## 3.2. Controles de diseño

Los controles de diseño son **contenedores** que permiten definir el **posicionamiento** de cada uno de los **controles gráficos** que contienen. Existen varios tipos de controles de diseño en el framework .NET que responden a la gran mayoría de necesidades en términos de definición de interfaz gráfica.

### 3.2.1. Grid

El control Grid representa una **cuadrícula** (o matriz) en la que se definen filas y columnas. Cada uno de los componentes gráficos se ubica en esta matriz mediante un índice de fila y de columna que puede extenderse sobre varias filas y/o columnas.

Observación: el control Grid es el contenedor principal por defecto definido en el modelo de ventanas WPF de Visual Studio.

Para definir una cuadrícula con tres filas y tres columnas se escribe el siguiente código:

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

</Grid>

Las propiedades **RowDefinitions** y **ColumnDefinitions** del control Grid son **colecciones** que pueden contener, respectivamente, tantos objetos RowDefinition o ColumnDefinition como sea necesario.

En tiempo de ejecución, la ventana que contiene esta definición de cuadrícula puede aparecer vacía… En efecto, por defecto nuestra cuadrícula es transparente y está vacía. Para visualizar la cuadrícula es posible definir su propiedad ShowGridLines a True.

<Grid ShowGridLines="True">

Es posible definir la altura de cada una de las filas mediante la propiedad Height de cada objeto RowDefinition y el ancho de cada columna mediante la propiedad Width de los objetos de tipo ColumnDefinition.

Existen tres formas de especificar estos tamaños, que son, por orden de prioridad:

* Un valor en número de píxeles.
* Auto: la fila (o columna) se adapta a su contenido. Si no tiene contenido, no será visible.
* \* (Star): el uso de este modo de medida es algo más complejo. Se asocia un coeficiente a cada medida “Star”. Este coeficiente vale 1 si no se especifica otra cosa de manera explícita. El tamaño efectivo se calcula según la siguiente fórmula:

Tamaño=Tamaño libre restante \*((Coeficiente para la fila/columna)/(suma de los coeficientes de las filas/columnas))

A continuación, se muestra un ejemplo que muestra la aplicación de los distintos tipos de medida.

<Grid ShowGridLines="True">

<Grid.RowDefinitions>

<RowDefinition Height="123"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="30"/>

<RowDefinition Height="2\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

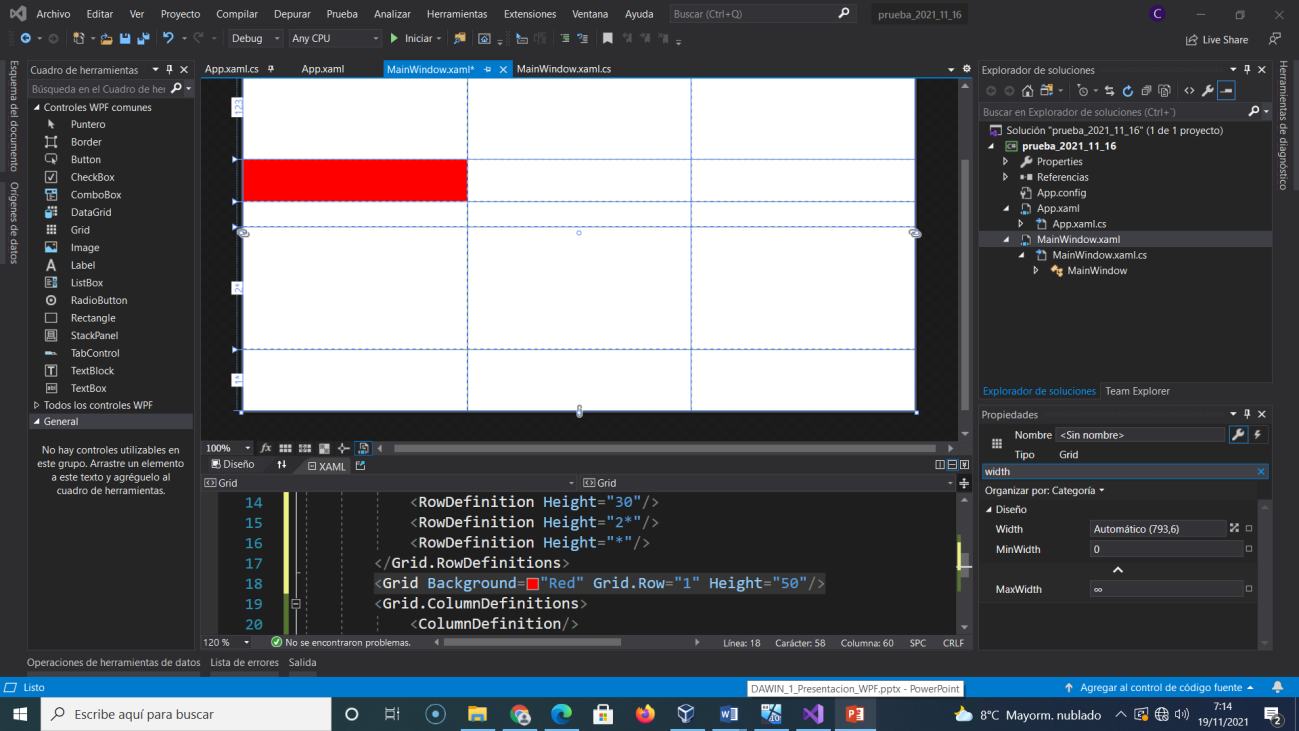
<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

</Grid>

Para **posicionar** controles en la cuadrícula es necesario declararlos y dotar de valor a cada una de las propiedades asociadas Grid.Row y Grid.Column. el valor por defecto de estas propiedades es 0, lo cual quiere decir que, si no se utilizan de manera explícita en un control, este se posicionará en la primera fila y la primera columna de la cuadrícula. <Grid Background="Red" Grid.Row="1" Height="50"/>



### 3.2.2. StackPanel

El control StackPanel permite **organizar** vertical u horizontalmente varios componentes visuales. Su propiedad **Orientation** controla el sentido en el que se apilarán los elementos y su valor por defecto es **Vertical**.

**Ejemplos con StackPanel**

Este ejemplo muestra en tres filas distintas unos caracteres:

<Grid>

<StackPanel Orientation="Vertical">

<TextBlock Text="Primera fila" />

<TextBlock Text="Segunda fila" />

<TextBlock Text="Tercera fila" />

</StackPanel>

</Grid>

En un segundo ejemplo, incluyo las tres filas anteriores en un botón y con colores diferentes:

<Grid>

<Button>

<Button.Width>200</Button.Width>

<Button.Height>60</Button.Height>

<Button.Content>

<StackPanel Orientation="Vertical">

<TextBlock Foreground="DeepPink" Text="Primera fila"/>

<TextBlock Foreground="ForestGreen" Text="Segunda fila"/>

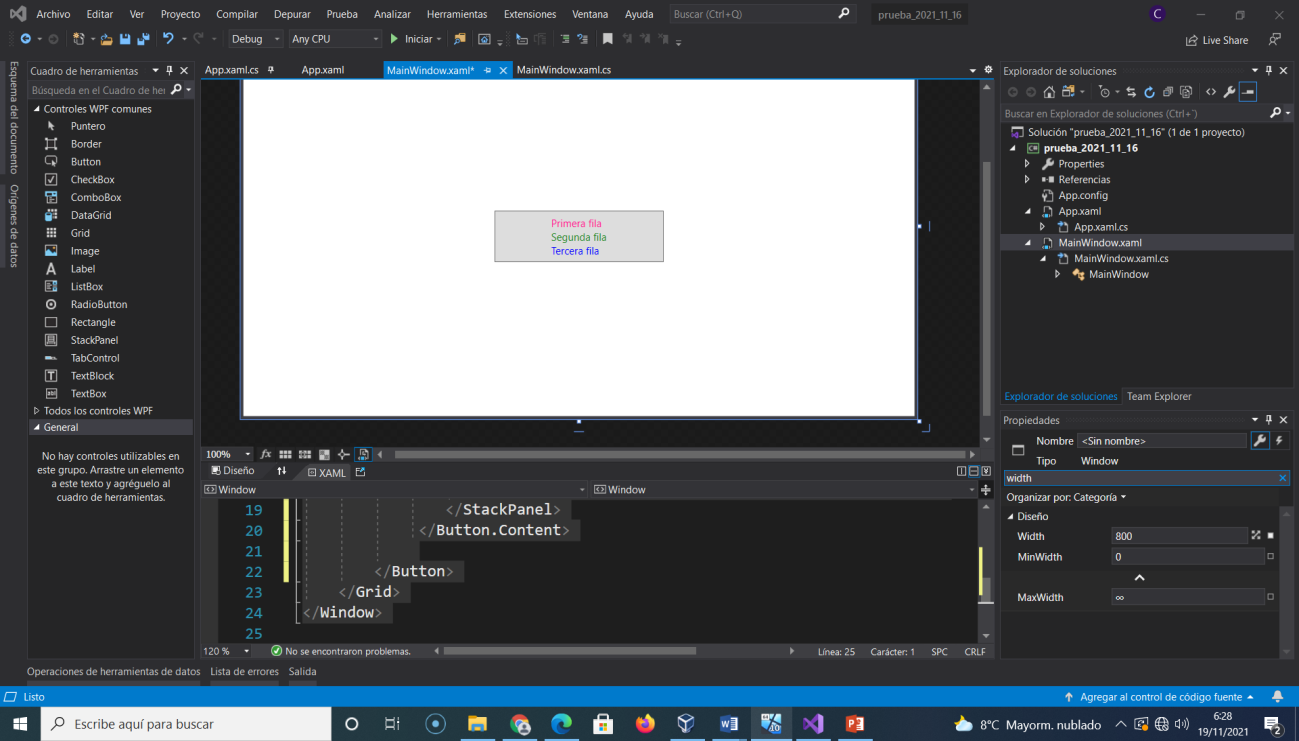
<TextBlock Foreground="Blue" Text="Tercera fila"/>

</StackPanel>

</Button.Content>

</Button>

</Grid>



### 3.2.3. DockPanel

El control DockPanel ofrece la posibilidad de anclar sus controles hijos sobre cada uno de sus cuatro lados. Para ello utiliza la propiedad asociada DockPanel.Dock sobre los controles que se desea anclar.

<Grid>

<DockPanel>

<Grid DockPanel.Dock="Right" Background="Green" Width="50"></Grid>

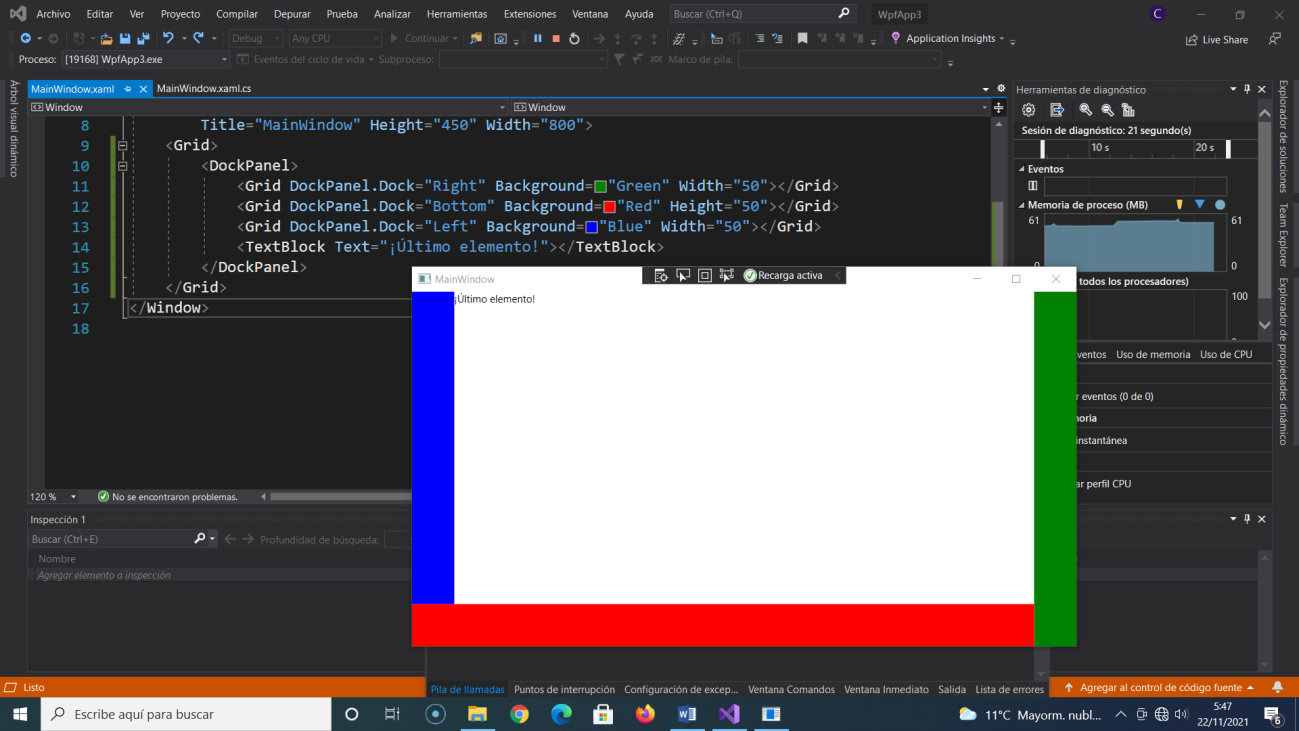
<Grid DockPanel.Dock="Bottom" Background="Red" Height="50"></Grid>

<Grid DockPanel.Dock="Left" Background="Blue" Width="50"></Grid>

<TextBlock Text="¡Último elemento!"></TextBlock>

</DockPanel>

</Grid>



Es importante destacar que los controles se anclan en función de su orden de instanciación. Esto implica que el código anterior y el siguiente ejemplo no produzcan el mismo resultado.

<Grid>

<DockPanel>

<Grid DockPanel.Dock="Bottom" Background="Red" Height="50"></Grid>

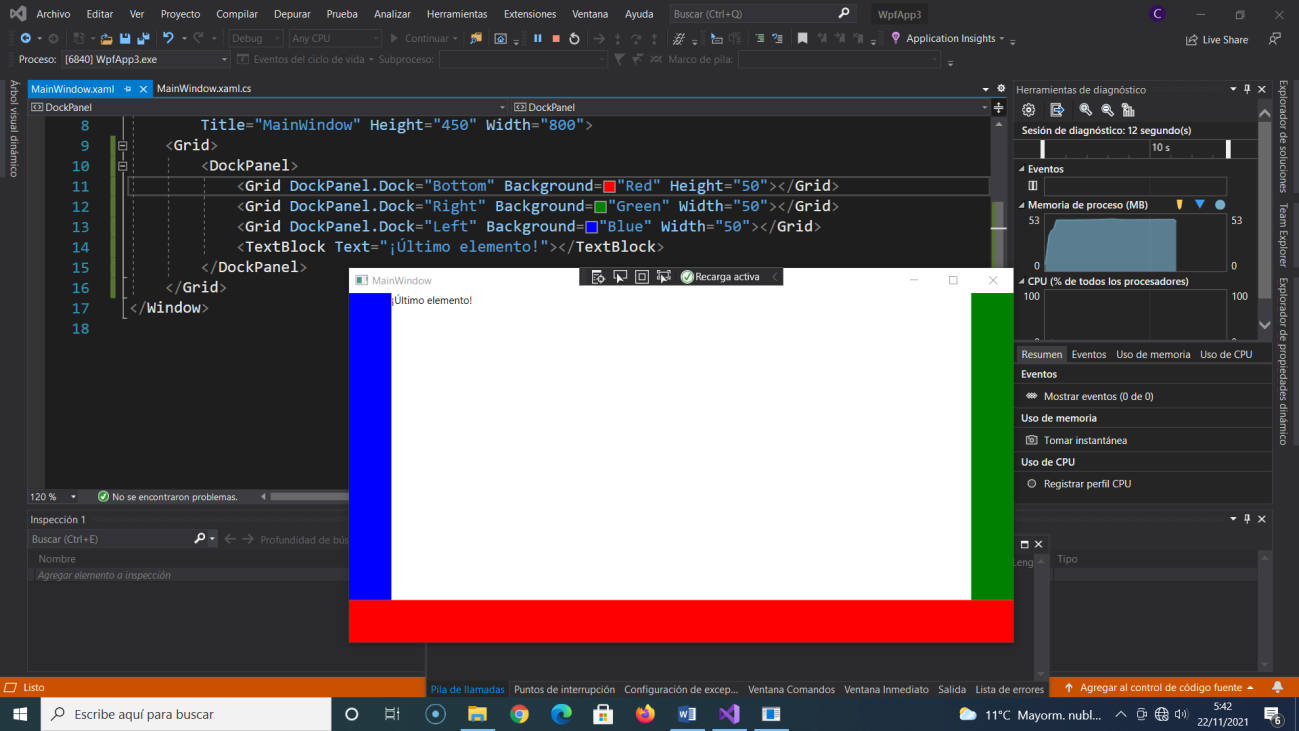
<Grid DockPanel.Dock="Right" Background="Green" Width="50"></Grid>

<Grid DockPanel.Dock="Left" Background="Blue" Width="50"></Grid>

<TextBlock Text="¡Último elemento!"></TextBlock>

</DockPanel>

</Grid>



Es posible anclar varios controles al mismo borde: éstos se apilarán de manera relativa respecto a este borde, en orden de instanciación.

<Grid>

<DockPanel>

<Grid DockPanel.Dock="Right" Background="Green" Width="50"></Grid>

<Grid DockPanel.Dock="Bottom" Background="Yellow" Height="50"></Grid>

<Grid DockPanel.Dock="Bottom" Background="Red" Height="50"></Grid>

<Grid DockPanel.Dock="Bottom" Background="Yellow" Height="50"></Grid>

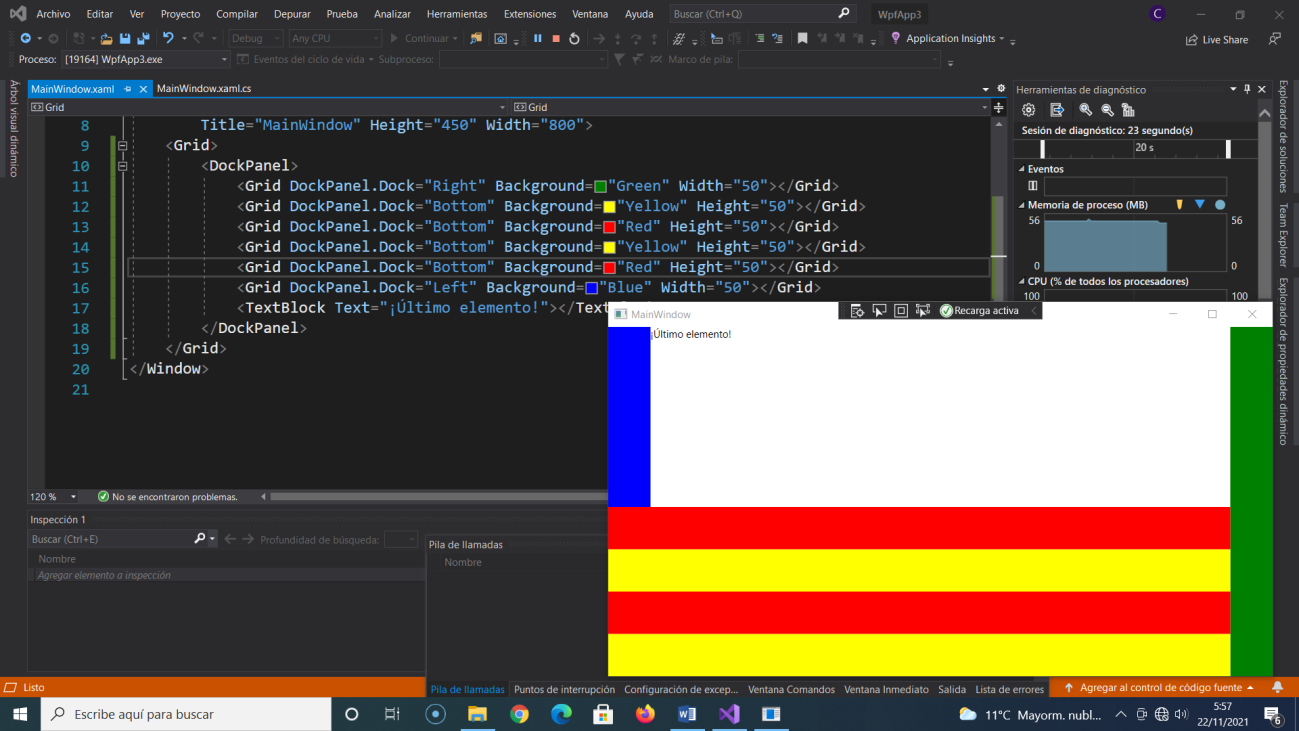
<Grid DockPanel.Dock="Bottom" Background="Red" Height="50"></Grid>

<Grid DockPanel.Dock="Left" Background="Blue" Width="50"></Grid>

<TextBlock Text="¡Último elemento!"></TextBlock>

</DockPanel>

</Grid>



La clase DockPanel tiene la propiedad LastChildFill que permite configurar cuál debe ser el comportamiento de su último control hijo. Si esta propiedad tiene el valor True entonces el elemento cubrirá la totalidad del espacio restante en el DockPanel.

### 3.2.4. WrapPanel

El control WrapPanel podría describirse como un StackPanel mejorado (si bien la clase WrapPanel no hereda de StackPanel). Del mismo modo que el control StackPanel, permite organizar varios componentes gráficos.

La verdadera diferencia entre ambos controles reside en el hecho de que el WrapPanel organiza el contenido en una nueva fila (o columna) cuando el espacio disponible es insuficiente para la representación deseada.

Cambio el ancho de la ventana de 800 a 400.

<Grid>

<WrapPanel>

<Label Content="1" FontSize="20" Height="50" Width="100" BorderThickness="1" BorderBrush="Black" />

<Label Content="2" FontSize="20" Height="50" Width="100" BorderThickness="1" BorderBrush="Black" />

<Label Content="3" FontSize="20" Height="50" Width="100" BorderThickness="1" BorderBrush="Black" />

<Label Content="4" FontSize="20" Height="50" Width="100" BorderThickness="1" BorderBrush="Black" />

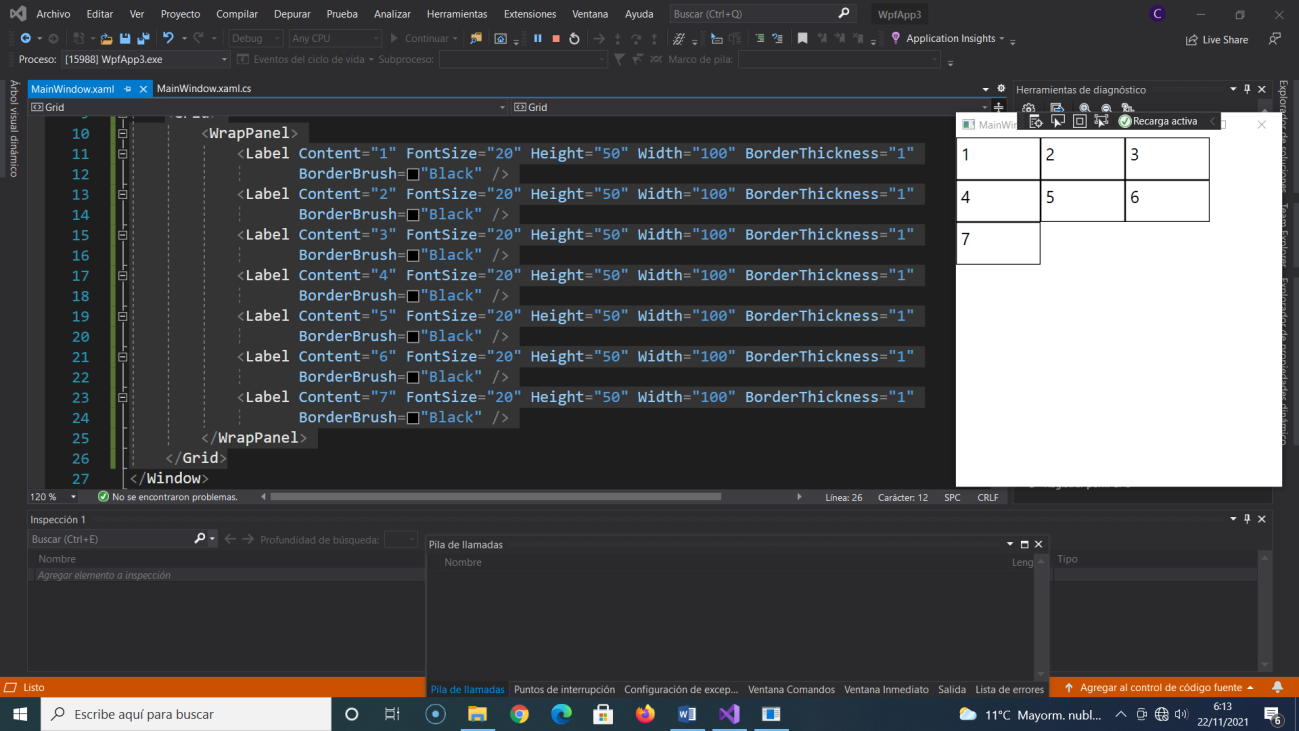
<Label Content="5" FontSize="20" Height="50" Width="100" BorderThickness="1" BorderBrush="Black" />

<Label Content="6" FontSize="20" Height="50" Width="100" BorderThickness="1" BorderBrush="Black" />

<Label Content="7" FontSize="20" Height="50" Width="100" BorderThickness="1" BorderBrush="Black" />

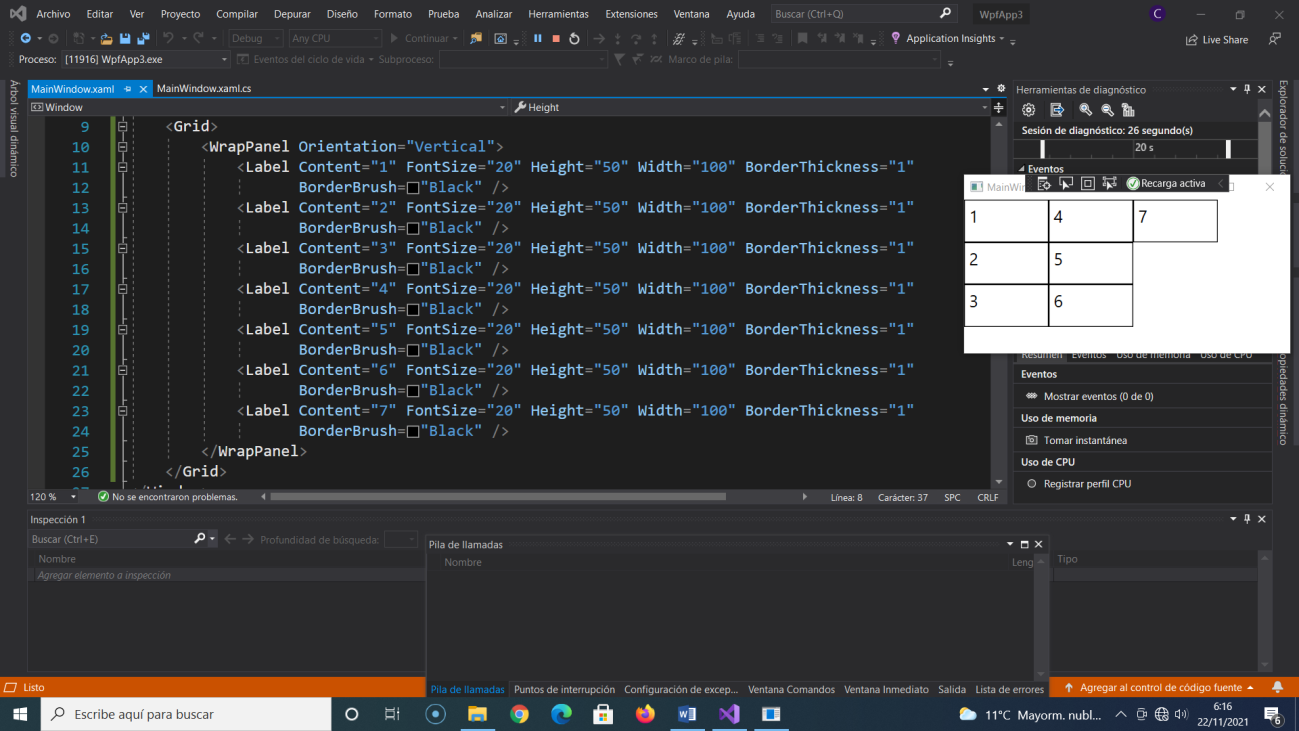
</WrapPanel>

</Grid>



Por defecto, la propiedad Orientation del WrapPanel es Horizontal. Cambiando este valor a Vertical (cambio el alto de la ventana de 450 a 250) la ventana se verá así:

<WrapPanel Orientation="Vertical">



El control WrapPanel ocupa, por defecto, todo el espacio disponible en su control padre, lo que permite tener un comportamiento dinámico si se redimensiona el elemento padre en tiempo de ejecución, el WrapPanel se redimensiona también automáticamente y sus elementos hijo pueden reposicionarse en función de las nuevas dimensiones.

### 3.2.5. Canvas

El control Canvas está pensado para contener controles gráficos posicionados de manera absoluta.

El posicionamiento de los elementos se implementa mediante las propiedades asociadas correspondientes Canvas.Top, Canvas.Left, Canvas.Right y Convas.Bottom. estas propiedades permiten, respectivamente, especificar la distancia entre el control posicionado y el borde del control Canvas. El siguiente código posiciona dos rectángulos, uno a 15 píxeles de los bordes izquierdo y superior, y el otro a 15 píxeles de los bordes derecho e inferior.

<Grid>

<Canvas>

<Rectangle Width="50" Height="30" Fill="Green"

Canvas.Top="15" Canvas.Left="15" />

<Rectangle Width="50" Height="30" Fill="red"

Canvas.Right="15" Canvas.Bottom="15" />

</Canvas>

</Grid>

