TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

Programación III

Guía teórica práctica 1. Parte 1.

Autora: Tamara Gisele Herrera

UTN|FRGP

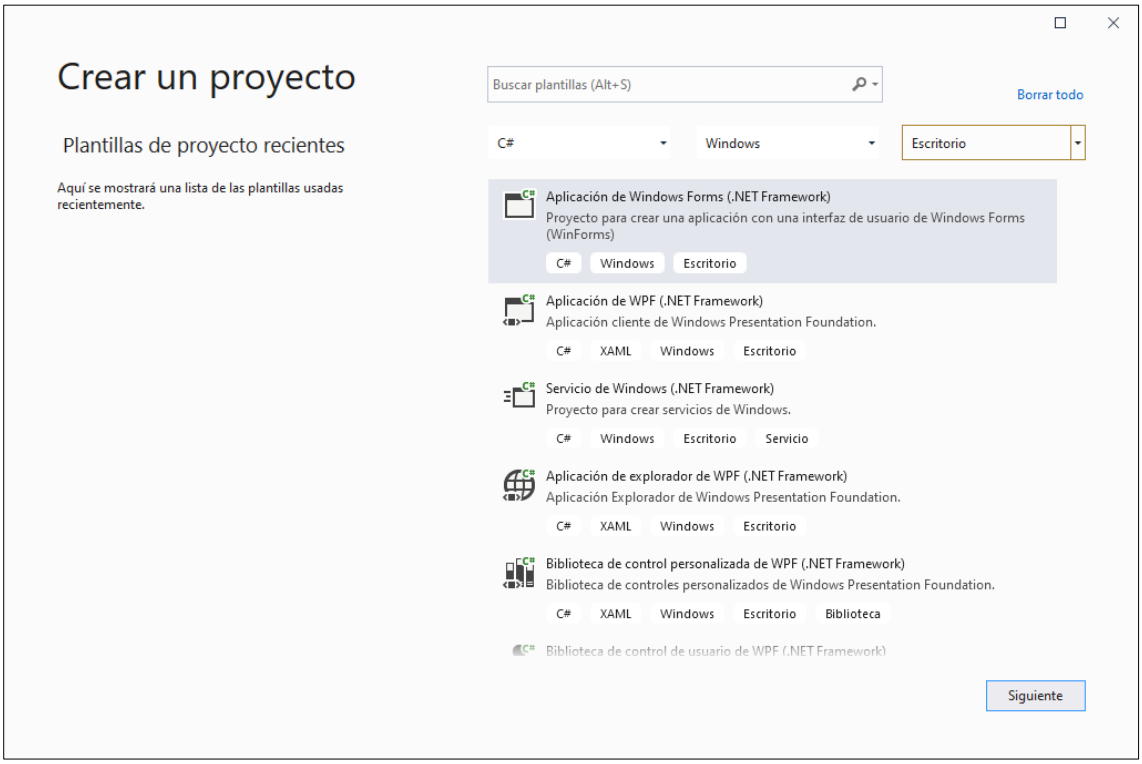
Contenido

Aplicación Windows

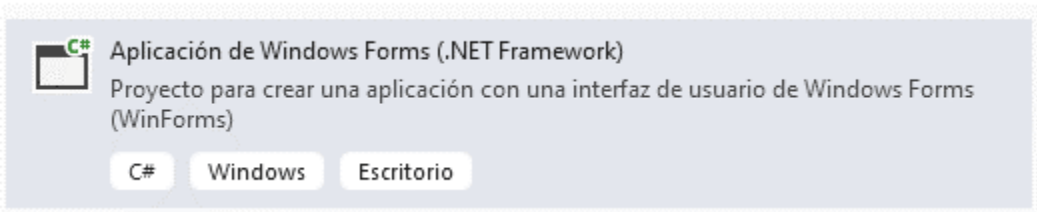
Crear un nuevo proyecto

1. Vamos Archivo → Nuevo → Proyecto

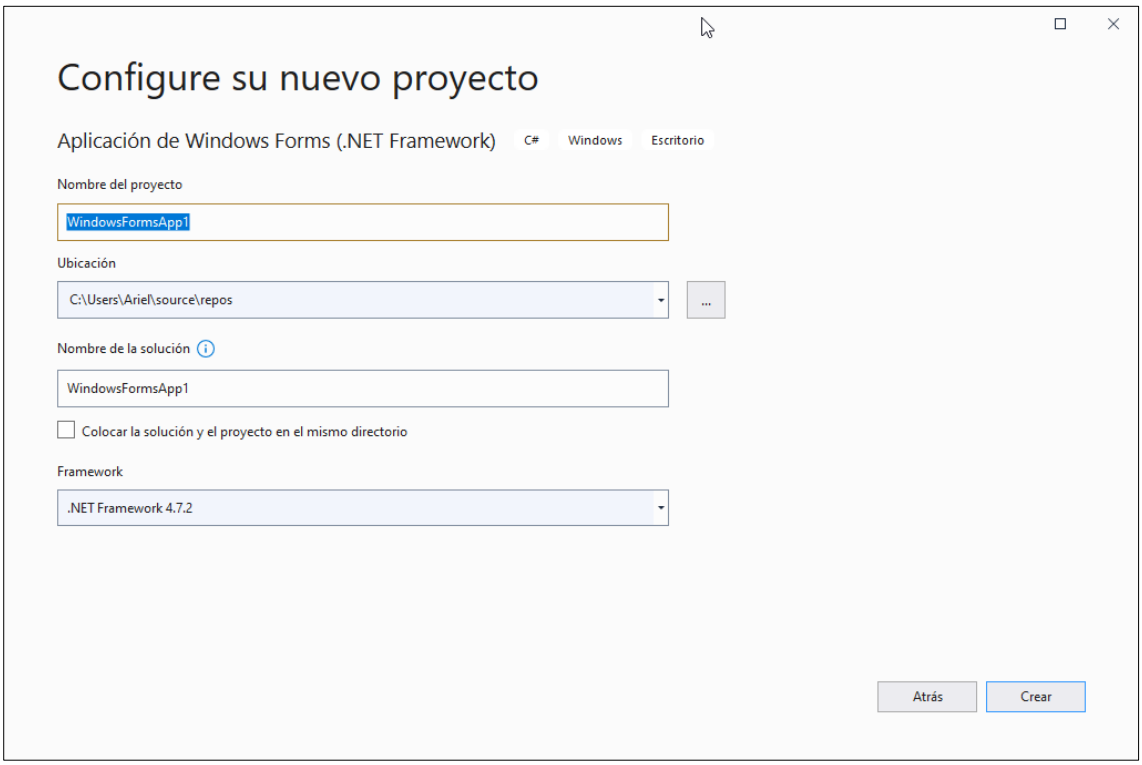
Esto abrirá un cuadro de dialogo para seleccionar el tipo de proyecto nuevo.



2. En lenguaje seleccionamos **C#**, en plataforma seleccionamos **Windows** y en tipo de proyecto seleccionamos **Escritorio**. Aparecerán las plantillas disponibles, seleccionamos **Aplicación de Windows Forms (.NET Framework)**



3. Damos clic en siguiente. En la nueva ventana colocamos un nombre del proyecto, seleccionamos la ubicación, el nombre de la solución (una solución es un contenedor de proyectos) y en Framework dejamos el que viene por defecto.



4. Por último pulsamos crear y se creará el proyecto.

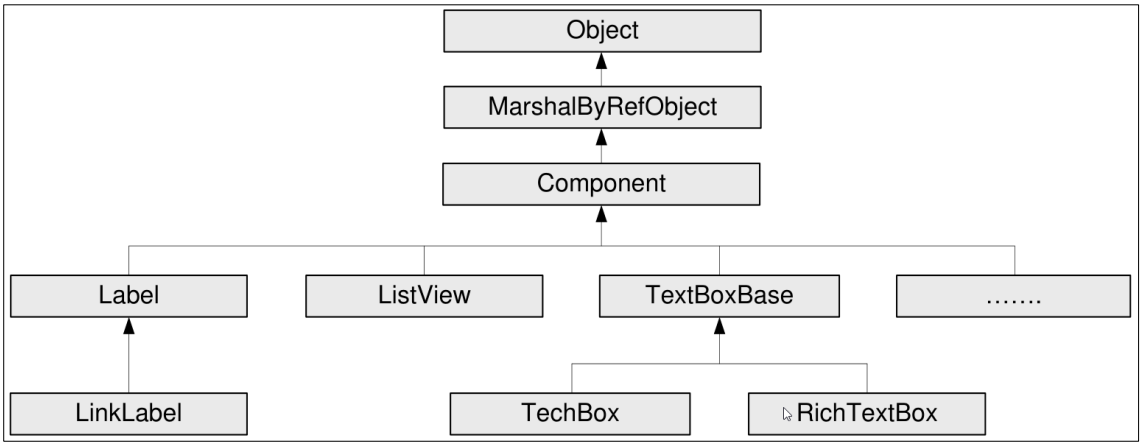
Aclaración:

- Al seleccionar un proyecto desde una plantilla se agregan archivos, código y se realiza una configuración inicial del proyecto.
- Podes seleccionar la versión del .NET Framework con la cual trabajar. Recomiendo desde 4.5 en adelante.
- La opción colocar la solución y el proyecto al mismo directorio crea directorio (carpeta) donde coloca los archivos del proyecto y la solución (de extensión .sln)

Controles

Cuando se trabaja con **Aplicaciones Windows Forms**, se está trabajando con el espacio de nombre (namespace) **System.Windows.Forms**, este espacio de nombre está incluido en los archivos Form de nuestra aplicación con la sentencia using.

Muchos de los controles que se utilizaran derivan de la clase **System.Windows.Forms.Control**, en esta clase se definen la funcionalidad básica de los con- troles por eso algunas de las propiedades y eventos son iguales. Y algunas de estas clases son la base para otros controles.



Propiedades

Todos los controles tienen una cantidad de propiedades que sirven para manejar el comportamiento de los controles. La clase base para la mayoría de los controles es **System.Windows.Forms.Control**.

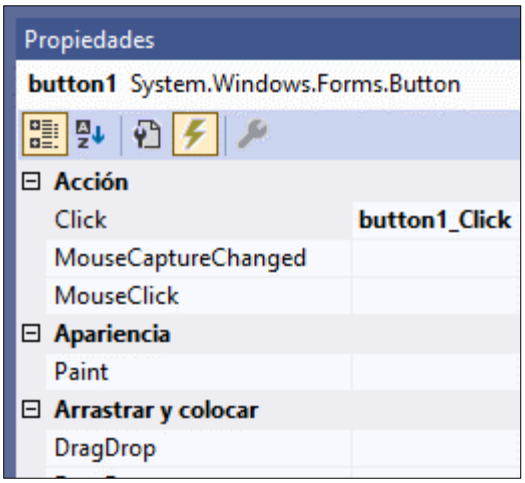
Las propiedades más comunes de estos controles son:

NOMBRE	DISPONIBILIDAD	DESCRIPCIÓN
Anchor	Lectura/Escritura	Especifica cómo se comporta el control cuando el recipiente cambia de tamaño.
BackColor	Lectura/Escritura	Color de fondo del control
Bottom	Lectura/Escritura	Se especifica la distancia desde la parte superior de la ventana a la parte inferior del control.
Dock	Lectura/Escritura	Le permite hacer un acoplamiento de control a los bordes de una ventana.
Enabled	Lectura/Escritura	Habilita o Deshabilita el Control.
ForeColor	Lectura/Escritura	EI color de primer plano del control.
Height	Lectura/Escritura	La altura del control.
Left	Lectura/Escritura	La distancia del borde izq. del control al Borde izq. de la ventana.
Name	Lectura/Escritura	El nombre del control.
Parent	Lectura/Escritura	El padre del control.
Right	Lectura/Escritura	Distancia del borde derecho del control al borde izq. De la ventana.
Tablndex	Lectura/Escritura	EI número de orden del control en el Form (Recipiente).
TabStop	Lectura/Escritura	Especifica si el control puede ser accedido mediante la tecla Tab..
Tag	Lectura/Escritura	Este valor no es usado por el control y se utiliza para guardar información sobre el control en sí mismo. Cuando esta propiedad es cambiada mediante el diseñador de Formularios solo se le puede asignar un valor de tipo string.
Top	Lectura/Escritura	Distancia del borde sup. del control con respecto al borde sup. de la ventana.
Visible	Lectura/Escritura	Si es visible o no.
Width	Lectura/Escritura	Largo del control.

Eventos

Los eventos son generados por los controles, generalmente esto pasa cuando el usuario realiza alguna acción sobre el control, por ej. Un clic sobre un botón. La clase **System.Windows.Forms.Control** define varios eventos comunes para la mayoría de los controles.

Para acceder a los eventos existen varias formas, una de ellas es hacer doble clic sobre el control, lo cual lo llevara al evento predeterminado del control. Otra forma es en la ventana de propiedades hacer clic en el icono de eventos y luego seleccionar el evento deseado.



Otra forma es directamente desde la ventana del código del formulario.

La lista de algunos de los eventos comunes es:

NOMBRE	DESCRIPCIÓN
Click	Ocurre cuando se hace click en un control. En algunos casos este evento curre cuando el usuario aprieta Enter
DoubleClick	Ocurre cuando se hace doble click en un control. Cuando se maneja el evento Click en algunos controles como el botón entonces el evento DoubleClick no llega ser llamado.
DragDrop	Ocurre cuando se completa una operación de drag y drop, es decir, cuando un objeto es arrastrado y luego el usuario libera el botón del mouse
DragEnter	Ocurre cuando un objeto es arrastrado y se ingresa en los límites de otro control.
DragLeave	Ocurre cuando un objeto es arrastrado y se sale de los límites de otro control.
DragOver	Ocurre cuando un objeto fue arrastrado por encima de otro control.
KeyDown	Ocurre cuando se presiona una tecla mientras el control tiene el foco. Este evento ocurre siempre antes de los eventos KeyPress y KeyUp.
KeyPress	Ocurre cuando se presiona una tecla mientras el control tiene el foco. Este evento ocurre siempre después del evento KeyDown y antes del evento KeyUp. KeyDown pasa el código de la tecla que fue presionada mientras que KeyPress el valor de carácter de la tecla que fue presionada.
KeyUp	Ocurre cuando una tecla es soltada mientras un control tiene el foco. Este evento ocurre siempre después de los eventos KeyDown y KeyPress.
GotFocus	Ocurre cuando un control recibe el foco. No utilice este evento para efectuar validaciones. Para ello se utilizan los eventos Validating y Validated.
LostFocus	Ocurre cuando un control pierde el foco. No utilice este evento para efectuar validaciones. Para ello se utilizan los eventos Validating y Validated.
MouseDown	Ocurre cuando el puntero del mouse se ubica sobre un control y un botón del mouse es presionado. Esto no es 10 mismo que el evento Click porque el evento MouseDown ocurre apenas el botón es presionado y antes de que sea liberado.
MouseMove	Ocurre permanentemente cuando el mouse se mueve sobre un control.
MouseUp	Ocurre cuando el puntero del mouse se ubica sobre un control y un botón del mouse es liberado.
Paint	Ocurres cuando dibujamos un control.
Validated	Este evento se dispara cuando un control con la propiedad CausesValidation en valor true esta por recibir el foco. Se dispara después de que el evento Validating finaliza e indica que la validación fue completada.
Validating	Este evento se dispara cuando un control con la propiedad CausesValidation en valor true esta por recibir el foco. EI control que esta por ser validado es el control que pierde el foco, no el que lo está recibiendo.

Practica 1 (Form)

1. Creamos un proyecto nuevo con el nombre MiPrimerAplicacion.
2. Analizamos los archivos que nos creó el proyecto.
3. Cambiamos el nombre del form1, con la propiedad Name.
4. Cambiamos el título del formulario, con la propiedad Text. (run)
5. Cambiamos el color de fondo del formulario, con la propiedad BackColor. (run)
6. Cambiamos la posición del formulario, con la propiedad StartPosition. (run)
7. Cambiamos la vista inicial del formulario (normal, minimizado o maximizado), con la propiedad WindowState. (run)
8. Manejar el evento load del formulario.
    a. Dobleclick sobre el formulario carga el método del evento load por defecto.

```
private void Form1_Load(object sender, EventArgs e)
{

}
```

    b. Mostramos un mensaje con MessageBox.Show. (run)

```
private void Formulario1_Load(object sender, EventArgs e)
{
    MessageBox.Show("Bienvenidos a C#");
}
```

9. Manejar el evento FormClosed del formulario.
    a. Mostramos un mensaje con messagebox.show. (run)

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    MessageBox.Show("Chau Chau ...");
}
```

10. No mostrar el botón de minimizar del formulario.
11. No mostrar el botón de maximizar del formulario.
12. Modificar el valor de la propiedad Opacity.

El control Button

El control Button deriva de la clase System.Windows.Forms.ButtonBase que brinda una funcionalidad básica, además el usuario puede usar esta clase para crear sus propios buttons personalizados.

De esta clase básica derivan otros 3 controles, Button, CheckBox, RadioButton.

Normalmente los Buttons se usan para aceptar o cancelar un DialogBox, llamar a otro formulario o aplicación y dar acción a los datos introducidos en un formulario.

Propiedades más comunes:

PROPIEDADES	DESCRIPCIÓN
FlatStyle	Cambia el Estilo del botón. Apariencia Plana o 3D.
Enabled	Habilita o Deshabilita el botón. (True o False)
Image	Carga una imagen.
ImageAlign	Alinea la imagen en el botón.

Los controles Label y LinkLabel

Las etiquetas (Label) son los controles más comunes que se pueden encontrar en una aplicación Windows y básicamente sirven para mostrar texto en los formularios.

En este .NET Framework existen dos controles Label y son.

- Label: etiqueta estándar.
- LinkLabel: igual a la estándar, pero permite un hipervínculo.

Propiedades más comunes:

PROPIEDADES	DESCRIPCIÓN
Borderstyle	Especifica el tipo de borde de la etiqueta. Por defecto sin borde.
Flatstyle	Estilo de la Etiqueta, plana o 3D.
Image	Carga una imagen.
ImageAlign	Alinea la imagen en el botón.
LinkArea	(LinkLabel) Especifica el rango del texto que debe mostrarse como link.
Linkcolor	(LinkLabel) Color del link.
LinkVisited	Cambia el color si el link ya se visitó.
TextAlign	Alinea el texto.
VisitedLinkColor	Color para link visitado.

Practica 2 (Control Button)

1. Agregar un Control Button al Formulario.
2. Analizamos el archivo Form1.Designer.cs
3. Cambiamos el nombre del Control Button, con la propiedad Name.
4. Cambiamos la etiquete del Control Button, con la propiedad Text. (Run)
5. Cambiamos el color del Control Button, con la propiedad BackColor. (Run)
6. Cambiamos el estilo del Control Button, con la propiedad FlatStyle. (Run)
7. Coloco un icono al Control Button, con la propiedad Image.
8. Deshabilitar el Control Button, con la propiedad Enabled.
9. Manejar el evento Click del Control Button.
    a. DobleClick sobre el Control Button carga el método del evento Click por defecto.

```
private void btnBoton_Click(object sender, EventArgs e)
{

}
```

    b. Mostramos un Mensaje con MessageBox.Show. (Run)

```
private void btnBoton_Click(object sender, EventArgs e)
{
    MessageBox.Show("Se disparo el evento Click", "Atención");
}
```

10. Al dispararse el evento Click, se cambie el color del Formulario.

```
private void btnBoton_Click(object sender, EventArgs e)
{
    //MessageBox.Show("Se disparo el evento Click", "Atención");
    this.BackColor = Color.Blue;
}
```

11. Manejar el evento Click del Formulario. Determinar que botón del Mouse se pulsó.

```
private void btnBoton_Click(object sender, EventArgs e)

{
    MouseEventArgs click = (MouseEventArgs)e;
    if (click.Button == MouseButtons.Left)
        MessageBox.Show("Presiono el botón Izquierdo", "Atención");
    else if (click.Button == MouseButtons.Right)
        MessageBox.Show("Presiono el Botón Derecho", "Atención");
    else if (click.Button == MouseButtons.Middle)
        MessageBox.Show("Presiono el botón del Medio", "Atención");
}
```

Practica 3 (control Label y Linklabel)

1. Agregar un Label al formulario.
2. Analizamos el archivo Form1.designer.cs
3. Cambiamos el nombre del Label, con la propiedad Name.
4. Cambiamos el texto del Label, con la propiedad Text.
5. Desplegamos el cuadro de dialogo de la propiedad Font y cambiamos tipo letra, tamaño y estilo.
6. Cambiamos el color del texto, desplegando la paleta de colores personalizados de la propiedad ForeColor.
7. Aplicamos un Borde 3D con la propiedad BorderStyle.
8. Manejar el evento MouseMove del control Label.
    a. En la ventana de eventos elijo MouseMove.

```
private void lblEtiqueta_MouseMove(object sender, MouseEventArgs e)
{


}
```

    b. Cambiamos el color de la propiedad BackColor. (Run)

```
private void lblEtiqueta_MouseMove(object sender, MouseEventArgs e)
{
    lblEtiqueta.BackColor = Color.Cyan;
}
```

9. Al dispararse el evento MouseLeave, se restablezca el color de fondo de la etiqueta.

```
private void lblEtiqueta_MouseLeave(object sender, EventArgs e)
{
    lblEtiqueta.BackColor = System.Drawing.SystemColors.Control;
}
```

10. Cambiar el estilo del cursor al dispararse los eventos MouseMove y MouseLeave.

```
private void lblEtiqueta_MouseMove(object sender, MouseEventArgs e)
{
```

```
    lblEtiqueta.BackColor = Color.Cyan;

    lblEtiqueta.Cursor = Cursors.Hand;
}

private void lblEtiqueta_MouseLeave(object sender, EventArgs e)
{
    lblEtiqueta.BackColor = System.Drawing.SystemColors.Control;
    lblEtiqueta.Cursor = Cursors.Arrow;
}
```

El control TextBox

Los cuadros de textos se utilizan siempre que deseemos que el usuario introduzca algún tipo de datos, en ellos se pueden introducir cualquier tipo de caracteres.

El control TextBox deriva de la clase System.Windows.Forms.TextBoxBase, esta clase brinda la funcionalidad básica para la manipulación de texto, como cortar, pegar y también brinda todos los eventos básicos.

De la clase TextBoxBase también deriva el control RichTextBox.

Propiedades más comunes:

PROPIEDADES	DESCRIPCIÓN
CausesValidation	Estando en true se disparan dos eventos (Validating y Validated) al recibir el foco.
CharacterCasing	Convierte el texto ingresado a minúscula, mayúscula o normal.
MaxLength	Cantidad máxima de caracteres que acepta.
Multiline	True o False, para aceptar multilíneas de texto.
PasswordChar	Especifica si el carácter de la contraseña debe sustituir a los caracteres ingresados.
ReadOnly	Especifica si es de solo lectura.
ScrollBars	En caso de ser multilínea si se muestra el Scroll.

Eventos más comunes:

EVENTOS	DESCRIPCIÓN
Enter, Leave, Validating, Validated	Estos cuatro eventos se ejecutan en este orden, son conocidos como eventos Foco. Cada vez que un control recibe el foco se disparan estos eventos, salvo Validating y Validated cuya propiedad CauseValidation del control que recibe el foco debe estar en true.
KeyDown, KeyPress, KeyUp	Estos eventos permiten controlar lo que se introduce en los controles. KeyDown y KeyUp, reciben el código de la tecla que se pulsa. KeyPress recibe el carácter de la tecla que se pulsa.
TextChanged	se dispara cada vez que hay un cambio en el TextBox.

Practica 4 (control TextBox)

1. Agregar un control TextBox al formulario.
2. Analizamos el archivo form1.designer.cs .
3. Cambiamos el nombre del control TextBox, con la propiedad Name.
4. Cambiamos la cantidad de caracteres que acepta el control TextBox, con la propiedad MaxLength. (run)
5. Cambiamos la propiedad CharacterCasing del control TextBox, para que cambie a mayúsculas los caracteres que se ingresan. (run)
6. En el evento Click del botón creado anteriormente, cancelamos las líneas de código anteriores y colocamos el código para cambiar el color de fondo del TextBox (propiedad BackColor) si el TextBox se encuentra vacio. (run)

```
private void btnBoton_Click(object sender, EventArgs e)
{
    //MessageBox.Show("Se disparo el evento Click", "Atención");
    //this.BackColor = Color.Blue;
    if (txtApellido.Text == "")
        txtApellido.BackColor = Color.Red;
    else
        txtApellido.BackColor = System.Drawing.SystemColors.Control;
}
```

7. Manejar el evento KeyPress, para ingresar solo números

```
private void txtApellido_KeyPress(object sender, KeyPressEventArgs e)
{
    if ((e.KeyChar < 48 || e.KeyChar > 59) && e.KeyChar != 8)
        e.Handled = true;
}
```

8. Agregar otro control TextBox, cambiar la propiedad Name.
9. Colocar en true la propiedad MultiLine del nuevo control TextBox. (run)
10. Cambiar la propiedad ScrollBars del nuevo control TextBox. (run)
11. Manejar el evento Leave del nuevo control TextBox. Para mostrar cuantos caracteres se ingresaron una vez que el control pierde el foco.

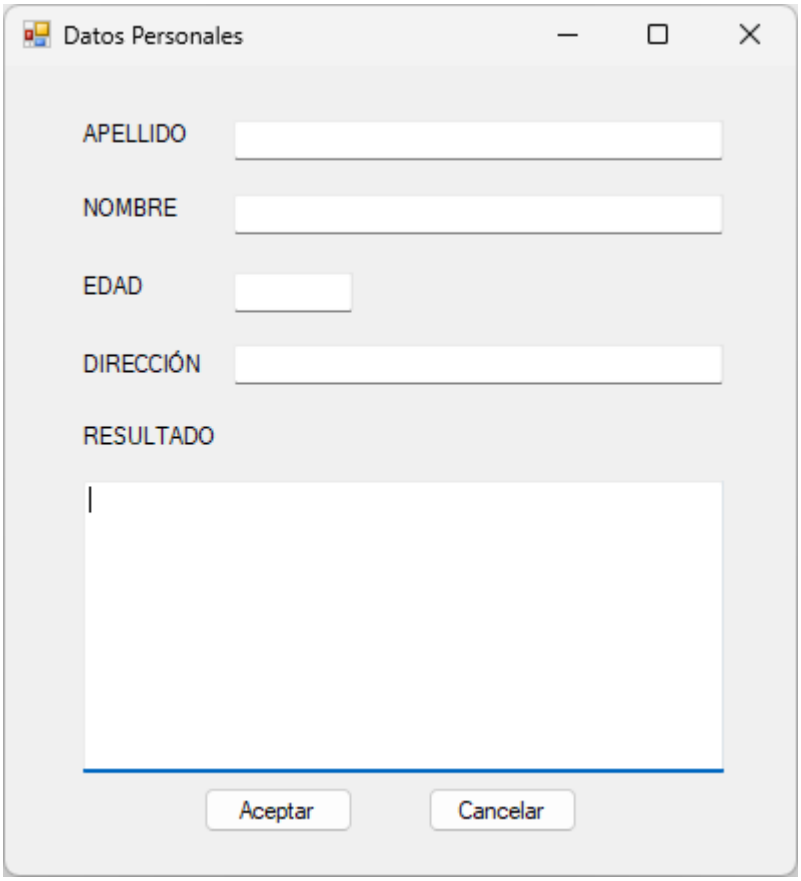
```
private void textNuevo_Leave(object sender, EventArgs e)
```

```
{
    MessageBox.Show("Tiene " + textNuevo.Text.Length + " Caracteres");
}
```

Practica 5 (aplicación windows)

Practica 5 (aplicación Windows)

Generar un formulario con los controles y diseño que se muestran en la siguiente imagen:



Al presionar el botón aceptar se debe validar que los TextBox Apellido, Nombre, Edad y Dirección tengan datos, en caso de estar vacíos marcarlos de color rojo.

Si pasa la validación los datos se deben escribir en el Text de resultado (TextBox multilínea) con el siguiente formato:

Apellido y Nombre: xxxxxxxxxxxxxxx

Edad: xxx

Dirección: xxxxxxxxxxxxxxxxxxxxx

En el campo Edad solo debe aceptar números.

En todos los campos limitar la cantidad de caracteres y pasarlos a mayúsculas. Al presionar el botón cancelar se debe cerrar la aplicación.