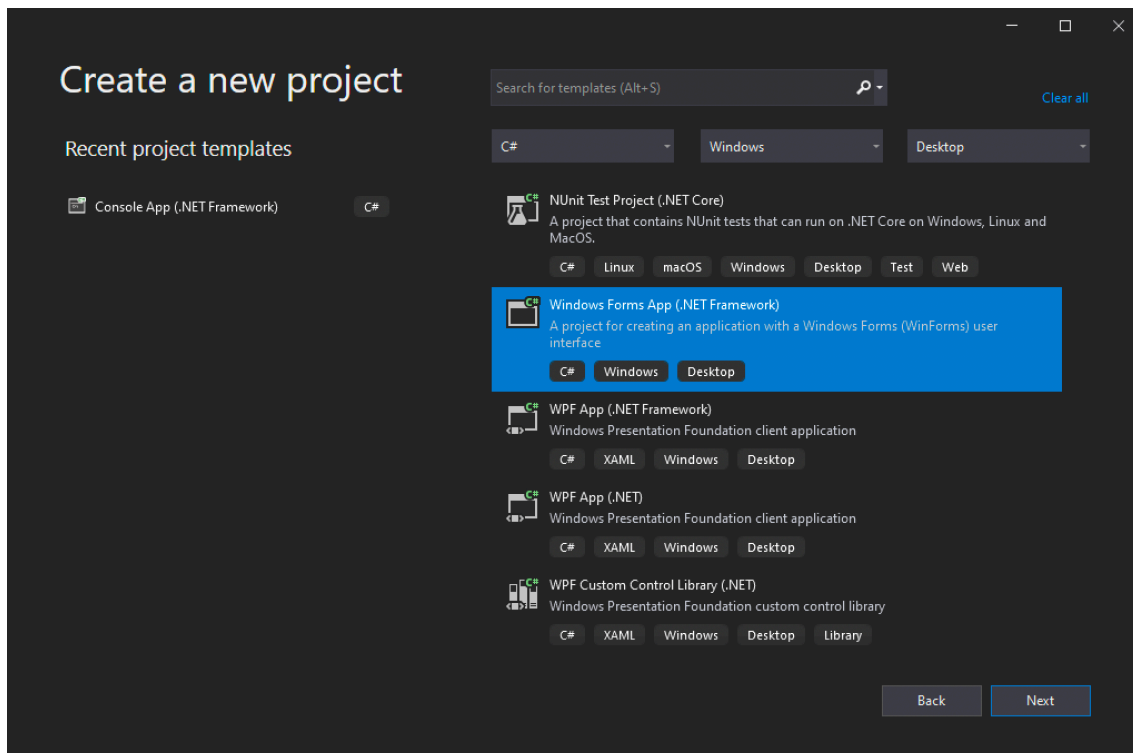


APLICACIONES WINDOWS I

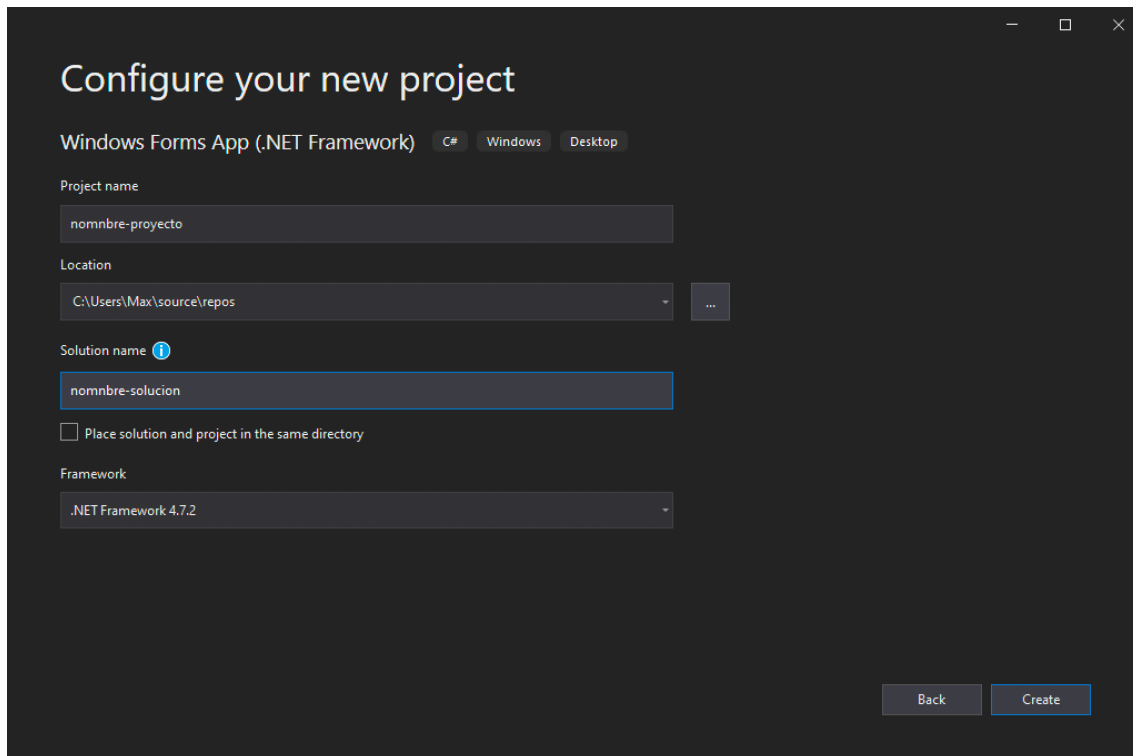
En este documento encontrarás una serie de ejercicios para trabajar con aplicaciones de escritorio con .Net y aplicar distinto tipo de controles.

CREAR UN NUEVO PROYECTO

1. Vamos Archivo > Nuevo > Proyecto esto abrirá un Dialog Box para seleccionar el tipo de Proyecto Nuevo.
2. En la parte izquierda tenemos la ventana de los templates recientes, es decir, los tipos de proyectos que hayamos creado recientemente; En los desplegables de la parte superior podemos ir seleccionando las opciones deseadas para poder filtrar y encontrar más rápido el tipo de proyecto. En ese caso seleccionamos C#, Windows y finalmente Desktop.
3. De las opciones listadas, seleccionaremos “Windows Forms App (.NET Framework)”. Es muy importante seleccionar la opción con “.Net Framework” ya que es la versión que trabajaremos.
4. A continuación presionamos en “Siguiente”.



5. En la siguiente pantalla debemos asignar un nombre para el proyecto y un nombre para la solución (paquete de proyectos) además de establecer la ruta en la que guardaremos dicha solución.



Configure your new project

Windows Forms App (.NET Framework) C# Windows Desktop

Project name
nomnbre-proyecto

Location
C:\Users\Max\source\repos

Solution name ⓘ
nomnbre-solucion

☐ Place solution and project in the same directory

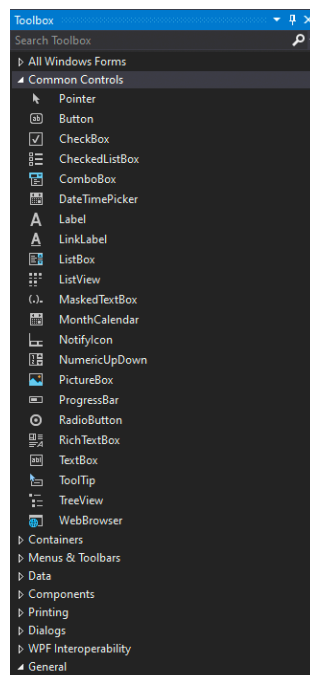
Framework
.NET Framework 4.7.2

Back Create

CONTROLES

Cuando se trabaja con Aplicaciones Windows Forms, se esta trabajando con el espacio de nombre(namespace) `System.Windows.Forms`, este espacio de nombre esta 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 controles 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.

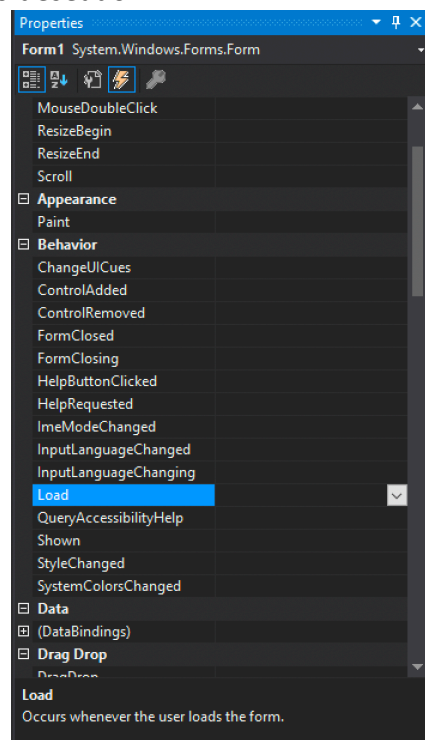
Las propiedades más comunes de estos controles son:

NOMBRE	DISPONIBILIDAD	DESCRIPCIÓN
Anchor	Lectura/Escritura	Especifica como 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	Allows you to make a control dock to the edges of a window. See below for a more detailed explanation of this property.
Enabled	Lectura/Escritura	Habilita o Deshabilita el Control.
ForeColor	Lectura/Escritura	El 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.
TabIndex	Lectura/Escritura	El numero de orden del control en el Form (Recipiente).
TabStop	Lectura/Escritura	Specifies whether the control can be accessed by the Tab key.
Tag	Lectura/Escritura	This value is usually not used by the control itself, and is there for you to store information about the control on the control itself. When this property is assigned a value through the Windows Form designer, you can only assign a string to it.
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

Un evento es básicamente una función que se lanza bajo determinadas circunstancias. Los eventos son disparados por los controles, generalmente esto sucede cuando el usuario realiza alguna acción sobre un control, por ejemplo, un click 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 (el rayito) y luego seleccionar el evento deseado.



Algunos eventos comunes:

NOMBRE	DESCRIPCIÓN
Click	Occurs when a control is clicked. In some cases, this event will also occur when a user presses Enter.
DoubleClick	Occurs when a control is double-clicked. Handling the Click event on some controls, such as the Button control will mean that the DoubleClick event can never be called.
DragDrop	Occurs when a drag-and-drop operation is completed, in other words, when an object has been dragged over the control, and the user releases the mouse button.

DragEnter	Occurs when an object being dragged enters the bounds of the control.
DragLeave	Occurs when an object being dragged leaves the bounds of the control.
DragOver	Occurs when an object has been dragged over the control.
KeyDown	Occurs when a key becomes pressed while the control has focus. This event always occurs before KeyPress and KeyUp.
Name	Description
KeyPress	Occurs when a key becomes pressed, while a control has focus. This event always occurs after KeyDown and before KeyUp. The difference between KeyDown and KeyPress is that KeyDown passes the keyboard code of the key that has been pressed, while KeyPress passes the corresponding char value for the key.
KeyUp	Occurs when a key is released while a control has focus. This event always occurs after KeyDown and KeyPress.
GotFocus	Occurs when a control receives focus. Do not use this event to perform validation of controls. Use Validating and Validated instead.
LostFocus	Occurs when a control loses focus. Do not use this event to perform validation of controls. Use Validating and Validated instead.
MouseDown	Occurs when the mouse pointer is over a control and a mouse button is pressed. This is not the same as a Click event because MouseDown occurs as soon as the button is pressed and before it is released.
MouseMove	Occurs continually as the mouse travels over the control.
MouseUp	Occurs when the mouse pointer is over a control and a mouse button is released.
Paint	Occurs when the control is drawn.
Validated	This event is fired when a control with the CausesValidation property set to true is about to receive focus. It fires after the Validating event finishes and indicates that validation is complete.
Validating	Fires when a control with the CausesValidation property set to true is about to receive focus. Note that the control which is to be validated is the control which is losing focus, not the one that is receiving it.

PRACTICA 1 (FORM)

1. Creamos un Proyecto nuevo con el nombre MiPrimerAplicacion.
2. Analizamos los archivos que nos creo el Proyecto.
3. Cambiamos el nombre del Form1, con la propiedad Name.
4. Cambiamos el titulo del Formulario, con la propiedad Text.
5. Cambiamos el color de fondo del Formulario, con la propiedad BackColor.
6. Cambiamos la posición del Formulario, con la propiedad StartPosition.
7. Cambiamos la vista inicial del Formulario (Normal, Minimizado o Maximizado), con la propiedad WindowState.
8. Manejar el evento Load del Formulario.

1. DoubleClick sobre el Formulario carga el método del evento Load por defecto.

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

}
```

2. Mostramos un Mensaje con MessageBox.Show.

```
private void Form1_Load(object sender, EventArgs e)
{
    MessageBox.Show("Bienvenidos a C#");
}
```

9. Manejar el evento FormClosed del Formulario.

1. Mostramos un Mensaje con MessageBox.Show.

```
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
<i>FlatStyle</i>	Cambia el Estilo del botón. Apariencia Plana o 3D.
<i>Enabled</i>	Habilita o Deshabilita el botón.(True o False)
<i>Image</i>	Carga una imagen(BMP, ICO ETC)
<i>ImageAlign</i>	Alinea la imagen en el botón.

CONTROLES LABEL Y LINKLABEL

Las etiquetas (Label) son los controles mas 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
<i>BorderStyle</i>	Especifica el tipo de borde de la Etiqueta. Por defecto sin borde.
<i>FlatStyle</i>	Estilo de la Etiqueta, plana o 3D.
<i>Image</i>	Cargar una Imagen.
<i>ImageAlign</i>	Alinear una imagen en la Etiqueta.
<i>LinkArea</i>	(LinkLabel)Especifica el rango del texto que debe mostrarse como Link.
<i>LinkColor</i>	(LinkLabel) Color del Link.
<i>Links</i>	---

LinkVisited	Cambia el color si el Link ya se visito.
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 etiqueta 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.

DobleClick sobre el Control Button carga el método del evento Click por defecto.

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

```
private void Form1_Click(object sender, EventArgs e)
{
    MouseEventArgs click = (MouseEventArgs)e;

    if (click.Button == MouseButton.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. En la ventana de eventos elijo MouseMove.

```

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

```

9. Cambiamos el color de la propiedad BackColor.

```

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

```

10. 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;
}

```

11. 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
<i>CausesValidation</i>	Estando en true se disparan dos eventos(Validating y Validated) al recibir el foco.
<i>CharacterCasing</i>	Convierte el Texto ingresado a Minúscula, Mayúscula o Normal.
<i>MaxLength</i>	Cantidad Máxima de caracteres que acepta.
<i>Multiline</i>	True o False, para aceptar multilíneas de texto.
<i>PasswordChar</i>	Especifica si el carácter de la contraseña debe sustituir a los caracteres ingresados.
<i>ReadOnly</i>	Especifica si es de solo Lectura.
<i>ScrollBars</i>	En caso de ser multilínea si se muestra el Scroll.
<i>SelectedText</i>	
<i>SelectionLength</i>	
<i>SelectionStart</i>	
<i>WordWrap</i>	

Eventos más comunes:

PROPIEDADES	DESCRIPCIÓN
<i>Enter</i> <i>Leave</i> <i>Validating</i> <i>Validated</i>	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.
<i>KeyDown</i> <i>KeyPress</i> <i>KeyUp</i>	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.
<i>TextChanged</i>	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.
5. Cambiamos la propiedad CharacterCasing del Control TextBox, para que cambie a mayúsculas los caracteres que se ingresan.
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 vacío.

```
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 txtNuevo_Leave(object sender, EventArgs e)
{
    MessageBox.Show("Tiene " + txtNuevo.Text.Length + " Caracteres");
}
```

PRÁCTICA 5 (APLICACIÓN WINDOWS 2)

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



The image shows a Windows-style dialog box titled "Datos Personales". It has a standard title bar with minimize, maximize, and close buttons. The main area contains four input fields: "APELLIDO", "NOMBRE", "EDAD", and "DIRECCIÓN". The "EDAD" field is smaller than the others. Below these fields is a large multi-line text area labeled "RESULTADO". At the bottom of the dialog are two buttons: "Aceptar" and "Cancelar".

Al presionar el botón aceptar se debe validar que los text 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: XXXXXXXXXXXXXXXX

Edad: XXX

Dirección: XXXXXXXXXXXXXXXXXXXX

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.