UT2. Desarrollo de Interfaces mediante XAML

Desarrollo de Interfaces

Índice

Lenguajes de descripción de interfaces basados en XML. XAML

- **XAML** Es un lenguaje de marcas empleado para la creación de interfaces en el modelo de programación .NET Framework de Microsoft. Las aplicaciones creadas podrán ejecutarse en entornos Windows.
- Mediante XAML se pueden crear lo que se conoce como **RIA (Rich Interface Applications)** que contienen abundante material gráfico.
- XAML consta de una serie de elementos XML para representar los principales componentes gráficos, así como la distribución, paneles y manejadores de eventos. Puede hacerse programando directamente la interfaz mediante un editor de texto, o mediante el entorno de desarrollo gráfico.

Lenguajes de descripción de interfaces basados en XML. XAML

- La realización de cálculos se añaden en un archivo independiente. Esto permite al equipo de desarrollo y al de diseño trabajar por separado, sin interferir mutuamente en su trabajo.
- Proceso de **mapeado**: se realiza en tiempo de ejecución, los elementos de la interfaz se conectan con objetos del framework .NET y los atributos con propiedades de estos objetos para integrarlos en la aplicación. Para facilitar la traducción de XAML a código .NET se ha creado una relación para cada elemento XAML con una clase de .NET.

Lenguajes de descripción de interfaces basados en XML. XAML

XAML: “Xtensible Application Markup Language“

- Microsoft
- Basado en XML que define objetos y sus propiedades.
- Sintaxis: definir la UI para la “Windows Presentation Foundation" (WPF)
- Independencia del código de la aplicación.

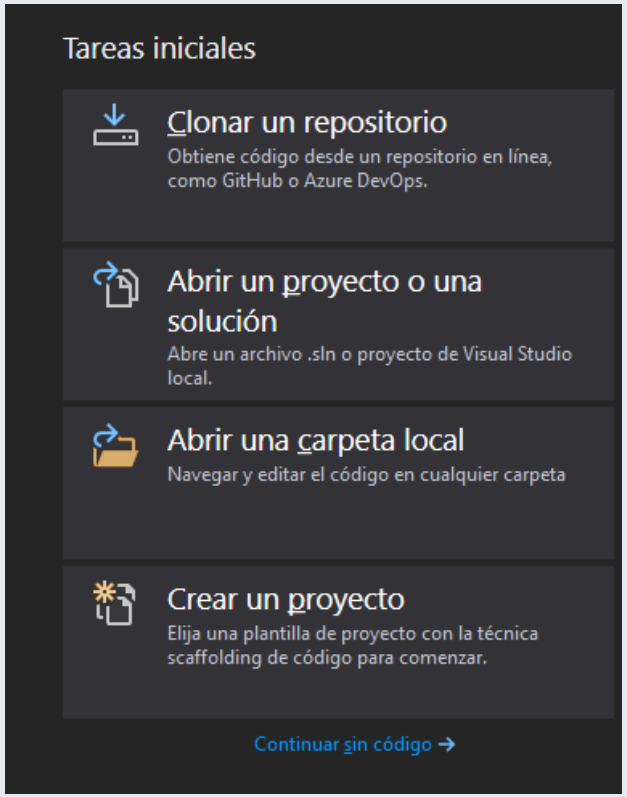
Tutoriales: http://www.scriptol.com/xaml/

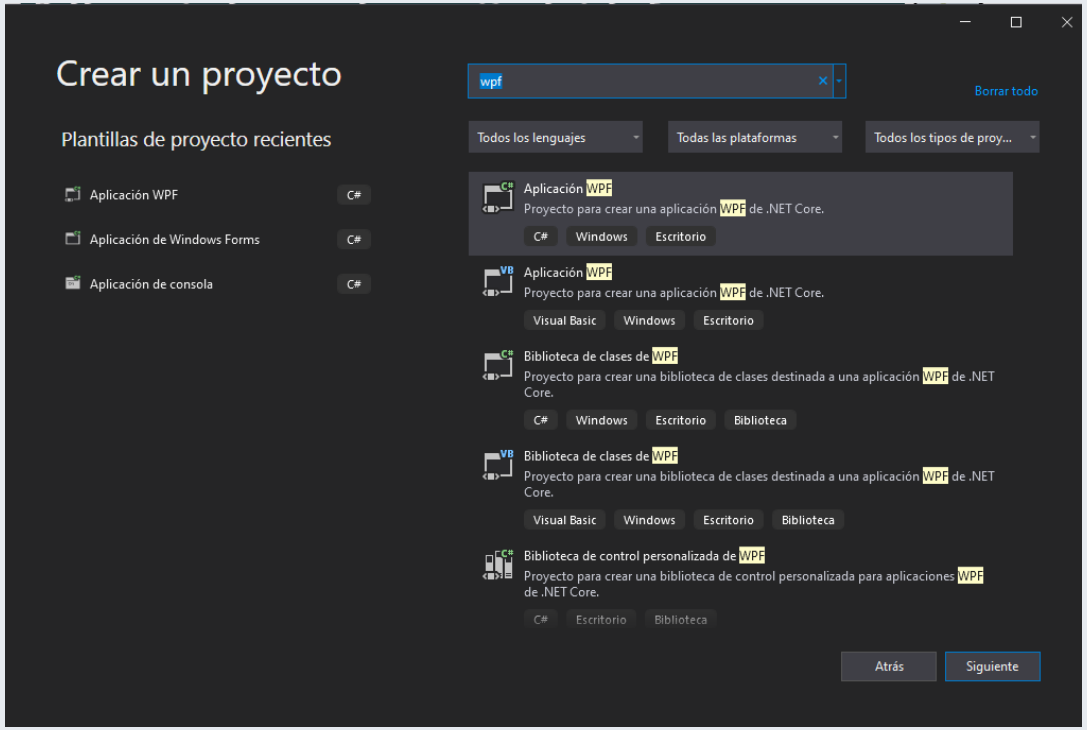
Lenguajes de descripción de interfaces basados en XML. XAML

- **WPF** (**Windows Presentation Foundation**). Es una serie de ensamblados y herramientas del framework .NET. Está destinado a proporcionar una API (Interfaz de programación de aplicaciones) para crear interfaces de usuario enriquecidas y sofisticadas para Windows.
- **Combina diferentes plataformas de desarrollo**. Coge cosas del desarrollo **web**, de las aplicaciones de Internet enriquecidas o **RIA** (del inglés Rich Internet Applications) y por supuesto del desarrollo de aplicaciones para **Windows**.
- Del desarrollo web hereda la **utilización de un lenguaje de meta etiquetas** para el desarrollo de la interfaz gráfica UI y los estilos. De las aplicaciones **RIA hereda los gráficos** vectoriales, animaciones y el soporte para multimedia.
- Además de las aportaciones de otras plataformas, WPF incorpora nuevas funcionalidades como soporte 3D, tipografía avanzada y documentos similares al PDF.

Lenguajes de descripción de interfaces basados en XML. XAML

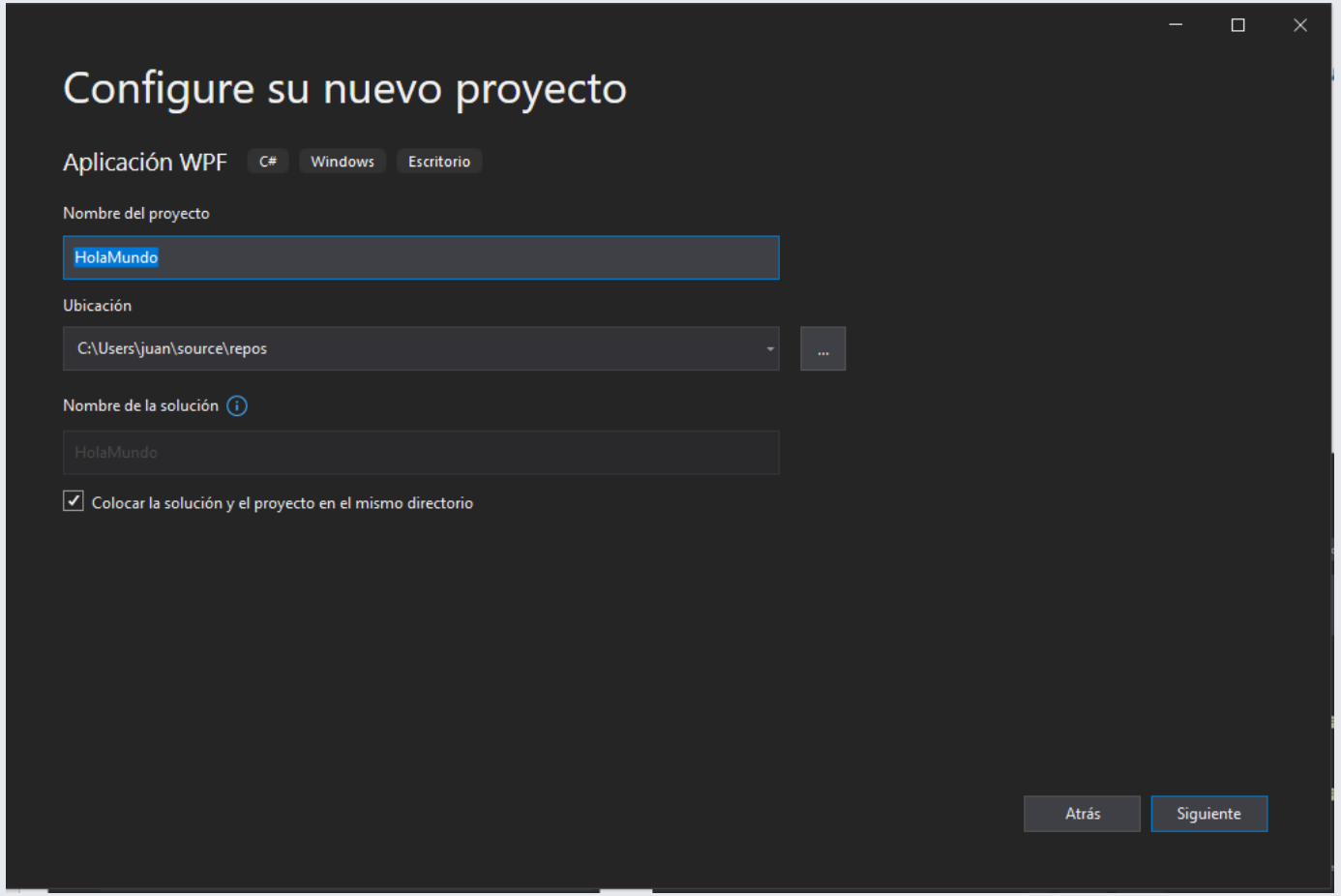
Crear un nuevo proyecto WPF ”Hola Mundo”





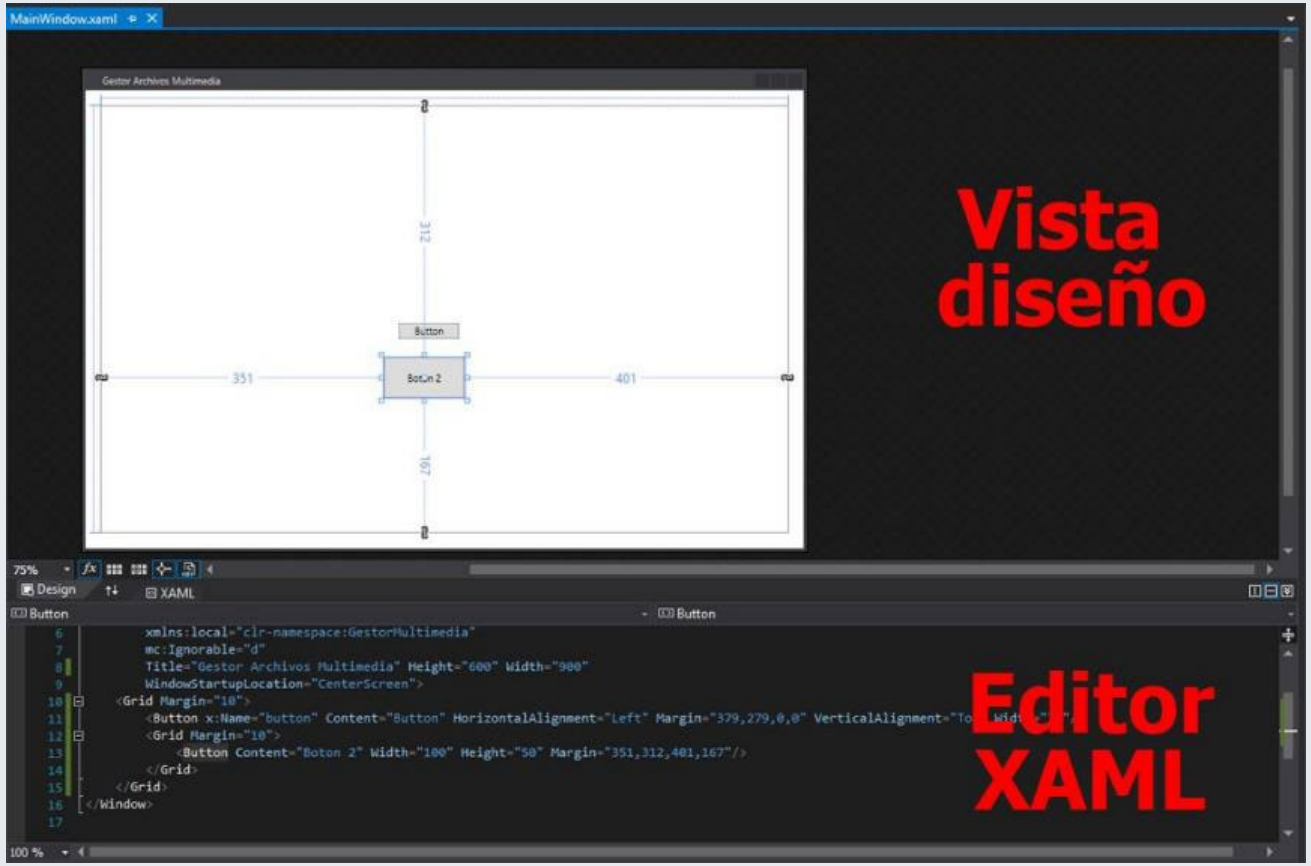
Lenguajes de descripción de interfaces basados en XML. XAML

Crear un nuevo proyecto WPF ”Hola Mundo”



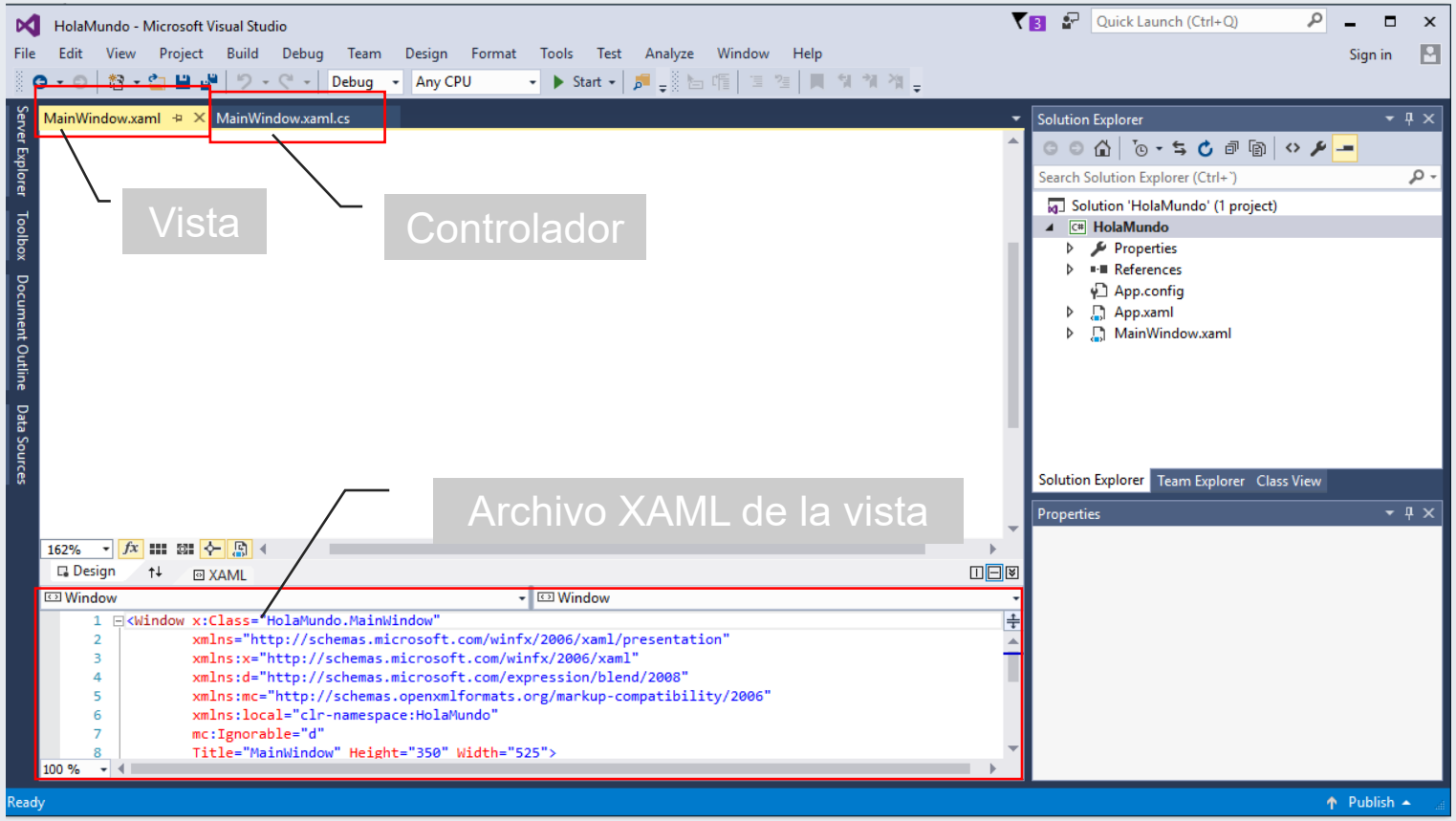
Lenguajes de descripción de interfaces basados en XML. XAML

Vistas del proyecto WPF ”Hola Mundo”



Lenguajes de descripción de interfaces basados en XML. XAML

Vistas del proyecto WPF ”Hola Mundo”



Lenguajes de descripción de interfaces basados en XML. XAML

- En el documento XAML, vemos un documento XML en el que el elemento raíz es **Window**. En la propia etiqueta Window se incluyen los atributos que definen la ventana que se creará.
- Dentro de esta etiqueta raíz nos aparece el contenedor principal de los componentes de la interfaz. Por defecto será de tipo **Grid**.
- Si agregamos una etiqueta y un botón a la interfaz, desde el menú de **Cuadro de herramientas**, veremos que se agregan dos nuevas etiquetas en el documento XAML que representan los componentes <Label> y <Button>. El texto de cada control se encuentra en el campo **Content** de su etiqueta correspondiente, aunque puede editarse, también, desde el panel de **Propiedades**. Al hacer doble click en el botón, nos aparece el fichero **Controlador** para definir el evento **Click** que también aparecerá en el documento XML.
- Aunque inicialmente no aparece definido, es conveniente poner un nombre **Name** a todos los controles para identificarlos.

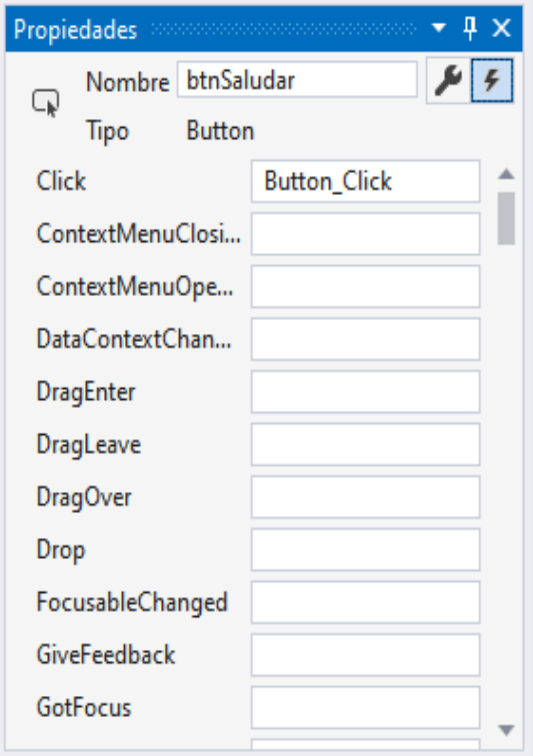
Lenguajes de descripción de interfaces basados en XML. XAML

Vistas del proyecto WPF ”Hola Mundo”

Lenguajes de descripción de interfaces basados en XML. XAML

Vistas del proyecto WPF ”Hola Mundo”

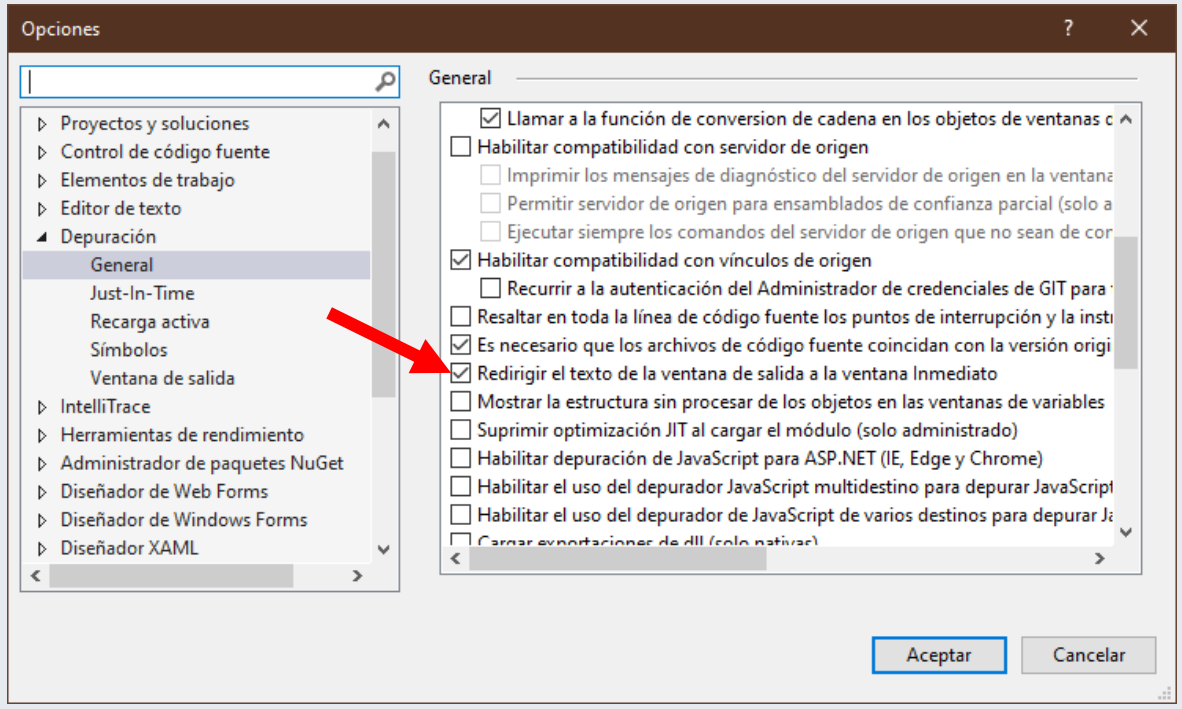
```
namespace UT_2_Ejemplo_Hola_Mundo
{
    /// <summary>
    /// Lógica de interacción para MainWindow.xaml
    /// </summary>
    2 referencias
    public partial class MainWindow : Window
    {
        0 referencias
        public MainWindow()
        {
            InitializeComponent();
        }

        0 referencias
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            lblSaludo.Content = "Hola Mundo";
        }
    }
}
```



Lenguajes de descripción de interfaces basados en XML. XAML

Vistas del proyecto WPF ”Hola Mundo”

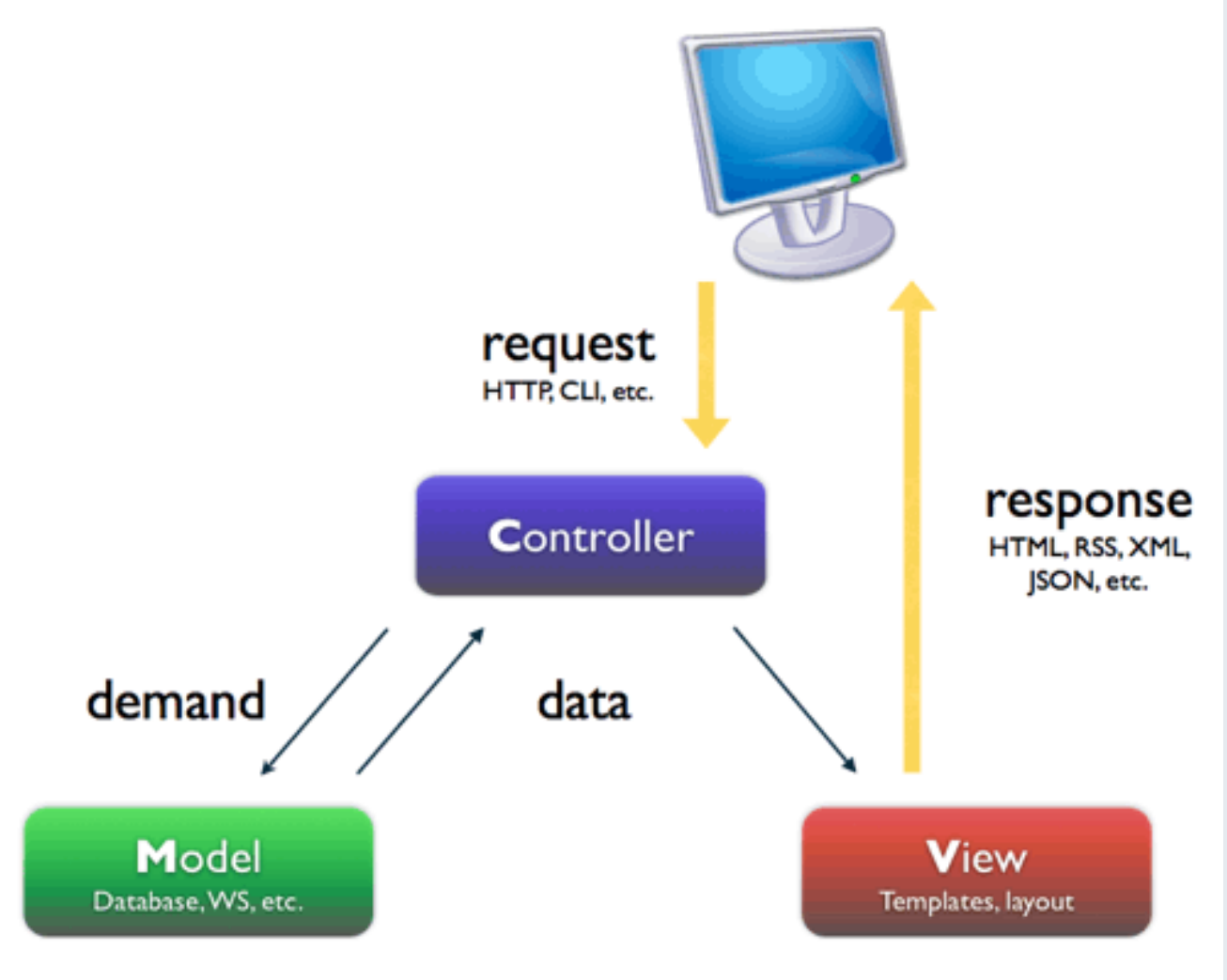
- Menú Depurar -> Opciones
- Para conseguir mostrar los mensajes Debug.WriteLine(””) en la ventana Inmediato (que está en Depurar->Ventanas).



Lenguajes de descripción de interfaces basados en XML. XAML. Arquitectura MVC

- La **arquitectura MVC** se basa en la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, (Model, Views & Controllers).
- **Modelos** -> Es la capa donde se trabaja con los datos (ficheros, bases de datos, red, ...)
- **Vistas** -> Visualización de las interfaces de usuario (interfaz gráfica, interfaz en modo texto, ...)
- **Controladores** -> Contiene el código necesario para responder a las acciones que se solicitan en la aplicación (enlace entre las vistas y los modelos)

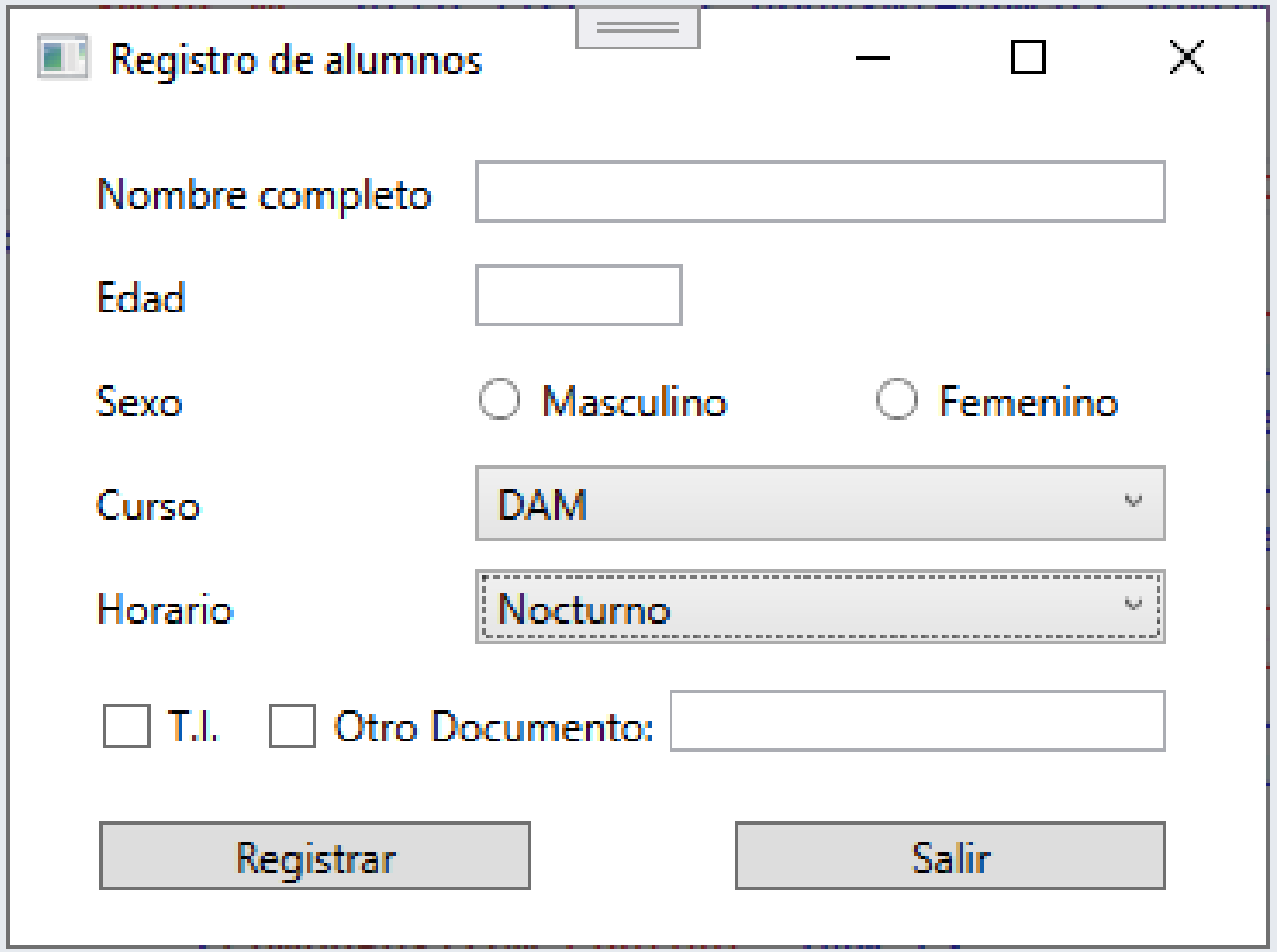
Lenguajes de descripción de interfaces basados en XML. XAML. Arquitectura MVC



Lenguajes de descripción de interfaces basados en XML. XAML. Arquitectura MVC

- El usuario **realiza una solicitud** que le llega al controlador.
- El **controlador comunica tanto con modelos como con vistas**. A los modelos les solicita datos o les manda realizar actualizaciones de los datos. A las vistas les solicita la salida correspondiente, una vez se hayan realizado las operaciones pertinentes según la lógica del negocio. Fichero con extensión **.xaml.cs.** Es la parte del **Código de programación.**
- Para producir la salida, en ocasiones **las vistas pueden solicitar más información a los modelos**.
- Las vistas envían al usuario la salida. Es la vista de **Diseño** en el fichero **.xaml**

Lenguajes de descripción de interfaces basados en XML. XAML.
Ejemplo interfaz de registro de alumnos



Lenguajes de descripción de interfaces basados en XML. XAML.
Ejemplo interfaz de registro de alumnos

```
<Window x:Class="Ejemplos.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:Ejemplos" mc:Ignorable="d"
       Title="Registro de alumnos" Height="280" Width="380">
    <Grid>
     <Label Content="Nombre completo" HorizontalAlignment="Left"
            Margin="20,10,0,0" VerticalAlignment="Top"/>
     <TextBox HorizontalAlignment="Left" Margin="135,14,0,0" Text=""
            TextWrapping="Wrap" VerticalAlignment="Top" Width="200"/>
     <Label Content="Edad" HorizontalAlignment="Left"
            Margin="20,40,0,0" VerticalAlignment="Top"/>
     <TextBox HorizontalAlignment="Left" Margin="135,44,0,0" Text=""
TextWrapping="Wrap" VerticalAlignment="Top" Width="60"/>
     <Label Content="Sexo" HorizontalAlignment="Left"
            Margin="20,70,0,0" VerticalAlignment="Top"/>
```

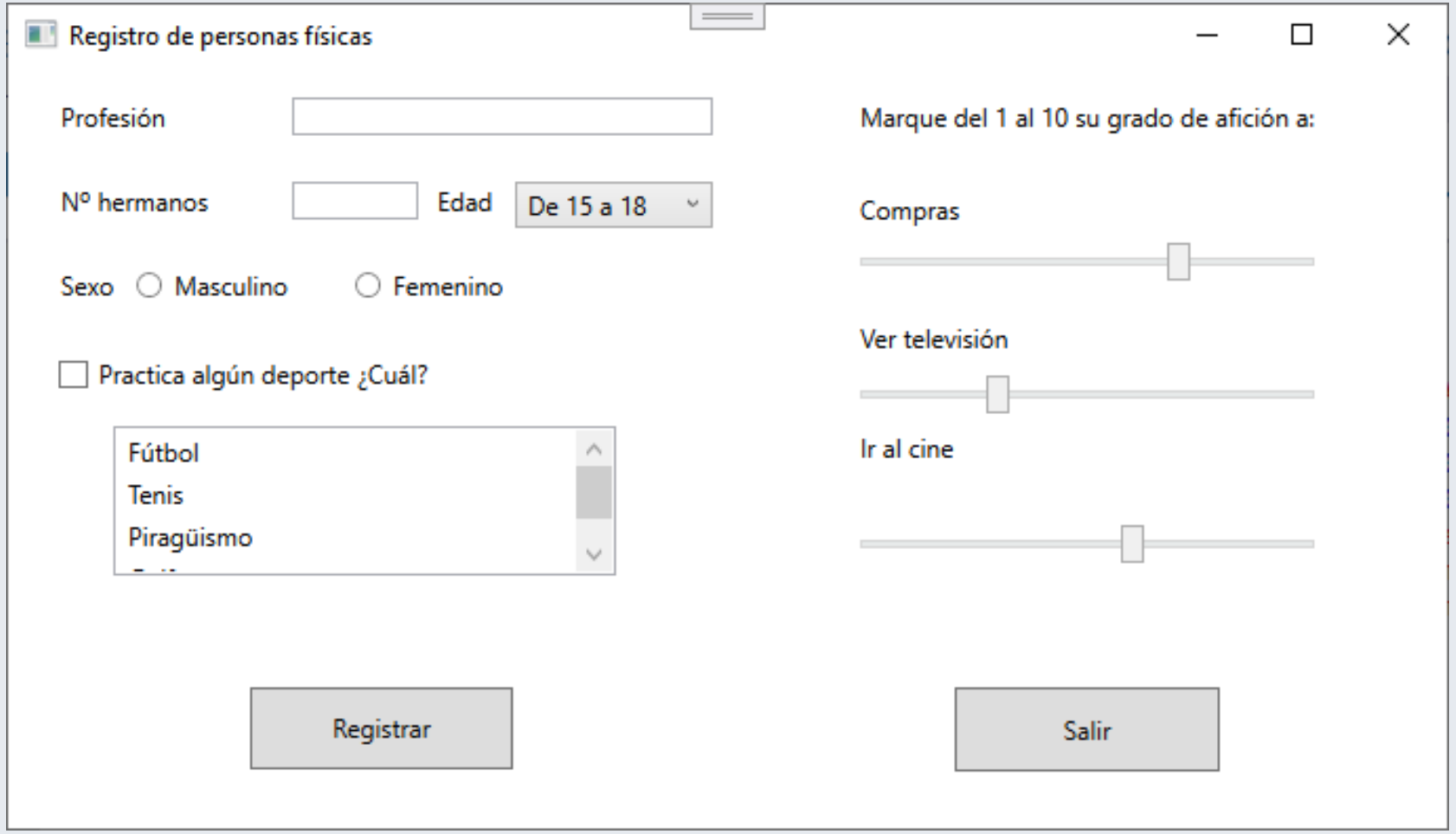
Lenguajes de descripción de interfaces basados en XML. XAML.
Ejemplo interfaz de registro de alumnos

```
<RadioButton Content="Masculino" HorizontalAlignment="Left"
      Margin="135,76,0,0" VerticalAlignment="Top"/>
<RadioButton Content="Femenino" HorizontalAlignment="Left"
      Margin="250,76,0,0" VerticalAlignment="Top"/>
<Label Content="Curso" HorizontalAlignment="Left"
      Margin="20,100,0,0" VerticalAlignment="Top"/>
<ComboBox HorizontalAlignment="Left" Margin="135,102,0,0"
      VerticalAlignment="Top" Width="200">
            <ComboBoxItem Content="SMR"/>
            <ComboBoxItem Content="DAM"/>
            <ComboBoxItem Content="DAW"/>
            <ComboBoxItem Content="ASIR"/>
</ComboBox>
<Label Content="Horario" HorizontalAlignment="Left"
Margin="20,130,0,0" VerticalAlignment="Top"/>
```

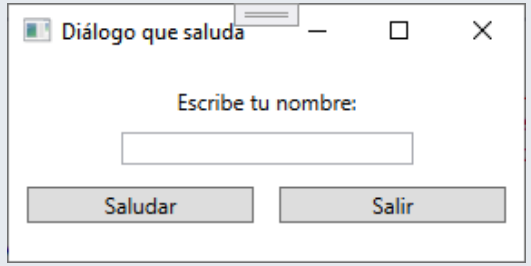
Lenguajes de descripción de interfaces basados en XML. XAML.
Ejemplo interfaz de registro de alumnos

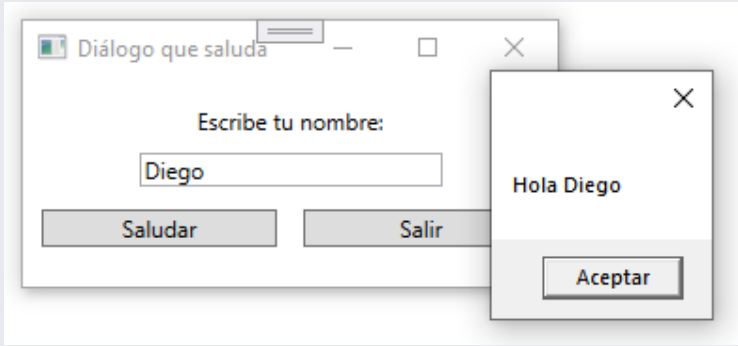
```
    <ComboBox HorizontalAlignment="Left" Margin="135,132,0,0"
        VerticalAlignment="Top" Width="200">
      <Button Content="Diurno"/>
      <CheckBox Content="Vespertino"/>
      <RadioButton Content="Nocturno"/>
    </ComboBox>
    <CheckBox Content="T.I." HorizontalAlignment="Left"
        Margin="26,170,0,0" VerticalAlignment="Top"/>
    <CheckBox Content="Otro Documento:" HorizontalAlignment="Left"
        Margin="74,170,0,0" VerticalAlignment="Top"/>
    <TextBox HorizontalAlignment="Left" Margin="191,167,0,0"
        TextWrapping="Wrap" Text="" VerticalAlignment="Top"
        Width="144"/>
    <Button Content="Registrar" HorizontalAlignment="Left"
        Margin="26,205,0,0" VerticalAlignment="Top" Width="125"/>
    <Button Content="Salir" HorizontalAlignment="Left"
        Margin="210,205,0,0" VerticalAlignment="Top" Width="125"/>
  </Grid>
</Window>
```

Lenguajes de descripción de interfaces basados en XML. XAML.
Ejemplo interfaz de registro de personas físicas



Lenguajes de descripción de interfaces basados en XML. XAML.
Ejemplo de cuadro de diálogo





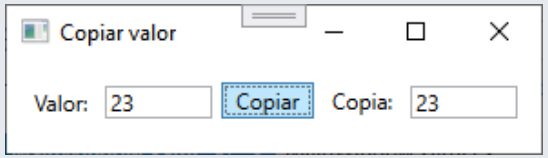

```
<Window x:Cl...
    ...    Title="Diálogo que saluda" Height="150" Width="300">
    <Grid>
        <Label Name="lblInstrucciones" Content="Escribe tu nombre:"
            HorizontalAlignment="Center" Margin="0,10,0,0"
            VerticalAlignment="Top"/>
        <TextBox Name="txtNombre"  HorizontalAlignment="Center" Width="161"
            Margin="0,40,0,0" TextWrapping="Wrap" VerticalAlignment="Top"/>
        <Button Name="btnSaludar" Content="Saludar" Margin="10,70,0,0"
            HorizontalAlignment="Left" VerticalAlignment="Top" Width="125"
            Click="btnSaludar_Click"/>
        <Button Name="btnSalir" Content="Salir“ HorizontalAlignment="Right"
            Margin="0,70,10,0" VerticalAlignment="Top" Width="125“ Click="btnSalir_Click"/>
     </Grid>
</Window>
```

Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de cuadro de diálogo

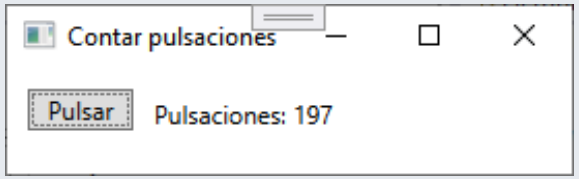
```
public partial class MainWindow : Window
{
    public MainWindow(){
            InitializeComponent();
    }
    private void btnSaludar_Click(object sender,RoutedEventArgs e) {
            MessageBox.Show("Hola " + txtNombre.Text);
    }
    private void btnSalir_Click(object sender,RoutedEventArgs e){
            Application.Current.Shutdown();
    }
}
```

Lenguajes de descripción de interfaces basados en XML. XAML.Otros ejemplos

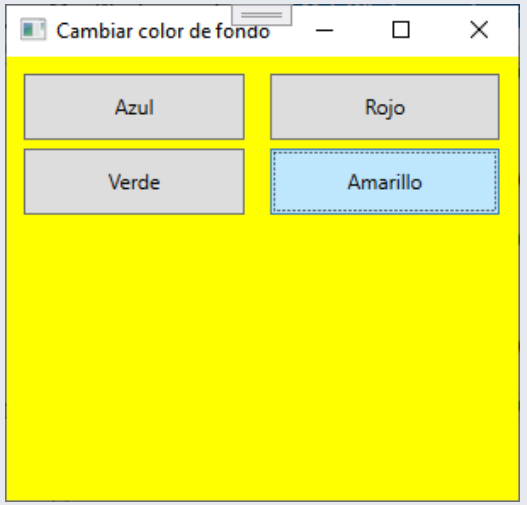
- Copiar un valor de un campo a otro. Usa el atributo **Text**



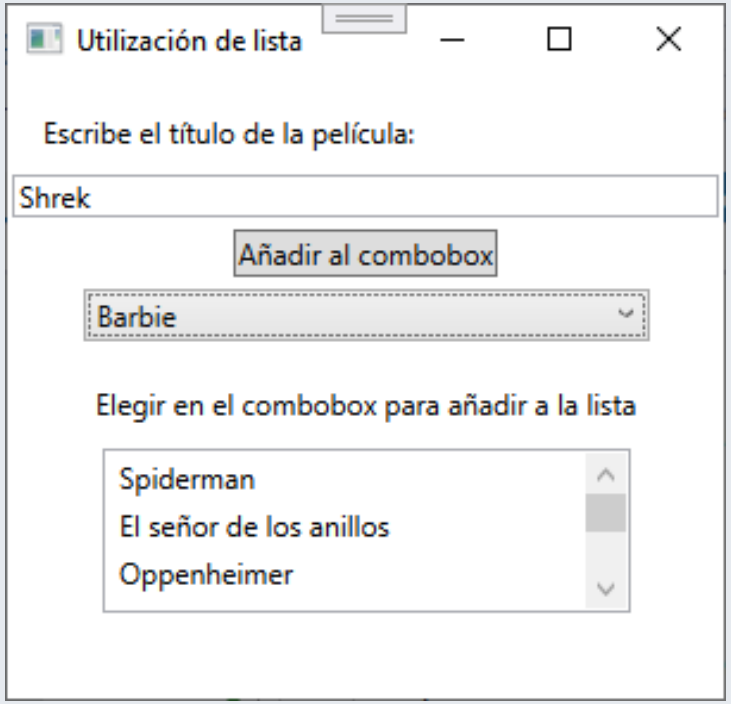
- Contador de pulsaciones sobre un botón. Usa el atributo **Content**



- Cambiar color de fondo de la aplicación. Usa el atributo **Background** y la clase **SolidColorBrush**

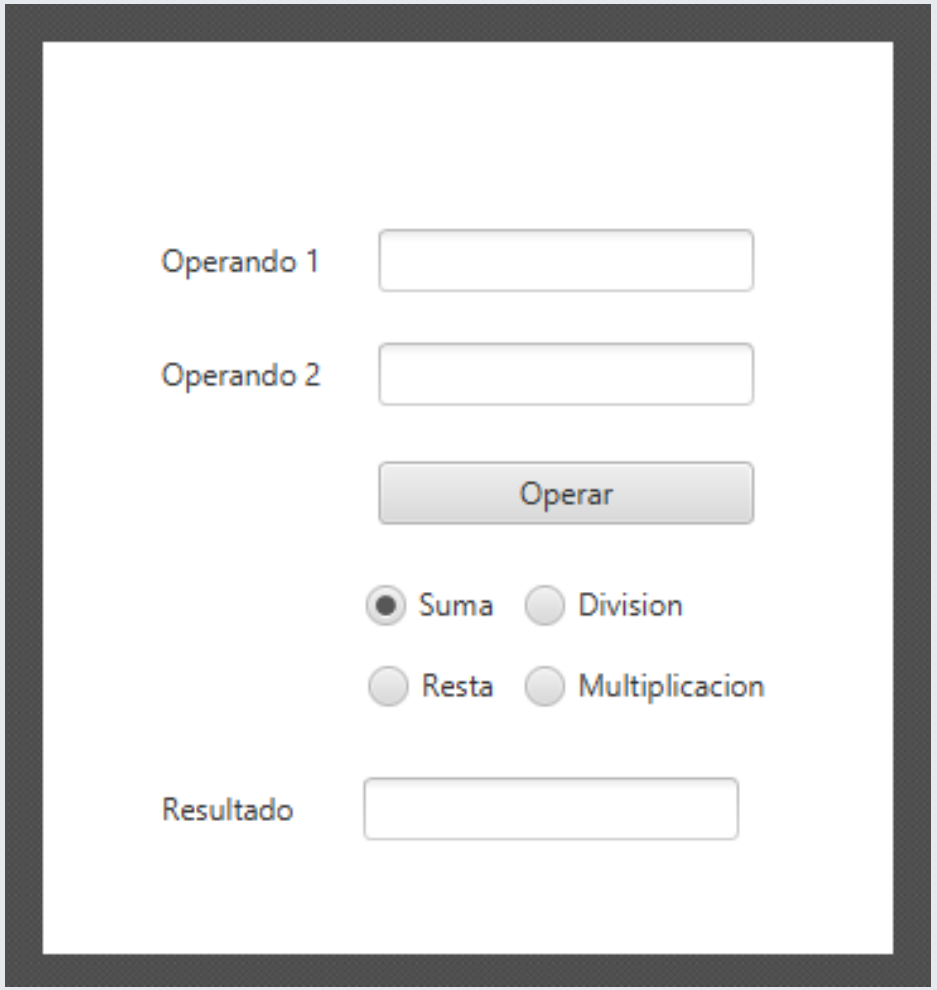


Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de añadir a una lista y seleccionar un elemento




```
btnAnyadir_Click:
cmbPeliculas.Items.Add(txtPelicula.Text);
cmbPeliculas_SelectionChanged:
lstPeliculas.Items.Add(cmbPeliculas.SelectedValue);
```

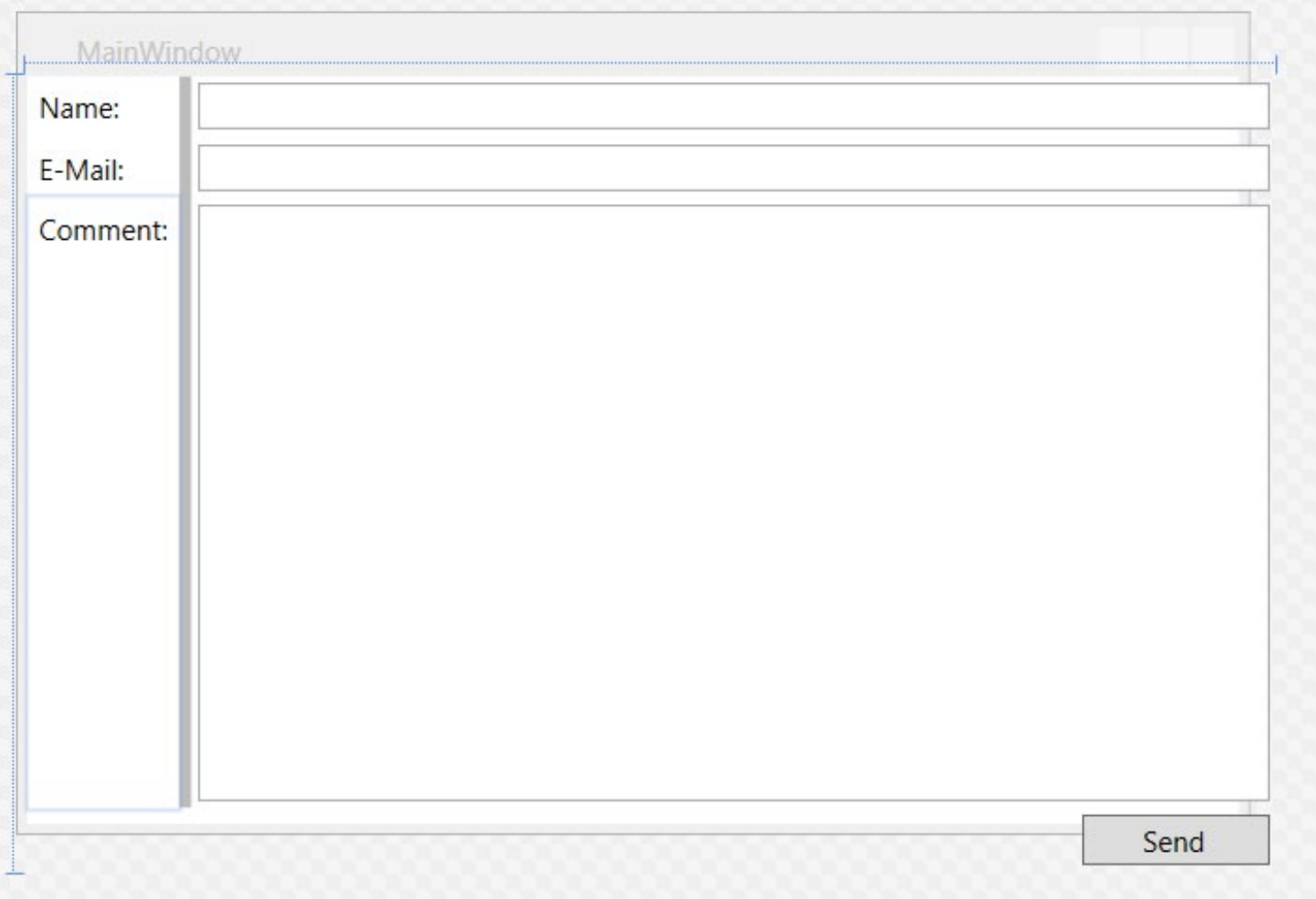
Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de calculadora con cuatro operaciones



Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: Grid**

- Contenedor **Grid:** Es el contenedor por defecto, tiene un estilo tabla con filas y columnas. Permite dividir el contenido en filas, cada una con una altura que puede ser diferente, y columnas, que tienen cada una un ancho que también puede ser variable. Lo primero que se hará será definir las filas y las columnas, y una vez definidas colocamos los elementos a través de las propiedades **Grid.Row** y **Grid.Column**.
- Tiene la posibilidad de fundir tanto filas como columnas.
- Sus tamaños pueden ser absolutos: en píxeles, relativos: en porcentajes, o automáticos utilizando *
- Si queremos hacer variable en tiempo de ejecución alguna columna o fila podemos utilizar el control **GridSplitter**.

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: Grid**

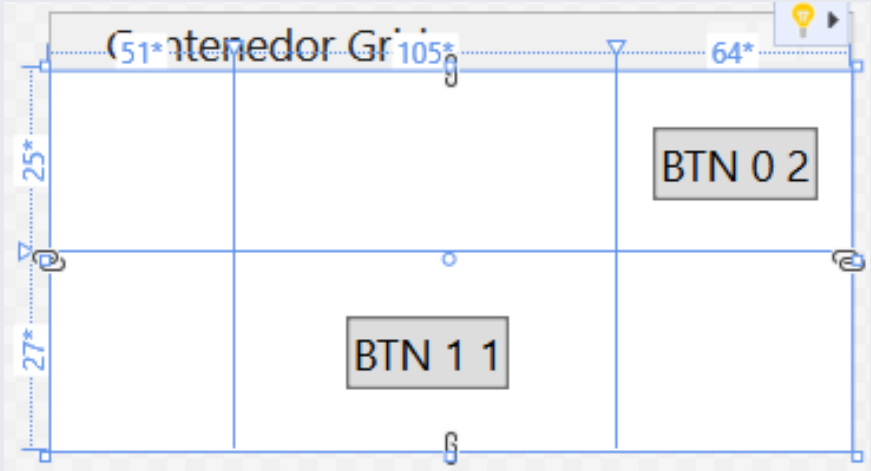


Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: Grid**

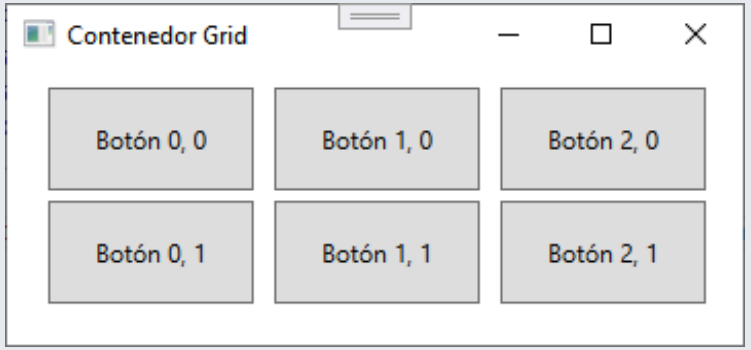
```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*" />
        <RowDefinition Height="28" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label Grid.Row="0" Grid.Column="0" Content="Name:"/>
    <Label Grid.Row="1" Grid.Column="0" Content="E-Mail:"/>
    <Label Grid.Row="2" Grid.Column="0" Content="Comment:"/>
    <GridSplitter HorizontalAlignment="Right"
            VerticalAlignment="Stretch"
            Grid.RowSpan="3" Grid.Column="1" ResizeBehavior="PreviousAndNext"
            Width="5" Background="#FFBCBCBC"/>
    <TextBox Grid.Column="2" Grid.Row="0" Margin="3" />
    <TextBox Grid.Column="2" Grid.Row="1" Margin="3" />
    <TextBox Grid.Column="2" Grid.Row="2" Margin="3" />
    <Button Grid.Column="2" Grid.Row="3" HorizontalAlignment="Right"
            MinWidth="80" Margin="3" Content="Send" />
</Grid>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Ejemplo de **Grid** diseñada con XAML

```
<Grid Name="grdContenedor">
  <Grid.RowDefinitions>
    <RowDefinition Height="25*"/>
    <RowDefinition Height="27*"/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="51*"/>
    <ColumnDefinition Width="105*"/>
    <ColumnDefinition Width="64*"/>
  </Grid.ColumnDefinitions>
  <Button Grid.Column="1" Grid.Row="1"...>
  <Button Grid.Column="2" Grid.Row="0"...>
</Grid>
```



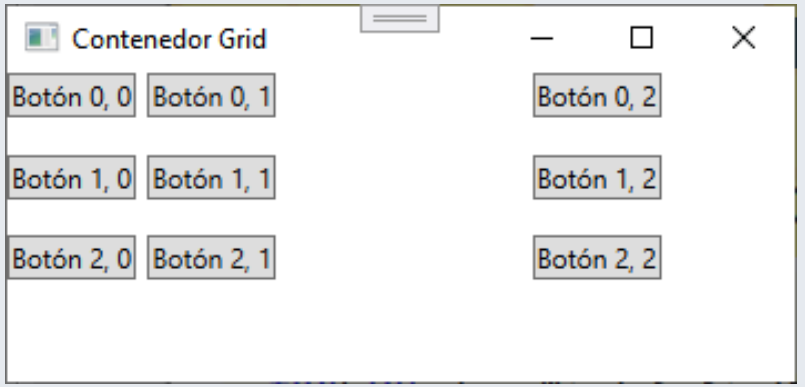
Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Añadir controles a un **Grid** que no tiene división en filas y columnas




```
for(int f = 0; f < 3; f++)
{
  for (int c = 0; c < 2; c++)
  {
    Button btnBoton = new Button();
    btnBoton.Width = 100;
    btnBoton.Height = 50;


    btnBoton.Content = "Botón " + f + ", " + c;
    btnBoton.HorizontalAlignment = HorizontalAlignment.Left;
    btnBoton.VerticalAlignment = VerticalAlignment.Top;

    //La colocación de los controles depende
    //de su alineación y del margen de cada control
    btnBoton.Margin = new Thickness(20 + f * 110, 10 + c * 55, 0, 0);
    grdContenedor.Children.Add(btnBoton);
  }
}
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Añadir controles a un **Grid** dividida en filas y columnas

Tenemos ya creadas 3 columnas y 3 filas

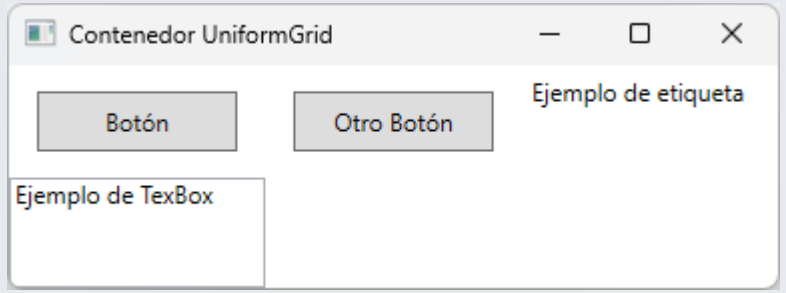



```
for (int f = 0; f < 3; f++)
{
  for (int c = 0; c < 3; c++)
  {
    Button btnBoton = new Button();
    btnBoton.Content = "Botón " + f + ", " + c;
    btnBoton.HorizontalAlignment = HorizontalAlignment.Left;
    btnBoton.VerticalAlignment = VerticalAlignment.Top;

    //La colocación de los controles depende de los métodos
    //estáticos SetRow y SetColumn de la clase Grid
    Grid.SetRow(btnBoton, f);
    Grid.SetColumn(btnBoton, c);
    grdContenedor.Children.Add(btnBoton);
  }
}
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: UniformGrid**

- **UniformGrid**: Distribución de elementos en una matriz cuadrada uniforme, es decir, según el tamaño de los objetos que contiene, distribuye el espacio de forma proporcional a todos.
- Funciona de la misma forma que **Grid** pero con la diferencia de que todas las filas tendrán el mismo alto y todas las columnas el mismo ancho.
- Se utiliza para crear tablas donde las celdas tienen dimensiones similares. Una vez definidas las filas y columnas, los controles se sitúan de forma automática conforme se crean: de izquierda a derecha y de arriba abajo.

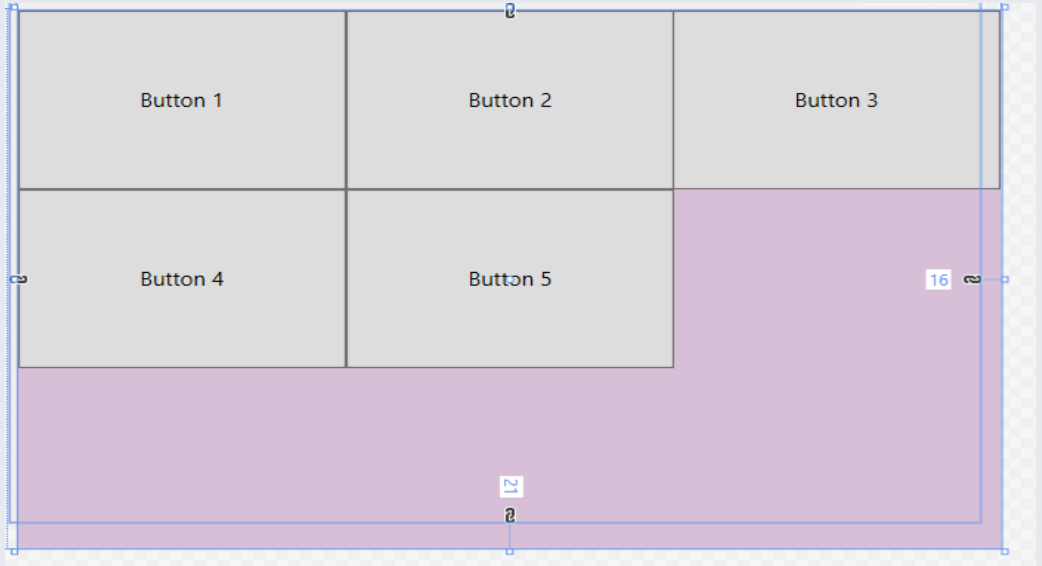
Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Ejemplo de **UniformGrid** diseñada con XAML




```
<UniformGrid Rows="2" Columns="3">
    <Button Content="Botón".../>
    <Button Content="Otro Botón".../>
    <Label Content="Label"/>
    <TextBox TextWrapping="Wrap".../>
</UniformGrid>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: UniformGrid**

- Si solo **Rows** se establece → UniformGrid calcula automáticamente el número de columnas necesarias.
- Si solo **Columns** se establece → calcula automáticamente el número de filas necesarias.
- Si ninguno se establece → intenta formar un cuadrado (mismo número de filas y columnas) según el total de hijos.

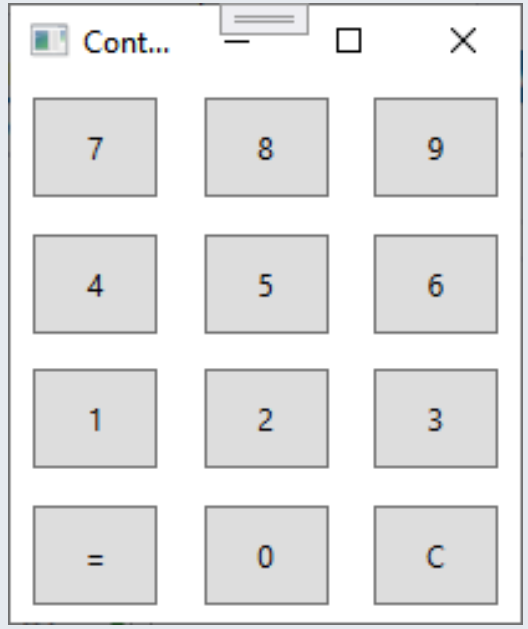



```
<UniformGrid Background="Thistle">
            <Button Content="Button 1"/>
            <Button Content="Button 2"/>
            <Button Content="Button 3"/>
            <Button Content="Button 4"/>
            <Button Content="Button 5"/>
</UniformGrid>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Añadir controles a un **UniformGrid**

Tenemos ya creadas 3 columnas y 4 filas

```
for (int f = 0; f < 4; f++)
{
  for (int c = 0; c < 3; c++)
  {
    Button btnBoton = new Button();
    btnBoton.Width = 50;
    btnBoton.Height = 40;
    if (f == 3)
       switch (c)
       {
         case 0: btnBoton.Content = "="; break;
         case 1: btnBoton.Content = "0"; break;
         case 2: btnBoton.Content = "C"; break;
       }
     else
     {
       btnBoton.Content = (9 - 3 * f) - (2 - c);
       btnBoton.HorizontalAlignment = HorizontalAlignment.Center;
       btnBoton.VerticalAlignment = VerticalAlignment.Center;
     }

     //La colocación de los controles se hace de
     //izquierda a derecha y de arriba a abajo
     ugrdContenedor.Children.Add(btnBoton);
  }
}
```

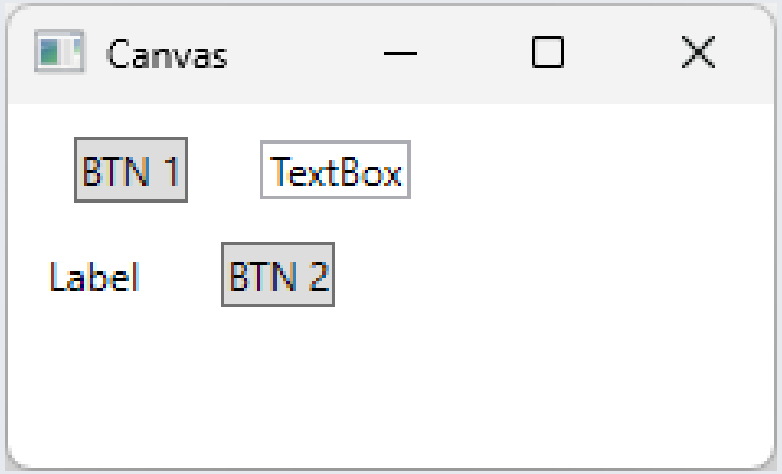


Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: Canvas**

- **Canvas**. Es el más simple e intuitivo de todos. Imita las características de un panel de Windows Forms. Permite asignar coordenadas absolutas donde el (0,0) del eje de coordenadas está en la esquina superior izquierda. Se trata de una ubicación precisa de elementos.
- Los objetos se van apilando uno encima del otro si no indicamos su posición absoluta.
- Define un área donde cada control hijo es posicionado en coordenadas absolutas.
- Se utiliza cuando se necesita control absoluto de las posiciones de los controles.
- No se recomienda usarlo a no ser que tengas alguna necesidad especial.

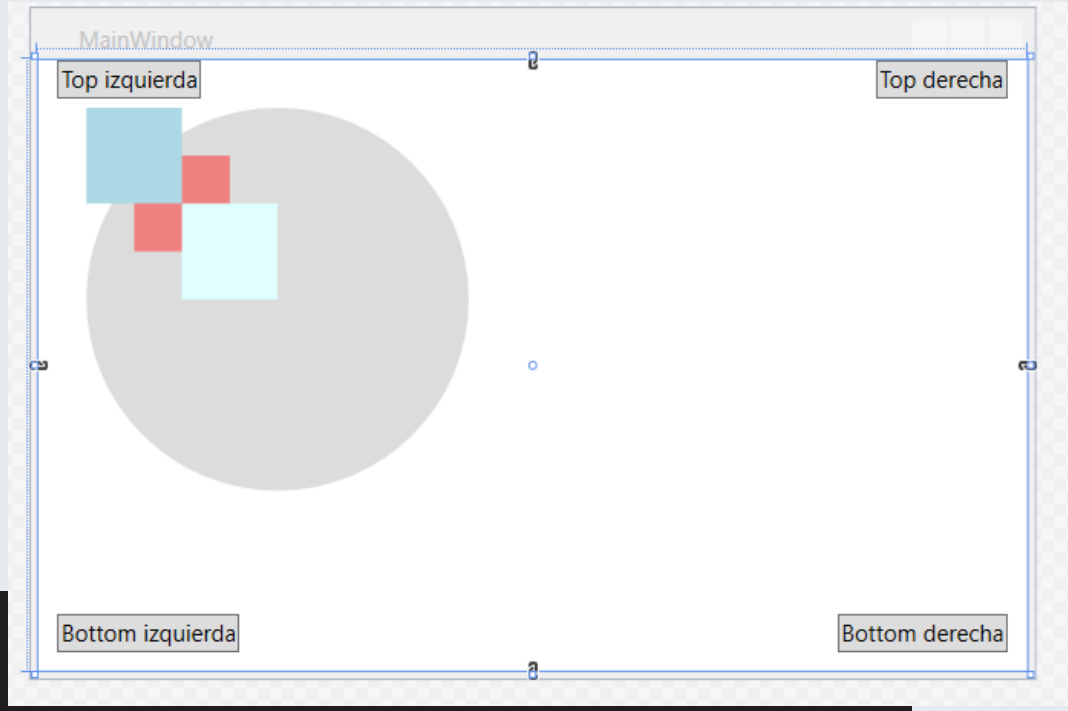
Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Ejemplo de **Canvas** diseñada con XAML

```
<Canvas Name="cnvContenedor">
    <Button Content="BTN 1"
            Canvas.Top="10" Canvas.Left="20"/>
    <Button Content="BTN 2"
            Canvas.Left="65" Canvas.Top="42"/>
    <Label Content="Label"
            Canvas.Left="7" Canvas.Top="39"/>
    <TextBox Canvas.Left="77" Canvas.Top="11".../>
</Canvas>
```



Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: Canvas**

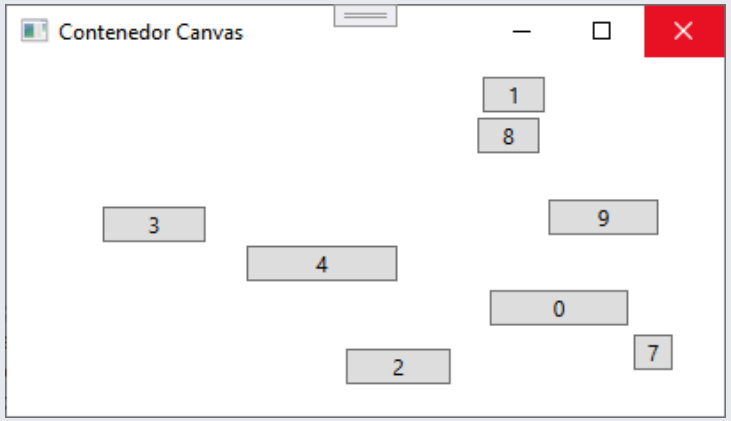
Las figuras también se pueden usar con Grid




```
<Canvas>
    <Button Canvas.Left="10">Top izquierda</Button>
    <Button Canvas.Right="10">Top derecha</Button>
    <Button Canvas.Left="10" Canvas.Bottom="10">Botton izquierda</Button>
    <Button Canvas.Right="10" Canvas.Bottom="10">Botton izquierda</Button>

    <Ellipse Panel.ZIndex="2" Fill="Gainsboro" Canvas.Left="25" Canvas.Top="25" Width="200" Height="200"/>
    <Rectangle Panel.ZIndex="3" Fill="LightBlue" Canvas.Left="25" Canvas.Top="25" Width="50" Height="50"/>
    <Rectangle Panel.ZIndex="2" Fill="LightCoral" Canvas.Left="50" Canvas.Top="50" Width="50" Height="50"/>
    <Rectangle Panel.ZIndex="4" Fill="LightCyan" Canvas.Left="75" Canvas.Top="75" Width="50" Height="50"/>
</Canvas>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Agregar componentes a un **Canvas**

Nota: hay que incluir un Canvas (puede ser manualmente) e incluir como nombre al formulario "cnvContenedor".

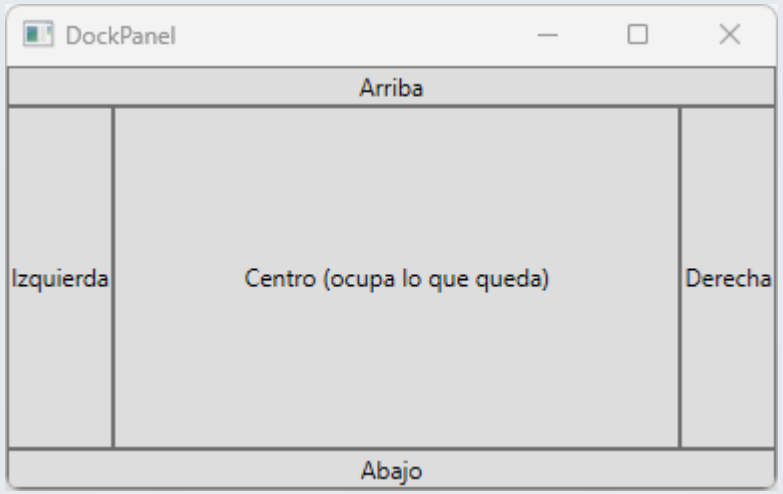



```
Random aleatorio=new Random();
for (int i = 0; i < 10; i++)
{
  Button btnBoton= new Button();
  btnBoton.Content = i;
  btnBoton.Width = aleatorio.Next(100);
  btnBoton.HorizontalAlignment = HorizontalAlignment.Center;
  btnBoton.VerticalAlignment = VerticalAlignment.Center;

  //La colocación de los controles depende de los métodos
  //estáticos SetLeft y SetTop de la clase Canvas
  Canvas.SetLeft(btnBoton, aleatorio.Next((int)this.Width));
  Canvas.SetTop(btnBoton, aleatorio.Next((int)this.Height));
  cnvContenedor.Children.Add(btnBoton);
}
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: DockPanel**

- **DockPanel**: contenedor de diseño que organiza los elementos secundarios (hijos) colocándolos en los bordes. Permite acoplar controles mapeando la pantalla en cinco partes diferentes: Top, Bottom, Left y Right (arriba, abajo, izquierda y derecha)
- La quinta parte será el espacio restante, normalmente en el centro, donde se expandirá el último componente al área sobrante. Si la propiedad **LastChildFill** es false no se expande.
- Se utiliza para dividir la ventana en zonas con funcionalidades específicas, como una barra de menú en la parte superior, una barra de estado en la parte inferior y el contenido principal en el centro.

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Ejemplo de **DockPanel** diseñada con XAML



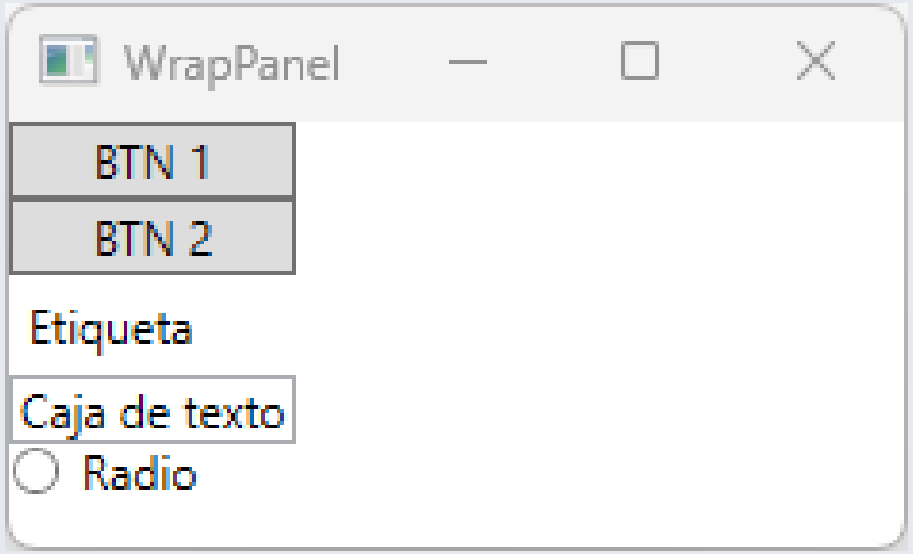

```xaml
<DockPanel LastChildFill="True">
    <Button Content="Arriba" DockPanel.Dock="Top" />
    <Button Content="Abajo" DockPanel.Dock="Bottom" />
    <Button Content="Izquierda" DockPanel.Dock="Left" />
    <Button Content="Derecha" DockPanel.Dock="Right" />
    <Button Content="Centro (ocupa lo que queda)" />
</DockPanel>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: WrapPanel**

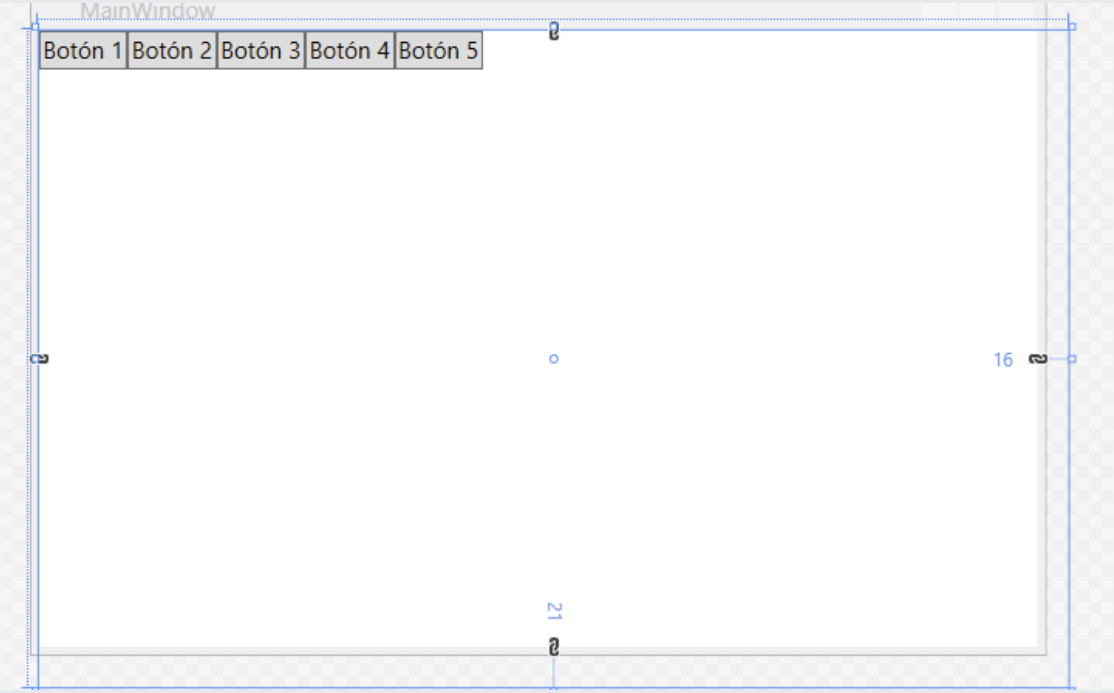
- **WrapPanel**: La posición de los objetos hijos es secuencial horizontal de izquierda a derecha o vertical de arriba hacia abajo. No ajusta los tamaños de los objetos.
- Se utiliza para posicionar los controles uno a continuación del otro, de forma similar a como se gestionan los componentes en los documentos HTML sin formato.
- Si falta espacio en la línea se pasa a la siguiente.
- Se pueden configurar para que se haga vertical.

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Ejemplo de **WrapPanel** diseñada con XAML

```
<WrapPanel Name="wrpContenedor" Orientation="Vertical">

    <Button Content="BTN 1" .../>
    <Button Content="BTN 2" .../>
    <Label Content="Etiqueta"/>
    <TextBox  Text="Caja de texto" .../>
    <RadioButton Content="Radio .../>

</WrapPanel>
```



Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: WrapPanel**

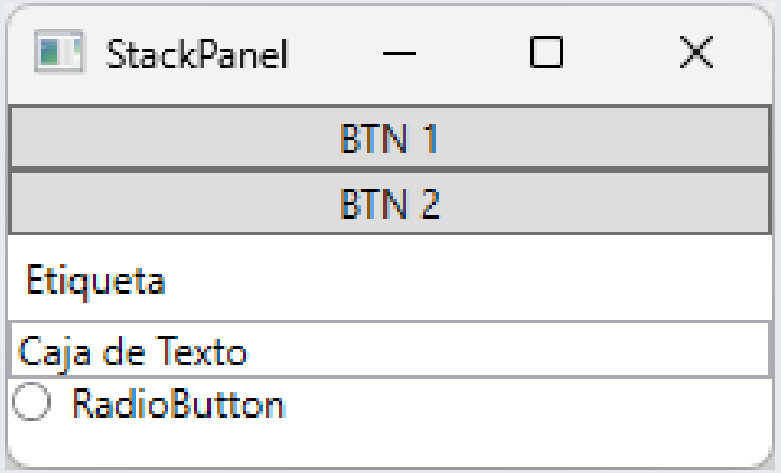



```
<WrapPanel Orientation="Horizontal">
            <Button Content="Botón 1" />
            <Button Content="Botón 2" />
            <Button Content="Botón 3" />
            <Button Content="Botón 4" />
            <Button Content="Botón 5" />
</WrapPanel>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: StackPanel**

- **StackPanel**: Es un contenedor que apila los objetos de manera horizontal o vertical. Por defecto se apilan de forma vertical pero repartiéndose sobre toda la superficie.
- Funciona de forma similar a como lo hace **WrapPanel** pero no se adapta si los controles ocupan demasiado sino que se expande.
- Se puede apilar de forma horizontal modificando el atributo **Orientation** = ”Horizontal”.
- Cada control se estira para ocupar todo el espacio (alto o ancho, dependiendo de su orientación) según se necesite.

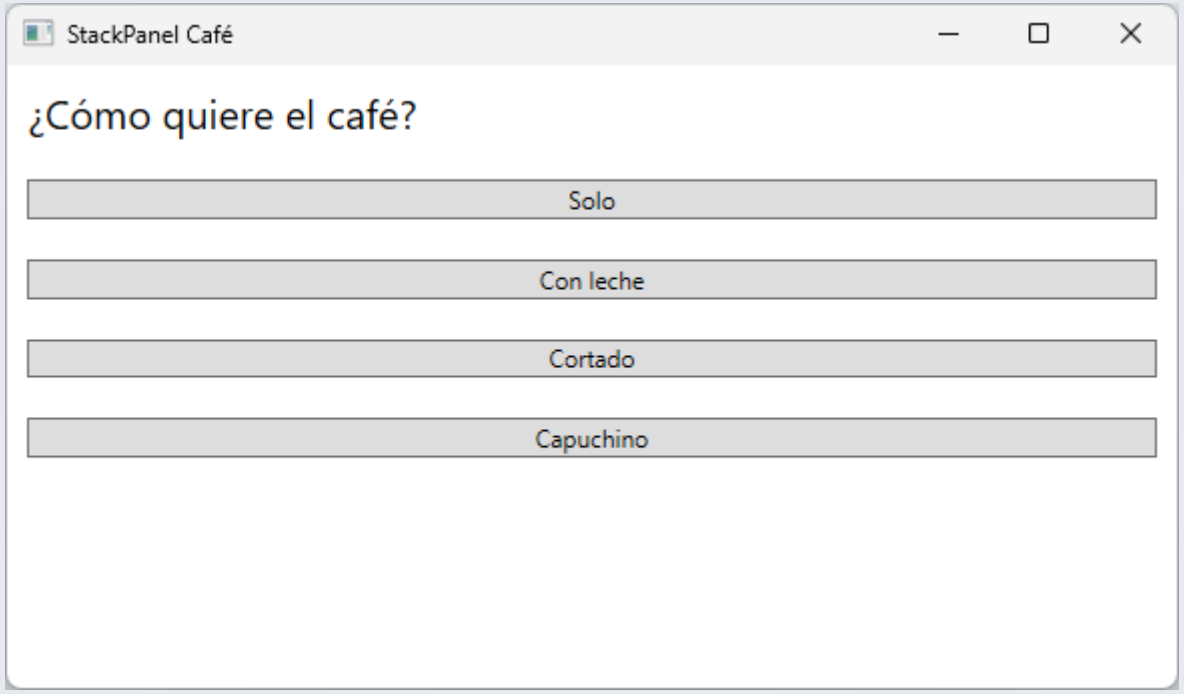
Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores.** Ejemplo de **StackPanel** diseñada con XAML




```
<StackPanel Name="contenedor" Orientation="Vertical">
    <Button Content="BTN 1"/>
    <Button Content="BTN 2"/>
    <Label Content="Etiqueta"/>
    <TextBox Text="Caja de Texto" TextWrapping="Wrap"/>
    <RadioButton Content="RadioButton"/>
</StackPanel>
```

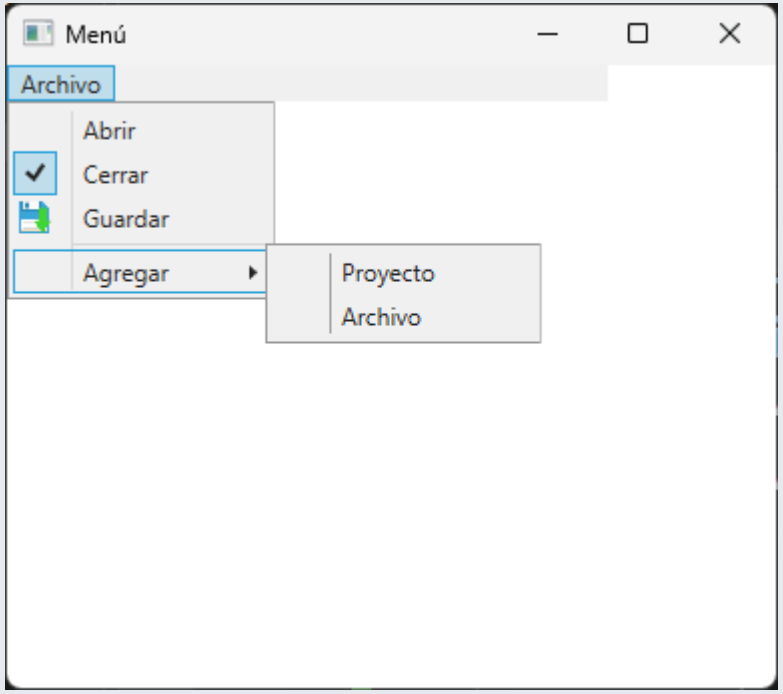
Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: StackPanel**

```
<StackPanel>
            <TextBlock Margin="10" FontSize="20">¿Cómo quiere el café?</TextBlock>
            <Button Margin="10">Solo</Button>
            <Button Margin="10">Con leche</Button>
            <Button Margin="10">Cortado</Button>
            <Button Margin="10">Capuchino</Button>
</StackPanel>
```

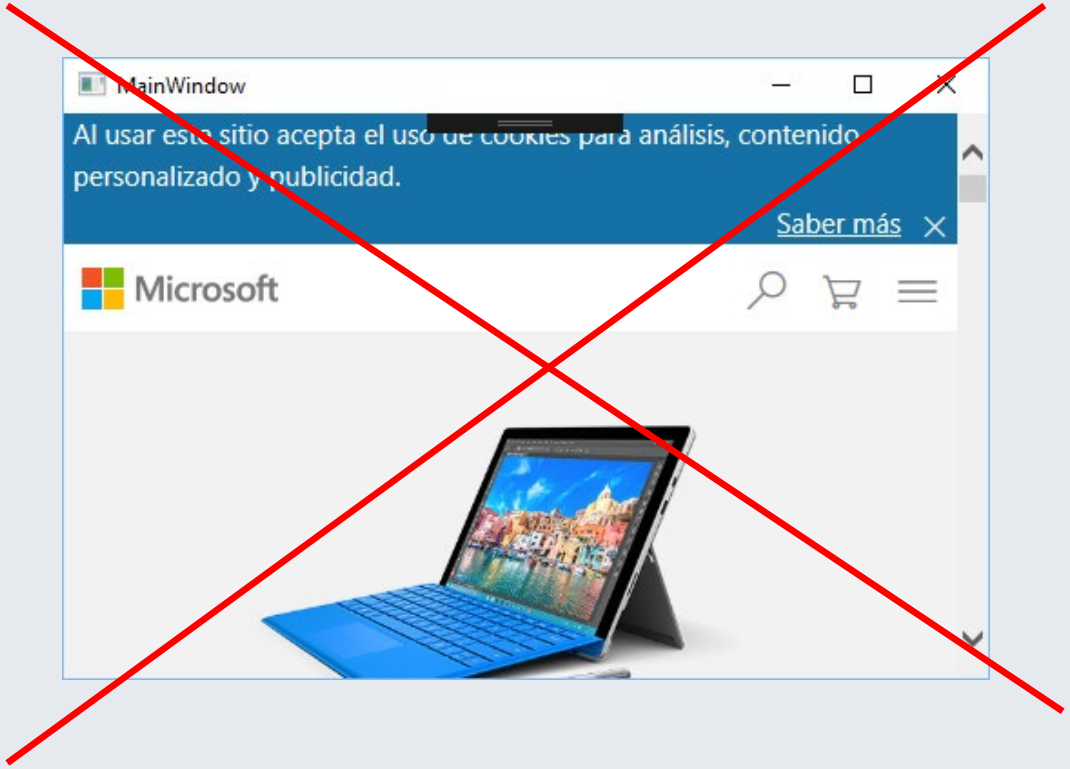


Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: Menú**

```
<Menu Name="menu1" Width="300" HorizontalAlignment="Left" VerticalAlignment="Top" >
    <MenuItem Header="_Archivo" >
          <MenuItem Header="A_brir" IsCheckable="true" />
          <MenuItem Header="_Cerrar" IsCheckable="true" IsChecked="True" />
          <MenuItem Header="_Guardar" >
              <MenuItem.Icon>
                  <Image Source="Save.ico" Width="16" Height="16" />
              </MenuItem.Icon>
          </MenuItem>
          <Separator />
          <MenuItem Header="Ag_regar" >
              <MenuItem Header="Proyecto" />
              <MenuItem Header="Archivo" />
          </MenuItem>
    </MenuItem>
</Menu>
```



Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: Frames**

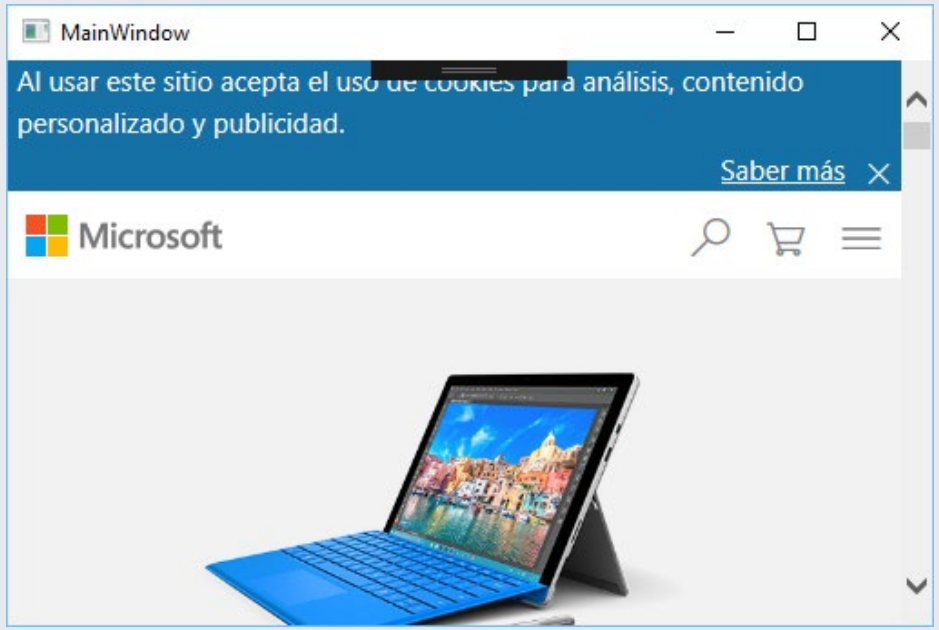


Nota: en la actualidad esto no funciona porque el navegador que está embebido en WPF es el Internet Explorer, y ya no funciona en ninguna página.

En la actualidad se usa **WebView2**, que tiene Edge como navegador.

```
<Grid>
        <Frame Name="miFrame" Source="http://www.microsoft.com" />
</Grid>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedores: WebView2**



WebView2 no viene instalado por defecto en los proyectos de WPF.

Hay que instalarlo en el menú de Administración de paquetes NuGet, y se llama “Microsoft.Web.WebView2”.

Además, hay que tener WebView2 Runtime en el ordenador, instalándolo con Evergreen Bootstrapper.

```
<Grid>
        <wv2:WebView2 Name="miWebView" Source="https://www.microsoft.com" />
</Grid>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedor - Border**

- **Border:** se utiliza para dibujar un borde y/o un fondo alrededor de otro elemento visual. Es un contenedor que puede tener un solo elemento hijo, y su función principal es encapsular contenido con un borde, color de fondo, esquinas redondeadas y márgenes.

```
<Border
   BorderBrush="Black"
   BorderThickness="2"
   CornerRadius="5"
   Background="LightGray"
   Padding="10"
   Margin="5">

   <TextBlock Text="Hola WPF!" />
</Border>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedor - Border**

Propiedades más comunes:

- **BorderBrush:** Define el color o pincel del borde. Ej.: "Black", "{DynamicResource AccentBrush}“
- **BorderThickness:** Define el grosor del borde (puede ser un número o varios valores para cada lado). Ej.: "2" o "2,4,2,4“
- **CornerRadius:** Redondea las esquinas del borde. Ej.: “10”
- **Background:** Establece el color o pincel del fondo del área dentro del borde. Ej.: "LightBlue“
- **Padding:** Espacio interno entre el borde y el contenido. Ej.: "10“
- **Margin:** Espacio externo alrededor del borde. Ej.: "5“
- **Child:** El elemento contenido dentro del <Border>. Ej.: <TextBlock> u otro control

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedor – Tab (Pestañas)**

Es un contenedor que permite organizar el contenido en pestañas (tabs), muy parecido a las pestañas de un navegador web.

```
<TabControl TabStripPlacement="Top">
            <TabItem Header="TabItem1">
            </TabItem>
            <TabItem Header="TabItem2">
            </TabItem>
</TabControl>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Contenedor – Tab (Pestañas)**

Propiedades útiles:

- **Items:** Colección de los elementos TabItem.
- **SelectedIndex:** Índice (0, 1, 2, …) de la pestaña actualmente seleccionada.
- **SelectedItem:** El TabItem actualmente seleccionado.
- **TabStripPlacement:** Posición de las pestañas: Top, Bottom, Left, Right.
- **ItemTemplate:** Permite personalizar la apariencia del encabezado de las pestañas.
- **ContentTemplate:** Permite definir una plantilla para el contenido.

Lenguajes de descripción de interfaces basados en XML. XAML. **Controles: Label – Access Text**

Label – AccessText

Representa la etiqueta de texto de un control.

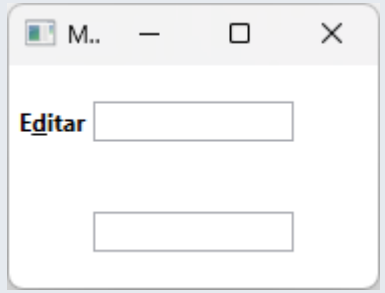
Esta clase proporciona compatibilidad funcional y visual con las teclas de acceso.

Se utiliza con frecuencia para permitir el acceso rápido desde el teclado a controles como *TextBox.*

Para asociar un control *Label* a otro control, establezca la propiedad *Target* en el control que debe obtener el foco cuando el usuario presione la tecla de acceso (lo veremos en el siguiente apartado).

Para establecer la tecla de acceso, agregue un guion bajo antes del carácter que debe ser la tecla de acceso. Se accede con Alt + la letra seleccionada.

```
<TextBox Name="txtBox2" …/>
<Label Target="{Binding ElementName=txtBox2}"/>
        <AccessText …>E_ditar</AccessText>
</Label>
<TextBox Name="txtBox3" …/>
```

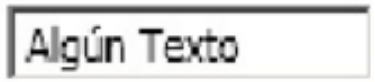


También se puede hacer con <AccessText Text="_Guardar"/> o con Label Content="_Guardar" Target….

Lenguajes de descripción de interfaces basados en XML. XAML. **Controles: TextBox y TextBlock**

TextBox
Representa un control que se puede utilizar para mostrar o editar texto sin formato.

```
<TextBox Name="tbSelectSomeText" Width="100" >
        Algún Texto</TextBox>
```



TextBlock
Proporciona un control ligero que permite mostrar pequeños fragmentos de contenido dinámico. TextBlock se ha diseñado para que sea ligero y se ha preparado específicamente para integrar pequeños fragmentos de contenido dinámico en una interfaz de usuario (UI). TextBlock proporciona un rendimiento óptimo en la presentación de líneas únicas y un rendimiento apropiado en la presentación de hasta unas pocas líneas de contenido.

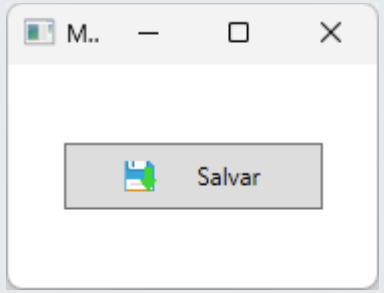
```
<TextBlock Name="textBlock1" TextWrapping="Wrap" FontSize="14">
        <Bold>TextBlock</Bold> esta diseñado para ser <Italic>liviano</Italic>
</TextBlock>
```

TextBlock esta diseñado para ser *liviano*

Lenguajes de descripción de interfaces basados en XML. XAML. **Controles: Button**

Button
Su propiedad de contenido es Content y su evento principal es Click.

En el siguiente ejemplo se muestra un botón que contiene una imagen y un texto formado por controles contenidos.




```
 <Button Width="129" Height="33">
    <StackPanel Orientation="Horizontal">
        <Image Source="/Save.ico" Width="16" Height="16" Margin="0,0,20,0"/>
        <TextBlock VerticalAlignment="Center">Salvar</TextBlock>
    </StackPanel>
</Button>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Controles: ListBox y ComboBox**

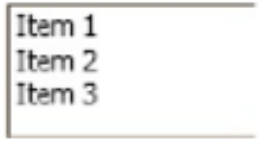
ListBox - ComboBox

El *ListBox* permite mostrar una lista de ítems desplegada. Las propiedades de contenido son las colecciones *Items* e *ItemsSource*. La propiedad *SelectionMode* permite establecer cuantos ítems pueden ser seleccionados. La opciones son Single (1 solo ítem), Multiple (varios ítems) y Extended (se pueden seleccionar varios ítems consecutivos usando la tecla Shift o no consecutivos usando la tecla Ctrl).

El *ComboBox* permite ocultar y desplegar la lista de ítems. Las propiedades *IsEditable* e *IsReadOnly* especifican cómo se comporta ComboBox cuando el usuario escribe una cadena para seleccionar un elemento o copia y pega un valor en el cuadro de texto.

En el siguiente ejemplo vemos el uso de un ListBox sencillo.

```
<ListBox Name="lb" Width="100" Height="55" SelectionChanged="PrintText"
SelectionMode="Single">
        <ListBoxItem>Item 1</ListBoxItem>
        <ListBoxItem>Item 2</ListBoxItem>
        <ListBoxItem>Item 3</ListBoxItem>
</ListBox>
```



Lenguajes de descripción de interfaces basados en XML. XAML. **Controles: CheckBox y RadioButton**

CheckBox
Representa un control que un usuario puede activar y desactivar. Pueden tener tres estados: activado, desactivado e indeterminado.

```
<CheckBox Name="Prueba" IsChecked="True">Prueba</CheckBox>
```

☑ Prueba

RadioButton
Representa un botón que el usuario puede seleccionar, pero no borrar. La propiedad *IsChecked* de *RadioButton* se puede establecer haciendo clic en él, pero sólo se puede borrar mediante programación. Normalmente se lo utiliza como elemento de un grupo de controles *RadioButton*. La selección está determinada por el estado de su propiedad *IsChecked*. Su evento principal es *Checked*.

```
<StackPanel Height="40"                                                    >
        <RadioButton Name="rb1" Checked="rb_Checked">Si</RadioButton>
        <RadioButton Name="rb2" Checked="rb_Checked">No</RadioButton>
        <RadioButton Name="rb3" Checked="rb_Checked">No sabe/no
        contesta</RadioButton>
</StackPanel>
```

○ Si
○ No
○ No sabe/no contesta

Nota: indeterminado significa que IsChecked=null

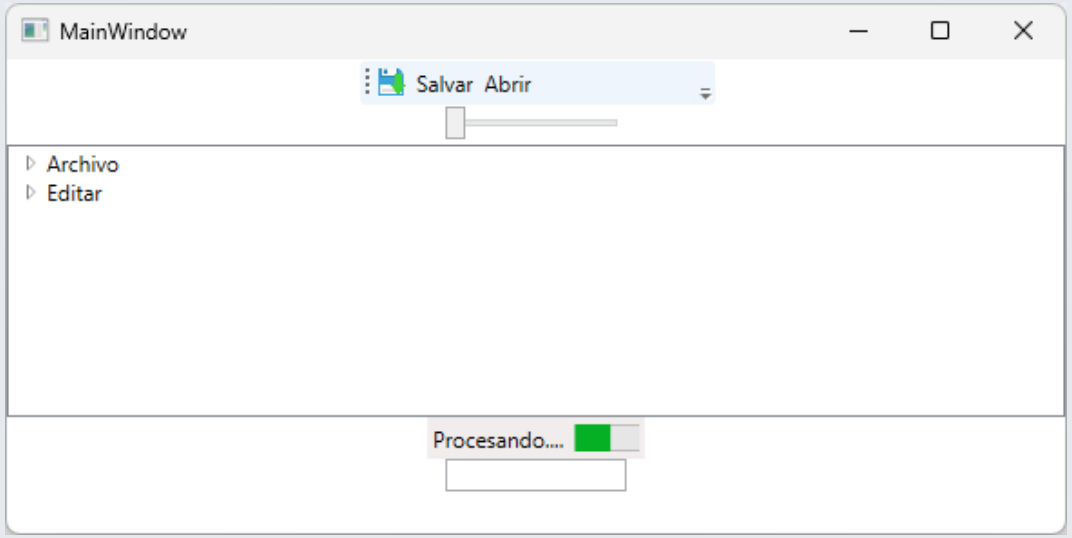
▣ **Indeterminate**

Se activa con IsThreeState=“True” y se usa cuando representa un conjunto de valores mixtos o selección parcial.

Lenguajes de descripción de interfaces basados en XML. XAML. **Controles avanzados.**

Otros Controles

- **ToolBar:** Identifica una barra de herramientas, como el control que contiene un conjunto de botones de comando en una ventana de la aplicación.
- **Slider:** Identifica un control deslizante.
- **TreeView:** Muestra datos jerárquicos, como una tabla de contenido, en una estructura de árbol.
- **StatusBar:** Identifica un control de barra de estado. Es solo un control visual, pero no es obligatorio. Las ProgressBar suelen colocarse dentro de este control.
- **ProgressBar:** Identifica un control de barra de progreso, que indica visualmente el progreso de una operación prolongada.
- **PasswordBox:** Representa un control diseñado para escribir y administrar las contraseñas.



Lenguajes de descripción de interfaces basados en XML. XAML. **Controles avanzados.**

```
<StackPanel>
    <ToolBar Height="26" Name="toolBar1" Width="200" >
        <Button>
            <Image Source="/Save.ico"/>
        </Button>
        <Button>Salvar</Button>
        <Button>Abrir</Button>
    </ToolBar>
    <Slider Height="21" Name="slider1" Width="100" />
    <TreeView Height="150">
        <TreeViewItem Header="Archivo">
            <TreeViewItem Header="Nuevo" />
            <TreeViewItem Header="Abrir" />
            <TreeViewItem Header="Cerrar" />
        </TreeViewItem>
        <TreeViewItem Header="Editar">
            <TreeViewItem Header="Cortar" />
            <TreeViewItem Header="Copiar" />
            <TreeViewItem Header="Pegar" />
        </TreeViewItem>
    </TreeView>
    <StatusBar Height="23" Name="statusBar1" Width="120" >
        <TextBlock Height="15" Text="Procesando...." />
        <ProgressBar Height="15" Name="progressBar1" Width="100" Value="20" />
    </StatusBar>
    <PasswordBox Name="pwdBox" Width="100" MaxLength="64" PasswordChar="*" />
</StackPanel>
```

Nota: dentro de StatusBar los componentes se pueden añadir directamente (como en el ejemplo) si la barra es simple. No necesitas alineación específica o se busca una definición rápida. O los componentes se pueden añadir con **StatusBarItem**, que se usaría si queremos que los elementos tengan alineación individual, respondan correctamente al redimensionado del StatusBar y si queremos que se organicen automáticamente según el estilo visual de Windows.

Lenguajes de descripción de interfaces basados en XML. XAML. **Controles avanzados: ToolBar.**

Si queremos mover una barra de herramientas podemos hacerlo usando <ToolBarTray>

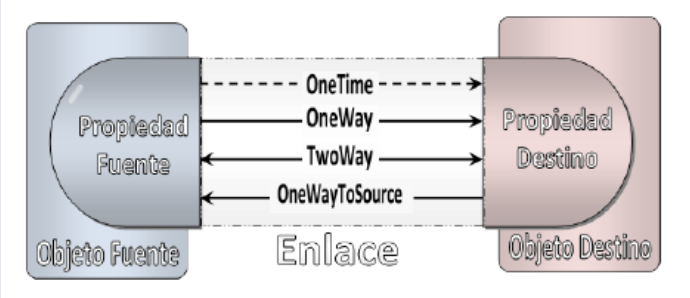
El usuario podrá arrastrar las toolbars dentro del ToolBarTray, horizontal o verticalmente.

Tambuién se puede controlar el movimiento con código.

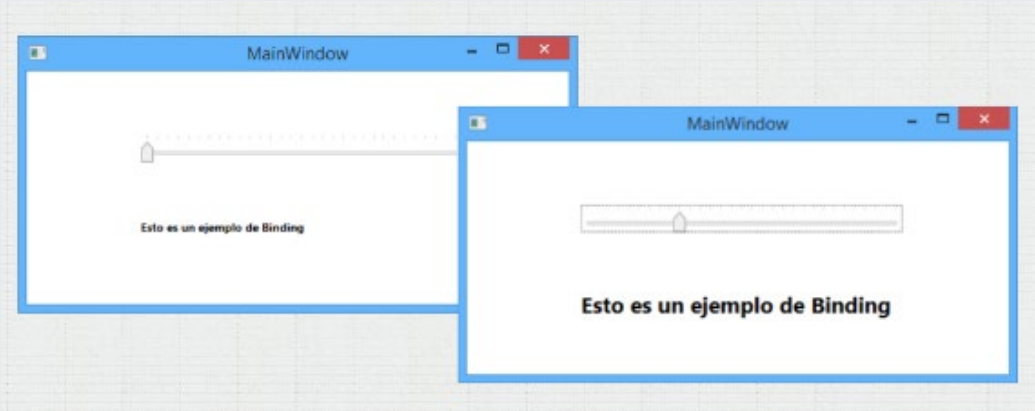
```
<ToolBarTray>
        <ToolBar Header="Archivo">
            <Button Content="Nuevo"/>
            <Button Content="Abrir"/>
        </ToolBar>

        <ToolBar Header="Editar">
            <Button Content="Cortar"/>
            <Button Content="Pegar"/>
        </ToolBar>
    </ToolBarTray>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos**

El enlace de datos es un mecanismo que establece una conexión entre la interfaz gráfica de usuario de una aplicación y los datos proporcionados por objetos pertenecientes a la lógica de negocio de dicha aplicación.



Mode especifica la dirección en la que fluyen los datos entre origen y destino.



Path indica la ruta dentro del objeto de origen a la propiedad que se quiere usar

```
<Slider Name="sldMiSlide" …>
```

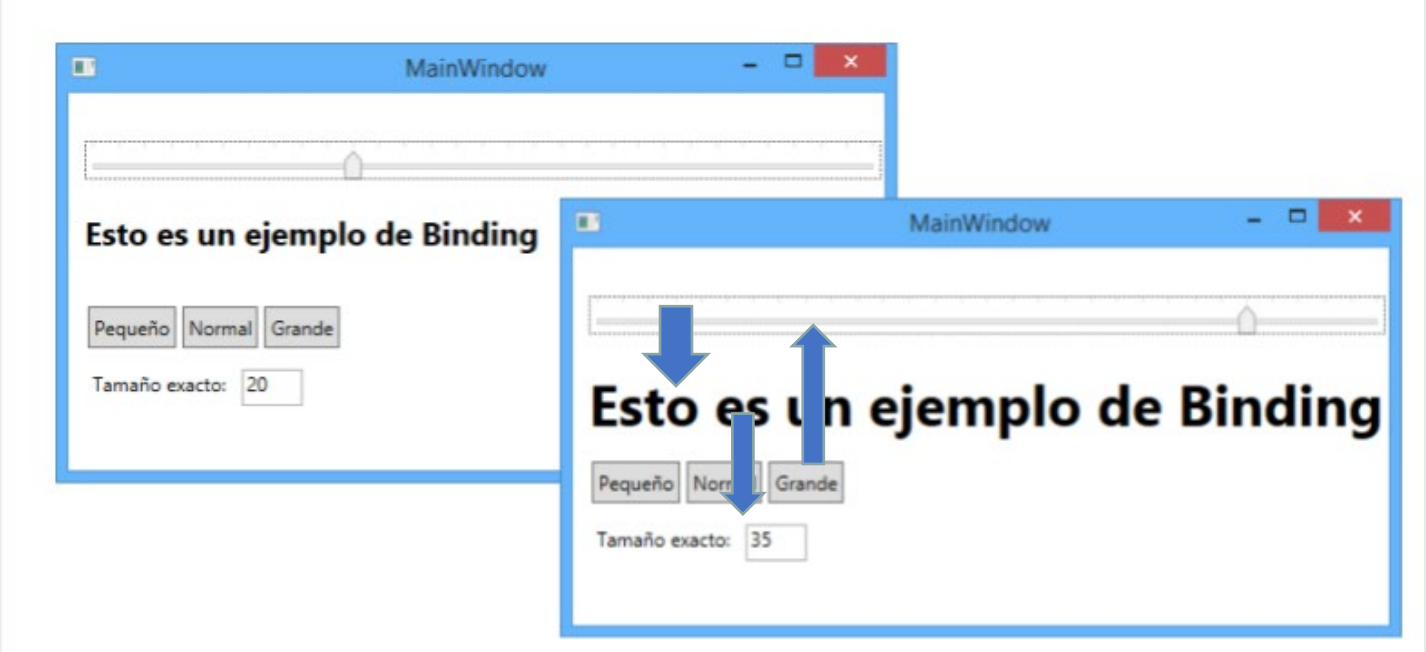
```
<TextBlock Name="txbEjemplo"
   FontSize="{Binding ElementName=MiSlide,Path=Value,Mode=OneWay}"
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos**

Valor de Mode	**Dirección del enlace**	**Descripción**
OneWay	◇ Fuente ➜ Destino	El valor fluye solo desde el *DataContext* (fuente) hacia el control (destino).
TwoWay	◇ Fuente ⇄ Destino	El valor se sincroniza en ambos sentidos. Si cambia en la UI, cambia en el modelo, y viceversa.
OneTime	◇ Fuente ➜ Destino (solo una vez)	Copia el valor una vez, al cargar el control. No se actualiza si el valor cambia después.
OneWayToSource	◇ Fuente ⇦ Destino	Flujo inverso: los cambios en el control actualizan el objeto fuente, pero no al revés.
Default	Depende de la propiedad	Usa el modo predeterminado según el tipo de propiedad del control.

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos** – Ejemplo 1

Hacer el data binding en ambos sentidos



Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos** – Ejemplo 1

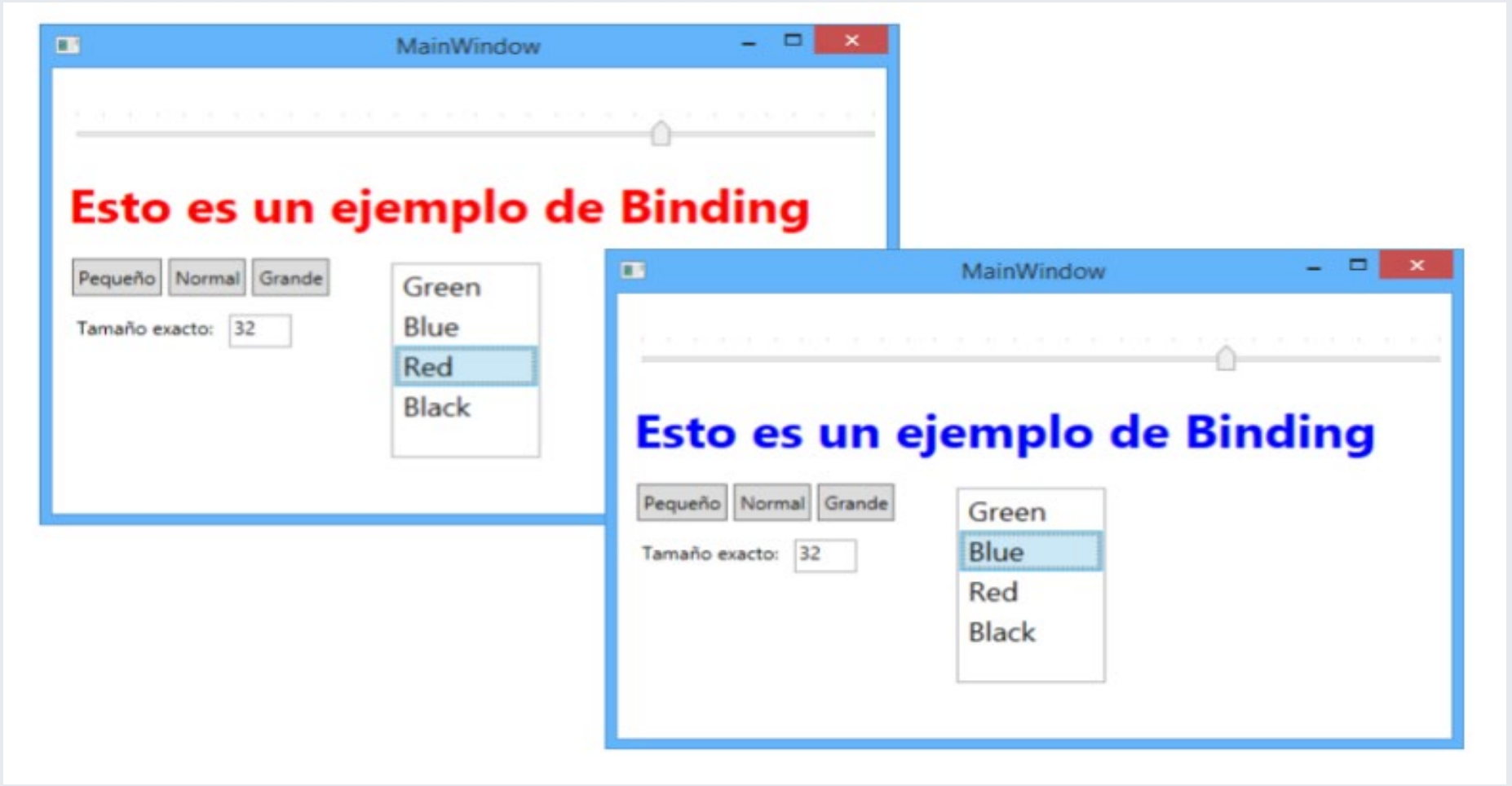
```
<WrapPanel Orientation="Vertical" HorizontalAlignment="Center">
       <Slider Name="sldMiSlider" Width="Auto" Margin="0,20,0,0"
               Minimum="10" Maximum="23"
               TickFrequency="1" IsSnapToTickEnabled="True"/>
       <TextBlock Name="txbTexto" Margin="0,20,0,0" HorizontalAlignment="Center"
                  FontSize="{Binding ElementName=sldMiSlider,Path=Value,Mode=TwoWay}">
           Esto es un ejemplo de Binding</TextBlock>
       <WrapPanel Margin="0,20,0,0" Orientation="Horizontal">
           <Button Name="btnPequeno" Width="100" Margin="10,0,0,0" Click="btnPequeno_Click">Pequeño</Button>
           <Button Name="btnMediano" Width="100" Margin="10,0,0,0" Click="btnMediano_Click">Normal</Button>
           <Button Name="btnGrande" Width="100" Margin="10,0,10,0" Click="btnGrande_Click">Grande</Button>
       </WrapPanel>
       <WrapPanel Margin="0,20,0,0" HorizontalAlignment="Center">
           <Label>Tamaño exacto:</Label>
           <TextBox Width="50" Text="{Binding ElementName=txbTexto,Path=FontSize,Mode=TwoWay}"
                    VerticalAlignment="Center"/>
       </WrapPanel>
   </WrapPanel>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos** – Ejemplo 1

Las pulsaciones de los botones no se pueden detectar con Data Binding: hay que capturar el evento.

```
public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void btnPequeno_Click(object sender, RoutedEventArgs e)
        {
            this.sldMiSlider.Value = 10;
        }

        private void btnMediano_Click(object sender, RoutedEventArgs e)
        {
            this.sldMiSlider.Value = 17;
        }

        private void btnGrande_Click(object sender, RoutedEventArgs e)
        {
            this.sldMiSlider.Value = 23;
        }
    }
```

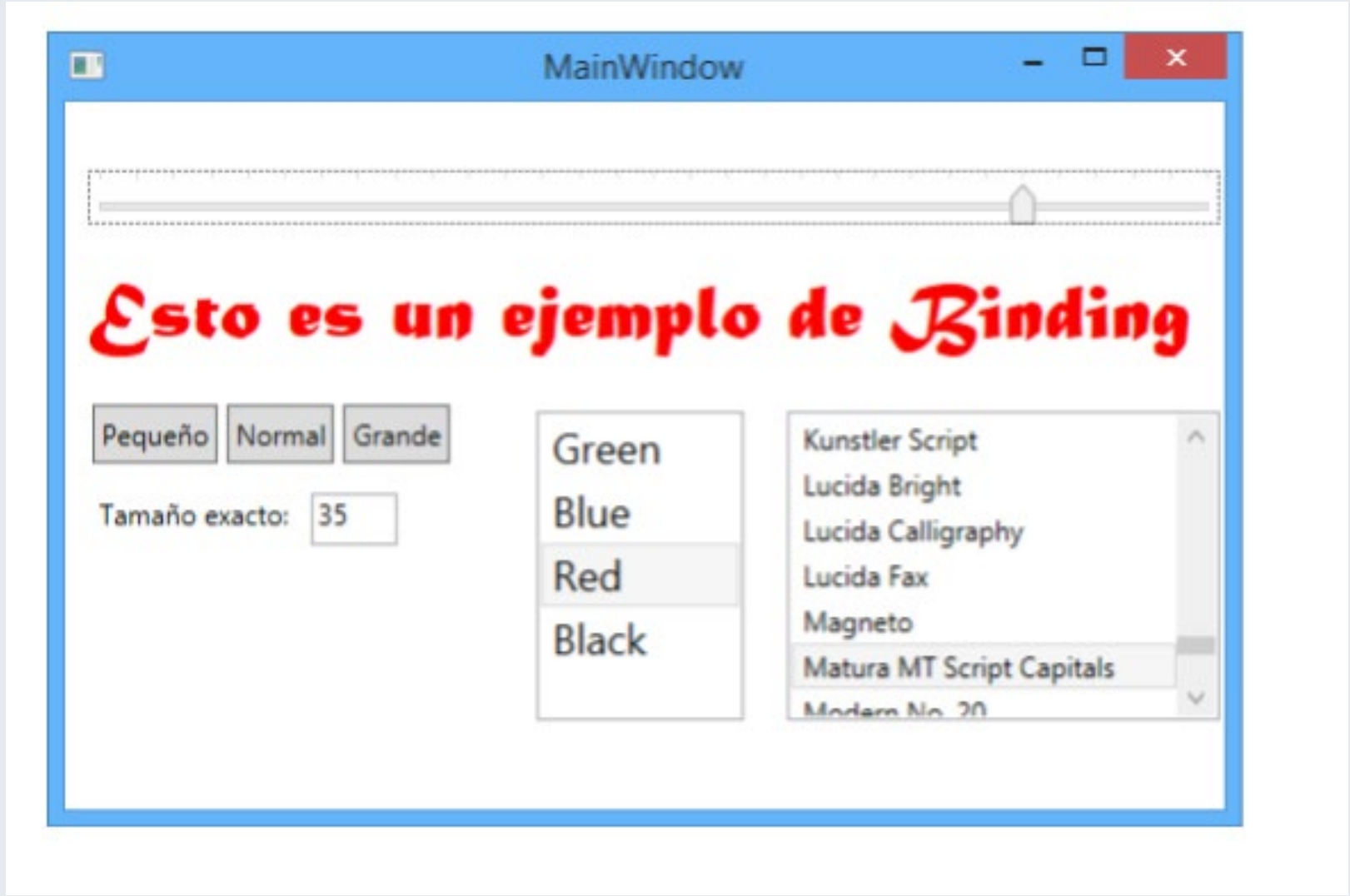
Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos** – Ejemplo 2



Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos** – Ejemplo 2

```
<StackPanel Margin="10,0,10,0">
       <Slider Name="sldMiSlider" Width="Auto" Margin="0,20,0,0"
               Minimum="10" Maximum="30"
               TickFrequency="1" IsSnapToTickEnabled="True"/>
       <TextBlock Name="txbTexto" Margin="0,0,0,0" Height="40" HorizontalAlignment="Center"
                  FontSize="{Binding ElementName=sldMiSlider,Path=Value,Mode=TwoWay}"
                  Foreground="{Binding ElementName=lstColor,Path=SelectedItem.Content}">
           Esto es un ejemplo de Binding</TextBlock>
       <WrapPanel Orientation="Horizontal">
           <WrapPanel Orientation="Vertical">
               <WrapPanel Margin="0,20,0,0" Orientation="Horizontal" HorizontalAlignment="Left">
                   <Button Name="btnPequeno" Width="50" Height="20"
                           VerticalAlignment="Top"  Margin="10,0,0,0"
                           Click="btnPequeno_Click">Pequeño</Button>
                   <Button Name="btnMediano" Width="50" Height="20"
                           VerticalAlignment="Top" Margin="10,0,0,0"
                           Click="btnMediano_Click">Normal</Button>
                   <Button Name="btnGrande" Width="50" Height="20"
                           VerticalAlignment="Top" Margin="10,0,10,0"
                           Click="btnGrande_Click">Grande</Button>
               </WrapPanel>
               <WrapPanel Margin="30,10,0,0" HorizontalAlignment="Left">
                   <Label>Tamaño exacto:</Label>
                   <TextBox Width="30" VerticalAlignment="Center"
                            Text="{Binding ElementName=txbTexto,Path=FontSize,Mode=TwoWay}"/>
               </WrapPanel>
           </WrapPanel>
           <ListBox Name="lstColor" Width="70" Margin="25,20,0,0">
               <ListBoxItem>Green</ListBoxItem>
               <ListBoxItem>Blue</ListBoxItem>
               <ListBoxItem>Red</ListBoxItem>
               <ListBoxItem>Black</ListBoxItem>
           </ListBox>
       </WrapPanel>
   </StackPanel>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos** – Ejemplo 3

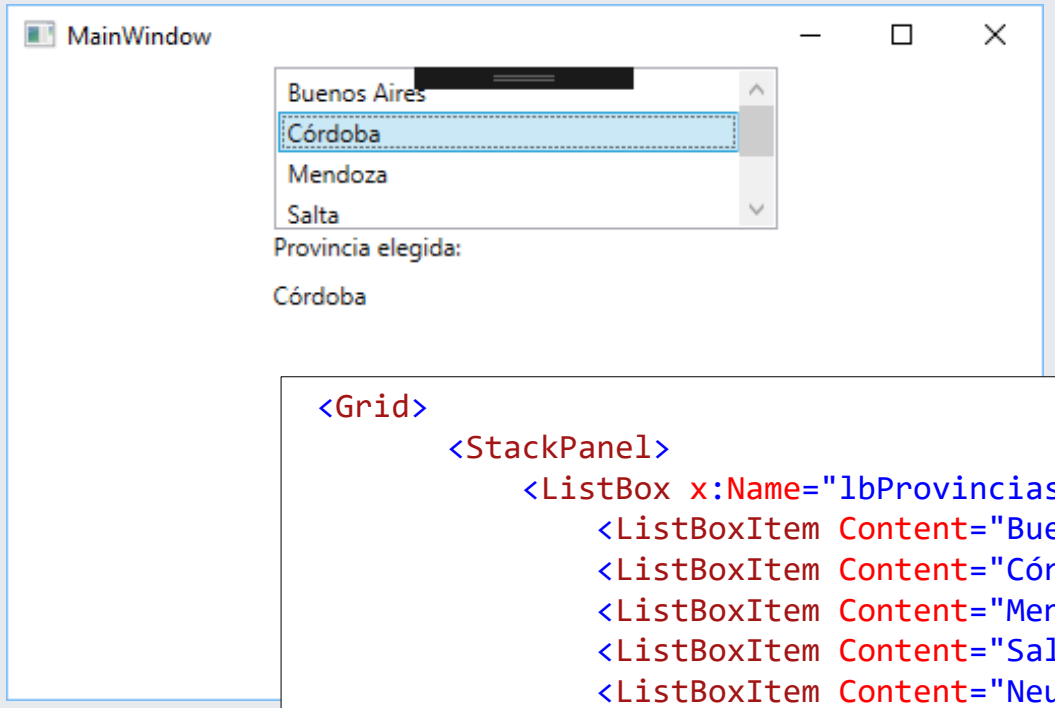


Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos** – Ejemplo 3

```
<TextBlock Name="LblEjemploTexto"
FontSize="{Binding ElementName=MiSlide, Path=Value,Mode=TwoWay}"
Foreground="{Binding ElementName=LstColores, Path=SelectedItem.Content}"
FontFamily="{Binding ElementName=LstFuentes, Path=SelectedItem}"
HorizontalAlignment="Left" Margin="10,73,0,0"
TextWrapping="Wrap" Text="Esto es un ejemplo de Binding"
VerticalAlignment="Top" Height="58" Width="497" FontWeight="Bold"/>
```

```
<ListBox Name="LstFuentes" HorizontalAlignment="Left" Height="136"
 Margin="317,136,0,0" VerticalAlignment="Top" Width="190"
ItemsSource="{x:Static Fonts.SystemFontFamilies}"/>
```

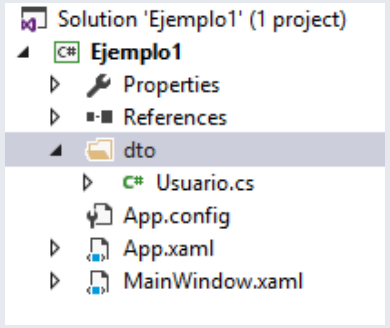
Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Enlazar datos** – Ejemplo 4




```
<Grid>
        <StackPanel>
            <ListBox x:Name="lbProvincias" Width="248" Height="80">
                <ListBoxItem Content="Buenos Aires"/>
                <ListBoxItem Content="Córdoba"/>
                <ListBoxItem Content="Mendoza"/>
                <ListBoxItem Content="Salta"/>
                <ListBoxItem Content="Neuquen"/>
            </ListBox>
            <TextBlock Width="248" Height="24" Text="Provincia elegida:" />
            <TextBlock Width="248" Height="24">
                <TextBlock.Text>
                    <Binding ElementName="lbProvincias" Path="SelectedItem.Content"/>
                </TextBlock.Text>
             </TextBlock>
        </StackPanel>
</Grid>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding -** Ejemplo de **rellenado de un combobox**

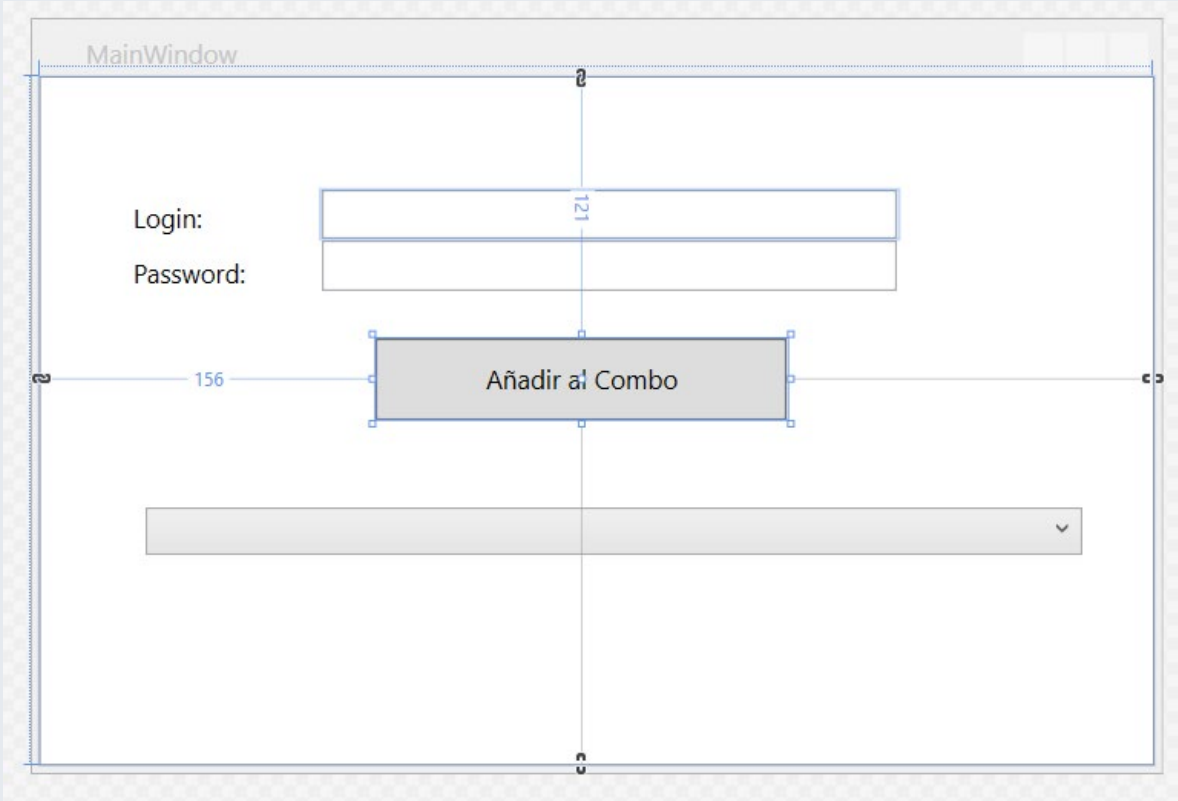
1.a) Creamos la clase Usuario




```
namespace Ejemplo1.dto
{
    class Usuario
    {
        public string Login { get; set; } // Implementamos métodos getter y setter
        public string Passw { get; set; }
        // Constructor
        public Usuario(String login, String passw)
        {
            this.Login = login;
            this.Passw = passw;
        }

        //Sobrecargamos el método ToString
        public override string ToString()
        {
            return Login + " " + Passw;
        }


    }
}
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding -** Ejemplo de **rellenado de un combobox**

1.b) Creamos la interfaz



El campo password lo podemos crear con <PasswordBox>

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding -** Ejemplo de **rellenado de un combobox**

1.c) Damos funcionalidad al botón

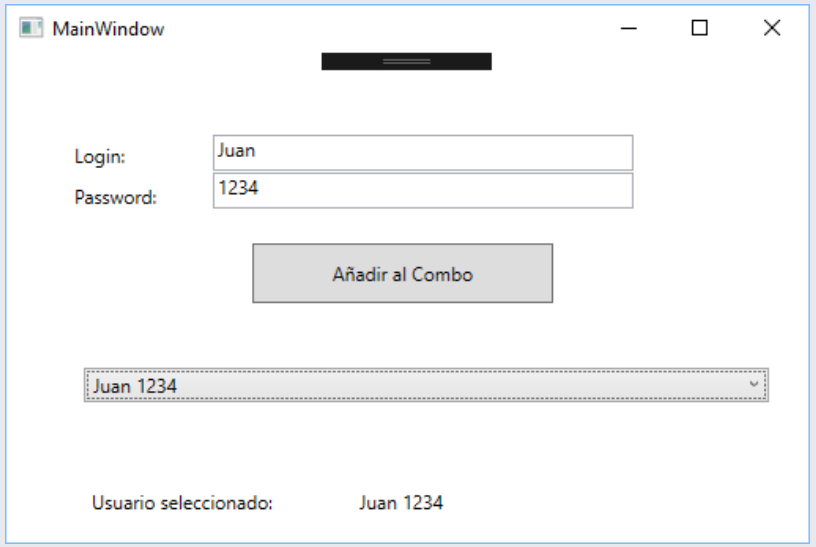
```
using System.Windows.Shapes;
using Ejemplo1.dto;  // Hacemos un import de la case usuario

namespace Ejemplo1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void button_Click(object sender, RoutedEventArgs e)
        {
            Usuario usuario = new Usuario(txtLogin.Text, txtPassw.Text);

            cmbUsuarios.Items.Add(usuario.ToString());
        }
    }
}
```

Si se ha usado <PasswordBox> se debe leer con txtPassw.Password

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding -** Ejemplo de **rellenado de un combobox**

2.a) Seleccionamos el evento SelectionChanged en el ComboBox e implementamos el código

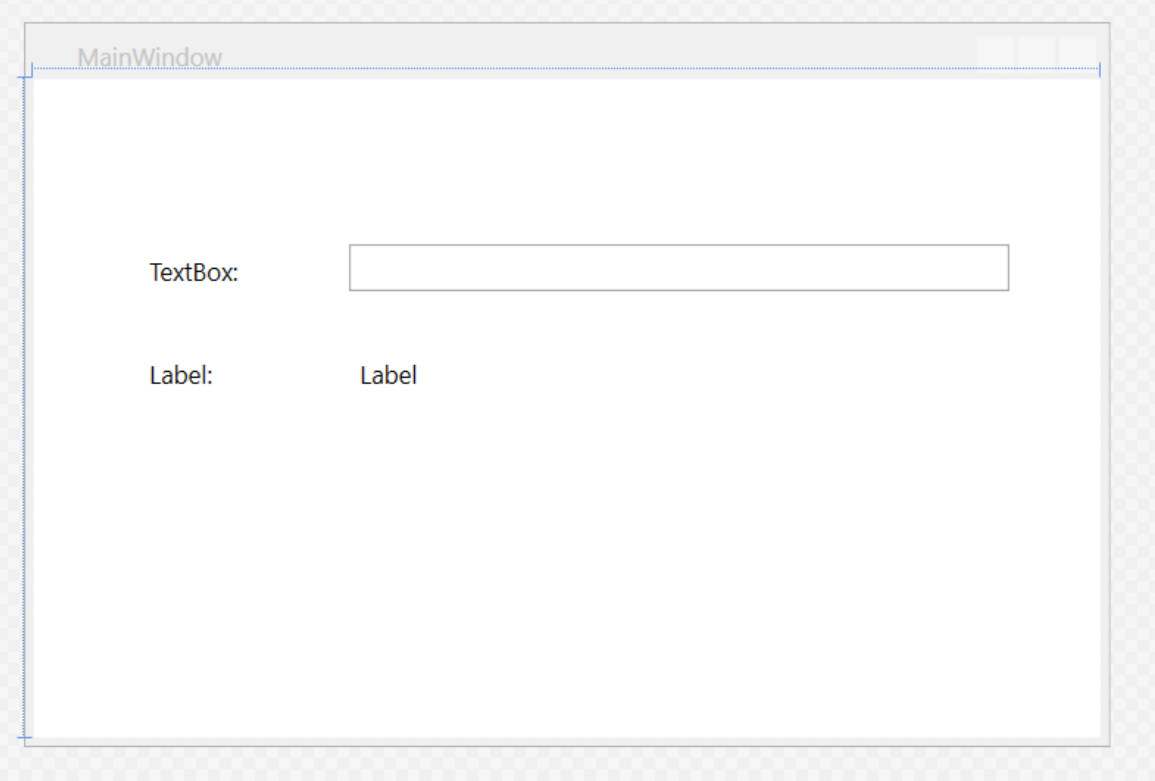
```
lblUsuario.Content = cmbUsuarios.SelectedItem.ToString();
```



O usamos Data Binding para hacerlo:

```
<Label Name="lblUsuario" Content="{Binding ElementName=cmbUsuarios,Path=SelectedItem}"/>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding -** Ejemplo de **replicar un TextBox en un Label**

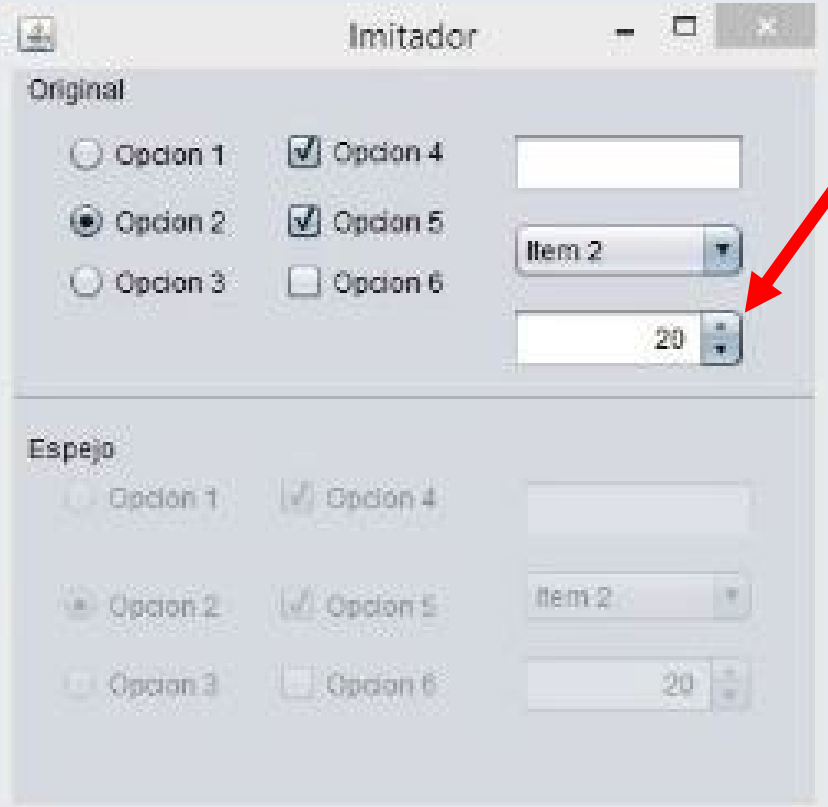



```
<Label x:Name="lblLabel" Content="{ Binding ElementName=txtBox, Path=Text}" …/>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding -** Ejemplo de **Espejo usando Binding**

- Vamos a crear un imitador, como si fuera un espejo.
- Tendremos dos pares de conjunto de elementos separados (puedes usar un separador) y cuando nosotros pinchamos en un elemento o escribimos en un campo, se debe cambiar el otro lado.
- Por ejemplo, si yo tengo un campo de texto y escribo en él, el campo de texto que es su reflejo también recibirá ese texto.
- Solo se puede modificar de un lado, el otro conjunto no se puede modificar, es decir, que no es bidireccional.

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding -** Ejemplo de **Espejo usando Binding**



Este control **no está por defecto**. Hay que instalarlo desde el Administrador de paquetes NuGet, y está en un paquete llamado **Extended.Wpf.Toolkit**

En el código XAML se incluye con:

```
<xctk:IntegerUpDown />
```

Y en la sección Windows hay que añadir:

```
xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding -** Ejemplo de **Espejo usando Binding**

Nota: en este caso he usado Grid con 9 filas y 3 columnas, pero se puede usar cualquier otro tipo de contenedor.

```
        <!-- ORIGINAL -->
        <Label Grid.Row="0" Grid.Column="0" FontWeight="Bold">Original</Label>

        <RadioButton Name="rdb1A" Grid.Row="1" Grid.Column="0" Content="Opción 1" GroupName="Original"
                     VerticalAlignment="Top" HorizontalAlignment="Center"/>
        <RadioButton Name="rdb2A" Grid.Row="2" Grid.Column="0" Content="Opción 2" GroupName="Original"
                     VerticalAlignment="Top" HorizontalAlignment="Center"/>
        <RadioButton Name="rdb3A" Grid.Row="3" Grid.Column="0" Content="Opción 3" GroupName="Original"
                     VerticalAlignment="Top" HorizontalAlignment="Center"/>

        <CheckBox Name="chk1A" Grid.Row="1" Grid.Column="1" Content="Opción 4" VerticalAlignment="Top" />
        <CheckBox Name="chk2A" Grid.Row="2" Grid.Column="1" Content="Opción 5" VerticalAlignment="Top" />
        <CheckBox Name="chk3A" Grid.Row="3" Grid.Column="1" Content="Opción 6" VerticalAlignment="Top" />

        <TextBox Name="txtA" Grid.Row="1" Grid.Column="2" Width="150" Height="25" VerticalAlignment="Top" HorizontalAlignment="Left"/>

        <ComboBox Name="cmbA" Grid.Row="2" Grid.Column="2" Width="150" Height="25" VerticalAlignment="Top" HorizontalAlignment="Left">
            <ComboBoxItem Content="Item 1"/>
            <ComboBoxItem Content="Item 2"/>
            <ComboBoxItem Content="Item 3"/>
        </ComboBox>

        <xctk:IntegerUpDown Name="intudA" Grid.Row="3" Grid.Column="2" Width="150" Height="25"
                    HorizontalAlignment="Left" VerticalAlignment="Top"/>

        <!-- SEPARADOR-->
        <Separator Grid.Row="4" Grid.ColumnSpan="3"/>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding -** Ejemplo de **Espejo usando Binding**

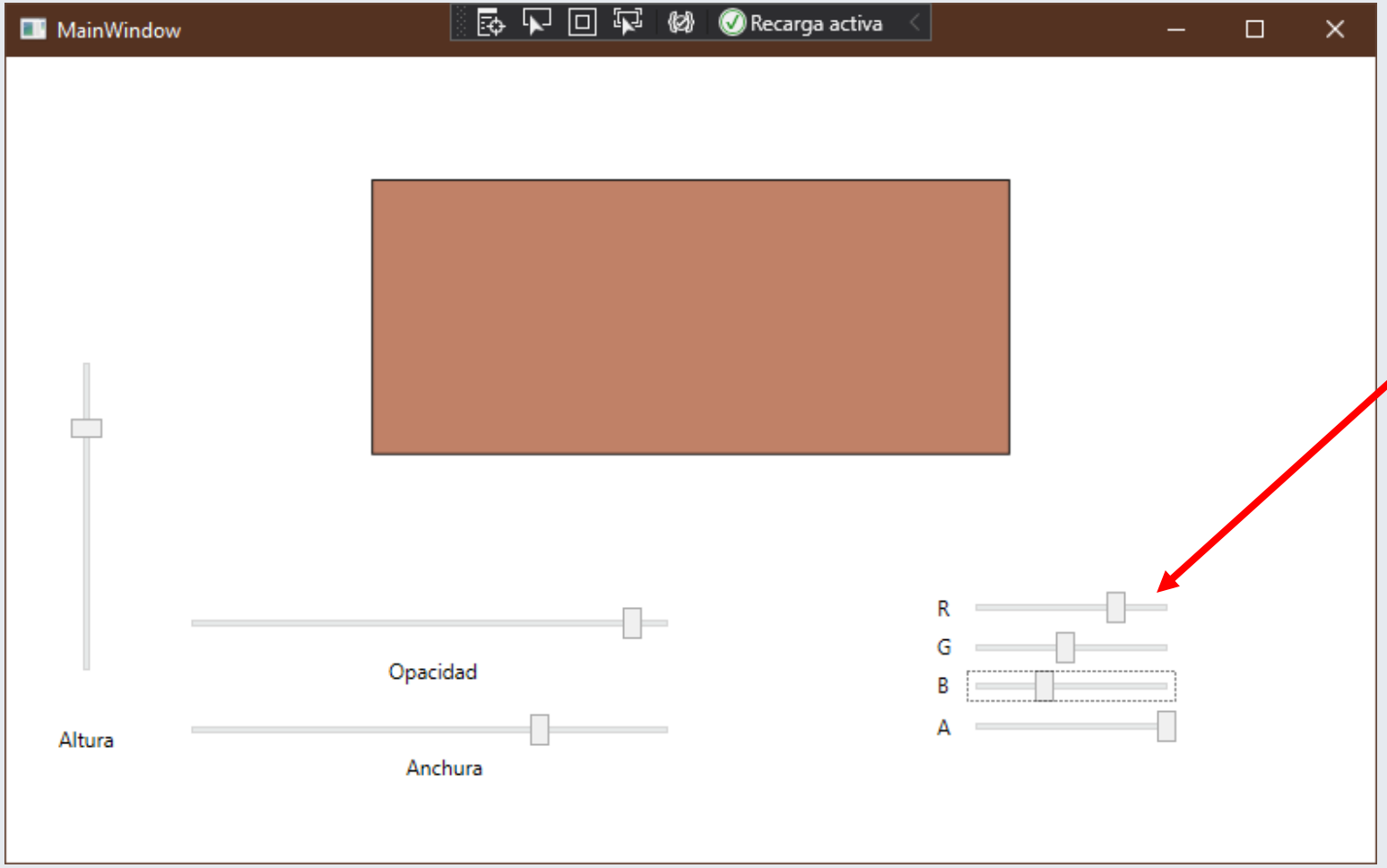
```
<!-- ESPEJO -->
        <Label Grid.Row="5" Grid.Column="0" FontWeight="Bold">Espejo</Label>

        <RadioButton Name="rdb1B" Grid.Row="6" Grid.Column="0" Content="Opción 1" GroupName="Espejo"
                     IsEnabled="False" VerticalAlignment="Top" HorizontalAlignment="Center"
                     IsChecked="{Binding ElementName=rdb1A,Path=IsChecked}"/>
        <RadioButton Name="rdb2B" Grid.Row="7" Grid.Column="0" Content="Opción 2" GroupName="Espejo"
                     IsEnabled="False" VerticalAlignment="Top" HorizontalAlignment="Center"
                     IsChecked="{Binding ElementName=rdb2A,Path=IsChecked}"/>
        <RadioButton Name="rdb3B" Grid.Row="8" Grid.Column="0" Content="Opción 3" GroupName="Espejo"
                     IsEnabled="False" VerticalAlignment="Top" HorizontalAlignment="Center"
                     IsChecked="{Binding ElementName=rdb3A,Path=IsChecked}"/>

        <CheckBox Name="chk1B" Grid.Row="6" Grid.Column="1" Content="Opción 4" VerticalAlignment="Top" IsEnabled="False"
                  IsChecked="{Binding ElementName=chk1A,Path=IsChecked}"/>
        <CheckBox Name="chk2B" Grid.Row="7" Grid.Column="1" Content="Opción 5" VerticalAlignment="Top" IsEnabled="False"
                  IsChecked="{Binding ElementName=chk2A,Path=IsChecked}"/>
        <CheckBox Name="chk3B" Grid.Row="8" Grid.Column="1" Content="Opción 6" VerticalAlignment="Top" IsEnabled="False"
                  IsChecked="{Binding ElementName=chk3A,Path=IsChecked}"/>

        <TextBox Name="txtB" Grid.Row="6" Grid.Column="2" Width="150" Height="25" IsEnabled="False" VerticalAlignment="Top" HorizontalAlignment="Left"
                 Text="{Binding ElementName=txtA,Path=Text}"/>

        <ComboBox Name="cmbB" Grid.Row="7" Grid.Column="2" Width="150" Height="25" IsEnabled="False" VerticalAlignment="Top" HorizontalAlignment="Left"
                  SelectedIndex="{Binding ElementName=cmbA,Path=SelectedIndex}">
            <ComboBoxItem Content="Item 1"/>
            <ComboBoxItem Content="Item 2"/>
            <ComboBoxItem Content="Item 3"/>
        </ComboBox>

        <xctk:IntegerUpDown Name="intudB" Grid.Row="8" Grid.Column="2" Width="150" Height="25"
                            HorizontalAlignment="Left" VerticalAlignment="Top" IsEnabled="False"
                            Value="{Binding ElementName=intudA,Path=Value}"/>
    </Grid>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Ejemplo de uso de Slider**

- Disponemos de un objeto rectángulo (rectangle) de color. También tenemos tres objetos Slider. Cada uno de ellos cambia la propiedad altura, ancho y opacidad del rectángulo.

- Le añadimos slider’s para establecer el color al rectángulo, y el brillo RGB y A(brillo) (valores entre 0 y 255) , pero no los implementamos.

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Ejemplo de uso de Slider**



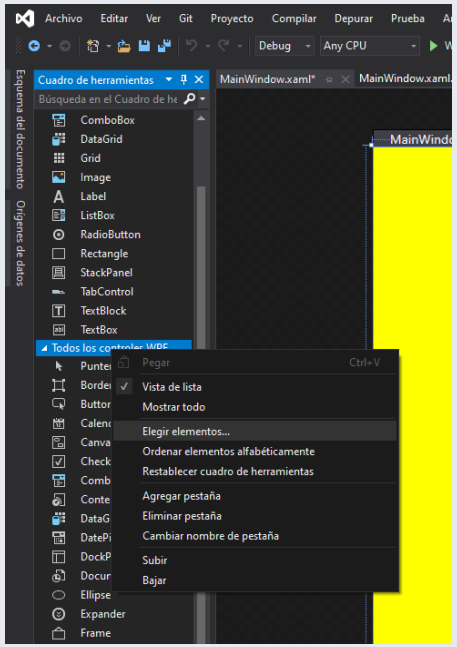
Esto lo dibujamos pero no lo vamos a implementar

Lenguajes de descripción de interfaces basados en XML. XAML. **Data Binding – Ejemplo de uso de Slider**

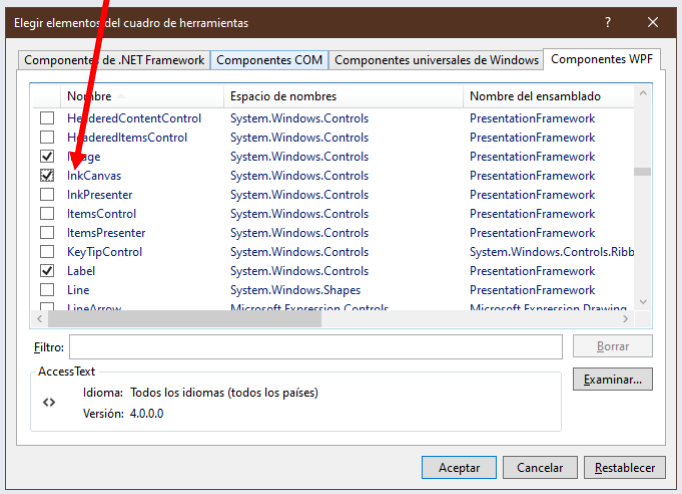
Nota: hay 2 final y 3 columnas con Grid

```
<Rectangle Grid.Row="0" Grid.Column="1" Fill="Brown"
           Height="{Binding ElementName=sldAltura,Path=Value}"
           Width="{Binding ElementName=sldAnchura,Path=Value}"
           Opacity="{Binding ElementName=sldOpacidad,Path=Value}"/>

<Grid Grid.Row="0" Grid.Column="0" Grid.RowSpan="2">
    <WrapPanel Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center">
        <Slider Name="sldAltura" HorizontalAlignment="Center" VerticalAlignment="Center"
                Orientation="Vertical" Height="200"
                Minimum="10" Maximum="100"/>
        <Label HorizontalAlignment="Center" VerticalAlignment="Center">Altura</Label>
    </WrapPanel>
</Grid>

<Grid Grid.Row="1" Grid.Column="1">
    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
        <Slider Name="sldOpacidad" HorizontalAlignment="Center" VerticalAlignment="Center"
                Width="200" Minimum="0" Maximum="1"/>
        <Label HorizontalAlignment="Center" VerticalAlignment="Center">Opacidad</Label>
        <Slider Name="sldAnchura" HorizontalAlignment="Center" VerticalAlignment="Center"
                Width="200" Minimum="10" Maximum="200"/>
        <Label HorizontalAlignment="Center" VerticalAlignment="Center">Anchura</Label>
    </StackPanel>
</Grid>
```

Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de uso de **InkCanvas**

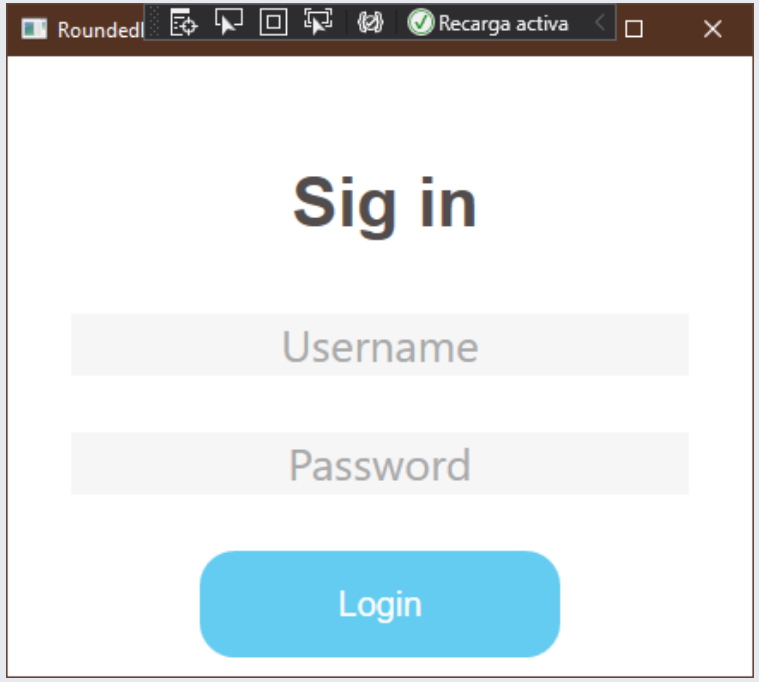


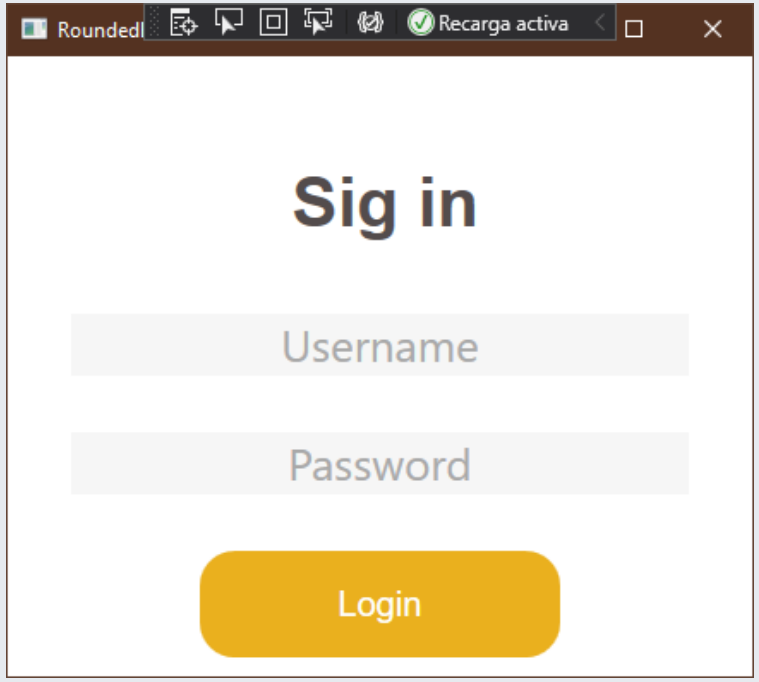
Esto en la versión actual de Visual Studio da un error y no lo añade




```
<StackPanel Background="Yellow">
        <InkCanvas Name="inkPizarra" Height="400" Width="250" VerticalAlignment="Top" HorizontalAlignment="Center" Margin="0,20,0,0"/>
        <WrapPanel HorizontalAlignment="Center" Margin="0,30,0,0">
            <Button Name="btnLimpiar" Content="Limpiar" Height="34" Width="100" Click="btnLimpiar_Click"/>
            <Button Name="btnSalir" Content="Salir" Height="34" Width="100" Margin="20,0,0,0" Click="btnSalir_Click"/>
        </WrapPanel>
    </StackPanel>
```

Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de **Formulario de Acceso**





Al colocar el puntero del ratón sobre el botón

Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de **Formulario de acceso**

```
<Window x:Class="RoundedButton_Juan.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:RoundedButton_Juan"
        mc:Ignorable="d"
        Title="RoundedButton" Height="400" Width="450" WindowStartupLocation="CenterScreen">
```

- **Manual:** si quieres colocar la ventana en una posición exacta (usando Left y Top).
- **CenterScreen:** para centrar la ventana principal de la app al iniciarse.
- **CenterOwner**: para diálogos o ventanas hijas que deben aparecer centradas respecto a la ventana que las abre.

Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de **Formulario de acceso**

```
 <StackPanel>
        <TextBlock ../>
        <TextBox ../>
        <TextBox ../> ó <PasswordBox ..>

<!-- Botón con estilo -->
        <Button Style="{StaticResource BotonRedondeado}"
                Content="Login" Width="170" Height="50" Margin="0,50,0,0"
                FontSize="16" FontWeight="SemiBold" Foreground="White" />
```

Style="{StaticResource BotonRedondeado}“: esto aplica un diseño para el botón llamado “BotonRedondeado”, que definiremos en una sección:

- **<Window.Resources>**: se aplica sólo a la ventana actual.
- **<Application.Resources>**: se aplica a toda la aplicación.

x:Key: define el nombre “BotonRedondeado”

Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de **Formulario de acceso**

- **Setter:** asigna un valor a una propiedad de un control.
- **Trigger:** cambia una propiedad de un control al ocurrir un evento.
- **ControlTemplate:** define como se dibuja visualmente un control.
- **TemplateBinding:** es una forma especial de hacer binding que hace que el se lea la propijedad gráfica de otro control. Por ejemplo, <Border Background="{TemplateBinding Background}“ asigna al background del border el backgraund con control que contiene al border.
- **ContentPresenter:** muestra el content del control.
- **TargetType:** aplica el estilo a todos los controles del tipo indicado.

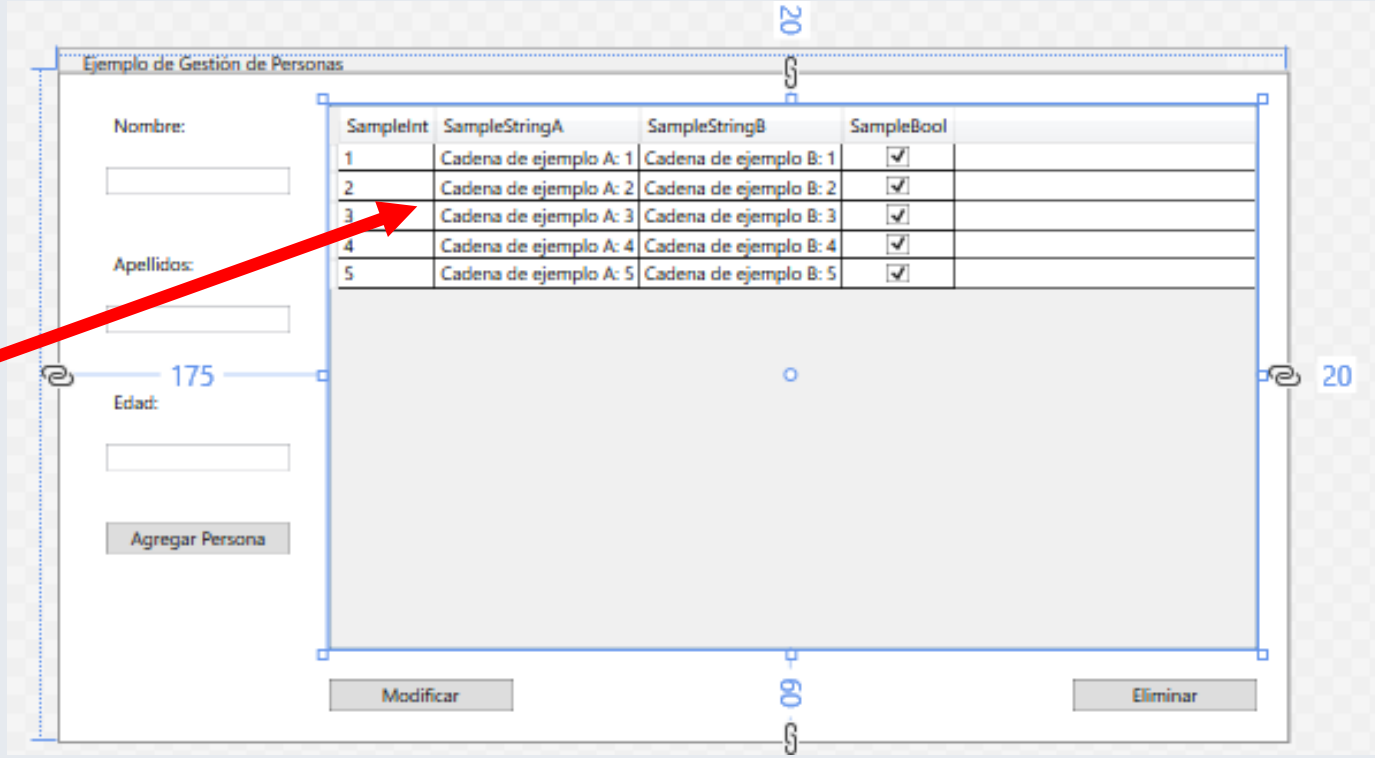
Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de **Formulario de acceso**

```
<Window.Resources>
        <Style x:Key="BotonRedondeado" TargetType="Button">
            <Setter Property="Background" Value="#4FC3F7"/>
            <Setter Property="BorderBrush" Value="Transparent"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}" CornerRadius="10">
                            <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="#F4B400"/>
                </Trigger>
                <Trigger Property="IsPressed" Value="true">
                    <Setter Property="Background" Value="#F1CA68"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>
```

Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

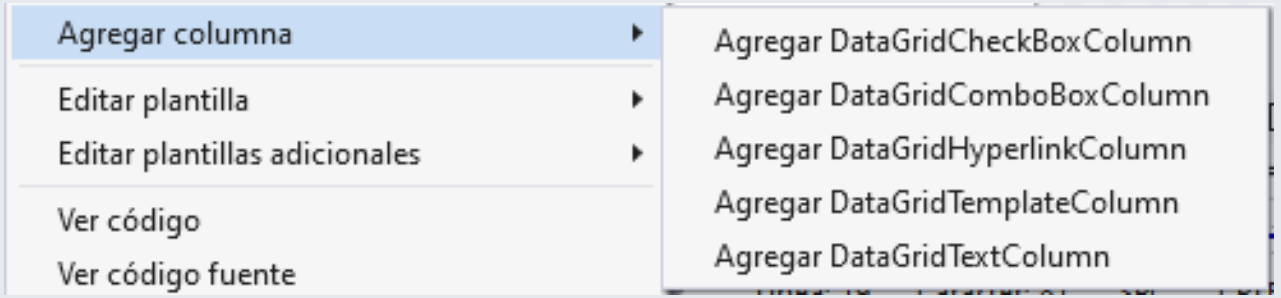
El siguiente ejemplo utilizará un control **DataGrid** que es una control que sirve para mostrar, editar y gestionar datos en forma de tabla. Es como un Excel avanzado.

Cuando se crea tiene datos de ejemplo.



Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

- Las columnas de la tabla se deben editar. Puede ser mediante la creación de un **enlace de datos** en el **ItemSource** que permite enlazar los datos con una consulta de base de datos, o cualquier otro origen de datos.
- Para modificar las columnas debemos pulsar con el botón derecho del ratón sobre el DataGrid, y elegir el tipo de control que se quiere utilizar en esta columna



Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

Los tipos de columnas de la tabla que se pueden seleccionar son:

- **DataGridCheckBoxColumn**: CheckBox que se utiliza para mostrar valores booleanos.
- **DataGridComboBoxColumn**: ComboBox usado para mostrar una elección de un listado.
- **DataGridHyperlinkColumn**: Hiperenlace que se usa para enlazar a una URL.
- **DataGridTemplateColumn**: Permite utilizar otros tipos de componentes específicos y personalizados.
- **DataGridTextColumn**: Cadena de texto.

Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

En XAML, en la etiqueta **DataGrid**, añadimos:

```
AutoGenerateColumns= "False“
```

y eliminaremos el enlace de datos de ejemplo:

```
d:ItemsSource="{d:SampleData ItemCount=5}“
```

AutogenerateColumns = False: permite definir manualmente las columnas que se mostrarán en el DataGrid. Permite definir su orden, los encabezados, el formato de los datos y si son editables o no. Si lo ponemos a True se genera todo automáticamente.

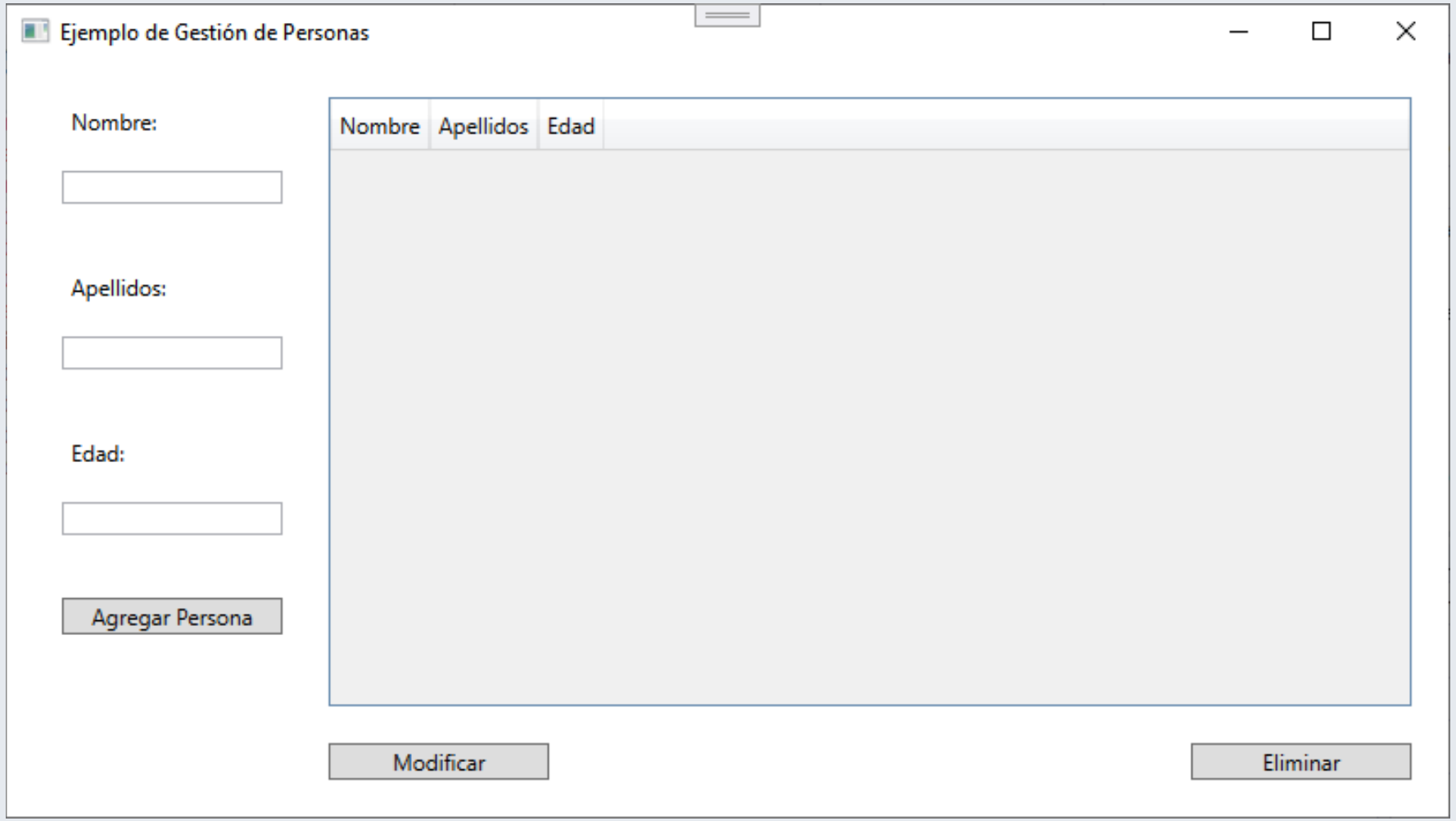
Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

Además agregaremos 3 columnas de tipo texto añadiendo un Binding:

```
<DataGrid.Columns>
    <DataGridTextColumn Header="Nombre" Binding="{Binding Nombre}" />
    <DataGridTextColumn Header="Apellidos" Binding="{Binding Apellidos}" />
    <DataGridTextColumn Header="Edad" Binding="{Binding Edad}" />
</DataGrid.Columns>
```

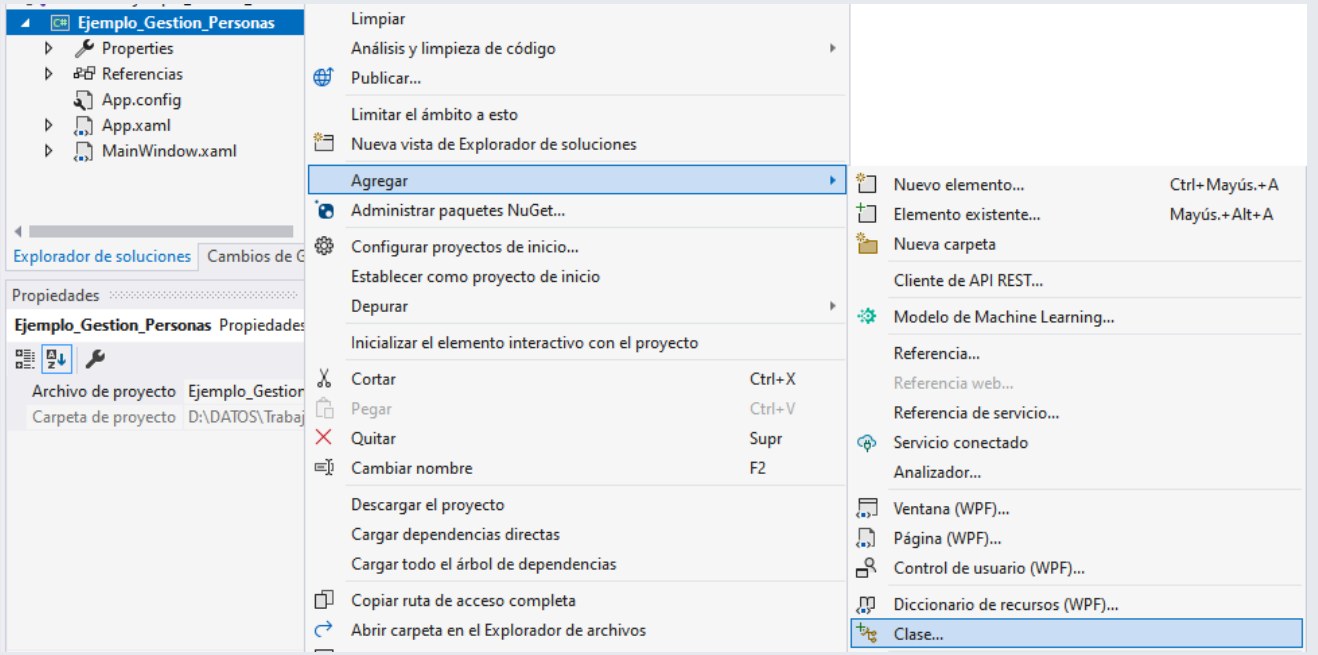
- Para que aparezca el encabezado de cada columna en la tabla se debe modificar en las propiedades, en la sección Encabezado, la propiedad Header.
- En las propiedades del DataGrid se pueden editar los atributos del encabezado como dimensiones de columnas

Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**



Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

- Para poder añadir elementos al DataGrid se deben realizar varias acciones preparatorias.
- El primer paso será crear una nueva clase que implemente los elementos que se van a agregar a la lista. En este caso una clase ”Persona”.
- Pulsando con el botón derecho sobre el proyecto seleccionamos **Agregar → Clase**



Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

- En la nueva clase se crearán los atributos privados correspondientes a las columnas de la tabla, en este caso nombre, apellidos y edad.

```
private String nombre;
private String apellidos;
private int edad;
```

- Para crear el constructor pulsamos con el botón derecho sobre el nombre de la clase y elegimos **Acciones rápidas y refactorizaciones…** y después en **Generar constructor a partir de miembros...**
- En la ventana que aparece, elegimos los campos que queremos pasar como parámetros.

Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

- Para cada atributo de la clase volvemos a elegir **Acciones rápidas y refactorizaciones…** y ahora elegimos **Encapsular campo: ---- (y usar propiedad).** Así generamos los Getters y Setters como propiedades accesibles desde el exterior. Si es necesario, se añadirán más miembros.

```
public Persona(string nombre, string apellidos, int edad) {
    this.nombre = nombre;
    this.apellidos = apellidos;
    this.edad = edad;
}

public string Nombre { get => nombre; set => nombre = value; }
public string Apellidos { get => apellidos; set => apellidos = value; }
public int Edad { get => edad; set => edad = value; }
```

Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

- Una vez que se tiene la clase, se creará una lista de objetos de este tipo. Será un miembro privado de la clase que contiene la aplicación.
- Representará el **Modelo** en el modelo **MVP** y será la que contendrá la lista de elementos del datagrid.

```
public partial class MainWindow : Window {
     private List<Persona> lstPersonas;
     public MainWindow(){
       InitializeComponent();
       lstPersonas = new List<Persona>();
       dgvPersonas.ItemsSource = lstPersonas;}
     private void btnAgregar_Click(object sender, RoutedEventArgs e){
       lstPersonas.Add(new Persona(txtNombre.Text, txtApellidos.Text,
                                   int.Parse(txtEdad.Text)));
       dgvPersonas.Items.Refresh();
  } }
```

Nota: *hay que comprobar que en los TextBox hay algo porque si están vacíos da un error*

Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

- El botón **Modificar** puede utilizarse para recoger los datos de la tabla. Posteriormente se puede añadir la funcionalidad de modificar desde los campos de texto originales. Para recoger los datos del DataGrid utilizamos este código:

```
private void btnModificar_Click(object sender, RoutedEventArgs e)
{
    Persona persElegida = (Persona) dgvPersonas.SelectedItem;
    txtNombre.Text = persElegida.Nombre;
    txtApellidos.Text = persElegida.Apellidos;
    txtEdad.Text = persElegida.Edad.ToString();
}
```

Nota: *hay que comprobar si se ha seleccionado algún dato (SelectedItem!=null) porque sino da un error*

Nota: Con este código no se llegan a modificar los datos del DataGrid, tan solo se copian al formulario. Esto queda como ejercicio.

Lenguajes de descripción de interfaces basados en XML. XAML. **DataGrid** - Ejemplo de **gestión de tabla de personas**

- En caso de modificar los datos desde la propia tabla, éstos se modificarían al mismo tiempo tanto en el datagrid como en el almacén de datos.
- El botón **Eliminar** puede utilizarse para eliminar registros de la lista y de la tabla.

```
private void btnEliminar_Click(object sender, RoutedEventArgs e)
{
   lstPersonas.Remove((Persona)dgvPersonas.SelectedItem);
   dgvPersonas.Items.Refresh();
}
```

Autoría:
Diego J. García
Rosa María Zapata Calle

Actualizado en 2025: Francisco J. Sánchez