Tema 1.4.7. Diseños dinámicos con XAML

El sistema de diseño XAML permite usar el ajuste de tamaño automático de los elementos, paneles de diseño y estados visuales como ayuda para crear una interfaz de usuario adaptativa. Con un diseño adaptativo, puedes hacer que la aplicación tenga un gran aspecto en pantallas con diferentes tamaños de ventana de aplicación, resoluciones, densidades de píxeles y orientaciones. También puedes usar XAML para cambiar la posición, ajustar el tamaño, redistribuir, mostrar u ocultar, reemplazar y rediseñar la interfaz de usuario de tu aplicación, como se explica en Técnicas de diseño con capacidad de respuesta. Aquí describimos cómo implementar diseños adaptativos con XAML.

Índice

1. Diseños fluidos con propiedades y paneles

La base de un diseño adaptativo es hacer un uso adecuado de las propiedades y los paneles de diseño de XAML para cambiar la posición, ajustar el tamaño y redistribuir el contenido de forma fluida.
El sistema de diseño de XAML admite diseños tanto estáticos como fluidos. En un diseño estático, se definen en los controles tamaños de píxeles y posiciones explícitas. Cuando el usuario cambia la resolución u orientación de su dispositivo, la interfaz de usuario no cambia. Los diseños estáticos pueden recortarse en distintos factores de forma y tamaños de pantalla. Por otro lado, los diseños fluidos se reducen, crecen y vuelven a fluir para responder al espacio visual disponible en un dispositivo.

En la práctica, usa una combinación de elementos estáticos y fluidos para crear la interfaz de usuario. Seguirás usando elementos y valores estáticos en algunas partes, pero debes asegurarte de que el conjunto de la interfaz de usuario se adapte a las distintas resoluciones, tamaños de pantalla y vistas.
Aquí describimos cómo usar las propiedades de XAML y los paneles de diseño para crear un diseño fluido.

1.1. Propiedades de diseño

Las propiedades de diseño controlan el tamaño y la posición de un elemento. Para crear un diseño fluido, usa variaciones de tamaño automáticas o proporcionales para los elementos, y permite que los paneles de diseño establezcan la posición de sus elementos secundarios según sea necesario.
Aquí se muestran algunas propiedades de diseño habituales y cómo usarlas para crear diseños fluidos

1.1.1 Alto y ancho

Las propiedades **Height** y **Width** especifican el tamaño de un elemento. Puedes usar valores fijos medidos en píxeles efectivos o puedes usar la variación de tamaño automática o proporcional.
La variación de tamaño automática cambia el tamaño de los elementos de la interfaz de usuario de modo que se ajusten a su contenido o contenedor primario. La variación de tamaño automática también se puede usar con las filas y columnas de una cuadrícula. Para usar la variación de tamaño automática, establece las propiedades Height o Width de los elementos de interfaz de usuario en **Auto**.

Nota:
El hecho de que el tamaño de un elemento cambie para ajustarse a su contenido o contenedor depende de cómo el contenedor primario controla la variación de tamaño de sus elementos secundarios. Para más información, consulta **Paneles de diseños** más adelante en este artículo.

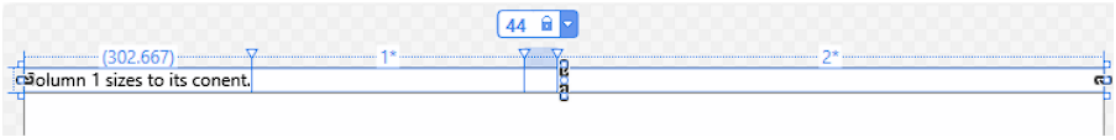
La *variación de tamaño proporcional* distribuye el espacio disponible entre las filas y columnas de una cuadrícula en proporciones ponderadas. En XAML, los valores de star se expresan como * (o *n** para el tamaño ponderado star). Por ejemplo, para especificar que una columna es 5 veces más ancha que la segunda columna en un diseño de 2 columnas, use "5*" y "*" para las propiedades **Width** de los elementos **ColumnDefinition** .
En este ejemplo, se combinan variaciones de tamaño fijas, automáticas y proporcionales en una clase **Grid** con 4 columnas.

Columna	Ajuste de tamaño	Description
Columna_1	**Auto**	La columna se ajustará a su contenido.
Columna_2	*	Después de que se calculen las columnas Auto, la columna recibe parte del ancho restante. Columna_2 será la mitad de ancha que Columna_4.
Columna_3	**44**	La columna tendrá un ancho de 44 píxeles.
Columna_4	2*	Después de que se calculen las columnas Auto, la columna recibe parte del ancho restante. Columna_4 será el doble de ancha que Columna_2.

El ancho de columna predeterminado es "*", de modo que no es necesario establecer explícitamente el valor de la segunda columna.

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition/>
        <ColumnDefinition Width="44"/>
        <ColumnDefinition Width="2*"/>
    </Grid.ColumnDefinitions>
    <TextBlock Text="Column 1 sizes to its content." FontSize="24"/>
</Grid>
```

En el Diseñador XAML de Visual Studio, el resultado tiene este aspecto.



1.1.2 Restricciones de tamaño

Cuando usas la variación de tamaño automática en la interfaz de usuario, es posible que aún tengas que establecer restricciones en el tamaño de un elemento. Puedes establecer las propiedades MinWidth/MaxWidth y MinHeight/MaxHeight para especificar valores que limitan el tamaño de un elemento al tiempo que permiten un cambio de tamaño fluido.
En una Grid, MinWidth/MaxWidth también se puede usar con las definiciones de columna y MinHeight/MaxHeight se puede usar con las definiciones de fila.

1.1.3 Alineación

Usa las propiedades **HorizontalAlignment** y **VerticalAlignment** para especificar cómo un elemento se debe colocar en su contenedor primario.

- Los valores de **HorizontalAlignment** son **Left**, **Center**, **Right** y **Stretch**.
- Los valores de **VerticalAlignment** son **Top**, **Center**, **Bottom** y **Stretch**.

Mediante la alineación **Stretch**, puedes conseguir que los elementos llenen todo el espacio que se les proporciona en el contenedor primario. Stretch es el valor predeterminado para ambas propiedades de alineación. Sin embargo, algunos controles, como **Button**, invalidan este valor en su estilo predeterminado. Cualquier elemento que pueda tener elementos secundarios puede tratar el valor Stretch de las propiedades HorizontalAlignment y VerticalAlignment de manera exclusiva. Por ejemplo, un elemento que usa el valor predeterminado Stretch colocado en una Grid se ajusta para rellenar la celda que lo contiene. El mismo elemento se coloca en tamaños de Canvas según su contenido. Para obtener más información sobre cómo cada panel controla el valor Stretch, consulta el artículo paneles de diseño.
Para más información, consulta el artículo Alineación, margen y espaciado y las páginas de referencia **HorizontalAlignment** y **VerticalAlignment**.

1.1.4 Visibilidad

Puedes mostrar u ocultar un elemento si estableces su propiedad **Visibility** en uno de los valores de enumeración **Visibility**: **Visible**, **Collapsed** y **Hidden.** Cuando un elemento tiene el estado Collapsed, no ocupa espacio en el diseño de la interfaz de usuario.
Puedes cambiar la propiedad Visibility de un elemento en código o en un estado visual. Cuando el objeto Visibility de un elemento cambia, también cambian todos sus elementos secundarios. Puedes reemplazar secciones de la interfaz de usuario al mostrar un panel mientras contraes otro.
Si no especificas nada de forma predeterminada, tomará "Visible". Como puede ver en la siguiente imagen, los 3 botones son visibles y 2 de ellos ni siquiera tienen una Visibilidad configurada.

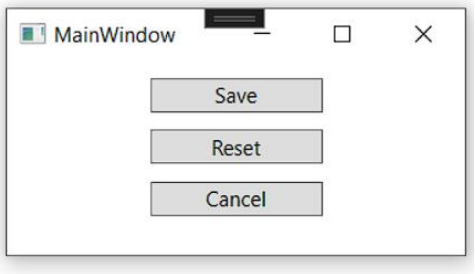


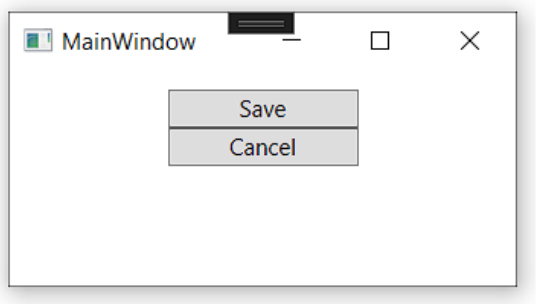
Figure 1: Visibility="Visible"

El primer y tercer botón ni siquiera tienen Visibilidad configurada, aun así, son visibles, pero para el segundo botón estamos especificando explícitamente que el valor de Visibilidad sea "Visible", al hacerlo estamos anulando su valor con su valor predeterminado. Es como asignar 0 al valor int.

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <Button x:Name="ButtonSave"
            Margin="0 10 0 0"
            Height="20"
            Width="100"
            Content="Save"/>
    <Button x:Name="ButtonReset"
            Margin="10"
            Visibility="Visible"
            Height="20"
            Width="100"
            Content="Reset"
            Grid.Row="1"/>
    <Button x:Name="ButtonCancel"
            Height="20"
            Width="100"
            Content="Cancel"
            Grid.Row="2"/>
</Grid>
```

Ahora cambiemos la Visibilidad a "Collapsed" para el segundo botón:

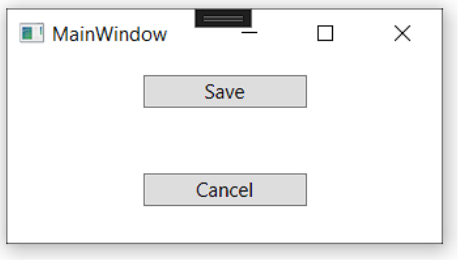
```
<Button x:Name="ButtonSave"
        Margin="0 10 0 0"
        Height="20"
        Width="100"
       Content="Save"/>
<Button x:Name="ButtonReset"
        Margin="10"
        Visibility="Collapsed"
        Height="20"
        Width="100"
        Content="Reset"
       Grid.Row="1"/>
<Button x:Name="ButtonCancel"
        Height="20"
        Width="100"
        Content="Cancel"
        Grid.Row="2"/>
```



Visibility="Collapsed"

El botón de “Reset” desaparece y el botón de “Cancel” se agrupa con el de “”. Ahora si cambio el valor a "Hidden”:

```
<Button x:Name="ButtonReset"
        Margin="10"
        Visibility="Hidden"
        Height="20"
        Width="100"
        Content="Reset"
        Grid.Row="1"/>
```



Visibility="Hidden"

El botón “Reset” desaparece, pero dejamos el lugar vacío.

Si tienes elementos en la interfaz de usuario con el estado **Collapsed** de manera predeterminada, los objetos se seguirán creando en el inicio, aunque no sean visibles. Puede aplazar la carga de estos elementos hasta que se muestren mediante el **atributo x:Load** para retrasar la creación de los objetos. Esto puede mejorar el rendimiento del inicio. Para obtener más información, consulte **atributo x:Load**.

2. Recursos de estilo

No hace falta establecer cada valor individualmente en un control. Suele ser más eficiente agrupar valores de propiedad en un recurso **Style** y aplicar el Style a un control. Esto es especialmente cierto cuando necesites aplicar los mismos valores de propiedad a muchos controles. Para obtener más información sobre los estilos, consulta Controles de estilo.

3. Paneles de diseño

Para colocar objetos visuales, debes ponerlos en un panel u otro objeto contenedor. El marco de XAML proporciona diversas clases Panel como **Canvas**, **Grid**, **RelativePanel** y **StackPanel**, que sirven como contenedores y permiten colocar y organizar los elementos de la interfaz de usuario dentro de ellas.
La principal consideración al elegir un panel de diseño es cuál será la posición del panel y el tamaño de sus elementos secundarios. Es posible que también debas considerar cómo se colocarán los elementos secundarios cuando se superpongan.
Esta es una comparación de las principales funciones de los controles de panel que se proporcionan en el marco XAML.

Control de panel	Descripción
Canvas	**Canvas** no admite la interfaz de usuario fluida; se controlan todos los aspectos de posicionamiento y ajuste de tamaño de los elementos secundarios. Normalmente se usa para casos especiales, como la creación de gráficos o la definición de áreas estáticas pequeñas de una interfaz de usuario adaptativa de mayor tamaño. Puedes usar código o estados visuales para cambiar la posición de los elementos en tiempo de ejecución. • Los elementos se colocan de forma absoluta mediante las propiedades adjuntas Canvas.Top y Canvas.Left. • La disposición se puede especificar explícitamente mediante la propiedad adjunta Canvas.ZIndex. • Los valores Stretch de HorizontalAlignment/VerticalAlignment se omiten. Si el tamaño de un elemento no se establece explícitamente, su tamaño se ajusta a su contenido. • El contenido secundario no se recorta visualmente si su tamaño es mayor que el panel. • El contenido secundario no se ve restringido por los límites del panel.
Grid	**Grid** admite la variación de tamaño fluida de elementos secundarios. Puedes usar código o estados visuales para cambiar la posición y redistribuir elementos. • Los elementos se organizan en filas y columnas mediante las propiedades adjuntas Grid.Row y Grid.Column. • Los elementos pueden abarcar varias filas y columnas mediante las propiedades adjuntas Grid.RowSpan y Grid.ColumnSpan. • Los valores Stretch de HorizontalAlignment/VerticalAlignment se respetan. Si el tamaño de un elemento no se establece explícitamente, se amplía para rellenar el espacio disponible en la celda de la cuadrícula. • El contenido secundario se recorta visualmente si su tamaño es mayor que el panel. • El tamaño del contenido se ve restringido por los límites del panel, por lo que el contenido desplazable muestra barras de desplazamiento si es necesario.
RelativePanel	• Los elementos se organizan en relación con el borde o el centro del panel y en relación entre ellos. • Los elementos se colocan mediante una variedad de propiedades adjuntas que controlan la alineación del panel, la alineación de elementos de mismo nivel y la posición de los elemento del mismo nivel. • Los valores Stretch de HorizontalAlignment/VerticalAlignment se omiten, a menos que las propiedades adjuntas RelativePanel de la alineación provoquen una ampliación (por ejemplo, un elemento se alinea a los bordes derecho e izquierdo del panel). Si el tamaño de un elemento no se establece explícitamente y no se amplía, su tamaño se ajusta a su contenido.

Control de panel	Descripción
	• El contenido secundario se recorta visualmente si su tamaño es mayor que el panel. • El tamaño del contenido se ve restringido por los límites del panel, por lo que el contenido desplazable muestra barras de desplazamiento si es necesario.
StackPanel	• Los elementos se apilan en una única línea vertical u horizontal. • Los valores Stretch de HorizontalAlignment/VerticalAlignment se respetan en la dirección opuesta a la propiedad Orientation. Si el tamaño de un elemento no se establece explícitamente, se amplía para rellenar el ancho disponible (o el alto si la Orientation es Horizontal). En la dirección especificada por la propiedad Orientation, el tamaño de un elemento se ajusta a su contenido. • El contenido secundario se recorta visualmente si su tamaño es mayor que el panel. • El tamaño del contenido no está restringido por los límites del panel en la dirección especificada por la propiedad Orientation, por lo que el contenido desplazable se amplía más allá de los límites del panel y no se muestran barras de desplazamiento. Tienes que restringir explícitamente el alto (o el ancho) del contenido secundario para mostrar sus barras de desplazamiento.
VariableSizedWrapGrid	• Los elementos se organizan en filas o en columnas que se ajustan automáticamente a una nueva fila o columna cuando se alcanza el valor MaximumRowsOrColumns. • La organización de los elementos en filas o en columnas se especifica mediante la propiedad Orientation. • Los elementos pueden abarcar varias filas y columnas mediante las propiedades adjuntas VariableSizedWrapGrid.RowSpan y VariableSizedWrapGrid.ColumnSpan. • Se omiten los valores extendidos de HorizontalAlignment y VerticalAlignment. El tamaño de los elementos se ajusta según se especifica en las propiedades ItemHeight y ItemWidth. Si no se establecen estas propiedades, toman sus valores del tamaño de la primera celda. • El contenido secundario se recorta visualmente si su tamaño es mayor que el panel. • El tamaño del contenido se ve restringido por los límites del panel, por lo que el contenido desplazable muestra barras de desplazamiento si es necesario.
WrapPanel	• Similar al anterior, colocar〈 cada uno de sus controles hijos uno junto al otro, horizontalmente (predeterminado) o verticalmente, hasta que no haya mas espacio. En este ultimo caso, se ajustar〈 a la siguiente linea y continuara .