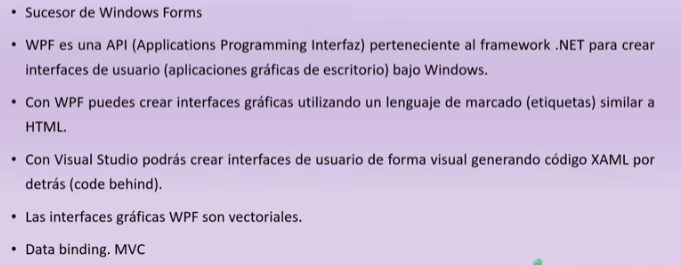
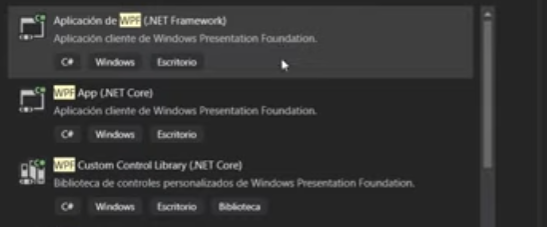
Windows Presentation Fundation (WPF)

Es el sucesor de Windows Forms



Vectorial significa que se la puede escalar, o sea, modificar su tamaño, sin que pierdan definición. (Vectores Vs Mapa de bits)



Es la primera de las opciones.

Agrega un textblock. Automáticamente en el XAML (el que parece HTML) agrega la etiqueta del mismo. Ahí mismo hay una etiqueta Text a la cual le podés dar valor.

Hay modificaciones mas fáciles desde vista diseño y otras desde este código.

<grid>

Aquí va el contenido

</grid>

Button

Hay otra etiqueta para button <Button>

Dentro de la etiqueta de apertura es donde se modifican las propiedades: content=”Enviar!”

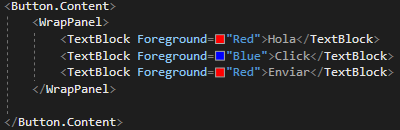
Width y height

<Button Content="Enviar !" Width="150" Height="100" FontSize="24"/>

Al final agregué una barra. Eso señala que la etiqueta ahí mismo tiene apertura y cierre.

Si se quieren enviar 3 bloques de texto dentro del mismo botón, es mas sencillo hacerlo desde el código.

Wrappanel/Stackpanel



<Button.Content>

<WrapPanel>

<TextBlock Foreground="Red">Hola</TextBlock>

<TextBlock Foreground="Blue">Click</TextBlock>

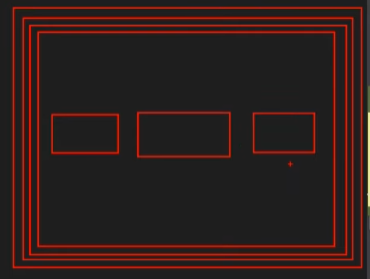
<TextBlock Foreground="Red">Enviar</TextBlock>

</WrapPanel>

</Button.Content>

Luego, lo crea desde C#.

Public MainWindow() Es el método desde el que se inicia la interfaz gráfica



Ahí representa al grid, wrappanel, button y los 3 bloques de texto dentro del mismo. Quizas no fui del todo preciso, pero son cajas dentro de cajas.

public MainWindow()

{

InitializeComponent();

Grid miGrid = new Grid();

this.Content = miGrid;

Button miBoton = new Button();

WrapPanel miWrap = new WrapPanel();

miBoton.Width=200;

miBoton.Height=100;

TextBlock txt1 = new TextBlock(); //Aquí se instancia un objeto de tipo textblock

TextBlock txt2 = new TextBlock();

TextBlock txt3 = new TextBlock();

txt1.FontSize = 15;

txt2.FontSize = 15;

txt3.FontSize = 15;

txt1.Text = " Click "; //Acá se le da el valor este

miWrap.Children.Add(txt1); //Acá le agrega el txt a Wrap

txt2.Text = " Enviar ";

miWrap.Children.Add(txt2);

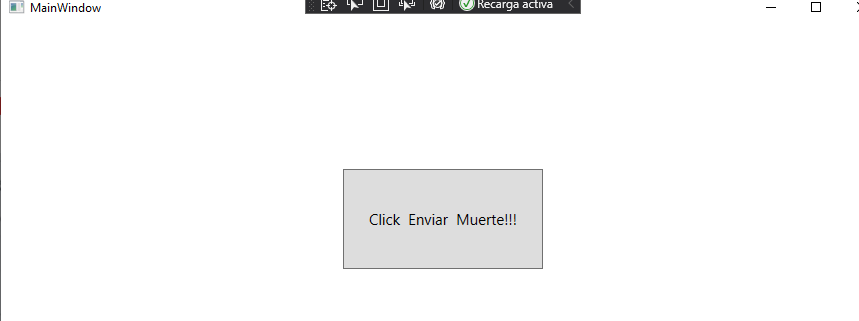
txt3.Text = " Muerte!!! ";

miWrap.Children.Add(txt3);

miBoton.Content = miWrap;

miGrid.Children.Add(miBoton); //Children porque es el hijo del Grid

}



Etiqueta Stackpanel. Organizador de elementos dentro de la interfaz gráfica.

<StackPanel>

<Button Content="Dale" HorizontalAlignment="Left" FontSize="21" Width="115" Margin="10"></Button>

<Button Content="Dale 2" Click="Button\_Click" HorizontalAlignment="Center" FontSize="21" Width="115" Margin="10"></Button>

<Button Content="Dale 3" HorizontalAlignment="Right" FontSize="21" Width="115" Margin="10"> </Button>

</StackPanel>

Los apila, uno debajo de otro.

Margin 15 (pixeles) alrededor del botón. Para cada lado

Click="Button\_Click" Eso para que el botón responda a un evento. Al poner eso, automáticamente aparece en C#:

private void Button\_Click(object sender, RoutedEventArgs e)

{

}

Lo que se programe ahí, será lo que se ejecute cuando se toque el botón.

Eventos

3 tipos de eventos: Directos (no tiene propagación), burbuja y tunelados(hacia abajo).

Eventos burbuja: Si el evento ocurre en el botón, se propaga hacia arriba.

El evento .Click es bubling se propaga hacia arriba. Así que primero se leerá por consola le has dado al botón 2, luego le has dado al panel. <StackPanel Name="Panel" ButtonBase.Click="Panel\_Click">

<Button Content="Dale" HorizontalAlignment="Left" FontSize="21" Width="115" Margin="10"></Button>

<Button Content="Dale 2" Click="Button\_Click" HorizontalAlignment="Center" FontSize="21" Width="115" Margin="10"></Button>

<Button Content="Dale 3" HorizontalAlignment="Right" FontSize="21" Width="115" Margin="10"> </Button>

</StackPanel>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Console.WriteLine("Le has dado al botón 2");

}

private void Panel\_Click(object sender, RoutedEventArgs e)

{

Console.WriteLine("Le has dado al panel");

}

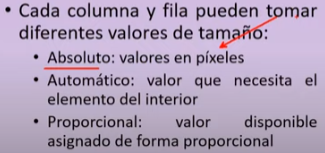




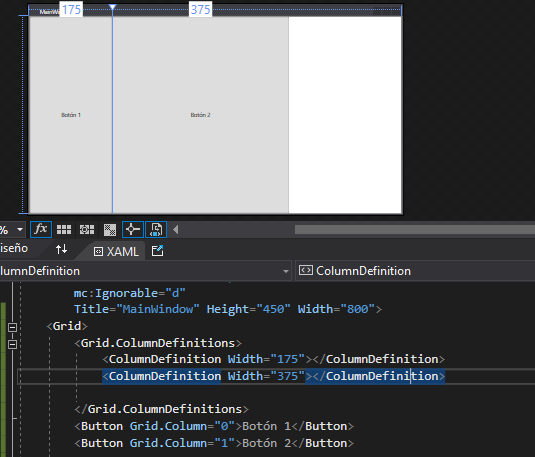
Luego, cambia el tipo de evento, de ButtonBase o Click a PreviewMouseLeftButtonDown. Ahora, primero, al tocar el botón 2 va a ejecutar el panel, y luego el botón 2.

Grid

Grillas. Divide una interfaz gráfica en celdas (Columnas y filas)



Gridcolumndefinitions y gridroydefinitions. Se abren y se cierran.



El 0 y el 1 entrecomillados refieren a la columna en la que se van a ubicar. Se empieza contando por el 0.

El 175 375 es el ancho expresado en pixeles, y usando un valor absoluto.

A la derecha hay un espacio blanco, restante.

Si uno le pone a una columna un valor en pixeles, y a otra un asterisco, esa segunda columna va a ocupar lo que quede de ancho.

Si uno simplemente les pone a todas un asterisco van a ocupar espacios “equitativos” o quizás igualitarios.

Si no se pone nada, es automático. Se ajusta al ancho necesario para que entre el contenido.

También se le puede poner a una \* y a la segunda 2\*. La segunda va a ocupar el doble de espacio.

Para las filas. Se agrega grid.rowdefinition y adentro el rowdefinition.

Si al botón no se le espcifica la fila, se ubica en la primera.

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Button Grid.Column="0" Grid.Row="1">Botón 1</Button>

<Button Grid.Column="1">Botón 2</Button>

<Grid.RowDefinitions>

<RowDefinition Height="\*"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

</Grid.RowDefinitions>

</Grid>

ColumnSpan significa cuantas columnas debe ocupar esa columna. Si hay 4 columnas y vos le pones ColumnSpan=”4”. Esa columna va a ocupar todo el ancho. Se expande.

>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

</Grid.RowDefinitions>

<Button Grid.Column="0" Grid.Row="0">Botón 1</Button>

<Button Grid.Column= "1" Grid.Row="0">Botón 2</Button>

<Button Grid.Column="2" Grid.Row="0">Botón 3</Button>

<Button Grid.Column="3" Grid.Row="0">Botón 4</Button>

<Button Grid.Column="0" Grid.Row="1">Botón 5</Button>

<Button Grid.Column= "1" Grid.Row="1">Botón 6</Button>

<Button Grid.Column="2" Grid.Row="1">Botón 7</Button>

<Button Grid.Column="3" Grid.Row="1">Botón 8</Button>

<Button Grid.Column="0" Grid.Row="2">Botón 9</Button>

<Button Grid.Column= "1" Grid.Row="2">Botón 10</Button>

<Button Grid.Column="2" Grid.Row="2">Botón 11</Button>

<Button Grid.Column="3" Grid.Row="2">Botón 12</Button>

<TextBlock Grid.ColumnSpan="4" Grid.Row="3" HorizontalAlignment="center" VerticalAlignment="center" FontSize="40"> Pepito</TextBlock>

</Grid>

Primero se crean las columnas, luego las filas, luego se ubica a los botones allí y luego crea un textblock, al que le pone el columnspan.

Cuando se ejecuta las líneas azules no se ven.



Grids anidados.

<Grid Grid.Column="0" Grid.Row="3">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Button Grid.Column="0" Grid.Row="0">Botón 1b</Button>

<Button Grid.Column= "1" Grid.Row="0">Botón 2b</Button>

<Button Grid.Column="2" >Botón 3b</Button>

<Button Grid.Column="3" Grid.Row="0">Botón 4b</Button>

</Grid>

</Grid>

