## 3.4. Controles de edición de texto

Los controles de edición de texto son puntos de entrada privilegiados para registrar información por parte de los usuarios. Ofrecen cierta libertad en cuanto al contenido y constituyen verdaderos puntos de creación de datos.

### 3.4.1. TextBox

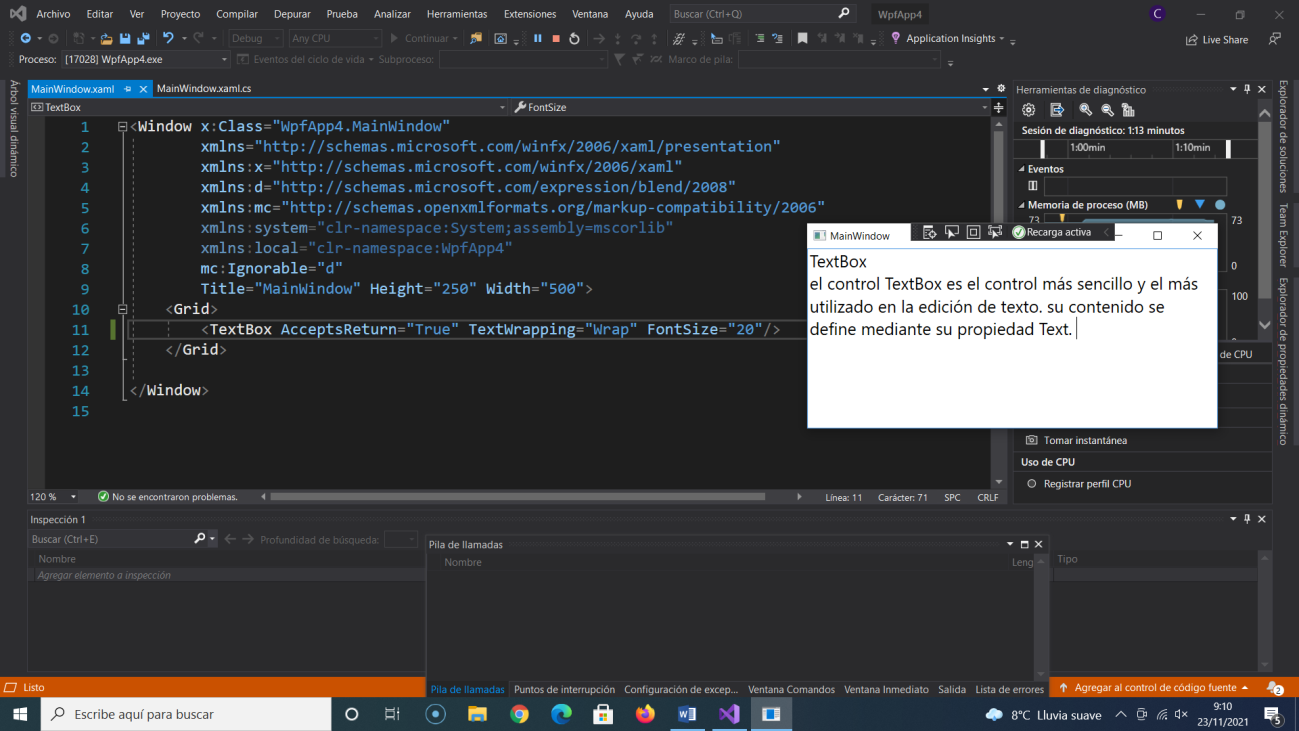
El control **TextBox** es el control más sencillo y el más utilizado en la edición de texto. Su contenido se define mediante su propiedad **Text**. Por defecto, es imposible saltar de línea en este control.

La propiedad **AcceptsReturn**, cuando se define a **True**, permite aceptar un retorno de línea mediante la tecla Enter. Existe otra propiedad que permite insertar un salto de línea automática cuando el texto es más largo que el control: **TextWrapping**. Basta con definir su valor a **Wrap** para obtener el resultado esperado.

El siguiente código permite obtener el siguiente resultado:

Texto ejemplo:

<TextBox AcceptsReturn="True" TextWrapping="Wrap" FontSize="20"/>



### 3.4.2. RichTextBox

El texto que se ha presentado en la sección anterior puede resultar difícil de leer, pues todos los elementos de texto están formateados de la misma manera. Para resolver este problema, el control RichTextBox permite mostrar texto en formato RTF (como la aplicación WordPad de Windows). Editando este texto en WordPad y realizando un copiar/pegar hacia un componente RichTextBox es posible obtener el siguiente resultado:

En MainWindow.xaml

Title="MainWindow" Height="450" Width="500">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button x:Name="btnNegrita" Grid.Row="0" Content="Poner en negrita el texto seleccionado"

Click="btnNegrita\_Click" />

<Button x:Name="btnCursiva" Grid.Row="1" Content="Poner en cursiva el texto seleccionado"

Click="btnCursiva\_Click"/>

<RichTextBox Grid.Row="2" AcceptsReturn="True" x:Name="zonaTexto" FontSize="20"/>

</Grid>

En MainWindow.xaml.cs

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

zonaTexto.Selection.ApplyPropertyValue(FontWeightProperty, FontWeights.Bold);

}

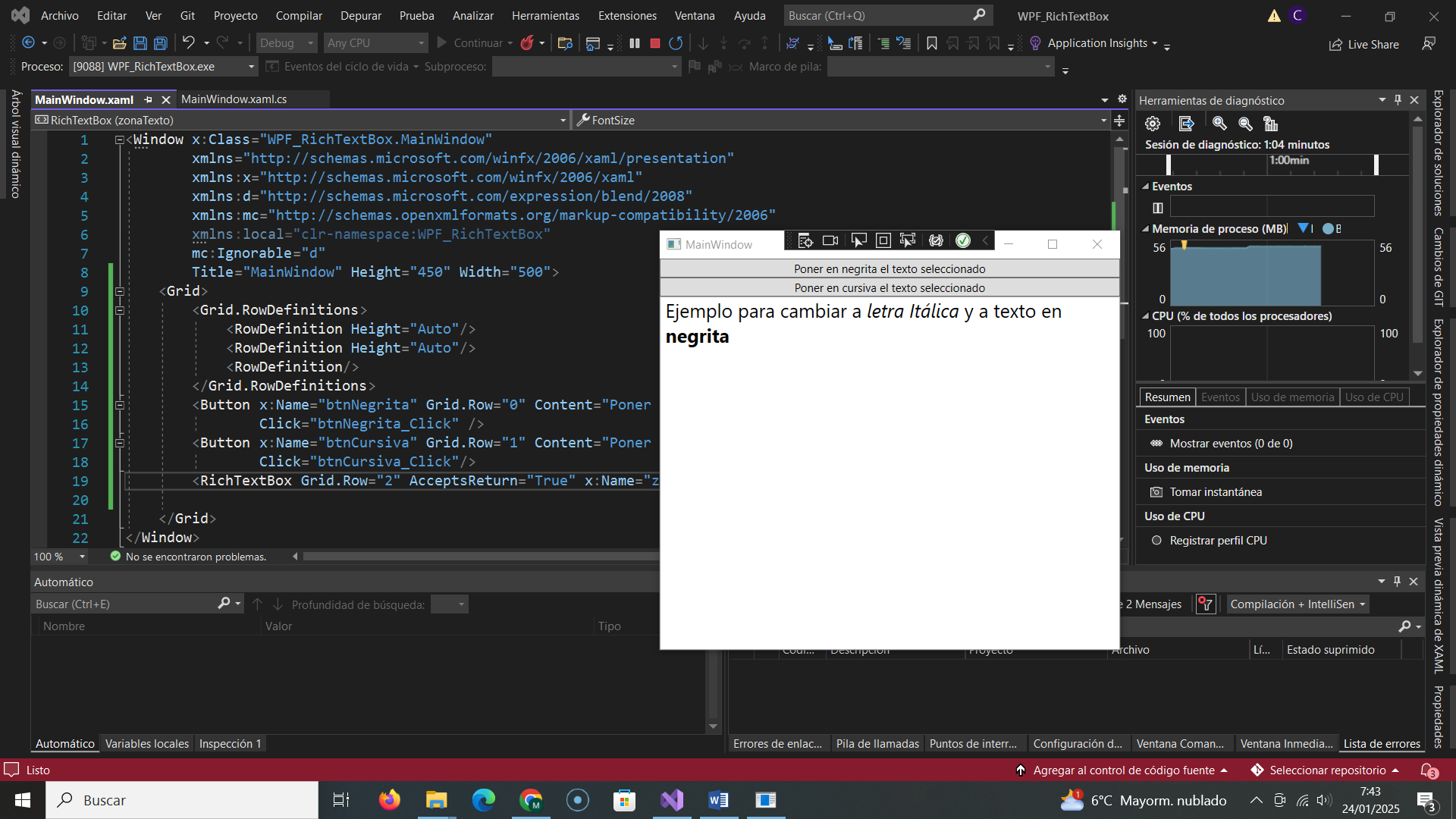
private void btnCursiva\_Click(object sender, RoutedEventArgs e)

{

zonaTexto.Selection.ApplyPropertyValue(FontStyleProperty,FontStyles.Italic);

}

}



### 3.4.3. PasswordBox

Las reglas de seguridad informática incluyen, todas ellas, la no compartición de contraseñas para evitar cualquier uso no deseado de un componente o aplicación. La primera manera de evitar que una contraseña caiga en las manos indebidas es su ocultación. Esta solución se utiliza, por otro lado, en la práctica totalidad de editores.

Con WPF, la implementación de esta ocultación puede realizarse mediante el control PasswordBox. Este control reemplaza automáticamente los caracteres escritos por el carácter especificado en su propiedad PasswordChar, preservando la privacidad del contenido escrito en su propiedad Password.

Title="MainWindow" Height="150" Width="400">

<Grid>

<Canvas>

<TextBlock Text="Password: " FontSize="20" Canvas.Left="20" Canvas.Top="25"/>

<PasswordBox Width="150" Height="30" Canvas.Right="80" Canvas.Top="20"/>

</Canvas>

</Grid>

