciones integradas, puede crear sus propias funciones personalizadas.

Ejemplo

La siguiente función pide al usuario que confirme su deseo de abandonar la aplicación y devuelve True si el usuario responde que Sí, y False en caso contrario.

```
Function Terminar() As Boolean
   If MsgBox("¿Desea salir del programa?", _
       vbQuestion + vbYesNo) = vbYes Then
       Terminar = True
   Else
       Terminar = False
    End If
End Function
```

5. Declaración de procedimientos

Sintaxis de un procedimiento Sub

```
[Private | Public | Friend] [Static] Sub NomProc
([lista de argumentos])
   <secuencia de instrucciones>
End Sub
```

Sintaxis de un procedimiento Function

```
[Private | Public | Friend] [Static] Function
NomProc ([lista de argumentos]) [As <Type>]
   <secuencia de instrucciones>
End Function
```

Para crear un procedimiento **Sub** o **Function**, se deben respetar los siguientes pasos:

- Determine el **alcance** del procedimiento.
- Declare el procedimiento según su tipo con la palabra clave **Sub** o **Function**, seguida del nombre del procedimiento.
- Defina los **argumentos** que se deban pasar como parámetros al procedimiento e indíquelos entre paréntesis después del nombre del procedimiento.
- En el caso de una función, si es necesario, indique el tipo del valor devuelto después de la palabra clave **As**.
- Escriba el código que permita efectuar la operación deseada. Si es necesario, use **Exit Sub** o **Exit Function** para salir del procedimiento. En el caso de una función, asigne el resultado al nombre de la función.
- Finalice el procedimiento con **End Sub** o **End Function**.

6. Alcance de los procedimientos

El alcance de un procedimiento determina la extensión de su uso.

Un procedimiento **Public** se puede llamar desde **todos los módulos** de todos los proyectos de Excel.

Un procedimiento **Private** solamente se puede llamar desde un procedimiento dentro del mismo módulo.

La palabra clave **Static** indica que las variables locales del procedimiento se mantienen entre una llamada y otra.

Si no se indica Public o Private o Friend, **los procedimientos son públicos por defecto**.

7. Argumentos de los procedimientos

Los argumentos se usan para transferir a los procedimientos parámetros en forma de datos. La cantidad de argumentos puede variar de 0 a varios.

Para declarar un argumento, basta con especificar su nombre. Sin embargo, la sintaxis completa para declarar un argumento es la siguiente:

```
[Optional] [ByVal | Byref] [ParamArray]
<variable> [As type]
```

La opción `Optional`	indica que el argumento es opcional y puede omitirse. Los argumentos opcionales se deben ubicar al final de la lista de argumentos y ser de tipo **Variant**.
La opción `Byval`	indica que el argumento se pasa por valor. El procedimiento accede a una copia de la variable; su valor inicial no se modifica por el procedimiento al que se pasa.
La opción `Byref` (opción por defecto)	indica que el argumento se pasa por referencia. En este caso, el procedimiento puede acceder a la variable propiamente dicha; de esta manera, su valor real se puede modificar por el procedimiento al que se pasa.
La palabra clave `ParamArray`	se usa únicamente como último argumento de la lista para indicar que se trata de una matriz opcional de elementos de tipo **Variant**. No se puede usar con las palabras clave `ByVal`, `ByRef` u `Optional`.
`Variable`	especifica el nombre del argumento. Para las variables de matriz, no especificar su dimensión.
`Type`	especifica el tipo de datos del argumento que se pasa al procedimiento (**Byte**, **Boolean**, **Integer**, **Long**, etc.).

8. Argumentos con nombre

La transferencia de argumentos a un procedimiento según su orden de aparición es a veces difícil de realizar, especialmente cuando hay parámetros opcionales. De la misma manera, la legibilidad de las llamadas a procedimientos con muchos parámetros no siempre es fácil.

Los argumentos con nombre facilitan la transferencia de argumentos gracias a las siguientes ventajas:

- El orden de los argumentos con nombre no es importante.
- Los argumentos opcionales pueden omitirse.

La sintaxis de los argumentos con nombres es:

```
NomArg := valor
```

Ejemplo

El siguiente código VBA:

```
If MsgBox("¿Desea salir de la aplicación?", _
   vbYesNo + vbQuestion, "Gestión de ventas") = vbYes Then
   Application.Quit
End If
```

puede transformarse en:

```
If MsgBox(Prompt:= "¿Desea salir de la aplicación?", _
   Buttons:=vbYesNo + vbQuestion, _
   Title:="Gestión de ventas") = vbYes Then
   Application.Quit
End If
```

También se puede modificar el orden de los parámetros:

```
If MsgBox(Prompt:= "¿Desea salir de la aplicación?", _
   Title:="Gestión de ventas", _
   Buttons:=vbYesNo + vbQuestion) = vbYes Then
   Application.Quit
End If
```

El nombre de los argumentos aparece automáticamente en el entorno VBE a medida que se escribe la instrucción. Los argumentos opcionales aparecen entre corchetes.

9. Llamar a un procedimiento

Sintaxis

```
[Call] NomProc [lista de argumentos]
```

Si se indica la palabra clave `Call`, debe colocar la lista de argumentos entre paréntesis.

- Para almacenar el resultado de una función en una variable, use la siguiente sintaxis:

```
<variable> = NomProc ( [lista de argumentos] )
```

- Para llamar a un procedimiento de otro módulo, use la siguiente sintaxis:

```
NomMódulo.NomProcedimiento
```

Ejemplo

```
ThisWorkbook.Salir_Apli
```

- Para llamar a un procedimiento de otro libro, use la siguiente sintaxis:

```
Application.Run "NomLibro!NomMódulo.NomProcedimiento"
```

Ejemplo

```
Application.Run "Ventas.xlsm!ThisWorkbook.Salir_Apli"
```

Para ejecutar este comando, el libro Ventas.xlsm debe estar abierto.

10. Llamar a una función VBA en una fórmula de Excel

Las funciones VBA se pueden usar en las fórmulas de Excel. Todas las funciones declaradas en **Public** están disponibles en el asistente para funciones de Excel (categoría **Funciones Definidas por el usuario**).

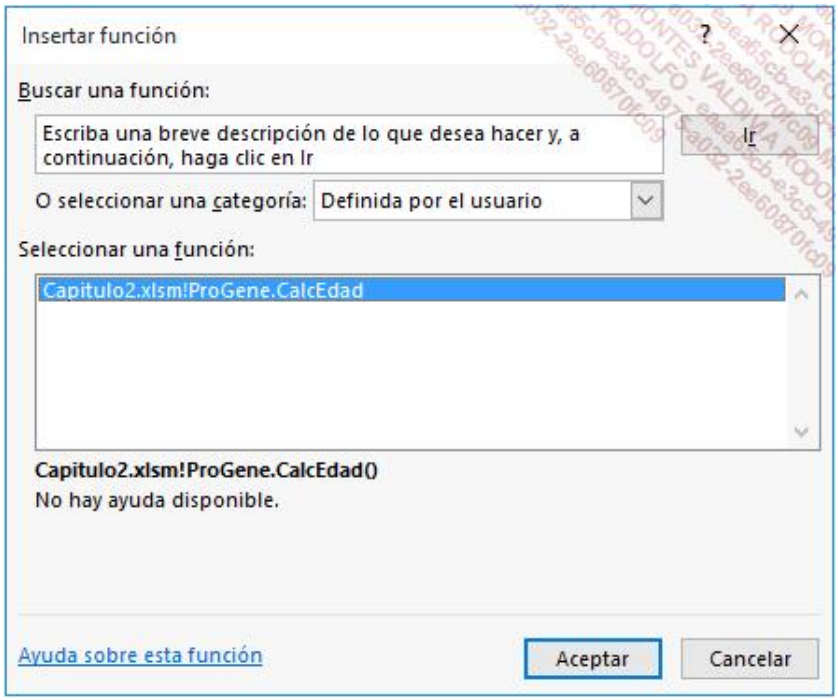
Ejemplo

Este ejemplo usa una función VBA que calcula la edad de una persona a partir de su fecha de nacimiento.

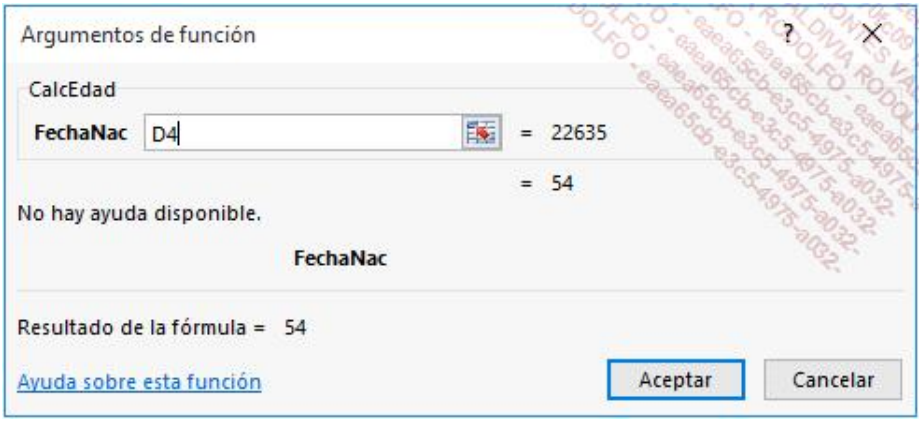
```
Function CalcEdad(fechaNac As Date)
   Dim zFecha As Date
   ' Calcula la edad en función de la fecha de nacimiento
   CalcEdad = Abs(DateDiff("YYYY", fechaNac, Date))
   zFecha = DateAdd("YYYY", CalcEdad,  fechaNac)
   If zFecha > Date Then CalcEdad = CalcEdad - 1
End Function
```

Para usar esta función en Excel:

- Seleccione la opción **Insertar función** de la pestaña **Fórmulas**.
- En el cuadro de diálogo **Insertar función**, en la lista **O seleccionar una categoría**, elija **Definida por el usuario**; la función CalcEdad estará ahora accesible:



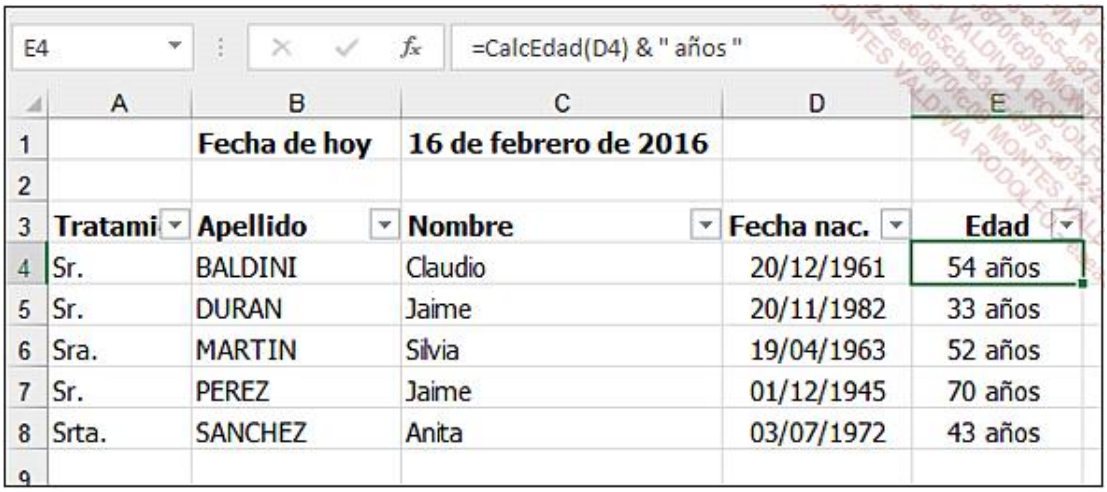
Seleccione la función y haga clic en **Aceptar**; el cuadro de diálogo le pedirá los argumentos de la función, como se ve aquí:



Este ejemplo muestra la importancia del nombre de los argumentos de las funciones: cuanto más explícitos sean estos, más fácil resultará usar la función en Excel.

Ahora puede modificar la fórmula para incluir el texto "años" y extender la fórmula a toda la lista.

El resultado en Excel es el siguiente:



11. Ejemplos de procedimientos y funciones

Copiar el contenido de una tabla de valores en la hoja de Excel activa.

```
Sub Mostrar_Tabla()
   Dim vTabVal As Variant
   Dim oCelda As Range
   Dim i As Integer
   ' Muestra el contenido de la tabla en la hoja de cálculo activa
   vTabVal = Array("Buenos Días", 1.244, "=A1+12", "=A2+12")
   For i = 0 To 3
      Set oCelda = Range("A" & i + 1)
      If MCell(oCelda, vTabVal(i)) Then
         MsgBox "La celda se actualizó con éxito"
      Else
         MsgBox "La celda no se pudo actualizar"
      End If
   Next i

End Sub
```

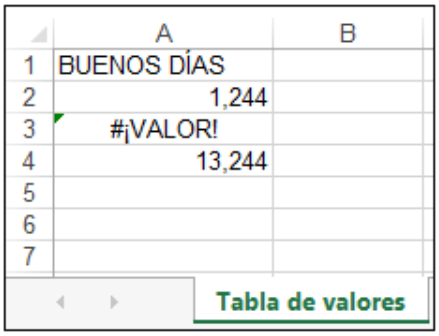
El código de estos ejemplos se debe escribir en un módulo estándar o en el módulo **ThisWorkbook**.

La función MCell informa sobre el valor asignado a una celda. Devuelve True si la celda se actualizó correctamente, y False en caso contrario.

```
Private Function MCell(oCelda As Range, _
            Valor As Variant) As Boolean
   ' Actualización de una celda a partir de un valor
   MCell = False
   If Not IsEmpty(oCelda) Then Exit Function
   oCelda.Value = Valor
   If oCelda.Text <> "#VALOR!" Then
      MCell = True
   End If
```

```
End Function
```

Si prueba este ejemplo, obtendrá el siguiente resultado:



	A	B
1	BUENOS DÍAS	
2	1,244	
3	#¡VALOR!	
4	13,244	
5		
6		
7		

Tabla de valores

La tercera celda no se pudo actualizar.

Variables

Las variables permiten almacenar valores intermedios durante la ejecución del código VBA para usarlos luego en cálculos, comparaciones, pruebas…

Las variables se identifican por un **nombre** que permite hacer referencia al valor que contienen y un **tipo** que determina la naturaleza de los datos que pueden almacenar.

1. Tipos de variables

Numéricas

Tipo	Rango	Tamaño en bytes
Byte	0 a 255	1
Integer (entero)	-32 768 a 32 767	2
Long (entero largo)	-2 147 483 648 a 2 147 483 647	4
Single (real simple de coma flotante)	-3,402823E38 a 1,401298E-45 (valores negativos) 1,401298E-45 a 3,402823E38 (valores positivos)	4
Double (real doble de coma flotante)	-1,79769313486231E308 a 4,94065645841247E-324 (valores negativos) 4,94065645841247E-324 a 1,79769313486231E308 (valores positivos)	8
Currency (monetario de punto fijo)	-922 337 203 685 477,5808 a 922 337 203 685 477,5807	8
Decimal	+/-79 228 162 514 264 337 593 543950 335 sin separador decimal; +/-7,9228162514264337593543950335 con 28 cifras a la derecha del separador decimal; el menor número distinto de cero es +/-0.0000000000000000000000000001	12

Cadenas de caracteres

El tipo es **String**. Existen dos tipos de cadenas:

- Las cadenas de longitud variable pueden contener aproximadamente dos mil millones de caracteres.
- Las cadenas de longitud fija pueden contener de 1 a aproximadamente 64 KB de caracteres.

Ejemplo

```
' Cadena de longitud variable
Dim sDomicilio As String
' Cadena de longitud fija (20 caracteres)
Dim sNombre As String * 20
```

Boolean o lógica

El tipo es **Boolean** (o booleano). La variable puede tomar los valores True (Verdadero) o False (Falso), que es su valor por defecto. Ocupa dos bytes.

Fecha

El tipo es **Date**. La variable puede tomar los valores de fecha y de hora del 1 de enero del año 100 al 31 de diciembre de 9999. Ocupa ocho bytes.

Variant

Las variables de tipo Variant pueden contener datos de todo tipo, además de los valores especiales **Empty**, **Error** y **Null**.

Usar el tipo de dato Variant ofrece más flexibilidad en el tratamiento de datos. Por ejemplo, si una variable de tipo Variant contiene cifras, se puede usar su valor real o su representación en forma de cadena, según el contexto.

De todas formas, las variables de tipo Variant requieren 16 bytes de memoria para números y 22 bytes más la longitud de la cadena para los caracteres; esto puede ser perjudicial en el caso de procedimientos largos o en módulos complejos.

Ejemplo

```
Sub Variable_Variant()
'  Declaración de la variable "Valx" como Variant
    Dim Valx As Variant
'  Asignación de una sucesión de valores a la variable
'  y mostrar el tipo del resultado
'  10 retorna un valor de tipo Integer
    Valx = 10
    MsgBox Valx & " es de tipo " & TypeName(Valx)
'  Ejemplo que retorna un valor de tipo String
    Valx = "Ejemplo"
    MsgBox Valx & " es de tipo " & TypeName(Valx)
'  Esta multiplicación retorna un valor de tipo Double
    Valx = 12500.32 * 1E+21
    MsgBox Valx & " es de tipo " & TypeName(Valx)
'#1/1/99# retorna un valor de tipo Date
    Valx = #1/1/99#
    MsgBox Valx & " es de tipo " & TypeName(Valx)
'  True retorna un valor de tipo Boolean
    Valx = True
    MsgBox Valx & " es de tipo " & TypeName(Valx)

End Sub
```

Objeto

El tipo es **Object**. Para crear una variable que contenga un objeto, comience por declarar la variable como tipo

Objeto y luego asígnele un objeto.

- Para declarar una variable Objeto:

Si el tipo de objeto no se conoce, use la sintaxis:

```
InstrucciónDeDeclaración NomVariable As Object
```

Si se conoce el tipo de objeto, use la sintaxis:

```
InstrucciónDeDeclaración NomVariable As TipoObjeto
```

Ejemplo

```
Sub Variables_Objeto()
'  oTest se declara como objeto
'  ShtNomCli se declara como hoja de cálculo
'  oGrafico se declara como gráfico
    Dim oTest As Object
    Dim ShtNomCli As Worksheet
    Dim oGrafico As Chart
End Sub
```

- Para asignarle un objeto a una variable Objeto, use la instrucción **Set**:

```
Set NomVariable = ObjetoaAsignar
```

Ejemplo

Declaración de una variable ZonaTest destinada a contener un objeto Range y asignación de las celdas A6 a B15 a esta variable:

```
Dim oZonaTest As Range
Set oZonaTest = Range("A6:B15")
```

- Para finalizar la asociación entre una variable y un objeto determinado, use la siguiente sintaxis:

```
Set NomVariable = Nothing
```

Definido por el usuario (o personalizado)

Los tipos de datos personalizados se crean con la instrucción **Type** usada a nivel de módulo.

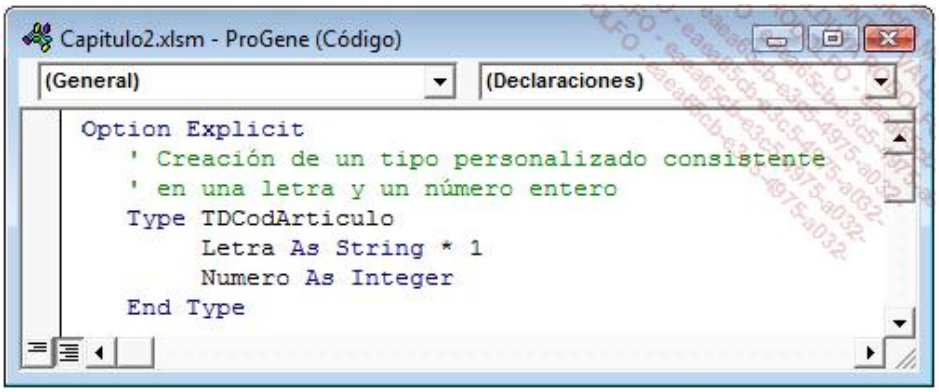
Sintaxis

```
Type NomTipoPerso
     NomElemento1 As TipoDatos
     NomElemento2 As TipoDatos
     ...
End Type
```

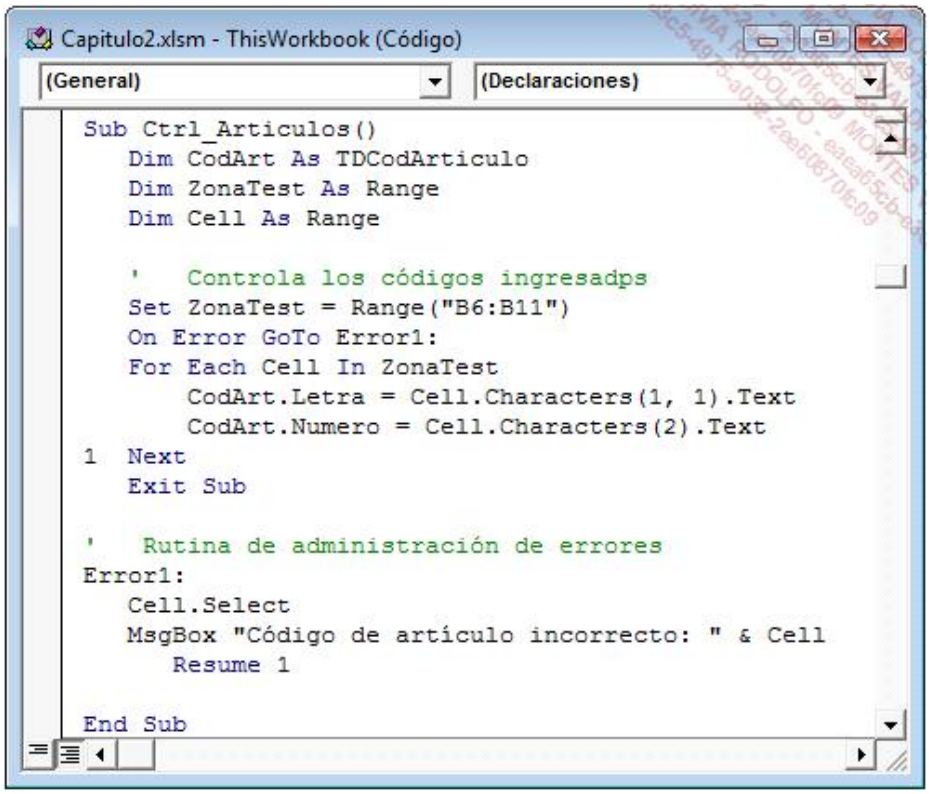
La definición del tipo solamente se puede hacer en la sección de declaración de un módulo.

Ejemplo

Declaración de un tipo personalizado constituido por una letra y un número entero en el módulo de código ProGene.



Uso del tipo personalizado para controlar los códigos de artículos introducidos de la celda B6 a la celda B11 de la hoja de cálculo activa. En caso de error, se muestra un mensaje.



2. Declaración de variables

Para crear una variable, debe declararla, es decir, darle un nombre. Luego puede usar ese nombre para modificar el valor de la variable, usar ese valor, etc.

La declaración de variables en VBA puede ser **implícita** o **explícita**.

a. Declaraciones implícitas

Se hacen directamente al asignar un valor a un nombre de variable. El tipo de datos será entonces el tipo por defecto, o sea, **Variant**.

Ejemplo

```
i = 12
dImporte = 12000
sNombre = "Juan Carlos"
```

b. Declaraciones explícitas

Requieren el uso de una **instrucción de declaración** (Dim, Public, Private, etc.). Si el tipo de la variable no se indica, la variable resultará del tipo por defecto, o sea, **Variant**.

Se puede imponer la declaración implícita de variables usando la instrucción **Option Explicit** en la sección de declaración de cada módulo. Para insertar esta instrucción automáticamente en cada nuevo módulo, active la opción **Requerir declaración de variables** del menú **Herramientas** - **Opciones** - pestaña **Editor**.

Ejemplo

```
Dim I
Private dImporte As Double
Public sNombre as String
```

Para optimizar la velocidad de ejecución del código VBA, se recomienda declarar las variables en forma explícita.

c. Sintaxis de las instrucciones de declaración

```
<InstrucciónDeclaración> NomVariable [As <TipodeDatos>]
```

Donde `InstrucciónDeclaración` corresponde a una de las siguientes cuatro instrucciones: `Dim`, `Public`, `Private` o `Static`.

`Dim`	Las variables declaradas con la instrucción `Dim` **a nivel de módulo** están disponibles para todos los procedimientos del módulo. No son accesibles desde ningún otro módulo. Las variables declaradas con la instrucción `Dim` **a nivel de procedimiento** solamente están disponibles dentro del procedimiento.
`Private`	**Solamente a nivel de módulo**. Las variables `Private` solamente están disponibles para el módulo en el que son declaradas.
`Public`	**Solamente a nivel de módulo**. Las variables declaradas con la instrucción `Public` son accesibles desde **el conjunto de módulos** de todos los proyectos de Excel abiertos. Si la instrucción `Option Private Module` se especifica en la sección de declaración del módulo, las variables solamente son públicas dentro del proyecto que las recibe.
`Static`	**Solamente a nivel de procedimiento**. Las variables declaradas con la instrucción `Static` conservan su valor mientras dure la ejecución del código.

3. Declaración de los tipos de variables

a. Declaraciones explícitas del tipo

El tipo de la variable se especifica en la declaración, tras la palabra clave **As**.

Ejemplo

```
Sub TotalAcum()
    Dim iTotal As Integer
    Static iAcum As Integer
    iTotal = iTotal + 10
    iAcum = iAcum + 10
'  Devuelve 10 en cada ejecución del procedimiento
    MsgBox iTotal
'  Devuelve 10 en la primera ejecución, luego 20 la segunda vez,
'  30 la tercera, etc.
    MsgBox iAcum
End Sub
```

Puede declarar más de una variable en una misma instrucción, pero atención, el tipo de datos solo se tendrá en cuenta para la última variable, el tipo Variant se asignará a las otras.

b. Declaraciones implícitas del tipo

El tipo de variable se declara usando un sufijo en el momento de su utilización o por la instrucción **DefType**.

Empleo de un sufijo

Debe agregar uno de los siguientes caracteres al nombre de la variable:

Sufijo	Tipo de datos
%	Integer
&	Long
!	Single
#	Double
@	Currency
$	String

Ejemplo

Declara la variable como tipo Cadena (String).

```
Dim Nombre$
```

Declara la variable como tipo Monetario (Currency).

```
Dim Deuda@
```

DefType

Estas instrucciones se emplean en la zona de declaración del módulo, para definir los tipos de datos por defecto de las variables cuyos nombres comienzan por los caracteres especificados.

Lista de instrucciones **DefType**:

Instrucción	Tipo de datos
DefBool	Boolean
DefDbl	Double
DefInt	Integer
DefDate	Date
DefLng	Long
DefStr	String
DefCur	Currency
DefObj	Object
DefSng	Single
DefVar	Variant
DefByte	Byte

Ejemplo

Todas las variables cuyos nombres comienzan por una letra comprendida entre I y K y por la letra N son variables de tipo entero (Integer).

```
DefInt I-K,N
```

Las variables que comienzan por una letra comprendida entre A y H serán de tipo cadena (String).

```
DefStr A-H
```

La instrucción siguiente declara la variable Identificador como tipo Variant y las variables Superficie y Latitud como tipo Entero.

```
Dim Identificador, Superficie As Integer
Dim Latitud As Integer
```

c. Convención de nombres de variables

Se recomienda utilizar una convención de nombres para las variables. Permiten estandarizar el código y hacerlo más legible y más fácil de mantener por diferentes desarrolladores.

La convención consiste en utilizar dos prefijos en el nombre de las variables:

- El primero para indicar el ámbito de la variable: p o pb o pb_ para pública, m o mo para módulo, y ningún prefijo para las locales.
- La segunda para indicar su tipo.

Tipo	Prefijo	Ejemplo
Boolean	b o bln	bVisible o blnVisible
Double	d o dbl	dTotal o dblTotal
Integer	i o int	iEdad o intEdad
Long	l o lng	lCanTot o lngtCanTot
Object	o u obj	oCelda u objCelda
String	s o str	sNombre o strNombre
Variant	v o var	vParam o varParam

4. Matrices

Puede crear una variable matriz cuando necesite trabajar con un grupo de valores relacionados.

Para crear una variable matriz, use la siguiente sintaxis:

```
<InstrucciónDeDeclaración> NomMatriz(índices) [As<TipodeDatos>]
```

Donde para **(índices)**:

- Si se omite este argumento: la matriz tendrá dimensión libre.
- Si se indica un valor: la matriz tendrá un número determinado de elementos.
- Si se indica `LimiteInf To LimiteSup`: la matriz tendrá un número de elementos determinado y números de índice específicos.

Por defecto, el valor menor del índice de una matriz es 0.

Para modificar el valor menor del índice, use la instrucción **`Option Base`** `MenorValorÍndice` en la sección de declaración del módulo o use la sintaxis `LimiteInf` **`To`** `LimiteSup` en el argumento **(índices)**.

Para definir el contenido de una matriz, use la función **Array** (la variable debe ser una tabla de dimensión libre) o despliegue los datos en una hoja o asigne una por una cada variable de la matriz usando los índices.

Ejemplo

El siguiente ejemplo muestra, en la hoja activa de Excel, la lista de los factores de conversión a euros para cinco países.

```
Const FacFRF = 6.55957
Const FacBEF = 40.3399
Const FacDEM = 1.95583
Const FacESP = 166.386
```

```
Const FacITL = 1936.27
---------------------------------------------------------------
Sub Muestra_Factor()
   Dim Pais As Variant
   Dim Factor(5) As Double
   Dim i As Integer
   ’ Lista de países
   Pais = Array("Francia", "Bélgica", "Alemania", _
           "España", "Italia")

   ’ Lista de los factores por país
   Factor(0) = FacFRF
   Factor(1) = FacBEF
   Factor(2) = FacDEM
   Factor(3) = FacESP
   Factor(4) = FacITL

   ’ Muestra los países y sus factores en la hoja de cálculo Euro
   For i = 0 To 4
        With Sheets ("Euro")
              .Cells(i + 1, 1) = Pais(i)
              .Cells(i + 1, 2) = Factor(i)
        End With
   Next i

End Sub
```

En este ejemplo, también es posible usar una matriz de dos índices. El código VBA del procedimiento queda como sigue:

```
Sub Muestra_Factor2
’ Matriz de dos índices
   Dim TabFactor(5, 1)
   Dim i As Integer

   ’ Lista de países y sus factores
   TabFactor(0, 0) = "Francia"
   TabFactor(0, 1) = FacFRF
   TabFactor(1, 0) = "Bélgica"
   TabFactor(1, 1) = FacBEF
   TabFactor(2, 0) = "Alemania"
   TabFactor(2, 1) = FacDEM
   TabFactor(3, 0) = "España"
   TabFactor(3, 1) = FacESP
   TabFactor(4, 0) = "Italia"
   TabFactor(4, 1) = FacITL

   ’ Muestra los países y sus factores en la hoja de cálculo Euro
   For i = 0 To 4
        With Sheets ("Euro")
            .Cells(i + 1, 1) = TabFactor(i, 0)
            .Cells(i + 1, 2) = TabFactor(i, 1)
        End With
   Next i
```

```
End Sub
```

5. Constantes

Una constante permite asignar un nombre explícito a un valor.

a. Constantes personalizadas

La declaración de una constante se hace con la instrucción **Const** en la sección de declaración de un módulo o en un procedimiento.

```
Const NomConstante [As <TipoDato>] = <expresión>
```

As TipoDato	El tipo de dato no puede ser un objeto (Object) ni un tipo personalizado (Type).
expresión	No puede ser una función definida por el usuario, ni una función intrínseca de Visual Basic.

Ejemplo

Declaración de algunas constantes.

```
Sub Constantes()
   ’  Ejemplos de constantes autorizadas:
    Const Val1 = "Mega+"
    Const Val2 = 148
    Const Val3 = 125.45
   ’  El tipo de datos es Double
    Const Val4 As Single = 125.45
   ’  Permite ahorrar memoria
     Const Val5 = Val2 * Val3
     Const Val6 = Val1 & " cuesta " & Val2
   ’  Ejemplo de constante no autorizada:
   ’  Por el uso de una función VB
    Const Val6 = Fix(Val4)
End Sub
```

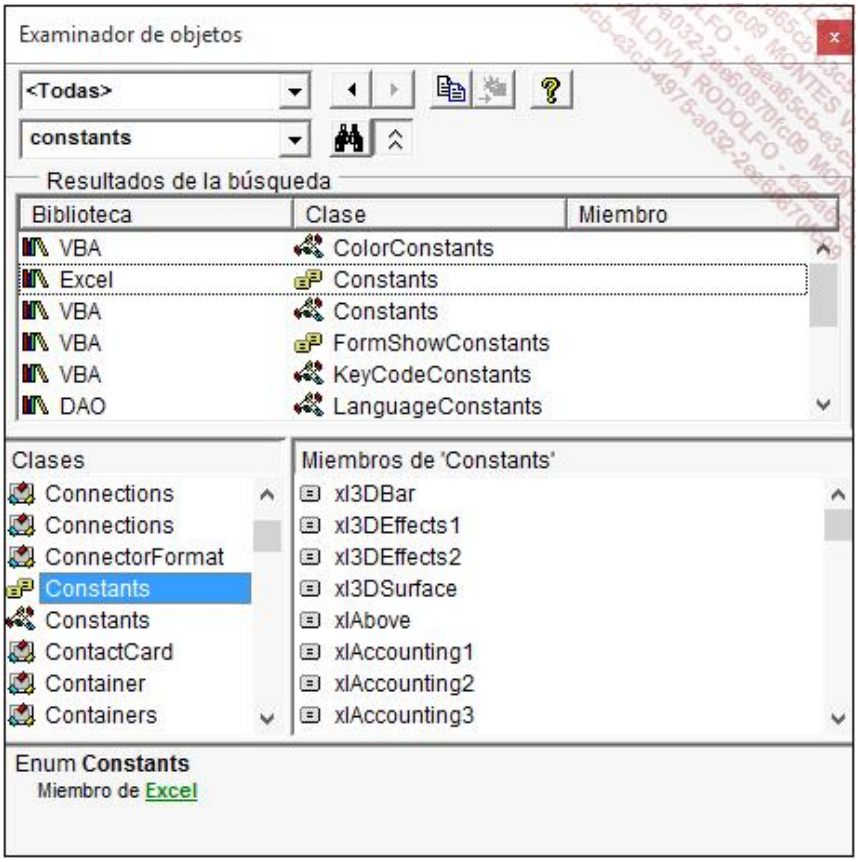
Para crear una constante accesible a todos los libros abiertos, se debe declarar en la sección de declaración de un módulo y situar la instrucción **Public** antes de la instrucción **Const**.

b. Constantes integradas

Las constantes usadas por los objetos de Microsoft Excel van precedidas por las letras "xl"; las constantes usadas con otras instrucciones y funciones de Visual Basic van precedidas por las letras "vb", y las constantes de Microsoft Office van precedidas por las letras "mso".

- Para mostrar la lista de constantes integradas, abra el examinador de objetos haciendo clic en el icono  o pulsando la tecla de función [F2]. Escriba la palabra **constants** en la lista desplegable **Texto de**

búsqueda (la segunda situada a la izquierda) y luego haga clic en el icono  .



Estructuras de decisión

Es conveniente testear las condiciones específicas antes de ejecutar las instrucciones.

Las estructuras de decisión, llamadas también alternativas o bifurcaciones condicionales, permiten, tras una evaluación, optar por uno u otro bloque de código.

Se distinguen dos instrucciones de bifurcación condicional:

- If ... Then ... Else
- Select ... Case

La función IIf también se puede usar para definir un valor en función de una condición. Ejemplo: Port = IIf(Cantidad < 100, 100, 0).

1. Instrucción If

Permite ejecutar ciertas instrucciones en función del resultado de una condición.

If...Then

```
If <condición> Then <instrucción> [:<instrucción>]
```

Si hay varias instrucciones, sepárelas por el signo de puntuación : (dos puntos). Esta sintaxis se usa especialmente para pruebas cortas y simples.

Ejemplo

Si la celda A1 está vacía, emite un bip y muestra un mensaje.

```
Sub Test_Celda_A1()
  If IsEmpty(Range("A1")) Then Beep: MsgBox "Olvidó el título"
End Sub
```

If...Then...End If

```
If <condición> Then
   <instrucción1>
   <instrucción2>
...
End If
```

Ejemplo

```
Sub Test_Titulo()
'  Si la celda A1 no está vacía
'  entonces ponerla en negrita y pintarla de rojo
```

```
    If Not IsEmpty(Range("A1")) Then
        With Range("A1")
            .Font.Bold = True
            .Interior.ColorIndex = 3
        End With
    End If
End Sub
```

If...Then...Else...End If

```
If <condición> Then
   <instrucciones>
Else
   <instrucciones>
End If
```

Ejemplo

Al cambiar la moneda (€ o US$) de la celda C3, modificar el formato del rango D6:F11.

```
Private Sub Workbook_SheetChange(ByVal Sh As Object, _
ByVal Target As Range)
'  Modifica la celda C3 de la hoja Artículos
If Sh.Name = "Artículos" And Target.Address = "$C$3" Then
   Aplicar_Formato
End If
End Sub
```

Este procedimiento modifica el formato de las celdas en función de la moneda elegida.

```
Sub Aplicar_Formato()
'   Formato € o $
Range("D6:F11").NumberFormat = "0.00"
If UCase(Range("C3")) = "EURO" Then
   Range("D6:F11").NumberFormat = "0.00 €"
Else
   Range("D6:F11").NumberFormat = "0.00"" $"""
End If
End Sub
```

If... Then...ElseIf...Else...End If

```
If <condición> Then
   <instrucciones>
ElseIf <condición> Then
   <instrucciones>
ElseIf <condición> Then
   <instrucciones>
...
```

```
        ...
Else
   <instrucciones>
End If
```

Ejemplo

Este procedimiento modifica los textos de las celdas seleccionadas: si la última letra es minúscula, pasa todo a mayúsculas; si no, pasa todo a minúsculas con la primera letra en mayúscula.

```
Sub Mayus_Minus()
   Dim oCelda As Range
   Dim iCodAscii As Integer

   '   Recorre las celdas de la selección
   For Each oCelda In Selection
      If IsEmpty(oCelda) Then
          Beep
          MsgBox "La celda " & oCelda.Address & " está vacía"
      Else
         '   Código Ascii de la última letra
         iCodAscii = Asc(Right(oCelda, 1))
         '   Si está en mayúsculas
         If iCodAscii >= 65 And iCodAscii <= 90 Then
            oCelda.Value = UCase(Left(oCelda, 1)) _
                           & LCase(Right(oCelda, Len(oCelda) - 1))
         '   Si está en minúsculas
         ElseIf iCodAscii >= 97 And iCodAscii <= 122 Then
               oCelda.Value = UCase(oCelda)
         Else
               MsgBox "El último carácter de la celda: " _
               & oCelda.Address & " no es una letra"
         End If
      End If
   Next
End Sub
```

2. Instrucción Select Case

Select Case

Ejecuta una secuencia de instrucciones específicas en función del valor de una expresión.

```
Select Case <ExpresiónTest>
   Case <ListaExpresiones>
        <instrucciones>
   Case <ListaExpresiones>
        <instrucciones>
...
   ...
   Case Else
```

```
        <instrucciones>
End Select
```

<listaExpresiones> puede tomar una de las siguientes formas:

- valor (ejemplo: `Case 10`)
- lista de valores (ejemplo: `Case 1, 5, 10`)
- rango de valores (ejemplo: `Case 1 To 5`)
- expresión condicional (ejemplo: `Case Is >= 5`)

Ejemplo

Llamada a un procedimiento que determina la fórmula de cálculo del total en función de la cantidad, del precio y del flete.

```
Private Sub Calc_Total()
   '   Llama a la función Total usando como parámetros
   '   las celdas Cant, Precio y Flete
   Range("D2") = Total(Range("A2"), Range("B2"), Range("C2"))
End Sub
```

La función Total devuelve una fórmula de Excel. El flete es gratuito a partir de dos unidades pedidas; el porcentaje de descuento también depende de la cantidad pedida.

```
Function Total(oCant As Range, oPrecio As Range, oFlete As Range) _
As String

   '   Determinación de la fórmula de cálculo del total
   '   en función de la cantidad ordenada
   Select Case oCant
      Case 1
          Total = "=" & oPrecio.Address & "+" & oFlete.Address
      Case 2 To 10
          Total = "=" & oCant.Address & "*" & oPrecio.Address
      Case 11 To 100
          Total = "=" & oCant.Address & "*" & oPrecio.Address & "* 0.95"
      Case 101 To 1000
           Total = "=" & oCant.Address & "*" & oPrecio.Address & "* 0.9"
      Case Else
           Total = "Error en la cantidad"
   End Select
End Function
```

Estructuras en ciclo

Las estructuras en ciclo (o repetitivas) permiten repetir la ejecución de un conjunto de instrucciones.

Se distinguen varios tipos de estructuras en ciclo:

- Do...Loop
- While...Wend
- For...Next
- For Each...Next

Do...Loop y **While...Wend** repiten las operaciones en función de una cierta condición, mientras que **For...Next** repite las operaciones una cantidad de veces determinada por un contador.

For Each...Next permite recorrer los elementos de una colección.

1. Instrucción Do...Loop

Ejecuta un bloque de instrucciones un número indeterminado de veces.

Sintaxis 1

Las instrucciones se ejecutan, mientras que la condición devuelve el valor True.

```
Do While <Condición>
      <Instrucciones>
Loop
```

Sintaxis 2

Las instrucciones se ejecutan una primera vez sin condición y, luego, mientras la condición devuelva True.

```
Do
    <Instrucciones>
Loop While <Condición>
```

Ejemplo

El siguiente código solicita al usuario que escriba un número mientras que el valor introducido no sea numérico o superior a 100.

```
Sub Introducir_Numero ()
   Dim vRespuesta as Variant
   Do
      vRespuesta = InputBox("Introduzca un número > 100")
   Loop While (Not IsNumeric(vRespuesta) Or vRespuesta <= 100)
End Sub
```

Sintaxis 3

Las instrucciones se ejecutan hasta que la condición toma el valor True (mientras que la condición devuelva el valor False).

```
Do Until <Condición>
    <Instrucciones>
Loop
```

Sintaxis 4

Las instrucciones se ejecutan una primera vez sin condición y luego hasta que la condición devuelva el valor True.

```
Do
    <Instrucciones>
Loop Until <Condición>
```

Ejemplo

El siguiente código solicita al usuario que escriba un número hasta que el valor introducido sea numérico y mayor que 100.

```
Sub Introducir_Numero()
   Dim vResponse as Variant
   Do
      vResponse = InputBox("Introduzca un número > 100")
   Loop Until (IsNumeric(vResponse) And vResponse > 100)
End Sub
```

2. Instrucción While...Wend

Ejecuta una serie de instrucciones en un ciclo mientras se cumple la condición especificada.

Sintaxis

```
While <condición>
    <instrucciones>
Wend
```

Ejemplo

```
Sub Introducir_Precio()
   '   Pide la introducción de un precio mientras que
   '   sea vacío o incorrecto
   While IsEmpty(Range("C9")) Or Not IsNumeric(Range("C9"))
      Range("C9") = InputBox("Introduzca el precio del producto")
   Wend
End Sub
```

3. Instrucción For...Next

Ejecuta un bloque de instrucciones según el valor de un contador.

```
For <contador>=<inicio> To <fin> [Step <incremento>]
    <instrucciones>
Next
```

Ejemplo

El siguiente código muestra un procedimiento que inserta los totales trimestrales en una matriz de resultados mensuales; además, el procedimiento elimina los totales si ya los había.

```
Sub Totales_Trimestrales()
   Dim iTrim As Integer
   Dim i As Integer
   Dim oCelda as Range
   '   Inserta los totales trimestrales cada 3 meses,
   '   los pone en negrita con un borde
   iTrim = 1
   For i = 5 To 17 Step 4
       If Left(Cells(2, i), 4) <> "Trim" Then
          Cells(2, i).EntireColumn.Insert
          Cells(2, i) = "Trim. " & iTrim
          iTrim = iTrim + 1
          Range(Cells(3, i), Cells(11, i)).Select
          Selection.FormulaR1C1 = "=SUM(RC[-3]:RC[-1])"
          Range(Cells(2, i), Cells(11, i)).Font.Bold = True
          For Each oCelda In Range(Cells(2, i), Cells(11, i))
               oCelda.BorderAround ColorIndex:=1, Weight:=xlThin
          Next oCelda
        End If
      Range("A1").Activate
   Next
End Sub
_____________________________________________________________
Sub Suprime_Totales_Trimestrales()
   Dim i As Integer

   '   Suprime los totales trimestrales si ya los había
   For i = 5 To 17
         If Left(Cells(2, i), 4) = "Trim" Then
            Cells(2, i).EntireColumn.Delete
         End If
   Next i
End Sub
```

Este procedimiento muestra en la hoja de cálculo "Colores" los diferentes colores de relleno y el valor de la propiedad ColorIndex correspondiente.

```
Sub Muestra_Colores()
   Dim i As Integer
      With Sheets("Colores")
           For i = 1 To 56
```

```
            Cells(i, 1).Interior.ColorIndex = i
            Cells(i, 2) = i
        Next i
    End With
End Sub
```

4. Instrucción For Each...Next

Ejecuta un bloque de instrucciones para cada elemento de una colección de objetos o de una matriz.

```
For Each <elemento> In <Grupo>
    <Instrucciones>
Next <elemento>
```

Ejemplo

Este procedimiento aplica un color de letra a las celdas en función de su contenido.

```
Sub Colores_Celda()
   Dim oZonaaModificar As Range
   Dim oCelda As Range

' Aplica un color en función del valor de la celda
   Set oZonaaModificar = Range("B3:Q11")
   For Each Celda In oZonaaModificar
        Select Case oCelda
            Case Is < 1000
                oCelda.Font.Color = RGB(150, 150, 250)
            Case Is < 5000
                oCelda.Font.Color = RGB(90, 100, 250)
            Case Is < 10000
                oCelda.Font.Color = RGB(10, 20, 250)
            Case Is < 20000
                oCelda.Font.Color = RGB(5, 10, 175)
            Case Else
                oCelda.Font.Color = RGB(5, 5, 100)
       End Select
   Next
End Sub
```

Ejecución de varias acciones sobre un objeto

```
With objeto
    <Instrucciones>
End With
```

Ejemplo

```
Sub Paginacion()
```

```
'  Define la paginación de la hoja activa
'  Redimensiona las columnas y procede con la impresión
   With ActiveSheet
      With .PageSetup
           .Orientation = xlLandscape
           .LeftMargin = Application.InchesToPoints(0.5)
           .RightMargin = Application.InchesToPoints(0.5)
           .TopMargin = Application.InchesToPoints(0.5)
           .BottomMargin = Application.InchesToPoints(0.5)
           .LeftHeader = ""
           .CenterHeader = "&A"
           .RightHeader = ""
           .LeftFooter = ""
           .CenterFooter = "Page &P"
           .RightFooter = ""
       End With
       .Columns("A:Q").EntireColumn.AutoFit
       .PrintOut
   End With
End Sub
```

5. Salir de las estructuras de control

La instrucción **Exit For** permite salir directamente de un ciclo **For** o **For Each**, mientras que **Exit Do** sale directamente de un ciclo **Do**.

Ejemplo

```
Sub Introducir_Fecha()
   Dim vValor as Variant
   '  Fuerza la introducción de una fecha en la celda A1
   '  Si no se introduce ningún valor: se sale del ciclo
   Range("A1") = ""
   Do While Not IsDate(Range("A1"))
        strVal = InputBox("Escriba una fecha")
        If vVal <> "" Then
           If IsDate(vVal) Then Range("A1") = vVal
        Else
            Exit Do
        End If
   Loop
End Sub
```

Reglas de escritura del código

1. Comentarios

Los comentarios permiten documentar el código VBA para hacerlo más legible.

```
REM comentario
```

o

```
' comentario
```

Al validar la línea de comentario, esta se muestra, por defecto, en verde.

2. Carácter de continuación

Una instrucción VBA puede escribirse en muchas líneas usando un guión bajo "_" precedido de un espacio.

Ejemplo

```
'   Pide la introducción de un precio en tanto que
'   esté vacío o sea incorrecto
Dim vPrecio as Variant
Do While IsEmpty(vPrecio) Or Not IsNumeric(vPrecio) _
   Or vPrecio < 50 Or vPrecio > 500
   vPrecio = InputBox("Escribir un importe comprendido entre " _
          & "50 y 500 ")
Loop
```

3. Sangrías

Las sangrías (o tabulaciones) permiten una mayor legibilidad del código. Es especialmente importante usarlas en las estructuras de control (sobre todo si hay varias instrucciones If anidadas) y las estructuras de decisión.

- Para generar las sangrías, use la tecla [Tab].
- Para retroceder a la tabulación precedente, use las teclas [Mayús][Tab].
- Para modificar el tamaño de la tabulación (cuatro espacios por defecto), seleccione **Opciones** en el menú **Herramientas**, haga clic en la pestaña **Editor** y modifique el valor **Ancho de tabulación**.

4. Nombres de los procedimientos, variables y constantes

Los nombres de los procedimientos, constantes, variables y argumentos deben respetar las siguientes reglas:

- El primer carácter debe ser una letra.
- No se diferencian minúsculas de mayúsculas (se aceptan letras acentuadas), aunque se respetan unas y otras.

- No se usan nombres reservados en Visual Basic o palabras clave con restricciones.
- No se usan el punto, el espacio ni los signos !, $, # y @.
- Un nombre no puede tener más de 255 caracteres.
- Para los procedimientos **Function**, no se usa un nombre igual a una referencia de celda.
- No se indican varias veces los mismos nombres de variables y de constantes en un mismo nivel de alcance.

Operadores

Los operadores permiten realizar operaciones aritméticas con variables o constantes, comparar variables entre ellas, evaluar varias condiciones, etc.

Se distinguen varios tipos de operadores:

- Operadores aritméticos.
- Operadores de comparación.
- Operadores lógicos.
- Operador de concatenación.

El operador de asignación es el signo =. El valor de la expresión situada a la derecha del signo igual se asigna a la variable situada a la izquierda del signo (ejemplo: IntA = 12, IntA = Intb * 12).

1. Operadores aritméticos

Permiten efectuar cálculos aritméticos con variables o constantes.

Operador	Cálculo realizado
+	Adición
-	Sustracción
/	División con resultado de un número con coma flotante
Mod	Resto de la división entre dos números
\	División con resultado entero
*	Multiplicación
^	Potenciación

2. Operadores de comparación

Comparan dos valores o dos cadenas de caracteres.

Operador	Cálculo realizado
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
=	Igual a
<>	Distinto de

La instrucción **Option Compare** utilizada a nivel de módulo permite declarar el método de comparación por defecto que conviene usar en la comparación de cadenas.

Puede tomar uno de estos tres valores posibles:

- La opción **Compare Binary** (opción por defecto) realiza la comparación de cadenas basada en el orden derivado de la representación binaria interna de los caracteres: A < B < E < Z < a < b < e < z < Á < Ê < Ø < á < ê...
- La opción **Compare Text** realiza la comparación de cadenas sin distinguir mayúsculas de minúsculas: (A=a) < (Á=á) < (B=b) < (E=e) < (Ê=ê) < (Z=z) < (Ø=ø)...
- La opción **Compare Database** realiza la comparación de cadenas basada en el orden determinado por el identificador de parámetros regionales de la base de datos en la que se realiza la comparación de cadenas.

3. Operadores lógicos

Permiten evaluar simultáneamente dos (o más) valores booleanos o expresiones que devuelven este tipo de valor. Generalmente se usan con la instrucción If.

Operador	Cálculo realizado
AND	Si todas las expresiones tienen el valor True, el resultado es True. Si una de las expresiones tiene el valor False, el resultado es False.
OR	Si por lo menos una de las expresiones tiene el valor True, el resultado es True (o inclusivo).
XOR	Si una y solo una de las expresiones tiene el valor True, el resultado es True (o exclusivo).
NOT	Devuelve el contrario de la expresión.
Eqv	Devuelve True si las dos expresiones son idénticas.

Ejemplo

```
(A>=1) AND (A=<9) devuelve True si A está comprendido entre 1 y 9,
NOT (A >= 10) devuelve True si A es estrictamente menor que 10,
(A>0) OR (B>0) OR (C>0) devuelve True si al menos uno de los
valores es positivo.
```

4. Operador de concatenación

El operador de concatenación es el signo **&**. Engancha cadenas de caracteres, valores y expresiones.

Ejemplo

Concatenación del apellido y el nombre.

```
SApeNom = sApellido & " " & sNombre
```

5. Prioridad de los operadores

Cuando hay varios operadores en una misma expresión, cada uno de ellos se evalúa en un orden predeterminado, llamado prioridad de los operadores.

Los operadores se evalúan en el siguiente orden: operadores aritméticos, operadores de comparación, operadores lógicos.

Los operadores de comparación tienen la misma prioridad; es decir, son evaluados por orden de aparición, de izquierda a derecha.

Los operadores aritméticos y lógicos se evalúan en el siguiente orden de prioridad:

Aritmético	Lógico
^	Not
*, /, \	And
Mod	Or
+, -	Xor
	Eqv

El uso de paréntesis permite modificar la prioridad para que un elemento de una expresión sea evaluado antes que los otros. Las operaciones encerradas entre paréntesis siempre se evalúan antes que las otras.

Ejemplo

```
La expresión "3 + 4 * 5" da como resultado 23.
La multiplicación (4 * 5) se efectúa antes que la adición ( + 3)
La expresión "(3 + 4) * 5" devuelve 35. La adición se efectúa con
prioridad.
```


Se aconseja usar paréntesis para mejorar la legibilidad del código.

Presentación

VBA Excel es un lenguaje de programación **orientado a objetos**, si bien no dispone de todas las funcionalidades de los lenguajes de este tipo.

La mayoría de los elementos que maneja Excel son objetos: los libros, las hojas de cálculo, los rangos de celdas, las celdas, etc.

Los objetos se organizan según un **modelo jerárquico**: ciertos objetos contienen otros objetos que pueden, a su vez, contener otros. Estos objetos se llaman **contenedor** u **objetos parent**. Por ejemplo, el objeto **Application** es un contenedor de objetos **Workbook** (libros abiertos en Excel), que a su vez contiene objetos **Worksheet** (hojas de cálculo de un libro). El contenedor principal es el objeto **Application**.

Un conjunto de objetos del mismo tipo constituye una **colección** (colección Workbooks: conjunto de libros abiertos en Excel; colección Worksheets: conjunto de hojas de cálculo de un libro).

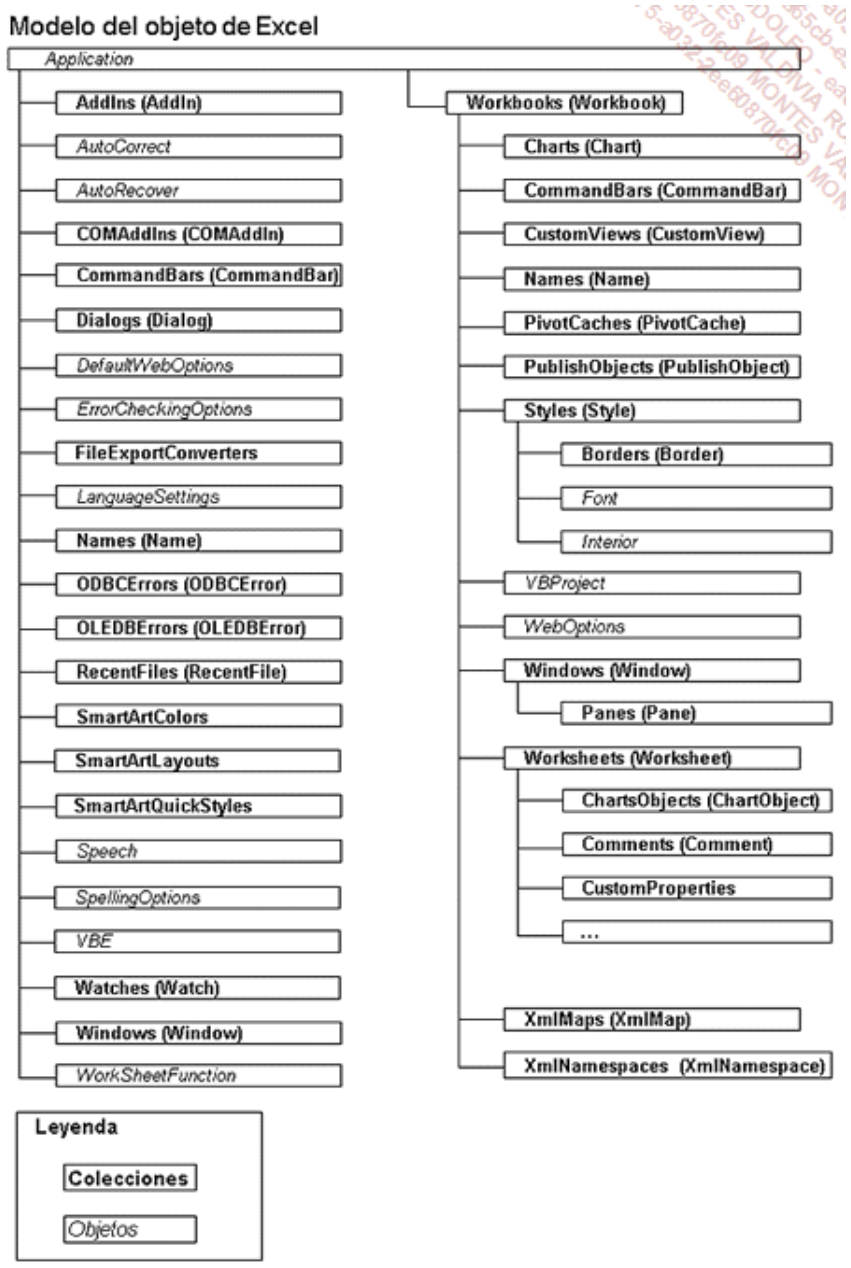
Un objeto dispone de un conjunto de características llamadas **propiedades** (por ejemplo, para el objeto **Application**: la propiedad **UserName** representa el nombre del usuario, la propiedad **Version** devuelve el número de versión de Microsoft Excel) y de comportamientos o acciones llamados **métodos** (por ejemplo, para el objeto **Application**, el método **FindFile** muestra el cuadro de diálogo **Abrir**; el método **Quit** sale de Excel, etc.).

A un objeto le suceden **eventos** provocados por el usuario (por ejemplo: la apertura de un libro, un clic en un botón de comando, el cambio de la celda activa, etc.) o por el sistema.

Las **clases** son modelos que permiten crear objetos de un mismo tipo. Los objetos de una misma clase heredan sistemáticamente todos los métodos, propiedades y eventos de su clase. Es posible crear clases de objetos con VBA Excel usando módulos de clase.

El modelo de objeto de Excel

1. Presentación



2. Principales objetos y colecciones

Objetos

Application

Objeto que hace referencia a la aplicación de Microsoft Excel activa.

AutoCorrect

Objeto que representa los atributos de autocorrección de Microsoft Excel.

AutoRecover

Objeto que representa las opciones de recuperación automática de un libro. Estas macros son accesibles desde Excel a partir de la pestaña **Guardar** del menú **Herramientas - Opciones**.

DefaultWebOptions

Objeto que devuelve los atributos usados por Excel para la apertura o grabación de una página web.

ErrorCheckingOptions

Objeto que representa las opciones de comprobación de errores de la aplicación de Excel.

LanguageSettings

Objeto que contiene información sobre la configuración de idioma de Excel.

Speech

Objeto que contiene los métodos y propiedades que se relacionan con las funciones de síntesis de voz.

SpellingOptions

Objeto que representa las opciones de ortografía de la aplicación.

VBE

Objeto que representa al Editor de Visual Basic.

WorkSheetFunction

Objeto que contiene todas las funciones disponibles en Excel. Este objeto permite obtener el resultado de una función aplicada a un rango de celdas. Ejemplo: Prom = Application.WorksheetFunction.Average (Selection)

Colecciones

AddIns

Colección que contiene todos los complementos (objetos AddIn).

COMAddIns

Representa los complementos COM actualmente instalados en Microsoft Excel.

CommandBars

Colección de barras de herramientas de la aplicación activa (objetos CommandBar).

Dialogs

Colección de los cuadros de diálogo integrados de Excel.

FileExportConverters

Colección de los convertidores de archivos para guardar libros de Excel.

Names

Colección de todos los nombres (celdas y rangos con nombre) del libro activo.

ODBCErrors

Colección de todos los errores ODBC generados por la última operación efectuada en un informe de tabla dinámica o en una tabla de consulta.

OLEDBErrors

Colección que representa la información relacionada con el error devuelto por la consulta OLE DB más reciente.

RecentFiles

Colección de los últimos archivos usados.

SmartArtColors

Colección de los estilos de colores SmartArt cargados actualmente en la aplicación.

SmartArtLayouts

Colección de los diagramas de diseño cargados actualmente en la aplicación.

SmartArtQuickStyles

Colección de los estilos rápidos de SmartArt cargados actualmente en la aplicación.

Watches

Colección de objetos que representa los rangos de inspección cuando la hoja de cálculo se recalcula.

Windows

Colección de todas las ventanas de la aplicación de Excel o de un libro.

Workbooks

Colección de los libros (objeto Workbook) abiertos.

Worksheets

Colección de las hojas de cálculo (objeto Worksheet) de un libro.

Principios de uso de los objetos y las colecciones

1. Propiedades

Las propiedades sirven para describir un objeto. Ciertas propiedades son de solo lectura y no se pueden, por lo tanto, modificar con el código VBA.

Sintaxis

```
{<objeto> | <variable objeto>}.<propiedad>
```

Ejemplo

```
'   Modificación del puntero del ratón
Application.Cursor = xlWait

'   Muestra la versión de la aplicación de Excel activa
'   Esta propiedad es de solo lectura
MsgBox Application.Version
Application.Cursor = xlDefault
```

2. Propiedades que representan objetos

Los objetos globales y los objetos definidos en el código a través de clases manejadas por VBA tienen ciertas propiedades cuyo valor se actualiza automáticamente por el sistema.

Estas **propiedades específicas** permiten acceder directamente a ciertos objetos: ventana activa, libro activo, celdas de la hoja activa, etc. La siguiente tabla muestra las propiedades específicas más usadas.

Propiedad	Objeto Parent	Objeto devuelto
ActiveCell	Application Window	Objeto **Range** que representa la primera celda activa de la ventana activa o especificada.
ActiveChart	Application Window Workbook	Objeto **Chart** que representa el gráfico activo.
ActiveControl	Frame Page UserForm	Objeto **Control** que representa el control (ActiveX) activo.
ActivePane	Window	Objeto **Pane** que representa el panel activo de la ventana activa.
ActiveSheet	Application Window Workbook	Objeto **Worksheet** que representa la hoja activa del libro activo o del libro especificado.
ActiveWindow	Application	Objeto **Window** que representa la ventana activa.
ActiveWorkBook	Application	Objeto **Workbook** que representa el libro de la ventana activa.
Parent	Objetos múltiples	Devuelve el objeto contenedor.
Selection	Application Windows	Objeto **Range** que representa la celda o las celdas seleccionadas.
ThisCell	Application	Devuelve la celda por la que la función definida

		por el usuario es llamada como objeto Range.
ThisWorkbook	Application	Objeto **Workbook** que representa el libro sobre el que se ejecuta el código de la macro actual.

Las propiedades específicas que devuelven un objeto Range (Cells, Offset, Columns, Rows, etc.) se explican en detalle en el capítulo Objetos de Excel.

3. Métodos

Los métodos permiten realizar acciones sobre los objetos.

Son similares a los procedimientos:

- Pueden usar o no argumentos.
- Ciertos métodos pueden devolver un valor, como los procedimientos **Function**, y otros no, como los procedimientos **Sub**.

Sintaxis de método que no devuelve un valor

```
{<objeto> | <variable objeto>}.<método> [<Lista de argumentos>]
```

Ejemplo

```
'   Activa la segunda hoja de cálculo del libro activo
Application.Goto ActiveWorkbook.Worksheets(2).Range("A1")
'   Selecciona un rango de celdas
Range("A1:C12").Select
'   Borra las celdas seleccionadas
Selection.Clear
'   Guarda el libro activo con un nuevo nombre
ActiveWorkbook.SaveAs "C:\presupuesto\presupuesto2.xlsm"
```

Como en el caso de los procedimientos, los distintos argumentos de un método se separan por comas. Si un argumento opcional no se define explícitamente, el método usará un valor por defecto.

Sintaxis de método que devuelve un valor

```
<variable> = {<objeto> | <variable objeto>}.<método>
[<Lista de argumentos>]
```

Ejemplo

```
'   Muestra el cuadro de diálogo Abrir
Dim strFileName as String

strFileName = Application.GetOpenFilename _
      (FileFilter:="Libros Excel (*.xlsm), *.xlsm", _
```

```
      Title:="Seleccione el archivo que desea abrir")
'   Si selecciona un archivo, lo abre
If strFileName <> False Then
   Workbooks.Open strFileName
End If
```

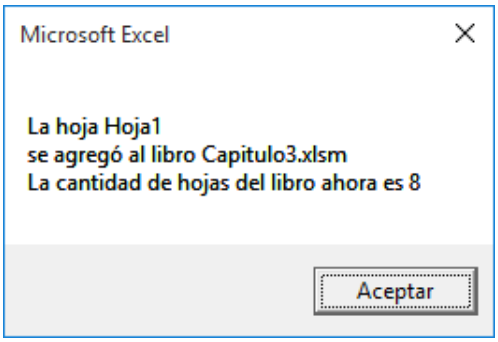
4. Eventos

Un evento es una **acción específica** que se realiza sobre un cierto objeto. Microsoft Excel responde a varios tipos de eventos: apertura o cierre de un libro, selección de celdas, agregar una hoja de cálculo, etc. Los eventos resultan generalmente de una acción del usuario.

El uso de un procedimiento asociado a eventos le permite definir su propio código como respuesta a un evento que se produce en un libro, una hoja o un formulario.

Ejemplo

Cuando se agrega una nueva hoja de cálculo al libro, aparece un mensaje para el usuario.




```
Private Sub Workbook_NewSheet(ByVal Sh As Object)

MsgBox "La hoja " & Sh.Name & Chr(13) & _
       "se agregó al libro " & _
       ActiveWorkbook.Name & Chr(13) & _
       "La cantidad de hojas del libro ahora es " & _
       ActiveWorkbook.Worksheets.Count

End Sub
```

La administración de eventos es uno de los aspectos más importantes en el desarrollo de aplicaciones de Excel; el capítulo Administración de eventos está íntegramente dedicado a este tema.

5. Colecciones

Para hacer referencia a un objeto de una colección, puede usarse cualquiera de las siguientes sintaxis:

```
NomColección!NomObjeto
NomColección![NomObjeto]
NomColección("NomObjeto")
```

```
NomColección(var)
```

donde var representa una variable de tipo String que contiene el nombre del objeto.

```
NomColección(index)
```

donde index representa el número del índice del objeto en la colección.

Para asegurar una mejor legibilidad del código, se aconseja usar siempre la misma sintaxis. Las sintaxis tercera y quinta se recomiendan porque permiten activar el asistente del editor de código. Además, la quinta sintaxis es más útil para recorrer los objetos de una colección.

Atención: el primer elemento de la mayoría de las colecciones lleva el índice 1. Use los índices solamente para recorrer una colección. Evite, por ejemplo, usar ActiveWorkbook.ActiveSheet(3) para hacer referencia a una hoja de cálculo del libro activo, ya que el índice de la hoja puede cambiar (si mueve las hojas o si elimina alguna).

Ejemplo

El siguiente código activa la hoja de cálculo Hoja1 del libro Presupuesto.xlsx. El código usa las colecciones Workbooks y Worksheets.

```
Workbooks("Presupuesto.xlsm").Worksheets("Hoja1").Activate
```

o

```
Workbooks![Presupuesto.xlsm].Worksheets!Hoja1.Activate
```

o

```
Workbooks![Presupuesto.xlsm].Worksheets![Hoja1].Activate
```

Recorrer una colección: este código cambia el nombre de las hojas de cálculo del libro activo.

```
Dim i As Integer
For i = 1 To ActiveWorkbook.Worksheets.Count
      ActiveWorkbook.Worksheets(i).Name = "Presupuesto N° " & i
Next i
```

Una colección también se puede recorrer con la instrucción For Each Next.

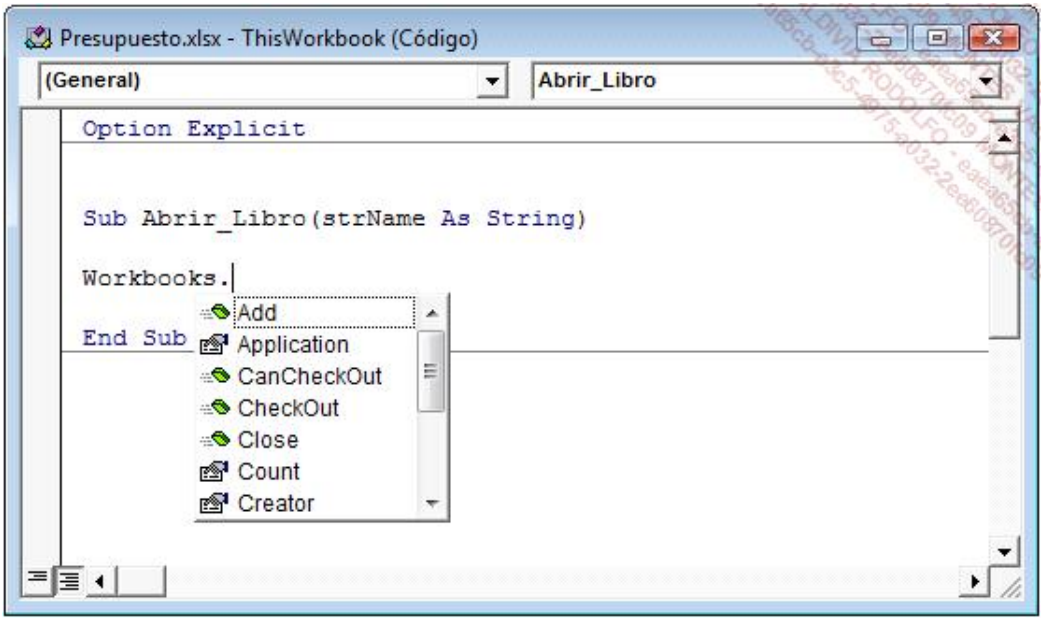
```
Dim oHoja As Worksheet
For Each oHoja In ActiveWorkbook.Worksheets
      oHoja.Name = "Presupuesto N° " & oHoja.Index
Next oHoja
```

6. Redacción automática de instrucciones

El editor VBA incluye una tecnología que lo ayuda en el empleo de objetos. Cuando usted escribe el nombre de un objeto o de una colección reconocida por VBA seguido de un punto, se despliega una lista con los métodos y las propiedades del objeto. Si selecciona un método, el asistente lo ayuda a indicar los argumentos que le corresponden.

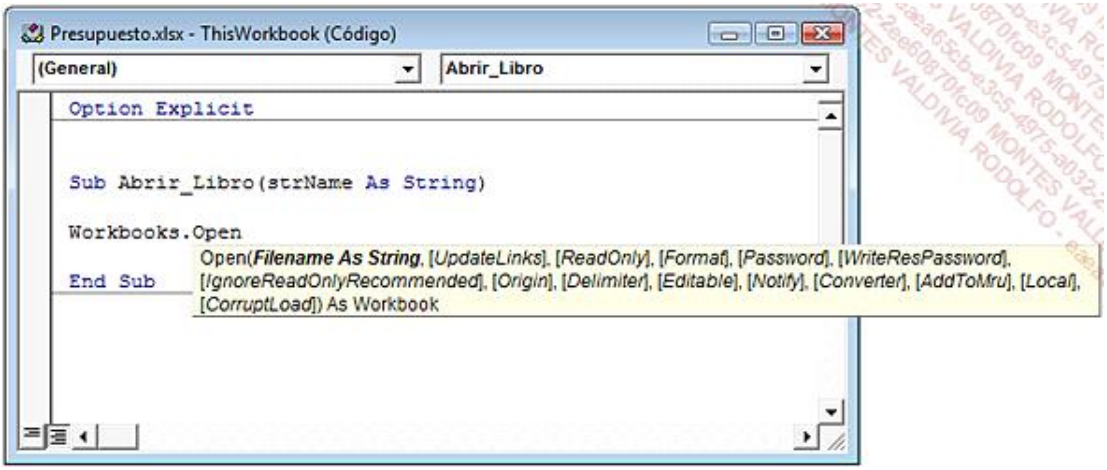
Ejemplo

- Escriba el nombre de la colección Workbooks seguido de un punto; aparece la siguiente lista desplegable.



El icono representa los métodos, el icono representa las propiedades y las colecciones.

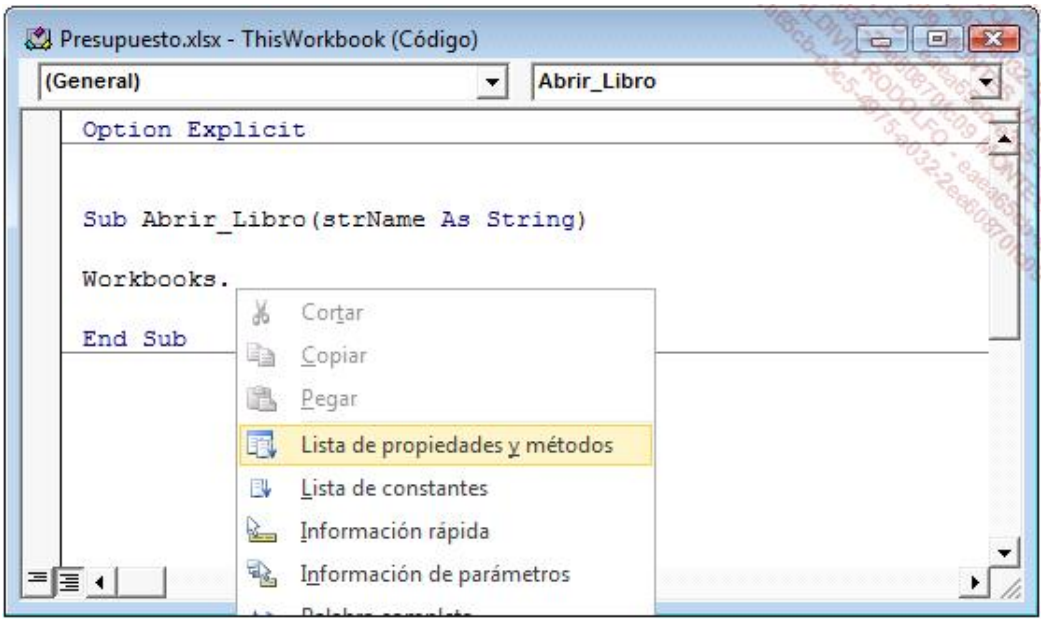
- Puede hacer que la lista avance escribiendo las primeras letras del método, propiedad o colección buscada o usando la barra de desplazamiento. Para seleccionar un elemento de la lista, haga doble clic en él.
- Escriba un punto si acaba de seleccionar un objeto para ver la lista de sus propiedades y métodos. Si selecciona un método, escriba un espacio para obtener la lista de configuración del método.
- Para seguir este ejemplo, seleccione el método **Open** y luego escriba un espacio.



Aparecerá una lista con los argumentos del método. Los argumentos opcionales aparecen entre corchetes. El argumento actual se verá en negrita. Si para un argumento dado existe una lista de valores predefinidos, aparecerá una lista desplegable con las constantes correspondientes.

También puede activar la lista de propiedades y métodos de la siguiente manera:

- Ubique el cursor detrás del punto situado después del método.
- Haga clic en el botón derecho del ratón para hacer aparecer el menú contextual.
- Seleccione la opción **Lista de propiedades y métodos**.



Instrucciones usadas con los objetos

1. La instrucción With

La instrucción **With** permite acceder varias veces al mismo objeto, indicándolo una sola vez.

Ofrece varias ventajas:

- Optimización del tiempo de ejecución del código.
- Ganar tiempo en la escritura del código.
- Mejor legibilidad del código.

Sintaxis

```
With <Objeto>
    <código que usa los métodos y propiedades>
    <que se relacionan con el objeto>
End With
```

Ejemplo

Agregar y modificar una hoja de cálculo.

```
With ActiveWorkbook
   '   Crea una hoja después de la última hoja del libro activo
   .Worksheets.Add , .Worksheets(.Worksheets.Count)
   '   Modifica el nombre de la nueva hoja y el valor de la celda A1
   With .ActiveSheet
      .Name = "Síntesis"
      .Range("A1") = "Revisión del Presupuesto"
   End With
End With
```

2. La instrucción For Each...Next

La instrucción **For Each... Next** permite recorrer los objetos de una colección o de una tabla.

Sintaxis

```
For Each <elemento > In <grupo>
    <secuencia de instrucciones>
    [Exit For]
    <secuencia de instrucciones>
Next <elemento>
```

Ejemplo

Modificación del contenido de la celda A1 y de la cantidad de hojas del libro activo.

```
Sub NumPresupuesto()
   Dim Hoja As Worksheet

   For Each Hoja In ActiveWorkbook.Worksheets
      Hoja.Cells(1, 1) = "PRESUPUESTO N° " & Hoja.Index
      Hoja.Name = "Presupuesto " & Hoja.Index
   Next Hoja
End Sub
```

3. La instrucción If TypeOf

La instrucción **If TypeOf** permite comprobar el tipo de un objeto.

Sintaxis

```
If TypeOf <Objeto> Is <TipoObjeto> Then
    <código que usa los métodos y propiedades>
    <relacionados con el objeto>
End If
```

Ejemplo

```
If TypeOf obj.Parent Is Worksheet Then_...
```

4. La instrucción Set

La instrucción **Set** permite atribuir la referencia de un objeto a una variable, llamada **variable objeto**.

Esta instrucción puede usarse para **crear un nuevo objeto** (usando en ese caso un método que permita crear el objeto) o para **hacer referencia a un objeto ya existente**.

Sintaxis

```
Set <Objeto> = [New] <expresión objeto>
O Set <Objeto> = Nothing
```

objeto	Es una variable de tipo String, que contiene el nombre del objeto que es necesario crear.
La palabra clave New	Permite crear una nueva instancia de la clase. Si la variable <Objeto> contiene una referencia a un objeto, esta última se abandona.
<expresión objeto>	Puede ser el nombre de un objeto o de una variable objeto del mismo tipo. Es decir, una función o método que devuelva un objeto del mismo tipo.
Nothing	Reinicializa la variable objeto y libera el conjunto de recursos del sistema y la memoria asociadas al objeto.

Ejemplo

Crear un libro con dos hojas y asignarle un nombre a cada una de ellas.

```
Sub Crear_Libro()
Dim oLibro As Workbook
Dim i As Integer
  ’   Crear un nuevo libro
Set oLibro = Application.Workbooks.Add
 ’   Eliminar las hojas de la tercera hasta la última
With oLibro
   Application.DisplayAlerts = False
   For i = .Worksheets.Count To 3 Step -1
       .Worksheets(i).Delete
   Next i
   Application.DisplayAlerts = True
   ’   Asignar los nombres a las hojas 1 y 2
   .Worksheets(1).Name = "Ventas Año 2013"
   .Worksheets(2).Name = "Ventas Año 2014"
   .SaveAs ThisWorkbook.Path & "\Histórico"
End With

End Sub
```

Modificar una hoja en un libro abierto.

```
Dim oLibro As Workbook
Dim oHoja As Worksheet

Set oLibro = Application.Workbooks![Histórico.xlsx]
Set oHoja = oLibro.Worksheets![Ventas Año 2014]
With oHoja
     .Name = "Ventas 2014"
     .Range("A2") = "Ventas del año 2014"
End With
Set oHoja = Nothing
```

El Examinador de objetos

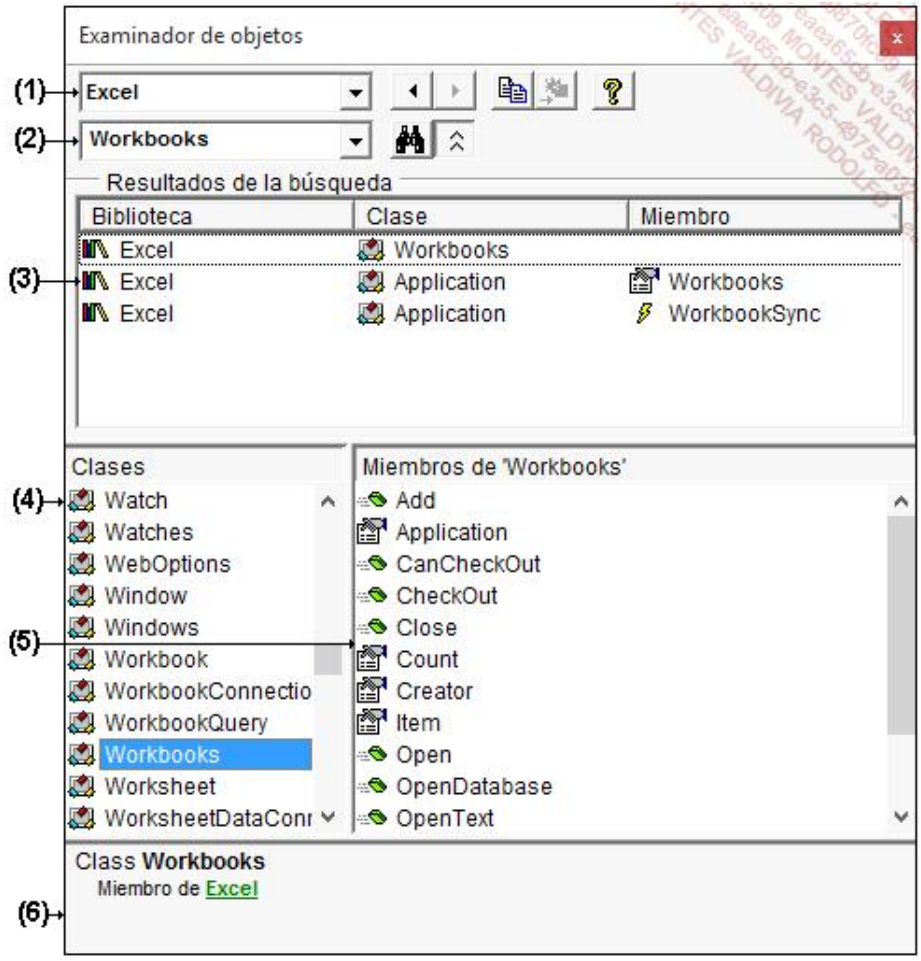
1. Presentación

Dada la cantidad y diversidad de objetos de Excel, es útil poder encontrar la información que se relaciona con los distintos objetos.

El Examinador de objetos muestra la información relativa a los objetos, métodos, propiedades, eventos y constantes.

Puede acceder al Examinador de objetos de distintas maneras:

- **Ver - Examinador de objetos** o  (barra de herramientas **Estándar**) o [F2].



1. Lista de bibliotecas actualmente cargada.

2. Texto buscado: objeto, propiedad, colección, evento, método, etc.

3. Resultados de la búsqueda: lista de las clases de objetos (objetos y colecciones) y de sus miembros (objeto, colección, propiedad, evento o método). La palabra buscada puede estar en la lista de clases o en la de sus miembros.

4. Clases de objetos de la biblioteca: la clase de objeto seleccionada en la lista **Resultados de la búsqueda** aparece remarcada.

5. Propiedades (icono), **métodos** (icono), **eventos** (icono) y **constantes** (icono) se relacionan con la clase de objeto seleccionada o remarcada en la lista de la izquierda.

6. Detalle del elemento seleccionado.

2. Búsqueda en el Examinador de objetos

Para hacer una búsqueda en el Examinador de objetos, proceda de la siguiente manera:

- Indique la palabra buscada en la segunda lista desplegable.
- Haga clic en el icono  . Si la lista **Resultados de la búsqueda** muestra muchos renglones, desplácese hasta el que le interese; la parte inferior de la ventana se actualizará.

El objeto Application

El objeto **Application** representa la aplicación de Microsoft Excel activa. Es el objeto por defecto y, por lo tanto, normalmente es opcional (ejemplo: *Version* equivale a *Application.Version*).

Este objeto contiene:

- Las **propiedades relativas al entorno de Excel** (por ejemplo, las opciones de Excel) y a la **presentación de la interfaz** (puntero del ratón, texto de la barra de estado, tamaño y estado de la ventana de la aplicación, etc.).
- Hay distintos **métodos** para realizar acciones en el entorno de Excel.
- **Propiedades** que devuelven objetos y colecciones de primer nivel (objetos y colecciones del modelo de objetos de Excel, como Workbooks o Charts).
- **Propiedades específicas** que hacen referencia directa a objetos (como ActiveCell, ActiveSheet o ActiveWindow).

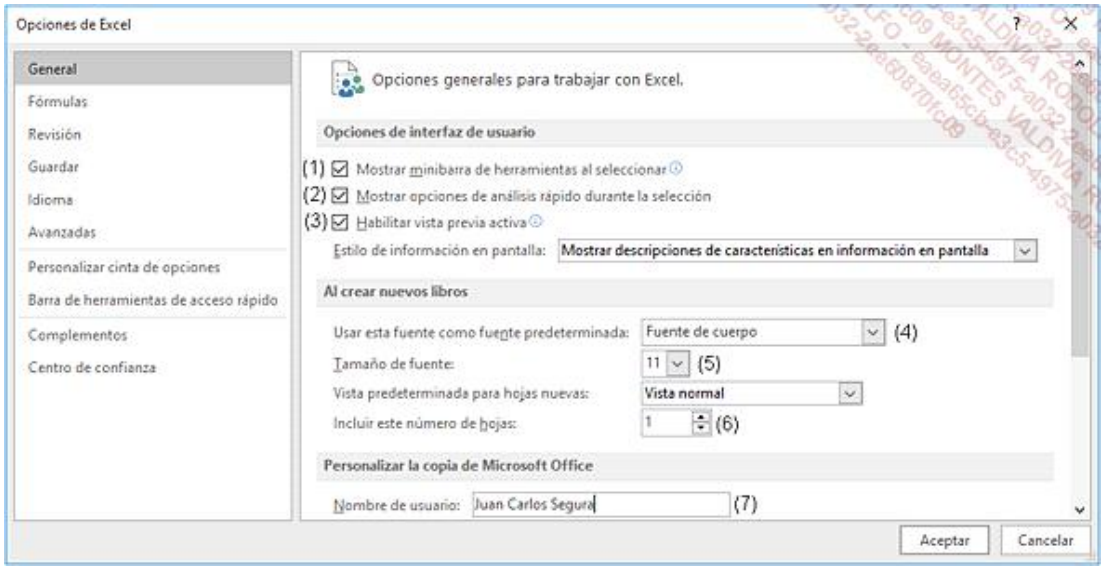
Las propiedades que hacen referencia a objetos se explican en el capítulo La programación de objetos en Excel.

1. Propiedades que representan las opciones de Excel

Las principales opciones de Excel se pueden definir o devolver a partir de propiedades del objeto **Application**. La mayoría de estas propiedades son de lectura y escritura.

Para acceder a las opciones de Excel en la versión 2013, haga clic en la pestaña **Archivo** y luego en **Opciones**.

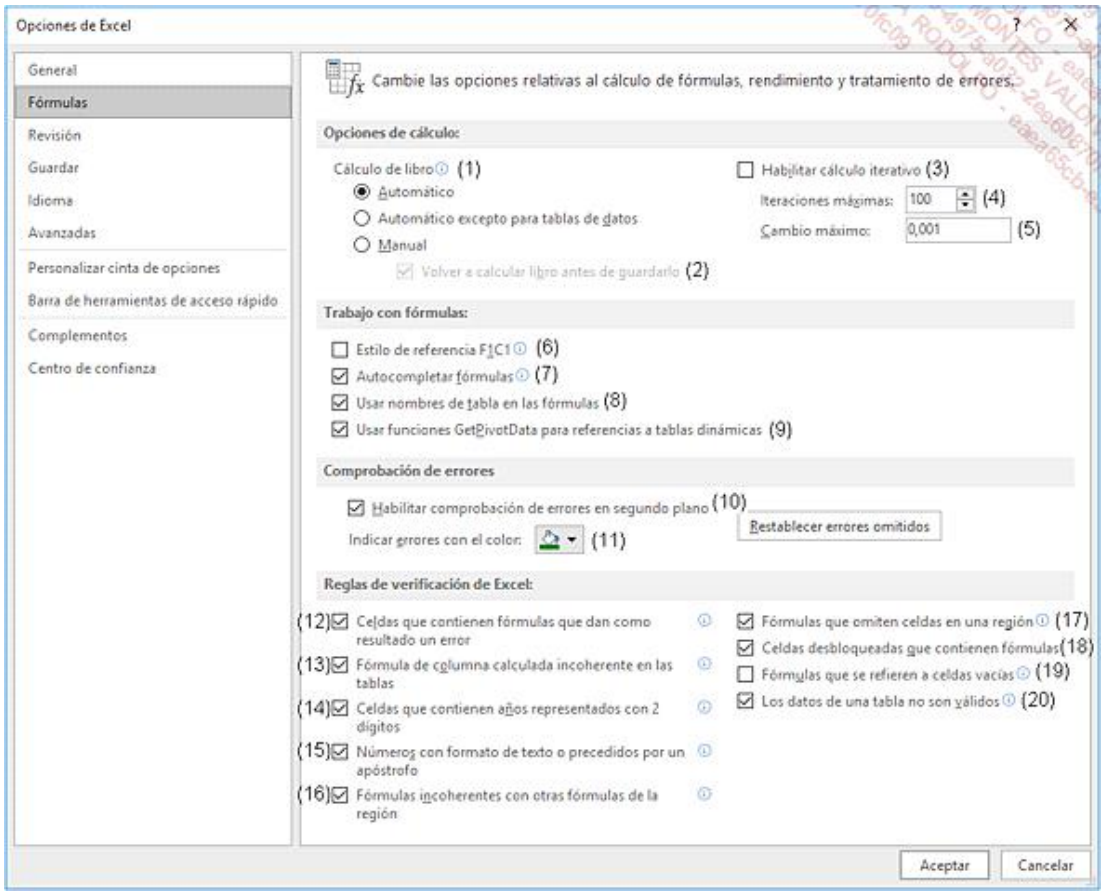
a. Opciones de la categoría General



N.°	Propiedades	Valores devueltos
1	ShowSelectionFloaties	Booleano
2	ShowQuickAnalysis	Booleano
3	EnableLivePreview	Booleano
4	StandardFont	Entero largo

5	StandardFontSize	Entero largo
6	SheetsInNewWorkbook	Entero largo
7	UserName	Cadena de caracteres

b. Opciones de la categoría Fórmulas



Opciones relacionadas con el modo de recálculo

N.°	Propiedades	Valores devueltos
1	Calculation	Constantes: **xlCalculationAutomatic** **xlCalculationManual** **xlCalculationSemiautomatic**
2	CalculateBeforeSave	Booleano
3	Iteration	Booleano
4	MaxIterations	Entero largo
5	MaxChange	Doble

Opciones relativas a las fórmulas

N.°	Propiedades	Valores devueltos

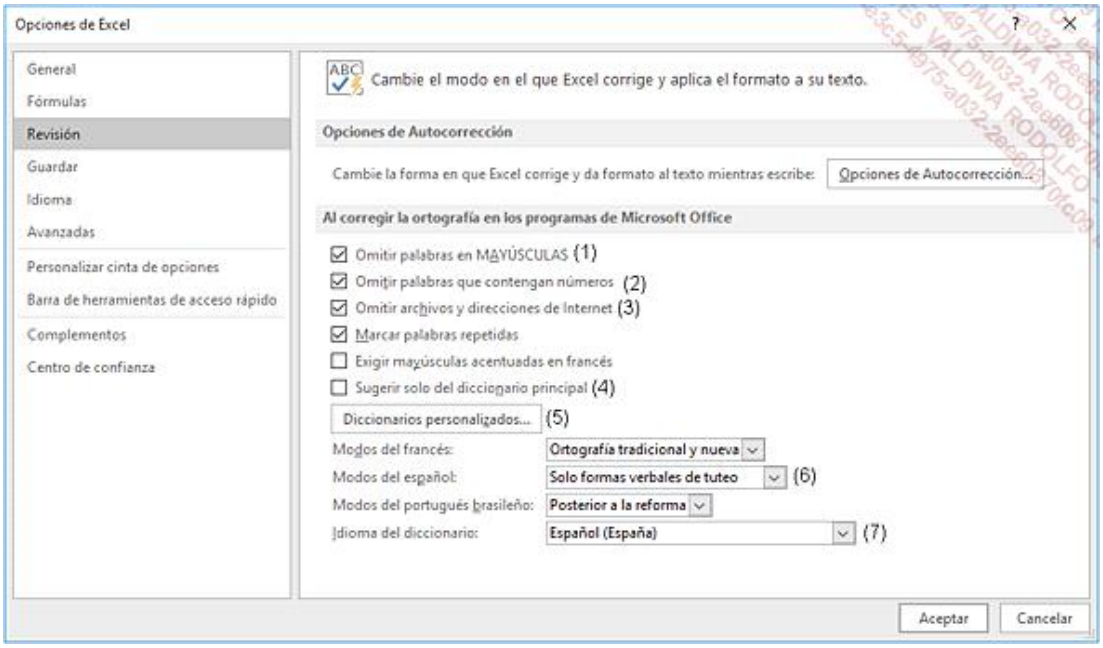
6	ReferenceStyle	Constantes: **xlA1** **xlR1C1**
7	DisplayFormulaAutoComplete	Booleano
8	GenerateTableRefs	Constantes: **xlGenerateTableRefStruct** **xlGenerateTableRefA1**
9	GenerateGetPivotData	Booleano

Opciones de comprobación de errores

Las siguientes propiedades dependen de la propiedad **ErrorCheckingOptions** del objeto **Application**. Esta propiedad devuelve un objeto **ErrorCheckingOptions**, que representa las opciones de comprobación de errores para una aplicación.

N.°	Propiedades	Valores devueltos
10	BackgroundChecking	Booleano
11	IndicatorColorIndex	Constante **xlColorIndex**
12	EvaluateToError	Booleano
13	InconsistentTableFormula	Booleano
14	TextDate	Booleano
15	NumberAsText	Booleano
16	InconsistentFormula	Booleano
17	GenerateTableRefs	Booleano
18	UnlockedFormulaCells	Booleano
19	EmptyCellReferences	Booleano
20	ListDataValidation	Booleano

c. Opciones de la categoría Revisión

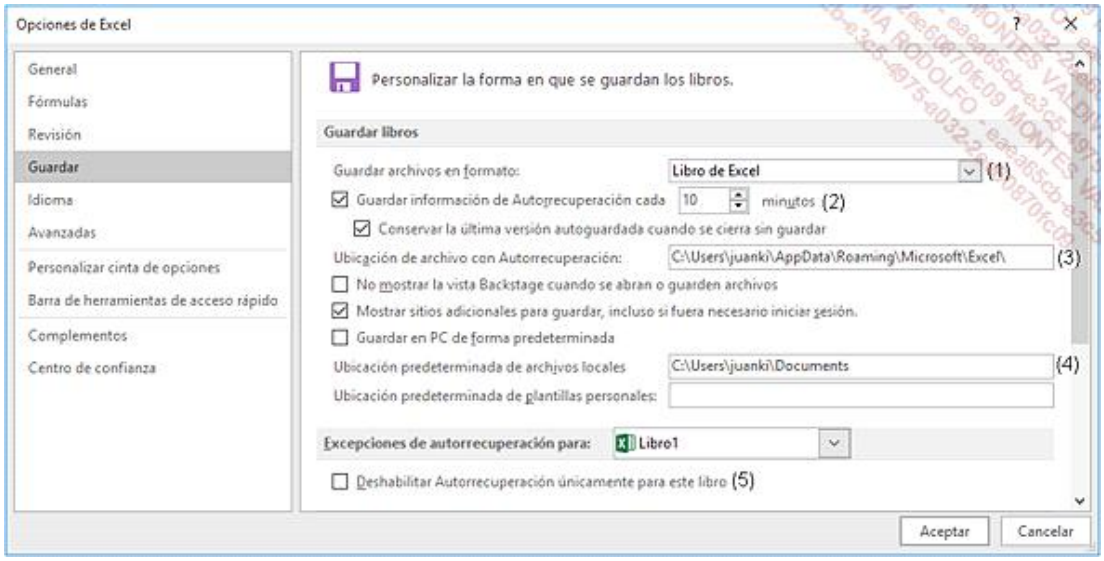


Opciones de revisión ortográfica

Las siguientes propiedades dependen de la propiedad **SpellingOptions** del objeto **Application**. Esta propiedad devuelve un objeto **SpellingOptions**, que representa las opciones de revisión ortográfica para una aplicación.

N.°	Propiedades	Valores devueltos
1	IgnoreCaps	Booleano
2	IgnoreMixedDigits	Booleano
3	IgnoreFileNames	Booleano
4	SuggestMainOnly	Booleano
5	UserDict	String
6	SpanishModes	Constante **xlSpanishModes**
7	DictLang	Entero largo

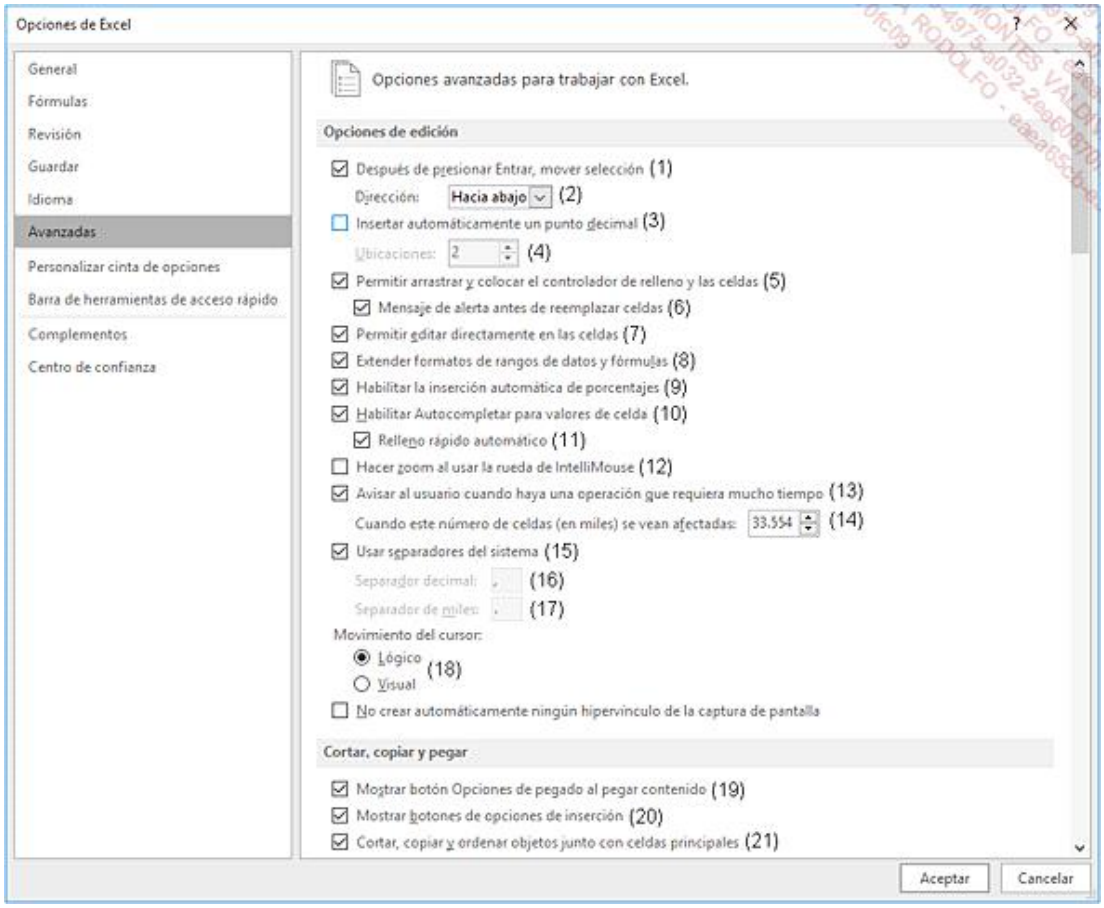
d. Opciones de la categoría Guardar



N.°	Propiedades	Valores devueltos
1	DefaultSaveFormat	Constante **xlFileFormat**
2	AutoRecover.Time	Entero largo
3	AutoRecover.Path	Cadena de caracteres
4	DefaultFilePath	Cadena de caracteres
5	EnableAutoRecover	Booleano

La opción 5 se aplica a libros (por ejemplo, Application.ActiveWorkbook).

e. Opciones de la categoría Avanzadas

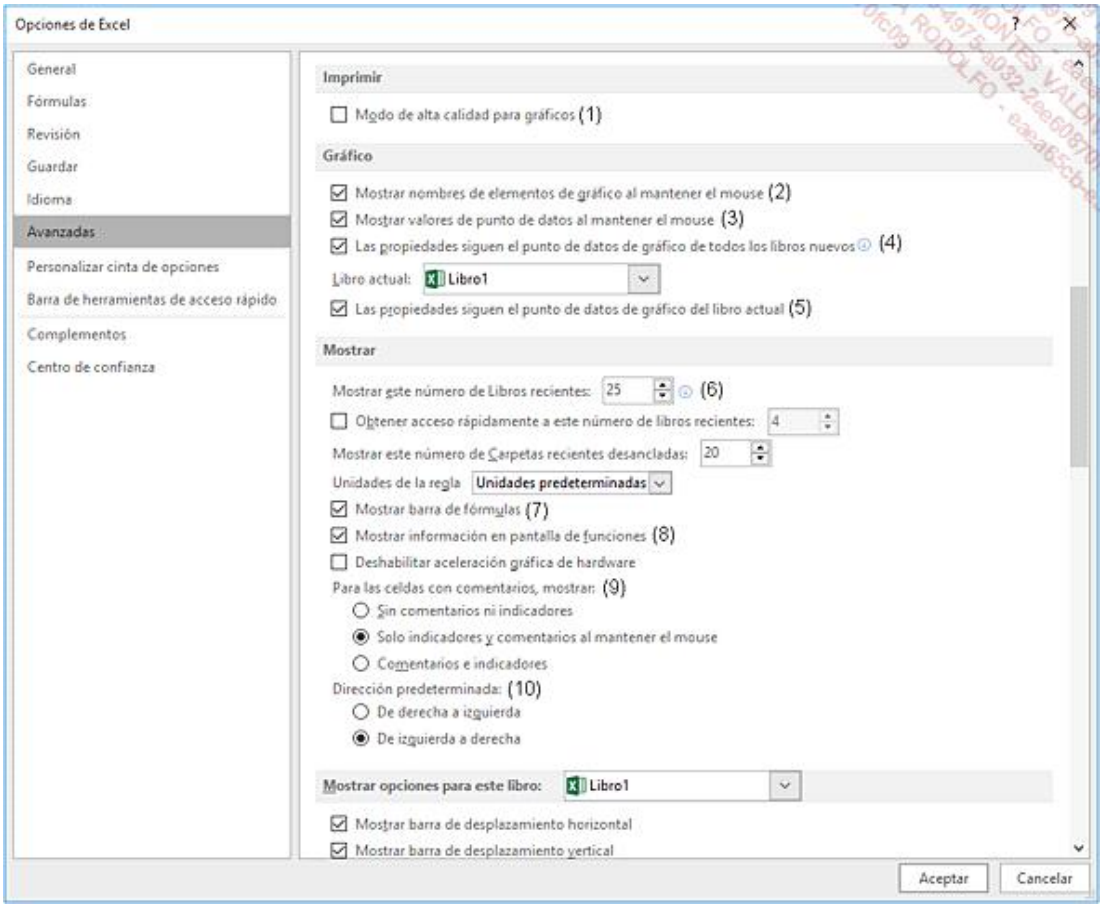


N.°	Propiedades	Valores devueltos
1	MoveAfterReturn	Booleano
2	MoveAfterReturnDirection	Constantes: **xlDown** **xlUp** **xlToRight** **xlToLeft**
3	FixedDecimal	Booleano
4	FixedDecimalPlaces	Entero largo
5	CellDragAndDrop	Booleano
6	AlertBeforeOverwriting	Booleano
7	EditDirectlyInCell	Booleano
8	ExtendList	Booleano
9	AutoPercentEntry	Booleano
10	EnableAutoComplete	Booleano
11	FlashFill	Booleano
12	RollZoom	Booleano
13	EnableLargeOperationAlert	Booleano
14	LargeOperationCellThousandCount	Entero largo
15	UseSystemSeparators	Booleano
16	DecimalSeparator	Cadena de caracteres
17	ThousandsSeparator	Cadena de caracteres

18	CursorMovement	Constantes: **xlVisualCursor** **xlLocgicalCursor**

Opciones de Cortar, Copiar y Pegar

N.°	Propiedades	Valores devueltos
19	DisplayPasteOptions	Booleano
20	DisplayInsertOptions	Booleano
21	CopyObjectsWithCells	Booleano

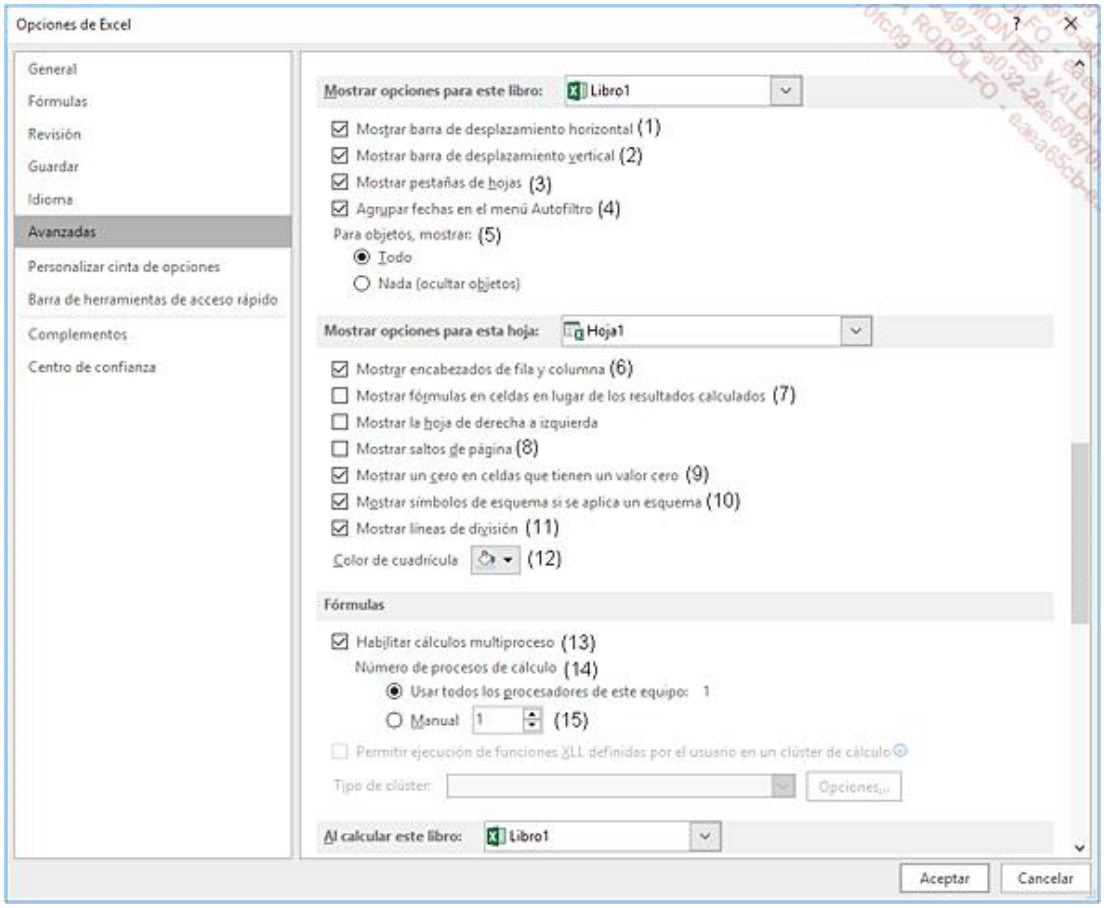


Opciones para gráficos

N.°	Propiedades	Valores devueltos
1	HighQualityModeForGraphics	Booleano
2	ShowChartTipNames	Booleano
3	ShowChart TipValues	Booleano
4	ChartDataPointTrack (object Application)	Booleano
5	ChartDataPointTrack (object Workbook)	Booleano

Opciones de visualización

N.°	Propiedades	Valores devueltos
6	RecentFiles.Maximum	Entero largo
7	DisplayFormulaBar	Booleano
8	DisplayFunctionToolTips	Booleano
9	DisplayCommentIndicator	Constantes: **xlNoIndicator** **xlIndicatorOnly** **xlCommentAndIndicator**
10	DefaultSheetDirection	Constantes: **xlRTL** (de derecha a izquierda) **xlLTR** (de izquierda a derecha)



Opciones de visualización para libros

Las siguientes propiedades dependen de la propiedad **ActiveWindow** del objeto **Application**. Esta propiedad devuelve un objeto **Window** que representa la ventana activa.

N.°	Propiedades	Valores devueltos
1	DisplayHorizontalScrollBar	Booleano
2	DisplayVerticalScrollBar	Booleano
3	DisplayWorkbookTabs	Booleano

4	AutoFilterDateGrouping	Booleano
5	DisplayDrawingObjects	Constantes: **xlDisplayShapes** (todos) **xlHide** (ninguno) Esta opción se aplica a libros (objeto **Workbook**).

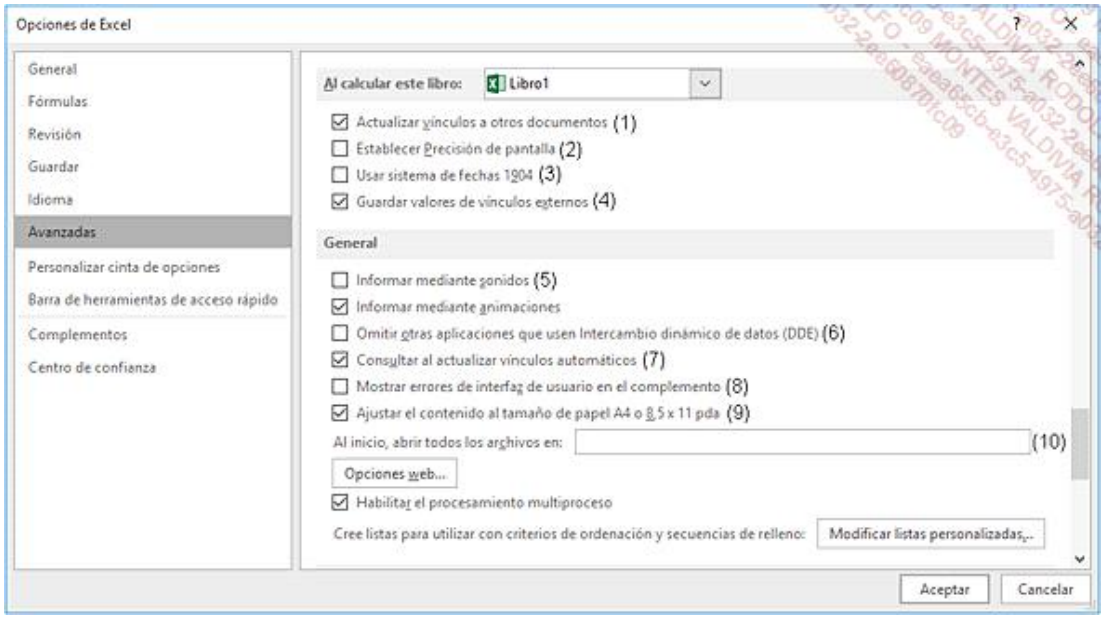
Opciones de visualización para las hojas de cálculo

Las siguientes propiedades dependen de la propiedad **ActiveWindow** del objeto **Application**.

N.°	Propiedades	Valores devueltos
6	DisplayHeadings	Booleano
7	DisplayFormulas	Booleano
8	DisplayPageBreaks	Booleano
9	DisplayZeros	Booleano
10	DisplayOutline	Booleano
11	DisplayGridlines	Booleano
12	GridlineColorIndex	Constante **xlColorIndex**

Opciones relacionadas con las fórmulas

N.°	Propiedades	Valores devueltos
13	MultiThreadedCalculation.Enabled	Booleano
14	MultiThreadedCalculation.ThreadMode	Constantes: **xlThreadModeAutomatic** **xlThreadModeManual**
15	MultiThreadedCalculation.ThreadCount	Entero



Opciones relacionadas con el cálculo en los libros

Las siguientes propiedades se aplican a un libro (objeto Workbook).

N.°	Propiedades	Valores devueltos
1	SaveLinkValues	Booleano
2	PrecisionAsDisplayed	Booleano
3	Date1904	Booleano
4	UpdateRemoteReferences	Booleano

Opciones generales

N.°	Propiedades	Valores devueltos
5	EnableSound	Booleano
6	IgnoreRemoteRequests	Booleano
7	AskToUpdateLinks	Booleano
8	WarnOnFunctionNameConflict	Booleano
9	MapPaperSize	Booleano
10	AltStartupPath	Cadena de caracteres
11	Asigna las propiedades de los objetos WebOptions y DefaultWebOptions	

2. Propiedades relativas a la presentación de la aplicación

DisplayAlerts

Booleano. Muestra (True) u oculta los mensajes, sobre todo de alerta, cuando se ejecuta una macro.

Height

Real doble. Altura de la ventana.

Left

Real doble. Distancia entre el borde izquierdo de la pantalla y el borde izquierdo de la ventana principal de Microsoft Excel.

Top

Real doble. Distancia entre el borde superior de la pantalla y el borde superior de la ventana principal de Microsoft Excel.

Width

Real doble. Distancia entre los bordes izquierdo y derecho de la ventana de la aplicación.

Caption

Cadena de caracteres. Nombre que se muestra en la barra de títulos de la ventana de Microsoft Excel.

Cursor

Constante. Aspecto del puntero del ratón en Excel.

Constantes	**xlDefault**	Puntero por defecto.
	xlIBeam	Puntero en I.
	xlNorthwestArrow	Flecha Noroeste.
	xlWait	Reloj de arena.

DisplayFullScreen

Booleano. Indica si Excel funciona en modo de pantalla completa.

FormulaBarHeight

Entero largo. Altura, en número de líneas, de la barra de fórmulas.

ScreenUpdating

Booleano. Permite desactivar (False) la actualización de la pantalla durante la ejecución del código VBA.

ShowMenuFloaties

Booleano. Indica si las minibarras de herramientas deben aparecer cuando el usuario hace clic en el botón derecho del ratón sobre la ventana del libro.

StatusBar

Cadena de caracteres. Texto de la barra de estado.

Visible

Booleano. Indica si la ventana principal de la aplicación está visible.

Ejemplo

El siguiente código permite optimizar el tiempo de ejecución de una macro (ej.: macro que modifica el contenido de un gran número de celdas de Excel o que realiza cálculos en las celdas).

```
Sub Macro_Optimizada()

'   Desactiva la actualización de la pantalla
Application.ScreenUpdating = False
'   Oculta los mensajes de alerta
Application.DisplayAlerts = False
'   Desactiva el cálculo manual
Application.Calculation = xlCalculationManual
```

```
'   Escribir aquí el código de la macro
'...
'...

'   Actualización de la pantalla
Application.ScreenUpdating = True
'   Muestra mensajes de alerta
Application.DisplayAlerts = True
'   Activa el cálculo automático
Application.Calculation = xlCalculationAutomatic

End Sub
```

3. Propiedades varias

AutoFormatAsYouTypeReplaceHyperlinks

Booleano. Indica si Excel convierte en hipervínculos las rutas de Internet de forma automática a medida que se las escribe.

AutomationSecurity

Constante. Modo de seguridad que emplea Microsoft Excel al abrir archivos con macros.

Constantes: **msoAutomationSecurityByUI**
msoAutomationSecurityForceDisable
msoAutomationSecurityLow

ActivePrinter

Cadena de caracteres. Nombre de la impresora activa.

CalculationInterruptKey

Constante. Indica la tecla que puede interrumpir el recálculo en Microsoft Excel.

Constantes: **xlAnyKey**
ptxlEscKey
xlNoKey

CalculationState

Constante. Indica el estado de cálculo de la aplicación, para el recálculo en curso en Microsoft Excel.

Constantes: **xlCalculating**
xlDone
xlPending

ClipboardFormats

Variant. Devuelve una matriz que contiene los formatos que se encuentran actualmente en el Portapapeles.

CutCopyMode

Constante. Devuelve o define el estado del modo Cortar o Copiar.

Constantes:	**False**:	No está en modo Cortar ni Copiar.
	xlCopy:	Está en modo Copiar.
	xlCut:	Está en modo Pegar.

DataEntryMode

Constante. Devuelve o define el modo de entrada de datos.

Constantes:	**xlOn**:	Modo de entrada de datos activado.
	xlOff:	Modo de entrada de datos desactivado.
	xlStrict:	Modo de entrada de datos activado y tecla [Escape] desactivada.

EnableCancelKey

Constante. Controla la forma en que Microsoft Excel trata la manera en que el usuario interrumpe el procedimiento en curso pulsando la combinación de teclas [Ctrl][Pausa].

Constantes	**xlDisabled**:	No se interrumpe.
	xlInterrupt:	Interrumpe el procedimiento en curso y pasa al modo Depurar.
	xlErrorHandler:	En caso de interrupción, el procedimiento genera un error (código de error 18).

EnableEvents

Booleano. Permite desactivar (False) los eventos del objeto Application.

EnableMacroAnimations

Booleano. Indica si las animaciones de macro están activadas.

FileValidation

Constante. Devuelve o define la forma en que Excel valida los archivos antes de abrirlos.

Constantes:	**msoFileValidationDefault**
	msoFileValidationSkip

FileValidationPivot

Constante. Devuelve o define la forma en que Excel valida el contenido de la memoria caché en un informe de tabla dinámica.

Constantes:	**xlFileValidationPivotDefault**:	Valida el contenido de

	la memoria caché según la configuración predeterminada y del Registro.
xlFileValidationPivotRun:	Valida el contenido de todas las memorias cachés.
xlFileValidationPivotSkip:	No valida el contenido de las memoria caché

FindFormat

Devuelve o define los criterios de búsqueda para el tipo de formato de celda que hay que encontrar.

GenerateGetPivotData

Booleano. Indica si Excel puede obtener datos de un informe de tabla dinámica.

Hinstance

Entero largo. Devuelve el controlador de instancia de la instancia que llama a Microsoft Excel.

HinstancePtr

Variant. Devuelve un controlador a la instancia de Microsoft Excel 2016 representada por el objeto Application.

Hwnd

Entero largo. Devuelve un objeto que designa el identificador de ventana superior de la ventana de Microsoft Excel.

MapPaperSize

Booleano. Indica si los documentos se ajustan automáticamente cuando cambia el tamaño de papel.

MouseAvailable

Booleano. Indica si hay un ratón disponible.

OrganizationName

Cadena de caracteres. Nombre de la empresa.

PreviousSelections

Variant. Devuelve una matriz de objetos Range que contiene los últimos cuatro rangos seleccionados.

PrintCommunication

Booleano. Indica si está activa la comunicación con la impresora.

RecordRelative

Booleano. True si las macros se graban usando referencias relativas.

ReplaceFormat

Booleano. Establece los criterios de sustitución que se emplean para sustituir formatos de celda (se usa con la propiedad **FindFormat**).

SaveISO8601Dates

Booleano. Indica si Excel guarda los valores de fechas y horas en formato ISO 8601.

TemplatesPath

Cadena de caracteres. Ruta de acceso local de la carpeta donde se guardan las plantillas.

Version

Cadena de caracteres. Número de versión de la aplicación de Excel activa.

4. Métodos del objeto Application

a. Métodos que actúan sobre las fórmulas y los cálculos

Calculate

Fuerza un cálculo de los datos para todos los libros abiertos.

CalculateFull

Fuerza un recálculo completo de los datos en todos los libros abiertos.

CalculateFullRebuild

Para todos los libros abiertos, fuerza un recálculo completo de los datos y vuelve a establecer las dependencias.

CheckAbort

Detiene el recálculo.

ConvertFormula

Convierte las referencias de celda en una fórmula, pasando del estilo de referencia A1 al estilo R1C1.

Evaluate

Calcula la expresión pasada como argumento y devuelve el resultado. La expresión debe corresponder a una fórmula en inglés.

b. Métodos que actúan sobre las celdas

DoubleClick

Equivale a hacer doble clic sobre la celda activa.

GoTo

Selecciona un rango o un procedimiento Visual Basic en cualquier libro y activa ese mismo libro si no lo está.

Intersect

Devuelve un objeto **Range** que representa la intersección de dos o más rangos.

Union

Devuelve la unión de, al menos, dos rangos.

c. Métodos que actúan sobre las listas personalizadas

AddCustomList

Agrega una lista personalizada.

DeleteCustomList

Elimina una lista personalizada.

GetCustomListContents

Devuelve una lista personalizada (matriz de cadena de caracteres).

GetCustomListNum

Devuelve el número de la lista personalizada correspondiente a una matriz de cadena de caracteres.

d. Métodos que muestran los cuadros de diálogo

Los métodos **GetOpenFileName**, **GetSaveAsFileName** e **InputBox** se explican en el capítulo Cuadros de diálogo.

e. Métodos relacionados con las acciones en Excel

ExecuteExcel4Macro

Ejecuta una función macro Microsoft Excel 4.0 y devuelve su resultado.

OnKey

Ejecuta un procedimiento especificado cuando el usuario pulsa una tecla o una combinación de teclas.

OnRepeat

Define el comando del menú **Repetir** y el nombre del procedimiento ejecutado al seleccionar la opción **Repetir** (menú **Edición**), después de la ejecución del procedimiento que define esta propiedad.

OnTime

Programa la ejecución de un procedimiento en un momento determinado.

OnUndo

Define el texto de la opción de menú **Deshacer** y el nombre del procedimiento ejecutado cuando se selecciona la opción **Deshacer** (menú **Edición**), después de la ejecución del procedimiento que define esta propiedad.

Quit

Sale de Microsoft Excel.

Repeat

Repite la última operación ejecutada desde la interfaz de usuario.

SaveWorkspace

Guarda el área de trabajo en curso.

SendKeys

Simula la pulsación de teclas en la aplicación activa.

Undo

Deshace la última operación realizada desde la interfaz de usuario.

RecordMacro

Graba el código si se activa el grabador de macros.

Run

Ejecuta un procedimiento o llama a una función.

Wait

Hace una pausa en la ejecución de la macro hasta un momento especificado. Devuelve el valor **True** cuando llega la hora especificada.

f. Métodos relativos al correo

MailLogon

Se conecta y abre una sesión de correo MAPI o de Microsoft Exchange. Si Microsoft Mail no se está

ejecutando aún, se debe usar este método para establecer una sesión de correo que permita el uso de funciones de distribución de documentos o de mensajes.

MailLogoff

Cierra una sesión de correo MAPI abierta por Microsoft Excel.

g. Otros métodos

ActivateMicrosoftApp

Activa una aplicación de Microsoft. Si esta ya está en ejecución, el método la activa. Si no lo está, el método abre una nueva instancia de la aplicación (ejemplo: Application.ActivateMicrosoftAPP xlMicrosoftWord).

CentimetersToPoints

Convierte centímetros en puntos (un punto equivale a 0,035 centímetros).

CheckingSpelling

Comprueba la ortografía de una palabra y devuelve **True** si la palabra se encontró en uno de los diccionarios.

DisplayXMLSourcePane

Abre el panel de tareas Office XML Source y muestra la asignación XML especificada por el argumento XmlMap.

Help

Muestra un tema de ayuda.

InchesToPoints

Convierte pulgadas en puntos.

MacroOptions

Opciones del cuadro de diálogo **Opciones de macro**.

RegisterXLL

Carga un recurso de código XLL y registra automáticamente las funciones y comandos que contiene.

Volatile

Define una función personalizada como volátil. Una función volátil se recalcula cada vez que se realiza un cálculo en cualquier celda de la hoja de cálculo.

5. Ejemplos de códigos que usan el objeto Application

a. Modificación de la interfaz de Excel

```
Sub Interfaz()

   With Application
      ’   Título de la ventana de la aplicación
      .Caption = "Aplicación " & .Name _
                 & " Versión " & .Version
      ’   Texto de la barra de estado
      .StatusBar = "Ejemplos VBA Excel 2016"
      ’   Estilo de referencia L1C1
     .ReferenceStyle = xlR1C1
      ’   Maximiza la ventana de la aplicación
      .WindowState = xlMaximized
      ’   Modifica la fuente por defecto
      .StandardFont = "Verdana"
      .StandardFontSize = 11
      ’   Oculta la barra de herramientas formato
      ’   Empleo de la colección CommandBars
      .CommandBars(_Formatting").Visible = False
      ’   Muestra la barra de herramientas Visual Basic
      .CommandBars("Visual Basic").Visible = True
   End With

End Sub
```

b. Creación de una lista personalizada

```
Dim i As Integer
Dim iNumList As Integer
Dim TabList As Variant

With Application
   ’   Selecciona las columnas 1, 3 y 5
   .AddCustomList Array("Este", "Norte", "Oeste", "Sur", "Centro")
   ’   Copia el contenido de la lista en una matriz
   NumList = .GetCustomListNum(Array("Este", "Norte", _
   "Oeste",  "Sur", "Centro"))
   TabList = .GetCustomListContents(NumList)
   ’   Muestra el contenido de la lista en columnas
   For i = LBound(TabList, 1) To UBound(TabList, 1)
       Cells(i, 1) = TabList(i)
   Next i
End With
```

c. Selección de columnas no consecutivas

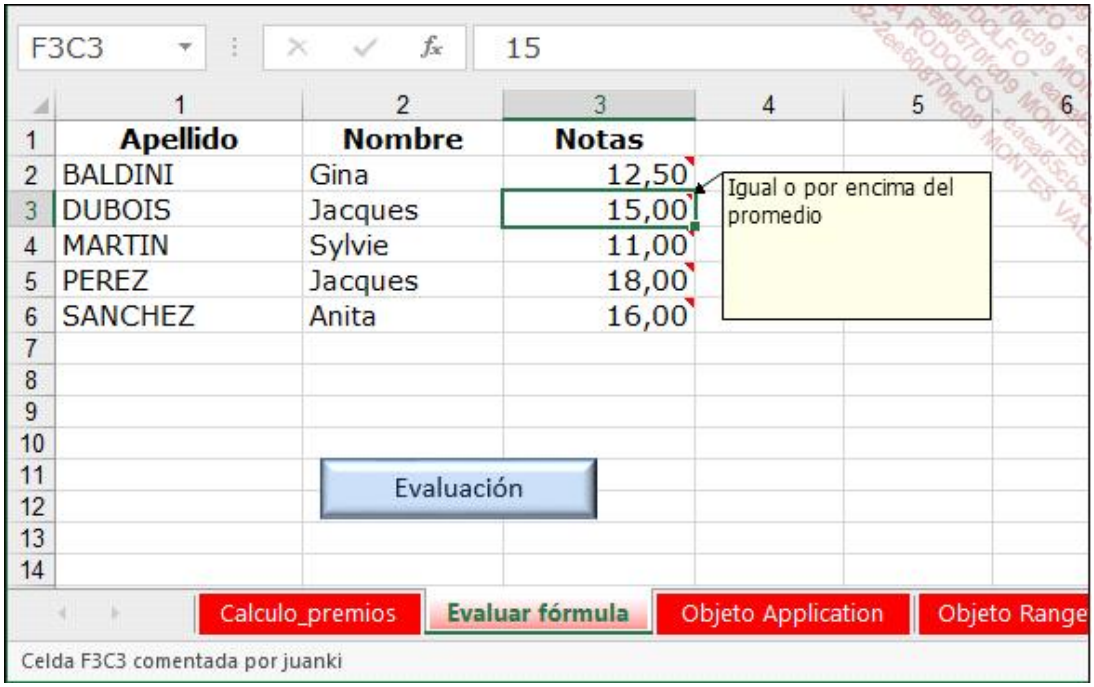
```
Dim oMultipleRange As Range
```

```
'   Crea un objeto Range formado por las columnas 1,3 y 5
Set oMultipleRange = Application.Union(Cells(1, 1), _
     Cells(1, 3), Cells(1, 5)).EntireColumn
'   Pone en negrita las celdas y las selecciona
oMultipleRange.Font.Bold = True
oMultipleRange.Select
```

d. Evaluación del resultado de una fórmula

Este ejemplo **calcula el promedio y el valor máximo de un rango de celdas** que contiene notas. Las notas se comparan, a continuación, con estos valores y se agrega un comentario para cada una.

```
Sub Evaluacion()
   Dim dProm As Double
   Dim dMax As Double
   Dim oNotas As Name
   Dim oCelda As Range
   '   Selecciona del rango de celdas llamado Notas
   Sheets("Evaluacion formula").Activate
   Set oNotas = ThisWorkbook.Names("Notas")
   oNotas.RefersToRange.Select
   '   Elimina los comentarios
   Selection.ClearComments
   '   Calcula el promedio y el máximo
   dProm = Evaluate("Average(notas)")
   dMax = Evaluate("Max(notas)")
   '   Muestra un comentario para cada nota
   For Each oCelda In Selection
       With oCelda
             Select Case .Value
                Case Is = dMax
                   .AddComment "El mejor"
                Case Is < dProm
                   .AddComment "Por debajo del promedio"
                   Case Else
                   .AddComment "Igual o por encima del promedio"
             End Select
       End With
   Next oCelda

End Sub
```

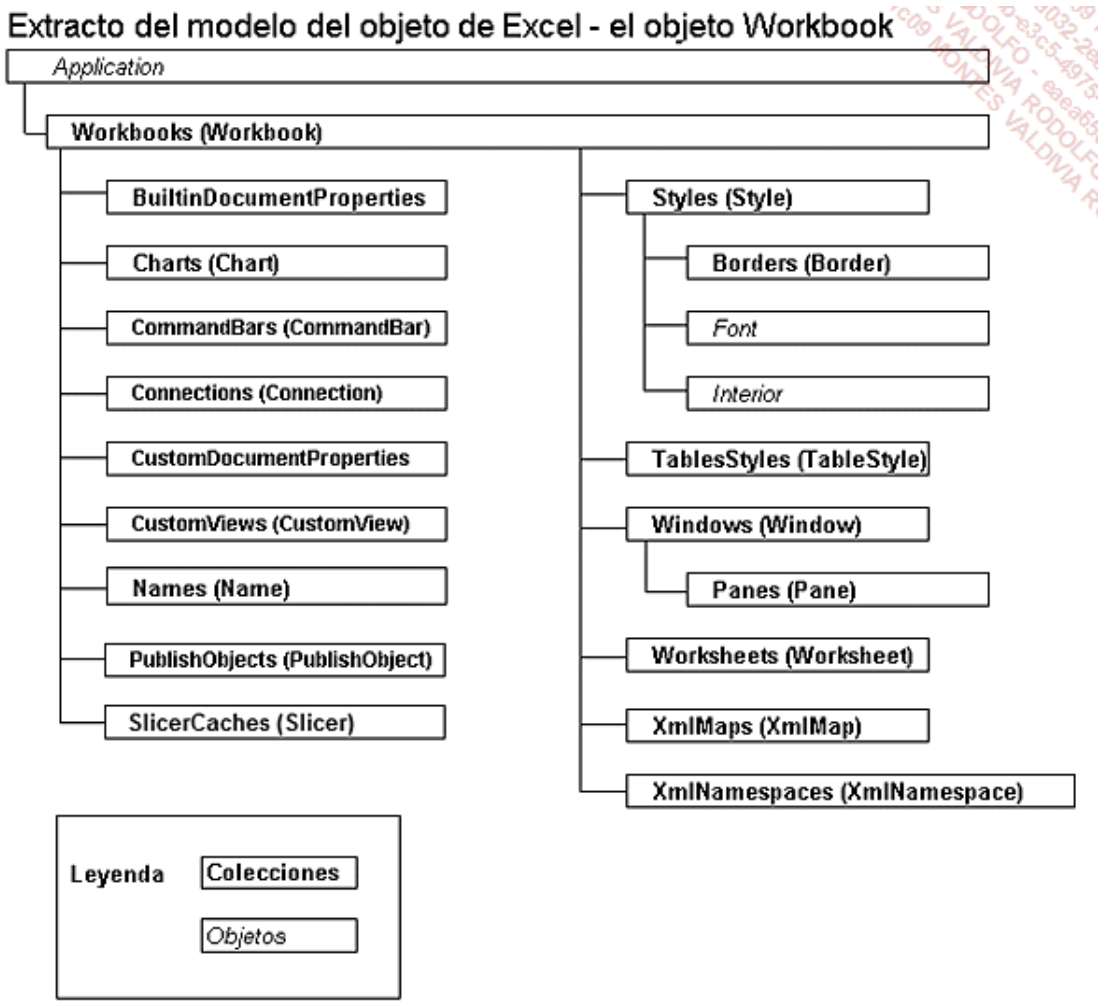


Objeto Workbook

Este objeto representa un libro de Microsoft Excel. El objeto Workbook es un miembro de la colección **Workbooks**.

Las siguientes propiedades del objeto **Application** devuelven un objeto Workbook:

- Workbooks
- ActiveWorkbook
- ThisWorkbook



1. Objetos y colecciones

Objetos

Theme

Objeto que representa el tema aplicado al libro.

VBProject

Objeto que representa el proyecto de Visual Basic asociado a un libro.

WebOptions

Opciones relativas a la grabación y apertura de una página web.

Colecciones

BuiltinDocumentProperties

Colección de las propiedades (autor, título, objeto, palabras clave, etc.) del libro.

Charts

Colección de los gráficos de un libro.

CommandBars

Colección de las barras de comandos de Excel.

Connections

Colección de las conexiones a orígenes de datos para el libro.

CustomDocumentProperties

Colección de las propiedades de un libro (título, autor, comentarios, etc.).

CustomViews

Colección de las vistas personalizadas de un libro.

Names

Colección de los rangos con nombre de un libro.

PivotTables

Colección de tablas dinámicas contenidas en un libro.

PublishObjects

Colección de los elementos de un libro grabado como página web y que se pueden actualizar.

SlicerCaches

Colección de los objetos SlicerCaches asociados a un libro.

Styles

Colección de los estilos de un libro.

TableStyles

Colección de los distintos estilos aplicables a una tabla.

Windows

Colección de las ventanas de la aplicación de Excel.

Worksheets

Colección de las hojas de cálculo de un libro.

XmlMaps

Colección de los objetos **XmlMap** que se han agregado a un libro. Estos objetos se usan para administrar la relación entre los rangos de lista y los elementos de un esquema XML.

XmlNamespaces

Colección de los espacios de nombres XML contenidos en el libro especificado.

2. Propiedades

a. Propiedades relativas a la actualización y registro de libros

CreateBackup

Booleano. Indica si se crea una copia de seguridad cuando se graba el archivo.

EnableAutoRecover

Booleano. Activa o desactiva la opción **Autorrecuperación**.

Saved

Booleano. Indica si el libro especificado no ha sido modificado después de la última grabación.

SaveLinkValues

Booleano. Indica si Microsoft Excel guarda los valores de los vínculos externos con el libro.

UpdateLinks

Constante. Parámetro del libro para la actualización de los vínculos OLE incorporados.

UpdateRemoteReferences

Booleano. Indica si Microsoft Excel actualiza las referencias remotas del libro.

b. Propiedades relativas a libros compartidos

AutoUpDateFrequency

Entero largo. Devuelve o define el tiempo, en minutos, entre dos actualizaciones automáticas en el libro compartido. Si esta propiedad recibe el valor 0, la actualización solamente se hace al guardar el libro.

AutoUpDateSaveChanges

Booleano. Indica si las modificaciones realizadas en el libro compartido se transmiten a los otros usuarios cuando el libro se actualiza automáticamente.

ChangeHistoryDuration

Entero largo. Devuelve o establece el número de días que se muestra en el historial de cambios del libro compartido.

ConflictResolution

Constante. Devuelve o define la forma en que se resuelven los conflictos cuando se actualiza un libro compartido.

HighlightChangesOnScreen

Booleano. Indica si las modificaciones en el libro compartido se resaltan en la pantalla.

KeepChangeHistory

Booleano. Indica si está habilitado el seguimiento de cambios en el libro compartido.

ListChangesOnNewsheet

Booleano. Indica si las modificaciones del libro compartido se muestran en una nueva hoja de cálculo.

MultiUserEditing

Booleano. Indica si el libro está abierto como una lista compartida.

RevisionNumber

Entero largo. Devuelve la cantidad de veces que el libro fue grabado mientras estaba abierto como lista compartida.

ShowConflictHistory

Booleano. Indica si la hoja de cálculo **Historial de conflictos** está visible en el libro abierto como lista compartida.

UserStatus

Variant. Devuelve una matriz de dos dimensiones indexada a partir de 1, con la información de cada usuario que tenga abierto el libro como lista compartida.

HasPassword

Booleano. Indica si el libro está protegido con contraseña.

Password

Cadena de caracteres. Devuelve o define la contraseña necesaria para abrir el libro.

PasswordEncryptionAlgorithm

Cadena de caracteres. Devuelve el algoritmo que usa Microsoft Excel para encriptar las contraseñas del libro.

PasswordEncryptionProvider

Cadena de caracteres. Devuelve el nombre del proveedor del algoritmo de cifrado que utiliza Microsoft Excel para encriptar las contraseñas del libro especificado.

PasswordEncryptionKeyLength

Entero largo. Indica la longitud de la clave del algoritmo que utiliza Microsoft Excel para encriptar las contraseñas del libro especificado.

PasswordEncryptionFileProperties

Booleano. Indica si Microsoft Excel encripta las propiedades de archivo del libro.

ProtectStructure

Booleano. Indica si el orden de las hojas de cálculo del libro está protegido.

ProtectWindows

Booleano. Indica si las ventanas del libro están protegidas.

ReadOnly

Booleano. Indica si el libro fue abierto en modo de solo lectura.

ReadOnlyRecommanded

Booleano. Indica si el libro fue grabado como recomendado para solo lectura.

RemovePersonalInformation

Booleano. Indica si la información personal del libro se puede eliminar.

VBASigned

Booleano. Indica si el proyecto VBA del libro tiene firma digital.

WritePassword

Cadena de caracteres. Devuelve o define la contraseña de escritura para el libro.

WriteReserved

Booleano. Indica si el libro está protegido contra escritura.

WriteReservedBy

Cadena de caracteres. Indica el nombre del usuario que está autorizado a sobreescribir el libro.

c. Otras propiedades

AccuracyVersion

Entero. Devuelve o define si ciertas funciones de la hoja de cálculo usan los últimos algoritmos de precisión para calcular sus resultados.

Valores

0: Usar los algoritmos más precisos y más recientes (opción por defecto)

1: Usar los algoritmos de Excel 2007 o de versiones anteriores

2: Usar los algoritmos de Excel 2010

CaseSensitive

Booleano. Indica si Excel distingue las mayúsculas y las minúsculas en la comparación de contenidos.

Colors

Variant. Devuelve o define los colores de la paleta del libro. La paleta tiene 56 entradas, cada una representa un valor RGB.

Date1904

Booleano. Indica si el libro usa el sistema de fechas 1904.

DefaultPivotTableStyle

Variant. Especifica el estilo de tabla de la colección **TableStyles** que se usa como estilo por defecto para las tablas dinámicas.

DefaultSlicerStyle

Variant. Especifica el estilo del objeto **TableStyle**, utilizado como estilo por defecto para los segmentos (o slicers).

DefaultTableStyle

Variant. Especifica el estilo de tabla de la colección **TableStyles** que se usa por defecto.

DisplayDrawingObjects

Constante. Devuelve o define la manera en que se muestran las formas.

EnvelopeVisible

Booleano. Indica si son visibles el encabezado de composición de los mensajes de correo y la barra de herramientas de sobre.

FullNameURLEncoded

Cadena de caracteres. Nombre del libro; incluye su ruta en el disco.

HasVBProject

Booleano. Indica si un libro contiene código VBA. Esta propiedad es especialmente útil para determinar si un libro se debe grabar en un formato que acepte las macros.

IsAddin

Booleano. Indica si el libro se ejecuta como complemento.

PrecisionAsDisplayed

Booleano. Indica si los cálculos en el libro se realizan usando solamente los decimales visibles en las celdas.

ShowPivotChartActiveFields

Booleano. Indica o define si el panel de tareas de filtro de gráfico dinámico es visible.

ShowPivotTableFieldList

Booleano. Indica si se puede mostrar la lista de campos de una tabla dinámica.

TemplateRemoveExtData

Booleano. Indica si las referencias de datos externos se eliminan cuando el libro se graba como plantilla.

3. Lista de métodos

a. Métodos que actúan directamente sobre los libros

AddToFavorites

Agrega el libro especificado a la lista de favoritos de la barra de herramientas web.

ApplyTheme

Aplica el tema especificado al libro.

CheckInWithVersion

Guarda un libro en un servidor, a partir de un ordenador local, y define el libro como de solo lectura para evitar que sea modificado localmente.

Close

Cierra el libro especificado.

DeleteNumberFormat

Elimina del libro un formato númerico personalizado.

ExportAsFixedFormat

Publica un libro en formato PDF o XPS.

MergeWorkbook

Fusiona, en un libro abierto, los cambios realizados en otro libro.

NewWindows

Crea una copia de la ventana especificada.

OpenDatabase

Abre una base de datos y muestra la información en un nuevo libro. Devuelve un objeto **Workbook**.

Post

Envía el libro especificado a una carpeta pública. Este método solamente funciona con un cliente Microsoft Exchange conectado a un servidor Microsoft Exchange.

PrintOut

Imprime el libro especificado.

PrintPreview

Muestra la vista preliminar del libro especificado.

PurgeChangeHistoryNow

Elimina las entradas del registro de cambios del libro especificado.

RefreshAll

Actualiza los rangos de datos externos y los informes de tablas dinámicas del libro especificado.

Route

Distribuye el libro usando la lista de distribución activa.

Save

Guarda las modificaciones del libro especificado.

SaveAs

Guarda el libro especificado en otro archivo (equivale a la opción **Guardar como** del menú **Archivo**).

SaveAsCopy

Guarda una copia del libro activo en un nuevo archivo sin modificar el libro abierto en la memoria.

UpdateFromFile

Actualiza un libro de solo lectura a partir de la versión del libro guardada en el disco si esta versión es más reciente que la copia del libro cargado en memoria. Si la copia del disco no fue modificada después de cargar el libro en memoria, la copia del libro residente en memoria no se recarga.

b. Métodos relativos a la seguridad

ChangeFileAccess

Modifica los permisos de acceso al libro, lo que puede implicar la necesidad de cargar, desde el disco, una versión actualizada.

LockServerFile

Bloquea el libro en el servidor para evitar su modificación.

Protect

Protege el libro especificado para que no se pueda modificar.

ProtectSharing

Guarda el libro e impide que sea compartido.

UnProtect

Quita la protección del libro especificado.

UnprotectSharing

Desactiva la protección que impide compartir el libro y lo graba.

c. Métodos relativos a libros compartidos

AcceptAllChanges

Acepta todas las modificaciones hechas al libro compartido especificado.

CanCheckIn

Devuelve una variable Booleano que indica si Excel puede extraer un libro especificado desde un servidor.

ExclusiveAccess

Atribuye al usuario actual un acceso exclusivo al libro abierto como lista compartida.

HighlightChangesOptions

Controla cómo se muestran los cambios en un libro compartido.

RejectAllChanges

Impide los cambios sobre el libro compartido especificado.

RemoveUser

Desconecta el usuario especificado del libro compartido.

d. Métodos relacionados con datos vinculados

BreakLink

Convierte las fórmulas vinculadas a otros orígenes de Microsoft Excel u orígenes OLE en valores.

ChangeLink

Modifica un vínculo entre dos documentos.

EnableConnections

Activa las conexiones de datos en un libro.

FollowHyperlink

Muestra un documento de la caché si ya ha sido transferido a la máquina local. De lo contrario, este método resuelve el hipervínculo, transfiere el documento de destino a la máquina local y muestra el documento en la aplicación apropiada.

LinkInfo

Devuelve información acerca de la fecha y el estado de actualización del vínculo.

LinkSources

Devuelve una matriz de vínculos al libro. Los nombres de la matriz son los nombres de los documentos vinculados, ediciones o servidores DDE u OLE. Este método devuelve **Empty** si no hay vínculos en el libro.

OpenLinks

Abre los documentos de origen de uno o más vínculos.

OpenXml

Abre un archivo XML en un nuevo libro. Devuelve un objeto **Workbook**.

ReloadAs

Vuelve a cargar un libro basado en un documento HTML usando la codificación de documentos especificada.

SetLinkOnDate

Define el nombre de un procedimiento ejecutado a cada actualización de un vínculo DDE.

UpdateLink

Actualiza uno o más vínculos de Microsoft Excel, DDE o OLE.

WebPagePreview

Muestra la vista previa del libro especificado, tal como se vería al ser guardado como página web.

e. Métodos relativos al envío de libros

SendFaxOverInternet

Envía una hoja de cálculo como fax a los destinatarios especificados.

SendMail

Envía un mensaje de correo electrónico con el libro especificado.

SendForReview

Envía un mensaje de correo electrónico con el libro que hay que revisar a los destinatarios especificados.

EndReview

Termina la revisión de un archivo enviado para este fin con el método **SendForReview**.

ReplyWithChanges

Envía un mensaje de correo electrónico al autor de un libro enviado para revisión y le informa de que

la revisión ha sido realizada.

f. Otros métodos

Los métodos relativos a la importación y exportación de archivos al formato XML (SaveAsXMLData, XmlImport, etc.) se explican en el capítulo Internet.

4. Ejemplos de códigos que usan el objeto Workbook

Para probar estos ejemplos, debe crear una carpeta C:\Ventas con la base de ejemplo de Access Contador.mdb.

a. Creación de un libro de Excel

El siguiente ejemplo permite:

- Cerrar todos los libros abiertos, excepto el libro activo.
- Crear un nuevo libro.
- Proteger el libro con contraseña.
- Agregar el libro a la lista de Favoritos.
- Guardar y cerrar el libro.

```
Private Sub NuevoLibro()
Dim oLibro As Workbook
Dim i As Integer
Dim j As Integer

'   Cierra los libros (excepto el libro activo)
'   y guarda los cambios

For Each oLibro In Workbooks
    If oLibro.Name <> ThisWorkbook.Name Then
       oLibro.Close True
    End If
Next oLibro

'   Crea un nuevo libro
Set oLibro = Application.Workbooks.Add
With oLibro
'   Protege el libro con contraseña
   .Password = "Ventas"
   .WritePassword = "W_Ventas"
'   Guarda el libro
   .SaveAs "C:\Ventas\Ventas por región"
'   Agrega el libro a los favoritos de la barra de menú Web
   .AddToFavorites
'   Cierra el libro
   .Close
End With

End Sub
```

b. Importar una base de datos y exportarla al formato HTML

El siguiente ejemplo muestra cómo:

- Abrir la tabla Clientes del archivo "Northwind 2016.accdb" en un nuevo libro.
- Exportar esta información en un archivo HTML.
- Abrir el archivo HTML.

```
Sub CreaHTMLFile()
Dim oLibro As Workbook

'   Importa la tabla Clientes de la base Access Northwind 2016
'   en un nuevo libro
Set oLibro = Workbooks.OpenDatabase _
          (Filename:=ThisWorkbook.Path & "\Northwind 2016.accdb", _
          CommandText:="SELECT * FROM CLIENTS")
'   Exporta los clientes a un archivo Html
ActiveWorkbook.SaveAs Filename:=ThisWorkbook.Path & "\Clientes.htm", _
FileFormat :=xlHtml, ReadOnlyRecommended:=False, CreateBackup:=False
'   Abre el archivo Html
Workbooks.Open Filename:=ThisWorkbook.Path & "\Clientes.htm"
End Sub
```

c. Mostrar las propiedades de un libro

Este ejemplo muestra los nombres y los valores de las diferentes propiedades del libro. Algunas de estas propiedades son accesibles al hacer clic en la pestaña **Archivo** y luego en la sección **Información**.


```
Sub Propiedades()
   Dim i As Integer
   Dim ObjProp As Object
   '   Muestra la lista de propiedades del libro
   i = 1
   For Each ObjProp In ThisWorkbook.BuiltinDocumentProperties
      On Error Resume Next
      ActiveSheet.Cells(i, 2) = ObjProp.Value
      ActiveSheet.Cells(i, 1) = ObjProp.Name
      i = i + 1
   Next
End Sub
```

d. Exportar un libro al formato PDF

Para ejecutar este código, debe previamente ejecutar el programa de instalación del complemento Excel SaveAsPDFandXPS.exe (este programa se entrega con los ejemplos).

```
Sub Export_Pdf()
   '   Exporta el archivo al formato PDF y lo abre en Acrobat Reader
   ThisWorkbook.ExportAsFixedFormat Type:=xlTypePDF, _
                                Filename:="Capítulo4", _
                                IncludeDocProperties:=True, _
                                OpenAfterPublish:=True
```

```
End Sub
```

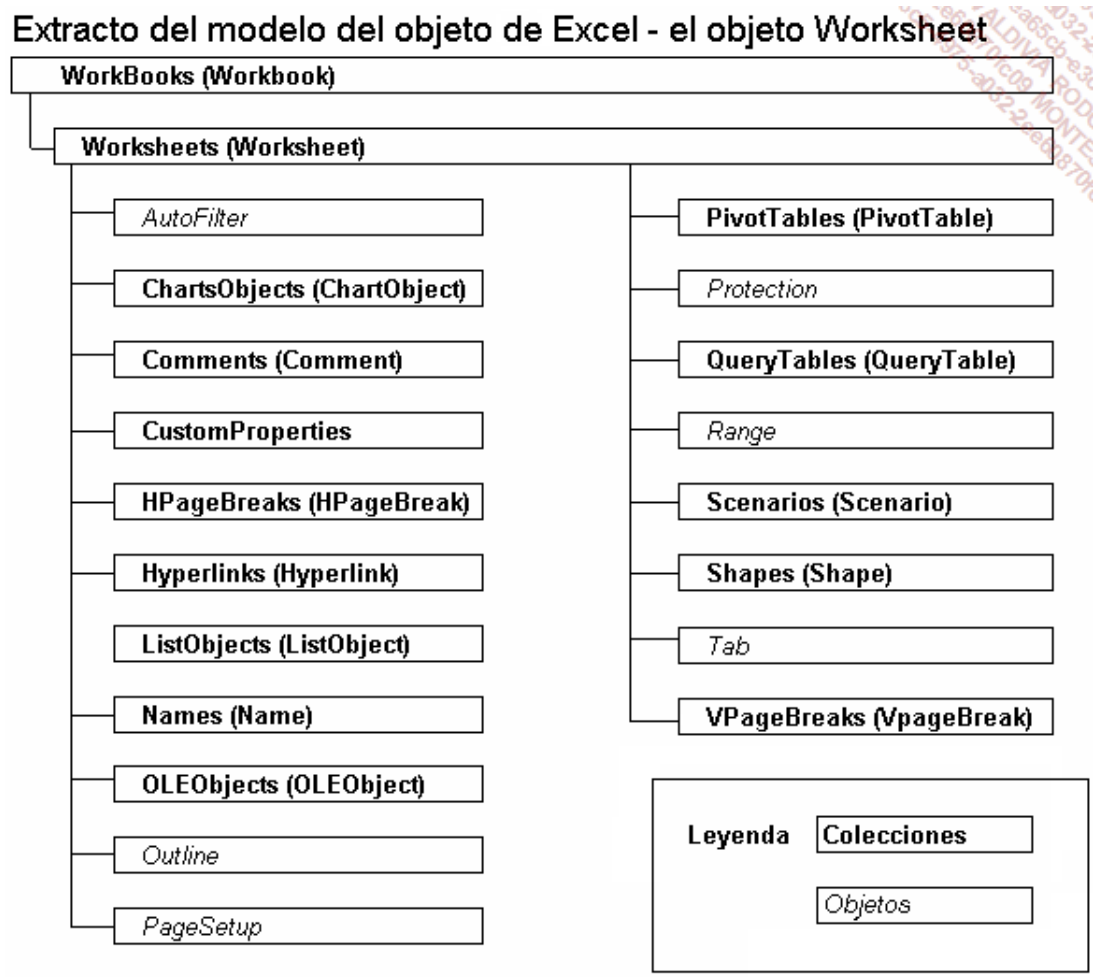
El objeto Worksheet

Este objeto representa una hoja de cálculo Excel. El objeto **Worksheet** es un miembro de la colección **WorkSheets** del objeto **Workbook**.

Las siguientes propiedades del objeto **Application** devuelven un objeto **Worksheet**:

- Worksheets
- ActiveSheet

1. Lista de objetos y colecciones



2. Objetos y colecciones

Objetos

AutoFilter

Objeto que representa el autofiltro de la hoja de cálculo especificada.

OutLine

Objeto que representa el esquema de la hoja de cálculo especificada.

PageSetup

Objeto que representa las opciones de configuración de página de la hoja de cálculo especificada.

Protection

Objeto que representa las opciones de protección para la hoja de cálculo especificada. Estas opciones son accesibles en Excel a través del menú **Herramientas - Protección - Proteger hoja**.

Range

Objeto que representa una celda o un rango de celdas (una fila, una columna, etc.).

Tab

Objeto que representa la pestaña de la hoja de cálculo especificada.

Colecciones

ChartObjects

Colección de los gráficos incrustados en la hoja de cálculo especificada.

Comments

Colección de todos los comentarios de celda de la hoja de cálculo especificada.

CustomProperties

Colección de objetos **CustomProperty** que representa la información complementaria (metadatos para XML o etiquetas inteligentes).

HPageBreaks

Colección de los saltos de página horizontales en la zona de impresión de la hoja especificada.

Hyperlinks

Colección de los hipervínculos de la hoja de cálculo especificada.

ListObjects

Colección de las listas de la hoja de cálculo especificada.

Names

Colección de los rangos de celdas con nombre de la hoja de cálculo especificada.

OLEObjects

Colección de los objetos ActiveX y objetos OLE vinculados o incrustados en la hoja de cálculo especificada.

PivotTables

Colección de los informes de tabla dinámica de la hoja de cálculo especificada.

QueryTables

Colección de las tablas de hoja de cálculo creadas a partir de datos enviados desde un origen de datos externo.

Scenarios

Colección de los escenarios de la hoja de cálculo especificada.

Shapes

Colección de todas las formas (autoformas, formas libres, objetos OLE o imágenes) presentes en la hoja de cálculo especificada.

VPageBreaks

Colección de los saltos de página verticales en la zona de impresión de la hoja especificada.

3. Propiedades

AutoFilterMode

Booleano. Indica si las flechas del menú desplegable de los autofiltros aparecen en la hoja de cálculo especificada.

ConsolidationFunction

Constante (**xlMax**, **xlMin**, **xlSum**, etc.). Devuelve la función usada para la consolidación actual.

ConsolidationOptions

Matriz de booleanos que representa las opciones relativas a la consolidación (rótulos en la fila superior, rótulos en la columna izquierda, vínculos con los datos de origen).

ConsolidationSources

Matriz de cadenas de caracteres que contiene los nombres de las hojas de origen para la consolidación actual de la hoja de cálculo especificada.

DisplayPageBreaks

Booleano. Indica si se muestran los saltos de página (automáticos y manuales) de la hoja especificada.

DisplayRightToLeft

Booleano. Corresponde a la opción **Ver la hoja actual de derecha a izquierda** del cuadro de diálogo **Herramientas** - **Opciones** - pestaña **Internacional**.

EnabledAutofilter

Booleano. Indica si las flechas del Autofiltro están activas solamente cuando está activa la protección de solo interfaz de usuario.

EnableCalculation

Booleano. Indica si Excel recalcula automáticamente la hoja de cálculo cuando es necesario.

EnableFormatConditionsCalculation

Booleano. Devuelve o define si los formatos condicionales se aplican automáticamente si es necesario.

EnableOutLining

Booleano. Indica si los símbolos del esquema están activos cuando está activa la protección de solo interfaz de usuario.

EnablePivotTable

Booleano. Indica si los controles y las acciones de la tabla dinámica están activos cuando está activa la protección de solo interfaz de usuario.

EnableSelection

Constante. Devuelve o define los elementos que se pueden seleccionar en la hoja (xlNoRestrictions, xlNoSelection, xlUnlockedCells).

FilterMode

Booleano. Indica si hay aplicado un filtro a la hoja especificada.

MailEnvelope

Representa el encabezado de los mensajes de correo electrónico para la hoja especificada.

Name

Cadena de caracteres que contiene el nombre de la hoja de cálculo.

PrintedCommentPages

Entero largo. Devuelve la cantidad de páginas de comentarios que se imprimen para la hoja de cálculo especificada.

ProtectContents

Booleano. Indica si el contenido de la hoja especificada está protegido.

ProtectDrawingObjects

Booleano. Indica si las formas gráficas están protegidas.

ProtectionMode

Booleano. Indica si está activa la protección de solo interfaz de usuario.

ProtectScenarios

Booleano. Indica si están protegidos los escenarios de hoja de cálculo.

ScrollArea

Cadena de caracteres. Devuelve o establece el rango en el que está permitido el desplazamiento de la hoja de cálculo especificada.

StandardHeight

Real doble. Devuelve el alto estándar (valor por defecto) de las filas de la hoja de cálculo especificada.

StandardWidth

Real doble. Devuelve el ancho estándar (valor por defecto) de las columnas de la hoja de cálculo especificada.

Type

Constante. Devuelve o define el tipo de la hoja de cálculo especificada (xlChart, xlDialogSheet, etc.).

Visible

Booleano. Indica si la hoja de cálculo especificada está visible.

4. Métodos

Activate

Activa la hoja de cálculo especificada. Equivale a hacer clic sobre la pestaña de la hoja.

Calculate

Recalcula las celdas de la hoja de cálculo especificada.

CheckSpelling

Efectúa la verificación ortográfica de la hoja de cálculo especificada (equivale a la opción **Ortografía** del menú **Herramientas**).

CircleInvalid

Rodea con un círculo las entradas incorrectas en la hoja de cálculo especificada.

ClearArrows

Borra las flechas de auditoría de la hoja de cálculo especificada.

ClearCircles

Borra los círculos que rodean las entradas incorrectas de la hoja de cálculo.

Copy

Hace una copia de la hoja de cálculo especificada (antes o después de una de las hojas del libro).

Delete

Elimina la hoja de cálculo especificada.

Evaluate

Calcula la expresión pasada como argumento y devuelve el resultado. La expresión debe corresponder a una fórmula de cálculo en inglés.

ExportAsFixedFormat

Publica una hoja de cálculo en formato PDF o XPS.

Move

Mueve la hoja de cálculo especificada a una posición dada (antes o después de una de las hojas del libro).

Paste

Pega el contenido del portapapeles en la hoja de cálculo especificada.

PasteSpecial

Pega el contenido del portapapeles en la hoja de cálculo especificada respetando el formato especificado (pegado especial).

PrintOut

Imprime la hoja de cálculo especificada.

PrintPreview

Muestra la vista preliminar de la hoja de cálculo especificada.

Protect

Protege la hoja de cálculo especificada.

ResetAllPageBreaks

Redefine los saltos de página de la hoja de cálculo especificada.

SaveAs

Guarda la hoja de cálculo en un nuevo libro.

Select

Selecciona la hoja de cálculo.

SetBackgroundPicture

Define el gráfico de fondo de la hoja de cálculo especificada.

ShowAllData

Muestra todas las filas de la lista actualmente filtrada.

ShowDataForm

Muestra los datos de la hoja de cálculo especificada como formulario (corresponde a la opción **Formulario** del menú **Datos**).

UnProtect

Desactiva la protección de la hoja de cálculo especificada.

Los métodos relativos a mapas de datos XML (**XmlDataQuery**, **XmlMapQuery**, etc.) se explican en el capítulo Internet.

5. Ejemplos de códigos que usan el objeto Worksheet

a. Ordenar las hojas de cálculo de un libro

El siguiente ejemplo permite:

- Ordenar las hojas de un libro.
- Modificar el color de las pestañas de cada hoja.

```
Sub OrganizarHojas()
   Dim oHoja As Worksheet
   Dim i As Integer

   '   Ordena las hojas del libro activo
```

```
   OrdenaHojas ActiveWorkbook

   '   Recorre las hojas de cálculo
   For i = 1 To ActiveWorkbook.Worksheets.Count
       Set oHoja = ActiveWorkbook.Worksheets(i)
       With oHoja
          '   Modifica el color de la pestaña
          .Tab.Color = vbRed
       End With
   Next i

   End Sub

   Private Sub OrdenaHojas(oLibro1 As Workbook)
   Dim i As Integer
   Dim j As Integer
   '   Procedimiento para ordenar alfabéticamente las hojas de cálculo
   '   de un libro
   With oLibro1
       For i = 1 To .Worksheets.Count
           For j = 1 To i - 1
               If .Worksheets(i).Name < .Worksheets(j).Name Then
                  .Worksheets(i).Move before:=.Worksheets(j)
               End If
           Next j
       Next i
   End With

End Sub
```

b. Protección de las hojas de cálculo de un libro

Este ejemplo protege las hojas de cálculo de un libro permitiendo las siguientes autorizaciones: formato de celdas, agregar columnas y filas, ordenamiento y autofiltros. Las otras operaciones (eliminar filas o columnas, modificación de escenarios, etc.) quedan prohibidas.

```
Sub ProtegeHojas()
   Dim oShtCurrent As Worksheet

   For Each oShtCurrent In ActiveWorkbook.Worksheets
        oShtCurrent.Protect Password:="Contraseña", _
          contents:=True, AllowFormattingCells:=True, _
          AllowInsertingColumns:=True, AllowInsertingRows:=True, _
          AllowSorting:=True, AllowFiltering:=True
   Next oShtCurrent

End Sub
```

c. Ordenar una tabla

Este ejemplo ordena una tabla según tres columnas: ciudad, apellido y nombre.

```
Sub Ordena_Clientes()

    '   Ordena la tabla de clientes por ciudad, apellido y nombre
    Application.Goto Reference:="Clientes"
    With ActiveWorkbook.Worksheets("Clientes").Sort
         .SortFields.Clear
         .SortFields.Add Key:=Range("C2:C30"), SortOn:=xlSortOnValues, _
          Order:=xlAscending, DataOption:=xlSortNormal
         .SortFields.Add Key:=Range("B2:B30"), SortOn:=xlSortOnValues, _
          Order:=xlAscending, DataOption:=xlSortNormal
         .SortFields.Add Key:=Range("D2:D30"), SortOn:=xlSortOnValues, _
          Order:=xlAscending, DataOption:=xlSortNormal
          .SetRange Range("A1:G30")
          .Header = xlYes
          .MatchCase = False
          .Orientation = xlTopToBottom
          .Apply
    End With
End Sub
```

El objeto Range

El objeto Range representa un rango de celdas y puede estar constituido por:

- Una celda.
- Una fila.
- Una columna.
- Un rango de celdas contiguas.
- Un rango de celdas no contiguas.
- Un rango 3D.

1. Propiedades y métodos que devuelven un objeto Range

Propiedades que devuelven un objeto Range

Propiedad	Objeto contenedor	Objeto devuelto
ActiveCell	Application Window	Objeto Range que representa la **primera celda** activa de la ventana activa o especificada.
Areas	Range	Colección que agrupa **todos los rangos** de una selección múltiple.
Cells	Application Range Worksheet	Objeto Range que representa **una celda o una colección de celdas**: • De la hoja activa si el objeto contenedor es Application. • Del rango especificado si el objeto contenedor es Range. • De la hoja de cálculo especificada si el objeto contenedor es Worksheet.
CircularReference	Worksheet	Objeto Range que representa el rango que contiene **la primera referencia circular** de la hoja.
Columns	Application Range Worksheet	Objeto Range que representa **las columnas**: • De la hoja activa si el objeto contenedor es Application. • Del rango especificado si el objeto contenedor es Range. • De la hoja especificada si el objeto contenedor es Worksheet.
CurrentRegion	Range	Objeto Range que representa el objeto Range especificado, limitado por toda combinación de filas y columnas vacías.
Dependents	Range	Objeto Range que representa el rango que contiene **todas las celdas dependientes** de una celda dada. Puede ser una selección múltiple (unión de objetos Range) si hubiera muchas celdas dependientes.
DirectDependents	Range	Objeto Range que representa el rango que contiene todas las celdas **directamente dependientes** de una celda dada.
DirectPrecedents	Range	Objeto Range que representa el rango que contiene todas las celdas **directamente antecedentes** de una celda dada.
EntireColumn	Range	Objeto Range que representa **una o más columnas enteras** del rango especificado.
EntireRow	Range	Objeto Range que representa **una o más filas enteras** del rango

		especificado.
End	Range	Objeto Range que representa **la celda situada al final de la zona** de rango especificado. Corresponde a la combinación de teclas [Fin] [Flecha arriba], [Fin][Flecha abajo], [Fin][Flecha izquierda] o [Fin] [Flecha derecha].
Next	Range	Objeto Range que representa la **siguiente celda** del rango especificado.
Offset	Range	Objeto Range especificado desplazado una o más filas o columnas.
Precedents	Range	Objeto Range que representa el rango que contiene **todas las celdas antecedentes** de una celda dada.
Previous	Range	Objeto Range que representa la **celda precedente** de una celda dada.
Range	Application Range Worksheet	Objeto Range que representa **un rango de celdas**: • De la hoja activa si el objeto contenedor es Application. • Del rango especificado si el objeto contenedor es Range. • De la hoja especificada si el objeto contenedor es Worksheet.
Rows	Application Range Worksheet	Objeto Range que representa **todas las filas**: • De la hoja activa si el objeto contenedor es Application. • Del rango especificado si el objeto contenedor es Range. • De la hoja especificada si el objeto contenedor es Worksheet.
UsedRange	Worksheet	Objeto Range que representa el rango usado en su totalidad por la hoja de cálculo especificada.

Las propiedades **Next** y **Previous**, aplicadas a los objetos **Worksheet** y **Chart**, devuelven un objeto tipo **Worksheet** que representa las hojas siguiente y anterior, respectivamente.

Métodos que devuelven un objeto Range

Método	Objeto contenedor	Objeto devuelto
Intersect	Application	Objeto Range que representa **la intersección** rectangular de varios rangos.
Union	Application	Objeto Range que representa **la unión** de varios rangos contiguos o discontiguos.

2. Sintaxis de las propiedades que devuelven un objeto Range

Cells

```
Objeto.Cells ([RowIndex],[ColumnIndex])
```

`RowIndex` Número de fila de la celda.

`ColumnIndex` Número de columna de la celda.

Si no se indica ningún argumento, Cells devuelve la colección de celdas del rango especificado.

Ejemplo

El siguiente ejemplo modifica el contenido y el color de celdas.

```
Sub LlenaHoja()
   Dim oCelda As Range

   '   Modifica el contenido de la celda B1 de la hoja activa
   Application.Cells(1, 2) = "Enero"
   '   Modifica el contenido de la celda B2 de la hoja activa
   ActiveSheet.Range("A1:G10").Cells(2, 2) = "Febrero"
   '   Modifica el contenido de la celda B3 de la hoja activa
   ActiveSheet.Cells(3, 2) = "Marzo"
   '   Modifica el color de las celdas C1, C2, D1, D2
   For Each oCelda In Range("C1:D2")
         oCelda.Interior.Color = vbRed
   Next oCelda
End Sub
```

Range

```
Objeto.Range (Cell1,[Cell2])
```

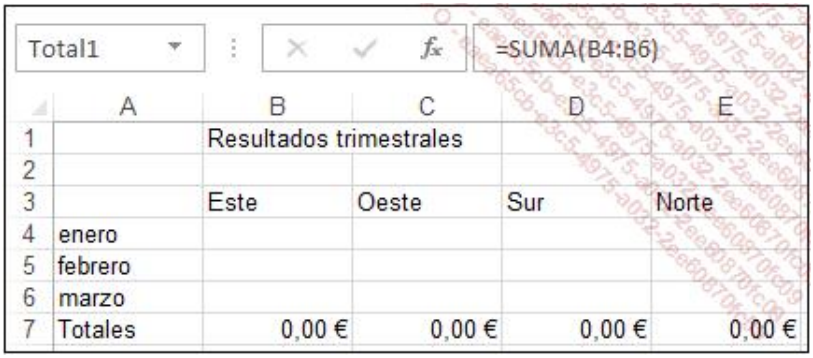
Donde `Cell1` y `Cell2` pueden ser:

- Una celda (por ejemplo: "A1").
- Un rango de celdas (por ejemplo: "A1:B7").
- Un nombre de celda (por ejemplo: "Totales").

Si `Cell2` está especificado, `Range` devuelve un rango de celdas contiguas que incluye los dos rangos especificados.

Ejemplo

Este ejemplo crea la siguiente tabla en una hoja de cálculo.



	A	B	C	D	E
1		Resultados trimestrales			
2					
3		Este	Oeste	Sur	Norte
4	enero				
5	febrero				
6	marzo				
7	Totales	0,00 €	0,00 €	0,00 €	0,00 €

```
Sub HojaResultado()
   Dim i As Integer

   With Application.ActiveSheet
      .Range("B1").Value = "Resultados trimestrales"

      '   Meses en columna
      For i = 1 To 3
          .Range("A" & i + 3).Value = _
          Format((DateValue("01/" & i & "/01")), "MMMM")
      Next i

      '   Regiones en fila
      Range("B3:E3").Value = Array("Este", "Oeste", "Sur", "Norte")
      '   Formato de celdas
      Range("B4:E7").NumberFormat = "# ##0.00 €"
      Range("A7").Value = "Totales"

      '   Nombra las celdas que contienen los totales
      '   Asigna una fórmula a las celdas con nombre
      Range("B7").Name = "Total1"
      Range("Total1").Formula = "=SUM(B4:B6)"
      Range("C7").Name = "Total2"
      Range("Total2").Formula = "=SUM(C4:C6)"
      Range("D7").Name = "Total3"
      Range("Total3").Formula = "=SUM(D4:D6)"
      Range("E7").Name = "Total4"
      Range("Total4").Formula = "=SUM(E4:E6)"
   End With

End Sub
```

OffSet

```
Objeto.OffSet ([rowOffset],[columnOffset])
```

rowOffset	Cantidad de filas de desplazamiento.
colOffset	Cantidad de columnas de desplazamiento.

`colOffset` y `rowOffset` pueden contener valores negativos.

Ejemplo

El siguiente ejemplo devuelve la dirección del rango que resulta de un desplazamiento de filas y columnas.

```
Sub DevuelveOffset()

   With Range("B5:C7")
      '   Desplazamiento de una fila hacia arriba
      '   Devuelve $A$5:$B$7
```

```
      MsgBox .Offset(0, -1).Address
      '   Desplazamiento de dos columnas hacia la derecha
      '   Devuelve $B$7:$C$9
      MsgBox .Offset(2, 0).Address
   End With

End Sub
```

Areas

```
Objeto.Areas ([Index])
```

Index	Número del rango en distintos rangos del objeto.

Si no se indica ningún argumento, Areas devuelve la colección de rangos especificada.

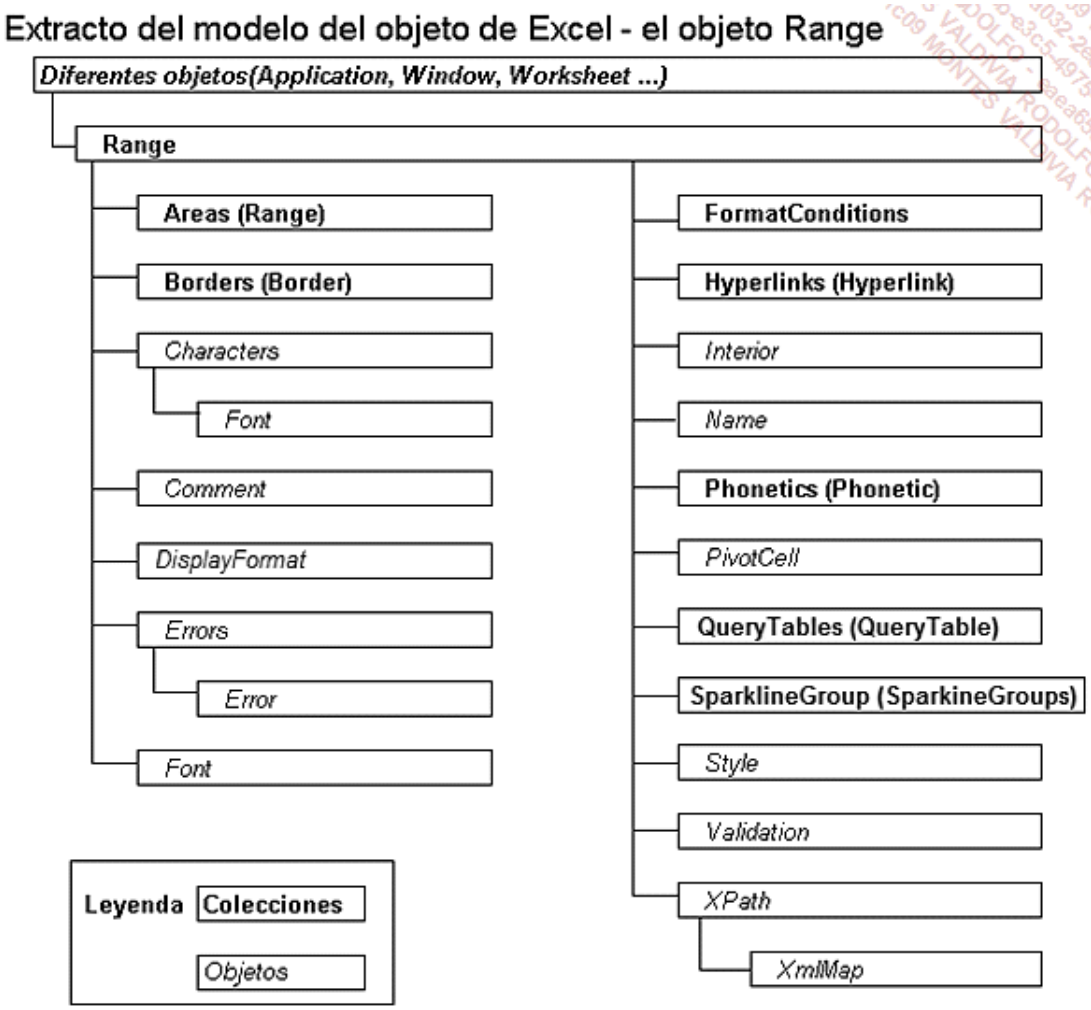
Ejemplo

Este ejemplo permite:

- Crear una zona formada por varios rangos de celdas discontinuos.
- Llenar el primer rango a partir de una matriz.
- Poner en negrita la fuente para todos los rangos.

```
Sub MuchosRangos()
   Dim oZonaTot As Range
   Dim i As Integer

   '   Unión de muchos rangos discontinuos
   Set oZonaTot = Union(Range("B3:E3"), Range("B1:B5"),  _
         Range("J2:F6"))

   With ZonaTot
   '   Primer rango completado a partir de una matriz
      .Areas(1).Value = Array("Este", "Oeste", "Sur", "Norte")
   '   Pone en negrita los caracteres de todos los rangos
      For i = 1 To 3
          .Areas(i).Font.Bold = True
      Next i
      .Select
   End With

End Sub
```

3. Lista de objetos y colecciones



Objetos

Characters

Objeto que representa los caracteres del texto de la celda especificada.

Comment

Objeto que representa el comentario asociado a la celda.

DisplayFormat

Objeto que representa las opciones de visualización para el rango especificado.

Errors

Objeto que representa los errores en el rango especificado.

Font

Objeto que contiene los atributos de fuente (nombre, tamaño, color, etc.) del rango especificado.

Interior

Objeto que representa el relleno de las celdas del rango especificado.

Name

Objeto que representa un nombre para un rango de celdas (celdas con nombre).

PivotCell

Objeto que representa una celda en un informe de tabla dinámica.

Style

Objeto que representa el estilo aplicado al rango especificado.

Validation

Objeto que representa la validación de datos aplicada al rango especificado.

XPath

Objeto que representa un XPath (ruta XML) mapeado en el rango de celdas especificado.

Colecciones

Areas

Colección de todos los rangos en una selección de muchas zonas.

Borders

Colección de todos los bordes del rango de celdas especificado.

FormatConditions

Colección de los formatos condicionales del rango especificado.

HyperLinks

Colección de los hipervínculos del rango especificado.

Phonetics

Colección de objetos que contienen la información de una cadena de texto fonética específica en una celda.

QueryTables

Colección de objetos que representa las tablas de hoja de cálculo creadas a partir de datos devueltos por un origen de datos externos.

SparklineGroups

Colección de los objetos que representan el conjunto de minigráficos para el rango especificado.

4. Propiedades

a. Propiedades relacionadas con la posición y el formato de las celdas

AllowEdit

True si el rango puede ser modificado en una hoja de cálculo protegida.

AddressLocal

Devuelve la referencia del rango especificado en el idioma del usuario.

Address

Devuelve la referencia del rango en el lenguaje de la macro.

Column

Devuelve el número de la primera columna de la primera zona del rango especificado.

ColumnWidth

Devuelve o define el ancho de todas las columnas del rango especificado.

HorizontalAlignment

Constante. Define o devuelve el tipo de alineación horizontal.

IndentLevel

Devuelve o define el nivel de sangría efectivo.

Row

Devuelve el número de la primera fila de la primera zona del rango.

RowHeight

Devuelve el alto, medido en puntos, de las filas del rango especificado.

UseStandardHeight

True si el alto de fila del objeto **Range** es igual al alto estándar de la hoja.

UseStandardWidth

True si el ancho de columna del objeto **Range** es igual al ancho estándar de la hoja.

VerticalAlignment

Constante. Define o devuelve el tipo de alineación vertical.

b. Propiedades que se relacionan con el contenido de celdas y con las fórmulas

CountLarge

Determina el valor máximo en el rango.

Formula

Devuelve o define la fórmula en el estilo de referencia A1.

FormulaLocal

Devuelve o define la fórmula del objeto, usando las referencias de estilo A1 en el idioma del usuario.

FormulaR1C1

Devuelve o define la fórmula del objeto, usando las notaciones de estilo R1C1.

FormulaR1C1Local

Devuelve o define la fórmula, usando las notaciones de estilo R1C1 en el idioma del usuario.

PrefixCharacter

Devuelve el prefijo de alineación de la celda.

Text

Valor de la celda especificada con el formato especificado (contenido visible de la celda).

Value

Valor de la celda especificada. Si la celda está vacía, la propiedad **Value** devuelve el valor **Empty** (use la función **IsEmpty** para testear este caso). Si el objeto **Range** contiene varias celdas, devuelve una matriz de valores (use la función **IsArray** para testear este caso).

WrapText

Booleano. **True** si Microsoft Excel inserta automáticamente retornos de carro en el texto del objeto.

c. Otras propiedades

MergeCells

True si el rango o el estilo contiene celdas combinadas.

ListHeaderRows

Devuelve la cantidad de filas de encabezado en el rango especificado.

5. Métodos

a. Métodos que devuelven un objeto

ColumnDifferences

Devuelve un objeto **Range** que representa las celdas donde el contenido es diferente al de la celda de comparación de cada columna.

Find

Busca una información específica en un rango y devuelve un objeto **Range** que representa la primera celda donde aparece esa información.

FindNext

Continúa una búsqueda (siguiente celda) iniciada con el método **Find**.

FindPrevious

Continúa una búsqueda (celda anterior) iniciada con el método **Find**.

RowDifferences

Devuelve un objeto **Range** que representa las celdas en las que el contenido es diferente al contenido de la celda de comparación de cada fila.

SpecialCells

Devuelve un objeto **Range** que representa las celdas que corresponden al tipo y al valor especificados.

b. Métodos que se relacionan con la presentación de las celdas

AddComment

Agrega un comentario al rango.

AutoFit

Modifica el ancho de las columnas del rango o el alto de las filas para ajustar sus datos.

BorderAround

Agrega un borde a un rango y define las propiedades **Color**, **LineStyle** y **Weight** del nuevo borde.

ClearComments

Quita todos los comentarios de celda del rango especificado.

ClearFormats

Elimina los formatos de las celdas.

ClearNotes

Borra las notas escritas y sonoras de todas las celdas del rango especificado.

InsertIndent

Inserta una sangría en las celdas del rango especificado.

Justify

Reorganiza el texto en un rango de forma que lo llene de manera uniforme.

Merge

Combina las celdas.

NoteText

Devuelve o define las notas de celdas asociadas a la celda ubicada en la esquina superior izquierda del rango.

Sort

Ordena un rango de valores.

TextToColumns

Redistribuye en varias columnas una columna de celdas que contiene texto.

UnMerge

Separa una celda combinada en celdas individuales.

c. Métodos relacionados con el contenido de las celdas

AllocateChanges

Efectúa la reescritura para para todas las celdas modificadas en un rango basado en una fuente de datos OLAP.

AutoFill

Ejecuta un llenado incremental en las celdas del rango especificado.

AutoComplete

Devuelve una coincidencia de la funcionalidad Autocompletar de la lista.

ClearContents

Borra el contenido de las celdas.

ClearHyperlinks

Elimina todos los hipervínculos del rango especificado.

Consolidate

Consolida datos que provienen de varios rangos situados en diferentes hojas de cálculo en un único rango situado en una única hoja de cálculo.

Copy

Copia al portapapeles el objeto **Range** del rango especificado.

CopyFromRecordSet

Copia el contenido de un objeto **Recordset** ADO o DAO en una hoja de cálculo, comenzando en la esquina superior izquierda del rango especificado.

CopyPicture

Copia el objeto seleccionado en el portapapeles en forma de imagen.

Cut

Corta el objeto y lo guarda en el portapapeles o lo pega en un destino especificado.

Delete

Elimina las celdas e indica cómo reemplazar las celdas eliminadas.

DiscardChanges

Descarta todos los cambios de las celdas revisadas del rango.

FillDown

Rellena un rango hacia abajo.

FillLeft

Rellena un rango hacia la izquierda.

FillRight

Rellena un rango hacia la derecha.

FillUp

Rellena un rango hacia arriba.

FunctionWizard

Inicia el Asistente para funciones en la celda situada en la esquina superior izquierda del rango.

Insert

Inserta celdas e indica cómo desplazar las celdas.

Parse

Redistribuye un rango de datos y lo divide en varias celdas. Distribuye el contenido del rango de manera que llene varias columnas adyacentes; el rango no puede tener más de una columna de ancho.

PasteSpecial

Efectúa el pegado especial de un objeto **Range** que proviene del portapapeles, en el rango especificado.

RemoveDuplicates

Elimina los datos repetidos en un rango de valores.

Replace

Busca y reemplaza caracteres en las celdas del rango especificado. El uso de este método no cambia la selección ni la celda activa.

d. Métodos relacionados con los nombres de celdas

ApplyNames

Define nombres para las celdas del rango especificado.

CreateNames

Crea nombres en el rango especificado en función de los rótulos de texto de la hoja.

ListNames

Pega una lista con los nombres de la hoja de cálculo que no están ocultos, comenzando por la primera celda del rango.

e. Métodos relacionados con los filtros

AdvancedFilter

Filtra o copia los datos de una lista en función de una zona de criterios.

AutoFilter

Filtra una lista.

f. Métodos relacionados con el modo esquema

ApplyOutlineStyles

Aplica los estilos del esquema al rango especificado.

AutoOutline

Crea automáticamente un esquema para el rango especificado. Si el rango tiene una única celda, Microsoft Excel crea un esquema para toda la hoja.

Group

En un esquema, aumenta el nivel del rango en el esquema. El rango debe ser una fila o una columna entera o un rango de filas o de columnas. Para un rango discontinuo de un informe de tabla dinámica, reagrupa el rango. Para una única celda del rango de datos de un campo de tabla dinámica, realiza un reagrupamiento numérico o cronológico en el campo.

Ungroup

Promueve un rango en un esquema (es decir, reduce su nivel de esquema). El rango especificado debe ser una fila o una columna o bien un rango de filas o columnas. Si el rango se encuentra en un informe de tabla dinámica, el método desagrupará los elementos incluidos en el rango.

g. Métodos que se relacionan con la herramienta de Auditoría

NavigateArrow

Desplaza una flecha de rastreo del rango especificado hacia la celda o las celdas precedentes, dependientes o que provocan un error.

ShowDependents

Muestra las flechas de rastreo que señalan las celdas dependientes directas del rango.

ShowPrecedents

Muestra las flechas de rastreo que señalan las celdas precedentes directas del rango.

ShowErrors

Muestra las flechas de rastreo a través de la estructura en árbol de las celdas precedentes a la celda

que origina el error y devuelve el rango que la contiene.

h. Otros métodos

Calculate

Calcula las fórmulas de todos los libros abiertos.

CalculateRowMajorOrder

Calcula el rango de celdas indicado a partir de su esquina superior izquierda y hasta la esquina inferior derecha en el orden fila-campo.

Dirty

Indica que el rango especificado se recalculará en la próxima actualización de la hoja.

ExportAsFixedFormat

Publica los datos de un rango de valores en formato PDF o XPS.

PrintOut

Imprime el rango de celdas.

Run

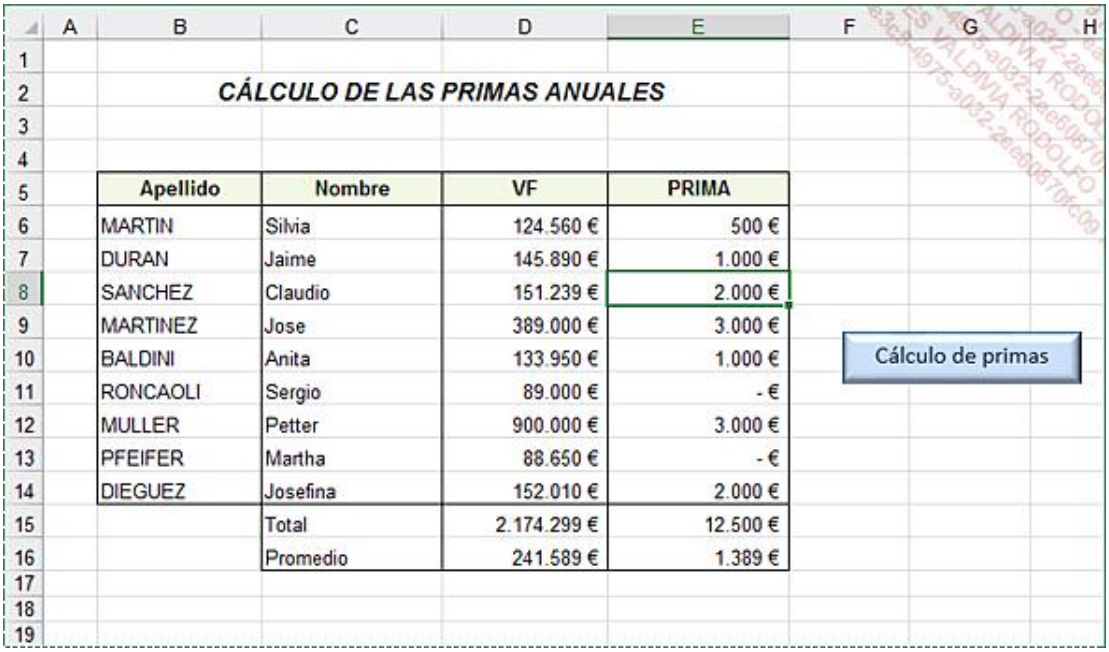
Ejecuta una macro.

Table

Crea una tabla de datos a partir de los valores de entrada y de las fórmulas definidas en una hoja de cálculo.

Ejemplos de uso de los objetos

1. Cálculo del importe de una prima



CÁLCULO DE LAS PRIMAS ANUALES

Apellido	Nombre	VF	PRIMA
MARTIN	Silvia	124.560 €	500 €
DURAN	Jaime	145.890 €	1.000 €
SANCHEZ	Claudio	151.239 €	2.000 €
MARTINEZ	Jose	389.000 €	3.000 €
BALDINI	Anita	133.950 €	1.000 €
RONCAOLI	Sergio	89.000 €	- €
MULLER	Petter	900.000 €	3.000 €
PFEIFER	Martha	88.650 €	- €
DIEGUEZ	Josefina	152.010 €	2.000 €
	Total	2.174.299 €	12.500 €
	Promedio	241.589 €	1.389 €

El rango de celdas "D6:D14" debe recibir el nombre VF.

Cuando el usuario hace clic en el botón de comando **Cálculo de primas**, se ejecutará el procedimiento **Calc_Prima**. Este procedimiento selecciona el rango de celdas llamado VF (celdas "D6:D14") y llama a la función **Prima** para calcular la prima y asignarla a la celda de la derecha.

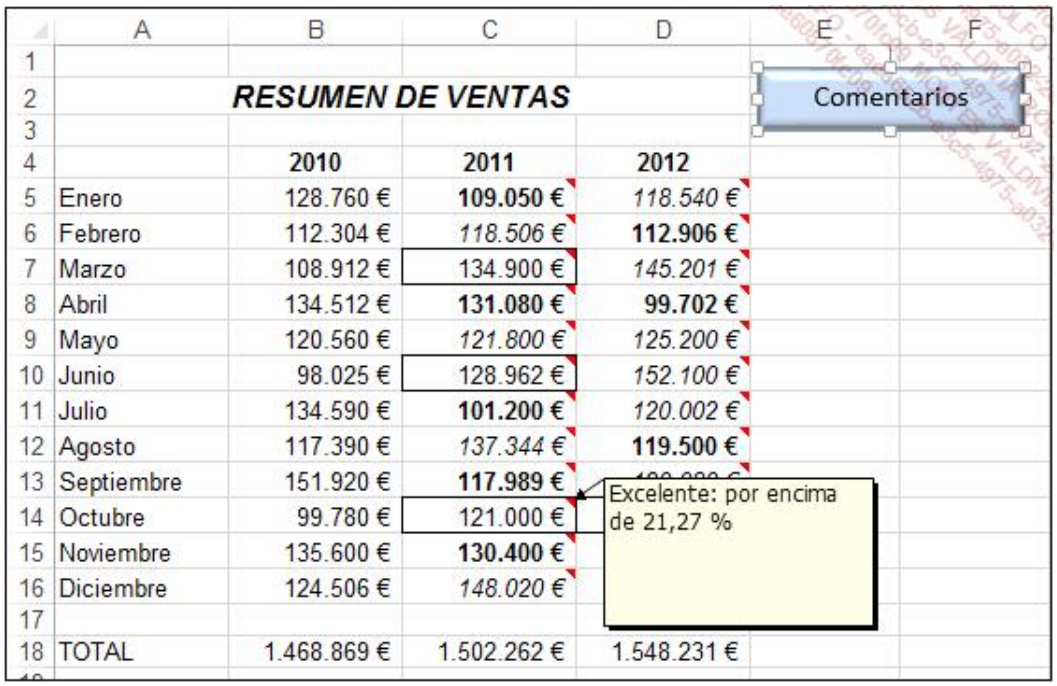
```
Sub Calc_Prima()
   Dim dblVFProm As Double
   Dim ocelda As Range

   '   Selección del rango llamado VF
   ThisWorkbook.Names("VF").RefersToRange.Select
   '   Cálculo del promedio de la selección
   dblVFProm = Evaluate("AVERAGE(VF)")
   '   Recorre las celdas de la selección
   '   La prima calculada se asigna a la celda de la derecha
   For Each ocelda In Selection
       Cells(ocelda.Row, ocelda.Column + 1) = _
       Prima(ocelda.Value, dblVFProm)
   Next ocelda
End Sub
```

La función **Prima** calcula la prima en función del VF (volúmen de facturación) y del promedio de los otros VF.

```
Function Prima(dblVF As Double, dblVFProm As Double) As Double
```

```
   '   Prima en función del importe VF
   Select Case dblVF
      Case Is < 100000
           Prima = 0
      Case Is < 125000
           Prima = 500
      Case Is < 150000
           Prima = 1000
      Case Else
           Prima = 2000
   End Select

   '   Si el VF es superior al promedio
   '   prima extra de 1000 €
   If dblVF > dblVFProm Then
      Prima = Prima + 1000
   End If

End Function
```

2. Asignar comentarios a las celdas



	A	B	C	D
1				
2		**RESUMEN DE VENTAS**		
3				
4		**2010**	**2011**	**2012**
5	Enero	128.760 €	**109.050 €**	*118.540 €*
6	Febrero	112.304 €	*118.506 €*	**112.906 €**
7	Marzo	108.912 €	134.900 €	*145.201 €*
8	Abril	134.512 €	**131.080 €**	**99.702 €**
9	Mayo	120.560 €	*121.800 €*	*125.200 €*
10	Junio	98.025 €	128.962 €	*152.100 €*
11	Julio	134.590 €	**101.200 €**	*120.002 €*
12	Agosto	117.390 €	*137.344 €*	**119.500 €**
13	Septiembre	151.920 €	**117.989 €**	
14	Octubre	99.780 €	121.000 €	
15	Noviembre	135.600 €	**130.400 €**	
16	Diciembre	124.506 €	*148.020 €*	
17				
18	TOTAL	1.468.869 €	1.502.262 €	1.548.231 €

Cuando el usuario hace clic en el botón **Comentarios**, se ejecuta el procedimiento **Mostrar_Comentarios**. Este procedimiento llama al procedimiento **Compara_Valor** para comparar cada una de las celdas seleccionadas con la celda situada a su izquierda.

```
Sub Mostrar_Comentarios()
   Dim oRng1 As Range
   Dim oRng2 As Range
   Dim oRngCurrent As Range
   Dim oCol As Object
```

```
   Dim i As Integer
   Dim j As Integer
   '   Borra los comentarios y los estilos de la selección actual
   Set oRngCurrent = ThisWorkbook.Worksheets("Ventas").Range("C5:D16")
   With oRngCurrent
       .ClearComments
       .Font.Bold = False
       .Font.Italic = False
       .Borders.LineStyle = xlLineStyleNone
   '   Recorre las columnas seleccionadas
   '   Compara el valor de cada celda de la columna
   '   con el de la celda situada a su izquierda
       For i = 1 To .Columns.Count
           Set oCol = .Columns(i)
           For j = 1 To oCol.Cells.Count
              Set oRng1 = oCol.Cells(j)
              Set oRng2 = Cells(oRng1.Row, oRng1.Column - 1)
              Compara_Valor oRng1, oRng2
           Next j
      Next i
   End With

End Sub
```

Al llamar a la función **Compara_Valor**, recibe como argumentos las celdas que hay que comparar. En función del porcentaje de evolución (negativo, < 20 %, > 20 %), se asigna un comentario y un formato a la primera celda.

```
Sub Compara_Valor(oRng1 As Range, oRng2 As Range)
   Dim dbl1, dbl2, dbl3 As Double
   Dim strEvol As String

   '   Compara los valores de dos celdas y asigna un comentario
   With oRng1
      dbl1 = oRng2.Value
      dbl2 = .Value
      dbl3 = (dbl2 - dbl1) / dbl1
      strEvol = Format(Abs(dbl3), "0.00 %")
      Select Case dbl3
         Case Is < 0
             .Font.Bold = True
             .AddComment "Atención: por debajo de " & strEvol
         Case Is < 0.2
             .Font.Italic = True
             .AddComment "Bien: por encima de " & strEvol
         Case Else
             .Borders.LineStyle = xlContinuous
             .AddComment "Excelente: por encima de " & strEvol
      End Select
   End With

End Sub
```

Las tablas dinámicas

La colección **PivotTables** contiene todos los objetos de tabla dinámica (objetos **PivotTable**) que hay en una hoja de cálculo.

1. El objeto PivotTable

Esta sección describe las colecciones, propiedades y métodos más comúnmente utilizados para la creación y modificación de tablas dinámicas.

a. Colecciones

CalculatedFields

Colección de todos los campos calculados de la tabla dinámica especificada.

ColumnFields

Colección de todos los campos de la tabla dinámica especificada que se muestran como campos de columna.

DataFields

Colección de todos los campos de la tabla dinámica especificada que se muestran como campos de datos.

HiddenFields

Colección de todos los campos del origen de datos de la tabla dinámica especificada que no se muestran.

PageFields

Colección de todos los campos de la tabla dinámica especificada que se muestran como campos de página.

RowFields

Colección de todos los campos de la tabla dinámica especificada que se muestran como campos de fila.

PivotFields

Colección de todos los campos del origen de datos de la tabla dinámica especificada, se muestren o no.

VisibleFields

Colección de todos los campos del origen de datos de la tabla dinámica especificada que se muestran.

Todas estas colecciones devuelven objetos **PivotField** que representan un campo de la tabla dinámica.

b. Propiedades

Las propiedades descritas a continuación corresponden a las diferentes opciones de las tablas dinámicas (a las que se puede acceder por la opción del menú contextual **Opciones de tabla dinámica**).

Están clasificadas por pestañas y ordenadas por orden de visualización en la pestaña.

Opciones de la pestaña Diseño y formato

Propiedad	Descripción	Tipo
MergeLabel	Corresponde a la opción **Combinar y centrar celdas con etiquetas**.	Booleano
CompactRowIndent	Número de caracteres de sangría de las etiquetas de fila en forma compacta.	Entero largo
DisplayErrorString	Indique si se muestra una cadena personalizada en las celdas con error.	Booleano
ErrorString	Cadena que se muestra en las celdas que contienen errores cuando la propiedad **DisplayErrorString** tiene el valor True.	Cadena de caracteres
DisplayNullString	Indica si se muestra una cadena personalizada en las celdas que contienen valores nulos.	Booleano
ErrorNull	Cadena que se muestra en las celdas que contienen valores nulos cuando la propiedad **DisplayNullString** tiene el valor True.	Cadena de caracteres
AsAutoFormat	Corresponde a la opción **Autoajustar anchos de columnas al actualizar**.	Booleano
PreserveFormating	Corresponde a la opción **Mantener el formato de la celda al actualizar**.	Booleano

Opciones de la pestaña Totales y filtros

Propiedad	Descripción	Tipo
ColumnGrand	Corresponde a la opción **Mostrar totales generales de las filas**.	Booleano
RowGrand	Corresponde a la opción **Mostrar totales generales de las columnas**.	Booleano
AllowMultipleFilters	Corresponde a la opción **Permitir varios filtros por campo**.	Booleano
SortUsingCustomLists	Corresponde a la opción **Usar listas personalizadas al ordenar**.	Booleano

Opciones de la pestaña Mostrar

Propiedad	Descripción	Tipo
ShowDrillIndicators	Corresponde a la opción **Mostrar botones para expandir y contraer**.	Booleano

DisplayContextTooltips	Corresponde a la opción **Mostrar información contextual sobre herramientas**.	Booleano
DisplayFieldsCaption	Corresponde a la opción **Mostrar títulos de campo y filtrar listas desplegables**.	Booleano
InGridDropZones	Corresponde a la opción **Diseño de tabla dinámica clásica**.	Booleano
ShowValuesRow	Corresponde a la opción **Mostrar la fila de valores**.	Booleano
FieldListSortAscending	Corresponde a la opción **Lista de campos**. Toma el valor True si se selecciona la opción **Ordenar de A a Z**.	Booleano

Opciones de la pestaña Impresión

Propiedad	Descripción	Tipo
PrintDrillIndicators	Corresponde a la opción **Imprimir botones para expandir o contraer**.	Booleano
RepeatItemsOnEachPrintedPage	Corresponde a la opción **Repetir etiquetas de fila en cada página impresa**.	Booleano
PrintTitles	Corresponde a la opción **Imprimir títulos**.	Booleano

Opciones de la pestaña Datos

Propiedad	Descripción	Tipo
SaveData	Corresponde a la opción **Guardar datos de origen con el archivo**.	Booleano
EnabledDrillDown	Corresponde a la opción **Habilitar Mostrar detalles**.	Booleano
RefreshOnFileOpen	Corresponde a la opción **Actualizar al abrir el archivo**.	Booleano

Opciones de la pestaña Texto alternativo

Propiedad	Descripción	Tipo
AlternativeText	Corresponde a la opción **Título**.	Booleano
Summary	Corresponde a la opción **Descripción**.	Booleano

c. Métodos

AddFields

Agrega campos de fila, de columna y de filtro a una tabla dinámica.

ChangePivotCache

Modifica el objeto **PivotCache** (caché de datos) de la tabla dinámica.

ClearAllFilters

Elimina todos los filtros aplicados a la tabla dinámica.

RefreshTable

Actualiza la tabla dinámica a partir del origen de datos.

2. Creación de una tabla dinámica

Para crear una tabla dinámica, en primer lugar hay que definir su caché de datos con el objeto **PivotCache**, y a continuación utilizar el método **CreatePivotTable** del objeto **PivotCache**.

Ejemplo

Este ejemplo permite crear una tabla dinámica a partir de una tabla de celdas llamada "TablaTiempos".

```
Dim oCache As PivotCache
Dim oPivotTable As PivotTable
'   Creación de la caché de la tabla
Set oCache = ThisWorkbook.PivotCaches.Create _
    (SourceType:=xlDatabase, SourceData:="TablaTiempos", _
    Version:=xlPivotTableVersion15)
'   Creación de la tabla dinámica
Set oPivotTable = oCache.CreatePivotTable _
    (TableDestination:=Sheets("TCD").Cells(1, 2), _
    TableName:="TCD1")
```

Los gráficos

Un gráfico está representado por un objeto **Chart**, que está en la colección **Shapes** (objetos **Shape**) que representan las formas u objetos dibujados en una hoja de cálculo. En el caso de un gráfico, el objeto **Shape** representa la zona del gráfico.

1. El objeto Shape

Esta sección describe las propiedades y métodos más comúnmente utilizados para crear y dar formato a las zonas de gráficos.

a. Propiedades

Propiedades	Descripción	Tipo
Chart	Devuelve un objeto **Chart** que representa el gráfico contenido en la forma.	Objeto
Fill	Devuelve un objeto **FillFormat** que representa el formato de relleno de la forma.	Objeto
Line	Devuelve un objeto **LineFormat** que contiene las propiedades de formato del borde de la forma.	Objeto
TopLeftCell	Devuelve un objeto **Range** que representa la celda que está en la esquina superior izquierda de la forma.	Objeto
HasChart	Indica si la forma contiene un gráfico.	Booleano
Height	Altura de la forma en puntos.	Real simple
Left	Distancia, en puntos, entre el borde izquierdo de la forma y el borde izquierdo de la columna A.	Real simple
Top	Distancia, en puntos, entre el borde superior de la forma y el borde superior de la hoja de cálculo.	Real simple
Width	Largo de la forma en puntos.	Real simple

b. Métodos

Copy

Copia la forma en el Portapapeles.

CopyPicture

Copia la forma en el Portapapeles como una imagen.

Delete

Elimina la forma.

2. El objeto Chart

Esta sección describe las propiedades y métodos más comúnmente utilizados para la creación y formato de gráficos.

a. Colecciones

Axes

Colección de todos los ejes del gráfico.

SeriesCollections

Colección de todas las series de datos del gráfico.

b. Propiedades

Propiedad	Descripción	Tipo
ChartArea	Devuelve un objeto **ChartArea** que representa la zona del gráfico.	Objeto
ChartTitle	Devuelve un objeto **ChartTitle** que representa el título del gráfico.	Objeto
PlotArea	Devuelve un objeto **PlotArea** que representa la zona de trazado del gráfico.	Objeto
Legend	Devuelve un objeto **Legend** que representa la leyenda del gráfico.	Objeto
ChartType	Define el tipo de gráfico.	Constante
DisplayBlanksAs	Define la forma como se muestran las celdas vacías en el gráfico.	Constante
HasLegend	Indica si el gráfico tiene leyenda.	Booleano
HasTitle	Indica si el gráfico tiene un título visible.	Booleano

c. Métodos

ApplyChartTemplate

Aplica un tipo de gráfico estándar o personalizado.

ApplyDataLabels

Muestra las etiquetas de datos para todas las series.

Delete

Elimina el gráfico.

Export

Exporta el gráfico a un archivo de tipo imagen.

ExportAsFixedFormat

Exporta el gráfico a un formato especificado (PDF o XPS).

PrintOut

Imprime el gráfico.

PrintPreview

Muestra una vista previa antes de imprimir el gráfico.

SaveChartTemplate

Agrega una plantilla de gráfico personalizada a la lista de plantillas disponibles.

SetSourceData

Define el rango de datos de origen del gráfico.

3. Creación de un gráfico

El método **AddChart** de la colección de objetos **Shapes** permite crear un gráfico. La propiedad **HasChart** del objeto **Shape** indica si hay un gráfico en la forma.

Ejemplo

El siguiente ejemplo permite crear un gráfico de tipo Sectores en 3D y modificar la presentación del gráfico (título y esquinas redondeadas).

```
Dim oSheet As Worksheet
Dim oShape As Shape
Dim oChart As ChartSet
Set oSheet = ThisWorkbook.Sheets("RESUMEN ACTIVIDAD")
Set oShape = oSheet.Shapes.AddChart(Excel.XlChartType.xl3DPie)
If oShape.HasChart Then
   Set oChart = oShape.Chart
   With oChart
     .SetSourceData Source:= _
      oSheet.Range("'RESUMEN ACTIVIDAD'!$B$6:$D$33")
     .ChartTitle.Text = "Resumen del tiempo de trabajo"
     .Parent.RoundedCorners = True
   End With
End If
```

Ejemplo de aplicación

1. Presentación

El siguiente ejemplo permite generar automáticamente tablas y gráficos estadísticos sobre el reparto del tiempo de trabajo de los empleados por día, semana, actividad...

Los datos de origen están situados en la tabla de celdas llamada "TablaTiempos" que puede ver a continuación.

Extracto de los datos origen

ACTIVIDAD	TAREA	FECHA INICIO	FECHA FIN	DURACIÓN	SEMANA	DÍA	ACUMULADO SEMAN	VISIÓN
DESARROLLOS-PROGRAMACIÓN	Hojas de programación	05/01/2015 8:50	05/01/2015 10:11	1:21	S02	05/01/2015	3:39	DIARIO
ADMINISTRACIÓN	Recibos de pago	05/01/2015 10:11	05/01/2015 10:25	0:14	S02	05/01/2015	3:52	DIARIO
DESARROLLOS-PROGRAMACIÓN	Hojas de programación	05/01/2015 10:25	05/01/2015 11:39	1:14	S02	05/01/2015	5:07	DIARIO
PRODUCCIÓN	Seguimiento de Prod. en Taller	05/01/2015 11:39	05/01/2015 12:29	0:50	S02	05/01/2015	5:56	DIARIO
DIVERSOS	Email - Correo	05/01/2015 12:29	05/01/2015 12:44	0:15	S02	05/01/2015	6:12	DIARIO
PRODUCCIÓN	Ejecución de pedidos	05/01/2015 13:36	05/01/2015 14:15	0:39	S02	05/01/2015	7:21	DIARIO
DESARROLLO - INMOBILIARIO	Instalación de red eléctrica	05/01/2015 14:15	05/01/2015 15:43	1:28	S02	05/01/2015	8:49	DIARIO
DESARROLLO - MÁQUINAS	Optimización de máquinas	05/01/2015 15:43	05/01/2015 17:07	1:24	S02	05/01/2015	11:13	DIARIO
DIVERSOS	Email - Correo	06/01/2015 8:11	06/01/2015 8:13	0:02	S02	06/01/2015	12:20	DIARIO
DESARROLLO - MÁQUINAS	Optimización de máquinas	06/01/2015 8:13	06/01/2015 8:25	0:12	S02	06/01/2015	12:32	DIARIO
DIVERSOS	Email - Correo	06/01/2015 8:25	06/01/2015 9:00	0:35	S02	06/01/2015	13:07	DIARIO
DESARROLLO - MÁQUINAS	Optimización de máquinas	06/01/2015 9:00	06/01/2015 10:00	1:00	S02	06/01/2015	14:07	DIARIO
DESARROLLO - INMOBILIARIO	Instalación de red eléctrica	06/01/2015 10:00	06/01/2015 10:50	0:50	S02	06/01/2015	14:57	DIARIO
PRODUCCIÓN	Ejecución de pedidos	06/01/2015 10:50	06/01/2015 12:10	1:20	S02	06/01/2015	16:16	DIARIO

Ejemplo de tabla dinámica y gráfico generado

El Mundo de la Programación en tus Manos...!

DETODOPROGRAMACION.ORG

DETODOPROGRAMACION.ORG

Material para los amantes de la Programación Java, C/C++/C#,Visual.Net, SQL, Python, Javascript, Oracle, Algoritmos, CSS, Desarrollo Web, Joomla, jquery, Ajax y Mucho Mas…

VISITA

www.detodoprogramacion.org
www.detodopython.com
www.gratiscodigo.com

GRATISCODIGO.COM
Usa, lo que ya se creó...

Detodo Python.com

RESUMEN ACTIVIDAD

Pedro MARTÍN
Periodo del 01/01/2015 al 31/01/2015

ACTIVIDAD	TIEMPOS	PORCENTAJE
ADMINISTRACIÓN	27:13	17,33%
COMPRAS	08:27	5,38%
DESARROLLO - INMOBILIARIO	06:05	3,87%
DESARROLLO - MÁQUINAS	22:32	14,34%
DESARROLLO - PROGRAMAS	17:19	11,02%
DESARROLLOS - PROYECTOS	03:21	2,13%
DESARROLLOS-PROGRAMACIÓN	12:42	8,08%
DIVERSOS	17:10	10,93%
PRODUCCIÓN	42:16	26,91%
Total general	**157:05**	**100,00%**

2. Código VBA del ejemplo

El procedimiento **GeneraEstadisticas** permite generar todas las tablas dinámicas y gráficos. Hace una llamada al procedimineto **Creacion_TCD** para la creación de las tablas dinámicas.

```
Option Explicit
 Dim oWbk As Workbook
_____________________________________________________________
Sub GeneraEstadisticas ()
Dim oSheet As Worksheet
Dim oShape As Shape
Dim oChart As Chart
Dim iNumVision As Integer
Dim iNumAct As Integer
```

```
Dim i As Integer
Dim j As Integer
Dim k As Integer

'   Elimina las hojas existentes
Set oWbk = ThisWorkbook
Application.DisplayAlerts = False
For i = oWbk.Sheets.Count To 2 Step -1
    oWbk.Sheets(i).Delete
Next
Application.DisplayAlerts = True

'   Añade las nuevas hojas
For i = oWbk.Sheets.Count + 1 To 5
    oWbk.Sheets.Add after:=oWbk.Sheets(oWbk.Sheets.Count)
Next
oWbk.Sheets(2).Name = "RESUMEN TAREA"
oWbk.Sheets(3).Name = "RESUMEN ACTIVIDAD"
oWbk.Sheets(4).Name = "RESUMEN DIA"
oWbk.Sheets(5).Name = "RESUMEN VISION"

'    Tabla dinámica Resumen Tarea
Creacion_TCD "TCD1", "Resumen Tarea", "Actividad", "Tarea"

'    Tabla dinámica Resumen Actividad
Creacion_TCD "TCD2", "Resumen Actividad", "Actividad", ""

'    Tabla dinámica Resumen Día
Creacion_TCD "TCD3", "Resumen Dia", "Semana", "Día"

'    Tabla dinámica Resumen Visión
Creacion_TCD "TCD4", "Resumen Vision", "Visión", ""

'    Recupera el número de actividades
iNumAct = Sheets("TIEMPOS").Range("NUM_ACTIVIDADES")

'    Gráfico Resumen por Actividad
Set oSheet = oWbk.Sheets("RESUMEN ACTIVIDAD")
Set oShape = oSheet.Shapes.AddChart(Excel.XlChartType.xl3DPie)
If oShape.HasChart Then
   Set oChart = oShape.Chart
   With oChart
     .SetSourceData Source:= _
        oSheet.Range("'RESUMEN ACTIVIDAD'!$B$6:$D$" & iNbAct + 5)
     .ChartTitle.Text = "Resumen del Tiempo de trabajo"
     .Parent.RoundedCorners = True
     .ApplyDataLabels
     .ShowAllFieldButtons = False
   End With
End If

 '   Posición y tamaño
With oShape
     .Left = 50
     .Top = 200
```

```
     .ScaleWidth 1.15, msoFalse, msoScaleFromTopLeft
     .ScaleHeight 1.3, msoFalse, msoScaleFromTopLeft
     .IncrementLeft -400
     .IncrementTop 80
     .ThreeD.RotationY = 30
End With

 ’   Leyenda del gráfico
With oChart.Legend
     .Position = xlBottom
     .Left = 25
     .Top = 200
     .Width = 400
     .Height = 60
End With

  ’   Color de fondo del gráfico
Dim oFill As FillFormat
Set oFill = oShape.Fill
With oFill
     .ForeColor.RGB = RGB(240, 238, 228)
     .Solid
End With

  ’   Etiquetas de datos
With oChart.SeriesCollection(1).DataLabels
     .ShowPercentage = True
     .ShowValue = False
     .Position = 5
     .NumberFormat = "0%"
End With

’   Exporta el gráfico en formato PDF
oChart.ExportAsFixedFormat xlTypePDF, "C:\TABLA\RESUMEN_ACTIVIDAD"

’   Recupera el número de valores para la columna VISION
iNumAct = Sheets("TIEMPOS").Range("NUM_VISIONES")

’   Gráfico Resumen por Vision
Set oSheet = oWbk.Sheets("RESUMEN VISION")
Set oShape = oSheet.Shapes.AddChart(Excel.XlChartType.xl3DPie)
Set oChart = oShape.Chart
With oChart
     .SetSourceData Source:= _
      oSheet.Range("’RESUMEN VISION’!$B$6:$C$" & iNumVision + 5)
     .ChartTitle.Text = "Resumen del Tiempo de trabajo"
     .Parent.RoundedCorners = True
     .ApplyDataLabels
     .Legend.Delete
End With

  ’   Posición y tamaño
With oShape
     .Left = 30
     .Top = 200
```

```
     .ThreeD.FieldOfView = 5
End With

  ’   Color de fondo del gráfico
Set oFill = oShape.Fill
With oFill
    .ForeColor.RGB = RGB(240, 238, 228)
    .Solid
End With

  ’   Etiquetas de datos
With oChart.SeriesCollection(1).DataLabels
     .ShowPercentage = True
     .ShowValue = False
     .ShowCategoryName = True
     .Position = 5
     .NumberFormat = "0%"
End With

’   Exporta el gráfico en formato PDF
oChart.ExportAsFixedFormat xlTypePDF, "C:\TABLA\RESUMEN_Vision"

’    Color para las visiones
Dim lRed As Long, lGreen As Long, lBlue As Long
lRed = 0: lGreen = 0: lBlue = 0

’   Busca el color en el rango VISIONES
Dim sVision As String
Dim rngVision As Range
Set rngVision = Sheets("TIEMPOS").Range("VISIONES")
For j = 1 To iNumVision
    sVision = oSheet.Range("B" & 6 + j).Value & Space(0)
    If sVision <> "" And sVision <> "(vacio)" Then
       For k = 1 To rngVision.Rows.Count
           If rngVision.Cells(k, 1) = sVision Then
              ColorRGB rngVision.Cells(k, 2).Interior.Color, _
                         lRed, lGreen, lBlue
              Exit For
           End If
       Next k

       ’   Color en el gráfico
       With oChart.SeriesCollection(1).Points(j).Format.Fill
            .ForeColor.RGB = RGB(lRed, lGreen, lBlue)
       End With

       ’   Color en la tabla dinámica
       oSheet.Range("B" & 6 + j).Font.Color = _
              RGB(lRed, lGreen, lBlue)
    End If
Next j

’   Posición en la hoja
’   Resumen actividad
oWbk.Sheets("Resumen Actividad").Activate
```

```
Application.ScreenUpdating = True

End Sub
```

El procedimiento **Creacion_TCD** permite generar una tabla dinámica a partir de datos de la hoja "TIEMPOS" con los siguientes parámetros: nombre de la tabla dinámica, nombre de la hoja de destino y nombre de los campos de fila.

```
Private Sub Creacion_TCD(sTCDName As String, sSheetName As _
            String, sField1 As String, sField2 As String)
Dim iFil As Integer
Dim oSheet As Worksheet
Dim oCache As PivotCache

'   Creación de una tabla dinámica
iFil = 6
oWbk.Activate
Set oSheet = oWbk.Sheets("Tiempos")

'   Creación de la caché de la tabla
Set oCache = oWbk.PivotCaches.Create(SourceType:=xlDatabase, _
     SourceData:="TablaTiempos", Version:=xlPivotTableVersion15)

'   Creación de la tabla
oCache.CreatePivotTable _
     TableDestination:=oWbk.Sheets(sSheetName).Cells(iFil, 2), _
     TableName:=sTCDName
Set oSheet = oWbk.Sheets(sSheetName)

'   Filas de la Tabla
With oSheet.PivotTables(sTCDName).PivotFields(sField1)
     .Orientation = xlRowField
     .Position = 1
End With
If sField2 <> "" Then
    With oSheet.PivotTables(sTCDName).PivotFields(sField2)
         .Orientation = xlRowField
         .Position = 2
    End With
End If
oSheet.PivotTables(sTCDName).CompactLayoutRowHeader = _
   UCase(sField1)

'   Elimina los valores vacíos
oSheet.PivotTables(sTCDName).PivotFields(sField1) _
      .PivotFilters.Add Type:=16, Value1:="(vacio)"

'   Tiempos en duración y porcentaje
oSheet.PivotTables(sTCDName).AddDataField _
       oSheet.PivotTables(sTCDName).PivotFields("DURACIÓN"), _
       "TIEMPOS", xlsum
oSheet.PivotTables(sTCDName).AddDataField _
       oSheet.PivotTables(sTCDName).PivotFields("DURACIÓN "), _
       "PORCENTAJE", xlsum
```

```
oSheet.Columns("C").NumberFormat = "[hh]:mm"
With oSheet.PivotTables(sTCDName).PivotFields("PORCENTAJE")
     .Calculation = xlPercentOfTotal
     .NumberFormat = "0.00%"
End With
oSheet.Columns("B:C").EntireColumn.AutoFit

'   Formato de la hoja de cálculo
Formato oSheet

End Sub
```

El procedimiento **Formato** permite dar formato a las diferentes hojas de cálculo, mostrar un título y el nombre del empleado y el periodo.

```
Private Sub Formato(zSheet As Worksheet)

'   Dar formato a una hoja de cálculo
With zSheet

     '    Tamaño de las columnas
     .Columns("A").ColumnWidth = 15
     .Columns("E").ColumnWidth = 12

     '   Título
     .Cells(1, 1).FormulaR1C1 = zSheet.Name
     With .Range("A1:E1").Font
           .Bold = True
           .Name = "Calibri"
           .Size = 16
     End With

     '   Fusión de las celdas del título
     With .Range("A1:E1")
          .HorizontalAlignment = xlCenter
          .VerticalAlignment = xlCenter
          .Merge
      End With

      '   Nombre y periodo
      .Cells(3, 1).FormulaR1C1 = Sheets("TIEMPOS").Range("EMPLEADO")
      .Cells(4, 1).FormulaR1C1 = "Periodo del " _
                  & Sheets("TIEMPOS").Range("FECHA_INICIO") _
                  & " au " & Sheets("TIEMPOS").Range("FECHA_FIN")
      With .Range("A3:G4").Font
           .Bold = True
           .Name = "Calibri"
           .Size = 12
      End With

     '   Fusión de las celdas
      With .Range("A3:E3")
           .HorizontalAlignment = xlCenter
```

```
            .VerticalAlignment = xlCenter
            .Merge
      End With
      With .Range("A4:E4")
            .HorizontalAlignment = xlCenter
            .VerticalAlignment = xlCenter
            .Merge
      End With

End With

End Sub
```

La función **ColorRGB** permite recuperar los valores RGB de un color. En el ejemplo, los colores de los diferentes valores posibles del campo "Vision" se configuran en el rango llamado "VISIONES" de la hoja "TIEMPOS".

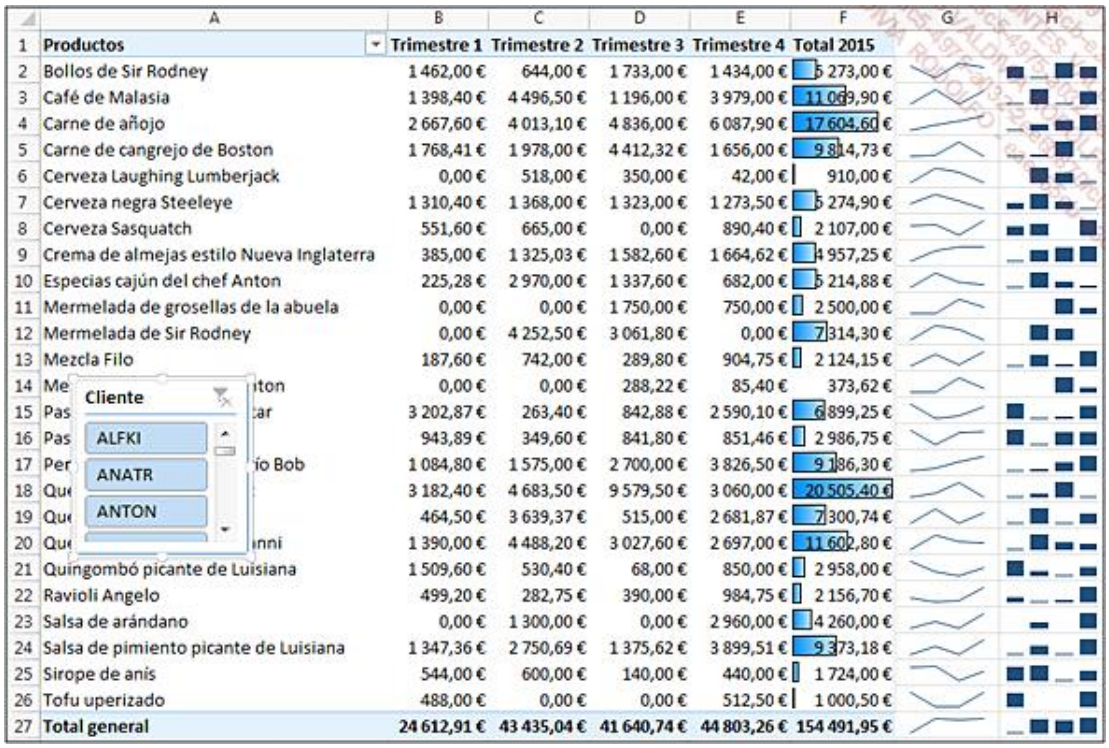
```
Public Function ColorRGB(ByVal lColor As Long, R As Long, _
       G As Long, B As Long)

   '   Devuelve los valores RGB de un color
   R = Int(lColor Mod 256)
   G = Int((lColor Mod 65536) / 256)
   B = Int(lColor / 65536)

End Function
```

Creación de una tabla dinámica con minigráficos

El ejemplo siguiente permite:

- Crear y personalizar una tabla dinámica utilizando las colecciones de objetos **PivotTables** y **PivotFields**.
- Agregar un formato condicional de tipo barra de datos, utilizando la colección de objetos **FormatConditions**.
- Agregar minigráficos utilizando la colección de objetos **SparklineGroups**.
- Agregar un segmento para filtrar los datos de un cliente, utilizando la colección de objetos **SlicerCaches**.

La hoja de cálculo, creada mediante código VBA, se visualiza del siguiente modo:



Productos	Trimestre 1	Trimestre 2	Trimestre 3	Trimestre 4	Total 2015
Bollos de Sir Rodney	1 462,00 €	644,00 €	1 733,00 €	1 434,00 €	5 273,00 €
Café de Malasia	1 398,40 €	4 496,50 €	1 196,00 €	3 979,00 €	11 069,90 €
Carne de añojo	2 667,60 €	4 013,10 €	4 836,00 €	6 087,90 €	17 604,60 €
Carne de cangrejo de Boston	1 768,41 €	1 978,00 €	4 412,32 €	1 656,00 €	9 814,73 €
Cerveza Laughing Lumberjack	0,00 €	518,00 €	350,00 €	42,00 €	910,00 €
Cerveza negra Steeleye	1 310,40 €	1 368,00 €	1 323,00 €	1 273,50 €	5 274,90 €
Cerveza Sasquatch	551,60 €	665,00 €	0,00 €	890,40 €	2 107,00 €
Crema de almejas estilo Nueva Inglaterra	385,00 €	1 325,03 €	1 582,60 €	1 664,62 €	4 957,25 €
Especias cajún del chef Anton	225,28 €	2 970,00 €	1 337,60 €	682,00 €	5 214,88 €
Mermelada de grosellas de la abuela	0,00 €	0,00 €	1 750,00 €	750,00 €	2 500,00 €
Mermelada de Sir Rodney	0,00 €	4 252,50 €	3 061,80 €	0,00 €	7 314,30 €
Mezcla Filo	187,60 €	742,00 €	289,80 €	904,75 €	2 124,15 €
Me	0,00 €	0,00 €	288,22 €	85,40 €	373,62 €
Pas	3 202,87 €	263,40 €	842,88 €	2 590,10 €	6 899,25 €
Pas	943,89 €	349,60 €	841,80 €	851,46 €	2 986,75 €
Per	1 084,80 €	1 575,00 €	2 700,00 €	3 826,50 €	9 186,30 €
Que	3 182,40 €	4 683,50 €	9 579,50 €	3 060,00 €	20 505,40 €
Que	464,50 €	3 639,37 €	515,00 €	2 681,87 €	7 300,74 €
Que	1 390,00 €	4 488,20 €	3 027,60 €	2 697,00 €	11 602,80 €
Quingombó picante de Luisiana	1 509,60 €	530,40 €	68,00 €	850,00 €	2 958,00 €
Ravioli Angelo	499,20 €	282,75 €	390,00 €	984,75 €	2 156,70 €
Salsa de arándano	0,00 €	1 300,00 €	0,00 €	2 960,00 €	4 260,00 €
Salsa de pimiento picante de Luisiana	1 347,36 €	2 750,69 €	1 375,62 €	3 899,51 €	9 373,18 €
Sirope de anís	544,00 €	600,00 €	140,00 €	440,00 €	1 724,00 €
Tofu uperizado	488,00 €	0,00 €	0,00 €	512,50 €	1 000,50 €
Total general	24 612,91 €	43 435,04 €	41 640,74 €	44 803,26 €	154 491,95 €

Cliente
ALFKI
ANATR
ANTON

Para ejecutar este ejemplo, realice las siguientes operaciones:

- Cree un nuevo libro basado en la plantilla **Informe de ventas** (teclee **Informe de ventas** en el cuadro **Buscar plantillas en línea** para acceder a esta plantilla).
- Guarde el libro como **Informe de ventas.xlsm**.
- Asigne el nombre **Ventas_2015** a todas las celdas de la hoja "Datos origen" (rango "A1:F278").
- Inserte el código del siguiente procedimiento en un nuevo módulo.
- Ejecute el código.

```
Sub Tabla_Dinamica()
Dim wSheet As Worksheet
Dim i As Integer
Dim NumLineas As Integer

    '   Selecciona el rango (origen de datos)
    Application.Goto Reference:="Ventas_2015"
```

```
'   Elimina la hoja TD_Productos si existe
For i = 1 To Worksheets.Count
   If Worksheets(i).Name = "TD_Productos" Then
      Application.DisplayAlerts = False
      Worksheets(i).Delete
      Application.DisplayAlerts = True
      Exit For
   End If
Next i

'   Crea la hoja TD_Productos
Set wSheet = Sheets.Add
wSheet.Name = "TD_Productos"

'   Inserta un gráfico en la hoja creada
ActiveWorkbook.PivotCaches.Create _
    (SourceType:=xlDatabase, SourceData:="VENTAS_2015", _
    Version:=xlPivotTableVersion15).CreatePivotTable _
    TableDestination:="TD_PRODUCTOS!R1C1", TableName:= _
    "TD Productos", DefaultVersion:=xlPivotTableVersion15

'   Muestra la lista de campos de la tabla dinámica
ActiveWorkbook.ShowPivotTableFieldList = True

'   Inserta el campo Producto y los 4 trimestres
With wSheet.PivotTables("TD Productos")

    .PivotFields("Producto").Orientation = xlRowField
    .PivotFields("Producto").Position = 1

     '   4 campos Trimestre
     For i = 1 To 4
         .AddDataField ActiveSheet.PivotTables("TD Productos") _
         .PivotFields("Tri " & i), "Trimestre " & i, xlSum
     Next i

     '   Campo Total 2015
     .AddDataField ActiveSheet.PivotTables("TD Productos") _
     .PivotFields("Total"), "Total 2015", xlSum

End With

'   Diseño de la tabla dinámica
With wSheet.PivotTables("TD Productos")

    '   Estilo de la tabla dinámica
    .TableStyle2 = "PivotStyleLight16"

    '   Tamaño de las columnas
    Columns("B:F").Select
    Selection.ColumnWidth = 13

    '   Título de los datos
    .CompactLayoutRowHeader = "Productos"
```

```
    .DataPivotField.Caption = "Ventas"

    '   Formato moneda para los valores numéricos
    For i = 1 To 4
       .PivotFields("Trimestre " & i).NumberFormat = "# ##0.00 €"
    Next i
    .PivotFields("Total 2015").NumberFormat = "# ##0.00 €"

End With

'   Número de filas de la tabla dinámica
Range("B2").Select
Selection.End(xlDown).Select
NumLineas = ActiveCell.Row

'   Formato condicional Barra de datos
'   en la última columna
Range("F2:F" & NumLineas - 1).Select
Selection.FormatConditions.AddDatabar
With Selection.FormatConditions(1)
    .BarColor.Color = 15698432
    .BarColor.TintAndShade = 0
    .BarFillType = xlDataBarFillGradient
    .BarBorder.Type = xlDataBarBorderSolid
End With

'   Añade un minigráfico de tipo líneas
Range("G2:G" & NumLineas).Select
Range("G2:G" & NumLineas).SparklineGroups.Add _
      Type:=xlSparkLine, SourceData:="B2:E" & NumLineas
With Selection.SparklineGroups.Item(1)
      .SeriesColor.ThemeColor = 5
      .SeriesColor.TintAndShade = -0.5
      .Points.Markers.Color.ThemeColor = 5
      .Points.Markers.Color.TintAndShade = 0.5
      .Points.Highpoint.Color.ThemeColor = 5
End With

'   Añade un minigráfico de tipo barras
Range("H2:H" & NumLineas).Select
Range("H2:H" & NumLineas).SparklineGroups.Add _
     Type:=xlSparkColumn, _
     SourceData:="B2:E" & NumLineas
With Selection.SparklineGroups.Item(1)
      .SeriesColor.ThemeColor = 5
      .SeriesColor.TintAndShade = -0.5
      .Points.Markers.Color.ThemeColor = 5
      .Points.Markers.Color.TintAndShade = 0.5
      .Points.Highpoint.Color.ThemeColor = 5
End With

'   Añade un segmento en el campo Cliente
ActiveWorkbook.SlicerCaches.Add _
  (wSheet.PivotTables("TD Productos"), "Cliente") _
  .Slicers.Add ActiveSheet, , "Cliente", "Cliente", _
```

```
      200, 20, 100, 100
    ActiveSheet.Shapes.Range(Array("Cliente")).Select

    '   Oculta la lista de campos de la tabla dinámica
    ActiveWorkbook.ShowPivotTableFieldList = False

End Sub
```

Presentación

El objetivo principal de los cuadros de diálogo es controlar el intercambio de información con el usuario: mostrar mensajes, pedir información, vista y entrada de datos, etc.

Existen tres tipos de cuadros de diálogo:

- Los cuadros de diálogo llamados **cuadros de diálogo integrados**, que permiten, por ejemplo, abrir o guardar un archivo, definir las opciones de Excel, imprimir hojas de cálculo, ordenar datos, etc.
- Los cuadros de diálogo **predefinidos**, que permiten mostrar un mensaje, hacer una pregunta al usuario o invitarlo a introducir una información.
- Los cuadros de diálogo **personalizados** o **formularios**, que permiten mostrar o introducir datos en una interfaz amigable. La creación de formularios personalizados se explica en el capítulo siguiente.

Cuadros de diálogo integrados

1. El objeto Dialog

Los cuadros de diálogo integrados son objetos **Dialog** pertenecientes a la colección **Dialogs** del objeto **Application**.

Sintaxis

Para mostrar un cuadro de diálogo, use el método **Show** según la siguiente sintaxis: `Application.Dialogs(xlDialog).Show`

donde `xlDialog` es una constante de Excel que indica el cuadro de diálogo que hay que mostrar.

Ejemplos de constantes xlDialog

Constante	Cuadro de diálogo
xlDialogBorder	Bordes
xlDialogFontProperties	Fuente
xlDialogDisplay	Opciones de visualización
xlDialogDefineName	Definir un nombre
xlDialogFormulaGoto	Ir a...
xlDialogOpenAbrir	Abrir
xlDialogSaveAs	Guardar como...
xlDialogSort	Ordenar

2. Los métodos GetOpenFileName y GetSaveAsFileName

Los métodos **GetOpenFileName** y **GetSaveAsFileName** del objeto **Application** muestran, respectivamente, los cuadros de diálogo **Abrir...** y **Guardar como...** del menú **Archivo**.

A diferencia de los objetos **Dialogs** correspondientes (constantes xlOpen y xlSaveAs), estos métodos no realizan ninguna acción; solamente permiten recuperar el nombre del archivo introducido o seleccionado por el usuario.

Sintaxis del método GetOpenFileName

```
Application.GetOpenFileName(FileFilter, FilterIndex, Title,
ButtonText, MultiSelect)
```

Todos los argumentos son opcionales.

`FileFilter`	Criterios de filtrado: nombre del filtro seguido de la extensión. Por ejemplo: "PáginaWeb (*.htm;*.html) ,*.htm;*.html".
`FileIndex`	Índice del criterio de filtrado por defecto.
`Title`	Título del cuadro de diálogo.
`ButtonText`	Etiqueta del botón **Abrir** (solamente para Macintosh).
`MultiSelect`	Indica si el usuario puede seleccionar varios archivos.

Sintaxis del método GetSaveAsFileName

```
Application.GetSaveAsFileName(InitialeFile, FileFilter,
FilterIndex, Title, ButtonText)
```

Todos los argumentos son opcionales.

InitialeFile	Nombre del archivo que aparece en la zona de texto **Nombre**. Si se omite este argumento, Excel usa el nombre del libro activo.

Los otros argumentos son los mismos que los argumentos del método **GetOpenFileName**.

Ejemplo

Este ejemplo permite:

- Mostrar el cuadro de diálogo **Abrir** con la posibilidad de seleccionar más de un archivo.
- Guardar en una matriz los nombres de los archivos seleccionados del tipo indicado (extensión xlsm) y que no se encuentren ya abiertos.
- Mostrar un mensaje que indica los archivos que se abrirán.
- Abrir esos archivos tras pedir confirmación.

```
Sub AbreLibros()
   Dim varFiles As Variant
   Dim xlFiles() As Variant
   Dim blnAbierto As Boolean
   Dim strMensaje As String
   Dim objWbk As Workbook
   Dim i As Integer
   Dim j As Integer

   '   Muestra el cuadro de diálogo Abrir
   varFiles = Application.GetOpenFilename _
            (filefilter:="Archivos Excel (*.xlsm),*.xlsm", _
            Title:="Seleccione los archivos que hay que abrir", _
            MultiSelect:=True)
   '   Prueba si los archivos han sido seleccionados
   If TypeName(varFiles) = "Variant()" Then
      ReDim xlFiles(UBound(varFiles))
      For i = 1 To UBound(varFiles)
          '   Controla la extensión del archivo
          If Right(varFiles(i), 4) = "xlsm" Then
             '   Prueba si el archivo ya está abierto
             blnAbierto = False
             For Each objWbk In Workbooks
                If objWbk.Path & "\" & objWbk.Name = varFiles(i) Then
                    blnAbierto = True
                End If
             Next objWbk
             '   Guarda el nombre del archivo en una matriz
             If Not blnAbierto Then
```

```
                j = j + 1
                xlFiles(j) = varFiles(i)
                strMensaje = strMensaje & varFiles(i) & vbCr
             End If
          End If
      Next i

      '   Abre todos los archivos de Excel tras confirmación
      If j > 0 Then
       strMensaje = "Confirme la apertura de los archivos: " _
                    & vbCr & strMensaje
         If MsgBox(strMensaje, vbYesNo + vbQuestion) = vbYes Then
            For i = 1 To j
                Workbooks.Open Filename:=xlFiles(i)
            Next i
         End If
      End If
   Else
      MsgBox "Ningún archivo seleccionado"
   End If
End Sub
```

Cuadros de diálogo predefinidos

1. La función InputBox

Muestra una pregunta (una solicitud de datos) y devuelve el texto escrito por el usuario.

```
InputBox(prompt,title,default,xpos,ypos,helpfile,context)
```

prompt	Cadena que aparecerá como mensaje.
title	Cadena que aparecerá en la barra de título.
default	Valor tomado por defecto.
xpos	Posición horizontal del cuadro de diálogo (expresado en twips).
ypos	Posición vertical del cuadro de diálogo (expresado en twips).
helpfile	Nombre del archivo de ayuda contextual.
context	Número del contexto de ayuda.

Ejemplo

Este ejemplo muestra un cuadro de diálogo que pide el nombre de las celdas que hay que borrar (las celdas reciben el nombre de cada mes).

```
Sub Borrar_Celdas_Nombradas()
   Dim sOpcion As String
   '   Pide introducir el mes que hay que borrar
   '   Si el mes se reconoce, borra las celdas con nombre
   '   Si no, muestra un mensaje de error
   sOpcion = InputBox( _
           Prompt:="¿Qué mes desea borrar?", _
           Title:="Borrar celdas")
   On Error GoTo Err
   Application.Goto reference:=sOpcion
   Selection.Clear
   Exit Sub

Err:
   MsgBox "No se puede borrar, nombre de celda inexistente"
End Sub
```

2. El método InputBox

Actúa como la función `InputBox`, pero permite controlar el tipo de datos que hay que introducir.

```
objeto.InputBox(prompt,title,default,left,top,helpfile,
helpContextID,type)
```

El objeto es obligatorio y debe ser un objeto **Application**.

prompt	Mensaje mostrado.
title	Título del cuadro de diálogo.
default	Valor tomado por defecto.
left	Posición horizontal del cuadro de diálogo (en puntos).
top	Posición vertical del cuadro de diálogo (en puntos).
helpfile	Nombre del archivo de ayuda en línea.
helpContextID	Número del contexto de ayuda.
type	Tipo de datos que se devolverá: 0: Fórmula 1: Número 2: Cadena 4: Valor Booleano 8: Referencia de celda 16: Valor de error 64: Matriz de valores en una selección de celdas.

Para aceptar varios tipos de datos, haga la suma de los valores. Por ejemplo, si se puede aceptar un texto o un número, indique el valor 3 (1 + 2) como tipo.

Ejemplo

Pide al usuario seleccionar la celda o las celdas que hay que pintar.

```
Sub Celdas_A_Pintar()
   Dim strRep as Range
   '   Si el usuario selecciona celdas,
   '   estas se pintan de rojo
   '   Si hace clic en Cancelar, termina el procedimiento
   On Error GoTo Err
   Set strRep = Application.InputBox( _
        Prompt:="Seleccione la celda o las celdas que hay que pintar", _
        Title:="Celda que hay que pintar", Default:="A1",  Type:=8)
       strRep.Interior.ColorIndex = 3

Err:
End Sub
```

3. La función MsgBox

Esta función muestra un mensaje en un cuadro de diálogo. Puede incluir un icono y de uno a tres botones.

Sintaxis de la instrucción

Usada cuando aparece un único botón.

```
MsgBox <message> [, [<type>][, <title>]]
```

Sintaxis de la función

Usada cuando aparece más de un botón. Permite saber qué botón activó el usuario, a través del valor devuelto.

```
MsgBox (<message> , [<buttons>][, <title>]
[,helpfile, context])
```

`message`	Texto del mensaje que aparece en el cuadro de diálogo.
`buttons`	Expresión numérica que representa la suma de los valores que especifican los botones que hay que mostrar, el tipo de icono que hay que usar, la identidad del botón por defecto y la modalidad del cuadro.
`title`	Texto en la barra de título.
`helpfile`	Archivo de ayuda que hay que usar.
`context`	Número del contexto de ayuda.

Valores del argumento Buttons

Constante simbólica	Valor	Significado
Número y tipo de los botones		
vbOKOnly	0	Muestra solamente el botón **Aceptar**.
vbOKCancel	1	Muestra los botones **Aceptar** y **Cancelar**.
vbAbortRetryIgnore	2	Muestra los botones **Anular**, **Reintentar** e **Ignorar**.
vbYesNoCancel	3	Muestra los botones **Sí**, **No** y **Cancelar**.
vbYesNo	4	Muestra los botones **Sí** y **No**.
vbRetryCancel	5	Muestra los botones **Reintentar** y **Cancelar**.
vbMsgBoxHelpButton	16384	Muestra un botón de ayuda.
Tipo de icono		
vbCritical	16	Muestra el icono 
vbQuestion	32	Muestra el icono 
vbExclamation	48	Muestra el icono 
vbInformation	64	Muestra el icono 
Botón por defecto		
vbDefaultButton1	0	Primer botón.
vbDefaultButton2	256	Segundo botón.
vbDefaultButton3	512	Tercer botón.
vbDefaultButton4	768	Cuarto botón.
Modalidad		
vbApplicationModal	0	Aplicación modal. El usuario debe responder al mensaje que aparece en el cuadro de mensajes antes de seguir trabajando en la aplicación actual.
vbSystemModal	4096	Sistema modal.

		Se suspenden todas las aplicaciones hasta que el usuario responda al mensaje que aparece en el cuadro de mensajes.
Presentación		
vbMsgBoxSetForeground	65536	Muestra la ventana del cuadro de mensaje en primer plano.
vbMsgBoxRight	524288	Alinea el texto a la derecha.
vbMsgBoxRtlReading	1048576	Especifica el orden de lectura de derecha a izquierda para los sistemas hebreo y árabe.

Los valores devueltos también se definen mediante constantes:

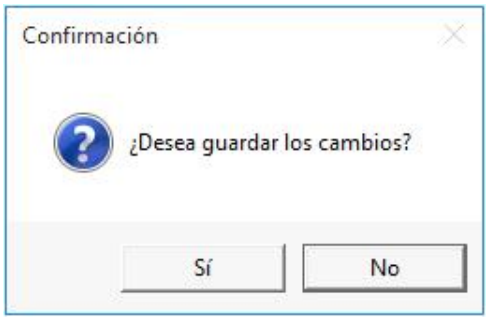
Constante	Valor devuelto	Botón elegido
vbOK	1	**Aceptar**
vbCancel	2	**Cancelar**
vbAbort	3	**Anular**
vbRetry	4	**Reintentar**
vbIgnore	5	**Ignorar**
vbYes	6	**Sí**
vbNO	7	**No**

Ejemplos: uso de la función MsgBox

```
iRep = MsgBox ("Por favor, confirme", 292, _
"Confirmación")
```

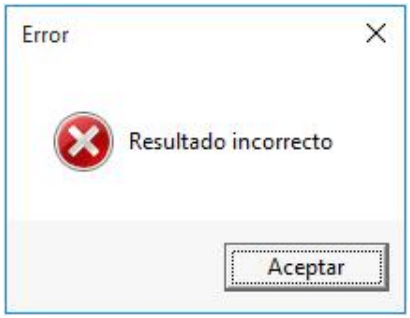
o

```
iRep = MsgBox ("¿Desea guardar los cambios?", _
vbYesNo + vbQuestion + vbDefaultButton2, _
"Confirmación")
```



Uso de la instrucción MsgBox

```
MsgBox "Resultado incorrecto",vbCritical,"Error"
```



Ejemplo

La siguiente macro pide al usuario que seleccione las columnas que se deben eliminar (la selección de columnas se puede hacer a partir de una o más celdas de la columna) y que confirme la eliminación.

```
Sub Confirmacion()
   Dim oCualCol As Range
   Dim vAConfirmar as Variant
   '   Elegir las columnas
   Set oCualCol = Application.InputBox( _
       Prompt:="Seleccione las columnas que desea eliminar", _
       Title:="Elija las columnas", _
       Type:=8)
   Set oCualCol = oCualCol.EntireColumn
   '   Selecciona las columnas y pide confirmación
   oCualCol.Select
   vAConfirmar = MsgBox( _
        Prompt:="¿Confirma la eliminación de las columnas seleccionadas?", _
       Title:="Eliminación de las columnas", _
       Buttons:=vbYesNo + vbExclamation + vbDefaultButton2)
   If vAConfirmar = vbYes Then oCualCol.Delete
End Sub
```

4. Constantes usadas en los cuadros de diálogo

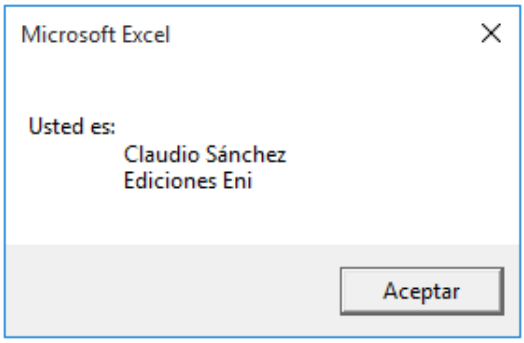
En los mensajes de los cuadros de diálogo, puede usar las siguientes constantes para insertar algunos caracteres especiales.

Carácter que desea insertar	Constante	Equivalente
Retorno de carro y salto de línea	vbCrLf	Chr(13) + Chr(10)
Salto de párrafo	vbCr	Chr(13)
Salto de línea	vbLf	Chr(10)
Carácter nulo	vbNullChar	Chr(0)
Diferente a una cadena de longitud nula	vbNullString	Cadena que tiene el valor cero
Tabulación	vbTab	Chr(9)
Retroceso	vbBack	Chr(8)

Estas constantes pueden usarse en otras instrucciones, además de los cuadros de diálogo.

Ejemplo

Para mostrar este cuadro de diálogo:



se utilizó el siguiente procedimiento:

```
Sub Identificacion()
   MsgBox "Usted es:" & vbCr & vbTab _
            & Application.UserName & vbCr & vbTab _
            & ThisWorkbook.BuiltinDocumentProperties("Company")
End Sub
```

Presentación

Los formularios (también llamados cuadros de diálogo personalizados, formularios personalizados, hojas del usuario o UserForm) permiten disponer de **interfaces de usuario simples y amigables** para la introducción, modificación o visualización de datos.

Los formularios personalizados son cuadros de diálogo sobre los que se puede:

- **Ubicar controles ActiveX**, tales como cuadros de entrada de texto, listas desplegables, botones de comando, etc.
- **Asociar código VBA** para responder a distintos eventos del usuario (clic en un botón de comando, introducción de texto en una zona, selección en una lista desplegable, etc.).

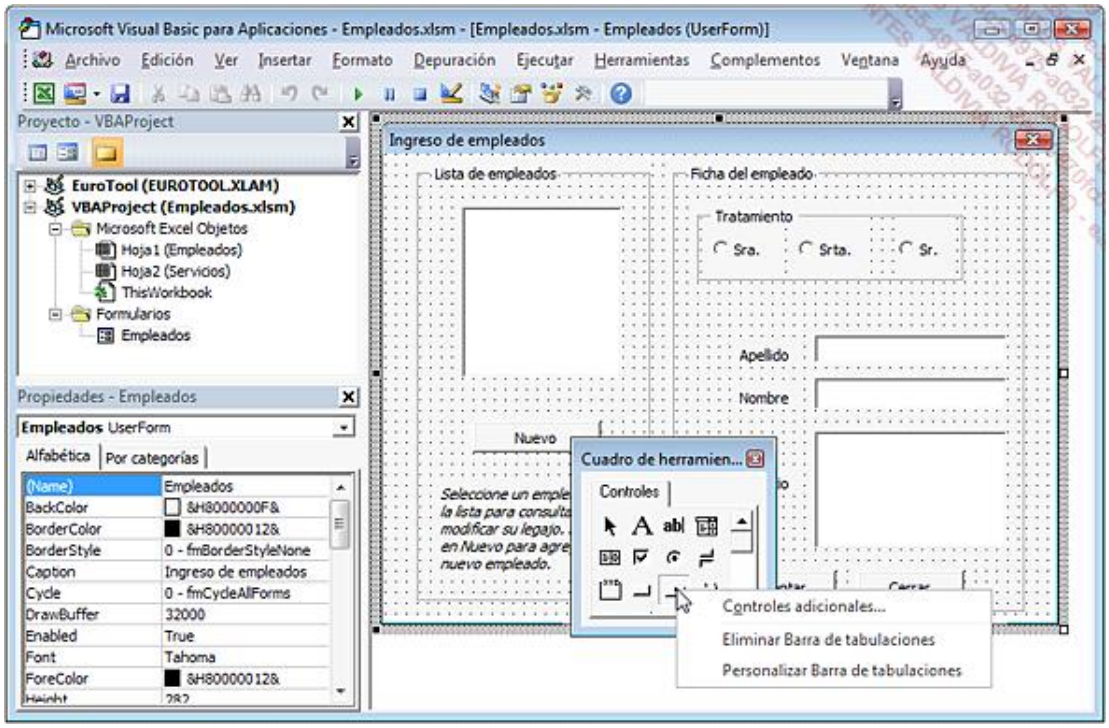
Crear un formulario

Un formulario se crea en una hoja **UserForm**.

- Para insertar una hoja UserForm, acceda a Microsoft Visual Basic y luego seleccione las opciones **Insertar - UserForm**.

  Se agregará una hoja llamada **UserForm n** (por ejemplo: UserForm1), aparecerá un formulario vacío y el cuadro de herramientas.

- Para mostrar la ventana de propiedades, seleccione las opciones **Ver - Ventana Propiedades**, haga clic en el botón  o pulse la tecla [F4].



(Name)

Nombre del formulario.

Caption

Texto en la barra de título.

- Para dimensionar el formulario, selecciónelo y arrastre los controladores de tamaño o indique las propiedades **Height** y **Width** del formulario.

La opción **Controles adicionales** permite agregar otros controles desde la barra de herramientas.

Lista de controles

Herramienta	Nombre	Objeto

	Etiqueta	Label
	Cuadro de texto	TextBox
	Cuadro combinado	ComboBox
	Cuadro de lista	ListBox
	Casilla de verificación	CheckBox
	Botón de opción	OptionButton
	Botón de alternar	ToggleButton
	Marco	Frame
	Botón de comando	CommandButton
	Barra de tabulaciones	TabStrip
	Página múltiple (selección de páginas)	Multipage
	Barra de desplazamiento	ScrollBar
	Botón de número (selección de valores)	SpinButton
	Imagen	Image
	RefEdit (selección de rangos)	RefEdit

Dibujar un control

- Seleccione el control que desea crear y arrastre el puntero para definir un área rectangular.
- Al soltar el botón del ratón, aparecerá el control y la herramienta **Seleccionar objetos**  pasará a ser la herramienta activa.

Para dibujar varios controles del mismo tipo, haga doble clic en la herramienta correspondiente.

Algunas propiedades

(Name)

Especifica el nombre del control.

Caption

Indica el texto de una etiqueta.

ControlTipText

Crea una etiqueta informativa.

Visible

Especifica si un control está oculto o visible.

Enabled

Determina si el foco puede estar sobre el control.

Value

Define el estado o el contenido de un control.

ControlSource

Vincula un control a una celda (cuadro de texto) o un rango de celdas (cuadro de lista).

Determinar el acceso a un control

- Para definir el orden de tabulación, seleccione las opciones:

  Ver - Orden de tabulación

- Para desactivar la posibilidad de usar la tecla [Tab] para acceder a un control, seleccione el control e indique **False** en la propiedad **TabStop**.
- Para asignar una tecla de acceso rápido, seleccione el control e indique la tecla de acceso en la propiedad **Accelerator**.

Si el acceso rápido se aplica a un control **Label**, el control que sigue al **Label** en el orden de tabulación recibirá el foco, y no el control **Label** propiamente dicho.

Tamaño de un control

- Para modificar el tamaño de un control, seleccione los controles y arrastre el controlador de tamaño o seleccione el control e indique las propiedades **Height** y **Width**, que determinan el alto y el ancho del control, en puntos.
- Para uniformizar los tamaños, seleccione los controles que desea dimensionar y vaya al menú **Formato - Igualar tamaño**.

  En función del tamaño deseado, elija **Ancho**, **Alto** o **Ambos**.

- Para ajustar el tamaño, seleccione los controles que desea ajustar y seleccione las opciones **Formato - Ajustar tamaño al contenido** o **Ajustar tamaño a la cuadrícula**.

Ubicar un control

- Para definir la posición de un control, seleccione el control que desea mover y arrastre el ratón o seleccione el control y asigne las propiedades **Left** y **Top** que indican la distancia entre el control y el borde izquierdo y superior del formulario.
- Para alinear controles entre ellos, seleccione los controles que desea alinear y a continuación vaya a la opción **Alinear** del menú **Formato**.

  Según el control de referencia, elija: **Izquierda**, **Centro**, **Derecha**, **Superior**, **Medio**, **Inferior** o **A la cuadrícula**.
- Para administrar el espaciado entre controles, seleccione los controles y vaya a las opciones **Formato - Espacio horizontal** o **Espacio vertical**.

  En función del espacio deseado, seleccione las opciones **Igualar**, **Aumentar**, **Disminuir** o **Quitar**.
- Para centrar un control en el formulario, seleccione el control y a continuación **Formato - Centrar en el formulario Horizontalmente** o **Verticalmente**.

Aplicar formato

Asigne las siguientes propiedades:

Font

Define la tipografía.

BackColor

Especifica el color de fondo.

ForeColor

Especifica el color de primer plano.

BorderColor

Especifica el color del borde.

BorderStyle

Especifica el tipo de borde.

SpecialEffect

Especifica el aspecto del objeto en la pantalla.

Administrar las futuras entradas

PasswordChar

Indica el carácter que hay que mostrar en lugar de los caracteres reales introducidos por el usuario.

MaxLength

Especifica el largo máximo de una entrada.

AutoTab

Fuerza una tabulación automática cuando una entrada alcanza el máximo largo de caracteres permitido.

AutoSize

Redimensiona automáticamente un control para mostrar todo su contenido.

AutoWordSelect

Especifica si una palabra o un carácter es la unidad básica utilizada para extender la selección.

DragBehavior

Indica si el sistema acepta la función arrastrar y soltar.

EnterKeyBehavior

Define el efecto de la tecla [Intro].

HideSelection

Indica si el texto seleccionado permanece resaltado cuando un control pierde el foco.

IntegralHeight

Indica si el control muestra las líneas completas de texto en una lista o líneas parciales en el sentido vertical.

Locked

Indica si se puede modificar un control.

MultiLine

Define si un control puede aceptar y mostrar varias líneas de texto.

SelectionMargin

Especifica si el usuario puede seleccionar una línea de texto al hacer clic a la izquierda del texto.

TabKeyBehavior

Determina si se permiten las tabulaciones en la zona de edición.

TextAlign

Indica el tipo de alineación del texto en un control.

WordWrap

Indica si se agrega automáticamente un retorno de carro al contenido de un control al final de una línea.

Resumen de propiedades por objeto

	Check Box	Combo Box	Command Button	Frame	Image	Label	ListBox	Multi Page
(Name)	X	X	X	X	X	X	X	X
Accelerator	X		X			X		
AutoSize	X	X	X		X	X		
AutoTab		X						
AutoWordSelect		X						
BackColor	X	X	X	X	X	X	X	X
BorderColor		X		X	X	X	X	
BorderStyle		X		X	X	X	X	
Caption	X		X	X		X		
ControlSource	X	X					X	
ControlTipText	X	X	X		X	X	X	X
DragBehavior		X						
Enabled	X	X	X	X	X	X	X	X
EnterKeyBehavior								
Font	X	X	X	X		X	X	X
ForeColor	X	X	X	X		X	X	X
Height	X	X	X		X	X	X	X
HideSelection		X						
IntegralHeight							X	
Left	X	X	X		X	X	X	X
Locked	X	X	X				X	
MaxLength		X						
MultiLine								
PassWordChar								
SelectionMargin		X						
SpecialEffect	X	X		X	X	X	X	
TabKeyBehavior								
TabStop	X	X	X	X			X	X
TextAlign		X					X	
Top	X	X	X		X	X	X	X
Value	X	X	X				X	X
Visible	X	X	X	X	X	X	X	X
Width	X	X	X		X	X	X	X

WordWrap	X		X			X		

	Option Button	Scroll Bar	Spin Button	TabStrip	TextBox	Toggle Button	RefEdit
(Name)	X	X	X	X	X	X	X
Accelerator	X					X	
AutoSize	X				X	X	X
AutoTab					X		X
AutoWordSelect					X		X
BackColor	X	X	X	X	X	X	X
BorderColor					X		X
BorderStyle					X		X
Caption	X				X		
ControlSource	X	X	X		X	X	
ControlTipText	X	X	X	X	X	X	X
DragBehavior					X		X
Enabled	X	X	X	X	X	X	X
EnterKeyBehavior					X		X
Font	X			X	X	X	X
ForeColor	X	X	X	X	X	X	X
Height	X	X	X	X	X	X	X
HideSelection					X		X
IntegralHeight					X		X
Left	X	X	X	X	X	X	X
Locked	X				X	X	X
MaxLength					X		X
MultiLine					X		X
PassWordChar					X		X
SelectionMargin					X		X
SpecialEffect	X				X	X	X
TabKeyBehavior					X		X
TabStop	X	X	X	X	X	X	X
TextAlign					X		X
Top	X	X	X	X	X	X	X
Value	X	X	X	X	X	X	X
Visible	X	X	X	X	X	X	X
Width	X	X	X	X	X	X	X
WordWrap	X				X	X	X

Personalizar un formulario

1. Escribir procedimientos

- Para mostrar la ventana de código de un control:

  haga doble clic en el control para el que desee asignar un código, o seleccione el control y seleccione las opciones:

  Ver - Código o [F7]

- Para insertar un nuevo evento, abra la lista de la derecha y seleccione el evento deseado.

  Si no indica un evento, el evento sugerido para la mayoría de los controles es el evento **Click**.

- Para volver a mostrar un control, seleccione las opciones:

  Ver - Objeto o [Mayús][F7]

2. Lista de eventos asociados a los principales controles

Activate

Ocurre cuando se activa la hoja.

AddControl

Ocurre cuando se inserta un control en una hoja.

AfterUpDate

Ocurre tras modificar datos.

BeforeDragOver

Ocurre cuando se está ejecutando una operación de arrastrar y soltar.

BeforeDropOrPaste

Ocurre cuando el usuario está a punto de colocar o pegar datos en un objeto.

BeforeUpDate

Ocurre antes de la modificación de datos.

Change

Ocurre cuando se modifica la propiedad **Value**.

Click

Ocurre cuando el usuario hace clic en un control o cuando selecciona definitivamente un valor para un control con más de un valor posible.

DblClick

Ocurre cuando el usuario hace doble clic.

DeActivate

Ocurre cuando la hoja deja de ser la ventana activa.

DropButtonClick

Ocurre cada vez que se muestra o se oculta una lista desplegable.

Enter

Ocurre antes de que un control reciba realmente el foco desde un control de la misma hoja.

Error

Ocurre cuando un control detecta un error y no puede devolver información del error al programa que lo ha llamado.

Exit

Ocurre inmediatamente antes de que un control pierda el foco en favor de otro control de la misma hoja.

Initialize

Ocurre después de que se carga un objeto, pero antes de que se muestre.

KeyDown

Ocurre cuando el usuario pulsa una tecla.

KeyPress

Ocurre cuando el usuario pulsa una tecla ANSI.

KeyUp

Ocurre cuando el usuario suelta una tecla.

Layout

Ocurre cuando cambia el tamaño de un control.

MouseDown

Ocurre cuando el usuario pulsa el botón del ratón.

MouseMove

Ocurre cuando el usuario mueve el ratón.

MouseUp

Ocurre cuando el usuario suelta el botón del ratón.

QueryClose

Se produce antes de cerrar la hoja.

RemoveControl

Ocurre cuando se elimina el control del contenedor.

Resize

Se produce cuando cambia el tamaño de la hoja.

Scroll

Ocurre cuando se vuelve a reubicar un cuadro de desplazamiento.

SpinDown

Ocurre cuando el usuario hace clic en la flecha inferior o izquierda del contador.

SpinUp

Ocurre cuando el usuario hace clic en la flecha superior o derecha del contador.

Terminate

Ocurre tras la descarga de la hoja.

Zoom

Ocurre cuando cambia el valor de la propiedad **Zoom**.

Resumen de eventos por control

	Check Box	Combo Box	Command Button	Frame	Image	Label	List Box	Multi page
Activate								
AddControl				X				X
AfterUpdate	X	X					X	
BeforeDragOver	X	X	X	X	X	X	X	X

BeforeDropOrPaste	X	X	X	X	X	X	X	X
BeforeUpdate	X	X					X	
Change	X	X					X	X
Click	X	X	X	X	X	X	X	X
DblClick	X	X	X	X	X	X	X	X
Deactivate								
DropButtonClick		X						
Enter	X	X	X	X			X	X
Error	X	X	X	X	X	X	X	X
Exit	X	X	X	X			X	X
Initialize								
KeyDown	X	X	X	X			X	X
KeyPress	X	X	X	X			X	X
KeyUp	X	X	X	X			X	X
Layout				X				X
MouseDown	X	X	X	X	X	X	X	X
MouseMove	X	X	X	X	X	X	X	X
MouseUp	X	X	X	X	X	X	X	X
RemoveControl				X				X
Terminate								
Scroll				X				X
SpinDown								
SpinUp								
Zoom				X				X
QueryCLose								
Resize								

	Option Button	**Scroll Bar**	**Spin Button**	**Tab Strip**	**Text Box**	**Toggle Button**	**User Form**	**Ref Edit**
Activate							X	
AddControl							X	
AfterUpdate	X	X	X		X	X	X	X
BeforeDragOver	X	X	X	X	X	X	X	X
BeforeDropOrPaste	X	X	X	X	X	X	X	X
BeforeUpdate	X	X	X		X	X		X
Change	X	X	X	X	X	X		X
Click	X			X		X	X	
DblClick	X			X	X	X	X	X
Deactivate							X	
DropButtonClick					X			X
Enter	X	X	X	X	X	X		X

Error	X	X	X	X	X	X	X	X
Exit	X	X	X	X	X	X		X
Initialize							X	X
KeyDown	X	X	X	X	X	X	X	
KeyPress	X	X	X	X	X	X	X	X
KeyUp	X	X	X	X	X	X	X	X
Layout							X	X
MouseDown	X			X	X	X	X	
MouseMove	X			X	X	X	X	X
MouseUp	X			X	X	X	X	X
RemoveControl							X	X
Terminate							X	
Scroll		X					X	
SpinDown			X					
SpinUp			X					
Zoom							X	
QueryCLose							X	
Resize							X	

Cancelar un evento

En ciertos casos, resulta necesario poder cancelar un evento.

Para ello, se debe asignar el valor **True** al argumento **Cancel** del procedimiento asociado a un evento.

Ejemplo

Si la fecha introducida es incorrecta, se cancela el evento Exit: el cursor queda situado en la zona de texto.

```
Private Sub txtFechaFin_Exit(ByVal Cancel As _
MSForms.ReturnBoolean)

If IsNull(txtFechaFin) Then Exit Sub
'  La fecha debe ser correcta
If Not IsDate(txtFechaFin) Then
   MsgBox "Fecha incorrecta", vbCritical
   Cancel = True
   Exit Sub
End If
 '  La fecha de fin debe ser >= fecha de inicio
If DateValue(txtFechaFin) < DateValue(txtFechaIni) Then
   MsgBox "La fecha de fin es anterior a la fecha de inicio", vbCritical
   Cancel = True
   Exit Sub
End If
End Sub
```

Solamente los eventos BeforeDragOver, BeforeDropOrPaste, BeforeUpdate, DblClick, Exit, Error y QueryClose tienen un argumento Cancel. Los demás eventos no pueden cancelarse.

3. Ejecutar y cerrar un formulario

- Para ejecutar un formulario desde de la hoja UserForm, seleccione las opciones:

  Ejecutar - Ejecutar Sub/UserForm o  o [F5]

- Para ejecutar un formulario desde un módulo, use el método **Show** o la instrucción **Load**.

Show (método)

Sintaxis

```
ObjetoUserForm.Show
```

Muestra el objeto **UserForm** indicado.

Load (instrucción)

Sintaxis

```
Load ObjetoUserForm
```

Carga el objeto sin mostrarlo.

- Para cerrar un formulario, use el método **Hide** o la instrucción **Unload**.

Hide (método)

Sintaxis

```
ObjetoUserForm.Hide
```

Oculta el formulario sin descargarlo.

Unload (instrucción)

Sintaxis

```
Unload ObjetoUserForm
```

Elimina el formulario de la memoria.

Eventos invocados

Los métodos e instrucciones de ejecución y de cierre de formularios desencadenan los siguientes eventos:

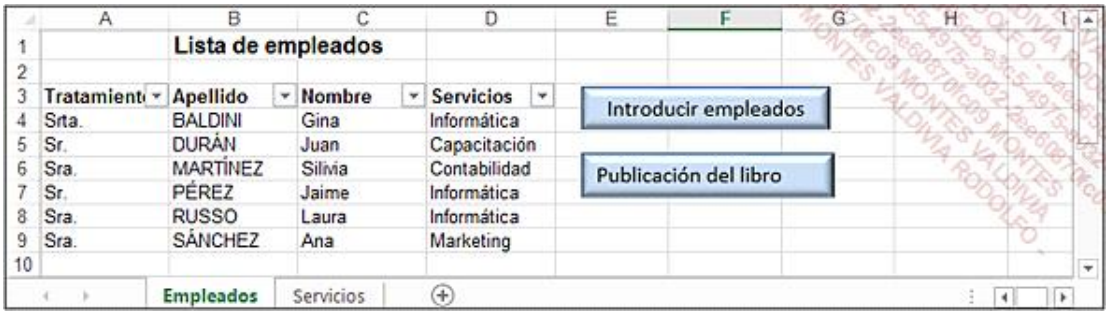
Método o instrucción	Eventos
Show	Initialize
	Activate
Load	Initialize
Hide	Sin evento
Unload	QueryClose
	Terminate

Ejemplo de formulario personalizado

1. Presentación

Este ejemplo muestra cómo crear un formulario personalizado para introducir o modificar las pestañas "empleados". El libro Empleados.xlsm contiene dos hojas de cálculo y un formulario.

La hoja "Empleados" contiene la lista de empleados:

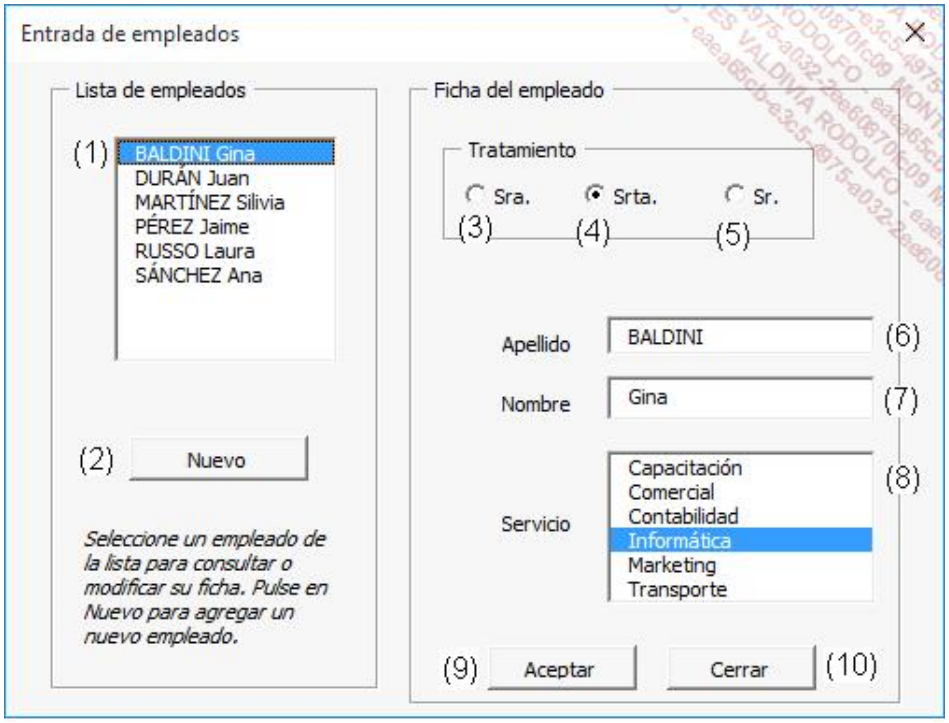


La hoja "Servicios" contiene la lista de servicios:

El formulario "Empleados" permite:

- Modificar la información de un empleado y actualizar la hoja de Excel Empleados.
- Crear un nuevo empleado y agregarlo a la lista de la hoja de Excel Empleados.

La lista de servicios se lee desde la hoja Servicios.



Lista de controles del formulario Empleados:

N.°	Tipo de control	Nombre
1	Cuadro de lista	lstEmpleados
2	Botón de comando	cmdNuevo
3	Botón de opción	optSra
4	Botón de opción	optSrta
5	Botón de opción	optSr
6	Cuadro de texto	txtApellido
7	Cuadro de texto	txtNombre
8	Cuadro de lista	lstServicios
9	Botón de comando	cmdAceptar
10	Botón de comando	cmdCerrar

2. Código asociado al botón macro de la pestaña Empleados

Este código está contenido en el módulo de clase ThisWorkbook.

```
Sub Mostrar_Formulario()
'   Muestra el formulario Empleados
Empleados.Show
End Sub
```

3. Código VBA asociado al formulario

```
Option Explicit
' Nombre de la aplicación
Const strAppName = "Entrada de empleados"
Dim bNuevo As Boolean
_____________________________________________________________________
Private Sub UserForm_Initialize()
   Dim oRng As Range
   Dim oCell As Range
   ' Muestra la lista de servicios
   With ThisWorkbook.Worksheets("Servicios")
        .Activate
         Set oRng = .Range("A1").CurrentRegion
        ' Ordena los servicios por orden alfabético
         oRng.Sort Key1:  =Range("A1")
         lstServicios.Clear
         For Each oCell In oRng
             If oCell.Text <> "" Then
                lstServicios.AddItem oCell.Text
             Else
                Exit For
             End If
         Next oCell
   End With
   ' Muestra la lista de empleados
   Mostrar_Empleados
   lstEmpleados.ListIndex = 0
   ' Nuevo empleado por defecto
   bNuevo = True
End Sub
_____________________________________________________________________
Private Sub Mostrar_Empleados()
   Dim oRng As Range
   Dim oLinea As Range
   ' Muestra la lista de empleados
   With ThisWorkbook.Worksheets("Empleados")
        .Activate
        Set oRng = .Range("A3").CurrentRegion
        Set oRng = .Range("A4:  D" & oRng.Rows.Count + 3)
        lstEmpleados.Clear
        For Each oLineaIn oRng.Rows
            If Cells(oLinea.row, 2) <> "" Then
               lstEmpleados.AddItem Cells(oLinea.row, 2) & " " &
               Cells(oLinea.row, 3)
            Else
               Exit For
            End If
        Next oLinea
   End With
End Sub
_____________________________________________________________________
Private Sub cmdAceptar_Click()
   Dim oRng As Range
   Dim i As Integer
    ' Control de datos introducidos
```

```
   If txtNombre = "" Or txtApellido = "" _
      Or IsNull(lstServicios) Then
      MsgBox "Nombre, apellido y servicio obligatorio", _
             vbExclamation, strAppName
      txtNombre.SetFocus
      Exit Sub
   End If
   If Not (optSr Or optSra Or optSrta) Then
      MsgBox "Tratamiento obligatorio", vbExclamation, strAppName
      optSr.SetFocus
      Exit Sub
   End If
   With ThisWorkbook.Worksheets("Empleados")
       ’  Agrega el empleado en la primera fila vacía
       If bNuevo Then
          Set oRng = .Range("A3").CurrentRegion
          i = oRng.Rows.Count + 3
       Else
       ’  Modifica el empleado seleccionado
          i = lstEmpleados.ListIndex + 4
       End If
       If optSr Then
          .Cells(i, 1) = "Sr."
       ElseIf optSra Then
           .Cells(i, 1) = "Sra."
       Else
           .Cells(i, 1) = "Srta."
       End If
       .Cells(i, 3) = Empleados.txtNombre
       .Cells(i, 2) = Empleados.txtApellido
       .Cells(i, 4) = Empleados.lstServicios
       ’   Ordena los empleados
       Ordenar_Empleados
   End With
   ’   Muestra la lista de empleados
      If bNuevo Then Mostrar_Empleados
      Inicializa_Empleados
End Sub
_____________________________________________________________
Private Sub cmdCerrar_Click()
   ’   Pide confirmación y cierra el formulario
   If MsgBox("¿Desea terminar la inserción de datos?", _
      vbQuestion + vbYesNo, strAppName) = vbYes Then
   Unload Me
   End If
End Sub
_____________________________________________________________
Private Sub cmdNuevo_Click()
   ’   Inicializa la pestaña Empleados
   Inicializa_Empleados
   bNuevo = True
End Sub
_____________________________________________________________
Private Sub Ordenar_Empleados()
   Dim oRng As Range
```

```
   '   Ordena la lista de empleados por apellido y nombre
   Set oRng = Worksheets("Empleados").Range("A3").CurrentRegion
   oRng.Sort Key1:=Range("B3"), Order1:=xlAscending, _
             Key2:=Range("C3"), Order2:=xlAscending, _
             Header:=xlYes
End Sub
_____________________________________________________________
Private Sub Inicializa_Empleados()
   Dim i As Integer
   '   Inicializa el formulario para la próxima inserción de datos
    With Empleados
        txtNombre = ""
        txtApellido = ""
        optSr = False
        optSra = False
        optSrta = False
        For i = 0 To lstServicios.ListCount - 1
            lstServicios.Selected(i) = False
        Next i
   End With
End Sub
_____________________________________________________________
Private Sub lstEmpleados_Click()
   Dim i As Integer
   Dim j As Integer
   '   Muestra el empleado seleccionado
   bNuevo = False
   i = lstEmpleados.ListIndex + 4
   With ThisWorkbook.Worksheets("Empleados")
       Select Case .Cells(i, 1)
           Case "Sr.":   optSr = True
           Case "Sra.":   optSra = True
           Case "Srta.":   optSrta = True
       End Select
       Empleados.txtNombre = .Cells(i, 3)
       Empleados.txtApellido = .Cells(i, 2)
       For j = 0 To Empleados.lstServicios.ListCount - 1
           If Empleados.lstServicios.List(j) = .Cells(i, 4) Then
              Empleados.lstServicios.ListIndex = j
           End If
       Next j
   End With
End Sub
```

Presentación

La versión 2016 de Microsoft Office permite personalizar la cinta de opciones agregando, relocalizando o eliminando pestañas y comandos (ver el capítulo Presentación). Esta innovación no permite tener cintas específicas para un libro determinado.

Sin embargo, existen dos soluciones para tener una cinta personalizada propia de un determinado libro:

- La utilidad **Custom UI Editor**, que permite crear una cinta personalizada por medio de código XML e interactuar con la cinta a través de procedimientos VBA.
- La colección **CommandBars**, que permite crear barras de herramientas y menús en la pestaña **Complementos**.

Personalización de la cinta con la utilidad Custom UI Editor

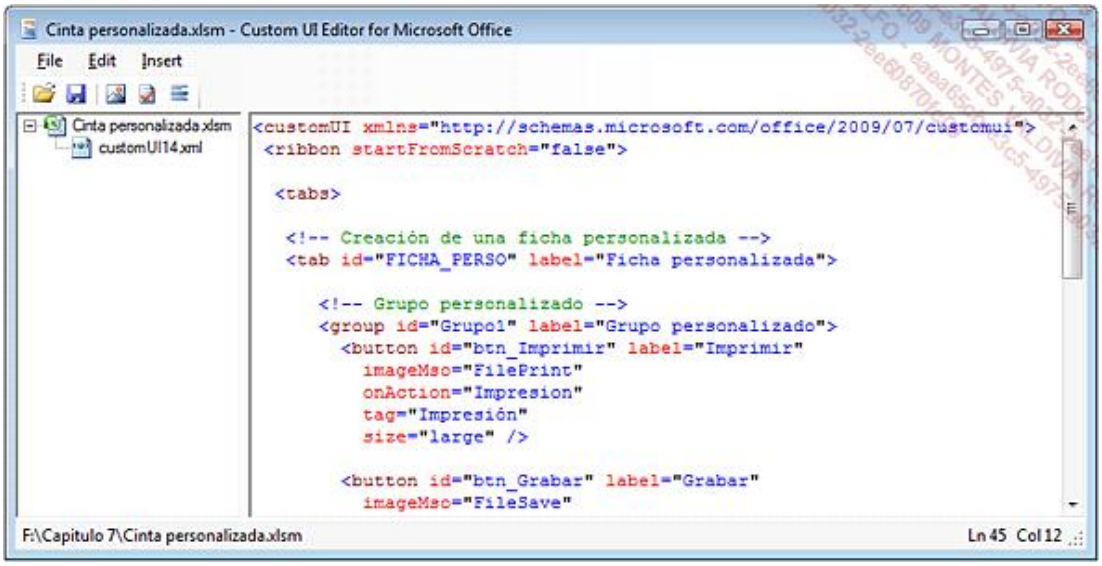
1. Presentación de la utilidad Custom UI Editor

La utilidad **Custom UI Editor** permite personalizar íntegramente la cinta de opciones asociada a un libro, por medio de código XML. Esta utilidad se puede descargar desde el sitio oficial de Microsoft. También está disponible con los ejemplos de este capítulo.

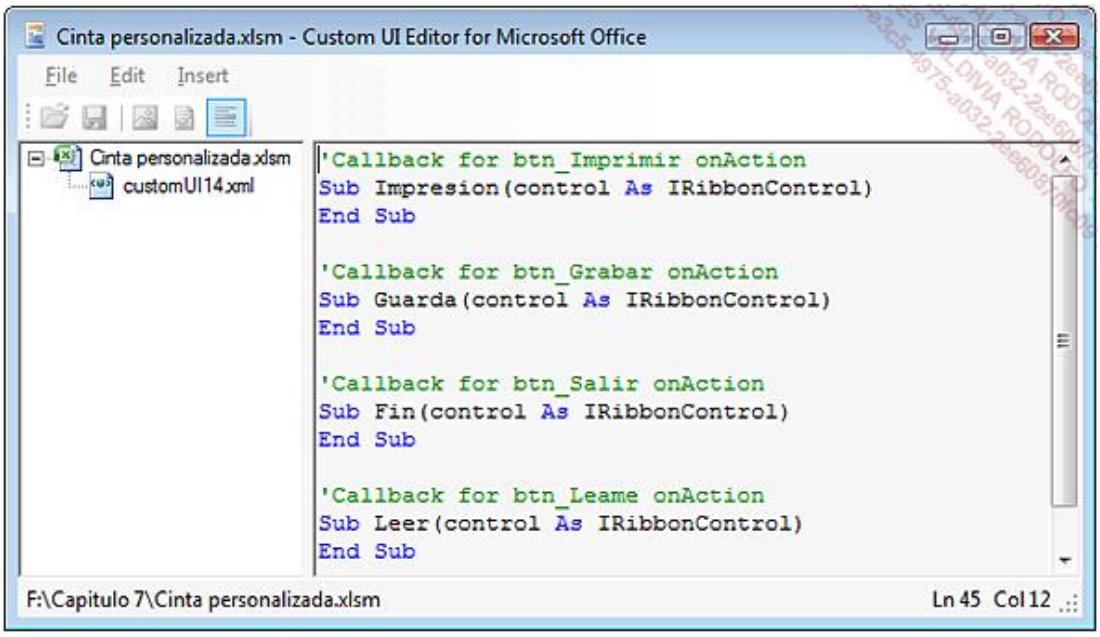
Para usar la utilidad **Custom UI Editor**, se debe disponer del paquete de redistribución .NET Framework 2. Si este no es el caso, un vínculo de descarga se lo propondrá al instalar la utilidad.

Una vez instalada la utilidad, podrá personalizar la cinta de opciones de un libro de la siguiente manera:

- Cree un nuevo libro en Excel 2016 y guárdelo en formato xlsm (libro habilitado para macros).
- Cierre el libro: si está abierto, el código XML no podrá grabarse desde la utilidad de personalización.
- Ejecute la herramienta **Custom UI Editor**.
- Abra el libro con la opción **Archivo** y luego **Abrir** o con el botón de comando  .
- Escriba el código XML de personalización de la interfaz.



- Haga clic en el botón **Validate**  para verificar la sintaxis del código.
- Haga clic en el botón **Generate CallBacks**  para generar el código VBA de los procedimientos asociados a las funciones personalizadas **onAction**.



- Copie el código en el Portapapeles: podrá luego pegarlo en un módulo asociado a su libro e incluir su propio código VBA en los distintos procedimientos.
- Guarde el libro con la opción **Save** de la pestaña **File** o con el botón de comando .
- Cierre la herramienta con la opción **Exit** de la pestaña **File**.
- Abra su libro desde Excel: aparecerá la nueva cinta de opciones.
- Usted deberá incluir los procedimientos **onAction** en un módulo VBA y personalizar el código asociado a los diferentes botones de comando.

2. Ejemplo de código XML de personalización

El siguiente código XML permite crear una pestaña personalizada con dos grupos que incluyen botones de comando de diferentes tamaños.

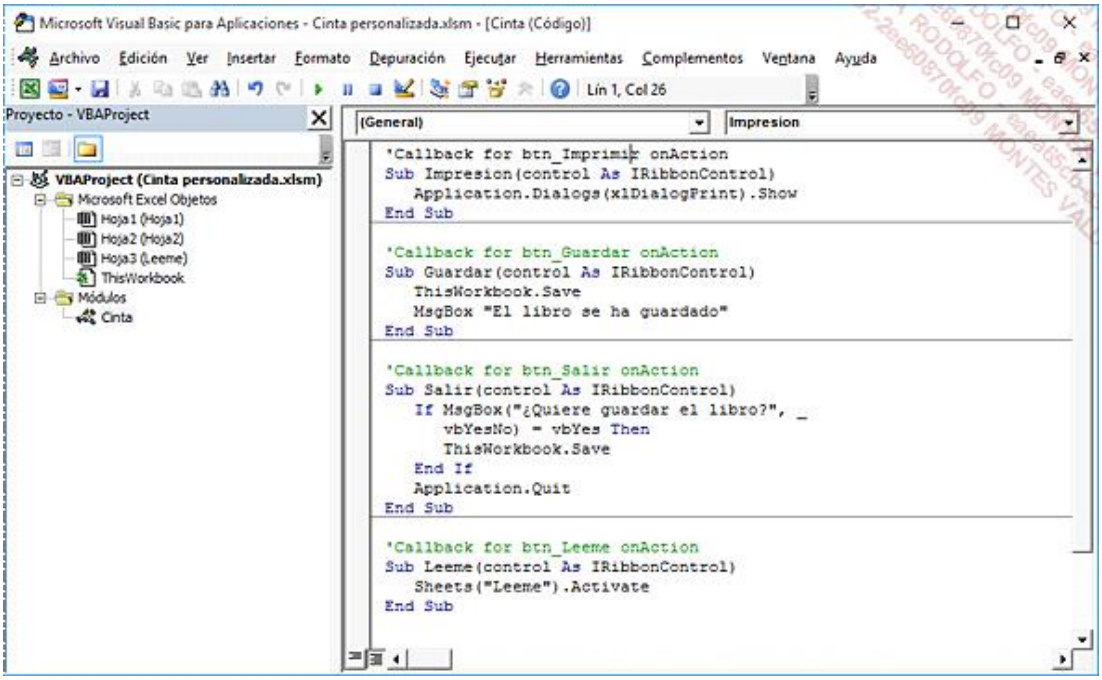

```
<customUI
  xmlns="http://schemas.microsoft.com/office/2009/07/customui">
 <ribbon startFromScratch="false">

  <tabs>

   <!-- Creación de una pestaña personalizada -->
   <tab id="PESTAÑA_PERSO" label="Pestaña personalizada">

      <!-- Grupo personalizado -->
      <group id="Grupo1" label="Grupo personalizado">
        <button id="btn_Imprimir" label="Imprimir"
```

```
          imageMso="FilePrint"
          onAction="Impresion"
          tag="Impresión"
          size="large" />

        <button id="btn_Grabar" label="Grabar"
          imageMso="FileSave"
          onAction="Guarda"
          tag="Grabar"
          size="normal" />

        <button id="btn_Salir" label="Salir"
          imageMso="PrintPreviewClose"
          onAction="Fin"
          tag="Salir"
          size="normal" />

      </group>

      <!-- Grupo Documentación -->
      <group id="Grupo2" label="Documentación">

        <button id="btn_Leame" label="Leame"
          imageMso="FunctionsLookupReferenceInsertGallery"
          onAction="Leer"
          tag="Leame"
          size="large" />
      </group>


   </tab>
  </tabs>
 </ribbon>
</customUI>
```

Para realizar este ejemplo:

- Cree el libro **Cinta personalizada.xlsm** desde Excel 2016. Guárdelo y ciérrelo.
- Abra ese libro desde la utilidad **Custom UI Editor** y escriba el código XML del ejemplo.
- Copie en el portapapeles los procedimientos VBA generados usando el botón **Generate CallBacks**  .
- Guarde el código XML y salga del **Custom UI Editor**.
- Abra el libro en Excel 2016. Aparecerá la nueva pestaña.
- Ahora puede pegar el código desde el portapapeles en un módulo estándar VBA y personalizarlo.

Ejemplo de código personalizado:



Para probar este código, debe crear en el libro una hoja llamada **Leeme**.

3. Etiquetas XML correspondientes a los distintos elementos de la cinta

a. Pestañas y grupos

Control	Etiqueta	Ejemplo
Pestaña de la cinta	<tab... </tab>	<tab id = "PestañaPers" label = "Pestaña personalizada" </tab>
Grupo de comandos	<group... </group>	<tab id = "GrupoPers" label = "Grupo personalizado" <group id = "GrupoPers1" label = "Tipo de factura" </group> </tab>

Las etiquetas <group... </group> deben estar incluidas en una pestaña (etiquetas <tab... </tab>).

Estos son los identificadores (atributo "idMso") de las pestañas estándares de Excel 2016:

TabHome	pestaña **Inicio**
TabInsert	pestaña **Insertar**
TabPageLayoutExcel	pestaña **Diseño de página**
TabFormules	pestaña **Fórmulas**
TabData	pestaña **Datos**
TabReview	pestaña **Revisar**
TabView	pestaña **Vista**
TabDeveloper	pestaña **Desarrollador**
TabAddIns	pestaña **Complementos**

Ejemplo:

El siguiente código permite ocultar las pestañas **Revisar** y **Desarrollador**.

```
<customUI
  xmlns="http://schemas.microsoft.com/office/2009/07/customui">
  <ribbon startFromScratch="false">
   <tabs>
    <tab idMso="TabReview" visible="false"/>
    <tab idMso="TabDeveloper" visible="false"/>
  </tabs>
 </ribbon>
</customUI>
```

b. Principales controles de la cinta de opciones

Los controles siguientes se pueden insertar dentro de grupos (etiquetas <group... </group>).

Control	Etiqueta	Ejemplo
Botón de comando	<button... />	<button id= "btVend1" label= "Servicios" imageMso = "MeetingsWorkspace" onAction = "MuestraVendedor" />
Botón de alternar	<toogleButton... />	<toggleButton id="btProd" label="Productos" imageMso="Functions RecentlyUsedtInsertGallery" size="grande" onAction="Tipo de factura" getPressed="Activar" />
Etiqueta	<labelControl ... />	<labelControl id="Titulo" label="Tipo de factura"/>
Cuadro combinado	<comboBox ... </comboBox>	<combo Box id="cmb_TasaIVA" label= "Tasa de IVA" onChange= "IVA" get Text="getIVA" > <item id= "itIva1" label="10 %" /> <item id= "itIva2" label="21 %" /> </comboBox>
Cuadro de lista	<dropDown ... </dropDown>	<dropDown id="cmb_TasaIVA" label= "Tasa de IVA" > <item id= "itIva1" label="10 %" /> <item id= "itIva2" label="21 %" /> </drop Down>
Casilla de verificación	<checkBox ... />	<checkBox id="chkPort"

		label="Gastos de flete?" onAction="GastoPort" getPressed="getchkPort" />
Campo de entrada	<editBox ... />	<editBox id="edFlete" label="Importe de los gastos de flete: "onChange="MostrarFlete" getText="getPort" getEnabled="ActivationPort"/>
Galería (o tabla) de controles	<gallery ... </gallery>	<gallery id="galPlazos" label = "Plazos" imageMso="StartTimer" size="large" > <button id= "itPlazo1" label="Recepción de la factura" onAction="Muestra plazo" imageMso="StartTimer" /> <button id= "itPlazo2" label="30 días" onAction="Muestra plazo" imageMso="StartTimer" /> <button id= "itPlazo3" label="30 días desde el fin de mes" onAction="Muestra plazo" imageMso="StartTimer" /> </gallery>
Barra de menú	<menu ... </menu >	<menu id="mnu_Vendedores" label="Vendedor" imageMso="AccessTableContactos" size="large"> <button id= "btVend1" label="Pedro LOPEZ" onAction="Muestra Vendedor" /> <button id= "btVend2" label="Laura DURAN" onAction="Muestra Vendedor" /> <button id= "btVend3" label="Lucas PEREZ" onAction="Muestra Vendedor" /> </menu>
Menú dinámico (el código VBA, a través de la función GetContent, debe proporcionar el contenido del menú)	<dynamicMenu ... />	<dynamicMenu id="dmnu_TasaIVA" label="Flete" getContent="Lista_Fletes" imageMso="ShowTimeZones" size="large" />

c. Atributos de los controles de la cinta de opciones

Los siguientes atributos permiten definir el contenido y la apariencia de las pestañas, los grupos y otros controles.

Atributo	Tipo de dato	Descripción
columns	Numérico	Cantidad de columas en una tabla de controles.

description	Texto (4096)	Descripción del control.
enabled	Booleano	Indica si el control está activo.
id	Texto (1024)	Identificador único del control.
idMso	Texto (1024)	Identificador de los controles predefinidos de Excel.
image	Texto (1024)	Imagen asociada al control.
imageMso	Texto (1024)	Identificador de una imagen predefinida de Excel.
insertAfterMso	Texto (1024)	Identificador del control predefinido después del cual se insertará el control especificado.
insertBeforeMso	Texto (1024)	Identificador del control predefinido antes del cual se insertará el control especificado.
invalidateContentonDrop	Boolean	Indica si el contenido del control se actualiza cada vez que se selecciona.
itemHeight	Numérico (píxeles)	Alto de un elemento de una tabla de controles (galería).
itemWidth	Numérico (píxeles)	Ancho de un elemento de una tabla de controles (galería).
itemSize	"large" o "normal"	Tamaño del los elementos de un menú.
keytip	Texto (3)	Método abreviado del control.
label	Texto (1024)	Texto (o etiqueta) del control.
maxLength	Numérico	Cantidad de caracteres en un cuadro de entrada.
rows	Numérico	Cantidad de líneas en una tabla de controles.
screentip	Texto (1024)	Texto de la pista del control.
showImage	Booleano	Indica si está visible la imagen del control.
showItemImage	Booleano	Indica si está visible la imagen de un elemento (ítem).
showItemLabel	Booleano	Indica si está visible el texto asociado a un elemento (ítem).
showLabel	Booleano	Indica si está visible el texto asociado a un control.
size	"large" o "normal"	Tamaño del control.
sizeString	Texto (1024)	Ancho del control.
startFromScratch	Booleano	Indica si están visibles las pestañas predefinidas de Office.
supertip	Texto (1024)	Texto del popup complementario del control.
tag	Texto (1024)	Propiedad complementaria que se puede usar a través de código VBA (ejemplo: hipervínculo a una página web).
visible	Booleano	Indica si está visible el control.

d. Resumen de los atributos para cada control

	Cinta de Office (ribbon)	**Pestaña (tab)**	**Groupo (group)**	**Botón de comando (button)**	**Botón Alternar (toggle Button)**	**Etiqueta (labelControl)**	**Cuadro combinado (comboBox)**
columns							
description				X	X		

enabled				X	X	X	X
id		X	X	X	X	X	X
idMso				X	X	X	X
image				X	X		X
ImageMso				X	X		X
insertAfterMso		X	X	X	X	X	X
insertBeforeMso		X	X	X	X	X	X
invalidateContentonDrop							X
itemHeight							
ItemWidth							
ItemSize							
keytip		X	X	X	X	X	X
label		X	X	X	X	X	X
maxLength							X
rows							
screentip			X	X	X	X	X
showImage				X	X		X
showItemImage							X
showItemLabel							X
showLabel				X	X	X	X
size				X	X	X	X
sizeString							X
startFromScratch	X						
supertip			X	X	X	X	X
tag				X	X	X	X
visible		X	X	X	X	X	X

	Cuadro de lista (dropDown)	**Casilla de verificación (checkBox)**	**Cuadro de entrada (editBox)**	**Galería de controles (gallery)**	**Barra de menú (menu)**	**Menú dinámico (dynamicMenu)**
columns				X		
description	X	X		X		
enabled	X	X	X	X	X	X
id	X	X	X	X	X	X
idMso	X	X	X	X	X	X
image	X		X	X	X	X
ImageMso	X		X	X	X	X
insertAfterMso	X	X	X	X	X	X
insertBeforeMso	X	X	X	X	X	X
invalidateContent onDrop				X		X

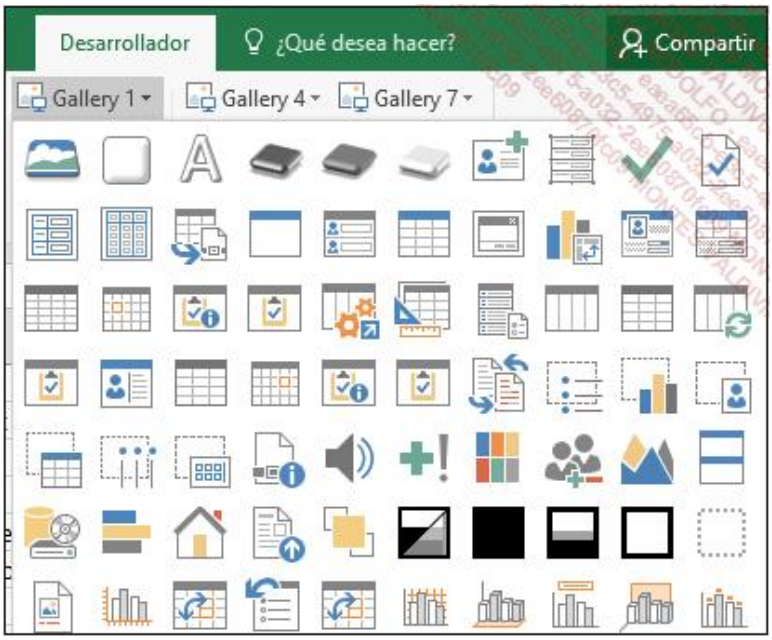
itemHeight				X		
ItemWidth					X	
ItemSize				X		
keytip	X	X	X	X	X	X
label	X	X	X	X	X	X
maxLength			X			
rows				X		
screentip	X	X	X	X	X	X
showImage	X		X	X	X	X
showItemImage	X			X		
showItemLabel	X			X		
showLabel	X		X	X	X	X
size	X	X	X	X	X	X
sizeString			X	X		
startFromScratch						
supertip	X	X	X	X	X	X
tag	X	X	X	X	X	X
visible	X	X	X	X	X	X

e. Imágenes de la galería de iconos de Microsoft Office

Los nombres de las imágenes usadas para personalizar la cinta de opciones (atributo ImageMso) corresponden al nombre del icono en la galería de Microsoft Office.

Para conocer la lista de nombres de iconos de esta galería:

- Abra en Excel el archivo **Office2007IconsGallery.xlsm** (disponible con los ejemplos de este libro o en la página de Microsoft).
- Seleccione la pestaña **Desarrollador**.
- Haga clic en alguno de los botones de **Gallery** para ver la lista de iconos.
- El nombre de la imagen aparecerá al señalar con el puntero del ratón el icono correspondiente (también puede hacer clic en el icono para mostrar su nombre en un cuadro de diálogo).



f. Funciones de llamadas Callbacks

Las funciones de llamada Callbacks permiten iniciar la ejecución de **procedimientos VBA**, con el objetivo de personalizar el contenido, la presentación y las acciones asociadas a los diferentes controles de la cinta de opciones.

La función **onAction** se ejecuta en el momento de la activación del control.

Las otras funciones (**getEnabled**, **getVisible**, **getLabel**...) permiten actualizar de manera dinámica algunos atributos en función de ciertas condiciones (por ejemplo, el campo "Cantidad de tasa portuaria" se desactiva cuando la casilla de opción "Tasa portuaria" no está seleccionada).

Función	Descripción	Control
onLoad	Procedimiento que se llama al cargar la cinta.	customUI
loadImage	Procedimiento para la carga de imágenes.	customUI
onAction	Procedimiento que se desencadena al activar un control.	Todos los controles
onChange	Procedimiento que se llama después de modificar el contenido del control y perder el foco.	comboBox, editBox
getText	Procedimiento que define el valor por defecto de un control.	comboBox, editBox
getContent	Procedimiento que define el código XML de personalización de un menú dinámico.	dynamicMenu
getitemCount	Procedimiento que define la cantidad de elementos del control.	comboBox, dropDown, gallery
getItemId	Procedimiento que define el identificador único (id) de cada elemento del control	
getItemImage	Procedimiento que define la imagen asociada a cada elemento del control.	
getItemLabel	Procedimiento que define la etiqueta asociada a cada elemento del control.	
getPressed	Procedimiento que indica si un botón de alternar	toggleButton, checkBox

	está activado o si una casilla está marcada.	
getSelectedItemId	Procedimiento que define el Id del elemento seleccionado por defecto dentro del control.	dropDown, gallery
getSelectedItemIndex	Procedimiento que define el índice del elemento seleccionado por defecto dentro del control.	dropDown, gallery
getDescription	Procedimiento que define el atributo "Description" del control.	Todos los controles que tienen el atributo personalizable por la función. Ejemplo: getImage se aplica a todos los controles que tengan el atributo Image. El tipo de datos que devuelve el procedimiento VBA debe ser el mismo que el del atributo correspondiente (por ejemplo, booleano para getEnable o cadena de caracteres para getImage), excepto para la funcion getSize (el procedimiento devuelve 1 para "normal" y 2 para "large".
getEnabled	Procedimiento que define si el control está activado o accesible.	
getKeytip	Procedimiento que define el método abreviado asociado al control.	
getImage	Procedimiento que define la imagen asociada al control.	
getItemLabel	Procedimiento que define la etiqueta del control.	
getItemHeigth	Procedimiento que define el alto de cada elemento de la galería.	
getItemWidth	Procedimiento que define el ancho de cada elemento de la galería.	
getItemScreentip	Procedimiento que define la pista de cada elemento de la galería.	
getItemSupertip	Procedimiento que define la pista suplementaria de cada elemento de la galería.	
getLabel	Procedimiento que define la etiqueta de un control.	
getScreentip	Procedimiento que define la pista de un control.	
getSupertip	Procedimiento que define la pista suplementaria de un control.	
getShowImage	Procedimiento que define si se debe mostrar la imagen del control.	
getShowLabel	Procedimiento que define si se debe mostrar la etiqueta del control.	
getSize	Procedimiento que define el tamaño del control.	
getVisible	Procedimiento que define si un control está visible.	

g. Uso de las funciones de llamada Callbacks

El principio de uso de estas funciones es el siguiente: en vez de indicar el valor de un atributo en el archivo XML de personalización (por ejemplo, visible = "False"), usted indica el nombre del procedimiento VBA que determina esa propiedad (por ejemplo, getVisible = "ProcVisible").

Las funciones de llamada devuelven información específica de la cinta de opciones por medio de parámetros (o argumentos), transferidos a los procedimientos VBA.

Para generar automáticamente los procedimientos VBA y los parámetros asociados, use el botón **Generate CallBacks**  de la utilidad **Custom UI Editor**.

Los principales parámetros son los siguientes:

Argumento	Descripción	Controles involucrados

Control	Objeto que representa el control que ha desencadenado el procedimiento. Este objeto tiene las propiedades "id" y "tag", que permiten, respectivamente, recuperar los valores de los atributos "id" y "tag".	Todos los controles
index	Determina el número de elemento para los controles de tipo lista.	Controles comboBox, dropDown, gallery y menu
pressed	Valor booleano que indica si la casilla está marcada (checkBox) o si el botón está presionado (toggleButton).	Controles CheckBox y toggleButton
Argumento precedido por la palabra clave **ByRef**	Argumento que permite recuperar y definir el atributo del control (por ejemplo: ByRef enabled, ByRef label).	Todos los controles

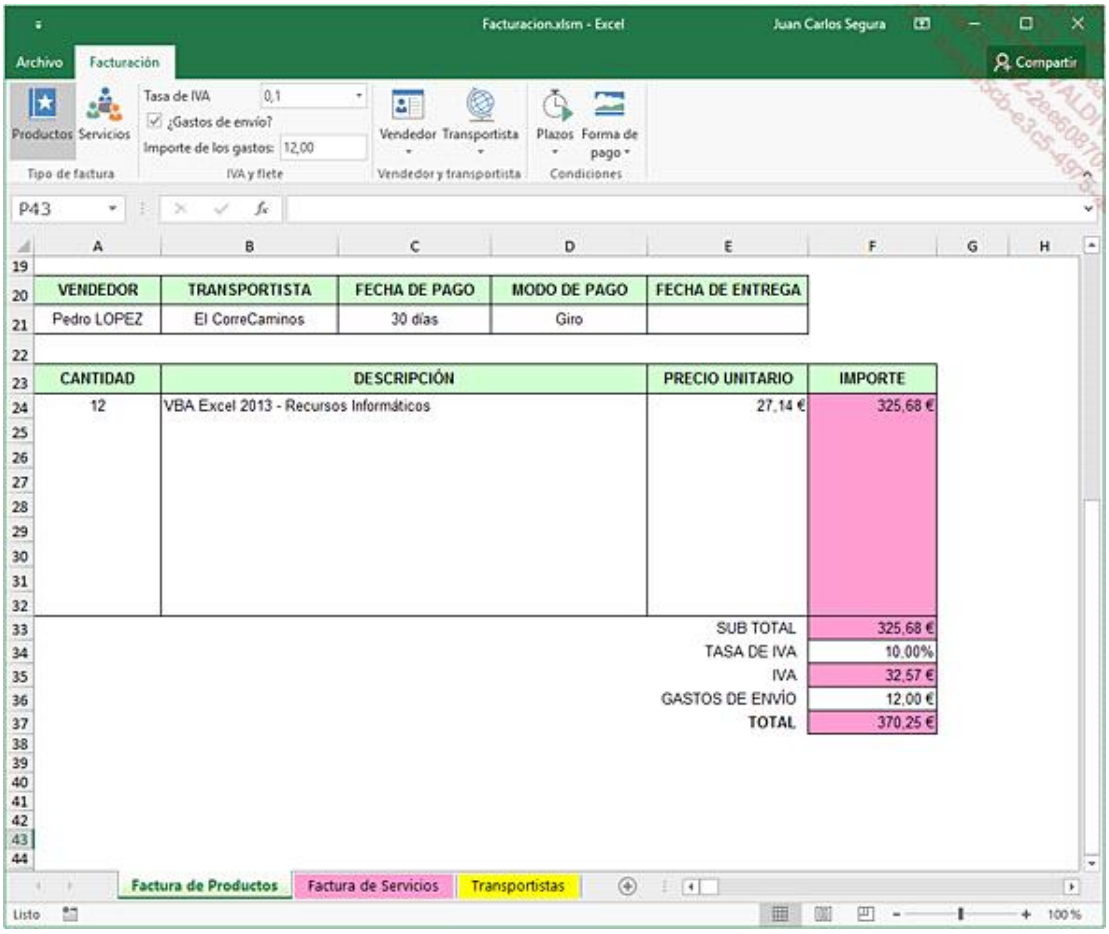
Ejemplo de cinta personalizada con el Custom UI Editor

1. Presentación

El siguiente ejemplo permite pedir la información de una factura a partir de la cinta de opciones.

La interacción con la hoja de Excel que contiene la información de la factura se hace en ambos sentidos:

- El ingreso de datos, a nivel de la cinta, afecta el contenido de las celdas de la hoja.
- La modificación de las celdas, dentro de la hoja de Excel, modifica los datos de la cinta.



La cinta consta de una pestaña de cuatro grupos que contienen los siguientes controles:

- Grupo "Tipo de factura": dos botones de alternar.
- Grupo "IVA y flete": una lista desplegable, una casilla de verificación y un cuadro de entrada.
- Grupo "Vendedor y transportista": un menú y un menú dinámico.
- Grupo "Condiciones": dos galerías de controles.

2. Código XML de la cinta

```
<customUI
```

```
xmlns="http://schemas.microsoft.com/office/2009/07/customui"
onLoad="CargaCinta" >

<!-- Oculta la cinta de opciones de Office -->
<ribbon startFromScratch="true">

<tabs>

 <!-- Creación de una pestaña personalizada -->
 <tab id="PESTAÑA_PERSO" label="Facturación">

    <!-- Grupo personalizado Tipo de factura -->
    <group id="Grupo1" label="Tipo de factura">

    <!-- Botones de alternar -->
    <toggleButton id="btProd" label="Productos"
        imageMso="FunctionsRecentlyUsedtInsertGallery"
        size="large" onAction="TipoFactura"
        getPressed="Activation" />
    <toggleButton id="btServ" label="Servicios"
        imageMso="MeetingsWorkspace"
        size="large" onAction="TipoFactura"
        getPressed="Activation" />
   </group>

    <!-- Grupo personalizado IVA y flete -->
    <group id="Grupo2" label="IVA y flete"
         getVisible="SelectGroup">

    <!-- Lista desplegable de tasas de IVA -->
    <comboBox id="cmbTasaIVA" label="Tasa de IVA" onChange="IVA"
       getText="getIVA" >
       <item id="itIVA1" label="10 %" />
       <item id="itIVA2" label="21 %" />
     </comboBox>

    <!-- Casilla Gastos de flete -->
    <checkBox id="chkFlete" label="¿Gastos de envío?"
      onAction="GastosFlete" getPressed="getchkFlete" />

    <!-- Cuadro de entrada Importe del flete -->
    <editBox id="edFlete" label="Importe de los gastos: "
      onChange="MostrarFlete" getText="getFlete"
      getEnabled="ActivarFlete" />
    </group>

    <!-- Grupo Vendedor y transportista -->
    <group id="Grupo3" label="Vendedor y transportista"
         getVisible="SelectGroup">

    <!-- Menú de vendedores -->
    <menu id="mnu_Vendedores" label="Vendedor"
        imageMso="AccessTableContacts" size="large">
        <button id="btVend1" label="Pedro LOPEZ"
           tag="Pedro LOPEZ" imageMso="ContactPictureMenu"
```

```
            onAction="MuestraVendedor" />
         <button id="btVend2" label="Laura DURAN"
            tag="Laura DURAN" imageMso="ContactPictureMenu"
            onAction="MuestraVendedor" />
         <button id="btVend3" label="Lucas PEREZ"
            tag="Lucas PEREZ" imageMso="ContactPictureMenu"
            onAction="MuestraVendedor" />
      </menu>

      <!-- Menú dinámico con la lista de transportistas -->
      <dynamicMenu id="dmnu_TasaIVA" label="Transportista"
           getContent="Lista_Transp"
           imageMso="ShowTimeZones" size="large" />
       </group>

      <!-- Grupo Condiciones -->
      <group id="Grupo4" label="Condiciones">

      <!-- Tabla de controles (galerías) -->
      <gallery id="galPlazo" label="Plazos"
               imageMso="StartTimer" size="large" >
         <button id="idPlazo1" label="A la recepción de la factura"
               onAction="MostrarPlazo"
               tag="De fecha de factura" imageMso="StartTimer" />
         <button id="idPlazo2" label="30 días"
               onAction="MostrarPlazo"
               tag="30 días" imageMso="StartTimer" />
         <button id="idPlazo3" label="30 días a fin de mes"
               onAction="MostrarPlazo"
               tag="30 días a fin de mes" imageMso="StartTimer" />
      </gallery>

      <gallery id="galPagos" label="Forma de pago"
               imageMso="PictureReflectionGalleryItem" size="large" >
          <button id="itModo1" label="Cheque" tag="Cheque"
               onAction="MostrarModo"
               imageMso="EnvelopesAndLabelsDialog" />
          <button id="itModo2" label="Giro" tag="Giro"
               onAction="MostrarModo"
               imageMso="ViewSlideShowView" />
          <button id="itModo3" label="Transferencia bancaria"
              tag="Transferencia bancaria"
              onAction="MostrarModo"
              imageMso="PictureReflectionGalleryItem" />
       </gallery>
     </group>
    </tab>
   </tabs>
  </ribbon>
</customUI>
```

3. Código VBA de personalización de la cinta (módulo "Cinta")

```
Option Explicit
Public objCinta As IRibbonUI
Public blnFlete As Boolean

'  Carga de la cinta
Sub CargaCinta(ribbon As IRibbonUI)

'  Variable objeto para usar la cinta
Set objCinta = ribbon

End Sub

'  Activa la hoja correspondiente al tipo de factura
Sub TipoFactura(control As IRibbonControl, pressed As Boolean)
If control.ID = "btProd" Then
    Sheets("Factura de productos").Select
Else
    Sheets("Factura de servicios").Select
End If

'  Muestra la cinta
objCinta = Invalidate
End Sub

'  Oculta los grupos 2 y 4 si es factura de servicios
Sub SelectGroup(control As IRibbonControl, ByRef returnedVal)
returnedVal = True
If control.ID = "Grupo2" And ActiveSheet.Name = "Factura de
servicios" Then
    returnedVal = False
End If
If control.ID = "Grupo3" And ActiveSheet.Name = "Factura
de servicios" Then
    returnedVal = False
End If
End Sub

'  Destaca el botón alternar seleccionado
Sub Activation(control As IRibbonControl, ByRef returnedVal)
returnedVal = False
If control.ID = "btProd" And _
       ActiveSheet.Name = "Factura de Productos" Then
    returnedVal = True
End If
If control.ID = "btServ" And _
       ActiveSheet.Name = "Factura de Servicios" Then
    returnedVal = True
End If

End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Recuperación de los valores de la factura    '
'   como valores por defecto de la cinta         '
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'   Recupera la tasa de IVA de la factura
Sub GetIVA(control As IRibbonControl, ByRef returnedVal)
   returnedVal = ""
   Application.Goto Reference:="TASA_IVA"
   returnedVal = ActiveCell.Value
End Sub

'   ¿Gastos de flete sobre la factura?
Sub getchkFlete(control As IRibbonControl, ByRef returnedVal)
   returnedVal = False
   Application.Goto Reference:="FLETE"
   If Val(ActiveCell.Value) > 0 Then returnedVal = True
   blnFlete = returnedVal
   If Not blnFlete Then
      Application.Goto Reference:="FLETE"
      ActiveCell.Value = ""
   End If
   objCinta.InvalidateControl "edFlete"
End Sub

'   Recupera el valor del flete de la factura
Sub getFlete(control As IRibbonControl, ByRef returnedVal)
   returnedVal = ""
   If blnFlete Then
      Application.Goto Reference:="FLETE"
      returnedVal = Format(ActiveCell.Value, "0.00")
   End If
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Recuperación de los valores de la cinta    '
'   y asignación a la factura                  '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'   Muestra la tasa de IVA seleccionada
Sub IVA(control As IRibbonControl, text As String)
   Application.Goto Reference:="TASA_IVA"
   ActiveCell.Value = text
End Sub

'   Si no hay gastos de flete, se desactiva el campo de entrada
Sub GastosFlete(control As IRibbonControl, pressed As Boolean)
    blnFlete = pressed
    If Not blnFlete Then
       Application.Goto Reference:="FLETE"
       ActiveCell.Value = ""
    End If
    objCinta.InvalidateControl "edFlete"
End Sub

'   Activa o desactiva el importe del flete
Sub ActivarFlete(control As IRibbonControl, ByRef returnedVal)
    returnedVal = blnFlete
```

```
End Sub

'   Asigna el importe del flete a la factura, después de controlar
Sub MostrarFlete(control As IRibbonControl, text As String)
'   El importe del flete debe ser numérico
If Not IsNumeric(text) Then
   MsgBox "Ingrese un valor numérico como importe del flete",
vbExclamation
Else
   Application.Goto Reference:="FLETE"
   ActiveCell.Value = text
   objCinta.InvalidateControl "edFlete"
End If

End Sub

'   Asigna el plazo seleccionado a la factura
Sub MostrarPlazo(control As IRibbonControl)
    If ActiveSheet.Name = "Factura de Productos" Then
      Application.Goto Reference:="PLAZO1"
    Else
      Application.Goto Reference:="PLAZO2"
    End If
    ActiveCell.Value = control.Tag
End Sub

'   Asigna el modo seleccionado a la factura
Sub MostrarModo(control As IRibbonControl)
    If ActiveSheet.Name = "Factura de Productos" Then
      Application.Goto Reference:="MODO1"
    Else
      Application.Goto Reference:="MODO2"
    End If
    ActiveCell.Value = control.Tag
End Sub

'   Asigna el vendedor seleccionado a la factura
Sub MuestraVendedor(control As IRibbonControl)
    Application.Goto Reference:="VENDEDOR"
    ActiveCell.Value = control.Tag
End Sub

'   Asigna el transportista seleccionado a la factura
Sub MuestraTransp(control As IRibbonControl)
   Application.Goto Reference:="TRANSPORTISTA"
   ActiveCell.Value = control.Tag
End Sub

'   Carga el menú dinámico con la lista de transportistas
'   mediante código XML
Sub Lista_Transp(control As IRibbonControl, ByRef returnedVal)
Dim i As Integer
Dim sXML As String
i = 1
sXML = "<menu xmlns=""http://schemas.microsoft.com/office/2006/01/
```

```
customui"">"
With Sheets("Transportistas")
    Do While .Cells(i, 1) <> ""
        sXML = sXML & AgregaItem(i, .Cells(i, 1))
        i = i + 1
    Loop
End With
returnedVal = sXML & " </menu>"
End Sub

'   Agrega un transportista al menú dinámico
Private Function AgregaItem(Index, Transportista As String)

AgregaItem = "<button " _
            & "id=" & Chr(34) & "btTransp" & Index & Chr(34) & " " _
            & "label=" & Chr(34) & Transportista & Chr(34) & " " _
            & "tag=" & Chr(34) & Transportista & Chr(34) & " " _
            & "imageMso=" & Chr(34) & "ShowTimeZones" & Chr(34) & " " _
            & "onAction=" & Chr(34) & "MuestraTransp" & Chr(34) & "/>"
End Function
```

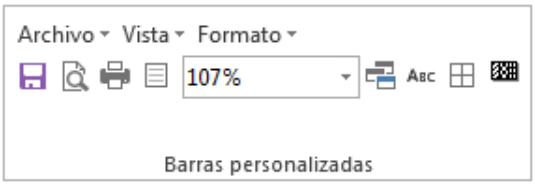
Personalización de la cinta de opciones mediante la colección CommandBars

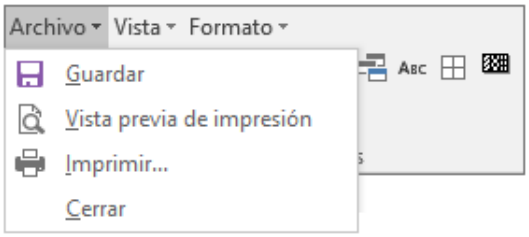
El acceso a esta colección permite realizar las siguientes operaciones con la ayuda del lenguaje VBA:

- Crear barras de herramientas personalizadas: se ubican automáticamente en el grupo **Barras de herramientas personalizadas** de la pestaña **Complementos** (la última de la derecha) y tiene el aspecto de las barras de herramientas de las versiones anteriores de Excel.
- Crear barras de menús personalizados: se pueden ubicar tanto en la pestaña **Complementos** como en una hoja Excel bajo la forma de menús contextuales (menús "popup").
- Crear una barra de comandos personalizada con el formato de Office 2016.
- Personalizar los comandos en su acción sobre las macros.

Ejemplos de barras de comandos

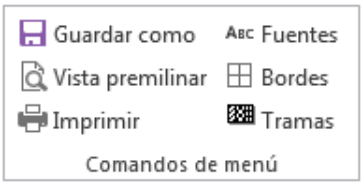
1. Barras de herramientas personalizadas





Se muestran dos barras de comandos: una barra de herramientas con seis botones y una barra de menús con tres opciones.

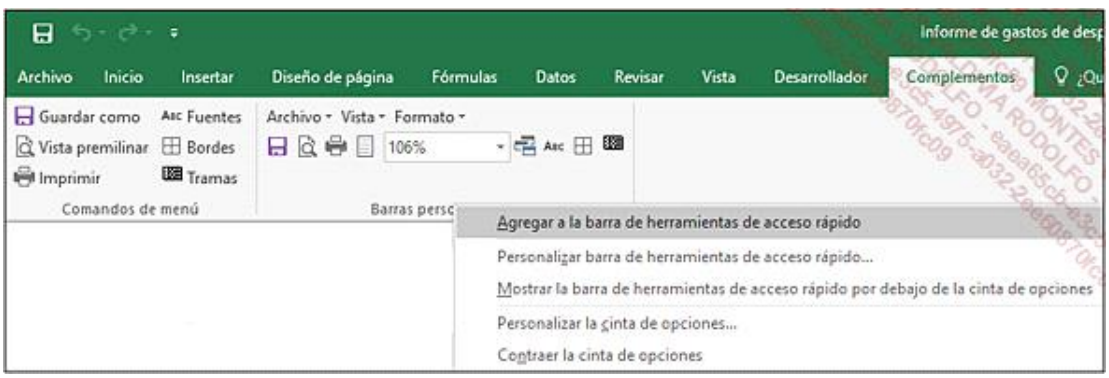
2. Comandos de menús en formato Office 2016



3. Agregar el grupo a la barra de herramientas de acceso rápido

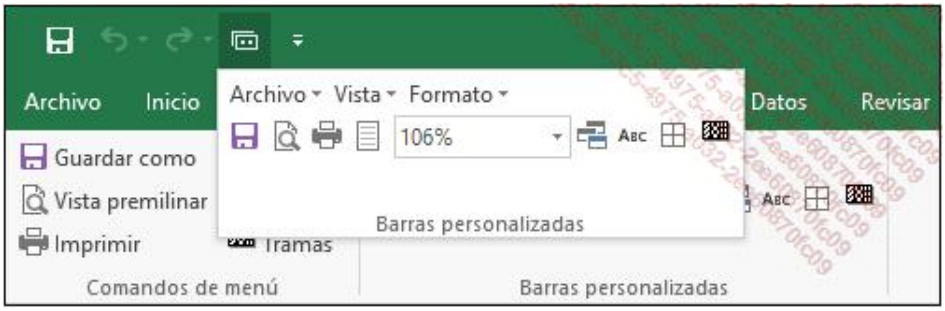
Las barras de comandos así creadas se pueden hacer accesibles desde la barra de herramientas de **acceso rápido** de la siguiente manera:

- Ubique el cursor bajo la barra de comandos y haga clic con el botón derecho del ratón.
- Seleccione la opción **Agregar a la barra de herramientas de acceso rápido**.

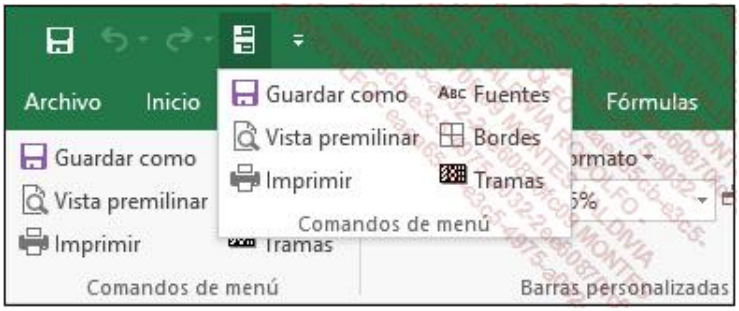


Normalmente, las barras de comandos son accesibles a través de botones de comandos agregados a la barra de herramientas de **acceso rápido**:

Barras de herramientas personalizadas



Comandos de menús



Barras de comandos

1. Terminología

a. Barra de comandos

Representa las barras de herramientas de Excel, las barras de herramientas personalizadas y las barras de menú.

b. Control

Representa un comando (botón de comando, opción de menú, etc.) de una barra de comandos.

2. Crear una barra de comandos

```
CommandBars.Add(Name, Position, MenuBar, Temporary)
```

Este método devuelve un objeto **CommandBar**.

Name	Nombre de la nueva barra de comandos.
Position	Posición de la nueva barra; puede adoptar una de las siguientes constantes: **msoBarLeft**: a la izquierda **msoBarTop**: arriba **msoBarRight**: a la derecha **msoBarBottom**: abajo **msoBarFloating**: no anclada **msoBarPopup**: menú contextual
MenuBar	Recibe el valor **True** si la nueva barra debe reemplazar la barra activa.
Temporary	Recibe el valor **True** en el caso de una barra temporal; las barras temporales se eliminan al cerrar la aplicación.

Ejemplo

Creación de una barra de menú y de una barra de herramientas. Estas barras solamente serán visibles después de agregar los controles asociados.

```
Dim Barra1 As CommandBar
Dim Barra2 As CommandBar

Sub Crear_Barras()

   '   Crea una barra de menú llamada "Menu1"
   Set Barra1 = CommandBars.Add(Name:="Menu1", _
       Position:=msoBarTop)
   '   Muestra la barra de menú creada
   Barra1.Visible = True
   '   Crea una barra de herramientas llamada "Menu2"
   Set Barra2 = CommandBars.Add(Name:="Menu2", _
       Position:=msoBarTop)
End Sub
```

Atención: si el código está escrito en el módulo de clase ThisWorkbook, es necesario indicar el objeto **Application** (ejemplo: Application.CommandBars).

3. Eliminar una barra de comandos

```
Expression.Delete
```

Expression	Expresión que devuelve el objeto **CommandBar** que hay que eliminar.

Ejemplo

Eliminar la barra de menú y la barra de herramientas (indispensable antes de crear nuevamente las barras).

```
Sub Eliminar_Barras()
   '   Elimina las barras de comandos personalizadas
   Application.CommandBars("Menu1").Delete
   Application.CommandBars("Menu2").Delete
End Sub
```

Las barras de comandos también pueden referenciarse por el nombre de la variable objeto.

```
Sub Eliminar_Barras()
      '   Elimina las barras de comandos personalizadas
      Barra1.Delete
      Barra2.Delete
End Sub
```

4. Mostrar una barra de comandos

La propiedad **Visible** permite mostrar u ocultar una barra de comandos. Después de crear una barra de comandos, la propiedad **Visible** tiene el valor **False**.

La propiedad **Enabled** permite activar o desactivar una barra de comandos. Si una barra de comandos está desactivada (Enabled = False), esta se elimina de la lista de barras de herramientas; la propiedad **Visible** ya no estará disponible.

Ejemplo

Muestra la barra de comandos Menu2 después de verificar que esta se encuentra disponible.

```
Sub Mostrar_Barras()

   '   Muestra la barra de comandos Menu2
   if Application.CommandBars("Menu2").Enabled = False then
      Application.CommandBars("Menu2").Enabled = True
   End if
```

```
   Application.CommandBars("Menu2").Visible = True

End Sub
```

Controles (opciones o botones de comando) de las barras de comandos

La colección de objetos **CommandBarControls** representa todos los controles de una barra de comandos.

Para acceder a esta colección, use la propiedad **Controls** de los objetos **CommandBar** y **CommandBarPopup**.

1. Agregar un control

```
Expression.Controls.Add(Type, Id, Parameter, Before, Temporary)
```

Este método devuelve un objeto **CommandBarButton**, **CommandBarComboBox** o **CommandBarPopUp**, que son objetos de tipo **CommandBarControls**.

Expression	Expresión que devuelve un objeto **CommandBar**; obligatorio.
Type	Tipo de control que hay que agregar; puede ser una de las siguientes constantes: **msoControlButton**: herramienta u opción de menú **msoControlEdit**: cuadro de entrada **msoControlDropDown**: cuadro de lista **msoControlComboBox**: cuadro de lista **msoControlPopUp**: menú contextual
Id	Entero que identifica un control integrado; si el valor del argumento es igual a 1 o si se omite, se agrega un control personalizado vacío del tipo indicado en la barra de comandos.
Parameter	En el caso de controles integrados, la aplicación contenedor lo usa para ejecutar el comando; en el caso de controles personalizados, este argumento puede servir para enviar información a los procedimientos Visual Basic o para almacenar la información en el control.
Before	Número que indica la posición del nuevo control en la barra de comandos; si no se especifica este argumento, el control se agrega al final de la barra de comandos.
Temporary	Recibe el valor **True** en el caso de un control temporal; los controles temporales se suprimen cuando se cierra la aplicación de Excel.

2. Especificar el título de un control

- Use la propiedad **Caption** del control.

En el caso de un menú, esta propiedad indica su título; en el caso de un botón, indica la etiqueta informativa que aparece.

3. Eliminar un control

```
Expression.Delete
```

Expression	Expresión que devuelve el objeto **CommandBarControls** que hay que eliminar.

4. Asociar un procedimiento a un control

- Use la propiedad **onAction** del control.

El nombre del procedimiento que desea asociar al control se debe indicar entre comillas.

Para mostrar la tecla de método abreviado del procedimiento asociado, use la propiedad **ShortCutText** del objeto **CommandBarButton**.

5. Otras propiedades

- Para activar o desactivar un control, use la propiedad **Enabled** del control.
- Para modificar el aspecto de la imagen de un botón, use la propiedad **FaceId** del objeto **CommandBarButton**.

Esta propiedad define el aspecto del botón, y no su función.

Ejemplos

Agregar un botón de comando personalizado a la barra de herramientas Menu2. Este botón abre el cuadro de diálogo Guardar como.

```
Sub Agregar_Control1()
    Dim m_Button as CommandBarButton
    ’  Agrega un botón de comando a la barra Menu2
     Set m_Button = Application.CommandBars("Menu2").Controls.Add _
           (Type:=msoControlButton)
    ’  Icono Guardar
     m_Button.FaceId = 3
    ’  Acción "GuardarComo"
     m_Button.OnAction = "GuardarComo"
End Sub
```

Procedimiento GuardarComo

```
Sub GuardarComo()
     ’   Cuadro de diálogo "GuardarComo"
     Application.Dialogs(xlDialogSaveAs).Show
End Sub
```

Agregar el menú Archivo y la opción Guardar como a la barra de comandos Menu1.

```
Sub Agrega_Control2()
   Dim m_Menu As CommandBarControl
   Dim m_Option As CommandBarControl
   ’ Agrega el menú Archivo
   Set m_Menu = Application.CommandBars("Menu1") _
                .Controls.Add (Type:=msoControlPopup)
   m_Menu.Caption = "Archivo"
```

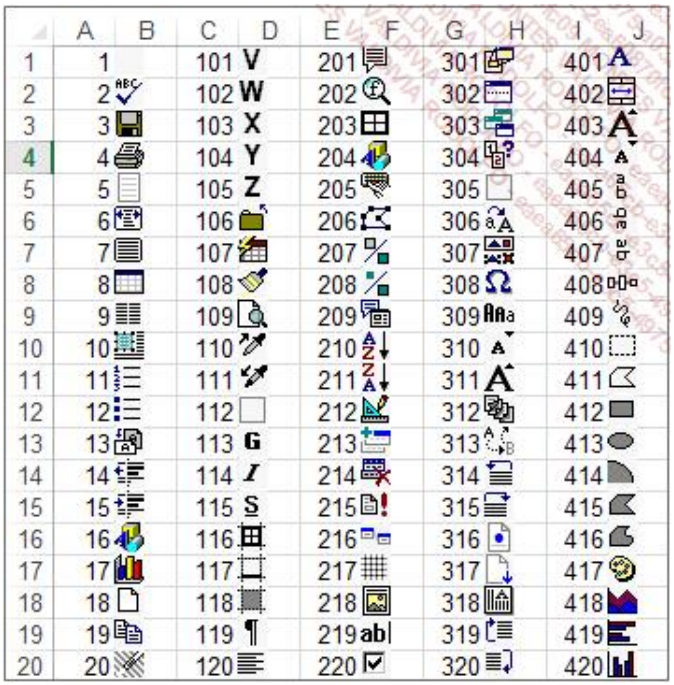
```
   ' Agrega el botón de comando
   Set m_Option = m_Menu.Controls.Add _
   (Type:=msoControlButton)
   m_Option.Caption = "Guardar Como"
   ' Icono Guardar
   m_Option.FaceId = 3
   ' Acción "GuardarComo"
   m_Option.OnAction = "GuardarComo"
End Sub
```

6. Lista de imágenes asociadas a los botones de comando

El siguiente procedimiento muestra, en la hoja de Excel activa, la lista de imágenes que se pueden asociar a los botones de comando (propiedad FaceId) y su número correspondiente.

```
Sub Muestra_Imagenes()
   Dim numFila As Integer
   Dim numCol As Integer
   Dim numImagen As Long
   Dim Menu1 As CommandBar
   Dim Button1 As CommandBarControl

   '   Crea una barra de herramientas temporal
   Set Menu1 = Application.CommandBars.Add _
       (Position:=msoBarFloating, temporary:=True)
   '   Agrega un botón de comando
   Set Button1 = Menu1.Controls.Add(msoControlButton)

   '   Modifica la imagen del botón de comando
   '   y la copia en una celda de Excel
   For numCol = 1 To 10 Step 2
     For numFila = 1 To 100
           numImagen = numImagen + 1
           Button1.FaceId = numImagen
           Button1.CopyFace
           ActiveSheet.Cells(numFila, numCol) = numImagen
           ActiveSheet.Paste Cells(numFila, numCol + 1)
     Next numFila
   Next numCol

   '   Redimensiona las columnas
   Columns("A:W").Select
   Selection.ColumnWidth = 4
   '   Elimina la barra de herramientas
   Menu1.Delete

End Sub
```

El resultado obtenido es el siguiente:

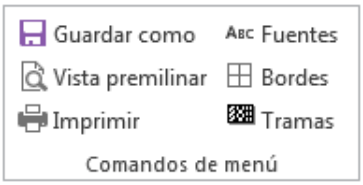


Ejemplos de menús personalizados

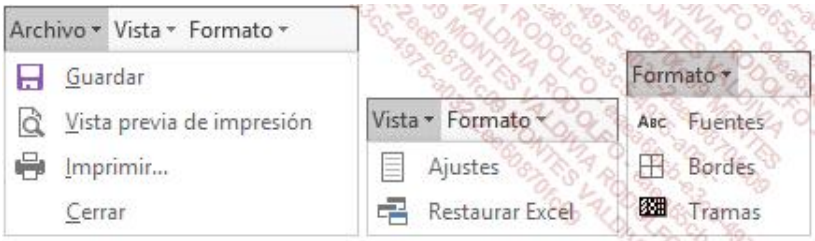
1. Presentación

Este ejemplo crea las siguientes barras de comandos:

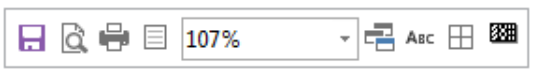
- Un grupo de comandos llamado "Comandos de menú":



- Una barra de menú con el título "Menú Gastos", dentro del grupo "Barras de herramientas personalizadas", que permite acceder a las siguientes opciones:



- Una barra de menú con el título "Gastos", dentro del grupo "Barras de herramientas personalizadas":



- Una barra de menú contextual que aparece cuando el usuario se sitúa en la zona llamada "Empleado" y hace clic con el botón secundario del ratón. Los empleados se extraen de la base de datos de Access Empleados.accdb (base de datos disponible con los ejemplos del libro).

2. Código de los ejemplos

Para crear los menús del siguiente ejemplo, se deben realizar las siguientes operaciones:

- Crear el documento de Excel.
- Definir un área de impresión.
- Definir un rango de celdas con el nombre **Empleado** que incluya las celdas Apellido y Número de empleado.
- Definir un rango de celdas con el nombre **INFORME GASTOS** que incluya las celdas que hay que imprimir.
- Asignar los siguientes nombres a las celdas que contienen la información del empleado: Empleado, Función, Población.
- Agregar un módulo estándar llamado **ProcMenus**; este módulo contendrá los procedimientos para crear las diferentes barras de comandos.
- Agregar un módulo estándar llamado **ProcAcciones**; este módulo contendrá los procedimientos personalizados asociados a los botones de comando.

3. Código del módulo de clase ThisWorkbook

```
option Explicit
_____________________________________________________________
Private Sub Workbook_Open()
   '   Muestra los menús personalizados
   Personalizar_Excel
   '   Ajusta el zoom
   Ajuste

End Sub
_____________________________________________________________
Private Sub Workbook_BeforeClose(Cancel As Boolean)
   '   Pide confirmación del cierre del libro
   If MsgBox("¿Desea cerrar el libro?", _
       vbQuestion & vbYesNo, ThisWorkbook.Name) = vbYes Then
      ' Muestra los menús de Excel
      Restaurar_Excel
   Else
      Cancel = True
   End If
End Sub
```

4. Código de la hoja "Nota de Gastos"

```
Option Explicit

Private Sub Worksheet_BeforeRightClick(ByVal Target As Excel.Range, _
          Cancel As Boolean)
   ’   Si la primera celda activa pertenece al rango llamado
   ’   "Empleado": mostrar el menú Empleados
   If Union(Target.Range("A1"), Range("Empleado")).Address = _
     Range("Empleado").Address Then
     CommandBars("Empleados").ShowPopup
     Cancel = True
End If
End Sub
```

5. Código del módulo ProcMenus

```
option Explicit

’   Declaración de variables
Dim m_Menu As CommandBarPopup
Dim m_Barra As CommandBar
Dim m_Option As CommandBarControl
Dim m_Button As CommandBarButton

Sub Personalizar_Excel()
   Dim i As Integer
   ’   Crea los menús y comandos personalizados
   Mostrar_Barra_Menús
   Mostrar_Barra_Herramientas
   Mostrar_Barra_Comandos
   ’   Menú contextual "Lista de empleados"
   Lista_Empleados
   ’   Oculta las barras de fórmulas, de estado
   ’   y los encabezados de fila y columna
   Application.DisplayFormulaBar = False
   Application.DisplayStatusBar = False
   Application.ActiveWindow.DisplayHeadings = False
End Sub
_____________________________________________________________________
Sub Restaurar_Excel()
   Dim i As Integer
   Dim m_Ctrl As CommandBarControl
   ’   Elimina los menús y barras de comandos
   On Error Resume Next
   Application.CommandBars("Menú Gastos").Delete
   Application.CommandBars("Gastos").Delete
   Application.CommandBars("Empleados").Delete
   On Error GoTo 0
   ’   Elimina los comandos personalizados
```

```
'   de la barra de comandos "Herramientas"
For Each m_Ctrl In Application.CommandBars("Tools").Controls
    If Not m_Ctrl.BuiltIn Then
       m_Ctrl.Delete
    End If
Next m_Ctrl
'   Vuelve a mostrar las barras de fórmulas, de estado
'   y los encabezados de fila y columna
Application.DisplayFormulaBar = True
Application.DisplayStatusBar = True
Application.ActiveWindow.DisplayHeadings = True
End Sub
Sub Mostrar_Barra_Menus()
On Error Resume Next
'    Elimina la barra de menú para recrearla
Application.CommandBars("Menu Gastos").Delete
On Error GoTo 0
'   Crea la barra de menú
Set m_Barra = Application.CommandBars.Add(Name:="Menu Gastos", _
    Position:=msoBarTop)
'   Muestra la barra de menú creada
Application.CommandBars("Menu Gastos").Visible = True
'   Agrega el menú "Archivo"
Set m_Menu = m_Barra.Controls.Add(Type:=msoControlPopup)
m_Menu.Caption = "Archivo"
'   Agrega los comandos del menú "Archivo"
'   Las acciones son las acciones por defecto
Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton, ID:=3)
m_Option.OnAction = "GuardarComo"
Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton, ID:=109)
Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton, ID:=4)
Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton, ID:=106)
'   Agrega el menú "Ver"
Set m_Menu = m_Barra.Controls.Add(Type:=msoControlPopup)
m_Menu.Caption = "Ver"
'   Agrega la lista desplegable de zooms
Set m_Option = m_Menu.Controls.Add(Type:=msoControlComboBox, ID:=1733)
'   Agrega los comandos del menú "Ver"
'   con la llamada a los procedimientos acción
Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton)
m_Option.FaceId = 175
m_Option.Caption = "Ajuste"
m_Option.OnAction = "Ajuste"
'   Agrega la opción " Restaurar Excel "
Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton)
m_Option.FaceId = 303
m_Option.Caption = "Restaurar Excel"
m_Option.OnAction = "Restaurar_Excel"
'   Agrega el menú "Formato"
Set m_Menu = m_Barra.Controls.Add(Type:=msoControlPopup)
m_Menu.Caption = "Formato"
'   Agrega los comandos del menú "Ver"
'   con la llamada a los procedimientos acción
Set m_Option = m_Menu.Controls.Add _
```

```
       (Type:=msoControlButton)
   m_Option.FaceId = 291
   m_Option.Caption = "Fuente"
   m_Option.OnAction = "Fuente"
   Set m_Option = m_Menu.Controls.Add _
      (Type:=msoControlButton)
   m_Option.Caption = "Bordes"
   m_Option.OnAction = "Bordes"
   m_Option.FaceId = 1704
   Set m_Option = m_Menu.Controls.Add _
   (Type:=msoControlButton)
   m_Option.Caption = "Tramas"
   m_Option.OnAction = "Tramas"
   m_Option.FaceId = 1988
End Sub
```

```
Sub Mostrar_Barra_Herramientas()
   On Error Resume Next
   '    Elimina la barra de herramientas para recrearla
   Application.CommandBars("Gastos").Delete
   On Error GoTo 0
   '   Crea una barra de menú llamada "Gastos"
   Set m_Barra = Application.CommandBars.Add(Name:="Gastos", _
       Position:=msoBarTop)
   '   Muestra la barra de herramientas personalizada
   Application.CommandBars("Gastos").Visible = True
   '   Agrega los botones de comando estándares en la barra
   '   de herramientas
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton, ID:=3)
   m_Option.OnAction = "GuardarComo"
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton, ID:=109)
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton, ID:=4)
   '   Agrega un botón de comando personalizado
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton)
   With m_Button
      .BeginGroup = True
      .FaceId = 175
      .OnAction = "Ajuste"
      .TooltipText = "Ajuste"
   End With
   '   Agrega la lista desplegable de zooms
   Set m_Option = m_Barra.Controls.Add _
       (Type:=msoControlComboBox, ID:=1733)
   '   Agrega la opción "Restaurar Excel"
   Set m_Option = m_Barra.Controls.Add _
      (Type:=msoControlButton)
   m_Option.FaceId = 303
   m_Option.Caption = "Restaurar Excel"
   m_Option.OnAction = "Restaurar_Excel"
   '   Agrega los botones "Formato"
   Set m_Button = m_Barra.Controls.Add _
```

```
        (Type:=msoControlButton)
    With m_Button
       .BeginGroup = True
       .FaceId = 291
       .OnAction = "Fuente"
       .TooltipText = "Fuente"
    End With
    Set m_Button = m_Barra.Controls.Add _
        (Type:=msoControlButton)
    With m_Button
       .FaceId = 1704
    .OnAction = "Bordes"
       .TooltipText = "Bordes"
    End With
    Set m_Button = m_Barra.Controls.Add _
        (Type:=msoControlButton)
    With m_Button
       .FaceId = 1988
       .OnAction = "Tramas"
       .TooltipText = "Tramas"
    End With
End Sub
```

```
Sub Mostrar_Barra_Comandos()
    '   Objeto que representa la barra de herramientas "Tools"
    Set m_Barra = Application.CommandBars("Tools")
    '   Agrega los botones de comando estándares en la barra
    '   de herramientas
    Set m_Button = m_Barra.Controls.Add _
        (Type:=msoControlButton, ID:=3)
    m_Button.Caption = "Guardar como"
    m_Button.OnAction = "GuardarComo"
    Set m_Button = m_Barra.Controls.Add _
        (Type:=msoControlButton, ID:=109)
    m_Button.Caption = "Vista preliminar"
    m_Button.OnAction = "VistaPreliminar"
    Set m_Button = m_Barra.Controls.Add _
        (Type:=msoControlButton, ID:=4)
    m_Button.Caption = "Imprimir"
    m_Button.OnAction = "Imprimir"
    '   Agrega los comandos de tipo "Formato"
    Set m_Button = m_Barra.Controls.Add _
        (Type:=msoControlButton)
    With m_Button
       .FaceId = 291
       .OnAction = "Fuente"
       .TooltipText = "Fuente"
       .Caption = "Fuente"
    End With
    Set m_Button = m_Barra.Controls.Add _
        (Type:=msoControlButton)
    With m_Button
       .FaceId = 1704
       .OnAction = "Bordes"
       .TooltipText = "Bordes"
```

```
      .Caption = "Bordes"
   End With
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton)
   With m_Button
      .FaceId = 1988
      .OnAction = "Tramas"
      .TooltipText = "Tramas"
      .Caption = "Tramas"
   End With
End Sub
_____________________________________________________________________
Public Sub Lista_Empleados()
   Dim Db As Database
   Dim rstEmp As Recordset
   Dim Legajo As Long
   '   Crea el menú "Empleados"
   On Error Resume Next
   Application.CommandBars("Empleados").Delete
   On Error GoTo 0
   Set m_Barra = CommandBars.Add _
      (Name:="Empleados", Position:=msoBarPopup, Temporary:=True)
   '   Abre la tabla Empleados
   Set Db = OpenDatabase(ActiveWorkbook.Path & "\Empleados.accdb")
   Set rstEmp = Db.OpenRecordset("SELECT * FROM Empleados ORDER BY
Apellido,  Nombre")
   '   Muestra la lista de empleados
   Do While Not rstEmp.EOF
      Set m_Button = m_Barra.Controls.Add(Type:=msoControlButton)
      With m_Button
         If rstEmp("Tratamiento") = "Sra" Or rstEmp("Tratamiento") =
"Srta" Then
            .FaceId = 2148
         Else
            .FaceId = 2103
         End If
         .Caption = UCase(rstEmp("Apellido")) & " " & rstEmp("Nombre")
         Legajo = rstEmp("Legajo")
         .OnAction = "Mostrar_Empleado(" & Legajo & ")"
      End With
      rstEmp.MoveNext
   Loop
   '   Cierra los objetos de Access
   rstEmp.Close
   Db.Close
End Sub
```

6. Código del módulo ProcAction

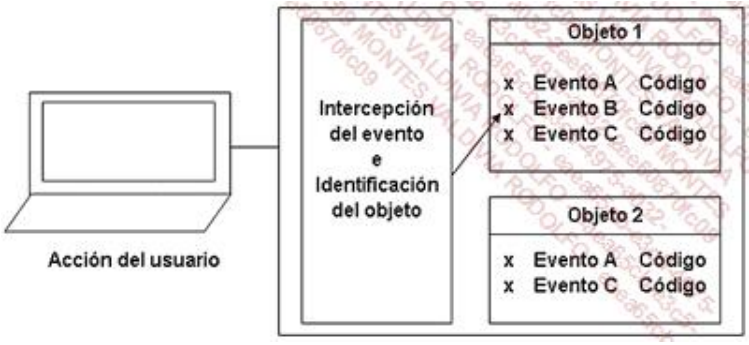
```
Sub Ajuste()
   '   Ajusta el zoom al contenido
   '   del rango llamado "NotaDeGastos"
```

```
   Application.Goto Reference:="INFORME GASTOS"
   ActiveWindow.Zoom = True
   Range("EMPLEADO").Select
End Sub
_____________________________________________________________________
Sub Mostrar_Empleado(Legajo As Long)
   Dim Db As Database
   Dim rstEmp As Recordset
   Dim strSql As String
   '   Abre la tabla "Empleados"
   Set Db = OpenDatabase(ActiveWorkbook.Path & "\Empleados.accdb")
   strSql = "SELECT * FROM Empleados WHERE [NumEmpleado] = " & NumEmp
   Set rstEmp = Db.OpenRecordset(strSql)
   '   Muestra las coordenadas del empleado seleccionado
   Range("EMPLEADO") = UCase(rstEmp("Apellido")) & " " & rstEmp("Nombre")
   Range("Función") = rstEmp("Cargo")
   Range("Población") = rstEmp("Municipio")
  '   Cierra los objetos de Access
  rstEmp.Close
   Db.Close
End Sub
_____________________________________________________________________
Sub Fuente()
   '   Muestra el cuadro de diálogo "Fuente"
   Application.Dialogs(xlDialogFormatFont).Show
End Sub
_____________________________________________________________________
Sub Bordes()
   '   Muestra el cuadro de diálogo "Bordes"
   Application.Dialogs(xlDialogBorder).Show
End Sub
_____________________________________________________________________
Sub Tramas()
   '   Muestra el cuadro de diálogo "Tramas"
   Application.Dialogs(xlDialogPatterns).Show
End Sub
_____________________________________________________________________
Sub GuardarComo()
   '   Cuadro de diálogo Guardar como
   Application.Dialogs(xlDialogSaveAs).Show
End Sub
_____________________________________________________________________
Sub VistaPreliminar()
   '   Vista preliminar
   ThisWorkbook.PrintPreview
End Sub
_____________________________________________________________________
Sub Imprimir()
   '   Impresión de la hoja de cálculo
   ActiveSheet.PrintOut
End Sub
```

Presentación

Un evento es una **acción** del usuario o del sistema reconocido por un objeto de Microsoft Excel. El evento desencadena un procedimiento, asociado al evento del objeto activo.

Estos procedimientos le permiten asociar un código personalizado en respuesta a un evento que se produce en un objeto de Excel (libro, hoja, formulario, gráfico, etc.).



Escritura de eventos

1. Eventos de libro, de hoja o de formulario

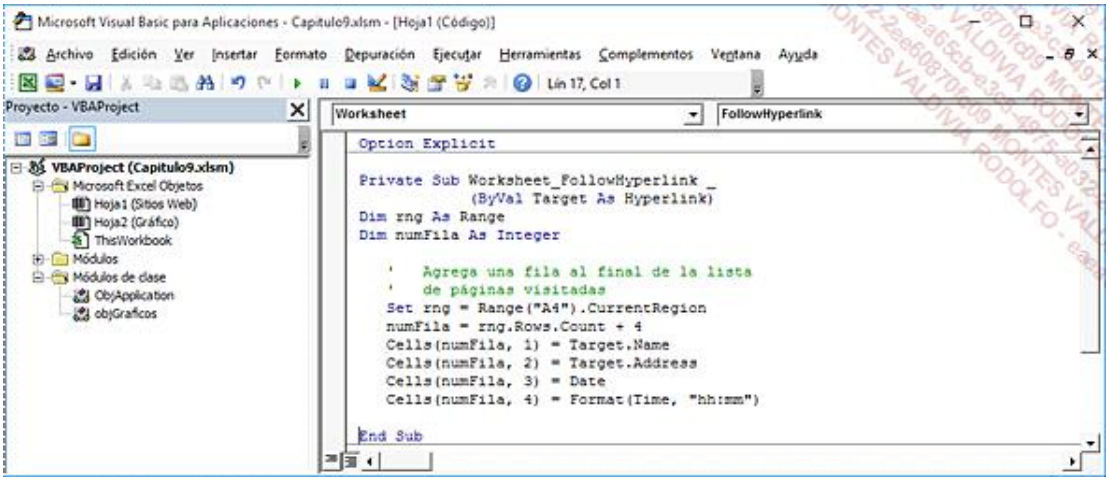
Usted puede acceder a los procedimientos de eventos asociados a un objeto de la siguiente manera:

- En la ventana **Explorador de proyectos**, haga doble clic en el objeto deseado (libro, hoja o formulario) para hacer que aparezca la ventana de código correspondiente.
- Abra la lista desplegable a la izquierda de la ventana de código y seleccione **Workbook**, **Worksheet** o **UserForm**, según el objeto seleccionado.
- También puede seleccionar un evento vinculado al objeto seleccionado en la lista desplegable de la derecha, para asociarle un código personalizado.

La ejecución de los procedimientos de eventos se puede desactivar en cualquier momento asignando el valor **False** a la propiedad **EnableEvents** del objeto **Application**.

Ejemplo

Este ejemplo muestra cómo obtener un listado histórico de todos los hipervínculos visitados en la hoja de cálculo activa.



Resultado en Excel:

	A	B	C	D	E	F	G	H
1								
2		LISTA DE PÁGINAS VISITADAS				PÁGINAS DISPONIBLES		
3								
4	Nombre	Dirección	Fecha	Hora		Microsoft		
5	Microsoft	http://www.microsoft.com/	18/09/2009	16:45		Microsoft Ibérica		
6	Microsoft Ibérica	http://www.microsoft.com/es/	04/11/2009	18:43		Ediciones ENI		
7	RENFE	http://www.renfe.es/	08/12/2009	17:42		Bolsa de Madrid		
8	Iberia	http://www.iberia.com/	12/01/2010	12:11		El tiempo		
9	Bolsa de Madrid	http://www.bolsamadrid.es/	13/01/2010	19:18		RENFE		
10	El tiempo	http://www.eltiempo.es/	15/02/2010	9:14		Iberia		
11	Microsoft	http://www.microsoft.com/	27/02/2010	10:02				
12	RENFE	http://www.renfe.es/	06/03/2010	0:35				
13	Ediciones ENI	http://www.ediciones-eni.com/	06/03/2010	0:36				
14								

Sitios Web | Grafico

2. Eventos del objeto Application

Se necesitan tres etapas para la escritura y ejecución de los eventos del objeto **Application**.

Etapa 1

- Inserte un módulo de clase:

  Insertar - Módulo de clase

  o abra la lista  y haga clic en **Módulo de clase**.

- Una vez insertado el módulo, asígnele un nombre.

Ejemplo

*Dele el nombre **ObjApplication** al módulo de clase.*

Etapa 2

- En el módulo de clase, cree un objeto **Application** con el siguiente código:

```
Public WithEvents NomObjeto As Application
```

Ejemplo

*Creación del objeto **oMiAplicación** como aplicación.*

```
Public WithEvents oMiAplicacion As Application
```

El objeto así creado queda disponible en la lista de la izquierda del módulo.

- Seleccione el objeto creado en la lista de la izquierda del módulo y luego seleccione el evento esperado en la lista de la derecha. Escriba el código de los procedimientos que desea generar.

Ejemplo

Creación de dos procedimientos de eventos: el primero realiza la inserción de una nueva hoja; el segundo, la creación de un nuevo libro.

```
Public WithEvents oMiAplicacion As Excel.Application
_____________________________________________________________
Private Sub oMiAplicacion_WorkbookNewSheet _
          (ByVal Wb As Workbook, ByVal Sh As Object)
   Dim oNomHoja As String
   '  Cada vez que se agrega una hoja, se pide al usuario
   '  que introduzca un nombre que a continuación se asignará a la hoja
   '  insertada tras las hojas existentes
   oNomHoja = InputBox("Introduzca el nombre de la hoja")
   ActiveSheet.Name = oNomHoja
```

```
   ActiveSheet.Move After:=Sheets(Sheets.Count)
End Sub
_____________________________________________________________________
Private Sub oMiAplicacion_NewWorkbook(ByVal Wb As Workbook)
   Dim iNbHojas As Integer
   Dim iNbActual As Integer
   Dim iDiferencia As Integer
   '  Por cada nuevo libro,
   '  solicitamos al usuario la cantidad de hojas
   '  Según el caso, se agregan o eliminan las hojas necesarias
   Do
      iNbHojas = Application.InputBox _
                 ("¿Cantidad de hojas?", Type:=1)
   Loop While iNbHojas = False
   iNbActual = Sheets.Count
   iDiferencia = iNbActual - iNbHojas
   '   Eliminar las hojas de más
   '   Eliminar los mensajes de alerta con el fin
   '   de no obtener mensajes en la eliminación de hojas
   Do While iDiferencia > 0
      Application.DisplayAlerts = False
      Sheets.Item(iDiferencia).Select
      ActiveWindow.SelectedSheets.Delete
      iDiferencia = iDiferencia - 1
   Loop

   '   Agregar hojas si es necesario
   '   Se desactivan los eventos para
   '   no indicar los nombres de las nuevas hojas
   Do While iDiferencia < 0
      Application.EnableEvents = False
      Sheets.Add
       iDiferencia =  iDiferencia + 1
   Loop
   '   Reactivar eventos y alertas
   Application.EnableEvents = True
   Application.DisplayAlerts = True
End Sub
```

Etapa 3

- Active un módulo cualquiera y conecte el objeto declarado en el módulo de clase con el objeto **Application** para las siguientes instrucciones:

```
Dim oNomVariable As New NomModuloDeClase

Sub NomProced ()
Set oNomVariable.NomObjeto = Application
End Sub
```

Ejemplo

Agregue el siguiente código en el módulo Declaraciones.

```
Option Explicit
Dim oApp As New ObjApplication
_____________________________________________________________
Sub InicializaMiAplicacion()
   Set oApp.oMiAplicacion = Application
End Sub
```

Finalmente, llame al procedimiento InicializaMiAplicacion al abrir el libro (módulo de clase ThisWorkbook).

```
Private Sub Workbook_Open()
   InicializaMiAplicacion
End Sub
```

Cuando se abra el libro, se ejecutarán automáticamente los procedimientos de eventos creados durante la etapa 2 y se agregarán los libros o las hojas. Estos procedimientos se desactivarán al cerrar el libro.

3. Evento asociado a un gráfico incrustado

La colección **Charts** (del objeto **Workbook**) contiene todos los gráficos del libro especificado.

Se necesitan tres etapas para la escritura y la ejecución de los eventos asociados a un gráfico incrustado.

Etapa 1

- Inserte un módulo de clase:

  Insertar - Módulo de clase

  o abra la lista  y haga clic en **Módulo de clase**.

- Una vez insertado el módulo, asígnele un nombre.

Ejemplo

*Dele el nombre **ObjGraficos** al módulo de clase.*

Etapa 2

- En el módulo de clase, cree un objeto gráfico para el siguiente código:

```
Public WithEvents NomObjeto As Chart
```

Ejemplo

*Creación del objeto llamado **oChart1** como gráfico incrustado.*

```
Public WithEvents oChart1 As Chart
```

El objeto así creado queda disponible en la lista de la izquierda del módulo.

- Seleccione el objeto creado en la lista de la izquierda del módulo y luego seleccione el evento esperado en la lista de la derecha. Escriba el código de los procedimientos que desea generar.

Ejemplo

Creación de dos procedimientos de eventos: uno desactiva el gráfico, el otro especifica sus dimensiones.

```
Option Explicit
Public WithEvents oChart1 As Chart
_____________________________________________________________
Private Sub oChart1_Deactivate()
   Dim sRespuesta As String
   '  Cada vez que se desactiva el gráfico
   '  se pregunta si hay que guardar el libro
   sRespuesta = MsgBox("¿Guardar los cambios?", vbYesNo)
   If sRespuesta = vbYes Then ActiveWorkbook.Save
End Sub
_____________________________________________________________
Private Sub oChart1_Resize()
   Dim oGrafico As Object
   '  Cada vez que cambia el tamaño del gráfico
   '  se muestran la primera y la última celda oculta
   Set oGrafico = Worksheets(2).ChartObjects(1)
   MsgBox "Este gráfico oculta la celda: " _
          & oGrafico.TopLeftCell.Address _
          & Chr(13) & "hasta la celda: " _
          & oGrafico.BottomRightCell.Address
End Sub
```

Etapa 3

- Active un módulo cualquiera y conecte el objeto declarado en el módulo de clase con el objeto gráfico incrustado para las siguientes instrucciones:

```
Dim oNomVariable As New NomModuloDeClase

Sub NomProced ()
Set oNomVariable.NomObjeto =
   WorkSheets(HojaDeGrafico).
   ChartObjects(NumeroDeGrafico).Chart
End Sub
```

Ejemplo

Para asociar los eventos al primer gráfico de la segunda hoja de cálculo, agregue el siguiente código en el módulo Declaraciones.

```
Dim obj As New ObjGraficos
_____________________________________________________________
Sub InicializaGrafico()
```

```
Set obj.Chart1 = Worksheets(2).ChartObjects(1).Chart
End Sub
```

Finalmente, llame al procedimiento InitMiAplicacion al abrir el libro (módulo de clase ThisWorkbook).

```
Private Sub Workbook_Open()
InicializaGrafico
End Sub
```

Al abrir este libro, los procedimientos creados durante la etapa 2 se ejecutarán automáticamente y se redimensionará o se desactivará el gráfico situado en la segunda hoja de cálculo. Estos procedimientos se desactivarán al cerrar el libro.

Eventos del objeto Application

AfterCalculate

Ocurre después del recálculo de los formularios del libro.

NewWorkBook

Ocurre al crear un nuevo libro.

ProtectedViewWindowActivate

Ocurre al activar una ventana protegida.

ProtectedViewWindowBeforeClose

Ocurre inmediatamente antes de cerrar una ventana protegida o un libro dentro de una ventana protegida.

ProtectedViewWindowBeforeEdit

Ocurre al modificar una ventana protegida.

ProtectedViewWindowDeactivate

Ocurre al cerrar una ventana protegida.

ProtectedViewWindowOpen

Ocurre cuando se abre un libro en una ventana protegida.

ProtectedViewWindowResize

Ocurre cuando se redimensiona una ventana que se encuentra en modo protegido.

SheetActivate

Ocurre al activar una hoja.

SheetBeforeDoubleClick

Ocurre al hacer doble clic en una hoja de cálculo, antes de la acción predeterminada para el doble clic.

SheetBeforeRightClick

Ocurre al hacer clic con el botón secundario del ratón en una hoja de cálculo, antes de la acción predeterminada.

SheetCalculate

Ocurre cuando se recalcula toda la hoja de cálculo o después de actualizar un gráfico al modificar sus datos.

SheetChange

Ocurre cuando las celdas de una hoja de cálculo se modifican por el usuario o por un vínculo externo.

SheetDeactivate

Ocurre al desactivar una hoja de cálculo.

SheetFollowHyperlink

Ocurre cuando el usuario hace clic en un hipervínculo en Microsoft Excel.

SheetPivotTableAfterValueChange

Ocurre cuando se modifica o recalcula una celda o rango de celdas de una tabla dinámica.

SheetPivotTableBeforeAllocateChanges

Ocurre al actualizar cambios en una tabla dinámica.

SheetPivotTableBeforeCommitChanges

Ocurre antes de validar cambios en una tabla dinámica vinculados a un origen de datos OLAP.

SheetPivotTableBeforeDiscardChanges

Ocurre antes de descartar cambios en una tabla dinámica.

SheetPivotTableUpdate

Ocurre al actualizar la hoja de informe de una tabla dinámica.

SheetSelectionChange

Ocurre cuando cambia la selección en cualquier hoja de cálculo (el evento no ocurre si la selección se hace sobre una hoja de gráfico).

WindowActivate

Ocurre al activar una ventana de libro.

WindowDeactivate

Ocurre cuando se desactiva una ventana de libro.

WindowResize

Ocurre al cambiar el tamaño de una ventana de libro.

WorkbookActivate

Ocurre cuando se activa un libro.

WorkbookAddinInstall

Ocurre cuando se instala un libro en forma de una macro complementaria.

WorkbookAddinUninstall

Ocurre cuando se desinstala una macro complementaria.

WorkbookAfterSave

Ocurre después de guardar un módulo de clase.

WorkbookAfterXMLExport

Ocurre después de exportar un archivo XML.

WorkbookAfterXMLImport

Ocurre después de importar un archivo XML.

WorkbookBeforeClose

Ocurre justo antes de cerrar un libro.

WorkbookBeforePrint

Ocurre antes de imprimir un libro abierto.

WorkbookBeforeSave

Ocurre antes de guardar un libro abierto.

WorkbookBeforeXMLExport

Ocurre antes de exportar un archivo XML.

WorkbookBeforeXMLImport

Ocurre antes de importar un archivo XML.

WorkbookDeactivate

Ocurre cuando se desactiva un libro abierto.

WorkbookNewChart

Ocurre al crear un nuevo gráfico en un libro.

WorkbookNewSheet

Ocurre cuando se crea una nueva hoja en un libro abierto.

WorkbookOpen

Ocurre cuando se abre un libro.

WorkbookPivotTableOpenConnection

Ocurre al abrir la conexión de un informe de tabla dinámica con su fuente de datos.

WorkbookPivotTableCloseConnection

Ocurre al cerrar la conexión de un informe de tabla dinámica con su fuente de datos.

WorkbookRowsetComplete

Ocurre cuando el usuario extrae el juego de grabación de una tabla dinámica OLAP.

WorkbookSync

Ocurre al sincronizar la copia local de una hoja de cálculo hecha a partir de un área de trabajo con la copia en el servidor.

Eventos del objeto Workbook

Activate

Ocurre cuando se activa el libro.

AddinInstall

Ocurre cuando el libro se instala en forma de una macro complementaria.

AddinUninstall

Ocurre cuando el libro se desinstala en forma de una macro complementaria.

AfterSave

Ocurre después de guardar el libro.

AfterXMLExport

Ocurre después de exportar un archivo XML.

AfterXMLImport

Ocurre después de importar un archivo XML.

BeforeClose

Ocurre antes de cerrar el libro; si el libro fue modificado, este evento se produce antes de invitar al usuario a guardar los cambios.

BeforePrint

Ocurre antes de imprimir el libro (o cualquiera de sus partes).

BeforeSave

Ocurre antes de grabar el libro.

BeforeXMLExport

Ocurre antes de exportar un archivo XML.

BeforeXMLImport

Ocurre antes de importar un archivo XML.

Deactivate

Ocurre al desactivar un gráfico, una hoja de cálculo o un libro.

NewChart

Ocurre al crear un nuevo gráfico en el libro.

NewSheet

Ocurre cuando se crea una nueva hoja en el libro.

Open

Ocurre cuando se abre el libro.

PivotTableOpenConnection

Ocurre al abrir la conexión de un informe de tabla dinámica con su fuente de datos.

PivotTableCloseConnection

Ocurre al cerrar la conexión de un informe de tabla dinámica con su fuente de datos.

RowsetComplete

Este evento se desencadena cuando el usuario extrae el juego de grabación de una tabla dinámica OLAP.

SheetActivate

Ocurre cuando se activa una hoja.

SheetBeforeDoubleClick

Ocurre al hacer doble clic en una hoja de cálculo, antes de la acción predeterminada para el doble clic.

SheetBeforeRightClick

Ocurre al hacer clic con el botón secundario del ratón en una hoja de cálculo, antes de la acción predeterminada.

SheetCalculate

Ocurre cuando se recalcula toda la hoja de cálculo o después de que se recalcule un gráfico al modificar sus datos.

SheetChange

Ocurre cuando las celdas de una hoja de cálculo se modifican por el usuario o por un vínculo externo.

SheetDeactivate

Ocurre cuando se desactiva una hoja de cálculo.

SheetFollowHyperlink

Se produce al hacer clic en cualquier hipervínculo en Microsoft Excel.

SheetPivotTableAfterValueChange

Ocurre cuando se modifica o recalcula una celda o rango de celdas de una tabla dinámica.

SheetPivotTableBeforeAllocateChanges

Ocurre al actualizar cambios en una tabla dinámica.

SheetPivotTableBeforeCommitChanges

Ocurre antes de validar cambios en una tabla dinámica vinculados a un origen de datos OLAP.

SheetPivotTableBeforeDiscardChanges

Ocurre antes de descartar cambios en una tabla dinámica.

SheetPivotTableChangeSync

Ocurre después de modificar una tabla dinámica.

SheetPivotTableUpdate

Ocurre al actualizar la hoja del informe de tabla dinámica.

SheetSelectionChange

Se produce cuando cambia la selección en una hoja de cálculo cualquiera (el evento no se produce si la selección está en una hoja de gráfico).

Sync

Ocurre al sincronizar la copia local de una hoja de cálculo hecha a partir de un área de trabajo con la copia en el servidor.

WindowActivate

Ocurre al activar un libro.

WindowDeactivate

Ocurre al desactivar un libro.

WindowResize

Ocurre cuando cambia el tamaño de la ventana.

Eventos del objeto Worksheet

Activate

Ocurre cuando se activa un libro, una hoja de cálculo, una hoja de gráfico o un gráfico incrustado.

BeforeDelete

Ocurre antes de eliminar una hoja.

BeforeDoubleClick

Ocurre al hacer doble clic en una hoja de cálculo o un gráfico incrustado, antes de la acción predeterminada para el doble clic.

BeforeRightClick

Ocurre al hacer clic con el botón secundario del ratón en una hoja de cálculo o un gráfico incrustado, antes de la acción predeterminada.

Calculate

Ocurre al recalcular la hoja de cálculo.

Change

Ocurre cuando algunas celdas de la hoja de cálculo están modificadas por el usuario o por un vínculo externo.

Deactivate

Ocurre al desactivar el gráfico, la hoja de cálculo o el libro.

FollowHyperlink

Ocurre al hacer clic en un hipervínculo de una hoja de cálculo.

PivotTableAfterValueChange

Ocurre cuando se modifica o recalcula una celda o rango de celdas de una tabla dinámica.

PivotTableBeforeAllocateChanges

Ocurre al actualizar cambios en una tabla dinámica

PivotTableBeforeCommitChanges

Ocurre antes de validar cambios en una tabla dinámica vinculados a un origen de datos OLAP.

PivotTableBeforeDiscardChanges

Ocurre antes de descartar cambios en una tabla dinámica.

PivotTableChangeSync

Ocurre después de modificar una tabla dinámica.

PivotTableUpdate

Ocurre después de actualizar un informe de tabla dinámica en una hoja de cálculo.

SelectionChange

Ocurre cuando cambia la selección en una hoja de cálculo.

Eventos del objeto Chart

Activate

Ocurre cuando se activa una hoja de gráfico o un gráfico incrustado.

BeforeDoubleClick

Ocurre al hacer doble clic en un gráfico incrustado o una hoja de gráfico, antes de la acción predeterminada para el doble clic.

BeforeRightClick

Ocurre al hacer clic con el botón secundario en un gráfico incrustado o una hoja de gráfico, antes de la acción predeterminada correspondiente.

Calculate

Ocurre después de que el gráfico se actualice con datos nuevos o modificados.

Deactivate

Ocurre cuando se desactiva el gráfico, la hoja de cálculo o el libro.

MouseDown

Ocurre al presionar el botón izquierdo o derecho del ratón cuando el puntero está sobre un gráfico.

MouseMove

Ocurre al cambiar la posición del puntero del ratón sobre un gráfico.

MouseUp

Ocurre al soltar el botón izquierdo o derecho del ratón cuando el puntero está sobre un gráfico.

Resize

Se produce al cambiar el tamaño del gráfico.

Select

Se produce al seleccionar un elemento del gráfico.

SeriesChange

Ocurre cuando el usuario modifica el valor de un punto de datos del gráfico.

Diferentes tipos de error

Se distinguen diferentes tipos de error en el lenguaje VBA:

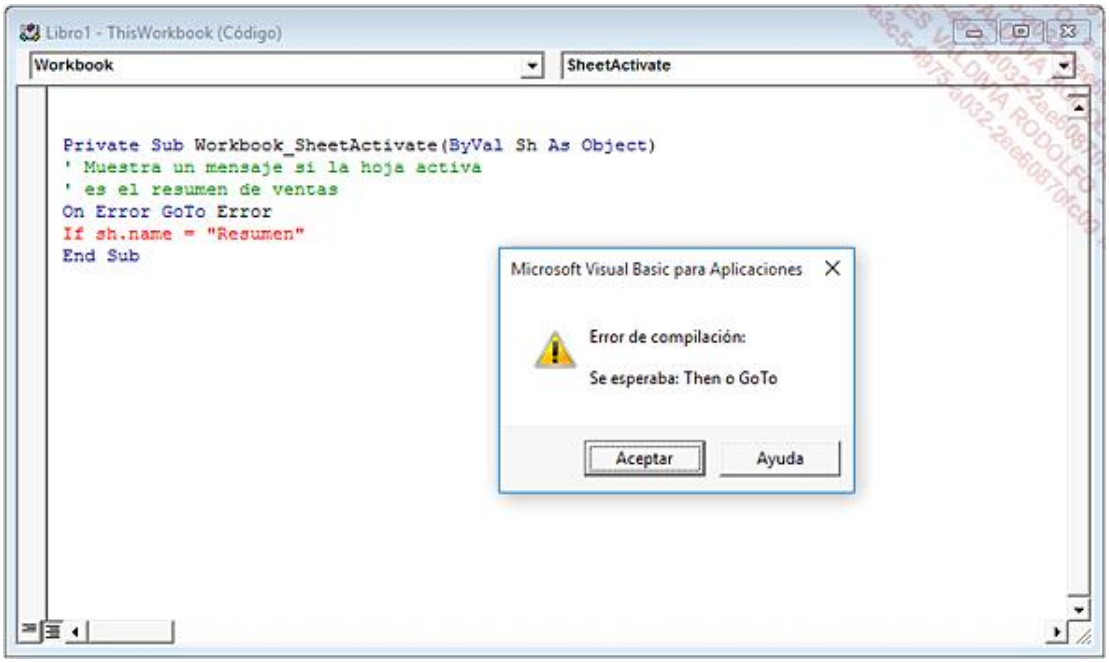
- Errores de sintaxis.
- Errores de compilación.
- Errores de ejecución.
- Errores de lógica.

1. Errores de sintaxis

Los errores de sintaxis se detectan automáticamente a medida que se introduce el código en VBA.

Para activar la comprobación de sintaxis, en el menú **Herramientas**, seleccione **Opciones**, luego seleccione la pestaña **Editor** y marque la casilla **Comprobación de sintaxis automática**.

Ejemplo



Los errores de sintaxis no corregidos provocarán un error de compilación; de ahí el mensaje que aparece.

2. Errores de compilación

Los errores de compilación se detectan cuando Excel trata de compilar el código.

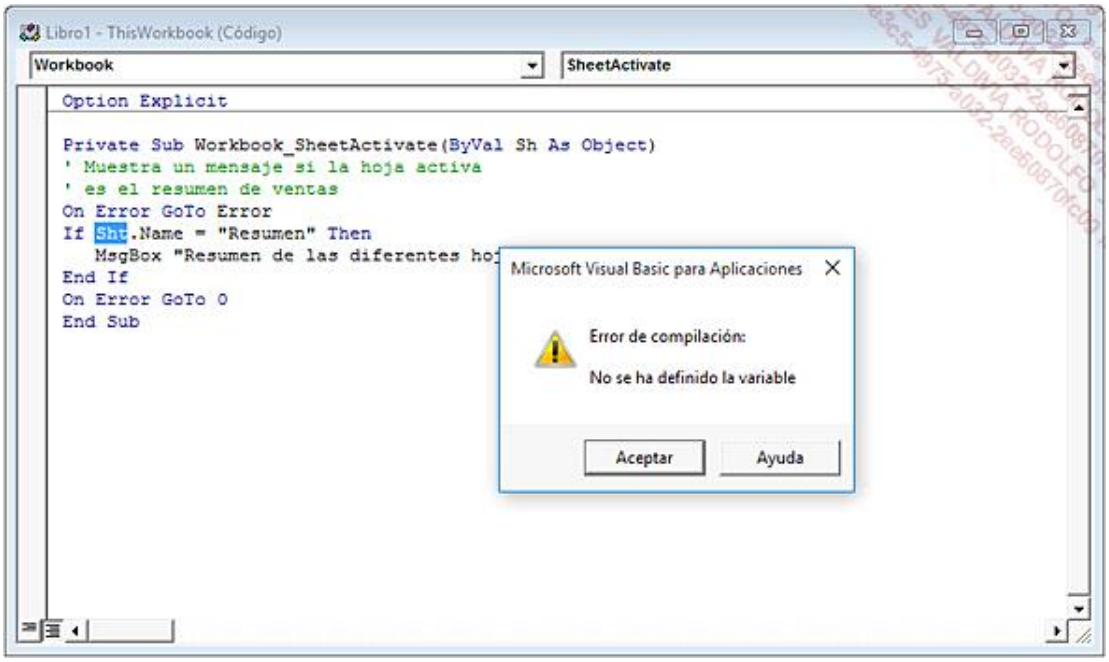
El código VBA puede compilarse de dos maneras:

- Bajo demanda, al seleccionar la opción **Compilar VBAProject** del menú **Depuración**. En este caso, el código se compila en su totalidad.
- Automáticamente **al ejecutar el código**. En este caso, solamente se compila el código contenido en los

procedimientos cuando se llaman por primera vez. Los procedimientos que no se llaman no se compilarán.

Se recomienda compilar el programa antes de ejecutarlo para ganar tiempo en la actualización.

Ejemplo



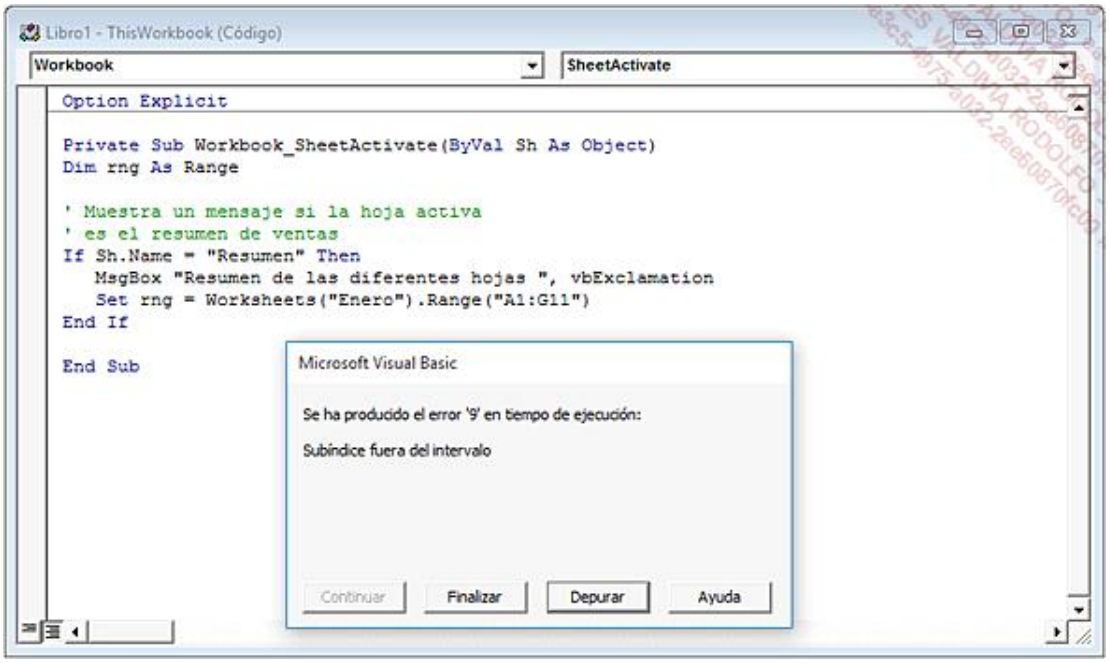
Es posible anticipar los errores de ejecución debidos a las variables no declaradas usando la instrucción **Option Explicit**. Si trata de usar un nombre de variable no declarado, se produce un error durante la compilación.

3. Errores de ejecución

Los errores de ejecución se detectan cuando Excel trata de ejecutar el código. Una instrucción, una operación, una llamada a una función, etc., inválidas provoca un error de ejecución. Por ejemplo, el uso de un índice erróneo en una colección o la asignación de un valor no numérico a una variable numérica puede provocar un error de ejecución.

Ejemplo

La hoja de cálculo "Enero" no existe en el libro activo.



4. Errores de lógica

Los errores de lógica tienen que ver con errores de razonamiento o con una mala traducción de un razonamiento en código VBA. Por ejemplo, un algoritmo de cálculo puede producir un error en el resultado si en su transcripción a VBA se omite o se traduce mal una operación o el algoritmo es erróneo.

Los errores de lógica son los más difíciles de detectar. En general, no producen un error de ejecución: pero producen un resultado distinto del esperado.

Para analizar este tipo de error, el entorno VBE dispone de herramientas de depuración que permiten ejecutar el código paso a paso y verificar el contenido de las variables a medida que se desarrolla el programa.

Depuración

1. Presentación

La depuración puede activarse de distintas maneras:

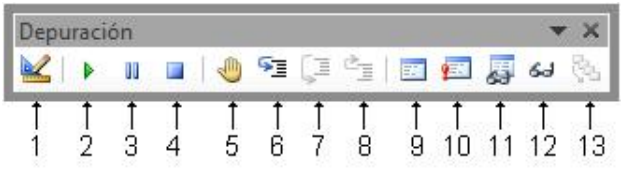
- Ejecutando el programa **paso a paso**.
- Insertando **puntos de interrupción** en el código VBA.
- Haciendo clic en el botón **Depurar** cuando se produce un error de ejecución.

Las diferentes herramientas de depuración permiten:

- Conocer en todo momento el valor de las variables o de las expresiones.
- Ejecutar instrucciones.
- Modificar interactivamente el código.
- Ejecutar el código paso a paso.
- Agregar puntos de interrupción.

2. La barra de herramientas Depuración

La barra de herramientas **Depuración** permite acceder directamente a las herramientas de depuración.



1. **Modo de diseño**: activa o desactiva el modo de diseño.

2. **Ejecutar** (método abreviado de teclado [F5]): ejecuta el código del procedimiento en curso, de la hoja UserForm activa o de una macro.

3. **Interrumpir** (método abreviado de teclado [Ctrl][Pausa]): interrumpe la ejecución del programa en curso y pasa al modo Interrupción.

4. **Restablecer**: borra el contenido de las variables y reinicializa el proyecto.

5. **Alternar punto de interrupción** (método abreviado de teclado [F9]): define o elimina un punto de interrupción en la línea actual; el código se ejecutará hasta el punto de interrupción, y luego pasará al modo depuración.

6. **Paso a paso por instrucciones** (método abreviado de teclado [F8]): ejecuta el código haciendo una interrupción después de cada instrucción del procedimiento en curso y de los procedimientos llamados.

7. **Paso a paso por procedimientos** (método abreviado de teclado [Mayús][F8]): ejecuta el código haciendo una interrupción después de cada instrucción del procedimiento en curso (las instrucciones de los procedimientos llamados se ejecutan de manera continua).

8. **Paso a paso para salir** (método abreviado de teclado [Ctrl][Mayús][F8]): ejecuta en forma continua las restantes líneas del procedimiento en curso.

9. **Ventana Locales**: muestra los valores de las variables locales del procedimiento.

10. **Ventana Inmediato** (método abreviado de teclado [Ctrl] G): muestra la ventana Inmediato, que permite ejecutar una instrucción de forma interactiva.

11. **Ventana Inspección**: muestra la lista de las variables de una inspección.

12. **Inspección rápida** (método abreviado de teclado [Mayús][F9]): muestra el valor de la expresión seleccionada.

13. **Pila de llamadas** (método abreviado de teclado [Ctrl] L): muestra la lista de llamadas de procedimiento cuya ejecución está en curso.

3. El objeto Debug

El objeto **Debug** permite enviar datos de salida a la ventana **Inmediato** durante la ejecución.

Métodos

Print

Muestra texto en la ventana **Inmediato**.

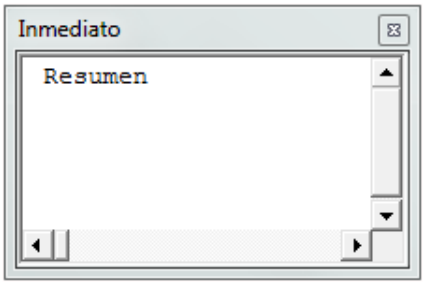
Assert

Suspende de forma condicional la ejecución de la línea en la que aparece el método.

Ejemplo

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
'   Muestra el nombre de la hoja activa en la ventana Inmediato
Debug.Print Sh.Name
If Sh.Name = "Resumen" Then ...
```

Resultado en la ventana Inmediato:



Administración de errores en VBA

Cuando se produce un error, VBA genera un error de ejecución que interrumpe la aplicación. Otros errores pueden hacer que el código VBA se comporte de manera imprevisible.

Para evitar esto, es posible manejar el error con la ayuda de las siguientes instrucciones y funciones:

On Error (instrucción)

Indica una secuencia de instrucciones que se ejecutará en caso de error.

Sintaxis 1

```
On Error GoTo línea
```

Activa la rutina de administración de errores que comienza en la línea indicada por el argumento `línea`.

El argumento `línea` debe ser una etiqueta o número de línea.

La `línea` debe pertenecer al mismo procedimiento que la instrucción **On Error**.

Si el argumento `línea` es un número de línea, debe ser obligatoriamente el primer carácter no vacío de la línea.

Sintaxis de la rutina de administración de errores

```
Línea:
   instrucciones
Resume
```

La instrucción **Resume** permite continuar la ejecución del código cuando termina la rutina de administración de errores, es decir, una vez resuelto el problema que produjo el error.

Hay tres sintaxis diferentes para **Resume**:

Resume 0	reanuda la ejecución del código donde el error se produjo.
Resume Next	reanuda a partir de la instrucción que sigue inmediatamente a la que generó el error.
Resume Línea	reanuda en la línea especificada por el argumento `Línea`.

Para impedir la ejecución del código de administración de errores cuando no hay errores, coloque una instrucción **Exit Sub**, **Exit Function** o **Exit Property** inmediatamente antes de la rutina de administración de errores.

Sintaxis 2

```
On Error Resume Next
```

Especifica que, en caso de error, la ejecución debe continuar.

Sintaxis 3

```
On Error GoTo 0
```

Permite interrumpir la administración de errores cuando el procedimiento está todavía en ejecución.

Ejemplo

Este procedimiento selecciona cada hoja y le cambia el nombre (por medio de un cuadro de diálogo) sucesivamente. Una rutina de administración de error se ejecuta cuando el nombre elegido es incorrecto o corresponde a un nombre existente.

```
Sub Errores_Nombre_Hojas()
   Dim oHojaTest As Worksheet
   Dim sNuevoNombre As String

   ’  En caso de error, se ejecutará la rutina "AdministracionDeErrores"
   On Error GoTo AdministracionDeErrores

   ’  Para cada hoja, seleccionarla y solicitar un nombre
   For Each oHojaTest In Sheets
       oHojaTest.Select
   1   sNuevoNombre = InputBox _
              (prompt:="Escriba el nombre para la hoja activa", _
               Default:=oHojaTest.Name)
   ’  Sale del procedimiento si el usuario hace clic en el
   ’  botón "Cancelar" o no indica ningún nombre
      If sNuevoNombre = "" Then Exit Sub
      oHojaTest.Name = sNuevoNombre
   Next
   ’  Desactiva la administración de errores
   On Error GoTo 0
   ’  Selecciona la primera hoja y guarda el libro
   Sheets(1).Select
   ActiveWorkbook.Save
   Exit Sub

   ’  Rutina de administración de errores que muestra
   ’  un mensaje y reanuda en la línea número 1
   AdministracionDeErrores:
      MsgBox "Nombre de hoja incorrecto o existente"
      Resume 1
End Sub
```

Error (función)

Devuelve un mensaje que corresponde a un número de error.

Sintaxis

```
Error (Código error)
```

Error (instrucción)

Simula la ocurrencia de un error.

Sintaxis

```
Error CódigoError
```

Los códigos de error personalizados deben tener un valor superior al de los códigos de error estándar e inferior a 65 535.

1. El objeto Err

El objeto **Err** contiene información que permite conocer el origen de un error de ejecución.

Propiedades

Description

Devuelve una cadena de caracteres que contiene una breve descripción del error.

HelpContext

Devuelve el identificador de contexto asociado a un tema de un archivo de ayuda.

HelpFile

Devuelve una cadena de caracteres que contiene la ruta de acceso completa del archivo de ayuda.

LastDLLError

Devuelve un código de error de sistema producido por una llamada a una biblioteca de vínculos dinámicos (DLL).

Number

Devuelve o establece un valor numérico que especifica el número del error.

Source

Devuelve o establece una cadena de caracteres que especifica el nombre del objeto o la aplicación que generó el error.

Métodos

Clear

Borra todas las propiedades establecidas del objeto **Err**.

Raise

Permite generar errores de ejecución.

Ejemplo

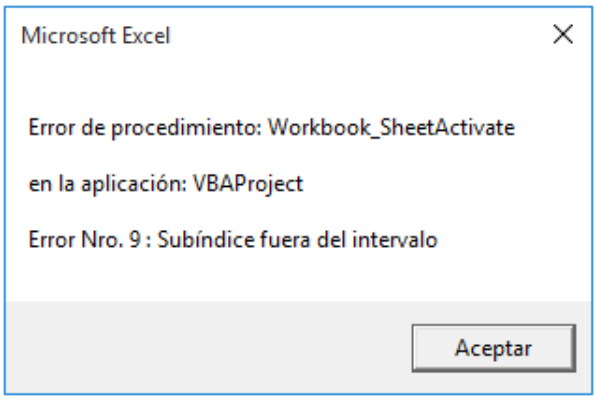
El siguiente código muestra un mensaje que brinda información sobre la naturaleza del error.

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
   Dim oRng As Range

   '  Muestra un mensaje si la hoja activada
   '  es el resumen de ventas
   On Error GoTo Error

   If Sh.Name = "Resumen" Then
      MsgBox "Resumen de las diferentes hojas ", vbExclamation
   Set oRng = Worksheets("Enero").Range("A1:G11")
   End If
   On Error GoTo 0
   Exit Sub
   '  En caso de error, mostrar un mensaje con
   '  la descripción del error encontrado
   Error:
      MsgBox "Error de procedimiento: " _
      & "Workbook_SheetActivate " _
      & vbCr & vbCr & "en la aplicación: " & Err.Source _
      & vbCr & vbCr & "Error Nro. " & Err.Number & " : " _
      & Err.Description
      Resume Next
End Sub
```

Para probar este ejemplo:

- Escribir el código en el módulo **ThisWorkbook**.
- Llamar una hoja **Resumen**.
- Poner el cursor en la hoja **Resumen**.

La ejecución de este código (si no existe la hoja Enero) devuelve el siguiente cuadro de mensaje:



Si la hoja **Enero** existe, no se produce ningún error.

La tecnología Automation

1. Presentación

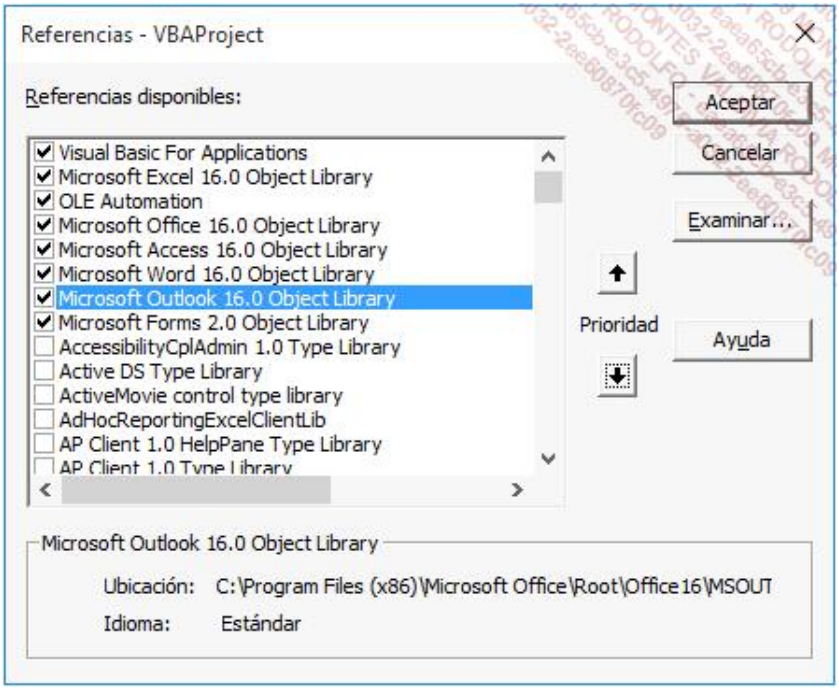
La tecnología **Automation**, llamada también **OLE** (*Object Linking and Embedding*) u OLE Automation, permite manipular los objetos de otra aplicación directamente desde Excel o VBA Excel.

Para poder funcionar, **Automation** necesita un cliente y un servidor, llamado **servidor OLE**. El servidor es la aplicación o el componente que realiza los servicios al cliente. El cliente (también llamado controlador), usa estos servicios para manejar la aplicación servidor y manipular sus objetos. Por ejemplo, si se ejecuta una combinación de correspondencia de Word desde VBA Excel, Excel es el cliente y Word el servidor OLE.

Una **biblioteca de objetos** es un archivo, generalmente con extensión olb, que provee la información necesaria para manipular los objetos puestos a disposición por un servidor. Se puede usar el Examinador de objetos para examinar el contenido de una biblioteca de objetos.

Para tener acceso a los objetos de otra aplicación, se debe hacer referencia a su biblioteca de objetos de la siguiente manera:

- Seleccione la opción **Referencias** del menú **Herramientas**. Aparece el cuadro de diálogo **Referencias - VBAProject** con todos los servidores OLE guardados en la base del registro.



- Active a continuación las referencias deseadas.

El número indicado con el nombre de la biblioteca de objetos corresponde a la versión de Microsoft Office (12 para la versión 2007, 14 para la versión 2010, 15 para la versión 2013 y 16.0 para la versión 2016).

2. Uso de la tecnología Automation

Para manipular los objetos de otra aplicación, proceda de la siguiente manera:

- Defina una variable objeto en el código VBA.

- Use las funciones **CreateObject** o **GetObject** para hacer referencia al objeto.

Ejemplos

Ejecutar Word

```
Dim oAppWord as Object
Set oAppWord = CreateObject("Word.Application")
```

o

```
Dim oAppWord As New Word.Application
```

Referencia a un documento de Word existente

```
Dim oDocWord As New Word.Document
Set oDocWord = GetObject("C:\Clientes\mailing.doc")
```

Si hay instalada más de una versión del mismo programa, se ejecutará la última versión guardada en la base de registro.

Sin embargo, puede precisar la versión del programa que desea ejecutar:

Ejemplo

Iniciar la aplicación Word 2013

```
Dim oAppWord as Object
Set oAppWord = CreateObject("Word.Application.15")
```

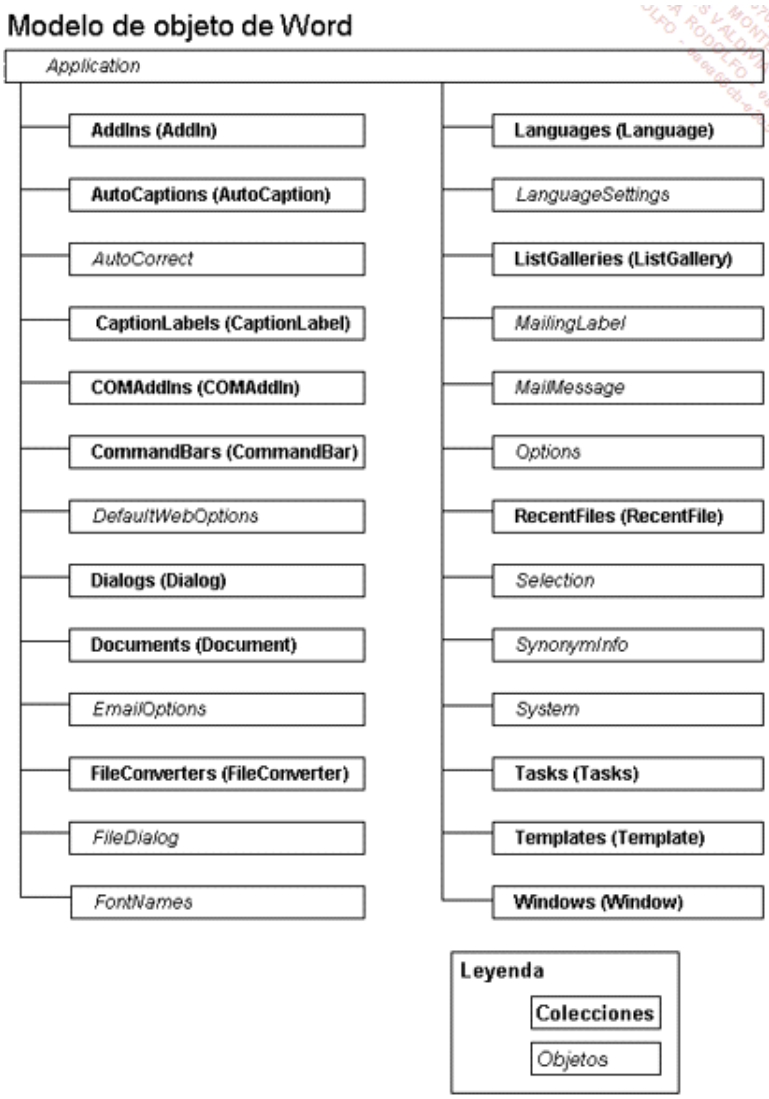
Las siguientes secciones describen cómo manejar distintos programas de Microsoft Office usando la tecnología Automation.

Los objetos, colecciones, métodos y propiedades de los modelos de objeto de Microsoft Office son numerosos. Solamente se describen a continuación los más usados.

Comunicación con Word desde Excel

1. El modelo de objeto de Word

Extracto del modelo de objeto de Word:



2. Principales colecciones del modelo de objeto de Word

AddIns

Colección de objetos **AddIn** que representa los modelos y complementos disponibles en Word, tanto si están cargados actualmente como si no lo están.

AutoCaptions

Colección de objetos **AutoCaption** que representa las leyendas agregadas automáticamente cuando esos elementos se insertan en un documento.

CaptionsLabels

Colección de objetos **CaptionLabel** que representa los rótulos de las leyendas agregadas automáticamente cuando esos elementos se insertan en un documento.

COMAddIns

Colección de objetos **COMAddin** que representan los complementos COM (*Component Object Model*), accesibles desde Word.

Dialogs

Colección de objetos **Dialog** que representan los cuadros de diálogo integrados de Word.

Documents

Colección de objetos **Document** que representan todos los archivos abiertos actualmente en Word.

FileConverters

Colección de objetos **Converter** que representan todos los conversores de archivo disponibles para abrir y guardar archivos desde Word.

Languages

Colección de objetos **Language** que representa todos los idiomas disponibles en Word para la comprobación de ortografía y gramática.

ListGalleries

Colección de objetos **ListGallery** que representa las tres bibliotecas de modelos de lista (con viñetas, números y jerarquía).

RecentFiles

Colección de objetos **RecentFile** que representa los últimos archivos abiertos en Word.

Tasks

Colección de objetos **Task** que representa el conjunto de aplicaciones en ejecución.

Templates

Colección de los objetos **Template** que representa los modelos actualmente disponibles.

Windows

Colección de objetos **Window** que representa todas las ventanas actualmente disponibles en Word.

3. Principales objetos del modelo de objeto de Word

AutoCorrect

Objeto que contiene la información relativa a la corrección automática de Word (opciones, excepciones, etc.).

DefaultWebOptions

Objeto que representa los atributos globales que usa Microsoft Word al guardar un documento como página web o al abrir una página web.

EmailOptions

Objeto que contiene los atributos globales que Microsoft Word utiliza al crear y editar mensajes de correo electrónico y las respuestas a esos mensajes.

FileDialog

Objeto que representa el cuadro de diálogo que le permitirá ejecutar acciones con un archivo (por ejemplo, abrir un archivo, guardarlo, etc.).

FontNames

Objeto que contiene los nombres de todas las fuentes disponibles.

LanguageSettings

Objeto que devuelve la información del diccionario de sinónimos.

MailingLabel

Objeto que representa a una etiqueta de correo.

MailMessage

Objeto que representa el mensaje de correo electrónico activo si Word es su editor de e-mail.

Options

Representa las opciones de la aplicación y del documento de Word.

Selection

Representa la selección actual dentro de una ventana o de un panel.

System

Contiene información sobre el sistema de su ordenador.

4. La colección Documents

La colección **Documents** está formada por todos los objetos **Document** abiertos en Word.

Principales métodos

Add

Crea un nuevo documento y lo agrega a la colección **Documents**.

Ejemplo: Documents.Add Template:="Normal"

Close

Cierra todos los documentos de Word abiertos.

Ejemplo: Documents.Close

Open

Abre el documento especificado y lo agrega a la colección **Documents**.

Ejemplo: Documents.Open FileName:="C:\Clientes\Informe.docx", ReadOnly:=True

Save

Guarda todos los documentos abiertos.

Ejemplo: Documents.Save

5. Objetos Document

Los objetos **Document** hacen referencia a un documento de Word. **ActiveDocument** designa al documento activo.

Principales métodos

Activate

Activa un documento ya abierto.

Ejemplo: Documents("Compras.docx").Activate

Close

Cierra un documento de Word abierto.

Ejemplo: Documents("Compras.docx").Close o ActiveDocument.Close

PrintOut

Imprime un documento o parte de él.

Ejemplo: imprimir las tres primeras páginas del documento activo.

ActiveDocument.ActiveWindow.PrintOut_Range:=wdPrintFromTo, From:="1", To_= "3"

PrintPreview

Muestra la vista preliminar de un documento.

Ejemplo: ActiveDocument.PrintPreview

Range

Devuelve un objeto **Range** (Selection).

Ejemplo: ActiveDocument.Range(0, 50).Bold = True

Save

Guarda un documento.

Ejemplo: ActiveDocument.Save

SaveAs

Guarda un documento con un nuevo nombre o en otro formato.

Ejemplo: ActiveDocument.SaveAs FileName:=strDocName

Principales colecciones

Characters

Colección de los caracteres de un documento.

Paragraphs

Colección de los párrafos de un documento.

Tables

Colección de las tablas contenidas en el documento.

Words

Colección de las palabras de un documento.

Ex : ActiveDocument.Words.Count

Principales objetos

AttachedTemplate

Devuelve un objeto **Template** que representa el modelo asociado al documento especificado.

DocumentTheme

Objeto que representa el tema de Office aplicado al documento especificado.

MailMerge

Objeto que permite acceder a las opciones de fusión y combinación de correspondencia en Word.

PageSetup

Objeto que representa las opciones de configuración de página.

Permissions

Objeto que representa las opciones de seguridad para el documento especificado.

WebOptions

Objeto que representa los atributos globales usados en Word para guardar un documento como página web.

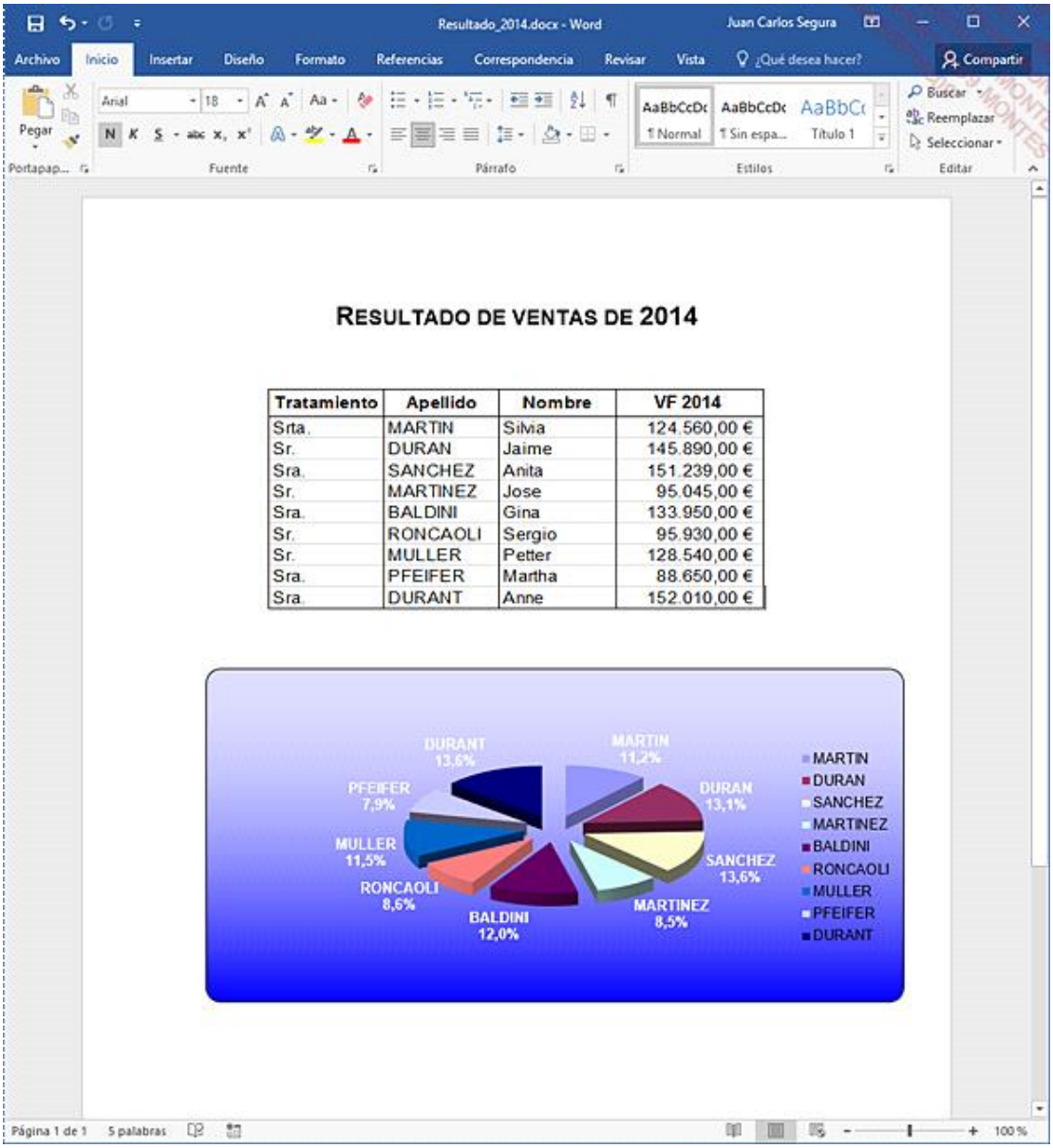
6. Ejemplo

Este ejemplo muestra cómo insertar un texto (título) y pegar una tabla y un gráfico de Excel en un nuevo documento de Word.

```
Public Sub CopiarEnWord()
   Dim oAppWord As New Word.Application
   Dim oDocWord As New Word.Document
   '  Agrega un nuevo documento
   With oAppWord
      .Visible = True
      Set oDocWord = .Documents.Add
      .Activate
   End With

   With oAppWord.Selection
       '   Agrega una línea de título y le aplica un formato
       .TypeText Text:="Resultado de ventas de 2014"
       .HomeKey Unit:=wdLine
       .EndKey Unit:=wdLine, Extend:=wdExtend
       .ParagraphFormat.Alignment = wdAlignParagraphCenter
       .Font.Size = 18
        With .Font
             .Name = "Arial"
             .Size = 18
             .Bold = True
             .Italic = False
             .SmallCaps = True
        End With

        '  Copia la tabla de Excel en el Portapapeles
       Range("A1:D10").Copy
        '  Pega la tabla en Word con vínculo
```

```
        .EndKey Unit:=wdLine
        .TypeParagraph
        .TypeParagraph
        .PasteSpecial Link:=True, DataType:=wdPasteOLEObject, _
                      Placement:=wdInLine, DisplayAsIcon:=False

        '  Copia el gráfico de Excel en el Portapapeles
        ActiveSheet.ChartObjects(1).Activate
        ActiveChart.ChartArea.Select
        ActiveChart.ChartArea.Copy
        '  Pega el gráfico en Word
        .TypeParagraph
        .TypeParagraph
        .Paste
    End With

    With oDocWord
         '   Guarda el documento de Word en la misma
         '   carpeta que el libro de Excel
         .SaveAs ThisWorkbook.Path & "\Resultado_2014.docx",  _
         Allowsubstitutions:=True
         '   Vista previa del resultado en Word
         .PrintPreview
    End With
    '   Reinicializa el objeto
    Set oAppWord = Nothing
End Sub
```

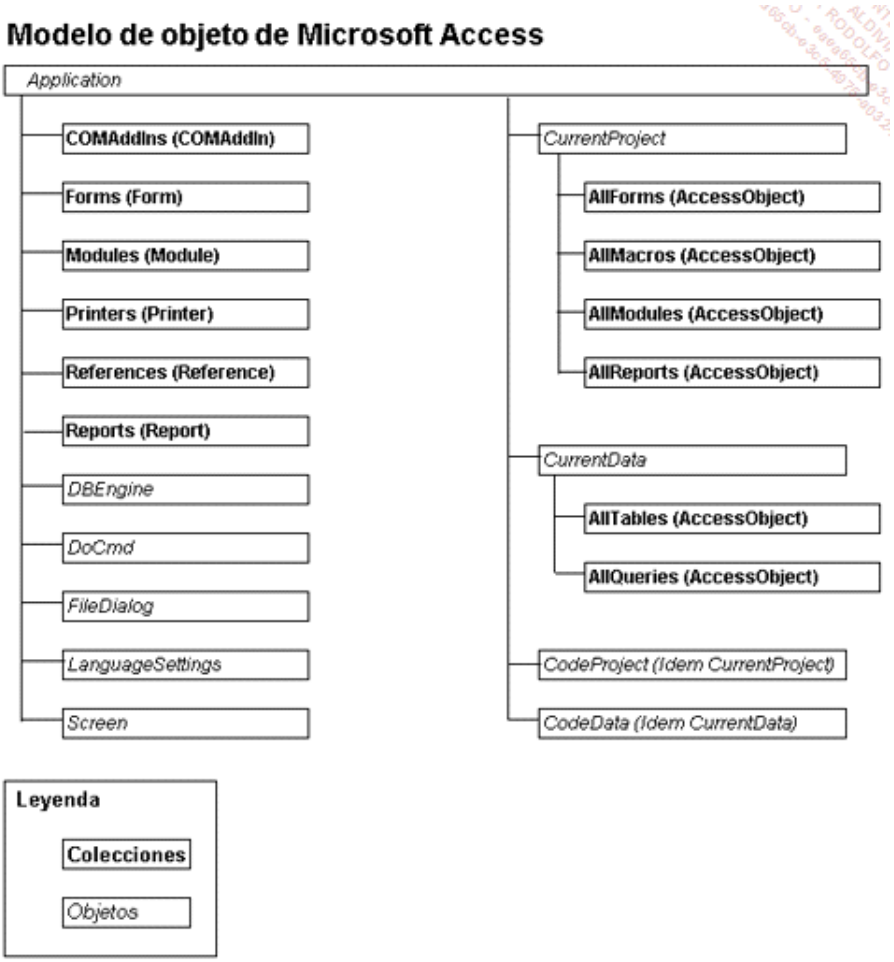
Resultado en Word 2016:



Comunicación con Access desde Excel

1. El modelo de objeto de Access

Extracto del modelo de objeto de Access:



2. Principales colecciones del modelo de objeto de Access

COMAddIns

Colección de macros complementarias (objetos **COMAddIn**) registrada en Access

Forms

Colección de formularios (objetos **Form**) abiertos actualmente en una base de datos de Access.

Modules

Colección de módulos estándares y módulos de clase (objetos **Module**) abiertos actualmente en una base de datos de Access.

Printers

Colección de objetos **Printer** que representan todas las impresoras disponibles actualmente en el sistema.

References

Colección de objetos **Reference** que representan las referencias VBA de una base de datos de Access.

Reports

Colección de listados (objetos **Report**) abiertos actualmente en una base de datos de Access.

3. Principales objetos del modelo de objeto de Access

DBEngine

Objeto que representa el motor de base de datos Microsoft Jet.

DoCmd

Objeto que permite realizar acciones de Access, como el cierre de ventanas, la apertura de formularios, de informes… La mayor parte de las acciones de macros se pueden convertir en VBA a través del objeto **DoCmd**.

FileDialog

Objeto que representa un cuadro de diálogo que permite realizar acciones sobre un archivo (ej.: apertura de un archivo, grabación de un archivo…).

LanguageSettings

Objeto que envía información de los parámetros de idioma de Access.

Screen

Objeto que permite acceder al formulario, al estado o al control que actualmente tiene el foco.

CurrentProject (o CodeProject)

Objeto que agrupa varias colecciones de objetos de Access específicos:

AllForms	Colección de todos los formularios de una base de datos de Access.
AllMacros	Colección de todas las macros de una base de datos de Access.
AllModules	Colección de todos los módulos de una base de datos de Access.
AllReports	Colección de todos los informes de una base de datos de Access.

CurrentData (o CodeData)

Objeto que agrupa varias colecciones de objetos de Access específicos:

AllTables	Colección de todas las tablas de una base de datos de Access.
AllQueries	Colección de todas las consultas de una base de datos de Access.

4. Ejemplos

a. Listar tablas de una base de Access

Este ejemplo muestra cómo ver la lista de tablas de la base de Access Northwind.accdb.

```
Sub Tablas_Access()
Dim oAppAccess As Access.Application
Dim i, j As Integer
  ' Inicia una sesión de Microsoft Access
Set oAppAccess = CreateObject("Access.application")
  ' Muestra la lista de tablas recorriendo
'   la colección "Alltables" del objeto "CurrentData"
With oAppAccess
   .OpenCurrentDatabase (ThisWorkbook.Path & "\Northwind.accdb")
   j = 2
   For i = 1 To .CurrentData.AllTables.Count - 1
       If Left(UCase(.CurrentData.AllTables(i).Name), 4) _
          <> "MSYS" Then
          Range("C" & j) = .CurrentData.AllTables(i).Name
          j = j + 1
       End If
   Next i
End With
  ' Sale de la aplicación Access
oAppAccess.Quit
  ' Reinicializa el objeto
Set oAppAccess = Nothing
End Sub
```

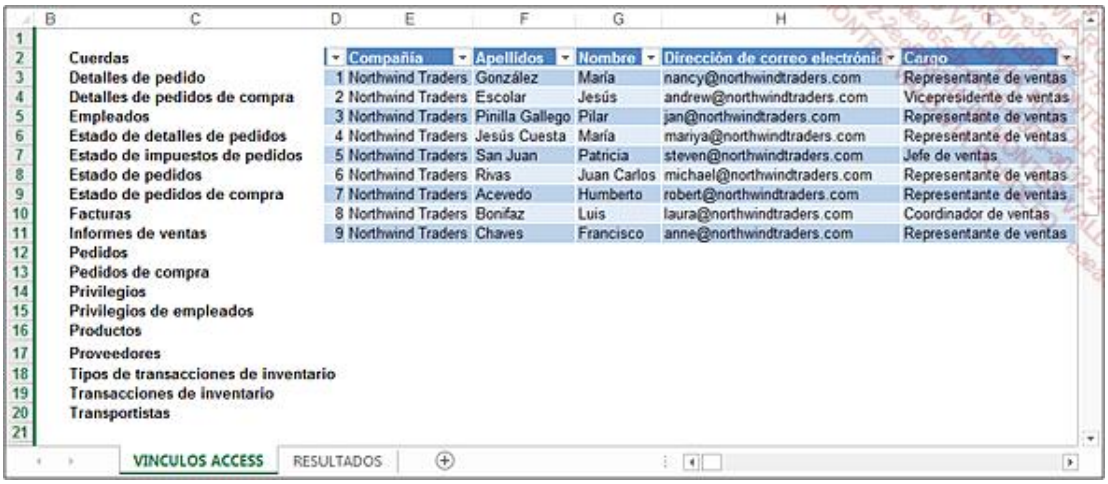
b. Mostrar una tabla de Access en Excel

El objeto **QueryTable** representa un rango de datos externo (base de datos, página web, etc.) contenida en una hoja de cálculo. Las propiedades de este objeto se explican en el capítulo dedicado a Internet.

```
Sub Mostrar_Tabla()
Dim i As Integer

'   Borra los datos de la última tabla mostrada
If ActiveSheet.ListObjects.Count > 0 Then
  For i = ActiveSheet.ListObjects(1).ListColumns.Count To 1 Step -1
    ActiveSheet.ListObjects(1).ListColumns(i).Delete
  Next i
End If

'   Muestra el contenido de la tabla seleccionada
'   haciendo una consulta en la base de datos "Northwind"
```

```
On Error GoTo 1:
If ActiveCell <> "" And ActiveCell.Column = 3 Then
   With ActiveSheet.ListObjects.Add(SourceType:=0, Source:=Array( _
        "OLEDB;Provider=Microsoft.ACE.OLEDB.16.0;", _
        "Data Source=" & ThisWorkbook.Path & "\Northwind.accdb;", _
        "Mode=Share Deny Write;Jet OLEDB:Engine Type=6"), _
        Destination:=Range("$D$2")).QueryTable
        .CommandType = xlCmdTable
        .CommandText = Array(ActiveCell)
        .RowNumbers = False
        .PreserveFormatting = True
        .BackgroundQuery = True
        .RefreshStyle = xlInsertDeleteCells
        .AdjustColumnWidth = True
        .PreserveColumnInfo = True
        .AdjustColumnWidth = True
        .PreserveColumnInfo = True
        .Refresh BackgroundQuery:=False
    End With
Else
    MsgBox "Debe seleccionar un nombre de tabla", vbExclamation
End If
On Error GoTo 0
Exit Sub
1:
MsgBox "La tabla seleccionada no se puede mostrar", _
       vbExclamation

End Sub
```

Resultado en Excel:



En este ejemplo, los datos importados se pueden actualizar en todo momento por medio del método **Refresh** del objeto **ListObject**.

```
Sub Actualiza_Datos()
   '   Actualiza todas las consultas
   '   de la hoja activa
```

```
   Dim oQTb As ListObject
   For Each oQTb In ActiveSheet.ListObjects
       If oQTb.SourceType = xlSrcQuery Then
          oQTb.Refresh
       End If
   Next
End Sub
```

c. Abrir una tabla o consulta de Access en un nuevo libro

El método **OpenDatabase** utilizado permite importar una tabla o el resultado de una consulta de Access en un nuevo libro de Excel.

```
Sub Abrir_Tabla()
   Dim sTableName As String
   Dim oWbk As Workbook
   Dim sWbkName As String
   Dim oFso As Object
   '  Controla la celda seleccionada
   If ActiveCell = "" Or ActiveCell.Column <> 3 Then Exit Sub
   '  Abre la tabla seleccionada
   sTableName = ActiveCell
   Set oWbk = Workbooks.OpenDatabase( _
       Filename:=ThisWorkbook.Path & "\Northwind.accdb", _
       CommandText:=Array(sTableName), CommandType:=xlCmdTable)
   '  Controla si el archivo de Excel ya existe
   sWbkName = ThisWorkbook.Path & "\" & sTableName & ".xlsx"
   If Dir(sWbkName) <> "" Then
      If MsgBox("Libro " & sTableName _
         & " ya existente, ¿desea eliminarlo?", _
         vbQuestion + vbYesNo) = vbYes Then
         Set oFso = CreateObject("Scripting.FileSystemObject")
         oFso.deletefile oWbkName
         Set oFso = Nothing
      Else
         Exit Sub
      End If
   End If
'  Guarda y cierra el libro creado
   oWbk.SaveAs sWbkName
   oWbk.Close
End Sub
```

Los datos importados también pueden actualizarse en todo momento.

Comunicación con Outlook desde Excel

1. Objetos de Outlook

AppointmentItem

Objeto que representa una reunión, una cita única o bien una cita o reunión periódicas.

ContactItem

Objeto que representa un contacto almacenado en una carpeta de contactos de Outlook.

Folder

Objeto que representa una carpeta de Outlook.

JournalItem

Objeto que representa una entrada del diario almacenada en una carpeta Diario (registro de todas las transacciones realizadas en Outlook durante un período dado).

MailItem

Objeto que representa un mensaje de correo electrónico (e-mail) dentro de una carpeta de mensajes de Outlook.

NoteItem

Objeto que representa una nota de Outlook dentro de una carpeta Notas.

PostItem

Objeto que representa una publicación de una carpeta pública que otros usuarios pueden examinar.

TaskItem

Objeto que representa una tarea almacenada en la carpeta Tareas de Outlook.

2. Acceso a los objetos de Outlook

a. Creación de un objeto (e-mail, contacto...) en Outlook

El método **CreateItem** del objeto **Application** permite crear una instancia de un objeto para un mensaje de correo o una cita.

Ejemplo:

Creación de un nuevo contacto en Outlook.

```
Sub Crea_Contacto()
Dim oAppOutlook As Outlook.Application
Dim oContacto As Outlook.ContactItem
' Inicia una sesión de Microsoft Outlook
Set oAppOutlook = CreateObject("outlook.application")
' Crea un nuevo contacto con un nombre y una dirección
Set oContacto = oApp Outlook.CreateItem(olContactItem)
oContacto.LastName = "PEREZ Patricia"
oContacto.MailingAddress = "pati.perez@yahoo.es"
' Guarda el contacto
oContacto.Save
End Sub
```

b. Acceso a los objetos (contactos, citas…) de Outlook

El método **GetDefaultFolder** del objeto **NameSpace** permite el acceso al conjunto de carpetas de Outlook. Este método usa un parámetro que permite especificar la carpeta que se desea usar.

Sintaxis del método:

```
expression.GetDefaultFolder(FolderType)
expression     Variable que representa un objeto NameSpace
FolderType      Constante que representa el tipo de carpeta que se devuelve:
               olFolderCalendar   carpeta Calendario
               olFolderContacts   carpeta Contactos
               olFolderDrafts     carpeta Borrador
               olFolderInbox      carpeta Bandeja de entrada
               olFolderJunk       carpeta Correo no deseado
               olFolderNotes      carpeta Notas
               olFolderOutbox     carpeta Bandeja de salida
               olFolderSentMail   carpeta Enviados
               olFolderTasks      carpeta Tareas
               olFolderToDo       carpeta Para hacer
```

Para tener acceso a la lista de objetos contenidos en una carpeta de Outlook (contactos, citas, etc.), proceda de la siguiente manera:

- Inicializar una variable objeto de tipo **NameSpace**.
- Usar el método **GetDefaultFolder** del objeto **NameSpace** para tener acceso a la carpeta deseada: el método devolverá un objeto **Folder**.
- Recorrer la colección **Items** del objeto **Folder** hasta llegar al conjunto de elementos de la carpeta (ciclo **For... Each**).

Ejemplo:

Este ejemplo da acceso al conjunto de reuniones en el calendario de Outlook.

```
Sub Reunion()
Dim oAppOutlook As Outlook.Application
Dim oNameSpace As Outlook.Namespace
```

```
Dim oFolder As Outlook.Folder
Dim oItem As AppointmentItem

' Abre una sesión de Microsoft Outlook
Set oAppOutlook = CreateObject("outlook.application")

' Recorre las reuniones en el calendario
Set oNameSpace = oAppOutlook.GetNamespace("MAPI")
Set oFolder = oNameSpace.GetDefaultFolder(olFolderCalendar)
For Each oItem In oFolder.Items
    Debug.Print oItem.Start & " " & oItem.End _
                & " " & oItem.Subject
Next
End Sub
```

3. Ejemplo de uso del objeto MailItem

Este ejemplo crea un mensaje de correo electrónico (e-mail) con un documento de Word como adjunto.

```
Sub Envia_Email()
Dim oAppOutlook As Outlook.Application
Dim oMail As Outlook.MailItem
'  Abre una sesión de Microsoft Outlook
Set oAppOutlook = CreateObject("outlook.application")
'  Crea un nuevo mensaje
Set oMail = oAppOutlook.CreateItem(olMailItem)
With oMail
   ' Título, Texto, Destinatarios, Adjuntos
   .Subject = "RESULTADO DE VENTAS"
   .Body = "Adjunto envío el documento con los resultados de
ventas del año 2014." _
           & Chr(13) & "Atentamente," & Chr(13) _
           & "Equipo comercial"
   .BodyFormat = olFormatHTML
   To = "laura@northwindtraders.com"
   To = "jaime@northwindtraders.com"
   .Attachments.Add ThisWorkbook.Path & "\Resultados_2014.docx"
   ' Envía el mensaje
   .Send
End With
' Cierra la aplicación Outlook
oAppOutlook.Quit
' Reinicializa el objeto
Set oAppOutlook = Nothing

End Sub
```

Objetos vinculados o incrustados

Es posible manipular los objetos vinculados o incrustados en Excel a través de la colección **OLEObjects** de objetos **OLEObject**.

El objeto emparentado puede ser un objeto **Worksheet** o un objeto **Chart**.

Ejemplo

El siguiente procedimiento modifica el tamaño y aplica un borde a todos los objetos incrustados en la hoja de cálculo Productos.

```
Sub Format_OLE()
Dim ole1 As OLEObject
For Each ole1 In Worksheets("Productos").OLEObjects
    With ole1
       .Height = 100
       .Width = 100
       .Border.Color = vbRed
       .Border.LineStyle = xlContinuous
       .Border.Weight = xlMedium
    End With
Next ole1
End Sub
```

1. Métodos del objeto OLEObject

Activate	Duplicate
Add	Item
BringToFront	Select
Copy	SendToBack
CopyPicture	UpDate
Cut	Delete

2. Propiedades del objeto OLEObject

AutoLoad

True si el objeto se carga automáticamente cuando se abre el libro que lo contiene.

OLEType

Devuelve `xlOLELink` o `xlOLEEmbed`.

Application	Name
AutoUpdate	Object
Border	Parent
Bottom	Right

Cell	Placement
Count	PrintObject
Creator	ProgId
Enabled	ShapeRange
Height	Shadow
Index	SourceName
Interior	Top
Left	TopLeftCell
LinkedCell	Visible
ListFillRange	Width
Locked	ZOrder

Métodos y propiedades relativos a los vínculos con Excel

1. Métodos y propiedades del objeto Workbook

ChangeLink (método)

Modifica un vínculo entre dos documentos.

```
ChangeLink (NAME,NEWNAME,type)
```

NAME	Nombre del vínculo que se debe modificar.
NEWNAME	Nuevo nombre del vínculo.
type	Tipo de vínculo que hay que devolver: xlLinkTypeExcelLinks, xlLinkTypeOLELinks.

LinkInfo (método)

Devuelve información acerca de la fecha y el estado de actualización del vínculo.

```
LinkInfo(NAME,LINKINFO,type,EditionRef)
```

NAME	Nombre del vínculo.
LINKINFO	Tipo de información que hay que devolver: (xlUpdateState o xlEditionDate).
type	Tipo de vínculo que se debe devolver: (xlLinkInfoOLELinks, xlLinkInfoPublishers o xllinkInfoSubscribers).
EditionRef	Si el vínculo es una edición, este argumento especifica la referencia a la edición.

LinkSources (método)

Devuelve una matriz Visual Basic con los vínculos del libro.

```
LinkSources(type)
```

type	Tipo de vínculo que hay que devolver: xlExcelLinks, xlOLELinks, xlPublishers, xlSubscribers.

OpenLinks (método)

Abre el documento de origen de un vínculo.

```
OpenLinks(NAME,readonly,type)
```

NAME	Nombre del vínculo.
Readonly	Apertura en modo de solo lectura (**True** o **False**).
type	Tipo del vínculo: (xlExcelLinks, xlOLELinks, xlPublishers o xlSubscribers).

SaveLinkValues (propiedad)

Propiedad que devuelve **True** si Microsoft Excel guarda los valores de los vínculos externos con el libro.

SetLinkOnData (método)

Crea un procedimiento que se ejecutará cada vez que se actualice un vínculo DDE.

```
SetLinkOnData(NAME,PROCEDIMIENTO)
```

NAME	Nombre del vínculo.
PROCEDIMIENTO	Nombre del procedimiento que se debe ejecutar cuando se actualiza el vínculo.

UpDateLink (método)

Actualiza un vínculo.

```
UpDateLink(NAME,type)
```

NAME	Nombre del vínculo.
type	Tipo de vínculo (xlLinkTypeExcelLinks o xlLinkTypeOLELinks).

2. Métodos y propiedades de otros objetos

AskToUpDateLinks (propiedad del objeto Application)

Propiedad que devuelve **True** si Excel invita al usuario a actualizar los vínculos cuando se abren los archivos que los contienen.

LinkSource (propiedad del objeto DocumentProperty)

Devuelve o define el origen de una propiedad de un documento personalizado con vínculos.

LinkToContent (propiedad del objeto DocumentProperty)

Propiedad que devuelve **True** si el valor de una propiedad de documento personalizada está vinculada al contenido del documento contenedor.

ActivateMicrosoftApp (método del objeto Application)

Activa una aplicación de Microsoft.

```
Application.ActivateMicrosoftApp(INDEX)
```

INDEX	Aplicación de Microsoft que hay que activar. Algunos ejemplos: xlMicrosoftWord, xlMicrosoftAccess, xlMicrosoft Project, etc.

Consultas por Internet

El siguiente procedimiento inserta en Excel una tabla procedente de un sitio Web:

- Haga clic en el botón 

situado en el grupo **Obtener datos externos** de la pestaña **Datos**.
- Escriba la dirección del sitio web deseado en la barra de direcciones del navegador.
- Seleccione las zonas que desea importar por medio del icono , ubicado en la parte superior de cada tabla; el icono se transformará en .
- Si lo desea, modifique las opciones de consulta haciendo clic en el botón **Opciones**.
- Haga clic en el botón de comando **Importar**.
- Seleccione el destino de la tabla en el cuadro de diálogo **Importar datos** y haga clic en **Aceptar**.

Ejemplo:

El siguiente ejemplo importa las cotizaciones de la Bolsa de Valores de Madrid desde el sitio www.bolsamadrid.es.

Resultado en Excel:

	A	B	C	D	E	F	G	H	I
1	Nombre	Últ.	% Dif.	Máx.	Mín.	Volumen	Efectivo (miles l)	Fecha	Hora
2	ABERTIS	13,73	0,11	13,755	13,56	975.270	13.353,29	29/02/2016	16:10
3	ACCIONA	69,75	2,03	70,15	68,49	239.995	16.712,50	29/02/2016	16:08
4	ACERINOX	10,1	3,02	10,25	9,604	1.406.439	14.106,83	29/02/2016	16:10
5	ACS	23,795	-0,23	23,85	23,24	483.906	11.435,50	29/02/2016	16:10
6	AENA	104,3	0,05	104,65	103,9	185.479	19.349,00	29/02/2016	16:09
7	AMADEUS	37,005	-0,3	37,175	36,67	1.863.902	68.946,50	29/02/2016	16:10
8	ARCELORMIT.	3,45	5,18	3,475	3,208	3.083.319	10.510,59	29/02/2016	16:10
9	BA.POPULAR	2,178	1,4	2,196	2,093	10.258.398	22.101,74	29/02/2016	16:10
10	BA.SABADELL	1,452	0,55	1,464	1,416	12.121.922	17.524,96	29/02/2016	16:10
11	BA.SANTANDER	3,703	0,46	3,742	3,591	31.017.914	113.954,76	29/02/2016	16:10
12	BANKIA	0,769	2,4	0,776	0,742	12.249.273	9.280,83	29/02/2016	16:09
13	BANKINTER	6,015	-0,03	6,075	5,92	944.534	5.664,22	29/02/2016	16:10
14	BBVA	5,793	0,43	5,847	5,603	24.300.897	139.618,66	29/02/2016	16:10
15	CAIXABANK	2,602	2,2	2,617	2,5	7.903.525	20.266,64	29/02/2016	16:10
16	DIA	4,614	-0,73	4,643	4,555	3.418.481	15.729,02	29/02/2016	16:09
17	ENAGAS	25,77	0,1	25,83	25,39	480.861	12.332,56	29/02/2016	16:08
18	ENDESA	16,52	-0,33	16,545	16,33	523.844	8.624,03	29/02/2016	16:10
19	FCC	6,355	0,75	6,457	6,283	575.275	3.672,32	29/02/2016	16:10
20	FERROVIAL	17,825	0,99	17,935	17,315	3.703.437	65.306,32	29/02/2016	16:10
21	GAMESA	17,26	1,47	17,3	16,8	7.952.187	136.375,19	29/02/2016	16:10
22	GAS NATURAL	16,01	-0,34	16,085	15,835	935.238	14.964,00	29/02/2016	16:10
23	GRIFOLS CL.A	20,065	1,31	20,36	19,425	532.699	10.690,57	29/02/2016	16:09
24	IAG	6,97	1,9	6,998	6,778	2.038.545	14.045,16	29/02/2016	16:10
25	IBERDROLA	5,931	0,07	5,943	5,861	5.019.117	29.638,78	29/02/2016	16:10
26	INDITEX	28,31	-0,68	28,46	27,975	1.462.779	41.292,43	29/02/2016	16:10
27	INDRA A	8,311	0,13	8,35	8,06	653.476	5.355,94	29/02/2016	16:10
28	MAPFRE	1,774	-1,28	1,794	1,753	4.896.158	8.699,12	29/02/2016	16:10
29	MEDIASET	9,743	3,22	9,76	9,35	1.200.210	11.507,09	29/02/2016	16:10
30	MERLIN	9,661	0,43	9,735	9,4	651.789	6.224,95	29/02/2016	16:10
31	OHL	5,268	3,46	5,286	4,963	2.066.849	10.727,34	29/02/2016	16:10
32	R.E.C.	72,79	0,33	72,86	71,54	221.064	15.959,90	29/02/2016	16:10
33	REPSOL	9,431	-0,47	9,568	9,221	7.511.102	70.636,73	29/02/2016	16:10
34	SACYR	1,617	11,36	1,637	1,437	11.121.371	17.371,23	29/02/2016	16:10
35	TEC.REUNIDAS	24,75	0,04	25,095	24,28	422.215	10.467,17	29/02/2016	16:10
36	TELEFONICA	9,173	0,53	9,29	8,941	10.842.132	98.780,33	29/02/2016	16:10

El objeto QueryTable

El objeto **QueryTable** (tabla de consulta) representa un rango de datos externos contenido en una hoja de cálculo. Estos datos pueden proceder de un origen externo, como una base de datos de Microsoft Access o SQL Server, o de datos extraídos con una **consulta web**.

El objeto **QueryTable** pertenece a la colección **QueryTables** del objeto **Worksheet**.

1. Propiedades del objeto QueryTable

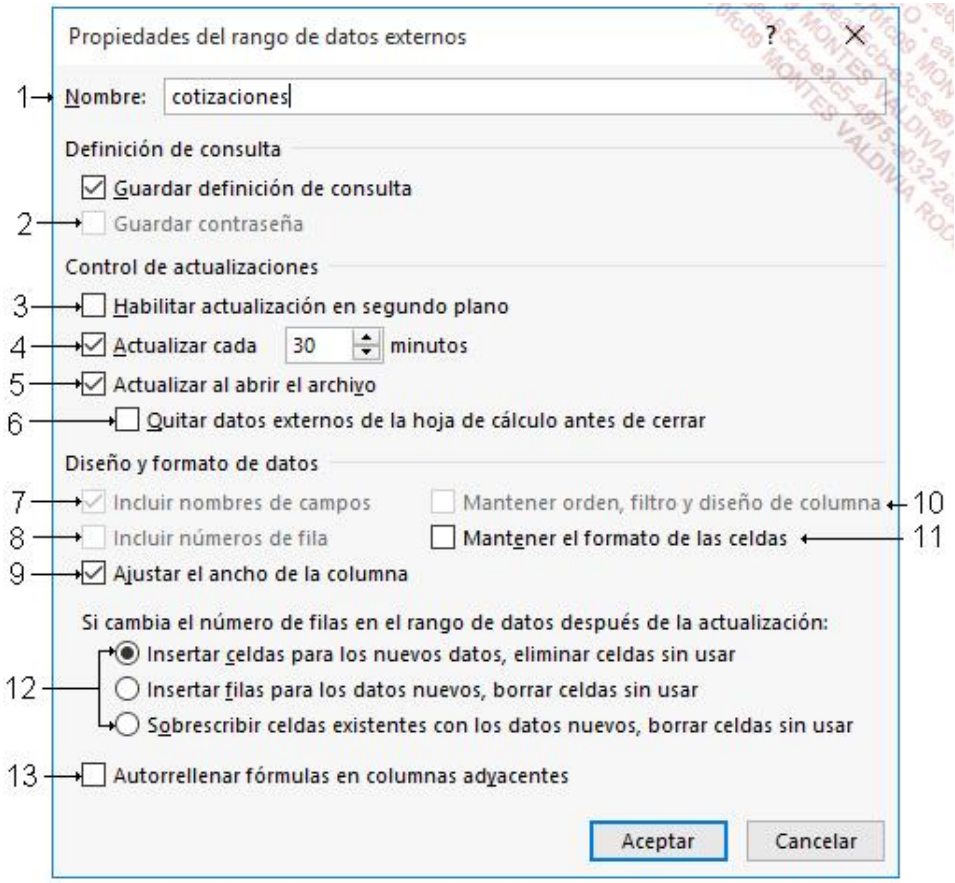
Propiedades del rango de datos externos

Ciertas propiedades del objeto **QueryTable** se corresponden con las opciones del cuadro de diálogo **Propiedades del rango de datos externos**.

Para mostrar este cuadro:

- Active el rango de datos externo haciendo clic en una de sus celdas.
- Seleccione el comando **Propiedades** del grupo **Conexiones** de la pestaña **Datos** (o haga clic en el botón derecho del ratón y seleccione **Propiedades del rango de datos** en el menú contextual).

Aparece la siguiente ventana:



N.°	Propiedades	Valores devueltos
1.	Name	Cadena de caracteres

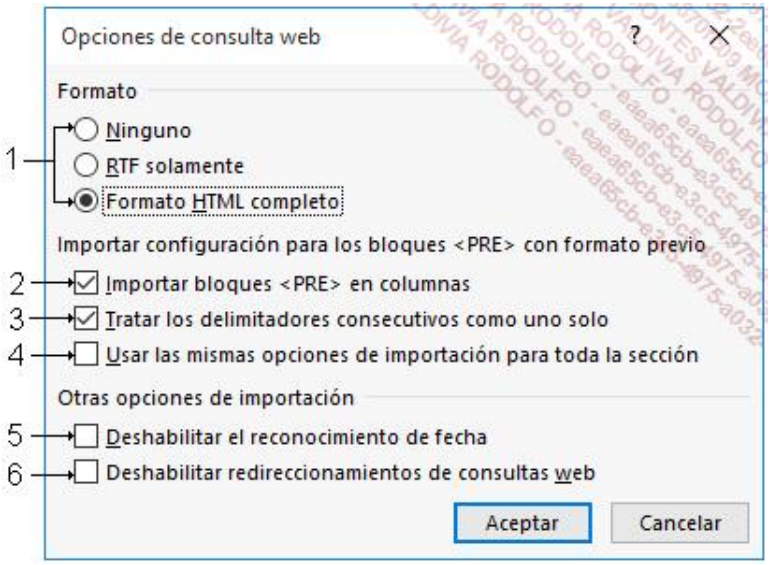
2.	SavePassword	Booleano
3.	BackgroundQuery	Booleano
4.	RefreshPeriod	Entero largo
5.	RefreshOnFileOpen	Booleano
6.	SaveData	Booleano
7.	FileNames	Booleano
8.	RowNumbers	Booleano
9.	AdjustColumnWidth	Booleano
10.	PreserveColumnInfo	Booleano
11.	PreserveFormatting	Booleano
12.	RefreshStyle	Constantes: **xlInsertDeleteCells** **xlInsertEntireRows** **xlOverwriteCells**
13.	FillAdjacentFormulas	Booleano

Opciones de las consultas Web

Ciertas propiedades del objeto **QueryTable** se corresponden con las opciones del cuadro de diálogo **Opciones de consulta web**.

Para mostrar este cuadro de diálogo:

- Coloque el cursor en el rango de datos externo de la consulta web.
- Haga clic en el botón derecho del ratón y seleccione la opción **Modificar consulta** en el menú contextual.
- Haga clic en el botón **Opciones**, arriba a la derecha del cuadro de diálogo **Modificar consulta web**.



N.°	Propiedades	Valores devueltos
1.	WebFormatting	Constantes: **xlWebFormattingNone** **xlWebFormattingRTF**

		xlWebFormattingAll
2.	WebPreFormattedTextToColumns	Booleano
3.	WebConsecutiveDelimiterAsOne	Booleano
4.	WebSingleBlockTextImport	Booleano
5.	WebDisableDateRecognition	Booleano
6.	WebDisableRedirections	Booleano

Otras propiedades usadas por las consultas web

Connection

Cadena de caracteres. URL del origen de datos web.

Destination

Objeto **Range**. Devuelve la celda de la esquina superior izquierda del rango de datos externos.

EditWebPage

Variant. URL de la página web.

EnabledEditing

Booleano. Indica si el usuario puede modificar la consulta.

EnableRefresh

Booleano. Indica si el usuario puede actualizar los datos del rango de datos externos.

MaintainConnection

Booleano. Indica si la conexión al origen de datos externos se mantiene después de la actualización y hasta que se cierra el libro.

Name

Nombre del rango de origen de datos externos.

QueryType

Constante. Indica el tipo de consulta que Microsoft Excel utiliza para rellenar el origen de datos externos (**xlWebQuery** para las consultas web).

ResultRange

Objeto **Range**. Representa el área de la hoja de cálculo ocupada por el origen de datos externos.

WebSelectionType

Constante. Establece un valor que determina qué parte de la página web se importará: toda la página, todas las tablas de la página o solo algunas de ellas.

Constantes: **xlWebEntirePage**, **xlWebAllTables**, **xlWebSpecifiedTables**

WebTables

Cadena de caracteres. Lista delimitada, por comas, de los nombres o números de índice de las tablas que hay que importar.

Métodos

CancelRefresh

Cancela todas las consultas en segundo plano del rango de datos externo especificado.

Delete

Elimina el rango de datos externo especificado.

Refresh

Actualiza el rango de datos externo.

ResetTimer

Restablece el temporizador automático de actualización utilizando el último intervalo establecido mediante la propiedad **RefreshPeriod** (frecuencia de actualización).

SaveAsODC

Guarda el origen de los datos externos como archivo de conexión de datos de Microsoft Office. (extensión .odc). No funciona con las consultas web.

2. Ejemplos

Creación de una consulta web

Este procedimiento crea una consulta web que importa una tabla especificada. Los argumentos permiten definir:

- La URL del sitio web.
- El nombre del rango de origen de datos externos.
- El destino en el libro (objeto **Range**).
- El número de la tabla que hay que importar.

```
Sub CreateWebQuery(strURL As String, strName As String, _
     oRngDesti As Range, sTable As Integer)

   '   Crea una consulta web
```

```
   '   Modifica las opciones de importar y las propiedades de
   '   la consulta
   '   Actualiza los datos por medio del método "Refresh"
   With ActiveSheet.QueryTables.Add(Connection:=strURL, _
        Destination:=oRngDesti)
       '   Propiedades de la consulta
       .Name = strName
       .FieldNames = True
       .RowNumbers = False
       .FillAdjacentFormulas = False
       .PreserveFormatting = True
       .RefreshOnFileOpen = True
       .BackgroundQuery = False
       .RefreshStyle = xlInsertDeleteCells
       .SavePassword = False
       .SaveData = True
       .AdjustColumnWidth = True
       .RefreshPeriod = 30
       '   Opciones de importar
       .WebSelectionType = xlSpecifiedTables
       .WebTables = sTable
       .WebFormatting = xlWebFormattingAll
       .WebPreFormattedTextToColumns = True
       .WebConsecutiveDelimitersAsOne = True
       .WebSingleBlockTextImport = False
       .WebDisableDateRecognition = False
       .WebDisableRedirections = False
       '   Actualiza los datos
       .Refresh BackgroundQuery:=False

       '   Corrige los datos
       Cells.Replace What:="Â", Replacement:="", LookAt:=xlPart, _
           SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
           ReplaceFormat:=False
       '   Da formato a los datos de la columna F
       Selection.NumberFormat = "#,##0"
   End With
End Sub
```

Ejemplo de llamada del procedimiento: importar la tabla número 13 de la página web http://www.bolsamadrid.es/esp/aspx/Mercados/Precios.aspx?indice=ESI100000000 en la hoja "Acciones" del libro activo.

```
Sub ImportarCotizaciones()
Dim oRng As Range
Dim sUrl As String
Dim sName As String
Dim i As Integer

'   Elimina las consultas existentes y borra los datos
ThisWorkBook.WorkSheets("Acciones").Select
If ActiveSheet.QueryTables.Count > 0 Then
   For i = ActiveSheet.QueryTables.Count To 1 Step -1
       ActiveSheet.QueryTables(i).Delete
```

```
       Range("A1:X100").Delete
   Next i
End If

'   Llama al procedimiento CreateQueryWeb
sUrl = "URL;http://www.bolsamadrid.es/esp/aspx/Mercados/Precios.aspx?indice=ESI100000000"
sName = "accind1_1"
Set oRng = ActiveWorkbook.Worksheets("Acciones").Range("A1")
CreateWebQuery sUrl, sName, oRng, 13
End Sub
```

Publicación de páginas web

Se puede crear una página web en formato html a partir de un libro, de una hoja de cálculo, de un gráfico o de un rango de celdas y actualizar la página al guardar el libro.

Para publicar una página web desde de Excel, siga estas instrucciones:

- Seleccione la opción **Guardar** de la pestaña **Archivo**.
- Despliegue las opciones de la lista **Tipo** y seleccione el tipo de archivo **Página web (*.htm; *.html)**.
- Haga clic en el botón **Publicar** para seleccionar los elementos que se van a publicar y las opciones de publicación (marque las opciones **Volver a publicar automáticamente cada vez que se guarde el libro** y **Abrir página en el explorador web**).
- Haga clic otra vez en el botón **Publicar** para crear la página web.

  La página se creará y aparecerá en su navegador.

En VBA, para asociar un elemento de un libro a una página web, se debe crear un objeto **PublishObject** (usando el método **Add** de la colección **Publish Objects**). Para publicar la página web, luego deberá usar el método Publish del objeto **PublishObject**.

1. Asociación de un elemento de libro a una página web

Sintaxis

```
PublishObjects.Add(SourceType, FileName, Sheet, Source,
HtmlType, DivID, Title)
```

Solamente son obligatorios los argumentos `SourceType` y `FileName`.

`PublishObjects`	Expresión que devuelve una colección **PublishObjects**.
`SourceType`	Tipo de elemento que se va a publicar (xlSourceSheet, SourceRange, xlSourceworkbook, xlSourceChart, xlSourceQuery, xlSourcePivotTable…).
`Sheet`	Nombre de la hoja de cálculo guardada como página web.
`Source`	Nombre del elemento que hay que publicar si se trata de un gráfico, de un informe de tabla dinámica o de una tabla de consulta.
`HtmlType`	Indica si el elemento publicado se guarda como componente de Microsoft Office Web interactivo o como texto e imágenes estáticas.
`DivId`	Identificador único usado en la etiqueta DIV de HTML para identificar el elemento en la página web.
`Title`	Título de la página web.

2. Publicación de la página web

Sintaxis

```
PublishObject.Publish(Create)
```

`PublishObject`	Expresión que devuelve un objeto **PublishObject** o una colección

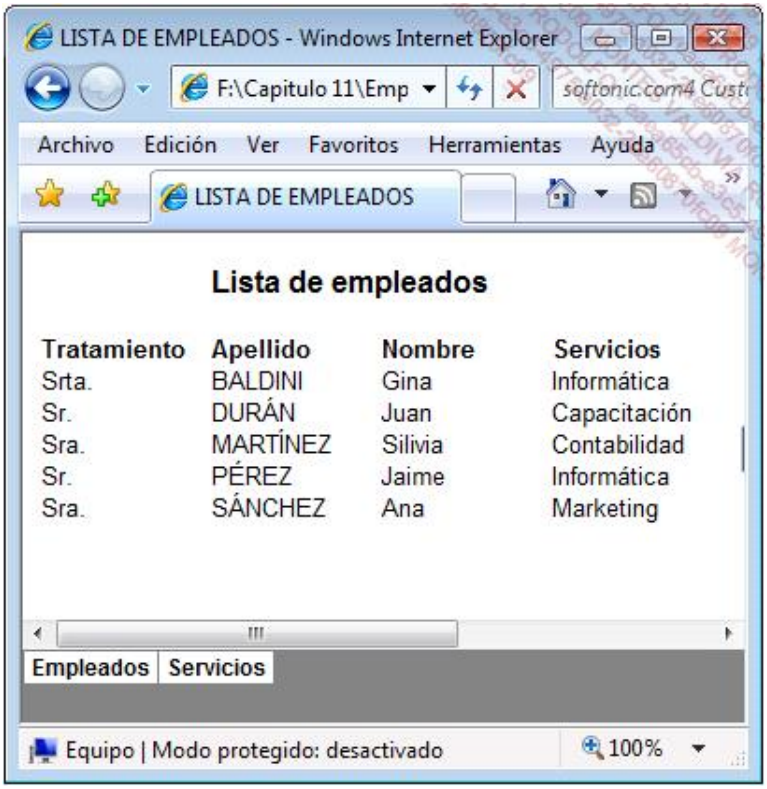
	PublishObjects.
Create	Si este argumento tiene el valor **True** y el archivo HTML ya existe, se sustituye. El valor por defecto es **False**.

3. Ejemplo

Publicación del libro Empleados: este libro tiene dos hojas de cálculo: Empleados y Servicios.

```
Sub Publicar()
   Dim WebPage As PublishObject

   '   Crea un objeto para guardar una página web
   Set WebPage = ActiveWorkbook.PublishObjects.Add _
       (xlSourceWorkbook, ThisWorkbook.Path & "\Empleados.html" _
       , , , xlHtmlStatic, , "LISTA DE EMPLEADOS")

   '   Publica la página
   With WebPage
      .Publish (True)
      .AutoRepublish = True
   End With

End Sub
```

Vista previa de la página web Empleados.html creada en Internet Explorer.

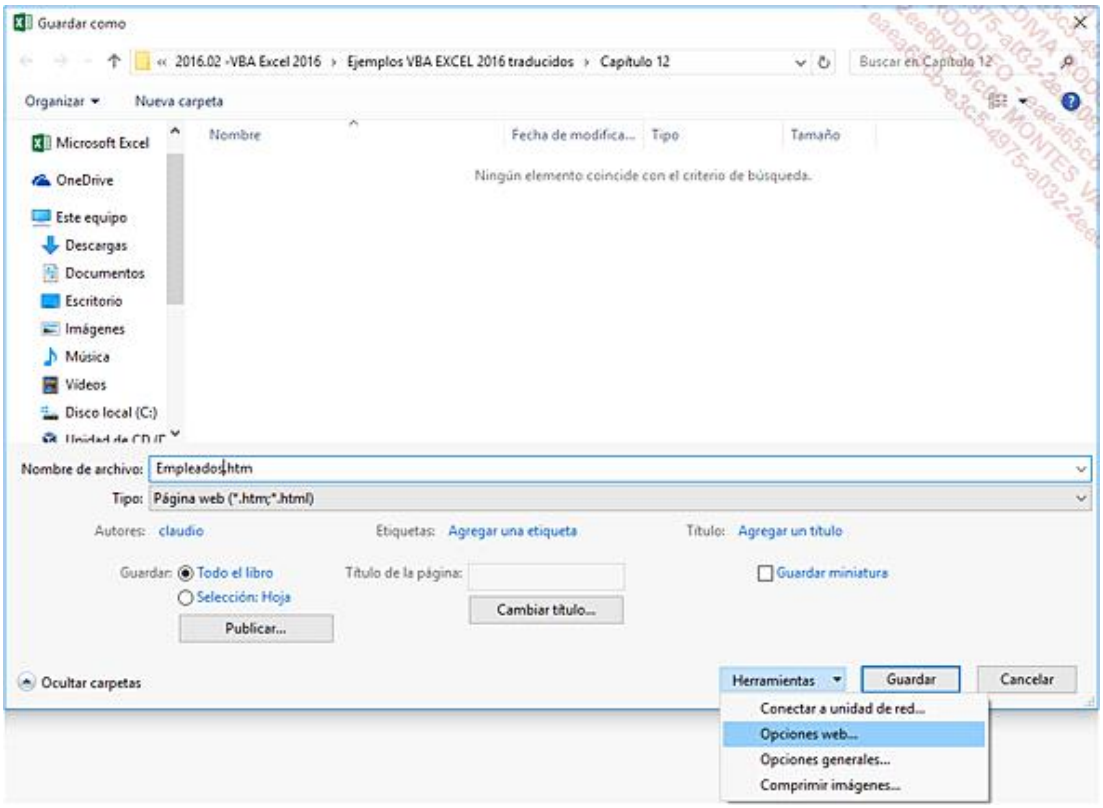


Los objetos WebOptions y DefaultWebOptions

Los objetos **WebOptions** y **DefaultWebOptions** contienen los atributos usados por Excel al guardar un documento como página web.

El objeto **DefaultWebOptions** contiene las opciones web por defecto de la aplicación Excel: su contenedor es el objeto **Application**. Las propiedades del objeto corresponden a los atributos accesibles al hacer clic en el botón **Opciones web** de la categoría **Opciones avanzadas** de las opciones de Excel.

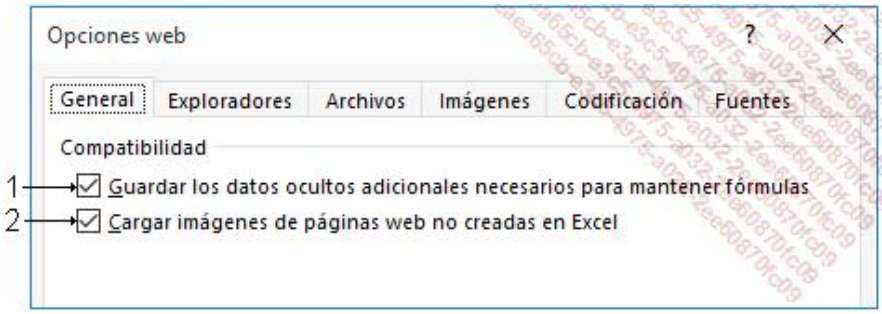
El objeto **WebOptions** contiene las opciones web del libro especificado: su contenedor es el objeto **Workbook**. Las propiedades del objeto corresponden a los atributos accesibles desde el botón **Herramientas/Opciones web** al guardar un libro en formato "Página web".



Ciertas propiedades son comunes a ambos objetos; las demás son específicas al objeto **DefaultWebOptions**.

1. Propiedades

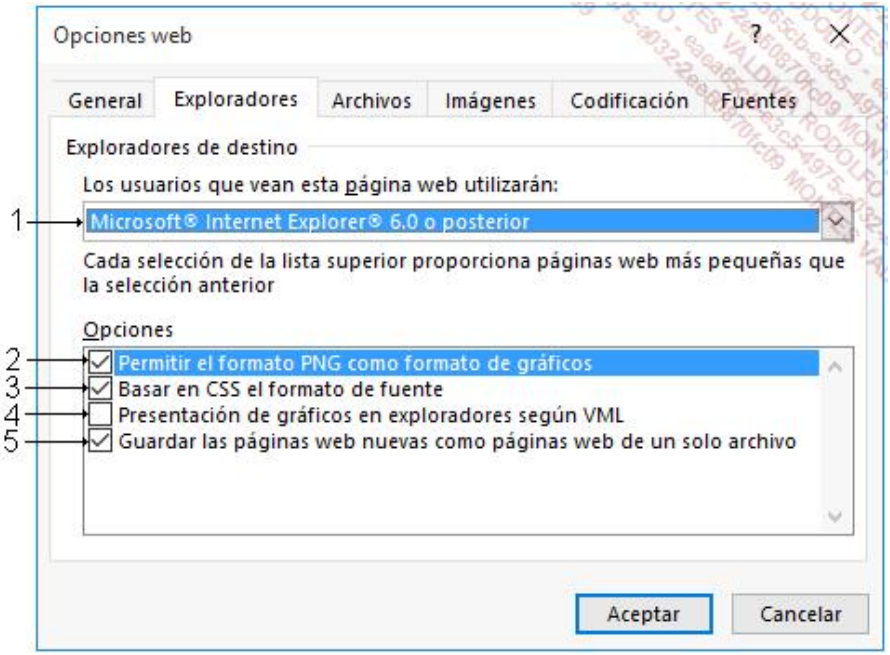
a. Opciones de la pestaña General



Propiedades del objeto DefaultWebOptions

1.	SaveHiddenData	Booleano
2.	LoadPictures	Booleano

b. Opciones de la pestaña Exploradores



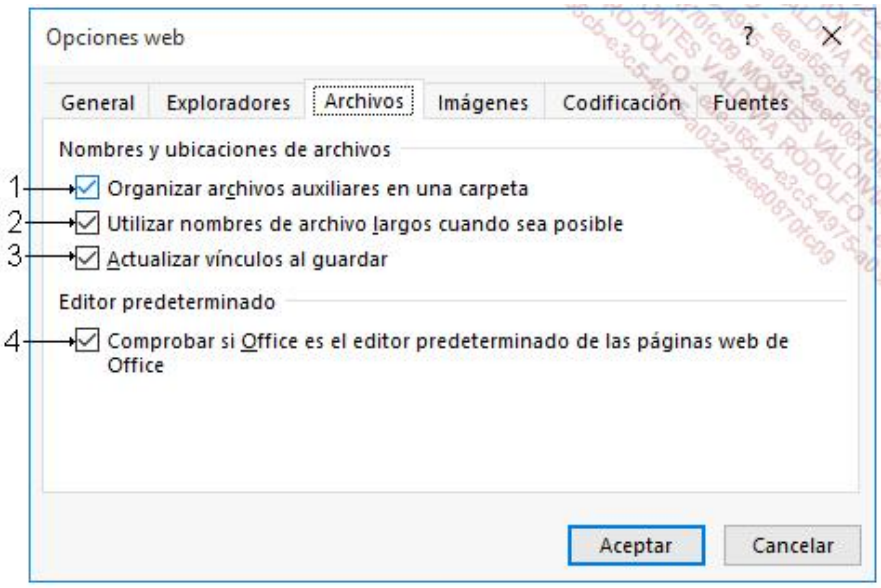
Propiedades comunes a ambos objetos

1.	TargetBrowser	Constantes: **msoTargetBrowserIE4 (IE5 o IE6)** **msoTargetBrowserV3 (o V4)**
2.	AllowPNG	Booleano
3.	RelyOnCSS	Booleano
4.	RelyOnVML	Booleano

Propiedades del objeto DefaultWebOptions

5.	SaveNewWebPagesAsWebArchives	Booleano

c. Opciones de la pestaña Archivos



Propiedades comunes a ambos objetos

1.	OrganizeInFolder	Booleano
2.	UseLongFileNames	Booleano

Propiedades del objeto DefaultWebOptions

3.	UpdateLinkOnSave	Booleano
4.	CheckIfOfficeIsHTMLEditor	Booleano

d. Otras propiedades

Propiedades comunes a ambos objetos

Encoding

Constante. Tipo de codificación usada (corresponde a la opción seleccionada en la lista desplegable de la pestaña **Codificación**).

FolderSuffix

Cadena de caracteres. Sufijo del archivo usado por Excel al guardar un documento como página web.

PixelsPerInch

Entero largo. Densidad (cantidad de píxeles por pulgada) de las imágenes y celdas de tablas de una página web.

ScreenSize

Constante. Tamaño mínimo de pantalla óptimo (ancho por alto, en píxeles) que se debe utilizar al ver

el documento con un explorador web (ejemplo: msoScreenSize 800 x 600, msoScreenSize 1024 x 768, etc.).

Propiedades del objeto DefaultWebOptions

AlwaysSaveInDefaultEncoding

Booleano. Indica si se usa la codificación predeterminada al guardar una página web.

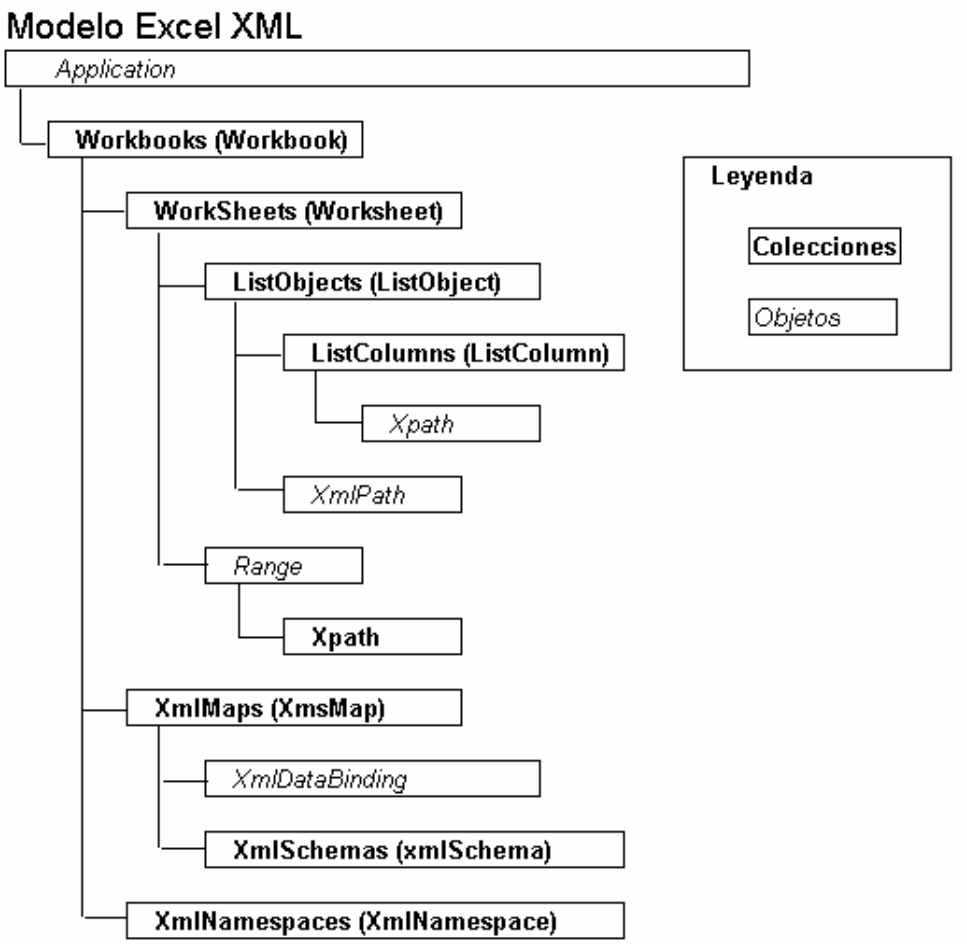
2. Método del objeto WebOptions

UseDefaultFolderSuffix

Establece el sufijo de carpeta para el libro especificado, correspondiente al idioma que haya seleccionado o instalado.

Importar, exportar y asignar archivos XML

El modelo de objeto de Excel XML presentado aquí describe los nuevos objetos que permiten cargar datos en formato XML en los libros de Excel.



1. Colecciones

ListObjects

Colección de las listas de una hoja de cálculo de Excel. Estas listas pueden contener datos XML.

ListColumns

Colección de las columnas de una lista de Excel.

XmlMaps

Colección de los objetos XML de un libro. Estos objetos se usan para controlar la relación entre los rangos de celdas de Excel y los elementos de un esquema XML.

XmlSchemas

Colección de los esquemas XML contenidos en un objeto XML.

XmlNamespaces

Colección de los espacios de nombres XML incluidos en el libro especificado.

2. Métodos del objeto Workbook

XmlImport

Importa un archivo de datos XML.

Ejemplo:

```
Sub ImportXML()
Dim oMapEmpleados As XmlMap

'   Importa el archivo empleados.xml en la hoja activa
ActiveWorkbook.XmlImport Url:=ActiveWorkbook.Path _
    & "\Empleados.xml", ImportMap:=MapEmpleados, _
    Overwrite:=True, Destination:=Range("A1")
oMapEmpleados.Name = "Empleados"
End Sub
```

SaveAsXMLData

Exporta los datos de un objeto XMLMap a un archivo XML.

Ejemplo:

```
Sub ExportXML()

'   Exporta el XMLMap al archivo Clientes2.xml
ActiveWorkbook.SaveAsXMLData _
      Filename:=ActiveWorkbook.Path & "\Clientes2.xml", _
      Map:=ActiveWorkbook.XmlMaps(1)
End Sub
```

3. Eventos del objeto Workbook

AfterXmlExport

Ocurre después de exportar un archivo XML.

AfterXmlImport

Ocurre después de importar un archivo XML.

BeforeXmlExport

Ocurre antes de exportar un archivo XML.

BeforeXmlImport

Ocurre antes de importar un archivo XML.

Ejemplo:

```
Private Sub Workbook_AfterXmlImport(ByVal Map As XmlMap, _
       ByVal IsRefresh As Boolean, ByVal Result As XlXmlImportResult)
If Result = xlXmlImportSuccess Then
   MsgBox "Importación exitosa"
Else
   MsgBox "Problema con la importación del archivo " & Map.Name
End If
End Sub
_____________________________________________________________________
Private Sub Workbook_BeforeXmlImport(ByVal Map As XmlMap, _
     ByVal Url As String, ByVal IsRefresh As Boolean, Cancel As Boolean)
If MsgBox("¿Desea importar el archivo " & Url & "?", _
   vbQuestion + vbYesNo) = vbNo Then
       MsgBox "Importación cancelada", vbExclamation
       Cancel = True
End If
End Sub
```

4. Métodos del objeto XmlMap

Delete

Permite quitar un XMLMap.

Ejemplo: ActiveWorkbook.XmlMaps(1).Delete

Import

Permite actualizar un mapeo XML a partir de un archivo XML.

Ejemplo: ActiveWorkbook.XmlMaps(1).Import "C:\Empleados.xml"

Export

Exporta a un archivo de datos XML el contenido de las celdas asignadas al objeto **XmlMap** especificado.

Ejemplo: ActiveWorkbook.XmlMaps(1).Export "C:\Empleados.xml"

El objeto HyperLink

El objeto **HyperLink** representa un **hipervínculo** contenido en una hoja de cálculo, un rango de celdas o un gráfico.

El objeto **HyperLink** pertenece a la colección **HyperLinks** de los objetos contenedores **Range**, **Workbook** y **Chart**.

1. Propiedades

Address

Cadena de caracteres. Dirección de la celda que contiene el hipervínculo especificado.

EmailSubject

Cadena de caracteres. Texto de la línea del asunto del correo electrónico del hipervínculo especificado (propiedad usada con los hipervínculos de los mensajes de correo electrónico).

Name

Cadena de caracteres. Nombre del hipervínculo.

Range

Objeto **Range**. Rango vinculado al hipervínculo.

ScreenTip

Cadena de caracteres. Texto de la etiqueta informativa del hipervínculo especificado.

Shape

Objeto **Shape**. Forma vinculada al hipervínculo especificado.

SubAddress

Cadena de caracteres. Ubicación dentro del documento a la que hace referencia el hipervínculo.

TextToDisplay

Cadena de caracteres. Texto que se mostrará para el hipervínculo especificado.

Type

Entero largo. Tipo del hipervínculo especificado.

2. Métodos

AddToFavorites

Agrega un acceso directo al hipervínculo en la carpeta Favoritos.

CreateNewDocument

Crea un nuevo documento vinculado al hipervínculo especificado.

Delete

Elimina el hipervínculo especificado.

Follow

Carga el documento de destino asociado al hipervínculo especificado y muestra el documento en la aplicación apropiada.

Presentación de las API

La interfaz de programación de Windows **API** (*Application Programming Interface*) ofrece funciones que le permiten controlar los aspectos más ínfimos del sistema. Puede extender y personalizar sus aplicaciones de Excel llamando a funciones de Windows API desde VBA. Si bien Excel continúa evolucionando y su lenguaje de programación nativo (VBA) integra cada vez más funciones de sistema, para ciertas tareas se deben usar las funciones API.

Una API es una serie de funciones que se pueden usar para trabajar con un componente, una aplicación o un sistema operativo. Se compone generalmente de uno o más archivos DLL (*Dynamic Link Library* o biblioteca de vínculos dinámicos).

La API más usada es la **API de Windows**, que incluye las DLL que forman el sistema operativo Windows. Cada aplicación de Windows interactúa directa o indirectamente con la API de Windows. Esto garantiza un comportamiento coherente de todas las aplicaciones que funcionan bajo Windows.

Las DLL de la aplicación Windows más usadas son las siguientes:

Kernel32.dll	Funciones de bajo nivel del sistema operativo, tales como la administración de memoria y de recursos.
User32.dll	Funciones de administración de Windows, tales como el tratamiento de mensajes, relojes, menús y comunicación.
GDI32.dll	Biblioteca GDI (*Graphics Device Interface*), que contiene las funciones de salida hacia los periféricos (gráficos, contexto de visualización y administración de fuentes).

Existen otras API, como por ejemplo la interfaz **MAPI** (*Mail Application Programming Interface*), que permite escribir **aplicaciones de correo electrónico**.

Para obtener más datos sobre las funciones de la API de Windows, existen dos fuentes de información:

- **El Visor de API** (archivo ejecutable ApiLoad.exe, incluido en Microsoft Office 2007 Developer y en Microsoft Visual Basic) permite mostrar las constantes, las declaraciones y los tipos de API. Los elementos seleccionados se pueden copiar del Visor de API hacia las aplicaciones VBA. La información del Visor de API se almacena en archivos de texto (win32api.txt, mapi32.txt, etc.) que luego se pueden exportar a una base de datos de Access para facilitar la consulta posterior.
- **La plataforma Microsoft SDK** (*Software Development Kit*) contiene la documentación completa de la API de Windows y está disponible gratuitamente en el sitio Microsoft Developer Network.

Llamar a una función de la API de Windows

Para llamar a una función de la API de Windows, se debe declarar con la instrucción **Declare** en la sección **Declaraciones** de un módulo del proyecto (generalmente un módulo específico de los procedimientos generales de la aplicación).

1. Sintaxis de la instrucción Declare

En las versiones de 32 bits de Visual Basic, se deben respetar las minúsculas y mayúsculas en los nombres de funciones y procedimientos de las DLL.

```
[Public|Private] Declare Sub <nom_proc> Lib "<nom_DLL>"_

[Alias "<nom_alias>"] [([lista_argumentos])]
[Public|Private] Declare Function <nom_fonc> Lib "<nom_DLL>"_
[Alias "<nom_alias>"] [([lista_argumentos])] [As <type>]
```

nom_proc, nom_fonc	Nombre del procedimiento o de la función usada en Visual Basic.
nom_DLL	Nombre de la DLL.
nom_alias	Nombre del procedimiento o de la función en la DLL.
lista_argumentos	[Optional][ByVal | ByRef][ParamArray] nomvariable[()][As type] (ver capítulo El lenguaje VBA).

Ciertas DLL no proveen un nombre para sus procedimientos y funciones, sino un número ordinal. La declaración de estos procedimientos o funciones usa la misma sintaxis, pero se debe definir el número ordinal a nivel del alias con un carácter numeral (#), seguido del número (ejemplo: Alias "#52").

2. Paso de argumentos

Las funciones y procedimientos de las DLL están escritos principalmente en lenguaje C y hacen, por lo tanto, referencia a su sintaxis. Por eso, el paso de argumentos a un procedimiento o una función de una DLL desde Visual Basic no siempre es simple. A nivel de las DLL que usan la sintaxis de lenguaje C, todos los argumentos se pasan por valor, salvo las matrices.

Las cadenas en lenguaje C se consideran como matrices de caracteres.

Ciertos argumentos de procedimientos de DLL pueden aceptar distintos tipos de datos (a semejanza de los Variant); se deben declarar como tipo Any (ejemplo: `variable As Any`). Para este tipo de argumento, Visual Basic considera que él mismo se pasa sistemáticamente por referencia; si hay que pasarlo por valor, se debe explicitar a nivel de la llamada (y no de la declaración), del procedimiento o de la función con la mención `ByVal`.

Por defecto, Visual Basic pasa los argumentos por referencia.

Lista de funciones API de Windows

Esta lista incluye las funciones de la API de Windows usadas habitualmente. Los ejemplos de uso de algunas de estas funciones se describen en la siguiente sección.

GetWindowsDirectory

Devuelve la ruta completa del directorio de Windows.

GetSystemDirectory

Devuelve la ruta completa del directorio de sistema de Windows.

GetSystemInfo

Devuelve una serie de datos sobre el sistema. Estos datos se almacenan en una estructura de tipo SYSTEM_INFO.

GetActiveWindow

Devuelve el identificador de la ventana activa.

FindExecutable

Retorna el nombre y la ruta de la aplicación asociados a un archivo (aplicación asociada a la extensión del archivo).

FindWindow

Devuelve el identificador de la ventana en función de su nombre y de la clase a partir de la cual se definió.

GetPrivateProfileString

Devuelve una opción de un archivo Ini a partir de un nombre de sección y de clave.

GetTempPath

Devuelve la ruta a la carpeta temporal del sistema.

GetUserName

Devuelve el nombre del usuario.

WNetGetUser

Devuelve el nombre de login de red.

SetFocus

Pone el foco de entrada en la ventana referenciada por su identificador.

Ejemplos de uso de funciones API de Windows

1. Recuperar el directorio Windows

Declaración de la función API:

```
' Esta función API devuelve el directorio Windows '
Private Declare Function GetWindowsDirectory _
           Lib "kernel32" Alias "GetWindowsDirectoryA" _
          (ByVal lpWindowsDir As String, _
           ByVal lpWindowsHeight As Long) _
           As Long
```

Llamada de la función API:

```
Function GetWinPath() As String

' Esta función VBA devuelve el directorio Windows
Dim StrResult As String
Dim StrProfile As String

StrResult = String(255, " ")
StrProfile = GetWindowsDirectory(StrResult, 255)

  ' Corta la cadena al primer carácter nulo
If StrProfile <> "" Then
   StrResult = Trim(StrResult)
   GetWinPath = Left(StrResult, InStr(1, StrResult, vbNullChar) - 1)
Else
   GetWinPath = ""
End If

End Function
```

2. Abrir la calculadora de Windows

Este ejemplo prueba si la calculadora de Windows está activa, y la inicia si no lo está.

Aquí se usan dos funciones API: la primera, **FindWindow**, busca la ventana "Calculadora"; la segunda, **FindExecutable**, busca la ubicación del archivo Calc.exe.

Declaración de la función API:

```
'   Esta función API busca una ventana
Public Declare Function FindWindow Lib "user32" Alias _ "FindWindowA" (ByVal lpClassName As String, ByVal  _
lpWindowName As String) As Long

'   Esta función API busca un archivo ejecutable
Public Declare Function FindExecutable Lib "shell32.dll" _
```

```
Alias "FindExecutableA" (ByVal lpFile As String, _
ByVal lpDirectory As String, ByVal lpResult As String) As Long
```

Llamar a las funciones API:

```
Private Function Calculadora() As Boolean
   Dim strClassName As String
   Dim strWindowName As String
   Dim Hwnd As Long
   Dim blnExe As Boolean
   Dim strRepCalc As String
   Dim strResult As Long

   ’   Inicialización
   Calculadora = False

   ’   Busca una ventana "Calculadora" activa
   strClassName = vbNullString
   strWindowName = "Calculadora"
   Hwnd = FindWindow(strClassName, strWindowName)
   ’   Si no la encuentra, busca el directorio
   ’   de Calc.exe y ejecuta la aplicación
   If Hwnd = 0 Then
      strRepCalc = String$(255, 0)
      strResult = FindExecutable("Calc.exe", "C:\", strRepCalc)
      If strResult = 0 Then
         MsgBox "Aplicación Calculadora no encontrada"
         Exit Function
      End If
      blnExe = Shell(strRepCalc, vbNormalFocus)
      If Not blnExe Then
         MsgBox "No se pudo ejecutar la aplicación"
         Exit Function
      End If
   Else
      MsgBox "La aplicación Calculadora ya está activa"
      Exit Function
   End If

   Calculadora = True
   MsgBox "Se inició la aplicación Calculadora"
End Function
```

El objeto FileSystemObject

El objeto **FileSystemObject** proporciona acceso al **sistema de archivos de un equipo**. Permite especialmente buscar, crear, eliminar y mover archivos o carpetas.

1. Métodos

Métodos relativos a los archivos

CopyFile

Copia uno o más archivos de un lugar a otro.

CreateTextFile

Crea un nombre de archivo especificado y devuelve un objeto TextStream que se puede utilizar para leer o escribir en un archivo.

DeleteFile

Elimina un archivo especificado.

FileExists

Devuelve un valor booleano que indica si existe el archivo especificado.

MoveFile

Mueve uno o más archivos de un lugar a otro.

OpenTextFile

Abre el archivo especificado y devuelve un objeto **TextStream** que se puede utilizar para leer el archivo o agregar datos.

Métodos relativos a las carpetas

CopyFolder

Copia una carpeta de un lugar a otro.

CreateFolder

Crea una carpeta.

DeleteFolder

Elimina una carpeta especificada junto con su contenido.

FolderExists

Devuelve un valor booleano que indica si existe la carpeta especificada.

MoveFolder

Mueve una o más carpetas de un lugar a otro.

Métodos relativos a las unidades de disco

DriveExists

Devuelve un valor booleano que indica si existe la unidad especificada.

GetDrive

Devuelve un objeto Drive que corresponde a la unidad para la ruta especificada.

GetDriveName

Devuelve una cadena que contiene el nombre de la unidad para una ruta especificada.

2. Propiedades

Drives

Devuelve una colección formada por los objetos **Drive** disponibles en el equipo local.

3. Ejemplo: copia de archivos de Excel

El siguiente ejemplo busca todos los archivos de Excel que hay en la misma carpeta que el libro activo y los copia en el directorio "C:\Archivos Excel".

```
Sub CopiaArchivos()
Dim oFso As Object
Dim strFile As String
'   Crea un nuevo directorio (salvo que ya exista)
Set oFso = CreateObject("Scripting.FileSystemObject")
If Not oFso.folderExists("C:\Archivos Excel") Then
   oFso.createfolder ("C:\Archivos Excel")
End If

'   Busca los archivos de Excel y los copia en el directorio
strFile = Dir(ThisWorkbook.Path & "\*.xls*", vbNormal)
Do While strFile <> ""
    oFso.copyfile ThisWorkbook.Path & "\" & strFile, "C:\Archivos Excel\"
    '   Archivo siguiente
    strFile = Dir
Loop
End Sub
```

Presentación general

La aplicación de Excel presentada en este capítulo permite administrar presupuestos realizados con Excel.

Las principales funcionalidades de esta aplicación son:

- Creación de un nuevo presupuesto a partir de un modelo.
- Búsqueda de presupuestos en función de criterios (cliente, fecha), con la posibilidad de abrir o de eliminar uno o más presupuestos.
- Creación de nuevos clientes y búsqueda de clientes.

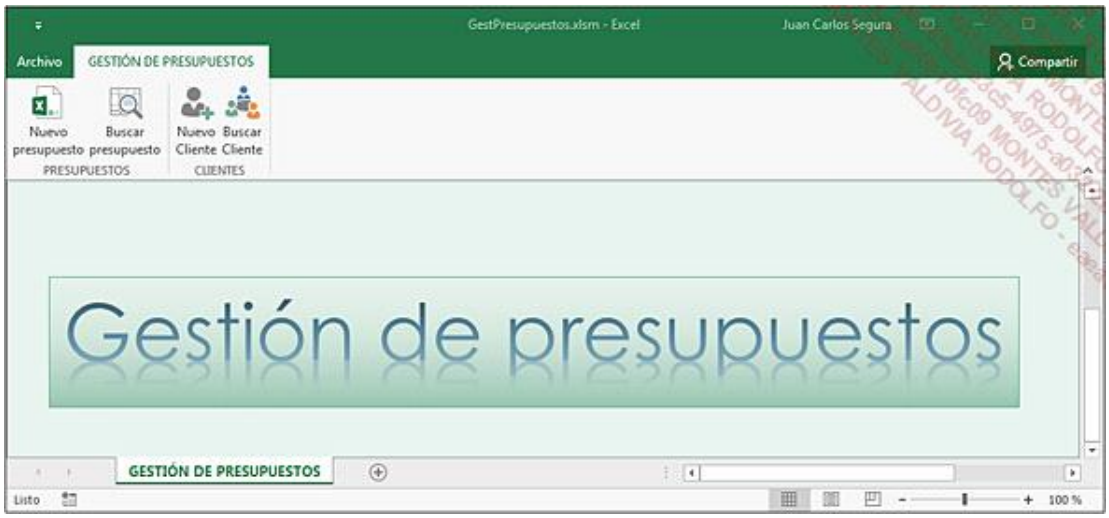
Todos los archivos necesarios para la aplicación se deben instalar en el mismo directorio. Estos archivos se entregan con los ejemplos del libro y son los siguientes:

- El archivo que contiene el código de la aplicación de Excel: GestPresupuesto.xlsm.
- La base de datos de Access que incluye la tabla de clientes y los formularios de búsqueda y de creación de un cliente: Clientes.accdb.
- La plantilla de Excel que sirve de base para la generación de presupuestos: Presupuesto.xltx.

Los presupuestos generados se presentan como archivos de Excel denominados de la siguiente manera: fecha de creación en la forma AAAAMMDD y extensión xlsx (ejemplo: 20130410.xlsx). Estos se generan en un subdirectorio de la aplicación que lleva el nombre del cliente.

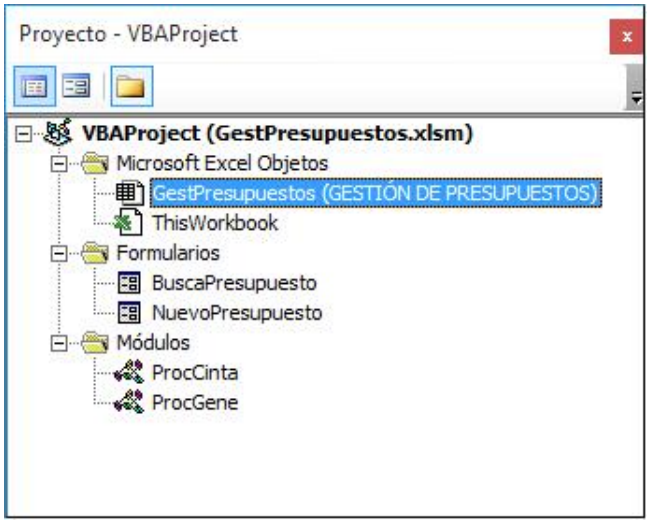
Descripción de la aplicación GestPresupuesto

Al abrir el archivo GestPresupuesto.xlsm, la presentación general de la aplicación es la siguiente:

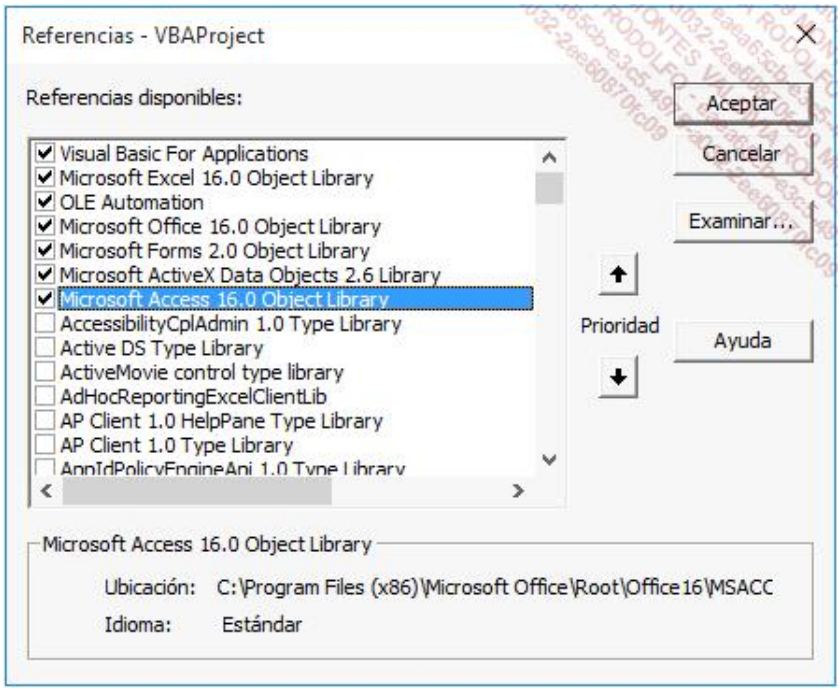


Esta aplicación comprende los siguientes elementos:

- Una cinta de Office personalizada creada a partir de la utilidad Custom UI Editor (encontrará una descripción detallada de esta utilidad en el capítulo Mejoras en la interfaz de usuario).
- Una única hoja de cálculo llamada **Gestión de presupuestos** que constituye la pantalla de bienvenida de la aplicación: título de la aplicación, menú específico de la aplicación. El módulo de clase **GestPresupuesto** asociado a esta hoja no contiene ningún código.
- Dos formularios, **BuscarPresupuesto** y **NuevoPresupuesto**, permiten, respectivamente, buscar y crear un nuevo presupuesto.
- Dos módulos estándares: el módulo **ProcCinta** contiene los procedimientos llamados desde los botones de comando de la cinta; el módulo **ProcGene** contiene las variables públicas y los procedimientos generales de la aplicación.



Esta aplicación necesita que se seleccionen las siguientes referencias:



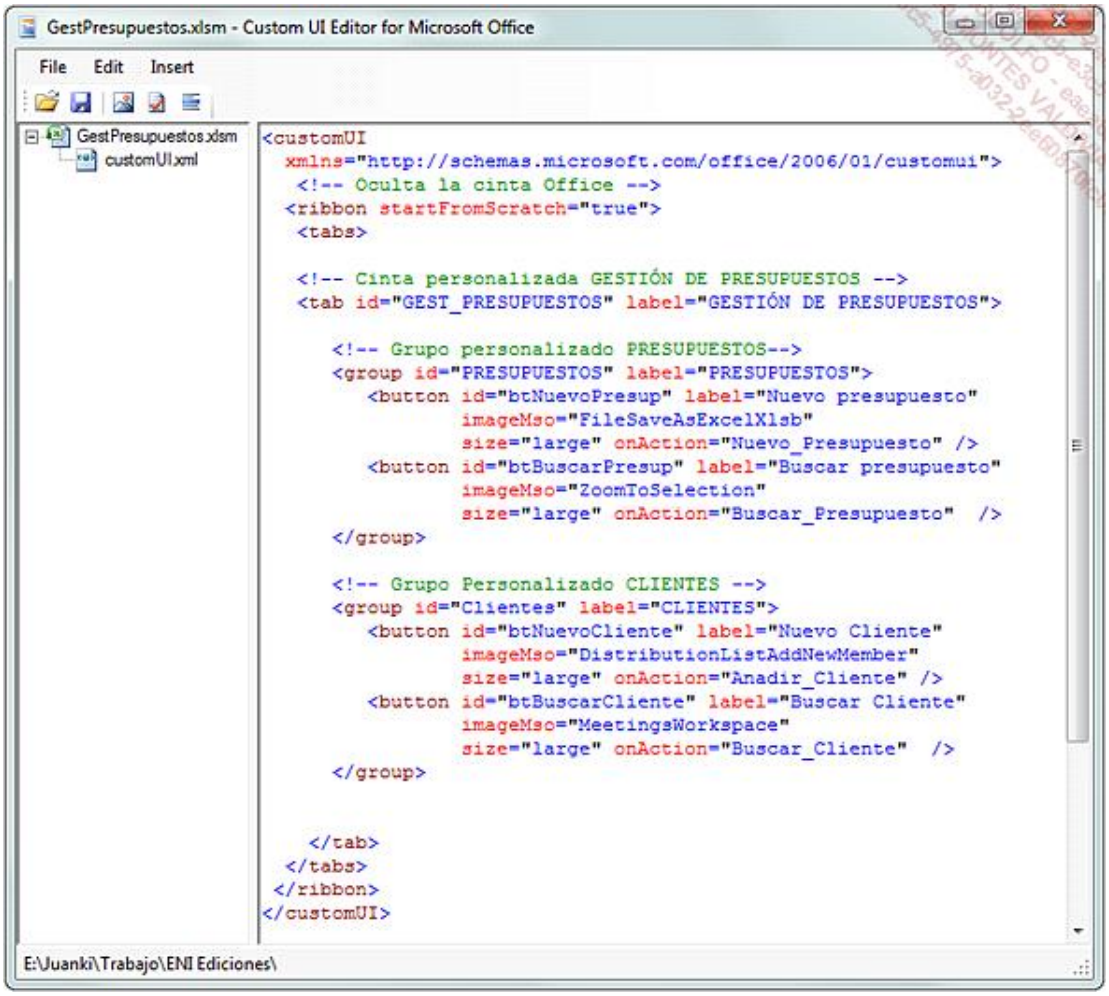
Cinta de Office 2013 personalizada

1. Presentación

La cinta personalizada está formada por la pestaña **GESTIÓN DE PRESUPUESTOS**, que contiene los siguientes elementos:

- Un grupo **PRESUPUESTOS** que incluye dos botones de comando: **Nuevo presupuesto** y **Buscar presupuesto**.
- Un grupo **CLIENTES** que incluye dos botones de comando: **Nuevo Cliente** y **Buscar Cliente**.

2. Código XML de la cinta personalizada




```
<customUI
  xmlns="http://schemas.microsoft.com/office/2006/01/customui">
   <!-- Oculta la cinta Office -->
  <ribbon startFromScratch="true">
   <tabs>

   <!-- Cinta personalizada GESTIÓN DE PRESUPUESTOS -->
   <tab id="GEST_PRESUPUESTOS" label="GESTIÓN DE PRESUPUESTOS">

      <!-- Grupo personalizado PRESUPUESTOS-->
      <group id="PRESUPUESTOS" label="PRESUPUESTOS">
         <button id="btNuevoPresup" label="Nuevo presupuesto"
                 imageMso="FileSaveAsExcelXlsb"
                 size="large" onAction="Nuevo_Presupuesto" />
         <button id="btBuscarPresup" label="Buscar presupuesto"
                 imageMso="ZoomToSelection"
                 size="large" onAction="Buscar_Presupuesto"  />
      </group>

      <!-- Grupo Personalizado CLIENTES -->
      <group id="Clientes" label="CLIENTES">
         <button id="btNuevoCliente" label="Nuevo Cliente"
                 imageMso="DistributionListAddNewMember"
                 size="large" onAction="Anadir_Cliente" />
         <button id="btBuscarCliente" label="Buscar Cliente"
                 imageMso="MeetingsWorkspace"
                 size="large" onAction="Buscar_Cliente"  />
      </group>


    </tab>
   </tabs>
  </ribbon>
</customUI>
```

Módulo ThisWorbook

1. Presentación

Este módulo permite:

- Modificar la presentación de Excel: oculta la barra de fórmulas y los encabezados de filas y columnas.
- Restablecer el entorno de Excel cuando se desactiva el libro: muestra la barra de fórmulas y los encabezados de filas y columnas.

2. Código VBA del módulo ThisWorkbook

```
Option Explicit
Dim TabMenu() As String
_____________________________________________________________
Private Sub Workbook_Open()
'   Ruta de la aplicación
strFolder = ThisWorkbook.Path & "\"
End Sub
Private Sub Workbook_Activate()
'   Oculta la barra de fórmulas y los encabezados
Application.DisplayFormulaBar = False
Application.ActiveWindow.DisplayHeadings = False
End Sub
_____________________________________________________________
Private Sub Workbook_Deactivate()
'   Muestra la barra de fórmulas y los encabezados
With Application
    .DisplayFormulaBar = True
    .ActiveWindow.DisplayHeadings = True
End With
End Sub
```

Formulario NuevoPresupuesto

1. Presentación

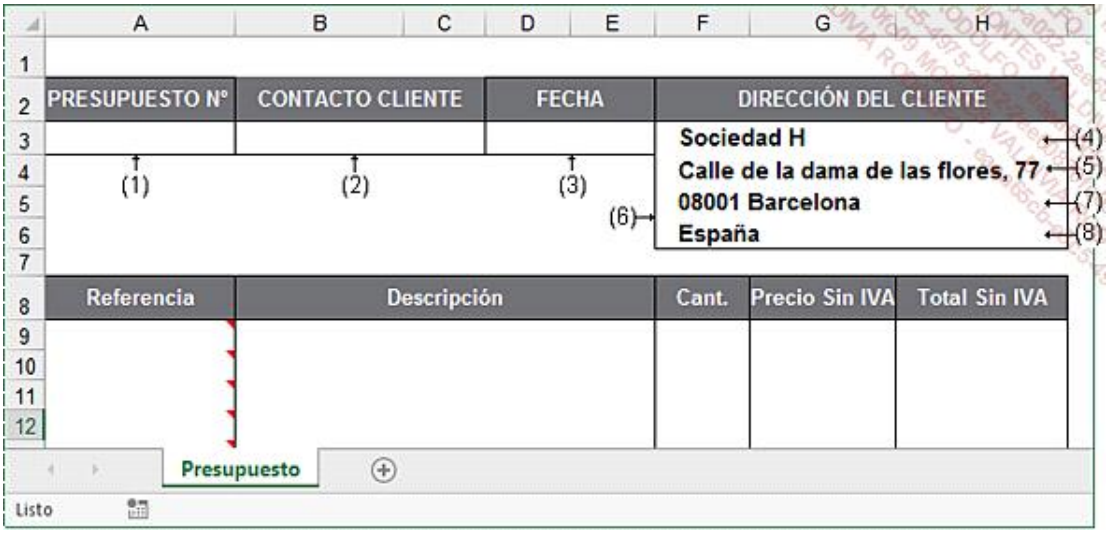
Este módulo permite:

- Crear un nuevo presupuesto a partir del modelo Presupuesto.xltx y guardar el presupuesto en el subdirectorio del cliente.
- Mostrar los datos del cliente en las celdas con nombre (CodCli, Empresa, Dirección, etc.) del libro.
- Aplicar al libro el tema seleccionado en el formulario.

2. Lista de controles

N.°	Nombre del control	Descripción
1.	cboClient	Cuadro de lista desplegable
2.	txtFecha	Cuadro de texto
3.	spinDate	Permite incrementar o decrementar la fecha en un día
4.	cboTema	Cuadro de lista desplegable
5.	cmdCrear	Botón de comando

3. Lista de celdas con nombre del modelo Presupuesto.xltx



1. Presupuesto	5. Direccion
2. Contacto	6. CPostal
3. Fecha	7. Ciudad
4. Empresa	8. Pais

4. Código VBA del formulario NuevoPresupuesto

```
Option Explicit
_____________________________________________________________
Private Sub UserForm_Initialize()
'   Muestra la lista de los clientes
Lista_Clientes ("NuevoPresupuesto")
'   Muestra los temas de Microsoft Office
Muestra_Temas_Office
'   Fecha por defecto
txtFecha = Format(Date, "DD/MM/YYYY")
End Sub
_____________________________________________________________
Private Sub txtDate_BeforeUpdate(ByVal Cancel As _
    MSForms.ReturnBoolean)
'   Controla la fecha introducida
If txtFecha <> "" Then Cancel = Not Ctrl_Fecha(txtFecha)
End Sub
_____________________________________________________________
Private Sub Muestra_Temas_Office()
Dim strPath As String
Dim strfile As String
'   Muestra la lista de los temas de Microsoft Office
cboTheme.Clear
'   Ruta de acceso a los temas
strPath = Left(Application.Path, Len(Application.Path) - 9) _
        & "\Document Themes 16\"
strfile = Dir(strPath & "*.thmx")
'   Muestra el nombre del archivo sin la extensión
Do While strfile <> ""
   cboTema.AddItem Left(strfile, Len(strfile) - 5)
```

```
   strfile = Dir
Loop
End Sub
```

```
Private Sub CmdCrear_Click()
Dim wbk As Workbook
Dim wbkName As String
Dim fso As Object
Dim strTheme As String
'   Controla la introducción de datos
If cboCliente = "" Or txtFecha = "" Then
   MsgBox "Cliente y fecha obligatorios", vbExclamation
   Exit Sub
End If
'   Cierra los libros (excepto ThisWorkbook)
For Each wbk In Workbooks
   If wbk.Name <> ThisWorkbook.Name Then
      wbk.Close
   End If
Next wbk
'   Comprueba que el libro no exista
wbkName = strFolder & cboCliente & "\" & Right(txtFecha, 4) & _
          Mid(txtFecha, 4, 2) & Left(txtFecha, 2) & ".xlsx"
If Dir(wbkName) <> "" Then
   MsgBox "El libro " & wbkName & " ya existe", vbExclamation
   Exit Sub
End If
'   Abre un nuevo libro basado en la plantilla Presupuesto.xltx
Set wbk = Workbooks.Add(Template:=strFolder & "Presupuesto.xltx")
'   Crear el subdirectorio del cliente si no existe
If Dir(strFolder & cboCliente, vbDirectory) = "" Then
   Set fso = CreateObject("Scripting.FileSystemObject")
   fso.createfolder (strFolder & cboCliente)
End If
'   Guarda el libro en este directorio
wbkName = Right(txtFecha, 4) & Mid(txtFecha, 4, 2) & Left(txtFecha, 2) _
                                                   & ".xlsx"
wbk.SaveAs strFolder & cboCliente & "\" & wbkName
wbk.Activate
'   Asigna las celdas del libro a partir de la tabla clientes
'   (Procedimiento del módulo ProcGene)
Muestra_Cliente wbk, cboCliente
wbk.ActiveSheet.Range("A9").Activate
'   Muestra el tema de Office seleccionado
strTheme = Left(Application.Path, Len(Application.Path) - 9) _
        & "\Document Themes 16\" & cboTema & ".thmx"
wbk.ApplyTheme strTheme
'   Cierra el formulario
Unload Me
End Sub
```

```
Private Sub SpinDate_SpinDown()
'   Día anterior
If IsDate(txtFecha) Then txtFecha = DateValue(txtFecha) - 1
End Sub
```

```
Private Sub SpinDate_SpinUp()
'   Día siguiente
If IsDate(txtFecha) Then txtFecha = DateValue(txtFecha) + 1
End Sub
```

Después de ejecutar este módulo, se habrá creado un nuevo presupuesto y se muestra en Excel para que el usuario lo complete:

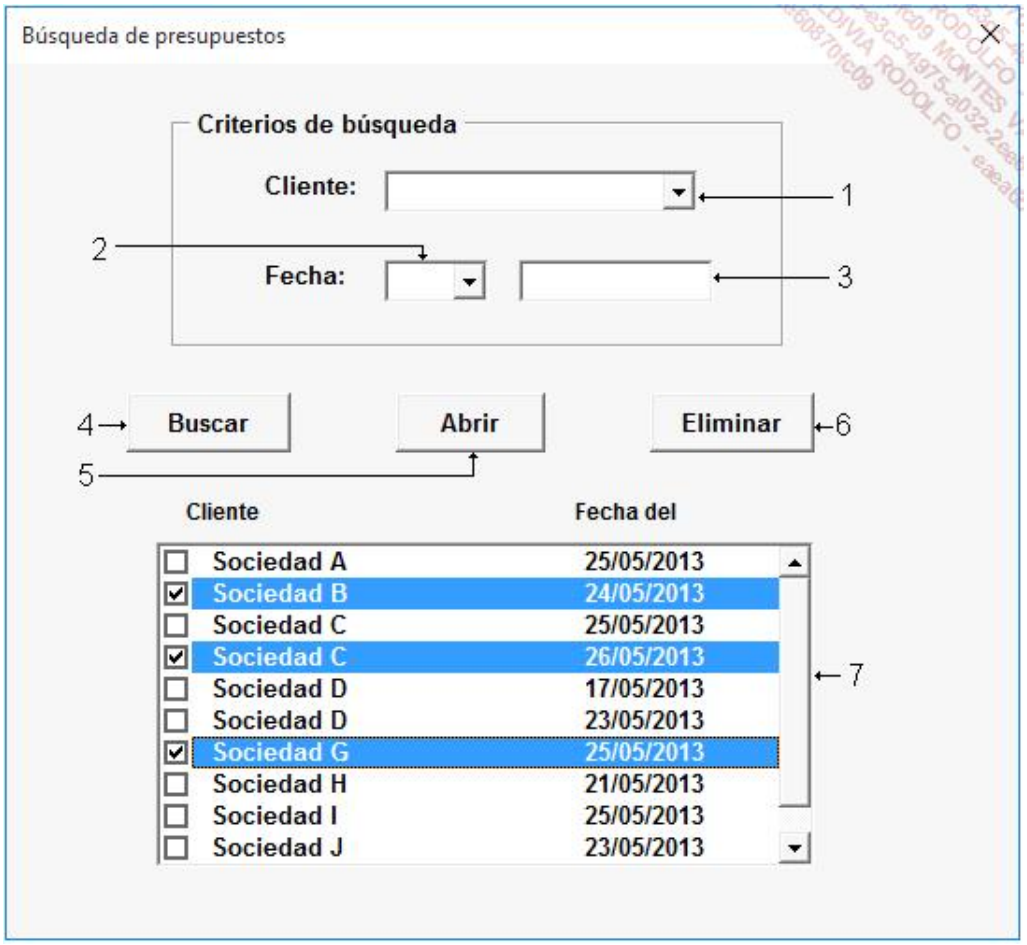
Formulario BuscarPresupuesto

1. Presentación

Este módulo permite:

- Buscar presupuestos en función del código de cliente o la fecha del presupuesto (si no se indica ningún criterio, se muestran todos los presupuestos disponibles).
- Abrir o eliminar uno o más presupuestos en la lista de presupuestos obtenida.

2. Lista de controles



N.°	Nombre del control	Descripción
1.	cboCliente	Cuadro de lista desplegable modificable.
2.	cboOpe	Cuadro de lista desplegable modificable que contiene los operadores de comparación (>= o <=).
3.	txtFecha	Cuadro de texto.
4.	cmdBuscar	Botón de comando.
5.	cmdSupr	Botón de comando.
6.	cmdAbrir	Botón de comando.

7.	lstPresup	Cuadro de lista desplegable.

3. Código VBA del formulario BuscarPresupuesto

```
Option Explicit

Private Sub UserForm_Initialize()
'   Operador para el campo Fecha
cboOpe.AddItem ">="
cboOpe.AddItem "<="
'   Permite seleccionar varios presupuestos
lstPresupuestos.MultiSelect = fmMultiSelectMulti
'   Casilla de verificación para seleccionar los presupuestos
lstPresupuestos.ListStyle = fmListStyleOption
'   Muestra los clientes en la lista desplegable
Lista_Clientes ("BuscaPresupuesto")
End Sub
_____________________________________________________________________
Private Sub txtFecha_BeforeUpdate _
        (ByVal Cancel As MSForms.ReturnBoolean)
' Controla la fecha introducida
If txtFecha <> "" Then Cancel = Not Ctrl_Fecha(txtFecha)
End Sub
_____________________________________________________________________
Private Sub CmdAbrir_Click()
Dim wbk As Workbook
Dim sPath As String
' Abrir los presupuestos seleccionados
On Error GoTo Err:
For j = 0 To lstPresupuestos.ListCount - 1
   If lstPresupuestos.Selected(j) Then
      '   Ruta del presupuesto
      sPath = ThisWorkbook.Path & "\" & lstPresupuestos.List(j, 0) _
      & "\" & Format(DateValue(lstPresupuestos.List(j, 1)), "YYYYMMDD")
      Set wbk = Workbooks.Open(sPath)
      wbk.Activate
   End If
Next j
On Error GoTo 0
Unload Me
Exit Sub
Err:
' No se pueden abrir dos presupuestos con el mismo nombre
' en la misma aplicación de Excel
If Err.Number = 1004 Then
   MsgBox Err.Description
End If
Resume Next
End Sub
_____________________________________________________________________
Private Sub cmdBuscar_Click()
Dim strCli As String
```

```
Dim i As Integer
' Modifica el puntero del ratón
Application.Cursor = xlWait
' Controla los datos introducidos
If (cboOpe <> "" And txtFecha = "") _
   Or (cboOpe = "" And txtFecha <> "") Then
   MsgBox "Debe introducir un operador y una fecha", vbExclamation
   Exit Sub
End If
' Muestra los presupuestos de un cliente o de todos los clientes
lstPresupuestos.Clear
If cboCliente = "" Then
   For i = 0 To cboCliente.ListCount - 1
       strCli = cboCliente.List(i)
       MuestraPresupuestos (strCli)
   Next i
Else
    MuestraPresupuestos (cboCliente)
End If
' Modifica el puntero del ratón
Application.Cursor = xlDefault
End Sub
```

```
Private Sub MuestraPresupuestos(strCli As String)
Dim strRep As String
Dim strPresupuestos As String
' Muestra la lista de presupuestos de un cliente
strRep = ThisWorkbook.Path & "\" & strCli & "\"
strPresupuestos = Dir(strRep & "*.*")
Do While strPresupuestos <> ""
   If Ctrl_Presupuestos(strPresupuestos) Then
      lstPresupuestos.AddItem strCli
      lstPresupuestos.List(lstPresupuestos.ListCount - 1, 1) =
FechaPresupuesto(Left(strPresupuestos, 8))
      i = i + 1
   End If
   strPresupuestos = Dir
Loop
End Sub

Function FechaPresupuesto(zPresupuesto)
'   Transforma el número de presupuesto en fecha
FechaPresupuesto = Right(zPresupuesto, 2) & "/" & Mid(zPresupuesto, 5, 2) _
         & "/" & Left(zPresupuesto, 4)

End Function
```

```
Private Function Ctrl_Presupuestos(strFileName As String) As Boolean
Dim dte
' Controla la fecha del presupuesto
Ctrl_Presupuestos = False
If cboOpe <> "" And txtFecha <> "" Then
   strFileName = Left(strFileName, Len(strFileName) - 5)
   dte = Right(strFileName, 2) & "/" _
       & Mid(strFileName, 5, 2) & "/" & Left(strFileName, 4)
```

```
   If Not IsDate(dte) Then Exit Function
      If cboOpe = ">=" And DateValue(dte) < DateValue(txtFecha) _
         Then Exit Function
      If cboOpe = "<=" And DateValue(dte) > DateValue(txtFecha) _
         Then Exit Function
End If
Ctrl_Presupuestos = True
End Function
_____________________________________________________________________
Private Sub cmdEliminar_Click()
Dim strLista As String
Dim fso As Object
Dim strfile As String
'  Muestra los presupuestos seleccionados
For i = 0 To lstPresupuestos.ListCount - 1
   If lstPresupuestos.Selected(i) Then
      strLista = strLista & vbCr _
         & lstPresupuestos.List(i, 0) & " de " & lstPresupuestos.List(i, 1)
   End If
Next i
'  Eliminar los presupuestos seleccionados después de pedir confirmación
If MsgBox("¿Quiere eliminar los siguientes presupuestos? " & strLista, _
   vbQuestion & vbYesNo) = vbYes Then
   Set fso = CreateObject("Scripting.FileSystemObject")
   For i = 0 To lstPresupuestos.ListCount - 1
      If lstPresupuestos.Selected(i) Then
        strfile = ThisWorkbook.Path & "\" & lstPresupuestos.List(i, 0) _
        & "\" & Format(DateValue(lstPresupuestos.List(i, 1)), "YYYYMMDD")
& ".XLSX"
         fso.Deletefile strfile
      End If
   Next i
End If
' Vuelve a actualizar la lista de presupuestos
cmdBuscar_Click

End Sub
```

Módulos ProcCinta y ProcGene

1. Código VBA del módulo ProcCinta

```
Option Explicit
_____________________________________________________________________
' Variables de módulos
Dim blnAccess As Boolean
Dim appAccess As Access.Application
_____________________________________________________________________
Sub Añadir_cliente(control As IRibbonControl)
'   Abre Access si no lo está y abre el formulario
'   de introducción de clientes en modo añadir
If AccessActivo Then
   appAccess.DoCmd.OpenForm "Ficha_Cliente"
   appAccess.DoCmd.GoToRecord , "", acNewRec
   Application.ActivateMicrosoftApp xlMicrosoftAccess
Else
   MsgBox "No se puede acceder a Access", vbExclamation
End If
End Sub
_____________________________________________________________________
Sub Buscar_Cliente(control As IRibbonControl)
'   Abre Access si no lo está y abre el formulario
'   de introducción de clientes
If AccessActivo Then
   appAccess.DoCmd.OpenForm "Ficha_Cliente"
   Application.ActivateMicrosoftApp xlMicrosoftAccess
Else
   MsgBox "No se puede acceder a Access", vbExclamation
End If
End Sub
_____________________________________________________________________
Private Function AccessActivo() As Boolean
'  Comprueba si se ha abierto Access
On Error GoTo Err:
If blnAccess Then
'  Comprueba si la base de datos Clientes está abierta
Else
'  Ejecuta Access y abre la base Clientes.accdb
ejecutaAccess:
   On Error GoTo Err2:
   Set appAccess = CreateObject("Access.application")
   appAccess.OpenCurrentDatabase (ThisWorkbook.Path & "\Clientes.accdb")
   appAccess.Visible = True
   blnAccess = True
End If
AccessActivo = True
Exit Function
Err:
blnAccess = False
GoTo ejecutaAccess:
Exit Function
```

```
Err2:
AccessActivo = False
End Function
_____________________________________________________________________
Sub Nuevo_Presupuesto(control As IRibbonControl)
'  Muestra el formulario de entrada de presupuestos
NuevoPresupuesto.Show
End Sub
_____________________________________________________________________
Sub Buscar_Presupuesto(control As IRibbonControl)
'  Muestra el formulario de búsqueda de presupuestos
BuscaPresupuesto.Show
End Sub
_____________________________________________________________________
```

2. Código VBA del módulo ProcGene

```
Option Explicit
_____________________________________________________________________
' Variables públicas
Public i As Integer
Public j As Integer
' Directorio de la aplicación
Public strFolder As String
' Objetos ADO
Private cnnCli As ADODB.Connection
Private rstCli As ADODB.Recordset
_____________________________________________________________________
Public Sub Lista_Clientes(FormName As String)
' Apertura de la base Clientes.accdb
If Not AbreBase Then Exit Sub
' Apertura de la tabla Clientes
Set rstCli = New ADODB.Recordset
With rstCli
   .ActiveConnection = cnnCli
   .CursorType = adOpenForwardOnly
   .LockType = adLockOptimistic
   .Open ("Clientes")
End With
' Muestra la lista de clientes extraídos de la tabla
' Clientes de la base de Access Clientes.accdb
Do While Not rstCli.EOF
   Select Case UCase(FormName)
      Case "BUSCAPRESUPUESTO"
           BuscaPresupuesto.cboCliente.AddItem rstCli("Sociedad")
      Case "NUEVOPRESUPUESTO"
           NuevoPresupuesto.cboCliente.AddItem rstCli("Sociedad")
   End Select
   rstCli.MoveNext
Loop
' Cierre de los objetos de Access
rstCli.Close
```

```
cnnCli.Close
End Sub
_____________________________________________________________________
Public Sub Muestra_Cliente(wbk As Workbook, strCli As String)
' Apertura de la base Clientes.accdb
If Not AbreBase Then Exit Sub
' Apertura de la tabla Clientes
Set rstCli = New ADODB.Recordset
With rstCli
   .ActiveConnection = cnnCli
   .CursorType = adOpenForwardOnly
   .LockType = adLockOptimistic
   .Open ("SELECT * FROM Clientes WHERE Sociedad = '" & strCli & "'")
End With
' Muestra datos del cliente en las celdas
' del libro Presupuestos
If Not rstCli.EOF Then
   With wbk.Sheets(1)
      .Range("PRESUPUESTO") = Left(wbk.Name, Len(wbk.Name) - 4)
      .Range("FECHA") = NuevoPresupuesto.txtFecha
      .Range("PRESUPUESTO") = Left(wbk.Name, Len(wbk.Name) - 4)
      .Range("SOCIEDAD") = rstCli("Sociedad")
      .Range("DIRECCION") = rstCli("Direccion")
      .Range("CPOSTAL") = rstCli("Codigo Postal")
      .Range("CIUDAD") = rstCli("Ciudad")
      .Range("PAIS") = UCase(rstCli("Pais"))
      .Range("CONTACTO") = UCase(rstCli("Apellido")) & " " & UCase(rstCli("Nombre"))
   End With
End If
' Cierre de los objetos de Access
rstCli.Close
cnnCli.Close
Set rstCli = Nothing
Set cnnCli = Nothing
End Sub
_____________________________________________________________________
Public Function AbreBase() As Boolean
' Apertura de la base Clientes.accdb
On Error GoTo Err:
Set cnnCli = New ADODB.Connection
With cnnCli
   .Provider = "Microsoft.ACE.OLEDB.16.0"
   .Open ThisWorkbook.Path & "\Clientes.accdb"
End With
On Error GoTo 0
AbreBase = True
Exit Function
Err:
On Error GoTo 0
AbreBase = False
MsgBox "Problema en la apertura de la base Clientes.accdb", _
               vbExclamation
End Function
_____________________________________________________________________
Function Ctrl_Fecha(zFecha As control) As Boolean
```

```
'   Control de la fecha
If zFecha <> "" Then
   '   Sustituye los . por /
   zFecha = Replace(zFecha, ".", "/")
   If IsDate(zFecha) Then
      zFecha = Format(DateValue(zFecha), "DD/MM/YYYY")
      Ctrl_Fecha = True
   Else
      MsgBox "Debe introducir una fecha en formato DD/MM/AAAA", _
                vbExclamation
      Ctrl_Fecha = False
   End If
End If
End Function
```

Lista de instrucciones

1. Cadenas de caracteres

LSet

Alinea a la izquierda una cadena de caracteres dentro de una variable tipo cadena.

Mid

Reemplaza una cantidad especificada de caracteres dentro de una variable cadena por los caracteres extraídos de otra cadena.

RSet

Alinea a la derecha una cadena de caracteres dentro de una variable tipo cadena.

2. Fecha Hora/Matemáticas

Date

Devuelve la fecha del sistema en curso.

Randomize

Inicializa el generador de números aleatorios.

Time

Devuelve la hora del sistema.

3. Declaración

Const

Declara las constantes que hay que utilizar en lugar de valores fijos.

Declare

Se utiliza a nivel de módulo para declarar las referencias a procedimientos externos en una biblioteca DLL o un recurso de código Macintosh.

Def*Type*

Define los tipos de datos por defecto de las variables y valores devueltos por procedimientos **Function** cuyos nombres comienzan con los caracteres especificados (`DefBool, DefInt, ..., DefStr`).

Dim

Declara variables y les reserva espacio de almacenamiento en la memoria.

Enum

Declara un tipo para una enumeración.

Event

Declara un evento definido por el usuario.

Function

Declara el nombre, los argumentos y el código que forma el cuerpo de un procedimiento **Function**.

Let

Asigna el valor de una expresión a una variable o a una propiedad (equivale al signo =).

Option Base

Define el menor valor del índice para las matrices: **0** o **1**.

Option Compare

Define el modo de comparación de cadenas: **Binary** o **Text**.

Option Explicit

Obliga la declaración explícita de todas las variables del módulo.

Option Private Module

Declara el módulo completo como privado.

Private

Declara las variables privadas y reserva su espacio de almacenamiento en la memoria.

Property Get

Declara el nombre, los argumentos y el código de un procedimiento **Property** que permite leer el valor de una propiedad.

Property Let

Declara el nombre, los argumentos y el código de un procedimiento **Property** que le asigna un valor a una propiedad.

Property Set

Declara el nombre, los argumentos y el código de un procedimiento **Property** que asigna una referencia a un objeto.

Public

Declara las variables públicas y les reserva espacio de almacenamiento en la memoria.

ReDim

Dimensiona variables de tipo tabla dinámica y les reserva espacio de almacenamiento en la memoria.

Set

Asigna una referencia a un objeto.

Static

Define las variables estáticas y les reserva espacio de almacenamiento en la memoria.

Sub

Declara el nombre, los argumentos y el código de un procedimiento **Sub**.

Type

Define un tipo de datos definido por el usuario.

4. Error

Error

Simula la ocurrencia de un error.

On Error

Activa una rutina de tratamiento de errores y especifica su ubicación dentro de un procedimiento.
También permite desactivar una rutina de tratamiento de errores.

Resume

Restablece la ejecución del código cuando termina una rutina de tratamiento de errores.

5. Archivo

Close

Finaliza las operaciones de entrada y salida en un archivo abierto con la instrucción **Open**.

FileCopy

Copia un archivo.

Get

Lee los datos de un archivo abierto y los guarda en una variable.

Input #

Lee los datos a partir de un archivo secuencial abierto y los asigna a variables.

Kill

Elimina los archivos del disco.

Line Input #

Lee una línea de datos a partir de un archivo secuencial abierto y la asigna a una variable de tipo cadena.

Lock...Unlock

Controla el acceso por parte de otros procesos a todo o parte de un archivo abierto mediante la instrucción **Open**.

Open

Permite ejecutar una operación de entrada y salida en un archivo.

Print #

Escribe los datos con formato en un archivo secuencial.

Put

Escribe el contenido de una variable en un archivo de disco.

Reset

Cierra todos los archivos de discos abiertos con la instrucción **Open**.

Seek

Define la posición de la próxima lectura y escritura en un archivo abierto con la instrucción **Open**.

SetAttr

Define los atributos de un archivo.

Width #

Asigna la longitud de la línea de salida a un archivo abierto con la instrucción **Open**.

Write #

Escribe datos en un archivo secuencial.

Ejemplo

El siguiente proceso permite leer un archivo de texto e importar los valores en celdas de Excel.

```
Sub ImportaArchivoTexto()
Dim oSht As Worksheet
Dim sNomArchivo As String
Dim iNumArchivo As Integer
Dim sLinea As String
Dim Tabla() As String
Dim iRow As Long
Dim iCol As Long
Dim i As Long

'   ¿Archivo import existe?
sNomArchivo = ThisWorkbook.Path & "\Clientes.txt"
If Dir(sNomArchivo, vbNormal) = "" Then
   MsgBox "Archivo " &sNomArchivo & " inexistente"
   Exit Sub
End If

'   Hoja de destino
Set oSht = ThisWorkbook.Sheets(1)
oSht.Cells.Clear

'    Apertura de archivo de texto
iNumArchivo = FreeFile
iRow = 1
Open sNomArchivo For Input As #iNumArchivo
'    Recorre las líneas del archivo de texto
Do While Not EOF(iNumArchivo)
    iCol = 1
    Line Input #iNumArchivo, sLinea
    Tabla = Split(sLinea, ";")
    '  Copia los campos de la línea en las celdas de Excel
    For i = LBound(Tabla) To UBound(Tabla)
    oSht.Cells(iRow, iCol) = Replace(Tabla(i), Chr(34), "")
    iCol = iCol + 1
  Next
  iRow = iRow + 1
Loop
Close #iNumArchivo

End Sub
```

El siguiente procedimiento permite crear un archivo de texto y escribir los datos de las celdas de Excel.

```
Sub ExportaArchivoTexto()
Dim oSht As Worksheet
```

```
Dim sNomArchivo As String
Dim iNumArchivo As Integer
Dim sLinea As String
Dim oRngExport As Range
Dim iNbRows As Long
Dim iNbCols As Long
Dim i, j As Integer
Dim vCelda As Variant

'   ¿Archivo Export existe?
sNomArchivo = ThisWorkbook.Path & "\Clientes.txt"
If Dir(sNomArchivo, vbNormal) <> "" Then
   Kill sNomArchivo
End If

'   Selección de celdas de Excel
Set oSht = ThisWorkbook.Sheets(1)
oSht.Range("A1").CurrentRegion.Select
Set oRngExport = Application.Selection
iNbRows = oRngExport.Rows.Count
iNbCols = oRngExport.Columns.Count

'   Creación del archivo de texto
iNumArchivo = FreeFile
Open sNomArchivo For Output As #iNumArchivo

'   Recorre las líneas y columnas de la hoja
For i = 1 To iNbRows
    For j = 1 To iNbCols
        vCelda = Trim(oRngExport.Cells(i, j).Value)
        If IsNumeric (vCelda) Then vCelda = Val(vCelda)
        If IsEmpty(oRngExport.Cells(i, j).Value) Then
           vCelda = Chr(32)
        End If
        '   Escritura en el archivo de texto
        If j = iNbCols Then
           sLinea = sLinea & vCell
           Write #iNumArchivo, sLinea
           sLinea = ""
        Else
           sLinea = sLinea & vCelda & ";"
        End If
    Next j
Next i
Close #1

End Sub
```

6. Estructuración

Call

Transfiere el control a un procedimiento **Sub, Function**, DLL o a un procedimiento de recursos de

código Macintosh.

Do...Loop

Repite un bloque de instrucciones mientras se cumple una condición o hasta que la condición se hace verdadera.

End

Termina un procedimiento o un bloque.

Exit

Sale de un bloque de código **`Do...Loop`**, **`For...Next`**, **`Function`**, **`Sub`** o **`Property`**.

For Each...Next

Repite un grupo de instrucciones para cada elemento de una matriz o de una colección.

For...Next

Repite un bloque de instrucciones un determinado número de veces.

GoTo

Realiza una bifurcación incondicional hacia una línea determinada de un procedimiento.

GoSub...Return

Realiza una bifurcación hacia una subrutina dentro de un procedimiento y luego retorna a la instrucción inmediatamente posterior a la bifurcación.

On GoSub y **On GoTo**

Realiza una bifurcación hacia una de las líneas especificadas, según el valor de una expresión dada.

Rem

Permite la entrada de comentarios (equivale al apóstrofo).

Select Case

Ejecuta uno o más grupos de instrucciones según el valor de una expresión dada.

Stop

Interrumpe la ejecución de un procedimiento.

If...Then..., **ElseIf...** y **Else...End If**

Permite la ejecución condicional de un grupo de instrucciones según el resultado de una expresión dada.

While...Wend

Ejecuta una serie de instrucciones mientras se cumpla una condición dada.

With

Ejecuta una serie de instrucciones sobre un único objeto o un tipo definido por el usuario.

7. Sistema

Beep

Emite una señal sonora.

ChDir

Cambia el directorio o la carpeta actual.

ChDrive

Cambia la unidad de disco actual.

MkDir

Crea un nuevo directorio o nueva carpeta.

Name

Modifica el nombre de un archivo, de un directorio o de una carpeta.

RmDir

Elimina un directorio o una carpeta existente.

8. Diversas

AppActivate

Activa una ventana de aplicación.

DeleteSetting

Elimina el valor de una sección o de una clave en la base de registro de Windows.

Erase

Reinicia los elementos de matrices de tamaño fijo y libera el espacio de almacenamiento asignado a matrices dinámicas.

Implements

Especifica una interfaz o una clase que se implementará en el módulo de clase donde aparece.

Load

Carga un objeto pero no lo muestra.

RaiseEvent

Elimina un evento declarado en el nivel de módulo dentro de una clase, formulario o documento.

SaveSetting

Guarda o crea una entrada para una aplicación en la base de registro de Windows.

SendKeys

Envía una o más pulsaciones de teclas a la ventana activa, como si se hubieran presionado desde el teclado. No disponible en Macintosh.

Unload

Elimina un objeto de la memoria.

Lista de funciones

Las funciones cuyo nombre termina con el signo $ devuelven valores en variables de tipo **String**, y no de tipo Variant.

1. Conversiones

CBool

Convierte una expresión a datos de tipo Boolean.

CByte

Convierte una expresión a datos de tipo Byte.

CCur

Convierte una expresión a datos de tipo Currency.

CDate

Convierte una expresión a datos de tipo Date.

CDbl

Convierte una expresión a datos de tipo Double (doble precisión).

CDec

Convierte una expresión a datos de tipo Decimal.

CInt

Convierte una expresión a datos de tipo Integer (nombre entero).

CLng

Convierte una expresión a datos de tipo Long (entero largo).

CSng

Convierte una expresión a datos de tipo Single (simple precisión).

CStr

Convierte una expresión a datos de tipo String.

CVar

Convierte una expresión a datos de tipo Variant.

CVErr

Devuelve un tipo Variant de un subtipo Error que contiene un número de error especificado por el usuario.

Format, Format$

Aplica un formato a una expresión según las instrucciones contenidas en una expresión de tipo formato.

FormatCurrency

Devuelve una expresión con formato en forma de valor de tipo **Currency** usando el símbolo monetario definido en el panel de control del sistema.

FormatDateTime

Devuelve una expresión con formato de fecha u hora.

FormatNumber

Devuelve una expresión con formato de número.

FormatPercent

Devuelve una expresión con formato de porcentaje (multiplicado por 100) con el carácter % al final.

Hex, Hex$

Devuelve una cadena de caracteres que representa el valor de un número escrito en forma hexadecimal.

Oct, Oct$

Devuelve una cadena que representa el valor octal de un número.

QBColor

Devuelve un valor que indica el código de color RGB correspondiente al número de color indicado.

RGB

Devuelve un número entero que representa el valor de un color RGB.

Str, Str$

Devuelve una cadena de caracteres que representa el número especificado.

StrConv

Devuelve un valor convertido al formato indicado.

Val

Devuelve el valor numérico contenido en una cadena de caracteres.

2. Cadenas de caracteres

Asc

Devuelve el código de carácter correspondiente al primer carácter de una cadena.

Chr, Chr$

Devuelve el carácter correspondiente al código de carácter especificado.

InStr

Devuelve la posición de la primera ocurrencia de una cadena dentro de otra cadena.

InStrRev

Devuelve la posición de la ocurrencia de una cadena dentro de otra, a partir del fin de la cadena.

LCase, LCase$

Devuelve una cadena con sus caracteres pasados a minúsculas.

Left, Left$

Devuelve un número especificado de caracteres de una cadena, comenzando desde la izquierda.

Len

Devuelve la cantidad de caracteres contenidos en una cadena o la cantidad de bytes necesarios para almacenar una variable.

LTrim, LTrim$

Devuelve una copia de una cadena eliminando los espacios a la izquierda.

Mid, Mid$

Devuelve un número especificado de caracteres extraídos de una cadena de caracteres.

Replace

Devuelve una cadena en la que una subcadena especificada se reemplaza por otra subcadena.

Right, Right$

Devuelve un número especificado de caracteres de una cadena, comenzando desde la derecha.

RTrim, RTrim$

Devuelve una copia de una cadena eliminando los espacios a la derecha.

Space, Space$

Devuelve una cadena formada por un número de espacios especificado.

StrComp

Devuelve un valor que indica el resultado de la comparación de cadenas.

String, String$

Crea una cadena constituida por una cadena de caracteres que se repite con la longitud especificada.

StrReverse

Devuelve una cadena que contiene los mismos caracteres que la cadena dada, pero en orden inverso.

Trim, Trim$

Devuelve una copia de una cadena eliminando los espacios a la izquierda y a la derecha.

UCase, UCase$

Devuelve una cadena con sus caracteres pasados a mayúsculas.

Ejemplo

Este procedimiento permite asignar un código de identificación según el sexo, apellido y nombre y el año de nacimiento.

	A	B	C	D	E
1	Identificador	Sexo	Apellido	Nombre	Fecha de nacimiento
2	1-ROS-P-1958	M	Rosell	Pedro	04/06/1958
3	2-DUR-V-1971	F	Duran	Verónica	24/01/1971
4	2-GON-C-1964	F	González	Claudia	16/10/1964
5	1-HER-J-1976	M	Hernández	Jaime	25/03/1976
6					

Importar Exportar Identificadores

Listo

```
Sub CalculaIdentificador()
  Dim i As Integer
  Dim codigo As String

  ' Recorre las filas de la hoja
  i = 2
```

```
  With ThisWorkbook.Sheets(1)
      Do While .Cells(i, 2) <> ""
        ' código 1 o 2 según el sexo
        ' seguido de las tres primeras letras del apellido en mayúsculas,
        ' más la inicial del nombre y el año de nacimiento
        If .Cells(i, 2) = "F" Then
            codigo = "2-"
        Else
            codigo = "1-"
        End If
        codigo = codigo & UCase(Left(.Cells(i, 3), 3) & "-" _
               & UCase(Left(.Cells(i, 4), 1)) & "-" _
               & Right(.Cells(i, 5), 4))
        .Cells(i, 1) = codigo
        i = i + 1
      Loop
  End With

End Sub
```

3. Matemáticas

Las funciones matemáticas se llaman funciones **intrínsecas**.

Abs

Devuelve el valor absoluto de un número.

Atn

Devuelve el arcotangente de un número.

Cos

Devuelve el coseno de un ángulo.

Exp

Devuelve e (la base de los logaritmos neperianos), elevado a una potencia dada.

Fix

Devuelve la parte entera de un número.

Int

Devuelve la parte entera de un número. La diferencia con la función **Fix** consiste en que, si el valor del argumento "número" es negativo, **Int** devuelve el primer entero negativo menor o igual al argumento, mientras que **Fix** devuelve el primer entero negativo mayor o igual al argumento.

Log

Devuelve el logaritmo neperiano de un número.

Rnd

Devuelve un número aleatorio.

Round

Devuelve un número redondeado a una cantidad especificada de posiciones decimales.

Sgn

Devuelve un número entero que indica el signo del argumento.

Sin

Devuelve el seno de un ángulo.

Sqr

Devuelve la raíz cuadrada de un número.

Tan

Devuelve la tangente de un ángulo.

Ejemplo

```
Sub FctsCalculos()
   'Diferencia entre Int y Fix
   nb1 = -125.45
   'Muestra -126
   MsgBox Int(nb1)
   'Muestra -125
   MsgBox Fix(nb1)
   'Devuelve un número aleatorio comprendido entre 1 y 49
   nb2 = Int(49*Rnd)+1
   MsgBox nb2
End Sub
```

Otras funciones, si bien no son intrínsecas, se pueden obtener a partir de funciones intrínsecas.

Algunos ejemplos:

Secante = 1 / Cos(X).

Cosecante = 1 / Sin(X).

Cotangente = 1 / Tan(X).

4. Financieras

DDB

Devuelve un valor que indica la amortización de un bien a lo largo de un período especificado (utiliza el método de amortización decreciente a tasa doble u otro método precisado).

FV

Devuelve un valor que indica el importe futuro de una anualidad basada en pagos constantes y periódicos y con una tasa de interés fija.

IPmt

Devuelve un valor que indica el importe, para un período dado, de una anualidad basada en pagos constantes y periódicos y con una tasa de interés fija.

IRR

Devuelve un valor que indica la tasa interna de retorno de una serie de movimientos de fondos periódicos (pagos y cobros).

MIRR

Devuelve un valor que indica la tasa interna de retorno modificada de una serie de movimientos de fondos periódicos (pagos y cobros).

NPer

Devuelve un valor que indica la cantidad de períodos de una anualidad basada en movimientos constantes y periódicos y con una tasa de interés fija.

NPV

Devuelve un valor que indica el valor actual neto de una inversión, calculada en función de una serie de movimientos de fondos periódicos (pagos y cobros) y según una tasa de descuento.

Pmt

Devuelve un valor que indica el importe de una anualidad basada en movimientos constantes y periódicos y con una tasa de interés fija.

PPmt

Devuelve un valor que indica el reembolso correspondiente a un período determinado de una anualidad basada en pagos periódicos y constantes con una tasa de interés fija.

PV

Devuelve un valor que indica el importe actual de una anualidad basada en pagos periódicos constantes que se van a realizar en el futuro, con una tasa de interés fija.

Rate

Devuelve un valor que indica la tasa de interés por período para una anualidad.

SLN

Devuelve un valor que indica la amortización de un bien para un período dado según el método lineal.

SYD

Devuelve un valor que indica la amortización global de un bien para un período dado.

5. Fechas y horas

Date, Date$

Devuelve la fecha del sistema en curso.

DateAdd

Devuelve un valor que representa la fecha correspondiente a una fecha dada, más un intervalo de tiempo especificado.

DateDiff

Devuelve un valor que indica la cantidad de intervalos de tiempo entre dos fechas dadas.

DatePart

Devuelve un valor que contiene el elemento especificado de una fecha dada.

DateSerial

Devuelve la fecha correspondiente a un año, un mes y un día especificados.

DateValue

Devuelve una fecha.

Day

Devuelve un número entero comprendido entre 1 y 31 que representa el día del mes.

Hour

Devuelve un número entero comprendido entre 0 y 23 que representa la hora del día.

Minute

Devuelve un número entero comprendido entre 0 y 59 que representa los minutos.

Month

Devuelve un número entero comprendido entre 1 y 12 que representa el mes del año.

MonthName

Devuelve una cadena que indica el mes especificado.

Now

Devuelve la fecha y la hora actuales tomadas del reloj del sistema.

Second

Devuelve un número entero comprendido entre 0 y 59 que representa los segundos.

Time, Time$

Devuelve la hora actual.

Timer

Devuelve la cantidad de segundos transcurridos desde la medianoche.

TimeSerial

Devuelve una fecha que contiene la hora exacta (horas, minutos y segundos).

TimeValue

Devuelve una hora.

WeekDay

Devuelve un número entero que representa el día de la semana.

WeekdayName

Devuelve una cadena que indica el día de la semana especificada.

Year

Devuelve un número entero que representa el año.

Ejemplo

Diversos cálculos con fechas y horas:

```
Sub CalcFechasyHoras ()
   '   Muestra la fecha del día
   MsgBox "Hoy es " & Date
```

```
   '   Muestra la cantidad de segundos transcurridos desde medianoche
   MsgBox "Medianoche fue hace " & _
          Timer & " segundos"
   '   Calcula y muestra el tiempo que resta de trabajo,
   '   suponiendo que la jornada termina a las 17 h 30
   Resto = TimeSerial(17 - Hour(Time), _
           30 - Minute(Time), 0 - Second(Time))
   MsgBox "Finalizando a las 17h30, falta " & Resto & _
          " horas de trabajo"
   '   Calcula y muestra el último día del mes en curso
   Final = DateSerial(Year(Now), Month(Now) + 1, 1) - 1
   MsgBox "el último día del mes en curso es " & Final
   '   Muestra el nombre del día de la semana de esa fecha
   '   (- 1 porque para Excel la semana empieza el domingo)
   MsgBox "Será un " & WeekdayName(Weekday(Final) - 1)
End Sub
```

6. Archivos, Sistema

CurDir, CurDir$

Devuelve la ruta de acceso actual.

Dir, Dir$

Devuelve el nombre de un archivo, de un directorio o de una carpeta que coincide con una plantilla o un atributo de archivo especificado o devuelve la etiqueta de volumen de una unidad de disco.

EOF

Devuelve un valor que indica si se ha llegado al final de un archivo.

FileAttr

Devuelve un valor que representa el modo del archivo para los archivos abiertos usando la instrucción **Open**.

FileDateTime

Devuelve la fecha y la hora de creación o de la última modificación de un archivo.

FileLen

Devuelve el tamaño de un archivo en bytes.

FreeFile

Devuelve un número que indica el siguiente número de archivo disponible para su uso en la instrucción **Open**.

GetAttr

Devuelve un número que representa los atributos de un archivo, directorio o carpeta o la etiqueta de un volumen.

Input, Input$

Devuelve los caracteres (bytes) leídos a partir de un archivo secuencial abierto.

Loc

Devuelve la posición de lectura y escritura actuales en un archivo abierto.

LOF

Devuelve la longitud en bytes de un archivo abierto con la instrucción **Open**.

Seek, Seek$

Devuelve la posición de lectura y escritura actuales en un archivo abierto con la instrucción **Open**.

Ejemplo

Procedimiento para mostrar los nombres, las fechas de última modificación y los tamaños de los cinco primeros archivos encontrados en la carpeta actual.

```
Sub ListaArchivos ()
   Dim strPath As String
   Dim strFile As String

   strPath = CurDir() & "\"
   strFile = Dir(strPath)
   For i = 1 To 5
       If i = 1 Then
          strFile = Dir(strPath)
       Else
          strFile = Dir()
       End If
       If strFile <> "" Then
          MsgBox "Archivo: " & strFile & Chr(13) & _
                 "Fecha: " & FileDateTime(strFile) & Chr(13) & _
                 "Tamaño: " & Format(FileLen(strFile), "# ##0")
      End If
   Next i
End Sub
```

7. Verificación de variables

IsArray

Devuelve un valor que indica si una variable es o no una matriz.

IsDate

Devuelve un valor que indica si una expresión se puede convertir a fecha.

IsEmpty

Devuelve un valor que indica si una variable ha sido o no iniciada.

IsError

Devuelve un valor que indica si una expresión es o no un valor de error.

IsMissing

Devuelve un valor que indica si se le pasó a un procedimiento un argumento opcional.

IsNull

Devuelve un valor que indica si una expresión contiene o no un valor válido.

IsNumeric

Devuelve un valor que indica si una expresión se puede interpretar como un número.

IsObject

Devuelve un valor que indica si un identificador representa una variable objeto.

TypeName

Devuelve una cadena que proporciona información acerca de una variable.

VarType

Devuelve un valor que indica el subtipo de una variable.

8. Interacción

CreateObject

Crea un objeto OLE Automation.

GetObject

Recupera un objeto OLE Automation en un archivo.

InputBox

Muestra un cuadro de diálogo con una invitación, espera que el usuario escriba un texto o pulse un botón y luego devuelve el contenido del cuadro de texto.

MsgBox

Muestra un mensaje en un cuadro de diálogo, espera que el usuario pulse un botón y luego devuelve un valor que indica el botón pulsado por el usuario.

Shell

Ejecuta un programa ejecutable.

9. Matrices

Array

Devuelve un dato de tipo **Variant** que contiene una matriz.

Filter

Devuelve una matriz de base cero que contiene un subconjunto de una matriz de cadenas basado en los criterios de filtrado especificados.

Joint

Devuelve una cadena creada por la unión de varias subcadenas contenidas en una matriz.

LBound

Devuelve el menor valor del índice disponible para la dimensión indicada en una matriz.

Split

Devuelve una matriz de una dimensión, basada en cero, que contiene la cantidad especificada de subcadenas.

UBound

Devuelve el mayor valor del índice disponible para la dimensión indicada en una matriz.

10. SQL

SQLBind

Especifica dónde se encuentran los resultados de la extracción con la función `SQLRetrieve`.

SQLClose

Cierra una conexión con un origen de datos externos.

SQLError

Devuelve información de error detallada cuando se utiliza después de que se produzca un error en

otra de las funciones ODBC.

SQLExec Query

Ejecuta una consulta en un origen de datos con una conexión establecida con la instrucción **SQLOpen**.

SQLGet Schema

Devuelve información acerca de la estructura del origen de datos de una conexión en particular.

SQLOpen

Establece una conexión con un origen de datos.

SQLRequest

Conecta con un origen de datos externo y ejecuta una consulta desde una hoja de cálculo; a continuación, devuelve el resultado de la consulta en forma de matriz Visual Basic.

SQLRetrieve

Recupera todos los resultados, o parte de ellos, de una consulta ejecutada con anterioridad.

SQLRetrieve, ToFile

Devuelve todos los resultados de una consulta ejecutada con anterioridad y los guarda en un archivo.

11. Diversas

CallByName

Ejecuta un método de un objeto, o establece o devuelve una propiedad de un objeto.

Choose

Selecciona y devuelve un valor a partir de una lista de argumentos.

DoEvents

Detiene momentáneamente la ejecución y cede el control al sistema operativo, para que este pueda procesar otros eventos.

Environ

Devuelve el valor asociado a una variable de entorno del sistema operativo.

GetAllSettings

Devuelve una lista de claves y sus valores respectivos (originalmente creados con la instrucción **SaveSetting**) a partir de la entrada de una aplicación en la base de registro de Windows.

GetSetting

Devuelve el valor de clave de una entrada de aplicación en la base de registro de Windows.

Iif

Devuelve uno u otro de dos argumentos según la evaluación de una expresión.

Spc

Función utilizada con la instrucción **`Print #`** o el método **`Print`** para posicionar la salida.

Switch

Evalúa una lista de expresiones y devuelve un valor o una expresión asociada a la primera expresión de la lista que tiene el valor **True**.

Tab

Función utilizada con la instrucción **`Print #`** o el método **`Print`** para posicionar la salida.

12. Solver

SolverAdd

Agrega una restricción al problema actual.

SolverChange

Modifica una restricción al problema actual.

SolverDelete

Elimina una restricción al problema actual.

SolverFinish

Indica a Excel qué debe hacer con los resultados y qué clase de informe debe crear al finalizar el proceso de resolución.

SolverFinishDialog

Es igual a la función **SolverFinish**, pero también muestra el cuadro de diálogo **Resultados de Solver** después de resolver el problema.

SolverGet

Devuelve la información relativa a la configuración de Solver.

SolverLoad

Carga la configuración de un modelo existente.

SolverOK

Define un modelo básico de Solver.

SolverOKDialog

Es igual a **SolverOK** pero también muestra el cuadro de diálogo **Solver**.

SolverOptions

Especifica las opciones avanzadas de un modelo.

SolverReset

Reinicia toda la configuración.

SolverSave

Guarda la configuración de un modelo.

SolverSolve

Procede con la resolución de un modelo.

El Mundo de la Programación en tus Manos...!

DETODOPROGRAMACION.ORG

DETODOPROGRAMACION.ORG

Material para los amantes de la Programación Java, C/C++/C#,Visual.Net, SQL, Python, Javascript, Oracle, Algoritmos, CSS, Desarrollo Web, Joomla, jquery, Ajax y Mucho Mas…

VISITA

www.detodoprogramacion.org
www.detodopython.com
www.gratiscodigo.com

GRATISCODIGO.COM
Usa, lo que ya se creó...

Detodo Python.com