Capıtulo 4

System.Windows.Forms (Formularios)

4.1. Introducción

Un formulario no es más que una clase.

Los formularios constan de propiedades, métodos y eventos.

4.2. Fundamentos de las ventanas

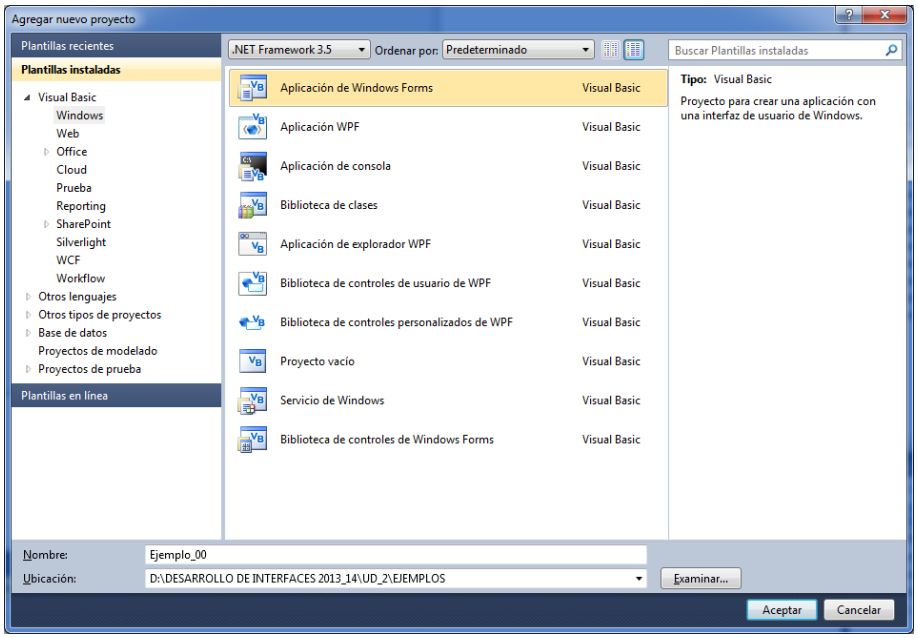
En su forma más cruda, una ventana se define como un área rectangular y limitada en la pantalla. Suele estar delimitada por un recuadro (visible o no) y a menudo contiene datos u otras ventanas.

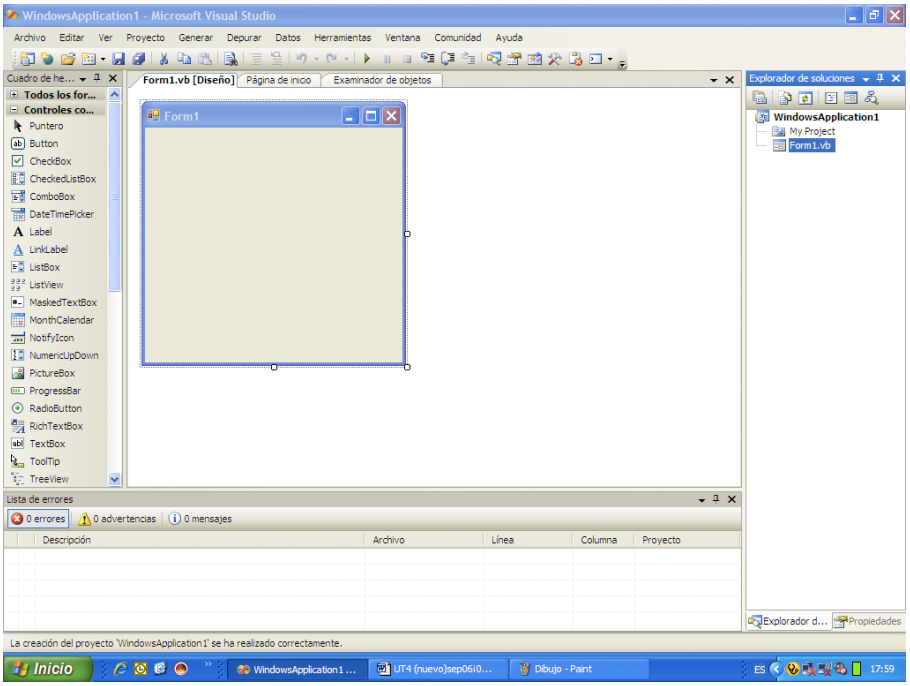
Las ventanas son el método principal de interacción entre sus aplicaciones y el usuario final. Por lo tanto, se encargan de recoger todas las entradas (del teclado, del ratón, etc.) de usuario y procesarlas de acuerdo a sus instrucciones. Además se utilizan para mostrar toda la información deseada, en el formato deseado.

En el entorno .NET FrameWork los formularios y controles se han convertido en clases completamente heredables.

Haciendo doble click sobre el formulario, vemos el código Form1_ Load el constructor lo crearemos nosotros.

```
1
2 Public Class Form1
3
```


Agregar nuevo proyecto
Plantillas recientes
Plantillas instaladas
Visual Basic
Windows
Web
Office
Cloud
Prueba
Reporting
SharePoint
Silverlight
WCF
Workflow
Otros lenguajes
Otros tipos de proyectos
Base de datos
Proyectos de modelado
Proyectos de prueba
Plantillas en línea
.NET Framework 3.5
Ordenar por: Predeterminado
Buscar Plantillas instaladas
Aplicación de Windows Forms
Aplicación WPF
Aplicación de consola
Biblioteca de clases
Aplicación de explorador WPF
Biblioteca de controles de usuario de WPF
Biblioteca de controles personalizados de WPF
Proyecto vacío
Servicio de Windows
Biblioteca de controles de Windows Forms
Visual Basic
Tipo: Visual Basic
Proyecto para crear una aplicación con una interfaz de usuario de Windows.
Nombre: Ejemplo_00
Ubicación: D:\DESARROLLO DE INTERFACES 2013_14\UD_2\EJEMPLOS
Examinar...
Aceptar
Cancelar


WindowsApplication1 - Microsoft Visual Studio
Archivo Editar Ver Proyecto Generar Depurar Datos Herramientas Ventana Comunidad Ayuda
Cuadro de he...
Todos los for...
Controles co...
Puntero
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
Form1.vb [Diseño]
Página de inicio
Examinador de objetos
Form1
Explorador de soluciones
WindowsApplication1
My Project
Form1.vb
Lista de errores
0 errores
0 advertencias
0 mensajes
Descripción
Archivo
Línea
Columna
Proyecto
Explorador d...
Propiedades
La creación del proyecto 'WindowsApplication1' se ha realizado correctamente.
Inicio
WindowsApplication1 ...
UT4 (nuevo)sep06i0...
Dibujo - Paint
ES
17:59

```
4       Public Sub New()
5
6           ' Llamada necesaria para el ~nDiseador de Windows Forms.
7           InitializeComponent()
8
9           ' Agregue cualquier óinicializacin édespus de la llamada a
                InitializeComponent().
10
11      End Sub
12
13      Private Sub Form1_Load(ByVal sender As System.Object , ByVal e As
            System.EventArgs) Handles MyBase.Load
14
15      End Sub
16  End Class
```

Lo primero que aparece es el constructor, que invoca al constructor de la superclase, a continuación manda ejecutar el procedimiento InitializeComponent codificado un poco más abajo. Este procedimiento inicializa los elementos (controles) colocados en el formulario.

Si queremos llevar a cabo alguna inicialización podemos codificarla después de la ejecución de este método.

Existe también otro método denominado Finalize (Dispose) que se ejecuta cuando se cierra la ventana, este método se utiliza para liberar recursos.

Los formularios son el lienzo sobre el que dibujar los controles necesarios para que el usuario final logre el objetivo de introducir datos, administrar el sistema, desarrollar aplicaciones o un número ilimitado de tareas.

El formulario es un control como cualquier otro del cuadro de herramientas, aunque también es muchas otras cosas.

4.3. Controles del cuadro de herramientas

Puntero No es un control. Se utiliza para seleccionar, mover y ajustar el tamaño de los objetos.

Label Una etiqueta permite mostrar un texto de una o más líneas que no puede ser modificado por el usuario. Son útiles para dar instrucciones al usuario.

LinkLabel Se trata de una etiqueta de Windows que puede mostrar hipervínculos.

```
1
2   Private Sub LinkLabel1_LinkClicked(ByVal sender As System.Object ,
        ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs)
        Handles LinkLabel1.LinkClicked
```

```
3           System.Diagnostics.Process.Start("http://microsoft.com")
4
5   End Sub
```

Button Un botón de pulsación tiene asociada una orden con él. Esta orden se ejecutará cuando el usuario haga clic sobre el botón.

TextBox Una caja de texto es un área dentro del formulario en la que el usuario puede escribir o visualizar texto.

Todos los controles disponen de un conjunto de propiedades (ventana que aparece debajo del explorador de soluciones), las propiedades más habituales son:

Name Nombre del objeto.

Text Almacena el texto que mostrará el control.

Font Indica el tipo de fuente que utilizará el control para mostrar su texto.

TextAlign Indica el tipo de alineación que se aplicará al texto respecto a los limites del control.

Location Hace referencia a un objeto de la clase Point que almacena las coordenadas de la esquina superior izquierda del componente.

Size Hace referencia a un objeto de la clase Size que almacena el tamaño (ancho y alto) del componente.

TabIndex Indica el orden Tab de un control. Todos los controles tienen un orden tab (0,1,2,3.etc.) por omisión, en función del orden en el que hayan sido añadidos al formulario. El control que quedará enfocado (seleccionado) cuando se arranca la aplicación será el de orden tab 0 y cuando se utilice la tecla tab para cambiar el foco a otro control, se seguirá el orden tab establecido. Sólo pueden tener el foco aquellos controles que tienen su propiedad TabStop a valor True.

4.4. Manejo de eventos

Cuando sobre algún componente ocurra algún suceso, normalmente se espera que pase algo. Lógicamente ese algo hay que programarlo y para poder hacerlo hay que saber como manejar el evento producido por el componente sobre el que ocurrió el suceso.

Ejemplo: Método que aparece cuando hacemos doble click sobre un botón con nombre Aceptar

```
1  Private Sub Aceptar_Click(ByVal sender As System.Object, ByVal e As
       System.EventArgs) Handles Aceptar.Click
```

El método Aceptar_Click se ejecutará cuando el usuario haga clic sobre botón. El primer parámetro del método hace referencia al objeto que produce el evento y el segundo contiene información que depende del evento producido.

La palabra reservada Handles permite definir que método se ejecutará para un determinado evento producido por un componente.

ControladorEvento() Handles Objeto1.evento1, objeto2.evento2, ...

Se ejcutará el controladorEvento, para el evento1 del objeto1 y el evento2 del objeto2

Otra forma de definir un controlador (método que responde al evento) para un evento producido por un determinado componente es por medio de la sentencia **AddHandler** cuya sintaxis es la siguiente:

AddHandler miObjeto.Evento, AddressOf ControladorDelEvento

Añade como respuesta al Evento de miObjeto el ControladorDelEvento

AddHandler es mucho más flexible ya que permite añadir o cambiar durante la ejecución un controlador, así como asociar múltiples controladores con un único evento. **RemoveHandler** permite desconectar un controlador de un evento.

RemoveHandler miObjeto.Evento, AddresOf ControladorDelEvento

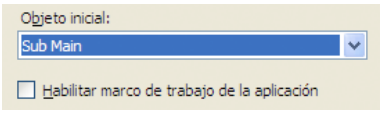
Ejemplo:

```
1  Public Sub saludoCastellano(ByVal sender As System.Object, ByVal e As
       System.EventArgs)
2          cajaSaludo.Text = "Hola"
3  End Sub
4
5  Public Sub saludoEuskara(ByVal sender As System.Object, ByVal e As
       System.EventArgs)
6          cajaSaludo.Text = "Kaixo"
7  End Sub
8
9  Private Sub castellano_CheckedChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles castellano.CheckedChanged
10         AddHandler Saludar.Click, AddressOf saludoCastellano
11 End Sub
12
13 Private Sub Euskara_CheckedChanged(ByVal sender As System.Object, ByVal e
       As System.EventArgs) Handles Euskara.CheckedChanged
14         AddHandler Saludar.Click, AddressOf saludoEuskara
```


```
15  End Sub
```

Ejemplo (La ejecución empieza por el método main)

Asegurarse, en propiedades del proyecto, que el objeto inicial sea el procedimiento Main y la opción Ḧabilitar.. . .ësté desactivada.



El form1 recoge los datos.

El form2 muestra los datos resultado.

Clase Controlador: El controlador, realiza las operaciones.

```
1   Public Class Controlador
2       Private oF1 As Form1
3       Private oF2 As Form2
4
5       Public Sub convertir(ByVal pesetas As Integer)
6           Dim oCalculadora = New Calculadora()
7           Dim resultado As Double = oCalculadora.convertirEuros(pesetas)
8           oF2.asignarResultado(resultado)
9           oF1.Hide()
10          oF2.Show()
11      End Sub
12
13      Public Sub comenzar()
14          oF2.Hide()
```


Aplicación Conversor
Conversor de euros
Cantidad en pesetas
Convertir


Resultado
Cantidad en euros
Label2
Volver
Salir

```
15          oF1.mostrarF1()
16      End Sub
17
18
19      Public Function getF1() As Form1
20          Return oF1
21      End Function
22
23      Public Function getF2() As Form2
24          Return oF2
25      End Function
26
27      Public Sub terminar()
28          'El émtodo dispose libera todos los recursos
29          oF2.Dispose()
30          oF1.Dispose()
31          Application.Exit()
32      End Sub
33
34      Public Sub New()
35          oF1 = New Form1(Me)
36          oF2 = New Form2(Me)
37      End Sub
38  End Class
```

Clase Form1:

```
1   Public Class Form1
2       Private oc As Controlador
3
4       Private Sub convertir_Click(ByVal  sender As System.Object , ByVal e As
            System.EventArgs) Handles convertir.Click
5           oc.convertir(Integer.Parse(cantidad.Text))
6       End Sub
7
8       Public Sub mostrarF1()
9           Me.Show()
10          'Limpiar la caja de texto
11          Me.cantidad.Clear()
12          'Colocar el cursor en la caja de texto
13          Me.cantidad.Focus()
14      End Sub
15
16      Public Sub New(ByRef oc As Controlador)
17
18          ' Llamada necesaria para el ~nDiseador de Windows Forms.
19          InitializeComponent()
20
21          ' Agregue cualquier óinicializacin édespus de la llamada a
                InitializeComponent().
22          Me.oc = oc
23      End Sub
24  End Class
```

Clase Form2:

```
1   Public Class Form2
2       Private oc As Controlador
```

```
3
4       Public Sub asignarResultado(ByVal  valor As Double)
5           resultado.Text  = Convert.ToString(valor)
6       End Sub
7
8       Private Sub Salir_Click(ByVal  sender As System.Object , ByVal e As
            System.EventArgs) Handles Salir.Click
9           oc.terminar()
10      End Sub
11
12      Private Sub Volver_Click(ByVal  sender As System.Object , ByVal e As
            System.EventArgs) Handles Volver.Click
13          oc.comenzar()
14      End Sub
15
16      Public Sub New(ByRef oc As Controlador)
17
18          ' Llamada necesaria para el ~nDiseador de Windows Forms.
19          InitializeComponent()
20
21          ' Agregue cualquier óinicializacin édespus de la llamada a
                InitializeComponent().
22          Me.oc = oc
23
24      End Sub
25  End Class
```

Clase Calculadora:

```
1   Public Class Calculadora
2
3       Public Function convertirEuros(ByVal  pesetas As Integer) As Double
4           Return pesetas / 166.386
5       End Function
6
7       Public Sub New()
8
9       End Sub
10  End Class
```

Main:

```
1   Module Module1
2       Sub Main()
3           'Dim oControlador  = New Controlador()
4           'oControlador.crearObjetos()
5           'Iniciar la óejecucin
6           Dim oc = New Controlador()
7           oc.comenzar()
8           Application.Run()
9       End Sub
10  End Module
```

4.4.1. Shared

La palabra clave Shared indica que un elemento está compartido. Los elementos compartidos no están asociados a una instancia específica de una clase, es decir, no hace falta crear un objeto para poder utilizarlo.

4.4.2. La clase Control

Todos los controles que colocamos en el formulario heredan de esta clase.

Propiedades públicas

AccessibilityObject	Obtiene **AccessibleObject** asignado al control.
AccessibleDefaultActionDescription	Obtiene o establece la descripción de la acción predeterminada del control que las aplicaciones cliente de accesibilidad utilizan.
AccessibleDescription	Obtiene o establece la descripción del control que las aplicaciones cliente de accesibilidad utilizan.
AccessibleName	Obtiene o establece el nombre del control que las aplicaciones cliente de accesibilidad utilizan.
AccessibleRole	Obtiene o establece la función accesible del control.
AllowDrop	Obtiene o establece un valor que indica si el control puede aceptar los datos que el usuario arrastra al mismo.
Anchor	Obtiene o establece los bordes del control que se acoplan a los bordes de su contenedor.
BackColor	Obtiene o establece el color de fondo del control.
BackgroundImage	Obtiene o establece la imagen de fondo que se muestra en el control.
BindingContext	Obtiene o establece BindingContext del control.
Bottom	Obtiene la distancia que existe entre el borde inferior del control y el borde superior del área cliente de su contenedor.
Bounds	Obtiene o establece el tamaño y la ubicación del control, incluidos los elementos de no cliente.
CanFocus	Obtiene un valor que indica si el control puede recibir el foco.
CanSelect	Obtiene un valor que indica si el control se puede seleccionar.
Capture	Obtiene o establece un valor que indica si el control ha capturado el mouse (ratón).

CausesValidation	Obtiene o establece un valor que indica si el control hace que se realice una validación de todos los controles que requieren validación cuando reciben el foco.
ClientRectangle	Obtiene el rectángulo que representa el área cliente del control.
ClientSize	Obtiene o establece el alto y el ancho del área cliente del control.
CompanyName	Obtiene el nombre de la compañía o del creador de la aplicación que contiene el control.
Container (se hereda de Component)	Obtiene IContainer que contiene Component.
ContainsFocus	Obtiene un valor que indica si el control, o uno de sus controles secundarios, tiene el foco de entrada en la actualidad.
ContextMenu	Obtiene o establece el menú contextual asociado al control.
Controls	Obtiene la colección de controles que contiene el control.
Created	Obtiene un valor que indica si se ha creado el control.
Cursor	Obtiene o establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
DataBindings	Obtiene los enlaces de datos del control.
DefaultBackColor	Obtiene el color de fondo predeterminado del control.
DefaultFont	Obtiene la fuente predeterminada del control.
DefaultForeColor	Obtiene el color de primer plano predeterminado del control.
DisplayRectangle	Obtiene el rectángulo que representa el área de presentación del control.
Disposing	Obtiene un valor que indica si el control está en proceso de eliminación .
Dock	Obtiene o establece el borde del contenedor principal al que está acoplado un control.
Enabled	Obtiene o establece un valor que indica si el control puede responder a la interacción del usuario.

Focused	Obtiene un valor que indica si el control tiene el foco de entrada.
Font	Obtiene o establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control.
Handle	Obtiene el identificador de ventana al que está enlazado el control.
HasChildren	Obtiene un valor que indica si el control contiene uno o más controles secundarios.
Height	Obtiene o establece el alto del control.
ImeMode	Obtiene o establece el modo de Editor de métodos de entrada (IME) del control.
InvokeRequired	Obtiene un valor que indica si el llamador debe llamar a un método de invocación cuando realiza llamadas a métodos del control porque el llamador se encuentra en un subproceso distinto al del control donde se creó.
IsAccessible	Obtiene o establece un valor que indica si el control es visible para las aplicaciones de accesibilidad.
IsDisposed	Obtiene un valor que indica si el control se ha eliminado.
IsHandleCreated	Obtiene un valor que indica si el control tiene un identificador asociado.
Left	Obtiene o establece la coordenada x del borde izquierdo de un control, en píxeles.
Location	Obtiene o establece las coordenadas de la esquina superior izquierda del control en relación con la esquina superior izquierda de su contenedor.
ModifierKeys	Obtiene un valor que indica cuál de las teclas modificadoras (MAYÚS, CTRL y ALT) está presionada.
MouseButtons	Obtiene un valor que indica cuál de los botones del mouse está presionado.
MousePosition	Obtiene la posición del cursor del mouse en coordenadas de pantalla.
Name	Obtiene o establece el nombre del control.
Parent	Obtiene o establece el contenedor

	principal del control.
ProductName	Obtiene el nombre de producto del ensamblado que contiene el control.
ProductVersion	Obtiene la versión del ensamblado que contiene el control.
RecreatingHandle	Obtiene un valor que indica si el control está volviendo a crear su identificador en la actualidad.
Region	Obtiene o establece la región de ventana asociada al control.
Right	Obtiene la distancia entre el borde derecho del control y el borde izquierdo de su contenedor.
RightToLeft	Obtiene o establece un valor que indica si los elementos del control se alinean para admitir configuraciones regionales utilizando fuentes de derecha a izquierda.
Site	Reemplazado. Obtiene o establece el sitio del control.
Size	Obtiene o establece el alto y el ancho del control.
TabIndex	Obtiene o establece el orden de tabulación del control en su contenedor.
TabStop	Obtiene o establece un valor que indica si el usuario puede dar el foco a este control mediante la tecla TAB.
Tag	Obtiene o establece el objeto que contiene datos sobre el control.
Text	Obtiene o establece el texto asociado al control.
Top	Obtiene o establece la coordenada y del borde superior del control, en píxeles.
TopLevelControl	Obtiene el control principal que no es secundario de ningún otro control de formularios Windows Forms. Normalmente, se trata del Form más externo en el que está contenido el control.
Visible	Obtiene o establece un valor que indica si se muestra el control.
Width	Obtiene o establece el ancho del control.

Métodos públicos

BeginInvoke	Sobrecargado. Ejecuta un delegado de forma asincró nica en el subproceso donde se

	creó el identificador subyacente del control.
BringToFront	Coloca el control al principio del orden Z.
Contains	Obtiene un valor que indica si el control especificado es un control secundario del control.
CreateControl	Obliga a que se cree el control, incluidos el identificador y los controles secundarios.
CreateGraphics	Crea el objeto Graphics para el control.
CreateObjRef (se hereda de MarshalByRefObject)	Crea un objeto que contiene toda la información relevante necesaria para generar un proxy utilizado para comunicarse con un objeto remoto.
Dispose (se hereda de Component)	Sobrecargado. Libera los recursos utilizados por Component.
DoDragDrop	Inicia una operación de arrastrar y colocar.
EndInvoke	Recupera el valor devuelto por la operación asíncrona representada por el objeto IAsyncResult que se pasa.
Equals (se hereda de Object)	Sobrecargado. Determina si dos instancias de Object son iguales.
FindForm	Recupera el formulario en el que se encuentra el control.
Focus	Establece el foco de entrada en el control.
FromChildHandle	Recupera el control que contiene el identificador especificado.
FromHandle	Devuelve el control actualmente asociado al identificador especificado.
GetChildAtPoint	Recupera el control secundario ubicado en las coordenadas especificadas.
GetContainerControl	Devuelve el siguiente ContainerControl (en sentido ascendente) de la cadena de controles principales del control.
GetHashCode (se hereda de Object)	Sirve como función hash para un tipo concreto, apropiado para su utilización en algoritmos de hash y estructuras de datos como las tablas hash.
GetLifetimeService (se hereda de MarshalByRefObject)	Recupera el objeto de servicio de duración actual que controla la directiva de duración de esta instancia.
GetNextControl	Recupera el siguiente control, hacia delante o hacia atrás, en el orden de tabulación de controles secundarios.
GetType (se hereda de Object)	Obtiene el objeto Type de la instancia actual.

Hide Compatible con .NET Compact Framework.	Oculta el control al usuario.
InitializeLifetimeService (se hereda de MarshalByRefObject)	Obtiene un objeto de servicio de duración para controlar la directiva de duración de esta instancia.
Invalidate	Sobrecargado. Invalida una región específica del control y hace que se envíe un mensaje de dibujo al control.
Invoke	Sobrecargado. Ejecuta un delegado en el subproceso que posee el identificador de ventana subyacente del control.
IsMnemonic	Determina si el carácter especificado es una tecla de acceso asignada al control en la cadena especificada.
PerformLayout	Sobrecargado. Obliga al control a aplicar la lógica de diseñ o a los controles secundarios.
PointToClient	Calcula la ubicación del punto especificado de la pantalla, en coordenadas de cliente.
PointToScreen	Calcula la ubicación del punto especificado de cliente en coordenadas de pantalla.
PreProcessMessage	Preprocesa los mensajes de entrada en el bucle de mensajes antes de enviarlos.
RectangleToClient	Calcula el tamaño y la ubicación del rectá ngulo de pantalla especificado, en coordenadas de cliente.
RectangleToScreen	Calcula el tamaño y la ubicación del rectá ngulo de cliente especificado, en coordenadas de pantalla.
Refresh	Obliga al control a invalidar su área cliente y, acto seguido, obliga a que vuelva a dibujarse el control y sus controles secundarios.
ResetBackColor	Restablece el valor predeterminado de la propiedad BackColor.
ResetBindings	Restablece el valor predeterminado de la propiedad DataBindings.
ResetCursor	Restablece el valor predeterminado de la propiedad Cursor.
ResetFont	Restablece el valor predeterminado de la propiedad Font.
ResetForeColor	Restablece el valor predeterminado de la propiedad ForeColor.

ResetImeMode	Restablece el valor predeterminado de la propiedad ImeMode.
ResetRightToLeft	Restablece el valor predeterminado de la propiedad RightToLeft.
ResetText	Restablece el valor predeterminado de la propiedad Text.
ResumeLayout	Sobrecargado. Reanuda la lógica de diseñ o habitual.
Scale	Sobrecargado. Ajusta la escala del control y de todos los controles secundarios.
Select	Sobrecargado. Activa un control.
SelectNextControl	Activa el siguiente control.
SendToBack	Envía el control al final del orden Z.
SetBounds	Sobrecargado. Establece los límites del control.
Show	Muestra el control al usuario.
SuspendLayout	Suspende temporalmente la lógica de diseñ o del control.
ToString (se hereda de Object)	Devuelve un objeto String que representa al objeto Object actual.
Update	Hace que el control vuelva a dibujar las regiones no válidas en su área de cliente.

Eventos públicos

BackColorChanged	Se produce cuando el valor de la propiedad BackColor cambia.
BackgroundImageChanged	Se produce cuando el valor de la propiedad BackgroundImage cambia.
BindingContextChanged	Se produce cuando el valor de la propiedad BindingContext cambia.
CausesValidationChanged	Se produce cuando el valor de la propiedad CausesValidation cambia.
ChangeUICues	Se produce cuando cambian las guías de la interfaz de usuario para el foco o el teclado.
Click	Se produce cuando se hace clic en el control.
ContextMenuChanged	Se produce cuando el valor de la propiedad ContextMenu cambia.
ControlAdded	Se produce cuando se agrega un nuevo control a Control.ControlCollection.
ControlRemoved	Se produce cuando se quita un control de Control.ControlCollection.

CursorChanged	Se produce cuando el valor de la propiedad Cursor cambia.
Disposed (se hereda de Component)	Agrega un controlador de eventos para escuchar al evento Disposed en el componente.
DockChanged	Se produce cuando el valor de la propiedad Dock cambia.
DoubleClick	Se produce cuando se hace doble clic en el control.
DragDrop	Se produce cuando termina una operación de arrastrar y colocar.
DragEnter	Se produce cuando se arrastra un objeto dentro de los límites del control.
DragLeave	Se produce cuando se arrastra un objeto fuera de los límites del control.
DragOver	Se produce cuando se arrastra un objeto sobre los límites del control.
EnabledChanged	Se produce cuando cambia el valor de la propiedad Enabled.
Enter	Se produce cuando se entra en el control.
FontChanged	Se produce cuando cambia el valor de la propiedad Font.
ForeColorChanged	Se produce cuando cambia el valor de la propiedad ForeColor.
GiveFeedback	Se produce durante una operación de arrastre.
GotFocus	Se produce cuando el control recibe el foco.
HandleCreated	Se produce cuando se crea un identificador para el control.
HandleDestroyed	Se produce cuando el identificador del control está en proceso de eliminación .
HelpRequested	Se produce cuando el usuario solicita ayuda para un control.
ImeModeChanged	Se produce cuando cambia la propiedad ImeMode.
Invalidated	Se produce cuando es necesario volver a dibujar un control.
KeyDown	Se produce cuando se presiona una tecla mientras el control tiene el foco.
KeyPress	Se produce cuando se presiona una tecla mientras el control tiene el foco.
KeyUp	Se produce cuando se suelta una tecla mientras el control tiene el foco.
Layout	Se produce cuando un control debe volver

	a colocar sus controles secundarios.
Leave	Se produce cuando el foco de entrada deja el control.
LocationChanged	Se produce cuando cambia el valor de la propiedad Location.
LostFocus	Se produce cuando el control pierde el foco.
MouseDown	Se produce cuando el puntero del mouse está sobre el control y se presiona un botón del mouse.
MouseEnter	Se produce cuando el puntero del mouse entra en el control.
MouseHover	Se produce cuando el puntero del mouse se sitúa encima del control.
MouseLeave	Se produce cuando el puntero del mouse deja el control.
MouseMove	Se produce cuando el puntero del mouse se mueve sobre el control.
MouseUp	Se produce cuando el puntero del mouse está encima del control y se suelta un botón del mouse.
MouseWheel	Se produce cuando la rueda del mouse se mueve mientras el control tiene el foco.
Move	Se produce cuando se mueve el control.
Paint	Se produce cuando vuelve a dibujarse el control.
ParentChanged	Se produce cuando cambia el valor de la propiedad Parent.
QueryAccessibilityHelp	Se produce cuando AccessibleObject proporciona ayuda para aplicaciones de accesibilidad.
QueryContinueDrag	Se produce durante una operación de arrastrar y colocar y permite al origen de arrastre determinar si la operación de arrastrar y colocar tiene que cancelarse.
Resize	Se produce cuando se cambia el tamaño del control.
RightToLeftChanged	Se produce cuando cambia el valor de la propiedad RightToLeft.
SizeChanged	Se produce cuando cambia el valor de la propiedad Size.
StyleChanged	Se produce cuando cambia el estilo del control.
SystemColorsChanged	Se produce cuando se modifican los colores del sistema.
TabIndexChanged	Se produce cuando cambia el valor de la propiedad TabIndex.
TabStopChanged	Se produce cuando cambia el valor de la propiedad TabStop.
TextChanged	Se produce cuando cambia el valor de la propiedad Text.
Validated	Se produce cuando finaliza la validación del control.
Validating	Se produce cuando el control se está validando.
VisibleChanged	Se produce cuando cambia el valor de la propiedad Visible.

Propiedades protegidas

CreateParams	Obtiene los parámetros de creación necesarios cuando se crea el identificador del control.
DefaultImeMode	Obtiene el modo de Editor de métodos de entrada (IME) predeterminado que admite el control.
DefaultSize	Obtiene el tamaño predeterminado del control.
DesignMode (se hereda de Component)	Obtiene un valor que indica si Component está actualmente en modo de diseño.
Events (se hereda de Component)	Obtiene la lista de controladores de eventos asociados a Component.
FontHeight	Obtiene o establece el alto de la fuente del control.
ResizeRedraw	Obtiene o establece un valor que indica si el control vuelve a dibujarse automáticamente cuando cambia de tamaño .
ShowFocusCues	Obtiene un valor que indica si el control debe mostrar rectángulos de foco.
ShowKeyboardCues	Obtiene un valor que indica si el control debe mostrar métodos abreviados de teclado.

Métodos protegidos

AccessibilityNotifyClients	Notifica a las aplicaciones cliente de accesibilidad los objetos AccessibleEvents especificados del control secundario especificado.
CreateAccessibilityInstance	Crea un nuevo objeto de accesibilidad para el control.
CreateControlsInstance	Crea una nueva instancia de la colección de controles para el control.
CreateHandle	Crea un identificador para el control.
DefWndProc	Envía el mensaje especificado al procedimiento de ventana predeterminado.
DestroyHandle	Destruye el identificador asociado a este control.
Dispose	Sobrecargado. Reemplazado. Libera todos los recursos que utiliza Control.
Finalize (se hereda de Component). Reemplazado.	Libera recursos no administrados y realiza otras operaciones de limpieza antes de que el recolector de elementos no utilizados reclame Component. En C# y C++ , los finalizadores se expresan mediante la sintaxis del destructor.
GetService (se hereda de Component)	Devuelve un objeto que representa el servicio suministrado por Component o por Container.
GetStyle	Recupera el valor del bit de estilo de control

GetTopLevel	especificado para el control. Determina si el control es de nivel superior.
InitLayout	Se llama a este método cuando el control se ha agregado a otro contenedor.
InvokeGotFocus	Provoca el evento GotFocus para el control especificado.
InvokeLostFocus	Provoca el evento LostFocus para el control especificado.
InvokeOnClick	Provoca el evento Click para el control especificado.
InvokePaint	Provoca el evento Paint para el control especificado.
InvokePaintBackground	Provoca el evento PaintBackground para el control especificado.
IsInputChar	Determina si un carácter es un carácter de entrada que el control reconoce.
IsInputKey	Determina si la tecla especificada es una tecla de entrada normal o una tecla especial que requiere preprocesamiento.
MemberwiseClone (se hereda de Object)	Crea una copia superficial del objeto Object actual.
OnBackColorChanged	Provoca el evento BackColorChanged.
OnBackgroundImageChanged	Provoca el evento BackgroundImageChanged.
OnBindingContextChanged	Provoca el evento BindingContextChanged.
OnCausesValidationChanged	Provoca el evento CausesValidationChanged.
OnChangeUICues	Provoca el evento ChangeUICues.
OnClick	Provoca el evento Click.
OnContextMenuChanged	Provoca el evento ContextMenuChanged.
OnControlAdded	Provoca el evento ControlAdded.
OnControlRemoved	Provoca el evento ControlRemoved.
OnCreateControl	Provoca el evento CreateControl.
OnCursorChanged	Provoca el evento CursorChanged.
OnDockChanged	Provoca el evento DockChanged.
OnDoubleClick	Provoca el evento DoubleClick.
OnDragDrop	Provoca el evento DragDrop.
OnDragEnter	Provoca el evento DragEnter.
OnDragLeave	Provoca el evento DragLeave.
OnDragOver	Provoca el evento DragOver.
OnEnabledChanged	Provoca el evento EnabledChanged.
OnEnter	Provoca el evento Enter.
OnFontChanged	Provoca el evento FontChanged.
OnForeColorChanged	Provoca el evento ForeColorChanged.

OnGiveFeedback	Provoca el evento GiveFeedback.
OnGotFocus	Provoca el evento GotFocus.
OnHandleCreated	Provoca el evento HandleCreated.
OnHandleDestroyed	Provoca el evento HandleDestroyed.
OnHelpRequested	Provoca el evento HelpRequested.
OnImeModeChanged	Provoca el evento ImeModeChanged.
OnInvalidated	Provoca el evento Invalidated.
OnKeyDown	Provoca el evento KeyDown.
OnKeyPress	Provoca el evento KeyPress.
OnKeyUp	Provoca el evento KeyUp.
OnLayout	Provoca el evento Layout.
OnLeave	Provoca el evento Leave.
OnLocationChanged	Provoca el evento LocationChanged.
OnLostFocus	Provoca el evento LostFocus.
OnMouseDown	Provoca el evento MouseDown.
OnMouseEnter	Provoca el evento MouseEnter.
OnMouseHover	Provoca el evento MouseHover.
OnMouseLeave	Provoca el evento MouseLeave.
OnMouseMove	Provoca el evento MouseMove.
OnMouseUp	Provoca el evento MouseUp.
OnMouseWheel	Provoca el evento MouseWheel.
OnMove	Provoca el evento Move.
OnNotifyMessage	Notifica al control los mensajes de Windows.
OnPaint	Provoca el evento Paint.
OnPaintBackground	Pinta el fondo del control.
OnParentBackColorChanged	Provoca el evento BackColorChanged cuando cambia el valor de la propiedad BackColor del contenedor del control.
OnParentBackgroundImageChanged	Provoca el evento BackgroundImageChanged cuando cambia el valor de la propiedad BackgroundImage del contenedor del control.
OnParentBindingContextChanged	Provoca el evento BindingContextChanged cuando cambia el valor de la propiedad BindingContext del contenedor del control.
OnParentChanged	Provoca el evento ParentChanged.
OnParentEnabledChanged	Provoca el evento EnabledChanged cuando cambia el valor de la propiedad Enabled del contenedor del control.
OnParentFontChanged	Provoca el evento FontChanged cuando cambia el valor de la propiedad

	Font del contenedor del control.
OnParentForeColorChanged	Provoca el evento ForeColorChanged cuando cambia el valor de la propiedad ForeColor del contenedor del control.
OnParentRightToLeftChanged	Provoca el evento RightToLeftChanged cuando cambia el valor de la propiedad RightToLeft del contenedor del control.
OnParentVisibleChanged	Provoca el evento VisibleChanged cuando cambia el valor de la propiedad Visible del contenedor del control.
OnQueryContinueDrag	Provoca el evento QueryContinueDrag.
OnResize	Provoca el evento Resize.
OnRightToLeftChanged	Provoca el evento RightToLeftChanged.
OnSizeChanged	Provoca el evento SizeChanged.
OnStyleChanged	Provoca el evento StyleChanged.
OnSystemColorsChanged	Provoca el evento SystemColorsChanged.
OnTabIndexChanged	Provoca el evento TabIndexChanged.
OnTabStopChanged	Provoca el evento TabStopChanged.
OnTextChanged	Provoca el evento TextChanged.
OnValidated	Provoca el evento Validated.
OnValidating	Provoca el evento Validating.
OnVisibleChanged	Provoca el evento VisibleChanged.
ProcessCmdKey	Procesa una tecla de comando.
ProcessDialogChar	Procesa un carácter de cuadro de diá logo.
ProcessDialogKey	Procesa una tecla de cuadro de diá logo.
ProcessKeyEventArgs	Procesa un mensaje de tecla y genera los eventos de control correspondientes.
ProcessKeyMessage	Procesa un mensaje de teclado.
ProcessKeyPreview	Muestra una vista preliminar de un mensaje del teclado.
ProcessMnemonic	Procesa un carácter de tecla de acceso.
RecreateHandle	Obliga a que se vuelva a crear el identificador del control.
ReflectMessage	Refleja el mensaje especificado en el control que está enlazado al identificador especificado.
RtlTranslateAlignment Sobrecargado.	Convierte la alineación actual en la alineación correspondiente para que admita texto de derecha a izquierda.
RtlTranslateContent	Convierte ContentAlignment que se especifica en ContentAlignment correspondiente para

	que admita texto de derecha a izquierda.
RtlTranslateHorizontal	Convierte HorizontalAlignment que se especifica en HorizontalAlignment correspondiente para que admita texto de derecha a izquierda.
RtlTranslateLeftRight	Convierte LeftRightAlignment que se especifica en LeftRightAlignment correspondiente para que admita texto de derecha a izquierda.
ScaleCore	Realiza la tarea de ajustar la escala de todo el control y de todos los controles secundarios.
Select	Sobrecargado. Activa un control.
SetBoundsCore	Realiza la tarea de configurar los límites especificados de este control.
SetClientSizeCore	Establece el tamaño del área cliente del control.
SetStyle	Establece el bit de estilo especificado en el valor especificado.
SetTopLevel	Establece el control como el control de nivel superior.
SetVisibleCore	Establece el control en el estado de visibilidad especificado.
UpdateBounds	Sobrecargado. Actualiza los límites del control.
UpdateStyles	Obliga a que los estilos asignados vuelvan a aplicarse al control.
UpdateZOrder	Actualiza el control en el orden Z de su control principal.
WndProc	Procesa los mensajes de Windows.

4.5. Diferentes Controles

4.5.1. Cajas de texto (TextBox)

Los cuadros de texto se utilizan para obtener entradas del usuario o para mostrar texto. El control TextBox se utiliza generalmente para el texto que se puede editar, aunque también puede configurarse como control de sólo lectura. Los cuadros de texto pueden mostrar varias líneas, ajustar el texto al tamaño del control y agregar formato básico. El control TextBox proporciona un único estilo de formato para el texto mostrado o escrito en el control. Para mostrar varios tipos de texto con formato, se debe usar el control RichTextBox.

El texto que se muestra en el control se encuentra almacenado en la propie-

dad Text. De forma predeterminada, en un cuadro de texto se puede escribir 2048 caracteres como máximo. Si establece la propiedad MultiLine en true, se podrá escribir un máximo de 32 KB de texto. La propiedad Enabled (True/False) indica si la caja de texto puede responder a la interacción con el usuario)

Eventos:

Lostfocus(Pierde el foco (cursor))
Enter (Cuando se entra en el control)
GotFocus(Cuando recibe el foco)
KeyPress (Cuando se presiona una tecla estando el foco en el control)
TextChanged(Se produce cuando se cambia el valor de la propiedad text)
Leave (Se produce cuando el foco de entrada deja el cursor)
Validating (Se produce cuando se está validando el control)
Validated (Se produce cuando se termina de validar el control)

Los eventos se ejecutan en el siguiente orden:

1. Enter
2. GotFocus
3. Leave
4. Validating
5. Validated
6. LostFocus

Los eventos Validating y validated no tienen sentido si la propiedad CausesValidation tiene el valor false.

4.5.2. Botones (Button)

El control **Button** permite al usuario hacer clic en él para ejecutar una acción . Cuando se hace clic en el botó n, da la sensación de que se ha presionado y soltado. Cada vez que el usuario hace clic en un botó n, se invoca al controlador del evento **Click**. El código se ubica en el controlador del evento **Click** para ejecutar la acción deseada.

El texto que se muestra en el botón se almacena en la propiedad **Text**. Si este texto supera el ancho del botón, se ajustará en la línea siguiente. No obstante, si el control no dispone del alto suficiente, el texto aparecerá cortado.

La propiedad **Text** puede contener una tecla de acceso, que permite al usuario presionar la tecla ALT junto con la tecla de acceso para "hacer clic.en el control. Por ejemplo, para establecer la letra Ïçomo tecla de acceso, debemos escribir "& Imprimir". La propiedad **Font** y la propiedad **TextAlign** controlan la apariencia del texto.

El control **Button** puede mostrar también imágenes, por medio de las propiedades **Image** e **ImageList**.

En cualquier formulario se puede designar un control **Button** como el botón que se utiliza para aceptar (también conocido como botón predeterminado). Siempre que el usuario presione la tecla ENTRAR hará clic en el botón predeterminado, independientemente del control del formulario que tenga el foco. La excepción a esta regla se da cuando el control con el foco es otro botón (en ese caso, se presionará el botón con el foco) o un cuadro de texto de varias líneas o un control personalizado que acapare la tecla ENTRAR.

Para designar el botón Aceptar en el diseñador debemos establecer en la propiedad AcceptButton del formulario el nombre del control Button.

En cualquier formulario Windows Forms se puede designar un control Button como el botón que se utiliza para cancelar. Siempre que el usuario presione la tecla ESC hará clic en el botón para cancelar, independientemente del control del formulario que tenga el foco. Habitualmente, este botón se programa para permitir que el usuario salga rápidamente de una operación sin confirmar ninguna acción .

Para designar el botón para cancelar en el diseñador debemos establecer en la propiedad CancelButton del formulario el nombre del control Button.

Button.PerformClick (Método)

Es posible llamar a este método para provocar el evento Click.
Aceptar.PerformClick()

4.5.3. Cuadros de mensaje (MessageBox.Show)

Muestra un cuadro de mensaje. Es un método sobrecargado, que admite desde un único parámetro hasta siete.

Text Texto que se va a mostrar en el cuadro de mensaje.

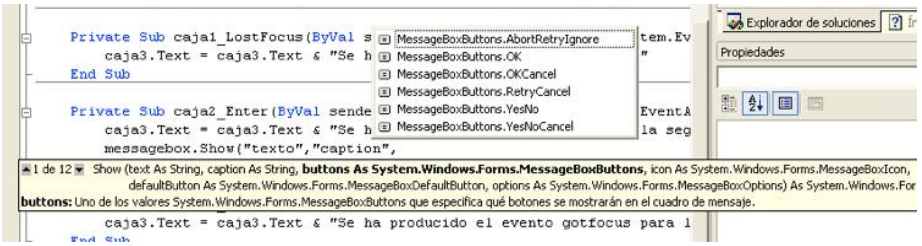
Caption Texto que se va a mostrar en la barra de título.

Buttons Un valor que especifica que botones se mostrarán en el cuadro de mensaje.

Icon Un valor que especifica que icono se mostrará en el cuadro de mensaje.

DefaultButton Un valor que especifica cual es el botón predeterminado.

Options Un valor que especifica las opciones de alineación del texto que mostrará el cuadro de mensaje.



MessageBox devuelve un valor que nos permite saber sobre que botón ha hecho click el usuario.

```
1  Dim ResultadoAs DialogResult
2  Resultado = MessageBox.Show(Me, Message, Caption,
       MessageBoxButtons.YesNo , _
3  MessageBoxIcon.Question ,MessageBoxDefaultButton.Button1,
       MessageBoxOptions.RightAlign)
4
5
6          If Resultado = DialogResult.Yes Then
```

4.5.4. MaskedTextBox

La clase **MaskedTextBox** es un control TextBox mejorado que admite una sintaxis declarativa para aceptar o rechazar los datos proporcionados por el

usuario. Con la propiedad Mask puede especificar la entrada siguiente sin escribir ninguna lógica de validación personalizada en su aplicación :

- Caracteres de entrada necesarios.
- Caracteres de entrada opcionales.
- El tipo de entrada esperada en una posición determinada de la máscara; por ejemplo, un dígito, o un carácter alfabético o alfanumérico.
- Los literales de máscara, o caracteres que deben aparecer directamente en el **MaskedTextBox**; por ejemplo, los guiones (-) en un número de teléfono o el símbolo de moneda en un precio.
- Procesamiento especial para los caracteres de entrada; por ejemplo, para convertir caracteres alfabéticos a mayúsculas.

Cuando se muestra un control **MaskedTextBox** en tiempo de ejecución , representa la máscara como una serie de caracteres de entrada y literales de cadena opcionales. Cada posición modificable de la máscara, que representa una entrada necesaria u opcional, se muestra con un único carácter de entrada. Por ejemplo, el signo de número (#) suele utilizarse como un marcador de posición para una entrada de caracteres numéricos. Se puede utilizar la propiedad PromptChar para especificar un carácter de entrada personalizado. La propiedad HidePromptOnLeave determina si el usuario ve los caracteres de entrada cuando el control pierde el foco de entrada.

A medida que el usuario escribe datos en el cuadro de texto enmascarado, los caracteres de entrada válidos reemplazan sus respectivos caracteres de entrada de modo secuencial. Si el usuario escribe un carácter de entrada no válido, no se realiza ningún reemplazo, sino que se emite un bip si la propiedad BeepOnError se establece en true y se produce el evento MaskInputRejected. Puede proporcionar su propia lógica de error personalizada controlando este evento.

Cuando el punto de inserción actual está en un literal de cadena, el usuario tiene varias opciones:

- Si se escribe un carácter distinto del carácter de entrada, se omitirá automáticamente el literal y el carácter de entrada se aplicará a la siguiente posición modificable, representada por el siguiente carácter de entrada.

- Si se escribe el carácter de entrada y la propiedad AllowPromptAsInput es true, la entrada sobrescribirá el carácter de entrada y el punto de inserción se desplazará a la siguiente posición de la máscara.
- Como siempre, las teclas de dirección pueden utilizarse para desplazarse a una posición anterior o siguiente.

El control **MaskedTextBox** cede realmente todo el procesamiento de la máscara a la clase System.ComponentModel.MaskedTextProvider especificada por la propiedad MaskedTextProvider. Este proveedor está ndar admite todos los caracteres Unicode salvo los suplentes y los caracteres combinados verticalmente; sin embargo, se puede utilizar la propiedad AsciiOnly para restringir la entrada a los conjuntos de caracteres a-z, A-Z y 0-9.

Propiedades útiles

MaskFull Para comprobar si el usuario ha escrito o no todos los datos de entrada necesarios.

Text Siempre recuperará la entrada del usuario con el formato especificado por la máscara y la propiedad TextMaskFormat.

ValidatingType Las máscaras no garantizan necesariamente que la entrada de un usuario representará un valor válido para un tipo determinado; por ejemplo, se puede escribir -9 como una edad en años. Puede comprobar que la entrada de un usuario representa un valor válido asignando una instancia del tipo de ese valor a la propiedad ValidatingType. Puede detectar si el usuario quita el foco de **MaskedTextBox** cuando contiene un valor no válido supervisando el evento TypeValidationCompleted. Si la validación de tipo tiene éxito, el objeto que representa el valor estará disponible a través de la propiedad ReturnValue del parámetro TypeValidationEventArgs.

Como ocurre con el control **TextBox**, varios métodos abreviados de teclado comunes no funcionan con **MaskedTextBox**. En particular, CTRL-R (justificar el texto a la derecha), CTRL-L (justificar el texto a la izquierda) y CTRL-L (centrar el texto) no surten ningún efecto.

Mask Es la propiedad predeterminada para la clase **MaskedTextBox**. Debe ser una cadena formada por uno o más de los elementos de enmascaramiento que se muestran en la tabla siguiente.

Elemento de enmascaramiento	**Descripción**
0	Dígito, necesario. Este elemento aceptará cualquier dígito único entre 0 y 9.
9	Dígito o espacio, opcional.
#	Dígito o espacio, opcional. Si esta posición está en blanco en la máscara, se representará como un espacio en la propiedad Text. Se permiten los signos más (+) y menos (-).
L	Letra, necesaria. Restringe la entrada a las letras ASCII a-z y A-Z. Este elemento de máscara es equivalente a [a-zA-z] en las expresiones regulares.
?	Letra, opcional. Restringe la entrada a las letras ASCII a-z y A-Z. Este elemento de máscara es equivalente a [a-zA-z]? en las expresiones regulares.
&	Carácter, necesaria. Si la propiedad AsciiOnly se establece en true, este elemento se comporta como el elemento ”L”.
C	Carácter, opcional. Cualquier carácter que no sea de control. Si la propiedad AsciiOnly se establece en true, este elemento se comporta como el elemento ?.
A	Alfanumérico, opcional. Si la propiedad AsciiOnly se establece en true, los únicos caracteres que aceptará son las letras ASCII a-z y A-Z.
a	Alfanumérico, opcional. Si la propiedad AsciiOnly se establece en true, los únicos caracteres que aceptará son las letras ASCII a-z y A-Z.
.	Marcador de posición de decimales. El carácter de presentación utilizado será el símbolo de posición de decimales apropiado para el proveedor de formato, según determine la propiedad **FormatProvider** del control.
,	Marcador de posición de miles. El carácter de presentación real utilizado será el marcador de posición de miles apropiado para el proveedor de formato, según determine la propiedad **FormatProvider** del control.
:	Separador de hora. El carácter de presentación utilizado será el símbolo de hora apropiado para el proveedor de formato, según determine la propiedad **FormatProvider** del control.
/	Separador de fecha. El carácter de presentación utilizado será el símbolo de fecha apropiado para el proveedor de formato, según determine la propiedad **FormatProvider** del control.
$	Símbolo de moneda. El carácter real mostrado será el símbolo

	de moneda apropiado para el proveedor de formato, según determine la propiedad **FormatProvider** del control.
<	Minúsculas. Convierte en minúsculas todos los caracteres que hay a continuación .
>	Mayúsculas. Convierte en mayúsculas todos los caracteres que hay a continuación .
|	Deshabilitar un cambio anterior a mayúsculas o minúsculas.
	Escape. Convierte un carácter de la máscara en un literal.
Todos los demás caracteres	Literales. Todos los elementos que no sean de la máscara aparecerán tal cual dentro de **MaskedTextBox**. Los literales siempre ocupan una posición estática en la máscara en tiempo de ejecución y el usuario no los puede desplazar ni eliminar.

Si cambia una máscara cuando **MaskedTextBox** ya contiene datos proporcionados por el usuario filtrados por una máscara anterior, **MaskedTextBox** intentará migrar esa entrada a la nueva definición de máscara. Si se produce un error, borrará la entrada existente. Al asignar como máscara una cadena de longitud cero se conservarán los datos existentes en el control. Cuando se utiliza con una máscara de longitud cero, **MaskedTextBox** se comporta como un control TextBox de una única línea.

Los símbolos de decimales (.), miles (), hora (:), fecha (/) y moneda ($) se muestran de manera predeterminada según define esos símbolos la referencia cultural de la aplicación . Puede forzar que se muestren símbolos para otra referencia cultural si utiliza la propiedad FormatProvider.

La propiedad InsertKeyMode controla la inserción de caracteres en la máscara en tiempo de ejecución . Los usuarios pueden desplazarse por la máscara utilizando las teclas de flecha a la izquierda y flecha a la derecha o el cursor del mouse, y pueden omitir posiciones opcionales de la máscara escribiendo un espacio en blanco.

En la tabla siguiente se muestran algunas máscaras de ejemplo.

Máscara	**Comportamiento**
00/00/0000	Una fecha (día, mes numérico, año) en formato de fecha internacional. El carácter /.es un separador de fecha lógico y aparecerá ante el usuario como el separador de fecha apropiado para la referencia cultural actual de la aplicación .
00->L<LL-0000	Una fecha (día, abreviatura del mes y año) en formato de Estados Unidos, en el que se muestra la abreviatura del mes de tres letras con una letra mayúscula inicial seguida de dos letras minúsculas.

(999)-000-0000	Número de teléfono de Estados Unidos, código de área opcional. Si los usuarios no desean escribir los caracteres opcionales, pueden escribir espacios en blanco o pueden situar el puntero del mouse en la posición de la máscara representada por el primer 0.
$999,999.00	Un valor de moneda en el intervalo de 0 a 999999. Los caracteres de moneda, miles y decimales se reemplazarán en tiempo de ejecución con sus equivalentes específicos de la referencia cultural.

4.5.5. MenuStrip

El componente **MenuStrip** muestra un menú en tiempo de ejecución. Si se agrega este componente, el Diseñador de menús permitirá configurar visualmente la estructura del menú principal. Todos los submenús del menú principal y los elementos individuales son objetos **MenuItem**. Cada objeto **MenuItem** puede ser un comando de la aplicación o un menú primario para otros elementos de submenú. Para enlazar **MenuStrip** con el objeto Form que lo mostrará, asigne MainMenuStrip a la propiedad Menu de Form.

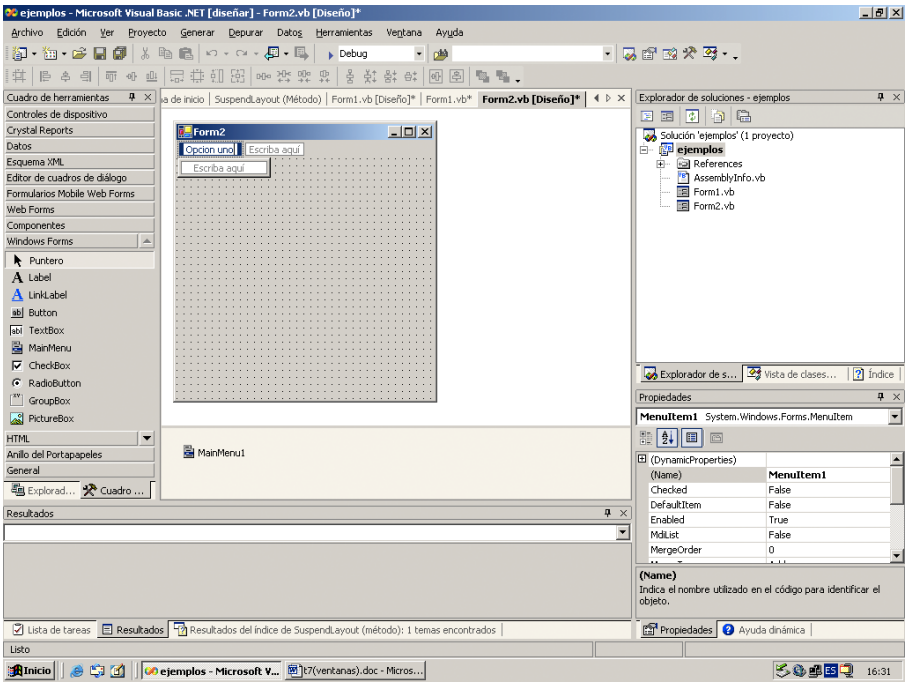
El asignar un menú a un formulario es algo que podemos hacer en diseño o en ejecución.

4.5.6. CheckBox

El control **CheckBox** de los formularios Windows Forms indica si una condición determinada está activada o desactivada. Se utiliza habitualmente para presentar al usuario una selección de tipo Sí/No o Verdadero/Falso. Se pueden utilizar grupos de casillas de verificación para mostrar múltiples opciones entre las cuales el usuario puede elegir una o más. Es similar al control **RadioButton**, aunque en este caso se puede seleccionar cualquier número de controles **CheckBox** agrupados.

El control **CheckBox** tiene dos importantes propiedades: Checked y CheckState. La propiedad Checked devuelve true o false. La propiedad CheckState devuelve CheckState.Checked o CheckState.Unchecked; o bien puede devolver también CheckState.Indeterminate. En el estado indeterminado, el cuadro se muestra atenuado para indicar que la opción no está disponible.

Cada vez que un usuario hace clic en un control **CheckBox** se produce el



evento Click. Debemos programar la aplicación para que ejecute una acción determinada, dependiendo del estado de la casilla de verificación .

4.5.7. RadioButton

Los controles **RadioButton** presentan al usuario un conjunto de dos o más opciones excluyentes entre sí. Aunque puede parecer que los botones de opción y las casillas de verificación funcionan de forma parecida, existe una diferencia importante: cuando un usuario selecciona un botón de opción, no puede seleccionar ninguno de los otros botones de opción del mismo grupo sin perder la selección de este botón.

Cuando se hace clic en un control **RadioButton**, su propiedad Checked se establece en true y se llama al controlador de eventos Click. El evento CheckedChanged se produce cuando cambia el valor de la propiedad Checked. Si la propiedad AutoCheck se establece en true (la opción predeterminada), al seleccionar el botón de opción se desactivarán automáticamente los demás botones de opción del grupo. Normalmente, esta propiedad sólo se establece en false cuando se utiliza código de validación para comprobar que el botón de opción seleccionado corresponde a una opción válida. El texto

que se muestra dentro del control se establece con la propiedad Text, que puede contener teclas de acceso directo. Una tecla de acceso directo permite al usuario "hacer clic.en el control; para ello, debe presionar la tecla ALT junto con la tecla de acceso.

Si se establece la propiedad Appearance en Appearance.Button, el control **RadioButton** puede tener la apariencia de un botó n de comando, que parece estar presionado cuando está seleccionado. Los botones de opción pueden mostrar también imágenes mediante las propiedades Image e ImageList.

Para agrupar botones de opción , debemos dibujarlos dentro de un contenedor como, por ejemplo, un control Panel, un control **GroupBox** o un formulario. Todos los botones de opción que se agregan directamente a un formulario se convierten en un grupo. Para agregar grupos independientes, deberá colocarlos dentro de paneles o cuadros de grupo.

4.5.8. GroupBox

Los controles **GroupBox** se utilizan para proporcionar un agrupamiento identificable para otros controles. Normalmente, los cuadros de grupo se utilizan para subdividir un formulario por funciones. Por ejemplo, podría tener un formulario de pedido que especifique opciones de envío, como el servicio de transporte urgente que se va a utilizar. La agrupación de todas las opciones en un cuadro de grupo ofrece al usuario una pista visual lógica. Los controles **GroupBox** y **Panel** son similares; sin embargo, el control **GroupBox** es el único de los dos que muestra un título y el control Panel es el único de los dos que puede tener barras de desplazamiento.

4.5.9. Picturebox

El control **PictureBox** se utiliza para mostrar gráficos en formato de mapa de bits, GIF, JPEG, metarchivo o icono. La imagen que se muestra está determinada por la propiedad Image, que se puede establecer en tiempo de ejecución o en tiempo de diseñ o. La propiedad SizeMode controla el ajuste entre la imagen y el control. Se puede establecer la propiedad SizeMode para:

- Alinear la esquina superior izquierda de la imagen con la esquina superior izquierda del control.
- Centrar la imagen dentro del control.
- Ajustar el tamaño del control a la imagen que muestra.

- Estirar cualquier imagen que se muestre para que se ajuste al control.

4.5.10. Panel

Los controles **Panel** se utilizan para proporcionar un agrupamiento identificable para otros controles. Normalmente, los paneles se utilizan para subdividir un formulario por funciones. Agrupar todas las opciones en un panel ofrece al usuario una pista visual lógica.

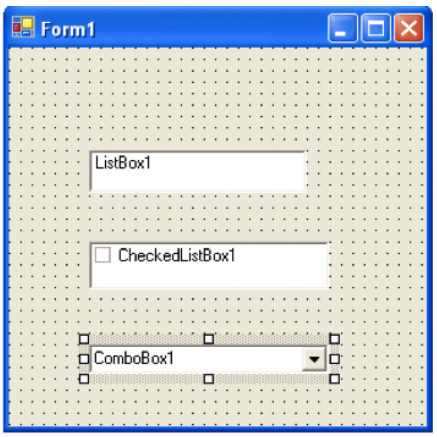
Los controles **Panel** y **GroupBox** son similares; sin embargo, el control **Panel** es el único de los dos que puede tener barras de desplazamiento y el control **GroupBox** es el único de los dos que muestra un título.

Para mostrar barras de desplazamiento, debemos establecer la propiedad AutoScroll en true. Para personalizar la apariencia del panel, debemos utilizar las propiedades BackColor, BackgroundImage y BorderStyle. La propiedad BorderStyle determina si el panel está rodeado por un borde invisible (None), una línea simple (FixedSingle) o una línea sombreada (Fixed3D).

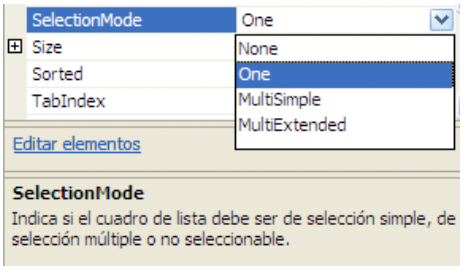
4.5.11. DatagridView

Relacionado con bases de datos.

4.5.12. ListBox, checkedlistbox, combobox



Un control **ListBox** muestra una lista de elementos de los cuales el usuario puede seleccionar uno o más.



Si el número total de elementos supera el número que se puede mostrar, se agrega automáticamente una barra de desplazamiento al control **ListBox**.

Si el número total de elementos supera el número que se puede mostrar, se agrega automáticamente una barra de desplazamiento al control **ListBox**.

Cuando la propiedad ScrollAlwaysVisible se establece en true, la barra de desplazamiento aparece, independientemente del número de elementos. La propiedad SelectionMode determina cuántos elementos de la lista pueden seleccionarse a la vez.

La propiedad SelectedIndex devuelve un valor entero que corresponde al primer elemento seleccionado en el cuadro de lista. Para cambiar mediante programación el elemento seleccionado, debemos cambiar el valor SelectedIndex en el código; el elemento correspondiente de la lista aparecerá resaltado en el formulario. Si no se selecciona ningún elemento, el valor de SelectedIndex es -1. Si se selecciona el primer elemento de la lista, el valor SelectedIndex es 0. Cuando se seleccionan múltiples elementos, el valor SelectedIndex refleja el elemento seleccionado que aparece primero en la lista.

Ejemplo: Listbox1, con la propiedad SelectionMode a MultiSimple.Se puede seleccionar mas de una linea.

```
1   Dim cadena As String
2         Dim l As Integer = Me.ListBox1.SelectedIndices.Count()
3         For i As Integer = 0 To ListBox1.SelectedIndices.Count() - 1
4             cadena = cadena & "" & ListBox1.SelectedIndices.Item(i)
5         Next
6         MessageBox.Show(l & "seleccionados: " & cadena)
```

En este ejemplo se recorre las líneas seleccionadas en listBox1 y se muestra un MessageBox con numero de líneas seleccionadas y la lista de índices de las fila seleccionada.

La propiedad SelectedItem es similar a SelectedIndex, pero devuelve el elemento en sí, habitualmente un valor de cadena.

```
1    Dim v As CVivienda
2          For i As Integer = 0 To ListBox1.SelectedItems.Count() - 1
3              v = ListBox1.SelectedItems.Item(i)
4              MessageBox.Show("Seleccionada: " & v.ToString)
5          Next
```

Para agregar o eliminar elementos de un control **ListBox**, se utilizan los métodos Items.Add, Items.Clear o Items.Remove. También podemos agregar elementos a la lista mediante la propiedad Items en tiempo de diseño.

```
1  For x = 1 To 50
2    listBox1.Items.Add("Item " & x.ToString())
3  Next x
```