# Introducción a WPF

## Contenedores

Son elementos que pueden contener y organizar otros elementos visuales, como controles o elementos de interfaz de usuario. Los contenedores son importantes para crear interfaces de usuario bien organizadas y estructuradas.

### 1. StackPanel

Este contenedor es muy utilizado a la hora de realizar listas, ya que organiza sus elementos secuencialmente en filas y/o columnas. Podemos apilar los elementos de dos formas:

- Orientación vertical: los elementos se colocan uno encima del otro.
- Orientación horizontal: los elementos se sitúan uno al lado del otro.

Utilizaremos la propiedad Orientation para posicionar los elementos de manera horizontal o vertical.

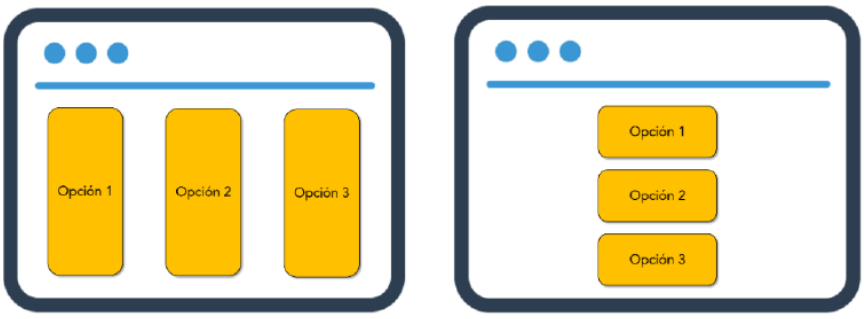


*Ilustración 1. StackPanel*

### 2. Grid (cuadrícula o rejilla)

Este es uno de los contenedores más utilizados en WPF. Permite organizar elementos en filas y columnas, lo que facilita la creación de diseños flexibles y complejos. Tiene un comportamiento muy parecido al objeto tabal de HTML, pudiendo unir columnas, celdas, etc.

Podemos utilizar diferentes propiedades para determinar la alineación vertical y horizontal de cada celda, además de muchas otras opciones.

Se pueden utilizar las propiedades RowDefinitions o ColumnDefinitions

Al utilizar RowDefinitions podemos utilizar valores absolutos en la propiedad Height=”100” o valores relativos como Height=”2*” o Height=”20*” y poner otro con Height=”80*”.

**Recomendación:** intentad utilizar valores que hagan un total de 10 o 100 para controlar mejor la interfaz. Nos resultará más fácil de esa manera dividir el *grid*.

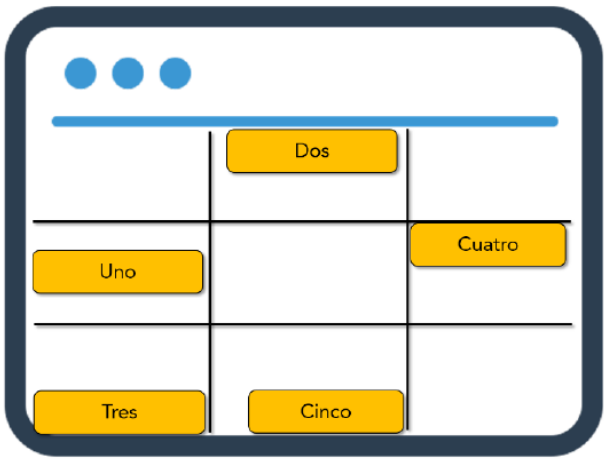


*Ilustración 2. Grid*

### 3. DockPanel

Este contenedor organiza sus elementos en regiones que pueden ser ancladas en la parte superior, inferior, izquierda, derecha o en el centro.

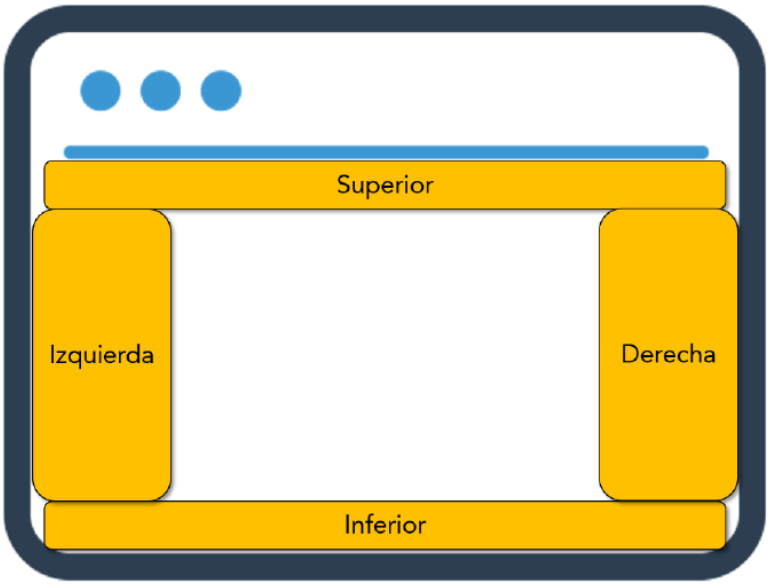


*Ilustración 3. DockPanel*

Hay una opción que nos permite determinar que un elemento ocupe todo el contenido, como por ejemplo para utilizarlo en el explorador de archivos y, dejar en la parte izquierda, opciones de menú.

Para conseguir este comportamiento del DockPanel utilizaríamos la propiedad LastChildFill.

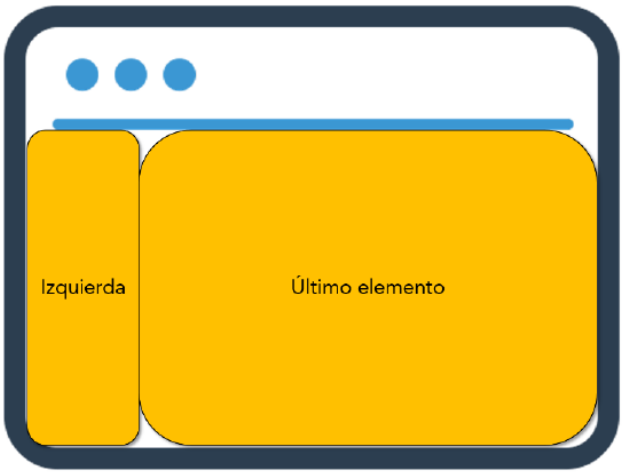


*Ilustración 4. Utilización de LastChildFill*

### 4. Canvas (lienzo)

Permite colocar elementos en coordenadas XY específicas. Es útil cuando se necesita un control total sobre la posición de los elementos. Permite agrupar elementos en diferentes coordenadas, aunque también podemos utilizar coordenadas relativas, haciendo uso de las referencias Canvas.Left, Canvas.Right, entre otras.

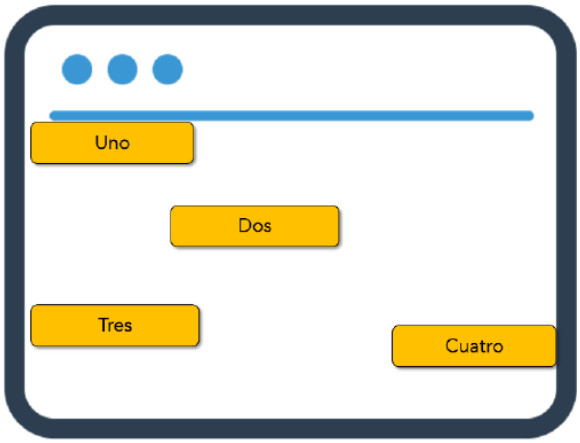


*Ilustración 5. Canvas*

### 5. WrapPanel

Similar al StackPanel, pero en lugar de seguir una sola dirección, coloca los elementos secundarios en posición secuencial de izquierda a derecha y traslada el contenido ala siguiente línea. Estos comportamientos pueden alterarse, de manera que podemos empezar de derecha a izquierda, por ejemplo.

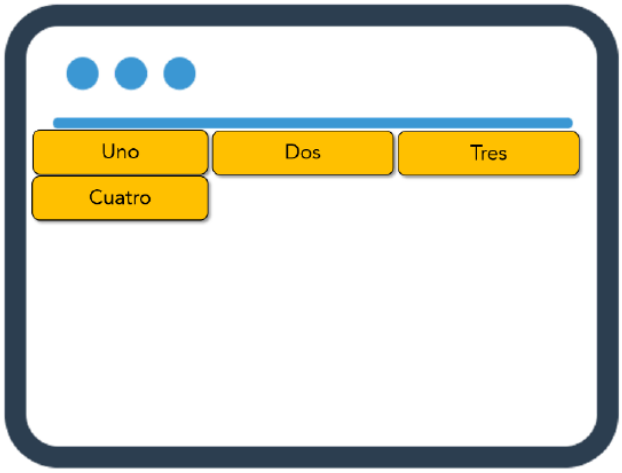


*Ilustración 6. WrapPanel*

## 6. UniformGrid

*UniformGrid* es un tipo de panel que organiza sus elementos secundarios en una cuadrícula uniforme, es decir, todas las celdas de la cuadrícula tienen el mismo tamaño. Sus características son:

- ✓ Todas las celdas tienen el mismo tamaño, proporcionando una distribución uniforme de los elementos dentro de este contendor.
- ✓ Se puede especificar el número de filas o columnas en el *UniformGrid* según nuestras necesidades.

*Ejemplo de utilización*

```
<UniformGrid Rows="3" Columns="3" Background="LightGray">
    <Button Content="1"/>
    <Button Content="2"/>
    <Button Content="3"/>
    <Button Content="4"/>
    <Button Content="5"/>
    <Button Content="6"/>
    <Button Content="7"/>
    <Button Content="8"/>
    <Button Content="9"/>
</UniformGrid>
```

*Ilustración 7. Ejemplo de creación UniformGrid en XAML*

También podemos crear los contenedores en código C#. Por ejemplo, para crear un contenedor de tipo *UniformGrid*, podemos hacerlo de la siguiente forma:

```
UniformGrid uniformGrid = new UniformGrid();
uniformGrid.Rows = 3;
uniformGrid.Columns = 3;

for (int i = 1; i <= 9; i++)
{
    Button button = new Button();
    button.Content = i.ToString();
    uniformGrid.Children.Add(button);
}
```

*Ilustración 8. Ejemplo creación UniformGrid en C#*

En la ilustración anterior vemos como crear un contenedor de tipo *UniformGrid* en código C#. Establecemos las filas y las columnas y después, mediante la utilización de un bucle, creamos botones asignándole un número como contenido y agregando cada botón al contendor mediante el método *Children.Add(button);*

### Consejos generales sobre contenedores

En general, todos los contenedores proporcionados por XAML deben seguir una serie de especificaciones:

- **No utilizar posiciones fijas** a través de coordenadas absolutas. Es preferible utilizar atributos como Alignment y Margin para asegurar un correcto funcionamiento.
- **No asignar tamaños** fijos a los elementos, es preferible hacer uso de la propiedad auto.
- **No abusar** de la utilización de Canvas, utilizarlo únicamente cuando sea necesario.
- **Utilizar** el contenedor StackPanel en los botones de diálogo.
- **Utilizar** el contenedor GridPanel a la hora de definir una interfaz de entrada de datos estática.

## Cuadros de diálogo

Los cuadros de diálogo definen la relación que existe entre la interfaz y el usuario. Son ventanas emergentes que se utilizan para interactuar con el usuario y solicitar información o confirmación en una aplicación de software. Estos cuadros de diálogo pueden variar en función de su propósito, por ejemplo, pueden ser utilizados para mostrar mensajes de advertencia, solicitar entrada de datos, confirmar una acción o presentar opciones al usuario.

Existen dos tipos principales de cuadros de diálogo: modales y no modales.

1) **Cuadro de diálogo modal**

Un cuadro de diálogo modal bloquea la interacción con la ventana principal de la aplicación hasta que el usuario lo cierra. Esto significa que el usuario no puede interactuar con la ventana

principal ni con ningún otro elemento de la aplicación hasta que haya respondido al cuadro de diálogo.

Los cuadros de diálogo modales se utilizan para interacciones críticas que requieren la atención inmediata del usuario o para acciones que deben completarse antes de continuar con la aplicación.

> **Ejemplo:** Un cuadro de diálogo que solicita al usuario guardar un archivo antes de cerrar una aplicación.

2) **Cuadro de diálogo no modal**

A diferencia de los modales, un cuadro de diálogo no modal no bloquea la interacción con la ventana principal de la aplicación. Esto significa que el usuario puede seguir interactuando con la ventana principal y con otros elementos de la aplicación mientras el cuadro de diálogo está abierto.

Los cuadros de diálogo no modales se utilizan para proporcionar información adicional o permitir al usuario realizar acciones secundarias sin interrumpir el flujo principal de trabajo.

> **Ejemplo:** Un cuadro de diálogo que proporciona información sobre el estado actual de la aplicación sin requerir una acción inmediata.

En resumen, la diferencia clave entre los cuadros de diálogo modales y no modales radica en cómo afectan la interacción con la ventana principal de la aplicación. Los modales bloquean la interacción hasta que se resuelve el cuadro de diálogo, mientras que los no modales permiten al usuario seguir interactuando con la ventana principal. Ambos tipos tienen sus usos específicos en el diseño de interfaces de usuario.