Curso de Programación .NET nivel I

Unidad 6

“Interfaz Gráfica”

Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

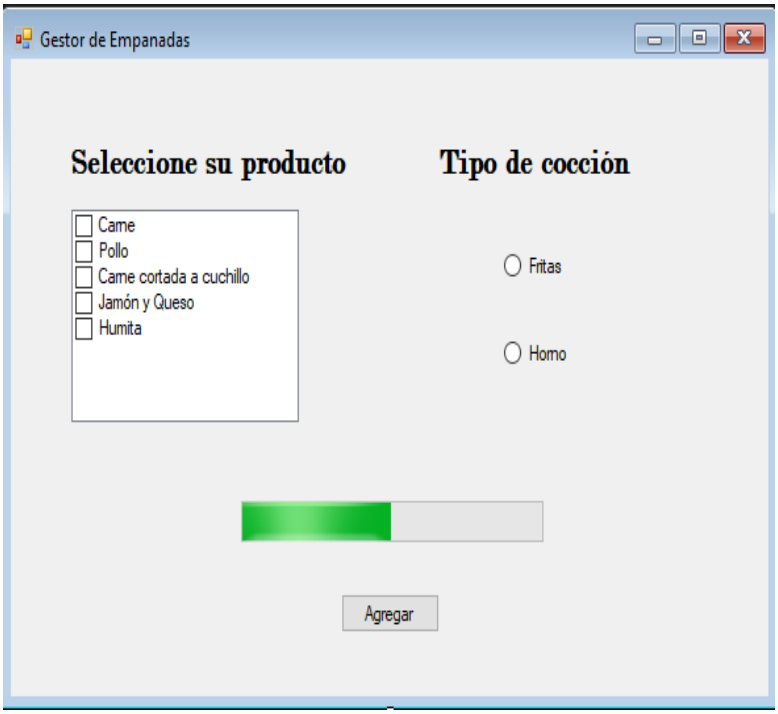
Índice

Contenido

Introducción

Bienvenidos a la unidad donde trataremos una de las tareas más complejas, divertidas y claves en todo sistema: La interfaz de usuario.

Hasta ahora la única interfaz de usuario que conocemos es la de consola, la cual puede ser deseable en miles de escenarios como ser de automatizaciones o procesos que corren en el “background”.
La realidad es que hoy en día una solución también debe proveer una interfaz de usuario visual amigable con botones, paneles y demás componentes; También debe ser simple e intuitiva para hacer que nuestros usuarios tengan una experiencia agradable.



Conceptos básicos de Winforms

Explicación

Una aplicación de Windows forms corre en una computadora de escritorio tal como lo hacen las aplicaciones consola. Este tipo de aplicación utiliza una ventana o Form que es la base de la visualización, los cuales a su vez contienen una colección de **componentes visuales** como etiquetas (labels), cajas de texto (textbox), lista de texto (listbox) etc. De ahora en adelante me voy a referir a ellos por su nombres en inglés ya que es la manera que lo van a encontrar en el sistema.

Estos componentes visuales se denominan **control;** Un Control es una clase con representación visual, es decir una clase ya implementada con métodos y propiedades que me permiten configurar y accionar sobre la interfaz de usuario.

Abajo vemos un ejemplo de una aplicación WinForm simple. Se trata de un selector de producto para agregar a una lista. El usuario va a seleccionar dos valores: el producto y el tipo de cocción para luego hacer click en Agregar para proceder.

Podemos ver la cantidad de controles que intervienen en este proceso:

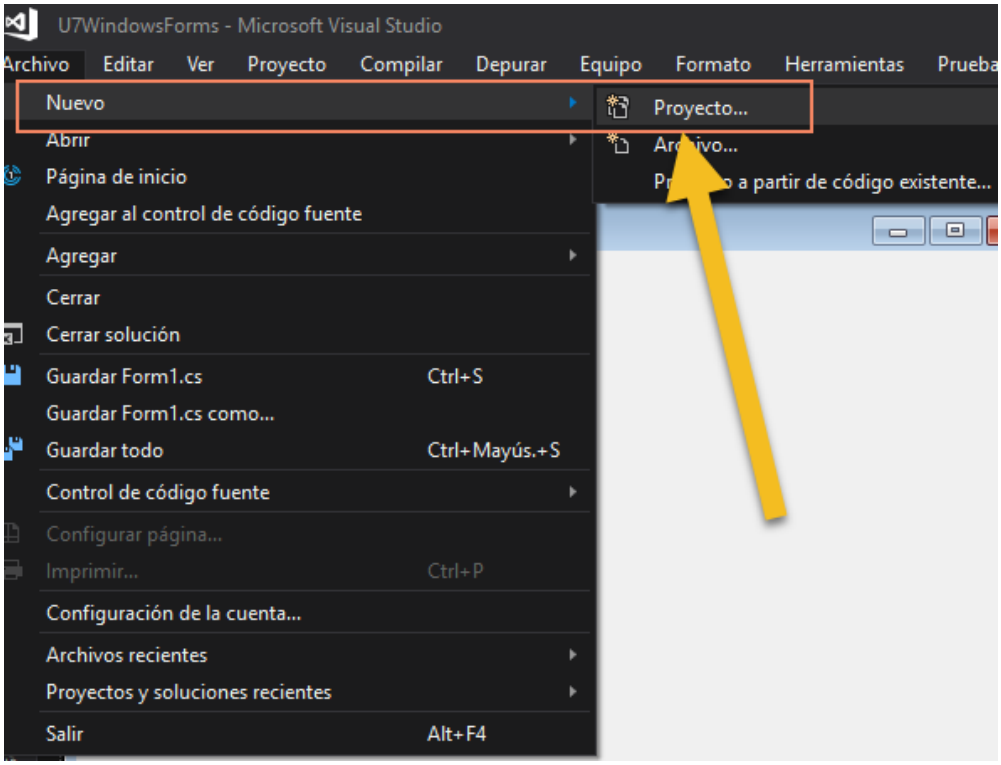
- un **Form** con el titulo “Gestor de empanadas” y dentro de el:
  - **Labels**: para guiar al usuario en el uso del sistemas
  - **CheckedListBox** y **RadioButton**: para seleccionar los productos
  - **ProgressBar**: para avisar a mi usuario el tiempo que va a tardar en procesar su pedido
  - **Button**: Por último un clásico en el diseño de UI, un botón que normalmente va a tener código al presionarse. Por ejemplo que obtenga los datos de los demás controles, los convierta y los guarde en un archivo de texto.

Implementando WinForms

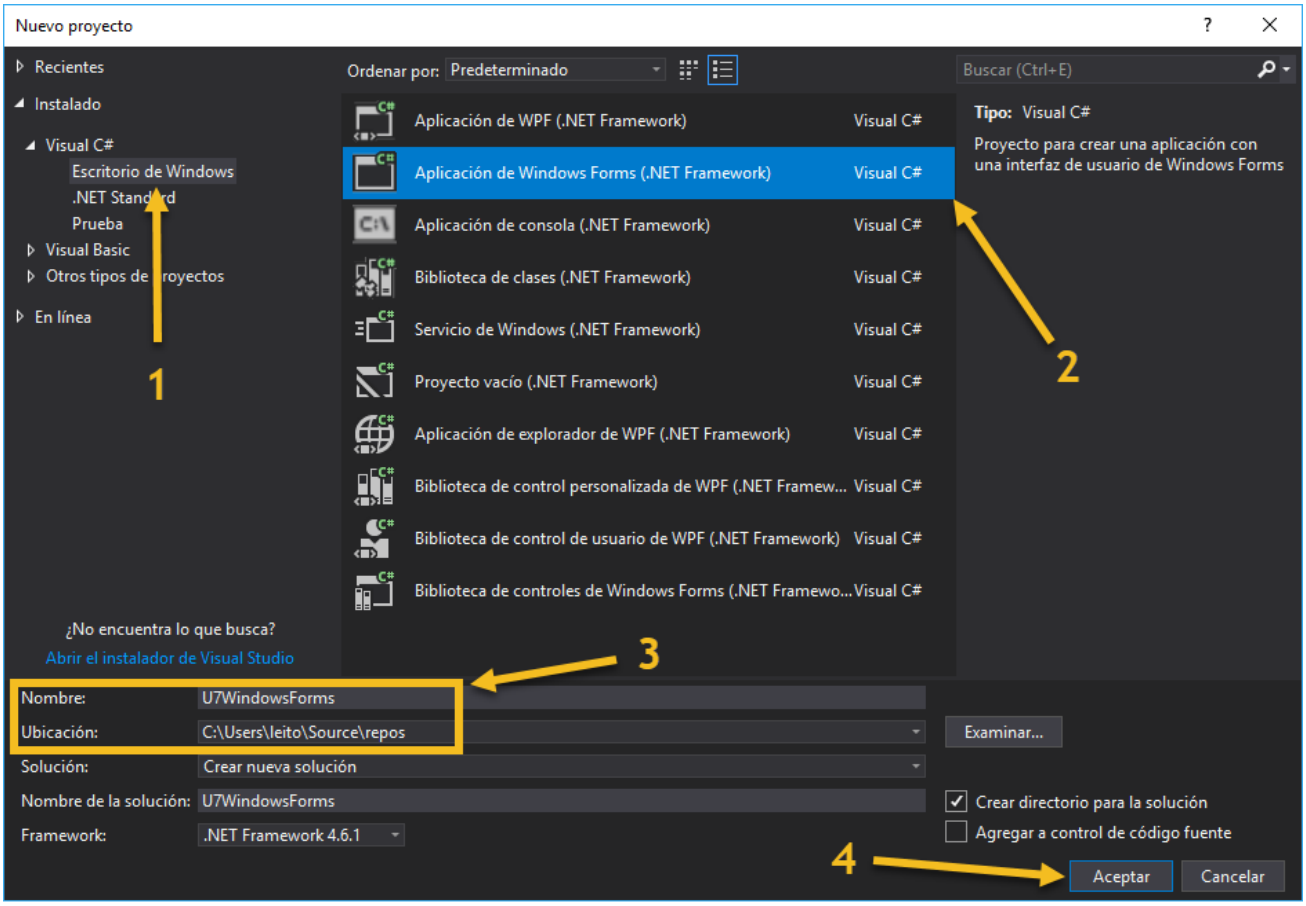
Creación de un proyecto WinForms

Vamos a ver como creamos una aplicación Winforms y configuramos Visual Studio para poder empezar a trabajar con esta solución

Paso 1: El primer paso consiste en la creación de un nuevo proyecto en Visual Studio, tal como veníamos haciendo. Para eso nos vamos al menú Archivo → Nuevo → Proyecto...

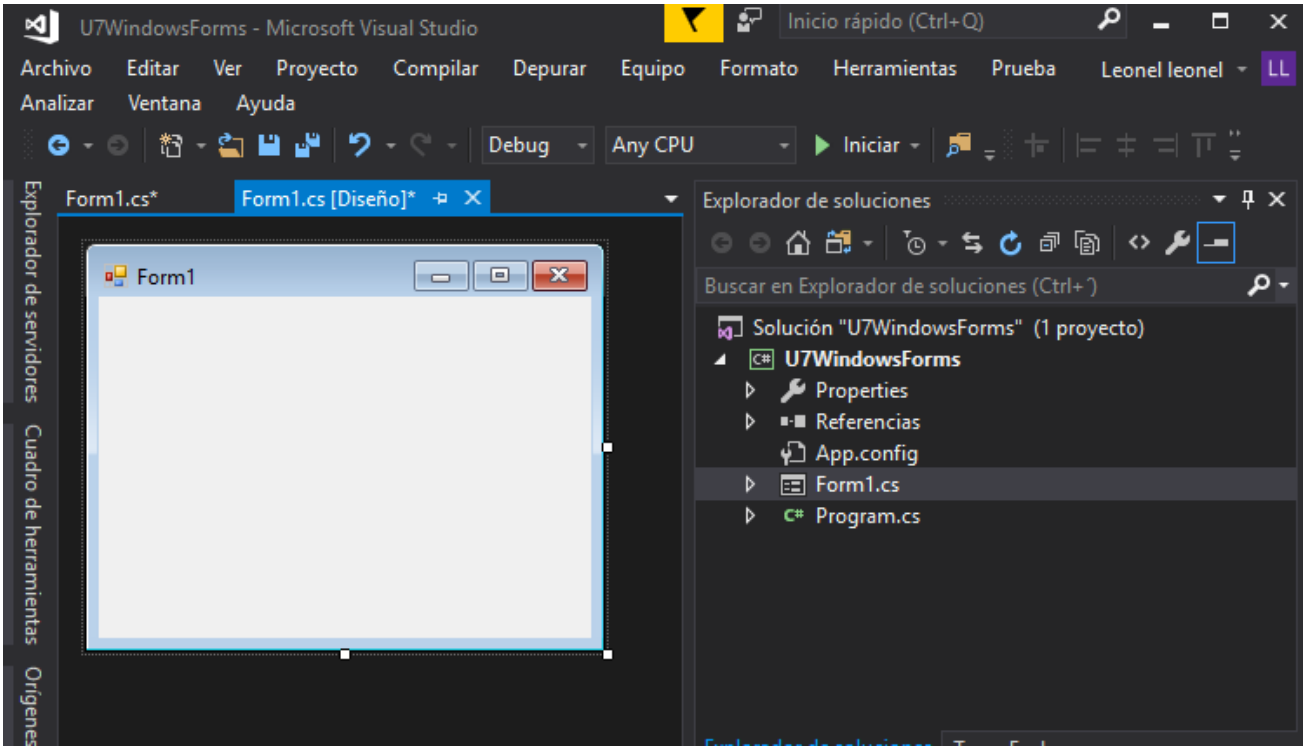


Paso 2: El segundo paso es el de elegir el tipo de proyecto adecuado, el cual es Aplicación de Windows Form



1- En el la ventana del lado izquierdo hay diferentes opciones de categorías de proyectos para Visual Studio, en nuestro caso vamos a clickear en “Escritorio de Windows”
2- Cuando hacemos el click anterior el panel central se va a actualizar, en él vamos a encontrar varios opciones de tipo de proyecto, seleccionamos con un click “Aplicacion de Windows Forms (.NET Framework)”
3- Le tenemos que dar un nombre a nuestra aplicación y también una carpeta donde queremos guardarlo.
4- En este último paso hacemos click en Aceptar para dejar que Visual Studio cree el proyecto.

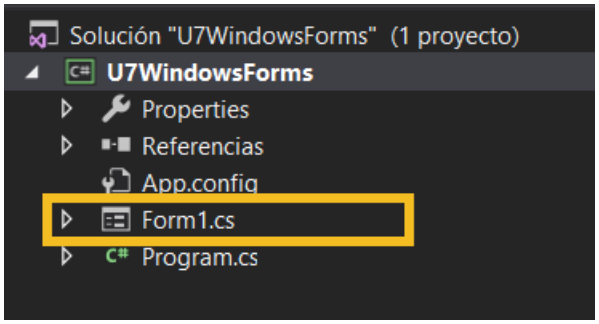
Si seguimos bien estos últimos pasos vamos a ver lo siguiente en Visual Studio:



Vamos a ver un diseñador de Form. Es en este diseñador que vamos a empezar a crear la Aplicación Visual de .NET

Diseñando el Form de Windows

Lo que vamos a tener de nuevo en un proyecto WinForm desde cero es el componente form en Visual Studio.

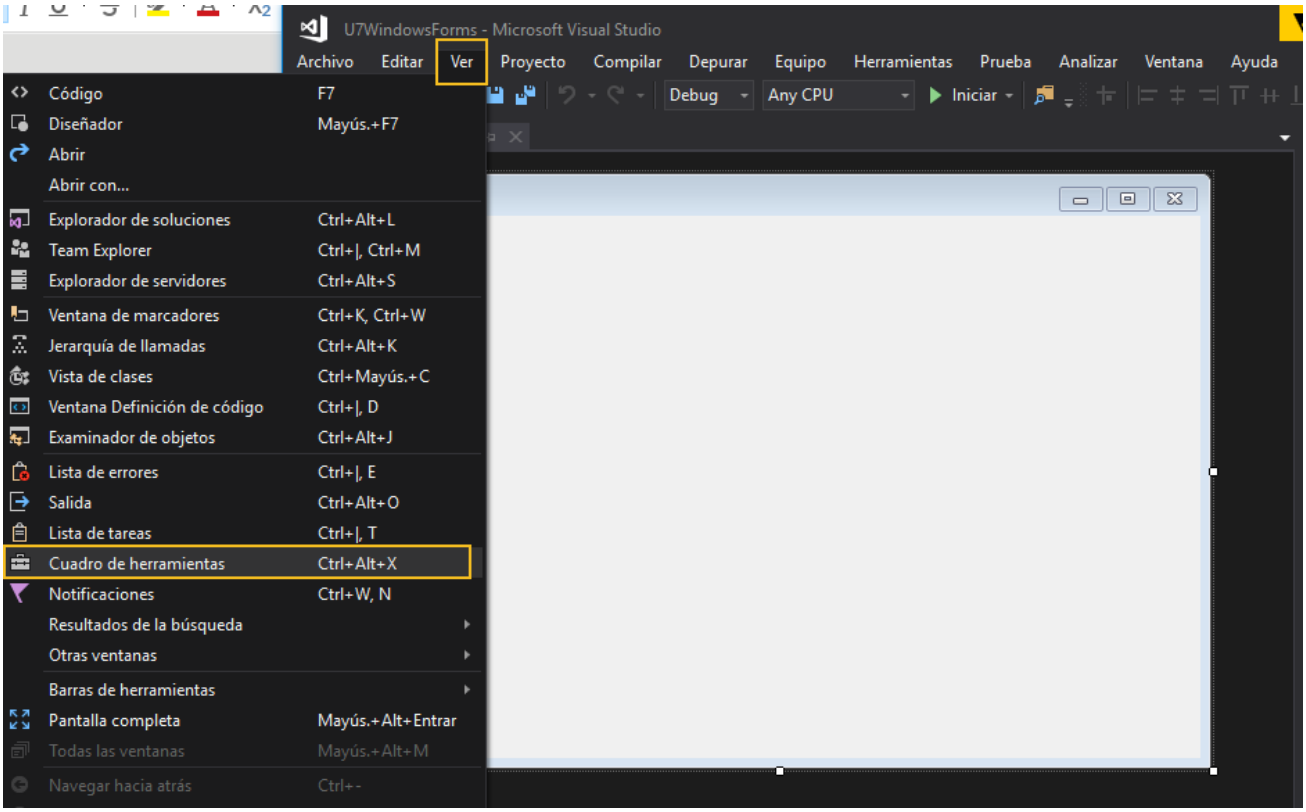


- **Componentes**
  - **Form1.cs**: va a contener el **código** como cualquier otra clase en .NET aunque incorpora la habilidad de tener controles por lo tanto también va a contar con el **diseño** de la ventana.
  - **Program.cs:** Ahora va a contener el código generado automáticamente para crear y mostrar el Form, el concepto del código inicial en Program.cs como en consola se puede seguir aplicando

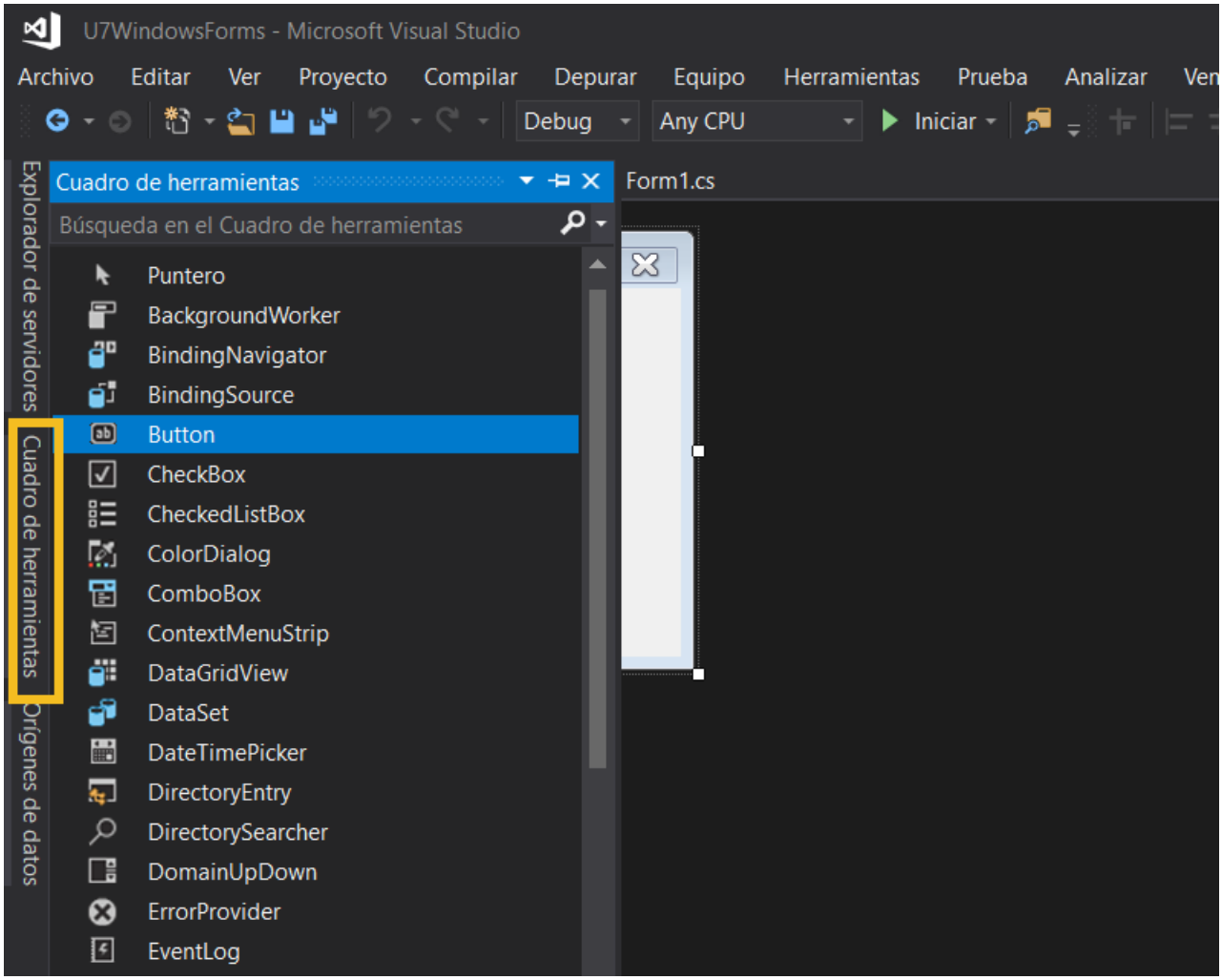
pero ahora tengo también la posibilidad de incluir código inicial en la clase del primer Form que aparezca en pantalla.

Si hacemos click en Form nos va a llevar al diseñador sobre el cual podemos arrastrar y soltar componentes visuales en él. Para comenzar a diseñar nuestro Form tenemos que activar el Toolbox de controles primeros que por defecto (no entiendo por qué) viene desactivado.

Activando el cuadro de herramientas visuales

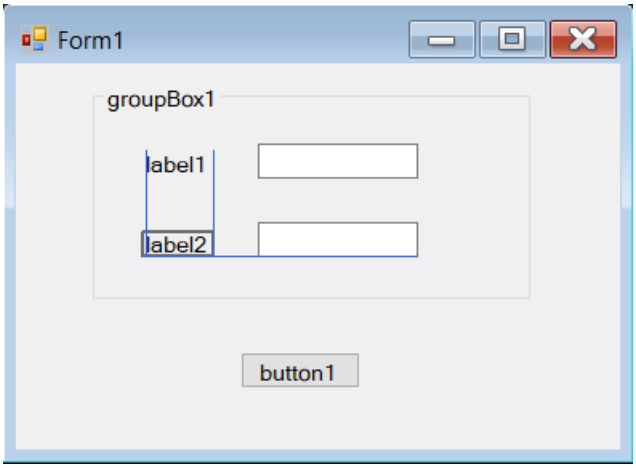


Una vez hecho esto vamos a ir bien a la izquierda de nuestra pantalla y vamos a hacer click en Cuadro de herramientas



Este cuadro de herramientas contiene los componentes visuales o los controles que podemos ir utilizando en nuestro Form para diseñar de manera visual (arrastrar y soltar) nuestra interfaz de usuario.

Pueden jugar un tirando componentes en el Form e ir armando diseños visuales de aplicaciones



Algo la verdad muy útil que ha sido introducido hace varias versiones de Visual Studio es la capacidad de alineación que tienen los componentes como vemos en la imagen anterior, que me permiten sin mucho esfuerzo alienar y centrar los controles en el Form.

Los controles Visuales

Introducción

Ya que hemos visto la mecánica de diseño en .NET veamos ahora como configurar mas propiedades de los controles además de las visuales (tamaño y posición) como llamar a sus métodos o como definir eventos; Un concepto nuevo del que hacen uso los controles. Como verán todo sigue el paradigma de objetos, propiedades y métodos, la base de todo.

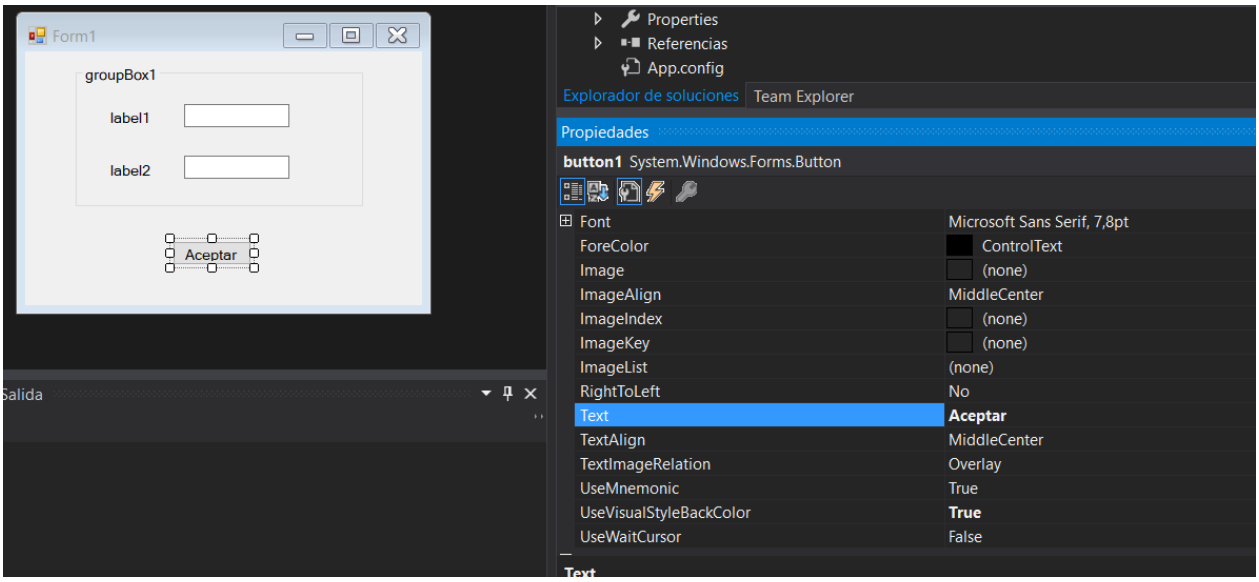
Los controles son objetos

Los controles son objetos que son creados a partir de clases. El selector de controles que vimos más arriba me presenta varias clases para seleccionar, cuando arrastro una de ellas hacia el diseñador estoy creando un nuevo objeto. En los controles visuales tenemos esta nueva manera de construir objetos obviamente todo esto es también realizable de la manera clásica de instanciación por código. Button btn = new Button(…).

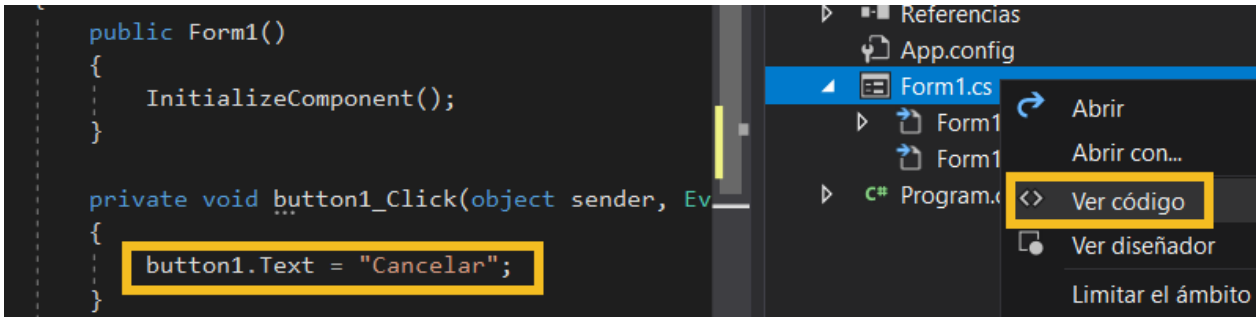
Accediendo a los controles

Vamos a ver como accedemos a los controles y para hacer esto tenemos dos maneras

- **Visualmente:** Por medio del gestor de propiedades el cual es un editor de los valores del objeto, en el cual puedo editarlos como en el caso del dibujo el Text del botón.



- **Por código:** Usando el nombre del objeto (button1 en mi caso) puedo acceder a la variable que contiene el objeto y operar en sus propiedades o métodos



Propiedades

Este concepto es lo mismo que ya hemos visto en las unidades de objetos, las propiedades son las características de los objetos; en el caso de los controles tenemos propiedades de color, posición, texto entre muchas otras que puedo definir según mis necesidades.

En la imagen anterior hay ejemplos de como acceder a las propiedades del objeto button en el cual cambiamos el texto del mismo.

Métodos

Así como los controles tienen propiedades también tienen métodos ya que son objetos. Una ventana puede cerrarse o un botón puede esconderse.

Para llamarlo tal como hicimos con las propiedades lo hacemos con métodos.

```
textBox1.Copy();
```

En el ejemplo de arriba llamo a la función de copiado de TextBox, lo que hace es copiar el contenido de un cuadro de texto al Portapapeles.

Eventos

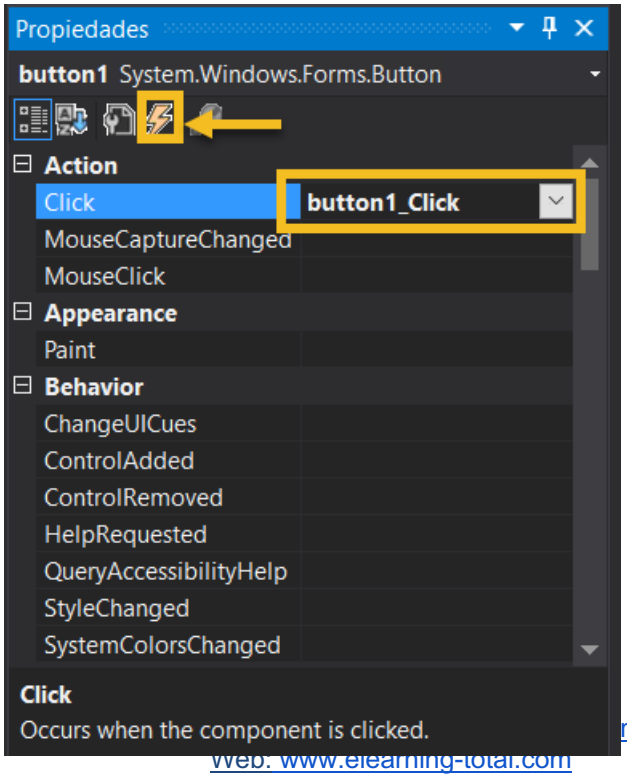
En lo referente a controles los métodos son utilizados pero donde generalmente nos centramos más es en los eventos.

Un evento es un mensaje enviado por un objeto para señalizar la ocurrencia de una acción. La acción puede ser una interacción por parte del usuario como lo es un Click de un botón o escribir en una caja de texto o también puede suceder por otra lógica de otro programa como cambiar el valor de una propiedad. El objeto que eleva el evento lo llamamos el “event sender”. El event sender desconoce que objeto o método va a recibir el evento el levanta.

Los controles vienen con una lista de eventos los cuales se pueden registrar, este proceso de registro de evento consta de dos partes:

- Selección de que tipo de Evento registrar. Por ejemplo: Click de un botón.
- Implementar código a ejecutarse cuando ese evento ocurra. Por ejemplo: Escribir un archivo de texto.

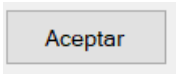
Para gestionar los eventos tenemos un apartado de eventos en el registro de objetos:



Para acceder a este panel, tenemos que hacer click en el rayo como muestra la imagen.
Para registrar un evento simplemente alcanza con hacer doble click en la celda derecha del evento que estemos interesados. Una vez registrado el evento el método para atenderlo nos va a aparecer en el código del form:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Boton Clickeado");
}
```

Ahora cuando el usuario haga click en el “button1” ese código va a ejecutarse.



Control Button

Explicación

El control Button es utilizado para generar una acción luego de ser clickeado.

Uso en la práctica

Generalmente los botones están presentes en todos los programas informáticos para ser utilizados por el usuario.

Los usos son muchos y entre los más comunes están:

- Botón para confirmar los datos de un formulario o restablecerlo.
- Agregar o quitar items de una lista.
- Guardar o cargar archivos.

Control Label

Explicación

El control label es usado para mostrar un texto o un mensaje al usuario dentro del Form. Este control es usado junto a otros controles.

Uso en la práctica

Un uso clásico es el de usar un label al lado de un textbox. El label entonces ayuda al usuario a informar que se espera llenar en un cuadro de texto.

Control Textbox

Explicación

Un control Textbox es un cuadro de texto usado para permitir al usuario ingresar texto para la aplicación del Form.

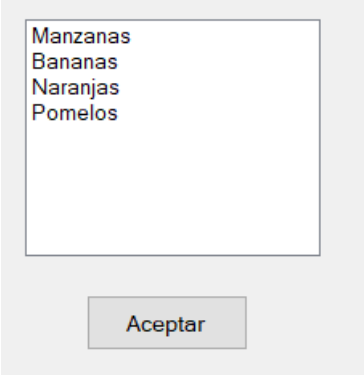
Uso en la práctica

Un uso muy común es el de un formulario para ingresar nuevos registros al sistema. Estas cajas de texto pueden tener distintos formatos por ejemplo en el caso que se quiera ingresar un password.
El ingreso de usuario y password es otro uso clásico de este control.

Control Listbox

Explicación

El control Listbox es una manera de presentar ítems para el usuario en formato de lista.



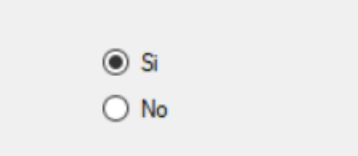
Uso en la práctica

Un uso natural es el de representar colecciones como listas o arrays de manera visual.

Control RadioButton

Explicación

Un Control RadioButton es usado para mostrar una lista de ítems exactamente como lo hace el ListBox con la



diferencia que podemos elegir uno para usarlo posteriormente

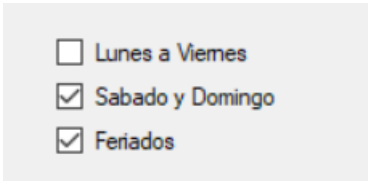
Uso en la práctica

Este es un componente visual para permitirle a nuestros usuarios tomar una decisión entre varias opciones que son mutuamente excluyentes. Por ejemplo para un formulario de venta de productos al momento de elegir el tipo de envió el cual puede ser prioritario o común, en este caso podemos usar un RadioButton con estas dos opciones.

Control CheckBox

Explicación

Un control de tipo CheckBox me permite mostrar una lista de ítems de manera similar al RadioButton, la única diferencia con este es la cantidad de ítems que puedo seleccionar. En el RadioButton solo puedo seleccionar uno; En el CheckBox puedo seleccionar de 0 a el total de ítems.



Uso en la práctica

El uso es similar al RadioButton, es decir que permite mostrar opciones para la toma de decisiones, con la diferencia que los ítems no son mutuamente excluyentes.