## 3.3. Controles de representación de datos

Cualquier aplicación gráfica necesita mostrar datos e información, bien sean de tipo texto o imagen, temporales o permanentes. En WPF existen varios controles disponibles para gestionar estor casos.

### 3.3.1. TextBlock

El control **Textblock** es el control mejor adaptado para representar texto. Su contenido se especifica mediante la propiedad **Text**.

<TextBlock Text=”Desarrollo con Visual Studio y C#” />

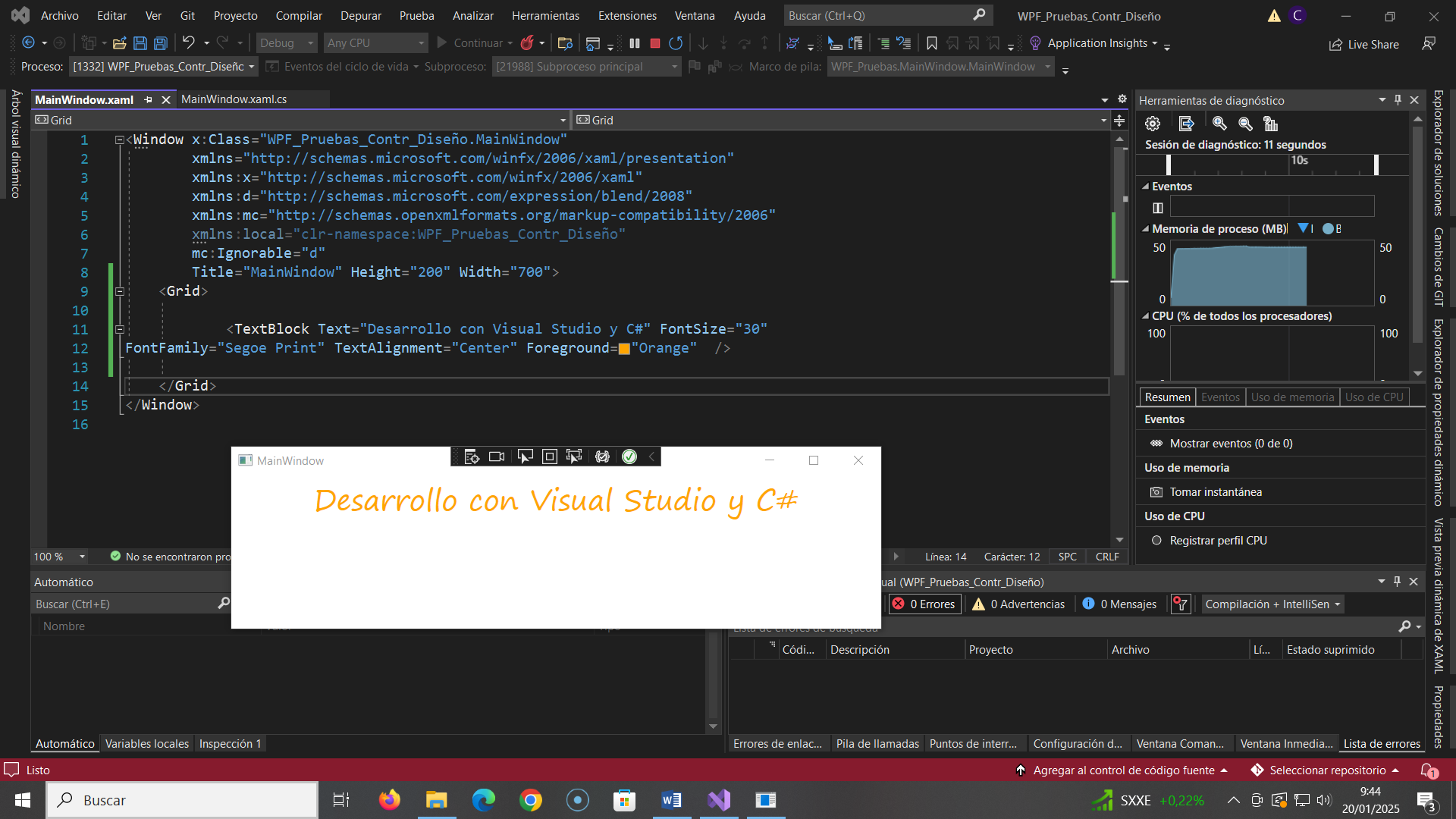
Existen otras propiedades que permiten controlar la manera en la que se representa el texto: color, tipo de letra o incluso la gestión de los saltos de línea.

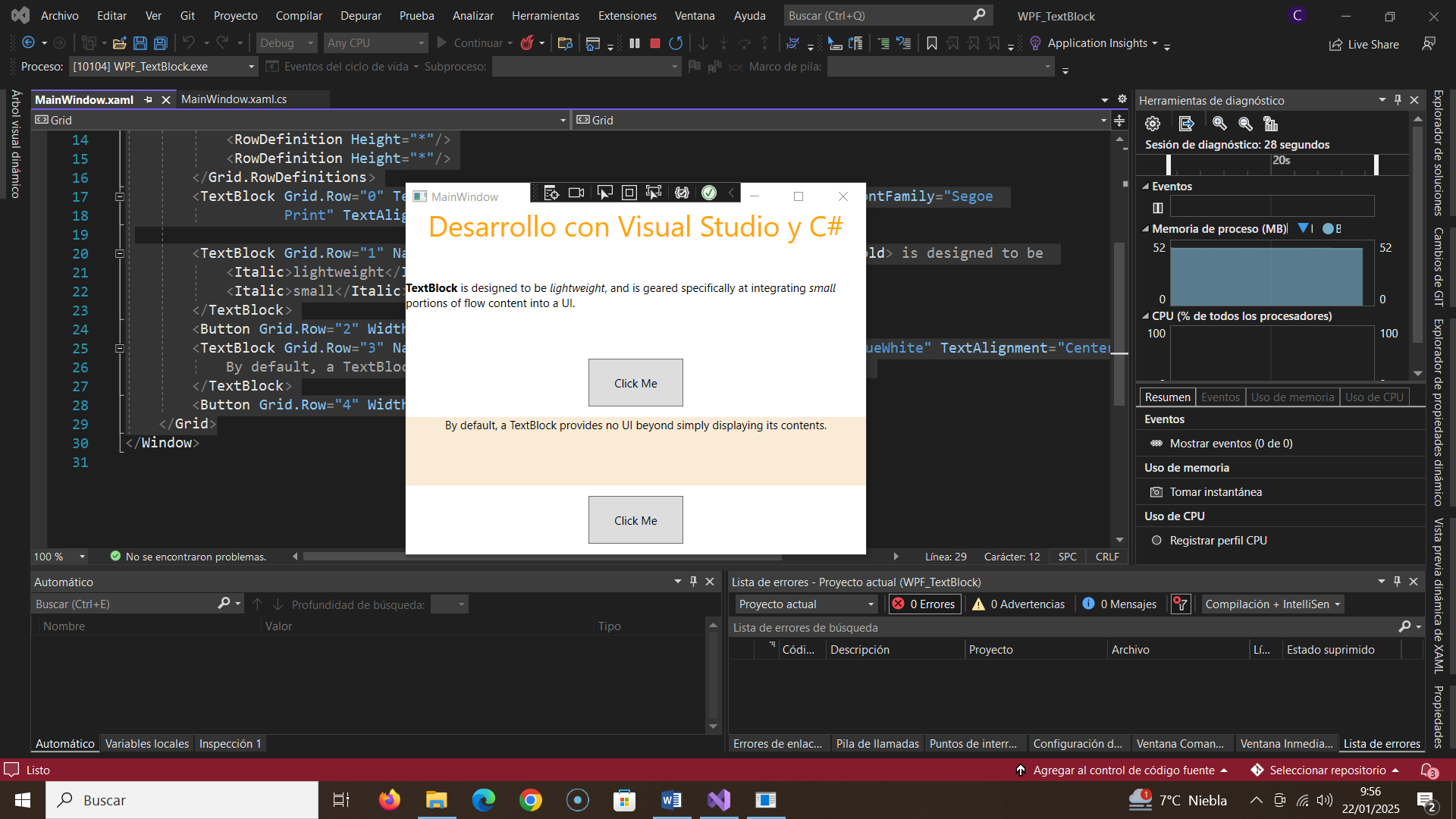
Las principales se enumeran en la siguiente tabla:

|  |  |
| --- | --- |
| **PROPIEDAD** | **CARACTERÍSTICAS** |
| Text | Define el texto que se muestra en el control |
| FontFamily | Define el tipo de letra utilizado en el control |
| FontSize | Define el tamaño de la tetra para mostrar el contenido. |
| FontStyle | Define el estilo de la letra: cursiva, normal u oblicua. |
| FontWight | Define el grosor de la fuente utilizada para el contenido. |
| Foreground | Define el color del texto. |
| TextAlignment | Define la alineación del texto (izquierda, derecha, centrada, justificada). |

<TextBlock Text="Desarrollo con Visual Studio y C#" FontSize="30"

FontFamily="Segoe Print" TextAlignment="Center" Foreground="Orange" />





Title="MainWindow" Height="400" Width="500">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Text="Desarrollo con Visual Studio y C#" FontSize="30" FontFamily="Segoe

Print" TextAlignment="Center" Foreground="Orange"/>

<TextBlock Grid.Row="1" Name="textBlock1" TextWrapping="Wrap"><Bold>TextBlock</Bold> is designed to be

<Italic>lightweight</Italic>, and is geared specifically at integrating

<Italic>small</Italic> portions of flow content into a UI.

</TextBlock>

<Button Grid.Row="2" Width="100" Height="50" Margin="10">Click Me</Button>

<TextBlock Grid.Row="3" Name="textBlock2" TextWrapping="Wrap" Background="AntiqueWhite" TextAlignment="Center">

By default, a TextBlock provides no UI beyond simply displaying its contents.

</TextBlock>

<Button Grid.Row="4" Width="100" Height="50" Margin="10">Click Me</Button>

</Grid>

### 3.3.2. Label

El control **Label**, en su uso más simple, es similar a un control TextBlock. No tiene propiedad Text sino una propiedad **Content**. El motivo de esta diferencia es el hecho de que el control Label **puede contener cualquier tipo de control**.

Este control implementa también una funcionalidad bastante interesante: permite **asignar el foco** a un control asociado mediante las **Access Keys**.

Las Access Keys son combinaciones de teclas que permiten acceder a un control. En Windows, esta combinación implica, generalmente, el uso de la tecla [Alt] algún carácter alfabético. Para utilizar esta funcionalidad es preciso:

* Definir un Label y asignar valor a su propiedad Content con una cadena de caracteres cuya letra esté precedida por el símbolo \_. Esta letra será a continuación la que se utilizará en la combinación de teclas [Alt]+ [letra].
* Definir un objeto que se asociará al Label. Este objeto recibirá el foco cuando se utilice la Access Key.
* Vincular el Label y el objeto asociado mediante la propiedad **Target** de Label utilizando el binding siguente:

Target={Binding ElementName=<nombre del objeto asociado>}”

Cambio el Height de la ventana a 150 para que aparezcan los controles más cerca el uno del otro.

Title="MainWindow" Height="300" Width="800">

<Grid>

<!-- Creo tres filas de la misma altura-->

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<!-- En la primera fila aparece el título -->

<StackPanel Grid.Row="0" >

<TextBlock Text="Desarrollo con Visual Studio y C#" FontSize="30"

FontFamily="Segoe Print" TextAlignment="Center" Foreground="Orange" />

</StackPanel>

<!-- En la segunda se pide el apellido-->

<StackPanel Grid.Row="1" Orientation="Horizontal" Height="25">

<Label Content="\_Apellido: " Target="{Binding ElementName=tbApellido}" />

<TextBox x:Name="tbApellido" Width="150" />

</StackPanel>

<!-- En la tercera el nombre -->

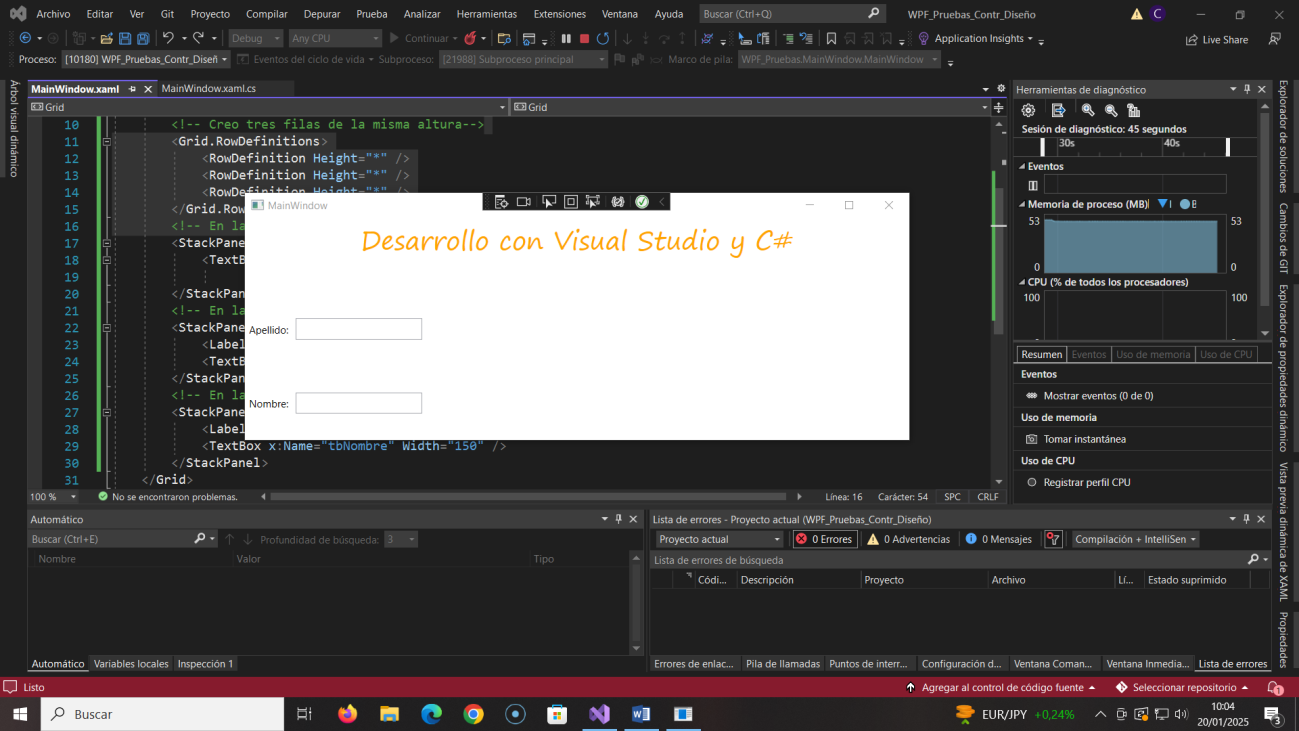
<StackPanel Grid.Row="2" Orientation="Horizontal" Height="25">

<Label Content="\_Nombre: " Target="{Binding ElementName=tbNombre}" />

<TextBox x:Name="tbNombre" Width="150" />

</StackPanel>

</Grid>



La combinación de teclas **[Alt]+A** pone el foco sobre el campo de edición **Apellido**, mientras que si pulsamos simultáneamente sobre **[Alt]+N** se asigna el foco al campo de edición **Nombre**.

### 3.3.3 Image

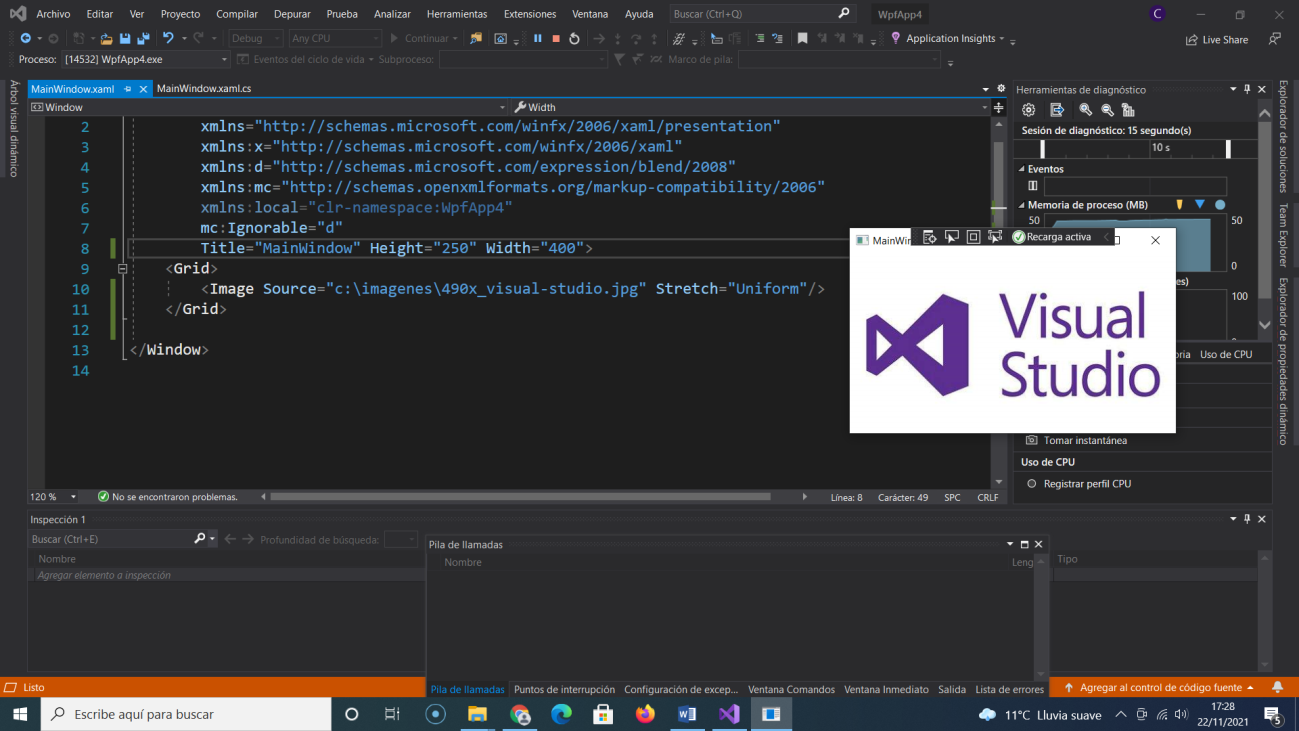
El control **Image** permite visualizar, entre otros, el contenido de archivos BMP, JPEG o PNG.

Su propiedad **Source** permite definir la ubicación de la imagen que se desea mostrar. Esta ubicación puede ser una ruta absoluta o relativa, una URI que apunte sobre alguno de los recursos de la aplicación o incluso una URL. En este último caso, la descarga de la imagen se realiza automáticamente antes de mostrarla.

La propiedad **Stretch** de este control define la manera en que se estirará la imagen en función del espacio disponible.

Ejemplo de imagen con trayectoria absoluta:

<Image Source="c:\imagenes\visualStudio.jpg" Stretch="Uniform"/>



### 3.3.4. ScrollViewer

Algunos controles necesitan más espacio que el que se les ha asignado. Habitualmente, este tipo de casos se gestiona mediante el uso de barras de desplazamiento horizontales o verticales, pero ciertos controles WPF no disponen de manera nativa, de estas barras. El control ScrollViewer es una implementación externa de estas barras que permite situarlas en cualquier control para habilitar el desplazamiento.

Title="MainWindow" Height="200" Width="400">

<Grid>

<ScrollViewer Height="100">

<TextBlock TextWrapping="Wrap" FontSize="14"

Text="1.- esto es una prueba de barras de desplazamiento

2.- esto es una prueba de barras de desplazamiento

3.- esto es una prueba de barras de desplazamiento

4.- esto es una prueba de barras de desplazamiento

5.- esto es una prueba de barras de desplazamiento

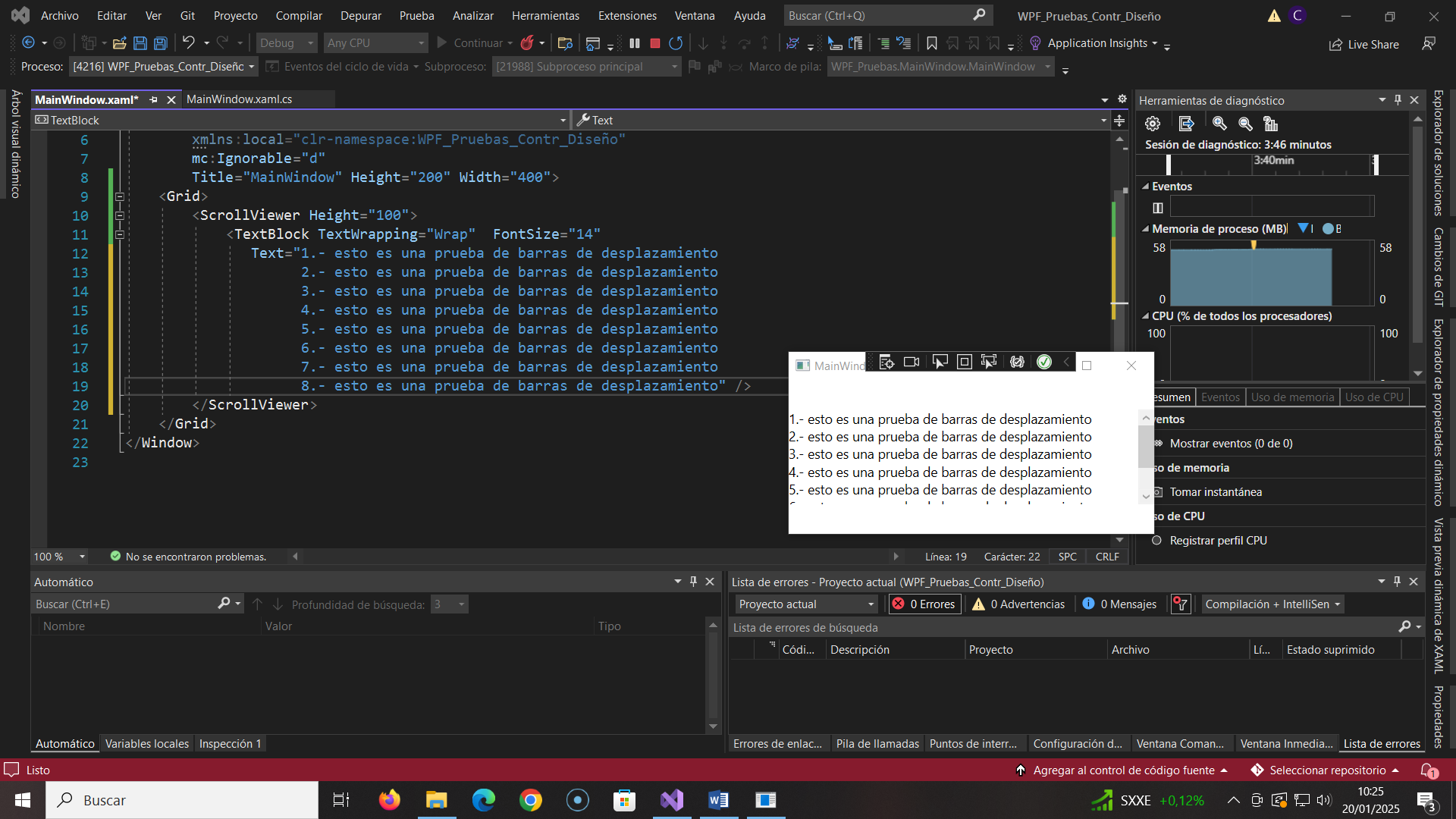
6.- esto es una prueba de barras de desplazamiento

7.- esto es una prueba de barras de desplazamiento

8.- esto es una prueba de barras de desplazamiento”/>

</ScrollViewer>

</Grid>



### 3.3.5 ItemsControl

Es posible visualizar una **lista de elementos** mediante un objeto de tipo **ItemsControl.** Este objeto puede contener una **colección** de objetos de **cualquier tipo**. Estos objetos se representan por defecto como una lista **vertical**, sin barra de desplazamiento. Cuando la representación de la lista es más grande que el control, la lista se **trunca**. Para gestionar este caso de uso, conviene situar el control de la lista dentro de un objeto de tipo **ScrollViewer**.

Observación:

Para agregar objetos de tipo **string** al control, se agrega el espacio de nombres XML **system** a la ventana mediante la siguiente declaración:  
xmlns:system=”clr-namespace:System;assembly=mscorlib”

xmlns:system="clr-namespace:System;assembly=mscorlib"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="400">

<Grid>

<ItemsControl Height="100">

<ItemsControl.Items>

<system:String>opción 1</system:String>

<system:String>opción 2</system:String>

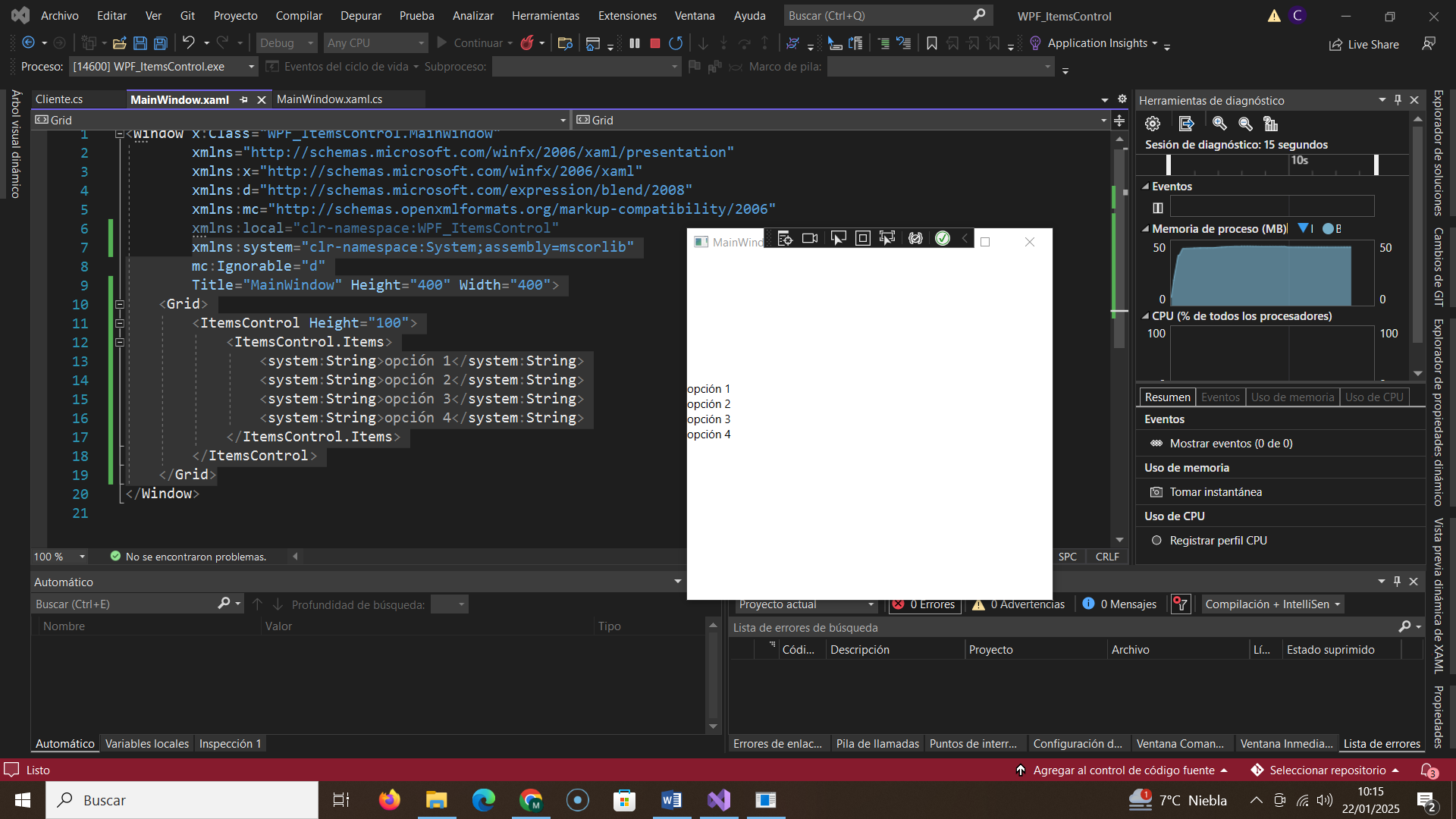
<system:String>opción 3</system:String>

<system:String>opción 4</system:String>

</ItemsControl.Items>

</ItemsControl>

</Grid>



La propiedad **Items**, al ser la propiedad por defecto del control, permite condensar el código omitiendo las declaraciones de apertura y cierre correspondientes a esta propiedad.

<ItemsControl Height="100">

<system:String>opción 1</system:String>

<system:String>opción 2</system:String>

<system:String>opción 3</system:String>

<system:String>opción 4</system:String>

</ItemsControl>

Es frecuente que los elementos contenidos por el control se definan de manera dinámica. Para ello, es preferible utilizar la propiedad **ItemsSource** asignándole valor con la declaración de un **binding**.

<ItemsControl Height=”30” ItemsSource=”{Binding ListaDeOpciones}” />

En ambos casos, cada elemento de la colección ListaDeOpciones genera un elemento en la lista visual. Ambas están vinculadas: el contexto de datos del elemento visual es el elemento correspondiente de la lista ListaDeOpciones.

Esta relación es importante cuando los datos proporcionados son de tipo complejo. La representación visual por defecto de cada elemento está determinada por el uso de su método ToString, aunque el resultado visual no siempre es el deseado.

<Grid>

<ItemsControl VerticalAlignment="Top" BorderBrush="Blue"

BorderThickness="2" Margin="10" Width="250"

Height="100">

<local:Cliente Apellidos="Moreno Blasco" Nombre="Eugenia" ImporteTotalCompras="230.89"/>

<local:Cliente Apellidos="Casajús Ansón" Nombre="Juan" ImporteTotalCompras="150.34"/>

<local:Cliente Apellidos="Roma Garriga" Nombre="Aurora" ImporteTotalCompras="36.50"/>

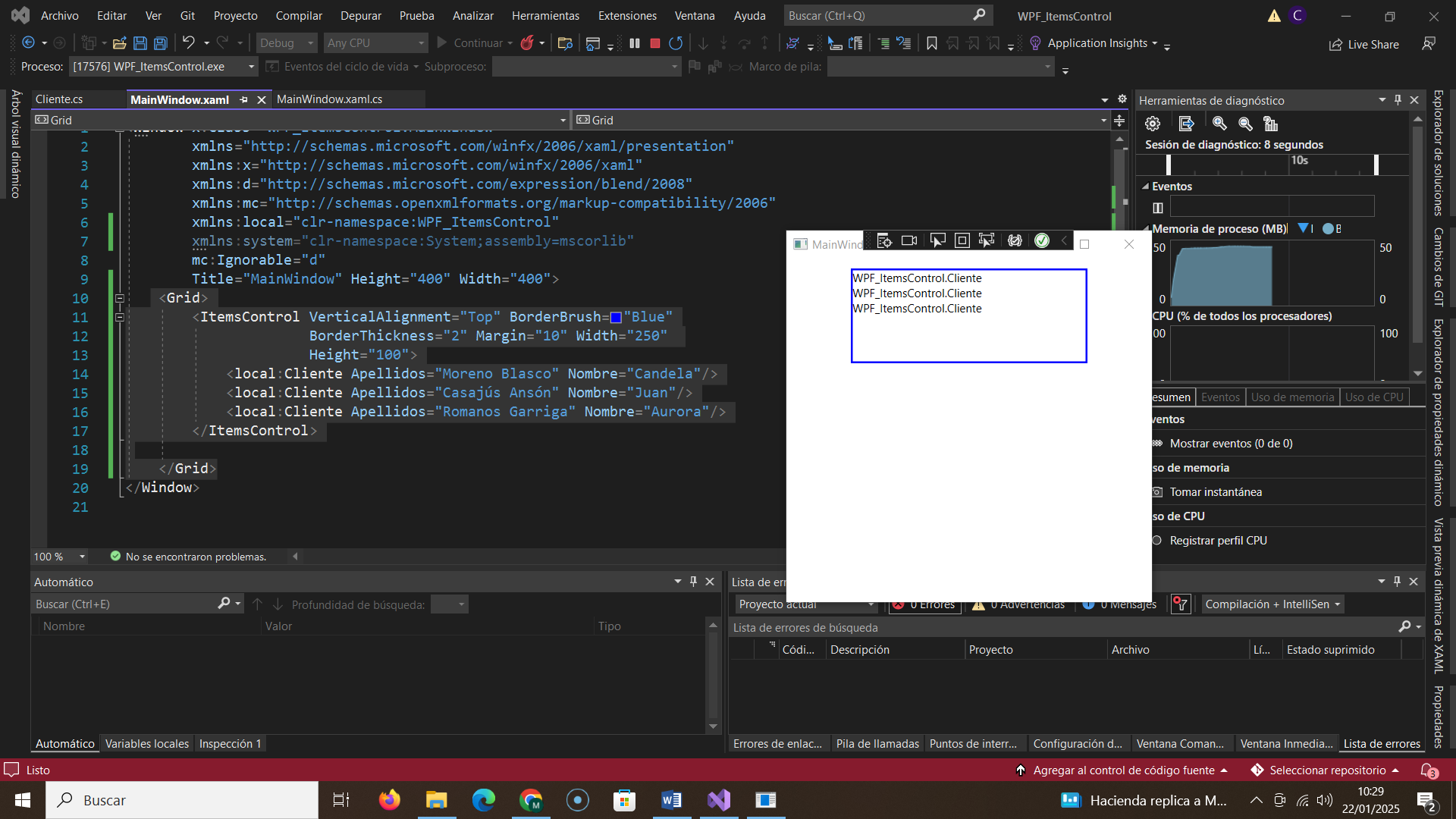
</ItemsControl>

</Grid>

Observación:

El espacio de nombres **local** se define previamente en la raíz del archivo.

La representación visual del control es la siguiente:



Para mejorar la representación visual de los datos se utiliza la propiedad **ItemTemplate** del control. Esta propiedad acepta como valor un objeto del tipo **DataTemplate**, que define un modelo visual para el contexto de datos.

El siguiente código XAML define una cuadrícula que contiene dos filas y dos columnas en las que se sitúan las propiedades de la clase Cliente mediante expresiones de bindign.

<Grid>

<ItemsControl VerticalAlignment="Top" BorderBrush="Blue"

BorderThickness="2" Margin="10" Width="250"

Height="100">

<local:Cliente Apellidos="Moreno Blasco" Nombre="Eugenia" ImporteTotalCompras="230.89"/>

<local:Cliente Apellidos="Casajús Ansón" Nombre="Juan" ImporteTotalCompras="150.34"/>

<local:Cliente Apellidos="Roma Garriga" Nombre="Aurora" ImporteTotalCompras="36.50"/>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0"

FontSize="18" Text="{Binding Apellidos}"/>

<TextBlock Grid.Row="0" Grid.Column="1"

FontSize="16" Text="{Binding Nombre}"/>

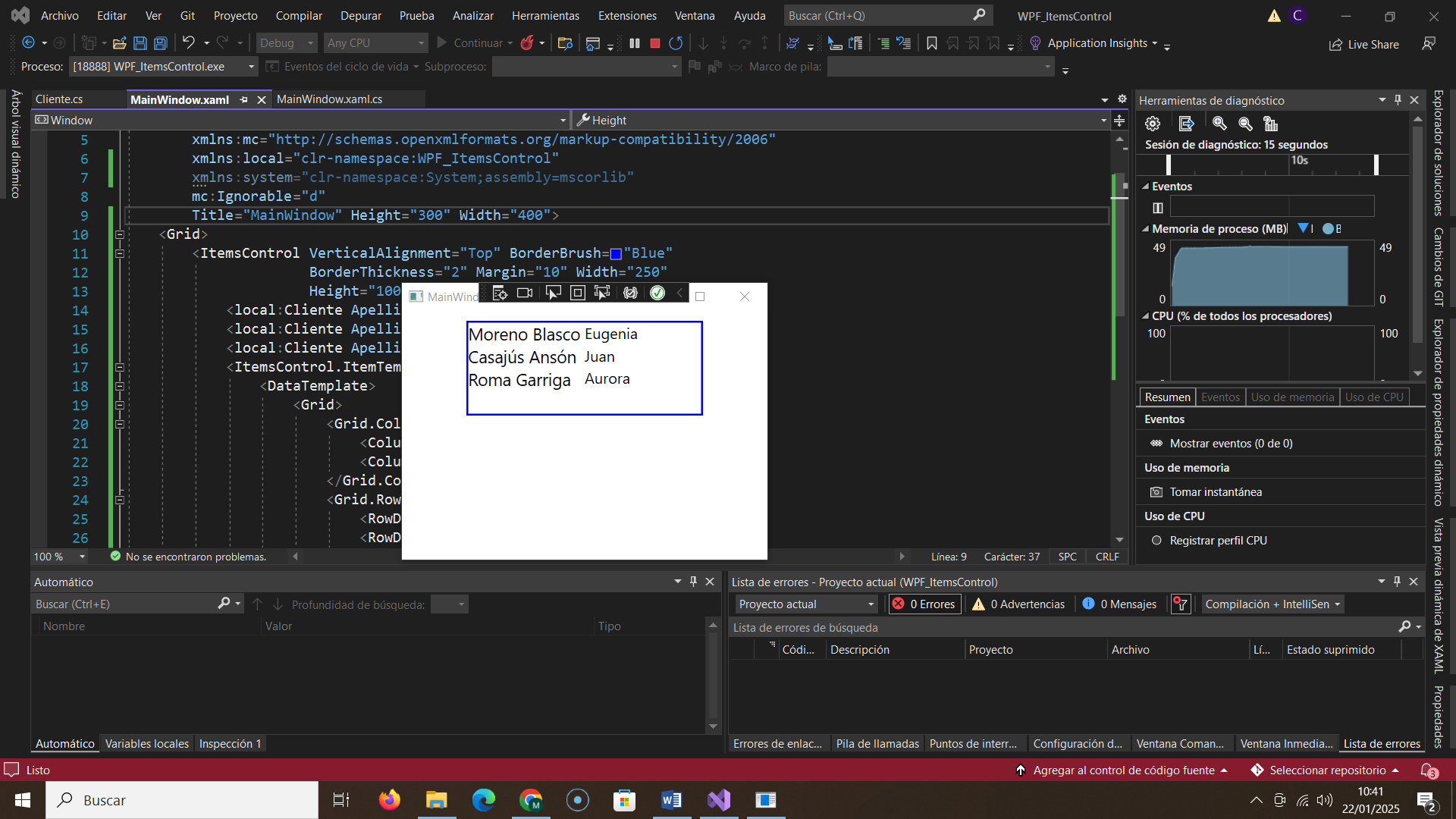
</Grid>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</Grid>



### 3.3.6 StatusBar

El control StatusBar representa una barra de estado que se sitúa, habitualmente, en la parte inferior de una ventana de aplicación. Esta barra contiene información relativa al trabajo en curso: número de página, estado de la copia de seguridad del archivo, etc.

Este control puede contener objetos de tipo StatusBarItem, cada uno de los cuales contiene a su vez los controles necesarios para la visualización de la información. Es posible definir una separación entre estos objetos mediante el control Separator.

<Grid>

<DockPanel LastChildFill="False">

<StatusBar DockPanel.Dock="Bottom" Height="30">

<StatusBarItem>Ningún archivo abierto</StatusBarItem>

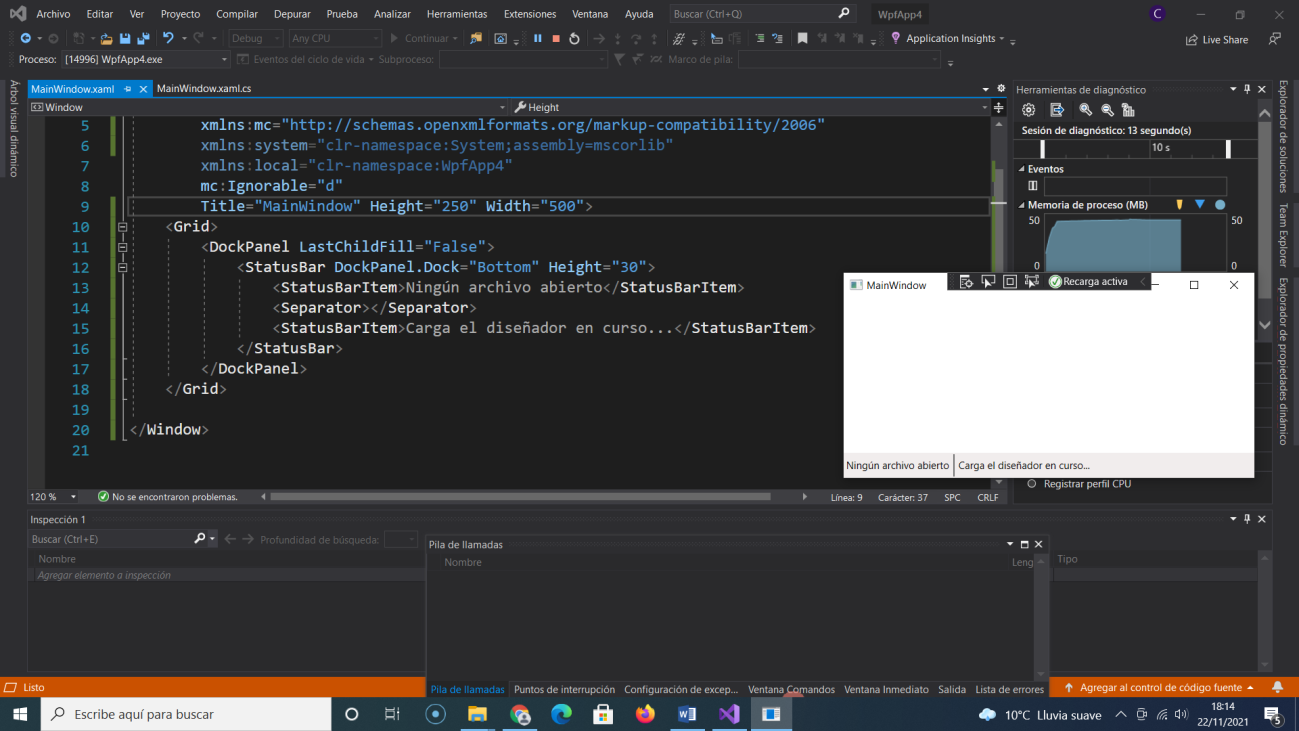
<Separator></Separator>

<StatusBarItem>Carga el diseñador en curso...</StatusBarItem>

</StatusBar>

</DockPanel>

</Grid>



### 3.3.7 ToolTip

El control ToolTip (información sobre herramientas) permite mostrar un mensaje de información cuando se pasa el cursor del ratón por encima de un control. Un caso de uso habitual es la visualización de un mensaje explicativo cuando se está introduciendo un dato erróneo (un formato incorrecto de correo electrónico, por ejemplo).

Los componentes gráficos poseen una propiedad ToolTip que se corresponde con el contenido que se desea mostrar. Ese contenido, en su forma más simple, es una cadena de caracteres, pero también puede ser más complejo y mostrar una imagen con texto complementario, por ejemplo.

<Grid>

<TextBlock Text="Este TextBlock tiene un Tooltip">

<TextBlock.ToolTip>

<ToolTip>

<TextBlock Text="Esto es un ToolTip"></TextBlock>

</ToolTip>

</TextBlock.ToolTip>

</TextBlock>

</Grid>

