



# **El entorno de desarrollo Visual Studio 2005**

Manejo básico del entorno

El diseñador de formularios

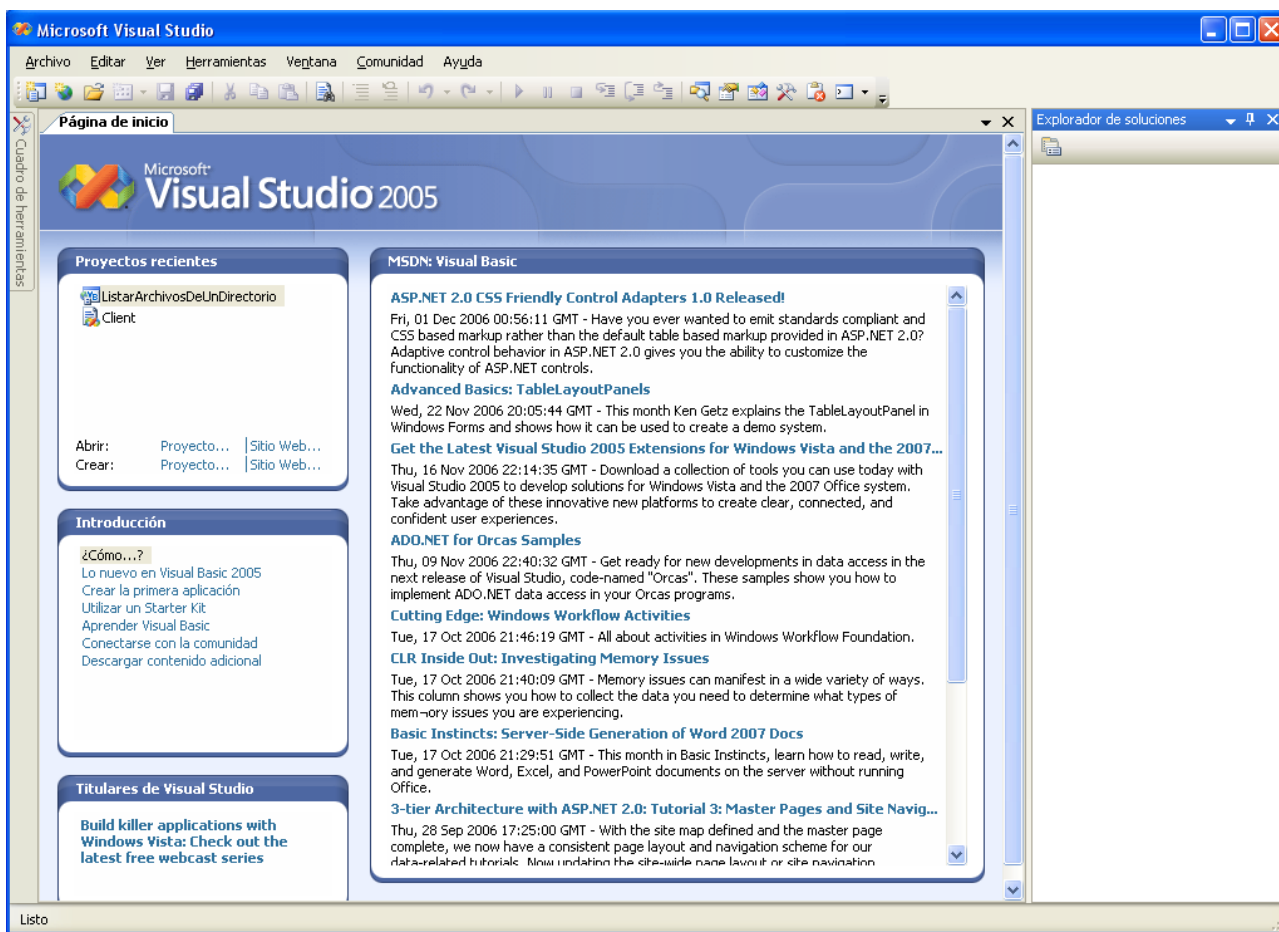
Visual Studio como herramienta de prototipado: simular la interacción

**Luís Rodríguez Baena**  
**Universidad Pontifica de Salamanca en Madrid**  
Facultad de Informática  
Escuela Universitaria de Informática  
2008

# El entorno de desarrollo Visual Studio 2005

## 1. La página de inicio

Al instalar el programa, y arrancarlo por primera vez, lo primero que aparece es la “Página de inicio”. Si la página se ha cerrado, es posible volver a visualizarla seleccionando “Página de inicio” de la opción “Otras ventanas” del menú “Ver”. También es posible restaurar el entorno de desarrollo a la situación inicial mediante la opción “Importar y exportar configuraciones...” del menú herramientas. En esta opción habría que elegir “Restablecer todas las configuraciones”, seleccionar si se quiere guardar la configuración actual y elegir algunas de las configuraciones predeterminadas (para este tutorial, se supone que se ha elegido la configuración predeterminada de “Configuración de desarrollo de Visual Basic”).



La Página de inicio está compuesta de cuatro secciones:

- **Proyectos recientes.** Desde aquí se podrá acceder a los últimos proyectos o sitios Web en los que se ha estado trabajando o crear un proyecto o sitio Web nuevo.
- **Introducción.** Con una lista de temas de ayuda, artículos técnicos, sitios Web, etc. La lista de temas, al igual que en las secciones siguientes, cambia según las características de los productos instalados o de la configuración que se haya aplicado.
- **Titulares de Visual Studio.** Con enlaces a información de productos y eventos de Microsoft.

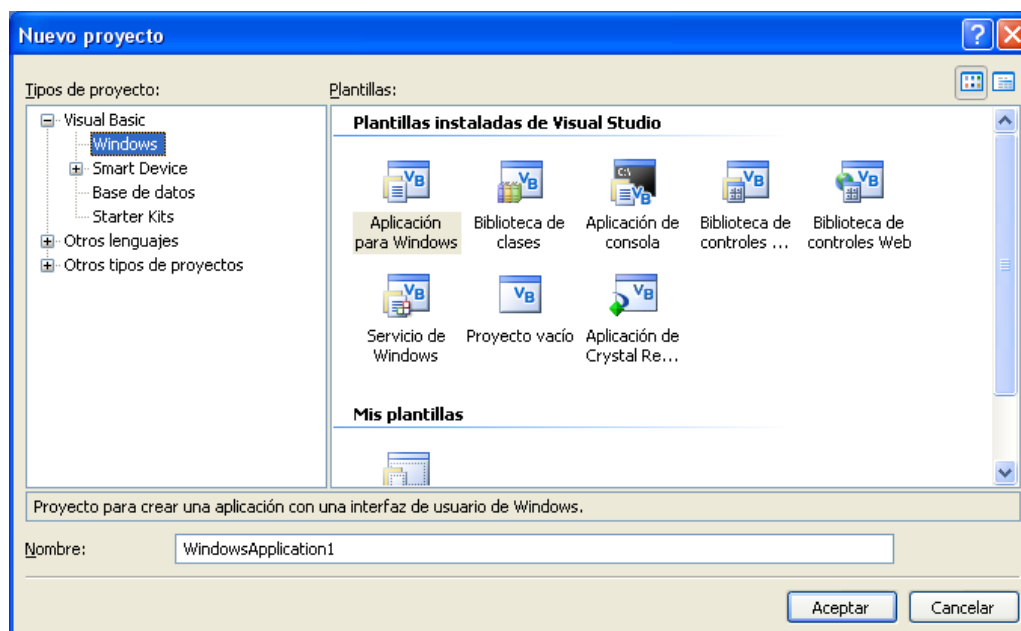
- **Noticias** (en nuestro caso “MSDN: Visual Basic”). Con una lista de artículos del canal RSS especificado en la configuración del entorno.

## 2. El proyecto

Una aplicación de Visual Studio se desarrolla alrededor de una *solución*. Una solución puede agrupar a varios proyectos, información acerca de los mismos, y una gran variedad de archivos (páginas HTML, documentos y esquemas XML, plantillas XSLT, mapas de bits, cursores, iconos, archivos de recursos y de texto, etc.).

Cada solución se corresponde con un archivo de definición de soluciones (.sln) almacenado en el disco que incluye información sobre los distintos elementos que lo componen, como proyectos asociados a la solución u otros elementos disponibles dentro de la misma y que no están asociados a ningún proyecto en particular. El archivo de definición de la solución puede compartirse entre los distintos programadores, pero cada uno de los usuarios tiene asociado un archivo con extensión .suo con que guarda información de la solución mientras que está siendo utilizada por el programador.

Para crear un nuevo proyecto hay que seleccionar el vínculo “Crear: Proyecto...” de la página de inicio o seleccionar la opción “Nuevo proyecto...” del menú archivo (CTRL+N).




En la columna de la izquierda aparecen los distintos tipos de proyectos y en la ventana de la derecha las plantillas asociadas a esos proyectos. En el caso de las plantillas asociadas a una aplicación que corra bajo Windows en Visual Basic, existen las siguientes:

Plantilla de proyecto	Se utiliza para crear
Plantilla de aplicaciones para Windows	Aplicaciones autónomas tradicionales para
Plantilla de biblioteca de clases	Clases o componentes reutilizables que puedan compartirse con otros proyectos. Este tipo de proyecto se considera que no tiene ventanas y no contendrá una clase de formulario Windows Forms.
Plantilla de aplicación de consola	Aplicaciones de línea de órdenes.
Plantilla de biblioteca de controles de Windows	Controles personalizados para utilizarlos en formularios Windows Forms.
Plantilla de biblioteca de controles Web	Controles personalizados para utilizarlos en páginas de formularios Web Forms.

Plantilla de proyecto	Se utiliza para crear
Plantilla de servicio de Windows	Aplicaciones de larga duración (se ejecutan durante toda la sesión de Windows) que no tienen una interfaz de usuario. Las aplicaciones de servicios de Windows (antes denominadas "servicios NT") pueden supervisar elementos tales como el rendimiento del sistema.
Plantilla de proyecto vacía	Proyecto vacío. La plantilla crea la estructura de archivos necesaria para almacenar la información de la aplicación; las referencias, archivos o componentes deben agregarse manualmente.

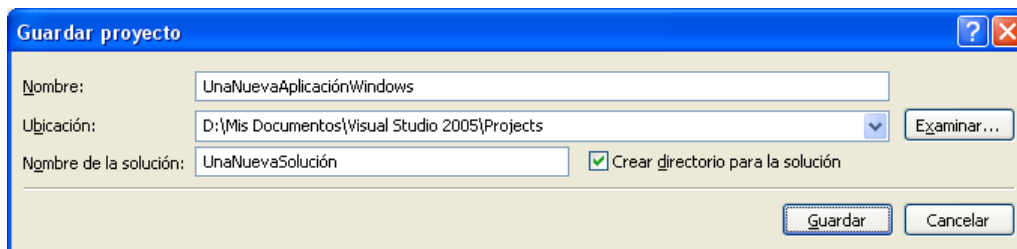
También se pueden hacer proyectos de Visual Basic para dispositivos inteligentes (Smart Devices) como Pocket PC 2005, SmartPhone o Windows CE 5.0), aplicaciones de bases de datos, etc. Desde esta ventana también se podrían seleccionar proyectos para otros lenguajes instalados con Visual Studio, proyectos para los programas de instalación, etc.

La parte inferior de la ventana se utiliza para asignar el nombre de proyecto. Por omisión, para una aplicación Windows, utiliza el nombre `WindowsApplicationN`, donde *N* es un número correlativo. Puesto que en un futuro podrá haber varias soluciones de este tipo, para poder identificarlas es conveniente dar al proyecto un nombre significativo.

El nombre de la solución y el directorio donde se almacenará podremos cambiarlos al guardar el proyecto. Por omisión todos los proyectos son temporales, es decir, no se guardan hasta que se da la orden con la opción "Guardar todo..." del menú archivo (CTRL+MAYÚS+S o el botón  de la barra de iconos). Estos archivos temporales se almacenan en la carpeta "Mis Documentos\Visual Studio 2005\Backup Files\" y desaparecen al cerrar el entorno o crear una nueva solución, por lo que es importante guardar el proyecto.

## 2.1. Archivos y directorios de la solución

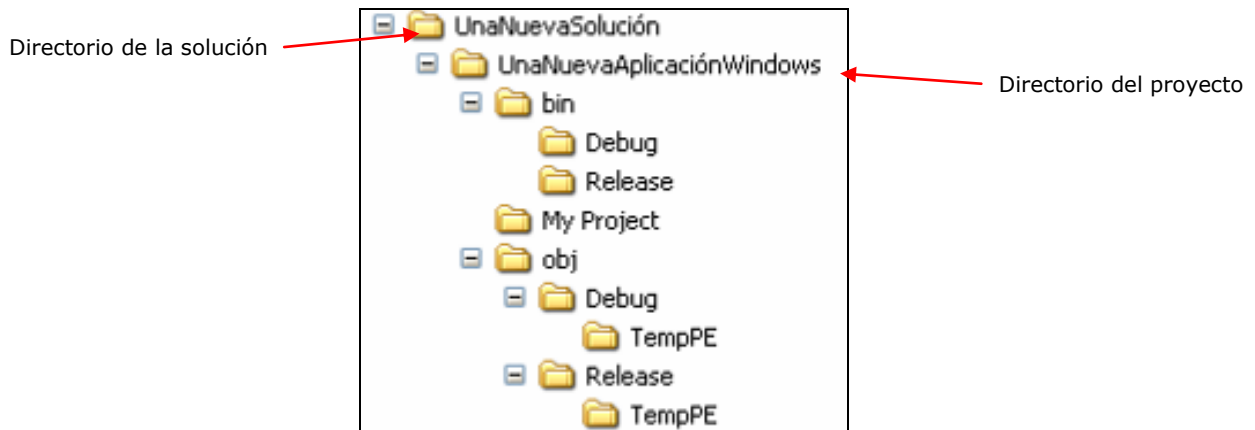
Al dar la opción de "Guardar todo..." aparece el siguiente cuadro de diálogo.



Aquí podremos cambiar el nombre del proyecto o el nombre de la solución. Además se podrá seleccionar donde se almacenará el mismo (por omisión creará un directorio con el nombre del proyecto a partir de la carpeta "Mis Documentos\Visual Studio 2005\Projects\"). Si se deja marcada la casilla de verificación creará una carpeta para la solución y dentro de ella una para el proyecto.


Es posible forzar al entorno a no trabajar con archivos temporales. En el cuadro de diálogo de la opción "Opciones..." del menú "Herramientas", se debería marcar la casilla de verificación "Guardar nuevos proyectos al crearlos" de la rama "General" de "Proyectos y soluciones".

Dentro de la carpeta `Projects` se creará un directorio con el nombre de la solución y dentro de ella, si se ha marcado la casilla de verificación, otro con el nombre del proyecto. Para trasladar el proyecto a otro ordenador para seguir con su desarrollo, lo más práctico será copiar **toda la carpeta de la solución**.



En el directorio de la solución se crea un archivo con extensión `.sln` y con el nombre de la solución (`UnaNuevaSolución.sln`). Se trata de un archivo de texto con los proyectos de los que consta la solución, con las propiedades de la solución y con el resto de elementos que contiene.

Además del archivo `.sln`, en el directorio de la solución se crea un archivo con extensión `.suo`. Se trata de un archivo binario que guarda información sobre las opciones asociadas a la solución como el tiempo que hace que no se ha abierto la solución o la personalización de sus propiedades. Este archivo permite que, cuando se abra el proyecto, se mantengan las características de personalización que el usuario ha realizado.

En el directorio del proyecto se crean además otras tres carpetas: `bin`, `My Project` y `obj`. El primero tiene a su vez dos carpetas, `Debug` y `Release`. La carpeta `Debug` es donde se guardan los resultados de la aplicación al ejecutarla (al pulsar la tecla F5, la opción “Iniciar depuración” del menú “Depurar” o el icono  de la barra de herramientas). La carpeta `Release` es donde se guardan los resultados de la aplicación al generarla (opción “Generar *NombreDelProyecto*” del menú “Generar”). La primera guarda la aplicación sin optimizar la compilación ya que almacena información de depuración. La segunda guarda el código optimizado y sería la versión definitiva de la aplicación.

En la carpeta `obj` se guardan las referencias a objetos COM (objetos que se utilizan en el proyecto y que no se han creado con el entorno de Visual Studio). La carpeta `My Project` guarda las propiedades del proyecto.

Como ya se dijo, el proyecto en sí mismo se guarda en un directorio que tiene, de forma predefinida, el mismo nombre que el proyecto. En él inicialmente se almacena el archivo de proyecto. En el caso de una aplicación de Visual Basic, se trata de un archivo con extensión `.vbproj`. En el caso de que se trate de proyectos en otros lenguajes podrá tener la extensión `.csproj` para C# (*C Sharp*) o `.vcproj` para Visual C++. El archivo `.vbproj` es un archivo con contenido XML que contiene información sobre el proyecto y los archivos que contiene.

Los archivos creados para el proyecto son los siguientes:

- Un archivo `Form1.vb` que contiene el código del formulario que crea la plantilla de una aplicación Windows. Se trata de un archivo de texto con el código fuente del formulario.
- Un archivo `Form1.resx`. Se trata de un archivo utilizado para editar y definir recursos de las aplicaciones. Los recursos son datos no ejecutables que se añaden a las aplicaciones. Puede tratarse de iconos, imágenes o cursores, textos con mensajes de error en varios idiomas para facilitar la internacionalización de la aplicación, etc.
- Un archivo `Form1.Designer.vb`. Se trata de un archivo que define los distintos controles que se incluyen en el formulario.

Inicialmente el proyecto se crea con un formulario llamado `Form1`. Por cada formulario que se añada al proyecto habrá uno de estos archivos.

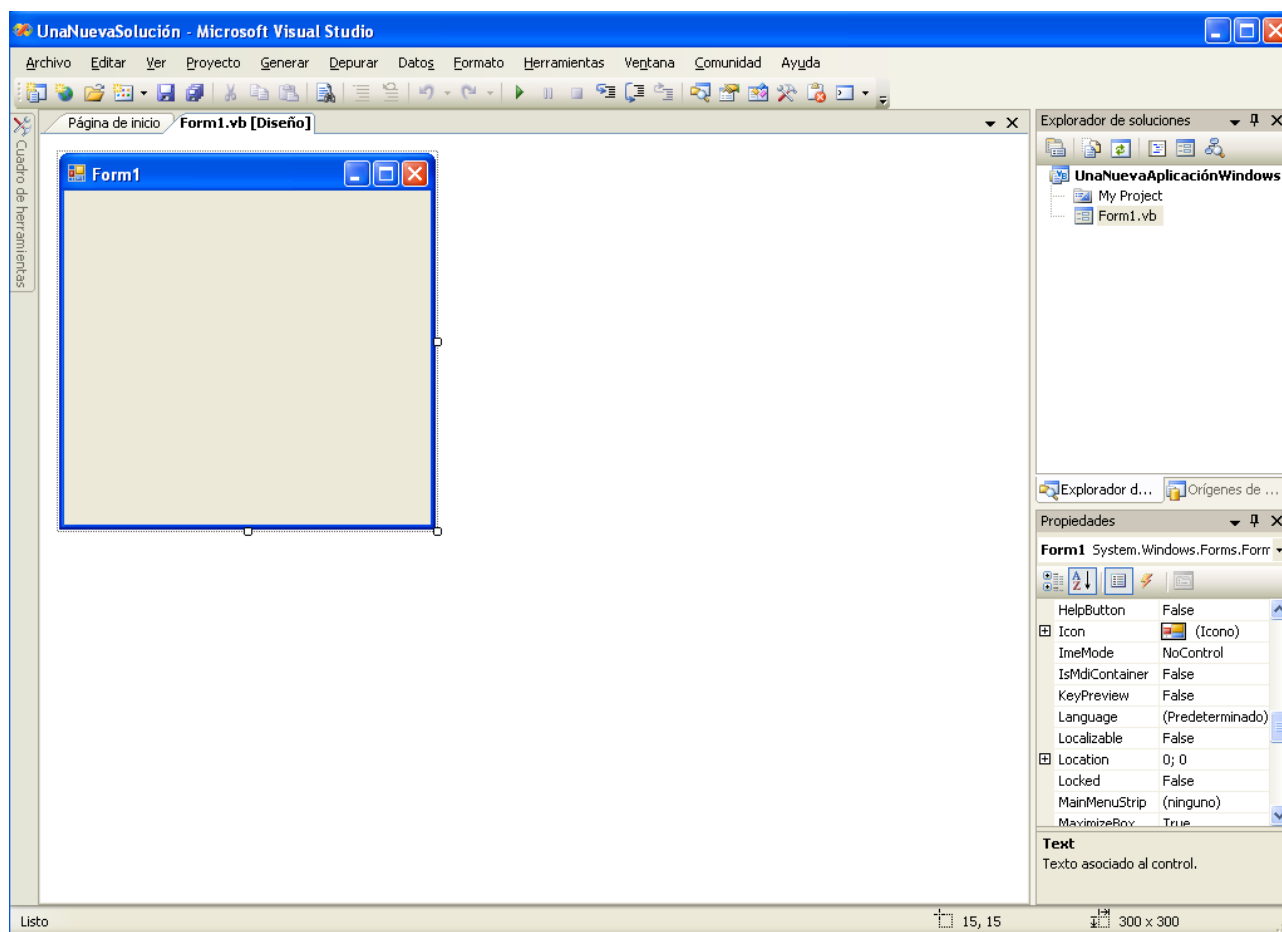
De cualquier forma, un proyecto podrá tener otros archivos. La lista de archivos de un proyecto local puede incluir los siguientes tipos de archivo:

Elemento de proyecto	Extensión	Propósito del elemento de proyecto
Formulario Windows Forms	.vb	Un formulario utilizado para crear aplicaciones para Windows.
Clase	.vb	Archivo de código que contiene una declaración de clase.
Interfaz	.vb	Una nueva interfaz.
Módulo (sólo Visual Basic)	.vb	Archivo para el almacenamiento de grupos de funciones.
Clase de componentes	.vb	Clase para crear componentes utilizando el diseñador visual.
Control de usuario	.vb	Una clase para crear un control de formularios Windows Forms utilizando el diseñador visual.
Servicio de Windows	.vb	Una clase para crear servicios de Windows.
DataSet	.xsd	Un archivo para crear un esquema XML con clases DataSet
Base de datos SQL	.mdf	Una base de datos SQL vacía para los datos locales.
Archivo XML	.xml	Un archivo XML en blanco.
Esquema XML	.xsd	Un archivo para crear un esquema para documentos XML.
Archivo de configuración	.settings	Un archivo de configuración del usuario en blanco.
Archivo de código	.vb	Un archivo de código en blanco.
Página HTML	.htm	Una página HTML que puede incluir códigos del lado del cliente
Formulario heredado	.vb	Un nuevo formulario basado en un formulario existente.
Control de usuario heredado	.vb	Un nuevo control basado en un control de formulario Windows Forms existente.
Control Web personalizado	.vb	Una clase para crear un control de servidores ASP.NET
Clase COM	.vb	Una clase que puede estar expuesta a COM.
Archivo de texto	.txt	Archivo de texto vacío.
Archivo XSLT	.xslt	Un archivo utilizado para transformar documentos XML.
Clase del instalador	.vb	Una clase que se va a invocar durante la configuración.
Crystal Report	.rpt	Un archivo de Crystal Report que publica datos en un formulario de Windows.
Archivo de mapa de bits	.bmp	Un archivo de imagen de mapa de bits en blanco, que puede utilizarse para crear imágenes simples.
Archivo de cursor	.cur	Un archivo para crear cursores personalizados.
Archivo de icono	.ico	Un archivo de imagen para crear un icono personalizado.
Archivo de recursos	.resx	Un archivo utilizado para editar y definir recursos de aplicaciones.
Archivo de información sobre el ensamblado	.vb	Un archivo que contiene información general acerca del ensamblado...

Elemento de proyecto	Extensión	Propósito del elemento de proyecto
Archivo de configuración de la aplicación	.config	Un archivo utilizado para configurar los valores de la aplicación.
Visualizador del depurador	.vb	Un visualizador del depurador sencillo
Diagrama de clase	.cd	Un diagrama de clase.
Informe	.rdlc	Un nuevo informe vacío.
Archivo JScript	.js	Un archivo de comandos que contiene código JScript.
Archivo VBScript	.vbs	Un archivo de comandos que contiene código VBScript.
Windows Script Host	.wsf	Un archivo que contiene una secuencia de comandos que se ejecuta como un programa de Windows.

### 3. El entorno de desarrollo

Una vez seleccionada la plantilla de la solución, se abre el diseñador de formularios del entorno de desarrollo integrado (IDE) de Visual Studio.



Si se está utilizando la configuración del entorno “Configuración de desarrollo de Visual Basic” la pantalla tendrá el aspecto de la figura anterior.

Como en cualquier aplicación Windows, la parte superior estará ocupada por la barra de menús y debajo la barra de herramientas. Inicialmente se muestra la barra de herramientas “Estándar”, aunque se pueden poner o quitar distintas barras de herramientas, que se pueden seleccionar pulsando con el botón derecho en la zona de la barra.



La parte central está ocupada por los distintos diseñadores de los documentos que tengamos abiertos. Al arrancar un nuevo proyecto con la plantilla “Aplicación para Windows”, aparece el “Diseñador de formularios” que muestra el formulario Windows Form con el que, por omisión, arrancará la aplicación.

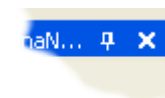
En la parte derecha del entorno aparece la ventana del Explorador de soluciones y, debajo la ventana de Propiedades. En la parte izquierda aparecen dos pestañas que abrirán otras ventanas al pasar el cursor sobre ellas: se trata del Explorador de servidores y la ventana del Cuadro de herramientas.

### 3.1. Navegación y configuración de ventanas

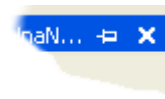
En el entorno de programación existen dos tipos de ventanas: las ventanas de herramientas y las ventanas de documentos. Las ventanas de documento se utilizan para abrir algún documento del proyecto. A las ventanas de herramientas se puede acceder mediante el menú “Ver” o “Depurar”, o mediante los botones situados a la derecha de la barra de herramientas.



Las ventanas de herramientas se pueden mover, mostrar, ocultar, acoplar o desacoplar. Con respecto a la ocultación, podemos hacer que quede fija o que se oculte automáticamente. Eso se consigue margando sobre la “chincheta” en la barra de título.



La ventana está fija



La ventana se muestra automáticamente al pasar el cursor por la pestaña



También se pueden mover por el entorno simplemente pulsando sobre el título de la ventana y desplazándola a otra área. Se pueden volver a acoplar arrastrándola hacia algún borde de la ventana o soltándola sobre otra ventana abierta.

Si se tienen dificultades para restaurar las ventanas a un estado original, siempre se puede volver a él mediante la opción “Importar y exportar configuraciones...” del menú “Herramientas”.

Las ventanas de documento inicialmente tienen el modo de organización por fichas, en la que aparece una ficha por cada documento abierto.

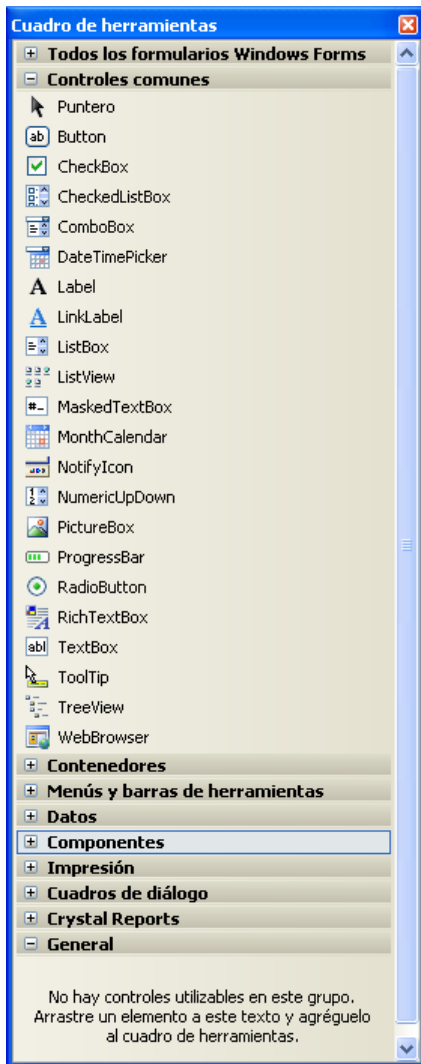


### 3.2. Ventana Cuadro de herramientas

Contiene los controles que se pueden arrastrar para agregarlos a los formularios. Existen varios grupos de controles disponibles cuyo número depende del tipo de diseñador activo en el editor. Cuando se está diseñando un formulario Windows, aparecerán las herramientas necesarias para



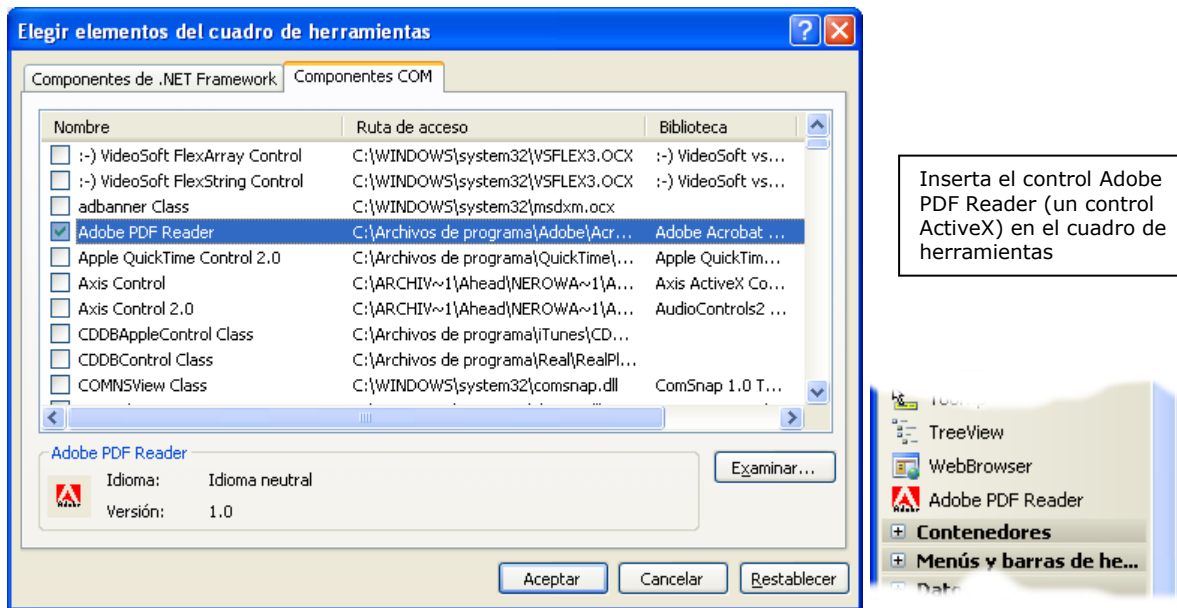
trabajar con formularios Windows, que serán distintas para trabajar con formularios Web y que, a su vez, están agrupados por categorías de controles.



## Personalización del cuadro de herramientas

Es posible crear nuevos grupos o agregar componentes adicionales a un grupo existente. Se pueden agregar otros componentes .NET o controles ActiveX de los utilizados en aplicaciones que no son .NET.

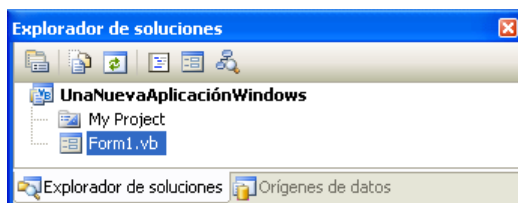
Para agregar nuevos componentes hay que seleccionar la opción “Elegir elementos del cuadro de herramientas” del menú “Herramientas” o del menú contextual que aparece al pulsar con el botón secundario sobre el nombre del grupo. Aparecerá una ventana con dos pestañas, una para agregar componentes ActiveX (componentes COM) y otra para agregar componentes .NET Framework. El componente seleccionado aparecerá en el grupo que aparezca abierto.



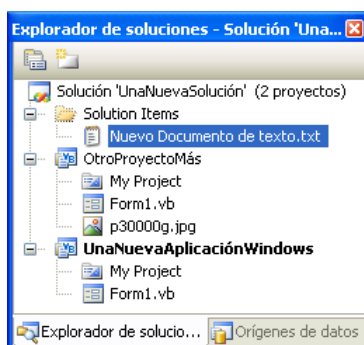
También es posible crear nuevos grupos de elementos mediante la opción “Agregar ficha” del menú contextual que aparece al pulsar con el botón derecho en la cabecera de alguna ficha existente. Podemos mover elementos de un grupo a otro, simplemente arrastrando el elemento, o modificar el orden en que aparecen utilizando las opciones “Subir” o “Bajar” del menú contextual o, simplemente arrastrando los elementos de un lugar a otro. De esta forma se pueden colocar en una ficha propia y en el orden deseado aquellos controles que más se utilicen.

### 3.3. Ventana del explorador de soluciones

Muestra una vista en árbol de todos los elementos de una solución. Inicialmente, la solución que aparece en la plantilla predeterminada de una aplicación para Windows sólo tiene un proyecto que sólo está compuesto por un formulario (`Form1.vb`), por lo que en la ventana sólo aparece una referencia a ese formulario y a “My Project”, que contiene las características del proyecto.



Pero una solución puede tener más de un proyecto y tener asociados a ella o a cada proyecto diversos archivos, referencias a bibliotecas de clases, módulos de código o cualquier otro tipo de archivo (archivos de texto, archivos html, iconos, cursores, etc.). Todos ellos aparecerán en el explorador de soluciones que permitirá navegar entre ellos. Además es posible abrir cualquier elemento pulsando dos veces sobre él, con lo que se abrirá el diseñador correspondiente para facilitar su edición.



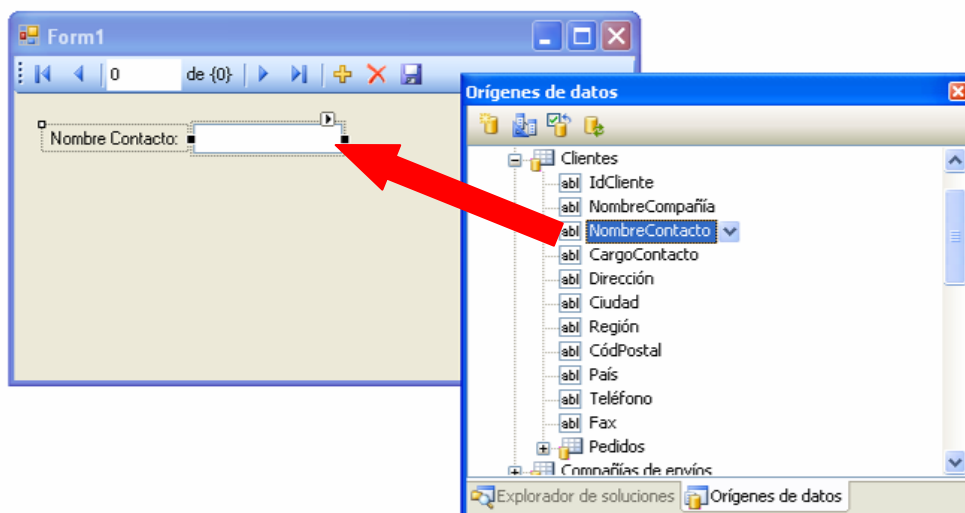
En la fila superior aparece una fila de botones cambia dependiendo del elemento seleccionado. Algunos de los botones:

Botón		Descripción
Ver código		Muestra el código para el archivo seleccionado en el Explorador de soluciones.
Ver diseñador		Muestra el diseñador para el archivo seleccionado en el Explorador de soluciones.
Actualizar		Actualiza el contenido del Explorador de soluciones.
Mostrar todos los archivos		Muestra todos los archivos, incluido el código de los archivos de formulario.
Propiedades		Muestra las propiedades del archivo seleccionado.
Ver diagrama de clases		Muestra una representación gráfico de los tipos utilizados en un proyecto.

El menú contextual de cada elemento de la solución también permitirá realizar distintas acciones como eliminar un componente, añadir uno nuevo, añadir un nuevo proyecto a la solución, generar la solución, etc.

### 3.4. Ventana Orígenes de datos

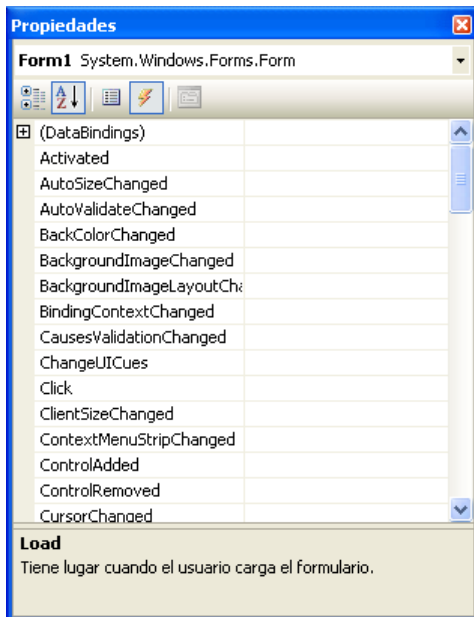
Muestra los orígenes de datos del proyecto en proyectos en los que los datos se toman de alguna fuente externa como bases de datos, servicios Web, etc. Desde esta ventana es posible crear la interfaz de usuario, simplemente arrastrando los distintos elementos de la ventana al diseñador de formularios.



### 3.5. Ventana de propiedades

Al trabajar con clases como formularios, controles, etc., normalmente necesitaremos modificar ciertos atributos (propiedades) de la clase. Estos atributos se pueden modificar mediante código (en tiempo de ejecución) o mediante la ventana de propiedades (en tiempo de diseño). La ventana de propiedades proporciona un acceso a las propiedades accesibles *en tiempo de diseño*<sup>1</sup>. La ventana muestra en la parte superior una lista de las clases disponibles en un entorno dado (por ejemplo, si estamos en el diseñador de un formulario, sólo mostrará los controles de ese formulario) y debajo una lista de las propiedades de la clase, indicando aquellas que se pueden modificar y aquellas que no, marcando estas últimas en gris.

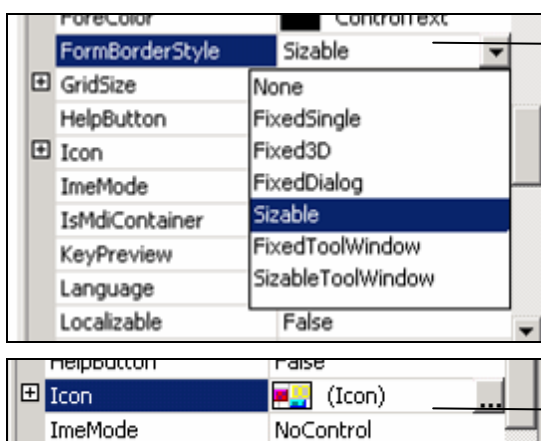
<sup>1</sup> Hay que notar que existen propiedades de sólo lectura y de lectura/escritura. También existen propiedades que sólo son accesibles en tiempo de ejecución, otras accesibles en tiempo de diseño y otras que son accesibles en ambas situaciones. La ventana de propiedades sólo mostrará las propiedades accesibles en tiempo de diseño



En la parte superior de la ventana aparece una lista desplegable que permite seleccionar el elemento del que queremos ver las propiedades y debajo la lista de propiedades que puede tomar varias vistas seleccionables mediante los siguientes botones:

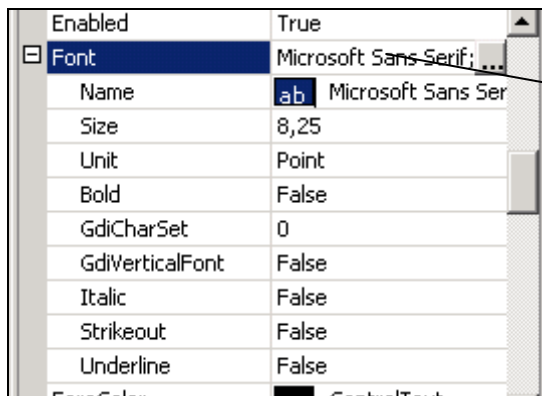
Botón		Descripción
Vista por categorías		Muestra las propiedades o eventos del elemento clasificado por categorías
Vista alfabética		Muestra las propiedades o eventos del elemento ordenados alfabéticamente
Propiedades		Muestra las propiedades del elemento
Eventos		Muestra los eventos del elemento
Páginas de propiedades		En aquellos elementos que lo permiten muestra su página de propiedades, un cuadro de diálogo que permite cambiar varias propiedades de forma conjunta

La modificación de las propiedades se podrá hacer introduciendo texto, seleccionando de una lista desplegable, o mediante el acceso a un submenú. En ocasiones, cuando la propiedad haga referencia a una clase (como en el caso de `Size`, `Location` o `Font`), podrá expandirse para tener acceso a sus propiedades.



A la propiedad `FormBorderStyle` se accede mediante una lista desplegable.

La propiedad `Icon` abre una ventana de diálogo Abrir.

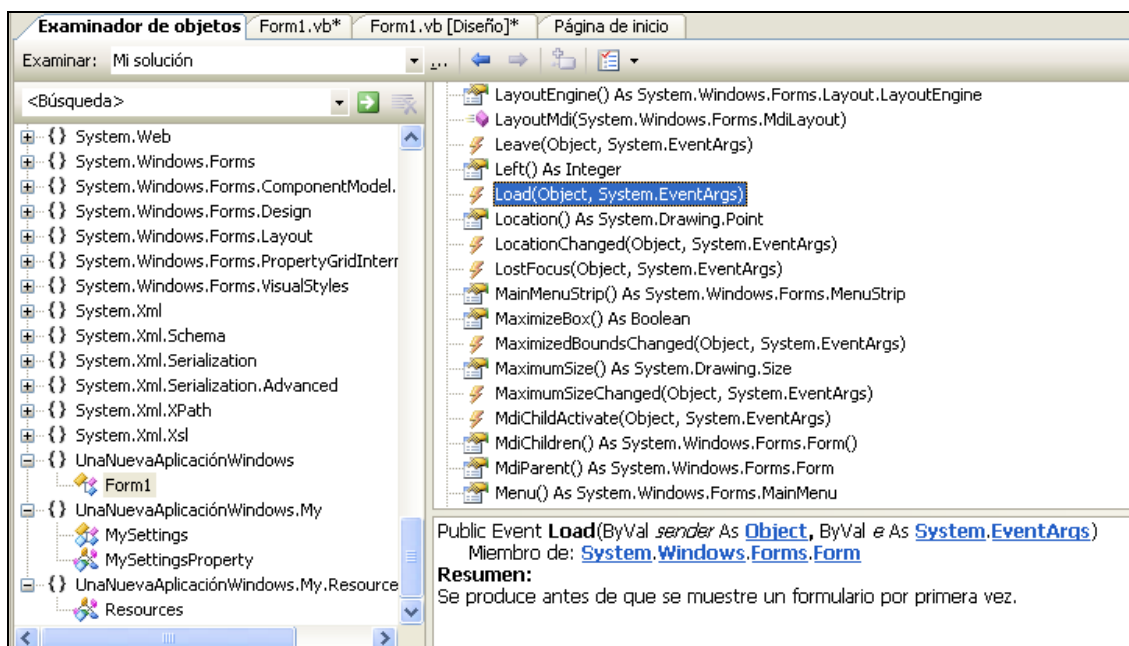


La propiedad `Font` hace referencia a un objeto de la clase `Font` por lo que se podrá acceder a sus propiedades individuales.

### 3.6. Ventana del examinador de objetos

El examinador de objetos permite acceder a las clases y sus miembros de todas las referencias cargadas en el proyecto. Es útil para saber todos los miembros de una clase (un control o cualquier otra clase no visual) y tener una idea aproximada de sus métodos, eventos y propiedades. El examinador de objetos mostrará también la declaración completa de los métodos.

El examinador de objetos no se muestra por omisión, sino que se abrirá en la ventana principal al seleccionar la opción “Examinador de objetos” del menú “Ver” o pulsando F2.







#### Algunos iconos del examinador de objetos y de la vista de clase

Iconos	Descripción	Iconos	Descripción
	Espacio de nombres		Método o función
	Clase		Operador
	Interfaz		Propiedad
	Estructura		Campo o variable
	Unión		Evento
	Enumeración		Constante
	Módulo		Elemento de tipo enumerado

## Iconos de señal

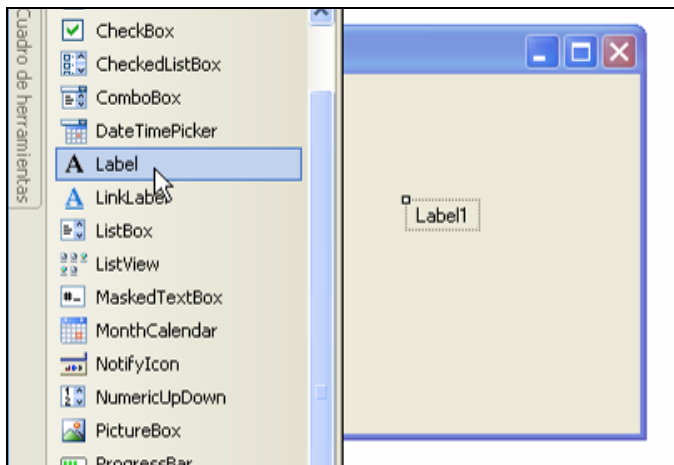
Los iconos de señal siguientes se aplican a todos los iconos anteriores e indican su accesibilidad.

Iconos	Descripción
<Sin icono de señal>	Public — Accesible desde cualquier lugar de este componente y desde cualquier componente que haga referencia a él.
	Protected — Accesible desde dentro del tipo o clase que lo contiene, o los derivados del tipo o clase que lo contiene.
	Private — Accesible desde el tipo o clase que lo contiene.
	Internal — Accesible sólo desde este componente.
	Friend — Accesible sólo desde dentro del proyecto.

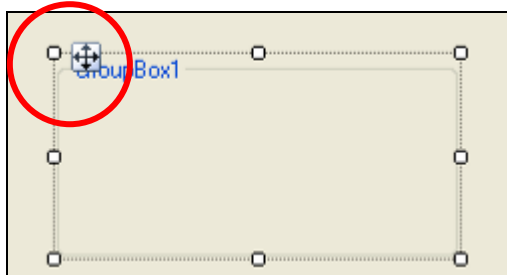
## 4. El diseñador de formularios

### 4.1. Tareas comunes: dibujar y colocar controles

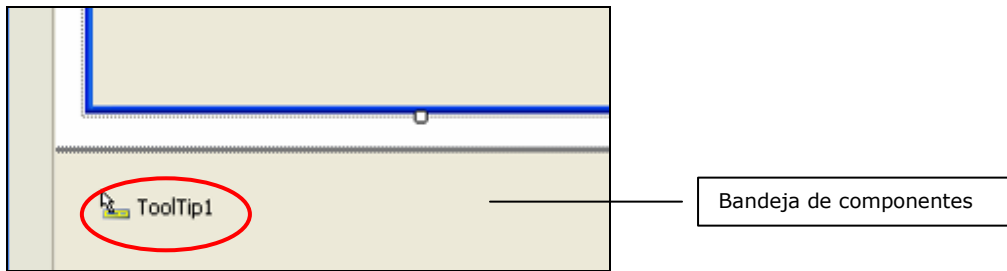
La plantilla de un proyecto basada en una aplicación Windows crea de forma predeterminada un formulario vacío sobre el que dibujaremos los controles. La forma de añadir un nuevo control a un formulario es abrir la caja de herramientas, seleccionar el grupo y el control deseado y hacer clic sobre el formulario o arrastrar en él hasta darle el tamaño requerido. De forma alternativa, es posible incluir un control haciendo doble clic sobre el control deseado. En este caso, si esta seleccionado el formulario, lo posicionará en la esquina superior izquierda del mismo; si está seleccionado otro control, el nuevo control lo posicionará sobre el mismo.



Para desplazar la mayoría de los controles simplemente hay que seleccionarlo y arrastrar con el ratón cuando el cursor toma la forma de una cruzeta. Sin embargo, para desplazar algunos controles es preciso utilizar el tirador que tienen en forma de cruzeta.



Existen controles que no son visibles. En versiones anteriores de Visual Basic, estos controles también aparecían sobre el formulario. En Visual Studio .NET, esos controles (como los cuadros de diálogo, el control `ToolTip`, o el control `Timer`) se dibujan sobre el formulario, pero aparecen en la bandeja de componentes situada debajo del formulario.



## Alinear controles y organizar controles

El entorno dispone de herramientas que permiten dar mayor precisión a la colocación de los controles mediante las características de alineación, que aparecen en la opción “Alinear” del menú “Formato” o en la barra de herramientas de “Diseño” (se puede seleccionar esa barra de herramientas u otra cualquiera pulsando con el botón derecho en una zona de menús).

Para seleccionar varios controles se pueden marcar varios controles mientras se mantiene pulsada la tecla CTRL o MAYÚS. El primer control seleccionado será el que actúe como control primario y los demás controles se alinearán a partir de él.

Las distintas opciones de alineación disponibles son:

Icono	Descripción	Icono	Descripción
	Alinea los controles por abajo		Alinea los controles a la izquierda
	Centra los controles horizontalmente		Alinea los controles a la derecha
	Alinea los controles por arriba		Centra los controles verticalmente

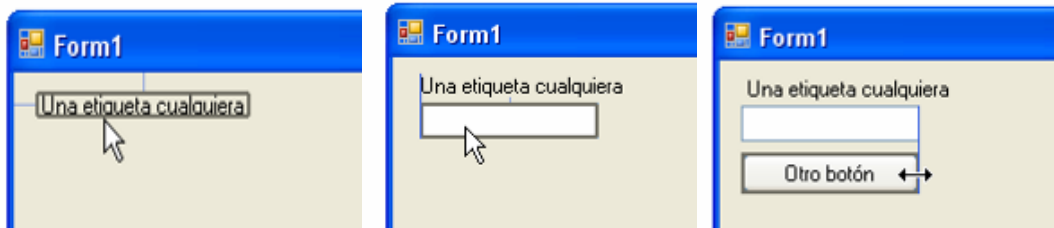
Además, mediante la misma barra de menús es posible realizar otras operaciones para organizar los controles:

Icono	Descripción	Icono	Descripción
	<b>Alinea el control a la cuadrícula</b>		<b>Igualar ancho</b>
	Iguala el tamaño del control a la cuadrícula		Igualar ancho
	Iguala tamaño (ancho y alto)		Iguala espacio horizontal
	Aumenta espacio horizontal		Disminuye espacio horizontal
	Quitar espacio horizontal		Iguala espacio vertical
	Aumenta espacio vertical		Disminuye espacio vertical
	Quitar espacio vertical		Centrar horizontalmente
	Centrar verticalmente		Enviar al frente
	Enviar al fondo		

La mayor parte de estas opciones de alineación sólo se pueden utilizar cuando existen varios controles seleccionados. En estos casos, la operación se realizará sobre el **último** control seleccionado.

El diseñador de formularios de VisualStudio 2005 también dispone de líneas de ajuste dinámicas (SnapLine) que permiten alinear, colocar e igualar el tamaño de los controles en tiempo de diseño. Las líneas de ajuste aparecen cuando se está moviendo un control o grupo de controles seleccionados cerca de una posición en la que se podría alinear, colocar o igualar el tamaño respecto a otros controles o respecto al borde de un contenedor. Los elementos seleccionados se ajustarán a la posición sugerida.





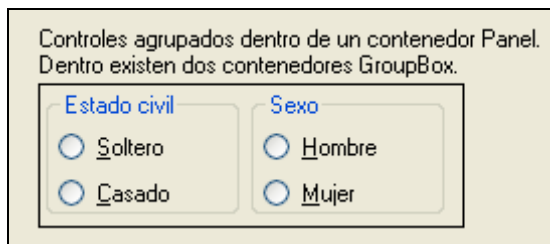
La distancia de la línea de ajuste sugerida será la suma de los valores de la propiedad `Margin` del control y de la propiedad `Padding` del contenedor.

### Contenedores y controles de un contenedor

Algunos controles Windows Forms del Framework permiten funcionar como contenedores de otros controles: se trata de las clases `Form`, `GroupBox` y `Panel`, que heredan de la clase `ContainerControl`. Para incluir controles dentro de un contenedor, simplemente hay que dibujarlos dentro o arrastrar un control dentro del contenedor.

La selección de varios controles dentro de un contenedor se debe hacer siempre utilizando las teclas `CTRL` o `MAYÚS`.

Los controles agrupados dentro de un contenedor se moverán, copiarán o pegarán con él, y se eliminarán cuando se elimine el contenedor.



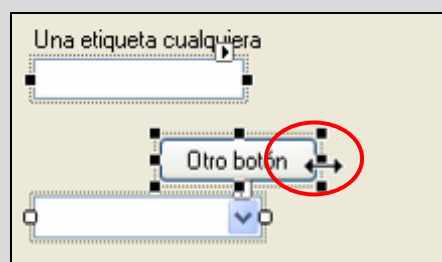
Existen además otros dos contenedores no visibles que se utilizarán para ajustar de forma dinámica la distribución de los controles que contiene al tamaño del formulario: el control `FlowLayoutPanel` y el control `TableLayoutPanel`. El uso de los mismos se tratará más adelante.

## 4.2. Modificación de propiedades en tiempo de diseño

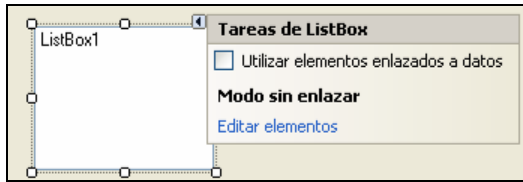
Para modificar las propiedades de un control en tiempo de diseño se utiliza la ventana de propiedades. Primero habría que seleccionar el control de la lista que aparece en la parte superior de la ventana o en el mismo formulario, con lo que aparecerían en la ventana todas las propiedades de lectura y escritura modificables en tiempo de diseño (existen otras propiedades que sólo se pueden modificar en tiempo de ejecución).

También es posible modificar las propiedades de varios controles, seleccionándolos en el formulario, con lo que en la ventana de propiedades aparecerán las propiedades comunes a todos.

Cuando hay varios controles seleccionados, se puede modificar el tamaño de todos a la vez mediante los tiradores de los controles.

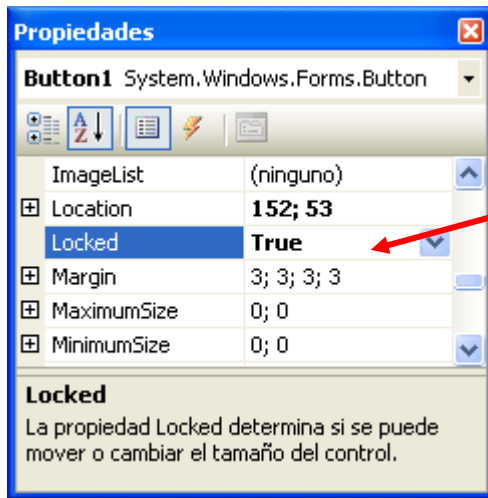


En algunos controles es posible acceder a las tareas más comunes pulsando sobre la flecha de la esquina superior izquierda.

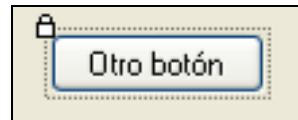


## Bloquear controles

Una vez que tenemos controles ubicados en algún lugar, podemos bloquear su ubicación para evitar moverlos por accidente. Para ello, tenemos que modificar la propiedad `Locked` del control y ponerla a `True`.



Al seleccionarlo, el control tendrá el siguiente aspecto en el formulario.



Los valores de esta propiedad se pueden modificar mediante una lista desplegable. En estos casos, podemos desplegar la lista, o, simplemente, pulsar dos veces sobre el valor.

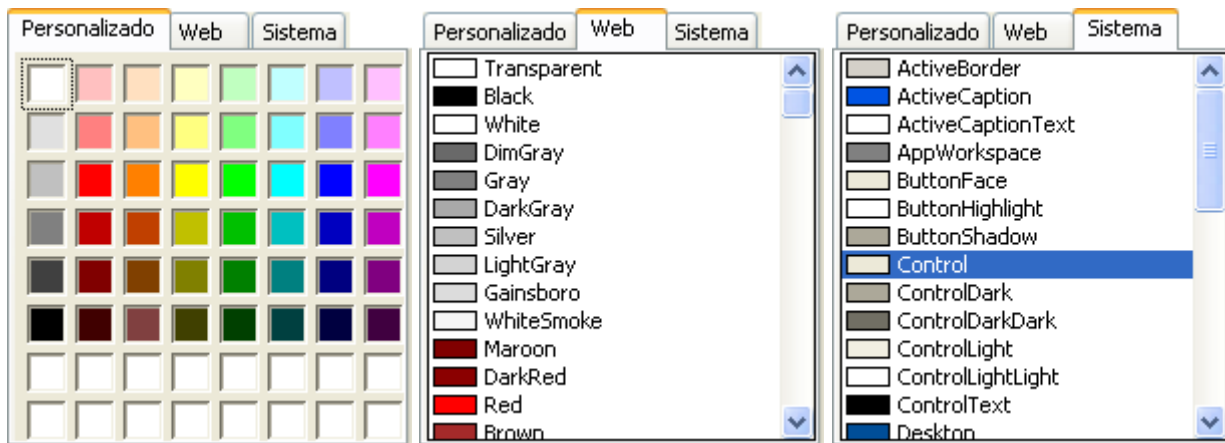
También es posible bloquear todos los controles de un formulario mediante la opción “Bloquear controles” del menú “Formato”.

## Nombrar controles

Cuando se coloca un control en el formulario, se le asigna un nombre predeterminado relacionado con el tipo de control y el número de controles del mismo tipo ya colocados (`Button1`, `Button2`, `Button3`,..., `TextBox1`, `TextBox2`, `TextBox3`,...). Esta mecánica evita al usuario tener que asignar un nombre a cada control que utiliza, pero también puede hacer que la lectura y el código de una aplicación se complique. Por ello es conveniente asignar a cada componente un nombre significativo mediante la propiedad `Name` que permita localizarlo en la lista de controles del proyecto y en el código. Esto es especialmente interesante si nos vamos a referir al control desde dentro del código. Siempre será más significativo leer `mensaje = Nombre.Text` que `mensaje = TextBox1.Text`. Mientras que en Visual Studio 6, Microsoft aconsejaba el uso de prefijos para nombrar controles (`txt` para controles `TextBox`, `lbl` para `Label`, `frm` para `Form`, etc.), en Visual Studio .NET no da ninguna recomendación especial para nombrar controles aunque es una buena costumbre que ayudará a clarificar el código de la aplicación.

## Propiedades que trabajan con el color

Todos los controles visibles tienen la propiedad `ForeColor` y `BackColor` para asignar el color de primer plano y de fondo. A la hora de elegir un color, se puede escoger de entre tres paletas distintas, para colores personalizados (colores RGB), colores Web o colores del sistema.

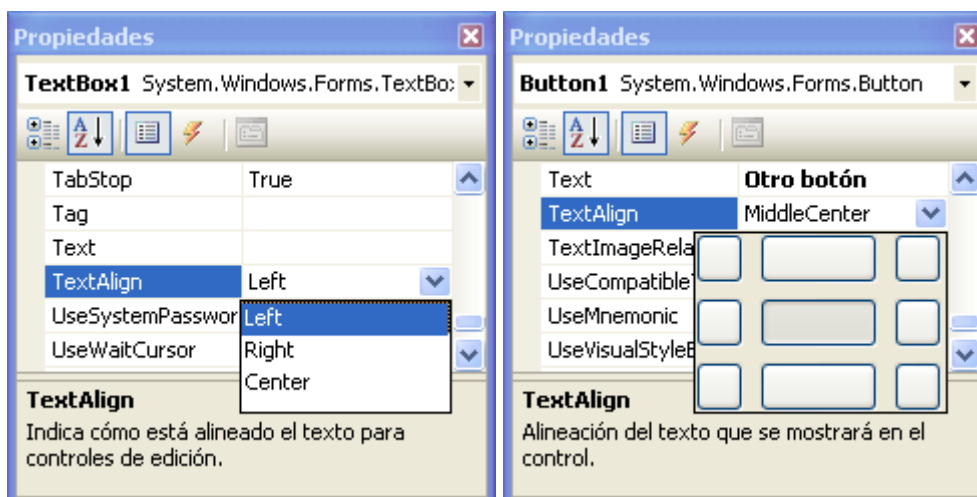


En la primera la elección de colores se basa en el estándar RGB, es decir mediante la combinación del componente rojo, verde y azul del color. La paleta de color Web utiliza la nomenclatura de colores que utiliza el World Wide Web Consortium (por ejemplo en la recomendación para hojas de estilo CSS3 o la especificación SVG 1.0). La paleta de colores del sistema utiliza los colores definidos por el usuario en las propiedades de la pantalla y utilizados Windows para los diferentes componentes del escritorio. Esta es la opción más recomendable en la mayoría de aplicaciones de escritorio, ya que permite que el aspecto de la interfaz cambie cuando se cambia la apariencia del sistema operativo.

## Trabajo con texto

La gran mayoría de los controles tienen la propiedad `Text` que se utiliza, dependiendo del control, para especificar texto estático o texto que el usuario puede cambiar. Estos controles asignan una propiedad `Text` predeterminada que coincide con el nombre que el entorno asigna al control.

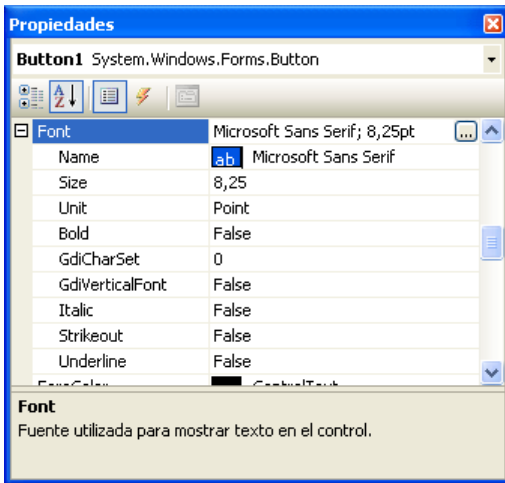
Normalmente, el texto aparece alineado a la izquierda. Sin embargo, algunos controles (`Label`, `TextBox`, `Button`, `CheckBox`, `RadioButton`, `NumericUpDown` y `DomainUpDown`) poseen la propiedad `TextAlign` que permite especificar la alineación. La selección del tipo de alineación se puede hacer seleccionando de una lista desplegable, cuya forma visual cambiará según el tipo de control.



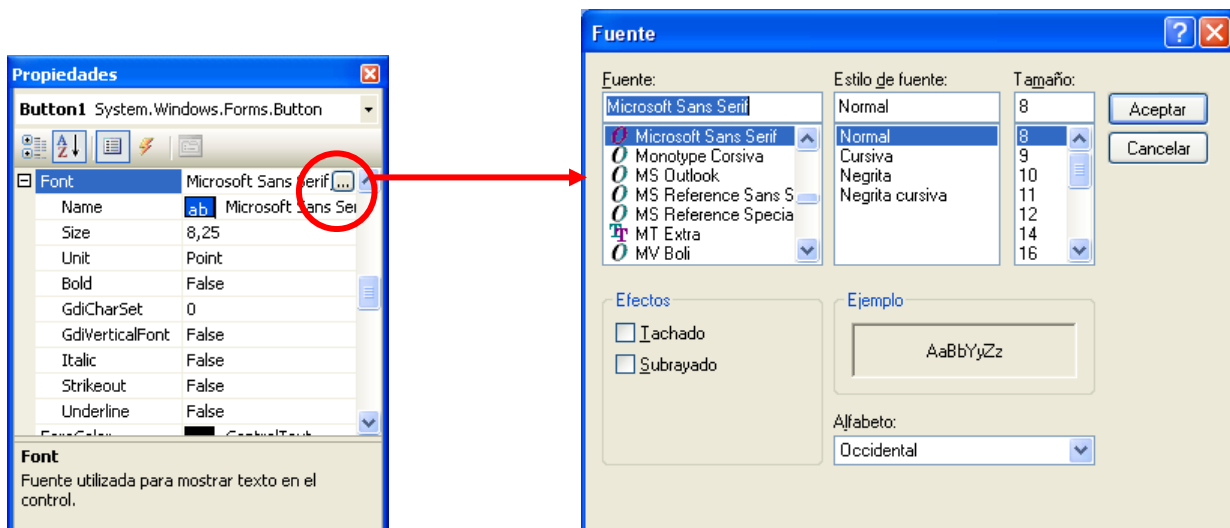
La propiedad `Font` permite cambiar la fuente utilizada en un control. En realidad, dicha propiedad hace referencia a un objeto de la clase `Font` al que es posible cambiar las propiedades mediante una paleta de selección de fuentes o desplegando los valores de las propiedades en la ventana de propiedades. La propiedad `Font` de cada control del formulario apunta a la propiedad `Font` de su contenedor, por lo que si se cambia en un contenedor todos sus controles heredarán la nueva

fuente (por ejemplo, al cambiarla en un formulario, todos los controles cambiarán la propiedad). Sin embargo, siempre será posible cambiar la propiedad en un único control.

Para cambiar la propiedad `Font` podemos desplegar la propiedad (pulsando el signo + de la propiedad) y cambiar cada uno de los atributos que componen el objeto.

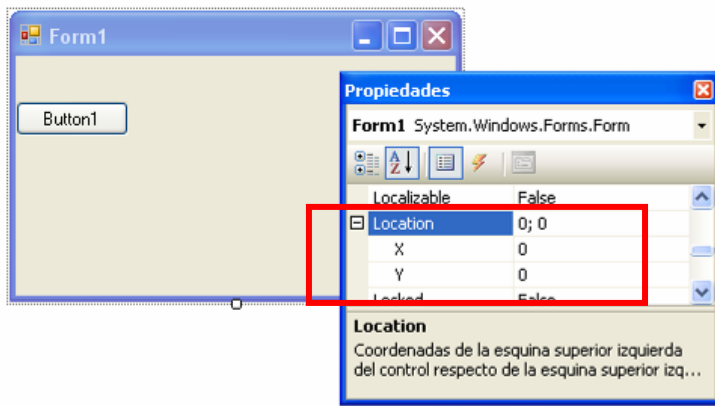


También podemos abrir la página de propiedades de la fuente pulsando al botón con los puntos suspensivos que aparece a la derecha y cambiar las propiedades mediante un cuadro de diálogo.

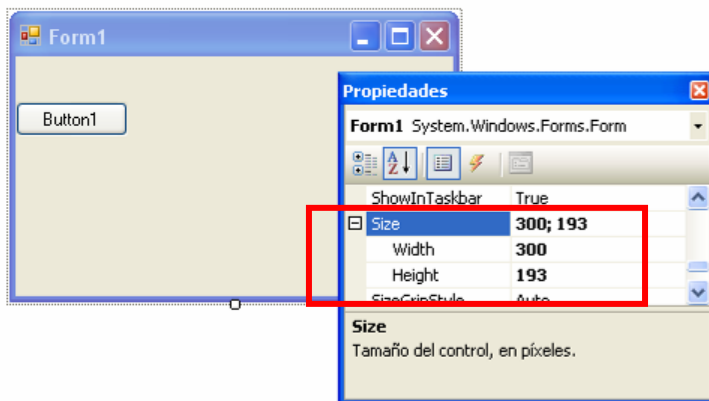


## Controlar el tamaño y la posición

Una vez colocados los controles es posible moverlos utilizando el ratón o arrastrándolos por el formulario o modificando la propiedad `Location` en la ventana de propiedades. La propiedad `Location` permite establecer el punto superior izquierdo del control respecto al contenedor donde está situado.



El tamaño de un control se modifica mediante la propiedad `Size`. La propiedad hace referencia a un objeto de la clase `Size` que tiene las propiedades `Width` y `Height` y el tamaño se asignará en píxeles.



En VisualStudio 2005 las medidas siempre se establecen en píxeles.

### Anclar y acoplar controles. Adaptar controles a los formularios

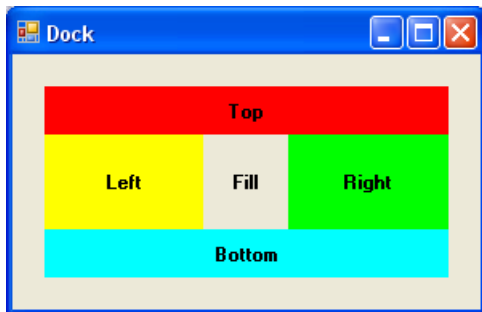
En versiones anteriores de Visual Basic, era muy laborioso adaptar el tamaño y la posición de los controles cuando se modificaba el tamaño del contenedor. Visual Studio .NET expone dos propiedades., `Anchor` y `Dock`, y dos nuevos controles `FlowLayoutPanel` y `TableLayoutPanel` que facilitan esta tarea.

La propiedad `Dock`, permite acoplar un control al borde de su contenedor. En tiempo de diseño se podrá seleccionar el borde del acople mediante la ventana de propiedades y que permite realizar el acople en la parte superior (`Top`), la inferior (`Bottom`), a la izquierda (`Left`), a la derecha (`Right`) o rellenar todo el hueco no ocupado por controles (`Fill`).





Todos los objetos contenedores exponen además la propiedad `Padding` que permite especificar el espacio de separación del control respecto al borde del contenedor. En los formularios anteriores, la propiedad `Padding` está a 0 (su valor predeterminado), mientras que en el formulario siguiente la propiedad está a 20 para todos los bordes.



La propiedad `Anchor` permite fijar un control a los bordes de su contenedor. Inicialmente su valor es `Top, Left`, por lo que al modificar el tamaño del contenedor no se modifica ni la posición ni su tamaño.

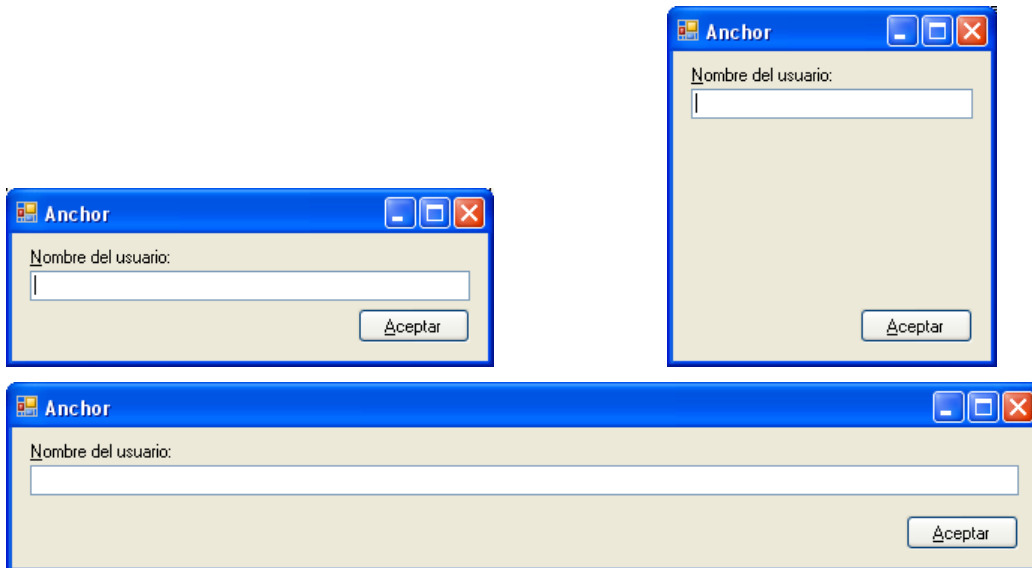


Algunas situaciones posibles de los anclajes podrían ser:

- Si se fija el control con los bordes superior derecho, el control se pegará al borde derecho del formulario y se desplazará horizontalmente cuando el contenedor se expanda o se reduzca.
- Si se ancla el control con los bordes inferior derecho, el control se desplazará y mantendrá su distancia con la esquina inferior derecha del formulario.
- Si el control se fija con los bordes izquierdo, superior y derecho, el control quedará pegado al borde superior del contenedor y cambiará su tamaño cuando el contenedor amplía o reduzca su tamaño.
- Si se fijan los bordes izquierdo y derecho el control cambiará de tamaño cuando se cambie la anchura del contenedor, y se moverá hacia arriba o hacia abajo para que su relación entre la distancia con la parte superior e inferior del contenedor sea proporcional a la que tenía en tiempo de diseño.
- Si se fijan los bordes superior e inferior, el control cambiará su altura cuando se cambie la altura del contenedor.

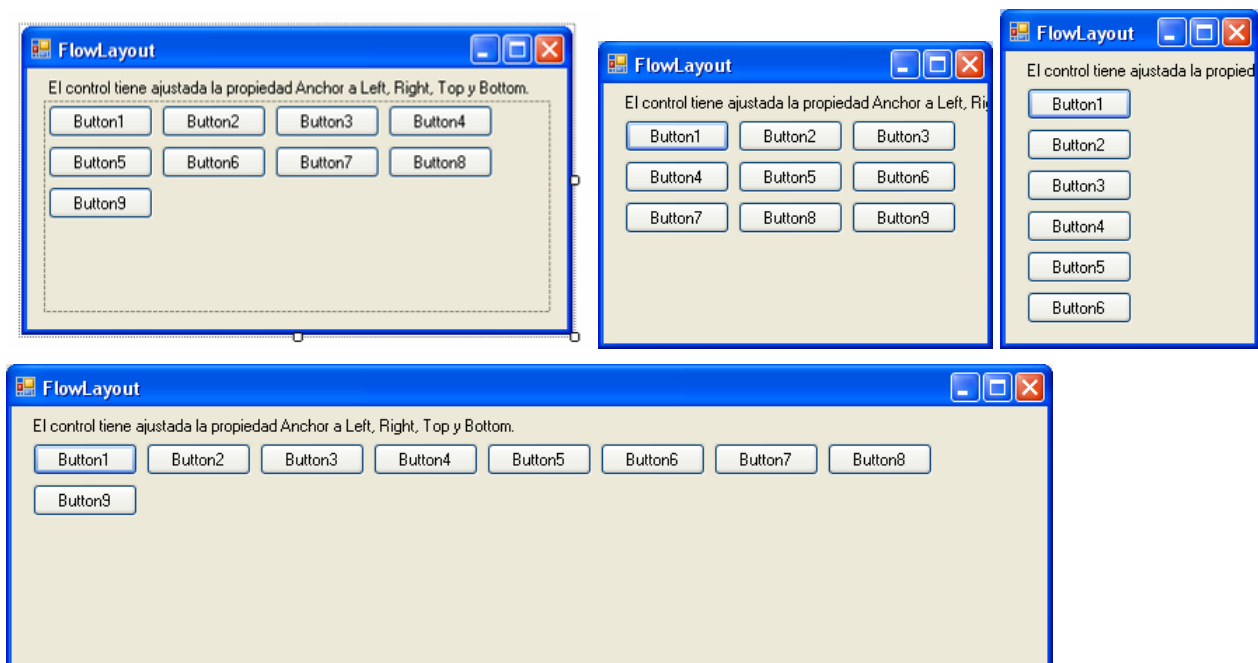
Para tener más control en la disposición de los controles en un formulario, la clase `Form`, permite asignar un objeto de la clase `Size` a las propiedades `MaximumSize` y `MinimumSize` para especificar el tamaño máximo y mínimo del mismo.

En el siguiente formulario, el texto estático se ha anclado en los bordes superior e izquierdo, el cuadro de texto se ha anclado en los bordes superior, izquierdo y derecho y el botón en los bordes inferior y derecho.



En Visual Studio 2005 los contenedores `FlowLayoutPanel` y `TableLayoutPanel` permiten un mayor control en tiempo de ejecución del tamaño y disposición de los controles en el formulario.

El control `FlowLayoutPanel` organiza los controles que contiene en una dirección del flujo horizontal o vertical, ajustando su contenido a la fila o columna siguiente.



El control `TableLayoutPanel` representa un panel que dispone su contenido de forma dinámica distribuyéndolo por filas y columnas, de forma que puede cambiar su tamaño en tiempo de ejecución dependiendo de las condiciones del entorno (resolución, tamaño de las etiquetas cuando se está traduciendo la aplicación a otro idioma, etc.).



Este control es el adecuado para utilizar con aplicaciones localizadas (aplicaciones para varios idiomas) o cuando se colocan los controles de forma dinámica en el formulario

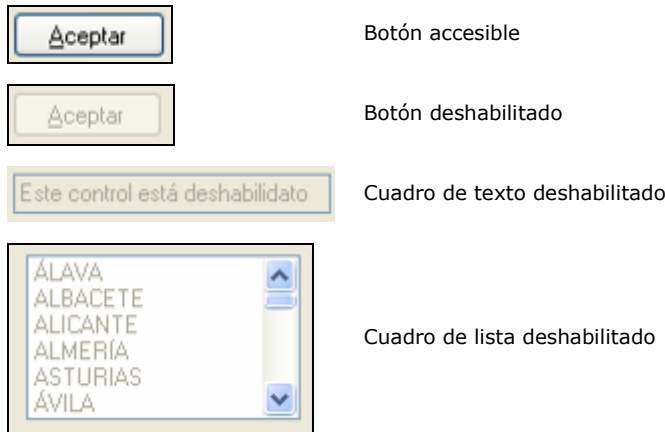
En la documentación de Visual Studio 2005, aparece un ejemplo completo de utilización de este contenedor en un formulario de entrada de datos de tamaño variable. El ejemplo se puede ver en “Cómo: Crear un formulario Windows Forms de entrada de datos de tamaño variable” en el enlace [ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.es/dv\\_fxmclict1/html/babdf198-404c-485d-a914-ed370c6ecd99.htm](http://ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.es/dv_fxmclict1/html/babdf198-404c-485d-a914-ed370c6ecd99.htm) de la ayuda de Visual Studio 2005

## Control del foco de entrada

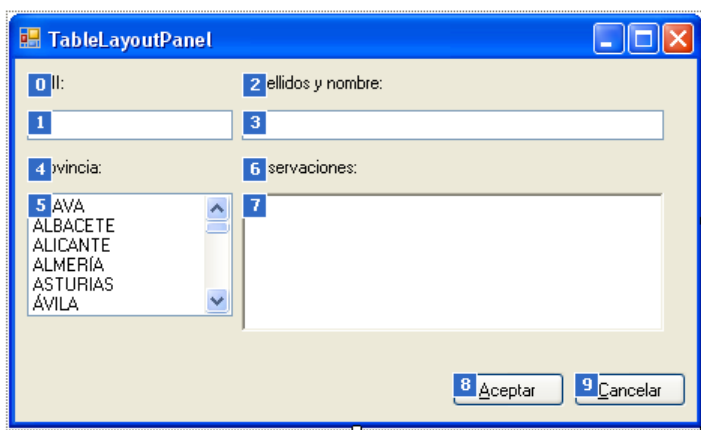
El concepto de *foco de entrada* se utiliza para indicar que control es el activo. Todos los controles visibles en los que se pueda interaccionar pueden coger el foco de entrada que irá cambiando al

pulsar las teclas TAB (siguiente control en el orden de tabulación) o MAYÚS+TAB (control anterior en el orden de tabulación).

Para que un control aparezca como no visible, se puede establecer la propiedad `Visible` a `False` en la ventana de propiedades. Si no queremos interactuar con un control se puede establecer la propiedad `Enabled` a `False`. En estos casos, si el control tiene asociado texto estático, el aspecto de la etiqueta aparecerá como deshabilitado.



Cuando se diseña un formulario, el orden de tabulación establecido por omisión está relacionado por el orden en que se ubican los controles en el formulario. La propiedad `TabIndex` indica, mediante un número entero, el orden de tabulación de los controles de un formulario (el primer control en tomar el foco es el que tenga la propiedad `TabIndex` a 0. Esta propiedad se puede modificar mediante la ventana de propiedades. Pero Visual Studio 2005 ofrece una forma más gráfica de mostrar el orden de tabulación de los controles, mediante la opción “Orden de tabulación” del menú “Ver” que aparece cuando el diseñador de formularios es la ventana activa. Con esta opción, aparecerá un número asociado a cada control que indicará su valor de la propiedad `TabIndex`; seleccionando los controles en que se desea establecer el orden de tabulación se cambiará el valor de la propiedad.



Podemos elegir que ciertos controles no intervengan en el orden de tabulación modificando la propiedad `TabStop`, presente en todos los controles visibles con capacidad de interacción, a `False`. Cuando se cambia el valor de la propiedad, se podrá interactuar con el control, pero al pulsar la tecla TAB, se saltará dicho control.

Hay que tener en cuenta que todos los controles visibles tienen un orden de tabulación, incluso aquellos en los que no se puede interactuar. Esto es importante para poder asignar teclas de acceso a controles que no tengan asociado texto estático (como los `TextBox` o los `ListBox`). En estos controles, la tecla de acceso será la definida en el texto estático que tenga un número de orden de tabulación anterior. Por ejemplo, en el formulario anterior, al pulsar Alt+P (inicial subra-

yada en el texto estático Provincia), el foco pasaría al cuadro de lista con las provincias, ya que la etiqueta de texto estático tiene el orden de tabulación 4 y la lista el valor de la propiedad `TabIndex` a 5.

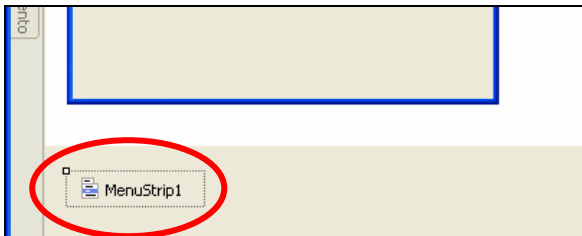
## 5. Visual Studio 2005 como herramienta de prototipado

En este documento se explicarán algunas nociones básicas para simular la interacción en una aplicación Windows desarrollada con Visual Studio. Su fin no es explicar la codificación de una aplicación Visual Basic .NET, sino dar algunas “recetas” para realizar un prototipo de una aplicación utilizando como herramienta de prototipado el propio entorno de desarrollo de Visual Studio.

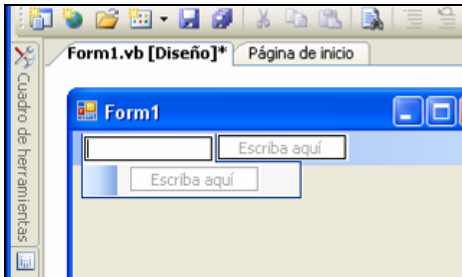
### 5.1. Manejo de algunos controles complejos

#### Menús

Los menús de una aplicación Windows Form descansan sobre el control `MenuStrip`. Al insertar el elemento en un formulario no aparece en el área normal del formulario, sino en la parte inferior del diseñador de formularios, en la bandeja de componentes.



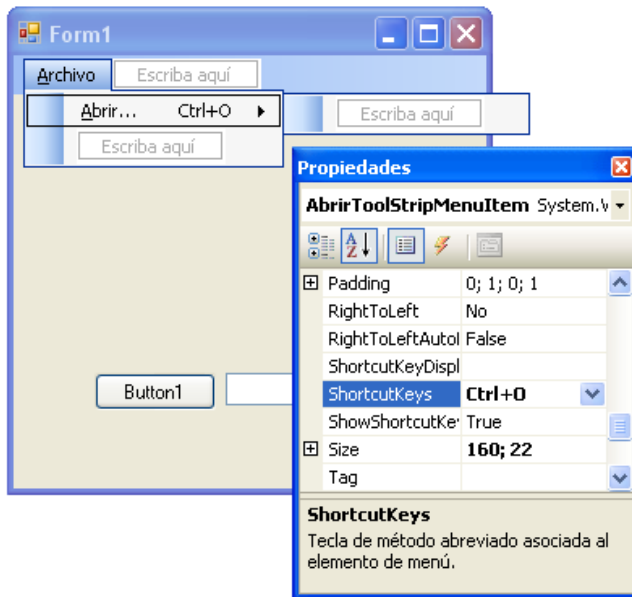
Al seleccionar el control, aparece un editor de menús en la parte superior del formulario donde se podrán poner las etiquetas de los menús. A medida que se rellenen los distintos elementos de menú se irán abriendo nuevos espacios para nuevos elementos de la barra de menús o de los submenús de cada elemento.



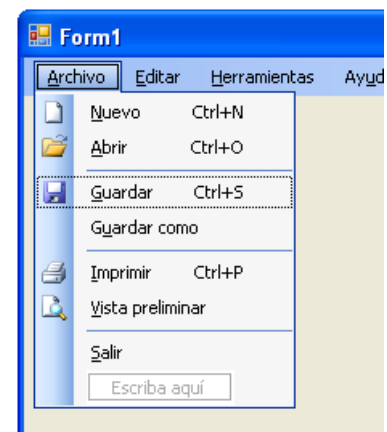
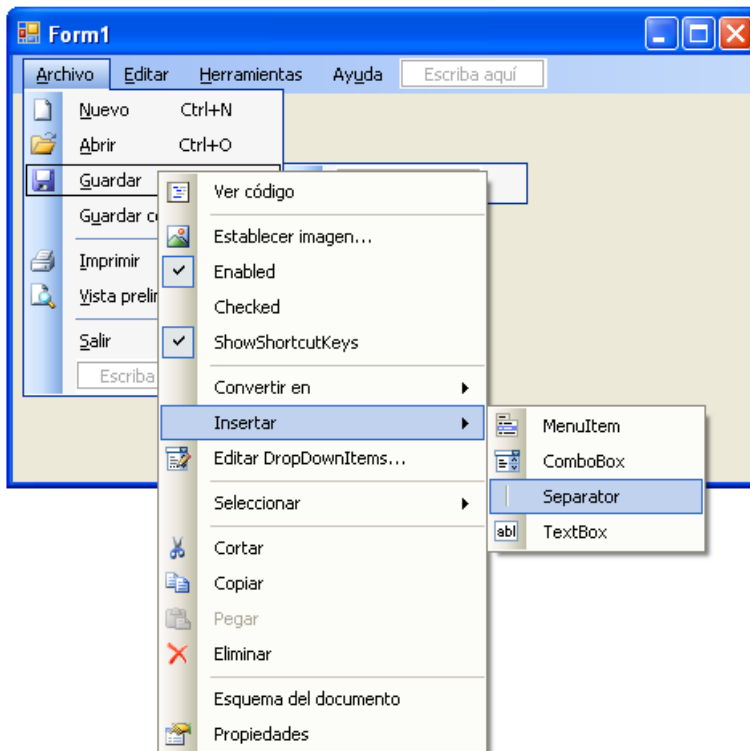
A la hora de poner el nombre no hay que olvidar las recomendaciones de estilo comentadas por Microsoft y recogidas en el tema 4:

- En el título de menú (propiedad `Text`), se puede utilizar el símbolo de *ampersand* (&) delante de un carácter para indicar la tecla de acceso rápido (las que aparecen subrayadas, en la figura que aparece más arriba sería el carácter A). Las recomendaciones de diseño de Microsoft, indican que **todos** los elementos de un menú deben tener una tecla de acceso rápido.
- Si una opción de menú implica la apertura de una ventana (por ejemplo, lo que ocurre en el menú Abrir en las aplicaciones Windows), el título debe aparecer seguido de tres puntos (por ejemplo, lo correcto sería “Abrir...”).
- En las opciones más utilizadas se pueden utilizar atajos de teclado (combinaciones de las teclas CTRL, ALT, SHIFT con otras). Para añadir a un elemento de menú un atajo de teclado hay que seleccionar el elemento y en la ventana de propiedades modificar la propiedad

ShortcutKeys. Los atajos de teclado se deben poner en aquellas etiquetas que no tienen submenús.

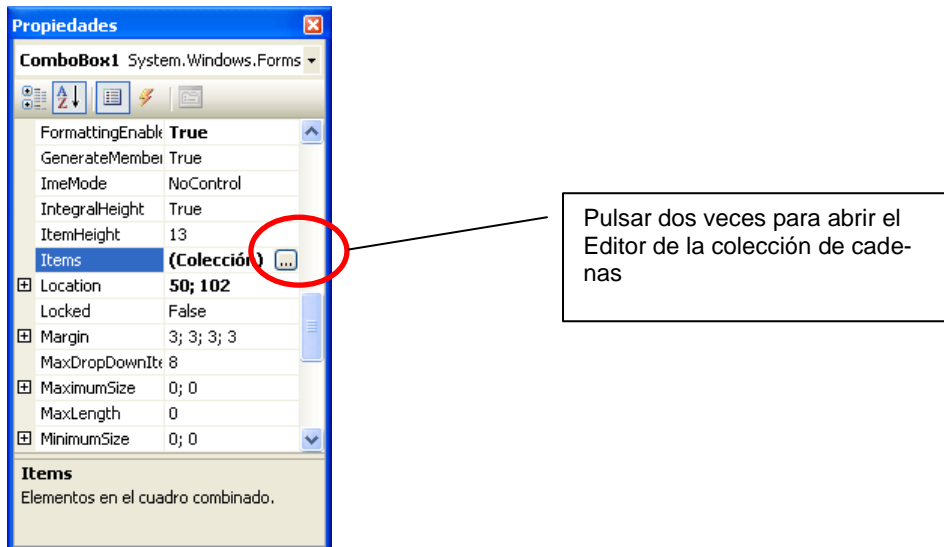


- Se pueden utilizar separadores para agrupar las distintas opciones de un menú desplegable. Para añadir un separador se puede utilizar el menú contextual e indicar la opción "Separator" del menú "Insertar"



## Rellenar con elementos las listas (ListBox, ComboBox, etc.) en tiempo de diseño

Los elementos de una lista se guardan en la propiedad `Items`.



Aparece entonces el editor de la colección de cadenas. Allí simplemente se escriben los elementos pulsando INTRO después de cada uno.



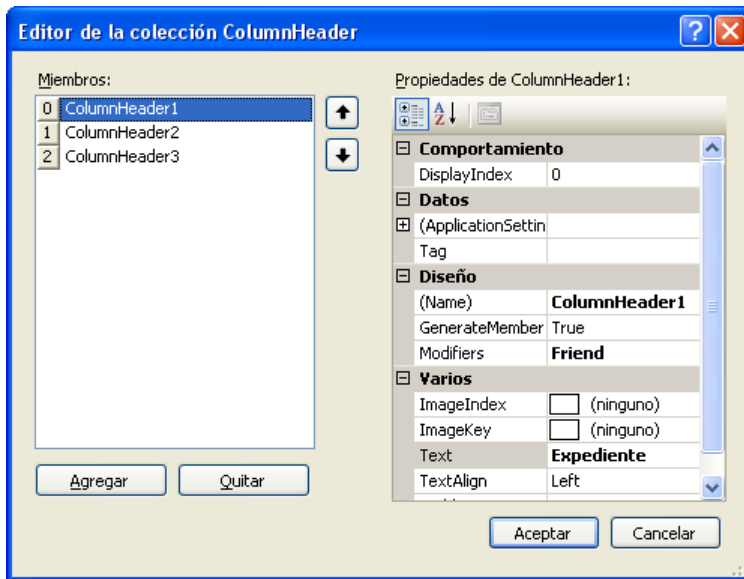
### Crear una rejilla con información tabular en tiempo de diseño

Se pueden crear rejillas utilizando el control `ListView`. Este control presenta distintas vistas, y una de ellas (`Details`) permite la vista en forma de filas y columnas. Por lo tanto, los pasos iniciales serían:

1. Crear un control `ListView`
2. Ir a la propiedad `View`
3. Seleccionar el valor `Details`.

### Definir las columnas

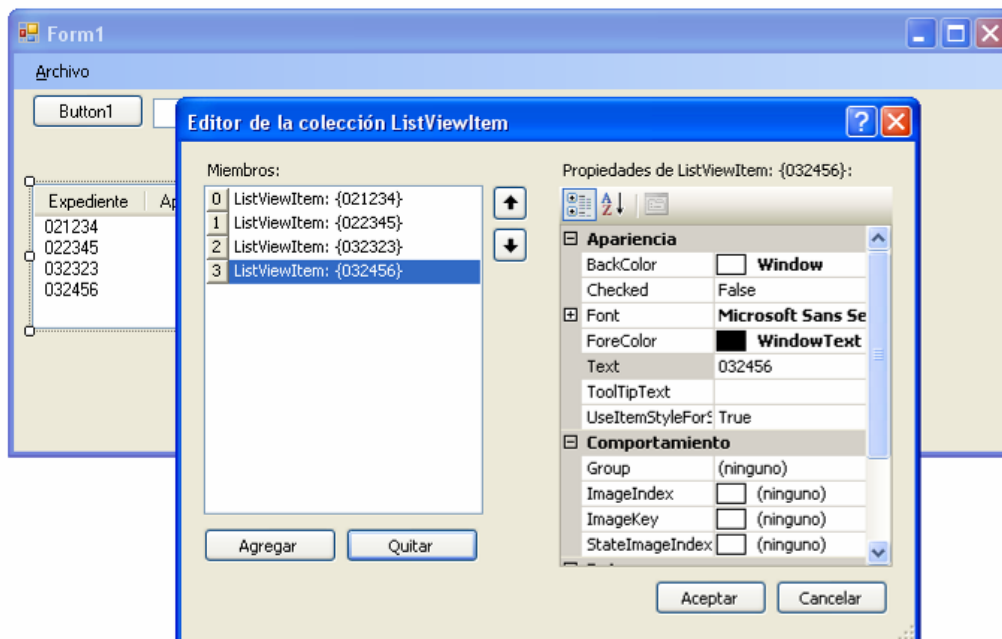
Para poder utilizar esta vista, primer hay que crear las columnas y sus encabezados. La propiedad `Columns`, permite crear las columnas y especificar sus encabezados.



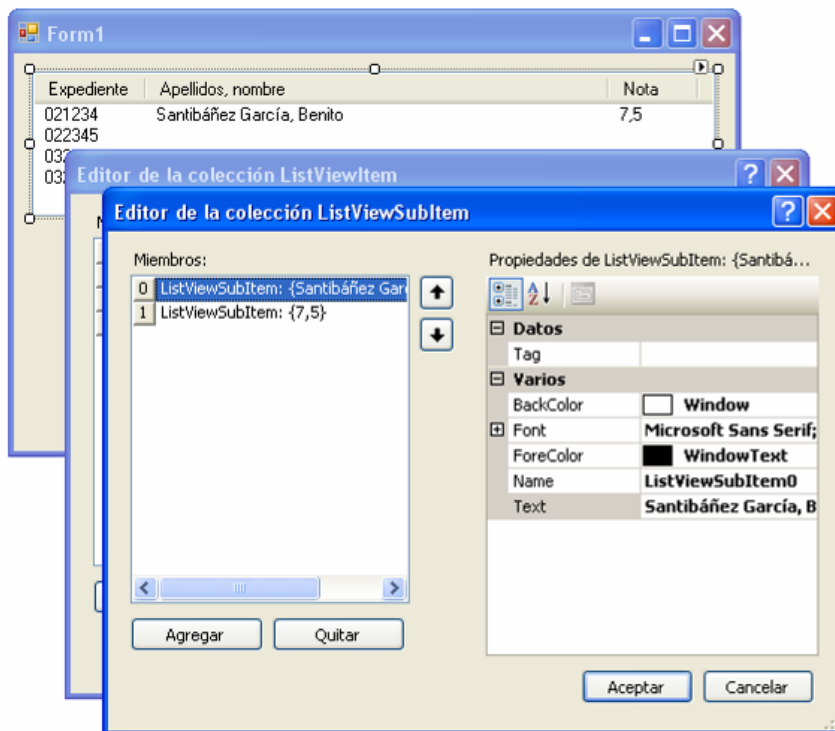
1. Para crear una nueva columna dar el botón “Agregar”.
2. Para añadir un título a la columna teclear el valor en la propiedad `Text`.

### Dar un contenido (fijo) a las columnas

La propiedad `Items` permite crear elementos en la lista. Cada elemento puede tener varios subitems. El texto de cada `Item` será el que aparezca en la primera columna. El texto de cada `subitem` será el que aparezca en las columnas siguientes.



1. Para crear nuevos elementos utilizar el botón Agregar.
2. Para dar un texto a cada elemento (el que aparece en la nueva columna) rellenar la propiedad `Text`.
3. Para crear subitems (columnas sucesivas) se utiliza la propiedad `SubItems` que accede a la ventana de edición de subitems de ese elemento.



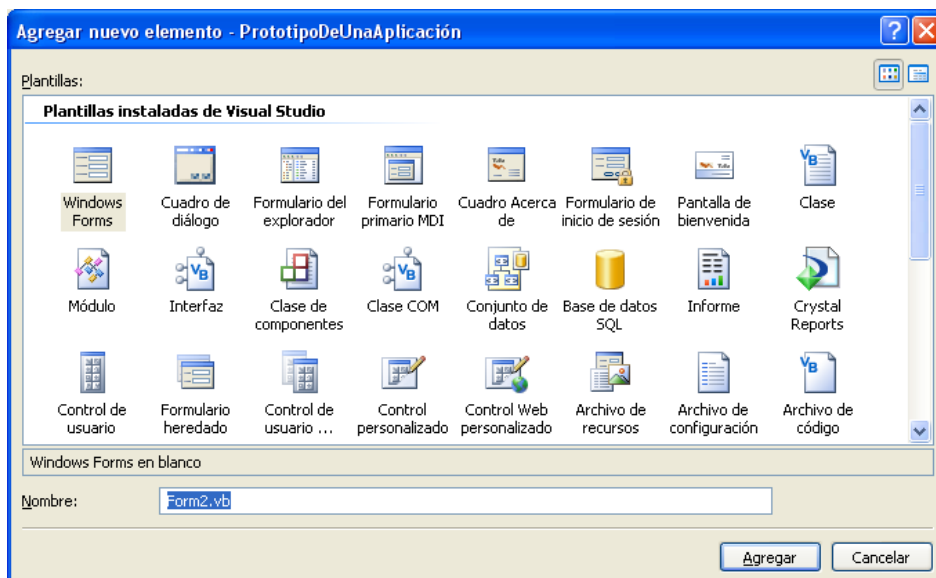
Por cada subitem (columna) que se quiera añadir se pulsa el botón Agregar y se teclea el nombre que se desea que aparezca en la celda.

## 5.2. Aplicaciones con varios formularios

Aunque la plantilla de una aplicación Windows estándar sólo tiene un formulario, la mayoría de las aplicaciones requerirán de varios formularios.

### Añadir nuevos formularios en un proyecto

En el menú "Proyecto", seleccionar la opción "Agregar Windows Forms". En el cuadro de diálogo Agregar nuevo elemento, seleccionar "Windows Forms". El nombre del nuevo formulario será Formxx.vb, aunque desde aquí se puede cambiar en el cuadro de texto de la parte inferior del cuadro de diálogo.

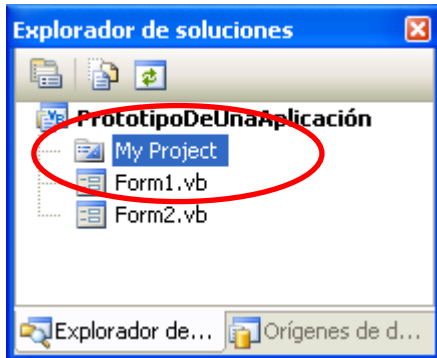




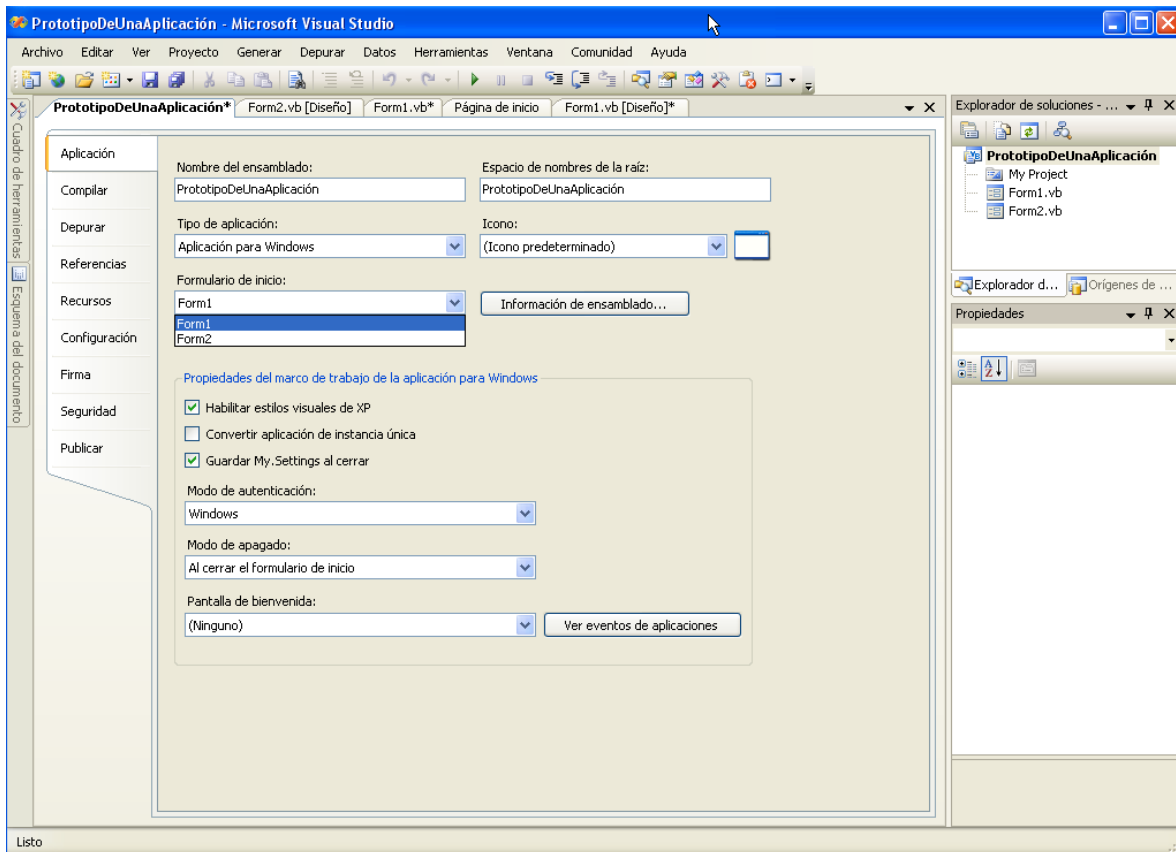
La opción “Windows Forms” inserta un nuevo formulario vacío, pero desde aquí también se pueden añadir otras plantillas prediseñadas de formularios y cuadros de diálogo estándar de Windows (exploradores, ventanas del tipo “Acerca de...”, inicio de sesión, pantalla de entrada a la aplicación, etc.).

## Formulario de arranque

Por omisión, el formulario de arranque es el primer formulario que se crea (Form1). Esto se puede cambiar en la ventana de propiedades del proyecto. A dicha ventana se puede acceder mediante la opción “Propiedades” del menú “Proyecto” o accediendo al elemento “My Project” desde la ventana del Explorador de soluciones.



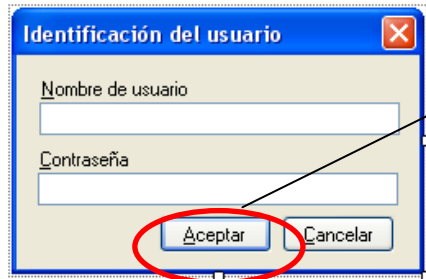
Aparece entonces la ventana de propiedades del proyecto. Dentro de la pestaña “Aplicación”, aparece una pantalla con información del proyecto. Ahí aparece un cuadro de diálogo objeto inicial en el que se muestran todos los elementos del proyecto y se debe seleccionar el formulario que se desea que aparezca cuando arranca la aplicación.



## 5.3. Simular la interacción con Visual Studio 2005

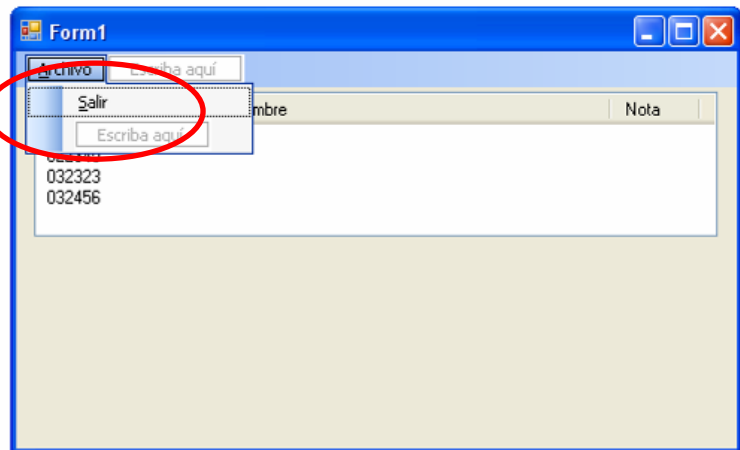
### Simular los clics de un control

Para acceder al código asociado a un control se debe pulsar dos veces sobre el control y aparecerá la ventana de código con el evento asociado al control.

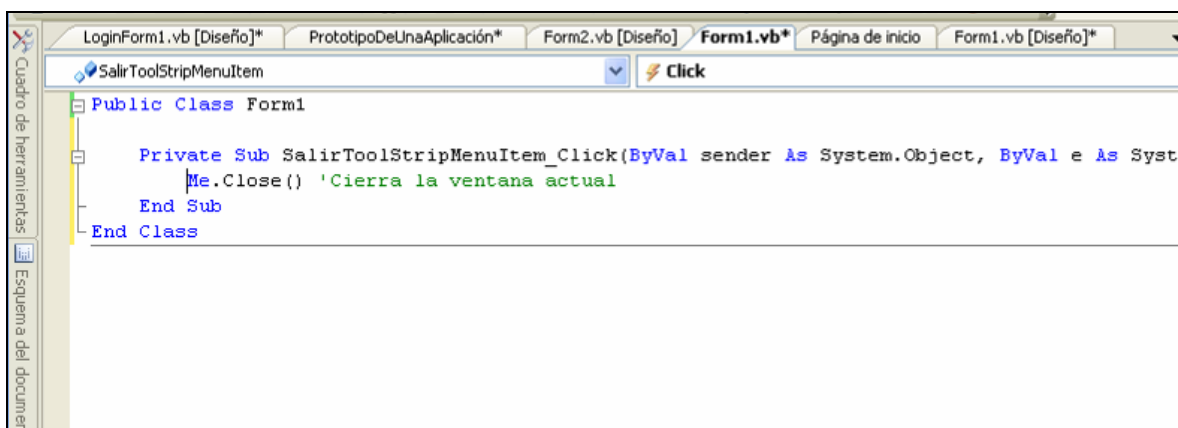


Hacer doble clic para sacar el código asociado al evento Click del botón Aceptar

Hacer doble clic para sacar el código asociado a la opción de menú Salir



Aparece entonces la ventana de código donde se puede introducir el código para simular el funcionamiento del control.



### Abrir una ventana en tiempo de ejecución

Para abrir una ventana nueva hay que realizar dos pasos:

1. Crea una nueva instancia de la clase de la ventana. El formulario debe estar creado previamente.

```
Dim frm As New Form4
```

Se declara una nueva variable (`frm`) y a ella se asigna una nueva instancia (`New`) de la clase `Form4`.

2. Se ejecuta el método `Show()` de la clase `Form` sobre la instancia creada anteriormente.

```
frm.Show()
```

La versión 2.0 de .NET Framework presenta una forma alternativa de realizar esta operación. La versión de Visual Basic incluida en .NET Framework 2.0 incluye una función `My` que proporciona información de la aplicación, a instancias predeterminadas de los objetos que la componen y a su entorno en tiempo de ejecución.

A diferencia de versiones anteriores, a través de la función `My` podemos acceder a las instancias de la colección de formularios que componen la aplicación mediante el objeto `My.Forms`, de forma que, por ejemplo, `My.Forms.Form1`, hará referencia a la instancia predeterminada de la clase `Form1`.

Por lo tanto, se podrá abrir una instancia de `Form1` desde otro formulario simplemente mediante la siguiente orden:

```
My.Forms.Form1.Show()
```

### Cerrar una ventana en tiempo de ejecución

El método `Close()` cierra una ventana, por lo que

```
Me.Close()
```

`Me` hace referencia a la instancia actual (el formulario donde se ha escrito); `Me.Close()` cierra la instancia (la ventana) actual.

Si se trata del único formulario que queda abierto en la aplicación, `Me.Close()` cerrará la aplicación.