
# Apuntes Programación Excel VBA VIII

*La función MsgBox. Trabajando con Userform en Excel Conceptos generales, una introducción*

*Jose Ignacio González Gómez*
*Departamento de Economía Financiera y Contabilidad - Universidad de La Laguna*
www.jggomez.eu

# INDICE

# 1 La función en MsgBox

**Fuente:** https://exceltotal.com/la-funcion-msgbox-en-vba/

## *1.1 Introducción*

La **función MsgBox en VBA** nos permite mostrar un mensaje dentro de un cuadro de diálogo en espera de que el usuario de Excel haga clic sobre alguno de los botones provistos. Si lo deseamos podemos tomar alguna acción específica después de conocer el botón pulsado por el usuario.

## *1.2 Argumentos de la función MsgBox*

La **función MsgBox en VBA** tiene 5 argumentos, los cuales serán explicados a continuación:

- **Prompt** (*obligatorio*): Es la cadena de texto que se mostrará como el mensaje dentro del cuadro de diálogo. La longitud máxima es de 1024 caracteres, pero depende del tipo de fuente utilizada.
- **Buttons** (*opcional*): Expresión numérica que proviene de la suma de ciertas constantes que representan el tipo de botón e iconos a desplegar.
- **Title** (*opcional*): Cadena de texto que se mostrará como el título del cuadro de diálogo. Si se omite, el título será el nombre de la aplicación.
- **HelpFile** (*opcional*): Cadena de texto con la ubicación del archivo de ayuda asociado al cuadro de diálogo. Si se especifica este argumento, debe indicarse también *Context*.
- **Context** (*opcional*): Valor numérico asignado por el autor al tema de ayuda. Si se especifica este argumento, debe indicarse también *HelpFile*.

Para dejar en claro el uso de cada uno de los argumentos de la función mostramos varios ejemplos y para cada uno de ellos crearemos un botón de comando ActiveX e insertaremos el código correspondiente en su evento Click.

## *1.3 Ejemplo 1: La función MsgBox en VBA*

Ya que solamente el primer argumento de la función MsgBox es obligatorio, podemos crear un mensaje informativo para el usuario con la siguiente línea de código:

```
1  Private Sub CommandButton1_Click()
2  MsgBox "Hola Mundo"
3  End Sub
```

Al ejecutar este código se mostrará un cuadro de diálogo como el siguiente


Ilustración 1

Al no haber especificado ningún valor para el segundo argumento, se muestra solamente el botón Aceptar. Y ya que tampoco hay un tercer argumento, el cuadro de diálogo tendrá el título "Microsoft Excel" que es el nombre de la aplicación.

### 1.4 Ejemplo 2: El argumento Title

Antes de revisar el segundo argumento de la función MsgBox, hablaremos sobre su tercer argumento que es el título del cuadro de diálogo. Para poner un título personalizado será suficiente indicarlo de la siguiente manera:

```
1  Private Sub CommandButton2_Click()
2  MsgBox "Hola Mundo", , "Mensaje especial"
3  End Sub
```

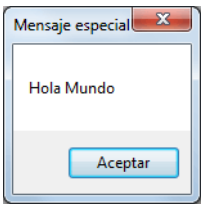



Ilustración 2

Esta instrucción mostrará el título del cuadro de diálogo como "Mensaje especial" y ya no se mostrará el nombre de la aplicación.

### 1.5 Ejemplo 3: El argumento Buttons

Hasta ahora solo hemos visto el botón Aceptar en el cuadro de diálogo, pero el segundo argumento de la función MsgBox nos permitirá indicar los botones que deseamos mostrar y también podremos elegir el icono desplegado y el comportamiento del cuadro de diálogo. La siguiente tabla indica los valores que podemos utilizar para este argumento:

| Constante | Valor | Descripción |
|---|---|---|
| vbOKOnly | 0 | Muestra el botón Aceptar |
| vbOKCancel | 1 | Muestra los botones Aceptar y Cancelar. |
| vbAbortRetryIgnore | 2 | Muestra los botones Anular, Reintentar y Omitir. |
| vbYesNoCancel | 3 | Muestra los botones Si, No y Cancelar. |
| vbYesNo | 4 | Muestra los botones Si y No. |
| vbRetryCancel | 5 | Muestra los botones Reintentar y Cancelar. |
| vbCritical | 16 | Muestra el icono Mensaje crítico. |
| vbQuestion | 32 | Muestra el icono Consulta de advertencia. |
| vbExclamation | 48 | Muestra el icono Mensaje de advertencia. |
| vbInformation | 64 | Muestra el icono Mensaje de información. |
| vbDefaultButton1 | 0 | El primer botón es el predeterminado. |
| vbDefaultButton2 | 256 | El segundo botón es el predeterminado. |
| vbDefaultButton3 | 512 | El tercer botón es el predeterminado. |
| vbDefaultButton4 | 768 | El cuarto botón es el predeterminado. |
| vbApplicationModal | 0 | Aplicación modal: el usuario debe responder al mensaje antes de continuar trabajando en la aplicación actual. |
| vbSystemModal | 4096 | Sistema modal: se suspenden todas las aplicaciones hasta que el usuario responda al cuadro de mensaje. |
| vbMsgBoxHelpButton | 16384 | Muestra el botón Ayuda. |
| VbMsgBoxSetForeground | 65536 | Especifica la ventana del cuadro de mensaje como ventana de primer plano. |
| vbMsgBoxRight | 524288 | Texto alineado a la derecha. |
| vbMsgBoxRtlReading | 1048576 | Especifica que el texto debe aparecer para ser leído de derecha a izquierda en los sistemas árabe y hebreo. |

Tabla 1

Todos estos valores los podemos dividir en cinco grupos.

1) El primer grupo (0, 1, 2, 3, 4, 5) nos permite indicar los botones que se mostrarán en el cuadro de diálogo
2) El segundo grupo (16, 32, 48, 64) determinará el tipo de ícono mostrado,
3) El tercer grupo (0, 256, 512, 768) es útil para indicar el botón predeterminado.
4) El cuarto grupo (0, 4096) es la modalidad del cuadro de diálogo
5) El último grupo, que son los valores restantes, nos permiten indicar la alineación del texto y si deseamos mostrar un botón de Ayuda.

Ya que los valores de cada grupo son excluyentes, solo tiene sentido elegir un valor de cada uno de ellos. De esta manera, si deseo mostrar los botones Si y No, y además mostrar un icono de mensaje de advertencia, entonces debo utilizar la siguiente instrucción:

```
1  Private Sub CommandButton3_Click()
2  MsgBox "Hola Mundo", vbYesNo + vbExclamation, "Mensaje especial"
3  End Sub
```

Para el segundo argumento podemos utilizar las constantes definidas para cada opción o utilizar directamente el valor numérico correspondiente de acuerdo a la tabla. El resultado de esta instrucción será el siguiente.

Para cada opción adicional debemos agregarla utilizando el símbolo de suma (+).

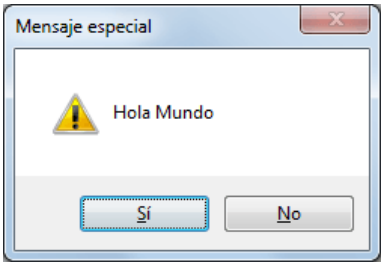



Ilustración 3

### 1.6 *Ejemplo 4: Conocer el botón pulsado*

Para conocer el botón que ha sido pulsado por el usuario, es necesario guardar el valor devuelto por la función MsgBox en una variable:

```
resultado = MsgBox("Hola Mundo", vbAbortRetryIgnore
+ vbQuestion, "Mensaje especial")
```

Antes de continuar debemos tener en cuenta algo importante sobre esta instrucción ya que, a diferencia de las anteriores, en esta ocasión los argumentos de la función MsgBox están encerrados en paréntesis. Esto se debe a una regla de programación en VBA la cual indica que cuando se llama a una función que devolverá un valor el cual será asignado a una variable, su lista de argumentos deberá estar rodeada por paréntesis. Así que no podemos olvidar colocar los paréntesis si queremos conocer el botón pulsado por el usuario.

Ahora bien, la variable resultado guardará el valor devuelto por la función MsgBox el cual puede ser cualquiera de las siguientes opciones dependiendo los botones que hayamos decidido mostrar:

| Constante | Valor | Descripción |
|---|---|---|
| vbOK | 1 | Aceptar |
| vbCancel | 2 | Cancelar |
| vbAbort | 3 | Anular |
| vbRetry | 4 | Reintentar |
| vbIgnore | 5 | Omitir |
| vbYes | 6 | Si |
| vbNo | 7 | No |

Tabla 2

Una vez que el usuario hace clic sobre un botón, la función MsgBox nos devuelve el valor correspondiente y podremos comparar dicho valor con las contantes mencionadas en la tabla anterior. Para nuestro ejemplo, utilizaré una sentencia *Select Case* para comparar la variable *resultado* con todas las opciones posibles:

```
Private Sub CommandButton4_Click()
resultado = MsgBox("Hola Mundo", vbAbortRetryIgnore + vbQuestion, "Mensaje especial")
Select Case resultado
    Case vbOK:
        MsgBox "Se pulsó el botón Aceptar"
    Case vbCancel:
        MsgBox "Se pulsó el botón Cancelar"
    Case vbAbort:
        MsgBox "Se pulsó el botón Anular"
    Case vbRetry:
        MsgBox "Se pulsó el botón Reintentar"
    Case vbIgnore:
        MsgBox "Se pulsó el botón Omitir"
    Case vbYes:
        MsgBox "Se pulsó el botón Si"
    Case vbNo:
        MsgBox "Se pulsó el botón No"
End Select
End Sub
```

Una vez identificado el botón sobre el cual el usuario ha hecho clic, mostraremos otro mensaje con el nombre de dicho ºbotón. En un caso real, esa línea de código sería reemplazada por las instrucciones que se deseas ejecutar de acuerdo a la respuesta del usuario. En nuestro ejemplo mostramos un cuadro de diálogo con los botones Anular, Reintentar y Omitir, así que si el usuario hace clic sobre el botón Reintentar se mostrará el mensaje que nos confirmará dicha acción.

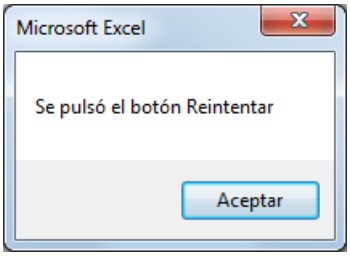


Ilustración 4

## 1.7 *Ejemplo 5: El botón de Ayuda*

Con los ejemplos realizados hasta el momento hemos cubierto la gran mayoría de opciones de uso de la **función MsgBox en VBA**. La verdad es que los últimos dos argumentos de la función son muy poco utilizados pero haremos un último ejemplo para mostrar su uso.

En el ejemplo 3 mostramos la tabla de posibles opciones para el argumento *Buttons* y una de ellas es la opción *vbMsgBoxHelpButton* la cual ocasionará que se muestre un botón con la leyenda Ayuda dentro del cuadro de diálogo. El único objetivo de este botón es mostrar un archivo externo con ayuda adicional para el usuario sobre los botones que puede a pulsar.

Para que el botón de Ayuda funcione correctamente es necesario utilizar el cuarto y quinto argumento de la función MsgBox. En el cuarto argumento debemos indicar la ruta donde se encuentra el archivo de ayuda, que generalmente es un archivo CHM, y el quinto argumento será utilizado para indicar el número del tema que será mostrado. Este último argumento es una característica de los archivos de ayuda, y es una configuración que se hace al construir ese tipo de archivos.

Para nuestro ejemplo utilizaremos un archivo de ayuda previamente elaborado y donde sabemos que existe el tema de ayuda 20000 y por lo tanto podemos utilizar las instrucciones siguientes para mostrarlo:

```
Private Sub CommandButton5_Click()
strRuta = ThisWorkbook.Path & "CHM-example.chm"
MsgBox "Hola Mundo", vbOKCancel + vbCritical + vbMsgBoxHelpButton, _
"Mensaje especial", strRuta, 20000
End Sub
```

Para este ejemplo es necesario colocar el archivo CHM en la misma carpeta que el libro de Excel ya que hemos utilizado la instrucción *ThisWorkbook.Path* para indicar que el archivo de ayuda se encuentra en la misma ubicación. Al hacer clic sobre el botón Ayuda, se abrirá una nueva ventana que nos permitirá visualizar el archivo indicado.

Con este último ejemplo hemos cubierto todos los argumentos de la **función MsgBox en VBA** y estamos listos para comenzar a crear nuestros propios mensajes dentro de nuestras aplicaciones de Excel.

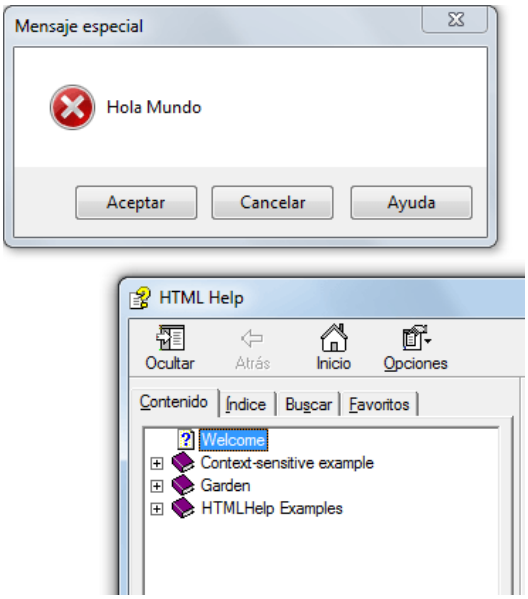



Ilustración 5

# 2 Conceptos y Tipos de UsersForms (Formularios)

## 2.1 *Objetivos Básicos de los Formularios en Excel*

Los cuadros de dialogo personalizados (UserForms) tienen como objetivos básicos:

| 1.- Introducción de Datos por parte del Usuario |
|---|
| **Función: InputBox** |
| **Método: InputBox**<br>*object .InputBox (Argumentos)* |

| 2.- Mostrar Mensajes y conseguir respuestas sencillas |
|---|
| **Función: MsgBox de VBA**<br>*MsgBox ("¿Continuar?", vbYesNo)* |

| 3.- Seleccionar un archivo o cuadro de dialogo |
|---|
| **Método: GetOpenFilename**<br>*object .GetOpenFilename (Argumentos)* |
| **Método: GetSaveAsFilename**<br>*object .GetSaveFilename (Argumentos)* |

| 4.- Mostrar los Cuadros de Dialogo Integrados en Excel |
|---|
| **Colección: Dialogs**<br>*Application. Dialogs (Argumentos)* |
| **CommandBars**<br>*Application CommandBars (Argumentos)* |

**Ilustración 6**

Debemos señalar que los Formularios de Excel conocidos como USERFORM o Formularios de Usuario tienen limitaciones y no poseen las características de los formularios de VBA que poseen características más avanzadas.

Los formularios Excel tienen una gran variedad de controles, los cuales tienen una variedad de propiedades, funciones y eventos.

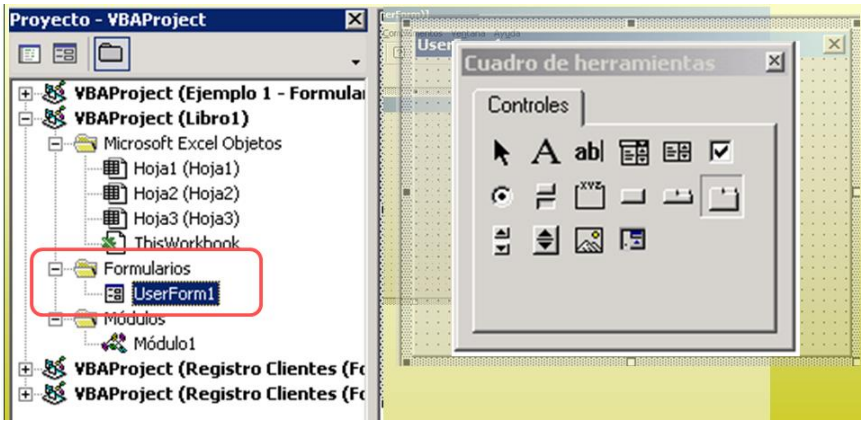



**Ilustración 7**

## 2.2 *Jerarquía de Objetos de UserForm*

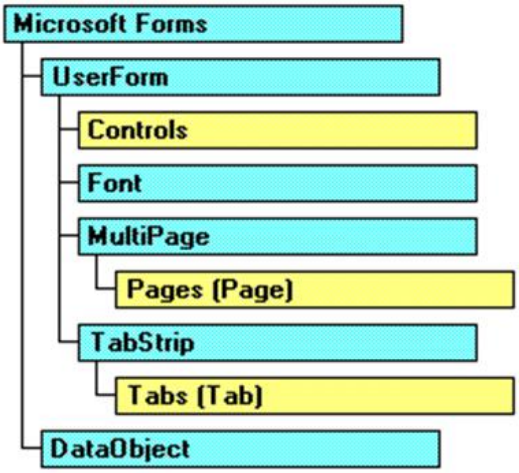



**Ilustración 8**

## 2.3 *Secuencia Básica en la Elaboración de los Cuadros de Dialogo Personalizados*

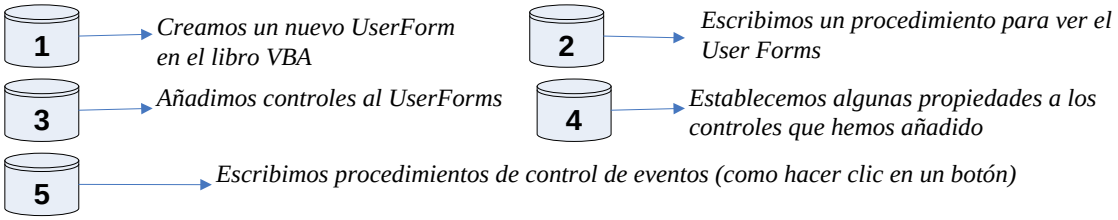
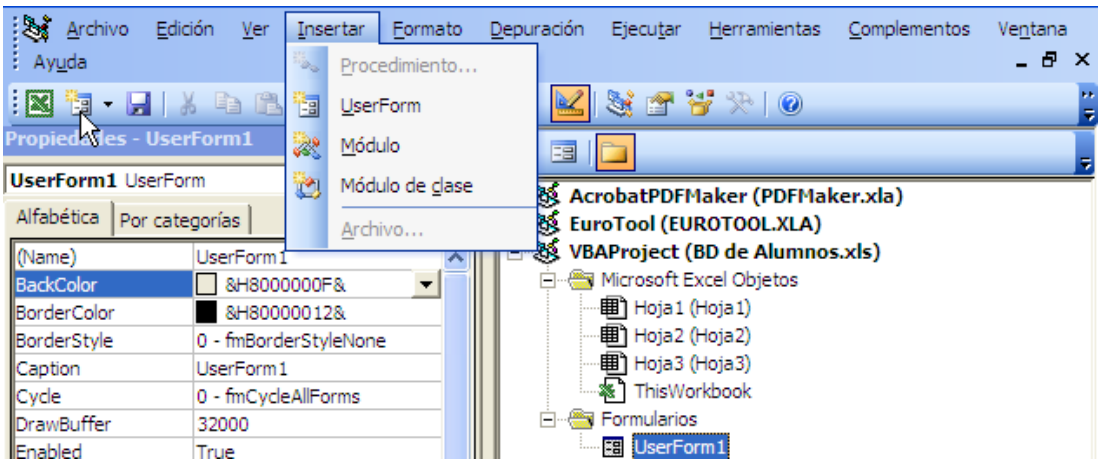



**Ilustración 9**

## 2.4 *Insertar y Mostrar los Formularios.*

### 2.4.1 Crear un Userform

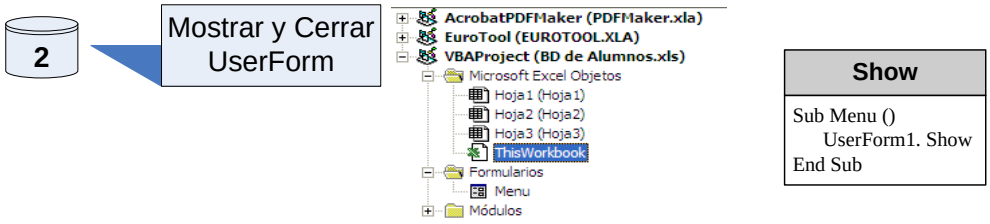

*Seleccionado el proyecto donde queremos crear el UserForm, escogemos Insertar>UserForm o bien el icono de acceso directo disponible al efecto*



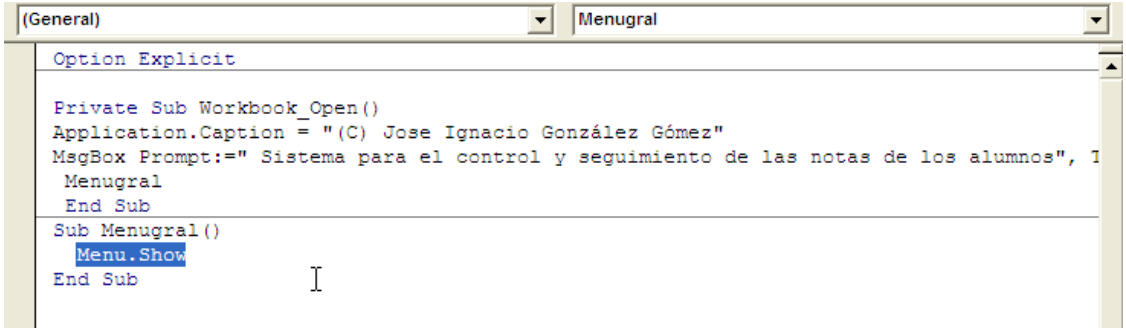

*Por defecto los UserForm tienen nombres como UserForm1, UserForm2, etc.. Para poderlo identificar más faciltmente podemos cambiarle el nombre en la ventana de propiedades del UserForm, en concreto en la propiedad Name*

**Ilustración 10**

### 2.4.2 Mostrar/Ocultar y Cerrar un UserForm




*Es necesario crear un procedimiento que debe situarse en un módulo estandar y no en el módulo de código del UserForm.*




*Para ocultar un formulario utilizamos el comando Hide* Userform1. Hide

*Para cerrar un formulario utilizamos el comando Unload* Unload Userform1

**Ilustración 11**

Por tanto para abrir un formulario la sentencia sería:

Menu Show

Para cerrar un formulario concreto la sentencia seria

Unload Menu

# 3 Controles y Propiedades en los Formularios

## 3.1 *Añadir Controles a los Formularios*


*Para añadir controles usamos el cuadro de herramientas. Si añadimos un botón de comando al formulario este tomara como nombre CommandButton1, CommandButton2, etc.. Es conveniente cambiarles el nombre para que sean más intuitivos al referirnos a ellos*

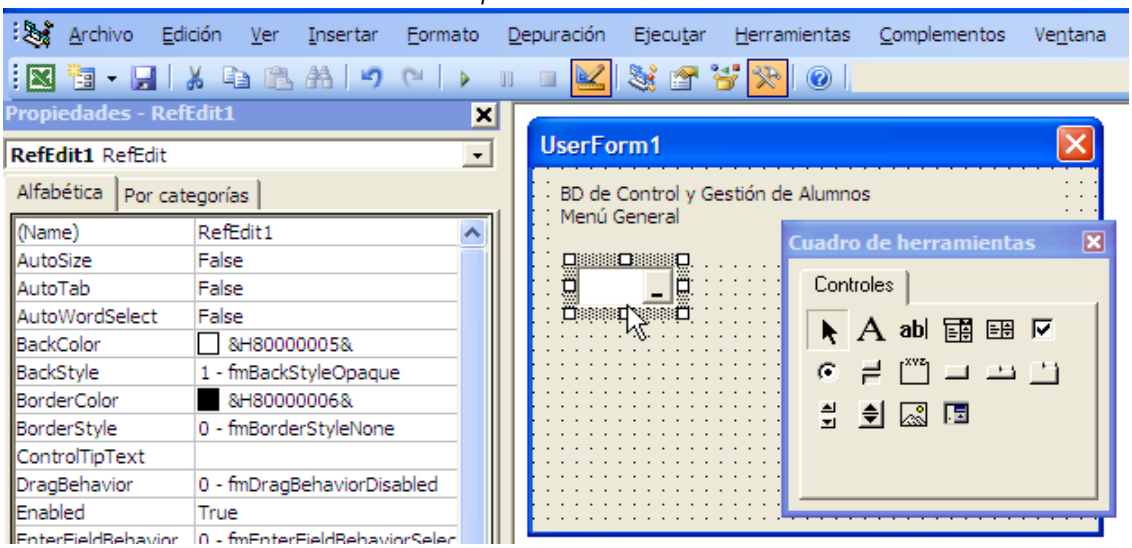
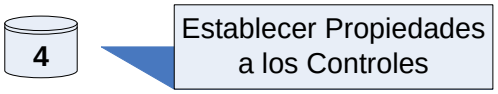


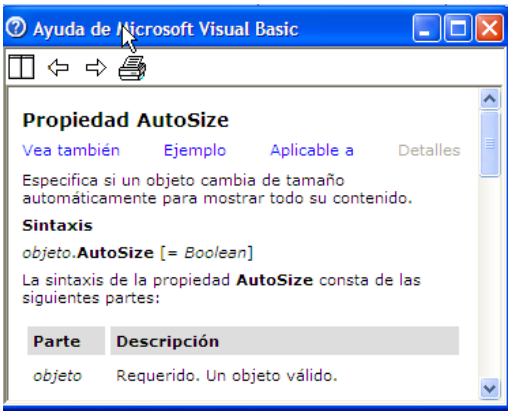

**Ilustración 12**

Un Control específico del Excel es el RefEdit que se utiliza cuando es necesario permitir al usuario seleccionar un rango de una hoja. Los demás controles son los típicos de todas las aplicaciones Microsoft.

## 3.2 *Establecer Propiedades a los Controles de los Formularios.*




*Cada control tiene varias propiedades que determinan el aspecto del control y su comportamiento. Estas propiedades se modifican en la ventana de Propiedades.*
*Aunque cada control tiene su propio conjunto de propiedades, muchos controles tienen algunas propiedades comunes, como Name, Font, etc.*
*Si seleccionamos en un formulario dos o mas controles y acudimos a la ventana de propiedades, solo se mostrara las propiedades compartidas por los controles.*
*Es conveniente, especialmente si vamos a manipular un control en VBA asignarle un nombre con significado más ajustado*
*Las mejores formas de conocer las propiedades de un control es a traves de la ayuda F!*



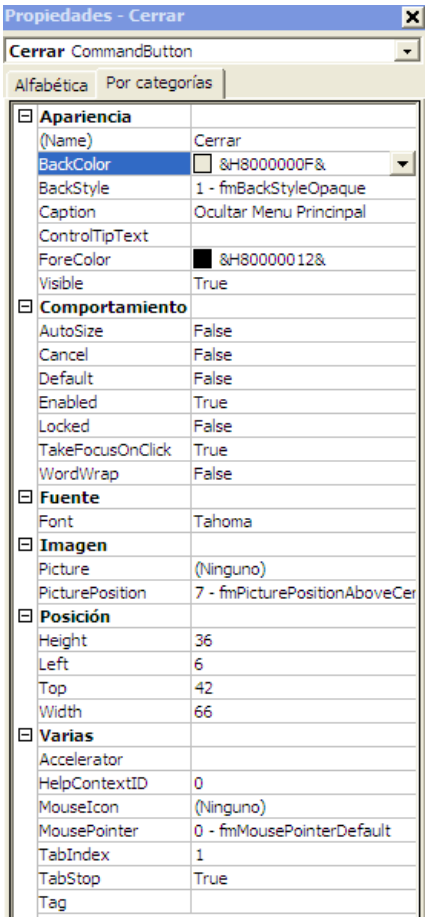




**Ilustración 13**

Existen diferentes tipos de propiedades del Formulario como

- Name
- Backcolor
- Caption
- Enable
- ScrollBars

Como ejemplo podemos ver también las propiedades más importantes de un ComboBox.

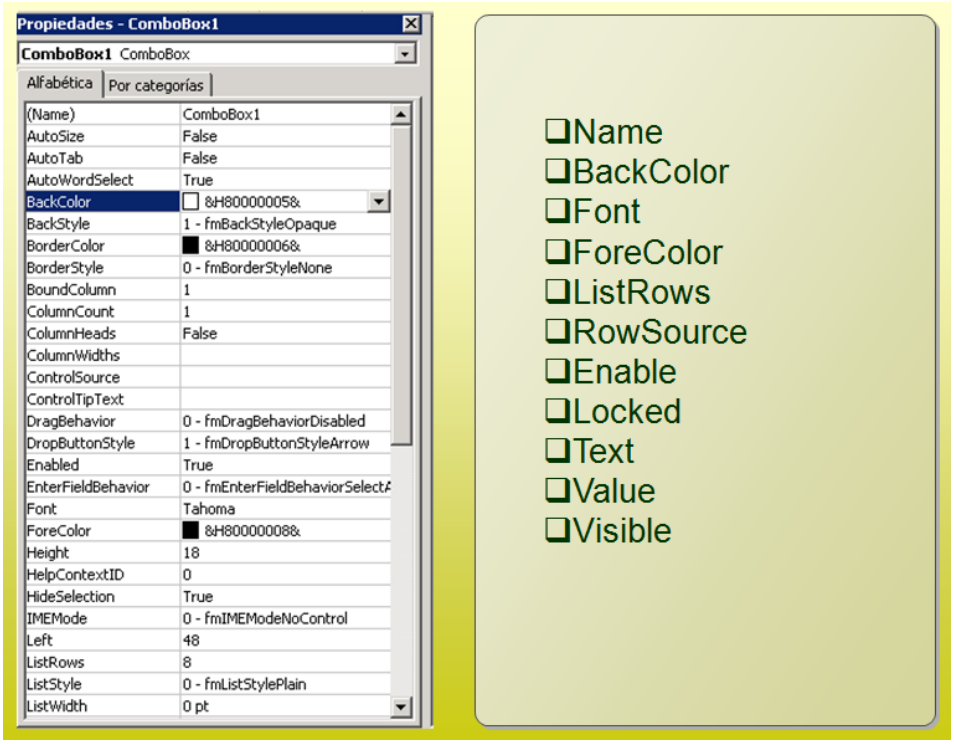


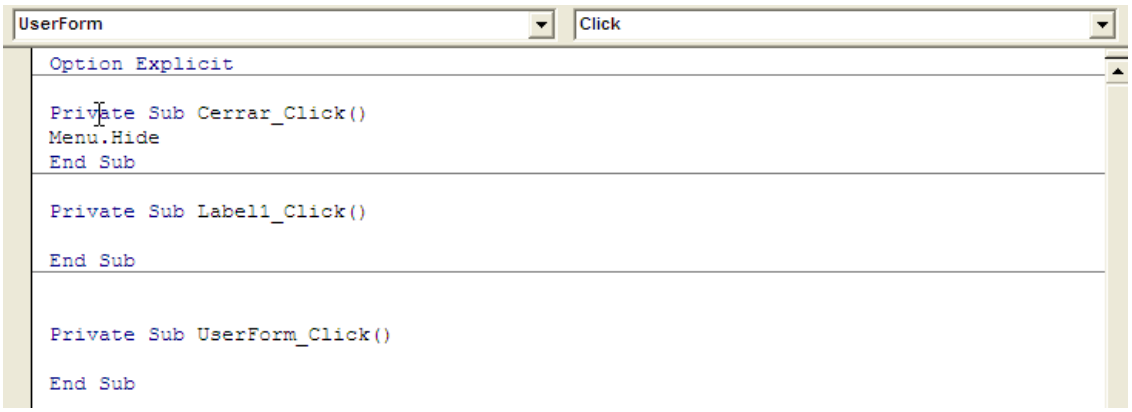

**Ilustración 14**

## 3.3 Procedimientos de Control de Eventos en Formularios.

5 **Procedimientos de Control de Eventos**

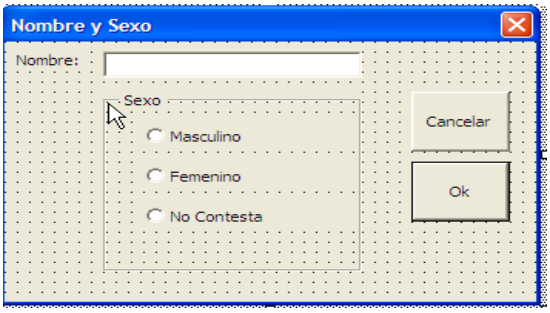
*Los usuarios interactuan con los UserForms de diversas formas, seleccionando una opción, haciendo clic, a todo esto se le denomina como hemos visto EVENTOS*

*Para que los eventos funcionen es necesario programarlos y para ello se hace uso de los PROCEDIMIENTOS que se ejecute cuando suceden estos eventos*




*Los Procedimientos correspondiente a cada evento (por ejemplo el clic sobre el botón cerrar de un Userform) deben colocarse en la ventana de código del propio UserForm que estamos tratando.*

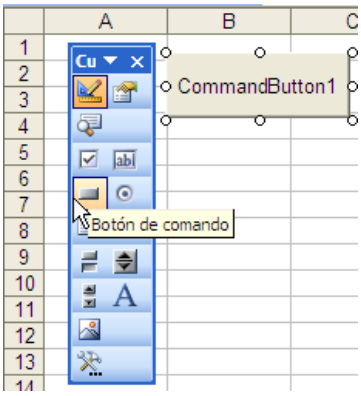
**EJEMPLO** *Vamos a crear un formulario para que nos de el nombre y sexo de un conjunto de personas*

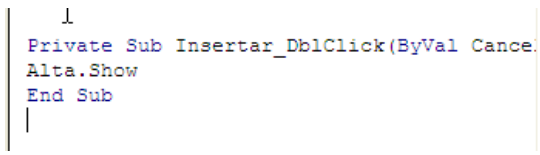



1 *Creamos el formulario como hemos visto anteriormente*

2 *Añadimos los controles correspondientes y los botones de comando*

3 *Creamos un botón en la hoja de calculo para acceder al formulario y lo progrmamos*




*Para programar los procedimientos de los eventos del formulario, tenemos que acudir al codigo del UserForm y en concreto al botón que hemos insertado en el mismo y que por tanto queremos programar su evento como puede ser el de Cancelar*

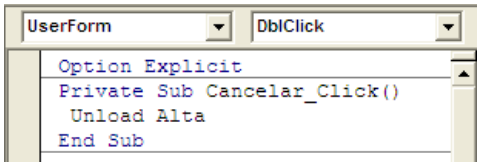



**Ilustración 15**

## 3.4 Tipos de Evento más comunes de un Control

Para saber qué tipos de eventos de un control, basta analizar el cuadro desplegable de eventos del control en concreto

Para conocer los destalles específicos de un evento podemos consultar la ayuda. Los procedimientos de Control de Eventos incorporan el nombre del objeto en el nombre del procedimiento, por tanto si cambiamos el nombre del control, debemos tener en cuenta que los cambios de nombre no se realizan automáticamente. Para facilitar las cosas, es

aconsejable proporcionar nombres a los controles antes de empezar a crear procedimientos de control de eventos

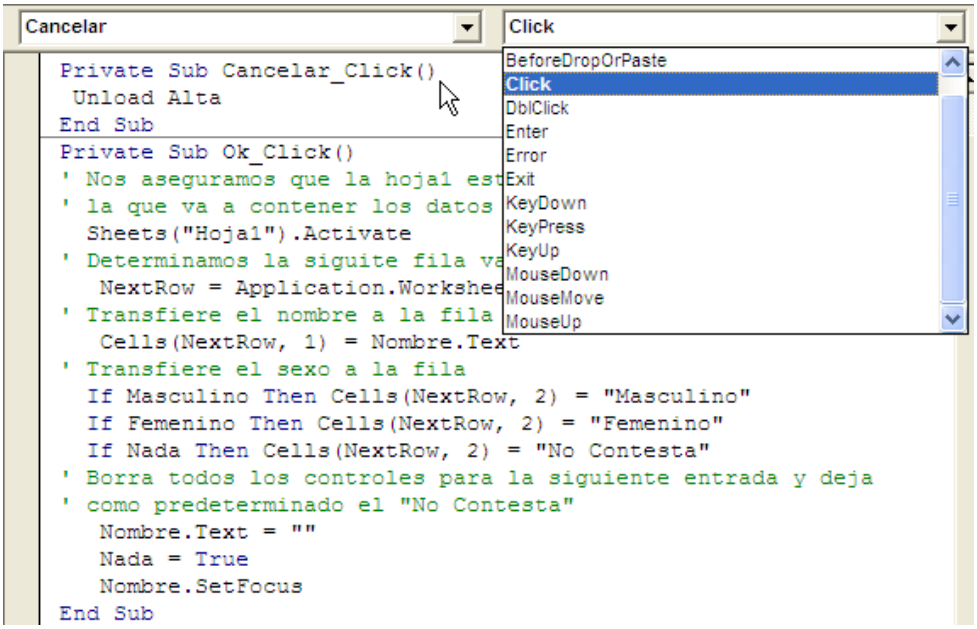



Ilustración 16

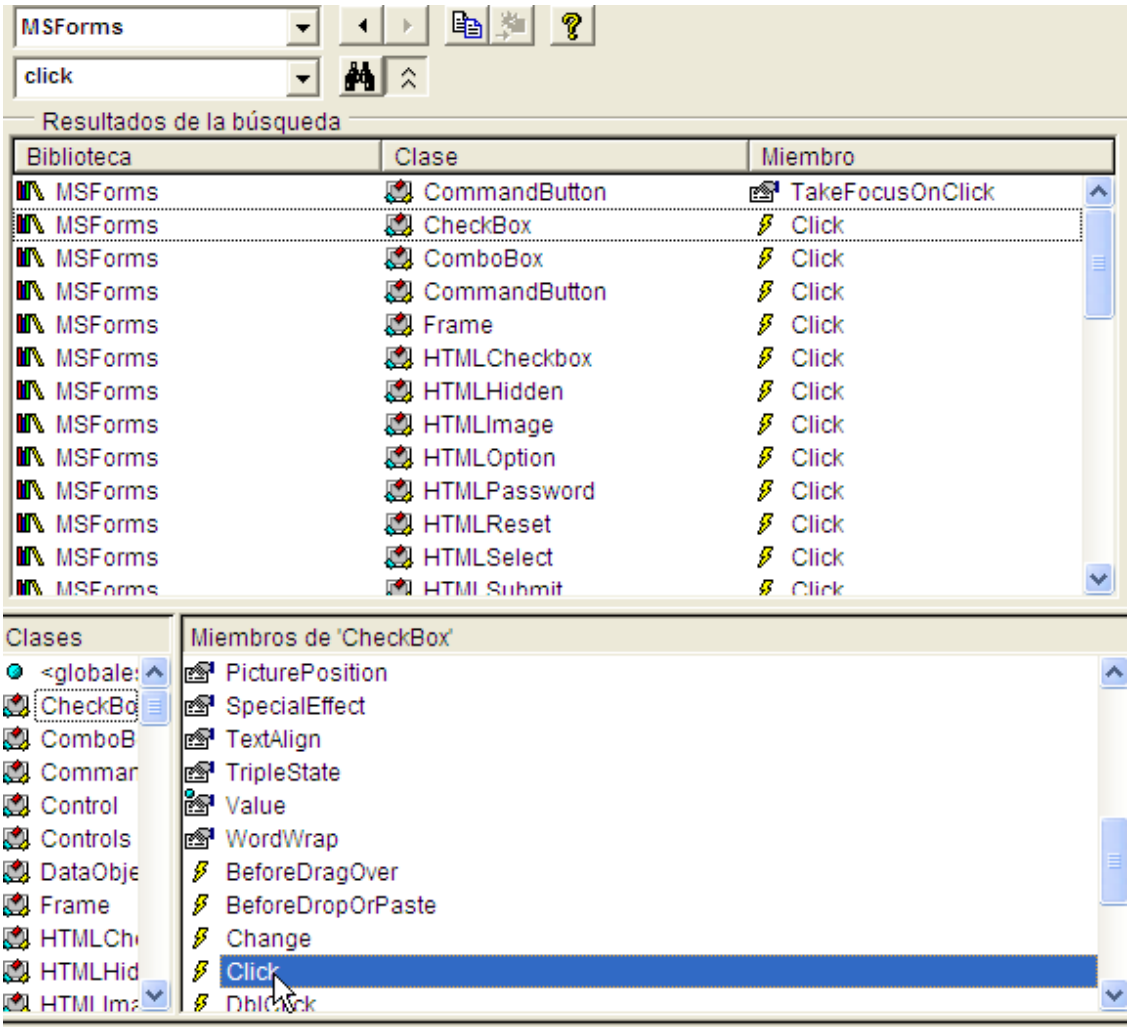
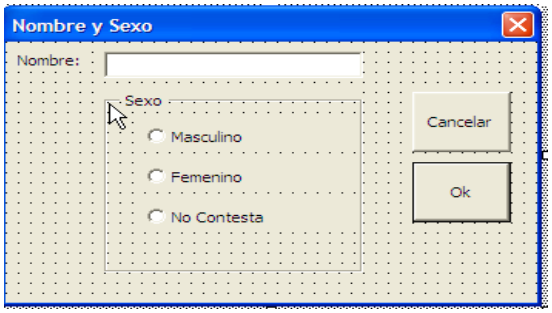



Ilustración 17

# 4 Ejemplo de Programación de un UserForm (Formulario)

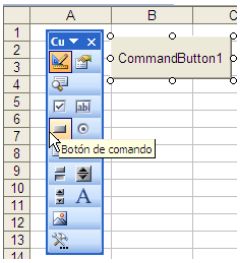
Ejemplo 1
Programación de UserForm

*Vamos a crear un formulario para que nos de el nombre y sexo de un conjunto de personas*




1 *Creamos el formulario como hemos visto anteriormente*

2 *Añadimos los controles correspondientes y los botones de comando*




3 *Creamos un botón en la hoja de calculo para acceder al formulario y lo progrmamos*

```
Private Sub Insertar_DblClick(ByVal Cance
Alta.Show
End Sub
```

*Para programar los procedimientos de los eventos del formulario, tenemos que acudir al codigo del UserForm y en concreto al botón que hemos insertado en el mismo y que por tanto queremos programar su evento como puede ser el de* ***Cancelar***

UserForm | DblClick

```
Option Explicit
Private Sub Cancelar_Click()
 Unload Alta
End Sub
```

*Este provcedimiento (Unload Alta) lo que hace es descargar el UserForm (Alta) Nombre y Sexo de la memoria*

*Pasamos a programar a continuación el botón* ***Ok***
*Hacemos doble click en boton Ok del UserForm Altas y nos introducimos en el control de eventos para el evento click del botón OK*

*En concreto vamos a programar este evento para que almacene los valores en la hoja de calculo, tal y como se muestra a continuación.*

Ok | Click

```
Option Explicit
Private Sub Cancelar_Click()
 Unload Alta
End Sub
Private Sub Ok_Click()

End Sub
```

Ok | Click

```
Private Sub Ok_Click()
' Nos aseguramos que la hoja1 esta activa que es
' la que va a contener los datos del Userform
  Sheets("Hoja1").Activate
' Determinamos la siguite fila vacia de la hoja activa
   NextRow = Application.WorksheetFunction.CountA(Range("A:A")) +
' Transfiere el nombre a la fila
   Cells(NextRow, 1) = Nombre.Text
' Transfiere el sexo a la fila
  If Masculino Then Cells(NextRow, 2) = "Masculino"
  If Femenino Then Cells(NextRow, 2) = "Femenino"
  If Nada Then Cells(NextRow, 2) = "No Contesta"
' Borra todos los controles para la siguiente entrada y deja
' como predeterminado el "No Contesta"
   Nombre.Text = ""
   Nada = True
   Nombre.SetFocus
End Sub
```

*Asi funciona el procedimiento:*
*1) Se asegura que la hoja apropiada (hoja1) esta activda.*
*2) Despues utiliza la función de Excel CONTARA para determinar la siguiente celda en blanco de la columna A*
*3) Transfiere el Cuadro del Texto Nombre a la Columna A que esta libre*
*4) Igual hace con los Generos*
*5) Finalmente prepara el formulario para dejarlo en blanco*

**Ilustración 18**

# 5 Bibliografía

https://exceltotal.com/la-funcion-msgbox-en-vba/