Comencemos a programar con

VBA - Access

Entrega 23

Objetos de Access
Formularios (2)

Eduardo Olaz

Propiedades de los formularios

Cuando estudiamos las clases vimos que en una clase podemos definir

- **Propiedades**
- **Métodos**
- **Eventos**.

Una propiedad es una característica de un objeto de una clase determinada que podemos leer, escribir o ambas cosas.

Los métodos son procedimientos definidos en una clase que pueden devolver valores, como en el caso de loas funciones, o sólo ejecutar una serie de instrucciones de código.

Los métodos pueden recibir parámetros, tanto por valor como por referencia.

Pueden devolver datos simples, como cadenas, valores numéricos, boleanos, etc… u objetos.

Lo dicho en el punto anterior es perfectamente aplicable a las propiedades.

Las propiedades de los formularios a las que podemos acceder, o establecer, afectan:

- Al diseño y aspecto del propio formulario, como su tamaño, posición, color…
- A los orígenes de datos a los que puede estar conectado, como vimos en la entrega anterior
- Otras características, como la forma en que se imprimirá en una impresora láser

Por la extensión del tema no podremos dar un repaso a todas las propiedades de los formularios, pero sí hablaremos de las más interesantes.

Aconsejo al lector que acuda a la ayuda de Access, donde podrá encontrar la información completa de cada una de las propiedades, incluyendo líneas de código.

Propiedades de Formato

Para hablar de las propiedades trataré de ajustarme a cómo están estructuradas en la ventana de propiedades de la versión de Access 2003.

Propiedad Título (Caption)

En un formulario controla el texto que aparecerá en la barra de título de un formulario.

Es una propiedad de tipo String y es de lectura – escritura.

Esta propiedad ya la usamos en el ejemplo del capítulo anterior

```
Caption = "Tabla Provincias"
```

Recordemos que la línea anterior es equivalente a:

```
Me.Caption = "Tabla Provincias"
```

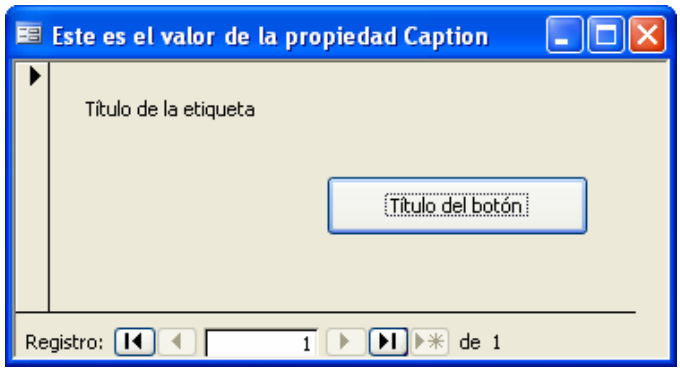
Como veremos más adelante hay otra serie de objetos que también poseen esta propiedad, como las etiquetas, informes, botones de comando, etc…

Vamos a poner en un formulario nuevo una etiqueta de nombre **`lblTitulo`** y un botón de comando de nombre **`cmdTitulo`**.

En el evento **Al cargar** del formulario escribimos:

```
Private Sub Form_Load()
    Caption = "Este es el valor de la propiedad Caption"
    lblTitulo.Caption = "Título de la etiqueta"
    cmdTitulo.Caption = "Título del botón"
End Sub
```

Al abrir el formulario veremos algo como esto:



Propiedad Presentación Predeterminada (DefaultView)

Especifica qué modo de presentación tendrá el formulario en el momento en que se abra.

Sus posibles valores son

Modo de Presentación	**Valor**	Descripción
Formulario simple	**0**	Valor Predeterminado. Muestra un registro cada vez.
Formularios continuos	**1**	Muestra múltiples registros, cada uno en su propia copia de la sección de detalle del formulario.
Hoja de datos	**2**	Muestra los datos del formulario organizados en filas y columnas como una hoja de cálculo correspondiéndose las filas a los registros y las columnas a los campos.
PivotTable	**3**	* Muestra el formulario como una tabla dinámica.
PivotChart	**4**	** Muestra el formulario como un gráfico dinámico.

(*) (**) Más adelante hablaremos sobre qué son las tablas y gráficos dinámicos.

De momento saber que es un sistema parametrizable para la presentación sintetizada de los datos agrupándolos por determinados valores.

Estas propiedades sólo pueden establecerse por código cuando el formulario está en modo diseño. Si tratamos de hacerlo en tiempo de ejecución nos dará el error Nº 2136.

Puede que más de uno se sorprenda, pero en un formulario por código no sólo podemos modificar sus características, sino incluso añadirle controles.

Algunos de estos procesos sólo pueden realizarse si ponemos el formulario en modo diseño.

Más adelante veremos todo esto con más detalle.

Propiedad AllowFormView

Controla si se permite mostrar el formulario en el modo de presentación **Formulario simple**.

Propiedad AllowDatasheetView

Controla si se permite el modo de presentación **Hoja de datos**

.Estas dos últimas propiedades son de tipo **Boleano**.

Propiedad Barras de Desplazamiento (ScrollBars)

Mediante esta propiedad podemos especificar la aparición o no, tanto de la barra de desplazamiento vertical como de la horizontal.

Es aplicable a formularios y cuadros de texto, teniendo en cuenta que los cuadros de texto sólo pueden tener barras verticales

Sus posibles valores son

Barras	**Valor**	Descripción
Ninguna	**0**	No aparecen barras de desplazamiento. Para los cuadros de texto es la opción predeterminada
Sólo horizontal	**1**	Barra de desplazamiento horizontal en el formulario
Sólo vertical	**2**	Aparece una barra de desplazamiento vertical.
Ambas	**3**	Es el valor predeterminado para los formularios. En el formulario aparecen barras de desplazamiento horizontal y vertical. No se aplica a cuadros de texto.

Propiedad Selectores de registro (RecordSelectors)

Permite Mostrar u ocultar la barra vertical selectora de registros.

Vamos a poner en un formulario un botón de comando con el nombre **`cmdSelectores`** y comprobemos el resultado cada vez que lo presionamos.

```
Private Sub cmdSelectores_Click()
    RecordSelectors = Not RecordSelectors
    If RecordSelectors Then
        cmdSelectores.Caption = "Ocultar selector de registro"
    Else
        cmdSelectores.Caption = "Mostrar selector de registro"
    End If
End Sub
```

Propiedad Botones de desplazamiento (NavigationButtons)

Permite Mostrar u ocultar los botones de desplazamiento entre registros y las etiquetas que indican su número.

Ponemos un botón con el nombre **cmdBotonesDeDesplazamiento** y comprobemos el resultado cada vez que lo presionamos.

```
Private Sub cmdBotonesDeDesplazamiento_Click()
    NavigationButtons = Not NavigationButtons
    If NavigationButtons Then
        cmdBotonesDeDesplazamiento.Caption = _
            "Ocultar Botones de Desplazamiento"
    Else
        cmdBotonesDeDesplazamiento.Caption = _
            "Mostrar Botones de Desplazamiento"
    End If
End Sub
```

Otras Propiedades de formato de tipo Bolean que pueden establecerse en tiempo de ejecución.

- **Separadores de registros (DividingLines)** especifica si habrá separadores de registros separando las secciones de un formulario o los registros mostrados en un formulario continuo.
- **Formulario movible (Moveable)** establece si el formulario se podrá o no desplazar por la pantalla.
- **Propiedad Activado (Enabled)** esta propiedad define si un objeto está activado y por tanto podrá responder a eventos.
- **Propiedad Bloqueado (Locked)** esta propiedad especifica si se pueden editar los datos de un control en la vista Formulario. Es muy útil para mostrar datos en un cuadro de texto, que no queremos que el usuario pueda cambiar.

Si queremos que se visualice en un formulario un campo, por ejemplo de tipo auto numérico, que ni queremos ni necesitamos que el usuario pueda editarlo, es habitual poner, para ese control, la propiedad **Enabled** a **False**, y la propiedad **Locked** a **True**.

Con eso podemos mantener la estética del formulario.

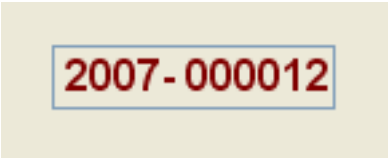
Yo personalmente suelo además poner el fondo como transparente.

Este es un ejemplo de formato para un TextBox, en el evento **Al cargar** del formulario.

```
Private Sub Form_Load()
    With txtidDato
        .Enabled = False      ' Sin activar
```

```
            .Locked = True          ' Bloqueado
            .BackStyle = 0          ' Fondo transparente
            .BorderStyle = 1        ' Borde fino
            .ForeColor = 128        ' Color Burdeos del texto
            .FontName = "Arial"     ' Fuente de tipo Arial
            .FontSize = 12          ' Tamaño de fuente 12 puntos
            .FontBold = True        ' Fuente en negrita
        End With
    End Sub
```

Así se vería el control:



Como hemos visto en el ejemplo, en cualquier momento podemos cambiar la forma como se muestra un control.

Animo al lector a que haga el siguiente ejemplo

Un formulario con una etiqueta y dos botones.

Cuando se pulsa uno de los botones aumenta el tamaño del texto.

Cuando se pulse el otro botón disminuiría su tamaño..

Propiedad Imagen (Picture)

Mediante esta propiedad podemos establecer la imagen que queremos mostrar como fondo del formulario, de un botón, informe, etc…

Admite distintos tipos de ficheros gráficos

Su sintaxis es

```
Objeto.Picture = "RutaYNombreDelFicheroGráfico "
```

```
Me.Picture = "c:\Imagénes\MiFoto.jpg"
cmdPaletas.Picture = "c:\Imagénes\Paleta.gif"
Picture = "c:\Imagénes\Escudo.bmp"
```

Nota:

Los Botones de Access no admiten mostrar simultáneamente un texto y una imagen.

Esto no es un problema insoluble, ya que podemos crearnos un fichero gráfico que incluya el texto, con cualquier herramienta de diseño.

Por ejemplo, aquí tenemos un botón con el gráfico típico de salida y el texto.

Podríamos incluso hacer cambiar el gráfico cuando se pulse el botón, y cuando se vuelve a soltar.

Para ello podemos aprovechar los eventos

- **Al bajar el Mouse (MouseDown)** que se produce al presionar un botón del ratón sobre el botón
- **Al subir el Mouse (MouseUp)** se produce al levantar el botón del ratón sobre el botón después de haberlo presionado

```
Private Sub cmdMiBoton_MouseDown( _
            Button As Integer, _
            Shift As Integer, _
            X As Single, Y As Single)
        cmdMiBoton.Picture = "C:\Pograma\BotonPresionado.bmp"
End Sub

Private Sub cmdMiBoton _MouseUp( _
            Button As Integer, _
            Shift As Integer, _
            X As Single, Y As Single)
        cmdMiBoton.Picture = "C:\Pograma\BotonNormal.bmp"
End Sub
```

Eventos del ratón MouseDown y MouseUp

En el controlador de estos eventos, podemos averiguar

- Mediante el parámetro **`Button`**, qué botón hemos presionado pudiendo tomar estos valores:

`acLeftButton`	→	1	Botón **izquierdo** del ratón
`acRightButton`	→	2	Botón **derecho**
`acMiddleButton`	→	4	Botón **central**

Podemos controlar también si hemos presionado, de forma simultánea alguna de estas teclas

- **Mayúscula** → 1
- **Control**. → 2
- **Alt** → 4

Y cualquier combinación de ellas (es la suma de los valores anteriores)

Este código nos mostrará en un cuadro de mensaje si se ha presionado alguna de las teclas anteriores mientras se pulsaba un botón del ratón.

```
Private Sub cmdMiBoton_MouseDown( _
            Button As Integer, _
            Shift As Integer, _
            X As Single, Y As Single)
    Select Case Shift
      Case 0
          MsgBox "Sólo el ratón"
      Case 1
          MsgBox "Tecla Mayúscula"
      Case 2
          MsgBox "Tecla Control"
      Case 3
          MsgBox "Teclas Mayúscula y Control"
      Case 4
          MsgBox "Tecla Alt"
      Case 5
          MsgBox "Teclas Mayúscula y Alt"
      Case 6
          MsgBox "Teclas Control y Alt"
      Case 7
          MsgBox "Teclas Mayúscula Control y Alt"
      End Select
End Sub
```

Estas combinaciones nos permiten definir diferentes comportamientos de un control al pulsarlo con el ratón dependiendo de si se ha pulsado simultáneamente una tecla **Mayúscula Control** o **Alt**, o incluso combinaciones de ellas.

Los parámetros **X** e **Y** contienen las coordenadas del ratón, en **Twips** respecto a la esquina superior izquierda del botón.

Más adelante veremos qué clase de unidad es un **Twip**.

El parámetro no es específico del evento **`MouseDown`**.

Por ejemplo, un cuadro de texto posee el evento **`KeyDown`**, que también posee el parámetro **`Shift`**, al que se accedería de forma similar a lo visto con **`MouseDown`** .

```
Private Sub txtDato_KeyDown( _
                KeyCode As Integer, _
```

```
                    Shift As Integer)
End Sub
```

Gestión de colores en un control

Propiedad Estilo del fondo (BackStyle)

Controla si el fondo de un control es opaco ó transparente.

0	**Transparente**	El control completo o parte de él serán transparentes
1	**Opaco**	El fondo del control tendrá el color definido por la propiedad **Color de fondo (BackColor)**

Propiedad Color de fondo (BackColor)

La propiedad que gobierna el color de fondo de la mayoría de los objetos es la propiedad **BackColor**.

Esta propiedad admite valores de tipo Long, en el rango:

del RGB(0,0,0) → 0 (Negro)

al valor RGB(255,255,255) → 16.777.215 → (Blanco)

Otra forma de acceder a los colores es mediante la función **QBColor**(Índice), que admite para el parámetro Índice, valores del 0 al 15

Índice	Color	Índice	Color
0	Negro	8	Gris
1	Azul	9	Azul claro
2	Verde	10	Verde claro
3	Aguamarina	11	Aguamarina claro
4	Rojo	12	Rojo claro
5	Fucsia	13	Fucsia claro
6	Amarillo	14	Amarillo claro
7	Blanco	15	Blanco brillante

Los veteranos que hayáis programado con aquellos “entrañables” lenguajes como:

GW Basic, Q Basic, T Basic, etc… identificaréis enseguida estos parámetros.

Y hay una tercera, que es la utilización de las constantes definidas en el módulo **ColorConstants**

Estas son:

vbBlack	→	Negro	→	0
vbBlue	→	Azul	→	16.711.680
vbCyan	→	Cyan (Azul primario)	→	16.776.960
vbGreen	→	Verde	→	65.280
vbMagenta	→	Magenta (Rojo primario)	→	16.711.935
vbRed	→	Rojo	→	255
vbWhite	→	Blanco	→	16.777.215
vbYellow	→	Amarillo	→	65.535

Propiedad Color del texto (ForeColor)

Esta propiedad define el color que va a mostrar el texto de un control.

Hay otras propiedades que definen aspectos del color en un objeto

BorderColor

Como su propio nombre indica, define el color de los bordes de un objeto.

Fijémonos en el siguiente código:

```
Private Sub Form_Load()
    With txtidDato
        .Enabled = False
        .Locked = True
        .BackStyle = 0
        .BackColor = vbYellow
        .BorderStyle = 1
        .ForeColor = vbBlue
        .FontName = "Arial"
        .FontSize = 12
        .FontBold = True
        .BorderWidth = 2
        .BorderColor = vbRed
    End With
End Sub
```

Define un cuadro de texto (**TextBox**) en el que lo que escribamos se verá en color azul, tendrá un borde rojo y el fondo del mismo será de color amarillo.

Propiedad Color de los bordes (BorderColor)

Define el color que van a mostrar los bordes de un control.

El grosor de los bordes lo define la propiedad **Ancho de los bordes (BorderWidth)**

Otras propiedades para el formato de objetos

Propiedad Estilo de los Bordes (BorderStyle)

Es una propiedad que funciona de forma diferente, si se aplica a un formulario o se aplica a un control.

En un **formulario** puede tomar los siguientes valores

0	**Ningún borde**	No aparece ningún borde ni ningún elemento relacionado con los mismos. No se puede cambiar el tamaño del formulario.
1	**Bordes delgados**	El formulario tiene un borde fino y puede mostrar los elementos relacionados. No se puede cambiar su tamaño.
2	**Bordes ajustables**	Es el valor predeterminado de Access. Puede mostrar los elementos relacionados con el borde, y ajustar su tamaño con el cursor.
3	**Cuadro de Diálogo**	El borde del formulario se pone de tamaño doble. Nos muestra una barra de título, un botón cerrar y un menú control. El formulario no puede ni maximizarse, ni minimizarse ni cambiar de tamaño. Además puede mostrarse como **Modal** (por encima de todas las demás ventanas) poniendo sus propiedades **Emergente (PopUp)** y **Modal** al valor **True**.

En un **control** sus valores pueden ser

0	**Transparente**	Predeterminado sólo para etiquetas, gráficos y sub-informes
1	**Sólido**	Línea sólida (Predeterminado)
2	**Guiones**	Línea de guiones
3	**Guiones cortos**	Línea de guiones cortos
4	**Puntos**	Línea de puntos
5	**Puntos separados**	Línea de puntos separados
6	**Guión punto**	Línea con una combinación de punto y raya
7	**Guión-punto-punto**	Línea con una combinación de raya punto y punto
8	**Sólido doble**	Líneas dobles

Propiedad ancho de los bordes (BorderWidth)

Especifica el ancho del borde de un control.

Es de tipo Byte. Valores admitidos:

0	**Trazo fino**	Predeterminado sólo para etiquetas, gráficos y sub-informes
1 a 6	**De 1 a 6 ptos.**	Ancho en puntos

En la ayuda de Access puede encontrar información completa de cómo hacer efectiva esta propiedad.

Ejercicio:

Vamos a crear un formulario en el que vamos a colocar tres barras que funcionarán al estilo de las **Barras de Progreso (ProgressBar)**.

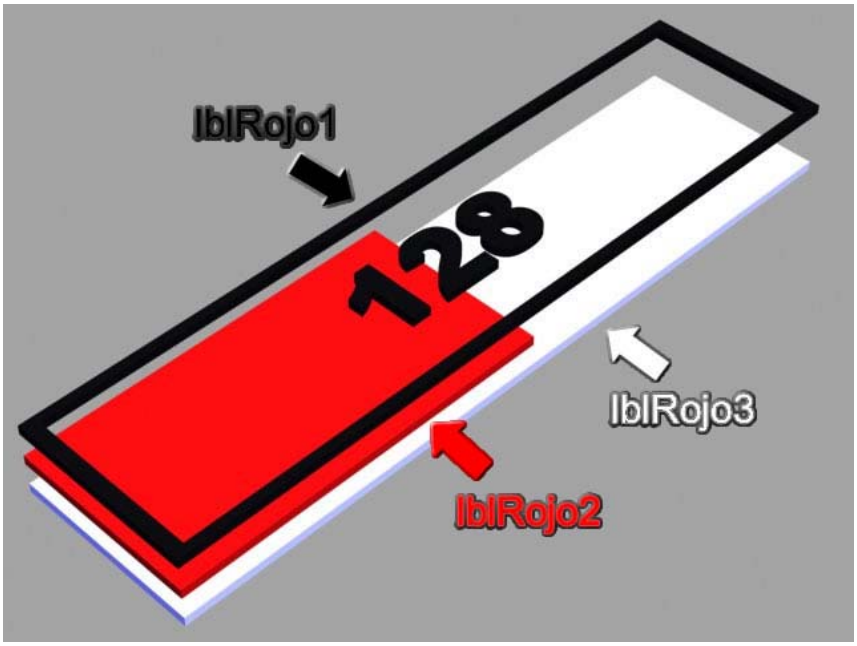
Haremos que al desplazar las barras, varíe la proporción de Rojo, Verde y Azul del fondo del formulario, con lo que irá variando su color.

Hay varios aspectos interesantes de este ejercicio.

Uno de ellos es que no vamos a utilizar ningún control **ActiveX**, como los que vienen en la librería **MsComCtlLyb**, para realizar las barras de desplazamiento.

Éstas las vamos a crear nosotros.

Fijémonos en el siguiente gráfico:



Para imitar la barra utilizaremos 3 etiquetas colocadas una encima de la otra.

La inferior **lblColor3** tendrá el fondo de color blanco.

La intermedia **lblColor2** será de color Rojo, Verde ó Azul, e irá cambiando su anchura cuando pulsemos con el cursor en la etiqueta superior y lo desplacemos.

La superior **lblColor1**, a diferencia de las otras dos, será transparente, además de poseer un borde negro y mostrar un valor numéricos, del **0** al **255**, también en negro.

"El viaje más largo empieza con el primer paso" (proverbio chino)

Para empezar con el proyecto, haremos nuestras primeras pruebas.

En un formulario colocamos una etiqueta con el texto Rojo, ocupando casi todo el ancho del mismo.

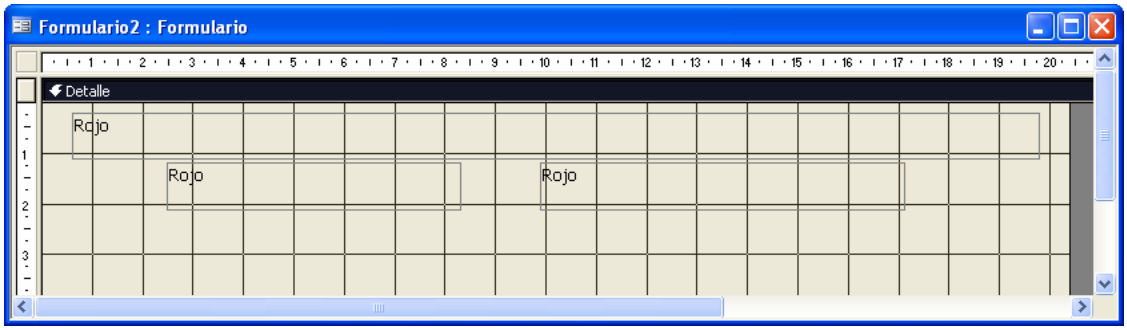
Le ponemos como nombre **lblRojo1**

Seleccionamos la etiqueta, la copiamos y la pegamos dos veces.

A estas dos nuevas etiquetas les ponemos como nombres **lblRojo2** y **lblRojo3**.

Estas dos últimas las hacemos más pequeñas y las ponemos cerca de la primera, pero por debajo de ella.

Quedarán con un aspecto similar a éste.



Ahora seleccionamos la etiqueta **lblRojo1** y pulsamos en la opción **Traer al frente** del menú **Formato**.

A continuación seleccionamos la etiqueta **lblRojo3** y pulsamos en la opción **Traer al frente** del menú **Formato**.

Con esto ya hemos conseguido que su orden se adapte al del gráfico anterior.

El siguiente paso es adaptar el tamaño de las etiquetas.

La inferior será igual a la superior, y la del medio tendrá una anchura mitad.

Además las colocaremos una encima de la otra.

Propiedad Izquierda (Left)

Especifica la distancia de la parte izquierda de un objeto a la parte izquierda del objeto que lo contiene.

Podemos especificar, por ejemplo la distancia de una etiqueta respecto del borde izquierdo de un formulario o de un informe. Es de Lectura y Escritura.

Existe otra propiedad semejante:

Propiedad Superior (Top)

Especifica la distancia de la parte superior de un objeto al borde superior del objeto que lo contiene.

Ambas son de tipo `Long` para los informes y de tipo Integer para los formularios.

Propiedades Alto y Ancho (Height) y (Width)

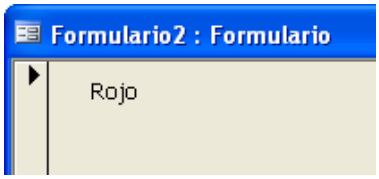
Height especifica la altura de un objeto

Width especifica su anchura

Vamos a utilizar estas propiedades para colocar y dimensionar correctamente las etiquetas.

En el evento **Al cargar** del formulario escribimos:

Abrimos el formulario, y no parece que suceda nada espectacular, ya que sólo nos muestra la palabra Rojo.



Pero esa es precisamente la demostración de que hemos conseguido una parte de lo que queríamos y es que las tres etiquetas están colocadas exactamente una encima de la otra.

Ahora vamos a mostrar las etiquetas, para lo que modificaremos el código.

En vez de ponerlo todo en el evento Al Cargar haremos que desde este evento llame al procedimiento **`FormateaEtiquetas`** que es el que hará todo el trabajo:

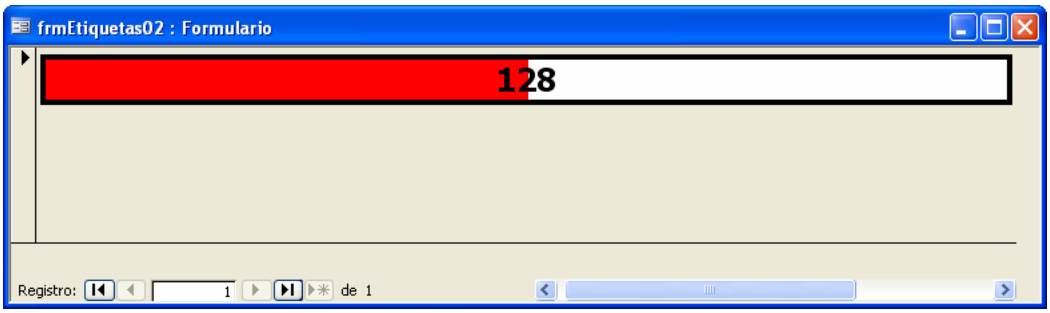
Si queremos mantener un "histórico" de los pasos que vamos a dar, guardamos este primer formulario con el nombre, **frmEtiquetas01**.

Y esta nueva versión mejorada, la grabamos con el nombre **frmEtiquetas02**.

```
Private Sub Form_Load()
    FormateaEtiquetas
End Sub

Private Sub FormateaEtiquetas()
    With lblRojo1
        .BorderStyle = 1          ' Borde de línea sólida
        .BorderWidth = 3          ' Anchura de 3 puntos
        .BackStyle = 0            ' Fondo transparente
        .TextAlign = 2            ' Texto centrado
        .Caption = 128            ' Texto inicial de la etiqueta
        .FontSize = 18            ' Tamaño de la fuente
        .FontBold = True          ' Texto en negrita
    End With

    With lblRojo2
        .Height = lblRojo1.Height
        .Width = lblRojo1.Width / 2
        .Caption = ""
        .BackStyle = 1            ' Fondo opaco
        .BackColor = vbRed        ' Color rojo para el fondo
        ' Salvo la anchura, sus propiedades geométricas _
          las ponemos como las de lblRojo1
        .Width = lblRojo1.Width / 2
```

```
        .Height = lblRojo1.Height
        .Left = lblRojo1.Left
        .Top = lblRojo1.Top
    End With

    With lblRojo3
        .Height = lblRojo1.Height
        .Width = lblRojo1.Width
        .Caption = ""
        .BackStyle = 1          ' Fondo opaco
        .BackColor = vbWhite    ' Color blanco
        ' Igualamos sus propiedades geométricas _
          con las de lblRojo1
        .Width = lblRojo1.Width
        .Height = lblRojo1.Height
        .Left = lblRojo1.Left
        .Top = lblRojo1.Top
    End With
End Sub
```

El resultado de este código es éste:



En el código aparece la propiedad **TextAlign**

Propiedad Alineación del texto (TextAlign)

Con esta propiedad podemos definir la colocación del texto que aparece en un control

0	**General**	Texto a la izquierda, números y fechas a la derecha
1	**Izquierda**	Todos los datos a la izquierda
2	**Centro**	Centrado
3	**Derecha**	Alineación a la derecha
4	**Distribuir**	El contenido se reparte por la anchura del control. **Ojo**: No es lo mismo que la Justificación del texto de Word.

El valor pasado a la propiedad **TextAlign** ha sido 2, con lo que vemos el valor 128 centrado en la etiqueta.

Antes de continuar vamos a dar formato al formulario.

Primero ponemos la propiedad Estilo de los bordes al modo Cuadro de diálogo, desde su ventana Propiedades.

A continuación, por código haremos las siguientes operaciones:

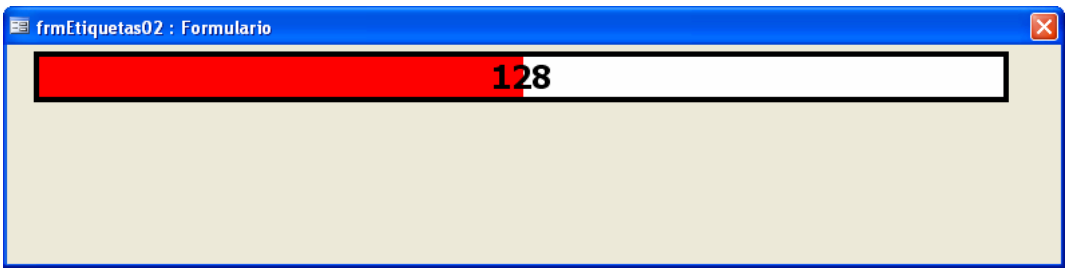
- Quitar la barra de desplazamiento que aparece en horizontal
- Quitar el selector de registros
- Quitar los botones de desplazamiento
- Quitar el separador de registros

Para ello crearemos el procedimiento **`FormateaFormulario`**, que será llamado desde el evento Al Cargar.

Para conseguirlo el código quedará así:

```
Private Sub Form_Load()
    FormateaEtiquetas
    FormateaFormulario
End Sub

Private Sub FormateaFormulario()
    ScrollBars = 0              ' Sin barras de desplazamiento
    RecordSelectors = False     ' Sin selector de registros
    NavigationButtons = False   ' Sin Botones desplazamiento
    DividingLines = False       ' Sin Separador de registros
    BorderStyle = 3             ' Estilo cuadro de diálogo
End Sub
```

Con estas modificaciones el formulario quedará así:



Vemos que el aspecto del formulario ha quedado mucho más limpio.

Tengo que reconocer que todas estas operaciones podrían haberse realizado sin escribir una sola línea de código, mediante la vista propiedades del formulario, pero al hacerlo de esta forma he pretendido dos cosas:

- Mostrar cómo se puede hacer por código
- Demostrar el alto grado de control que podemos tener sobre el diseño del mismo

El paso siguiente es conseguir que la barra roja se desplace cuando presionemos sobre las etiquetas y desplacemos el cursor sobre ellas.

El principal evento que vamos a utilizar es el **MouseMove**.

Vamos a analizarlo con más detenimiento:

Análisis de eventos

Antes de seguir vamos a analizar los gestores de los eventos que necesitaremos para conseguir nuestro objetivo.

Evento MouseMove

Este evento se produce cuando el usuario desplaza el ratón por encima de un objeto que lo genere.

La sintaxis de su gestor es:

```
Private Sub NombreDelobjeto_MouseMove( _
                    Button As Integer, _
                    Shift As Integer, _
                    X As Single, _
                    Y As Single)
```

Si nos fijamos, los parámetros son los mismos que los del evento que hemos analizado en uno de los puntos anteriores.

Por el momento, el parámetro que nos interesa es el **X**, que define la coordenada horizontal de la punta del ratón, respecto al borde izquierdo del control.

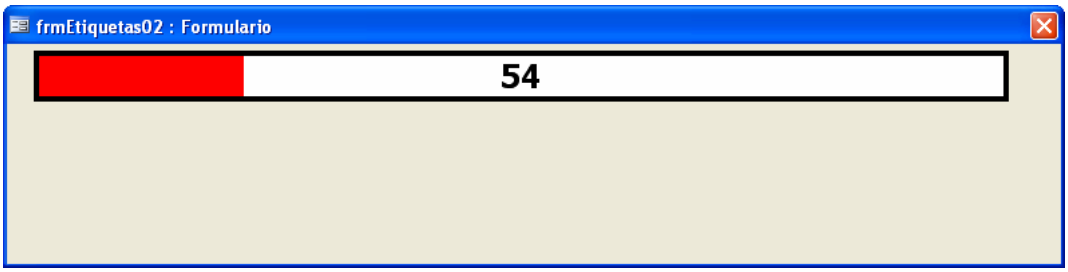
Lo interesante de esta coordenada es que, al estar en el mismo tipo de unidades que la propiedad `Width` de las etiquetas, podemos tomar su valor para definir la anchura de la etiqueta roja **lblRojo02**.

Añadimos este código para controlar el evento **MouseMove** de la etiqueta **lblRojo02**.

```
Private Sub lblRojo1_MouseMove( _
                    Button As Integer, _
                    Shift As Integer, _
                    X As Single, _
                    Y As Single)
        Dim bytRojo As Byte
        Dim lngAnchoEtiquetas As Long
        lngAnchoEtiquetas = lblRojo1.Width
        If X <= lblRojo1.Width And X >= 0 Then
            lblRojo2.Width = X
            bytRojo = Round(255 * X / lngAnchoEtiquetas)
            lblRojo1.Caption = bytRojo
        End If
End Sub
```

Abrimos el formulario, y vemos que ya hemos conseguido la parte más importante del objetivo, que el conjunto simule una barra de progreso, que aumente o disminuya cuando

pasemos el ratón por encima, y que el valor presentado en la etiqueta superior vaya de 0 a 255, en función de la posición del ratón.



Pero aquí faltan todavía cosas.

Hemos quedado que la barra debe moverse cuando movamos el ratón por encima, sólo si tenemos presionada la tecla izquierda del mismo.

En cambio se desplaza aunque no tengamos presionada la tecla izquierda.

Esto significa que en el gestor del evento, antes de cambiar el tamaño de la etiqueta, o que la etiqueta superior muestre el valor numérico, se debe comprobar si la tecla izquierda está presionada.

La comprobación la efectuaremos leyendo la variable Boleana **blnBotonBajado**, definida a nivel del módulo de clase del formulario.

Esta variable se pondrá a **True** cuando presionemos el botón izquierdo del ratón, y se pondrá a **False**, cuando levantemos el botón.

Para ello nos aprovecharemos del evento **MouseDown** de la etiqueta **lblRojo1**, y del evento **MouseUp** de la misma.

```
Option Explicit

Private blnBotonBajado As Boolean

- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -

Private Sub lblRojo1_MouseDown( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    blnBotonBajado = True
End Sub

Private Sub lblRojo1_MouseUp( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
```

```
        blnBotonBajado = False
    End Sub
```

Con esto ya hemos conseguido que las etiquetas funcionen como deseábamos.

¿Del todo?

Si nos fijamos en su funcionamiento, vemos que si presionamos el botón sobre la etiqueta, sin desplazarlo por encima de la misma, la barra roja no cambia hasta que movamos el botón.

Podemos evitar esta "pequeña deficiencia", llamando directamente al gestor del evento **MouseMove** desde el gestor del evento **MouseDown**.

Modificaremos el evento de la siguiente forma:

```
    Private Sub lblRojo1_MouseDown( _
                    Button As Integer, _
                    Shift As Integer, _
                    X As Single, Y As Single)
        ' Comprobamos que el botón pulsado es el izquierdo
        If Button = acLeftButton Then
            blnBotonBajado = True
            ' Llamamos al gestor del evento MouseMove _
              de lblRojo1
            lblRojo1_MouseMove Button, Shift, X, Y
        End If
    End Sub
```

Un asunto interesante de esto es comprobar que podemos llamar al gestor de un evento cualquiera como si fuera un procedimiento normal.

Por supuesto le deberemos pasar los parámetros adecuados, si es que los posee.

En este caso mediante la línea

```
lblRojo1_MouseMove Button, Shift, X, Y
```

le pasamos los mismos parámetros que recibe del evento **MouseDown** de **lblRojo1**

Antes de seguir con nuestro "pequeño proyecto" haremos unas pequeñas modificaciones, como poner la declaración de las variables **bytRojo** y **lngAnchoEtiquetas** a nivel del módulo, en vez de en el procedimiento `lblRojo1_MouseMove`.

Con esto el código del módulo de clase del formulario quedará:

Código intermedio del proyecto

```
    Option Compare Database
    Option Explicit

    Private blnBotonBajado As Boolean
    Dim bytRojo As Byte
    Dim lngAnchoEtiquetas As Long
```

```
Private Sub Form_Load()
    FormateaEtiquetas
    FormateaFormulario
End Sub

Private Sub lblRojo1_MouseDown( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    ' Comprobamos que el botón pulsado es el izquierdo
    If Button = acLeftButton Then
        blnBotonBajado = True
        ' Llamamos al gestor del evento MouseMove _
          de lblRojo1
        lblRojo1_MouseMove Button, Shift, X, Y
    End If
End Sub

Private Sub lblRojo1_MouseUp( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    blnBotonBajado = False
End Sub

Private Sub lblRojo1_MouseMove( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
        lngAnchoEtiquetas = lblRojo1.Width
        If blnBotonBajado Then
            If X <= lblRojo1.Width And X >= 0 Then
                lblRojo2.Width = X
                bytRojo = Round(255*X / lngAnchoEtiquetas)
                lblRojo1.Caption = bytRojo
            End If
        End If
End Sub
```

```
Private Sub FormateaFormulario()
     ' Sin barras de desplazamiento
     ScrollBars = 0
     'Sin selector de registros
    RecordSelectors = False
     ' Sin Botones de desplazamiento
     NavigationButtons = False
     ' Sin Separador de registros
     DividingLines = False
End Sub

Private Sub FormateaEtiquetas()
    With lblRojo1
        .BorderStyle = 1          ' Borde de línea sólida
        .BorderWidth = 3          ' Anchura de 3 puntos
        .BackStyle = 0            ' Fondo transparente
        .TextAlign = 2            ' Texto centrado
        .Caption = 128            ' Texto inicial _
                                    de la etiqueta
        .FontSize = 18            ' Tamaño de la fuente
        .FontBold = True          ' Texto en negrita
    End With

    With lblRojo2
        .Height = lblRojo1.Height
        .Width = lblRojo1.Width / 2
        .Caption = ""
        .BackStyle = 1            ' Fondo opaco
        .BackColor = vbRed        ' Color rojo para el fondo
        ' Salvo la anchura, ponemos _
          sus propiedades geométricas _
          como las de lblRojo1
        .Width = lblRojo1.Width / 2
        .Height = lblRojo1.Height
        .Left = lblRojo1.Left
        .Top = lblRojo1.Top
    End With

    With lblRojo3
        .Height = lblRojo1.Height
```

```
            .Width = lblRojo1.Width
            .Caption = ""
            .BackStyle = 1          ' Fondo opaco
            .BackColor = vbWhite    ' Color blanco
            ' Igualamos sus propiedades geométricas _
              con las de lblRojo1
            .Width = lblRojo1.Width
            .Height = lblRojo1.Height
            .Left = lblRojo1.Left
            .Top = lblRojo1.Top
        End With
    End Sub
```

Ya hemos conseguido una parte importante de nuestro proyecto.

Ahora guardamos el formulario **frmEtiquetas02** y para la versión siguiente lo guardamos como **frmEtiquetas03**.

Creación de una barra de progreso para los colores básicos Verde y Azul.

Vamos a poner seis etiquetas nuevas

lblVerde1 **lblVerde2** **lblVerde3**

lblAzul1 **lblAzul2** **lblAzul3**

A continuación modificamos el evento Load de la siguiente manera:

```
Private Sub Form_Load()
    FormateaEtiquetas "Rojo"
    FormateaEtiquetas "Verde"
    FormateaEtiquetas "Azul"
    FormateaFormulario
End Sub
```

Al procedimiento `FormateaEtiquetas` le estamos pasando un parámetro que antes no tenía.

Para que no nos de error, deberemos por tanto modificar la cabecera y el diseño del mismo.

Un inciso previo:

¿Cuál es el objetivo de todo esto?

Si utilizara el mismo sistema que en el código inicial, sería preciso repetir el procedimiento tres veces, una para cada color de etiquetas.

Nos podemos ahorrar todo este montón de código, si somos capaces de diseñar en el procedimiento un sistema que permita manejar las diferentes etiquetas de forma dinámica.

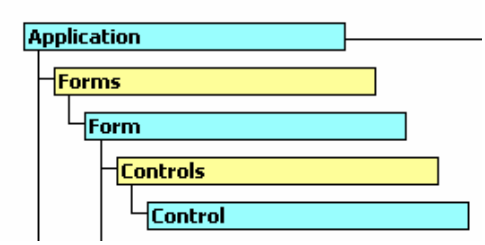
Asignación dinámica de controles desde la colección Controls.

Cuando analizamos las colecciones, vimos que había dos formas de recuperar un elemento de las mismas.

La primera era utilizar el índice del elemento contenido. `NombreColección(`**`Indice`**`)`

La segunda era utilizando una clave única de tipo **`string`** creada en el momento en que se añadía un elemento a la colección.

Con los controles de un formulario sucede algo semejante.



Además podemos realizar un recorrido por los mismos mediante la estructura **`For Each`**...

Vemos que el formulario tiene la colección **`Controls`**.

Esta colección comienza con el índice cero.

Acceso mediante `For Each ...`

En un módulo estándar vamos a escribir el siguiente código:

```
Public Sub ListaControles1(Formulario As Form)
    Dim ctl As Control
    For Each ctl In Formulario
        Debug.Print ctl.Name
    Next ctl
End Sub
```

Si llamamos a este procedimiento mediante

`ListaControles1 Form_frmEtiquetas03`

en mi caso el procedimiento escribe en la ventana inmediato lo siguiente:

```
lblRojo3
lblRojo2
lblRojo1
lblVerde3
lblVerde2
lblVerde1
lblAzul3
lblAzul2
lblAzul1
```

Acceso mediante un índice: `Controls(Índice)`

En el mismo módulo que en el procedimiento anterior escribimos:

```
Public Sub ListaControles2(Formulario As Form)
    Dim i As Integer
    Dim ctl As Control
    For i = 0 To Formulario.Controls.Count - 1
        Set ctl = Formulario.Controls(i)
        Debug.Print ctl.Name
    Next i
End Sub
```

Si llamamos al procedimiento mediante

`ListaControles2 Form_frmEtiquetas03`

El resultado es el mismo que en el procedimiento **`ListaControles1`**.

Acceso mediante el nombre del control: `Controls(Nombre)`

Se puede acceder a un control mediante su nombre.

Por ejemplo, este procedimiento muestra el nombre y la anchura de cada control accediendo a él mediante su nombre:

```
Public Sub ListaControles3(Formulario As Form)
    Dim ctl As Control
    Dim ctlNombre As Control
    Dim strNombre As String
    For Each ctl In Formulario
        strNombre = ctl.Name
        Set ctlNombre = Formulario.Controls(strNombre)
        Debug.Print ctlNombre.Name, ctlNombre.Width
    Next ctl
End Sub
```

Si ejecutamos **`ListaControles3 Form_frmEtiquetas03`** me muestra:

```
lblRojo3        4091
lblRojo2        3296
lblRojo1        10886
lblVerde3       4091
lblVerde2       3296
lblVerde1       7196
lblAzul3        4091
lblAzul2        3296
lblAzul1        7376
```

Vemos que efectivamente estamos asignando a la variable **`ctlNombre`**, de tipo Control, cada uno de los controles obtenidos desde la colección **`Controls`** a la que se le pasa el parámetro del nombre del control.

¿Cómo afecta esto a nuestro proyecto?

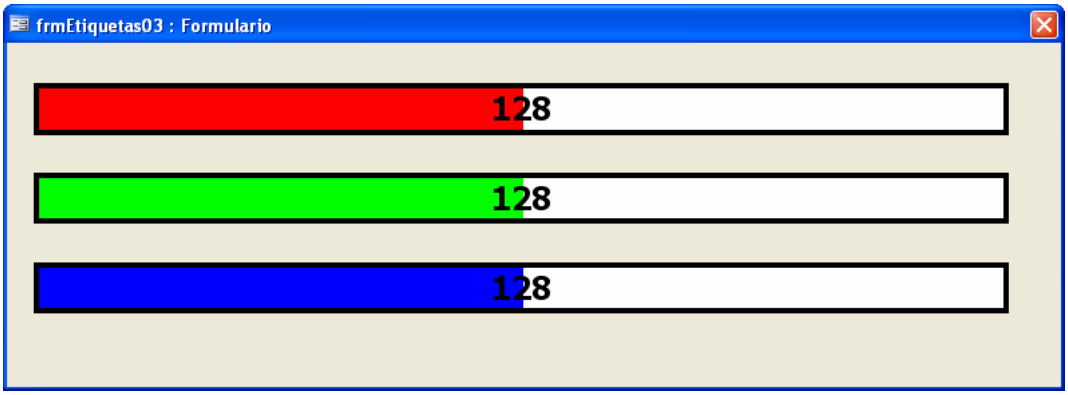
Sencillamente, si le pasamos al procedimiento `FormateaEtiquetas` una cadena que nos permita construir el nombre de la etiqueta, podremos ajustar el formato de cada etiqueta sin tener que repetir todo el código del mismo.

Podría ser esto:

```
Private Sub FormateaEtiquetas(ByVal NombreColor As String)
    Const SEPARACION As Long = 500
    Dim lngTop As Long
    Dim lngColor As Long
    Dim lbl As Label
    Dim strNombreEtiqueta As String

    Select Case NombreColor
    Case "Rojo"
        lngTop = SEPARACION
        lngColor = vbRed
    Case "Verde"
        lngTop = 3 * SEPARACION
        lngColor = vbGreen
    Case "Azul"
        lngTop = 5 * SEPARACION
        lngColor = vbBlue
    End Select

    strNombreEtiqueta = "lbl" & NombreColor & "1"
    Set lbl = Controls(strNombreEtiqueta)
    With lbl
        .Top = lngTop
        .Width = lblRojo1.Width
        .BorderStyle = 1          ' Borde de línea sólida
        .BorderWidth = 3          ' Anchura de 3 puntos
        .BackStyle = 0            ' Fondo transparente
        .TextAlign = 2            ' Texto centrado
        .Caption = 128            ' Texto inicial de la etiqueta
        .FontSize = 18            ' Tamaño de la fuente
        .FontBold = True          ' Texto en negrita
    End With
```

```
    strNombreEtiqueta = "lbl" & NombreColor & "2"
    Set lbl = Controls(strNombreEtiqueta)
    With lbl
        .Top = lngTop
        .Caption = ""
        .BackStyle = 1          ' Fondo opaco
        .BackColor = lngColor    ' Color para el fondo
        ' Salvo la anchura, sus propiedades geométricas _
          como las de lblRojo1
        .Width = lblRojo1.Width / 2
        .Height = lblRojo1.Height
        .Left = lblRojo1.Left
    End With

    strNombreEtiqueta = "lbl" & NombreColor & "3"
    Set lbl = Controls(strNombreEtiqueta)
    With lbl
        .Caption = ""
        .BackStyle = 1          ' Fondo opaco
        .BackColor = vbWhite    ' Color blanco
        ' Igualamos sus propiedades geométricas _
          con las de lblRojo1
        .Width = lblRojo1.Width
        .Height = lblRojo1.Height
        .Left = lblRojo1.Left
        .Top = lngTop
    End With
End Sub
```

Este es el resultado final:



Lógicamente sólo será operativa la barra roja, ya que no hemos definido los eventos para el resto de las barras.

Esta tarea es ya muy sencilla; prácticamente “copiar y pegar”.

Aunque no del todo.

Para controlar si tenemos pulsada la tecla sobre cada barra, crearemos tres variables boleanas y dos variable de tipo **`Byte`** adicionales.

```
Option Compare Database
Option Explicit

Private blnBotonRojoBajado As Boolean
Private blnBotonVerdeBajado As Boolean
Private blnBotonAzulBajado As Boolean
Dim bytRojo As Byte
Dim bytVerde As Byte
Dim bytAzul As Byte
Dim lngAnchoEtiquetas As Long
```

Cambiamos la variable en el código anterior, y añadimos los eventos para las otras barras

Este sería el código para las barras Verdes

```
Private Sub lblVerde1_MouseDown( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    ' Comprobamos que el botón pulsado es el izquierdo
    If Button = acLeftButton Then
        blnBotonRojoBajado = True
        ' Llamamos al gestor del evento _
                    MouseMove de lblRojo1
        lblVerde1_MouseMove Button, Shift, X, Y
    End If
End Sub

Private Sub lblVerde1_MouseUp( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    blnBotonRojoBajado = False
End Sub

Private Sub lblVerde1_MouseMove( _
                    Button As Integer, _
```

```
                        Shift As Integer, _
                        X As Single, _
                        Y As Single)
            If blnBotonRojoBajado Then
                If X <= lblRojo1.Width And X >= 0 Then
                    lblVerde2.Width = X
                 bytVerde = Round(255 * X / lngAnchoEtiquetas)
                    lblVerde1.Caption = bytVerde
                    ColoreaFormulario
                End If
            End If
    End Sub
```

Fíjese el lector que al final del procedimiento **lblVerde1_MouseMove** he añadido una llamada al procedimiento **ColoreaFormulario**.

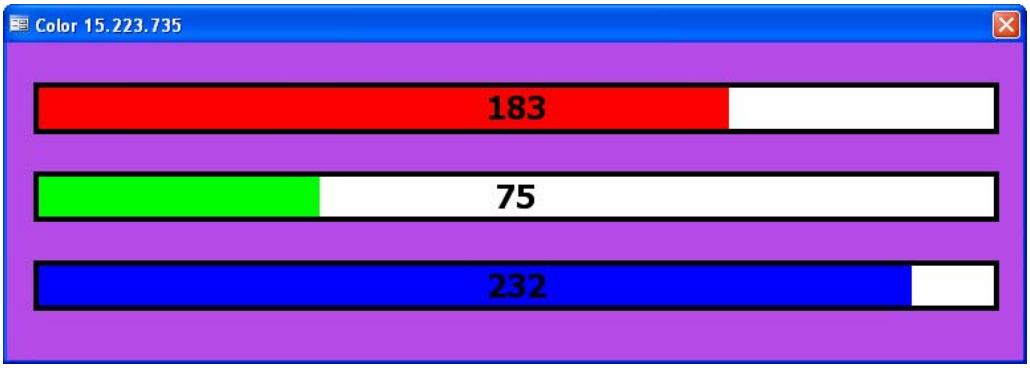
Este procedimiento será el que se encargue de colorear el formulario.

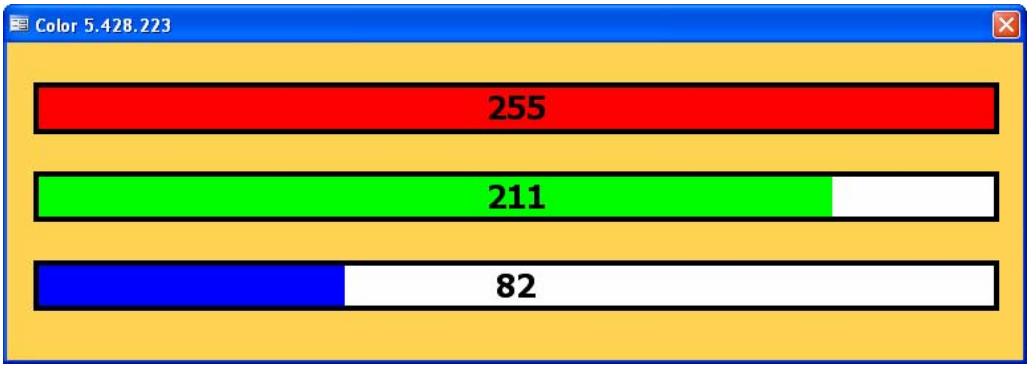
Su código es tan sencillo como este:

```
    Private Sub ColoreaFormulario()
        Dim lngColor As Long
        lngColor = RGB(bytRojo, bytVerde, bytAzul)
        Detalle.BackColor = lngColor
        Caption = "Color " & Format(lngColor, "#,##0")
    End Sub
```

Tras todo esto el formulario nos servirá para analizar la variación de los colores, en función de la proporción de cada color básico.

A continuación muestro diferentes resultados de ajustar las barras





Código final del proyecto

Aunque resulte un tanto reiterativo, aquí pongo el código completo del módulo de clase del formulario.

```
Option Compare Database
Option Explicit

Private blnBotonRojoBajado As Boolean
Private blnBotonVerdeBajado As Boolean
Private blnBotonAzulBajado As Boolean
Dim bytRojo As Byte
Dim bytVerde As Byte
Dim bytAzul As Byte
Dim lngAnchoEtiquetas As Long

Private Sub Form_Load()
    bytRojo = 128
    bytVerde = 128
    bytAzul = 128
    FormateaEtiquetas "Rojo"
    FormateaEtiquetas "Verde"
    FormateaEtiquetas "Azul"
    FormateaFormulario
    lngAnchoEtiquetas = lblRojo1.Width
    ColoreaFormulario
End Sub

Private Sub lblRojo1_MouseDown( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    ' Comprobamos que el botón pulsado es el izquierdo
    If Button = acLeftButton Then
        blnBotonRojoBajado = True
        ' Llamamos al gestor del evento MouseMove de lblRojo1
```

```
        lblRojo1_MouseMove Button, Shift, X, Y
    End If
End Sub

Private Sub lblRojo1_MouseUp( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    blnBotonRojoBajado = False
End Sub

Private Sub lblRojo1_MouseMove( _
                    Button As Integer, _
                    Shift As Integer, _
                    X As Single, _
                    Y As Single)
        lngAnchoEtiquetas = lblRojo1.Width
        If blnBotonRojoBajado Then
            If X <= lblRojo1.Width And X >= 0 Then
                lblRojo2.Width = X
                bytRojo = Round(255 * X / lngAnchoEtiquetas)
                lblRojo1.Caption = bytRojo
                ColoreaFormulario
            End If
        End If
End Sub

Private Sub lblVerde1_MouseDown( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    ' Comprobamos que el botón pulsado es el izquierdo
    If Button = acLeftButton Then
        blnBotonRojoBajado = True
        ' Llamamos al gestor del evento MouseMove de lblRojo1
        lblVerde1_MouseMove Button, Shift, X, Y
    End If
End Sub

Private Sub lblVerde1_MouseUp( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
```

```
    blnBotonRojoBajado = False
End Sub

Private Sub lblVerde1_MouseMove( _
                    Button As Integer, _
                    Shift As Integer, _
                    X As Single, _
                    Y As Single)
        If blnBotonRojoBajado Then
            If X <= lblRojo1.Width And X >= 0 Then
                lblVerde2.Width = X
                bytVerde = Round(255 * X / lngAnchoEtiquetas)
                lblVerde1.Caption = bytVerde
                ColoreaFormulario
            End If
        End If
End Sub

Private Sub lblAzul1_MouseDown( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    ' Comprobamos que el botón pulsado es el izquierdo
    If Button = acLeftButton Then
        blnBotonAzulBajado = True
        ' Llamamos al gestor del evento MouseMove de lblAzul1
        lblAzul1_MouseMove Button, Shift, X, Y
    End If
End Sub

Private Sub lblAzul1_MouseUp( _
                Button As Integer, _
                Shift As Integer, _
                X As Single, Y As Single)
    blnBotonAzulBajado = False
End Sub

Private Sub lblAzul1_MouseMove( _
                    Button As Integer, _
                    Shift As Integer, _
                    X As Single, _
                    Y As Single)
        lngAnchoEtiquetas = lblAzul1.Width
```

```
        If blnBotonAzulBajado Then
            If X <= lblAzul1.Width And X >= 0 Then
                lblAzul2.Width = X
                bytAzul = Round(255 * X / lngAnchoEtiquetas)
                lblAzul1.Caption = bytAzul
                ColoreaFormulario
            End If
        End If
End Sub

Private Sub FormateaFormulario()
    ScrollBars = 0              ' Sin barras de desplazamiento
    RecordSelectors = False     ' Sin selector de registros
    NavigationButtons = False   ' Sin Botones de desplazamiento
    DividingLines = False       ' Sin Separador de registros
End Sub

Private Sub FormateaEtiquetas(ByVal NombreColor As String)
    Const SEPARACION As Long = 500
    Dim lngTop As Long
    Dim lngColor As Long
    Dim lbl As Label
    Dim strNombreEtiqueta As String

    Select Case NombreColor
    Case "Rojo"
        lngTop = SEPARACION
        lngColor = vbRed
    Case "Verde"
        lngTop = 3 * SEPARACION
        lngColor = vbGreen
    Case "Azul"
        lngTop = 5 * SEPARACION
        lngColor = vbBlue
    End Select

    strNombreEtiqueta = "lbl" & NombreColor & "1"
    Set lbl = Controls(strNombreEtiqueta)
    With lbl
        .Top = lngTop
        .Width = lblRojo1.Width
        .BorderStyle = 1        ' Borde de línea sólida
        .BorderWidth = 3        ' Anchura de 3 puntos
```

```
            .BackStyle = 0          ' Fondo transparente
            .TextAlign = 2          ' Texto centrado
            .Caption = 128          ' Texto inicial de la etiqueta
            .FontSize = 18          ' Tamaño de la fuente
            .FontBold = True        ' Texto en negrita
        End With

        strNombreEtiqueta = "lbl" & NombreColor & "2"
        Set lbl = Controls(strNombreEtiqueta)
        With lbl
            .Top = lngTop
            .Caption = ""
            .BackStyle = 1           ' Fondo opaco
            .BackColor = lngColor    ' Color para el fondo
            ' Salvo la anchura, sus propiedades geométricas _
              como las de lblRojo1
            .Width = lblRojo1.Width / 2
            .Height = lblRojo1.Height
            .Left = lblRojo1.Left
        End With

        strNombreEtiqueta = "lbl" & NombreColor & "3"
        Set lbl = Controls(strNombreEtiqueta)
        With lbl
            .Caption = ""
            .BackStyle = 1          ' Fondo opaco
            .BackColor = vbWhite    ' Color blanco
            ' Igualamos sus propiedades geométricas _
               con las de lblRojo1
            .Width = lblRojo1.Width
            .Height = lblRojo1.Height
            .Left = lblRojo1.Left
            .Top = lngTop
        End With
    End Sub

    Private Sub ColoreaFormulario()
        Dim lngColor As Long
        lngColor = RGB(bytRojo, bytVerde, bytAzul)
        Detalle.BackColor = lngColor
        Caption = "Color " & Format(lngColor, "#,##0")
    End Sub
```

Recapitulemos

En esta entrega hemos analizado las características de algunas propiedades del formulario y de sus controles

Hemos visto cómo podemos manejar una serie de eventos

Cómo podemos, con un poco de imaginación, y bastante trabajo, emular un control mezcla de una barra de progreso y otro que se suele llamar **Slider**.

Hemos vuelto a ver cómo podemos gestionar los colores con varias técnicas diferentes.

Hemos repasado la asignación dinámica de controles a variables, creando instancias de los mismos, utilizando para ello la colección **Controls** del formulario..

Nos hemos construido un formulario que puede resultarnos “divertido”, e incluso útil cuando queramos manejar colores.

Sugerencia:

En una entrega anterior creamos la clase **clsColor** que permitía controlar los colores, e incluso generaba eventos cuando éste cambiaba.

En el código anterior la llamada al procedimiento **`ColoreaFormulario`** se realiza desde el gestor del evento **`MouseMove`** de las etiquetas superiores.

Quedaría mucho más “elegante” gestionar los colores mediante una instancia de la clase clsColor, y que sea el evento que se desencadena desde la clase, cuando se cambia su color, el que se encargue de dibujar el color del formulario y el **`Caption`** de las etiquetas superiores.

Nota:

En la entrega anterior, la 22, indicaba que en esta entrega se iba a desarrollar un formulario para jugar al bingo.

A la hora de redactar esta entrega he considerado que sería preciso, antes de abordar un proyecto de esas características, profundizar un poco en el manejo de las propiedades del formulario y los controles utilizando el código VBA.

Me comprometo a que en la próxima entrega desarrollaremos el proyecto.

Y espero que esa entrega se realice en breve.