2

INTRODUCCIÓN A LAS MACROS

2.1 INTRODUCCIÓN

En este capítulo vamos a ver algunos ejemplos de cómo hacer macros con eventos y operaciones o acciones básicas.

Para ello, nos valdremos de uno o varios botones y el editor de Visual Basic de Excel, en donde definiremos pequeños subprogramas que se denominan funciones o procedimientos, según sea el caso, y que vienen declaradas por las palabras reservadas `Function` y `Sub`.

2.1.1 Sub y Function

La palabra reservada `Sub`, comúnmente define un bloque denominado subrutina o procedimiento y representa un bloque de código que se utiliza para realizar una tarea determinada que NO devuelve ningún valor o resultado. Por ejemplo:

```
Sub CambiaColorTexto(celda As Range)
    Celda.Interior.Color = vbRed
End Sub
```

Por otro lado, la palabra reservada `Function`, comúnmente define un bloque denominado función o procedimiento de función definida por el usuario y representa un bloque de código que ejecuta una tarea determinada que devuelve un valor o resultado.

La principal diferencia entre **Sub** y **Function** es que **Function** se puede usar dentro de las celdas de Excel como si de una fórmula se tratase, justo después del símbolo igual (=). Por ejemplo:

```
Function Suma(a As Long, b As Long) As Long
    Dim total As Long

    total = a + b

    Suma = total
End Function
```

Al definir o declarar esta función dentro de un módulo del Editor de VBA, por ejemplo, Módulo 1, estaremos dando la capacidad a nuestro libro de Excel de poder llamarla en nuestra hoja de cálculo de la siguiente forma:

```
=Suma(A1, A2)
```

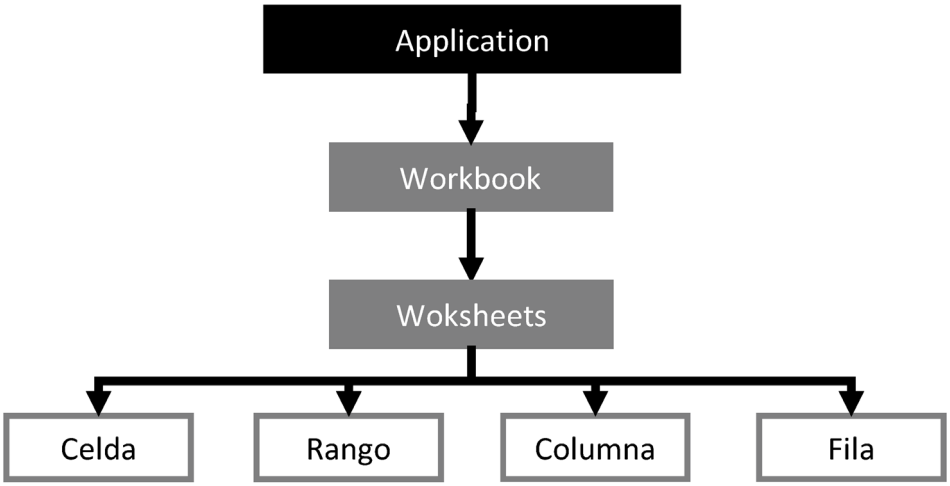
Esto es:

	A	B	C	D	E	F
1						
2	Número 1	123				
3	Número 2	456				
4	Suma	=Suma (B2, B3)				
5						

Al hacer esto, lo que debería aparecer en la celda B4 no es el contenido textual de "=Suma (B2, B3)", sino el valor 579.

2.1.2 Eventos

Como ya se ha comentado anteriormente, en Excel existen 3 grandes objetos que contienen la inmensa mayoría de los eventos comunes para ser utilizados en nuestras macros. En concreto, estamos hablando de los objetos **Application**, **Workbook** y **Worksheet**, que se explicarán más adelante.



Es importante destacar que, estos objetos, se presentan a modo de jerarquía tal y como se han mencionado, es decir, el primero en la jerarquía es **`Application`**, después está el objeto **`Workbook`** y, finalmente el objeto **`Worksheet`**.

Cada uno de estos objetos contiene los eventos del propio objeto y otros que suelen hacer referencia a objetos que están por debajo del mismo, sin embargo, no suelen llamarse igual. Por ejemplo, el objeto **`Workbook`** tiene un evento denominado **`SheetActivate`** que se activa cada vez que se selecciona una hoja de cálculo de un libro de Excel y que se corresponde con el evento **`Activate`** del objeto **`Worksheet`**.

No obstante, aunque son similares, no son idénticos. Esto es, aunque hacen o ejecutan la misma acción, su uso o declaración viene definida por una segunda premisa. Mientras que si queremos ejecutar el evento de activación en una hoja de cálculo concreta deberemos usar el evento **`Activate`** del objeto **`Worksheet`**, si lo que queremos es que se ejecute para todas, lo mejor será usar o recurrir al evento **`Activate`** del objeto **`Workbook`**.

Una posible forma de averiguar los posibles eventos de un objeto dado es recurrir al **`Examinador de objetos`**, accesible desde la opción **`Ver`** del menú principal del **`Editor de Visual Basic`**. Al abrir esta ventana o diálogo, podremos ver paneles, uno a la izquierda, que es donde están listados los objetos o clases disponibles y, otro a la derecha, que es el que nos permitirá ver sus eventos, propiedades y métodos asociados y disponibles para ese objeto o clase.

Por último, sólo destacaremos que los eventos se declaran y localizan dentro del objeto al que pertenezcan. Así, si lo que estamos definiendo es el evento `Activate` de la Hoja1 de un libro de Excel, el evento y/o su macro estará dentro de la opción del objeto en cuestión, como ya veremos más adelante.

2.2 GRABACIÓN DE MACROS

2.2.1 Macro 1: Hola mundo

Se trata de a crear una macro a través de la grabación de acciones automática que trae el Microsoft Excel que inserte el texto "Hola mundo" en la celda A1 y que le aplique un formato de color de fondo negro con color de texto en blanco.

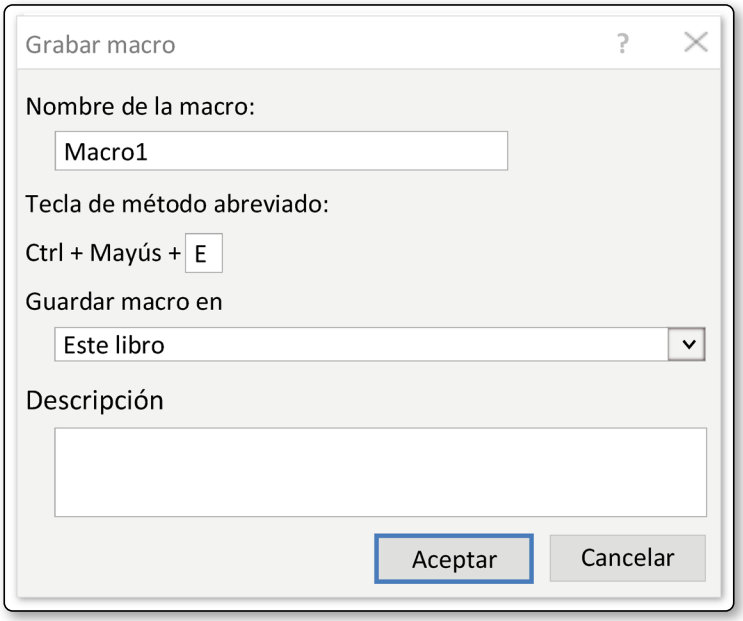
2.2.1.1 INTRODUCCIÓN AL GRABADOR DE MACROS

Antes de empezar a grabar macros es importante tener en cuenta una serie de premisas:

- Cuando se graba una macro, ésta, sólo se ejecutará en las celdas del rango seleccionado. Por tanto, si posteriormente se añaden más filas al rango, la macro no se aplicará en dichas filas.
- Si la macro que se va a crear o construir es larga y/o compleja siempre será mejor dividirla en varias macros más sencillas.
- No sólo las acciones ejecutadas en Excel se graban en una macro. Es decir, una macro de Excel puede almacenar varias operaciones concretas sobre una hoja de cálculo y, como remate final, la apertura del Microsoft Outlook para enviar por correo electrónico el archivo contenedor.

Dicho esto, para grabar una deberemos ir a la pestaña `Programador` y, una vez allí, pulsar en el icono de `Grabar Macro`. Si no se encuentra la pestaña **Programador**, puede que la pestaña se llame `Desarrollador`.

Debería aparecer algo como:



Como se puede apreciar en la imagen anterior, el primer dato que se nos solicita es el **nombre de la macro**. Este valor debe empezar por una letra y sólo admite letras, números y el símbolo de subrayado como posibles caracteres.

Si se desea, como es el caso de la imagen, se puede asignar un **método abreviado**. Para este cometido se puede recurrir a cualquier combinación, no obstante, la recomendación es que se use una tecla con **CTRL + MAYÚS** ya que la combinación de teclas anulará el método abreviado previamente asignado que tenga Excel.

En el desplegable de **Guardar macro en** podemos seleccionar varias opciones, aunque lo frecuente es dejar la opción por defecto, es decir, la opción de **Este libro**.

Ahora bien, es interesante saber qué, si la macro va a usarse en varios libros o archivos de Excel, la mejor opción es elegir **Libro de macros personal**. Con esto conseguiremos que esté disponible para todos los archivos Excel.

El cuadro de **Descripción** es opcional y nos permitirá introducir una descripción breve de lo que hace la macro o su funcionalidad.

Una vez rellenados todos los campos deberemos pulsar en **Aceptar** y, tras ello, ya podremos realizar todas nuestras acciones.

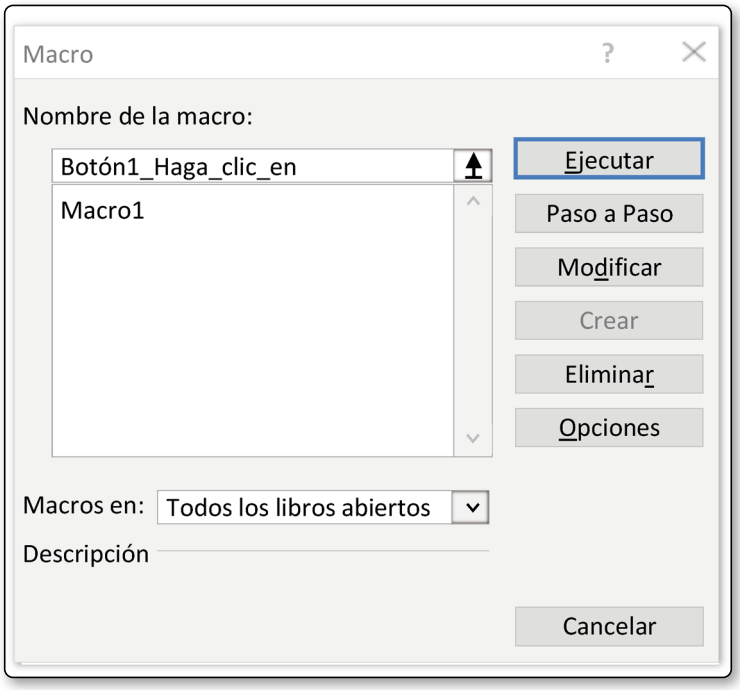
2.2.1.2 SOLUCIÓN

Primero deberemos posicionarnos en la celda A1 de la Hoja1 de nuestro libro de Excel y, en la pestaña **Inicio**, pulsar en el icono de **color de relleno**, que está representado por un bote de pintura en la sección de Fuente. Aquí pulsaremos sobre el color negro y, justo a la derecha, haremos clic en el icono de **color de fuente**, que está representado por una "A" mayúscula.

Posteriormente, escribiremos **Hola mundo** en la celda A1 y nos iremos a la pestaña de Programador. Aquí buscaremos el icono de **Detener grabación**, situado a la derecha del icono de **Macros** y que está a la derecha del icono de **Visual Basic**.

Si queremos ver el código fuente que se ha generado, lo que podemos hacer es ir a la pestaña Programador y pulsar en el icono de **Macros** que está a la derecha del icono de **Visual Basic**.

Al pulsar en **Macros**, debería aparecernos un cuadro de diálogo con todas las macros que tenemos disponibles. En esa lista seleccionamos nuestra macro, la que hemos denominado **Macro1**, y pulsamos el botón de **Modificar**.



Tras pulsar en el botón de modificar, deberíamos ver algo como:

```
Sub Macro1()
'
' Macro1 Macro
'
' Acceso directo: Ctrl+Mayús+E
'
    Range("A1").Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorLight1
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
    With Selection.Font
        .ThemeColor = xlThemeColorDark1
        .TintAndShade = 0
    End With
    ActiveCell.FormulaR1C1 = "Hola mundo"
    Range("A2").Select
End Sub
```

2.3 CREACIÓN DE MACROS POR CÓDIGO

2.3.1 Macro 2: Hola mundo

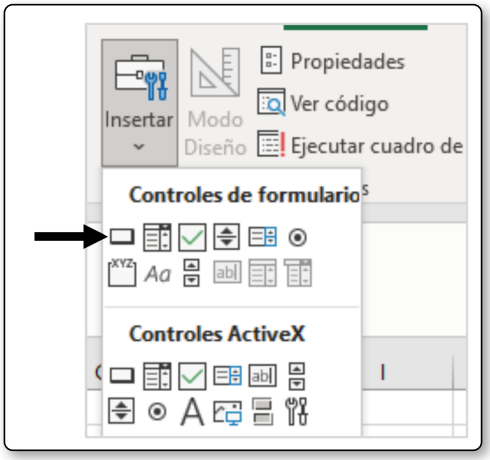
Se trata de crear un botón que, al hacer clic sobre él, nos muestre en la celda A1 el texto "Hola mundo".

2.3.1.1 SOLUCIÓN

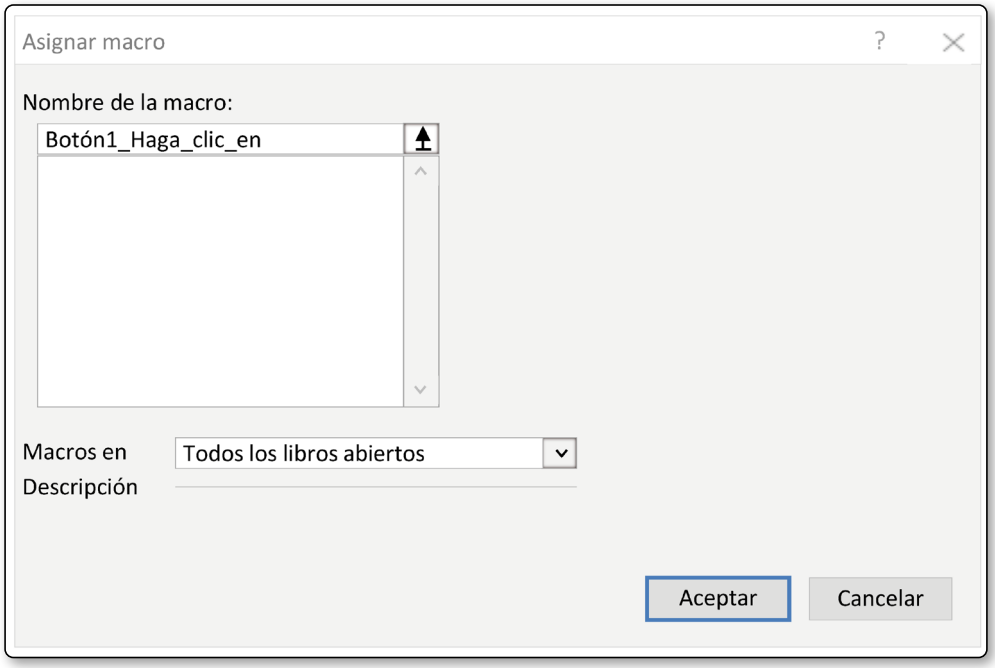
Antes de nada, y dado que es nuestra primera macro, deberemos comprobar que el documento de Microsoft Excel tiene activada la pestaña de "Programador". Si no es así, activarla como se ha indicado en el capítulo anterior.

Una vez que ya tengamos la opción de "Programador" habilitada, buscaremos y pulsaremos el icono que indica insertar, representado por un maletín con unas herramientas a la derecha.

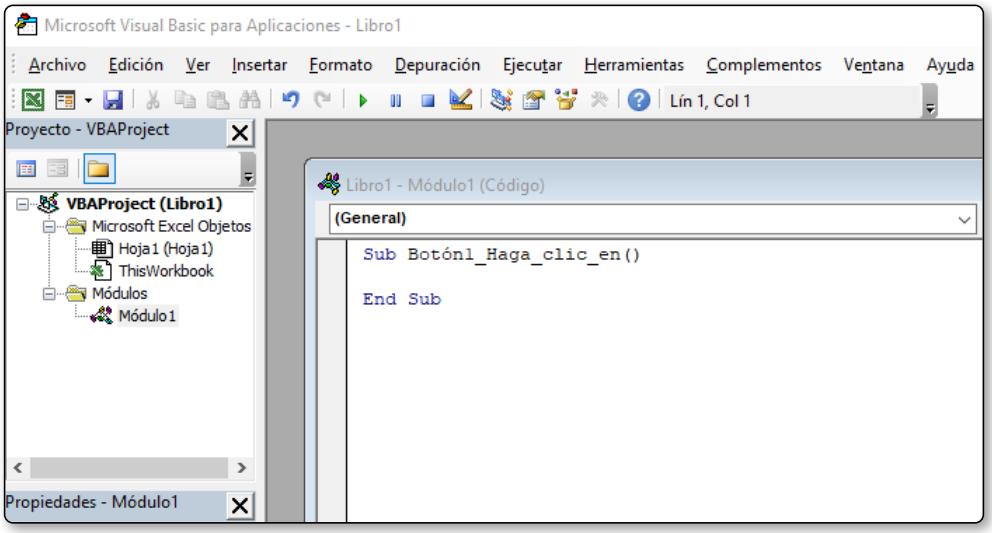
Tras haber pulsado la opción de "Insertar", deberemos seleccionar y presionar en el control de formulario denominado "Botón":



Lo que provocará que se nos muestre una pantalla solicitándonos el nombre de la macro. En este punto se nos asignará un nombre por defecto, por ejemplo, "Boton1_Haga_clic_en", pero podremos ponerle el nombre que deseemos, por ejemplo, "Boton_saludo".



Una vez que hayamos insertado el nombre de nuestra macro, nos aparecerá una pantalla a modo de diálogo emergente que titulada "Microsoft Visual Basic para Aplicaciones - Libro1" o algo similar. Dentro de esta pantalla deberíamos ver un diálogo que indica "Libro1 - Módulo1 (Código)".



Ahora, dentro del bloque **`Sub - End Sub`**, lo que deberemos escribir es lo siguiente:

```
Sub Botón1_Haga_clic_en()
    Hoja1.Cells(1, 1) = "Hola amigo!"
End Sub
```

Si ahora cerramos el diálogo "Microsoft Visual Basic para Aplicaciones - Libro1", que es donde hemos escrito nuestra macro, deberíamos ver un nuevo botón en nuestra hoja de cálculo con un nombre **`Botón 1`**.

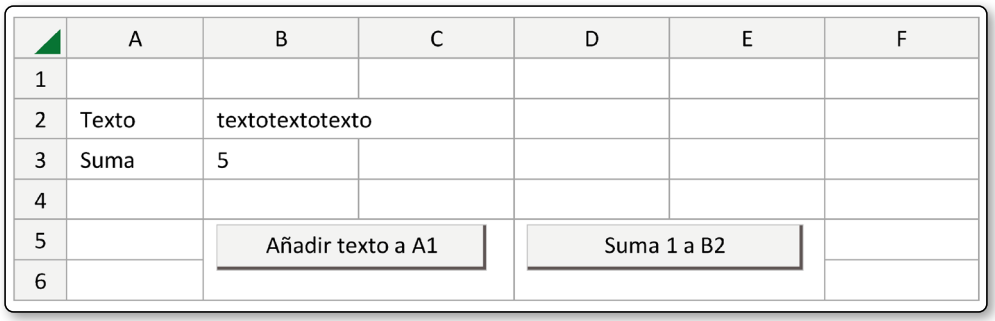
Si no nos gusta este nombre siempre podremos editarlo pulsando sobre el botón con el botón derecho de nuestro ratón y seleccionando la opción de **`Editar texto`**. No obstante, es posible que este proceso de renombrado haga que deje de funcionar la macro.

Si esto pasa, lo único de deberemos hacer es pulsar el botón derecho de nuestro ratón sobre el botón (otra vez) y seleccionar la opción de **`Asignar macro`**. Una vez que hayamos pulsado en esta opción nos saldrá la macro que creamos antes y, tras seleccionarla, sólo deberemos pulsar en **`Aceptar`**.

2.3.2 Macro 3: Anidación y suma en celdas separadas

Crear dos macros. Una que añada o concatene la palabra "texto" a la celda B2 y, otra, que sume uno al valor de la celda B3.

El resultado podría ser algo como lo siguiente:



2.3.2.1 SOLUCIÓN

Para resolver el añadido o concatenado con la celda B3, lo que haremos es crear un botón denominado **`Añadir texto`** y, después, en la pantalla del Editor de Visual Basic para Aplicaciones, añadir lo siguiente:

```
Sub AnadirTexto()
    Hoja1.Cells(2, 2) = Hoja1.Cells(2, 2) + "texto"
End Sub
```

Y para resolver el incremento de la celda B3, lo que haremos es crear un botón denominado **`Suma1`** y, después, en la pantalla del Editor de Visual Basic para Aplicaciones, añadir lo siguiente:

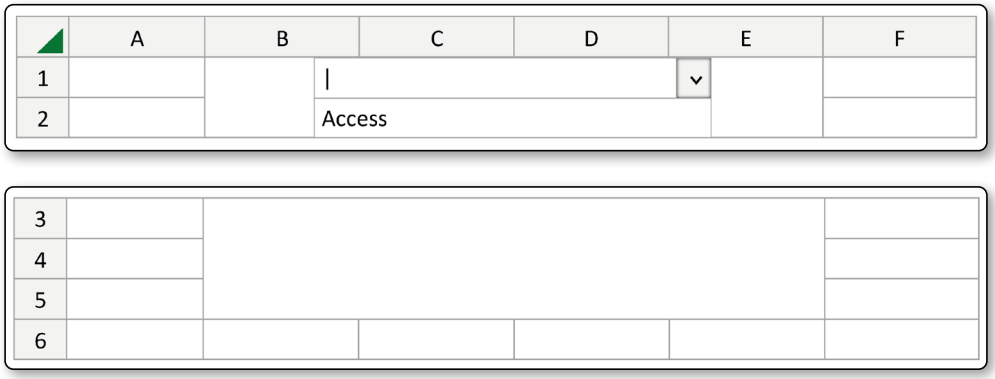
```
Sub Suma1()
    Hoja1.Cells(3, 2) = Hoja1.Cells(3, 2) + 1
End Sub
```

2.3.3 Macro 4: Rellenar un ComboBox

Insertar un elemento desplegable o cuadro combinado de ActiveX y rellenarlo cada vez que se active la hoja de Excel donde se encuentre.

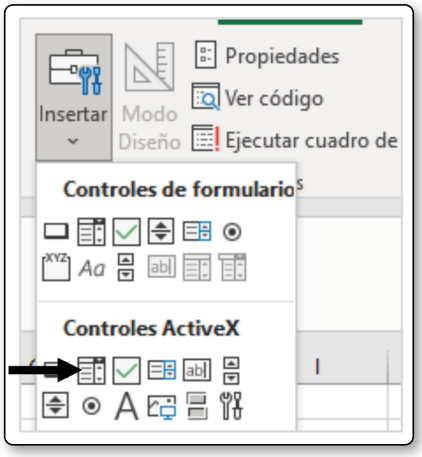
Los datos que deberá contener son: Access, Excel, OneNote, PowerPoint y Word.

El resultado debería ser algo similar a lo siguiente:



2.3.3.1 SOLUCIÓN

En primer lugar, lo que necesitamos es insertar un control de Cuadro Combinado o `ComboBox` en nuestra hoja de Excel. Para ello, lo que deberemos hacer es ir a la pestaña de `Programador` y, allí, pulsar en el icono de `Insertar`. Seguidamente, pulsaremos en el icono de `Cuadro combinado (Control ActiveX)`.

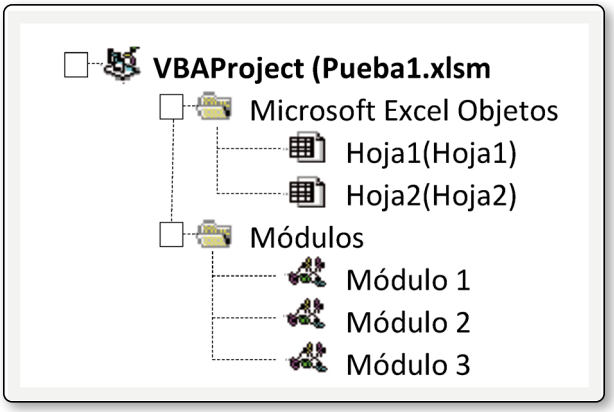


Después, nos iremos al editor de Visual Basic y, allí, insertaremos un código dentro del evento **Activate** del objeto **Worksheet**, el cual representa a todas y cada una de las hojas de un libro de Excel.

Aunque ya se verá más adelante, el objeto **Worksheet** es una pieza o miembro clave de la colección **Worksheets** que, a su vez pertenece al objeto **Workbook**.

Si hemos definido más de una hoja dentro de nuestro libro de Excel, el código que a continuación, se muestra, deberemos colocarlo en la hoja adecuada, es decir, deberemos seleccionar previamente el sitio donde se desea que actúe, en este caso la **Hoja1**, la cual cuelga directamente de **Microsoft Excel Objetos**.

Esto es, en la pantalla de *Microsoft Visual Basic para Aplicaciones*, debajo del menú y barra de botones, situado a la izquierda deberíamos poder ver la ventana de proyecto, que debería mostrar algo como:



Una vez hayamos seleccionado la página donde está el combo que acabamos de añadir realizando doble clic en el nombre de la hoja (en nuestro caso Hoja1), en el cuadro de diálogo o ventana que se nos abre y pegaremos el siguiente código:

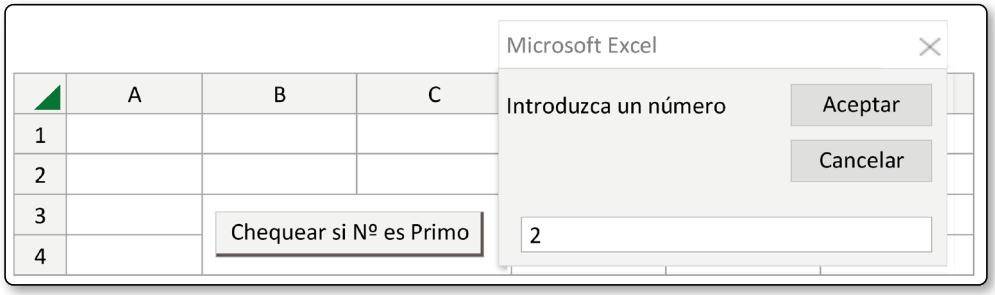
```
Private Sub Worksheet_Activate()
    ' Eliminamos todas las opciones de ComboBox1
    For i = ComboBox1.ListCount - 1 To 0 Step -1
            ComboBox1.RemoveItem i
    Next i

    ' Añadimos las opciones a ComboBox1
    With Me.ComboBox1
        .AddItem "Access"
        .AddItem "Excel"
```

```
            .AddItem "OneNote"
            .AddItem "PowerPoint"
            .AddItem "Word"
        End With
    End Sub
```

2.3.4 Macro 5: Detección de número primo

Ahora que nos hemos iniciado en las macros, vamos a trabajar con procedimientos y funciones. Para ello, recurriremos una macro sencilla que nos solicite un número a través de un diálogo emergente y verifique si es o no primo.

El resultado debería ser algo similar a:



2.3.4.1 SOLUCIÓN

Creo que ya todos sabremos que un número primo es aquel que sólo es divisible por sí mismo y por 1. Por tanto, si es divisible por cualquier otro número que no sea el valor introducido y 1, no será primo.

Para realizar esta comprobación podemos recurrir a varios métodos, aunque el más sencillo posiblemente sea el de ir comprobando si el número introducido es divisible por todos los números, desde 2, hasta la raíz cuadrada del mismo, o sea, nuestro número.

Dado que queremos usar procedimientos y funciones, lo que haremos es un procedimiento que actuará como activador de macro asociándolo a un botón y que será el encargado de solicitar el número, además de mostrar el resultado y, una función que, será la encargada de comprobar si el número introducido es o no primo.

Un posible resultado para este ejercicio podría ser:

```
Sub SolicitarNumero()
    Dim valor As String
    Dim primo As Boolean

    valor = InputBox("Introduzca un número")

    If IsNumeric(valor) = False Then Exit Sub

    If IsPrime(valor) Then
        MsgBox (CStr(valor) + " es un número primo")
    Else
        MsgBox (CStr(valor) + " NO es un número primo")
    End If
End Sub

Function IsPrime(x) As Boolean
    Dim n As Integer

    IsPrime = True
    n = 2
    Do While IsPrime And n <= Sqr(x)
        If (x Mod n = 0) Then
            IsPrime = False
        End If
        n = n + 1
    Loop
End Function
```