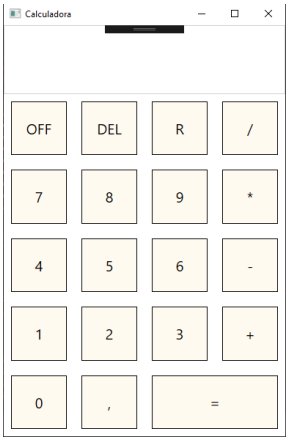
Projeto Semanal 1 – Calculadora



Desenvolvido em C# - WPF | Nuno Silva | 30171 | RSI-A 2º Ano

- ## Design da Aplicação

```
<Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
```

Primeiramente, comecei por desenhar os botões e o layout da calculadora.

Dentro do <grid>, através do Grid.ColumnDefinitions, comecei por por indicar o número de colunas que necessitava, ou seja, 4. Indiquei ainda que a largura seria "*" para esta se dimensionar a ela própria, ou seja, ser "responsive".

De seguida, defini as linhas através do Grid.RowDefinitions, sendo 6 linhas e obtendo uma altura (Height) igual a "*" de modo a ter também uma dimensão responsiva.

```
<Grid.Resources>
            <Style TargetType="{x:Type Button}">
                <Setter Property="Margin" Value="10"></Setter>
                <Setter Property="FontSize" Value="20"></Setter>
                <Setter Property="BorderBrush" Value="Black"></Setter>
                <Setter Property="Background" Value="FloralWhite"></Setter>
            </Style>
        </Grid.Resources>
```

De seguida, defini algumas propriedades dos botões, entre eles a margem (margin), o tamanho da letra (fontsize), a cor do contorno (BorderBrush) e a sua cor (Background).

No que diz respeito ao ecrã da calculadora, onde irão aparecer os valores, escolhi uma textbox, nomeada "Tb_Visor", indicando ainda que a mesma ocupa 4 colunas, para ocupar a totalidade da calculadora através do Grid.ColumnSpan.

Posteriormente adicionei todos os botões através do <Button>, com algumas propriedades importantes:

Grid.Column: A coluna onde estará localizado | Grid.Row – A linha onde estará localizado | Content – Texto dentro do botão

Para os botões executarem determinadas ações, atribuí a todos eles a propriedade Click="C_Click" que faz com que, quando clicados exerçam uma determinada função. Para os restantes botões procedi de igual forma.

```
1.  <TextBox x:Name="Tb_Visor" IsEnabled="False" Grid.ColumnSpan="4" FontSize="50"/>
2.
3.          <Button Grid.Column="0" Grid.Row="1" Content="OFF" Click="C_Click"/>
```

- ## Desenvolvimento da Aplicação

De seguida, criei 6 variáveis.

```
1.  double visor;
2.  bool soma = false;
3.  bool sub = false;
4.  bool mult = false;
5.  bool div = false;
6.  bool igual = false;
```

A variável visor, do tipo double, serve para armazenar os valores das operações e posteriormente, mostrar os mesmos.

Os booleans servem apenas para indicar se algum dos operadores foi clicado, de modo a quando clicar no "igual", este possa saber que tipo de operação é que vai realizar.

De seguida, para perceber em que botão é que o utilizador carregou, criei um switch que verifica o conteúdo do botão e posteriormente realiza uma ação:

```
1.  Button b = (Button)sender;
2.              switch (b.Content)
3.              {
4.              }
```

Tal como os restantes botões, a estrutura dentro do swtich statement processa-se da seguinte forma:

```
1.  case "OFF":
2.                      MessageBoxResult resposta = MessageBox.Show("A aplicação vai encerr
    ar!", "Confirmar Ação", MessageBoxButton.OKCancel);
3.                      if(resposta == MessageBoxResult.OK)
4.                      {
5.                          System.Windows.Application.Current.Shutdown();
6.                      }
7.                      break;
```

Existe um Case que verifica o conteúdo do botão. Neste caso, apresento o botão OFF – Serve para desligar a calculadora.

Envia uma mensagem de confirmação para o ecrã, de modo a garantir que o utilizador pretende mesmo fechar a calculadora. Aquando da resposta, ele lê a mesma e dentro do IF, verifica qual foi e realiza uma ação. Se a resposta for "OK" ele fecha a calculadora.

```
case "=":
                    if(soma == true)
                    {
                        visor += double.Parse(Tb_Visor.Text);
                        Tb_Visor.Text = Convert.ToString(visor);
                        soma = false;
                        igual = true;
                    }
```

Neste exemplo, demonstro o que acontece quando se carrega no botão IGUAL. Ele verifica que tipo de operação é realizada e, de acordo com a operação, altera o valor no visor da calculadora. De seguida, indica que a operação é igual a false, uma vez que já foi realizada e indica também que o botão igual foi pressionado.

Todos os botões contêm um mecanismo de verificação: Se o utilizador já tiver carregado no igual, ele reinicia a variável igual e o valor do "visor" de modo a permitir a realização de uma nova aplicação. Caso o igual não tenha sido pressionado, ele limita-se a acrescentar o valor pressionado ao valor que consta no ecrã.

```
case "1":
                    if (igual == true)
                    {
                        Tb_Visor.Text = "1";
                        visor = 0;
                        igual = false;
                    }
                    else
                    {
                        Tb_Visor.Text += "1";
                    }

                    break;
```

O botão "R" (Reset), reinicia a calculadora ou seja, define o valor da variável "visor" para 0, apaga o texto e de seguida, reinicia também as variáveis existentes.

```
case "R":
                    Tb_Visor.Text = "";
                    visor = 0;
                    soma = false;
                    sub = false;
                    mult = false;
                    div = false;
                    igual = false;
                    break;
```

Para não permitir operações com apenas uma vírgula no ecrã, ou seja, estas causariam erro, desenvolvi este excerto de código:

```
case ",":
                    if (igual == true)
                    {
                        MessageBox.Show("Escolha um número primeiro!");
                    }
                    else
                    {
                        Tb_Visor.Text += ",";
                    }
                    break;
```

Ou seja, se o utilizador já tiver carregado no igual e este pretenda inserir uma vírgula, este mecanismo de verificação no o vai permitir. Caso o igual não tenha sido pressionado, pode de facto inserir a vírgula que é incrementada ao valor que está no visor.

```
case "+":

                    if (Tb_Visor.Text != "," && Tb_Visor.Text != ",," && Tb_Visor.Text != ",,," && Tb_Visor.Text != ",,,,")
                    {
                        if (Tb_Visor.Text.Length != 0)
                        {
                            visor = double.Parse(Tb_Visor.Text);
                            Tb_Visor.Text = "";
                            soma = true;
                            igual = false;
                        }
                    }
                    else
                    {
                        MessageBox.Show("Não é possível efetuar operações com símbolos!");
                        Tb_Visor.Text = "";
                    }
                    break;
```

Neste pedaço de código é possível verificar o que acontece quando uma tecla de operação é pressionada.

Primeiro, o programa confirma que o utilizador não vai realizar uma soma de duas vírgulas ou mais (causaria erro), de seguida, verifica se realmente se encontra um número escrito para realizar a operação. Caso seja positivo, o programa realiza a operação e posteriormente indica à variável da respetiva operação, que a mesma se encontra realizada.

Isso acontece atribuindo o valor "TRUE" à variável boolean da mesma.

Caso se verifique que existe uma operação com vírgulas, o programa indica ao utilizador que não são possíveis operações com vírgula.

No código é possível verificar as restantes operações (sutrair, dividir, multiplicar) e verificar que o funcionamento é semelhante à soma, alterando apenas o operador e as suas operações.