

Bloque 2. Visual Basic .NET
UT 3. Confección de interfaces de usuario y creación de componentes visuales

TEMA 10

Interfaz gráfica. Controles

1. Introducción

Visual Basic .NET incorpora muchos controles disponibles para construir nuestros programas. Cada uno de estos controles tiene una función determinada: cuadros de texto, desplegables, etiquetas, gráficos,... Debemos familiarizarnos con estos controles incorporados porque son la base para construir las interfaces. Deberemos elegir los controles adecuados para facilitar la tarea de mostrar y recoger datos en los formularios.

Entre los controles más importantes tenemos:

Control	Función	Descripción
Label	Información	Muestra un texto al usuario. No puede editarse
LinkLabel	Información	Muestra un texto pero con capacidad de hipervínculo
TextBox	Edición de textos	Muestra un cuadro de texto editable por el usuario
RichTextBox	Edición de textos	Muestra texto en "formato de texto enriquecido" (RTF) con muchas posibilidades de formato.
ComboBox	Listas de selección	Muestra una lista desplegable de elementos
DomainUpDown	Listas de selección	Muestra una lista de elementos de texto que el usuario puede desplazar arriba-abajo
NumericUpDown	Listas de selección	Muestra una lista de elementos numéricos que el usuario puede desplazar arriba-abajo
ListBox	Listas de selección	Muestra una lista de elementos
ListView	Listas de selección	Muestra texto en formato de sólo texto, texto con iconos pequeños, con iconos grandes o informe
TreeView	Listas de selección	Muestra información jerárquica
PictureBox	Gráficos	Muestra gráficos de varios formatos
CheckBox	Asignar valores	Presenta opciones no excluyentes, toman individualmente valores si/no
CheckedListBox	Asignar valores	Lista de elementos con un Chekbox como elemento

RadioButton	Asignar valores	Presenta un grupo de opciones excluyentes entre si
TrackBar	Asignar valores	Muestra una escala donde el usuario puede seleccionar un valor
DateTimePicker	Asignar fechas	Muestra un calendario gráfico para seleccionar una fecha
Button	Control de comando	Inicia, detiene o interrumpe un proceso
Panel	Controles de grupo	Grupo de controles desplazable
GroupBox	Controles de grupo	Grupo de controles
TabControl	Controles de grupo	Proporciona páginas de fichas para agrupar controles

Algunas de las propiedades de estos controles son comunes, como el nombre, que indica el nombre del control. Por esto vamos a ver una lista de las propiedades comunes más importantes, así no las volveremos a ver más adelante cuando tratemos individualmente cada control y nos podremos centrar en sus propiedades específicas.

Propiedad	Descripción
(DataBindings)	Esta colección almacena los enlaces del control a los orígenes de datos (bases de datos)
(Name)	Nombre utilizado para identificar el control u objeto
AccessibleDescription	Descripción que se proporciona a clientes con accesibilidad
AccessibleRole	Función que se proporciona a clientes con accesibilidad
Anchor	Delimitador del control. Definen a qué bordes del contenedor está enlazado el control. Cuando un control está delimitado con relación a un borde, la distancia con el borde más cercano del control y el borde del control permanece constante.
CausesValidation	Indica si el control causa y genera eventos de validación
ContextMenu	Menú contextual que se activa cuando el usuario hace clic con el botón derecho de ratón (visto en los menús)
Dock	Ubicación de acoplamiento del control, con indicación de los bordes que están acoplados al control
Enabled	Indica si el control está habilitado
Location	Posición de la esquina superior izquierda del control respecto del contenedor
Locked	Determina si se puede cambiar o mover el tamaño del control
Modifiers	Nivel de visibilidad del objeto
Size	Tamaño del control en píxeles
TabIndex	Indica el orden de tabulación de dicho control
TabStop	Indica si el usuario puede usar la tecla TAB para poner el foco en dicho control.
Visible	Determina si el control estará visible u oculto

2. Controles.

2.1. Etiqueta: Label

A **Label** Es el control más sencillo que tenemos disponible, simplemente muestra un texto en nuestro formulario y no puede ser modificado por el usuario. Puedes utilizar etiquetas para añadir descripciones a otros controles para identificarlos por ejemplo, en los cuadros de texto.

Las etiquetas no pueden recibir el enfoque. Propiedades más importantes:

Propiedad	Descripción
AllowDrop	Determina si el control puede recibir avisos de "drag-drop", es decir, de arrastrar y soltar
AutoSize	Permite el redimensionado automático de la fuente de un texto
BackColor	Color de fondo.
BorderStyle	Determina si tiene un borde visible
Cursor	Determina el tipo de cursor gráfico que debe aparecer cuando se pase el ratón por encima: <i>mouseover</i>
Font	Tipo de letra utilizado
ForeColor	Color utilizado para el texto y gráficos en la etiqueta
Image	Imagen de fondo de la etiqueta
ImageAlign	Alineación de la imagen
ImageIndex	Índice de la imagen si se utiliza una lista de imágenes
ImageList	Lista de imágenes utilizada para obtener gráficos para las etiquetas
TabIndex	Índice que indica el orden de tabulación
Text	Texto de la etiqueta
TextAlign	Determina la alineación del texto en la etiqueta

Para establecer un texto en tiempo de diseño utiliza la propiedad Text. Y lo mismo en tiempo de ejecución, asigna a la propiedad Text el texto que quieras mostrar.


```
lblStatus.Text = "Buscando servidor..."
```

La propiedad AutoSize te ayuda a ajustar el tamaño del label según el texto tanto en tiempo de diseño como de ejecución de forma dinámica.

```
lblStatus.AutoSize = True
```

Otra propiedad, BorderStyle, permite definir un rectángulo normal o con efecto tridimensional alrededor de la etiqueta.

2.2. Etiqueta de enlace: LinkLabel

 **LinkLabel** Una de las deficiencias que tenían las versiones anteriores de Visual Basic es que no tenían un simple control Label que funcionase como un enlace de Internet o hipervínculo. Esto tan simple está solucionado con este control, es prácticamente igual que el anterior pero que atiende también al evento de enlace de Internet incorporando el mismo aspecto que cualquier hipervínculo. En ocasiones en pantallas de información, o en la típica "Acerca de..." que describe el autor y detalles del programa se suele introducir un hipervínculo a la web de la empresa o del autor. Veamos ahora las propiedades específicas de este control (además de las del Label)

Propiedad	Descripción
ActiveLinkColor	Determina el color del hipervínculo cuando el usuario hace clic en él.
DisabledLinkColor	Determina el color del hipervínculo cuando está deshabilitado
LinkBehavior	Determina el comportamiento del subrayado de un hipervínculo
LinkColor	Color del hipervínculo
LinkVisited	Determina si el hipervínculo debe procesarse como visitado

Las propiedades ActiveLinkColor, DisabledLinkColor, LinkColor y VisitedLinkColor determinan el color del hipervínculo. Por ejemplo, cuando se hace clic en el enlace puedes cambiar su color para indicar que se ha visitado

La propiedad LinkColor indica el color original, es decir, cuando no se ha hecho clic todavía sobre él. Cuando se hace clic ActiveLinkColor determina su color. La propiedad DisabledLinkColor determina el color del enlace cuando está deshabilitado. Por último la propiedad VisitedLinkColor determina el color del enlace una vez visitado. Para cambiar su color en tiempo de ejecución:

```
LinkWeb.ActiveLinkColor = Color.Red  
LinkWeb.DisabledLinkColor = Color.Blue  
LinkWeb.LinkColor = Color.Blue  
LinkWeb.VisitedLinkColor = Color.Purple
```

También puedes utilizar un valor decimal para los colores azul, verde y rojo:

```
With LinkWeb  
    .ActiveLinkColor = Color.FromArgb(255, 0, 0)  
    .DisabledLinkColor = Color.FromArgb(0, 0, 255)  
    .LinkColor = Color.FromArgb(0, 0, 255)  
    .VisitedLinkColor = Color.FromArgb(128, 0, 128)  
End With
```

Nota Para simplificar la escritura y cuando establecemos varias propiedades de un objeto podemos utilizar si recuerdas las instrucciones With... End With.

Es normal que cuando un enlace puede ser utilizado se ponga el texto subrayado, esto lo indica la propiedad LinkBehavior, que determina el comportamiento del subrayado del enlace:

- Siempre subrayado
- El vínculo sólo muestra texto subrayado cuando se sitúa el ratón sobre el texto del vínculo
- El texto del vínculo jamás aparece subrayado. El vínculo puede diferenciarse también de otros textos mediante la propiedad LinkColor del control LinkLabel.
- El comportamiento de esta configuración depende del conjunto de opciones que utilice el cuadro de diálogo Opciones de Internet en el Panel de control o Internet Explorer.

```
InkWebSite.LinkBehavior = LinkBehavior.AlwaysUnderline
```

Otra propiedad interesante es LinkVisited, que determina cuando un enlace debe marcarse como visitado. El evento clic de esta etiqueta es especialmente importante puesto que determina qué sucede cuando se hace clic en él. Utiliza el evento LinkClicked para ejecutar la acción apropiada, por ejemplo para iniciar el explorador Web:

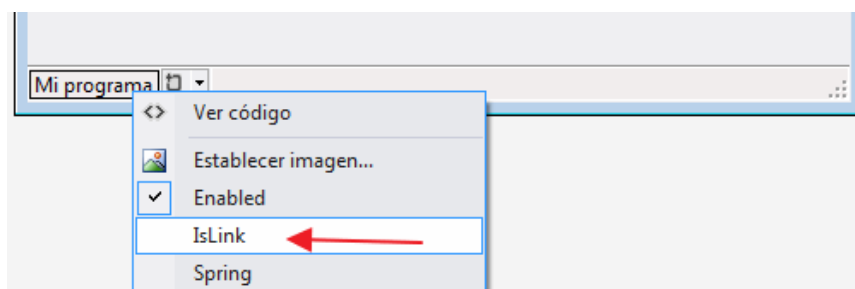
1. En el controlador del evento LinkClicked utiliza el método Process.Start para iniciar el explorador predeterminado con una URL. Para utilizar el método Process.Start necesitas añadir una referencia al "namespace" System.Diagnostics.
2. Establece la propiedad LinkVisited a True

Por ejemplo:


```
Private Sub InkWebSite_LinkClicked(ByVal Sender As Object, ByVal e As  
EventArgs)  
    'abre el enlace  
    System.Diagnostics.Process.Start("http://www.microsoft.com")  
    'marca el enlace como visitado  
    InkWebSite.LinkVisited = True  
End Sub
```

Como en muchos otros controles puede tener una imagen con la propiedad "image" o cargarla desde un contenedor "imagelist" como el que vimos el capítulo anterior.

Para las barras de controles como la de estado, había una opción específica para que los textos se comportaran como hipervínculos:



2.3. Cuadros de Texto: TextBox

 **TextBox** Los cuadros de texto son uno de los elementos más utilizados en las interfaces. Permiten mostrar textos en los que el usuario puede interactuar, modificando o eliminando su contenido. Una variante es la utilización de los cuadros para escribir contraseñas, en este caso, los caracteres que escribimos se sustituyen por un carácter especial que puede ser, por ejemplo, un '*'. Los cuadros de texto pueden mostrar y editar más de una línea, es decir, "multiline". Por defecto, los cuadros admiten hasta 2048 caracteres que es más que suficiente para nuestra interfaz pero puede aumentar hasta los 32.000 cuando se trabaja en múltiples líneas.

Aunque se utilizan normalmente para editar textos, pueden ser de sólo lectura, por ejemplo para mostrar datos que el usuario que está accediendo en ese momento no puede modificar por su nivel de acceso, por ejemplo.

Ahora veremos sus propiedades más importantes:

Propiedad	Descripción
AcceptsReturn	Obtiene o establece un valor que indica si, al presionar ENTRAR en un control TextBox multilínea, se crea una nueva línea de texto en el control o se activa el botón predeterminado del formulario
AcceptsTab	Obtiene o establece un valor que indica si al presionar la tecla TAB en un control de cuadro de texto multilínea se escribe un carácter TAB en el control en lugar de moverse el foco al siguiente control en el orden de tabulación.
AutoSize	Habilita el cambio de tamaño automático basado en el tamaño de fuente para controles de edición de una sola línea.
BackColor	Establece el color de fondo del control
BorderStyle	Establece el tipo de borde del control de cuadro de texto
CharacterCasing	Establece si el control TextBox modifica la condición de mayúscula o minúscula de los caracteres a medida que se escriben
Cursor	Establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
Font	Establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control
HideSelection	Establece un valor que indica si el texto seleccionado en el control de cuadro de texto continúa resaltado cuando el control pierde foco
Lines	Establece las líneas de texto de un control de cuadro de texto
MaxLength	Establece el número máximo de caracteres que el usuario puede escribir o pegar en el control de cuadro de texto.
MultiLine	Establece un valor que indica si es un control de cuadro de texto multilínea.
PasswordChar	Establece los caracteres utilizados para enmascarar caracteres de una contraseña en un control TextBox de una sola línea
ReadOnly	Establece un valor que indica si el texto del cuadro de texto es de sólo lectura.

ScrollBars	Establece qué barras de desplazamiento tienen que aparecer en un control TextBox multilínea
Text	Texto que muestra el cuadro.
TextAlign	Obtiene o establece cómo se alinea el texto en un control TextBox
WordWrap	Indica si un control de cuadro de texto multilínea ajusta las palabras de forma automática al principio de la línea siguiente cuando es necesario

Puedes mostrar varias líneas de texto en este control utilizando las propiedades MultiLine, WordWrap y ScrollBars. Haríamos esto:

1. Establecemos la propiedad MultiLine a True
2. Establecemos la propiedad ScrollBars al valor que queremos. Es decir, si queremos que se muestren las barras de desplazamiento verticales, las horizontales, ambas o ninguna.
3. Establecemos WordWrap al valor deseado para que ajuste las palabras de forma automática

Por ejemplo, desde el código sería:

```
With txtTextBox
    'Establece a True la propiedad de multilínea
    .Multiline = True
    'Activa las barras de desplazamiento
    .ScrollBars = ScrollBars.Both
    'Activa el ajuste de palabras
    .WordWrap = True
End With
```

Nota: Los cuadros de texto leen datos de tipo Texto. Por lo tanto, luego debo convertirlos al tipo adecuado. Por ejemplo, si estoy leyendo un valor que es la edad de una persona, tendré que convertir el valor de texto a numérico... (Cint, por ejemplo).

Puedes crear cuadros de texto de sólo lectura utilizando la propiedad ReadOnly. Si las activas, el comando "Copiar" estará disponible pero no el "Cortar" o "Pegar".

Con las propiedades MaxLength y PasswordChar podemos crear un campo de contraseña. Con la segunda propiedad mostramos el carácter que le indiquemos y con la primera el tamaño máximo del cuadro de texto, o en este caso, la contraseña:

```
With txtTextBox
    'Indica que el caracter que va a escribir es un asterisco
    .PasswordChar = "*"
    'Como máximo 10 caracteres
    .MaxLength = 10
End With
```

El problema más grande que podemos encontrarnos es cuando queramos escribir unas comillas. Por ejemplo, quiero asignar esta frase:

Jose Mª es un "manta"

Normalmente las asignaciones de cadena de caracteres las hacemos con ese mismo símbolo...

```
cuadro.Text = "Hola Jose"
```

Entonces... ¿cómo debo escribir las comillas? Muy sencillo tenemos varias formas aunque un poco liosas al principio:

- Con un juego de comillas adicional:

```
cuadro.Text = "Jose Mª es un "manta" "
```

(es decir con dos comillas a cada lado)


- Utilizando el carácter ASCII (34) de la comilla y concatenando:

```
cuadro.Text = "Jose Mª es un " & Chr(34) & "manta" & Chr(34)
```

- Definiendo una constante para el carácter de comillas

```
Const marcacomilla = """"  
cuadro.Text = "Jose Mª es un " & marcacomilla & "manta" & marcacomilla
```

2.4. Botón: Button

 **Button** Este control realiza una acción cuando el usuario hace clic sobre él o lo pulsa. Como ya hemos visto, cuando se hace clic sobre él se activa el evento "Click" que es donde colocaremos el código con la acción que queremos que haga.

Los botones no permiten el evento "doble-clic", si el usuario hace doble clic llamará dos veces al evento "click" si en la segunda ocasión estaba ya disponible el botón.

Las propiedades más importantes para este control son:

Propiedad	Descripción
BackColor	Obtiene o establece el color de fondo del control.
BackgroundImage	Obtiene o establece la imagen de fondo que se muestra en el control.
Cursor	Establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
DialogResult	Obtiene o establece un valor que se devuelve al formulario principal cuando se hace clic en el botón.

FlatStyle	Obtiene o establece la apariencia de estilo plano del control de botón.
Font	Obtiene o establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control.
Image	Obtiene o establece la imagen que se muestra en un control de botón.
ImageAlign	Obtiene o establece la alineación de la imagen en el control de botón.
ImageIndex	Obtiene o establece el valor de índice de la lista de imágenes correspondiente a la imagen mostrada en el control de botón
ImageList	Obtiene o establece el objeto ImageList que contiene el objeto Image que se muestra en un control de botón.
Text	Obtiene o establece el texto asociado al control.
TextAlign	Obtiene o establece la alineación del texto en el control de botón.

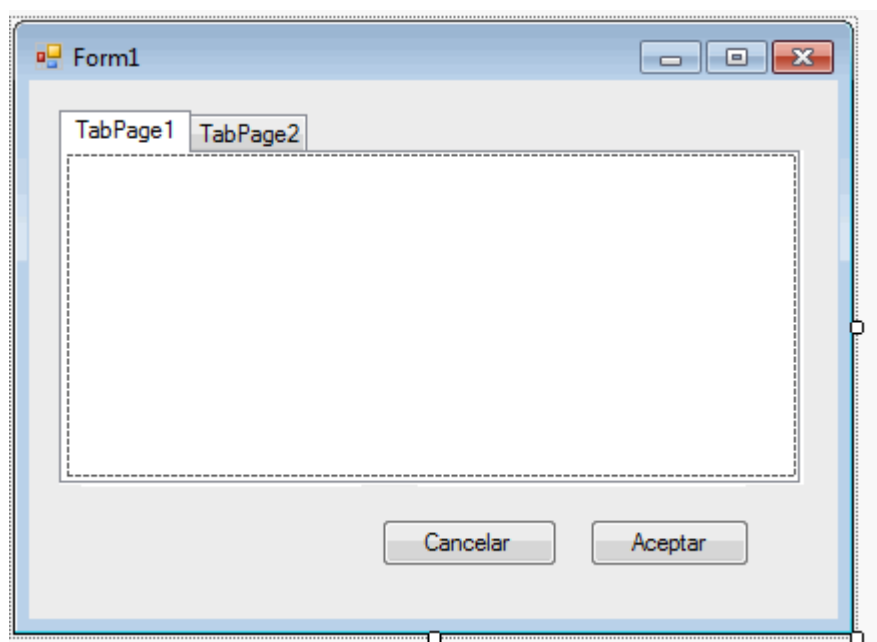
El usuario puede activar el botón de varias formas:

- Haciendo clic con el ratón
- Con el enfoque en el botón, pulsando la tecla de espacio o de "intro"
- El usuario presiona "Intro" cuando el botón es el "Aceptar" del formulario
- El usuario presiona la tecla "Escape" cuando el botón es el "Cancelar" del formulario
- El usuario presiona la tecla de acceso del botón (alt + letra subrayada)

Durante el programa podemos activar el botón con:

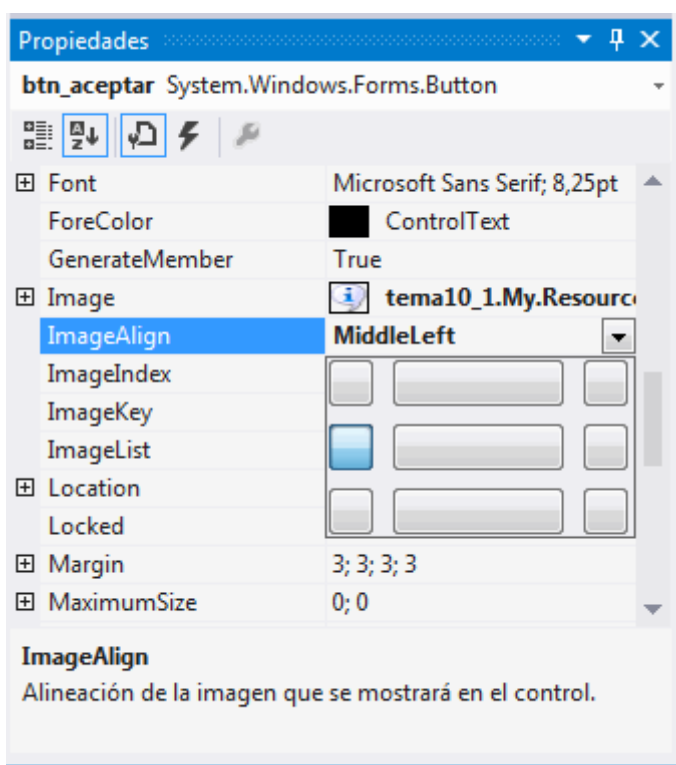
- el procedimiento del evento clic del botón
- llamando al método "PerformClick"

Podemos asignar un botón a una tecla para que se comporte como Aceptar o Cancelar. Es decir, tenemos dos botones en nuestro formulario en la parte inferior:



Y queremos que la tecla "Escape" e "Intro" hagan las funciones de los dos botones. Para esto le indicaremos en la propiedad "AcceptButton" del formulario el botón que quiero que debe simular con la tecla, en este caso "AcceptButton=boton_aceptar" y "CancelButton=boton_cancelar". Luego cuando estemos en este formulario y pulsemos la tecla de escape o enter equivaldrá a pulsar uno de esos botones. Es una funcionalidad muy habitual en los formularios de ahí que hayan tenido el detalle de ponerlo de una forma tan sencilla.

También podemos incorporar imágenes en nuestros botones mediante la propiedad "image" o con las propiedades ImageList, ImageKey e ImageIndex para indicar la imagen de un control contenedor de imágenes "ImageList":



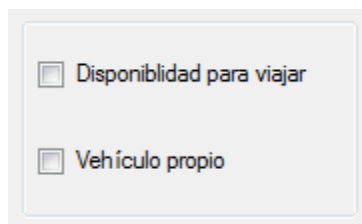
La propiedad "ImageAlign", como ves en la imagen, permite alinear la imagen dentro del botón:



2.5. Casilla de verificación: CheckBox

☒ **CheckBox** Las casillas de verificación indican un valor True/False. Se activan con el ratón y aparece una marca cuando su valor es True. Se utiliza para presentar valores independientes no

excluyentes con otros (veremos luego los botones de opción). Podemos agrupar varios controles CheckBox dentro de un GroupBox (como en este ejemplo):




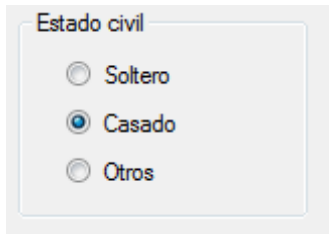
Tiene dos propiedades importantes: Checked y CheckState. La primera puedes ser True o False e indica si se cumple o no la opción. CheckState devuelve "CheckState.Checked" cuando se cumple o "CheckState.UnChecked" cuando no se cumple. Las propiedades más importantes son:

Propiedad	Descripción
Appearance	Controla la apariencia del control.
AutoCheck	Obtiene o establece un valor que indica si los valores Checked o CheckState y la apariencia de la casilla de verificación cambian automáticamente al hacer clic en ella.
BackColor	Obtiene o establece el color de fondo del control
BackgroundImage	Obtiene o establece la imagen de fondo que se muestra en el control.
CheckAlign	Obtiene o establece la alineación horizontal y vertical de una casilla de verificación en un control CheckBox.
Checked	Obtiene o establece un valor que indica si el estado de la casilla de verificación es el de activada.
Cursor	Obtiene o establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
FlatStyle	Obtiene o establece la apariencia de estilo plano del control de botón
Font	Obtiene o establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control.
Image	Obtiene o establece la imagen que se muestra en un control de botón.
ImageAlign	Obtiene o establece la alineación de la imagen en el control de botón.
ImageIndex	Obtiene o establece el valor de índice de la lista de imágenes correspondiente a la imagen mostrada en el control de botón
ImageList	Obtiene o establece el objeto ImageList que contiene el objeto Image que se muestra en un control de botón.
Text	Obtiene o establece el texto asociado al control.
TextAlign	Obtiene o establece la alineación del texto en el control de botón.

Los controles contenedores, es decir, los que pueden contener otros controles como el que hemos dicho del "GroupBox" se encuentran en el grupo *Contenedores*.

2.6. Botón de opción: RadioButton

 **RadioButton** El control RadioButton se utiliza para proporcionar una forma sencilla de elegir una opción entre varias siendo mutuamente excluyentes, es decir, sólo está activa una de ellas, por ejemplo:



Cuando se selecciona una opción la anterior queda desactivada. Para que funcionen de esta forma, es decir, que estén agrupadas y se excluyan entre ellas deben estar dentro de un GroupBox. Si quieres poner varios grupos de RadioButton en un formulario (algo normal) debes añadir otros tantos GroupBox.

El uso de estos dos controles es distinto, mientras el CheckBox agrupa opciones que puede tomar un valor cada una de ellas, en este caso sólo una de todo el grupo tendrá un valor a True. Las propiedades más importantes son:

Propiedad	Descripción
Appearance	Controla la apariencia del control.
AutoCheck	Obtiene o establece un valor que indica si los valores Checked o CheckState y la apariencia de la casilla de verificación cambian automáticamente al hacer clic en ella.
BackColor	Obtiene o establece el color de fondo del control
BackgroundImage	Obtiene o establece la imagen de fondo que se muestra en el control.
CheckAlign	Obtiene o establece la alineación horizontal y vertical de una casilla de verificación en un control CheckBox.
Checked	Obtiene o establece un valor que indica si el estado de la casilla de verificación es el de activada.
Cursor	Obtiene o establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
FlatStyle	Obtiene o establece la apariencia de estilo plano del control de botón
Font	Obtiene o establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control.
Image	Obtiene o establece la imagen que se muestra en un control de botón.
ImageAlign	Obtiene o establece la alineación de la imagen en el control de botón.
ImageIndex	Obtiene o establece el valor de índice de la lista de imágenes correspondiente a la imagen mostrada en el control de botón
ImageList	Obtiene o establece el objeto ImageList que contiene el objeto Image que se muestra en un control de botón.

Text	Obtiene o establece el texto asociado al control.
TextAlign	Obtiene o establece la alineación del texto en el control de botón.

Ejercicio: Recogida de datos

Vamos a realizar un pequeño ejemplo para ver cómo recogemos los datos seleccionados por un usuario con los controles que hemos visto hasta ahora. El programa sería:

Ejemplo de recogida de datos:

Nombre:

Apellidos:

Estado civil

☐ Soltero

☒ Casado

☐ Otros

☐ Servicio militar cumplido

☐ Vehículo propio

Cancelar Aceptar

Este ejemplo con estos valores:

Ejemplo de recogida de datos:

Nombre:

Apellidos:

Estado civil

☒ Soltero

☐ Casado

☐ Otros

☐ Servicio militar cumplido

☒ Vehículo propio

Cancelar Aceptar

Produce el siguiente resultado:

Recogida de datos

El nombre escrito es:Alfredo Ruiz.Tu estado civil es: Soltero .No has hecho el servicio militar y Si tienes vehículo propio.

Aceptar

2.7. Cuadro de texto enriquecido: RichTextBox




RichTextBox

El control de texto enriquecido o **RichTextBox** permite mostrar y leer texto de forma parecida al cuadro de texto. Pero la diferencia está en que este control permite utilizar el formato de texto RTF, donde podremos seleccionar una fuente diferente, color, tamaño de texto,... es decir, un mini editor como, por ejemplo, el de un correo electrónico.

También se pueden añadir listas, viñetas y párrafos y abrir y guardar archivos, por lo que la diferencia con el control de texto es ya bastante notable. Prácticamente todas las propiedades del TextBox están aquí presentes a las que se añadirán las referentes al formato de los textos...

Propiedad	Descripción
AcceptsTab	Obtiene o establece un valor que indica si al presionar la tecla TAB en un control de cuadro de texto multilínea se escribe un carácter TAB en el control en lugar de moverse el foco al siguiente control en el orden de tabulación
AutoSize	Obtiene o establece un valor que indica si el tamaño de RichTextBox se ajusta automáticamente cuando se cambia la fuente asignada al control.
AutoWordSelection	Obtiene o establece un valor que indica si la selección automática de palabras está habilitada
BackColor	Obtiene o establece el color de fondo del control
BorderStyle	Obtiene o establece el tipo de borde del control de cuadro de texto.
BulletIndent	Obtiene o establece la sangría que se utiliza en el control RichTextBox cuando se aplica el estilo de viñeta al texto.
Cursor	Obtiene o establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
DetectURLs	Obtiene o establece un valor que indica si RichTextBox debe aplicar formato a un localizador de recursos universal (URL) automáticamente cuando se escriba en el control.
Font	Obtiene o establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control.
HideSelection	Obtiene o establece un valor que indica si el texto seleccionado en el control de cuadro de texto continúa resaltado cuando el control pierde foco.
Lines	Obtiene o establece las líneas de texto de un control de cuadro de texto.
MaxLength	Obtiene o establece el número máximo de caracteres que el usuario puede escribir o pegar en el control RichTextBox
MultiLine	Controla si puede admitir más de una línea.
RightMargin	Obtiene o establece el tamaño de una sola línea de texto del control
ScrollBars	Obtiene o establece el tipo de barras de desplazamiento que se muestran en el control RichTextBox.
Text	Obtiene o establece el texto asociado al control.
WordWrap	Indica si un control de cuadro de texto multilínea ajusta las palabras de forma automática al principio de la línea siguiente cuando es necesario.

2.8. Cuadro de lista: ListBox

 **ListBox** El ListBox o cuadro de lista es uno de los controles más utilizados. Permite seleccionar uno o más elementos de una lista, aunque también estos elementos pueden ser casillas de verificación o CheckBox, pero en estos casos la lista de elementos suele ser reducida. Una barra vertical indica que se puede hacer scroll o desplazamiento por el control y si le ponemos la opción de "MultiColumn" podremos desplazarnos también horizontalmente. Las propiedades más importantes son:

Propiedad	Descripción
BackColor	Obtiene o establece el color de fondo del control
BorderStyle	Obtiene o establece el tipo de borde del control de cuadro de texto.
ColumnWidth	Obtiene o establece el ancho de las columnas de un control ListBox de varias columnas.
Cursor	Obtiene o establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
DataSource	Obtiene o establece el origen de datos de este objeto ListControl.
DisplayMember	Obtiene o establece una cadena que especifica la propiedad del origen de datos cuyo contenido se desea mostrar.
DrawMode	Obtiene o establece el modo de dibujo del control
Font	Obtiene o establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control.
HorizontalExtent	Obtiene o establece el ancho por el que puede desplazarse la barra de desplazamiento horizontal de un control ListBox.
HorizontalScrollBar	Obtiene o establece un valor que indica si se muestra una barra de desplazamiento horizontal en el control.
IntegralHeight	Obtiene o establece un valor que indica si el control debe cambiar de tamaño para evitar que se muestre sólo una parte de los elementos
Items	Obtiene los elementos del control ListBox.
MultiColumn	Obtiene o establece un valor que indica si el control ListBox admite varias columnas.
ScrollAlwaysVisible	Obtiene o establece un valor que indica si la barra de desplazamiento vertical se muestra siempre
SelectionMode	Obtiene o establece el método en el que se seleccionan los elementos del control ListBox.
Sorted	Obtiene o establece un valor que indica si los elementos del control ListBox se ordenan alfabéticamente.
UseTabStops	Obtiene o establece un valor que indica si el control ListBox puede reconocer y expandir los caracteres de tabulación al dibujar sus cadenas
ValueMember	Obtiene o establece una cadena que especifica la propiedad del origen de datos a partir de la cual se va a dibujar el valor

Como se trata de una lista de elementos será una matriz de una dimensión donde un índice me dirá en qué elemento estoy o podré moverme al elemento número x. Si a esto le añadimos un par de métodos para eliminar o añadir elementos a la lista, tenemos ya todo lo necesario para trabajar con los cuadros de lista.

La propiedad *SelectionMode* determina la selección simultánea de varios elementos. Si está activado como "SelectionMode.MultiSimple" el usuario puede seleccionar más un elemento simplemente haciendo clic en los elementos de la lista. Si está establecida como "SelectionMode.MultiExtended" el usuario puede seleccionar más de un elemento manteniendo pulsada la tecla de "mayúsculas" o la de "control", como en las listas normales de Windows.

Como en otros controles la propiedad *SelectedIndex* almacena el índice del elemento seleccionado (recuerda lo de la matriz). Si es una selección múltiple esta propiedad contiene el índice del primer elemento seleccionado.

Para recuperar una colección que contenga todos los elementos seleccionados en un control **ListBox** de selección múltiple, se utiliza la propiedad *SelectedItems* o si lo que quieres es la lista de los índices utilizaremos *SelectedIndex* para obtener todos los índices seleccionados en un **ListBox** de selección múltiple.

Nota: Recuerda que al igual que una matriz el primer elemento es el 0. Por lo tanto, si ponemos *SelectedIndex=0* estamos seleccionando el primer elemento.

Como los elementos que tenemos en un cuadro de lista (esto nos sirve también para los dos próximos controles) son una colección de "Items", veamos las propiedades y métodos que tenemos para manipular esta colección:

Propiedad	Descripción
Count	Devuelve el número de elementos de la colección
Item	Obtiene o establece el elemento con ese índice
Métodos	Descripción
Add	Añade un elemento a la lista
AddRange	Añade un grupo de elementos a la lista
Clear	Borra toda la colección
Contains	Determina si un elemento pertenece a la colección
IndexOf	Devuelve el índice del elemento
Insert	Inserta un elemento en el índice especificado
Remove	Borra el elemento especificado
RemoveAt	Borra el elemento con el índice especificado
ToString	Devuelve un string que representa al objeto actual

Utilizaremos el método "Add" o "Insert" para añadir elementos a la lista. Si es una lista sin ordenación (Sorted=false) los añade al final. Con Insert los insertamos en una posición determinada, por ejemplo:

```
'Añade un elemento a la lista:  
lstEmpleados.Items.Add("Jose Rodriguez")  
'Añade un elemento en la primera posición de la lista  
lstEmpleados.Items.Insert(0, "Jose Rodriguez")
```

Para eliminar elementos utilizaremos el método "Remove", siguiendo con el ejemplo esta instrucción elimina el primer elemento del cuadro de lista:

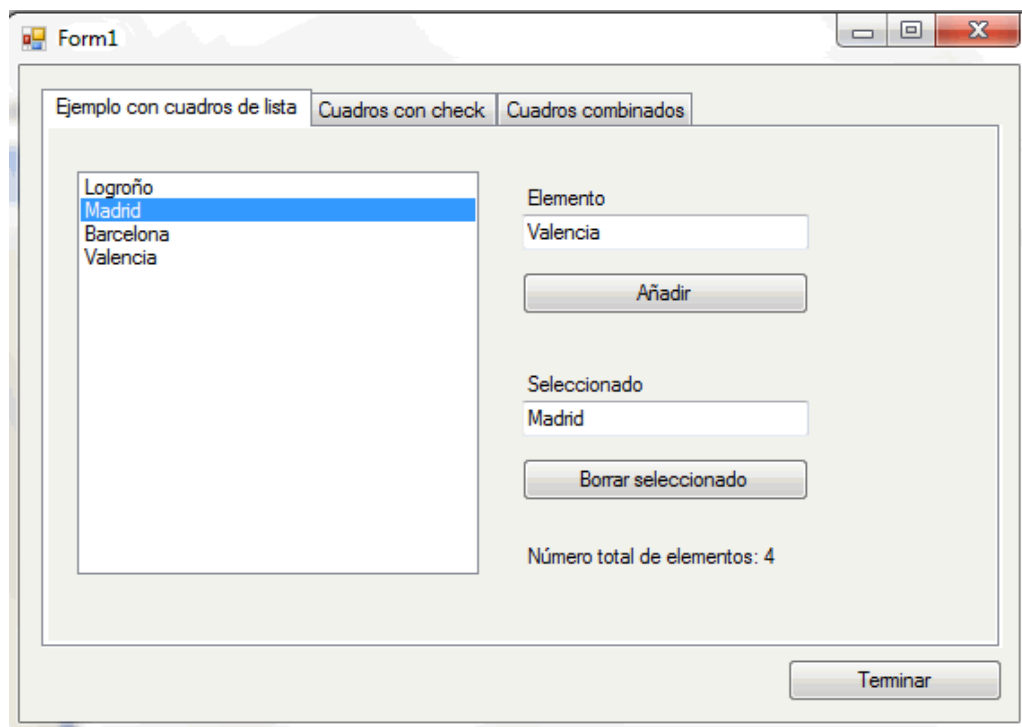
```
lstEmpleados.Items.Remove(0)
```

Para limpiar todo el cuadro de lista y así eliminar todos los elementos, utilizaremos el método "Clear"

```
'Limpia todo el cuadro  
lstEmpleados.Items.Clear()
```

Ejercicio: Cuadro de lista

En este caso vamos a hacer un práctico ejemplo donde podemos ver prácticamente todo lo que necesitamos de cuadros de lista. El resultado será este:



El programa debe añadir los elementos que se escriban en el cuadro de texto "elemento". En la parte de abajo iremos escribiendo el total de elementos de la lista. Para hacer alguna variación sólo dejaremos insertar elementos si no están ya, de esta forma no permitimos repetidos.


Por otro lado, cada vez que se haga clic sobre el cuadro de lista escribiremos su contenido en la cuadro de texto "seleccionado" y podremos eliminarlo utilizando el botón "Borrar seleccionado". Dejaremos seleccionado el elemento anterior de la lista al elemento borrado.

Notas:

1. Para añadir un elemento a la lista primero hago una comprobación de que hay algo escrito en el cuadro de texto, luego hago otra comprobación para ver si el elemento existe o no ya en la lista. Si ya existe muestro un mensaje y, en caso contrario, lo añado a la lista y actualizo la etiqueta con el número total de elementos.
2. Para mostrar el número de elementos de la lista escribimos el valor de la propiedad Count de los elementos de la lista.
3. Para detectar si se ha seleccionado un elemento utilizaremos el evento "SelectedIndexChanged" y luego la propiedad "SelectedItem" me devolverá el elemento seleccionado. Si se utiliza SelectedIndex tendríamos el índice del elemento seleccionado.
4. Eliminar un elemento: En este caso con "SelectedItem" sabremos si se ha seleccionado algún elemento. Dejaremos seleccionado el elemento anterior de la lista cuando se borre. Almaceno el actual con "SelectedIndex" que luego utilizaré para realizar la selección del anterior.

Si quisiéramos borrar todos los elementos podríamos utilizar el método Clear.

2.9. Cuadro de lista de casillas de verificación: CheckListBox

 **CheckedListBox** Una variante muy interesante del cuadro de lista es el cuadro de lista de casillas de verificación. En él podemos mostrar una lista de opciones con una casilla de verificación a la izquierda, por ejemplo:



Por ejemplo, se puede utilizar para una selección múltiple de una forma más homogénea que poniendo los controles de forma individual. Ya que se tratarían como una matriz.

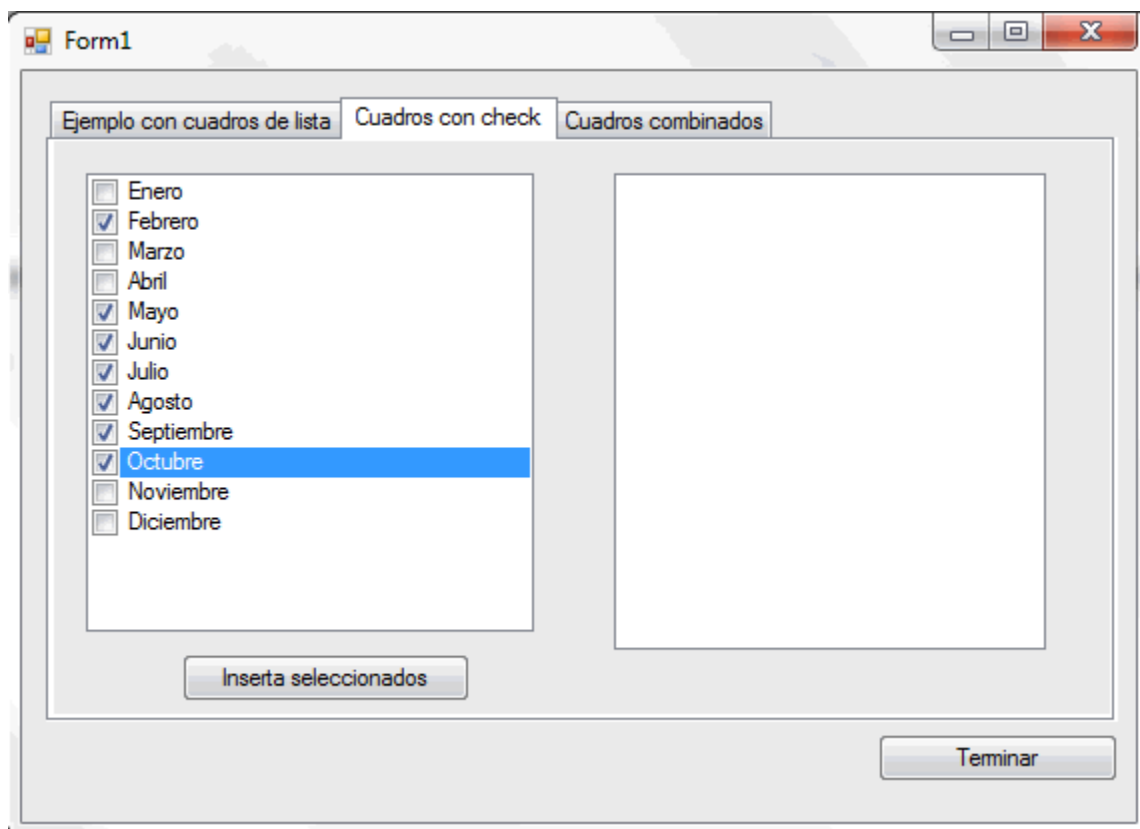
Para cambiar el aspecto de los checkbox utilizaremos la propiedad "TreeDCheckCBoxes" que le da un aspecto plano o en tres dimensiones al control. La propiedad "CheckOnClick" determina si la casilla de verificación debe cambiar de estado al hacer clic por primera vez en un elemento. Las propiedades MultiColumn y ScrollAlwaysVisible permiten hacer que se puedan poner varias columnas de controles y la forma en la que se deben mostrar las barras de desplazamiento. Por último la propiedad "Sorted" nos ordena automáticamente los elementos de este control.

Podemos añadirlos como ya hemos visto en el cuadro de lista normal en la propiedad Items.

Ejercicio: Cuadro de lista 2

Continuamos con el ejercicio anterior ampliándolo como se muestra seguidamente.

En nuestro programa añadiremos a la lista de la derecha sólo los elementos que el usuario haya marcado:




Nota: Para conseguir esto debemos saber qué elementos son los que están marcados a la izquierda. La selección se hace por doble clic a menos que lo indiquemos con la propiedad "CheckOnClick"

Este control dispone de varias propiedades para poder extraer los que están seleccionados. Por ejemplo, "CheckedItems" almacena la colección de los elementos seleccionados. Como todas las colecciones tenemos una propiedad "Count" que nos indica cuantos elementos tiene. Hay recorrer este bucle para ir añadiéndolos en la lista de la derecha. Simplemente los extraeremos indicando con un índice el elemento de la matriz: CheckedItems (x) y lo recuperamos en forma de String con el método: CheckedItem.ToString() (recuerda los métodos del cuadro de lista)

Si lo que quieres es una colección con los índices de los elementos seleccionados utilizaremos la colección "CheckedIndexCollection"

2.10. Vista de lista: Control ListView

 **ListView** El control ListView representa el objeto de la clase ListView, que muestra información matricial, tal como se muestra en un Excel o en una tabla de Word.



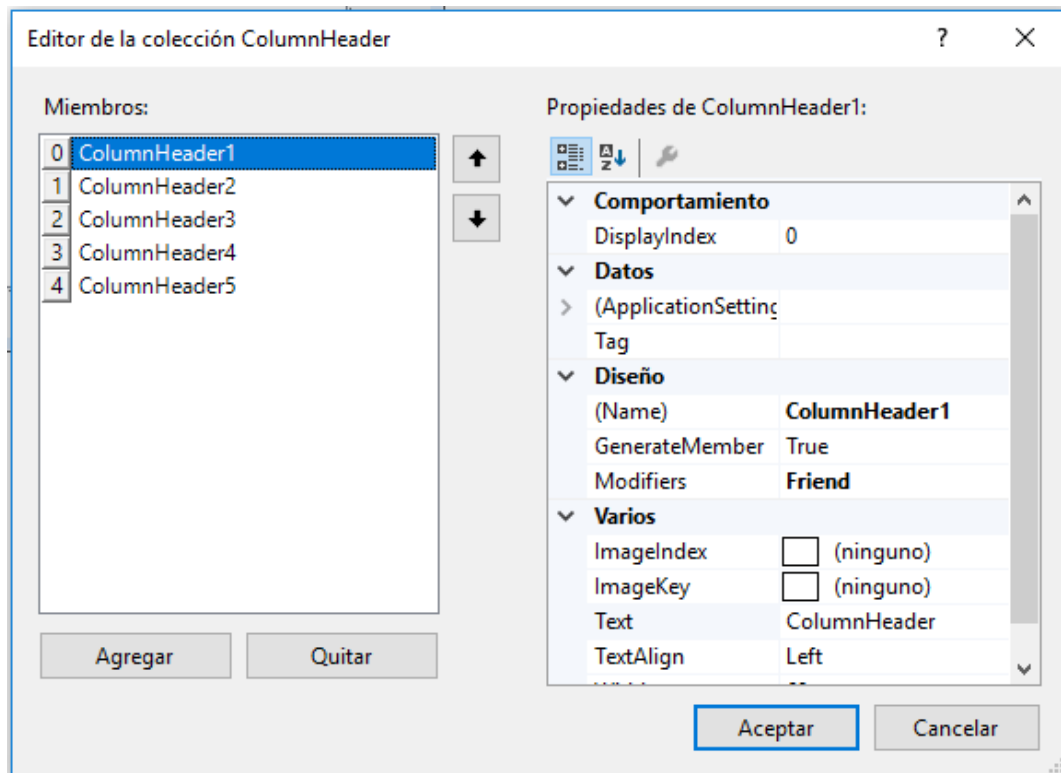
CÓDIGO	PRODUCTO	PRECIO	CANTIDAD	SUBTOTAL(€)

Las propiedades más importantes son:

Propiedad	Descripción
GridLines	Asigna una rejilla al control ListView
View	Asigna una determinada vista al objeto ListView. Con mucha frecuencia usaremos la vista <i>Details</i> (Detalles).
Columns	Permite asignar columnas al objeto ListView, de tal forma que se asigna una cabecera adecuada para el usuario.

Veamos a continuación un ejemplo de cómo implementar el ListView anterior. Este ListView permitirá presentar el código, descripción, precio, cantidad y subtotal de una venta de productos.

Seleccionaremos el objeto ListView en el formulario. Modificaremos la propiedad *View* para que tome el valor *Details*, la propiedad *GridLines* tomará el valor *True*. Finalmente, mediante la propiedad *Columns* agregaremos los textos mostrados en la cabecera.



Para agregar las cabeceras deben crearse cinco columnas. Para ello haremos clic cinco veces sobre el botón *Agregar*; luego seleccionaremos uno a uno los miembros (columnHeader1) y en la propiedad *Text*, mostrada en la misma ventana, modificaremos la cabecera. Cuando terminemos con todas las columnas haremos clic en el botón *Aceptar*.

Se puede implementar la misma aplicación mediante el siguiente código:

```
Public Class frm_ListView
    0 referencias
    Private Sub frm_ListView_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        With lvFactura
            .View = View.Details
            .GridLines = True
            .Columns.Add("CÓDIGO", 60)
            .Columns.Add("PRODUCTO", 150)
            .Columns.Add("PRECIO", 100)
            .Columns.Add("CANTIDAD", 100)
            .Columns.Add("SUBTOTAL(€)", 120)
        End With
    End Sub
End Class
```

Para que el control ListView muestre una rejilla, se usa la sentencia `lvFactura.GridLines=True`; para visualizar en forma de columnas se usa la sentencia `lvFactura.View=View.Details`; y finalmente, para agregar los encabezados se usa la sentencia `lvFactura.Columns.Add("Título", Ancho de columna)`.

Ejercicio: Media de notas

Crea una aplicación que permita determinar el la nota media de una asignatura de un alumno dependiendo de cuatro notas, en el que cada nota tiene un peso correspondiente.

Hay que tener en cuenta los siguientes aspectos:

- El formulario debe contemplar los datos solicitados en la aplicación.
- El usuario introducirá el nombre del alumno y sus cuatro notas.
- Los pesos de las notas son 10%, 20%, 30% y 40% según el orden de entrada.
- La media de las notas resulta del valor porcentual de cada nota según el peso.
- Mostrará los resultados en un formato numérico de dos decimales.
- Los resultados se mostrarán en un control ListView.

ALUMNO	NOTA1	NOTA2	NOTA3	NOTA4	PROMEDIO
Luis	8,00	5,00	7,00	9,00	7,50

Nota: Para agregar una nueva fila al ListView deberemos declarar un objeto *fila* de la clase *ListViewItem* y que esa fila esté compuesta por cada uno de los ítems de la columna que hemos de añadir a la fila mediante *fila.SubItems.Add()*

2.11. Cuadro combinado: Combobox



ComboBox

El cuadro combinado es otra variante del cuadro de lista. Aunque más que una variante es una combinación entre un cuadro de texto y un cuadro de lista. Por un lado, aparece como un cuadro de texto pero tiene asociado una lista, este cuadro de texto tiene una flecha que al pulsarla presenta el cuadro desplegable:

Cuadro combinado sin desplegar:



Cuando escribimos en el cuadro de texto se selecciona el elemento de la lista que coincida. Si la lista está desplegada, cuando el usuario selecciona un elemento de la lista éste aparece en el cuadro de texto.

La parte de la lista del cuadro combinado es similar al anterior control de cuadro de lista. En este control tendremos que tener cuidado con lo que dejamos escribir al usuario en la parte del cuadro de texto, ya que puede escribir un texto y no corresponder con ningún elemento de los existentes en la lista. Así que una de las posibilidades de funcionamiento que tiene es no dejar escribir para seleccionar un elemento de la lista (es la forma de uso más utilizada).

Este control se utiliza mucho en los formularios, ya que los cuadros de lista habituales ocupan mucho espacio en pantalla. Los cuadros combinados ofrecen el mismo resultado pero con un espacio menor.

Los cuadros combinados tienen tres estilos que determinan su funcionamiento:

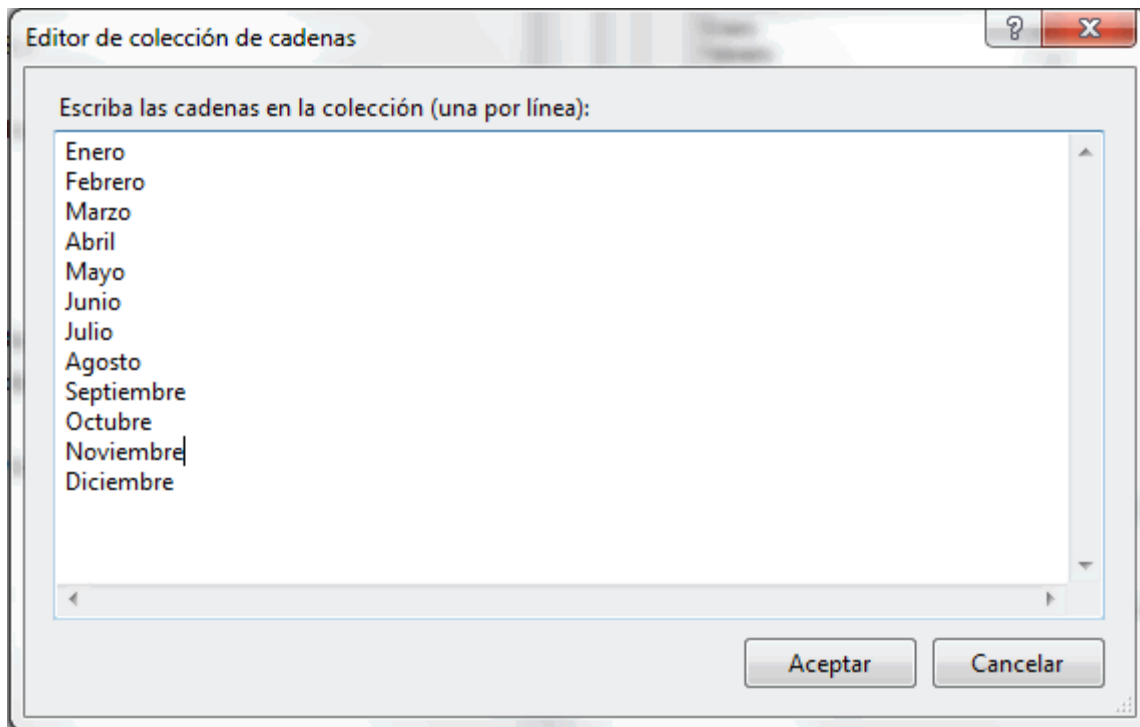
- Simple: La parte correspondiente al texto puede editarse. La parte correspondiente a la lista siempre se ve.
- DropDown: La parte correspondiente al texto puede editarse. El usuario debe hacer clic en el botón de flecha para mostrar la parte correspondiente a la lista.
- DropDownList: El usuario no puede editar directamente la parte correspondiente al texto. El usuario debe hacer clic en el botón de flecha para mostrar la parte correspondiente a la lista. (la más utilizada como hemos comentado antes)

En el primer caso el cuadro combinado tiene un cuadro de texto con un cuadro de lista siempre visible. En el segundo caso también podemos editar el cuadro de texto pero disponemos de una

flecha para que se despliegue el cuadro de lista y en el tercer caso disponemos también de una lista desplegable pero no podemos editar el cuadro de texto.

Las propiedades son muy parecidas a las del cuadro de lista, sólo hay una que nos interesa y es la anterior, la que define su forma de funcionar: "DropDownStyle"

Para añadir elementos en tiempo de diseño pulsaremos la opción Items del cuadro de propiedades y escribiremos los elementos que queramos que tenga inicialmente este control:



Desde el código utilizaremos el método "Add" o "Insert" para añadir elementos a la lista. Si es una lista sin ordenación (Sorted=false) los añade al final, con Insert los insertamos en una posición determinada, por ejemplo:

```
'Añade un nombre  
lstEmpleados.Items.Add("Jose Rodriguez")  
'Inserta un nombre al principio de la lista  
lstEmpleados.Items.Insert(0, "Jose Rodriguez")
```

Para eliminar elementos utilizaremos el método "Remove", siguiendo con el ejemplo esta instrucción elimina el primer elemento del cuadro combinado:

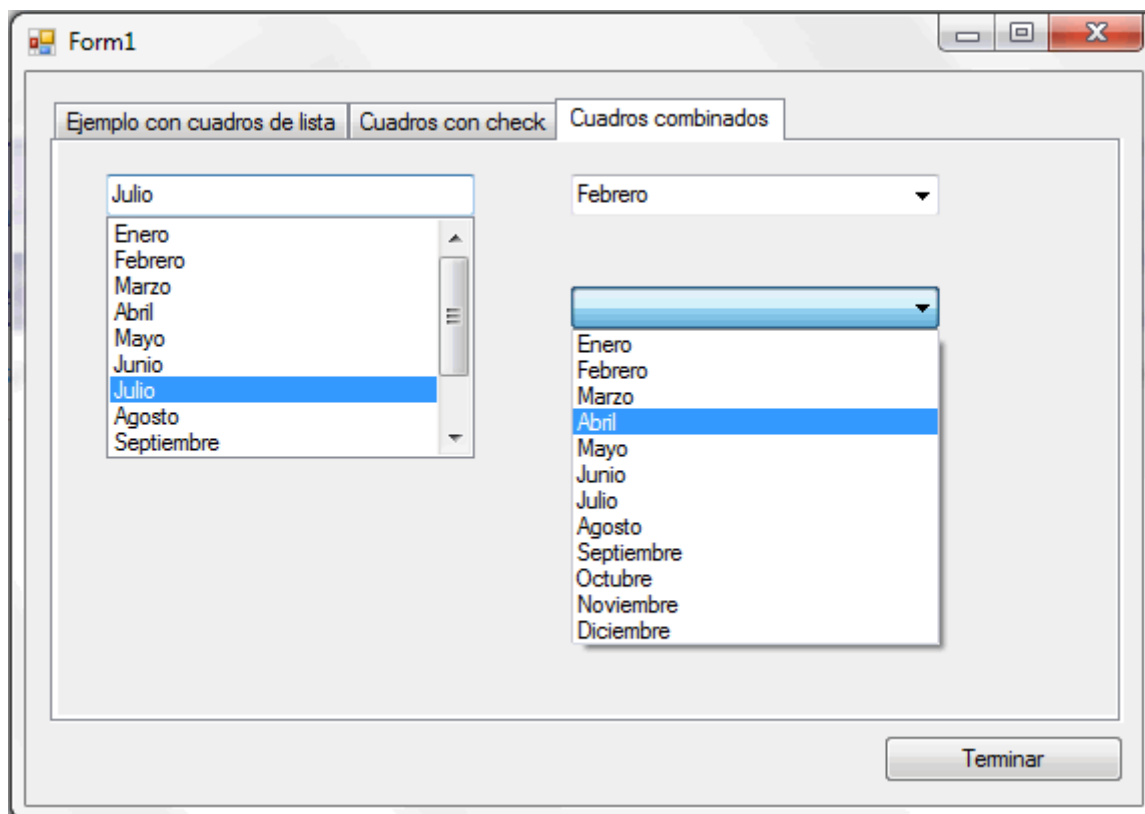
```
lstEmpleados.Items.Remove(0)
```

Para limpiar todo el cuadro combinado y así eliminar todos los elementos, utilizaremos el método "Clear"

```
'Elimina todos los elementos  
lstempleadosItems.Clear()
```


Nota: Aunque tengamos una lista de "Items" asignada al cuadro debemos establecer la propiedad "Text" con algún valor. Si no lo hacemos, al ejecutar el programa el valor que nos aparecerá es el de "ListBox1" o vacío. Para solucionarlo, basta con escribir a mano en la propiedad Text el primer elemento. Puede ser que no lo sepamos en la carga del formulario (evento Load) así que lo que podemos hacer es activar el elemento que queramos, en este caso: `Lista.SelectedIndex=0`

En el ejemplo resumen están dibujados tres controles con los tres estilos permitidos. Como las propiedades y métodos son prácticamente iguales que los cuadros de lista no lo repetiremos aquí.



El primer caso es con el estilo (propiedad `DropDownStyle`) simple. Muestra un cuadro de texto y un cuadro de lista solidario, a medida que escribimos algo en el cuadro de texto se actualiza la lista. El segundo caso tiene el valor "DropDown" en la misma propiedad. En este caso muestra un cuadro de texto y botón para desplegar la lista. El último caso es el del "DropDownList" similar al anterior pero que no permite escribir en el cuadro de texto y solo espera que el usuario seleccione un elemento de la lista.

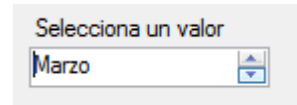
La ventaja o desventaja de los dos primeros es que el usuario puede escribir libremente un texto pero puede no corresponder con ningún elemento de la lista. El tercer caso por fuerza debe ser uno de la lista porque no deja escribir. Si utilizamos este control para asegurarnos de leer un valor correcto de la lista, con los dos primeros podemos tener problemas ya que al pulsar "Aceptar" en el formulario podemos tener un valor incorrecto, al no pertenecer a la lista. En este caso debemos comprobar que lo que ha escrito el usuario es un elemento de la lista utilizando el método que busca un elemento en la lista: `"FindStringExact"`.

2.12. Control DomainUpDown



DomainUpDown

Este control es muy útil para construir nuestras interfaces. Nos permite seleccionar un elemento pero sin desplegar una lista sino simplemente haciendo clic en las flechas arriba-abajo nos desplazamos por la lista de elementos que le hemos asignado. Por ejemplo, en el caso anterior, con los meses del año: Con lo que de una forma sencilla podemos controlar lo que el usuario debe seleccionar. Lógicamente la lista que asignemos a este control debe ser de datos que tengan una secuencia: días de la semana, meses,... es decir, que el usuario pueda encontrar de una forma sencilla. Las propiedades más importantes son:



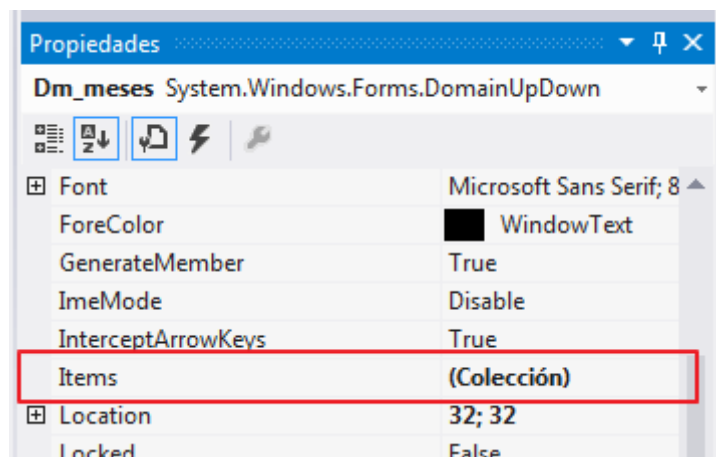
Propiedad	Descripción
BackColor	Obtiene o establece el color de fondo del control
BorderStyle	Obtiene o establece el tipo de borde del control de cuadro de texto.
Cursor	Obtiene o establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
Font	Obtiene o establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control.
InterceptArrowsKeys	Indica se las teclas de arriba/abajo aumentarán o disminuirán el valor del control
Items	Lista de los elementos que puede seleccionar el usuario
ReadOnly	Indica si el cuadro de edición es de sólo lectura.
Sorted	Obtiene o establece un valor que indica si los elementos del control se ordenan alfabéticamente.
Text	Contiene el valor seleccionado
TextAlign	Alineación del texto en el cuadro.
UpDownAlign	Indica la forma en la que el control de flechas ubica los botones de flecha arriba y abajo y su relación con su cuadro de edición
Wrap	Indica si los valores saltan al otro extremo de la lista de elementos

La propiedad Text contiene el elemento seleccionado. La propiedad Items como en otros controles contiene la lista de elementos posibles. Además de asignar elementos a esta lista en tiempo de diseño también podemos hacerlo en tiempo de ejecución mediante programación (como hemos visto en los controles anteriores).

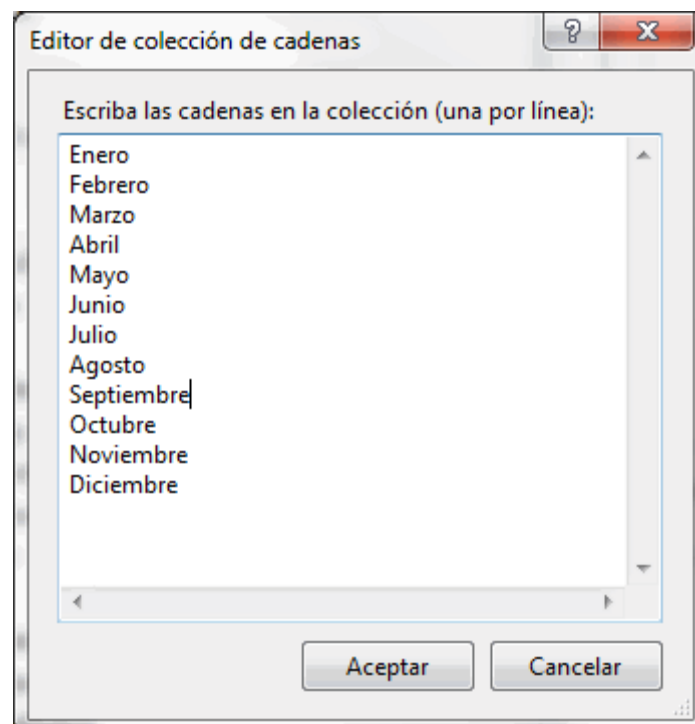
Cuando la propiedad "ReadOnly" está asignada al valor "True" no se puede escribir en el control y debemos seleccionar obligatoriamente uno de la lista. Esta es la forma más común de utilizar este control.

La propiedad "UpDownAlign" permite que los botones de subir/bajar se coloquen a la izquierda en lugar del sitio predeterminado que es la derecha. Una variante de este control es el control "NumericUpDown" que veremos a continuación.

Este control es mucho más parecido a las anteriores listas que se componen también de una colección de elementos. Así que los métodos y propiedades de Items serán los mismos. Por ejemplo, cargamos unos datos en la colección Items desde el IDE o desde una matriz como hemos visto en otras ocasiones, en este caso haciendo clic directamente en el botón "Items" del IDE:



Para poder obtener el cuadro de inserción de datos:

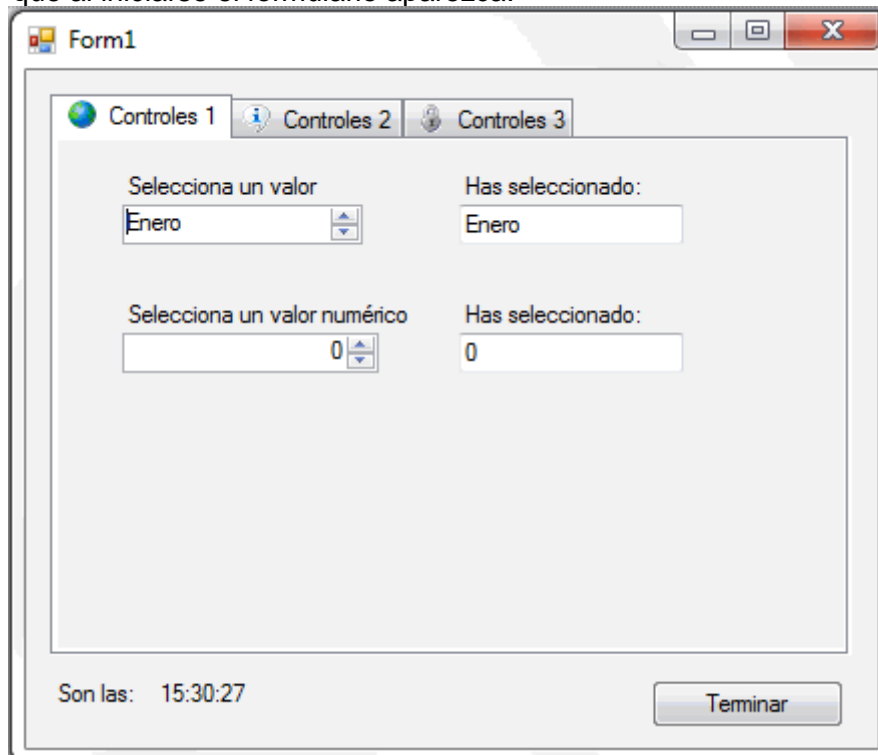


Pero todavía falta algo. Al iniciar el programa debemos indicarle el elemento que queremos que aparezca seleccionado, de lo contrario escribirá un texto genérico y podría despistar a los

usuarios. En el evento Load del formulario haremos lo siguiente: En un cuadro de texto escribiremos la selección del usuario así que también lo iniciamos con este mismo valor:

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As  
System.EventArgs) Handles Me.Load  
    'Utilizo este evento para cargar datos en el cuadro de diálogo:  
    'Por ejemplo selecciono el primer elemento del control  
    'y escribo el cuadro de texto del primer elemento seleccionado del  
    control UpDown  
    dm_Meses.SelectedIndex = 0  
    txt_mes.Text = dm_Meses.SelectedItem  
End Sub
```

Para conseguir que al iniciarse el formulario aparezca:




Ahora debemos hacer que se actualice el cuadro de la derecha cada vez que se modifique nuestro control, para esto utilizaremos el evento SelectedItemChanged:

```
Private Sub dm_Meses_SelectedItemChanged(ByVal sender As Object, ByVal e As  
System.EventArgs) Handles dm_Meses.SelectedItemChanged  
    'Actualizo el cuadro de texto de la selección  
    txt_mes.Text = dm_Meses.SelectedItem  
End Sub
```

Así que cuando esto suceda se actualizará el cuadro de texto de nuestro ejemplo. Como ves las propiedades, métodos y eventos son muy similares a las anteriores, al fin y al cabo se comporta como un cuadro combinado pero sin desplegarse la lista.

Recordemos que la propiedad Sorted (que está desactivada) nos permitiría mantener una colección de este tipo ordenada.

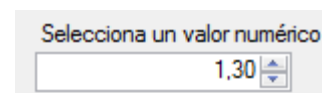
2.13. Control NumericUpDown

 **NumericUpDown** Este control permite seleccionar un valor numérico con los botones subir-bajar. En este caso la variante, o mejor dicho la ventaja, es que al ser numérico, simplemente indica una sucesión de valores. Las propiedades más interesantes son:

Propiedad	Descripción
BackColor	Obtiene o establece el color de fondo del control
BorderStyle	Obtiene o establece el tipo de borde del control de cuadro de texto.
Cursor	Obtiene o establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
DecimalPlaces	Número de decimales a mostrar
Font	Obtiene o establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control.
Hexadecimal	Indica si debe mostrar los valores en hexadecimal
Increment	El incremento/decremento que sufrirá en cada clic.
InterceptArrowsKeys	Indica si las teclas de arriba/abajo aumentarán o disminuirán el valor del control
Maximum	Valor máximo que puede tomar el control
Minimun	Valor mínimo que puede tomar el control
ReadOnly	Indica si el cuadro de edición es de sólo lectura.
TabIndex	Obtiene o establece el orden de tabulación del control en su contenedor
ThousandsSeparator	Indica si debe poner un separador de miles cada tres posiciones.
UpDownAlign	Indica la forma en la que el control del flechas ubica los botones de flecha arriba y abajo y su relación con su cuadro de edición
Value	Valor actual del control

La propiedad Value es la que almacena el valor que tiene el control, el usuario utilizará entonces las flechas arriba/abajo para incrementar/decrementar ese valor lo que indique la propiedad "Increment". Su valor predeterminado es 1.

Podemos establecer la propiedad Increment con un valor inferior a 1 y mostrar tantos decimales como le indiquemos en la propiedad "DecimalPlaces":



Para dar formato a los números también podemos ponerle un punto como separador de miles cada tres dígitos. Las propiedades "Maximun" y "Minimum" indican obviamente el intervalo de valores que comprende este control.

En este caso ya no nos encontramos con colecciones porque simplemente es un control que necesita un valor mínimo, un máximo y un incremento. Simplemente los recorrerá hasta el valor máximo. Como en los casos anteriores, para garantizar que se muestra un valor correcto en la carga del formulario iniciamos los valores en su creación:

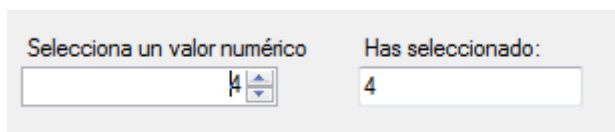
```
Me.num_numero.Value = 0
Me.txt_numero.Text = Me.num_numero.Value
```

Y para detectar si ha habido algún cambio utilizaremos el evento habitual de los cuadros de texto:


```
Private Sub num_numero_ValueChanged(sender As Object, e As EventArgs)
    'Cuando cambie su valor
    Me.txt_numero.Text = Me.num_numero.Value
End Sub
```

Nota importante Si queremos que el usuario no pueda escribir dentro de este cuadro y del anterior y así obligarle a seleccionar un valor con el cursor podemos poner la propiedad de sólo lectura "ReadOnly" a True.

Finalmente como los cuadros numéricos se suelen ajustar a la derecha. Se lo indicamos en la propiedad TextAlign:



2.14. Cuadro de imagen: PictureBox

 **PictureBox** El control PictureBox se utiliza para mostrar imágenes en varios formatos: BMP, GIF, JPG,... Sirve, por ejemplo, para mostrar un logotipo de la empresa, una foto de la base de datos o fondos de formularios. Las propiedades más interesantes de este control son:

Propiedad	Descripción
BackColor	Obtiene o establece el color de fondo del control
BackgroundImage	Imagen de fondo utilizada en el control
BorderStyle	Tipo de contorno del control
Cursor	Tipo de cursor que se muestra al pasar el ratón por encima.
Image	Imagen mostrada en el control

Podemos controlar por programación la imagen que queremos mostrar, esto será útil cuando trabajemos con bases de datos, por ejemplo, donde un control de imagen puede contener distintos archivos. Para establecer una imagen en tiempo de ejecución utilizaremos la propiedad `ImageLocation` del control y le asignaremos la ruta de la imagen:

```
logoempresa.ImageLocation="c:\imagen.jpg"
```

El objeto `Image` es muy sencillo y lo veremos cuando terminamos de ver algún detalle más de este control...

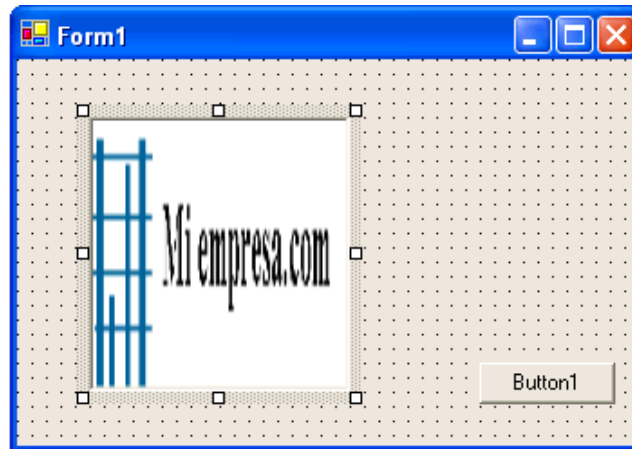
Para indicar una imagen en tiempo de diseño, obviamente buscaremos la propiedad "Image" en la página de propiedades. Tendremos la posibilidad de explorar en las carpetas para seleccionar la imagen y finalmente se mostrará en el formulario:



El control no se ha ajustado al gráfico. Si el tamaño con el que si hemos dibujado el control no es el mismo que el de la imagen, tendremos que ajustarlo con el ratón. Para este ajuste es donde entra la otra propiedad interesante de este control. La Propiedad "**SizeMode**" permite ajustar el comportamiento del control para adaptarlo a nuestro formulario, mira los valores que puede tener, te será más sencillo de ver:

- **AutoSize**. El tamaño de **PictureBox** debe ajustarse igual que el tamaño de la imagen que contiene.
- **CenterImage**. La imagen se muestra en el centro si **PictureBox** es más grande que la imagen. Si la imagen es más grande que **PictureBox**, la imagen se coloca en el centro de **PictureBox** y se recortan los bordes exteriores.
- **Normal**. La imagen se coloca en la esquina superior izquierda de **PictureBox**. La imagen se recorta si es más grande que el objeto **PictureBox** que la contiene
- **StretchImage**. La imagen situada dentro de **PictureBox** se estira o encoge para ajustarse al tamaño de **PictureBox**.

Esta última puede deformarnos la imagen así que trátala con cuidado en tiempo de ejecución ya que si el tamaño es muy desproporcionado la intentará ajustar en el control y puede quedar:



Aunque lógicamente se puede realizar el cambio del tipo de ajuste en tiempo de ejecución mediante programación:

```
logoempresa.SizeMode = PictureBoxSizeMode.AutoSize
```

Si queremos borrar su contenido en nuestro programa. Por ejemplo, si estamos consultando una base de datos y no existe una foto para esta persona podemos borrar su contenido asignándole el valor vacío "Nothing"

```
logoempresa.Image = Nothing
```

Formatos gráficos

Vamos a enumerar los formatos que debes utilizar según su contenido. Esto es válido para VB o para Internet en la elaboración de páginas web.

- BMP. Nunca se debe utilizar. Es el formato estándar de Windows y es de muy buena calidad pero al no ser comprimido no se puede utilizar.
- GIF. Es el perfecto para gráficos de pocos colores: iconos, logotipos, ... además se pueden crear "Gifs animados" que producen pequeños gráficos en movimiento (utilizados en las páginas web)
- JPG. Este es el utilizado para fotografías. Se generan al escanear fotografías o en las cámaras digitales. Es el que utilizaremos siempre para este tipo de imágenes. Tiene una compresión muy alta y muy buena calidad
- TIFF. Es el estándar para documentos escaneados. Es mejor que el JPG ya que la imagen es totalmente fiel a la original. El JPG modifica algo la imagen al comprimirlo, el TIFF no. Para que te hagas una idea una variante del TIFF es el formato de imagen que utilizan los FAX.

Clase Image

Es una clase base que proporciona funcionalidad para las clases descendentes Bitmap y Metafile. Los formatos BitMap son los que hemos visto antes y los Metafile son los generados con vectores, por ejemplo, con programa de CAD o de diseño gráfico.


Esta clase pertenece al espacio de nombres System.Drawing. En la siguiente tabla veremos las propiedades y métodos más utilizados, entre los métodos verás el que hemos utilizado antes para cargar el gráfico:

Propiedad	Descripción
FrameDimensionList	Obtiene una matriz de GUID que representa las dimensiones de los marcos de este objeto Image .
Height	Obtiene el alto de este objeto Image
HorizontalResolution	Obtiene la resolución horizontal, en píxeles por pulgada, de este objeto Image
PhysicalDimension	Obtiene el ancho y el alto de esta imagen.
PixelFormat	Obtiene el formato de píxeles de este objeto Image
RawFormat	Obtiene el formato de este objeto Image .
Size	Obtiene el ancho y el alto de esta imagen, expresado en píxeles.
VerticalResolution	Obtiene la resolución vertical, en píxeles por pulgada, de este objeto Image
Width	Obtiene el ancho de este objeto Image

Métodos

Método	Descripción
Clone	Crea una copia exacta de este objeto Image .
FromFile	Sobrecargado. Crea un objeto Image a partir del archivo especificado.
FromStream	Sobrecargado. Crea un objeto Image a partir de la secuencia de datos especificada.
GetPixelFormatSize	Devuelve la profundidad de color (número de bits por píxel) del formato de píxel especificado.
GetThumbnailImage	Devuelve una vista en miniatura de este objeto Image .
RotateFlip	Este método gira, voltea o gira y voltea el objeto Image .
Save	Sobrecargado. Guarda esta imagen en la secuencia especificada con el formato especificado
ToString	Devuelve un objeto String que representa al objeto Object actual.

2.15. Panel

 **Panel** Este control no realiza ninguna función especial de programación, su finalidad es la de agrupar controles. Es uno de los controles “contenedores” porque su misión será la de organizar nuestro formulario conteniendo distintos controles. Es un elemento visual que permite agrupar controles que traten sobre un tema. Por ejemplo, en un cuadro de diálogo donde mostramos las propiedades de un fichero, podemos agrupar en un panel todas estas: archivo, sólo lectura, sistema, oculto...


Otra de sus funciones es para comodidad del diseñador ya que al estar agrupados permite mover todo el grupo de controles sólo con arrastrar este panel. Y finalmente, puesto que todos los controles están dentro de este panel, si cambiamos su propiedad visible a true/false lógicamente podremos ocultar o mostrar todos los controles a la vez. Las propiedades más interesantes son:

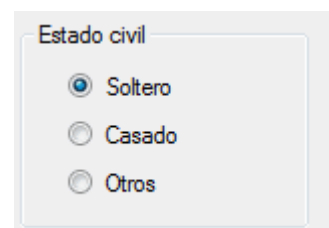
Propiedad	Descripción
AutoScroll	Obtiene o establece un valor que indica si el contenedor permitirá que el usuario se desplace a los controles situados fuera de los límites visibles.
AutoScrollMargin	Obtiene o establece el tamaño del margen de desplazamiento automático
AutoScrollMinSize	Obtiene o establece el tamaño mínimo del desplazamiento automático
DockPadding	Esta propiedad controla el borde dentro de este control para componentes acoplados.

Las propiedades se refieren sólo a su aspecto visual como el comportamiento de las barras de desplazamiento (scroll) o colores. Por ejemplo, para poner un color:

```
Panel1.BackColor=color.gray
```


2.16. GroupBox

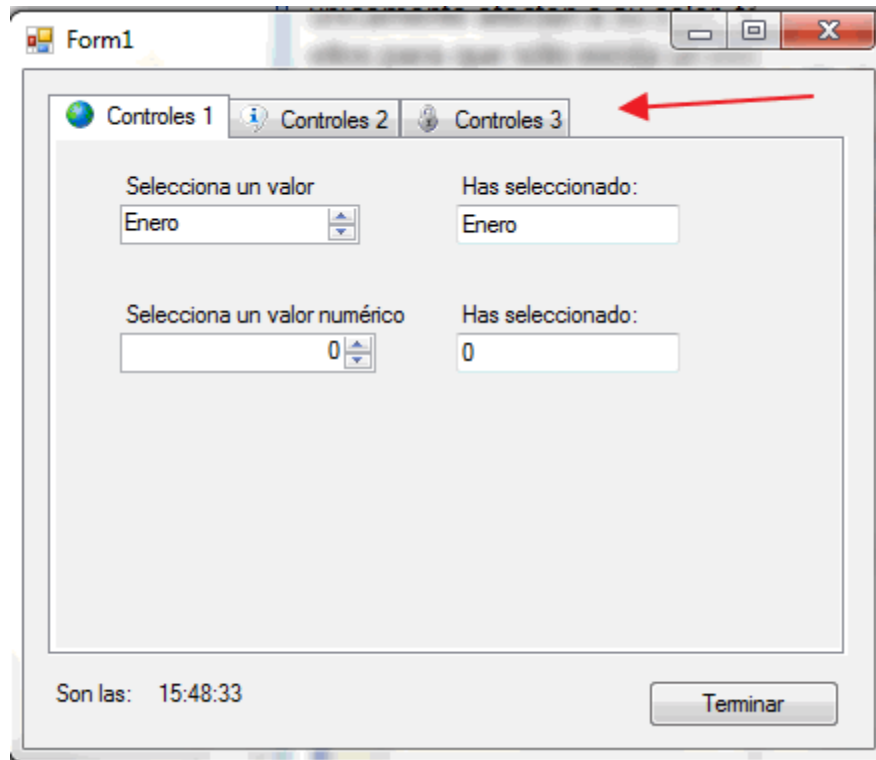
 **GroupBox** Es un control con función similar a la anterior, es decir, para agrupar controles. Pero en este caso con algunas particularidades. Por ejemplo, nunca va a tener barras de desplazamiento. Pero por otro lado permite tener un título, además su aspecto visual es distinto. Por ejemplo, este es el aspecto del que utilizamos en un ejercicio de este tema.



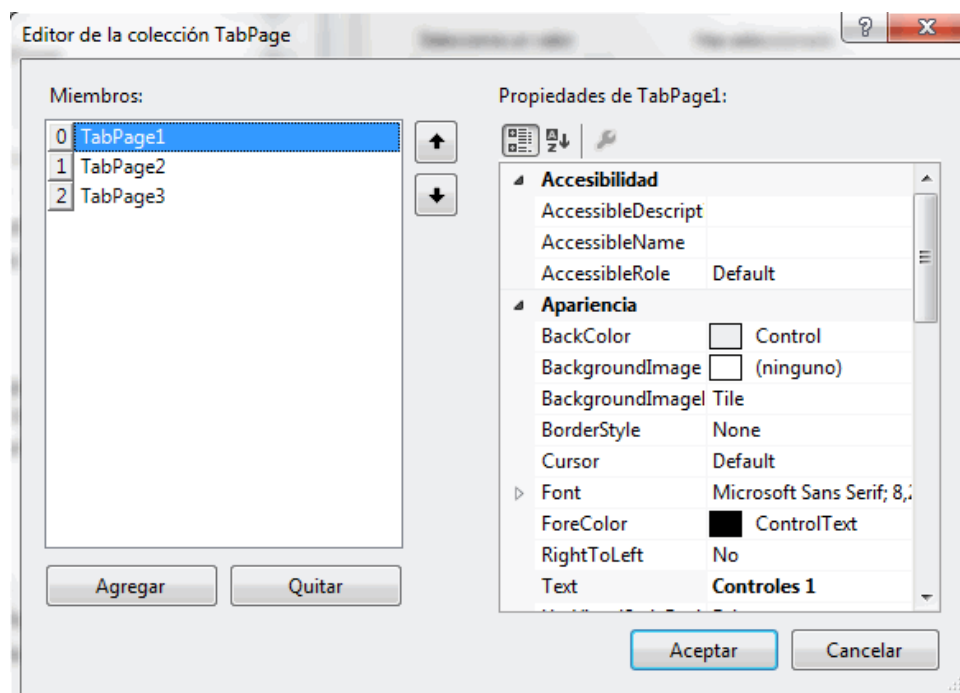
Se utiliza para agrupar controles que traten del mismo tema y así estructurar la interfaz. Las propiedades no las enumeramos en esta ocasión porque son similares a las vistas anteriormente y únicamente afectan a su color, fondo y tipo de letra del título. Además de ser un contenedor visual, es la forma de que funcionen correctamente los botones de opción ya que hace que se excluyan entre ellos para que sólo exista un elemento seleccionado.

2.17. Control de fichas: TabControl

 **TabControl** Es un control muy utilizado en los cuadros de diálogo y en las páginas de propiedades representado por una colección de fichas, permitiendo agrupar los controles por temas o por contenidos.



La propiedad más importante de este control es "TabPage" que es la que contiene la colección de solapas o páginas de propiedades. Pulsa en esta propiedad en el diseñador y te aparecerá esta página, añade tres fichas....:



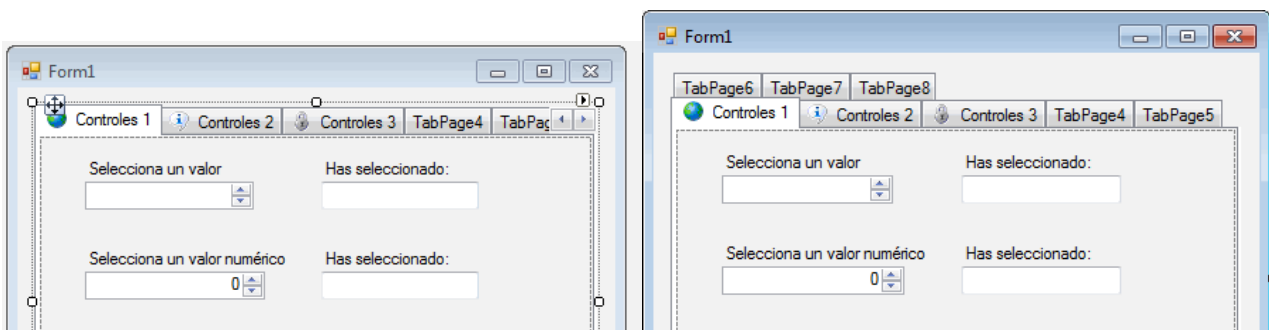
Cada una de las solapas tiene su propia colección de propiedades que afectan a su contenido de forma individual. Las propiedades más importantes son:

Propiedad	Descripción
Alignment	Obtiene o establece el área del control (por ejemplo, a lo largo de la parte superior) donde se alinean las fichas
Appearance	Obtiene o establece el aspecto visual de las fichas del control.
Font	Tipo de letra de los títulos de las fichas
HotTrack	Obtiene o establece un valor que indica si las fichas del control cambian de apariencia cuando el mouse pasa sobre ellas.
ImageList	Obtiene o establece las imágenes que se van a mostrar en las fichas del control.
ItemSize	Obtiene o establece el tamaño de las fichas del control.
MultiLine	Obtiene o establece un valor que indica si se puede mostrar más de una fila de fichas.
ShowToolsTips	Obtiene o establece un valor que indica si se muestra la información sobre herramientas de una ficha cuando el mouse pasa sobre la ficha.
SizeMode	Obtiene o establece la forma en la que se ajusta el tamaño de las fichas del control.

La apariencia (Appearance) puede ser uno de esos valores:

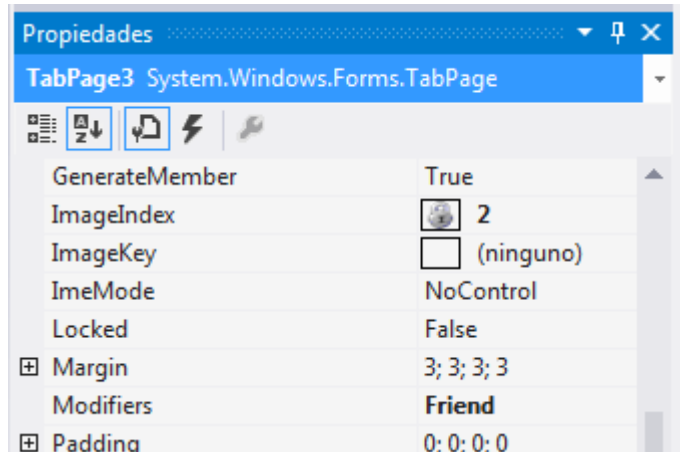
- Buttons. Las fichas tienen la apariencia de botones tridimensionales.
- FlatButtons. Las fichas tienen el aspecto de botones planos.
- Normal. Las fichas tienen la apariencia estándar de fichas.

La propiedad MultiLine determina si se pueden colocar más de una línea de fichas, esto es muy útil cuando existen muchas fichas. Este es el aspecto en tiempo de ejecución de dos controles de fichas con esta propiedad a true o false:



En un caso las fichas las podremos desplazar hacia los lados y en el otro las tendremos todas visibles. Observando interfaces se aprende mucho, ahora las analizarás cada vez que veas un formulario de los programas que utilices.

Estos controles se componen del control principal de tipo **TabControl** y de varios controles que son las **TabPages** que hemos visto antes. En concreto se trata nuevamente de una colección, esta vez de fichas llamadas **TabPage**. Si vemos en el IDE, al hacer clic sobre el control podemos ver su clase:

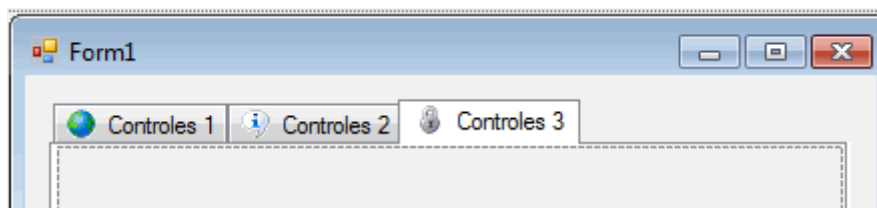


La propiedad **"HotTrack"** hace que al pasar el ratón por encima de las fichas cambie de color el título.

Practica con los gráficos y demás propiedades para añadirle iconos a las fichas mediante controles **"ImageList"** y **"ToolTipText"** para ayudar a describir las páginas a los usuarios. Por ejemplo, para poner un icono a la izquierda de la ficha:

1. Añade un control **ImageList** al formulario.
2. Añade imágenes a la lista de imágenes.
3. Establece la propiedad **ImageList** del control **TabControl** como el control **ImageList**.
4. Establece la propiedad **ImageIndex** del objeto **TabPage** como el índice de una imagen adecuada de la lista.

Para conseguir esto:



Con un solo evento podremos ver qué solapa se ha seleccionado. En nuestro ejemplo nos servirá para activar un control Timer en cuanto se pulse la tercera solapa.


```
Private Sub tab_fichas_SelectedIndexChanged(ByVal sender As Object,
    'Podemos detectar la ficha de dos formas con el índice
    'y con el nombre de la ficha de la colección
    If Me.tab_fichas.SelectedIndex = 2 Then
        'Activamos el control de tiempo
        Me.timer_progreso.Enabled = True
    End If
    'La otra forma sería
    'If Me.tab_fichas.SelectedTab.Name = "Controles 3" Then
End Sub
```

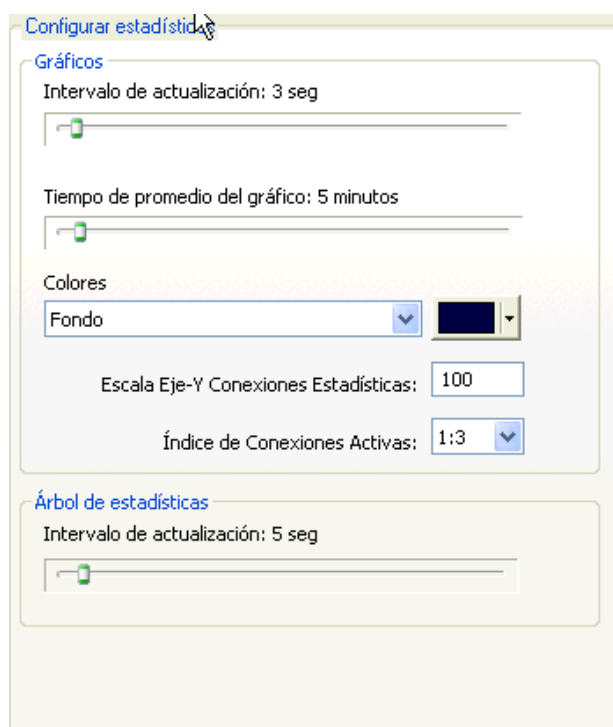
Tenemos dos formas de saber qué ficha se ha pulsado cuando se active el evento "SelectedIndexChanged"

- Utilizando la propiedad **SelectexIndex** que comenzando de 0 me indica el número de ficha que se ha hecho clic
- Utilizando el nombre de la ficha seleccionada. Como es un objeto obtendremos su nombre con **SelectedTab.Name**

Este control es muy útil porque a menudo necesitaremos saber qué controles se han seleccionado para mostrar unos cálculos...

2.18. TrackBar

 **TrackBar** El control TrackBar es uno de los típicos controles que hace que nuestras interfaces ganen mucho en personalidad. Son muy sencillos de utilizar y visualmente causan gran efecto y además al usuario le es muy sencillo e intuitivo su uso. Por ejemplo:



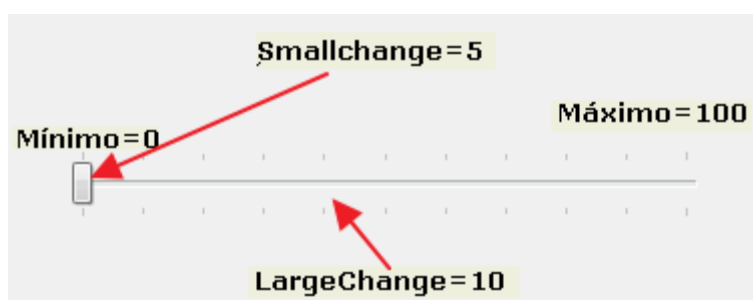
Es una forma sencilla de que el usuario seleccione un valor, en lugar de escribir un cuadro de texto o un control Arriba/Abajo utilizaremos este tipo de controles. Veamos ahora su funcionamiento.

Este control es similar a una barra de desplazamiento que nos va a permitir desplazarnos por un intervalo numérico. Por ejemplo, para un ajuste para el volumen, brillo, contraste... el usuario puede arrastrarlo con el ratón o utilizar las flechas de cursor arriba/abajo para desplazarse. Su

funcionamiento es muy sencillo y se parece bastante al control "Up/Down" que vimos antes ya que necesita un valor mínimo, otro máximo y el incremento en cada iteración del usuario. Veamos sus propiedades:

Propiedad	Descripción
AutoSize	Obtiene o establece un valor que indica si el alto o el ancho de la barra de seguimiento se va a ajustar de forma automática.
BackColor	Obtiene o establece el color de fondo del control
Cursor	Obtiene o establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
LargeChange	Obtiene o establece un valor que se suma o resta a la propiedad Value cuando el control deslizante se mueve una larga distancia.
Maximun	Obtiene o establece el límite superior del intervalo con el que trabaja este TrackBar .
Minimun	Obtiene o establece el límite inferior del intervalo con el que trabaja este control TrackBar .
Orientation	Obtiene o establece un valor que indica la orientación vertical u horizontal de la barra de seguimiento.
SmallChange	Obtiene o establece el valor que se suma o resta de la propiedad Value cuando el control deslizante se mueve una distancia pequeña.
TickFrecuency	Obtiene o establece un valor que especifica la distancia entre las marcas de paso dibujadas en el control.
Value	Obtiene o establece un valor que indica cómo mostrar las marcas de paso en la barra de seguimiento.

Tiene una pequeña diferencia y es que aquí nos encontramos con dos propiedades "SmallChange" y "LargeChange" que son nuevas pero que aparecerán en más controles. Hace referencia a si se hace clic en medio de la barra del control. Es decir, observa una barra de desplazamiento (scroll) cualquiera, por ejemplo, la que tienes ahora a la derecha en el visor de pdf. Si haces clic en las flechas arriba-abajo la pantalla se desplaza un valor pero si haces clic en medio de la barra se desplaza mucho más, es decir el "SmallChange" lo producen las flechas y el "LargeChange" cuando se pulsa en medio. Mira este gráfico a ver si lo ves mejor:

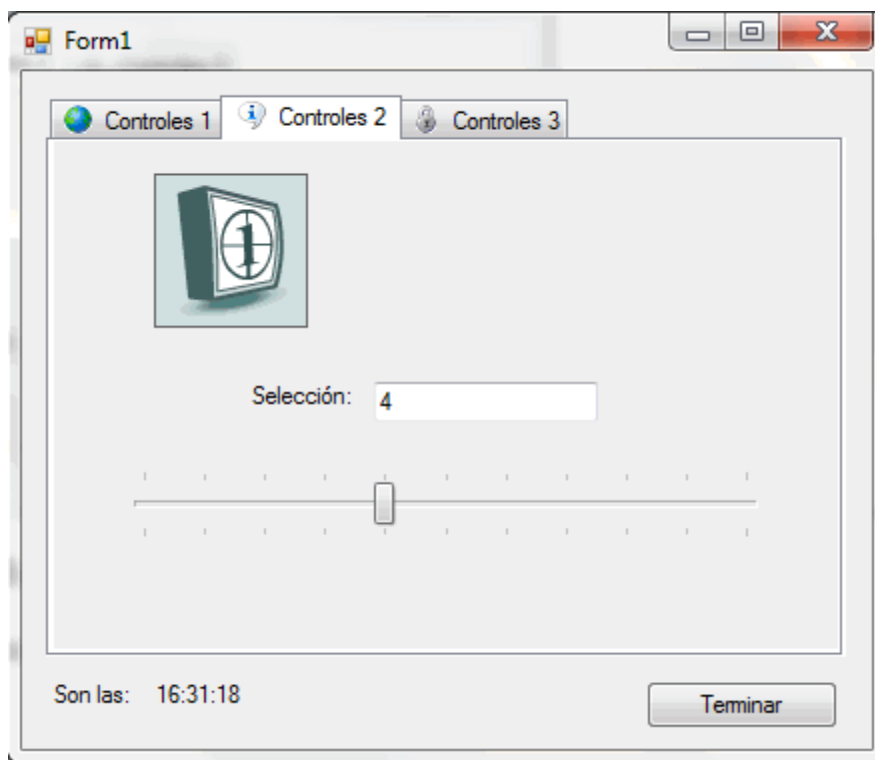


La propiedad "Value" es la que almacena el valor actual. Además si utilizamos el teclado la propiedad "Smallchange" es el incremento utilizado al utilizar las flechas. En cambio el LargeChange se utiliza con las teclas página arriba-abajo.

Las propiedades máximo y mínimo indican los valores del intervalo. La propiedad "ThickFrequency" configura el número posiciones o marcas que tiene el control. Finalmente "ThickStyle" indica si las marcas aparecen encima, en la propia barra o en los dos sitios. Practica un poco en el IDE y verás las diferencias...

En nuestro ejemplo pintaremos un control de este tipo con los valores 1 hasta 100, pondremos 5 como valor de cambio pequeño y 10 como grande. Utilizaremos también el evento habitual para detectar cambios y así modificar la posición de un control.

Además pintaremos una imagen, que como es un gif con animación aparecerá animado y modificaremos su valor de la posición para que se mueva según el valor del control:




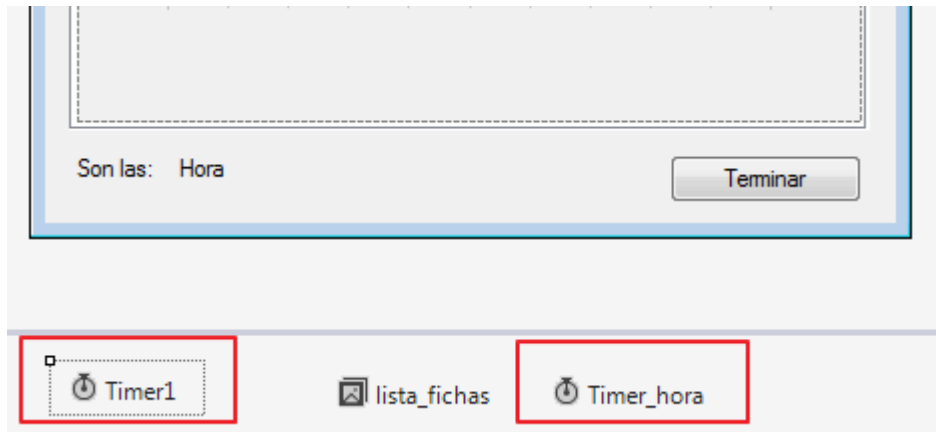
El evento para actualizar los datos como siempre:

```
Private Sub track_datos_ValueChanged(ByVal sender As Object, ByVal e As EventArgs)
    'Escribo la selección y muevo el control imagen
    Me.txt_seleccion.Text = Me.track_datos.Value
    Me.PictureBox1.Left = 10 + (10 * Me.track_datos.Value)
End Sub
```

Como vas viendo a lo largo de los ejemplos es muy importante iniciar bien los valores de los controles: comprobar sus valores de inicio al ejecutarse la aplicación, poner nombres sencillo para referirnos en el código a ellos de una forma más sencilla y prever todas las operaciones o eventos que pueda proporcionar el usuario.

2.19. Control temporizador: Timer

 **Timer** En ocasiones tendremos la necesidad de tener una especie de cronómetro que me permita realizar determinadas acciones cada cierto tiempo. Por ejemplo, que cada 5 minutos mire a ver si tenemos un nuevo mensaje de correo en nuestro buzón o compruebe un proceso. Para realizar esto disponemos del control "Timer". Como en otros casos es un control que no necesita interfaz, así que cuando añadamos uno a nuestro proyecto se presentará de esta forma:



Las propiedades más importantes son:

Propiedad	Descripción
Enabled	Habilita o deshabilita su funcionamiento o la generación de eventos
Interval	Frecuencia de tiempo en milisegundos.

Por un lado debemos habilitar el control, indicando un tiempo en milisegundos para que se active. Una vez en marcha y con un tiempo de activación, debemos ver cómo trabajar cuando se cumpla ese intervalo que será cuando dispare el evento "Tick": Se produce cuando ha transcurrido el intervalo de tiempo especificado y está activado el temporizador.

La longitud de los intervalos está definida por la propiedad **Interval**, cuyo valor se expresa en milisegundos. Cuando el componente está habilitado, el evento **Tick** se produce a cada intervalo. Aquí es donde se agrega el código que se va a ejecutar:

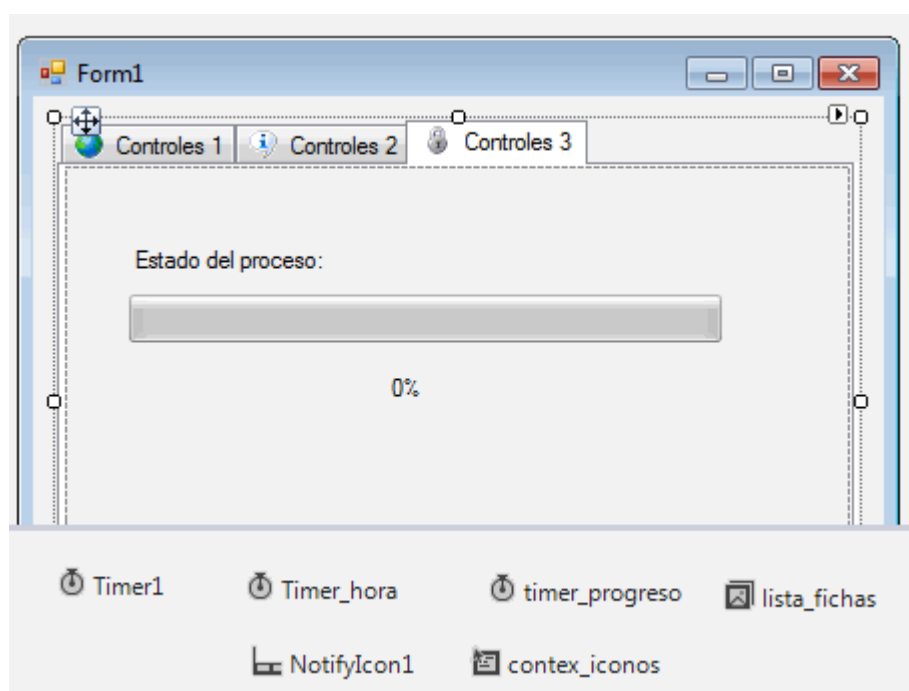
```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
End Sub
```

Ejemplo

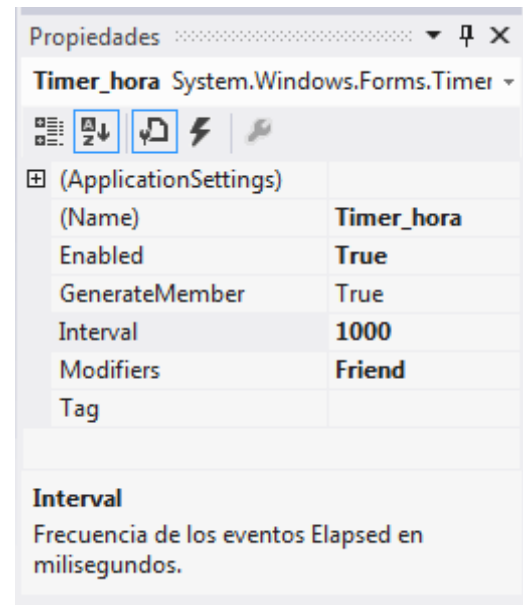
Este ejemplo efectúa un seguimiento de la hora del día con incrementos de un segundo. Utiliza un control **Button**, un control **Label** y un componente **Timer** en un formulario. La propiedad **Interval** se establece como 1.000 (igual a un segundo). En el evento **Timer**, el título de la etiqueta se establece como la hora actual. Al hacer clic en el botón, la propiedad **Enabled** se establece como **false**, lo que hace que el temporizador deje de actualizar el título de la etiqueta. El código siguiente asume un formulario con un control **Button** denominado Button1, un control **Timer** denominado Timer1 y un control **Label** denominado Label1.

```
Private Sub InitializeTimer()  
    ' Ejecuta ese procedimiento cada segundo  
    Timer1.Interval = 1000  
    ' Lo habilito.  
    Timer1.Enabled = True  
    Button1.Text = "Enabled"  
End Sub  
  
Private Sub Timer1_Tick(ByVal Sender As Object, ByVal e As  
EventArgs) Handles Timer1.Tick  
    ' Escribe la hora  
    Label1.Text = DateTime.Now  
End Sub
```

En nuestro ejemplo vamos a utilizar varios, uno de ellos para escribir la hora de una forma parecida al ejemplo anterior y otro para escribir un mensaje cada 10 segundos. En ambos casos los añadiré a mi proyecto ya que son dos temporizadores diferentes:



Para que funcionen deben estar habilitados así que cambiaré la propiedad de los dos Enabled=true. Ahora sólo me falta indicarles un intervalo en milisegundos: 1000 para escribir cada segundo la hora y 10000 para escribir cada 10 segundos un mensaje en pantalla. Para esto modificamos en el IDE la propiedad Interval:




Cuando se "disparen" los eventos provocados por el cumplimiento del intervalo haremos las acciones necesarias en nuestro ejemplo:

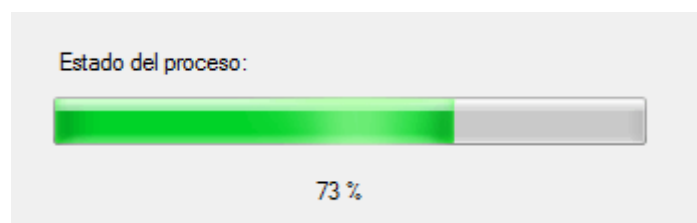
```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    'Escribo un mensaje cada 10 segundo
    MessageBox.Show("Hola, mensaje escrito cada 10 segundos", "Mi programa", MessageBoxButtons.OK, MessageBoxIcon.Information)
End Sub

Private Sub Timer_hora_Tick(sender As Object, e As EventArgs) Handles Timer_hora.Tick
    'Escribimos cada segundo la hora del sistema
    Me.lb_hora.Text = TimeString
End Sub
```

En un caso mostramos un mensaje y con el otro la hora del sistema.

2.20. Control de barra de progreso: ProgressBar

 **ProgressBar** Este control informa al usuario del estado de un proceso o mejor dicho de su progreso. Cuando estemos ejecutando un proceso largo es muy conveniente indicarle al usuario en qué estado está, por una parte, para ver si no está detenido y por otra parte para informarle de cuánto se ha cumplido:



Las propiedades son casi las mismas que en el caso anterior, la diferencia está en que este control no admite la intervención del usuario, simplemente muestra un gráfico del porcentaje completado. Las propiedades más interesantes son:


Propiedad	Descripción
Maximun	Límite superior del intervalo con el que trabaja ProgressBar
Minimun	Límite inferior del intervalo con el que trabaja ProgressBar
Value	Valor actual del control.

El funcionamiento es muy sencillo, sólo debemos indicarle el intervalo con Mínimo-Máximo y luego indicarle el valor. Por ejemplo, tenemos un control que le hemos puesto el valor 0 como mínimo y 100 como máximo, el siguiente bucle mostrará su progreso entero al ir asignándole el valor adecuado a la propiedad Valor:

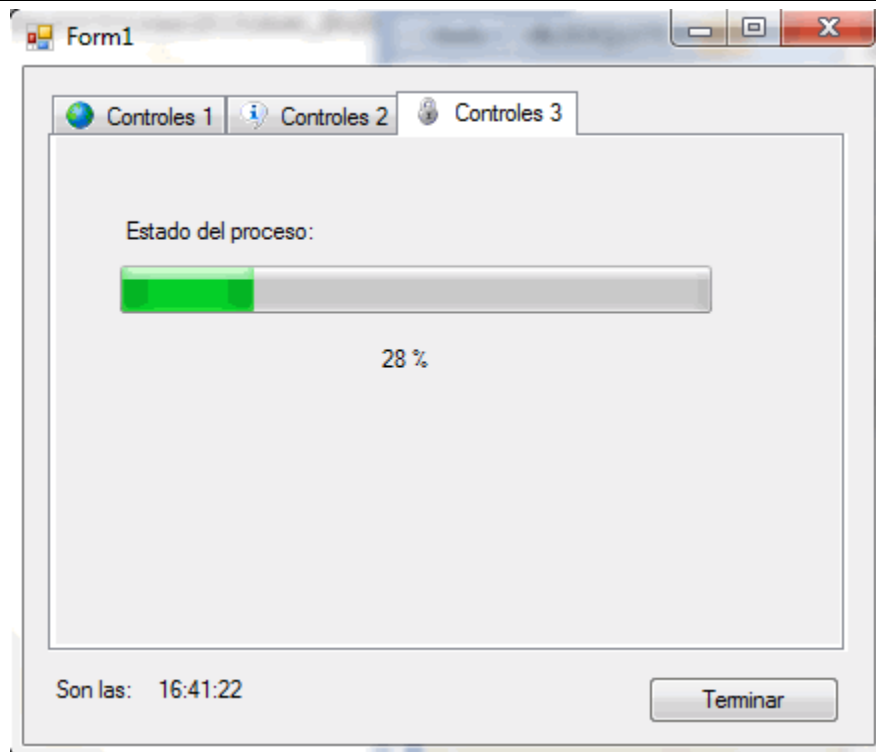
```
dim i as byte
for i=1 to 100
    progressbar1.value=i
next
```

En el ejemplo utilizaremos un control de tiempo para que cada 100 milisegundos cambie el valor de este control y simule su funcionamiento. Como variante activaremos el control de tiempo sólo cuando se hace clic sobre el control de las fichas como hemos visto antes:

```
Private Sub tab_fichas_SelectedIndexChanged(sender As Object, e As EventArgs)
    'Podemos detectar la ficha de dos formas con el índice
    'y con el nombre de la ficha de la colección
    If Me.tab_fichas.SelectedIndex = 2 Then
        'Activamos el control de tiempo
        Me.timer_progreso.Enabled = True
    End If
    'La otra forma sería
    'If Me.tab_fichas.SelectedTab.Name = "Controles 3" Then
End Sub
```



Ahora simplemente debemos actualizar el valor de la propiedad "Value" del control y de una etiqueta de información que pondremos cada vez que se cumplan los 100 ms del control de tiempo:



El código para hacer esto:

```
Private Sub timer_progreso_Tick(sender As Object, e As EventArgs) Handles timer_progreso.Tick
    'Si hemos llegado al final desactivamos el control timer para que ya no siga...
    If Me.pg_progreso.Value = Me.pg_progreso.Maximum Then
        Me.timer_progreso.Enabled = False
        Exit Sub
    End If

    'Incremento su valor
    Me.pg_progreso.Value = Me.pg_progreso.Value + 1

    'Escribo el porcentaje completado
    lb_completado.Text = (Me.pg_progreso.Value / Me.pg_progreso.Maximum) * 100 & " %"


End Sub
```

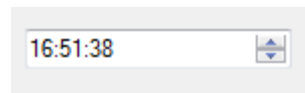
Primero incrementamos el valor del cuadro de progreso, luego actualizamos el porcentaje completado. Lógicamente como no siempre el máximo será el valor 100 lo calculo según sea el valor máximo:

$$(Me.pg_progreso.Value / Me.pg_progreso.Maximum) * 100$$

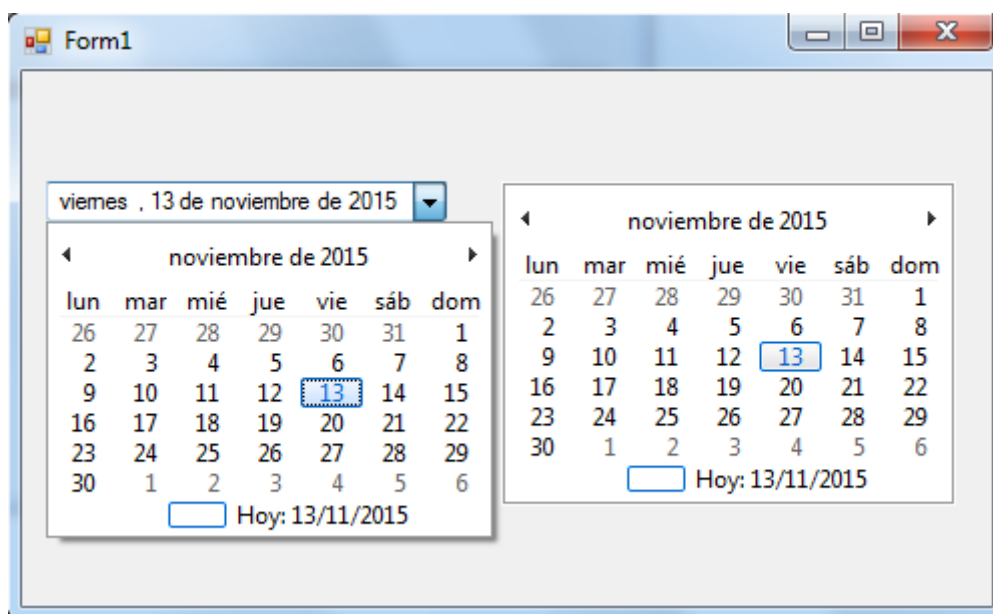
Y le concateno la cadena " %". Finalmente si he llegado al valor máximo (propiedad Maximum) deshabilito el control timer para que ya no se siga disparando este evento.

2.21. Fecha/hora: DateTimePicker y MonthCalendar

 **DateTimePicker** Estos dos controles permiten mostrar o seleccionar una fecha. Los datos más complicados de recoger y tratar en las interfaces son las fechas. La aparición de este control ha simplificado enormemente la captura de estos datos mostrando al usuario un calendario para seleccionar la fecha. Ya no hay que realizar ninguna comprobación de formato porque el dato devuelto es con toda seguridad una fecha.



El control "DateTimePicker" es más extenso que el "MonthCalendar". Éste último sólo permite seleccionar una fecha de un calendario. El primero es mucho más versátil y más cómodo porque el calendario se esconde detrás de una especie de cuadro combinado. En el cuadro de texto podemos seleccionar o ver la fecha pero si hacemos clic en la flecha se nos presentará el calendario con lo que simplificamos la interfaz al estar oculto y no ocupar un área importante en pantalla. El aspecto de los dos es:



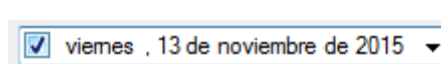
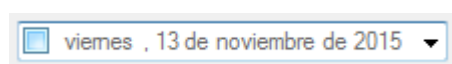
Como es un cuadro combinado (cuadro de texto + calendario) el usuario también puede escribir la fecha desde el cuadro de texto, validando la fecha y no permitiendo un valor erróneo en esta captura de datos.

Este control también puede mostrar sólo horas, con lo que puede capturar o mostrar un dato de este tipo. Por ejemplo, activando las propiedades "ShowUpDown=true" y "Format=Time" el resultado es:

En este control la propiedad más importante es la que almacena el dato fecha/hora que es "Value". Veamos las propiedades más importantes de este control:

Propiedad	Descripción
CalendarFont	Obtiene o establece el estilo de fuente que se aplica al calendario.
CalendarForeColor	Obtiene o establece el color de primer plano del calendario.
CustomFormat	Obtiene o establece la cadena de formato personalizado de fecha y hora.
DropDownAlign	Obtiene o establece la alineación del calendario desplegable en el control de selector de fecha y hora.
Format	Obtiene o establece el formato de fecha y hora que se muestra en el control.
MaxDate	Obtiene o establece la fecha y hora máximas que se pueden seleccionar en el control.
MinDate	Obtiene o establece la fecha y hora mínimas que se pueden seleccionar en el control.
ShowCheckBox	Obtiene o establece un valor que indica si se muestra una casilla de verificación a la izquierda de la fecha seleccionada.
ShowUpDown	Obtiene o establece un valor que indica si se utiliza un control de flechas para ajustar el valor de fecha y hora
Value	Obtiene o establece el valor de fecha y hora asignado al control.

La propiedad ShowCheckBox es otra propiedad importante, normalmente está establecida a False y el usuario puede seleccionar una fecha con normalidad. Pero puede que en nuestra interfaz nos interese incluir este campo o no para realizar una consulta a una base de datos, por ejemplo. Podemos entonces incluir esta casilla de verificación para confirmar que el usuario además quiere incluir este valor en la consulta. En estas dos imágenes puedes ver el efecto cuando está activada la casilla de verificación y cuando no:



Como siempre en tiempo de ejecución podemos utilizar estas propiedades:

```
chk_fecha.ShowCheckBox=True
```

Otra propiedad importante es la del formato de la fecha/hora que se muestra en el cuadro de texto: propiedad Format. Esta propiedad determina el formato de fecha y hora en que se muestra la fecha. El formato de fecha y hora se basa en la configuración regional del sistema operativo del usuario.

Si queremos personalizarla para poner un estilo distinto que los preestablecidos podemos utilizar la propiedad "CustomFormat".

Para mostrar los valores literales de cadena que contienen separadores de fecha y hora o cadenas de formato, es necesario utilizar caracteres de escape en la subcadena. Por ejemplo,

para mostrar la fecha como "June 06 at 3:00 PM", hay que establecer la propiedad **CustomFormat** en "MMMM dd 'at' t:mm tt". Si la subcadena "at" no está escrita entre caracteres de escape, el resultado será "Junio 06 aP 3:00PM" debido a que el carácter "t" se leerá como la cadena de formato AM/PM de una letra.

Las cadenas de formato pueden combinarse para después aplicarse a la fecha y la hora. Por ejemplo, para mostrar la fecha y la hora como 06/01/2001 12:00 PM, esta propiedad se debe establecer en "dd/MM/yyyy hh':'mm tt". Ahora veremos la tabla de formatos personalizados y cómo construirlos. Se muestran los caracteres que se pueden utilizar para crear formatos de fecha y hora definidos por el usuario. A diferencia de versiones anteriores de Visual Basic, estos caracteres de formato distinguen mayúsculas de minúsculas.

Carácter	Descripción
(:)	Separador de hora. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de hora. Este separador horario separa horas, minutos y segundos cuando se da formato a valores horarios. El carácter real utilizado como separador de hora en los resultados con formato viene determinado por el valor LocaleID del sistema.
(/)	Separador de fecha. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de fecha. Este separador separa el día, mes y año cuando se da formato a los valores de fecha. El carácter real utilizado es el especificado como separador de fecha en la configuración local de su sistema.
(%)	Se utiliza para indicar que el carácter siguiente debe leerse como formato de una sola letra sin tener en cuenta las posibles letras finales. También se emplea para indicar que un formato de una sola letra se lea como formato definido por el usuario. Para obtener información más detallada, vea la explicación siguiente.
d	Muestra el día como un número sin cero a la izquierda (por ejemplo, 1). Use %d si es el único carácter en el formato numérico definido por el usuario.
dd	Muestra el día como un número con cero a la izquierda (por ejemplo, 01).
ddd	Muestra el día de forma abreviada (por ejemplo, Dom).
dddd	Muestra el día de forma completa (por ejemplo, Domingo).
M	Muestra el mes como un número sin cero a la izquierda (por ejemplo, enero se representa como 1). Use %M si es el único carácter en el formato numérico definido por el usuario.
MM	Muestra el mes como un número con cero a la izquierda (por ejemplo, 01/12/01), doce de enero de 2001.
MMM	Muestra el mes en forma abreviada (por ejemplo, Ene).
MMMM	Muestra el mes en forma completa (por ejemplo, Enero).
gg	Especifica la era histórica (por ejemplo, A.D.)

h	Muestra la hora como un número sin ceros a la izquierda y en formato de doce horas (por ejemplo, 1:15:15 PM). Use %h si es el único carácter en el formato numérico definido por el usuario.
hh	Muestra la hora como un número con ceros a la izquierda y en formato de doce horas (por ejemplo, 01:15:15 PM).
H	Muestra la hora como un número sin ceros a la izquierda y en formato de doce horas (por ejemplo, 1:15:15 PM). Use %H si es el único carácter en el formato numérico definido por el usuario.
HH	Muestra la hora como un número con ceros a la izquierda y en formato de doce horas (por ejemplo, 01:15:15).
m	Muestra los minutos como un número sin ceros a la izquierda (por ejemplo, 12:1:15). Use %m si es el único carácter en el formato numérico definido por el usuario.
mm	Muestra los minutos como un número con ceros a la izquierda (por ejemplo, 12:01:15).
s	Muestra los segundos como un número sin ceros a la izquierda (por ejemplo, 12:15:5). Use %s si es el único carácter en el formato numérico definido por el usuario.
ss	Muestra los segundos como un número con ceros a la izquierda (por ejemplo, 12:15:05).
F	Muestra fracciones de segundos. Por ejemplo, ff muestra centésimas de segundo, mientras que ffff muestra diez milésimas de segundo. Puede utilizar hasta siete símbolos f en el formato definido por el usuario. Use %f si es el único carácter en el formato numérico definido por el usuario.
T	Usa el reloj de doce horas; muestra una A mayúscula para cualquier hora entre medianoche y mediodía, y una P mayúscula para cualquier hora entre mediodía y medianoche. Utilice %t si éste es el único carácter del formato numérico definido por el usuario.
tt	Usa el reloj de doce horas y muestra la leyenda AM en mayúsculas para cualquier hora entre medianoche y mediodía; y PM en mayúsculas para cualquier hora entre mediodía y medianoche.
y	Muestra el año sin cero inicial. Utilice %y en caso de que éste sea el único carácter en su formato numérico definido por el usuario.
yy	Muestra el año en formato numérico de dos dígitos sin cero inicial, si procede.
yyy	Muestra el año en formato numérico de cuatro dígitos.
yyyy	Muestra el año en formato numérico de cuatro dígitos.
z	Muestra el desplazamiento de zona horaria sin cero a la izquierda (por ejemplo, -8). Use %z si es el único carácter en el formato numérico definido por el usuario.
zz	Muestra el desplazamiento de zona horaria con un cero a la izquierda (por ejemplo, -08).
zzz	Muestra el desplazamiento completo de zona horaria (por ejemplo, -08:00).

Ejemplo

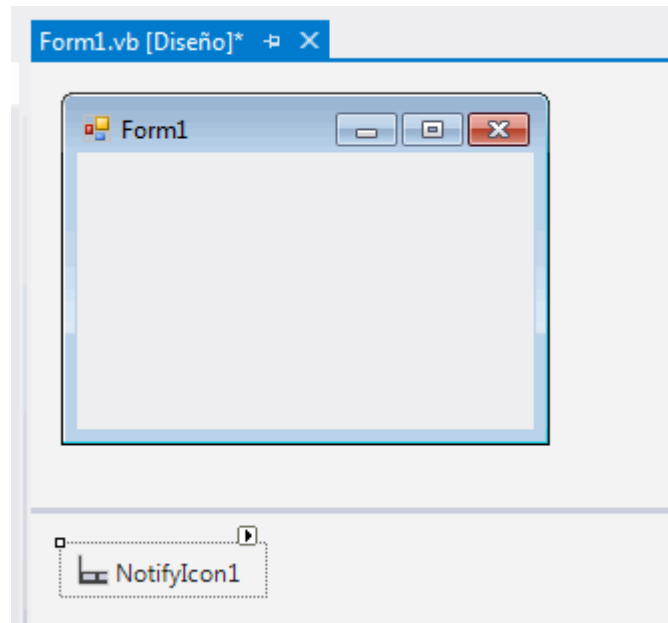
A continuación se muestran algunos ejemplos de formatos de hora y fecha definidos por el usuario y correspondientes al 7 de diciembre de 1958, a las ocho horas, cincuenta minutos y treinta y cinco segundos de la tarde (Diciembre 7, 1958, 8:50 PM, 35 segundos):

Formato	Muestra
M/d/yy	12/7/58
d-MMM	7-Dic
d-MMMM-yy	7-Diciembre-58
d MMMM	7 Diciembre
MMMM yy	Diciembre 58
hh:mm tt	08:50 PM
h:mm:ss t	8:50:35 P
H:mm	20:50
H:mm:ss	20:50:35
M/d/yyyy H:mm	12/7/1958 20:50

2.22. Control de iconos de notificación. NotifyIcon



NotifyIcon Un sencillo pero útil control es el de los iconos de notificación, mostrados a la derecha en la barra de tareas de Windows. Este es otro control "oculto", es decir, cuando lo añadamos a nuestro proyecto no se verá dentro de la interfaz sino en la parte exterior indicando que está disponible para utilizarse:



Los iconos del área de estado son accesos directos a procesos que se ejecutan en segundo plano como, por ejemplo, un programa de protección antivirus o un control de volumen. Estos procesos no incluyen sus propias interfaces de usuario. La clase **NotifyIcon** proporciona una manera de programar en esta funcionalidad. Las propiedades más interesantes son:

Propiedad	Descripción
ContextMenuStrip	Menú contextual que se desplegará con el icono
Icon	Icono que mostrará.
Text	Texto de la información sobre herramientas que se muestra cuando el ratón pasa sobre un icono del área de estado.
Visible	Obtiene o establece un valor que indica si el icono es visible en la Bandeja del sistema de Windows.

La propiedad **Icon** define el icono que aparece en el área de estado. Los menús emergentes de un icono se definen con la propiedad **ContextMenuStrip**. La propiedad **Text** asigna texto de información sobre herramientas. Para que el icono aparezca en el área de estado, la propiedad **Visible** debe establecerse en **true**.

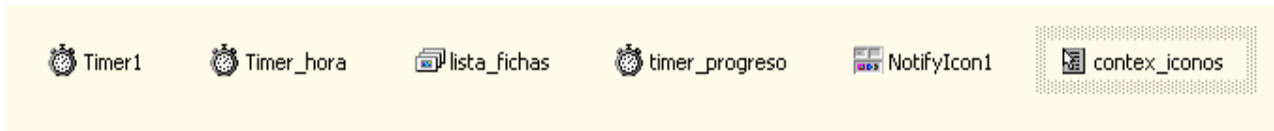
Los eventos a que atenderá son los habituales de los controles: clic, dobleclic, mousemove... pero no nos harán falta programarlos ya que basta con modificar la propiedad "ContextMenuStrip" para asignarle unas opciones u otras dependiendo del estado en que se encuentre.

Para cargar un icono en tiempo de ejecución utilizaremos las instrucciones ya conocidas de carga de gráficos desde el disco:

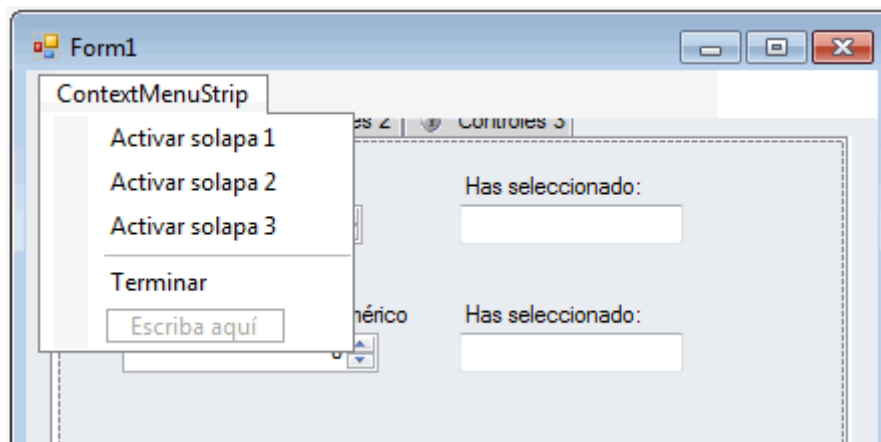
```
notifyIcon1.Icon = New Icon("icono.ico")
```

Pero si no lo queremos hacer así podemos por ejemplo añadir dos iconos al programa estando sólo uno de ellos visible, cuando queramos cambiar su estado lo ponemos oculto y mostramos el segundo.

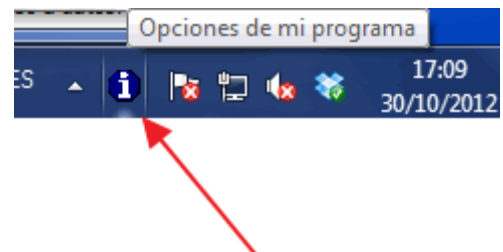
En nuestro ejemplo final crearemos un menú contextual que mostraremos con el botón de derecho del icono. Así que añadimos un control de este tipo y un menú contextual (ContextMenuStrip):



Creamos el menú contextual con estos valores:



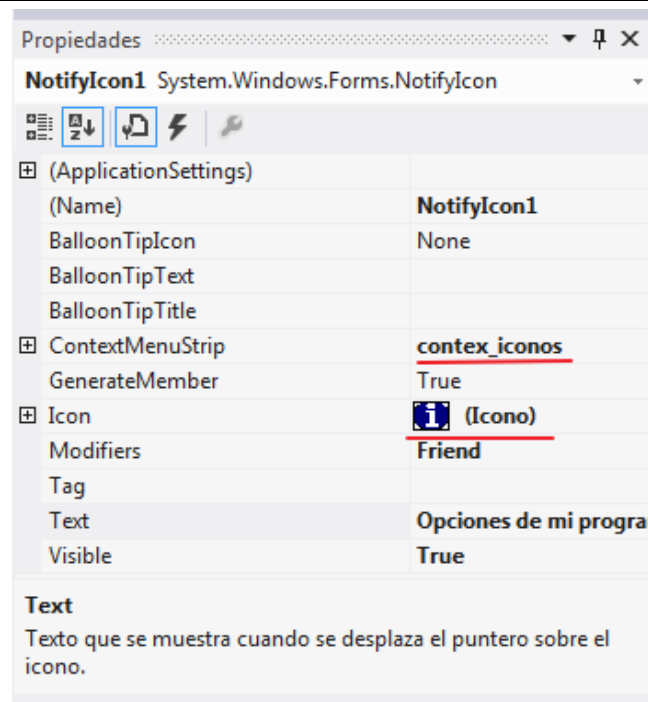
Que son las acciones que quiero hacer... cuando se esté ejecutando el programa podremos cambiar de una solapa a otra directamente desde aquí e incluso terminar el programa. Con el botón derecho sobre este icono:



Se muestra este menú:



Para conseguir esto, una vez tenemos creado nuestro menú añadimos un control de notificaciones y le asignamos las propiedades:



Es muy importante que se ponga la propiedad "Text" ya que es lo que nos mostrará al poner el ratón encima del icono. Añadimos un icono y le asignamos con "ContextMenuStrip" el menú que hemos creado. Podemos cambiar de forma dinámica el icono como hemos dicho antes pero en este ejemplo dejaremos el mismo. Veamos ahora las acciones para activar las solapas.

```
Private Sub ActivarSolapa1ToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ActivarSolapa1ToolStripMenuItem.Click
    TabControl1.SelectedIndex = 0
End Sub

Private Sub ActivarSolapa2ToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ActivarSolapa2ToolStripMenuItem.Click
    TabControl1.SelectedIndex = 1
End Sub

Private Sub ActivarSolapa3ToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ActivarSolapa3ToolStripMenuItem.Click
    TabControl1.SelectedIndex = 2
End Sub

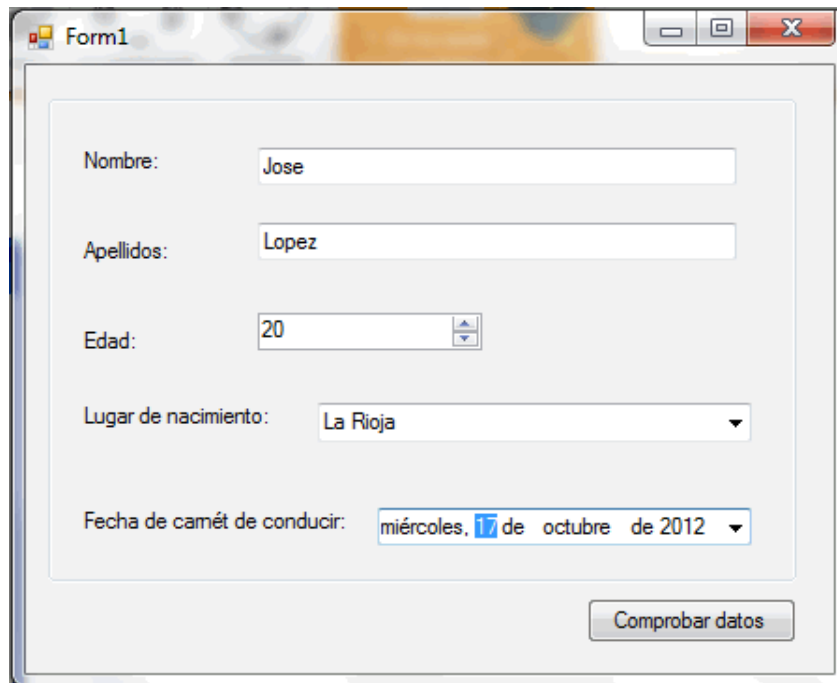
Private Sub TerminarProgramaToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TerminarProgramaToolStripMenuItem.Click
End
End Sub
```

Lógicamente hemos utilizado el evento clic de cada uno de estas opciones de menú y hemos activado la solapa correspondiente.

Ejercicios

Ejercicio 1

Crea un programa que tenga esta interfaz:

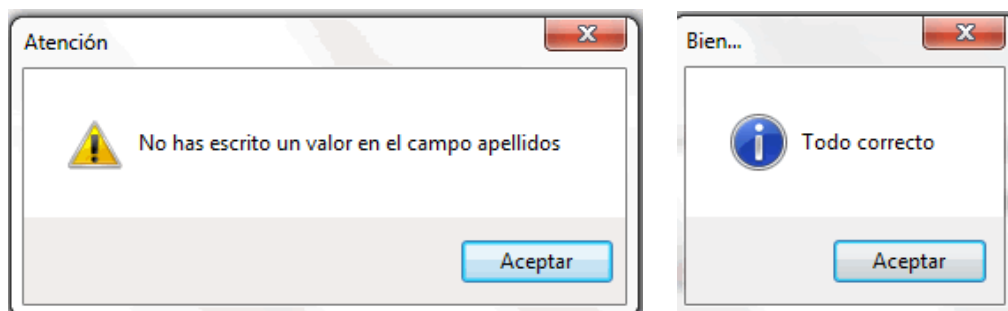


The screenshot shows a Windows application window titled 'Form1'. Inside the window is a form with the following fields and controls:

- Nombre:** A text box containing the value 'Jose'.
- Apellidos:** A text box containing the value 'Lopez'.
- Edad:** A numeric spinner box set to the value '20'.
- Lugar de nacimiento:** A dropdown menu with 'La Rioja' selected.
- Fecha de camét de conducir:** A date picker showing 'miércoles, 17 de octubre de 2012'.
- Comprobar datos:** A button located at the bottom right of the form.

Cuando se pulse "Comprobar" debe comprobar que:

- Se haya escrito algo en el campo del nombre
- Se haya escrito algo en el campo del apellido
- Se haya seleccionado una ciudad
- La fecha de carné no puede ser posterior al día de hoy
- Muestra un mensaje de todo correcto si se cumplen los campos



Nota:

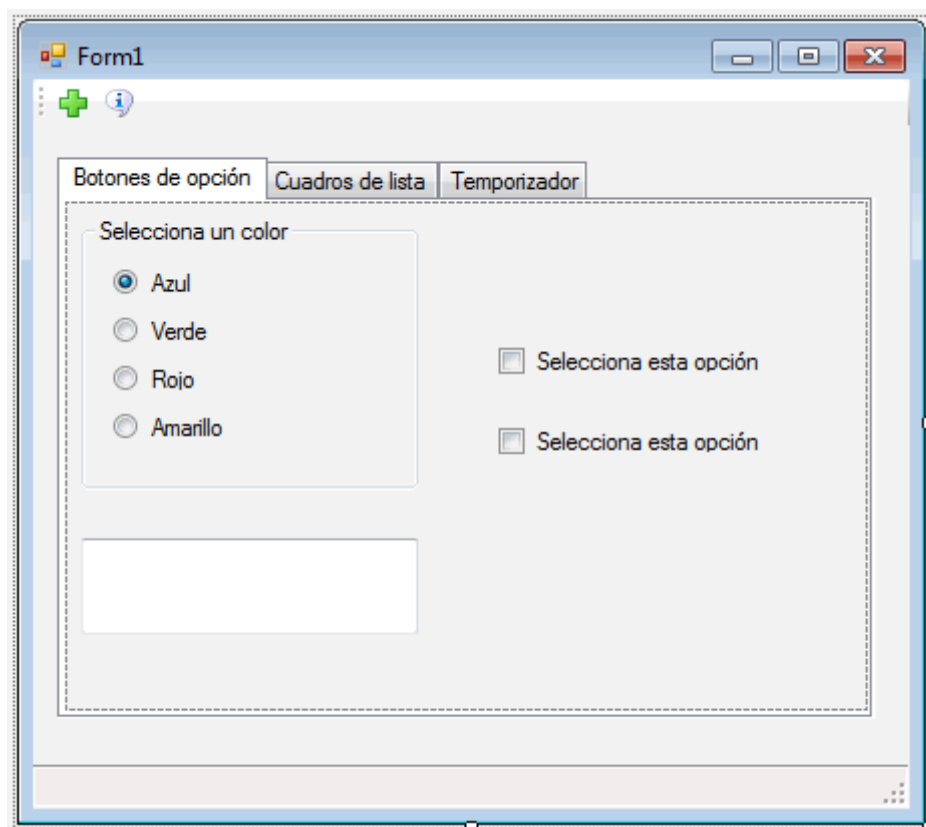
- Rellena el cuadro combinado con unos elementos de prueba en el evento Load de formulario. El primero de ellos debe ser "Selecciona un valor" que es el activo predeterminado
- Pon el cursor o el enfoque en el campo que no cumpla con las condiciones

Ejercicio 2

Crea un programa que tenga esta interfaz:

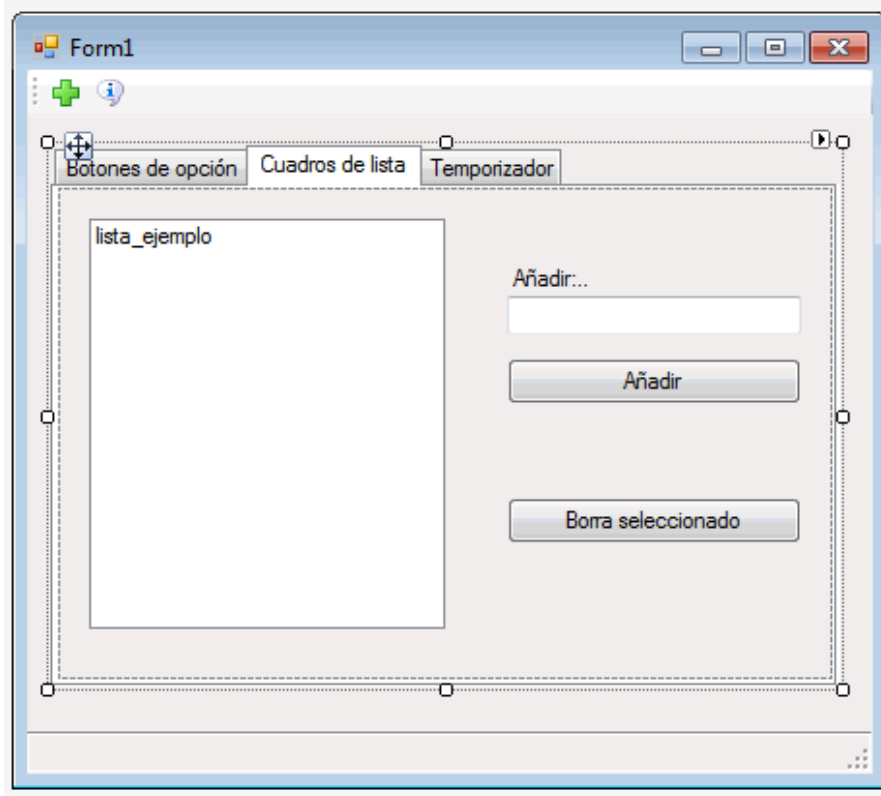
Ficha 1

- Que cambie el color del fondo de un cuadro de texto multilinea
- Que dos casillas de verificación indiquen en una barra de estado que están activas. Además pulsarán o soltarán dos botones de una barra de herramientas



Ficha 2

- Rellenar un cuadro de lista con los elementos que se vayan introduciendo. No se permiten repetidos, ni en blanco, ni que superen los 30 caracteres.
- Eliminar el elemento seleccionado y dejar marcado el inmediatamente inferior



Ficha 3

Cuando se active la tercera solapa se pondrá en marcha un reloj que mostrará la hora actualizada cada segundo en un nuevo panel creado mediante código en la barra de estado

Cada segundo un trackbar y un progress bar mostrarán el segundo actual

