Argentina $8,90 (recargo al interior $0,20)) / México: $45

USERS

Microsoft

Curso teórico y práctico de programación

# Desarrollador .net 6

Con toda la potencia de **Visual Basic .NET** y **C#**

La mejor forma de aprender a programar desde cero

★★★★★

Basado en el programa Desarrollador Cinco Estrellas de Microsoft

## ASP.NET

Introducción al desarrollo Web
Lenguajes de Cliente y Servidor

SALIDA LABORAL
La mejor preparación para rendir los exámenes
GARANTIZADA

## Formularios Web

Conceptos - Eventos
Ciclo de vida



ISBN 978-987-1347-43-8
00006
9 789871 347438

# RedUSERS

COMUNIDAD DE TECNOLOGIA

¡SUMATE!

# EL SITIO Nº1 DE TECNOLOGIA

**Noticias al instante // Entrevistas y coberturas exclusivas // Análisis y opinión de los máximos referentes // Reviews de productos // Trucos para mejorar la productividad // Registrate, participa, y comparte tus opiniones**


## SUSCRIBITE

**SIN CARGO** A CUALQUIERA DE NUESTROS **NEWSLETTERS** Y RECIBÍ EN TU CORREO ELECTRÓNICO TODA LA INFORMACIÓN DEL UNIVERSO TECNOLÓGICO ACTUALIZADA **AL INSTANTE**


**INGRESÁ A**
redusers.com/suscribirse-al-newsletter
**¡Y REGISTRATE YA!**



Foros · Encuestas · Tutoriales · Agenda de eventos · Videos · ¡Y mucho más!

**redusers.com**

Seguinos en:
www.facebook.com/redusers
www.twitter.com/redusers
www.youtube.com/redusersvideos

PRACTICA 01

# Desarrollo de un explorador de archivos

En esta práctica aplicaremos los diferentes conocimientos obtenidos sobre las herramientas de programación.

A continuación, iniciaremos el desarrollo de una aplicación para Windows, un explorador de archivos que nos permita inspeccionar los directorios y archivos de nuestra PC, visualizar sus propiedades y permitir su ejecución. Trabajaremos con Visual Basic .NET y, a la vez, en C#. El objetivo de esta sección es poder aplicar en un caso concreto todo lo que hemos aprendido hasta ahora.

## Objetivo de desarrollo

La finalidad de este proyecto es armar un explorador de archivos similar a Windows Explorer. Para hacerlo, es preciso saber qué elementos debemos mostrar y utilizar durante el desarrollo. También tenemos que hacer un relevamiento acerca de qué funciones incorporar en el programa para que éste sea efectivo. La aplicación que crearemos mostrará los recursos denominados unidades en una computadora. Estas unidades contienen carpetas y archivos, y están identificadas con una letra, para diferenciarse entre sí.
Nuestras necesidades, entonces, serán:

- Analizar los objetivos que queremos cubrir con nuestra aplicación.
- Analizar los eventos que deben ocurrir en nuestra aplicación, que mostrarán contenidos de los recursos y realizarán operaciones con ellos.
- Crear funciones que permitan optimizar el código de la aplicación, reutilizarlo y no tener necesidad de repetirlo.
- Armar un formulario contenedor de los componentes que visualizarán los recursos de la computadora.
- Permitir múltiples opciones para la visualización de los archivos y carpetas, de modo que el usuario pueda personalizar la aplicación a su gusto y no tenga que adaptarse a un estándar impuesto por nosotros como programadores.
- Controlar cambios que ocurran en la computadora, los cuales pueden modificar o incorporar recursos, y hacer que éstos puedan visualizarse en nuestra aplicación.
- Estudiar la posibilidad de expandir el funcionamiento de nuestra aplicación mediante sentencias y comandos que enriquezcan el accionar de nuestro software, y permitan depurar y controlar errores que puedan ocurrir en él.

Es muy importante hacer un relevamiento acerca de qué funciones incorporar en el programa para que éste sea efectivo.

## Manos a la obra

Para comenzar, crearemos un proyecto nuevo del tipo Aplicación para Windows, cuyo nombre será **WinExplorer**. Debemos tener en cuenta que este ejemplo será realizado utilizando Visual Basic .NET, pero en las páginas y en el sitio Web correspondiente a este curso, encontraremos el código fuente también para C#, con el fin de comparar las diferencias entre ambos lenguajes.
La interfaz de usuario se diseña en cualquiera de las dos herramientas de desarrollo prácticamente de la misma manera. Una vez iniciado el proyecto, realizaremos algunos

### Tabla 5 | Componentes de la aplicación WinExplorer

| Formulario | Componente | Nombre | Función |
|---|---|---|---|
| **frmExplorer** | **Formulario principal de la aplicación** | | |
| | ToolStripMenu | VerToolStripMenuItem | Visualiza submenús con distintas vistas de archivos. |
| | ToolStripMenu | ArchivoToolStripMenuItem | Visualiza submenú Salir de la aplicación. |
| | ToolStrip | ToolStripCtl | Barra de herramientas que contiene funciones. |
| | ToolStripContainer | ToolStripContainerCtl | Contenedor de controles dentro del form. |
| | TreeView | TreeViewCtl | Visualizador de recursos y unidades. |
| | ListView | ListViewCtl | Visualizador de archivos y subcarpetas. |
| | StatusStrip | StatusStripCtl | Barra de tareas de la aplicación. |
| **frmPropiedades** | **Formulario visualización de propiedades** | | |
| | TabControl | TabControlPropiedades | Contenedor de controles dentro del form. |
| | txtNombre | TextoBox | Nombre del archivo. |
| | lblDirectorioValue | Label | Muestra el nombre del directorio actual que contiene el archivo o la subcarpeta. |
| | lblTamañoValue | Label | Muestra el tamaño del archivo actual. |
| | lblCreadoValue | Label | Muestra fecha de creación del archivo. |
| | lblModificadoValue | Label | Muestra fecha de última modificación del archivo. |
| | lblAccesoValue | Label | Muestra fecha de la última vez que se accedió al archivo. |
| | lblAtributos | Label | |
| | chkSoloLectura | CheckBox | Muestra si el archivo es de sólo lectura. |
| | chkSistema | CheckBox | Muestra si el archivo es un archivo de sistema. |
| | chkOculto | CheckBox | Muestra si es un archivo oculto. |
| | btnAceptar | Button | Cierra la ventana de propiedades. |

LA INTERFAZ DE USUARIO LA PODREMOS REALIZAR UTILIZANDO EL ENTORNO DE VISUAL BASIC . NET O C#; YA QUE AMBAS PERMITIRÁN OBTENER EL MISMO RESULTADO FINAL.

cambios en las propiedades del formulario Form1, que se crea automáticamente.

A continuación, agregamos a frmExplorer los menús y las barras de herramientas que contendrá la aplicación. Estos controles se ubican en la caja de herramientas de la aplicación, dentro de la solapa Menús y barras de herramientas. Se llaman ToolStripContainer (cuyo nombre cambiaremos por **ToolStripContainerCtl**) y MenuStrip (su nombre será **MenuStripCtl**).

Una vez que ubicamos ambos controles, vamos a crear las opciones del menú. Como vemos en pantalla, es sencillo: basta con ir armando la estructura del menú a medida que lo escribimos. Creamos los menús y sus submenús asociados y, luego, cambiaremos algunas propiedades. Esto puede observarse en la Figura 023 y en la Tabla 6.

Las opciones del menú **Ver** nos permitirán cambiar la vista de los archivos tal como lo hace el Explorador de Windows al utilizar un

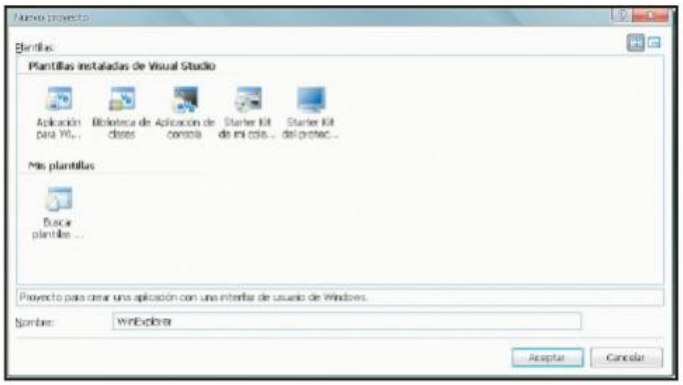



FIGURA 022 | Selección del tipo de proyecto.

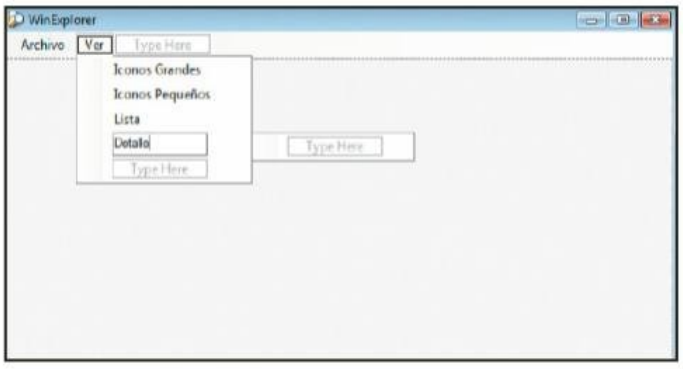



FIGURA 023 | Creación del menú de la aplicación.

## Tabla 6 | Propiedades de Form1

| Propiedad | Valor |
|---|---|
| Name | frmExplorer |
| Text | WinExplorer |
| StartPosition | CenterScreen |
| WindowState | Maximized |
| Icon | Seleccionar uno a gusto o cargar el que viene adjunto al código fuente en la Web de este curso. |

> Para proporcionar un acceso rápido a los comandos más comunes, crearemos una barra de herramientas, utilizando el control denominado ToolStrip.

control Listview. La propiedad **Tag** se usará como clave para unificar el código.
Para asignar imágenes a los elementos del menú, conviene recurrir a ellas como recursos en la aplicación. Esto nos permitirá manipular dichas imágenes desde distintos lugares de la aplicación. Para almacenar los recursos de imágenes, desde la ventana Explorador de soluciones seleccionamos el icono de la barra de herramientas denominado Propiedades; aparece una pestaña de propiedades de la aplicación. En la solapa **Recursos** agregamos las imágenes necesarias utilizando la opción **Imagen/Agregar Recurso**. Luego indicamos el formato de imagen que queremos agregar (JPG, PNG, ICO, BMP, GIF). Al finalizar el procedimiento, se nos solicitará guardar el archivo de recursos, que quedará junto al resto de los que componen nuestro proyecto.
Para asignar las imágenes correspondientes a los elementos del menú, simplemente, desplegamos el control **MenuStrip**, y en cada uno de los submenús que llevan imágenes, hacemos un clic con el botón derecho del mouse y seleccionamos **Establecer imagen**.
Para proporcionar un acceso rápido a los comandos más comunes, vamos a crear una barra de herramientas. Utilizamos un control denominado ToolStrip; lo ubicamos debajo del menú principal y agregamos los elementos mencionados en la sección 1 de la Tabla 7.
Una vez finalizada la creación del menú

## Tabla 7 | Menú de la aplicación y barra de herramientas

**1- Control MenuStripCtl**

| Menú | Submenú | Propiedades |
|---|---|---|
| Archivo | | Tag: Archivo |
| | Salir | Tag: Salir |
| | | ShortCutKeys: ALT + F4 |
| Ver | | Tag: Ver |
| | Iconos grandes | Tag: IconosGrandes |
| | Iconos pequeños | Tag: IconosPequeños |
| | Lista | Tag: Lista |
| | Detalles | Tag: Detalle |

**2- Control ToolStrip**

| Botón | Propiedades |
|---|---|
| Separator | |
| Button | Text: Iconos Grandes |
| | Tag: ToolIconosGrandes |
| Button | Text: Iconos Pequeños |
| | Tag: ToolIconosPequeños |
| Button | Text: Lista |
| | Tag: ToolLista |
| Button | Text: Detalle |
| | Tag: ToolDetalle |
| Separator | |
| Button | Text: Propiedades |
| | Tag: ToolPropiedades |
| Button | Text: Salir |
| | Tag: ToolSalir |

contextual, asignamos las imágenes correspondientes a los distintos ítem de la misma manera en que lo hicimos con el control MenuStrip. También necesitamos agregar a la aplicación una barra de estado, que permitirá ver en cada momento la ruta completa del archivo seleccionado o el path de la carpeta, si no existe un archivo seleccionado. Para hacerlo, agregamos un control denominado StatusStrip en la parte inferior del form y generamos un elemento llamado LabelStatus, al que cambiaremos las siguientes propiedades:

- Name: ToolStripStatuslblDescripcion
- Text: WinExplorer
- Spring: True
- TextAlign: MiddleLeft

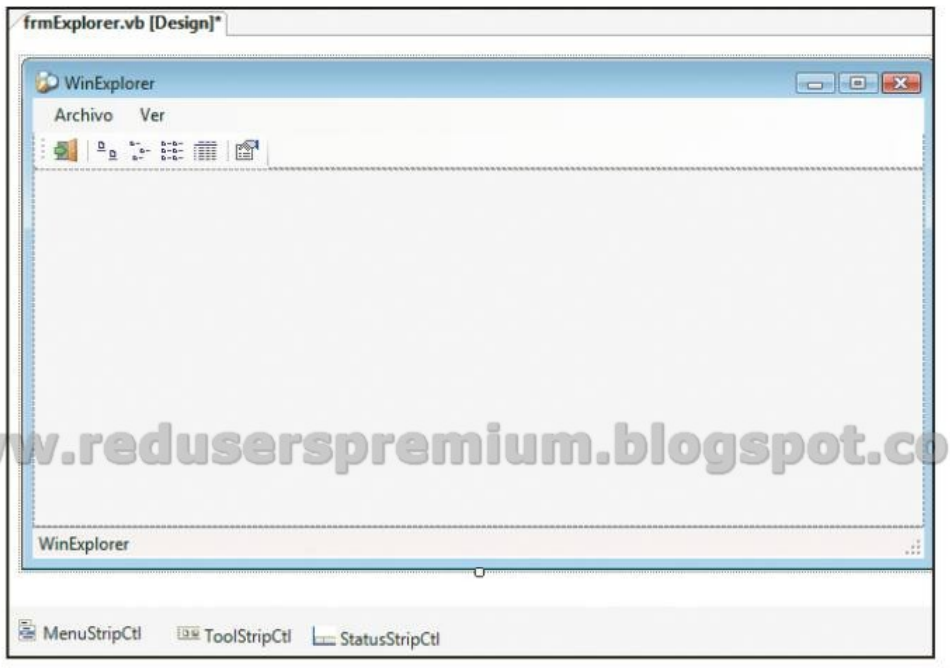
Con esto, tendremos lista la estructura de menús, barra de herramientas y barra de estado.

## Directorios y archivos

A continuación, vamos a crear la interfaz en donde se verán los directorios, las unidades de disco y los archivos en forma jerárquica. Para hacerlo, necesitamos una estructura de árbol para directorios y unidades de disco; el control adecuado en este caso es **TreeView**. También precisamos agregar una lista para ver los archivos que hay en los directorios, que permita cambiar su forma de visualización. El control **SplitContainer** nos permitirá dividir la pantalla en dos paneles verticales y ubicar los controles en él, con la

> El control SplitContainer permite dividir la pantalla en dos paneles horizontales con la ventaja de poder redimensionar su ancho proporcionalmente en tiempo de ejecución.

FIGURA 024 | Interfaz final de menús, barra de herramientas y barra de estado.



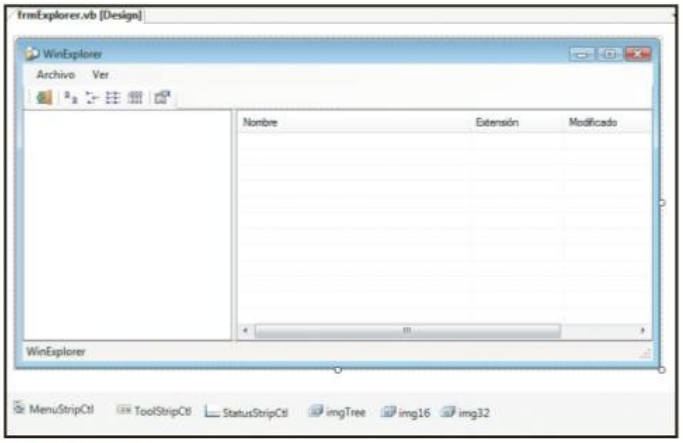



FIGURA 025 | Interfaz visual de WinExplorer casi terminada.

ventaja de poder redimensionar su ancho proporcionalmente en tiempo de ejecución. Entonces, como primer paso, agregamos el control **SplitContainer** al centro del formulario. Veremos en él dos paneles libres: en el Panel1, el de la izquierda, agregamos el control TreeView; y en el Panel2 arrastramos el control ListView. En ambos casos cambiamos su propiedad **Dock** a **Fill**. Acto seguido, cambiamos el nombre del control TreeView a TreeViewCtl, y a ListView le ponemos ListViewCtl. Para ver las diferentes carpetas y unidades en el control TreeView necesitamos algunos iconos que permitan diferenciar las unidades de disco, las carpetas especiales y las comunes. Primero arrastramos un control ImageList, al cual le cambiamos el nombre de imgTree y le asignamos las imágenes utilizando la propiedad **Images**. En la ventana que se presenta, asignamos una imagen para cada uno de los siguientes elementos y cambiamos su propiedad Name según muestra el sector 1 de la Tabla 8.

Una vez incorporadas las imágenes, asignamos el control ImageList al control TreeView. Para esto, seleccionamos el TreeView y cambiamos el valor de su propiedad ImageList = **imgTree**.

En el control ListView mostraremos los archivos y algunas de sus propiedades. Necesitaremos una imagen para cada archivo (por simplicidad, en este caso utilizaremos una para cualquier tipo). En este ejemplo vamos a usar dos controles imageList: uno para los iconos de 16 píxeles y otro para los de 32 píxeles. Denominamos a cada uno **img16** e **img32**, respectivamente, y asignamos una única imagen (la misma en el tamaño correspondiente a cada ImageList) con su propiedad **Name** en **File**.

## Tabla 8 | ImageList y ListView

**1- Control ImageList**

| Elemento (imagen) | Name |
|---|---|
| Escritorio | Desktop |
| Unidad de Disco | Drive |
| Carpeta Cerrada | FolderClosed |
| Carpeta Abierta | FolderOpen |
| MiPc | MiPC |
| Mis Documentos | MyDocuments |
| Mi Musica | MyMusic |
| Mis imágenes | MyPictures |

**2- ListView**

| Propiedad | Valor | Función |
|---|---|---|
| FullRowSelect | True | Selecciona la fila en vez de sólo la primera columna. |
| GridLines | True | Muestra las líneas de la grilla. |
| LargeImgList | Img32 | Asigna el control ImageList que contiene imágenes de 32 píxeles. |
| SmallImgList | Img16 | Asigna el control ImageList que contiene imágenes de 16 píxeles. |
| MultiSelect | False | Previene la selección múltiple de elementos. |
| View | Details | Muestra la vista Detalles inicialmente. |

Una vez que las cargamos, cambiamos las propiedades en el ListView según se observa en el sector 2 de la Tabla 8.
Queda aún por crear las columnas que mostrarán la información extendida de los archivos. Seleccionamos la propiedad **Columns** en la ventana correspondiente y generamos las siguientes modificando estas propiedades:

- Columna1: Name:ColNombre, Text:Nombre, Width:100
- Columna2: Name:ColExtension, Text:Extensión
- Columna3: Name:ColModificado, Text:Modificado
- Columan4: Name:ColTamaño, Text:Tamaño

También necesitamos prever algo importante en nuestra aplicación: las computadoras actuales suelen estar conectadas en red para compartir recursos y, eventualmente, conectamos a ellas dispositivos externos como: pendrives, reproductores de MP3 y MP4, y otros. Estos dispositivos suelen crear nuevas unidades de disco temporales en la PC, con el fin de poder intercambiar archivos desde y hacia ellos. Para que WinExplorer detecte estos cambios de unidades, agregaremos un control Timer, que generará, de manera automática, un evento cada cierto período de tiempo. Esto lo utilizaremos para actualizar las unidades de disco.
Entonces, agregamos un control Timer al formulario y cambiamos sus propiedades **Name** a TimerCtl e **Interval** a 5000 (5 segundos). Para el segundo punto utilizamos un menú contextual, que proporcionará las opciones de Ejecutar y Propiedades, y será asignado al ListView en su propiedad ContextMenú. Los elementos del menú contextual deben tener la propiedad Tag con Ejecutar y Propiedades, respectivamente.

> El control Timer genera, de manera automática, un evento cada cierto período de tiempo. En este caso, lo utilizaremos para actualizar las unidades de disco.

## Menú contextual

Para acceder a las propiedades de un archivo seleccionado, se requiere otro formulario, de modo que agregamos uno nuevo y le ponemos como nombre **frmPropiedades**. Éste tiene un conjunto de controles Label que muestran la información de un archivo, y varios controles Checkbox para sus atributos. No entraremos en detalle aquí respecto al diseño de la interfaz, ya que es sencilla y variará dependiendo de los atributos que queramos observar dentro del menú. Lo único que debemos tener en cuenta es que debe proporcionar una propiedad llamada **Archivo** del tipo **System.IO.FileInfo**. Esta propiedad será la que se ocupe de brindar la información para visualizar.

### ! El código completo

Podremos encontrar el código completo de esta aplicación, en ambos lenguajes (VB.NET y C#), listo para descargar y probar, en el sitio oficial de este curso. Recordemos que la dirección de la página Web en cuestión es: http://desarrollador.redusers.com.

El evento Shown sólo se produce la primera vez que se muestra un formulario; no se genera al minimizarlo, restaurarlo o dibujarlo otra vez.

## Haciendo que la aplicación funcione

Llegó la hora de escribir el código con el que WinExplorer cobrará vida. Comencemos por poner código al formulario **frmPropiedades**, ya que es bastante sencillo. En el evento **Shown** de **frmPropiedades** mostramos las propiedades del archivo (tales como nombre, tamaño y algunos de sus atributos), y en el evento clic del botón salimos del form. El evento Shown sólo se produce la primera vez que se muestra un formulario; no se genera al minimizarlo, maximizarlo, restaurarlo, ocultarlo, mostrarlo, o invalidarlo y dibujarlo otra vez. El código que analizamos a continuación es el correspondiente a VB.NET y C#. Lo más importante de la siguiente sección es poder visualizar y comprender las similitudes y diferencias entre ambos lenguajes.

### El código en distintos objetos

En principio declaramos un objeto **_Archivo**, del tipo **IO.FileInfo** (namespace que permite realizar operaciones sobre los archivos del sistema). Le asignamos una propiedad por la cual se obtiene información del archivo, y otra propiedad de salida por la cual se envía la información al control que la solicite.

VB.NET

```
'Código correspondiente a
 FrmPropiedades_Declaraciones
Private _Archivo As IO.FileInfo
Public Property Archivo() As IO.FileInfo
  Get
    Return _Archivo
  End Get
  Set(ByVal value As IO.FileInfo)
    _Archivo = value
  End Set
End Property
```

C#

```
  private System.IO.FileInfo _Archivo;
  public System.IO.FileInfo Archivo
  {
    get { return _Archivo; }
    set { _Archivo = value; }
  }
```

A continuación, vemos el código en VB.NET y C# que corresponde al botón **Aceptar**. Este código es el que nos permite cerrar la ventana de propiedades:

VB.NET

```
'Código correspondiente al botón
 Aceptar_Click
Private Sub btnAceptar_Click(ByVal sender
As System.Object, ByVal e As
System.EventArgs) Handles btnAceptar.Click
  Me.Close()
End Sub
```

C#

```
  private void btnAceptar_Click(object
  sender, EventArgs e)
  {
    this.Close();
  }
```

En el evento **Shown** ingresamos el código correspondiente para que los controles Label en **frmPropiedades** muestren la información correspondiente al archivo seleccionado.
Cabe notar que declaramos una variable del tipo Double, denominada tamaño, que contendrá el correspondiente al archivo. Dado que, por ejemplo, un archivo de 1 Kb equivale a 1024 bytes, utilizaremos la función matemática de redondeo **Math.Round**. Veamos cómo hacer esto en ambos lenguajes:

VB.NET

```
Private Sub frmPropiedades_Shown(ByVal
sender As Object, ByVal e As
System.EventArgs) Handles Me.Shown
    txtNombre.Text = Archivo.Name
    lblDirectorioValue.Text =
    Archivo.DirectoryName
    Dim Tamaño As Double
    Tamaño = Math.Round(Archivo.Length /
    1024, 0)
    If Tamaño < 1 Then
      lblTamañoValue.Text = Archivo.Length &
      " bytes"
    Else
      lblTamañoValue.Text = Tamaño & " KB"
    End If

    lblCreadoValue.Text =
    Archivo.CreationTime.ToString
    lblModificadoValue.Text =
    Archivo.LastWriteTime.ToString
    lblAccesoValue.Text =
    Archivo.LastAccessTime.ToString
    chkSoloLectura.Checked =
    Archivo.Attributes And
    IO.FileAttributes.ReadOnly
    chkSistema.Checked = Archivo.Attributes
    And IO.FileAttributes.System
    chkOculto.Checked = Archivo.Attributes
    And IO.FileAttributes.Hidden
End Sub
```

C#

```
private void frmPropiedades_Shown(object
sender, EventArgs e)
{
  txtNombre.Text = Archivo.Name;
  lblDirectorioValue.Text =
  Archivo.DirectoryName;
  double Tamaño;
  Tamaño = Math.Round((double)
  (Archivo.Length / 1024), 0);
  if (Tamaño < 1)
  {
    lblTamañoValue.Text = Archivo.Length
    + " bytes";
  }
  else
  {
    lblTamañoValue.Text = Tamaño + " KB";
  }
  lblCreadoValue.Text =
  Archivo.CreationTime.ToString();
  lblModificadoValue.Text =
  Archivo.LastWriteTime.ToString();
  lblAccesoValue.Text =
  Archivo.LastAccessTime.ToString();
  if (( Archivo.Attributes &
  System.IO.FileAttributes.ReadOnly)==
  System.IO.FileAttributes.ReadOnly) {
  chkSoloLectura.Checked = true; }
  if ((Archivo.Attributes &
  System.IO.FileAttributes.System) ==
  System.IO.FileAttributes.System) {
  chkSistema.Checked = true; }
  if ((Archivo.Attributes &
  System.IO.FileAttributes.Hidden) ==
  System.IO.FileAttributes.Hidden) {
  chkOculto.Checked = true; }
}
```

El corazón de esta aplicación son los métodos CargarArbol y SeleccionarNodo.

## Formulario principal

El corazón de la aplicación son los métodos **CargarArbol** y **SeleccionarNodo**. El primero es llamado en el evento Load del formulario, y carga inicialmente el árbol con las unidades de disco y las carpetas especiales. No carga las subcarpetas de cada uno: esto se hace "a demanda", a medida que es necesario para agilizar la aplicación, y lo hacemos en el método **SeleccionarNodo**, que ocurre al seleccionar un ítem del árbol. A continuación, veremos el código del método CargarArbol; este código es para VB.NET, como dijimos al principio de la práctica:

```
'cargamos items en el arbol
Private Sub CargarArbol()
  Dim oNodo As TreeNode
  Dim oChild As TreeNode
  TreeViewCtl.Nodes.Clear()
  'Cargamos MIPC
  oNodo = TreeViewCtl.Nodes.Add(KeyMIPC,
  "Mi PC")
  oNodo.ImageKey = "MiPC"
  oNodo.SelectedImageKey = "MiPC"
  oNodo.Tag = "MiPC"

  'cargamos los discos disponibles
  For Each oDrive As IO.DriveInfo In
  IO.DriveInfo.GetDrives
    oChild =
    oNodo.Nodes.Add(oDrive.RootDirectory.
    FullName, oDrive.RootDirectory.FullName)
    oChild.Tag =
    oDrive.RootDirectory.FullName
    oChild.ImageKey = "Drive"
    oChild.SelectedImageKey = "Drive"
    oChild = Nothing
  Next

  'Cargamos Escritorio
  oNodo = New TreeNode
  oNodo.Tag = My.Computer.FileSystem.
  SpecialDirectories.Desktop
  oNodo.Text = "Escritorio"
  oNodo.SelectedImageKey = "Desktop"
  oNodo.ImageKey = "Desktop"
  TreeViewCtl.Nodes.Add(oNodo)

  'Cargamos Mis Documentos
  oNodo = New TreeNode
  oNodo.Tag = My.Computer.FileSystem.
  SpecialDirectories.MyDocuments
  oNodo.Text = "Mis Documentos"
  oNodo.SelectedImageKey = "MyDocuments"
  oNodo.ImageKey = "MyDocuments"
  TreeViewCtl.Nodes.Add(oNodo)

  'cargamos mis imágenes
  oNodo = New TreeNode
  oNodo.Tag = My.Computer.FileSystem.
  SpecialDirectories.MyPictures
  oNodo.Text = "Mis Imágenes"
  oNodo.SelectedImageKey = "MyPictures"
  oNodo.ImageKey = "MyPictures"
  TreeViewCtl.Nodes.Add(oNodo)

  'cargamos mi música
  oNodo = New TreeNode
  oNodo.Tag = My.Computer.FileSystem.
  SpecialDirectories.MyMusic
  oNodo.Text = "Mi Música"
  oNodo.SelectedImageKey = "MyMusic"
  oNodo.ImageKey = "MyMusic"
  TreeViewCtl.Nodes.Add(oNodo)

End Sub
```

EL MÉTODO SELECCIONARNODO ES COMPLEJO, YA QUE TIENE UNA DOBLE FUNCIONALIDAD: VERIFICAR SI EXISTEN SUBDIRECTORIOS PARA EL NODO SELECCIONADO Y, A LA VEZ, MOSTRAR LOS ARCHIVOS DE ESA CARPETA EN EL CONTROL LISTVIEW.

Como vemos en el código, primero eliminamos cualquier ítem que exista anteriormente dentro del control TreeView y cargamos los nodos a mano. Lo importante aquí es asignar la clave correcta a estos nodos desde el principio, para poder obtenerlos luego y, a su vez, utilizar la propiedad **Tag** de cada elemento para almacenar el path o ruta completo, que permita una fácil localización cuando lo necesitemos. La asignación de las imágenes se realiza indicando sus nombres, y se corresponden con los asignados en el control ImageList e **imgTree**.

El método SeleccionarNodo es un poco más complejo, ya que tiene una doble funcionalidad: verificar si existen subdirectorios para el nodo seleccionado y cargarlos si es necesario, a la vez que debe mostrar los archivos de esa carpeta en el control ListView. Parte del código del método SeleccionarNodo es el siguiente:

```
Private Sub SeleccionarNodo(ByVal e As
System.Windows.Forms.TreeViewEventArgs)
    'al seleccionar un nodo mostramos su ruta
    ToolStripStatuslblDescripcion.Text =
    e.Node.Tag

    'si no tiene hijos, cargamos sus
    subcarpetas
    'tenememos en cuenta carpetas especiales
    'como MIPC que solo contiene las
    unidades de disco
    If e.Node.Tag <> "MiPC" Then
        'si es ditinto a mi pc vemos si tiene
        cargadas las subcarpetas
        If e.Node.Nodes.Count = 0 Then
            Dim oChild As TreeNode
            Dim SubFolders As String()
            Try
                SubFolders = IO.Directory.
                GetDirectories(e.Node.Tag)
            Catch ex As IO.IOException
                MessageBox.Show("La unidad de
                disco " & e.Node.Tag & " no está
                disponible.", Me.Text, Message
                BoxButtons.OK, MessageBoxIcon.
                Exclamation)
                Exit Sub
            Catch ex As
            UnauthorizedAccessException
                MessageBox.Show("Ud. no tiene los
                suficientes permisos para acceder
                a este recurso.", Me.Text,
                MessageBoxButtons.OK,
                MessageBoxIcon.Exclamation)
                Exit Sub
            Catch ex As Exception
                MessageBox.Show(ex.Message,
                Me.Text, MessageBoxButtons.OK,
                MessageBoxIcon.Exclamation)
                Exit Sub
            End Try

            'recorro las subcarpetas y las cargo
            For Each oFolder As String In
            SubFolders
```

```
            oChild = New TreeNode
            oChild.Tag = oFolder
            oChild.Text =
            IO.Path.GetFileNameWithout
            Extension(oFolder)
            oChild.ImageKey = "FolderClosed"
            oChild.SelectedImageKey =
            "FolderClosed"
            e.Node.Nodes.Add(oChild)
            oChild = Nothing
          Next
        End If

        `A continuación, una vez cargadas las
        subcarpetas mostramos en el control
        listview los archivos del directorio
        seleccionado

    End Sub
```

Lo primero que hacemos es mostrar la ruta de la carpeta en la barra de estado; luego, determinamos si el nodo es MiPC o uno diferente, ya que sabemos que sus subcarpetas, en realidad, son las unidades de disco y éstas son actualizadas con el control Timer. Si es alguna otra, verificamos si ya hemos cargado sus subcarpetas; de no ser así, lo hacemos. Finalmente, cargamos en el control **listview** los archivos de la carpeta seleccionada. Éste es uno de los lugares en donde utilizamos la propiedad Tag del nodo, en la que hemos almacenado la ruta completa de la carpeta en disco. También realizamos algunos controles de errores, como pueden ser unidades de disco no preparadas o falta de permiso para visualizar el contenido de alguna carpeta en particular.

## Actualizando archivos

Para actualizar las unidades de disco en el evento **Tick** del Timer, verificamos si la cantidad ha cambiado y recargamos los nodos de MiPC. A continuación, vemos el código correspondiente:

```
    Private Sub TimerCtl_Tick(ByVal sender As
    Object, ByVal e As System.EventArgs)
    Handles TimerCtl.Tick
      TimerCtl.Stop()

       `verifico que la cantidad de drives
        haya cambiado antes de actualizar
      Dim oNodo As TreeNode
      oNodo = TreeViewCtl.Nodes.Find(KeyMIPC,
      False)(0)
      If IO.DriveInfo.GetDrives.Length <>
      oNodo.Nodes.Count Then

        'Cambió. Recargamos
        oNodo.Nodes.Clear()
        'cargamos los discos disponibles
        For Each oDrive As IO.DriveInfo In
        IO.DriveInfo.GetDrives
          Dim oChild As TreeNode
          oChild =
          oNodo.Nodes.Add(oDrive.RootDirectory.
          FullName, oDrive.RootDirectory.
          FullName)
          oChild.Tag =
          oDrive.RootDirectory.FullName
          oChild.ImageKey = "Drive"
          oChild.SelectedImageKey = "Drive"
          oChild = Nothing
        Next
        TreeViewCtl.SelectedNode = oNodo
        SeleccionarNodo(New
        TreeViewEventArgs(oNodo))
      End If

      TimerCtl.Start()

    End Sub
```

Lo importante en esta rutina es detener el funcionamiento del control Timer antes de realizar la recarga, ya que, dadas las circunstancias, podría volver a generarse el evento y tener una llamada reentrante a él. Después de efectuar la recarga, iniciamos otra vez el control Timer.

## Cambio de visualización

El control ListView tiene la capacidad de cambiar su vista y, para eso, creamos los menús y los botones del control toolbar. Al realizar clic en cualquiera de ellos, se llama al método **CambiarVista**, que, además, actualiza el estado del resto de los ítem de los menús.

## Propiedades y ejecución de un archivo

El menú contextual que agregamos sirve para visualizar las propiedades del archivo seleccionado. Al hacer clic en esa opción del menú, se llama al método **VerPropiedades**.

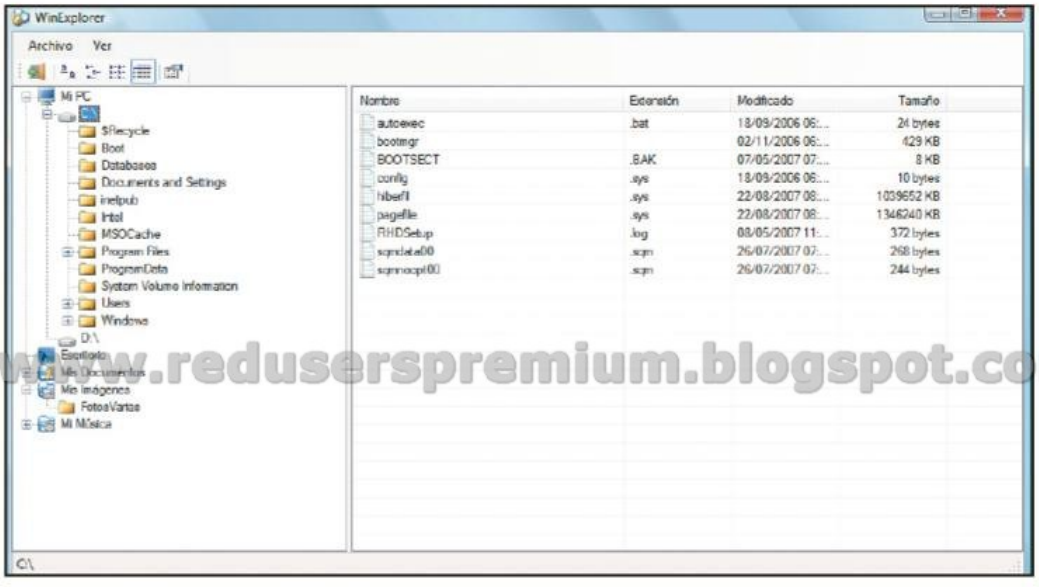
## Otros eventos

También hemos codificado diferentes eventos del control TreeView para modificar el icono de la carpeta (abierta o cerrada) y el evento Load del Form, asignando de manera dinámica los manejadores de eventos a los elementos del menú y a la barra de herramientas, para ver las diferencias de sintaxis entre VB.NET y C#. Los invitamos a estudiar el código completo de la aplicación y a realizar algunas de las mejoras propuestas:

> Es importante que exista una rutina que detenga el funcionamiento del control Timer antes de realizar la recarga, ya que podría volver a generarse el evento y tener una llamada reentrante a él.

- Permitir operaciones de Copiar, Cortar y Pegar archivos.

FIGURA 026 | Así veremos nuestra aplicación funcionando en Windows.



- Visualizar en el ListView tanto las carpetas como los archivos, tal como lo hace el Explorador de Windows real.
- Mostrar los iconos en los archivos de acuerdo con el programa asociado.
- Crear, eliminar y modificar carpetas y archivos.

## Expandir la funcionalidad de la aplicación

A lo largo de esta sección, hemos visto las funciones y la sintaxis básica en C# y VB.NET para crear un explorador de archivos de Windows con los elementos básicos para su funcionamiento. Podemos investigar un poco más sobre sintaxis en el lenguaje que nos guste o en ambos, para poder expandir la funcionalidad de WinExplorer al incorporarle acciones como las siguientes: copiar, mover y eliminar archivos (enviándolos a la Papelera), crear nuevas carpetas, mover y copiar carpetas con subcarpetas y archivos incluidos, visualizar en cada archivo el icono correspondiente a su extensión definido en Windows, y otras funcionalidades más que se nos ocurran, y que generalmente se incluyen en este tipo de aplicaciones. En la Tabla 8, veremos además una serie de comandos destinados a realizar algunas de estas sugerencias que permitirán potenciar aún más nuestra aplicación WinExplorer. Éstas son algunas de las funciones o métodos que permitirán expandir la funcionalidad. Los invitamos a probarlas y mejorar así este sistema.

## Tabla 9 | Namespace System.IO

| Clase File | |
|---|---|
| **Método** | **Descripción** |
| Copy | Copia un archivo existente a un archivo nuevo. |
| Create | Crea un archivo en la ruta de acceso especificada. |
| Delete | Elimina el archivo especificado. Este método borra directamente el archivo, sin pasar por la Papelera de reciclaje. Se recomienda probarlo con suma precaución. |
| Exists | Determina si existe un archivo especificado. |
| GetAttributes | Obtiene el objeto FileAttributes del archivo especificado. |
| GetCreationTime | Devuelve la fecha y hora de creación del archivo especificado. |
| GetLastAccessTime | Devuelve la fecha y hora de la última vez que se accedió al archivo especificado. |
| GetLastWriteTime | Devuelva la fecha y hora en que se escribió el archivo o directorio por última vez. |
| Move | Mueve un archivo especificado hacia el origen indicado. |
| SetAccessControl | Establece los atributos FileAttributes especificados para el archivo de la ruta indicada. |
| SetCreationTime | Establece la fecha y hora en la que se creó el archivo. |
| **Clase Directory** | |
| Move | Mueve un directorio especificado hacia una nueva ubicación. |
| Delete | Elimina un directorio vacío especificado. |
| Delete(string, boolean) | Elimina un directorio especificado, e indica los subdirectorios que contiene. |
| CreateDirectory | Crea un directorio en el lugar especificado. |
| GetDirectories | Permite obtener los subdirectorios de una ruta especificada. |

# El primer repaso

A continuación haremos un repaso general de los puntos más importantes de cada una de las herramientas que vimos hasta ahora.

Esta sección tiene por objetivo repasar aquellos conceptos que serán evaluados en el examen de certificación de Microsoft. Veremos primero una síntesis de los conceptos más importantes. Luego, repasaremos algunas preguntas del examen. Al final, encontraremos ejercicios prácticos que nos ayudarán a fijar los conceptos aprendidos.

## La programación y los lenguajes

- La programación se enfoca en el desarrollo de soluciones basadas en software o programas informáticos de computadoras. Estos programas se sustentan en un conjunto de instrucciones que una computadora pueda interpretar y ejecutar.
- Los programas se escriben en un lenguaje de programación. El lenguaje de programación es "un idioma" utilizado para controlar el comportamiento de una computadora a través de reglas sintácticas y semánticas que definen su estructura y el significado de cada uno de sus elementos. Este lenguaje permite a un programador especificar qué datos serán controlados, operados y almacenados en una computadora.

## El programador

- Un programador, básicamente, es una persona que se dedica a escribir programas para computadoras.
- El programador es el encargado de implementar algoritmos que resuelvan problemas operativos o mejoren la productividad en una empresa, mediante uno o varios lenguajes de programación, que puedan ser entendidos por la computadora.

## La plataforma .NET

Microsoft .NET es una plataforma de desarrollo y ejecución de aplicaciones. Sus características principales son:

- Las aplicaciones se ejecutan en un entorno aislado del sistema, denominado runtime, lo cual lo hace flexible, seguro y portable.
- Está 100% orientado a objetos.
- Permite el desarrollo de software complejo y robusto.
- Brinda un único modelo de programación para desarrollar diferentes tipos de aplicaciones.
- Se integra fácilmente con las aplicaciones generadas en versiones anteriores de Visual Studio, como COM.
- Permite integrar aplicaciones de otras plataformas y sistemas operativos implementando estándares como XML, SOAP y WSDL.

# LA ESTRELLA I

## Elementos de la plataforma .NET

- .NET Framework
- CLR
- BCL
- Lenguajes de programación y compiladores
- Herramientas y documentación

## El modelo de ejecución

El modelo de desarrollo se basa en dos etapas:

- La escritura del código fuente.
- La compilación del assembly o ejecutable. Éste genera un código intermedio o MSIL, que luego deriva la ejecución a CLR (*Common Language Runtime*), encargado de traducir el código al lenguaje máquina para que pueda ser ejecutado. Esto se conoce como compilación JIT.

## Principales lenguajes

- Visual Basic .NET es la evolución de VB 6.0. Fue rescrito por completo para hacerlo totalmente orientado a objetos. Mantiene gran parte de su sintaxis similar a la de su antecesor.
- C# es un nuevo lenguaje diseñado, específicamente, para la plataforma .NET. Su sintaxis es similar a las de C y Java.

## Principales sentencias

- SENTENCIA IF: Representa la estructura de selección más simple en cualquier lenguaje de programación .NET. Permite ejecutar una porción de código si es que se cumple determinada condición. Se puede combinar con **else** para ejecutar otra porción de código en caso de que no se cumpla la sentencia **if**.
- SELECT CASE: En Visual Basic .NET, la sentencia **select case** equivale a **switch** de C#. Se puede denominar como un caso generalizado de la sentencia if, pero en vez de ejecutar una porción de código dependiendo de una condición lógica, select case divide la ejecución en un grupo de casos disjuntos.
- SENTENCIA FOR: Permite ejecutar un bloque de código una cantidad determinada y fija de veces. Con la palabra **Step**, podemos indicar el paso o valor en el cual se incrementa el contador de cada iteración.
- SENTENCIA WHILE: Permite ejecutar un bloque de código repetidas veces mientras se cumpla una condición lógica. En Visual Basic .NET el bloque de ejecución se cierra con la palabra clave **loop**.

## Procedimientos o funciones

Para optimizar el desarrollo de nuestros programas y no tener que repetir un código que pueda ser reutilizable, podemos crear procedimientos o funciones que lo almacenen. Al momento de tener que usarlo otra vez, sólo tendremos que llamar al procedimiento o función que lo contiene.

## Clases y objetos

Las clases son abstracciones de objetos de la rea lidad que se quieren modelar. Los objetos son instancias particulares de las clases; cada uno pertenece a una clase (agrupados por propiedades comunes), pero tiene identidad propia.
Las clases son definidas por un nombre, y en su interior agrupan métodos y propiedades.

## Conceptos sobre bases de datos

Una base de datos es un conjunto de información organizada que está almacenada en algún soporte no volátil, y que se puede consultar, administrar y actualizar a través de un sistema o programa, conocido como sistema de gestión de base de datos (SGDB o DBRMS). Éste permite realizar operaciones como agregar, modificar, borrar, consultar, cruzar, crear, administrar la seguridad y efectuar backup, entre otras. Las bases de datos más utilizadas en informática son las relacionales.

# Preguntas de ejemplo

## 1 | ¿Qué es .NET?

**A.** .NET es un sistema operativo, que incluye lenguajes de programación y entornos de desarrollo.
**B.** Es un sistema operativo compuesto por una interfaz gráfica, un paquete de oficina, un programa de conexión a Internet y un navegador Web.
**C.** .NET es una plataforma de desarrollo compuesta por un entorno de ejecución, bibliotecas de funcionalidad (Class Library), lenguajes de programación, compiladores, herramientas de desarrollo y guías de arquitectura.

Respuesta correcta C: .NET es una plataforma que engloba distintas aplicaciones, servicios y conceptos, y que, en conjunto, permiten el desarrollo y la ejecución de aplicaciones.

## 2 | Las características de .NET son:

**A.** Es una plataforma que, simplemente, continúa en su versión a Visual Studio 6, manteniendo las mismas características que él y sólo optimizada para Windows XP.
**B.** Es una plataforma de ejecución intermedia, 100% orientada a objetos, con soporte multilenguaje y que permite el desarrollo de aplicaciones de misión crítica.
**C.** Es un sistema operativo que viene a reemplazar a Windows XP e incluye algunas herramientas para programación de escritorio.

Respuesta correcta B: .NET es una plataforma de ejecución intermedia. Las aplicaciones son ejecutadas a través de un componente de software llamado entorno de ejecución, que se encarga de manejar el ciclo de vida de cualquier aplicación .NET, desde su inicio hasta su finalización.

## 3 | Otras ventajas conocidas de .NET son:

**A.** Está compuesto por un único lenguaje de programación, que permite aprenderlo y no complicarse con diversas alternativas.
**B.** Permite desarrollar sólo aplicaciones para servidores, lo cual facilita su uso desde cualquier computadora conectada a un servidor de red o servidor de Internet.
**C.** Provee de un modelo único de programación para cualquier tipo de aplicación y dispositivo de hardware. Se integra fácilmente con la plataforma Microsoft y con aplicaciones desarrolladas en otras plataformas.

Respuesta correcta C: .NET fue diseñado para proveer de un único modelo de programación para todo tipo de aplicaciones (Windows, de consola, Web, móviles, etc.) y para cualquier dispositivo de hardware (PC, Pocket PC, teléfonos celulares inteligentes -también llamados smartphones-, Tablet PC, etc.). Tiene la posibilidad de interactuar e integrarse fácilmente con aplicaciones desarrolladas en plataformas anteriores, como objetos COM. .NET se integra fácilmente con aplicaciones desarrolladas en otras plataformas Microsoft, y también con aquellas desarrolladas en otras plataformas de software, sistemas operativos o lenguajes de programación, respetando estándares como XML, HTTP, SOAP, WSDL y UDDI.

## 4 | ¿Cómo está compuesta la plataforma de ejecución intermedia?

**A.** El entorno Microsoft .NET contiene un entorno de desarrollo compuesto por: lenguajes de programación, librerías de funcionalidad y entornos de ejecución. Todo esto integra una aplicación .NET que corre sobre el sistema operativo de la familia Windows.
**B.** El entorno .NET está compuesto por librerías DLL y archivos BAT, que se encargan de hacer funcionar los ejecutables sólo para correr aplicaciones bajo Windows y DOS.
**C.** El entorno .NET dispone de ejecutables que llaman a otros ejecutables para mejorar el funcionamiento del sistema operativo y los paquetes de oficina. Éstos funcionan únicamente sobre la plataforma Linux.

Respuesta correcta A: Se ubica entre el sistema y las aplicaciones con las que interactúan los usuarios. Agrupa los principales lenguajes de programación de la familia Microsoft, como Visual Basic .NET, C#, J# y Visual C ++ .NET. También lo acompaña el Framework .NET, que dispone de los controles y funciones necesarios para desarrollar las aplicaciones.

# Ejercicios con lenguajes de programación

## Visual Basic .NET

**EJERCICIO 1:** Realizar una aplicación Windows que permita incluir dos números en distintas cajas de texto y, a la vez, dé la posibilidad de realizar las operaciones básicas de suma, resta, multiplicación y división entre ellos. El resultado debe mostrarse en una etiqueta, y a través de un botón, debe poder copiarse en el Portapapeles de Windows. Para este ejercicio debemos agregar el uso del método **ClipBoard()**, que permite acceder al Portapapeles de Windows.

**EJERCICIO 2:** Realizar una aplicación de consola que nos solicite: nombre y fecha de nacimiento, y a través de este último dato, nos diga qué edad tenemos y los días que llevamos vividos. Incorporaremos el método **DateDiff()**, que permite realizar operaciones entre un rango de fechas u horas y devolver el resultado.

## Visual C #

**EJERCICIO 1:** Crear una aplicación de consola que contenga una clase denominada Monedas, en la que se almacene, en el tipo de variable Dólar, el cambio en pesos; y en otra variable Euro, el cambio en pesos. Para Dólar asignamos un valor de $ 3.20 y para Euro, de $ 4.25. El programa debe preguntarnos de cuántos pesos disponemos para comprar monedas extranjeras, y debe darnos el importe que podamos comprar en euros y en dólares.

**EJERCICIO 2:** Modificar el Ejercicio 1 para que indique, mediante un mensaje en pantalla, que si la compra de euros o dólares da un resultado menor a 100, no es posible realizar la operación, debido a que el mínimo para venta de monedas extranjeras es de 100 unidades.

# Ejercicios con bases de datos

## SQL Server 2005

A continuación, haremos ejercicios prácticos en SQL Server Management Studio 2005 sobre la base de datos Northwind. Si no disponemos de esta base de datos, podremos descargarla desde el sitio web de este curso: http://desarrollador.redusers.com.



**EJERCICIO 1:** En la base de datos Northwind, modificar la tabla **Customers** para que contenga un campo denominado **AlternativeContactName** de longitud 30, y un campo **Comments** con la máxima longitud permitida de texto.

**EJERCICIO 2:** Realizar una consulta de selección de la tabla **Employees**, donde el campo **Title** tenga el valor **'Sales Representative'**, ordenado por el campo **LastName** en forma descendente. Mostrar los resultados en pantalla en formato Texto.

**EJERCICIO 3:** Realizar una consulta de actualización de la tabla **Employees**, donde el campo **Title** tenga el valor **'Sales Representative'**. Cambiar dicho valor por **'Representante de ventas'**. Deberemos utilizar la sentencia **UPDATE** para este propósito.

# ASP.NET

## Introducción a la programación Web

# 4

## Contenidos

Prácticamente con el nacimiento de Internet nace lo que conocemos como programación Web. Desde entonces, se inició la pelea por llevar los sistemas de escritorio a la gran Red de redes. En este capítulo, realizaremos una introducción a los conceptos fundamentales.

## Temas tratados

- Introducción al desarrollo Web
- Introducción a ASP.NET
- Los formularios Web
- Los controles Web

# ASP.NET

Analizaremos cada una de las características del desarrollo de aplicaciones Web a través del entorno Microsoft .NET.

## Introducción al desarrollo Web

Repasaremos aquellos conceptos fundamentales que hacen a la programación de aplicaciones para la Web.

- Modelo cliente-servidor
- HTML y XHTML
- Lenguajes de cliente y lenguajes de servidor
- Problemáticas en el desarrollo Web

## Introducción a ASP.NET

En este capítulo, nos introduciremos en el mundo del desarrollo de aplicaciones Web a través del entorno Microsoft .NET.

- ¿Qué es ASP.NET?
- Problemas que resuelve
- Requerimientos

## Formularios Web

Aprenderemos a trabajar con formularios, uno de los elementos característicos de la programación de sitios Web.

- Conceptos fundamentales
- Eventos
- Ciclo de vida

## Controles Web

Haremos una introducción a estos objetos y luego profundizaremos en cada uno de ellos, indicando sus propiedades y posibles usos.

- Controles simples
- Controles de lista
- Controles de acción
- Controles de ingreso



## Ejercicios prácticos

En cada sección de este capítulo, encontraremos ejemplos prácticos para comenzar a probar cada uno de los conceptos aprendidos.

- Ejemplos
- Casos reales
- Consejos de los expertos

# Desarrollo de aplicaciones Web

Nos introducimos en el mundo de la programación Web. Primero conoceremos cuáles son sus principales características.

El desarrollo de aplicaciones Web consiste en la creación de programas que residen en un servidor Web y que pueden ser utilizados en forma de páginas Web desde un navegador de Internet. Éstos emplean distintas tecnologías basadas en Internet para procesar los datos requeridos por el o los usuarios que están utilizando el sistema. Algunas de ellas son: PHP, ASP, CGI-BIN, ColdFusion y Jsp; cada una precisa software del lado del servidor para analizar el código y generar la página pedida. Dichas tecnologías se combinan en el servidor con otros servicios que permiten acceder a bases de datos y, así, poder armar un sitio altamente funcional y rico en información, sin tener que lidiar a cada momento con él para actualizar las páginas que lo componen.

El usuario que accede al sitio que dispone de páginas programadas con alguna de estas tecnologías siempre verá, a través de su navegador, páginas estáticas. Éstas son armadas sobre la base de la información enviada o solicitada por el usuario, y el resultado es devuelto por el servidor al usuario en forma de una página similar a las HTML.

## Esquema de funcionamiento de páginas Web

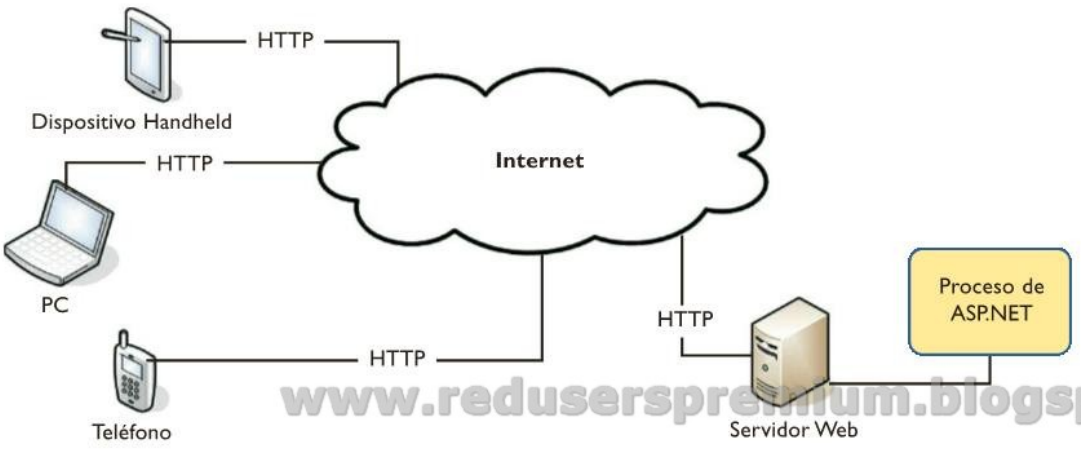





FIGURA 001 | Ejemplo del esquema de funcionamiento de una página Web.

Algunos ejemplos clásicos de aplicaciones Web que utilizamos a diario son: periódicos, foros de información, home banking (banca electrónica), y otros. Cada uno está compuesto por una aplicación que se encarga de procesar el contenido de la base de datos; permitir el alta, baja y modificación de información; y brindar la posibilidad de generar páginas Web que nos permitan hacer las consultas correspondientes.

Los actores fundamentales en el desarrollo de aplicaciones Web son el cliente y el servidor.

## Esquema de programación Web con textos

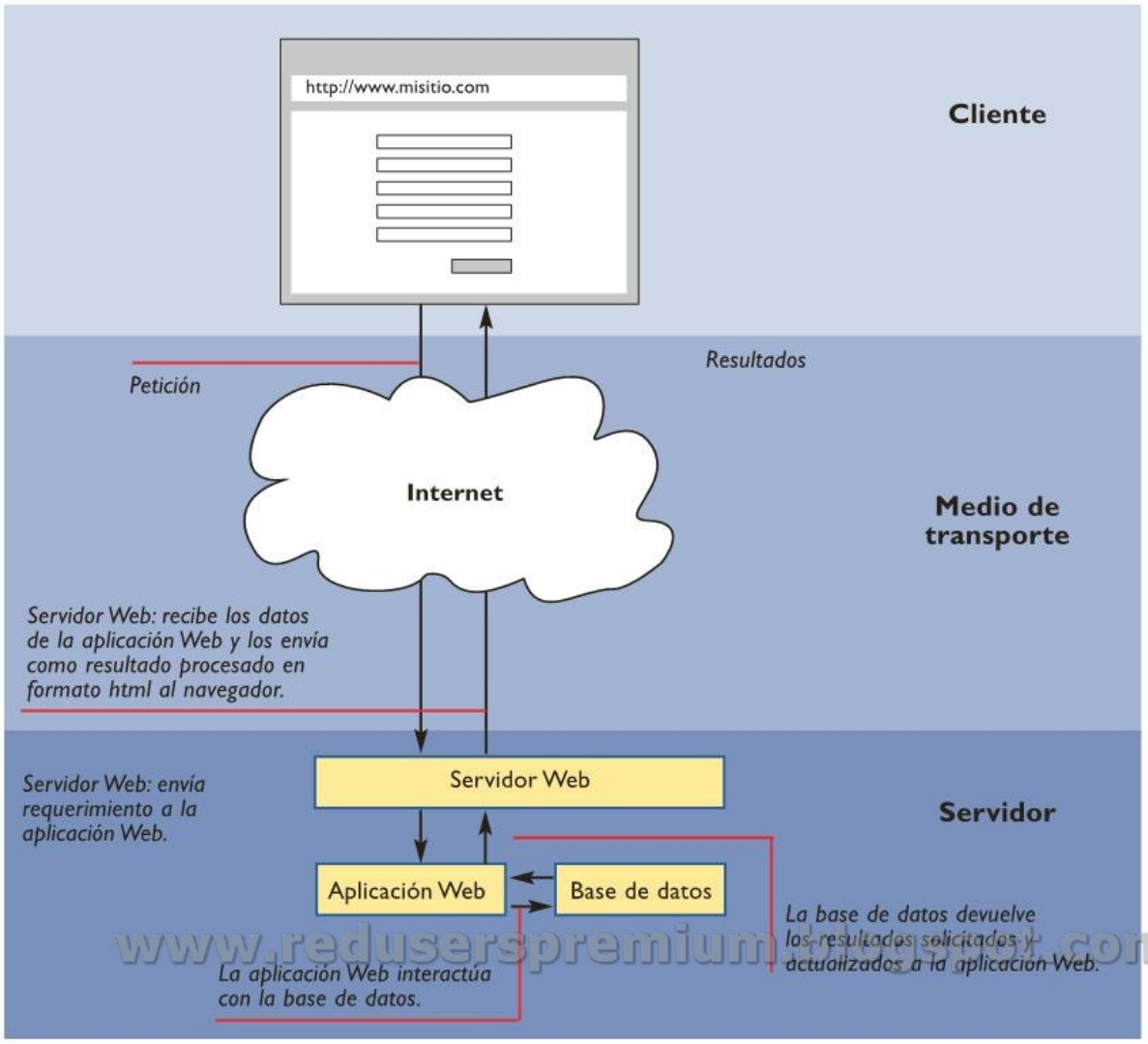





FIGURA 002 | Esquema de programación Web con páginas dinámicas.

EL DESARROLLO DE APLICACIONES WEB CONSISTE EN LA CREACIÓN DE PROGRAMAS QUE RESIDEN EN UN SERVIDOR WEB Y QUE PUEDEN SER UTILIZADOS EN FORMA DE PÁGINAS WEB DESDE UN NAVEGADOR DE INTERNET.

La comunicación entre el servidor y el cliente se realiza mediante el protocolo HTTP. Un cliente solicita una página Web al servidor, que la busca dentro de su base de almacenamiento y la entrega al cliente. Éste visualiza la página recibida utilizando una aplicación llamada navegador. Internet Explorer y Firefox son los programas de este tipo más usados. El navegador reside en la máquina del cliente y sólo interpreta HTML. Para agregar lógica en las páginas Web, se han creado numerosos lenguajes de desarrollo del lado del servidor, mediante los cuales podemos modificar el contenido de las páginas pedidas en función de determinados parámetros. De esta manera, las páginas ya no son entregadas directamente por el servidor, sino que antes son analizadas por software que se ejecuta junto al servidor Web, llamado servidor de aplicaciones.

Los navegadores pueden ejecutar código en la máquina cliente; el ejemplo más conocido es JavaScript. El código cliente se ejecuta en el contexto del navegador, dentro de la máquina del cliente. De esa manera, modifica el contenido de la página descargada sin tener que realizar nuevas peticiones al servidor Web. Desafortunadamente, los navegadores no interpretan el HTML de la misma manera, y tampoco lo hacen con el código cliente (el código escrito en JavaScript, por ejemplo). Un desarrollador tiene que enfrentarse a diario con problemas que incluyen distintos servidores Web, diferentes versiones de los servidores de aplicaciones, diversas versiones de los navegadores y distintos navegadores. Las posibilidades son muchas, y esto significa un gran problema para los programadores. De a poco, gracias al esfuerzo que hacen las empresas para implementar los estándares y a las innovaciones tecnológicas, nos acercamos a modelos de desarrollo en los cuales estas cuestiones tienden a desaparecer.

## Modelo cliente-servidor

En un principio, dijimos que los dos actores más importantes en el desarrollo de aplicaciones Web eran el cliente y el servidor. Veamos este tema con un poco más de profundidad. Un usuario se conecta a un sitio utilizando un navegador. Escribe la dirección única de la página y da la orden de ejecución. En ese instante, el servidor donde reside la página recibe la petición, procesa el pedido y entrega la página en cuestión. Ambos, el cliente y el servidor, "charlan" en HTTP. Cuando la página está lista, es descargada por el cliente, y en ese momento el usuario comienza a recibir la información que la integra.



Este modelo se conoce como modelo cliente-servidor. Las razones de su nombre son obvias, ya que está determinado por los principales actores que forman parte de él.

El servidor es una o varias PCs que contienen software para atender peticiones en HTTP.

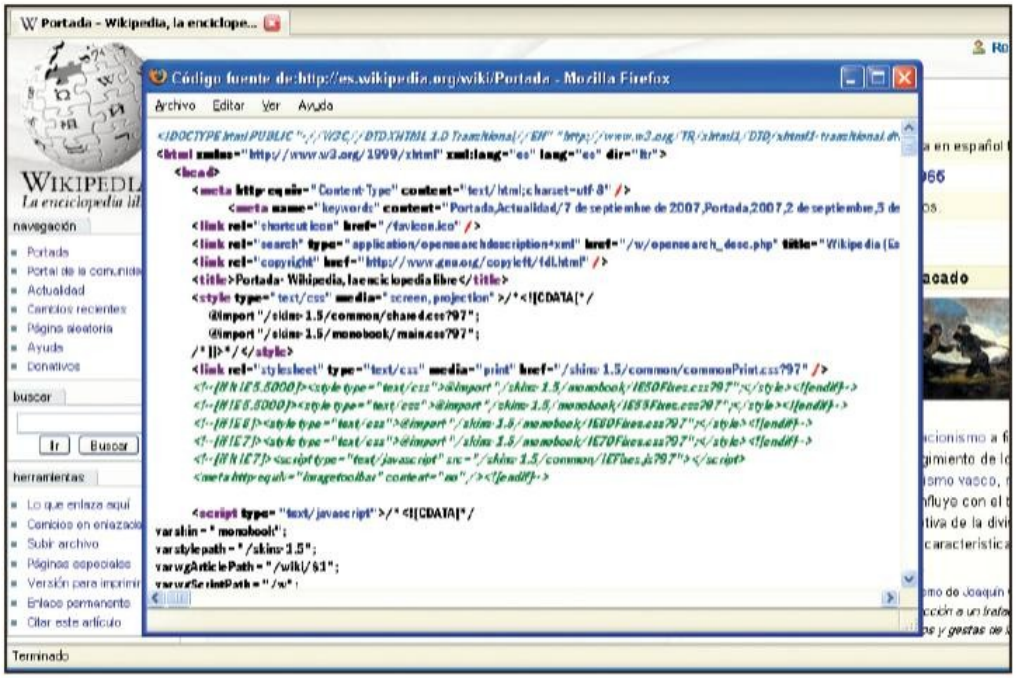



FIGURA 003 | Desde cualquier navegador web, podremos acceder al código fuente de cualquier sitio. En las páginas siguientes de este curso veremos en detalle qué función cumple cada uno de estos elementos.

Los servidores de este tipo se conocen como servidores Web. El cliente puede ser cualquier computadora que ejecuta un sistema operativo y, sobre él, un navegador. El modelo cliente-servidor puede extenderse a otros tipos de clientes, como dispositivos móviles, o incluso a otros servidores. Existen muchas aplicaciones que "sirven" a otros servidores. No necesariamente utilizamos el término "cliente" para referirnos a una persona; debe ser entendido como el que inicia el pedido.

De la misma manera, el servidor es quien recibe el pedido y lo entrega. Debemos acostumbrarnos a pensar estos términos sólo en el contexto indicado. Actualmente, es posible poner un servidor Web dentro de un chip muy pequeño. Y también podemos ejecutar cliente y servidor en la misma máquina, algo que los desarrolladores solemos hacer. Puede ser que un dispositivo móvil tenga una dirección única, cuente con un servidor Web y "sirva", de esa manera, las peticiones de los clientes. El modelo cliente-servidor no implica la existencia de sólo dos elementos; en ocasiones, las "capas" pueden ser varias. El modelo se refiere a la "lógica" y no a la física. Físicamente, el servidor puede estar estructurado como una solución de cluster que contiene muchos equipos comportándose como uno solo; las variaciones son diversas.

PARA AGREGAR LÓGICA EN LAS PÁGINAS WEB SE HAN CREADO NUMEROSOS LENGUAJES DE DESARROLLO DEL LADO DEL SERVIDOR, MEDIANTE LOS CUALES PODEMOS MODIFICAR EL CONTENIDO DE LAS PÁGINAS PEDIDAS EN FUNCIÓN DE DETERMINADOS PARÁMETROS.

Argentina $8,90 (recargo al interior $0,20)) / México: $45

USERS

Microsoft

Curso teórico y práctico de programación

# Desarrollador .net 6

Con toda la potencia de **Visual Basic .NET** y **C#**

La mejor forma de aprender a programar desde cero

★★★★★

Basado en el programa Desarrollador Cinco Estrellas de Microsoft

## ASP.NET

Introducción al desarrollo Web
Lenguajes de Cliente y Servidor

SALIDA LABORAL
La mejor preparación para rendir los exámenes
GARANTIZADA

## Formularios Web

Conceptos - Eventos
Ciclo de vida



ISBN 978-987-1347-43-8
00006
9 789871 347438