Tema 7: Controles III

- Cuadros de dialogo comunes.
- La clase DateTimePicker
- La clase MonthCalendar.
- La clase Timer.
- La clase ToolTip.
- La clase TabControl
- La clase TreeView.
- La clase ListView
- La clase ImageList
- La clase ToolStrip
- La clase StatusStrip

Cuadros de diálogo comunes

- Windows ofrece una serie de cuadros de diálogos comunes que puede utilizar cualquier aplicación Windows.
- Desde .NET es posible acceder a los siguientes cuadros de diálogos:
  - Abrir archivo (clase OpenFileDialog).
  - Seleccionar carpeta (clase FolderBrowserDialog).
  - Guardar como (clase SaveFileDialog).
  - Colores (clase ColorDialog).
  - Fuentes (clase FontDialog).
  - Imprimir (clase PrintDialog).
  - Configurar página (clase PageSetupDialog).
  - Vista previa (clase PrintPreviewDialog).
- Todas las clases tienen los siguientes miembros comunes:
  - Método ShowDialog(). Muestra el cuadro y devuelve DialogResult.Ok o DialogResult.Cancel dependiendo del botón que pulse el usuario.
  - Método Reset() que restaura todas las propiedades a su valor por omisión.
- Los cuadros de diálogos no realizan por si ninguna acción, sólo sirven para seleccionar valores mediante propiedades, que se utilizarán más, tarde.

Cuadro de diálogo OpenFileDialog

- Muestra el típico cuadro de diálogo para seleccionar uno o varios archivos.
  - Se utiliza para seleccionar un archivo.
  - En principio, aparecen todos los archivos de la carpeta inicial o la que indique la propiedad InitialDirectory.
    - ✓ La propiedad Title permite especificar el título del cuadro de diálogo, por omisión aparecerá la cadena “Abrir”.

- Recuperar el archivo seleccionado.
  - La propiedad FileName guarda el nombre del archivo seleccionado.
  - Si se escribe directamente el nombre del archivo en el cuadro de texto correspondiente es posible detectar si el archivo o la carpeta existe mediante las propiedades lógicas CheckFileExist y CheckPathExist.
    - ✓ Si están a True, se detectará si existe ese nombre de archivo o carpeta al pulsar el botón Aceptar del cuadro de diálogo.

Cuadro de diálogo OpenFileDialog (II)

- Filtrar los archivos seleccionados.

  - La propiedad Filter permite añadir elementos a la lista de tipos de archivos.
    - ✓ El valor de esa propiedad será una cadena con el siguiente formato:
      - Texto1|filtro1|Texto2|filtro2….
      - Para que aparezcan todos los archivos, archivos txt o algunos archivos gráficos:

```
OpenFileDialog1.Filter = "Todos los archivos (*.*) |*.*|" & _
                         "Archivos de texto (*.txt) |*.txt| " & _
                         "Archivos gráficos |*.gif;*.bmp;*.jpg“
```

  - La propiedad FilterIndex permite decidir mediante un número entero el índice del tipo de archivo que aparecerá inicialmente.
    - ✓ La propiedad ShowReadOnly es un valor lógico que permite que aparezca la casilla de verificación “Abrir como sólo lectura”.
  - La propiedad ReadOnlyChecked permitirá recuperar el valor introducido por el usuario en esa casilla para su posterior proceso.

Cuadro de diálogo OpenFileDialog (III)

- Selección múltiple.
  - La propiedad Multiselect permite seleccionar múltiples archivos del cuadro de diálogo.
  - En este caso los archivos seleccionados se cargan en un array de cadena representado por la propiedad FileNames.
  - Ejemplo: carga en un ListBox nombres de los archivos seleccionados.

```
If OpenFileDialog1.ShowDialog() = Windows.Forms.DialogResult.OK Then
        'Recorre todos los elementos del array FileNames
        For Each str As String In OpenFileDialog1.FileNames
                'FileNames contiene la especificación completa del archivo
                'Para obtener sólo el nombre, troceo la cadena con Split
                Dim aux() As String = str.Split("\")
                'y me quedo con el último elemento del array resultante
                ListBox1.Items.Add(aux(aux.GetUpperBound(0)))
        Next
End If
```

Cuadro de diálogo OpenFileDialog (IV)

- Ejemplo: seleccionar un archivo gráfico y cargarlo en un control PictureBox.


```
Private Sub Cargar_Click (ByVal sender As System.Object, _
                          ByVal e As System.EventArgs) Handles Cargar.Click
    OpenFileDialog1.Filter = "Todos los archivos (*.*)|*.*|" & _
                             "Archivos gráficos|*.bmp;*.gif;*.jpg;*.png"
    OpenFileDialog1.FilterIndex = 1
    If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
        PictureBox1.Image = Image.FromFile(OpenFileDialog1.FileName)
    Else
        PictureBox1.Image = Nothing
    End If
End Sub
```

Cuadro de diálogo FolderBrowserDialog

- Permite seleccionar una carpeta a partir de un cuadro de diálogo estándar.
- La propiedad SelectedPath devuelve el nombre de la carpeta seleccionada.
- Es posible activar u ocultar el botón “Nueva carpeta” mediante la propiedad ShowNewFolderButton.
- La carpeta inicial será la que indique la propiedad RootFolder.
  - Puede tomar alguno de los valores de la enumeración Environment.SpecialFolder:
    - ✓ Desktop. El escritorio.
    - ✓ MyComputer. Mi PC.
    - ✓ Personal. Mis documentos.
    - ✓ MyMusic. Mi música.
    - ✓ MyPictures. Mis imágenes.
    - ✓ ProgramFiles. Archivos de programa

Cuadro de diálogo SaveFileDialog

- Similar al cuadro Abrir archivo pero con distinta funcionalidad.
  - Se utiliza para dar un nombre a un archivo.
- Mantiene las propiedades Title, InitialDirectory,Filename, Filter, FilterIndex.
- No admite multiselección.
  - La propiedad Filenames pierde su sentido.
- Las propiedades CheckFileExist, CheckPathExist, ShowReadOnly y ReadOnlyCheck existen, pero pierden su utilidad.
- Como nuevas propiedades tiene CreatePrompt y OverwritePrompt que avisan si se va a crear un archivo nuevo o se va a sobreescribir un archivo.
- La propiedad DefaultExt permite incluir una cadena para la extensión por omisión del archivo.

Cuadro de diálogo ColorDialog

- Permite seleccionar un color de la paleta.
- Propiedad FullOpen.
  - Con un valor True, el cuadro de diálogo se abre con la paleta de colores personalizados.
- Propiedad AllowFullOpen.
  - Con un valor True, permite a los usuarios elegir un color personalizado.
- La propiedad Color permite devuelve el color seleccionado al pulsar el botón Aceptar.

```
If ColorDialog1.ShowDialog() = DialogResult.OK Then
        Button5 ColorDialog1 Button5.BackColor =
        ColorDialog1.Color
End If
```

Cuadro de diálogo FontDialog

❑ Permite seleccionar una fuente cuyas características devolverá en la propiedad Font que se puede asignar a la propiedad Font de cualquier clase que disponga de ella.

```
If FontDialog1.ShowDialog() = DialogResult.OK Then
Label1 FontDialog1 Label1.Font = FontDialog1.Font
End If
```

❑ En principio aparecen todas las fuentes, pero podemos seleccionar las deseadas mediante un valor lógico en las propiedades:

- AllowVectorFonts. Admite o no fuentes vectoriales.
- AllowVerticalFonts. Admite o no fuentes verticales.
- FixedPichOnly. Admite sólo fuentes de paso fijo.

✓ Las propiedades MaxSize y MinSize permiten definir el tamaño máximo y mínimo de la lista de tamaños de fuentes.

Cuadro de diálogo FontDialog (II)

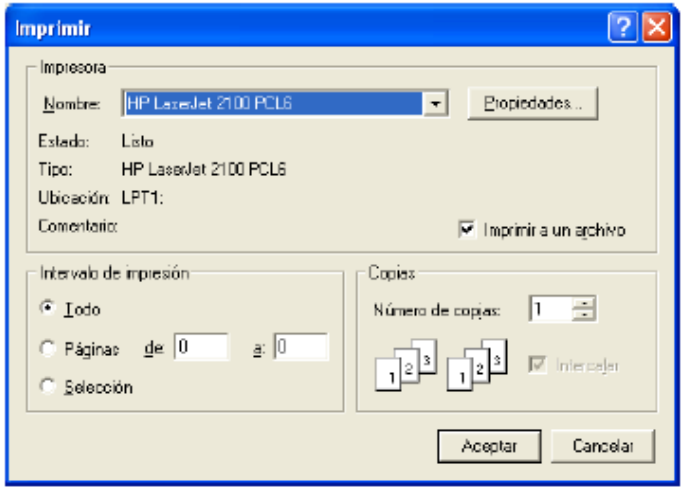
- Características especiales.
  - La propiedad ShowEffects permite mostrar las casillas de verificación de subrayado y tachado.
  - La propiedad ShowColor permite mostar y seleccionar de una lista de colores.
    - ✓ El color seleccionado será recogido en la propiedad Color que será necesario asignar independientemente de la fuente seleccionada.

      Label1.ForeColor = FontDialog1.Color
- La propiedad ShowApply permite mostrar en el cuadro de diálogo el botón aplicar.
  - El evento Apply permitirá controlar si se ha pulsado dicho botón y, mediante código, mostrar una vista previa de la nueva fuente.

```
Dim fuente As Font = Label1.Font 'Para poder recuperar la fuente si se pulsa Cancelar
If FontDialog1.ShowDialog() = DialogResult.OK Then
        Label1.Font = FontDialog1.Font
Else
        Label1.Font = fuente 'Si se pulsa Cancelar se vuelve a la fuente original
End If

Private Sub FontDialog1_Apply (ByVal sender As System.Object, ByVal e As System EventArgs) Handles FontDialog1.Apply
        Label1.Font = FontDialog1.Font
End Sub
```

Cuadro de diálogo PrintDialog

- Permite especificar las propiedades de un objeto PrinterSettings,necesario para realizar una impresión.
- Especificar que propiedades podemos cambiar:
  - AllowPrintToFile. Habilita o deshabilita la casilla de verificación Imprimir a un archivo.
  - AllowSelection. Habilita o deshabilita el botón de radio Selección.
  - AllowSomePages. Habilita o deshabilita el botón de radio Páginas para imprimir sólo un intervalo de páginas.



- PrintToFile. Devuelve un valor lógico si está marcada la casilla de verificación.
- La propiedad PrinterSettings devuelve el objeto PrinterSettings modificado por el cuadro de diálogo.

Cuadro de diálogo PrintDialog (II)

- Propiedades del objeto PrinterSettings.
  - PrinterName. Nombre de la impresora seleccionada.
  - Copies. El número de copias marcado en el cuadro número de copias.
  - Collate. Devuelve o establece si se ha marcado la casilla de verificación intercalar.
  - PrintRange. Devuelve o establece el botón de radio marcado en el intervalo de impresión.
    - ✓ Puede tomar alguno de los siguientes valores:
      - PrintRange.AllPages.
      - PrintRange.Selection.
      - PrintRange.SomePages.
  - FromPage. Devuelve o establece la página inicial.
  - ToPage. Devuelve o establece la página final.

Introducción a la impresión

❑ Para poder trabajar con las clases de impresión es necesario importar el espacio de nombres al comienzo del código: Imports System.drawing.printing

❑ Para realizar la impresión es preciso seguir los siguientes pasos:

- Crear y definir un objeto PrintDocument.
  - ✓ Se puede crear arrastrándolo desde la caja de herramientas a la bandeja de componentes o declarando una variable PrintDocument a nivel de módulo.
- Crear y definir un objeto PrinterSettings.
  - ✓ Se puede realizar a partir de la propiedad PrinterSettings del cuadro de diálogo PrintDialog.
  - ✓ Las especificaciones de impresión del objeto PrintDocument se tomarán a partir de este objeto PrinterSettings.
  - ✓ El método Print de la clase PrinterSettings comenzará la impresión.

```
If PrintDialog1.ShowDialog() = DialogResult.OK Then
        PrintDocument1.PrinterSettings = PrintDialog1.PrinterSettings
        PrintDocument1.Print()
End If
```

Introducción a la impresión (II)

- El contenido de lo que se va a imprimir se debe realizar dentro del evento PrintPage de la clase PrintDocument.

```
Private Sub PrintDocument1_PrintPage (ByVal sender As System.Object, _
                                      ByVal e As System.Drawing.Printing.PrintPageEventArgs) _
                                      Handles PrintDocument1.PrintPage
...
End Sub
```

- PrintPage recibe un argumento de la clase PrintPageEventArgs que se utiliza para tomar las características de la página y el objeto Graphics que se imprimirá.
  - e.MarginBounds. Rectángulo con los márgenes de la página.
  - e.PageBounds. Rectángulo con los límites físicos de la página.
  - e.HasMorePages. Valor lógico que indica si hay o no más páginas a imprimir.
  - e.PageSettings. Configuración de la página para la página actual.

Introducción a la impresión (III)

❑ e.Graphics definir lo que se va a imprimir con los métodos DrawXXX de la clase Graphics

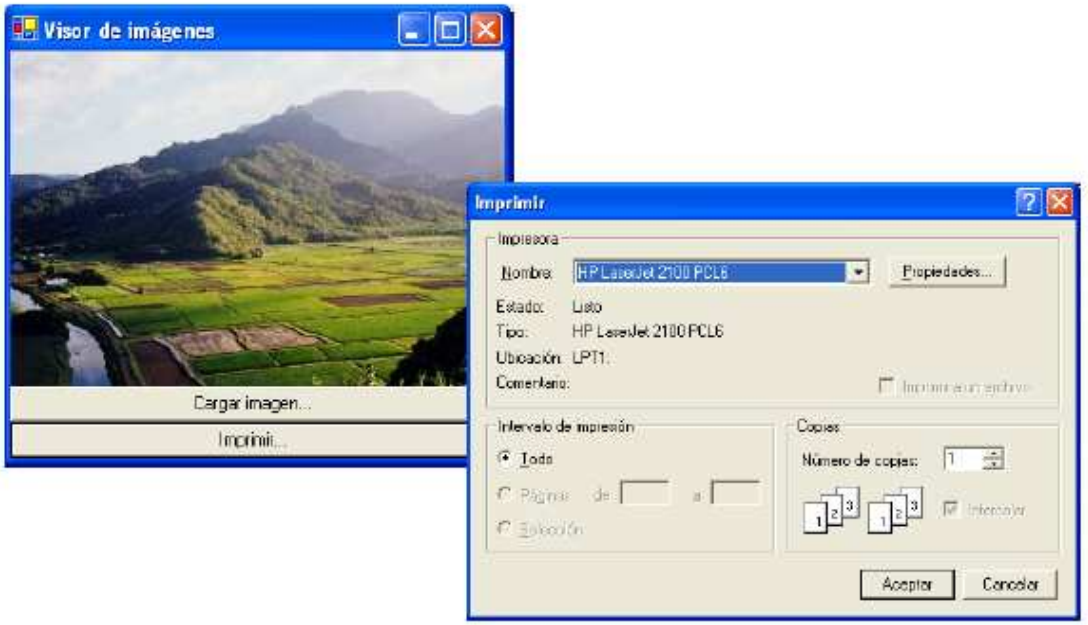
- e.Graphics.DrawString(*cadena*,*fuente*,*pincel*,*X*,*Y*).
  - ✓ Imprime la cadena con la fuente seleccionada y el color del pincel seleccionado en la posición de la página X,Y.
- e.Graphics.DrawImage(image,X,Y,ancho,alto).
  - ✓ Imprime el objeto Image especificado en la posición X e Y de la página con un ancho y alto específico.

```
'Imprime el contenido de TextBox1
Private Sub PrintDocument1_PrintPage (ByVal sender As System.Object, _
                                      ByVal e As System.Drawing.Printing.PrintPageEventArgs)_
                                      Handles PrintDocument1.PrintPage
    Dim margenIzq As Single = e.MarginBounds.Left
    Dim margenSup As Single = e.MarginBounds.Top
    e.Graphics.DrawString (TextBox1.Text, Me.Font, Brushes.Black, margenIzq, margenSup)
End Sub
```

✓ En http://msdn.microsoft.com/en-us/library/xdt36c58.aspx se puede encontrar información adicional sobre la impresión impresión.

Introducción a la impresión (IV)

- Ejemplo: añadir al ejemplo de carga de una imagen un botón para imprimir la imagen.

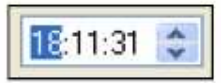


Introducción a la impresión (V)

```
Imports System.Drawing.Printing
...
Private Sub Imprimir_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Imprimir.Click
        PrintDialog1.AllowPrintToFile = False
        PrintDialog1.AllowSelection = False
        PrintDialog1.AllowSomePages = False
        PrintDialog1.PrinterSettings = New PrinterSettings
        If PrintDialog1.ShowDialog() = DialogResult.OK Then
                PrintDocument1.PrinterSettings = PrintDialog1.PrinterSettings
                Try
                        PrintDocument1.Print()
                Catch ex As Exception
                        MessageBox.Show ("Error de impresión", "Imprimir imagen", _
                                                MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
                End Try
        End If
End Sub

Private Sub PrintDocument1_PrintPage (ByVal sender As System.Object, _
                                                ByVal e As System.Drawing.Printing.PrintPageEventArgs) _
                                                Handles PrintDocument1.PrintPage
        e.Graphics.DrawImage (PictureBox1.Image, e.MarginBounds.Left, e.MarginBounds.Top, _
                                        e.MarginBounds.Width, PictureBox1.Image.Height)
End Sub
```

Clase DateTimePicker

- Proporciona un mecanismo para la introducción de valores de tipo DateTime.



  - Presenta el aspecto de una lista deplegable que se despliega como un calendario.
  - Es posible limitar las fechas a visualizar mediante las propiedades MaxDate y MinDate.
- Obtener el valor.
  - La propiedad Text obtiene o establece una cadena con el valor que aparece en el control.
  - La propiedad Value obtiene o establece un valor de tipo DateTime.
  - La propiedades Day, Month, Year, DayOfWeek, Hour, Minute, Second y Millisecond devuelven las partes de la fecha.
- Formato del control.
  - La propiedad Format permite establecer el formato de fecha y hora que se visualiza.
  - Es posible mostrar un control para seleccionar horas marcando la propiedad ShowUpDown a True y Format a la propiedad Time.

Clase MonthCalendar

- Muestra una interfaz gráfica en forma de calendario mediante la que el usuario puede manejar información relativa a fechas.
  - El número de meses que aparece se puede modificar mediante la propiedad CalendarDimensions.

- Permite obtener o establecer rangos de fechas.
  - La propiedad MaxSelectionCount permite determinar el número máximo de días seleccionados.
  - La propiedades SelectionStart y SelectionEnd obtienen o establecen la fecha de inicio y fin.
  - La propiedad SelectionRange establece o devuelve un objeto de tipo SelectionRange que contiene dos fechas con el inicio y el fin del periodo.

```
'Selecciona el día actual y los tres siguientes y muestra las fechas de inicio y fin en etiquetas
MonthCalendar1.SelectionRange =New SelectionRange(Now(), Now().AddDays(3))
Label1.Text = MonthCalendar1.SelectionRange.Start
Label2.Text = MonthCalendar1.SelectionRange.End
```

Clase Timer

- Implementa un temporizador que produce un evento en los intervalos fijados por el programador.
- El evento Tick, se producirá cada vez que se cumple el intervalo previsto por el programador y el control está activado.
- La propiedad Interval permite fijar en milisegundos el intervalo de tiempo.
- La propiedad Enabled admite un valor lógico que permite activar o desactivar el temporizador.

```
'Muestra en Label1 un reloj que se actualiza cada milisegundo
Private Sub Timer1_Tick (ByVal sender As Object, ByVal e As System.EventArgs) _
                                Handles Timer1.Tick
        'Label1 muestra la hora del sistema actualizada
        'La propiedad Interval se debe establecer a 1000
        'La propiedad Enabled se debe establecer a True
        Label1 DateTime Now ToString("hh:mm:ss")
        Label1.Text = DateTime.Now.ToString( ss )
End Sub
```

Clase ToolTip

- Muestra texto de ayuda cuando el curso se para sobre el control.

- El control ToolTip se coloca en la bandeja de componentes.
  - Se puede usar un único control ToolTip para todos los componentes del formulario.

    ✓ En tiempo de diseño, para asociar un control con un ToolTip, en la ventana de propiedades **del control al que se quiere añadir el texto** se introduce el texto en “ToolTip en ToolTip1”.

    ✓ En tiempo de ejecución, se puede hacer mediante el método SetToolTip. ToolTip1.SetToolTip(Button1, "Guardar cambios")

Clase ToolTip (II)

- Propiedades que controlan el tiempo de retardo del control.
  - InitialDelay, tiempo en milisegundos que el usuario debe apuntar al control asociado para que aparezca la información del control.
  - ReshowDelay, tiempo que tarda en aparecer el texto cuando el ratón de mueve desde un control asociado a otro.
  - AutoPopDelay, tiempo durante el cual se muestra la información del control asociado.
  - AutomaticDelay, permite establecer las demás propiedades en función del valor asignado a esta propiedad:
    - ✓ Si AutomaticDelay tiene el valor N…
      InitialDelay se establece en N, ReshowDelay se establece en N/5 y AutoPopDelay se establece en 5N.

Clase TabControl

❑ Muestra un formulario con múltiples fichas similares a las pestañas de las carpetas.

- Cada ficha puede tener varios controles.
- Se utilizan para cuadros de diálogo con varias páginas (por ejemplo para páginas de propiedades).

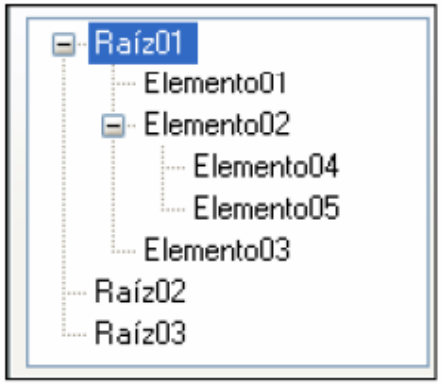
❑ La propiedad TabPages hace referencia a una colección con las fichas.

- Cada ficha es un objeto de tipo TabPages.
  - ✓ Cada objeto TabPage es un contenedor de otros controles.
- Se pueden añadir en tiempo de diseño o en tiempo de ejecución con el método Add.

```
TabControl1.TabPages.Add(New TabPage("Otra ficha"))
```

Clase TreeView

- Muestra un conjunto de elementos organizado de forma jerárquica.
- Cada elemento es un objeto de la clase TreeNode a la que se accede por medio de la propiedad Nodes del elemento.
  - La propiedad Nodes del control TreeView contendría una colección de nodos de tipo nodo raíz.
  - Cada nodo sería un objeto de la clase TreeNode que a su vez tiene una colección de nodos representada también por su propiedad Nodes.
    - ✓ Cada nodo permite tener asociado un icono con la propiedad ImageKey.
      - ImageKey es un número que hacer referencia a una imagen almacenada en un control ImageList al que se referencia por la propiedad ImageList.



Colección Nodes de TreeView:
- Raíz01, Raíz02 y Raíz03

Colección Nodes de Raíz01:
- Elemento01, Elemento02, Elemento03.

Colección Nodes de Elemento02:
- Elemento04, Elemento05.

Clase TreeView (II)

- Añadir nodos mediante programación.
  - El método Add de la colección Nodes permite añadir nodos.
    - ✓ Recibe como argumento una cadena con el nombre de la etiqueta o un objeto de la clase TreeNode.
      - Añade un nodo raíz al objeto TreeView.

```
TreeView1.SelectedNode.Nodes.Add("Nuevo nodo")
```

      - Añade un nodo al nodo seleccionado.

```
Dim miNodo As New TreeNode("Nuevo nodo")
'La propiedad SelectedNode referencia al nodo seleccionado
TreeView1.SelectedNode.Nodes.Add(miNodo)
```

- Eliminar nodos mediante programación.
  - El método Remove de la clase TreeNode elimina el nodo y sus nodos hijos.
- Contenido de un nodo.
  - La propiedad Text de la clase TreeView muestra la etiqueta del nodo seleccionado.
  - La propiedad Text de la clase TreeNode muestra la etiqueta del nodo.
  - La propiedad FullPath de la clase TreeNode devuelve una cadena con la ruta de acceso del nodo

Clase TreeView (III)

- Ejemplo: Añadir y eliminar nodos mediante programación.
  - El botón Añadir insertará un nuevo nodo a partir del nodo seleccionado con la etiqueta del cuadro de texto.
    - ✓ Si la casilla de verificación Nodo raíz está activada, lo añadirá como nodo raíz.
  - El botón Eliminar quitará el nodo seleccionado y todos sus hijos.
    - ✓ Si el nodo tiene hijos, un cuadro de mensaje informará de la incidencia y permitirá cancelar la operación.

Clase TreeView (IV)

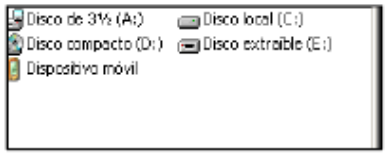
```
Private Sub Button1_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        If (Not CheckBox1.Checked) And (Not TreeView1.S SelectedNode Is Nothing) Then
                TreeView1.SelectedNode.Nodes.Add(TextBox1.Text)
        Else
                TreeView1.Nodes.Add(TextBox1.Text)
        End If
End Sub

Private Sub CheckBox1_CheckedChanged (ByVal sender As System.Object, ByVal e As System.EventArgs)_
                                                Handles CheckBox1.CheckedChanged
        If CheckBox1.Checked Then
                TreeView1.SelectedNode = Nothing
        End If
        End Sub

Private Sub Button2_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        If TreeView1.SelectedNode.GetNodeCount(True) <> 0 Then
                If MessageBox.Show ("El nodo tiene hijos ¿Desea continuar?", _
                                        "Eliminar un nodo",MessageBoxButtons.YesNo, MessageBoxIcon.Question, _
                                         MessageBoxDefaultButton.Button2)=Windows.Forms.DialogResult.Yes Then
                        TreeView1.SelectedNode.Remove()
                End If
        Else
                TreeView1.SelectedNode.Remove()
        End If
End Sub
```

Clase ListView

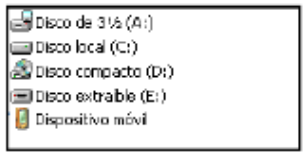
- Muestra una lista de elementos con iconos o en columnas.

- Puede mostrar cuatro vistas mediante los valores de la propiedad View:
  - LargeIcon.
  - SmallIcon.
  - List.
  - Details.

LargeIcon



SmallIcon



List

Details

Clase ListView (II)

- Agregar elementos.
  - Cada elemento es un dato de tipo ListViewItem.
  - Se pueden agregar en tiempo de diseño mediante el editor de la propiedad Items de la ventana de propiedades.
    - ✓ Items hace referencia a una colección de ListViewItems.
  - En tiempo de ejecución se añaden con el método Add de la colección ListViewItemsCollection.

```
'Añade un elemento con la etiqueta "Elemento 1"
ListView1.Items.Add("Elemento 1")
‘Añade un nuevo elemento miItem
Dim miItem As New ListViewItem("Elemento 1")
ListView1.Items.Add(miItem)
'Añade un nuevo elemento "Elemento 1", con el primer icono de la lista de imágenes.
ListView1.Items.Add("Elemento 2", 0)
```

Clase ListView (III)

- Agregar elementos en columnas.
  - El formato tabular sólo está disponible con la propiedad View establecida a Details.
  - Es necesario crear las colunmas mediante el editor de columnas al que se accede mediante la propiedad Columns.
  - El contenido de la primera columna corresponde al elemento ListViewItem.
    - ✓ Cada columna siguiente es un elemento de la colección SubItems de la clase ListViewItem.
      - Se puede contruir un ListViewItem con sus elementos a partir de un array de cadenas.

```
Dim items() As String = New String() {TextBox1.Text, TextBox2.Text, TextBox3.Text}
ListView1.Items.Add(New ListViewItem(items))
```

Clase ListView (IV)

- Referencia a los elementos.
  - Propiedad SelectedIndices. Devuelve una colección con los índices seleccionados.
    - ✓ Hay que tener en cuenta que se pueden seleccionar varios elementos.
      - ListView1.SelectedIndices(0).Item, devuelve un entero con el índice el primer elemento seleccionado.
  - Propiedad SelectedItems. Devuelve una colección de ListViewItem con los elementos seleccionados.
    - ListView1.SelectedItems(0).Text, devuelve la etiqueta del primer elemento seleccionado.
  - Propiedad FocusedItem. Devuelve el ListViewItem que ha recibido el foco.
- Referencia a los subelementos.
  - Se realiza a partir de la colección SubItems del elemento.
    - ListView1.SelectedItems(0).SubItems(0).Text, devuelve la etiqueta de l a primer subelemento de la fila seleccionada.

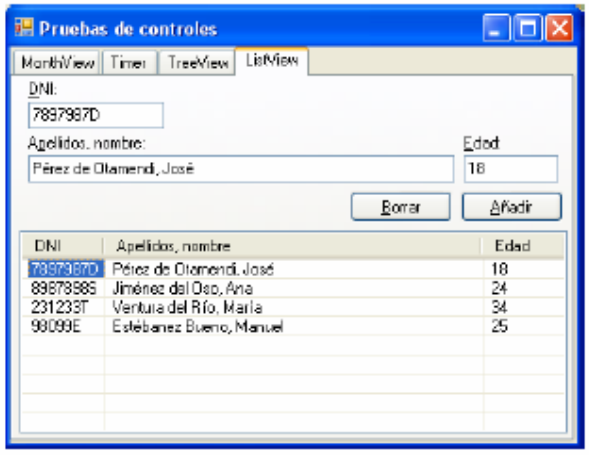
Clase ListView (V)

- Control del elemento seleccionado.
  - Evento ItemActivate. Se produce cuando se activa un elemento.
  - Evento SelectedIndexChanged. Se produce cuando se cambia el índice del elemento seleccionado.
    - ✓ Se produce antes de ItemActivate.
- Eliminación de elementos.
  - Método RemoveAt de la colección ListViewItemsCollection, indicándole el índice del elemento a borrar.
    - ListView1.Items.RemoveAt(ListView1.SelectedIndices(0))
  - Método Remove de la colección ListViewItemsCollection, indicándole el elemento a borrar.
    - ListView1.Items.Remove(ListView1.FocusedItem)

Clase ListView (VI)

❑ Ejemplo: Almacenar datos de personas en un control ListView.

- Al pulsar el botón Añadir, se almacenarán los datos de los cuadros de texto en el ListView.
- Al pulsar el botón Borrar, se eliminará el elemento seleccionado.
- Al pulsar sobre un elemento del ListView aparecerán sus datos en los cuadros de texto.



Clase ListView (VII)

```
Private Sub Button1_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) _
                                Handles Button1.Click
        Dim items() As String = New String() {TextBox1.Text, TextBox2.Text, TextBox3.Text}
        ListView1.Items.Add(New ListViewItem(items))
End Sub

Private Sub Button2_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) _
                                Handles Button2.Click
        ListView1.Items.Remove(ListView1.FocusedItem)
End Sub

Private Sub ListView1_SelectedIndexChanged (ByVal sender As System.Object,
                                                ByVal e As System.EventArgs)_
                                                Handles ListView1.ItemActivate
        Dim item As ListViewItem
        item = ListView1.SelectedItems(0)
        TextBox1.Text = item.Text
        TextBox2.Text = item.SubItems(1).Text
        TextBox3.Text = item.SubItems(2).Text
End Sub
```

Clase ImageList

- Se utiliza como contenedor de imágenes que podrán ser utilizadas por otros controles a partir del índice de cada imagen.

- Se puede hacer referencia a un ImageList en los controles ListView, TreeView, TabControl, Button, CheckBox, RadioButton y Label.

  - En el control ListView la referencia a la lista de imágenes se hace mediante las propiedades LargeImageList y SmallImageList.

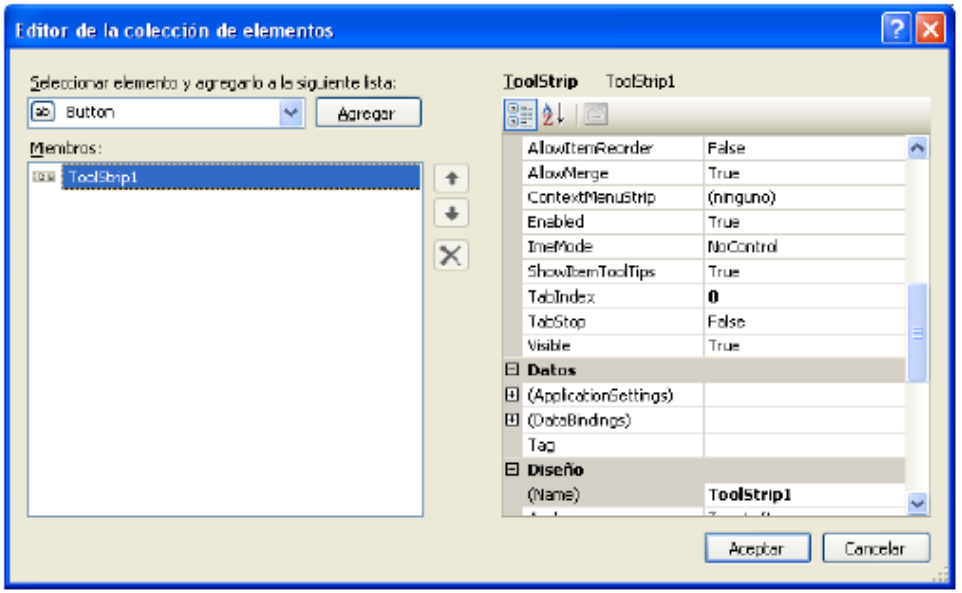
  - En el resto, a partir de la propiedad ImageList.

    - ✓ La imágenes se gestionan en tiempo de diseño mediante el "Editor de la colección Images" accesible por la propiedad Images del ImageList.

    - ✓ La referencia a una imagen concreta de la selección se hace en cada control por medio de la propiedad ImageIndex.

Clase ToolStrip

- Crea una barra de herramientas que puede contener botones, cuadros de texto, botones desplegables, ComboBox, etiquetas o separadores.
  - Para agregar elementos en tiempo de diseño a la barra utiliza la colección Items de la ventana de propiedades.



Clase ToolStrip (II)

- Elementos de ToolStrip.
  - Button (clase ToolStripButton).
    - ✓ Mediante la propiedad DisplayStyle podemos asociarle una imagen (propiedad Image), un text (propiedad Text) o imagen y texto.
    - ✓ La propiedad ToolTipText permite asociarle un texto con información de la herramienta.
    - ✓ El evento Click del control ToolStripButton permite controlar su comportamiento.
  - Label (clase ToolStripLabel).
    - ✓ Representa texto y/o imágenes no seleccionables.
    - ✓ Presenta características similares al control Label.
  - SplitButton (clase ToolStripSplitButon).
    - ✓ Combina un botón con un menú desplegable con las mismas posibilidades que la clase ToolStripMenuItem, con las características ya apuntadas en el apartado de menús.
    - ✓ Registra las acciones, tanto al pulsar sobre el botón, como al desplegar la opciones lista y seleccionar las opciones.

Clase ToolStrip (III)

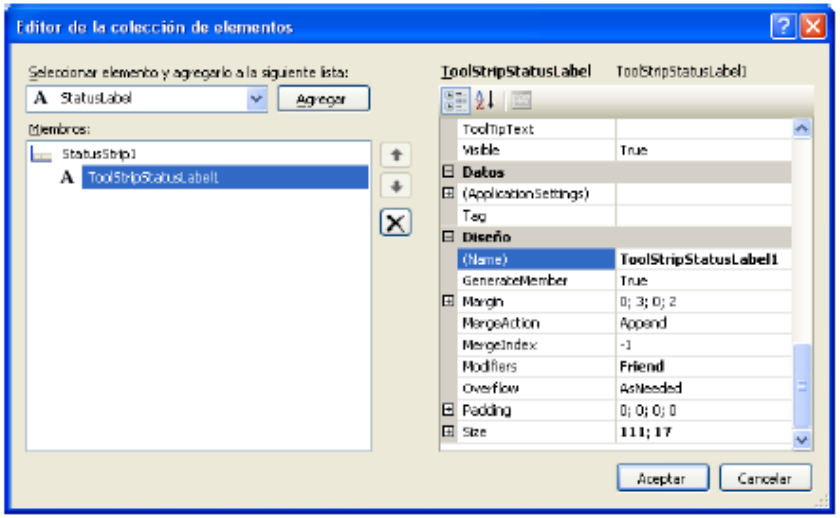
- Elementos de ToolStrip (continuación).
  - DropDownButton (clase ToolStripDropDownButon).
    - ✓ Al pulsarlo muestra un menú desplegable.
      - El Click del botón sólo despliega el menú.
  - Separator (clase ToolStripSeparator).
  - ComboBox (clase ToolStripComboBox).
    - ✓ Representa un ComboBox, con características similares.
  - TextBox (clase ToolStripTextBox).
    - ✓ Representa un TextBox con características similares.
  - ProgressBar (clase ToolStripProgressBar).
    - ✓ Representa una barra de progreso.
      - Las propiedades Minimum y Maximum representan los valores mínimos de la barra.
      - La propiedad Value representa el valor actual de la barra.
      - La propiedad Step representa el valor con el que se incrementará la barra de progreso al ejecutar el método PerformStep.
      - El método Increment, permite incrementar la propiedad Value al margen del valor que tenga Step.

Control StatusStrip

- Barra de estado que muestra información sobre los objetos que se visualizan en un formulario o de las acciones que se están ejecutando.

- Normalmente está compuesto de objetos ToolStripStatusLabel, aunque también puede mostrar ToolStripDropDownButton, ToolStripSplitButton y ToolStripProgessBar.

- Para añadir estos controles se utiliza el editor de la colección de elemento StatusStrip.



Control StatusStrip (II)

- Elemento ToolStripStatusLabel.
  - Propiedad Spring.
    - ✓ Determina si la etiqueta ocupa todo el espacio disponible de la barra de estado.
  - Propiedad BorderSides.
    - ✓ Indica que bordes de la etiqueta se van a mostrar.
  - Propiedad BorderStyle.
    - ✓ Tipo de borde que se va a mostrar.
- Ejemplo.
  - Muestra información sobre un botón al pasar el cursor sobre él.

```
Private Sub Button1_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.MouseHover
        'Al pasar el ratón sobre el botón se muestra el texto en la barra de estado
        ToolStripStatusLabel1.Text = "Guardar los cambios"
End Sub

Private Sub Form2_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.MouseHover
        'Es necesario para borrar el texto cuando se sale del botón
        ToolStripStatusLabel1.Text = ""
End Sub
```