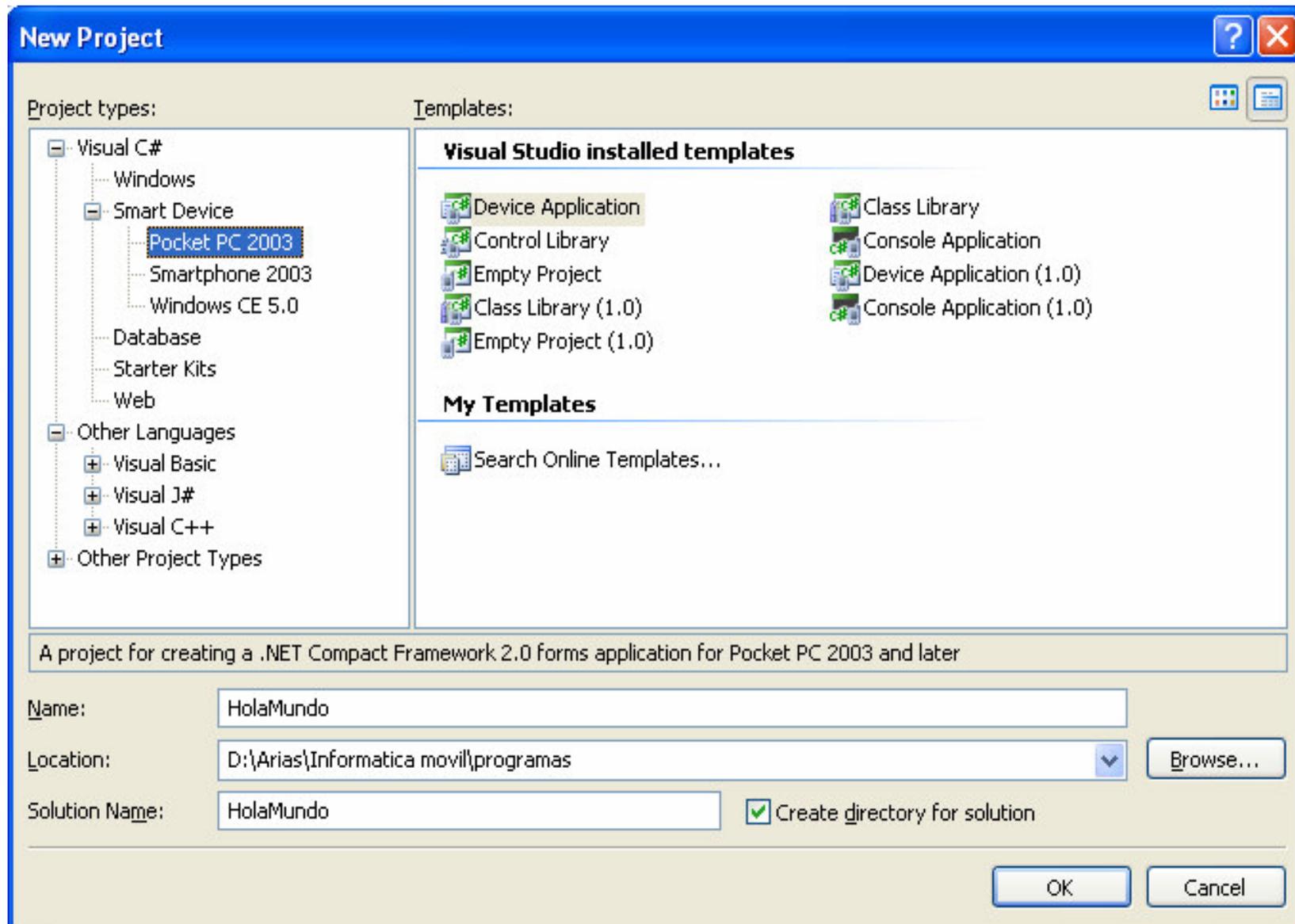


---

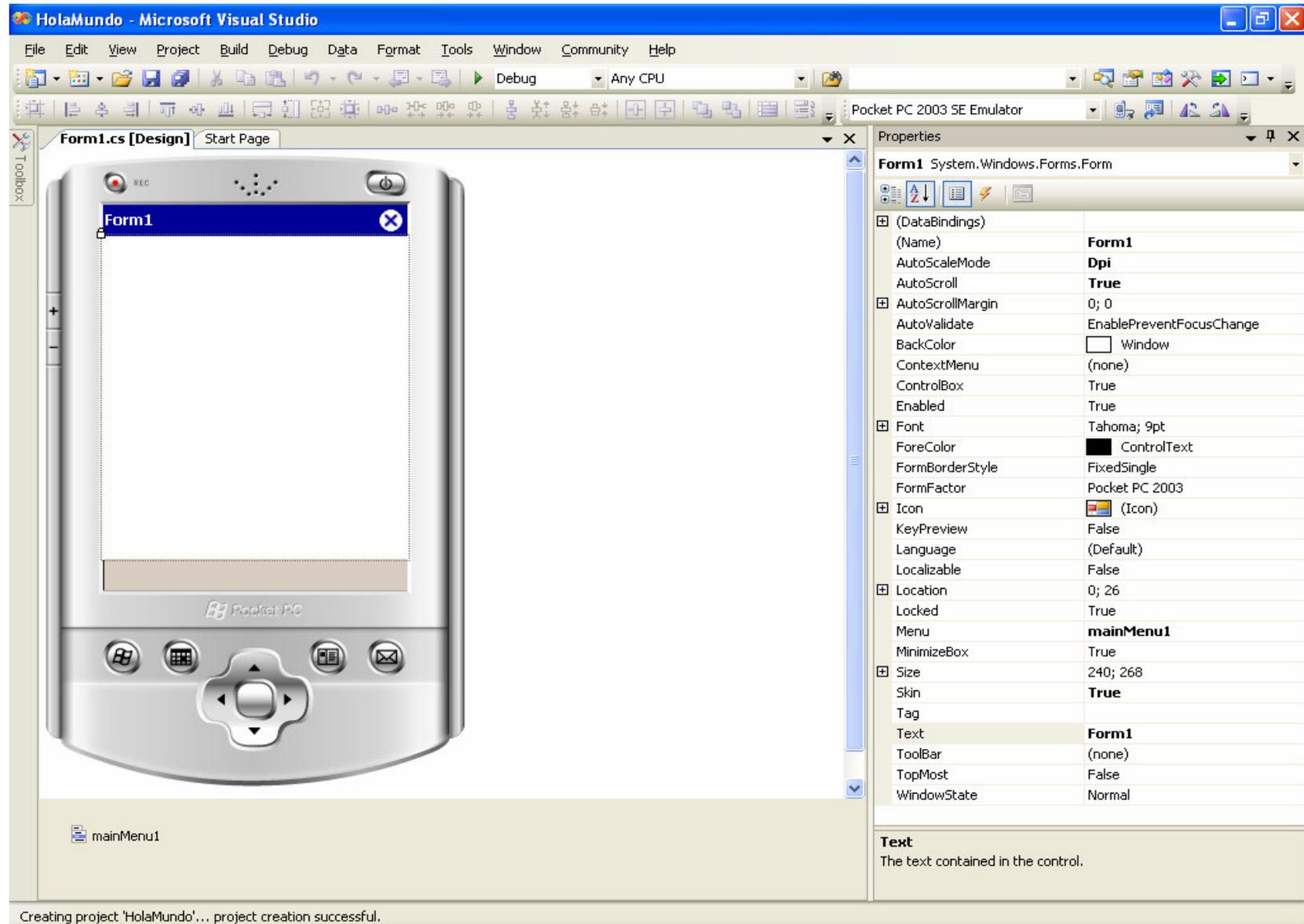
# Desarrollo con Visual Studio .NET Compact Framework para PocketPC



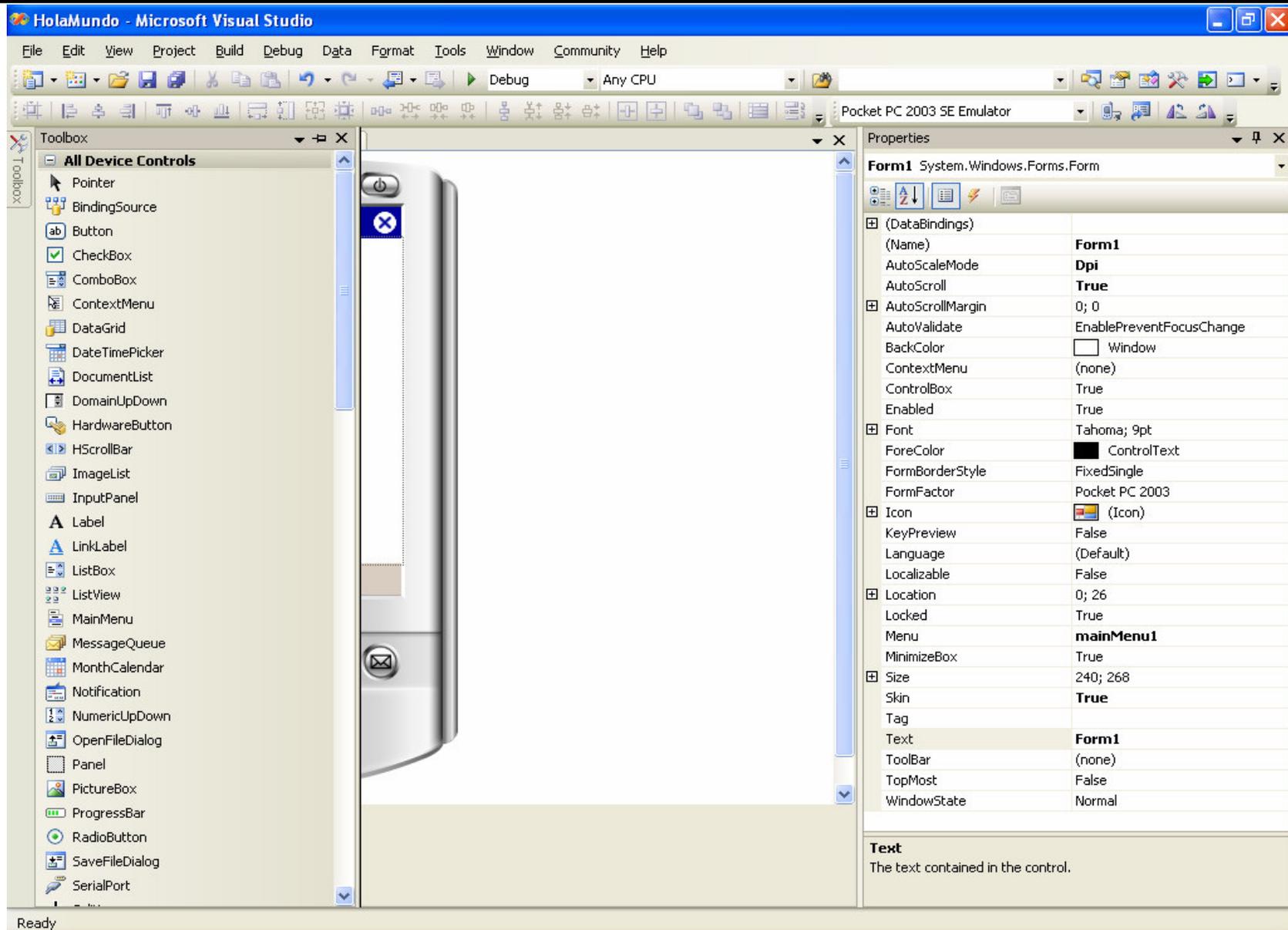
# Creación de un proyecto nuevo



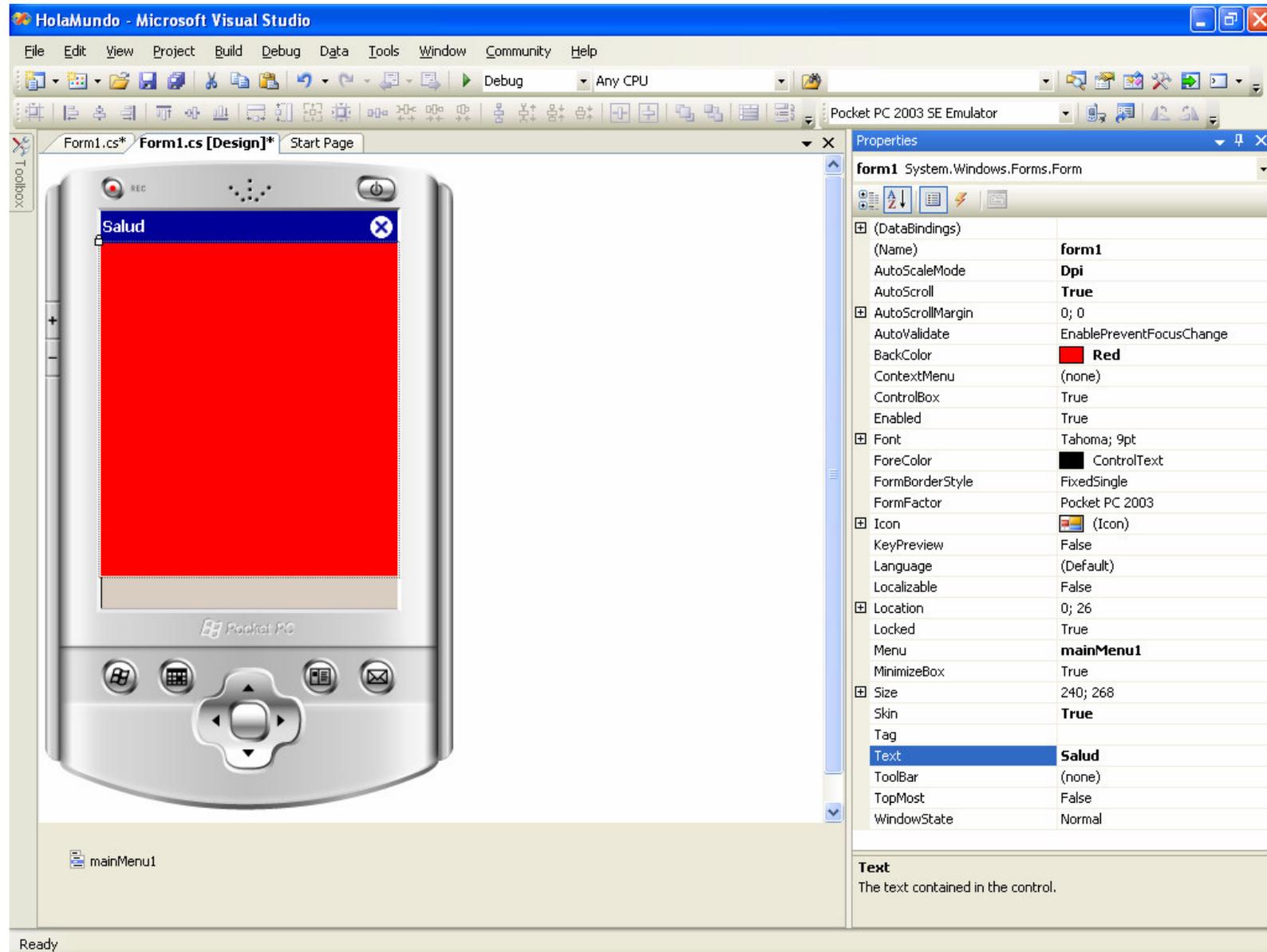
# Entorno de desarrollo



# Cuadro de Herramientas



# Ventana de Propiedades



# Controles del Programa

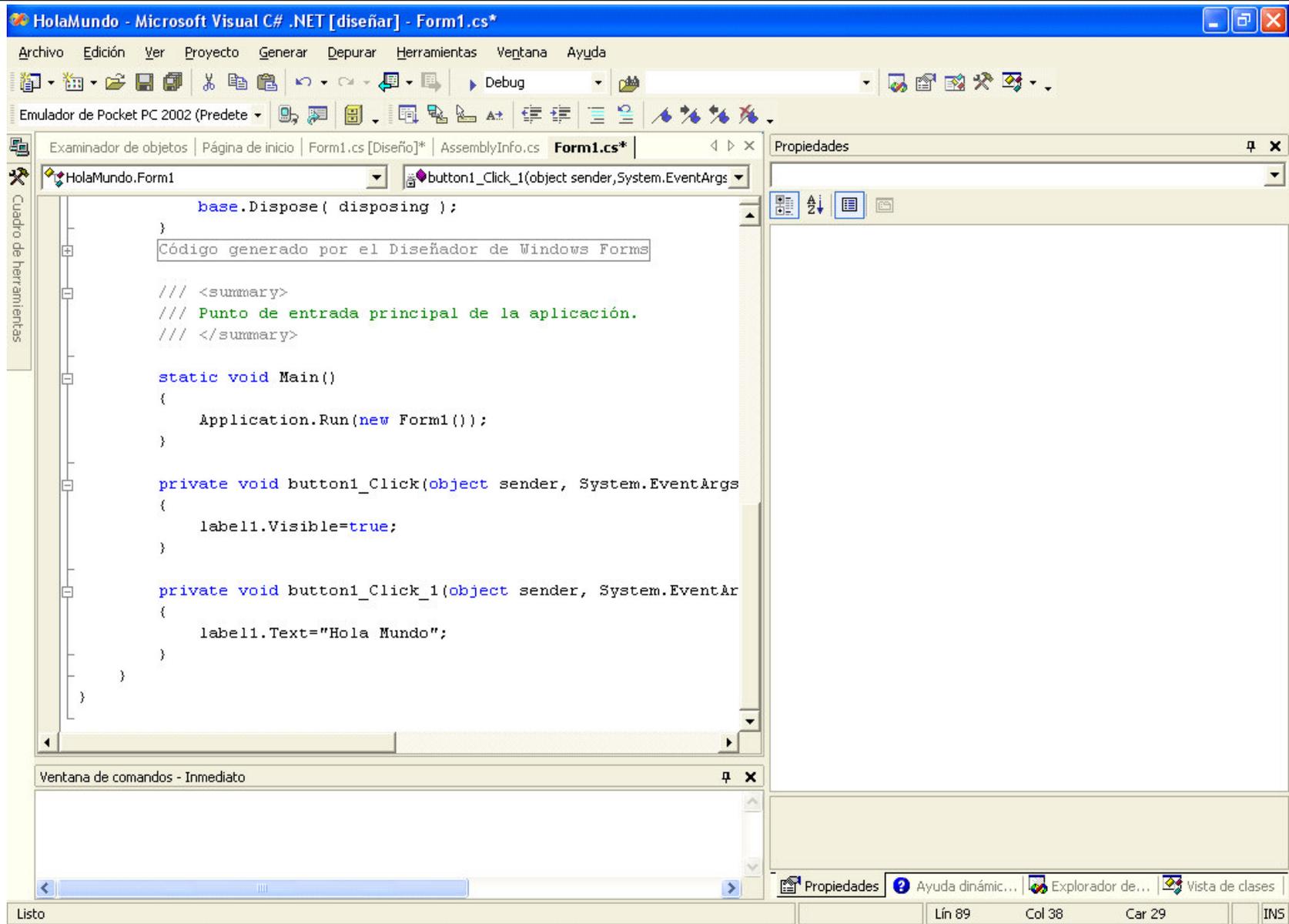
The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Menu Bar:** Archivo, Editar, Ver, Proyecto, Generar, Depurar, Datos, Formato, Herramientas, Ventana, Comunidad, Ayuda.
- Toolbox:** Located on the left side, labeled 'Cuadro de herramientas'.
- Designer View:** Shows a mobile device emulator with a screen titled 'Saludar'. The screen has a red background and a button labeled 'Saluda' at the bottom.
- Properties Window:** Located on the right, showing properties for 'label1 System.Windows.Forms.Label'.

Property	Value
(Name)	label1
Anchor	Top, Left
BackColor	Red
ContextMenu	(ninguno)
Dock	None
Enabled	True
Font	Tahoma; 9pt
ForeColor	ControlText
GenerateMember	True
Location	53; 48
Locked	False
Modifiers	Private
Size	100; 20
Tag	
Text	
TextAlign	TopLeft
Visible	True
- Error List:** Located at the bottom, showing '0 errores', '0 advertencias', and '0 mensajes'.

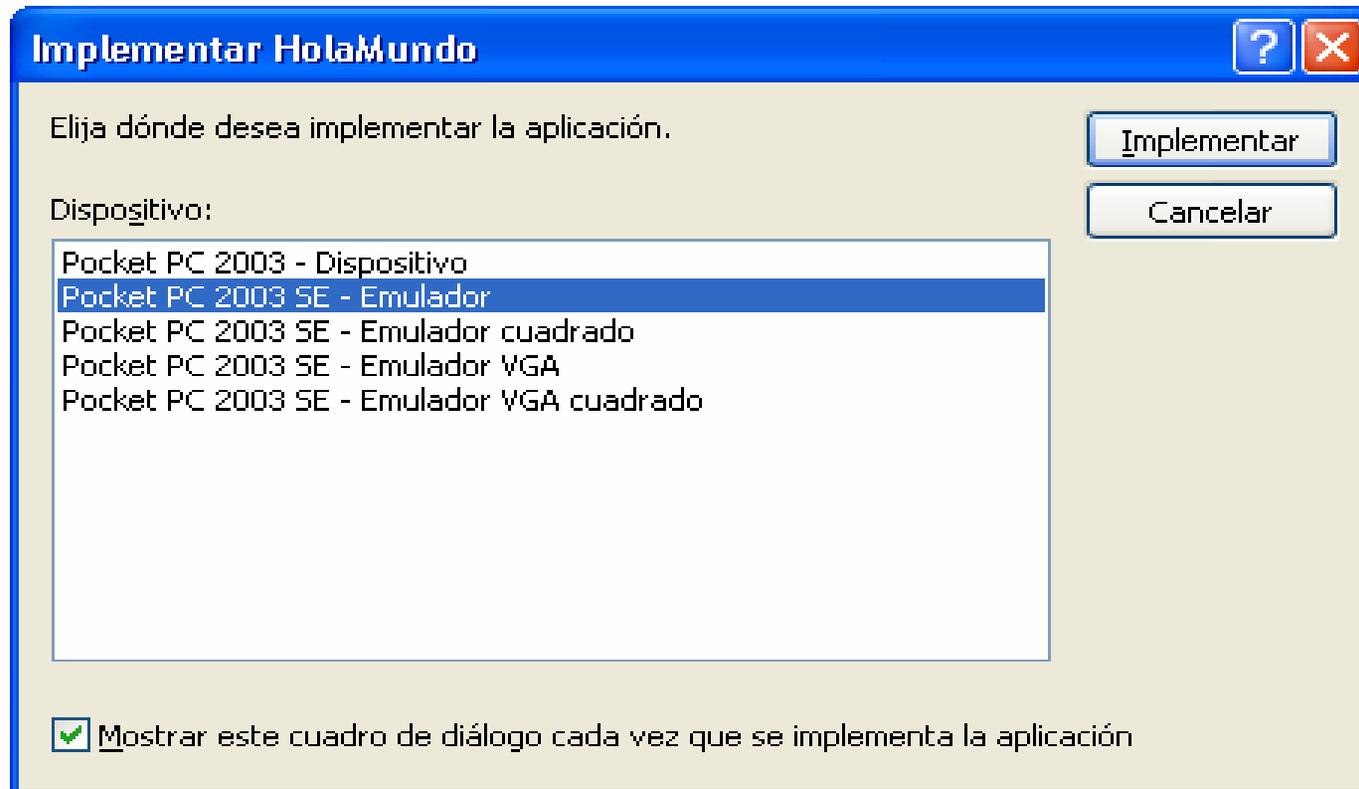


# Ventana de edición de código



# Selección del lugar de ejecución

---



# Ejecución en el simulador

---



# Código generado por Visual Studio I. Form1.cs

---

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
```

```
namespace HolaMundo
```

```
{
    public partial class form1 : Form
    {
        public form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            label1.Text = "Hola Mundo";
        }
    }
}
```

Se reparte la definición  
en varios archivos

Constructor

Delegado para  
el botón



# Código generado por Visual Studio II. Form1.designer.cs

---

```
namespace HolaMundo
{
    partial class form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        private System.Windows.Forms.MainMenu mainMenu1;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code
```

Continúa la declaración del form



# Código generado por Visual Studio II. Form1.designer.cs

```
private void InitializeComponent()
```

```
{
```

```
    this.mainMenu1 = new System.Windows.Forms.MainMenu();  
    this.button1 = new System.Windows.Forms.Button();  
    this.label1 = new System.Windows.Forms.Label();
```

Añade  
los controles

```
    //
```

```
    // button1
```

```
    //
```

Propiedades específicas

```
    this.button1.Location = new System.Drawing.Point(79, 177);  
    this.button1.Name = "button1";  
    this.button1.Size = new System.Drawing.Size(72, 20);  
    this.button1.TabIndex = 0;  
    this.button1.Text = "Saludar";  
    this.button1.Click += new System.EventHandler(this.button1_Click);
```

```
    //
```

```
    // label1
```

```
    .....
```

```
    // form1
```

```
    //
```

```
    this.BackColor = System.Drawing.Color.Red;  
    this.ClientSize = new System.Drawing.Size(240, 268);  
    this.Controls.Add(this.label1);  
    this.Controls.Add(this.button1);  
    this.Menu = this.mainMenu1;  
    this.Name = "form1";  
    this.Text = "Saludar";
```

```
}
```



# Código generado por Visual Studio III. Program.cs

---

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace HolaMundo
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [MTAThread]
        static void Main()
        {
            Application.Run(new form1());
        }
    }
}
```

Cuerpo del programa



# Detalles en el desarrollo



- Los formularios se ejecutan siempre a pantalla completa.
- La X minimiza pero no detiene el programa.
- Si queremos que la aplicación se detenga, se cambia la propiedad **MinimizeBox** de **true** a **false**.
- La X pasa a ser un OK



# Recomendaciones Microsoft

---

- No debería de haber más de una ocurrencia de la aplicación ejecutándose al mismo tiempo. Lo garantiza el sistema de ejecución reactivando una instancia ya creada en vez de crear una nueva.
- Si la aplicación utiliza el SIP (“*Software Input Panel*”) hay que tener en cuenta que ocupa los 80 bits inferiores de la pantalla.
- Una vez que una instancia ha sido creada, el usuario no debería de ser capaz de eliminarla de memoria o detenerla.
- Si se ejecuta una nueva aplicación, el formulario de la nueva oculta al de la anterior, haciéndola pasar a segundo plano y *desactivándola*. Al pulsar sobre la aplicación de nuevo, el sistema la *reactivará*, de nuevo.
- Las aplicaciones deberían liberar el máximo de recursos cuando son desactivadas ya que pueden estar mucho tiempo en ese estado.



# Recomendaciones Microsoft

---

- Cuando una aplicación es desactivada se genera el evento **Form.Deactivate**. Cuando se activa, se genera **Form.Activate**.
- Se puede (debe) incluir código para gestionar esos eventos:

```
private void Form1_Activate(object sender, System.EventArgs e)
{
    // código necesario para recuperar la activación
}
private void Form1_Deactivate(object sender, System.EventArgs e)
{
    // código necesario para liberar recursos
}
```



# Pantalla y Teclado

- Para mostrar un mensaje por pantalla, además de modificar las propiedades de los controles que usemos, podemos abrir una ventana:

**MessageBox.Show** .- Método sobrecargado. Retorna un valor de la enumeración **DialogResult**.

- El teclado direccional genera eventos en el formulario cuando se pulsan sus botones. El manejador recibe un evento del tipo **KeyEventArgs** que tiene la propiedad **KeyCode**. Los valores de esa propiedad son los mostrados en la tabla
- El control **InputPanel** corresponde al SIP. Si la propiedad **Enabled** es **true**, el SIP está desplegado. Genera el evento **EnabledChanged** cuando cambia el valor de la propiedad.

<b>Keys.Up</b>	Se pulsó el botón superior
<b>Keys.Down</b>	Se pulsó el botón inferior
<b>Keys.Left</b>	Se pulsó el botón izquierdo
<b>Keys.Right</b>	Se pulsó el botón derecho
<b>Keys.Return</b>	Se pulsó el botón central



# Controles

---

- El **RadioButton** es un control que se usa para mostrar un listado de elecciones mutuamente excluyentes. Cuando se selecciona uno, los demás se deseleccionan.
- Eventos expuestos:
  - **Click**.- Se activa cuando el usuario pulsa con el lápiz en el botón. No se activa si se cambia el estado por programa.
  - **CheckedChanged**.- Se lanza cuando el estado del botón cambia, bien por programa bien gráficamente.
- El **CheckBox** es un control similar con la diferencia de que sí pueden estar seleccionados varios a la vez.
- La propiedad **CheckState** determina si está seleccionado.
- Tiene tres estados: **Checked**, **Unchecked** y **Indeterminate**. Este último se puede usar cuando la propiedad **ThreeState** está a **true**. En ese caso, el control está marcado pero en gris.
- El cambio de estado se controla con el evento **CheckStateChanged**.



# Controles

- El **ComboBox** se usa para presentar una lista de opciones en una cantidad restringida de espacio de pantalla. Las opciones aparecen en una línea con una flecha lateral. Cuando se pulsa la flecha, las opciones se despliegan.
- Para meter opciones se utiliza la propiedad **Items**. También se puede añadir y borrar por programa invocando los métodos **Items.Add** y **Items.Remove**
- Para conocer el elemento seleccionado hay dos posibilidades:
  - Vía índice usando la propiedad **SelectedIndex**.
  - Usando la propiedad **SelectedItem**.
- Eventos expuestos:
  - **SelectedIndexChanged**.- Se activa cuando el usuario selecciona un elemento de la lista
- El control **ListBox** comparte las mismas propiedades y eventos que el **ComboBox**, pero se usa en los casos en los que se dispone de espacio para mostrar las opciones.



# Controles: **TextBox**

Propiedades

textBox1 System.Windows.Forms.TextBox

Apariencia

BackColor	Window
Font	Microsoft Sans Serif; 8,25pt
ForeColor	WindowText
ScrollBars	None
<b>Text</b>	<b>textBox1</b>
TextAlign	Left

Comportamiento

AcceptsReturn	False
AcceptsTab	False
ContextMenu	(ninguno)
Enabled	True
MaxLength	32767
Multiline	False
<b>PasswordChar</b>	
ReadOnly	False
Visible	True
WordWrap	True

Diseño

(Name)	textBox1
Location	<b>88; 48</b>
Locked	False
Modifiers	Private
Size	100; 20

**Text**

Texto que contiene el control.

Propiedades

textBox1 System.Windows.Forms.TextBox

Evento

KeyDown	
KeyPress	
KeyUp	

Foco

GotFocus	
LostFocus	
Validated	
Validating	

La propiedad cambió

EnabledChanged	
ParentChanged	
TextChanged	

**KeyDown**

Tiene lugar cuando se presiona una tecla por primera vez.



# Controles: **NumericUpDown**

Propiedades

numericUpDown1 System.Windows.Forms.NumericUpDown

Value 0

Increment 1

Maximum 100

Minimum 0

(Name) numericUpDown1

Location 56; 56

Locked False

Modifiers Private

Size 100; 20

(Name)  
Indica el nombre utilizado en el código para identificar el objeto.

Propiedades

numericUpDown1 System.Windows.Forms.NumericUpDown

ValueChanged

Se lanza por programa o cuando se usan las flechas. No cuando el usuario escribe en el control

ValueChanged  
Tiene lugar cuando cambia el valor del control numérico de flechas.



# Controles: DomainUpDown

Two screenshots of the Visual Studio Properties window for a `DomainUpDown` control. The left screenshot shows the 'Text' and 'Items' properties circled in red. The right screenshot shows the 'SelectedIndexChanged' event circled in red. Yellow callout boxes explain the 'Text' and 'SelectedIndexChanged' properties.

**Text**  
Texto que contiene el control.

**SelectedIndexChanged**  
Tiene lugar cuando el elemento seleccionado en el control cambia.

Texto que contiene el control

Lista de opciones que presentará

La propiedad **SelectedIndex** contiene el índice del elemento seleccionado.



# Controles: ProgressBar

The image displays two screenshots of the Visual Studio Properties window for a `ProgressBar` control. The left screenshot shows the **Comportamiento** (Behavior) tab, where the **Value** property is highlighted in blue and circled in red. The right screenshot shows the **Foco** (Focus) tab, where the **ParentChanged** event is highlighted in blue.

Propiedad	Valor
ContextMenu	(ninguno)
Enabled	True
Maximum	100
Minimum	0
Value	0
Visible	True

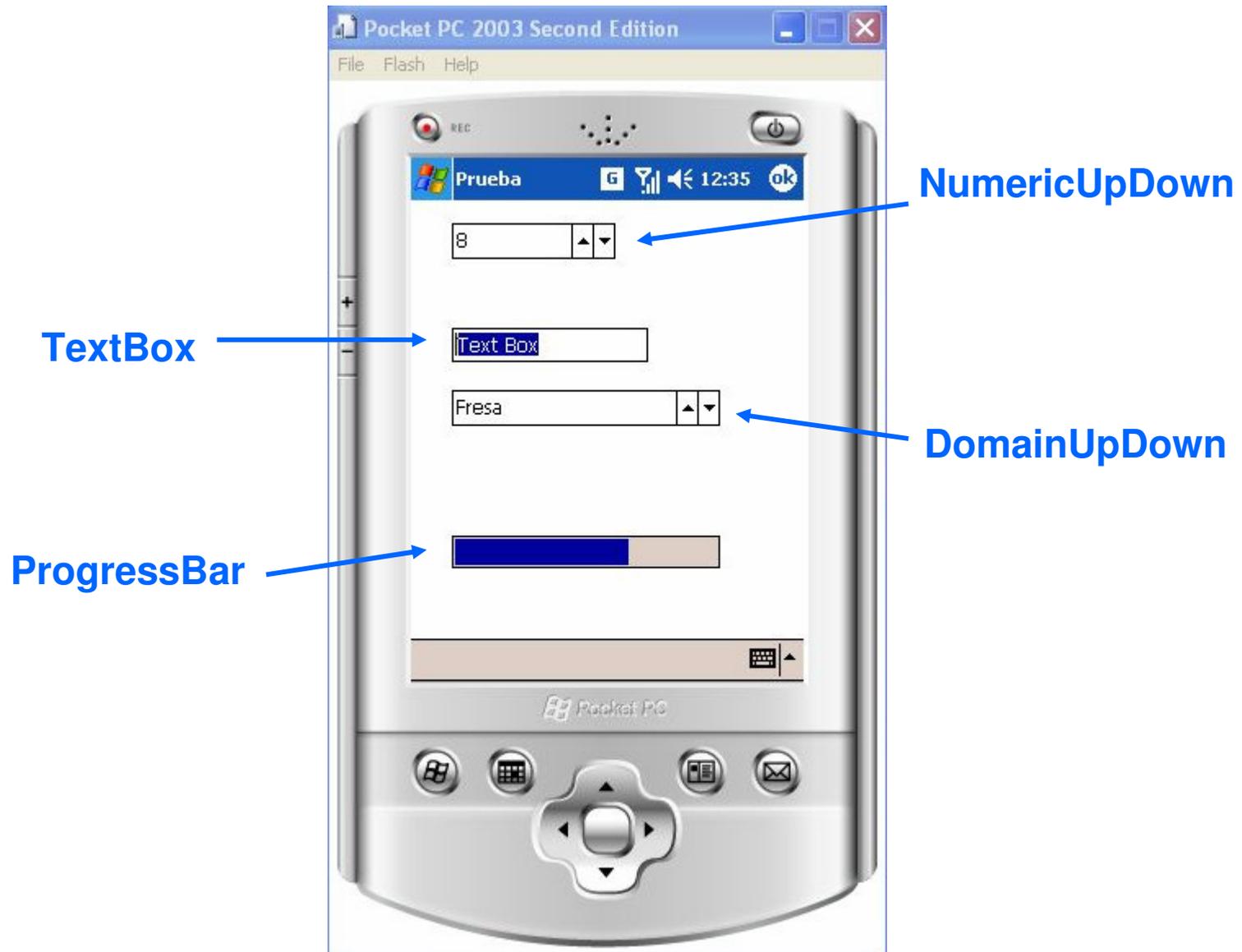
Evento	Descripción
Validated	
Validating	
La propiedad cambió	
EnabledChanged	
ParentChanged	

**Value**  
Valor actual de ProgressBar, en el intervalo especificado por las propiedades de mínimo y máximo.

**ParentChanged**  
Evento que se desencadena cuando se cambia el valor de la propiedad Parent de Control



# Controles



# Controles: **TrackBar**

**Propiedades** trackBar1 System.Windows.Forms.TrackBar

Categoría	Propiedad	Valor
Apariencia	Orientation	Horizontal
	SmallChange	1
	TickFrequency	1
	LargeChange	5
Comportamiento	ContextMenu	(ninguno)
	Enabled	True
	Maximum	10
	Minimum	0
Diseño	(Name)	trackBar1
	Location	48; 80
	Locked	False
	Modifiers	Private
	Size	136; 45
	Visible	True

**Propiedades** trackBar1 System.Windows.Forms.TrackBar

Categoría	Evento
Acción	ValueChanged
Evento	GotFocus
	LostFocus
	Validated
	Validating
La propiedad cambió	EnabledChanged
	ParentChanged

**Cambios en el indicador cuando se pulsa una tecla o cuando se pulsa la barra.**

**Distancia entre las marcas de la barra.**

Posición del control deslizante.

Tiene lugar cuando el valor del control cambia.



# Controles: HScrollBar - VScrollBar

**Propiedades** hScrollBar1 System.Windows.Forms.HScrollBar

Comportamiento	
ContextMenu	(ninguno)
Enabled	True
LargeChange	10
Maximum	91
Minimum	0
SmallChange	1
Value	0
Visible	True

**Propiedades** hScrollBar1 System.Windows.Forms.HScrollBar

Acción	
ValueChanged	

La propiedad cambió	
EnabledChanged	
ParentChanged	

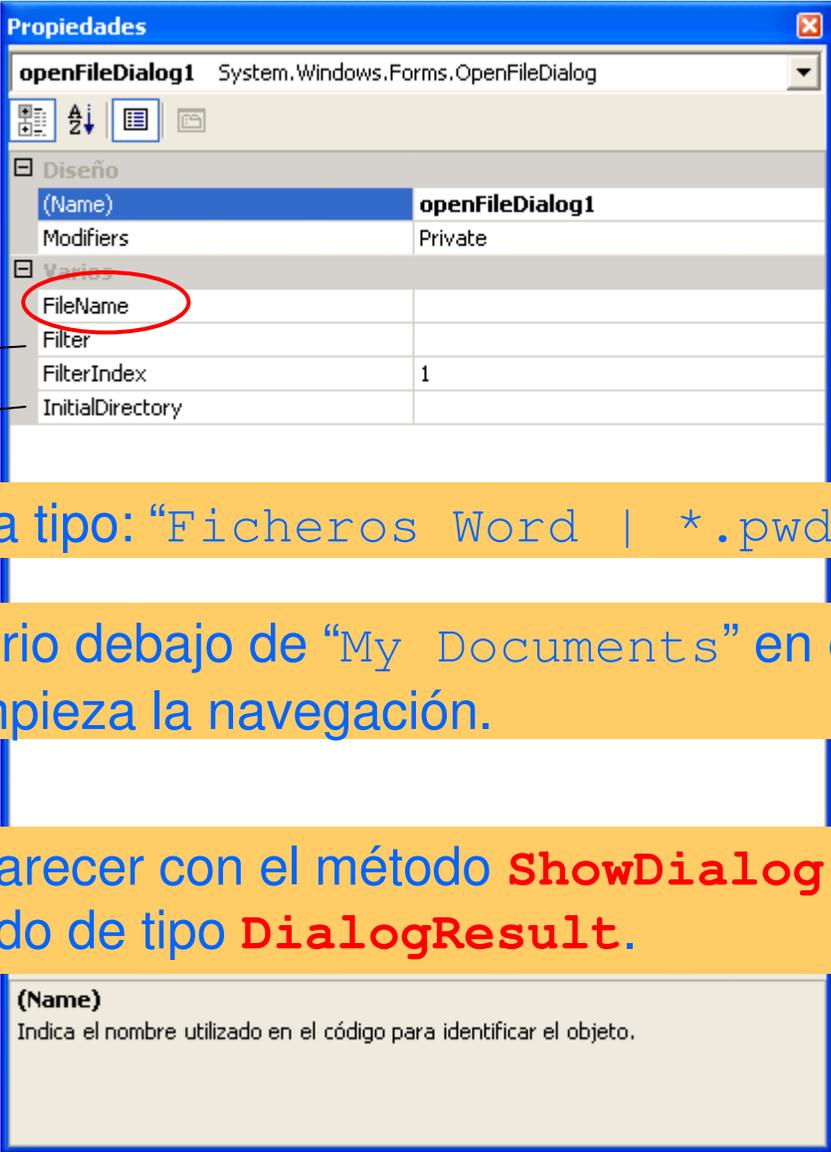
**Value**  
Posición del indicador.

**ValueChanged**  
Tiene lugar cuando el valor del control cambia.

Igual que en el caso anterior



# Controles: OpenFileDialog - SaveFileDialog



Propiedades	
openFileDialog1 System.Windows.Forms.OpenFileDialog	
Diseño	
(Name)	openFileDialog1
Modifiers	Private
Propiedades	
FileName	
Filter	
FilterIndex	1
InitialDirectory	

Cadena tipo: "Ficheros Word | \*.pwd"

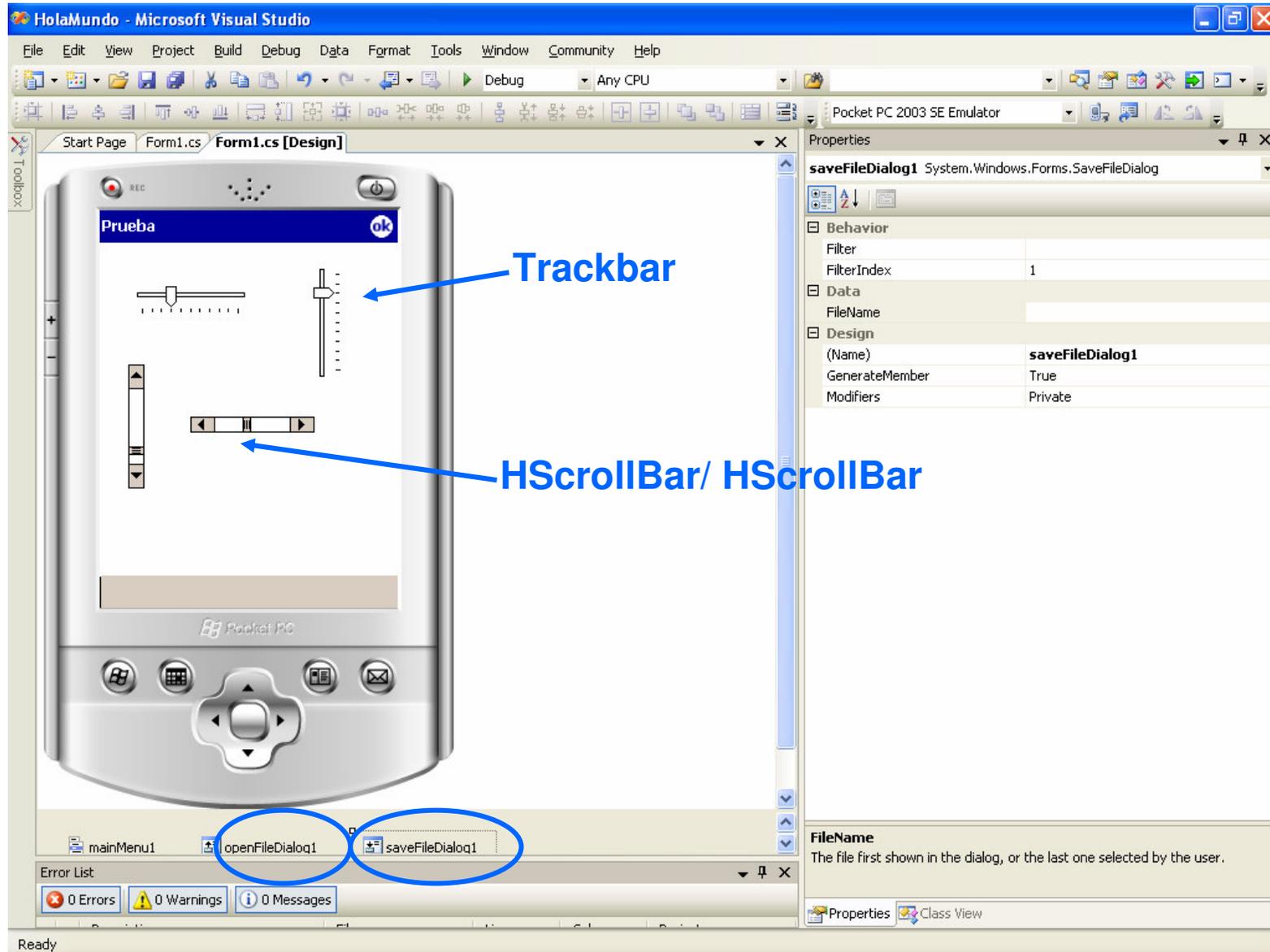
Directorio debajo de "My Documents" en el que empieza la navegación.

Se hacen aparecer con el método **ShowDialog** que retorna un enumerado de tipo **DialogResult**.

(Name)  
Indica el nombre utilizado en el código para identificar el objeto.



# Controles



# Controles: **Timer**

**Intervalo de activación, en milisegundos (cuando `Enabled` está a `true`)**

**Se lanza este evento cada vez que se cumple el intervalo**

Propiedad	Valor
Enabled	False
Interval	100
(Name)	timer1
Modifiers	Private

**Interval**  
Frecuencia de los eventos Elapsed en milisegundos.

**Tick**  
Tiene lugar cuando ha transcurrido el intervalo de tiempo especificado.



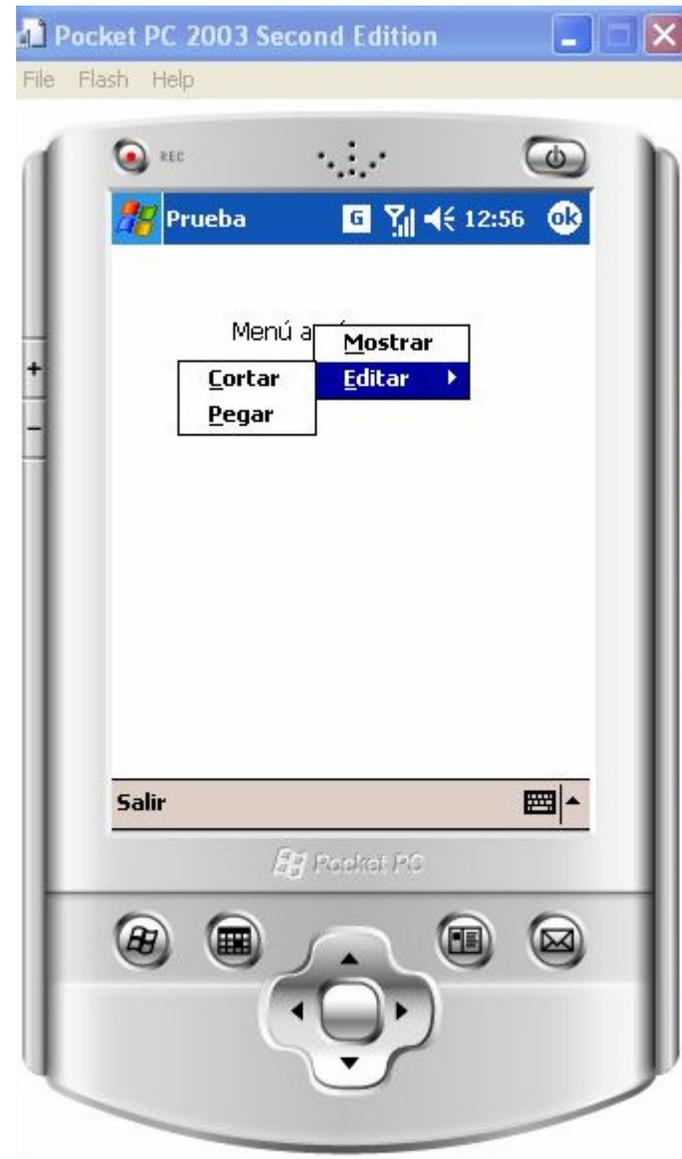
# Controles: **StatusBar** – **MainMenu** – **ContextMenu**

---

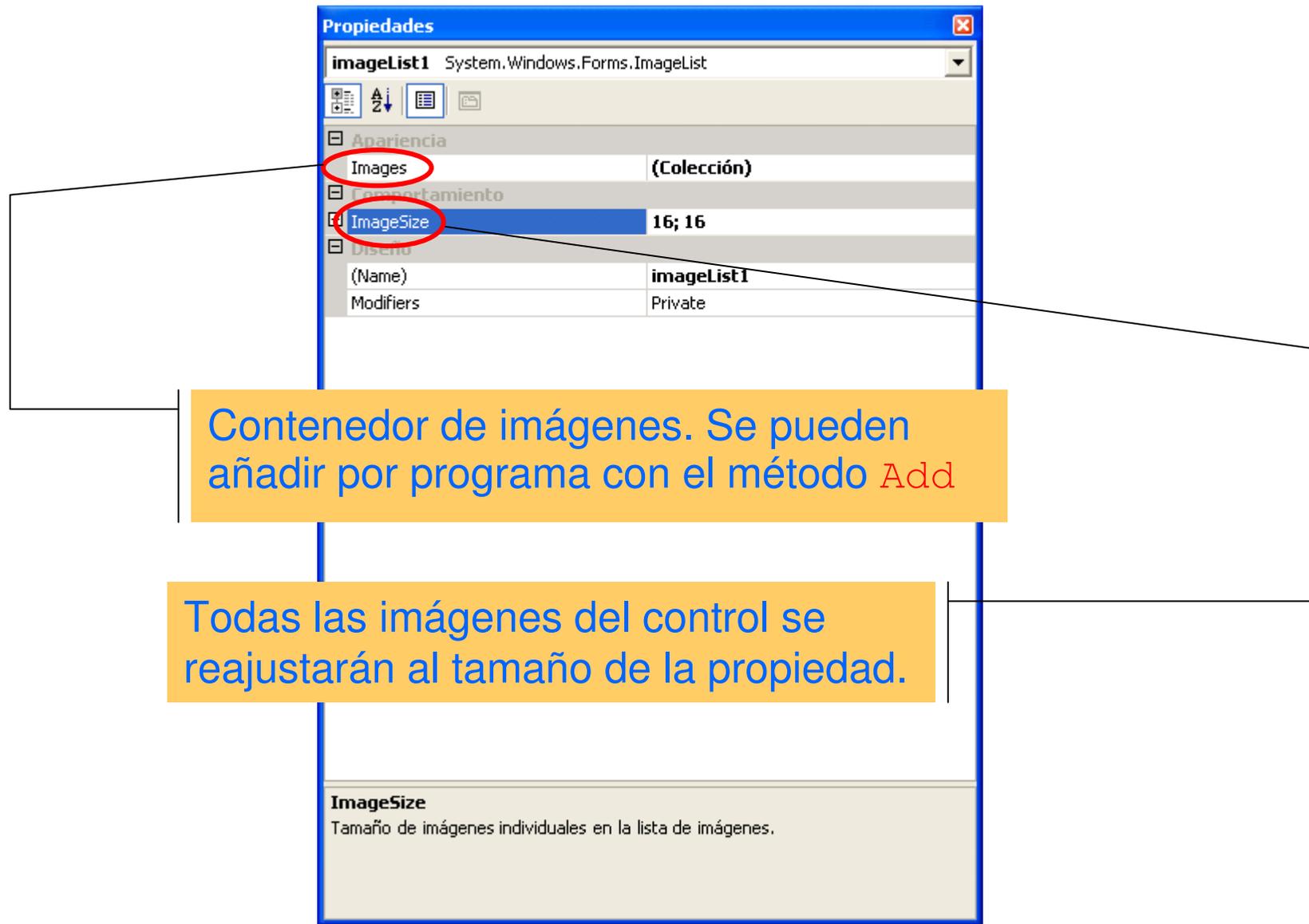
- No tienen representación gráfica en el Diseñador de Formularios. Aparecen en la parte baja del formulario fuera de la zona de pantalla.
- El control **StatusBar** es el área de información que se muestra (si se desea) encima del menú principal.
- El control **MainMenu** existe siempre por defecto ya sin él no se puede usar el SIP. Cada opción del Menú es un control **MenuItem** y se van añadiendo de manera gráfica.
- Los **MenuItem** pueden contener otros **MenuItem** dando lugar a submenús.
- Para incluir un separador en el menú, simplemente se crea un **MenuItem** asignándole a la propiedad **Text** el carácter '-'.  
• El control **ContextMenu** es prácticamente idéntico en su funcionalidad al **MainMenu** excepto que está asociado a otros controles. Es el típico menú que aparece cuando se pulsa y se mantiene.
- Para añadirle opciones al menú contextual se incorporan controles **MenuItem** de manera similar a como se hace en el **MainMenu** .



# Controles



# Controles: ImageList



Propiedades

imageList1 System.Windows.Forms.ImageList

Images (Colección)

ImageSize 16; 16

(Name) imageList1

Modifiers Private

Contenedor de imágenes. Se pueden añadir por programa con el método *Add*

Todas las imágenes del control se reajustarán al tamaño de la propiedad.

**ImageSize**  
Tamaño de imágenes individuales en la lista de imágenes.



# Controles: **ToolBar**

**Buttons** (Colección)

**ImageList**

**ButtonClick**

Contiene las imágenes de la barra de botones

Configura el aspecto de cada botón

Se activa cuando se pulsa un botón. Recibe como parámetro el objeto `ToolBarButtonClickEventArgs` cuya propiedad `Button` nos indica que botón ha sido pulsado:  
`if (e.Button == this.toolBarButton1)`

**Buttons**  
Colección de botones `ToolBarButtons` que forman esta barra de herramientas.

**ButtonClick**  
Tiene lugar cuando el usuario hace clic en un botón de la barra de herramientas.



# Controles: PictureBox

Las imágenes pueden estar en un fichero, en el propio ensamblado o en un control `ImageList`. El control `ImageList` permite cambiar el tamaño a las imágenes antes de presentarlas.

**Image**  
Imagen mostrada en el control PictureBox.

**ParentChanged**  
Evento que se desencadena cuando se cambia el valor de la propiedad Parent de Control



# Controles: ListView

**Propiedades** listView1 System.Windows.Forms.ListView

Categoría	Propiedad	Valor	
Apariencia	CheckBoxes	False	
	FullRowSelect	False	
	View	LargeIcon	
	Comportamiento	Columns	(Colección)
		ContextMenu	(ninguno)
		Enabled	True
HeaderStyle	Clickable		
Items	(Colección)		
LargeImageList	(ninguno)		
SmallImageList	(ninguno)		
Visible	True		
Diseño	(Name)	listView1	

**Propiedades** listView1 System.Windows.Forms.ListView

Categoría	Propiedad	Valor
Acción	ColumnClick	
	ItemActivate	
	ItemCheck	
Comportamiento	SelectedIndexChanged	
	...	
Foco	GotFocus	
	LostFocus	
	Validated	
	Validating	
	La propiedad cambió	EnabledChanged
ParentChanged		

Cuatro estilos: Details, LargeIcon, List, SmallIcon

Colección de imágenes a mostrar.

Elementos del control. Se asocian a las imágenes.

La propiedad `SelectedIndices` contiene los índices de los elementos seleccionados(1). También se puede usar la propiedad `ListView.Selected` para localizar el elemento seleccionado.



# Controles: TreeView

**Propiedades** treeView1 System.Windows.Forms.TreeView

Categoría	Propiedad	Valor	
Apariencia	Checkboxes	False	
	ContextMenu	(ninguno)	
	Enabled	True	
	ImageIndex	<input type="checkbox"/> (ninguno)	
	ImageList	(ninguno)	
	Nodes	(Colección)	
	SelectedImageIndex	<input type="checkbox"/> (ninguno)	
	ShowLines	True	
	ShowPlusMinus	True	
	ShowRootLines	True	
Comportamiento	AfterCheck		
	AfterSelect		
	GotFocus		
	LostFocus		
	Validated		
	Validating		
	La propiedad cambió		
	EnabledChanged		
	ParentChanged		

Contiene los nodos y subnodos del árbol. Se introducen en un control visual.

Si queremos imágenes al lado de los nodos.

Recibe el objeto **TreeViewEventArgs** que contiene la propiedad **Node** con el nodo activado. También **Action** que indica el cómo: **ByKeyboard**, **ByMouse**, **Collapse**, **Expand**, **Unknow**.

**AfterSelect**  
Tiene lugar cuando cambia la selección.



# Controles: TabControl

**Propiedades** tabControl1 System.Windows.Forms.TabControl

**Propiedades** tabControl1 System.Windows.Forms.TabControl

**Comportamiento**

**SelectedIndexChanged**

**Diseño**

**TabPage** (Colección)

Recibe la propiedad `SelectedIndex` indicando que página está activa.

Añade páginas de controles al proyecto.

**TabPage**  
TabPage en TabControl.

**SelectedIndexChanged**  
Tiene lugar cuando cambia la propiedad 'SelectedIndex' de este control.



# Controles: DataGrid

Define el aspecto de los datos presentados. La propiedad `DataSource` indica de dónde se obtienen.

Recibe como parámetro la posición X e Y de la pulsación. El método `HitTest` del `DataGrid` retorna la fila y columna pulsada. También retorna la propiedad `Type` que indica si es una cabecera de columna, fila, etc.



# Controles: LinkLabel

Propiedades

linkLabel2 System.Windows.Forms.LinkLabel

(DataBindings)	
(Name)	linkLabel2
Anchor	Top, Left
BackColor	Window
ContextMenu	(ninguno)
Dock	None
Enabled	True
Font	Tahoma; 9pt; style=Underlin
ForeColor	Blue
GenerateMember	True
Location	73; 57
Locked	False
Modifiers	Private
Size	100; 20
TabIndex	1
TabStop	True
Tag	
<b>Text</b>	<b>Panel 1</b>
TextAlign	TopLeft
Visible	True

**Text**  
Texto que contiene el control.

Propiedades Explorador de soluciones

Propiedades

linkLabel2 System.Windows.Forms.LinkLabel

(DataBindings)	
<b>Click</b>	<b>linkLabel2_Click</b>
EnabledChanged	
GotFocus	
LostFocus	
ParentChanged	
TextChanged	

**Click**  
Tiene lugar cuando se hace clic en el control.

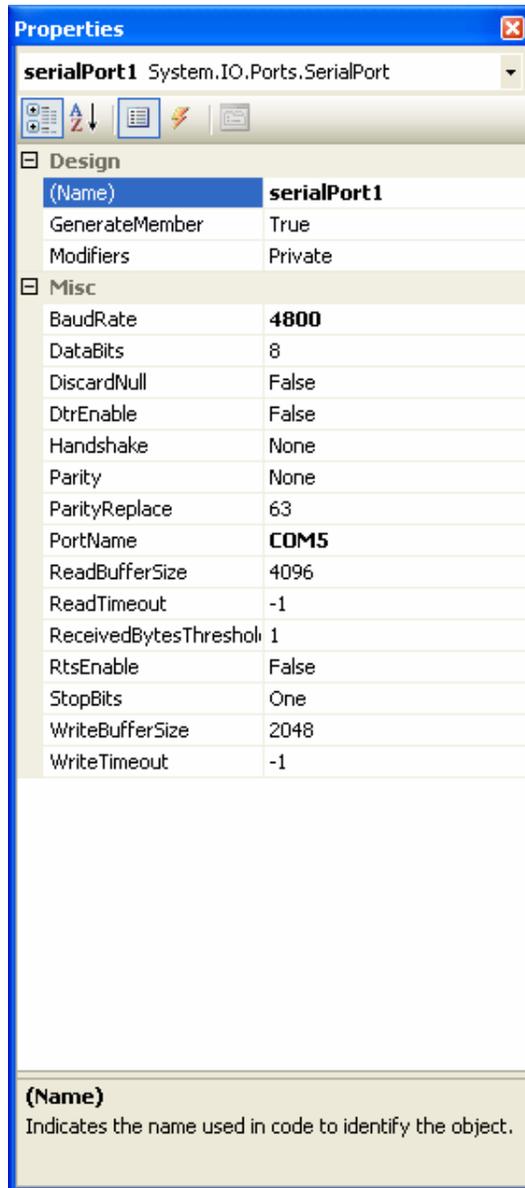
Propiedades Explorador de soluciones

Permite crear una etiqueta de texto navegable. Al puntear sobre la etiqueta se lanza el evento **Click** que nos permite cambiar de contexto, de formulario, activar una URL, etc.

Prácticamente es un botón de texto.



# Controles: **SerialPort**



**serialPort1** System.IO.Ports.SerialPort

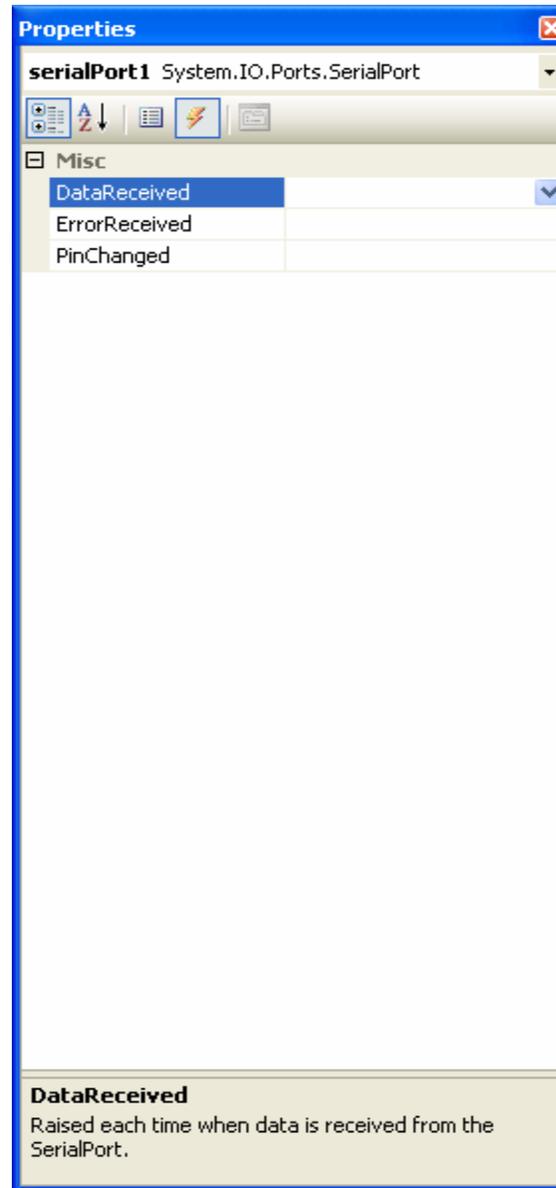
**Design**

(Name)	<b>serialPort1</b>
GenerateMember	True
Modifiers	Private

**Misc**

BaudRate	<b>4800</b>
DataBits	8
DiscardNull	False
DtrEnable	False
Handshake	None
Parity	None
ParityReplace	63
PortName	<b>COM5</b>
ReadBufferSize	4096
ReadTimeout	-1
ReceivedBytesThreshold	1
RtsEnable	False
StopBits	One
WriteBufferSize	2048
WriteTimeout	-1

**(Name)**  
Indicates the name used in code to identify the object.



**serialPort1** System.IO.Ports.SerialPort

**Misc**

DataReceived	
ErrorReceived	
PinChanged	

**DataReceived**  
Raised each time when data is received from the SerialPort.

Para utilizar este control hay que utilizar los métodos:

- Open
- Read
- ReadByte, ReadChar
- ReadLine
- Write
- WriteLine
- Close

La configuración del puerto se realiza a través de las propiedades. Otras propiedades:

- BytesToRead
- IsOpen
- .....



# Controles: **MonthCalendar**

The image shows two side-by-side screenshots of the Visual Studio Properties window for a `MonthCalendar` control. The left window shows the 'Appearance' and 'Behavior' sections, with 'Font' set to 'Tahoma; 9pt' and 'Enabled' set to 'False'. The right window shows the 'Action' section with 'DateChanged' selected, and the 'Property Changed' section with 'EnabledChanged' and 'ParentChanged' listed.

Permite configurar el aspecto:

`ShowWeekNumbers`,  
`CalendarDimensions`,  
`BoldedDates`,  
`AnnuallyBoldedDate`,  
`MonthlyBoldedDates`,  
`SelectionRange`,  
`BackColor`, `ForeColor`,  
`TitleBackColor`,  
`TitleForeColor`,...

Métodos:

`DateSelect`

`DateChanged`

Reciben como parámetro

`DateRangeEventArgs` que  
tiene los miembros

`End` y `Start`.



# Controles: DateTimePicker

**Formato de la fecha: Long, Short, Time y Custom**

**Máxima y mínima fecha elegible en el control**

Property	Value
Appearance	
CalendarFont	Tahoma; 9pt
Font	Tahoma; 9pt
<b>Format</b>	Long
ShowUpDown	False
Behavior	
ContextMenu	(none)
CustomFormat	
Enabled	True
MaxDate	31/12/9998
<b>MinDate</b>	01/01/1753
TabIndex	0
TabStop	True
<b>Value</b>	23/03/2007 12:29
Visible	True
Data	
(DataBindings)	
Tag	
Design	
Layout	
Anchor	Top, Left
Dock	None
Location	3; 35
Size	200; 22
Action	
ValueChanged	
Data	
(DataBindings)	
Focus	
GotFocus	
LostFocus	
Validated	
Validating	
Key	
KeyDown	
KeyPress	
KeyUp	
Property Changed	
EnabledChanged	
ParentChanged	

**Value**  
The current date/time value for this control.

Además de seleccionar una fecha, permite formatearla. Al seleccionar custom se pueden usar cadenas de formato:

d -> Día del mes: 9

dd -> día del mes: 09

ddd -> Día de la semana abreviado

dddd -> Día de la semana

M -> Mes: 9

MM -> Mes: 09

MMM -> Nombre abreviado del mes

MMMM -> Nombre del mes.

.....

El campo **Value** es de tipo **DateTime**.



---

# Desarrollo con .NET Compact Framework para SmartPhone



# Características Smartphone

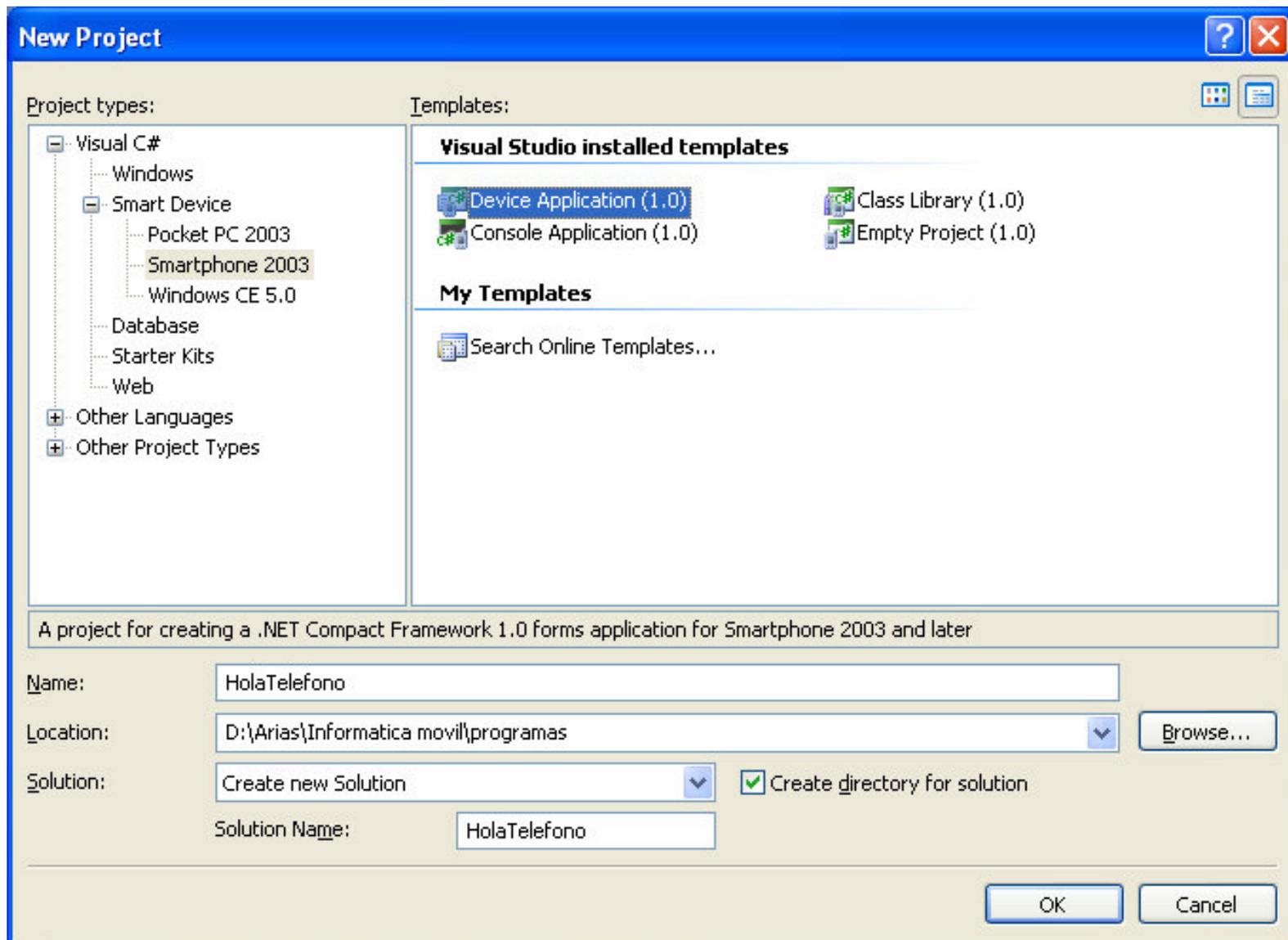
---

- Un Smartphone es un teléfono que ejecuta el sistema operativo Pocket PC.
- Se diferencia de un Pocket Pc normal en:
  - El tamaño de pantalla es de 176 x 220 píxeles en vez de 240 x 320.
  - La pantalla del Smartphone no es táctil. Todas las entradas del usuario se hacen pulsando botones físicos.
- Estas diferencias hacen que no se disponga de los siguientes controles:

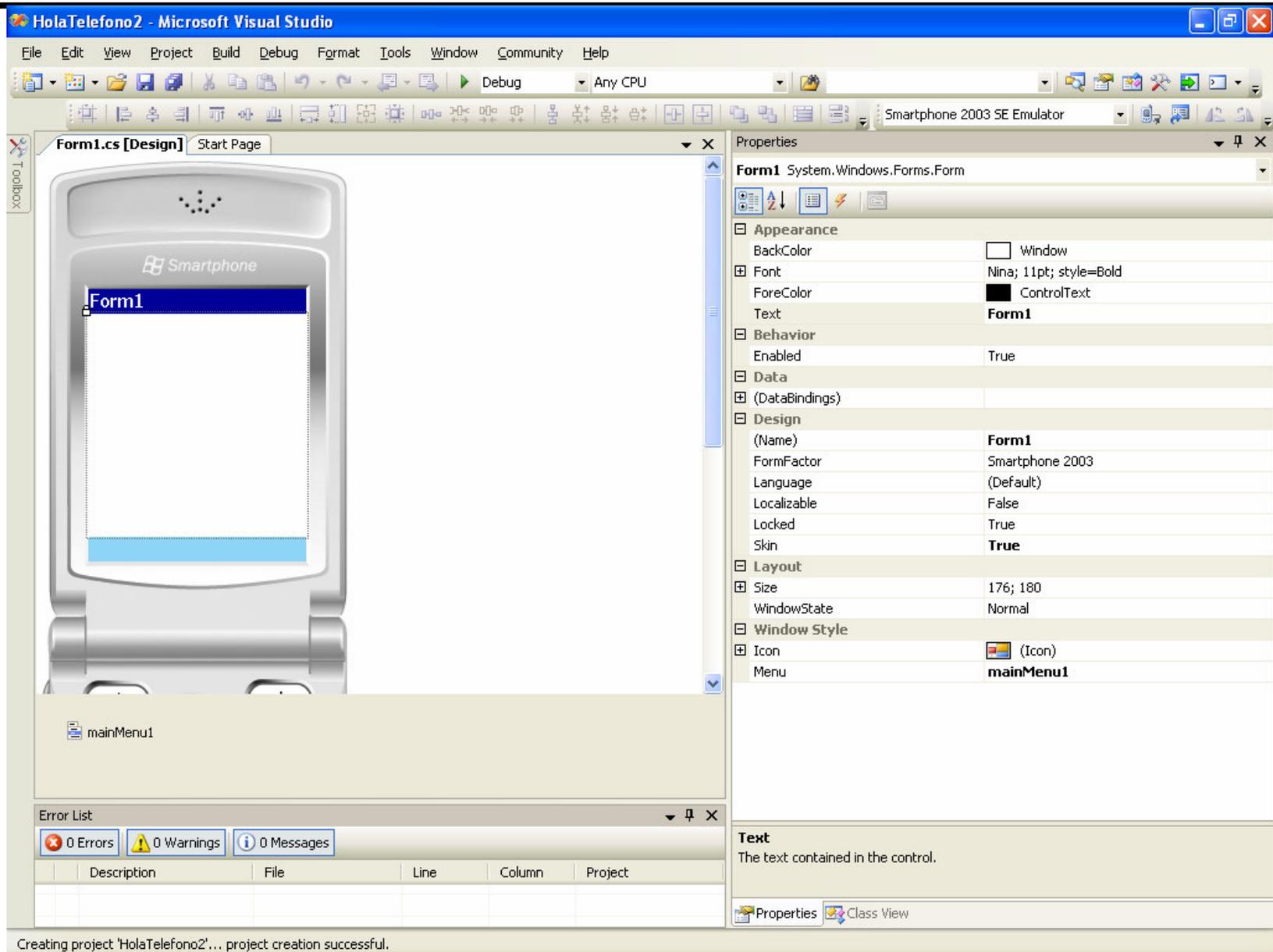
<code>Button</code>	<code>RadioButton</code>	<code>ListBox</code>
<code>TabControl</code>	<code>DomainUpDown</code>	<code>NumericUpDown</code>
<code>TrackBar</code>	<code>ContextMenu</code>	<code>ToolBar</code>
<code>StatusBar</code>	<code>OpenFileDialog</code>	<code>SaveFileDialog</code>
<code>InputPanel</code>		



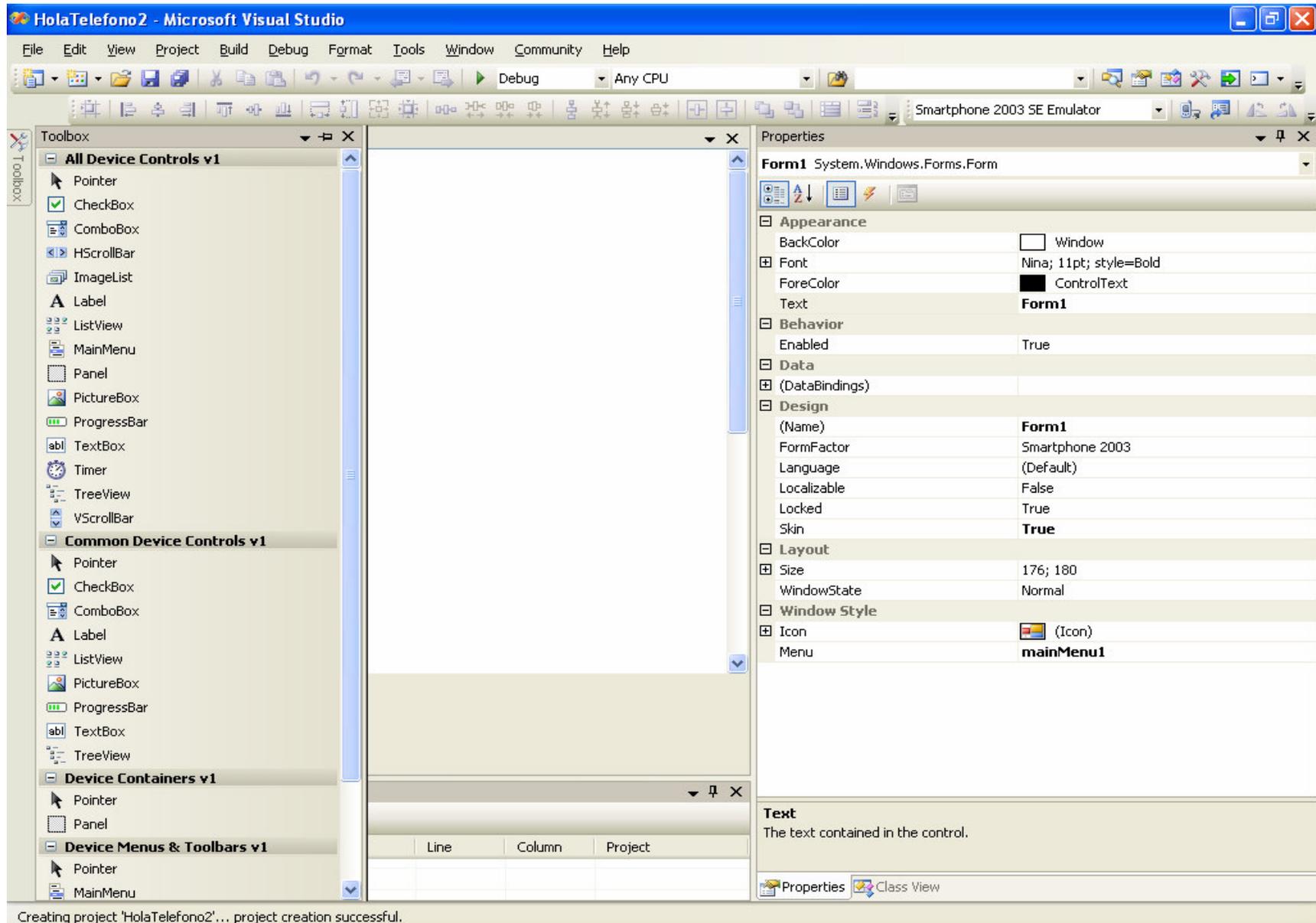
# Selección de la plataforma



# Entorno de trabajo



# Herramientas



# Simulador Smartphone

