UNIDAD 8:

LAYOUTS

Layouts son importantes

- Con WPF se introdujo un nuevo sistema de layouts para hacer más fluidas las interfaces de usuario: los layouts que no sean estáticos pueden adaptar su tamaño al espacio disponible.
- UWP, Xamarin y MAUI siguen usando este mismo sistema.
- Las aplicaciones ya no dependen de la resolución.
- Las interfaces se escalan para ajustarse a diferentes tamaños de pantalla.

Layouts son importantes

- Problema con Windows Forms.
- Esto no ocurre usando correctamente los Layouts.

1920x1080

1024x768

Tipos de Layouts

En MAUI tenemos los siguientes tipos de layouts:

- StackLayout
- HorizontalStackLayout
- VerticalStackLayout
- Grid
- FlexLayout
- AbsoluteLayout
- BindableLayout

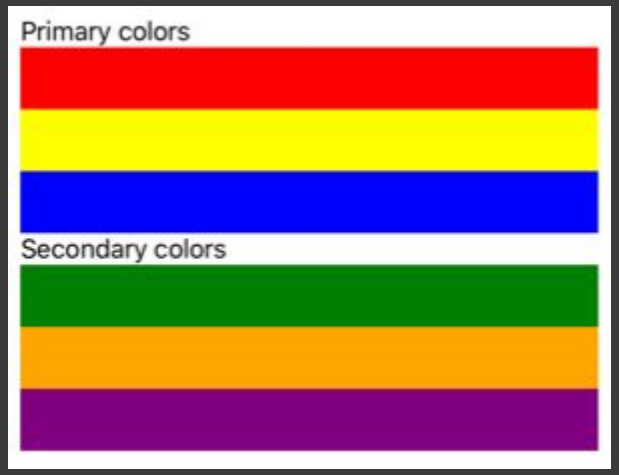
Buenas prácticas

- Evitar posiciones fijas: usar alineaciones y márgenes para posicionar elementos.
- Evitar tamaños fijos: darle a las propiedades Width y Height un valor “Auto” siempre que sea posible.
- Limitar el tamaño de los controles dándoles un tamaño máximo y/o mínimo.
- Los layouts contenedores pueden y deben estar anidados.
- No abusar del Grid. No intentar hacer toda la interfaz de usuario con un Grid.

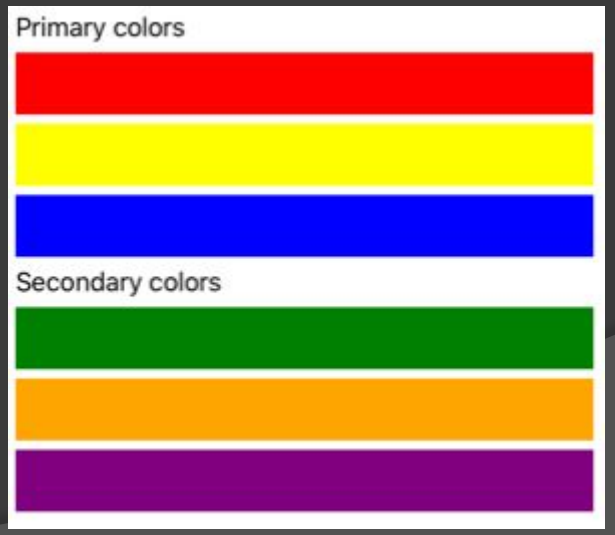
StackLayout

- Organiza los elementos en una única línea que se puede orientar horizontal o verticalmente.
- Todos los ItemControls como ComboBox, ListBox o MenuFlyout usan internamente un StackLayout.
- Los elementos se expanden hasta rellenar el espacio.
- La propiedad “Spacing” permite definir un espacio entre los elementos de la pila.

Sin Spacing

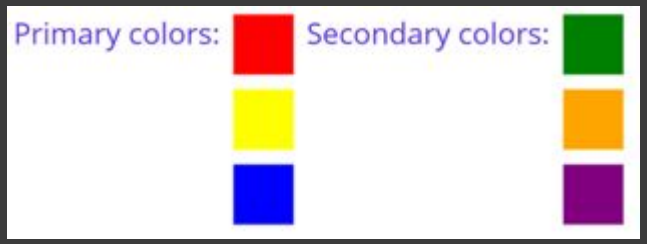


Spacing=6



HorizontalStackLayout y VerticalStackLayOut

- Son parecidos al StackLayout pero son específicos para colocarlos vertical u horizontalmente.
- Son un poco más eficientes que el StackLayout, y los usaremos cuando estemos seguros de que la posición del Stack no va a variar.
- Podemos mezclarlos:



Esto es un HorizontalStack como padre con dos VerticalStack como hijos.

Grid

- Es uno de los layouts más potentes por su versatilidad.
- Forma una tabla con filas y columnas.
- En una celda puede haber muchos elementos, pero lo normal es colocar sólo uno. Sin embargo, ese elemento puede ser otro contenedor que contenga otros elementos (por ejemplo un stacklayout en una celda).
- Anidar demasiados elementos puede afectar al rendimiento.
- Podemos usar **Grid.RowSpan** y **Grid.ColumnSpan** para combinar filas o columnas.

Grid: filas y columnas

- Un Grid tiene una fila y una columna por defecto. Para crear más filas tenemos que usar Grid.RowDefinitions y Grid.ColumnDefinitions.
- El tamaño se puede definir de las siguientes formas:
  - **Por defecto:** si no especificamos nada, se dividirá la anchura o altura de forma equitativa.
  - **Fija:** podemos fijar un tamaño usando Width="50"
  - **Auto:** tomará el espacio que se necesite dependiendo del contenido.
  - **Star (*):** tomará el espacio restante si sólo es una fila o columna, si son varias hará lo siguiente: por ejemplo, una fila con Width="*" y otra con Width="5*" indica que la primera tomará 1/6 y la segunda 5/6 de la anchura restante.

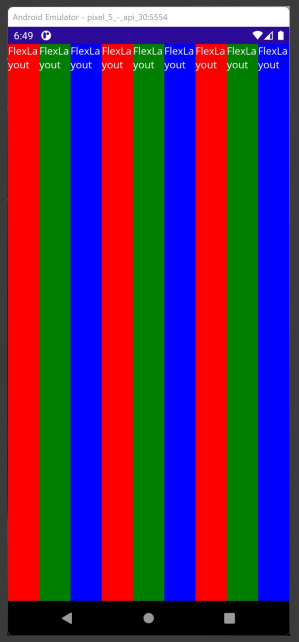
Grid: Rows and Columns

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="*”></RowDefinition>
        <RowDefinition Height="28”></RowDefinition>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto”></ColumnDefinition>
        <ColumnDefinition Width="200”></ColumnDefinition>
    </Grid.ColumnDefinitions>
</Grid>
```

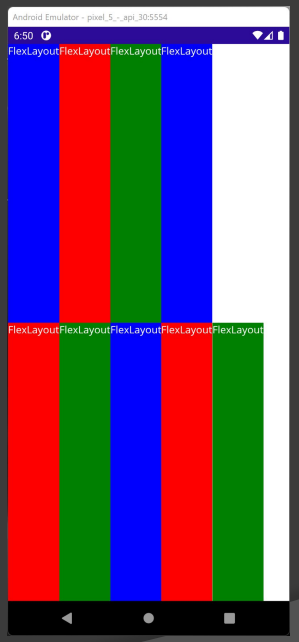
FlexLayout

- Es parecido al StackLayout pero los elementos se ajustan automáticamente a una nueva fila o columna cuando no les queda espacio.
- Para que los elementos se ajusten, se usa la propiedad Wrap.

<FlexLayout Wrap="NoWrap">



<FlexLayout Wrap="Wrap">



AbsoluteLayout

- Permite asignar posición y tamaño a sus elementos de manera explícita.
- La posición se determina partiendo de la esquinas superiores izquierdas del elemento hijo y del AbsoluteLayout.
- Tiene dos propiedades muy importantes:
  - LayoutBounds: representa la posición y el tamaño. Por ejemplo: AbsoluteLayout.LayoutBounds="0, 10, 200, 5" indica x=0, y =10, anchura= 200 y altura=5
  - LayoutFlags: permite indicar que los valores del LayoutBound son proporcionales al tamaño del AbsoluteLayot. Si su valor es None, se tomarán valores absolutos.

BindableLayout

- Permite que cualquier tipo de Layout pueda ser enlazado (Binding) a una colección de elementos.

```
<StackLayout BindableLayout.ItemsSource="{Binding User.TopFollowers}"
             Orientation="Horizontal"
             ...>
    <BindableLayout.ItemTemplate>
        <DataTemplate>
            <Image Source="{Binding}"
                   Aspect="AspectFill"
                   WidthRequest="44"
                   HeightRequest="44"
                   ... />
        </DataTemplate>
    </BindableLayout.ItemTemplate>
</StackLayout>
```

- No es aconsejable para el Grid ni para el AbsoluteLayout