# 2. Uso de controles

La interfaz gráfica de una aplicación WPF está compuesta por **controles** definidos en el **código XAML** o, con menor frecuencia, en C#.

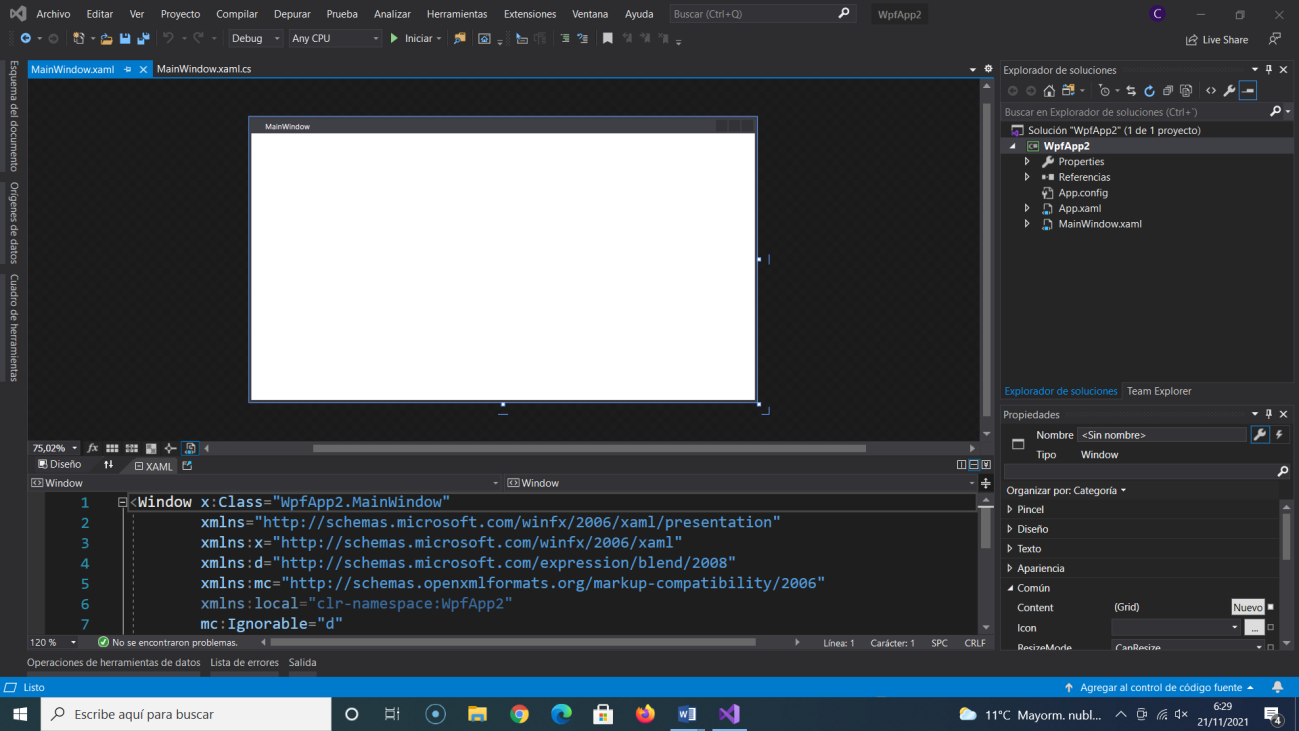
Estos controles forman parte de una jerarquía que permite tener propiedades comunes, la mayor parte del tiempo ligadas a su posicionamiento. Cada control hereda, de este modo, de un control padre al que se le agregan funcionalidades de cara a proveer un nuevo conjunto formado por un elemento visual y un comportamiento funcional asociado.

Visual Studio integra, de manera nativa, un diseñador visual que permite manipular controles de manera gráfica y, por tanto, más simple. Genera también, código XAML que define la interfaz y que es posible manipular directamente, cualquier modificación sobre el código tiene su reflejo en la parte de diseño gráfico, y viceversa.

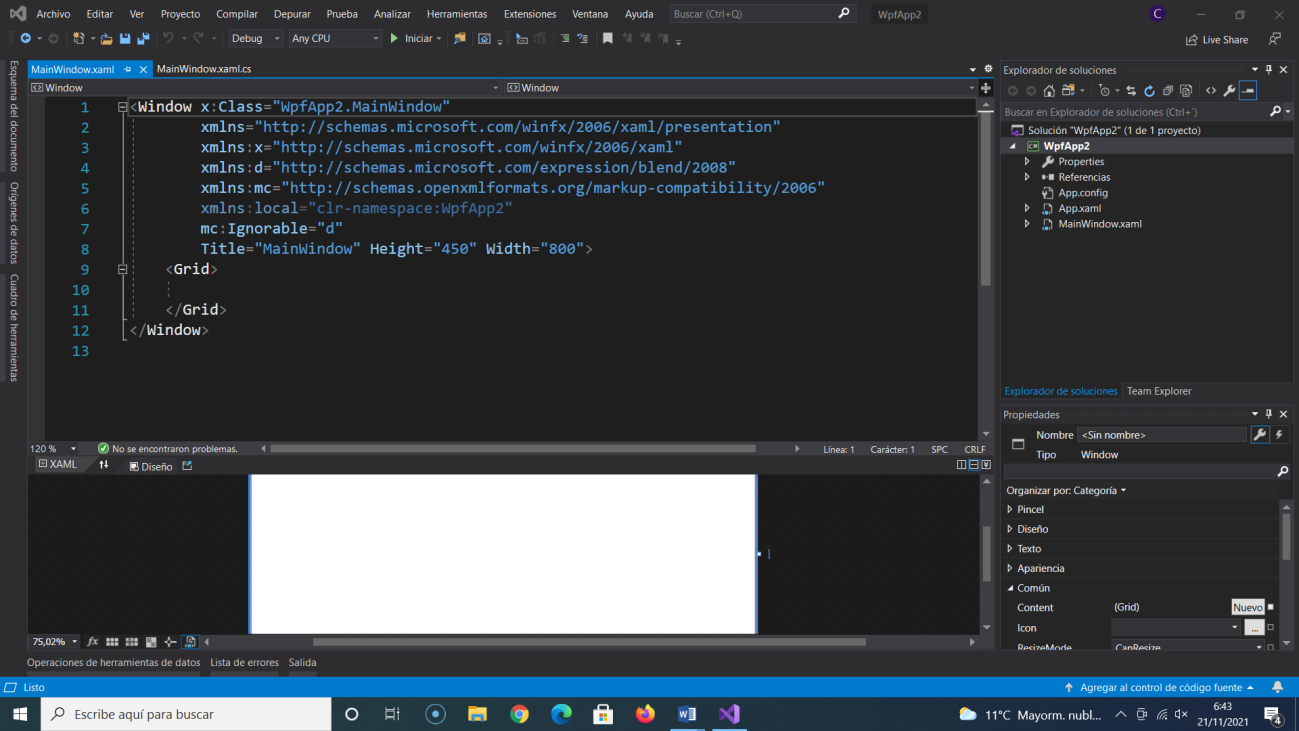
## 2.1. Agregar controles

Tras la creación de un nuevo proyecto de aplicación WPF, Visual Studio muestra, por defecto, varios elementos:

* El **Explorador de soluciones**,
* La ventana **Propiedades**,
* Una **ventana de edición** dividida en dos secciones: el diseñador visual en la zona superior, y el editor de código XAML situado en la zona inferior



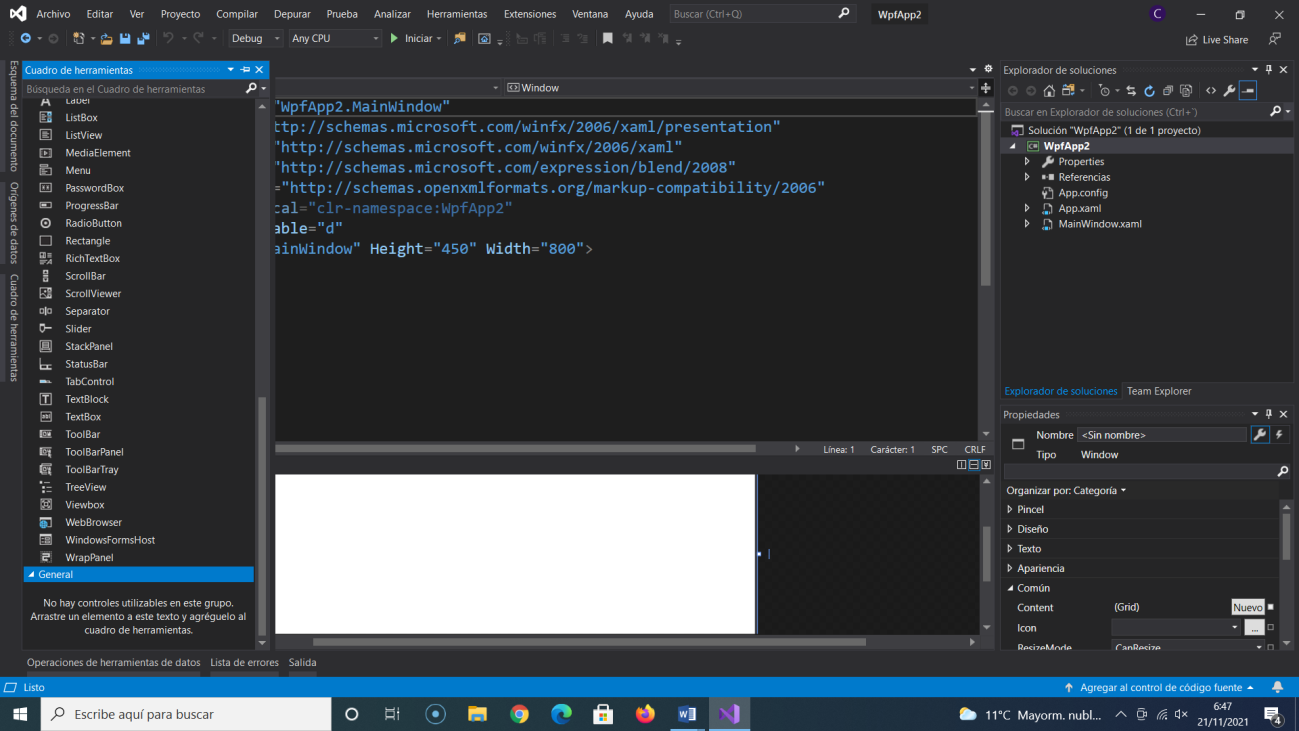
Se puede cambiar la posición de las dos secciones gracias al botón de doble flecha “Intercambiar paneles.



De esta forma, se ve completamente el código inicial XAML.

La ventana de edición está destinada a la modificación de la ventana principal de la aplicación, llamada **MainWindow**. El diseñador visual presenta una vista de esta ventana.

Varias ventanas ocultas automáticamente, se posicionan en la zona izquierda del entorno. La más importante es la ventana **Cuadro de herramientas**.

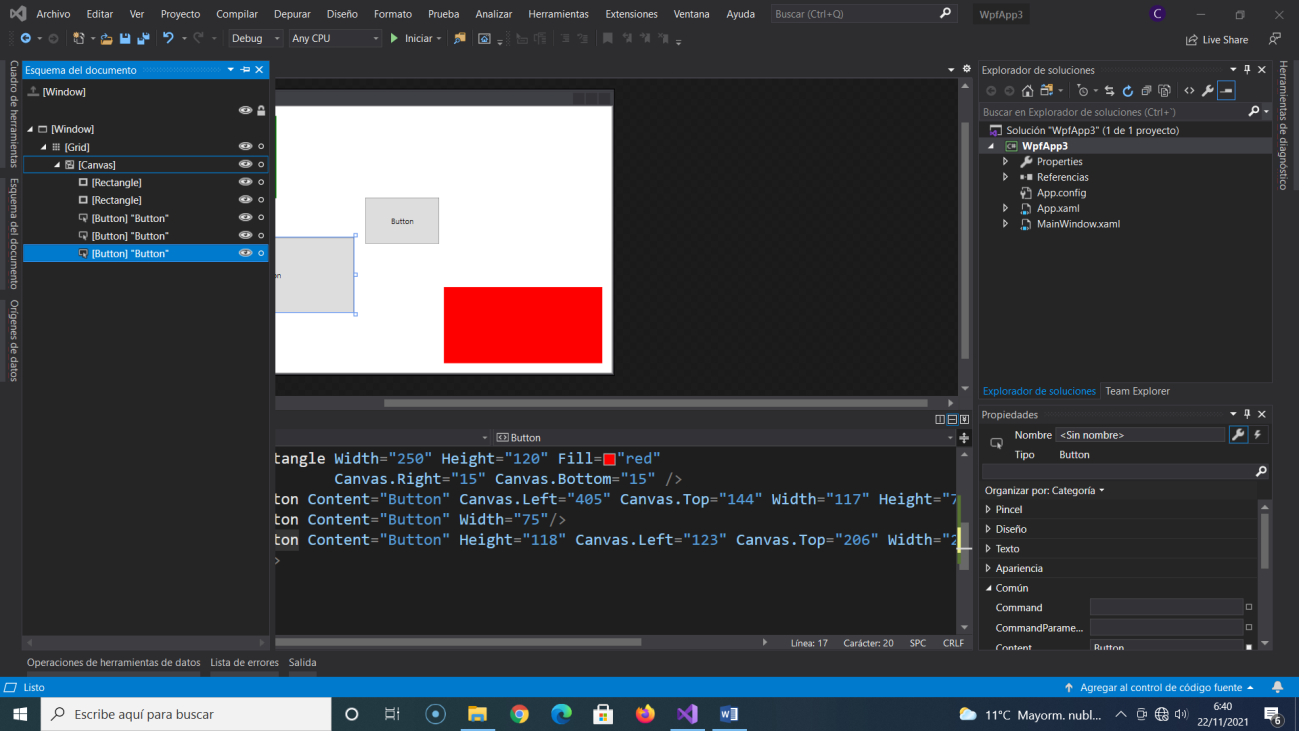


Esta ventana muestra distintos controles que pueden utilizarse en una interfaz WPF. Permite, también, seleccionarlos y situarlos con ayuda del ratón. Este enfoque resulta mucho más rápido que la edición del código fuente XAML.

Es posible utilizar tres soluciones para situar controles en una ventana WPF:

* Haciendo **doble clic** sobre un control en el cuadro de herramientas se sitúa un ejemplar del control en la ventana. El control se ubica, por defecto, en la esquina superior izquierda de la ventana. Sus propiedades de dimensionamiento son las especificadas por defecto.
* El uso de la acción **arrastrar y colocar** sobre un elemento del cuadro de herramientas hacia el diseñador visual permite situar un control en el lugar deseado. Sus dimensiones son las definidas por defecto.
* La **selección de un elemento** en el cuadro de herramientas permite diseñar una zona rectangular cuyo tamaño y posición los define el usuario. El control se situará en esta zona.

La ventana **Esquema del documento** ofrece una visualización rápida del árbol de la ventana en edición. Permite, también, agregar un control en una ubicación precisa del árbol. Puede utilizar la acción arrastrar y colocar sobre un elemento del cuadro de herramientas hacia esta ventana. Cuando se pasa por encima de esta ventana, aparecen algunas indicaciones que le ayudarán a situar el control en la ubicación lógica deseada.



El control creado en la arborescencia se ubica, por defecto, en su control contenedor y sus propiedades de dimensionamiento son las definidas por defecto por el diseñador visual. Este modo de creación del control es particularmente útil cuando la interfaz es compleja.

## 2.2. Posición y dimensionamiento de controles

Los controles se posicionan, por defecto, en las coordenadas 0;0 de su contenedor (cuando éste lo permite) y se corresponden con la esquina superior izquierda del control padre.

Evidentemente, la posición por defecto es en pocas ocasiones la deseada por el usuario. Los controles WPF poseen una gran cantidad de propiedades que permiten modificar este comportamiento, así como su dimensionamiento.

**Height y Width**

Las propiedades Height y Width permiten definir de manera absoluta o relativa la altura y anchura de un control. Pueden contener tres tipos de valores:

1. Un **valor numérico absoluto** en pixeles.
2. Un **valor numérico absoluto** seguido de una **unidad de medida**, px (píxeles), in (pulgadas), cm (centímetros), pt (puntos).
3. El valor **Auto**, que indica que el control debe dimensionarse en base a sus necesidades y dentro de los límites de su control contenedor. Algunos controles utilizarán, por defecto, todo el espacio disponible, mientras que otros adaptarán su tamaño en función de su contenido.

**HorizontalAlignment y VerticalAlignment**

La posición de un control en su control padre puede gestionarse de manera automática en función de una alineación.

Las propiedades **HorizontalAlignment** y **VerticalAlignment** permiten implementar este tipo de posicionamiento proporcionando su valor de enumeración que indica la alineación que se desea utilizar: **Left, Center, Right** o **Stretch** **horizontalmente**, y **Bottom, Center, Top** o **Stretch verticalmente**.

Los tres primeros valores son relativamente explícitos. Permiten alinear el control en el borde izquierdo, en el centro o en el borde derecho de su control padre. El valor Stretch permite imponer al control que utilice todo el espacio disponible horizontal o verticalmente.

**Margin**

La propiedad **Margin** de un control define una zona vacía alrededor de él que permite desplazarlo respecto a su posición normal. Existen cuatro valores asociados a esta propiedad que corresponden a los cuatro bordes del control (**izquierda, arriba, derecha y abajo**).

Cuando se modifican estos valores directamente en el código XAML deben declararse en el orden anterior.

<Button Margin=”0 10 0 0” />

En este caso, el control Button se desplazará 10 píxeles hacia abajo respecto a su posición normal en el contenedor.

Los valores definidos para esta propiedad pueden, como las propiedades Width y Height, especificar una unidad de medida particular. Visual Studio no soporta unidades de medida en la ventana de propiedades, por lo que es obligatorio, definirlas en el editor de código fuente XAML.

<Button Margin=”1cm 2px 3in 4pt” />

La propiedad Margin se utiliza en el diseñador visual de Visual Studio para posicionar un control en un contenedor de tipo Grid.

**Padding**

La propiedad Padding permite definir una zona, situada en el interior de un control, en la que no se mostrará el contenido. Para el control Button, por ejemplo, esta propiedad puede utilizarse para que el contenido no se superponga jamás sobre los bordes del control. Su valor se define de la misma manera que el valor de la propiedad Margin.

**La posición en un Canvas**

Cuando un contenedor es de **tipo Canvas**, el posicionamiento se realiza asignando valor a las propiedades Canvas.Top, Canvas.Left, Canvas.Right y Canvas.Bottom sobre cada control que contiene. Permiten especificar la distancia entre el borde del Canvas y el borde asociado al control. Aceptan valores numéricos, con o sin unidad de medida asociada.

## 2.3. Agregar un controlador de eventos a un control

Los controles visuales pueden producir numerosos **eventos** sobre los que es posible suscribirse desde el código C# asociado a la ventana. Es posible, también, suscribirse a estos eventos desde el código fuente XAML.

Para ello, basta con agregar a la declaración XAML de un control un atributo que tenga el siguiente aspecto:

<nombre del evento>=”>nombre del controlador>”

<Grid>

<Button Height=”50” Width=”120” Content=”Mi primer botón”

Click=”Button\_Click”>

</Button>

</Grid>

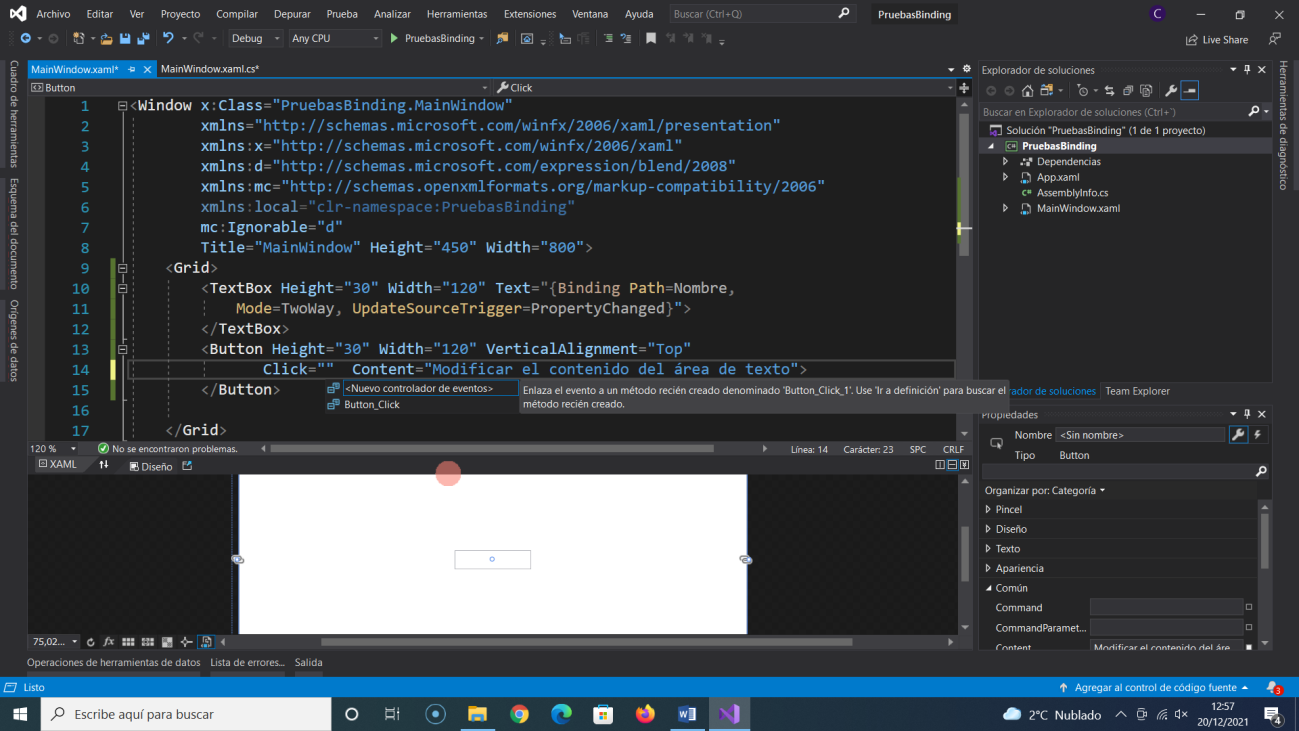
El controlador de eventos correspondiente debe encontrarse en el archivo de code-behind de la ventana en edición. Aquí, la gestión del clic sobre el botón se realiza mediante el método Button\_Click cuya definición es la siguiente:

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

{

}

**IntelliSense** proporciona una gran ayuda para realizar esta operación, (haciendo clic sobre <Nuevo controlador de eventos>). Permite, generar automáticamente un controlador correspondiente al tipo de delegado asociado al evento. Permite, siempre que es posible, asociar el evento a un controlador de eventos existente.



Haciendo doble clic sobre un control se genera también un controlador para el evento por defecto del control. El controlador y el evento se asocian automáticamente en el código XAML.