

# UT2. Desarrollo de Interfaces mediante XAML

Desarrollo de Interfaces

# Índice

- Elaboración de interfaces de usuario
- Componentes
- Herramientas de elaboración de interfaces
- Contenedores
- Controles de la interfaz
- Asociación de acciones a eventos
- Diálogos modales y no modales
- Edición de código
- Clases, propiedades y métodos

## 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

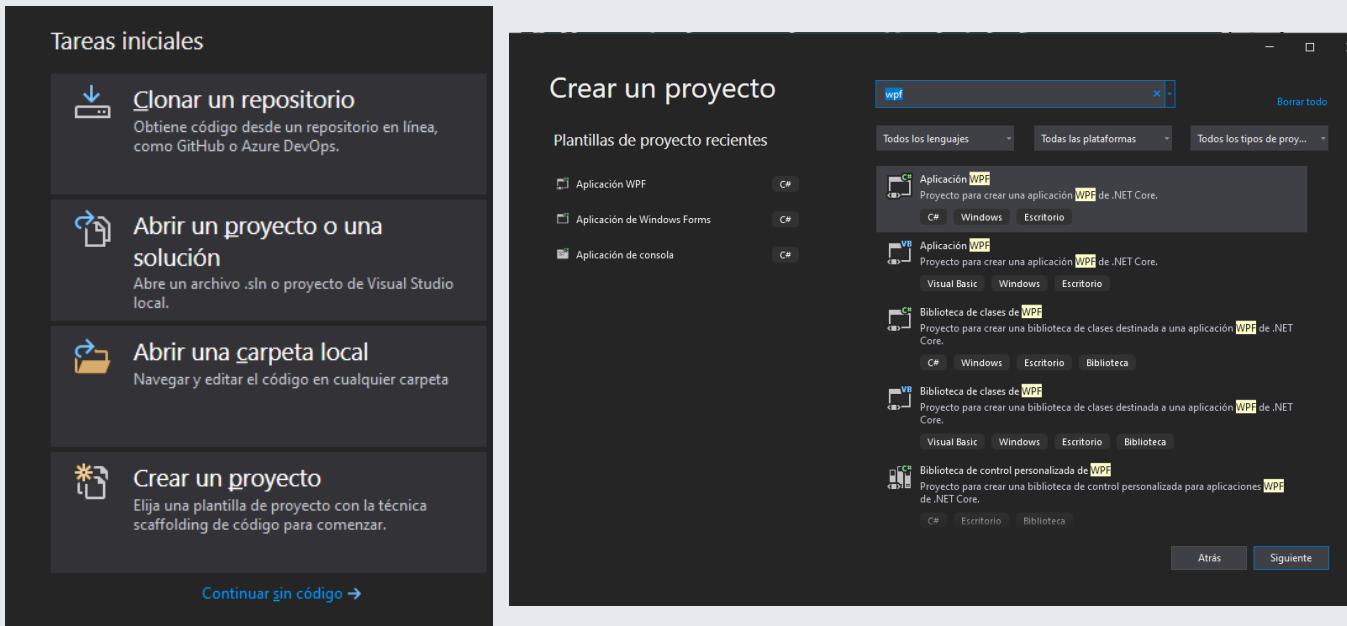
- **XAML:** “Xtensible Application Markup Language”
  - Microsoft
  - Basado en XML que define objetos y sus propiedades.
  - Sintaxis para definir la UI para la “Windows Presentation Foundation” (WPF)
  - Independencia del código de la aplicación.
    - Tutoriales: <http://www.scriptol.com/xaml/>

# 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.

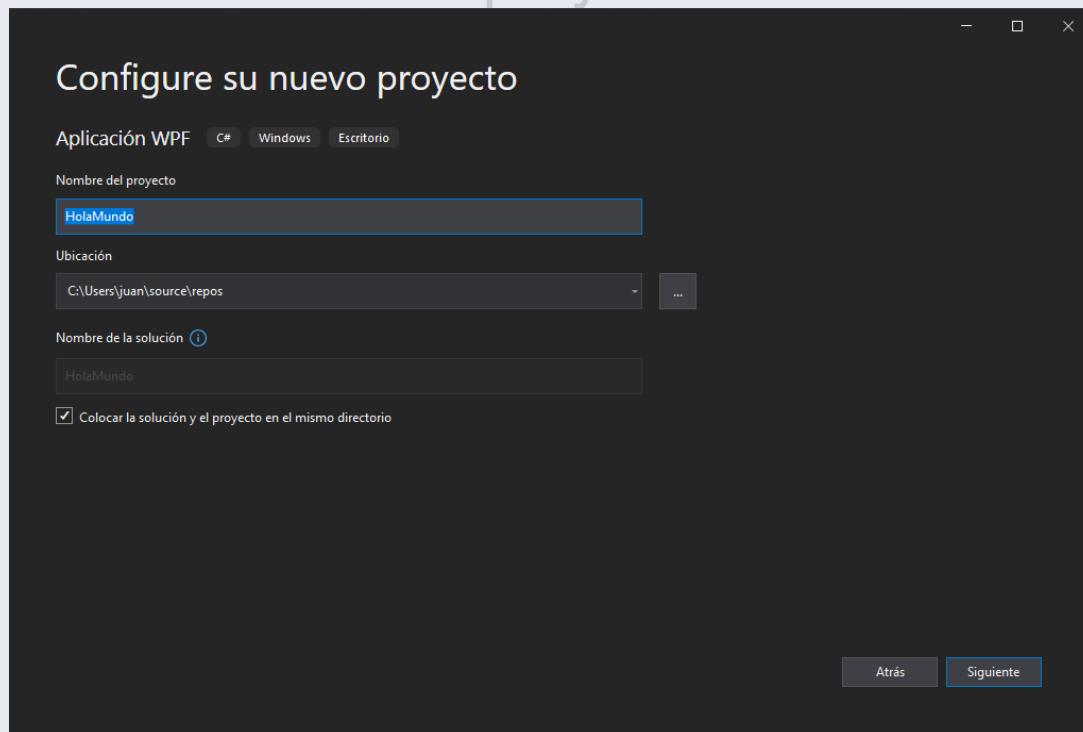
# Lenguajes de descripción de interfaces basados en XML. XAML

- Crear un nuevo proyecto WPF "Hola Mundo"



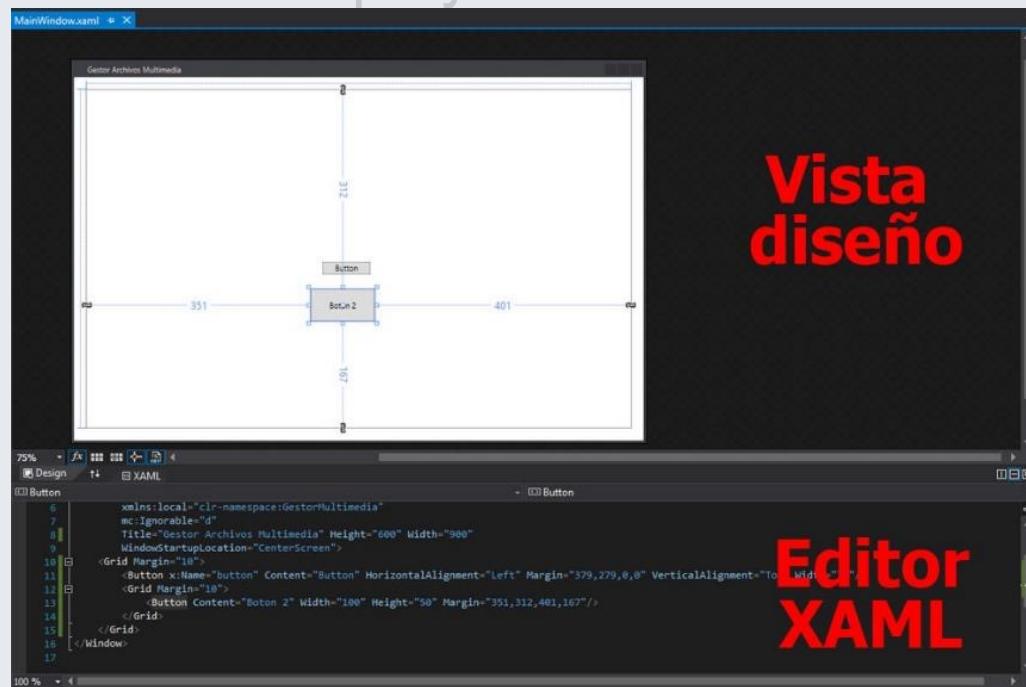
# Lenguajes de descripción de interfaces basados en XML. XAML

- Crear un nuevo proyecto WPF "Hola Mundo"



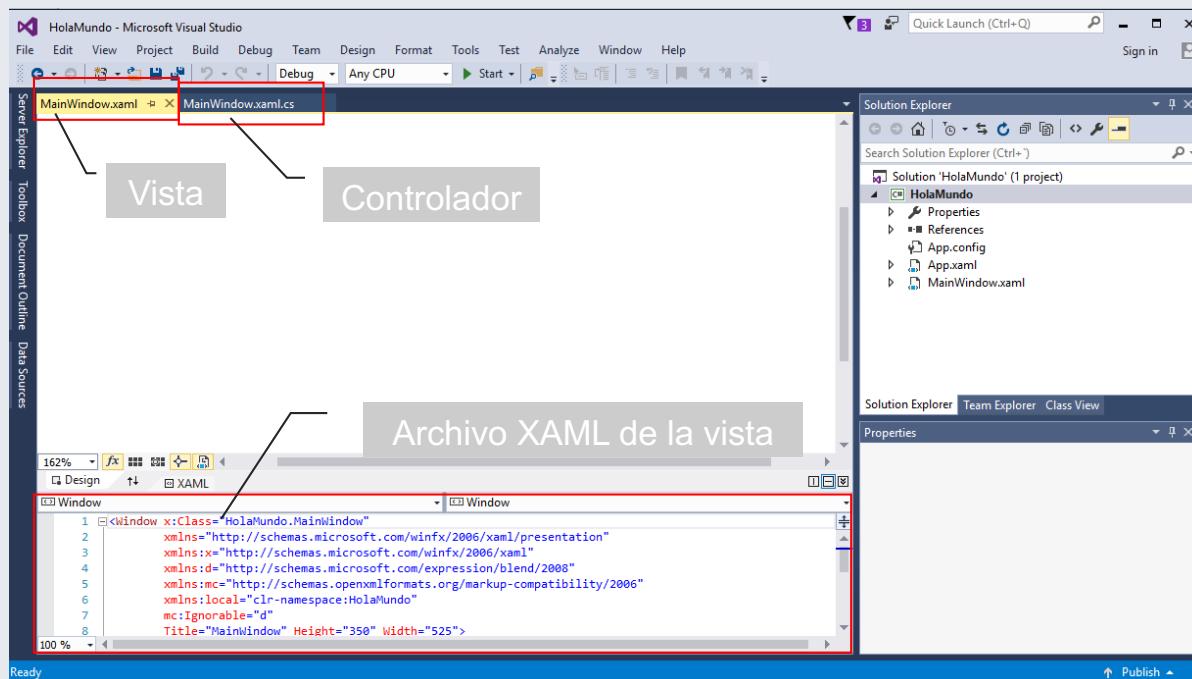
# Lenguajes de descripción de interfaces basados en XML. XAML

- 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"

The screenshot shows the Visual Studio IDE interface. On the left is the code editor displaying the `MainWindow.xaml.cs` file. On the right is the Properties window for the `btnSaludar` button.

**Properties Window (Properties) - btnSaludar**

Propiedad	Valor
Nombre	btnSaludar
Tipo	Button
Click	Button_Click
ContextMenuClose...	
ContextMenuOpen...	
DataContextChang...	
DragEnter	
DragLeave	
DragOver	
Drop	
FocusableChanged	
GiveFeedback	
GotFocus	

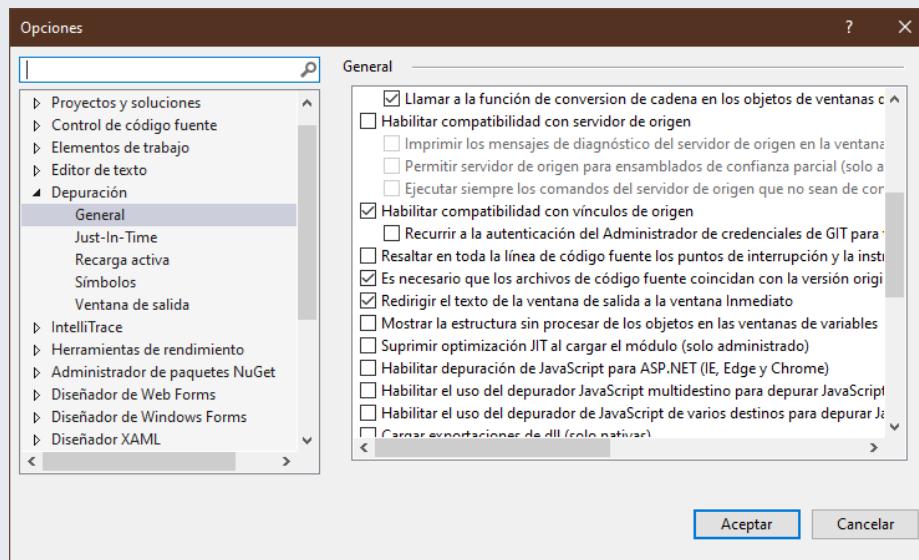
**Code Editor (MainWindow.xaml.cs)**

```
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

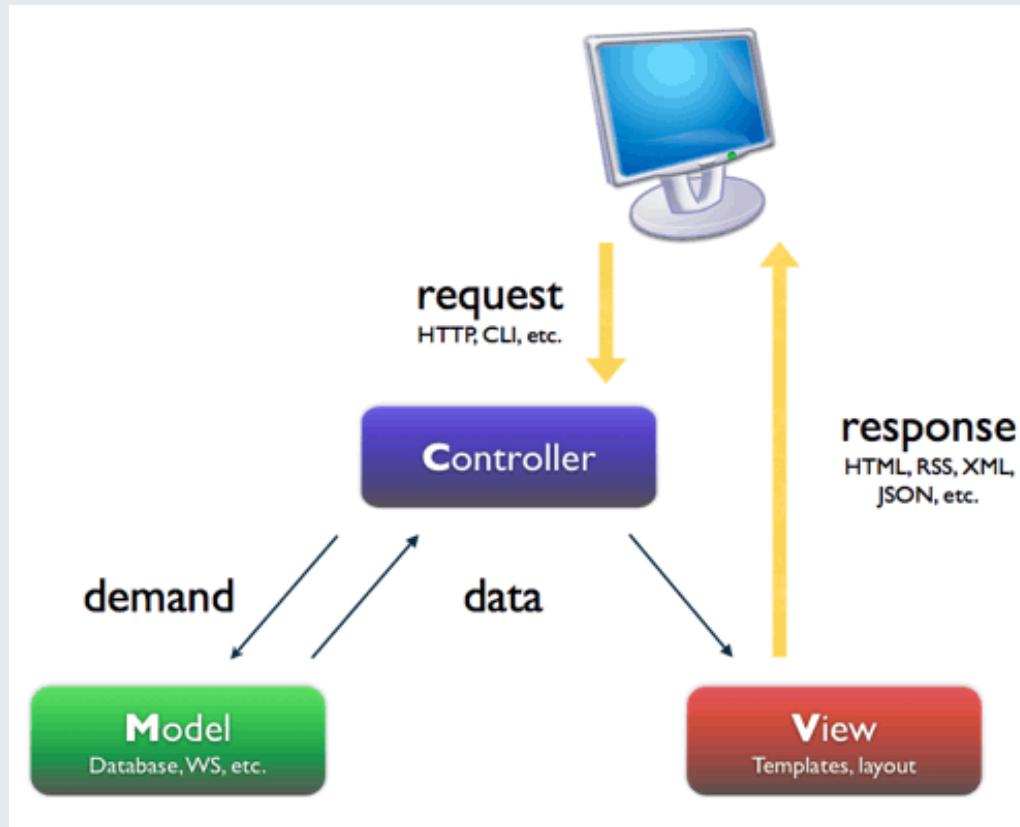
- Vistas del proyecto WPF "Hola Mundo",
  - Menú depurar □ Opciones
- Para conseguir mostrar los mensajes Debug.WriteLine("") en la ventana Inmediato,



## 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

The screenshot shows a Windows application window titled "Registro de alumnos". The interface includes the following fields:

- Nombre completo: A text input field.
- Edad: An empty text input field.
- Sexo: Radio buttons for "Masculino" (selected) and "Femenino".
- Curso: A dropdown menu currently showing "DAM".
- Horario: A dropdown menu currently showing "Nocturno".
- Checkboxes for "T.I." and "Otro Documento:" followed by a text input field.
- Buttons for "Registrar" and "Salir".

# 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

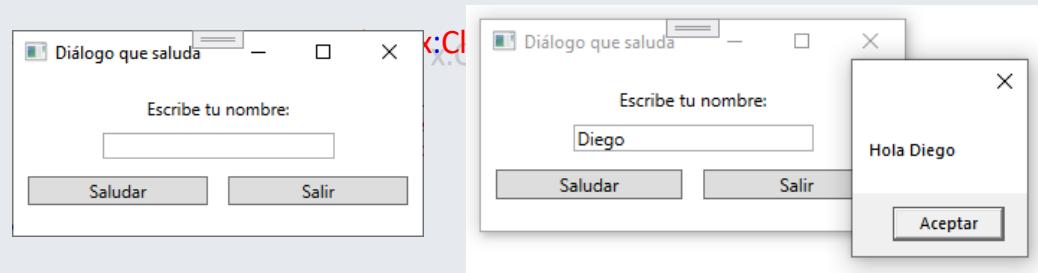
```
<ComboBox HorizontalAlignment="Left" Margin="135,132,0,0"
          VerticalAlignment="Top" Width="200">
    <Button Content="Diurno"/>
    <CheckBox Content="Vespertino"/>
    <RadioButton Content="Nocturno"/>
</ComboBox>
<CheckBox Content="T.I." HorizontalAlignment="Left"
          Margin="26,170,0,0" 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

**Registro de personas físicas**

Profesión	<input type="text"/>	Marque del 1 al 10 su grado de afición a:
Nº hermanos	<input type="text"/> Edad <input type="button" value="De 15 a 18"/>	Compras
Sexo	<input type="radio"/> Masculino <input type="radio"/> Femenino	Ver televisión
<input type="checkbox"/> Practica algún deporte ¿Cuál?		Ir al cine
<input type="listbox" value="Fútbol"/> Tenis Piragüismo		
<input type="button" value="Registrar"/> <input type="button" value="Salir"/>		

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



```
... 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"/>
    </Grid>
</Window>
```

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

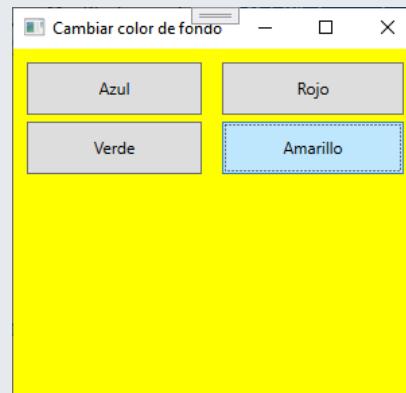
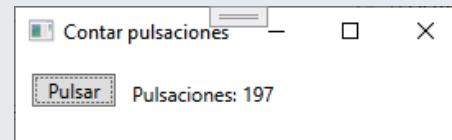
```
public partial class MainWindow : Window
{
    public MainWindow(){
        InitializeComponent();
    }

    private void btnSaludar_Click(object sender,
        RoutedEventArgs e){
        MessageBox.Show("Hola " + txtNombre.Text);
    }

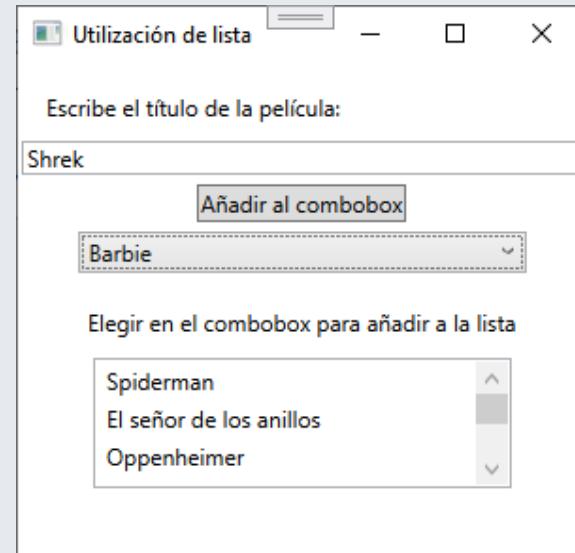
    private void btnSalir_Click(object sender,
        RoutedEventArgs e){
        App.Current.Shutdown();
    }
}
```

## 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

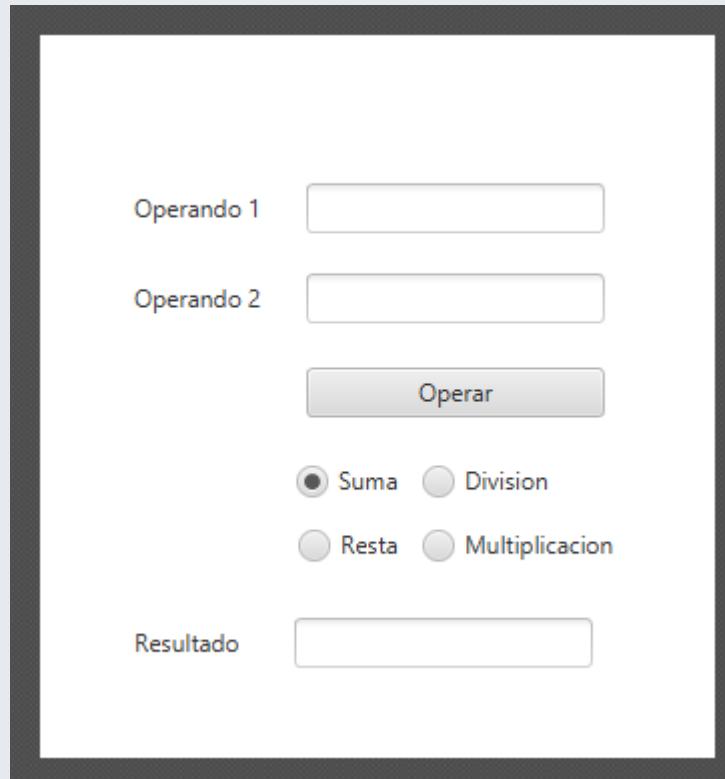


```
cmbPeliculas.Items.Add(txtPelicula.Text);
```

```
cmbPeliculas_SelectionChanged:
```

```
IstPeliculas.Items.Add(cmbPeliculas.SelectedValue);
```

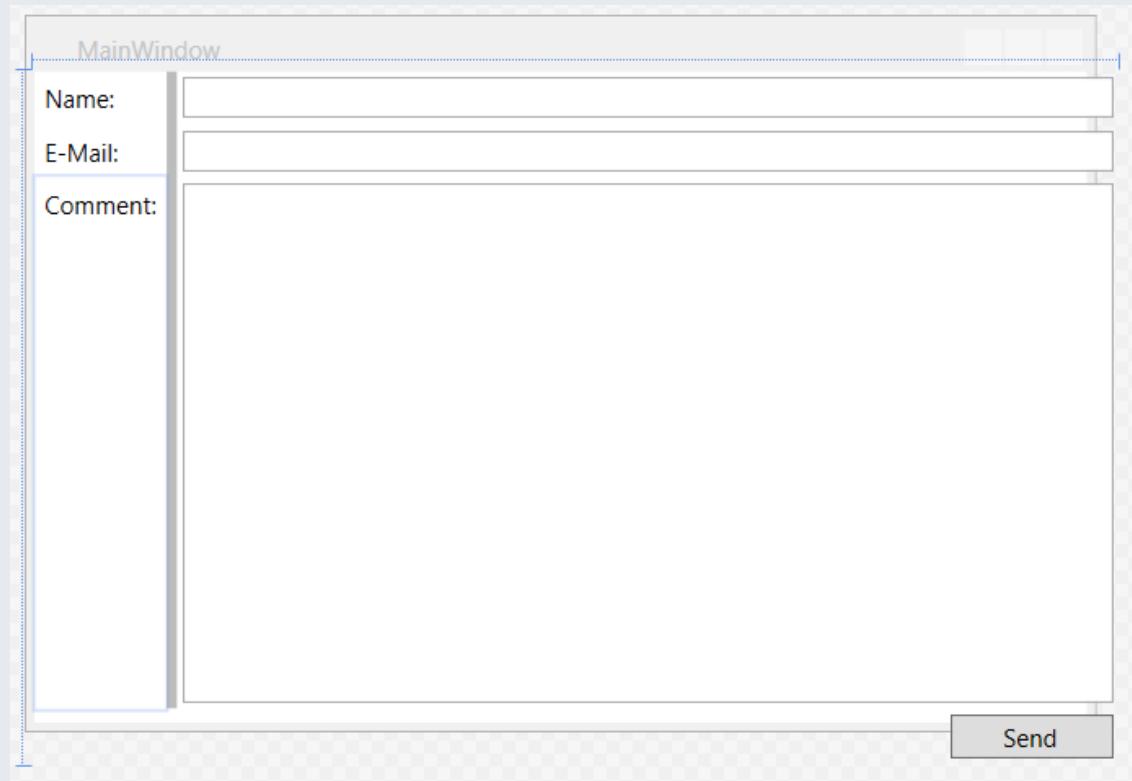
## Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de calculadora con cuatro operaciones



## Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores.

- Contenedor **Grid**: Es el contenedor por defecto, tiene un estilo tabla con filas y columnas. Permite dividir el contenido en filas, cada una con un ancho que puede ser diferente, y columnas, que tienen cada una un alto 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 el **\***
  - Si queremos hacer variable alguna columna o fila podemos utilizar el control **GridSplitter**.

## Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores. Grid

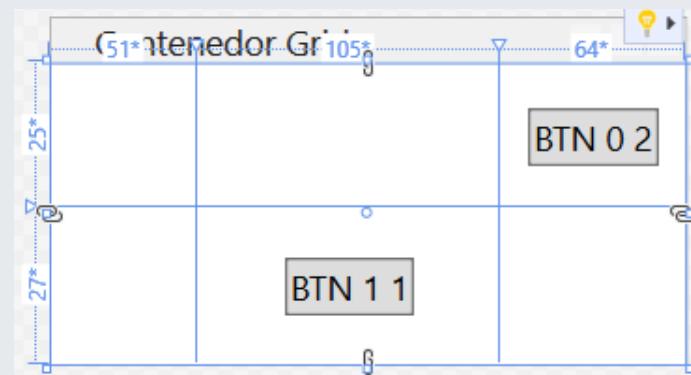


# Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores. Grid

```
<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>
```

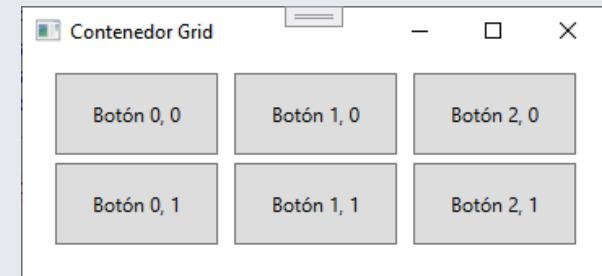
## Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores. Ejemplo de Grid diseñada con XAML

```
<Grid Name="contenedor">
    <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 i = 0; i < 3; i++){
    for (int j = 0; j < 2; j++){
        Button boton = new Button();
        boton.Width = 100;
        boton.Height = 50;
```



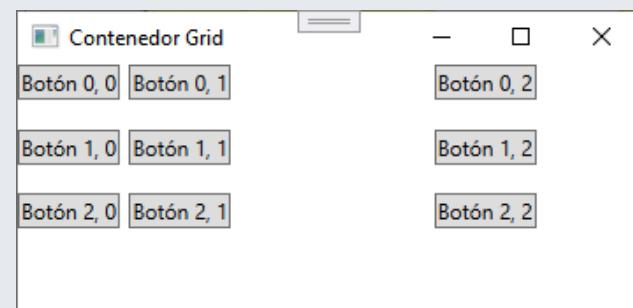
```
boton.Content = "Botón " + i + ", " + j;
boton.HorizontalAlignment = HorizontalAlignment.Left;
boton.VerticalAlignment = VerticalAlignment.Top;

//La colocación de los controles depende
//de su alineación y del margen de cada control

boton.Margin = new Thickness(20 + i * 110, 10 + j * 55, 0, 0);
contenedor.Children.Add(boton);
}
}
```

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

```
for (int i = 0; i < 3; i++){
    for (int j = 0; j < 3; j++){
        Button boton = new Button();
        boton.Content = "Botón " + i + "," + j;
        boton.HorizontalAlignment =
            HorizontalAlignment.Left;
        boton.VerticalAlignment = VerticalAlignment.Top;
        //La colocación de los controles depende de los
        //métodos
        //estáticos SetRow y SetColumn de la clase Grid
        Grid.SetRow(boton, i);
        Grid.SetColumn(boton, j);
        contenedor.Children.Add(boton);
    }
}
```

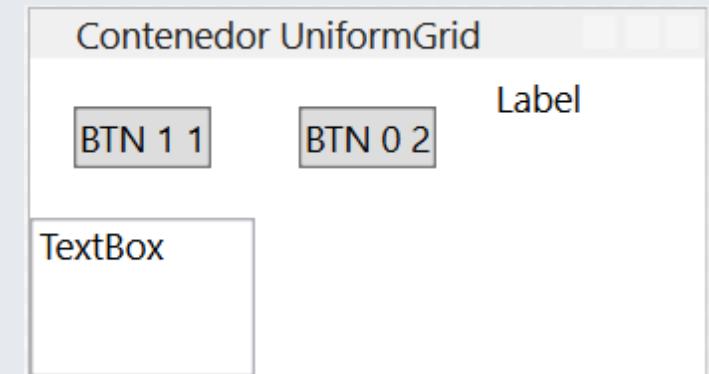


## Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores

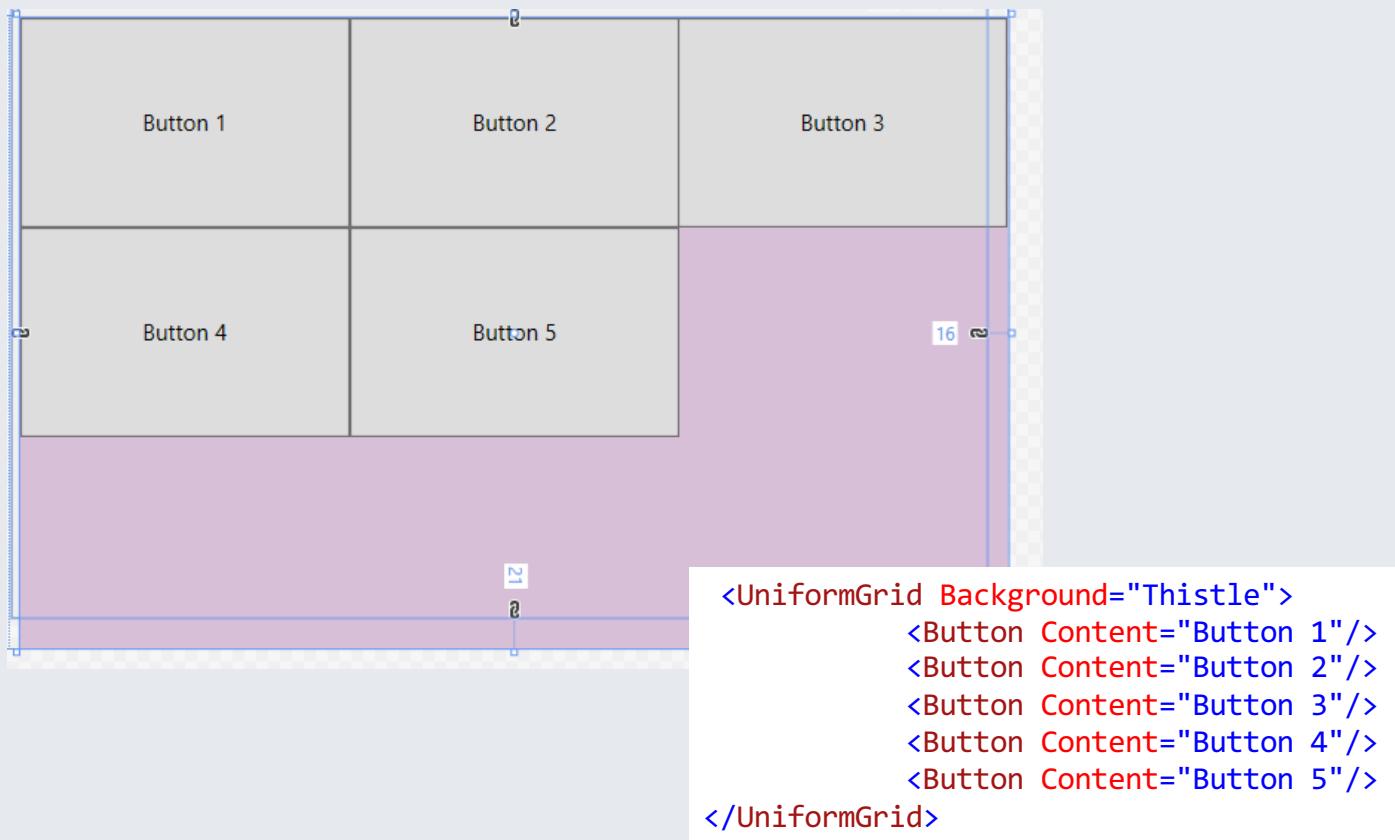
- **UniformGrid:** Distribución de elementos en una matriz cuadrada uniforme, es decir, según el tamaño de los objetos que contiene, distribuye el espacio de forma proporcional a todos.
- Funciona de la misma forma que **Grid** pero con la diferencia de que todas las filas tendrán el mismo alto y todas 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.

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

```
<UniformGrid Rows="2" Columns="3">
    <Button Content="BTN 1 1" .../>
    <Button Content="BTN 0 2" .../>
    <Label Content="Label"/>
<TextBox TextWrapping="Wrap" .../>
</UniformGrid>
```

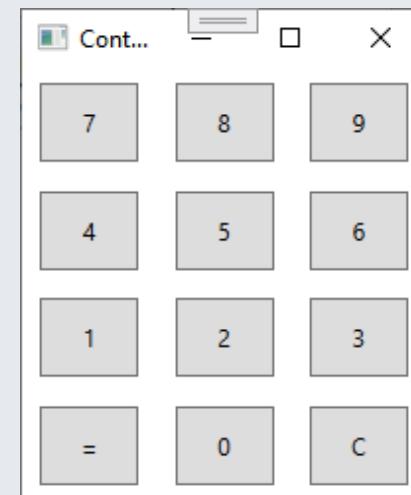


# Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores. UniformGrid



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

```
for (int i = 0; i < 4; i++){
    for (int j = 0; j < 3; j++){
        Button boton = new Button();
        boton.Width = 50;
        boton.Height = 40;
        if (i == 3)
            switch (j){
                case 0:
                    boton.Content = "="; break;
                case 1:
                    boton.Content = "0"; break;
                case 2:
                    boton.Content = "C"; break;
            }
        else
            boton.Content = (9 - 3 * i) - (2 - j);
        boton.HorizontalAlignment = HorizontalAlignment.Center;
        boton.VerticalAlignment = VerticalAlignment.Center;
        //La colocación de los controles depende de los métodos
        //estáticos SetRow y SetColumn de la clase Grid
        Grid.SetRow(boton, i);
        Grid.SetColumn(boton, j);
        contenedor.Children.Add(boton);
    }
}
```

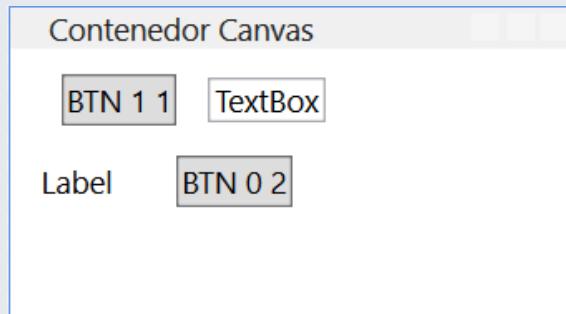


## Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores

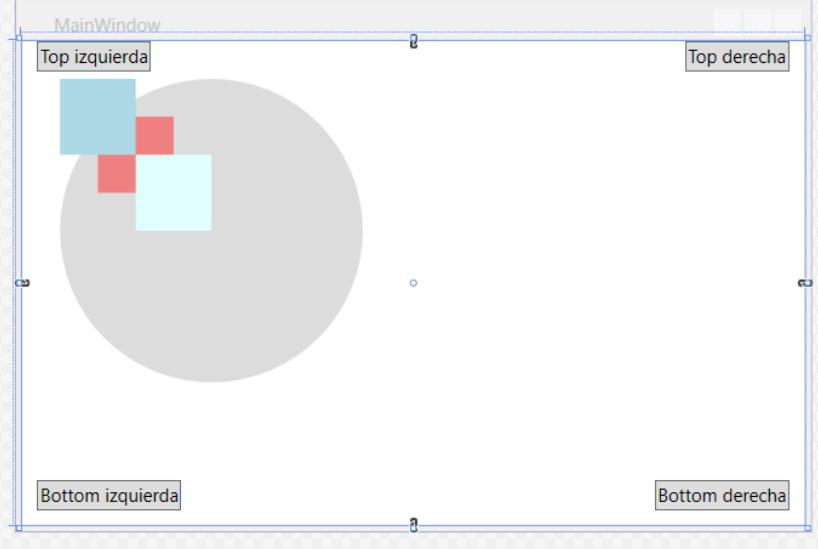
- **Canvas.** Es el más simple e intuitivo de todos. Imita las características de un panel de Windows Forms. Permite asignar coordenadas absolutas donde el (0,0) del eje de coordenadas está en la esquina superior izquierda. Se trata de una ubicación precisa de elementos.
- Los objetos se van apilando 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="contenedor">
    <Button Content="BTN 1 1"
        Canvas.Top="10" Canvas.Left="20" .../>
    <Button Content="BTN 0 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

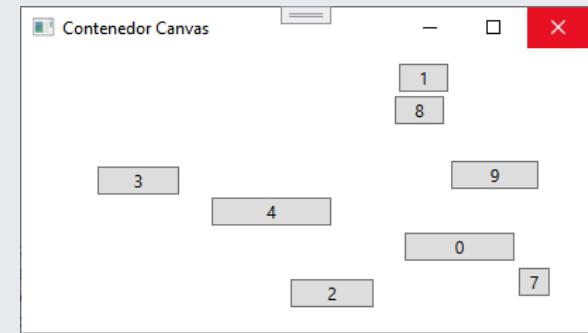


```
<Canvas>
    <Button Canvas.Left="10">Top izquierda</Button>
    <Button Canvas.Right="10">Top derecha</Button>
    <Button Canvas.Left="10" Canvas.Bottom="10">Bottom izquierda</Button>
    <Button Canvas.Right="10" Canvas.Bottom="10">Bottom derecha</Button>
</Canvas>

<Canvas>
    <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

```
Random aleatorio=new Random();
for (int i = 0; i < 10; i++){
    Button boton= new Button();
    boton.Content = i;
    boton.Width = aleatorio.Next(100);
    boton.HorizontalAlignment = HorizontalAlignment.Center;
    boton.VerticalAlignment = VerticalAlignment.Center;
    //La colocación de los controles depende de los métodos
    //estáticos SetLeft y SetTop de la clase Canvas
    Canvas.SetLeft(boton, aleatorio.Next((int)this.Width));
    Canvas.SetTop(boton, aleatorio.Next((int)this.Height));
    contenedor.Children.Add(boton);
}
```



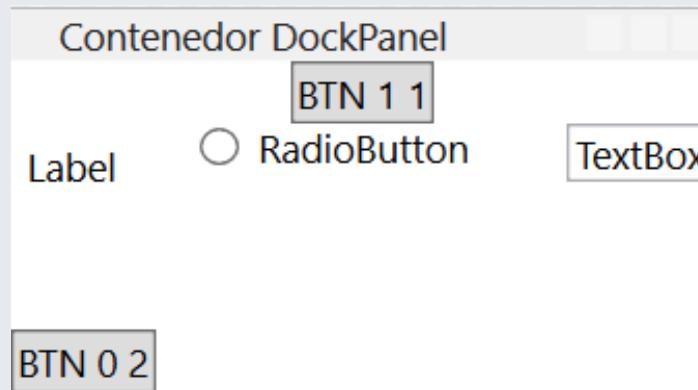
*Observaciones: Incluir un canvas (puede ser manualmente) e incluir como nombre al formulario "contenedor".*

## Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores

- **DockPanel:** La posición es relativa con respecto a otro objeto hijo dentro del contenedor.
- Permite acoplar controles mapeando la pantalla en cinco partes diferentes. Las posibles posiciones son Top, Right, Bottom y Left (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.
  - Se utiliza para dividir la ventana en zonas con funcionalidades específicas.

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

```
<DockPanel Name="contenedor">
    <Button Content="BTN 1 1"
        DockPanel.Dock="Top" .../>
    <Button Content="BTN 0 2"
        DockPanel.Dock="Bottom" .../>
    <Label Content="Label" DockPanel.Dock="Left"/>
    <TextBox Text="TextBox" DockPanel.Dock="Right" .../>
    <RadioButton Content="RadioButton" Width="95" .../>
</DockPanel>
```

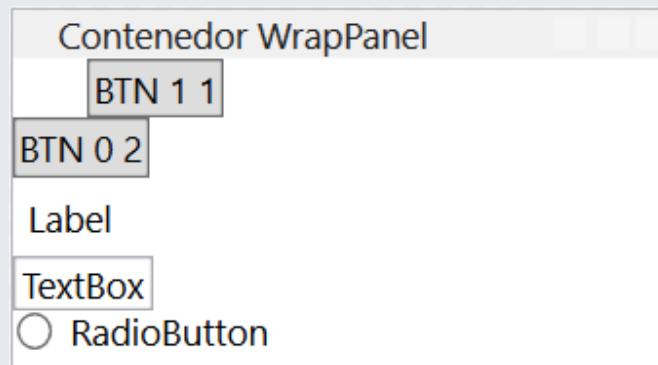


## Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores

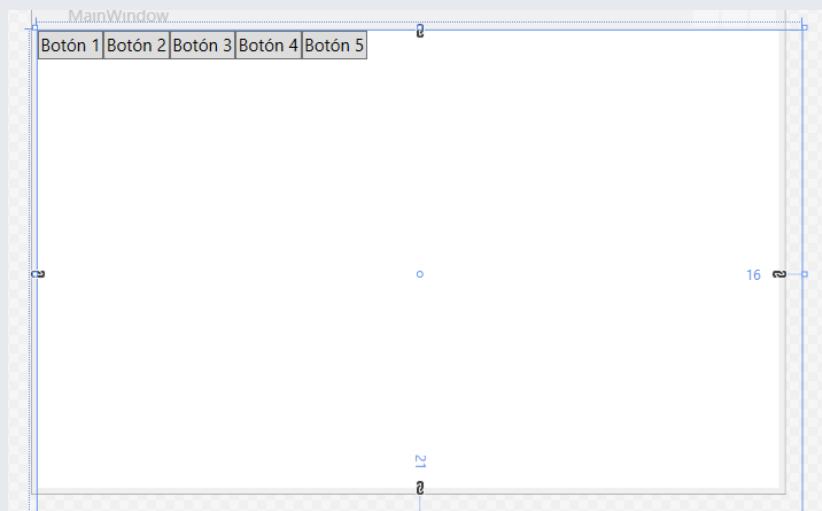
- **WrapPanel:** La posición de los objetos hijos es secuencial horizontal de izquierda a derecha o vertical de arriba hacia abajo. No ajusta los tamaños de los objetos.
  - Se utiliza para posicionar los controles uno a continuación del otro, de forma similar a como se gestionan los componentes en los 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="contenedor" Orientation="Vertical">  
    <Button Content="BTN 1 1" .../>  
    <Button Content="BTN 0 2" .../>  
    <Label Content="Label"/>  
    <TextBox Text="TextBox" .../>  
  
    <RadioButton Content="Radio ..." />  
</WrapPanel>
```



## Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores. WrapPanel



The screenshot shows a Windows application window titled "MainWindow". Inside the window, there is a horizontal container holding five buttons labeled "Botón 1" through "Botón 5". The container has a blue border and is positioned at the top of the window. The window itself has a light gray background and a title bar.

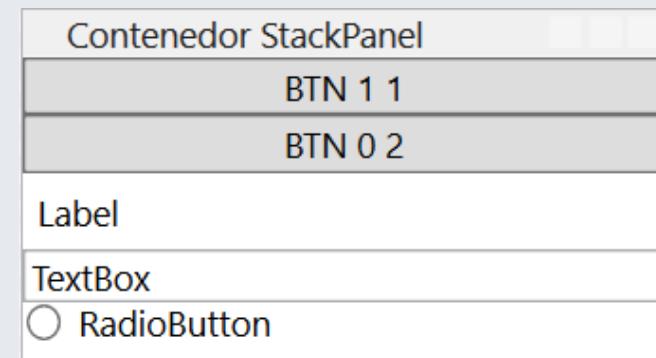
```
<WrapPanel Orientation="Horizontal">
    <Button Content="Botón 1" />
    <Button Content="Botón 2" />
    <Button Content="Botón 3" />
    <Button Content="Botón 4" />
    <Button Content="Botón 5" />
</WrapPanel>
```

## Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores

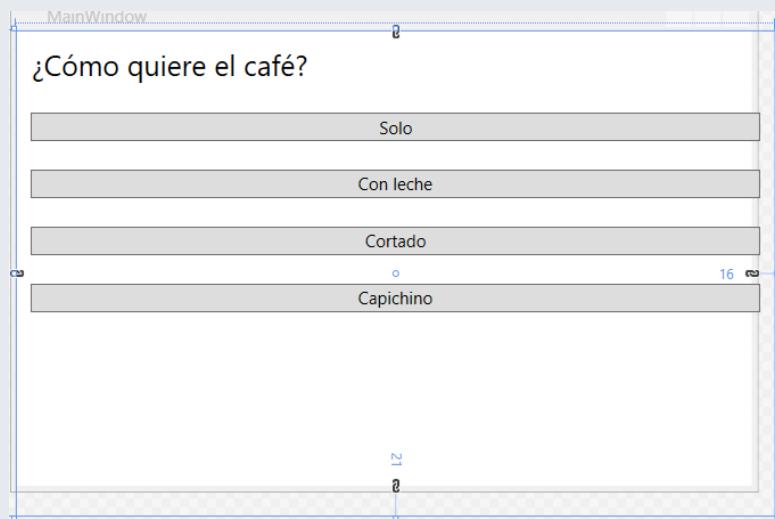
- **StackPanel:** Es un contenedor que apila los objetos de manera horizontal o vertical. Por defecto se apilan de forma vertical pero repartiéndose sobre toda la superficie.
  - Funciona de forma similar a como lo hace **WrapPanel** pero no se adapta si los controles ocupan demasiado sino que se expande.
- 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 1"/>  
    <Button Content="BTN 0 2"/>  
    <Label Content="Label"/>  
    <TextBox Text="TextBox"  
            TextWrapping="Wrap"/>  
        <RadioButton  
        Content="RadioButton"/>  
</StackPanel>
```

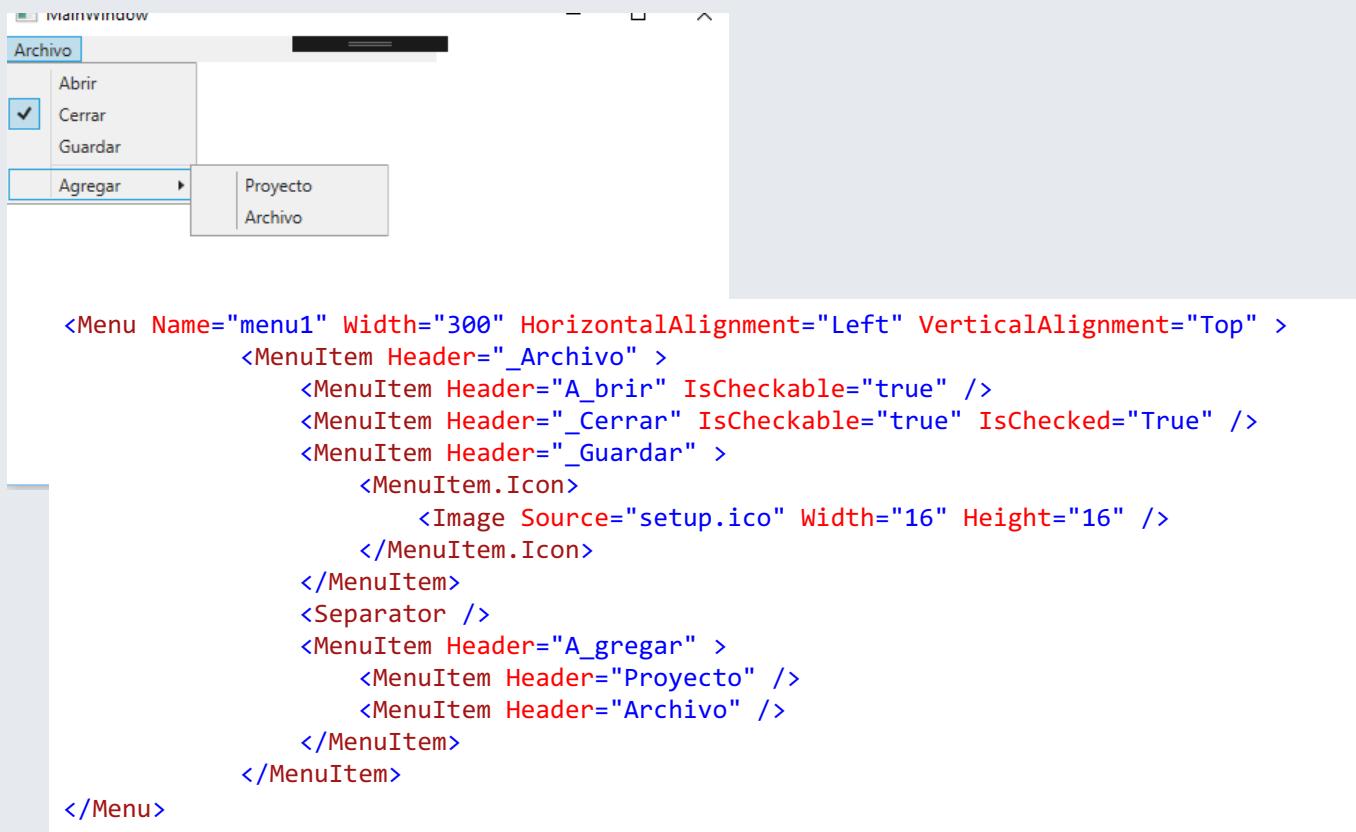


# Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores. StackPanel



```
<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. Menu



```
<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="_Guarda" >
            <MenuItem.Icon>
                <Image Source="setup.ico" Width="16" Height="16" />
            </MenuItem.Icon>
        </MenuItem>
        <Separator />
        <MenuItem Header="A_gregar" >
            <MenuItem Header="Proyecto" />
            <MenuItem Header="Archivo" />
        </MenuItem>
    </MenuItem>
</Menu>
```

# Lenguajes de descripción de interfaces basados en XML. XAML. Contenedores. Frame



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

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

En el siguiente ejemplo al presionar Alt+D, el foco se posiciona en *TextBox2*

```
<DockPanel Margin="0,10,3,3" Grid.Column="0" Grid.Row="5">
    <TextBox Name="textBox2" Width="100" Height="20"/>
    <Label Target="{Binding ElementName(textBox2)}">
        <AccessText FontSize="12" FontWeight="Bold">E_ditar</AccessText>
    </Label>
</DockPanel>
<TextBox Name="textBox3" Width="100" Height="20" Canvas.Top="30" />
```



# 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" >  
    Algun 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 Name="btn5" Canvas.Left="64" Canvas.Top="17" Width="129" Height="33"
Click="btnAceptar_Click" >
    <StackPanel Orientation="Horizontal" Width="Auto">
        <Image Source="ms.jpg" Width="20" Height="20"></Image>
        <TextBlock Text="Aceptar" TextAlignment="Right" VerticalAlignment="Center"
Width="69"></TextBlock>
    </StackPanel>
</Button>
```



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

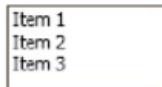
## **ListBox - ComboBox**

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>
```



## 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*.

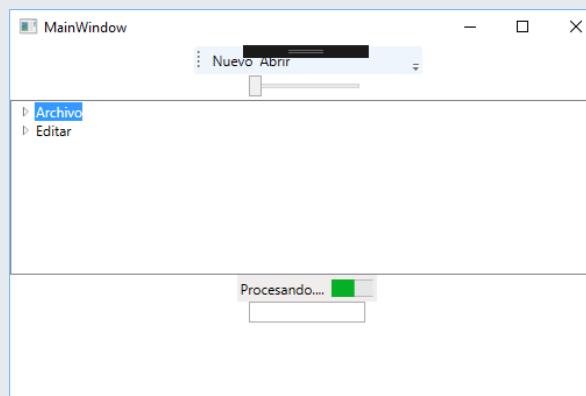
```
<StackPanel Height="40" Canvas.Left="35" Canvas.Top="43" Width="115">
    <RadioButton Name="rb1" Checked="rb_Checked">Si</RadioButton>
    <RadioButton Name="rb2" Checked="rb_Checked">No</RadioButton>
    <RadioButton Name="rb3" Checked="rb_Checked">No sabe/no contesta</RadioButton>
</StackPanel>
```

- Si
- No
- No sabe/no contesta

# 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.
- **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.



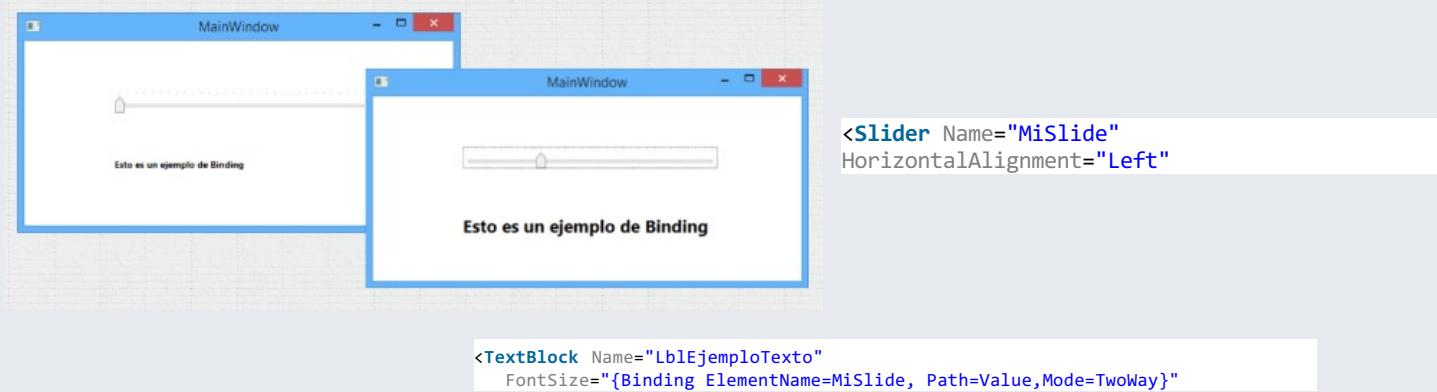
# Lenguajes de descripción de interfaces basados en XML. XAML. Controles avanzados

```
>
<StackPanel>
    <ToolBar Height="26" Name="toolBar1" Width="200" >
        <Button>
            <Image Source="setup.ico"/>
        </Button>
        <Button>Nuevo</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>
```

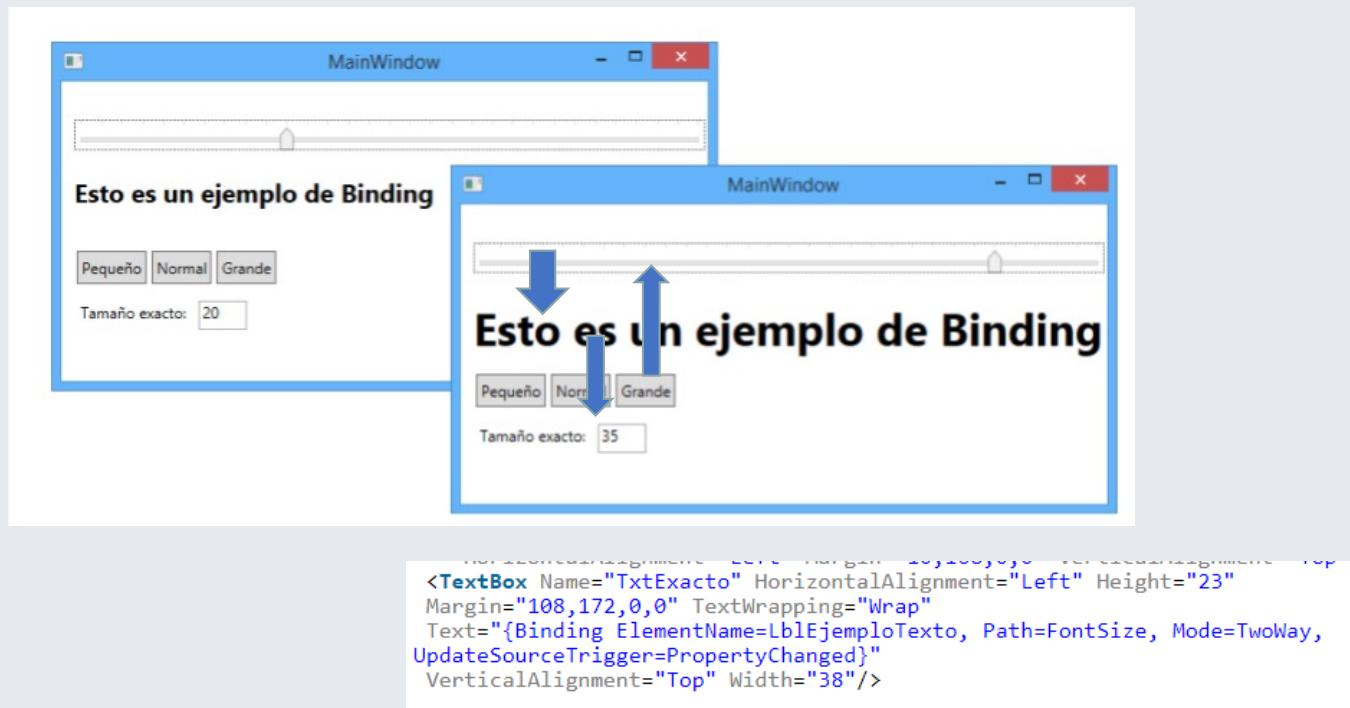
# Lenguajes de descripción de interfaces basados en XML. XAML. Data Binding – Enlazar datos - Ejemplo



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.



# Lenguajes de descripción de interfaces basados en XML. XAML. Data Binding – Enlazar datos - Ejemplo



# Lenguajes de descripción de interfaces basados en XML. XAML. Data Binding – Enlazar datos - Ejemplo

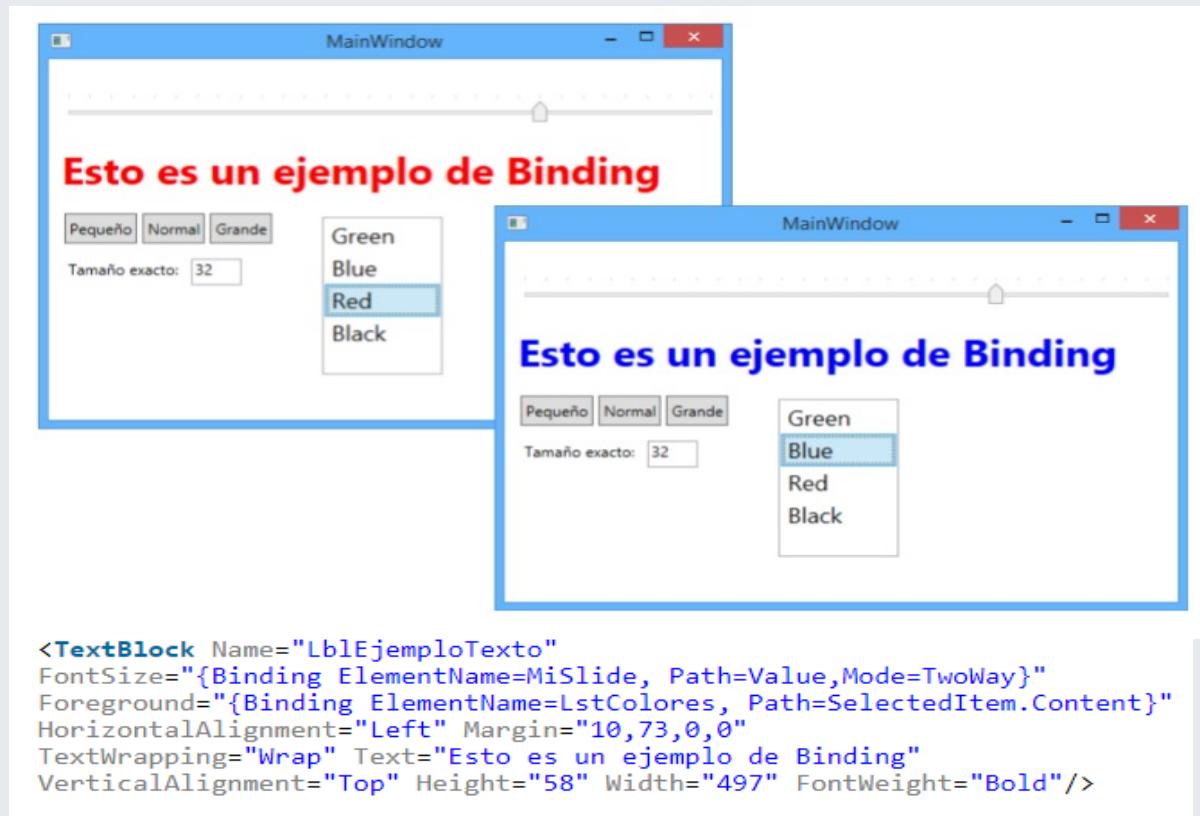
```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

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

    private void CmdSetNormal(object sender, RoutedEventArgs e)
    {
        this.MiSlide.Value = 24;
    }

    private void CmdSetGrande(object sender, RoutedEventArgs e)
    {
        this.MiSlide.Value = 34;
    }
}
```

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

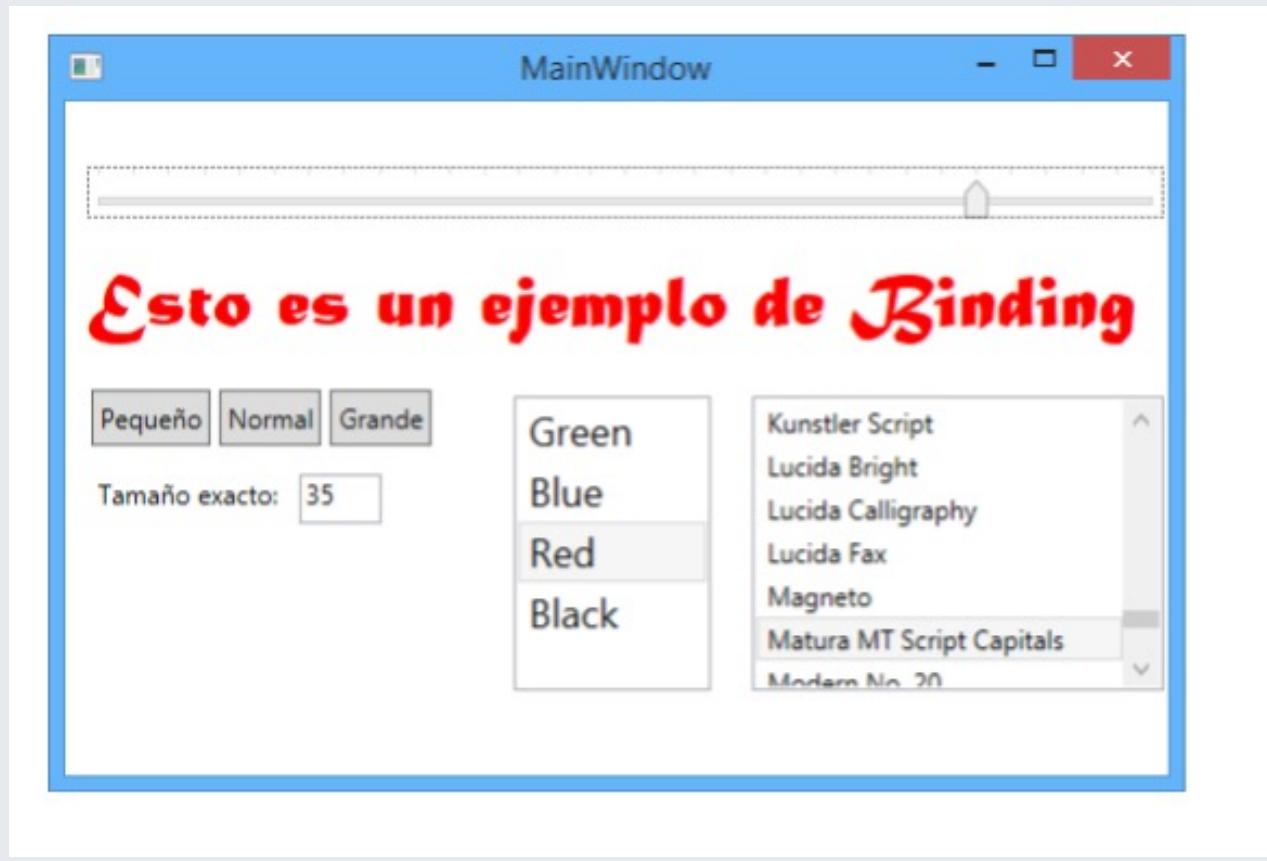


## Lenguajes de descripción de interfaces basados en XML. XAML. Data Binding – Enlazar datos – Ejemplo 2

```
<TextBlock Name="LblEjemploTexto"  
FontSize="{Binding ElementName=MiSlide, Path=Value, Mode=TwoWay}"  
Foreground="{Binding ElementName=LstColores, Path=SelectedItem.Content}"  
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="LstColores" HorizontalAlignment="Left" Height="136"  
Margin="207,136,0,0" VerticalAlignment="Top" Width="91" FontSize="18">  
    <ListBoxItem Content="Green"/>  
    <ListBoxItem Content="Blue"/>  
    <ListBoxItem Content="Red"/>  
    <ListBoxItem Content="Black"/>  
</ListBox>
```

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



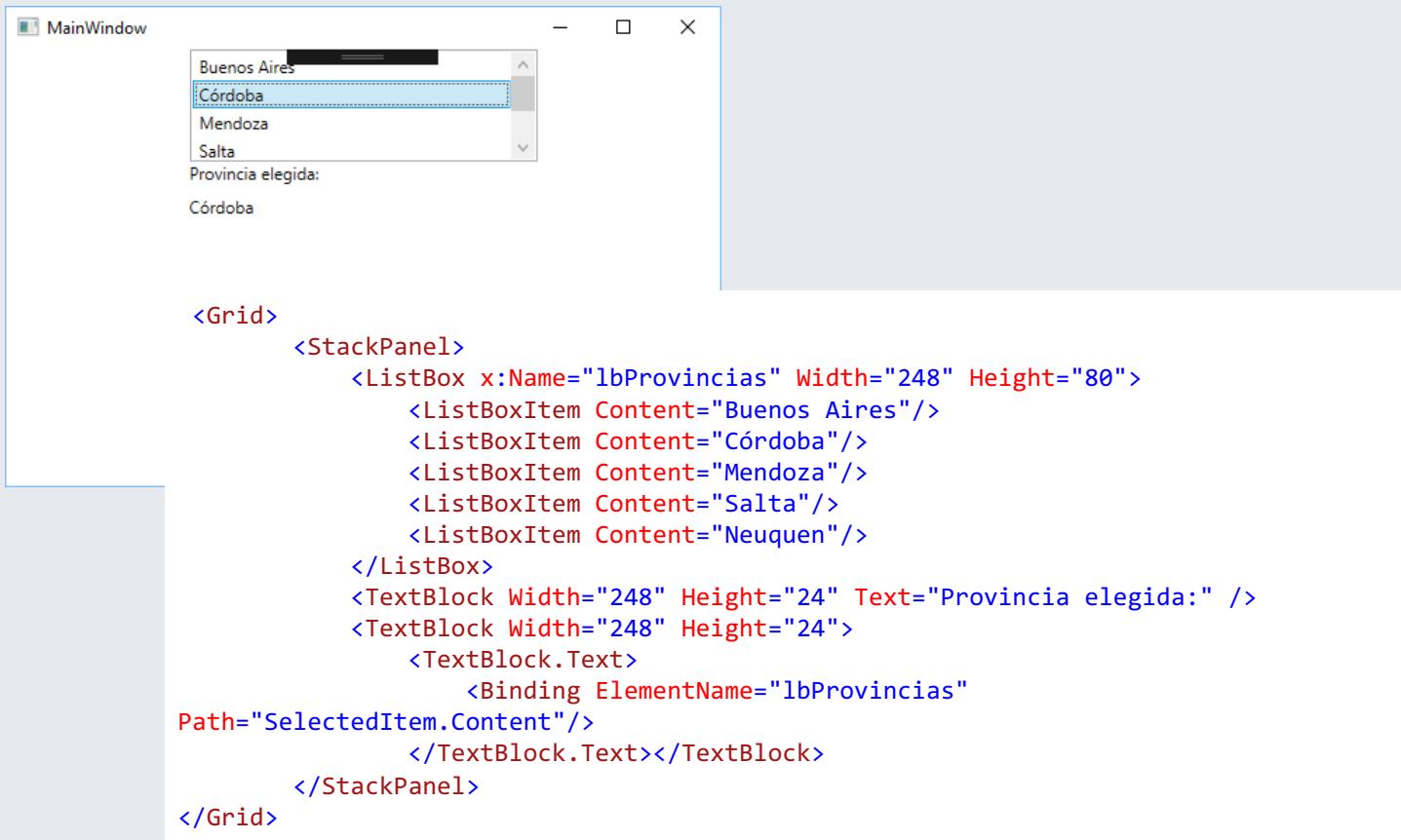
# Lenguajes de descripción de interfaces basados en XML. XAML. Data Binding – Enlazar datos – Ejemplo 3

```
<TextBlock Name="Lbtjemplotexto"
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}"/>
```

<https://wpf.csharp.com.es/binding/>

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

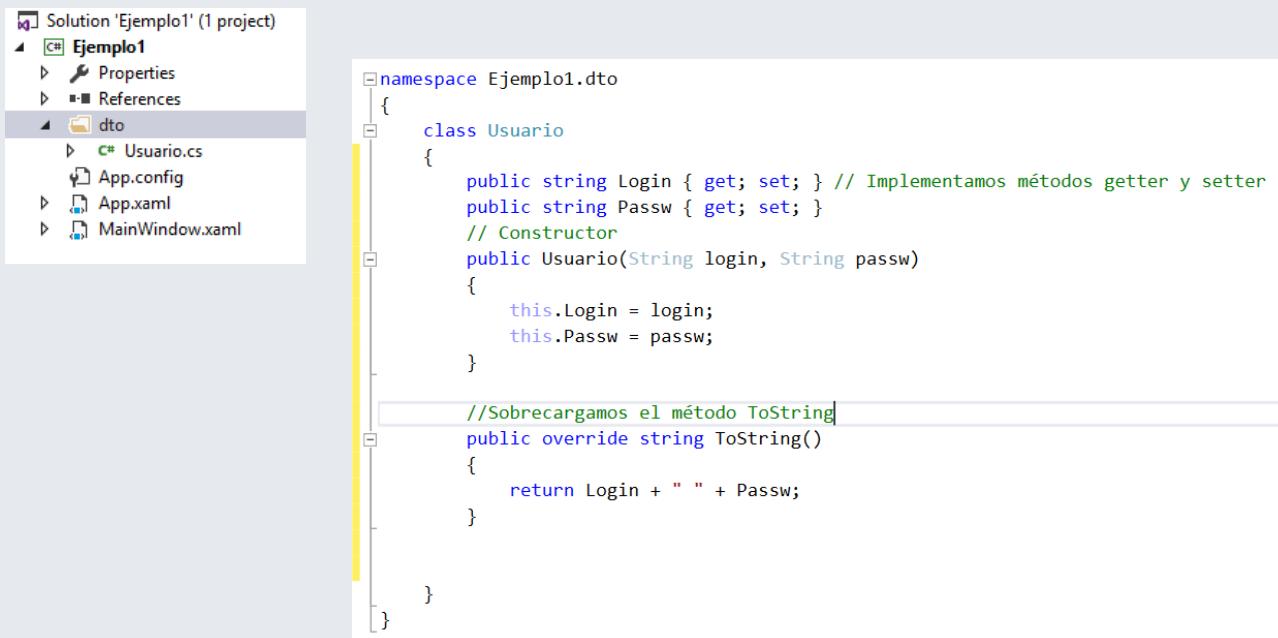


The screenshot shows a Windows application window titled "MainWindow". Inside the window, there is a list box containing four items: "Buenos Aires", "Córdoba", "Mendoza", and "Salta". The item "Córdoba" is currently selected, highlighted with a blue border. Below the list box, there is a text block labeled "Provincia elegida:" followed by the text "Córdoba".

```
<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. Ejemplo de relleno de un combobox

## 1.a) Creamos la clase Usuario



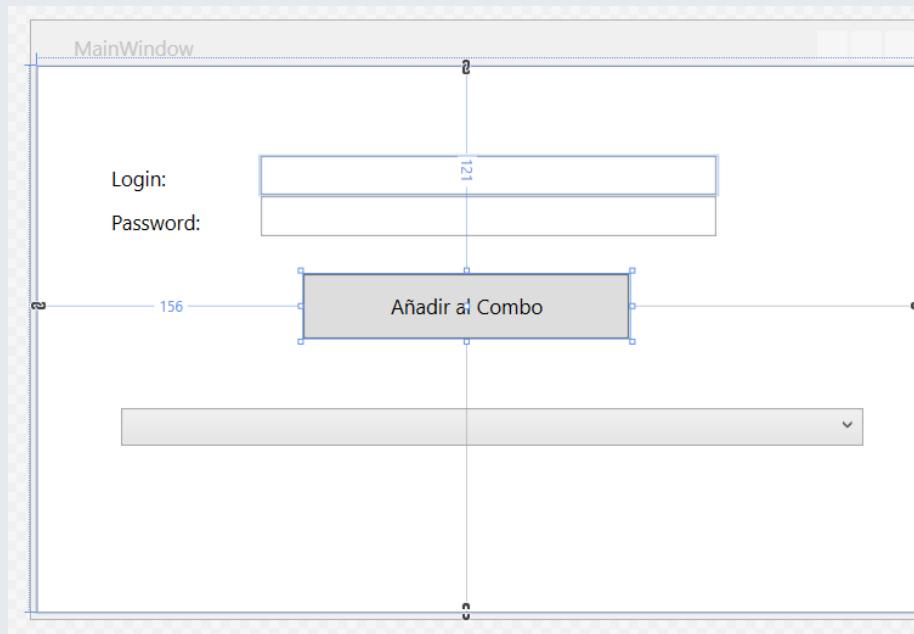
The screenshot shows the Visual Studio IDE interface. On the left, the Solution Explorer displays a project named 'Ejemplo1' containing a 'Properties' folder, a 'References' folder, a 'dto' folder, and files 'App.config', 'App.xaml', and 'MainWindow.xaml'. The 'Usuario.cs' file is selected in the 'dto' folder. On the right, the code editor shows the following C# code:

```
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. Ejemplo de relleno de un combobox

1.b) Creamos la interfaz



# Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de relleno de un combobox

## 1.c) Damos funcionalidad al botón

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

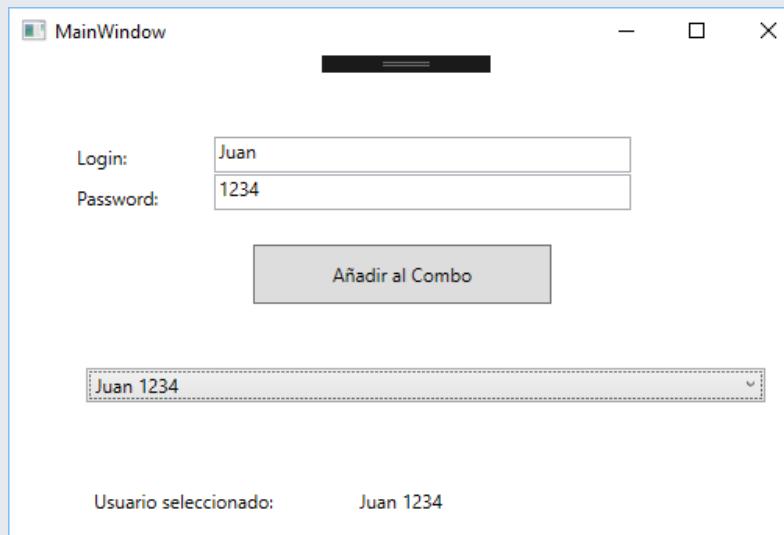
namespace Ejemplo1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

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

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

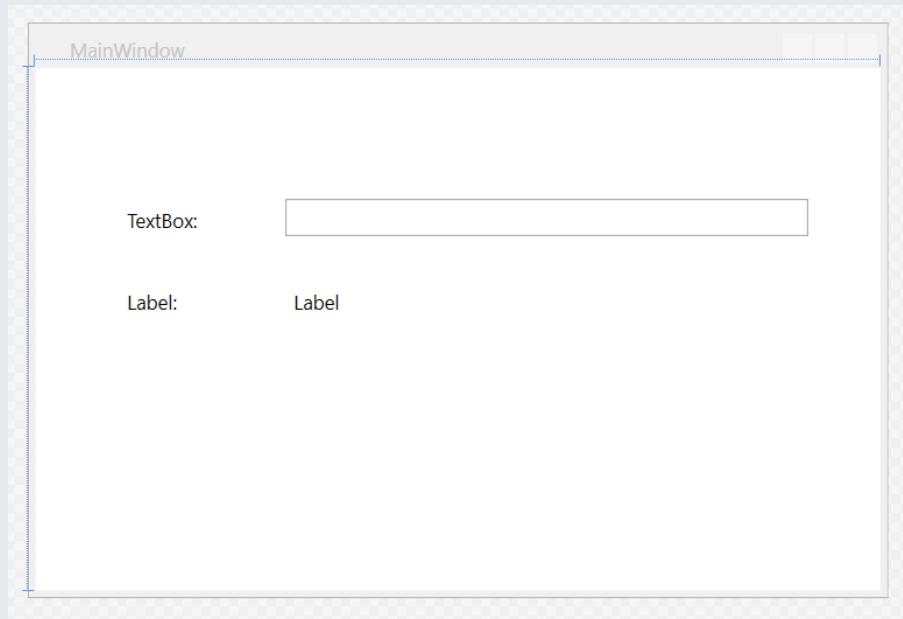
# Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de rellenado de un combobox

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



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

# Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de replicar un TextBox en un Label

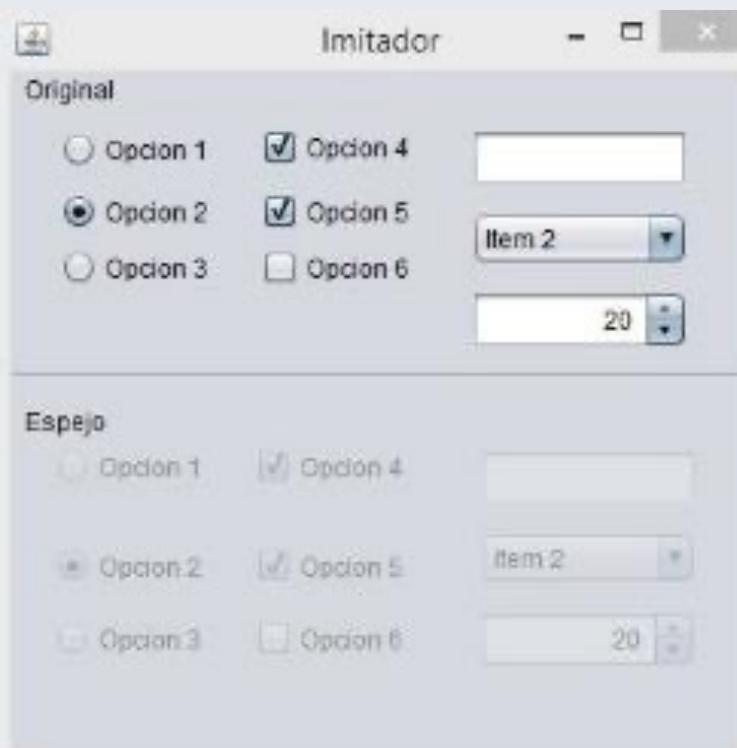


```
<Label x:Name="lblLabel" Content="{ Binding ElementName=txtBox, Path=Text}"  
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="153,130,0,0" Width="320"/>
```

## Lenguajes de descripción de interfaces basados en XML. XAML. 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.

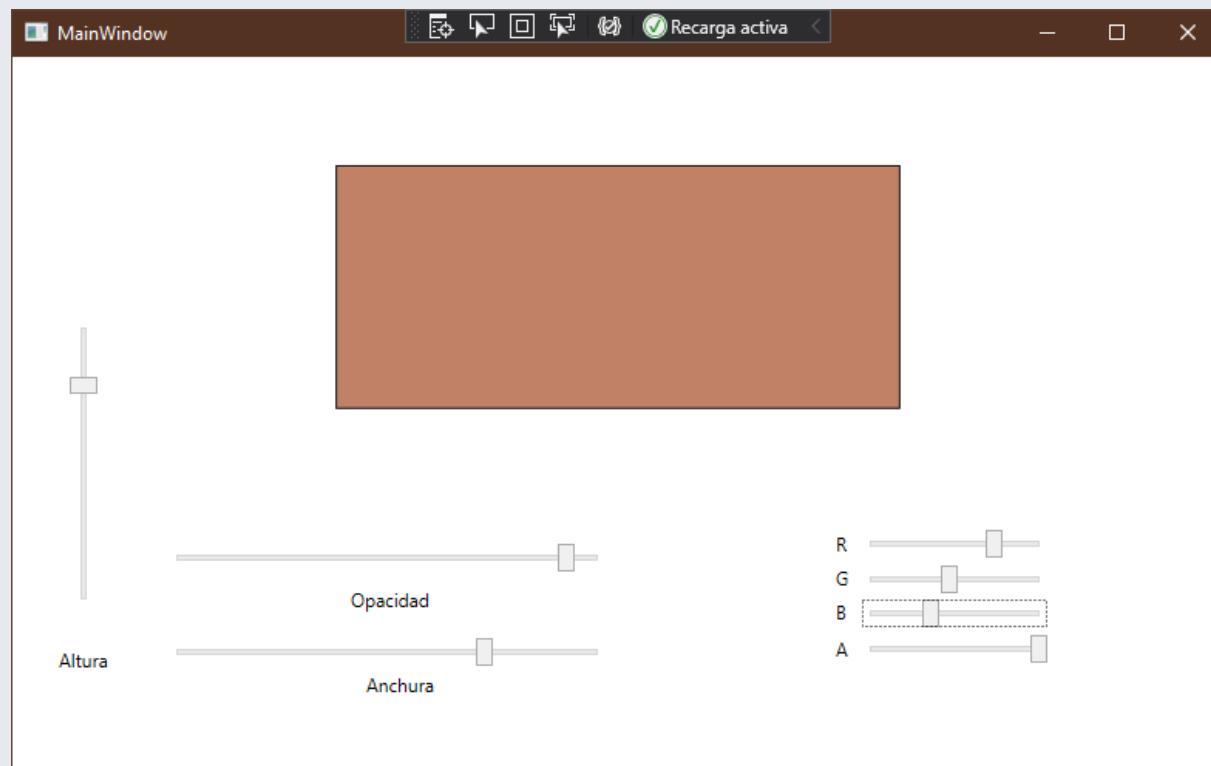
## Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de Espejo usando Binding



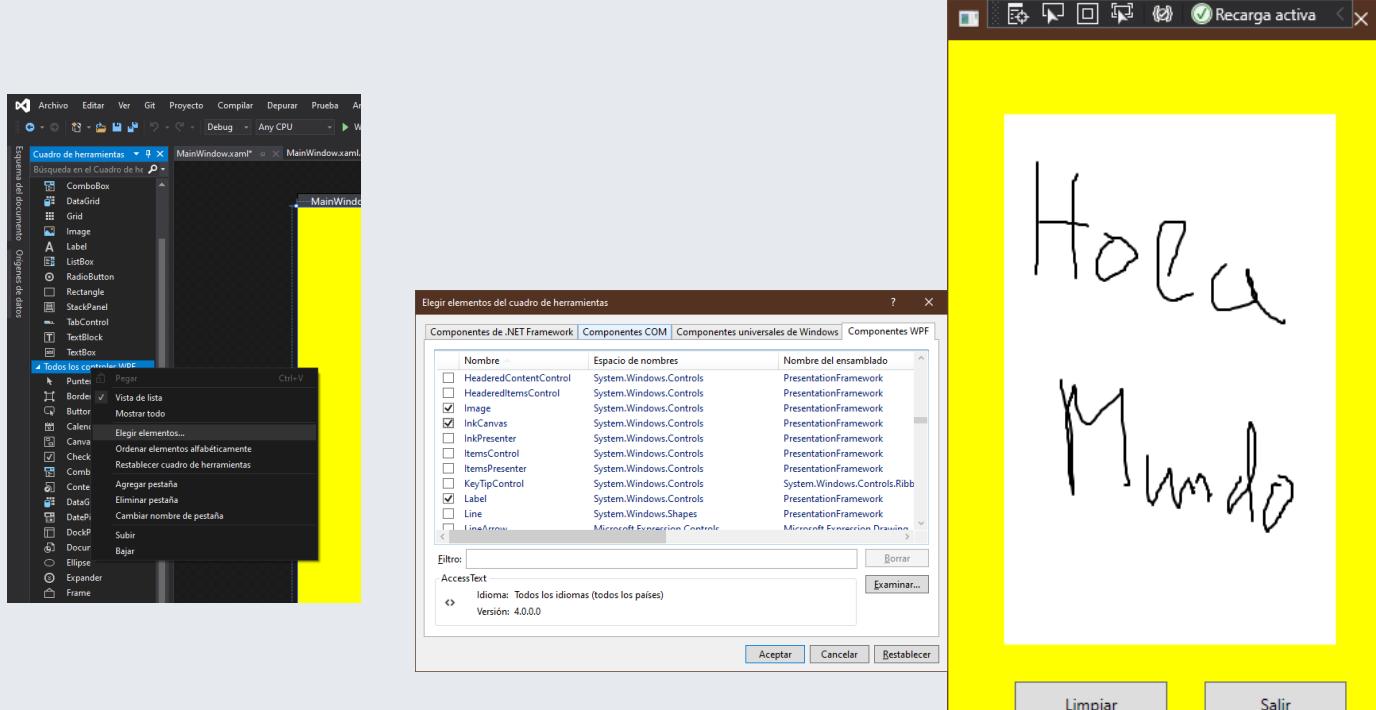
## Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de uso de Slider

- Disponemos de un objeto rectángulo (rectangle) de color azul. 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) . Inicializar los slider's a 0. Si ponemos otro valor dará problemas ya que los cuatro llaman al mismo método y algunos objetos no han sido creados en la primera llamada al método.

## Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de uso de Slider



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



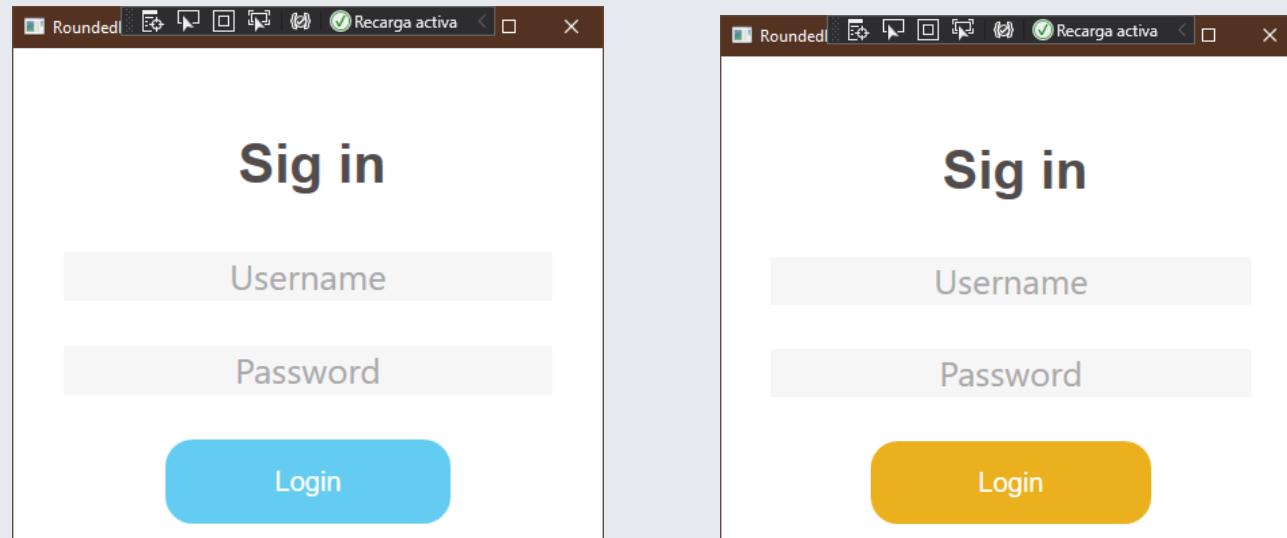
```
<Grid Background="Yellow">
    <InkCanvas x:Name="lnkPizarra" Height="400" Width="250"/>
    <Button x:Name="btnLimpiar" Content="Limpiar" HorizontalAlignment="Left"
Margin="50,483,0,0" VerticalAlignment="Top" Height="34" Width="115" Click="btnLimpiar_Click"/>
    <Button x:Name="btnSalir" Content="Salir" HorizontalAlignment="Left" Margin="193,483,0,0"
VerticalAlignment="Top" Height="34" Width="107" Click="btnSalir_Click"/>
```

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

```
private void grosor_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    paint.DefaultDrawingAttributes.Width = grosor.Value;
    paint.DefaultDrawingAttributes.Height = grosor.Value;
}

private void ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    Brush brush = new SolidColorBrush(Color.FromArgb((Byte)sldbrillo.Value,
(Byte)sldred.Value, (Byte)sldgreen.Value, (Byte)sldblue.Value));
    SolidColorBrush scbrush = (SolidColorBrush)brush;
    paint.DefaultDrawingAttributes.Color = scbrush.Color;
    rectcolor.Fill = brush;
}
```

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



Al colocar el puntero del ratón sobre el botón

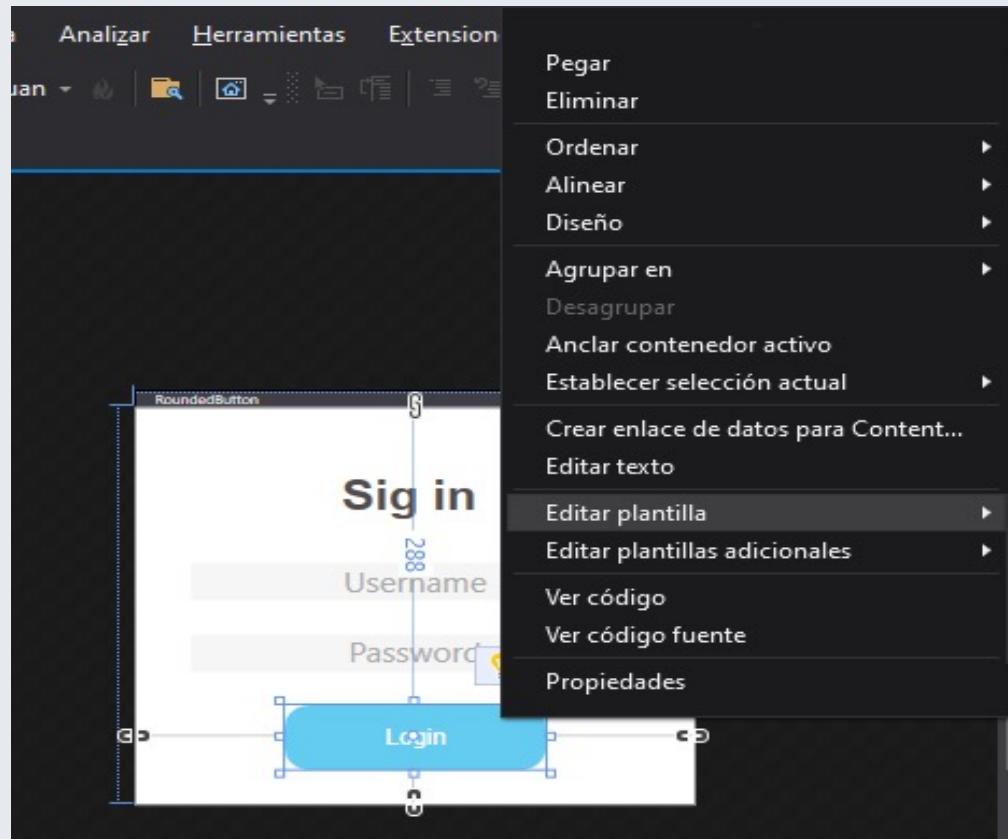
## Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de Formulario de acceso

```
<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">
```

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

```
<Grid>
    <TextBlock HorizontalAlignment="Left" Height="48" Margin="0,62,0,0" Text="Sig in" TextWrapping="Wrap"
    VerticalAlignment="Top" TextAlignment="Center"
        FontFamily="Arial"
        FontSize="40"
        FontWeight="Bold"
        Foreground="#FF534C4C" Width="440"/>
    <TextBox HorizontalAlignment="Center" Height="38" Margin="0,149,0,0" Text="Username" TextWrapping="Wrap"
    VerticalAlignment="Top" Width="362"
        FontSize="26"
        BorderBrush="{x:Null}"
        Background="#FFF6F6F6"
        Foreground="DarkGray"
        TextAlignment="Center"/>
    <TextBox HorizontalAlignment="Center" Height="38" Margin="0,218,0,0" Text="Password" TextWrapping="Wrap"
    VerticalAlignment="Top" Width="362"
        FontSize="26"
        BorderBrush="{x:Null}"
        Background="#FFF6F6F6"
        Foreground="DarkGray"
        TextAlignment="Center"/>
    <Button Style="{DynamicResource RoundedButtonStyle}" Content="Login" HorizontalAlignment="Center" Height="62"
Margin="0,288,0,0" VerticalAlignment="Top" Width="210"
        Background="#FF64CCF1"
        Foreground="White"
        FontFamily="Arial"
        FontSize="20"/>
</Grid>
```

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



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

```
<Setter Property="BorderThickness" Value="0"/>

<Border x:Name="border" CornerRadius="20" Background="{TemplateBinding
Background}" BorderThickness="{TemplateBinding BorderThickness}"
BorderBrush="{TemplateBinding BorderBrush}" SnapsToDevicePixels="true">

<Trigger Property="IsMouseOver" Value="true">
    <Setter Property="Background" TargetName="border" Value="#EAB01E"/>
    <Setter Property="BorderBrush" TargetName="border" Value="#EAB01E"/>
</Trigger>

<Trigger Property="IsPressed" Value="true">
    <Setter Property="Background" TargetName="border" Value="#F1CA68"/>
    <Setter Property="BorderBrush" TargetName="border" Value="#F1CA68"/>
</Trigger>
```

## Lenguajes de descripción de interfaces basados en XML. XAML. Ejemplo de gestión de tabla de personas

- El siguiente ejemplo utilizará un control **DataGrid** que es una tabla donde se incluyen datos. Inicialmente tiene datos de ejemplo no visibles.

The screenshot shows a Windows application window titled "Ejemplo de Gestión de Personas". On the left side, there are four text input fields labeled "Nombre", "Apellidos", and "Edad", each with a corresponding text box below it. Below these fields is a button labeled "Agregar Persona". To the right of these controls is a DataGrid control displaying five rows of data. The columns are labeled "SampleInt", "SampleStringA", "SampleStringB", and "SampleBool". The data in the grid is as follows:

SampleInt	SampleStringA	SampleStringB	SampleBool
1	Cadena de ejemplo A: 1	Cadena de ejemplo B: 1	<input checked="" type="checkbox"/>
2	Cadena de ejemplo A: 2	Cadena de ejemplo B: 2	<input checked="" type="checkbox"/>
3	Cadena de ejemplo A: 3	Cadena de ejemplo B: 3	<input checked="" type="checkbox"/>
4	Cadena de ejemplo A: 4	Cadena de ejemplo B: 4	<input checked="" type="checkbox"/>
5	Cadena de ejemplo A: 5	Cadena de ejemplo B: 5	<input checked="" type="checkbox"/>

At the bottom of the window, there are two buttons: "Modificar" on the left and "Eliminar" on the right. The window has a total width of 20 units, with 175 pixels allocated to the left panel and 20 pixels to the right panel. The DataGrid itself has a width of 60 pixels.

## Lenguajes de descripción de interfaces basados en XML. XAML. 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 y elegir el tipo de control que se quiere utilizar en esta columna



## Lenguajes de descripción de interfaces basados en XML. XAML. 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. 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}"
```

- Además agregaremos 3 columnas de tipo texto añadiendo un Binding:

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

- 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. Ejemplo de gestión de tabla de personas

Ejemplo de Gestión de Personas

Nombre	Apellidos	Edad
--------	-----------	------

Nombre:

Apellidos:

Edad:

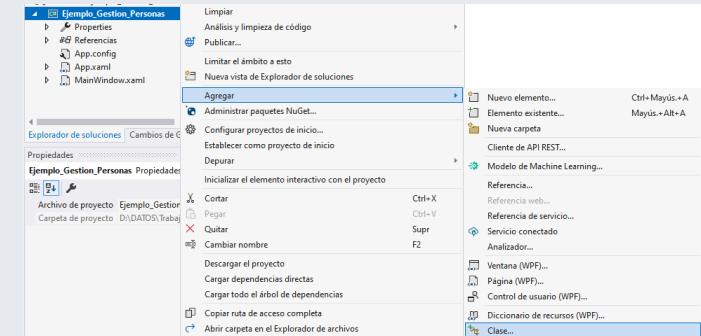
**Agregar Persona**

**Modificar**      **Eliminar**

The screenshot shows a Windows application window titled "Ejemplo de Gestión de Personas". On the left, there are three text input fields labeled "Nombre", "Apellidos", and "Edad", each with a corresponding empty input box below it. Below these fields is a button labeled "Agregar Persona". To the right of these controls is a data grid with three columns: "Nombre", "Apellidos", and "Edad". The grid is currently empty. At the bottom of the window, there are two buttons: "Modificar" on the left and "Eliminar" on the right.

# Lenguajes de descripción de interfaces basados en XML. XAML. 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 "Personas"
  - Pulsando con el botón derecho sobre el proyecto seleccionamos Agregar → Clase



## Lenguajes de descripción de interfaces basados en XML. XAML. 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...**
- En la ventana que aparece, elegimos los campos que queremos pasar como parámetro

## Lenguajes de descripción de interfaces basados en XML. XAML. 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**: 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. 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.ItemsSource.Refresh();  
    } }
```

## Lenguajes de descripción de interfaces basados en XML. XAML. 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();  
}
```

## Lenguajes de descripción de interfaces basados en XML. XAML. 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