Apéndice 5. Contacto con los entornos gráficos

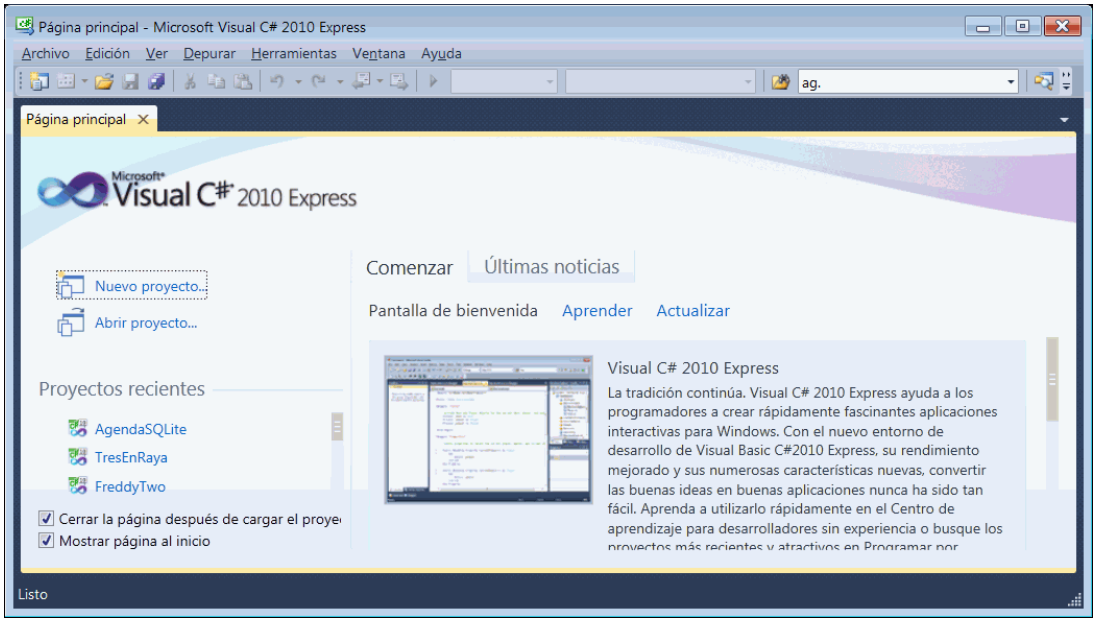
Ap5.1. Creación de formularios, botones y etiquetas

En C# podemos crear con una cierta facilidad programas en entornos gráficos, con menús botones, listas desplegables, etc.

La forma más cómoda de conseguirlo es usando herramientas que incluyan un editor visual, como Visual Studio o SharpDevelop.

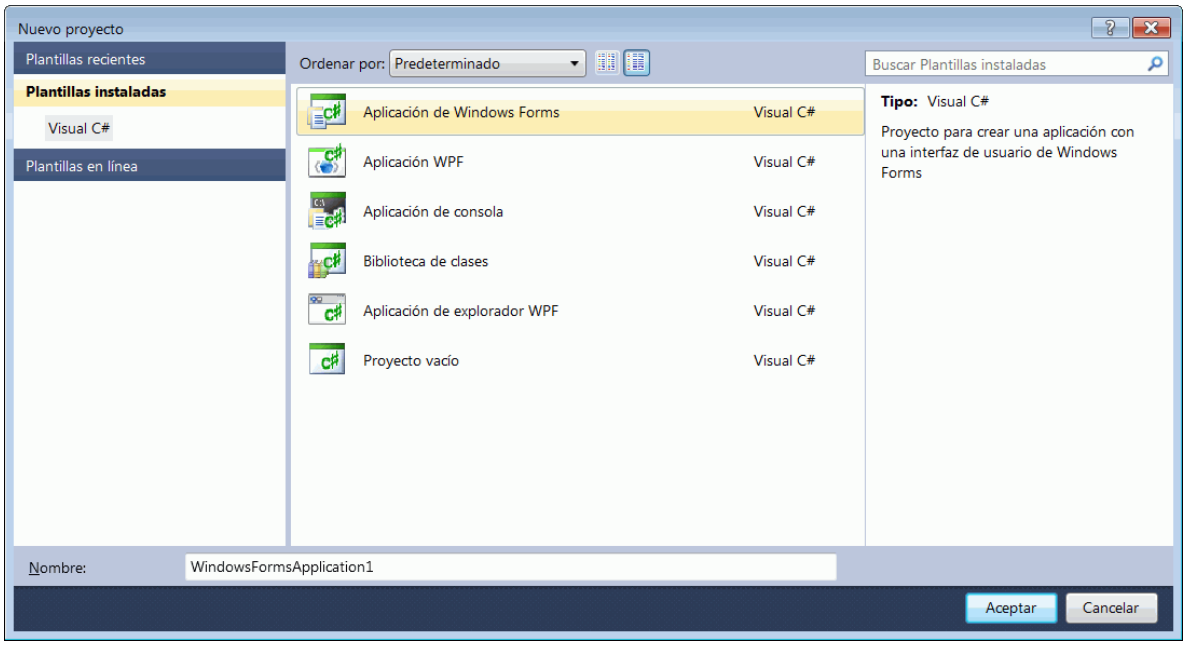
SharpDevelop necesita un ordenador menos potente que Visual Studio y tiene un manejo muy similar a éste. Aun así, dado que la versión Express de Visual Studio es gratis para uso personal y se mueve con una soltura razonable en un ordenador "moderno" (más de 1 Gb de memoria RAM, procesador de un núcleo de 2 Ghz o más, o bien procesador de doble núcleo), vamos a centrarnos en su manejo, en concreto para la versión Visual Studio 2010, dado que los cambios en versiones posteriores son mínimos.

Cuando lanzamos Visual Studio 2010, aparece una pantalla que nos muestra los últimos proyectos que hemos realizado y nos da la posibilidad de crear un nuevo proyecto:

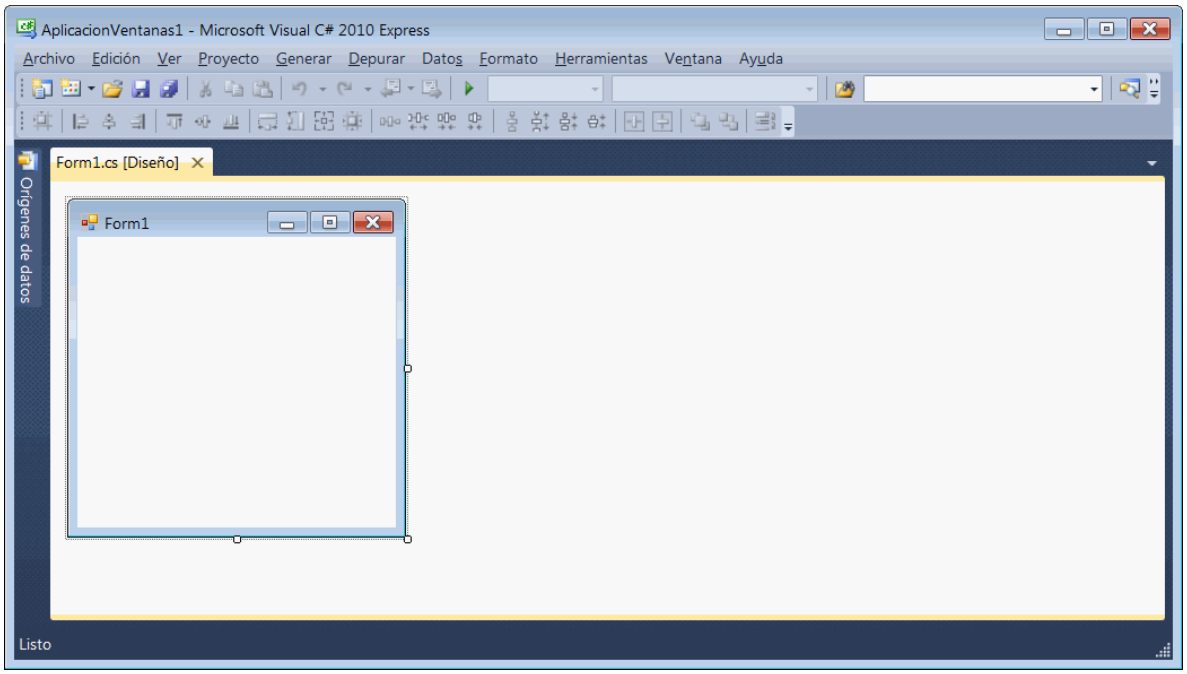


Al elegir un "Nuevo proyecto", se nos preguntará de qué tipo queremos que sea, así como su nombre. En nuestro caso, será una "**Aplicación de Windows Forms**", y como nombre se nos propondrá algo como "WindowsFormsApplication1", que nosotros podríamos cambiar por "AplicacionVentanas1" (como siempre, sin espacios ni acentos,

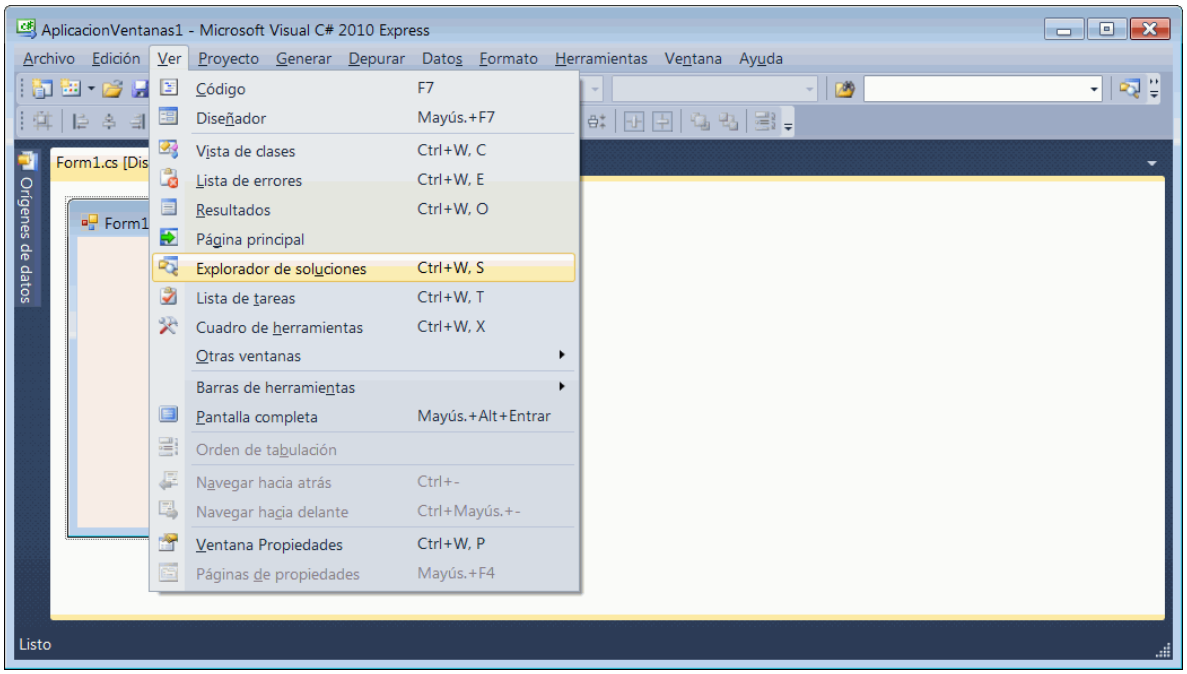
que suelen ser caracteres no válidos en un identificador -aunque Visual Studio sí permitiría los acentos-):



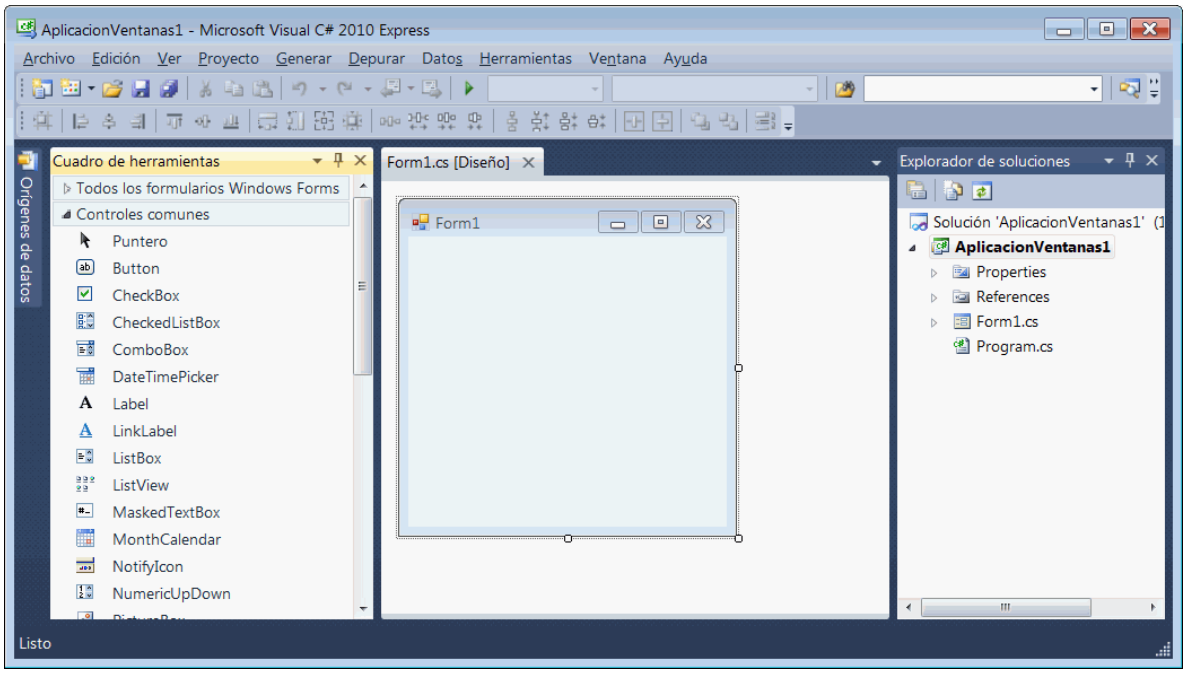
Nos aparecerá un formulario ("una ventana") vacío:



Si nuestra pantalla está así de vacía, nos interesará tener un par de herramientas auxiliares. La primera es el "Explorador de soluciones", que nos permitirá pasear por las distintas clases que forman nuestro proyecto. La segunda es el "Cuadro de herramientas", que nos permitirá añadir componentes visuales, como botones, menús y listas desplegables. Ambos los podemos añadir desde el menú "Ver":

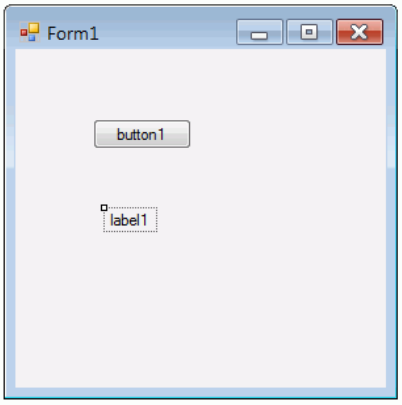


De modo que ahora nuestra pantalla debería ser algo así:

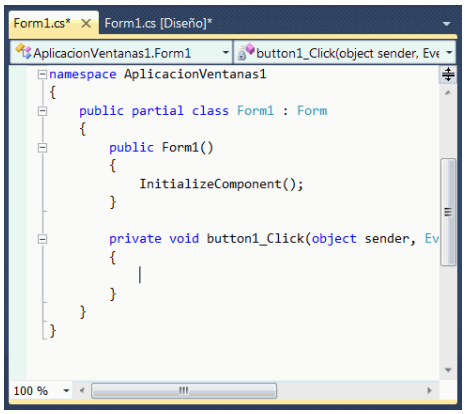


Como primer ejemplo, vamos a añadir a nuestro formulario un botón (Button) y una etiqueta de texto (Label), para hacer un primer programa que cambie el texto de la etiqueta cuando pulsemos el botón.

Hacemos clic en el componente Button del Cuadro de Herramientas (panel izquierdo) y luego clic dentro del formulario, para que aparezca un botón en esa posición. De igual modo, hacemos clic en Label para indicar que queremos una etiqueta de teto, y luego clic en cualquier punto de nuestro formulario. Nuestra ventana debería estar quedando así:



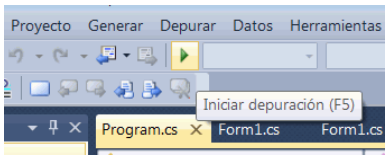
Ahora vamos a hacer que nuestro programa responda a un clic del ratón sobre el botón. Para eso, hacemos doble clic sobre dicho botón, y aparecerá un esqueleto de programa que podemos rellenar:



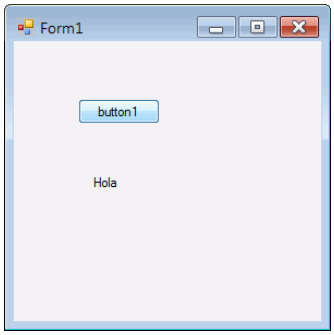
En ese hueco indicaremos lo que debe ocurrir cuando se pulse el botón ("button1_Click"). En nuestro caso, simplemente será cambiar el texto del "label", así:

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

Antes de ver con más detalle por qué ocurre todo esto y mejorarlo, vamos a comprobar que funciona. Pulsamos el botón de "Iniciar depuración" en la barra de herramientas:



Y nos aparecerá nuestra ventana, lista para probar. Si pulsamos el botón, cambiará el texto de la etiqueta:



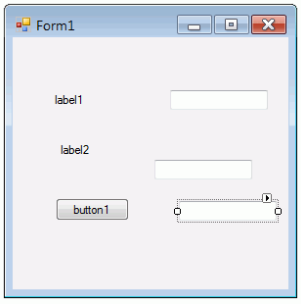
Antes de seguir, deberíamos grabar los cambios, porque algunas versiones de Visual Studio permiten "compilar a memoria", de modo que nuestro programa quizá todavía no está guardado.

Ahora vamos a crear un segundo proyecto, un poco más elaborado y que nos ayude a entender mejor el funcionamiento del entorno.

Ap5.2. Cambios de apariencia. Casillas de texto para sumar dos números

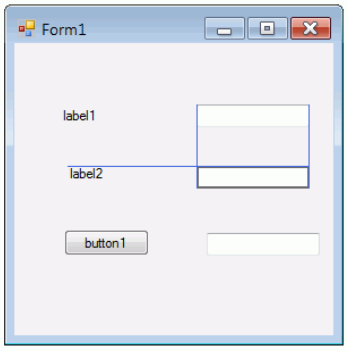
Hemos visto cómo colocar un botón y una casilla de texto, pero no hemos cambiado el texto del botón, ni su tamaño, ni el tipo de letra. Ahora vamos a hacer un programa que calcule y muestre la suma de dos números, y cuidaremos un poco más la apariencia.

Comenzamos por crear un nuevo proyecto llamado (por ejemplo) "AplicacionVentanas2", que también será una "Aplicación de Windows Forms".

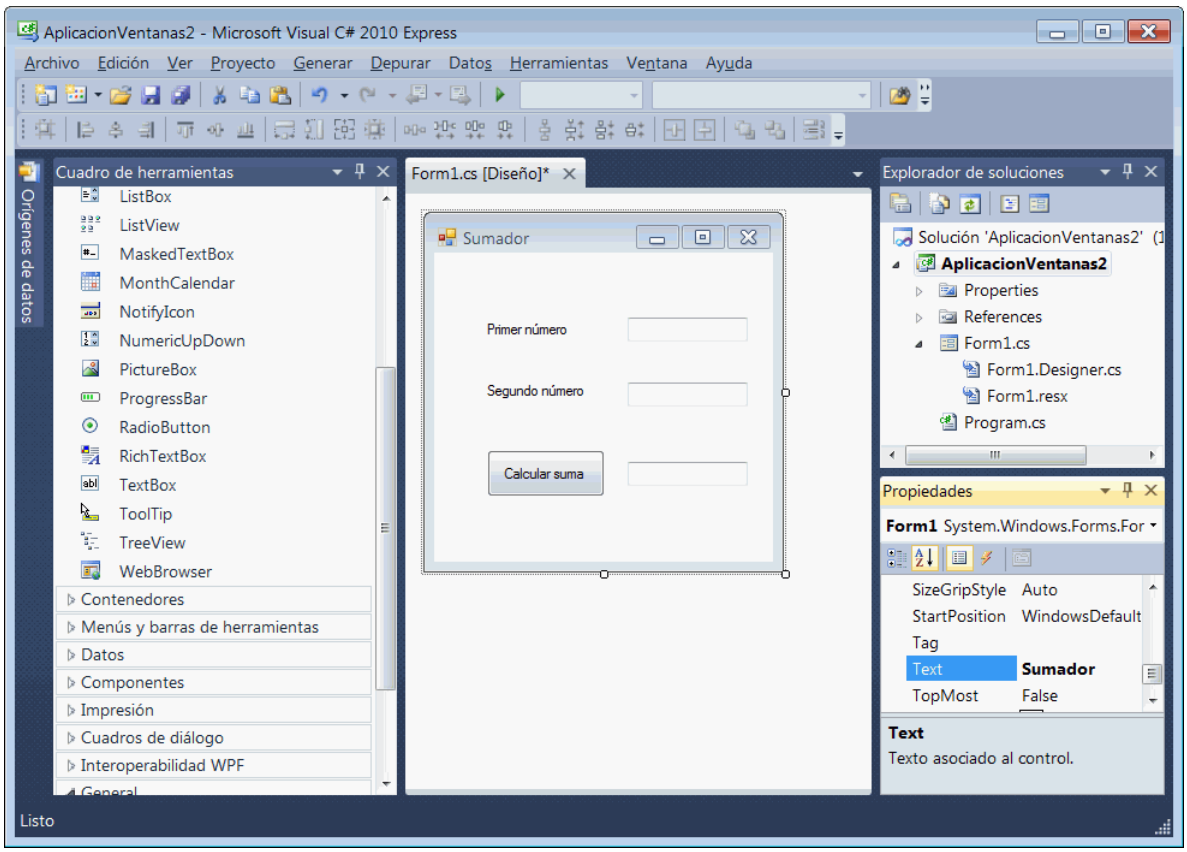


En el formulario, añadiremos dos Label para incluir textos explicativos, tres TextBox (casillas de introducción de texto) para los datos y el resultado, y también un botón para que se calcule dicho resultado. El resultado todavía será feo y, posiblemente, con los componentes desalineados:

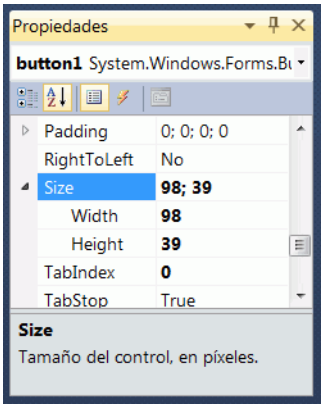
Vamos a comenzar por alinear los componentes. Si pinchamos y arrastramos uno de ellos con el ratón, cuando lo movamos es fácil que aparezcan unas rayas para avisarnos en el momento en que queden alineados:



Una vez que estén colocados, vamos a cambiar sus propiedades. Sabemos cambiar el texto de un Label desde código, pero también podemos hacerlo desde el diseñador visual. Basta con tener la ventana de "Propiedades", que podemos obtener al pulsar el botón derecho sobre uno de los componentes o desde el menú "Ver". Ahí tenemos la propiedad "Text", que es el texto que cada componente muestra en pantalla. Haciendo clic en cada uno de ellos, podemos cambiar el texto de todos (incluyendo el "nombre de la ventana"):



La anchura de las casillas de texto y el tamaño del botón los podemos cambiar simplemente pinchando y arrastrando con el ratón, pero también tenemos una propiedad "Size", con componentes X e Y:



En general, será interesante cambiar también el nombre (Name) de cada elemento visual: nuestro código resultará mucho más legible si un botón se llama (por ejemplo) "btSumar" que si conservamos el nombre original "button1".

Algunas de las propiedades son:

- Name, el nombre con el que se accederá desde el código.
- Text, el texto que muestra un elemento.
- ForeColor, el color con el que se muestra el componente.
- BackColor, el color de fondo.
- TextAlign, para indicar la alineación del texto (y poder centrarlo, por ejemplo).
- Enabled, para poder activar o desactivar un elemento.
- Location, la posición en que se encuentra (que podemos ajustar inicialmente con el ratón).
- Size, el tamaño (ancho y alto, que también se puede ajustar inicialmente con el ratón).
- Font, el tipo de letra.
- ReadOnly, para poder hacer que un TextBox sea sólo de lectura y no se pueda modificar su contenido.

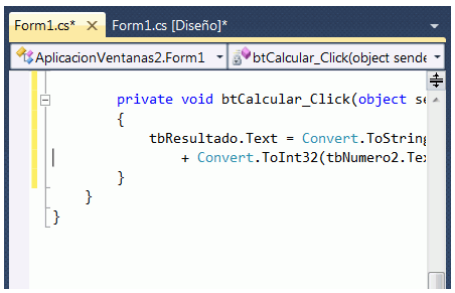
En nuestro caso, vamos a hacer los siguientes cambios:

- Las etiquetas de texto, en vez de label1 y label2, se llamarán lbNumero1 y lbNumero2.
- De igual modo, las casilla de texto, en vez de textBox1, textBox2, y textBox3 se llamarán tbNumero1, tbNumero2 y tbResultado.
- El botón, en vez de Button1, pasará a llamarse btCalcular.
- El formulario se llamará frmSumador.
- La casilla del resultado usará el mismo tipo de letra, pero en negrita (Bold). Además, será sólo de lectura.

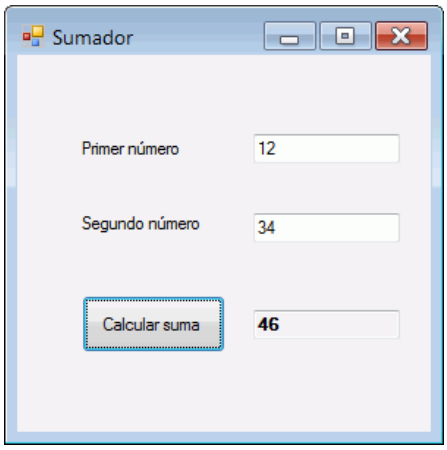
Nuevamente, para hacer que se calcule el resultado al pulsar el botón, hacemos doble clic sobre él. En este caso, el texto de tbResultado será la suma de los de los otros TextBox. Eso sí, como las casillas de texto contienen cadenas, habrá que convertir a int32 para poder sumar, y luego volver a convertir a string para almacenar el resultado:

```
private void btCalcular_Click(object sender, EventArgs e)
{
    tbResultado.Text = Convert.ToString( Convert.ToInt32(tbNumero1.Text)
        + Convert.ToInt32(tbNumero2.Text) );
}
```

Podemos volver a la vista de diseño en cualquier momento, haciendo clic en la pestaña que hay en la parte superior:



La apariencia del programa, ya en funcionamiento, sería algo como:



Es muy mejorable, pero algunas de las mejoras son parte de los ejercicios propuestos.

Ejercicios propuestos:

(Ap5.2.1) Mejora el ejemplo anterior para que las casillas tengan el texto alineado a la derecha y para que no falle si las casillas están vacías o contienen texto en lugar de números.

(Ap5.2.2) Crea una nueva versión del programa, que no solo muestre la suma, sino también la resta, la multiplicación, la división y el resto de la división.

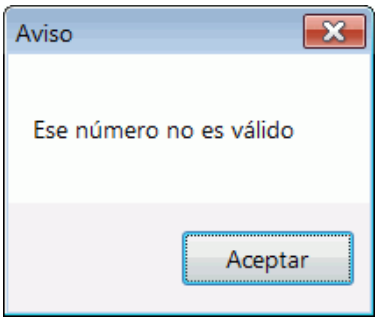
(Ap5.2.3) Un programa que muestre una ventana con un recuadro de texto, un botón y 3 etiquetas. En el recuadro de texto se escribirá un número (en sistema decimal). Cada vez que se pulse el botón, se mostrará en las 3 etiquetas de texto el equivalente de ese número en binario, octal y hexadecimal.
(Ap5.2.4) Haz una variante del programa anterior, que no use un botón para convertir el número a binario, octal y hexadecimal. En vez de eso, cada vez que en el recuadro de texto se pulse una tecla, se mostrará automáticamente en las 3 etiquetas de texto el equivalente del número actual. Pista: al hacer doble clic sobre una casilla de texto, aparecerá el evento "TextChanged", que es el que se lanza cuando se modifica el texto que contiene un TextBox.

Ap5.3. Usando ventanas predefinidas

En una aplicación basada en ventanas, típicamente tendremos que mostrar algún mensaje de aviso, o pedir una confirmación al usuario. Para ello podríamos crear un programa basado en múltiples ventanas, pero eso queda más allá de lo que pretende este texto.

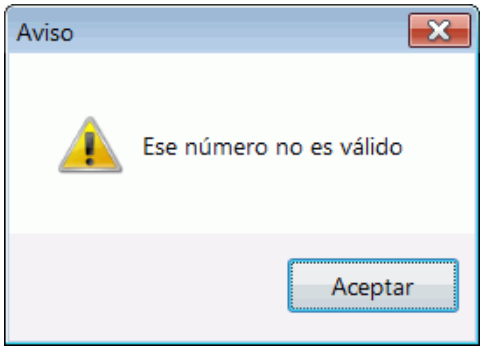
Una forma alternativa y sencilla de conseguirlo es usando "ventanas de mensaje". Éstas se pueden crear llamando a "MessageBox.Show", que tiene varias sintaxis posibles, según el número de parámetros que queramos utilizar. Por ejemplo, podemos mostrar un cierto texto de aviso en una ventana que tenga un título dado:

```
MessageBox.Show("Ese número no es válido", "Aviso");
```



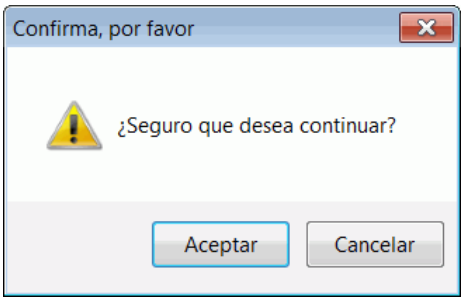
La segunda variante es indicar además qué botones queremos mostrar, y qué iconos de aviso:

```
MessageBox.Show("Ese número no es válido", "Aviso",
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
```



Y la tercera variante permite indicar además el que será el botón por defecto:

```
MessageBox.Show("¿Seguro que desea continuar?", "Confirma, por favor",
         MessageBoxButtons.OKCancel, MessageBoxIcon.Exclamation,
         MessageBoxDefaultButton.Button1);
```



En los casos en los que se muestran varios botones, nos interesará saber cuál de ellos se ha pulsado, y eso se puede conseguir mirando el valor devuelto por el MessageBox, que será de tipo DialogResult::

```
DialogResult respuesta = MessageBox.Show(
         "¿Seguro que desea continuar?",
         "Confirma, por favor",
         MessageBoxButtons.OKCancel,
         MessageBoxIcon.Exclamation,
         MessageBoxDefaultButton.Button1);
if (respuesta == DialogResult.OK) ...
```

Como se ve en estos ejemplos, tenemos algunos valores predefinidos para indicar qué botones o iconos queremos mostrar:

- Los botones (MessageBoxButtons) pueden ser: OK (Aceptar), OKCancel (Aceptar y Cancelar), AbortRetryIgnore (Anular, Reintentar y Omitir), YesNoCancel (Sí, No y Cancelar), YesNo (Sí y No), RetryCancel (Reintentar y Cancelar).
- Los iconos (MessageBoxIcon) pueden ser: None (ninguno), Hand (X blanca en un círculo con fondo rojo), Question (signo de interrogación en un círculo, no recomendado actualmente), Exclamation (signo de exclamación en un triángulo

con fondo amarillo), Asterisk (letra 'i' minúscula en un círculo), Stop (X blanca en un círculo con fondo rojo), Error (X blanca en un círculo con fondo rojo), Warning (signo de exclamación en un triángulo con fondo amarillo), Information (letra 'i' minúscula en un círculo).

- Los botones por defecto (MessageBoxDefaultButton) pueden ser: Button1 (el primero), Button2 (el segundo), Button3 (el tercero).

Si queremos que el usuario responda tecleando, no tenemos ninguna ventana predefinida que nos lo permita (sí existe un "InputBox" en otros entornos de programación, como Visual Basic), así que deberíamos crear nosotros esa ventana de introducción de datos desde el editor visual o mediante código, elemento por elemento.

Ejercicios propuestos:

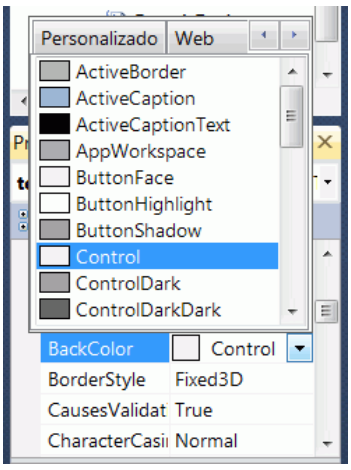
(Ap5.3.1) Mejora el ejercicio Ap5.2.1, para que muestre un mensaje de aviso si las casillas están vacías o contienen texto en lugar de números.

Ap5.4. Una aplicación con dos ventanas

Si queremos una ventana auxiliar, que permita al usuario introducir varios datos, deberemos crear un segundo formulario, y llamarlo desde la ventana principal. Vamos a ver los pasos necesarios.

En primer lugar, crearemos un nuevo proyecto, de tipo "Aplicación de Windows Forms", que llamaremos (por ejemplo) "DosVentanas" y entraremos a la vista de diseño para crear la que será la ventana principal de nuestra aplicación.

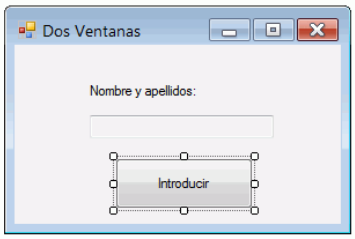
Vamos a crear en ella una casilla de texto (TextBox) en la que no se podrá escribir, sino que sólo se mostrarán resultados, y un botón que hará que aparezca la ventana secundaria, en la que sí podremos introducir datos.



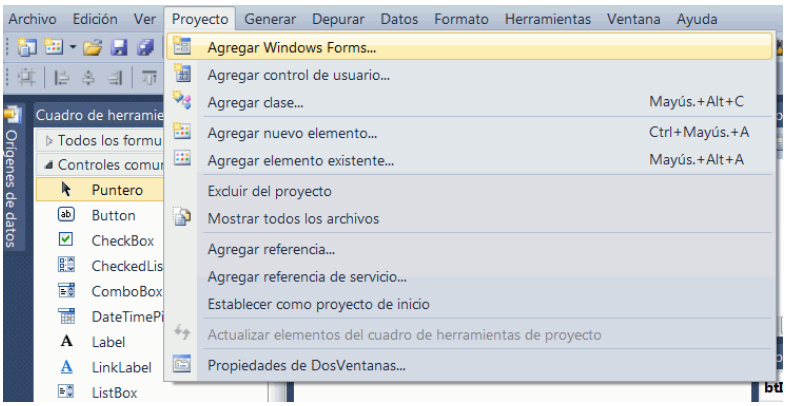
Comenzamos por crear el "TextBox" que mostrará el texto, y el "Label" que aclarará qué es ese texto. Cambiamos el "Text" de la etiqueta para que muestre "Nombre y apellidos", y cambiamos la propiedad "ReadOnly" de la casilla de texto para que sea "true" (verdadero), de modo que no se pueda escribir en esa casilla. También podemos cambiar el color de la casilla, para que sea más evidente que "no es una casilla normal". Visual Studio 2010 lo hace automáticamente, pero si un entorno más antiguo no lo hace por nosotros, podríamos pedir que fuera gris, como el resto de la ventana. Para eso, cambiamos su

propiedad BackColor (color de fondo). Es recomendable no usar colores prefijados, como el "gris", sino colores de la paleta de Windows, de modo que los elementos cambien correctamente si el usuario elige otra combinación de colores para el sistema operativo. Como el color de fondo de la ventana es "Control" (el color que tengan los controles de Windows), para la casilla de texto, escogeríamos también el color "Control".

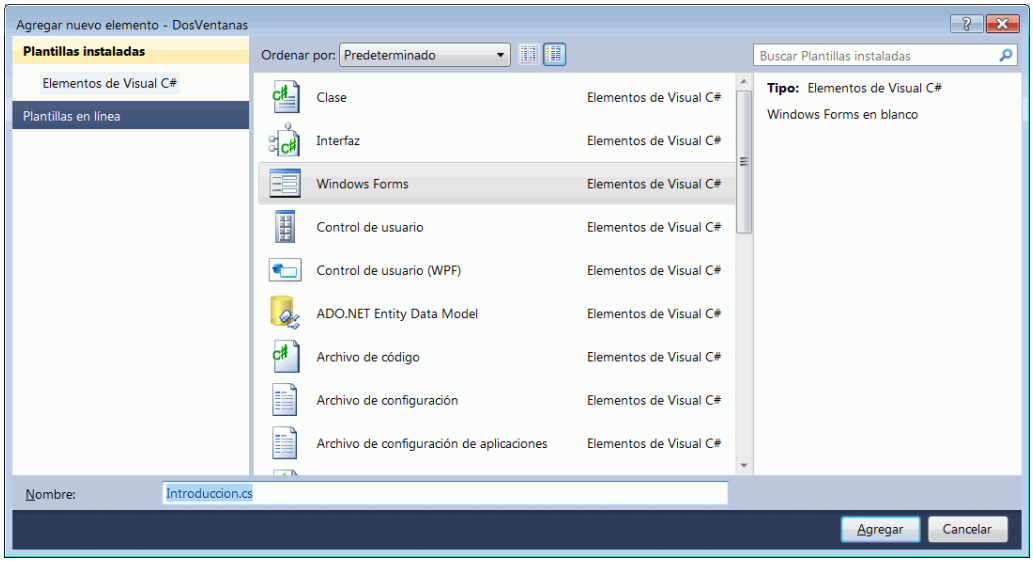
La apariencia de nuestra ventana debería ser parecida a ésta:



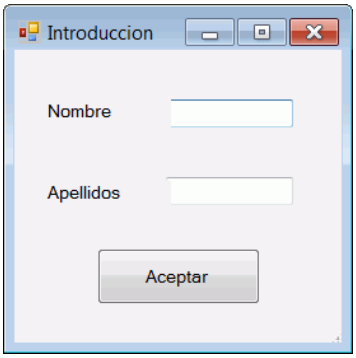
Para crear el segundo formulario (la ventana auxiliar), usamos la opción "Agregar Windows Forms ", del menú "Proyecto":



y le damos un nombre, por ejemplo "Introduccion.cs":



Nos aparecerá una nueva ventana, en su vista de diseño. Añadimos dos casillas de texto (TextBox), con sus etiquetas aclaratorias (Label), y el botón de Aceptar:



Llega el momento de añadir el código a nuestro programa. Por una parte, haremos que el botón "Aceptar" cierre la ventana. Lo podemos conseguir con doble clic, para que nos aparezca la función que se lanzará con el suceso Click del botón (cuando se pulse el ratón sobre él), y añadimos la orden "Close()" en el cuerpo de esa función, así:

```
void Button1Click(object sender, EventArgs e)
{
    Close();
}
```

Además, para que desde la ventana principal se puedan leer los datos de ésta, podemos hacer que sus componentes sean públicos, o, mejor, crear un método "Get" que devuelva el contenido de estos componentes. Por ejemplo, podemos devolver el nombre y el apellido como parte de una única cadena de texto, así:

```
public string GetNombreApellido()
{
```

```
    return tbNombre.Text + " " + tbApellidos.Text;
}
```

Ya sólo falta que desde la ventana principal se muestre la ventana secundaria y se lean los valores al terminar. Para eso, añadimos un atributo en la ventana principal, que represente la ventana auxiliar:

```
public partial class MainForm : Form
{
    Introduccion ventanaIntro;
        ...
```

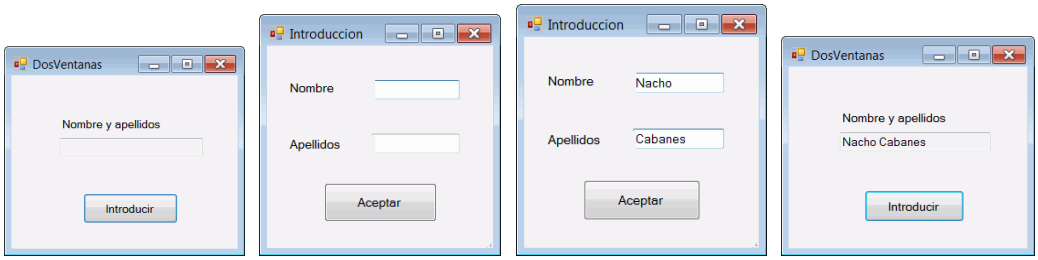
Y la inicializamos al final del constructor:

```
public MainForm()
{
    InitializeComponent();
    ventanaIntro = new Introduccion();
}
```

Finalmente, en el suceso Click del botón hacemos que se muestre la ventana secundaria usando ShowDialog, que espere a que se cierre ésta antes de permitirnos seguir trabajando en la ventana principal, y después leemos el valor que se había tecleado en dicha ventana:

```
void Button1Click(object sender, EventArgs e)
{
    ventanaIntro.ShowDialog();
    tbNombreApellido.Text = ventanaIntro.GetNombreApellido();
}
```

El resultado sería una secuencia como esta:

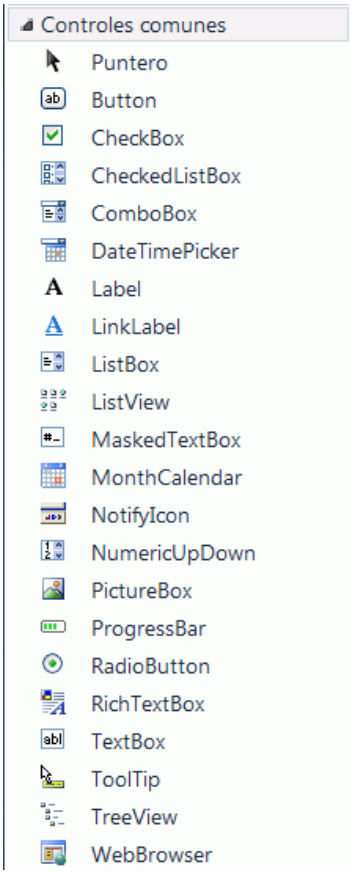


Ejercicios propuestos

(Ap5.4.1) Crea un programa que descomponga un número como producto de sus factores primos, usando ventanas.

(Ap5.4.2) Crea un programa que muestre una casilla de texto en la que el usuario puede introducir frases. Todas estas frases se irán guardando en un fichero de texto.

(Ap5.4.3) Crea una versión "con ventanas" de la "base de datos de ficheros" (ejemplo 04_06a).



Ap5.5. Otros componentes visuales

Por supuesto, hay muchos más controles visuales que los que hemos visto, pero en esto texto no se pretende dominar todos los componentes existentes, sino dar una alternativa a la entrada y salida convencional a través de consola.

Entre los otros muchos componentes existentes, y que quedan a la curiosidad del lector, cabe mencionar, por ser frecuentes y razonablemente sencillos de utilizar:

- CheckBox, para permitir al usuario escoger entre varias opciones, y RadioButton, cuando sólo se puede elegir una de dichas opciones.
- ListBox, para mostrar una lista de valores, y ComboBox para permitir introducir un valor o escoger de una lista desplegable.
- PictureBox, para mostrar una imagen.
- ProgressBar, para una barra de progreso.
- TextBox tiene una propiedad "MultiLine", para indicar si es de múltiples líneas. En ese caso, su contenido no se obtendría con "Text", sino recorriendo sus "Lines".
- También, dentro de otras categorías, podemos crear paneles para agrupar elementos, incluir menús y barras de herramientas, mostrar ventanas de diálogo estándar (como la de abrir un fichero, la de guardarlo, la de navegar por carpetas, la de elegir un color, la de elegir un tipo de letra), así como realizar otras muchas tareas de mayor complejidad, como crear componentes semiautomáticos para acceder a bases de datos, mostrar vistas previas de impresión, crear temporizadores, monitorizar cambios en el sistema de ficheros, etc.

Ap5.6. Dibujando con Windows Forms

Windows es un entorno gráfico, por lo que se podría suponer que deberíamos tener la posibilidad de trabajar en "modo gráfico" desde dentro de Windows, dibujando líneas, círculos y demás figuras básicas. En efecto, podemos usar las posibilidades de "System.Drawing" para crear una ventana gráfica dentro de nuestro formulario.

Deberemos preparar también las "plumas" ("Pen", para los contornos) y las "brochas" ("Brush", para los rellenos) que queramos usar. Un ejemplo que dibujara una línea roja y una elipse azul cuando pulsemos un botón del formulario podría ser así:

```
void Button1Click(object sender, EventArgs e)
{
  // Creamos la pluma, el relleno y la ventana gráfica
    System.Drawing.Pen contornoRojo = new System.Drawing.Pen(
      System.Drawing.Color.Red);
    System.Drawing.SolidBrush rellenoAzul = new System.Drawing.SolidBrush(
      System.Drawing.Color.Blue);
    System.Drawing.Graphics ventanaGrafica;
    ventanaGrafica = this.CreateGraphics();

    // Dibujamos
    ventanaGrafica.DrawLine(contornoRojo, 200, 100, 300, 400);
    ventanaGrafica.FillEllipse(rellenoAzul, new Rectangle(0, 0, 200, 300));

    // Liberamos la memoria que habíamos reservado
    contornoRojo.Dispose();
    rellenoAzul.Dispose();
    ventanaGrafica.Dispose();
}
```

Los métodos para dibujar líneas, rectángulos, elipses, curvas, etc. son parte de la clase Graphics. Algunos de los métodos que ésta contiene y que pueden ser útiles para realizar dibujos sencillos son:

- DrawArc, para dibujar un arco.
- DrawBezier, para una curva spline de Bézier definida por cuatro puntos (estructuras Point).
- DrawClosedCurve, para una curva spline cerrada, a partir de un array de puntos.
- DrawCurve, para una curva.
- DrawEllipse, para dibujar una elipse, a partir del rectángulo que la contiene.
- DrawIcon, para dibujar una imagen representada por un icono (Icon).
- DrawImage, para mostrar una imagen (Image).
- DrawLine, para una línea.
- DrawPolygon, para un polígono, a partir de un array de puntos.
- DrawRectangle, para un rectángulo.
- DrawString, para mostar una cadena de texto.
- FillEllipse, para rellenar el interior de una elipse.
- FillPolygon, para rellenar el interior de un polígono.
- FillRectangle, para rellenar el interior de un rectángulo.

Por otra parte, un ejemplo de cómo mostrar una imagen predefinida podría ser:

```
void Button2Click(object sender, EventArgs e)
```

```
{
  Graphics ventanaGrafica = this.CreateGraphics();
  Image imagen = new Bitmap("MiImagen.png");
  ventanaGrafica.DrawImage(imagen,20,20,100,90);
}
```

Esta imagen debería estar en la carpeta del programa ejecutable (que quizá no sea la misma que el fuente), y puede estar en formato BMP, GIF, PNG, JPG o TIFF.

Se puede encontrar más detalles sobre la clase Graphics en la referencia en línea (MSDN), por ejemplo en la página

http://msdn.microsoft.com/es-es/library/system.drawing.graphics_methods.aspx

Ejercicios propuestos

(Ap5.6.1) Crea una versión del "juego del ahorcado" (ejercicio 5.9.1.2) basada en Windows Forms, en la que, a medida que el usuario falle, se vaya "dibujando un patíbulo".