Jose Ignacio González Gómez.
Departamento de Economía Financiera y Contabilidad
Universidad de La Laguna
http://www.jggomez.eu/

Tema:

Apuntes Programación Excel VBA.

PARTE IV: Excel como base de datos. Trabajando con Userform

Indice

1 Excel como herramienta de Base de Datos.

1.1 Aspectos generales y limitaciones.

1.2 Evitar introducir valores duplicados, repetidos en una tabla Excel.

Fuente (16/03/2009) : http://hojas-de-calculo-en-excel.blogspot.com/2008/02/introducir-valores-nicos-no-repetidos.html Javier Marco

Excel, a través de la validación de datos, nos brinda la oportunidad de poder introducir valores únicos, es decir, no repetidos. Imaginemos que tenemos una hoja de cálculo, donde solo permitimos la entrada de datos en determinado rango de celdas, para lo cual, deberemos tener protegida la hoja de cálculo, y con todas las celdas bloqueadas, excepto esas celdas donde permitiremos la introducción de datos.

Si deseamos valores únicos (no repetidos), en ese rango de celdas donde vamos a permitir la introducción de datos (en nuestro ejemplo supondremos que el rango va desde A1 hasta A30), tan solo tendremos que seleccionar la primera celda del rango (en nuestro ejemplo A1), y en el menú Datos, seleccionaremos Validación. En el criterio de validación, seleccionaremos Personalizada, y en el apartado donde pone Fórmula, introduciremos esta:

```
=CONTAR.SI($A$1:$A$30;A1)=1
```

De tal forma que nos quede como en esta imagen:

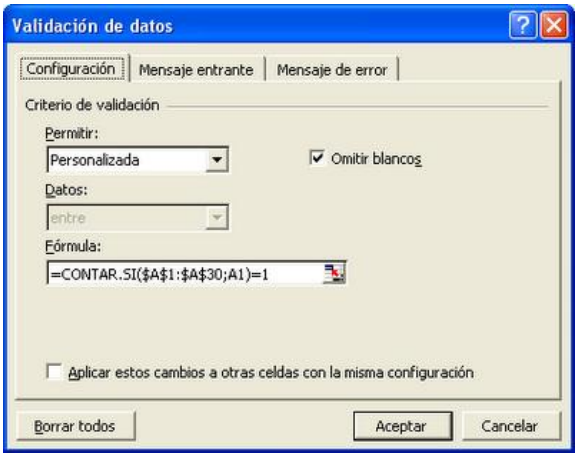


Ilustración 1

Ahora, tan solo nos quedará copiar la celda A1, en el resto de celdas del rango donde vamos a permitir la introducción de datos (en nuestro ejemplo, desde A2 hasta A30).

Si lo hemos hecho todo correctamente, en la celda A30 deberíamos tener esta fórmula de validación (la veremos desde el menú Datos, Validación:

```
=CONTAR.SI($A$1:$A$30;A30)=1
```

Ahora vamos a ver cómo funciona. Vamos a introducir una lista de datos que no se repitan desde A1 hasta A8, tal y como muestra la siguiente imagen:

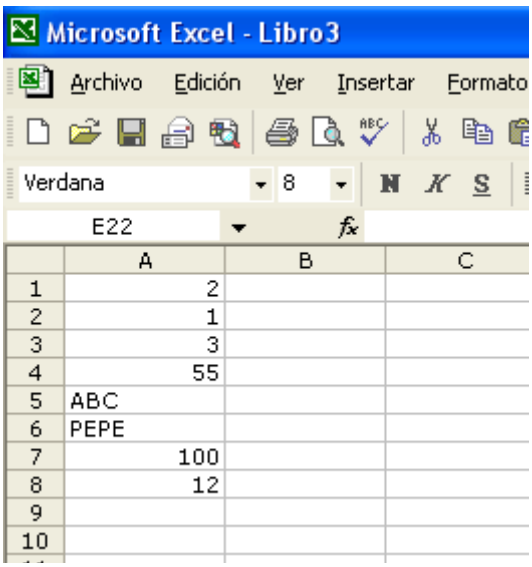


Ilustración 2

Ahora, si por ejemplo en la celda A9, introducimos una cifra que ya exista en alguna de las celdas anteriores (por ejemplo, introducimos el número 55, que ya tenemos en la ceda A4), nos aparecerá un mensaje de aviso (el que sale por defecto, aunque se puede personalizar a nuestro gusto, también desde el menú **Datos**, **Validación**) como este:

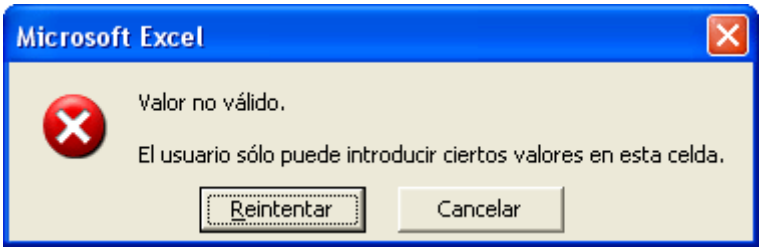


Ilustración 3

Y eso es todo. Un sencillo ejemplo de cómo podemos limitar las entradas de excel, para que no se repitan los datos.

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 *object .InputBox (Argumentos)*

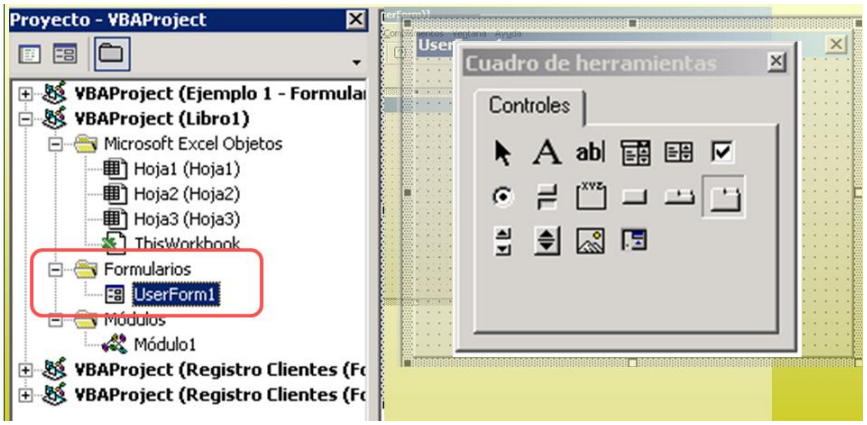
2.- Mostrar Mensajes y conseguir respuestas sencillas
Función: MsgBox de VBA *MsgBox ("¿Continuar?", vbYesNo)*

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

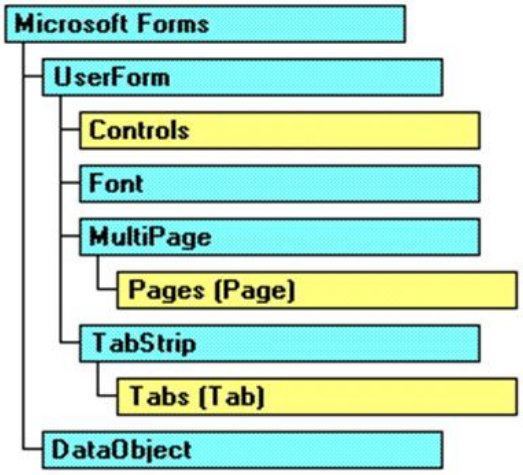
4.- Mostrar los Cuadros de Dialogo Integrados en Excel
Colección: Dialogs *Application. Dialogs (Argumentos)*
CommandBars *Application CommandBars (Argumentos)*

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.



2.2 Jerarquía de Objetos de UserForm



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

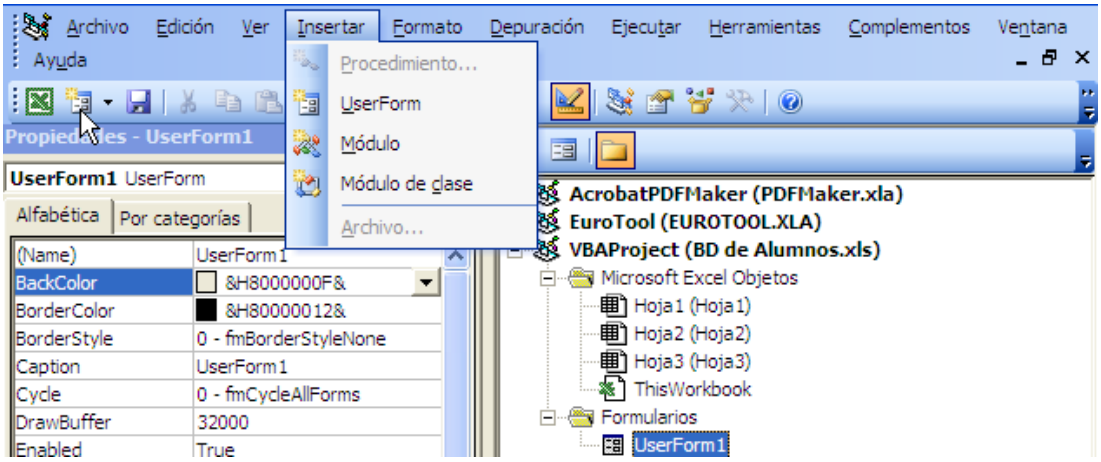
1. *Creamos un nuevo UserForm en el libro VBA*
2. *Escribimos un procedimiento para ver el User Forms*
3. *Añadimos controles al UserForms*
4. *Establecemos algunas propiedades a los controles que hemos añadido*
5. *Escribimos procedimientos de control de eventos (como hacer clic en un botón)*

2.4 Insertar y Mostrar los Formularios.

2.4.1 Crear un Userform

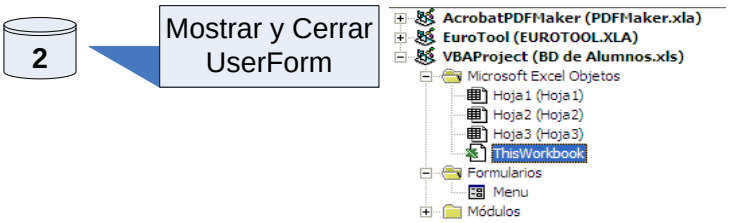
1. Insertar un UserForms

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

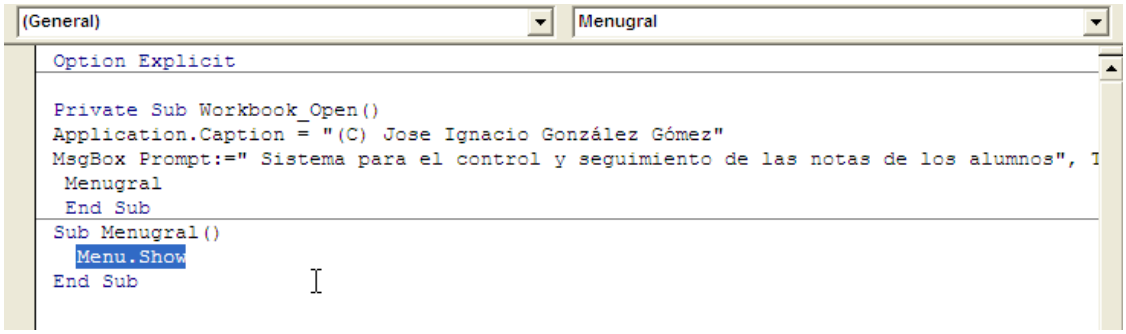


Por defecto los UserForm tienen nombres como UserForm1, UserForm2, etc.. Para poderlo identificar más faciltmente podemos cambiarle el nombre en la ventana de propiedades del UserForm, en concreto en la propiedad Name

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

Por tanto para abrir un formulario la sentencia sería:

Menu Show

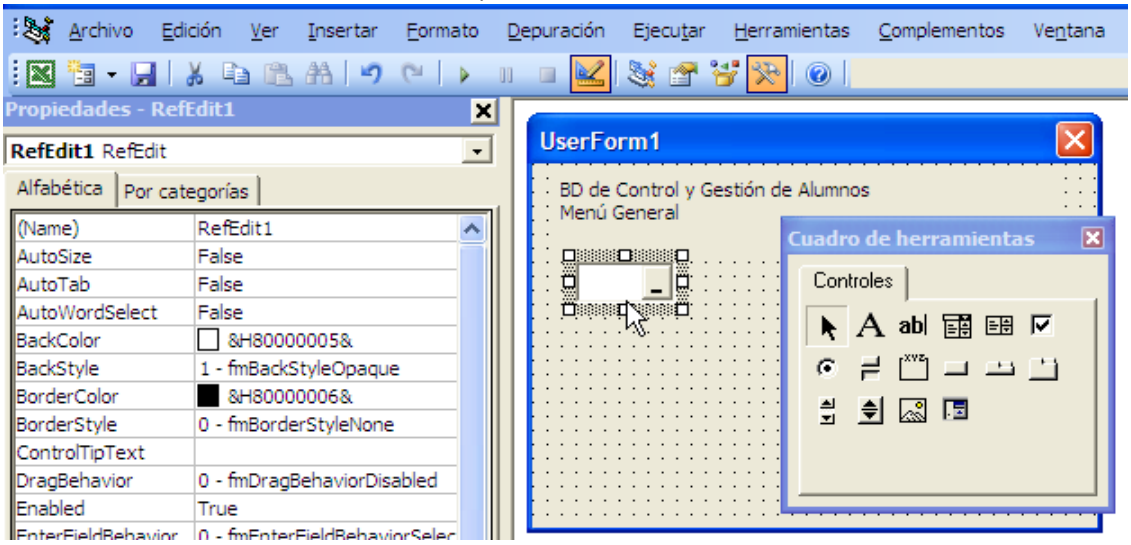
Para cerrar un formulario concreto la sentencia seria

Unload Menu

2.5 Controles y Propiedades en los Formularios

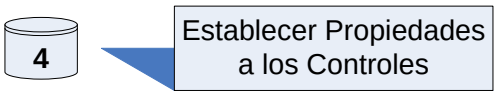
2.5.1 Añadir Controles a los Formularios

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

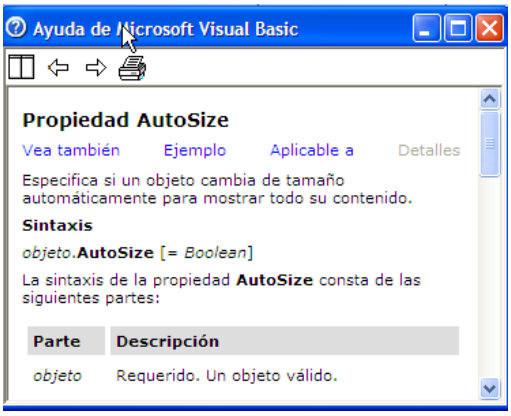


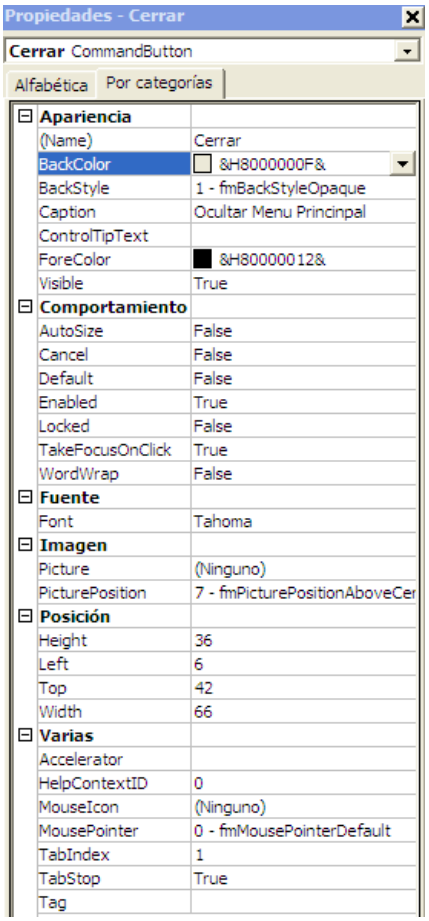
Un Control especifíco del Excel es el RefEdit que se utiliza cuando es necesario permitir al usuario seleccionar un rango de una hoja. Los demas controles son los tipicos de todas las aplicaciones Microsoft.

2.5.2 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!

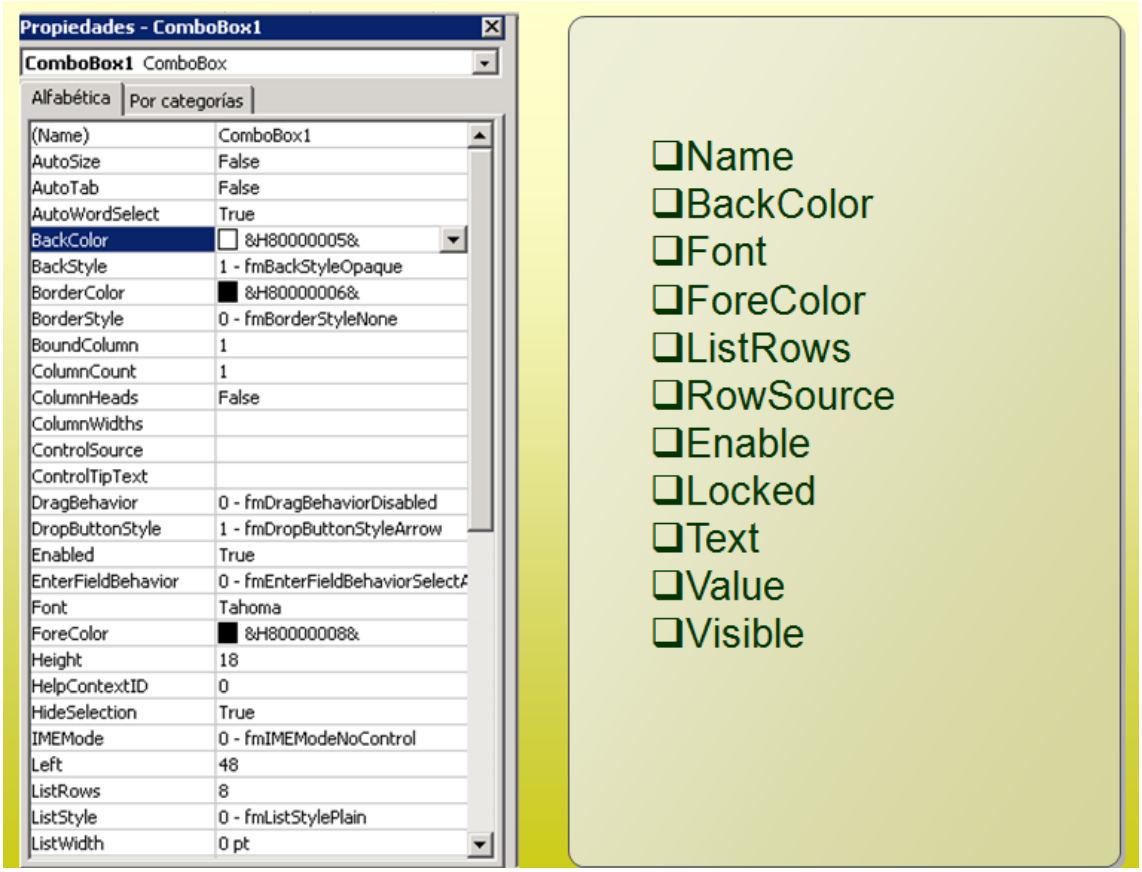




Existen diferentes tipos de propiedades del Formulario como

- Name
- Backcolor
- Caption
- Enable
- ScrollBars

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

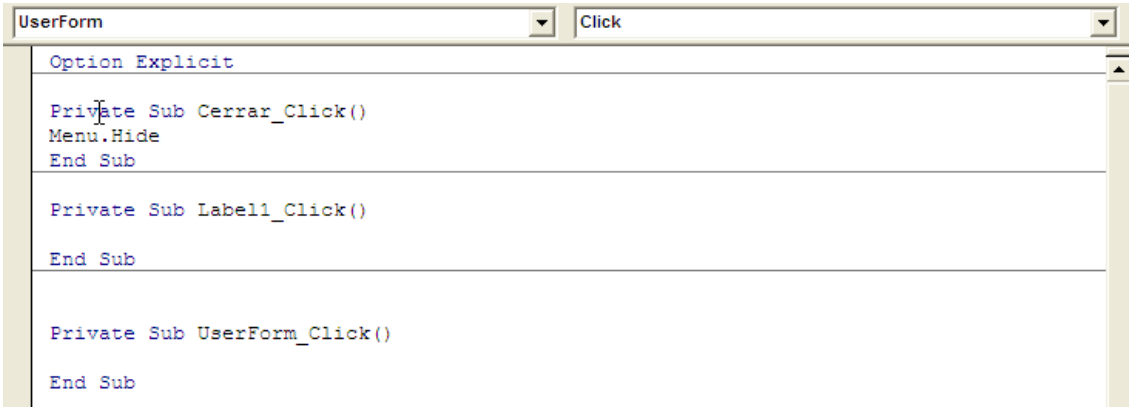


2.6 Procedimientos de Control de Eventos en Formularios.

5 Procedimientos de Control de Eventos

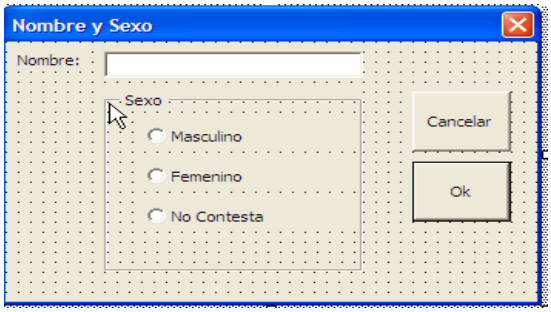
Los usuarios interactuan con los UserForms de diversas formas, seleccionando una opción, haciendo clic, a todo esto se le denomina como hemos visto EVENTOS

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.

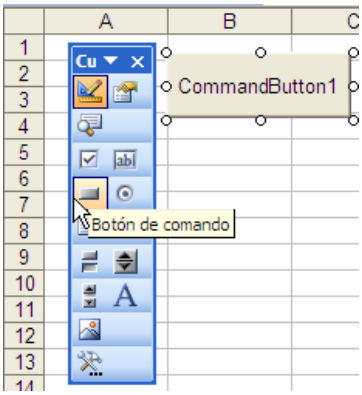
EJEMPLO *Vamos a crear un formulario para que nos de el nombre y sexo de un conjunto de personas*

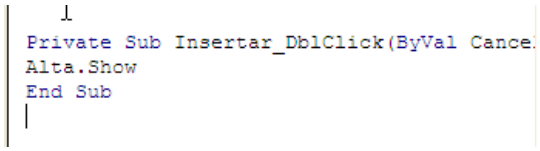


1 *Creamos el formulario como hemos visto anteriormente*

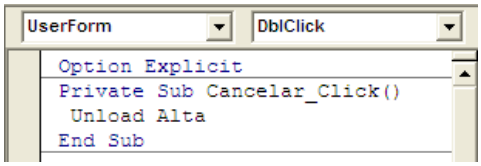
2 *Añadimos los controles correspondientes y los botones de comando*

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





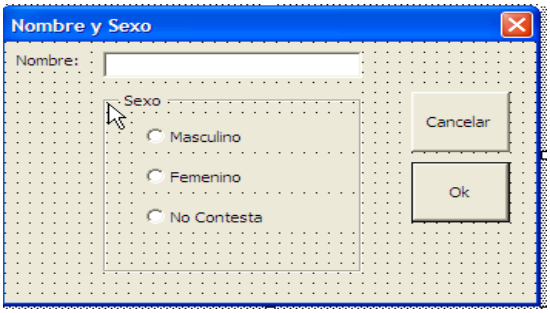
Para programar los procedimientos de los eventos del formulario, tenemos que acudir al codigo del UserForm y en concreto al botón que hemos insertado en el mismo y que por tanto queremos programar su evento como puede ser el de Cancelar



2.7 Ejemplo de Programación de un UserForm (Formulario)

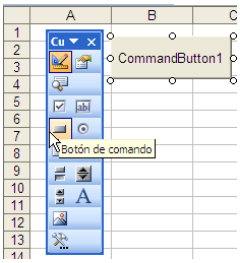
Ejemplo 1
Programación de UserForm

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*

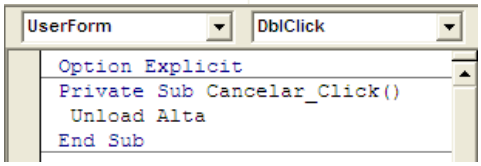
2 *Añadimos los controles correspondientes y los botones de comando*



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
```

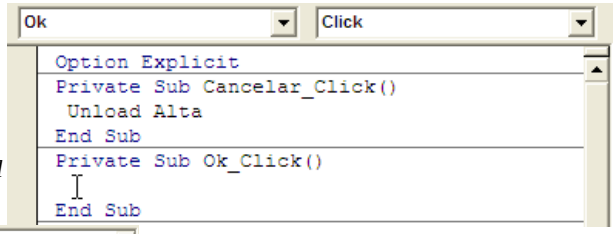
Para programar los procedimientos de los eventos del formulario, tenemos que acudir al codigo del UserForm y en concreto al botón que hemos insertado en el mismo y que por tanto queremos programar su evento como puede ser el de ***Cancelar***

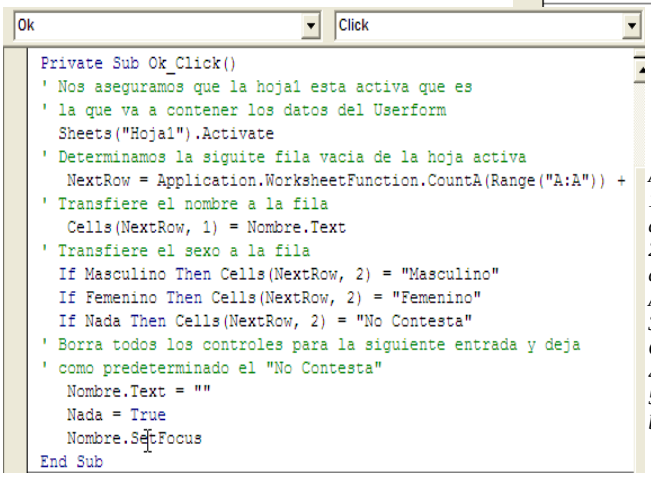


Este provcedimiento (Unload Alta) lo que hace es descargar el UserForm (Alta) Nombre y Sexo de la memoria

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.

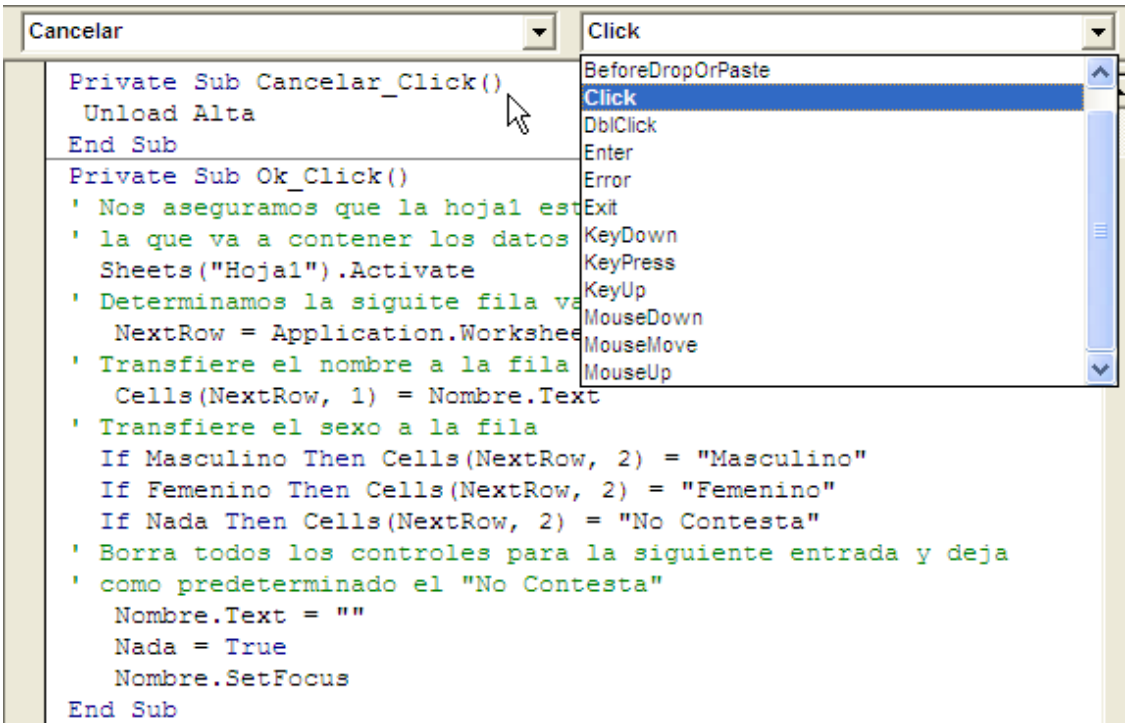




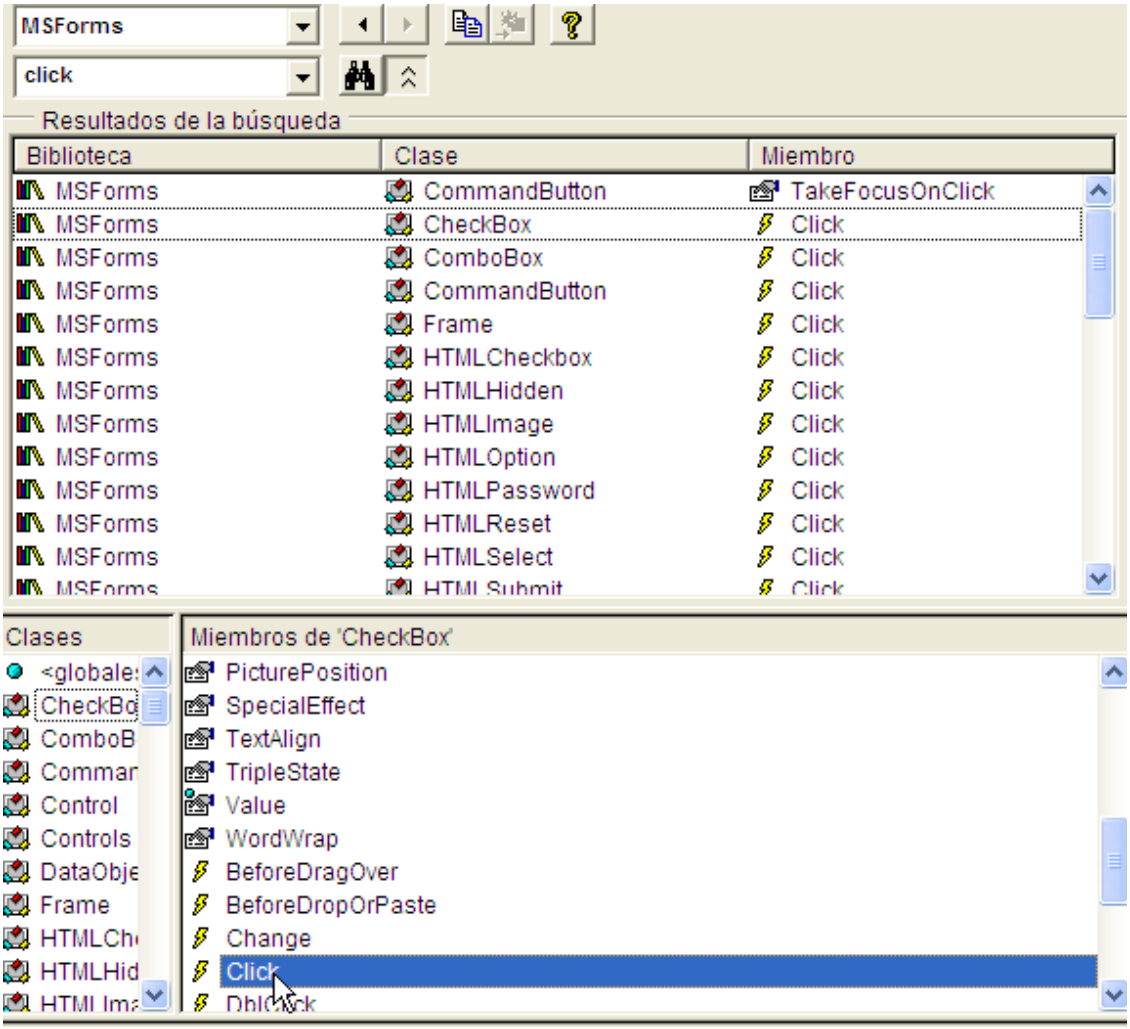
Asi funciona el procedimiento:
1) Se asegura que la hoja apropiada (hoja1) esta activda.
2) Despues utiliza la función de Excel CONTARA para determinar la siguiente celda en blanco de la columna A
3) Transfiere el Cuadro del Texto Nombre a la Columna A que esta libre
4) Igual hace con los Generos
5)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

2.8 Tipos de Evento de Un Control (Eventos más Comunes)

Para saber que 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



3 Código, Módulos y Procedimientos

Código:
Realizamos acciones en VBA ejecutando código VBA.

Módulos:
Los módulos VBA están almacenados en un libro de Excel, pero vemos o editamos módulos utilizando el editor de Visual Basic (VBE). Un módulo VBA consiste en procedimientos.

Procedimientos:
Un procedimiento es básicamente una unidad de código informático que realiza alguna acción. VBA admite dos tipos de procedimientos: Sub y Function.

* Sub: Un procedimiento Sub consiste en una serie de sentencias que pueden ejecutarse de varias maneras. Aquí tienes un ejemplo de un sencillo procedimiento Sub llamado Test. Este procedimiento realiza una sencilla suma y después muestra el resultado en un cuadro de mensaje.

```
Sub Test()
Sum = 1 + 1
MsgBox "La respuesta es " & Sum
End Sub
```

* Function: Además de los procedimientos Sub, un módulo VBA también puede tener un procedimiento Function. Un procedimiento Function devuelve un solo valor (o posiblemente una matriz). Se puede invocar un Function desde otro procedimiento de VBA o podemos usarlo en una fórmula de hoja. A continuación se muestra un ejemplo de una función llamada AgregaDos:

```
Function AgregaDos(arg1, arg2)
AgregaDos = arg1 + arg2
End Function
```

4 Formularios básicos para interactuar con base de datos en Excel

4.1 Planteamiento del problema

Nuestro interés en este caso está centrado en crear un formulario de alta/registro de datos en una tabla o base de datos en Excel, por ejemplo podría ser de usuarios o partes de trabajo, utilizando controles UserForm.

Se pretende por tanto insertar datos a través de un formulario y no directamente sobre la hoja Excel, así los datos captados del formulario se volcarán en una tabla Excel. Esto nos permite validar mediante código los datos insertados, y mantener cierta integridad y formato correcto de losdatos (por ej. través del control de fecha insertaremos una fecha válida correcta en formato).

Por otro lado el insertar de forma coherente la información, facilitará la aplicación de consultas o filtros sobre la tabla de datos.

4.2 Creación de un formulario en Excel para introducir datos en una tabla, Caso Fincas.

4.2.1 Introducción

En este caso nuestro objetivo es llevar un control de las fincas a través de una tabla en Excel y para ello queremos contar con un formulario de registro de entrada, consulta y modificación similar a los de Access tal y como mostramos en la Ilustración 5

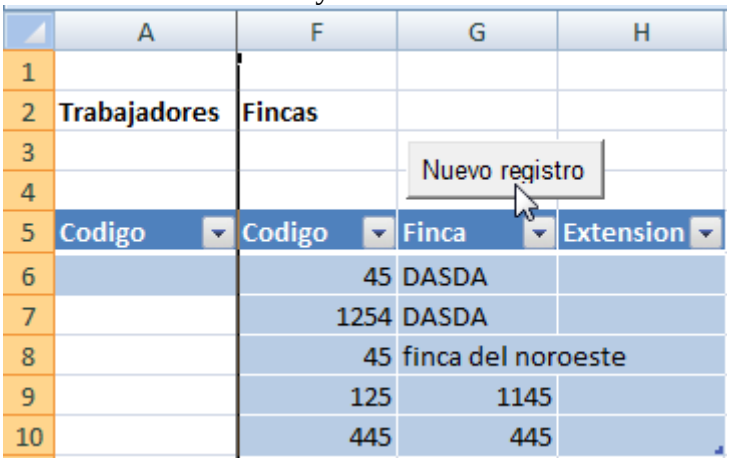


Ilustración 4

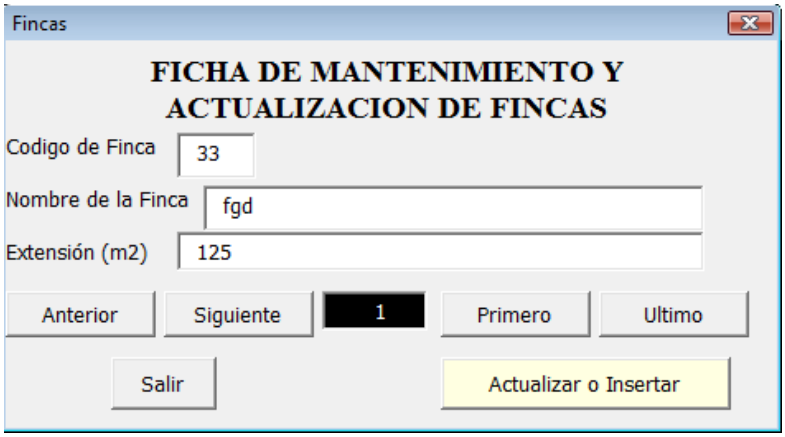


Ilustración 5

4.2.2 Creando los rangos de la tabla

Para alcanzar nuestro objetivo en primer lugar vamos a definir los rangos y la tabla a través de la opción de Excel Formulas – Administración de nombres tal y como vemos en la Ilustración 6.

Ilustración 6

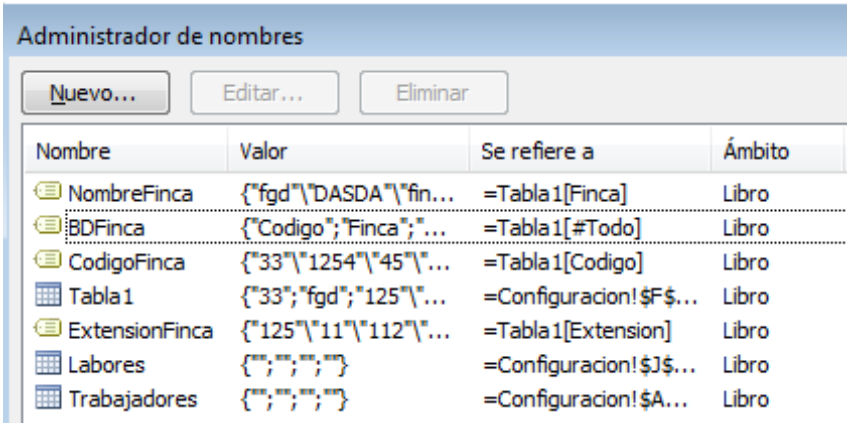


Ilustración 7

Así hemos creado los siguientes rangos basados en la tabla (Ver Ilustración 4):

- CodigoFinca que hace referencia a la columna Codigo de la Tabla BDFinca.
- NombreFinca que hace referencia a la columna Finca de la citada tabla.
- ExtensionFinca que hace referencia a la columna Extension de la tabla.
- BFinca le hemos asignado a toda la tabla que contiene los valores de la finca.

Una vez creada la definición de la tabla y los rangos correspondientes a cada columna pasamos a analizar la configuración del formulario, para ello retomamos la Ilustración 5.

4.3 Configuración del formulario.

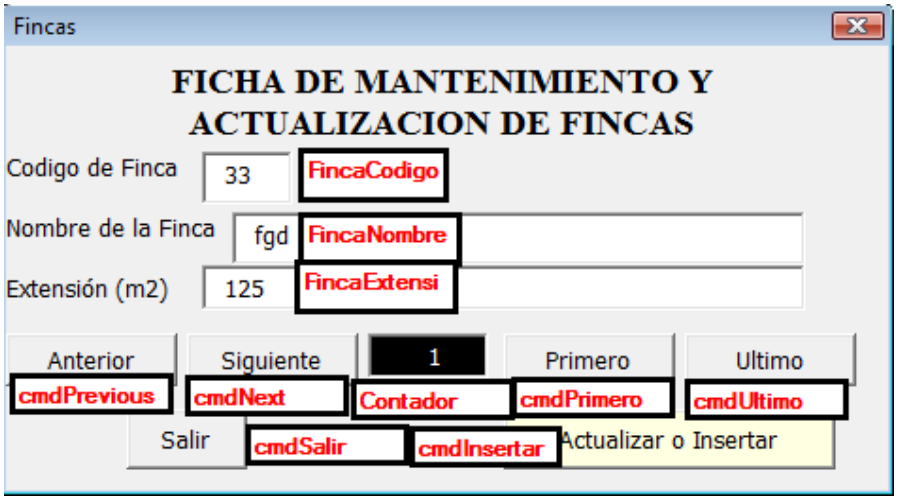


Ilustración 8

Como podemos ver el formulario se compone de cuatro Text Box o Cuadros de Texto:

- FincaCodigo: que muestra o inserta el código que asignemos a la finca.
- FincaNombre: que muestra o inserta el nombre de la Finca o Parcela
- FincaExtension: que almacena y muestra la extensión de la Finca.
- Contador: Muestra el número de registro en el que nos encontramos situados correspondiente en la base de datos.

Y también podemos observar la existencia de seis elementos Command Button o Botones de Comando:

- Un conjunto de botones que nos permite navegar a través de los diferentes registros de la tabla, Anterior-Siguiente-Primerio y Ultimo.
- Comando Salir que desactiva y cierra el formulario.
- Comando Actualizar o Insertar que modifica los datos del registro activo o situados en un nuevo registro almacena en la tabla los campos del mismo.

4.4 Código y explicación

4.4.1 Visión general del código

El código correspondiente a este formulario se estructura básicamente en los siguientes procedimientos y eventos asociados:

- Inicialización del formulario: Proceso que se realiza cada vez que se inicializa el formulario.
- Actualizardatos: evento que actualiza los datos del formulario.
- Contador: lleva el control sobre el número de registro o fila en la que nos encontramos correspondiente a la base de datos.

- Botones de navegación: evento que se produce cada vez que hacemos click sobre cada botón.
- Boton especial " Actualizar o Insertar" procedimiento que se ejecuta para almacenar o actualizar los campos en la base de datos.

Pasamos a continuación a estudiar cada uno de ellos, teniendo en cuenta que previamente hemos definido los rangos tal y como hemos comentado en el apartado

4.4.2 Inicialización del formulario.

```
Private Sub UserForm_Initialize()
' Nos aseguramos que la hoja Configuracion esta activa que es
' la que va a contener los datos del Userform
Sheets("Configuracion").Activate
' actualizamos todos los datos
Actualizadatos
End Sub
```

Ilustración 9

```
Private Sub UserForm_Initialize()
' Nos aseguramos que la hoja Configuracion esta activa que es
' la que va a contener los datos del Userform
Sheets("Configuracion").Activate
' actualizamos todos los datos
Actualizadatos
End Sub
```

Cada vez que inicializamos el formulario nos aseguramos que estamos en la hoja activa que contiene la base de datos sobre la que vamos a operar, en nuestro caso esa hoja es "Configuración" y posteriormente llamamos al procedimiento Actualizardatos que como veremos a continuación define la ubicación de cada campo de la base de datos.

4.4.3 Procedimiento "Actualizardatos"

```
Private Sub Actualizadatos()
' Esto viene a decir que el valor del campotexto
' por ejemplo FincaCodigo toma el valor definido por el rango Rango
'CodigoFinca y la fila marcada por Contador.txt
FincaCodigo.Text = Range("CodigoFinca")(Contador.Text).Value
FincaNombre.Text = Range("NombreFinca")(Contador.Text).Value
FincaExtension.Text = Range("ExtensionFinca")(Contador.Text).Value
End Sub
```

Ilustración 10

```
Private Sub Actualizadatos()
' Esto viene a decir que el valor del campotexto
' por ejemplo FincaCodigo toma el valor definido por el rango Rango
'CodigoFinca y la fila marcada por Contador.txt
FincaCodigo.Text = Range("CodigoFinca")(Contador.Text).Value
FincaNombre.Text = Range("NombreFinca")(Contador.Text).Value
FincaExtension.Text = Range("ExtensionFinca")(Contador.Text).Value
End Sub
```

Aquí definimos cual es el valor correspondiente a cada campo de texto del formulario o Text Button, por ejemplo el valor de FincaCodigo, se encuentra en el Rango CodigoFinca y en la fila marcada por el botón Contador, ese es el valor que ha de mostrar en el Text Button. Asi para el resto de campo de texto del formulario el procedimiento es el mismo.

Por tanto cada vez que ejecutamos el procedimiento Actualizar lo que realmente hace es determinar en qué fila se encuentra de la tabla o base de datos Excel, leer los campos de cada rango o columna y actualizar los valores en el formulario.

4.4.4 Procedimiento Contador_Afterapdate

Como hemos dicho anteriormente, este evento tiene como objetivo determinar la fila de la tabla o base de datos en la que nos encontramos en cada momento.

```
Private Sub Contador_AfterUpdate()
'go to the right row
If Contador.Text < 1 Then
'can't go above the top of the range
    Contador.Text = 1
ElseIf Contador.Text > Range("BDFinca").Rows.Count Then
'can't go more than 1 row below the end of the range
    Contador.Text = Range("BDFinca").Rows.Count
End If
Actualizadatos
'devuelve el control al primer campo del formulario
FincaCodigo.SetFocus
End Sub
```

Ilustración 11


```
Private Sub Contador_AfterUpdate()
'go to the right row
If Contador.Text < 1 Then
'can't go above the top of the range
    Contador.Text = 1
ElseIf Contador.Text > Range("BDFinca").Rows.Count Then
'can't go more than 1 row below the end of the range
    Contador.Text = Range("BDFinca").Rows.Count
End If
Actualizadatos
'devuelve el control al primer campo del formulario
FincaCodigo.SetFocus
End Sub
```


Básicamente es que si el valor del campo de texto contador es menor que 1, es decir que esta fuera de rango de la tabla le asigne al mismo el valor por defecto 1, es decir fila 1 y por tanto los registros del formulario se

actualizarían a la fila 1, ejecutando el procedimiento Actualizardatos y colocándose el cursor en el campo FincaCodigo.

Si el valor del campo contador es mayor que el numero de filas contenidas en la base de datos finca, por ejemplo que el usuario se haya puesto sobre el campo contador y haya solicitado situarse en la fila 200, cuando realmente la tabla solo contiene 50, obligamos a que en este caso el valor de contador asuma como máximo 50 y por tanto se situé el formulario en el último registro de la base de datos o tabla.

Si el usuario quiere ir al registro 2 de la base de datos, podría poner en el campo contador 2 y así todos los campos del formulario se actualizarían a la fila 2 de su correspondiente rango, esto se realiza a través de la llamada al procedimiento Actualizadatos.

4.4.5 Botones de navegación

Estos son eventos que se produce cada vez que hacemos click sobre cada botón. Veamos cada uno de ellos.

```
Private Sub cmdPrevious_Click()
'previous row
If Contador.Text > 1 Then Contador.Value = Contador.Value - 1
Actualizadatos
End Sub
Private Sub cmdNext_Click()
'next row
If Contador.Text < Range("BDFinca").Rows.Count Then Contador.Value = Contador.Value + 1
Actualizadatos
End Sub
Private Sub cmdPrimero_Click()
Contador.Value = 1
Actualizadatos
End Sub
Private Sub cmdUltimo_Click()
Contador.Text = Range("BDFinca").Rows.Count - 1
Actualizadatos
End Sub
Private Sub cmdSalir_Click()
' con esto cancelamos la operación del formulario fincas
Unload Me
End Sub
```

Ilustración 12

Tal y como se muestra en la Ilustración 12 lo que decimos es que si hacemos clic sobre el botón “Anterior” le asignamos a Contador el valor -1 al que tiene ahora ejecute el procedimiento Actualizardatos con lo que muestra los campos del formulario correspondientes a la fila anterior a la que nos encontrábamos.

De forma similar funciona el botón siguiente que lo que hace es desplazarnos a la fila siguiente a la que nos encontramos, en este caso añadiendo +1 al valor de contador.

El botón primero simplemente lo que hace es llevarnos al primer registro de la base de datos.

Evidentemente el botón último nos llevaría al registro final de la base de datos.

Si nos fijamos cada uno de estos eventos anteriores termina ejecutando el procedimiento Actualizadatos, para que los valores queden reflejados en el formulario

El botón salir cierra el formulario activo.

```
Private Sub cmdPrevious_Click()
'previous row
If Contador.Text > 1 Then Contador.Value = Contador.Value - 1
Actualizadatos
End Sub
Private Sub cmdNext_Click()
'next row
If Contador.Text < Range("BDFinca").Rows.Count Then Contador.Value =
Contador.Value + 1
Actualizadatos
End Sub
Private Sub cmdPrimero_Click()
Contador.Value = 1
Actualizadatos
End Sub
Private Sub cmdUltimo_Click()
Contador.Text = Range("BDFinca").Rows.Count - 1
Actualizadatos
End Sub
Private Sub cmdSalir_Click()
' con esto cancelamos la operación del formulario fincas
Unload Me
End Sub
```

4.4.6 Procedimiento Insertar

```
Private Sub cmdInsertar_Click()
' Antes de copiar los valores verificamos que los campos estan llenos
If FincaCodigo.Value = "" Then
       MsgBox "Debe introducir el codigo de finca", vbCritical, "Error en los parámetros"
      Exit Sub
   ElseIf FincaNombre.Value = "" Then
      MsgBox "Debe introducir descripción de la finca", vbCritical, "Error en los parámetros"
      Exit Sub
  ElseIf FincaExtension.Value = "" Then
      MsgBox "Debe introducir la Extension de la finca", vbCritical, "Error en los parámetros"
      Exit Sub
  ' Tambien verificamos que los campos tienen el formato correcto asociado al tipo de dato
  ElseIf Not IsNumeric(FincaCodigo.Text) Then
     MsgBox "Debe introducir un valor numerico al codigo de la finca", vbCritical, "Error en los parámetros"
      Exit Sub
End If
'Comienza a copiar los valores del UserForm a la hoja
Range("CodigoFinca")(Contador.Text).Value = FincaCodigo.Value
Range("NombreFinca")(Contador.Text).Value = FincaNombre.Value
Range("ExtensionFinca")(Contador.Text).Value = FincaExtension.Value
End Sub
```

Este procedimiento lo que hace es actualizar los valores de un registro si lo queremos modificar o bien insertar los valores nuevos en la tabla. Para ello debemos estar en un registro vacio del formulario.

Como podemos ver en el código en primer lugar verificamos antes de copiar que los campos del formulario que son requeridos no están vacios

En segundo verificamos también por ejemplo que el campo FincaCodigo sea de tipo numérico

Si todo es correcto simplemente insertamos los valores en su rango correspondiente.

```
Private Sub cmdInsertar_Click()
' Antes de copiar los valores verificamos que los campos estan llenos
If FincaCodigo.Value = "" Then
       MsgBox "Debe introducir el codigo de finca", vbCritical, "Error
en los parámetros"
      Exit Sub
   ElseIf FincaNombre.Value = "" Then
      MsgBox "Debe introducir descripción de la finca", vbCritical,
"Error en los parámetros"
      Exit Sub
  ElseIf FincaExtension.Value = "" Then
      MsgBox "Debe introducir la Extension de la finca", vbCritical,
"Error en los parámetros"
      Exit Sub
  ' Tambien verificamos que los campos tienen el formato correcto
asociado al tipo de dato
  ElseIf Not IsNumeric(FincaCodigo.Text) Then
     MsgBox "Debe introducir un valor numerico al codigo de la finca",
vbCritical, "Error en los parámetros"
      Exit Sub
End If
'Comienza a copiar los valores del UserForm a la hoja
Range("CodigoFinca")(Contador.Text).Value = FincaCodigo.Value
Range("NombreFinca")(Contador.Text).Value = FincaNombre.Value
Range("ExtensionFinca")(Contador.Text).Value = FincaExtension.Value
End Sub
```

4.5 Bibliografia.

Extraído y adaptado de Helen Toomik

5 Trabajando con Formularios y Componentes Web

5.1 Palabras claves

Controles ActiveX, Microsoft Office Web Components, ChartSpace, Spreadsheet, Spin, Conectar y bloquear texbox, enlazar textbox, etc..

5.2 Planteamiento del problema

	A	B	C	D	E	F	G	H
1	**Simulador Presupuestario**						Formulario	
2								
3	**Presupuesto Nº**							
4	Ingresos Previstos (Valor del					133.844,00	**Significación**	
5	Presupuesto)						Calculada	Estandar
6	Costes Variables Estimados					73.444,00	54,9%	21,3%
7	(Materia prima, tinta, etc.)							
8	Costes No Operativos Variables		**2,95%**			**3.942,41 €**		
9	Tot. Costes Fijos o Estructurales		254,0 hr	49,6 €/hr		**12.601,78 €**	9,4%	77,60%
10	***Horas Estimadas de Trabajo***				***Nivel de Actividad***			
11			**[100% / 76%]**		**[75% / 51%]**		**[- 50%]**	
12	**Dptos. Operativos**	**Nº de Hras**	**Unitario**	**Total**	**Unitario**	**Total**	**Unitario**	**Total**
13	Taller I	100,00	48 €/hr	4.775 €	59 €/hr	5.926 €	82 €/hr	8.230 €
14	Taller II	100,00	59 €/hr	5.937 €	72 €/hr	7.180 €	97 €/hr	9.666 €
15	Taller III	54,00	35 €/hr	1.890 €	45 €/hr	2.430 €	55 €/hr	2.970 €
16								
17								
18	**Subtotal**	254,0 hr	12.601,78 €	9,4%	15.536,49 €	11,6%	20.865,93 €	15,6%
19	= Resultado previsto		43.855,82 €	33%	40.921,10 €	31%	35.591,67 €	27%

Ilustración 13

Para sintetizar nuestro objetivo en este apartado presentamos a continuación, tal y como se muestra en la Ilustración 13, un caso en el que tenemos un conjunto de datos correspondientes a un simulador presupuestario. Las celdas en rojo son las variables independientes o aquellas que podemos cambiar mientras que el resto son dependientes o asociadas a una formula.

En este caso de análisis de sensibilidad consideramos oportuno contar con un formulario que nos permita el acceso a los distintos datos así como mostrar a través de un grafico incrustado los cambios en las principales variables relevantes del análisis.

Para ello hemos confeccionado un formulario al que accedemos a través del botón correspondiente de la ilustración anterior.

5.3 El formulario y análisis de los principales componentes

5.3.1 Primera aproximación al formulario "Facturas"

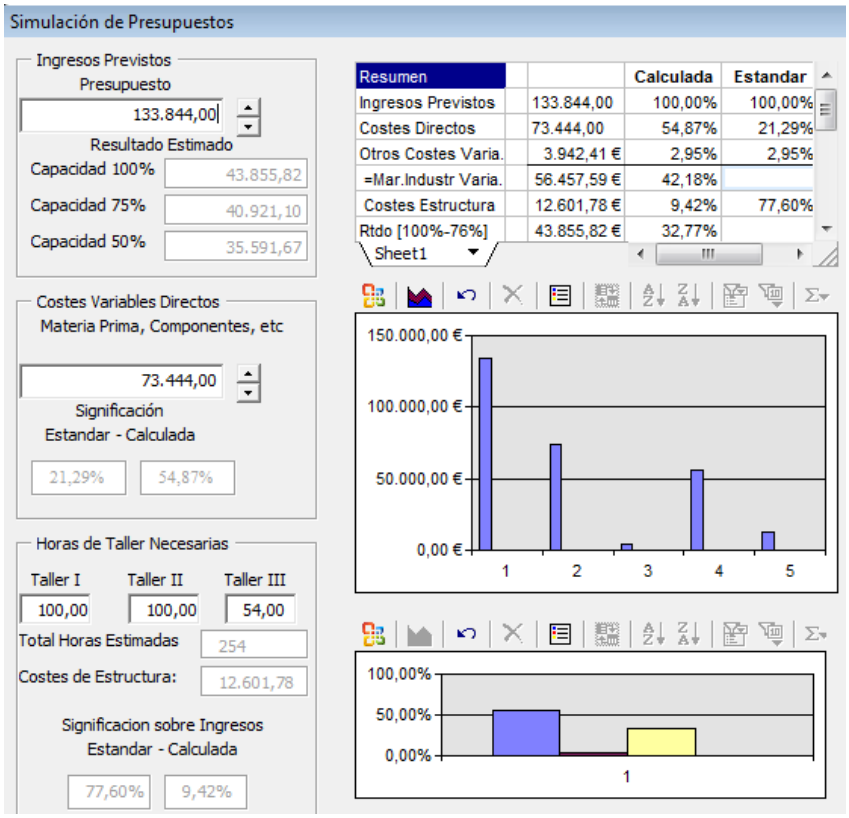


Ilustración 14

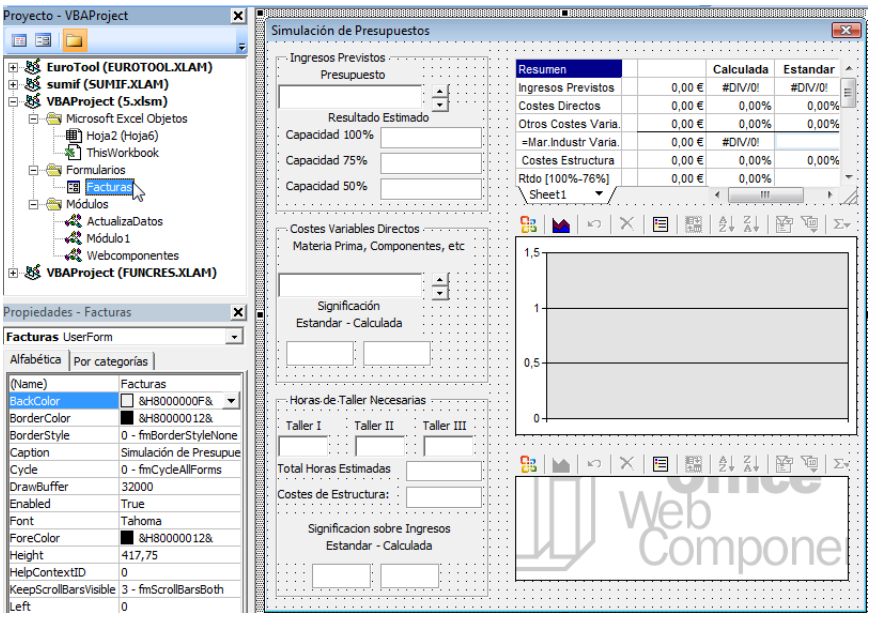


Ilustración 15

Este formulario central que hemos denominado "Facturas", contiene diferentes controles relacionados o vinculados con nuestro libro Excel y con diferentes hojas de cálculo del mismo, tal y como pasaremos a continuación a analizar.

Sin embargo destacan tipos de controles especiales, los primeros situados a la izquierda del formulario que son controles de Excel básicos o normales como control label o etiqueta, control cuadro de texto o text box, botón spin, etc. A la derecha nos encontramos dos controles diferenciados que forman parte del

conjunto de controles denominados Microsoft Office Web Components en concreto tenemos un Spreadsheet (u hoja de cálculo) y dos ChartSpace (o controles graficos), señalar que estos gráficos son dependientes de los datos disponibles en el control Spreadsheet.

5.3.2 Análisis de los textbox del formulario. Características principales. Formato del cuadro de texto.

Propiedades - Ingresos

Ingresos TextBox

Alfabética | Por categorías

(Name)	Ingresos
AutoSize	False
AutoTab	False
AutoWordSelect	True
BackColor	&H80000005&
BackStyle	1 - fmBackStyleOpaque
BorderColor	&H80000006&
BorderStyle	0 - fmBorderStyleNone
ControlSource	
ControlTipText	
DragBehavior	0 - fmDragBehaviorDisabled
Enabled	True
EnterFieldBehavior	0 - fmEnterFieldBehaviorSelectAll
EnterKeyBehavior	False
Font	Tahoma
ForeColor	&H80000008&
Height	18
HelpContextID	0
HideSelection	True
IMEMode	0 - fmIMEModeNoControl
IntegralHeight	True
Left	0
Locked	False
MaxLength	0
MouseIcon	(Ninguno)
MousePointer	0 - fmMousePointerDefault
MultiLine	False
PasswordChar	
ScrollBars	0 - fmScrollBarsNone
SelectionMargin	True
SpecialEffect	2 - fmSpecialEffectSunken
TabIndex	1
TabKeyBehavior	False
TabStop	True
Tag	
Text	
TextAlign	3 - fmTextAlignRight
Top	18
Value	
Visible	True
Width	102
WordWrap	True

Ilustración 16

Retomando la Ilustración 14 en concreto el textbox que hemos denominado Presupuesto (aunque internamente su identificador es Ingresos) vamos a estudiar sus principales propiedades para entender uno de los aspectos básicos en los que hemos programado uno de los principales elementos que componen este formulario.

En la Ilustración 16, podemos observar las propiedades asociados al citado TextBox.

Ademas de la propiedad de este elemento queremos también en este apartado destacar otras cuestiones asociadas al mismo como son los eventos cambio y después de actualizar.

Es decir, cuando introducimos un valor en este cuadro de texto queremos que este cambio se vea reflejado o este vinculado a una celda concreta de la hoja de cálculo, para ello es necesario definir un procedimiento que realice la citada acción.

En este sentido hemos creado el procedimiento Ingresos_change

```
Private Sub Ingresos_Change()
Range("e4").Value = Ingresos.Value
' Actualizar Spreadsheet3, vuelve a importar los datos
Webcomponentes.Importa_a_Spreadsheet
End Sub
```

Ilustración 17

Es decir se corresponde al evento cambio del textbox Ingreso y lo que hace es que cuando el valor del textbox cambie asigna este valor a la celda e4 de la hoja de cálculo activa.

También cuando esto ocurre los valores de la hoja de cálculo cambia y por tanto estos deben ser actualizados, en el Speedsheet, como justificaremos posteriormente.

También después de actualizar este elemento deseamos aplicar al mismo un formato así como otras acciones, para ello es necesario programar la citada acción asociado al evento AfterUpdate, tal y como mostramos en la Ilustración 18.

La función concreta es:

```
Ingresos.Text = Format(Ingresos.Text, "#,##0.00")
```

Conjuntamente realizamos otra serie de acciones o llamadas a módulos del libro que previamente hemos definido y explicaremos posteriormente como son actualizar los porcentajes, así como otros valores. Evidentemente al cambiar el valor de la variable Ingresos (TextBox) el resto de variables dependientes también cambiaran y por tanto será necesario actualizar los citados cambios en el formulario para que el mismo sea coincidente con los valores respectivos de las celdas de la hoja de cálculo.

```
Private Sub Ingresos_AfterUpdate()
Ingresos.Text = Format(Ingresos.Text, "#,##0.00")
'actualiza porcentajes llamamos al modulo ActualizaDatos
ActualizaDatos.Porcentajes
'actualiza los resultados al 100%, 75% y 50% llamamos al
ActualizaDatos.Resultados
'actualiza los costes de estructura
ActualizaDatos.Estructura
' Actualiza hoja de calculo Spreadsheet vuelve a importa
Webcomponentes.Importa_a_Spreadsheet
' Recargamos con datos el grafico1
Webcomponentes.grafico1actualiza
End Sub
```

Ilustración 18

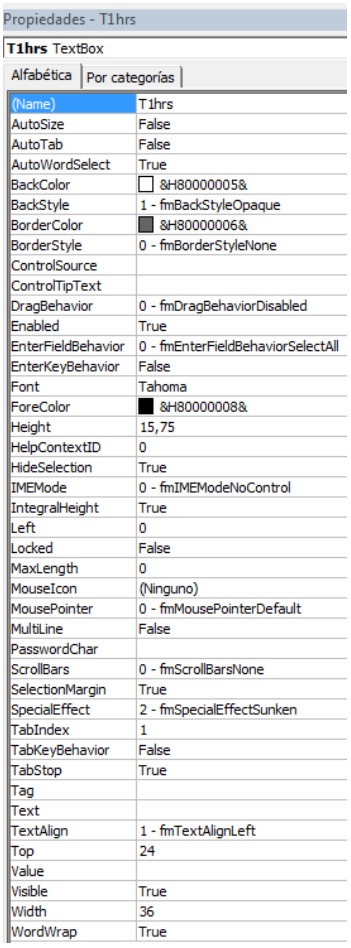


Ilustración 19

De igual forma nos encontramos con las propiedades y el comportamiento del TextBox que hemos denominado "Taller I". En este segundo ejemplo de programación del TextBox es muy similar al anterior.

En este caso en este control de cuadro de texto (y que internamente hemos llamado T1hrs) se estiman las horas necesarias de Taller I para llevar a cabo el trabajo o servicio y este elemento está asociado a la celda "B13" tal y como se puede ver en la Ilustración 13.

Igualmente asociado al mismo hemos programado dos acciones asociadas al evento Change así como al evento AfterUpdate

```
Private Sub T1hrs_Change()
Range("b13").Value = T1hrs.Value
' Actualizar Spreadsheet3, vuelve a importar los datos
Webcomponentes.Importa_a_Spreadsheet
End Sub
```

Ilustración 20

El objetivo de este evento Change es el mismo que en el caso analizado anteriormente y su significado por tanto es coincidente, comparar la Ilustración 17 y la Ilustración 20.

Igualmente tenemos asociado el evento AfterUpdate de la Ilustración 21 y que podemos comparar de igual manera con el descrito anteriormente correspondiente a la Ilustración 18

```
Private Sub T1hrs_AfterUpdate()
T1hrs.Text = Format(T1hrs.Text, "#,##0.00")
'actualiza porcentajes llamamos al modulo ActualizaDatos porcent
ActualizaDatos.Porcentajes
'actualiza los resultados al 100%, 75% y 50% llamamos al modulo
ActualizaDatos.Resultados
'actualiza los costes de estructura
ActualizaDatos.Estructura
End Sub
```

Ilustración 21

5.3.3 Bloqueando los textbox.

En algunos casos, en especial cuando las variables son dependientes nos puede interesar que el citado registro se muestre en el formulario pero que aparezca como bloqueado y el mismo se actualice en el momento de que cambien los valores del mismo. Este es el caso por ejemplo de los cuadros de texto correspondientes al resultado estimado para la capacidad del 100% o del total de horas estimadas necesarias de taller, tal y como podemos ver en la Ilustración 14 que aparecen con un color gris, síntoma de que no pueden ser editados los valores en cuanto que son celdas resultantes de una formula o calculo, es decir son variables dependientes.

Para analizar sus propiedades así como eventos principales asociados tomaremos como referencia la variable Resultado Estimado para la capacidad del 100% y que internamente hemos denominado al citado cuadro de texto como "Rtdo1".

Así tal y como se muestra en la Ilustración 23 y comparamos con la Ilustración 19 observamos que los únicos cambios se producen en la propiedad SpecialEffect que en este caso para que aparezca bloqueado le asignamos el valor a la citada propiedad "3 fm SpecialEffectEtched". Evidentemente también tenemos asignados un conjunto de enventos a este control concreto Change tal y como muestra la Ilustración 22 que simplemente lo que hace es cada vez que cambia su valor le vuelve a dar formato.

```
Private Sub Rtdo1_Change()
Rtdo1.Text = Format(Rtdo1.Text, "#,##0.00")
End Sub
```

Ilustración 22

Pero realmente para que el control quede bloqueado, hemos procedido a incorporar en el evento de inicialización del formulario la propiedad correspondiente, es decir el código:

```
Rtdo1.Enabled = False
```

Sin entrar a explicar detalladamente en este momento el código correspondiente al proceso de inicialización del formulario (ver Ilustración 24) exponemos a continuación el mismo para destacar simplemente como al inicializar el formulario de la Ilustración 14 procedemos a bloquear los Text Box o cuadros de texto deseados.

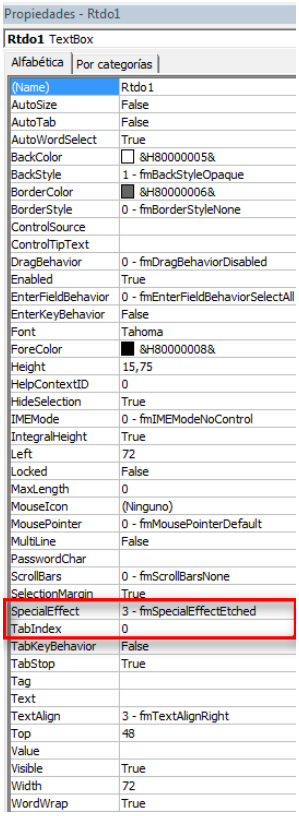


Ilustración 23

```
Private Sub UserForm_Initialize()
'Al inicializar el formulario Facturas hacemos lo siguiente
' Asignamos los valores por defecto a los campos del form y le damos forma
Ingresos.Text = Cells(4, 5)
Ingresos.Text = Format(Ingresos.Text, "#,##0.00")
GtosDirecVari.Text = Cells(6, 5)
GtosDirecVari.Text = Format(GtosDirecVari.Text, "#,##0.00")
GtosDirecVariPorEstandar.Text = Cells(6, 8)
GtosDirecVariPorEstandar.Text = Format(GtosDirecVariPorEstandar.Text, "#.0
GtosDirecVariPorCal.Text = Cells(6, 7)
GtosDirecVariPorCal.Text = Format(GtosDirecVariPorCal.Text, "#.0,0%")
T1hrs.Text = Cells(13, 2)
T1hrs.Text = Format(T1hrs.Text, "#,##0.00")
T2hrs.Text = Cells(14, 2)
T2hrs.Text = Format(T2hrs.Text, "#,##0.00")
T3hrs.Text = Cells(15, 2)
T3hrs.Text = Format(T3hrs.Text, "#,##0.00")
Rtdo1.Text = Cells(19, 3)
Rtdo1.Text = Format(Rtdo1.Text, "#,##0.00")
Rtdo2.Text = Cells(19, 5)
Rtdo2.Text = Format(Rtdo2.Text, "#,##0.00")
Rtdo3.Text = Cells(19, 7)
Rtdo3.Text = Format(Rtdo3.Text, "#,##0.00")
TotalHras.Text = Cells(9, 3)
TotalHras.Text = Format(TotalHras.Text, "#,##0.00")
TotalEstructura.Text = Cells(9, 5)
TotalEstructura.Text = Format(TotalEstructura.Text, "#,##0.00")
EstandarEstuc.Text = Cells(9, 8)
EstandarEstuc.Text = Format(EstandarEstuc.Text, "#.0,0%")
CalculadaEstuc.Text = Cells(9, 7)
CalculadaEstuc.Text = Format(CalculadaEstuc.Text, "#.0,0%")
' Bloquemaos los controles que deseamos aparecen en gris oscuro
' para que no se puedan meter datos son paneles exclusivamente informativo
GtosDirecVariPorEstandar.Enabled = False
GtosDirecVariPorCal.Enabled = False
Rtdo1.Enabled = False
Rtdo2.Enabled = False
Rtdo3.Enabled = False
TotalHras.Enabled = False
TotalEstructura.Enabled = False
EstandarEstuc.Enabled = False
CalculadaEstuc.Enabled = False
' Cargamos con datos el grafico1 y grafico 2
Webcomponentes.grafico1actualiza
Webcomponentes.grafico2actualiza
End Sub
```

Ilustración 24

El código de la Ilustración 24 está asociado al formulario facturas en concreto al evento inicializar, tal y como podemos leer en las líneas de comentarios (en color verde) en su penúltimo apartado procedemos al bloqueo de los controles de texto deseados asignándoles el valor False a la propiedad Enabled.

Junto a este conjunto de Text Box básicos que hemos analizado anteriormente nos encontramos con otros elementos que pasamos a continuación a analizar.

5.3.4 Control SpinButton (botón de numero) asociado a Text Box.

El objeto de este elemento SpinButton es facilitar los cambios o variaciones de los valores numéricos asociados a una celda o TextBox. Lo que realmente hace es incrementar o disminuir el valor numérico asociado al control de cuadro de texto.

Por tanto vamos analizar las propiedades y eventos asociados a este tipo de objeto para lo cual tomaremos como referencia los dos SpinButton usados en el formulario Facturas de la Ilustración 14, en concreto son el llamado SpinIngresos y SpinGtos.

Entre las propiedades de este tipo de objeto destacan básicamente tres, tal y como se muestra en la Ilustración 25 que son el valor máximo que deseamos asignar a este control, su valor mínimo así como el intervalo de cambio o variación (SmallChange) que deseamos asignar cada vez que pulsamos sobre el citado control, es decir es el valor de crecimiento o decrecimiento de valores.

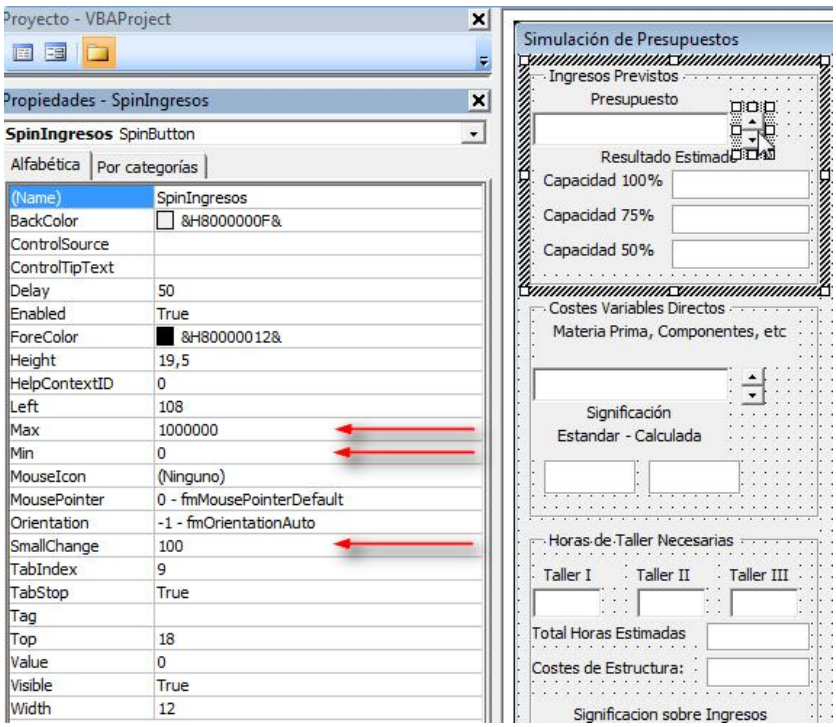


Ilustración 25

En referencia a los eventos asociados a este tipo de control son dos, el evento Enter y el Change tal y como se muestra en la Ilustración 26.

```
Private Sub SpinIngresos_Enter()
SpinIngresos.Value = Ingresos.Value
End Sub
Private Sub SpinIngresos_Change()
Ingresos.Value = SpinIngresos.Value
Range("e4").Value = Ingresos.Value
Ingresos.Text = Format(Ingresos.Text, "#,##0.00")
'actualiza porcentajes llamamos al modulo ActualizaDatos
ActualizaDatos.Porcentajes
'actualiza los resultados al 100%, 75% y 50% llamamos al
ActualizaDatos.Resultados
'actualiza los costes de estructura
ActualizaDatos.Estructura
End Sub
```

Ilustración 26

Así con el evento *SpinIngresos_Enter* lo que hace es que cada vez que se entre en este control el valor que toma por defecto es el que tiene el Text Box o Caja de Texto "Ingresos" del formulario activo.

Con el evento *SpinIngresos_Change*, es decir cada vez que cambia o pulsamos sobre el citado SpinButton se incrementa el valor de referencia y por tanto en primer lugar actualiza el mismo en la casilla de texto "Ingresos" del formulario y vuelve a dar formato al citado control de texto y además actualiza el valor de la celda que contiene esa referencia en la hoja de cálculo.

Como los cambios que se producen en este control afectan a variables dependientes, en concreto a la de ingresos, es necesario por tanto actualizar los datos de las mismas para que queden correctamente reflejados en el formulario. Para ello recurriremos a un conjunto de módulos a los que invocamos, y que trataremos posteriormente.

Igualmente las propiedades así como los eventos configurados para el control SpinGtos son similares al tratado anteriormente tal y como se puede ver en la Ilustración 27 y su comparación con la Ilustración 26.

```
Private Sub SpinGtos_Enter()
SpinGtos.Value = GtosDirecVari.Value
End Sub
Private Sub SpinGtos_Change()
GtosDirecVari.Value = SpinGtos.Value
Range("e6").Value = GtosDirecVari.Value
GtosDirecVari.Text = Format(GtosDirecVari.Text, "#,##0.00")
'actualiza porcentajes llamamos al modulo ActualizaDatos porcentajes
ActualizaDatos.Porcentajes
'actualiza los resultados al 100%, 75% y 50% llamamos al modulo Actua
ActualizaDatos.Resultados
'actualiza los costes de estructura
ActualizaDatos.Estructura
End Sub
```

Ilustración 27

5.4 Principales eventos del formulario facturas y módulos del proyecto.

5.4.1 Evento al inicializar el formulario

Una vez diseñado y construido el formulario Facturas con todos sus elementos tal y como muestra la Ilustración 15 es necesario que cuando este se abra cargue todos los valores de los TextBox o Cuadros de Texto de las celdas de las hojas de cálculo a la que está vinculado y asigne el formato correspondiente. Para llevar a cabo esta acción, hacemos uso del evento asociado al citado formulario denominado UserForm_Initialize y en la que podemos distinguir las siguientes acciones (ver Ilustración 24):

- Cargamos los valores de las distintas cajas de texto con sus vínculos de cada una de las celdas correspondientes.
- Damos los formatos correspondientes a cada uno de estos controles.
- Bloquemos los controles deseados, especialmente los de aquellas variables dependientes para impedir su modificación.
- Cargamos los valores correspondientes a los controles ActiveX de Microsoft Office Web Components, especialmente el Spreadsheet así como los ChartSpace con el fin de dibujar los gráficos correspondientes.

En resumen, con este procedimiento cada vez que inicializamos el formulario se vuelven a cargar los datos de los distintos cuadros de texto con los valores vinculados de las celdas, se le da el formato numérico correspondiente, se bloquean aquellos valores correspondientes a las variables dependientes y finalmente se importan los datos y se dibujan los gráficos de los componentes Spreadsheet.

5.4.2 Botón inicializar/abrir el formulario. Módulo 1

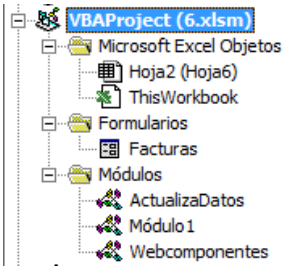


Ilustración 28

En nuestro libro Excel contamos con tres módulos (Ilustración 28), en concreto el Módulo1 es una macro o sentencia que lo que hace es abrir nuestro formulario "Facturas" y su código es tan simple como el que mostramos a continuación:

```
Sub MostrarFactura()
    Facturas.Show
End Sub
```

Por tanto lo que procede es asignar este código al botón Formulario de la Ilustración 13 a través de la asignación de la macro al citado botón "MostrarFactura" tal y como hemos denominado a este procedimiento.

5.4.3 Eventos especiales asociados a los textbox. Modulo Actualiza Datos.

En este modulo encontraremos tres procedimientos o rutinas tal y como se muestra en la Ilustración 29.

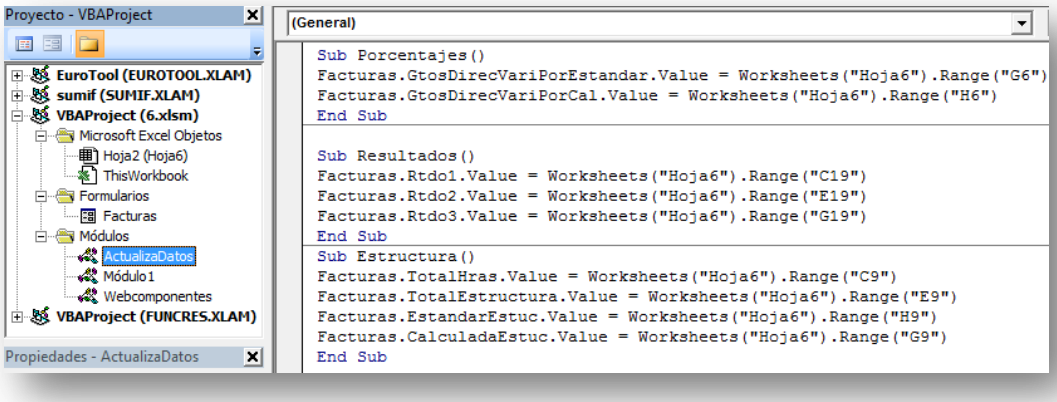


Ilustración 29

- ActualizaDatos.Porcentajes. Este procedimiento lo que hace es cargar o actualizar los valores del cuadro de porcentajes, según el siguiente código:

```
Sub Porcentajes()
        Facturas.GtosDirecVariPorEstandar.Value = Worksheets("Hoja6").Range("G6")
        Facturas.GtosDirecVariPorCal.Value = Worksheets("Hoja6").Range("H6")
End Sub
```

Es decir (Ilustración 30) asigna valores a los cuadros de porcentaje del formulario facturas contenidos en la Hoja 6 de nuestro libro y en el rango o celda correspondiente.

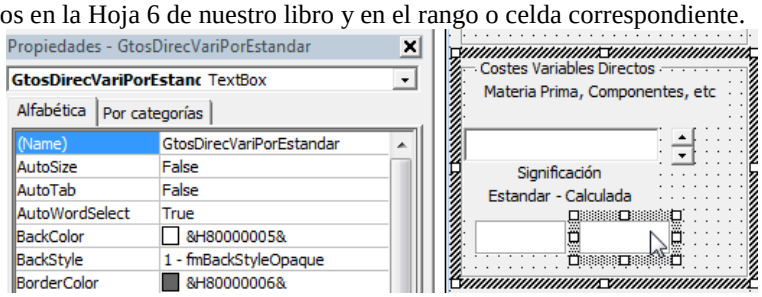


Ilustración 30

- ActualizaDatos.Resultados. En este caso el código asociados es:

```
Sub Resultados()
        Facturas.Rtdo1.Value = Worksheets("Hoja6").Range("C19")
        Facturas.Rtdo2.Value = Worksheets("Hoja6").Range("E19")
        Facturas.Rtdo3.Value = Worksheets("Hoja6").Range("G19")
End Sub
```

De forma similar al caso anterior, en este ocasión actualizamos y asignamos valores a los cuadros de texto los valores contenidos en las hojas especificadas y en el rango correspondiente.

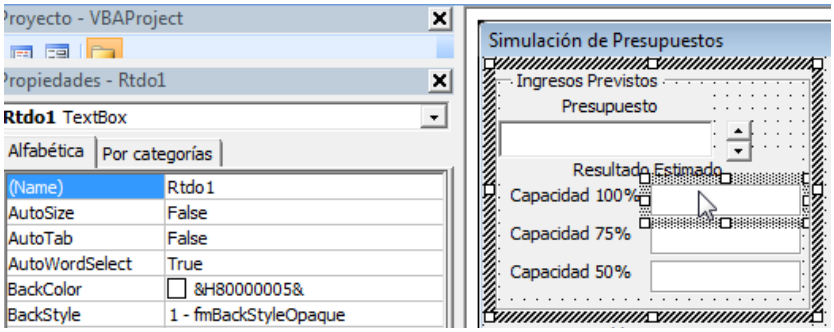


Ilustración 31

- ActualizarDatos.Estructura. El código asociado a esta parte del módulo es:

```
Sub Estructura()
        Facturas.TotalHras.Value = Worksheets("Hoja6").Range("C9")
```

```
        Facturas.TotalEstructura.Value = Worksheets("Hoja6").Range("E9")
        Facturas.EstandarEstuc.Value = Worksheets("Hoja6").Range("H9")
        Facturas.CalculadaEstuc.Value = Worksheets("Hoja6").Range("G9")
End Sub
```

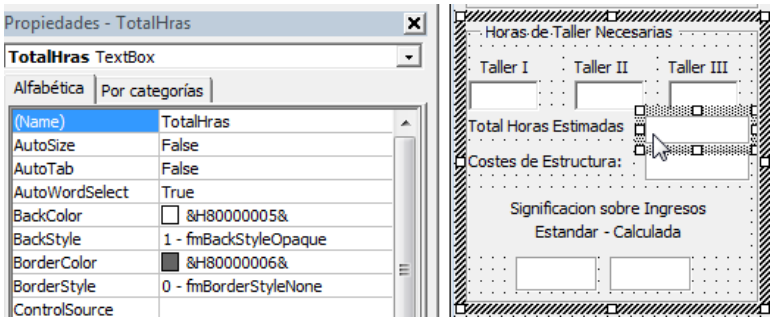


Ilustración 32

5.4.4 Llamada a los módulos desde los TextBbox independientes.

Por tanto tomando en consideración estos módulos, podemos ver que cada vez que cambien los valores de los textbox independientes nuestro objetivo se centra en actualizar los valores en el formulario, para ello hacemos una llamada a los módulos anteriores.

Por ejemplo, cada vez que cambiamos el cuadro de texto "Ingresos" del formulario "Facturas" (Ilustración 17 e Ilustración 18) llama al método ActualizaDatos y sus distintos procedimientos para reflejar estos cambios en las variables dependientes en el formulario.

De igual forma por ejemplo cuando otra variable independiente como puede ser el TextBox "T1hrs" (Ilustración 19, Ilustración 20 e Ilustración 21) cambia esto afecta a otras variables dependientes y por tanto debemos llamar a este modulo para proceder a reflejar estas actualizaciones.

En definitiva, en todas las variables independientes del formulario factura y para el evento AfterUpdate hemos añadido las siguientes sentencias que ejecutan básicamente el módulo ActualizaDatos.

```
'actualiza porcentajes llamamos al modulo ActualizaDatos porcentajes
        ActualizaDatos.Porcentajes
'actualiza los resultados al 100%, 75% y 50% llamamos al modulo ActualizaDatos resultados
        ActualizaDatos.Resultados
'actualiza los costes de estructura
        ActualizaDatos.Estructura
```

Señalar finalmente que el modulo que hemos denominado como webcomponente contiene las rutinas asociadas a los controles ActiveX de Microsoft Office Web Components, en concreto el Spreadsheet así como los ChartSpace , estos serán analizados con más detalle en el siguiente apartado.

6 Controles ActiveX, Spreadsheet y ChartSpace

6.1 Introducción.

Retomando la Ilustración 14 observamos en el margen derecho del formulario la existencia de dos tipos de componentes especiales de la familia de Controles ActiveX Microsoft Office Web Components, en concreto Spreadsheet y dos ChartSpace.

Estos controles nos van a permitir interactuar con el resto del libro y de sus hojas de cálculo así como representar los valores mediante gráficos.

Destacar previamente que para poder tener estos disponibles en nuestra cuadro de herramienta de la cinta de programación VBA es necesario cargarlo ya que estos no vienen incorporados por defecto, para ello accederemos desde Microsoft Visual Basic a la opción Herramientas-Controles adicionales tal y como mostramos en la Ilustración 33.

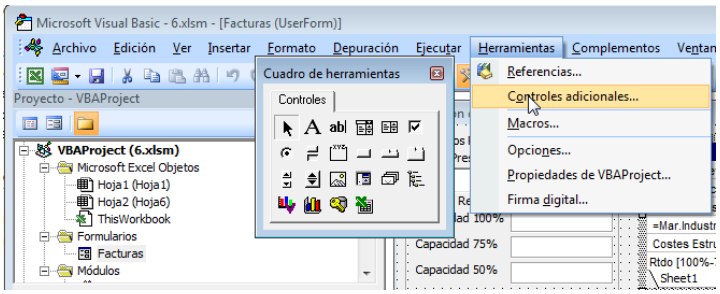


Ilustración 33

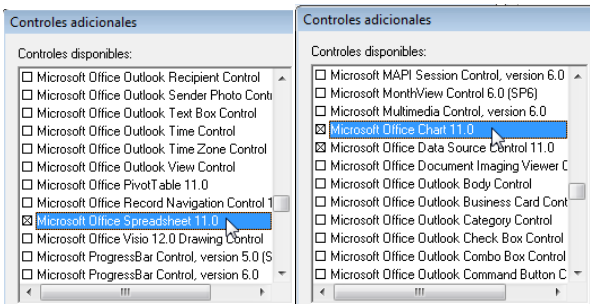


Ilustración 34

De esta forma accedemos a una pantalla adicional donde podremos seleccionar los controles necesarios que deseamos incorporar, en nuestro caso son los señalados en la Ilustración 34.

Como podemos observar de este cuadro de dialogo disponemos de multitud de controles que podemos añadir a nuestro cuadro de herramientas

En caso de que estos no esten disponibles en nuestro equipo deberemos descargarlo desde la web "Herramienta de Office XP: Web Components" o para la versión de Office 2007. Igualmente señalar tambien que si desarrollamos alguna aplicación excel con estos componentes el usuario final para poder usarlos se los debera tambien que descargar.

6.2 Análisis del componente Spreadsheet.

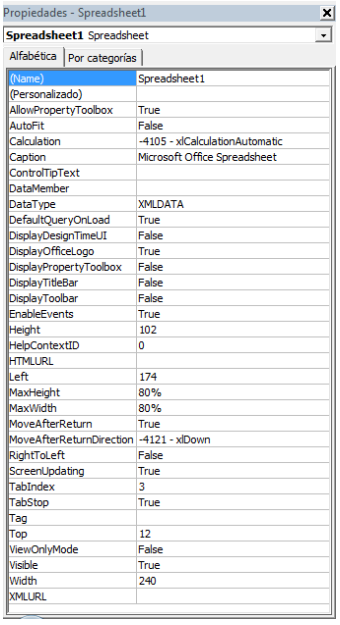


Ilustración 35

Partiendo de que el componente lo tenemos disponible, insertamos en nuestro formulario el citado Spreadsheet y podremos ver sus propiedades. Ademas si pulsamos sobre el mismo podremos acceder tambien a la configuración del mismo, para ello debemos situarnos dentro de una celda del Spreadsheet y con el botón derecho accedemos al conjunto de comandos y opciones diponibles, ver Ilustración 36.

Así podremos proteger las hojas de calculo a mostrar, el numero de ellas, asi como los desplazamientos, etc.Tambien nos permitira dar formato a las celdas a traves de la pestaña format.

Contamos ademas con la posibilidad de crear nuevas hojas (sheet), eliminarlas, modificarlas, etc.

Especial atención merece la pestaña formula a traves de la cual podemos incorporar formulas y referencias a las celdas de forma similar a como trabajamos tradicionalemente con nuestra hoja de calculo, evidentemente con ciertas peculiaridades.

Podremos ocultar los encabezados de fila y columna, disponer de un menu contextual asociado al citado componente.

Destacar que no podemos vincular directamente los datos de nuestra hoja de calculo a nuestro Spreadsheet1 sino que deberemos importalos a traves de una rutina.

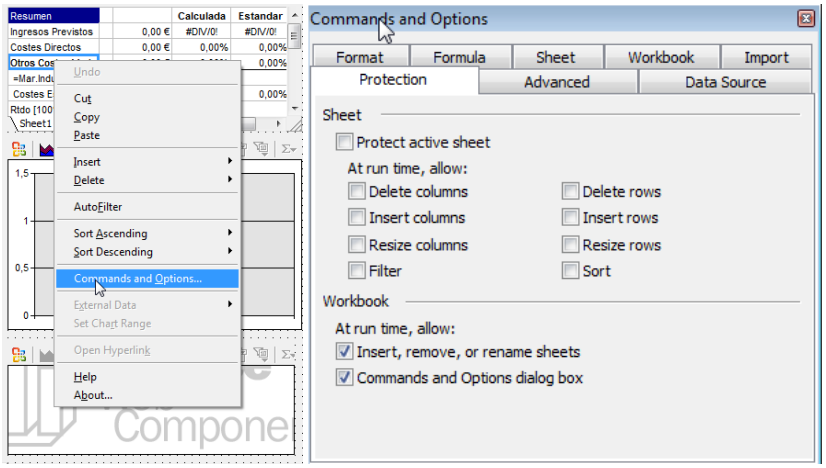


Ilustración 36

6.3 *Análisis del Módulo Webcomponentes. Caso Spreadsheet.*

Debemos tener en cuenta que con este componente no podemos leer de forma directa los datos de nuestra hoja de calculo, pero si podemos importarlos para trabajar con ellos en nuestro Spreadsheet. Así para trabajar con los datos disponibles en nuestro libro de trabajo hemos establecido un procedimiento dentro del modulo Webcomponentes que hemos denominado "Importa_a_SpreadSheet" tal y como se muestra en la Ilustración 37.

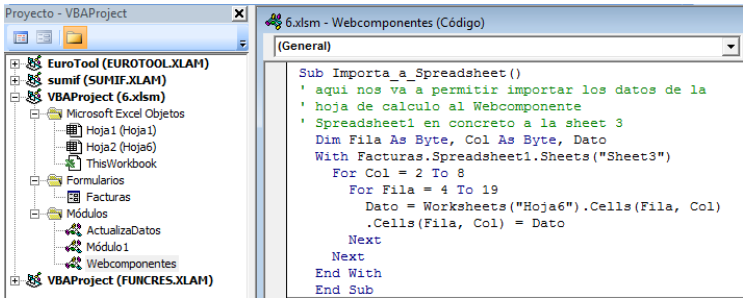


Ilustración 37

Lo que hace este procedimiento es leer los datos de nuestra "Hoja 6" (ver Ilustración 13) e incorporarlos a nuestro Spreadsheet en concreto a la hoja Sheet 3, tal y como vemos en la Ilustración 38 y posteriormente ocultamos para no estar visible para el usuario.

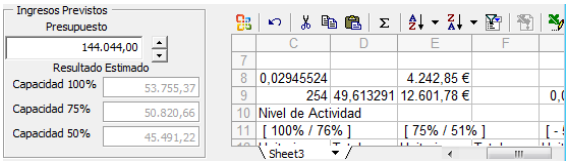


Ilustración 38

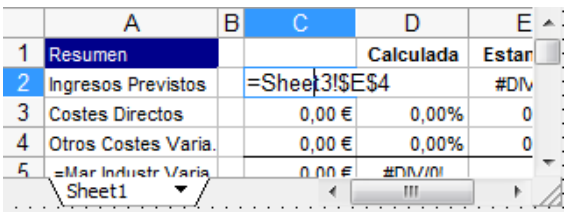


Ilustración 39

Una vez importados los datos necesarios podemos hacer referencia a los mismos dentro de cualquier Sheet del componente Spreadsheet con el nombre de la hoja y su celda de ubicación tal y como muestra la Ilustración 39.

Por ejemplo, leemos los Ingresos Previstos a traves de la formula:

```
=Sheet3!$E$4
```

Y que corresponde a los datos importados anteriormente a traves del procedimiento "Importa_a_SpreadSheet".

Debemos tener en cuenta que cada vez que modificamos un valor en la hoja de calculo debemos invocar este procedimiento para actualizar los mismos en el Spreadsheet, es decir a atraves de una llamada como:

```
Webcomponentes.Importa_a_Spreadsheet
```

Así por ejemplo en las Ilustración 17, Ilustración 18 e Ilustración 20 tenemos varios ejemplos que al cambiar el valor de algunos cuadros de texto independientes del formulario automaticamente actualizamos los valores del componente Spreadsheet en a traves de una llamada a la función anterior.

6.4 Análisis del Módulo Webcomponentes. Caso ChartSpace.

Respecto al componente ChartSpace como podemos observar de nuestro formulario de ejemplo "Facturas" presentamos dos gráficos que hemos denominado ChartSpace1 y ChartSpace2. Los datos necesarios para representar los valores deben estar contenidos en otro componente Spreadsheet, así por ejemplo podemos preparar un nuevo sheet para representar los gráficos.

Evidentemente, en este caso tambien cada vez que cambiemos los valores de las celdas independientes queremos que el gráfico se actualiza, para ello hemos creado dentro del modulo Webcomponentes dos procedimnientos uno para cada gráfico que hemos denominado "Grafico1actualiza" y "Grafico2actualiza" cuya unica diferencia son los datos ha representar, ver Ilustración 40.

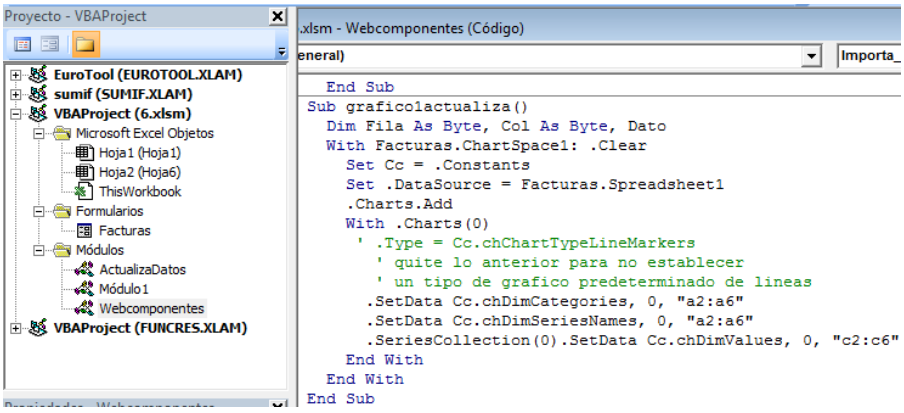


Ilustración 40

7 Bibliografía.

http://jbcascallar.wordpress.com/2009/09/09/manejo-de-controles-de-formulario-userform-ejemplo-de-un-formulario-de-alta-de-usuarios/