

Aula 2 - Controlos e Layouts

Resumo:

- *Layouts* em WPF
- Utilização de alguns controlos
- Estilos

Nota: Este tutorial é baseado num tutorial da Microsoft que pode ser encontrado em:

[https://msdn.microsoft.com/en-us/library/ms752299\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/ms752299(v=vs.110).aspx)

2.1. Aplicação tipo Navegador

Crie um novo Projeto “WPF application” com o nome DETI. Corra a aplicação e veja o que ocorre.

Modifique o projeto para que este seja uma aplicação para navegação entre páginas (e não entre janelas). Para tal, aceda ao ficheiro `MainWindow.xaml`, modifique o elemento `Window` para `NavigationWindow` e remova o componente `grid`. Pode ainda alterar outros parâmetros (por exemplo mude o título para DETI e altere o tamanho para 400 x 600).

Deve ainda modificar, no ficheiro `C#` associado, o objeto `window` para `NavigationWindow`. Corra a aplicação agora e compare o resultado.

2.2. Adição de uma janela

Crie uma nova página WPF (no projeto, seleccionar `add->new Item->Page(WPF)`) com o nome DETI-Home. No Ficheiro XAML mude o seu nome para DETI Home. Coloque esta janela como `source` no ficheiro `MainWindow.xaml` usando o código seguinte no fim do bloco `NavigationWindow`.

```
Title="DETI" Height="400" Width="600" Source="DETI-Home.xaml">
```

Da mesma forma, adicione outra página chamada DETI-Cursos que irá mostrar as disciplinas associado a um curso em concreto. Use como título para a página o nome DETI-Cursos.

2.3. Layout

O WPF disponibiliza vários *layouts* para organizar os conteúdos numa janela permitindo comportamentos e organização diferente dos componentes ao redimensionar as mesmas.


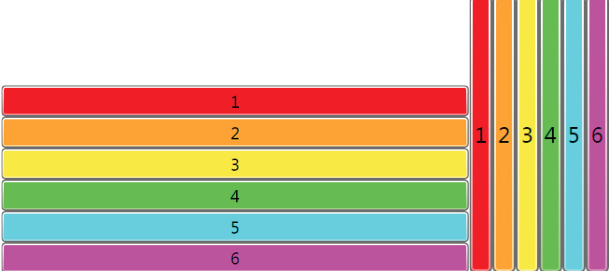
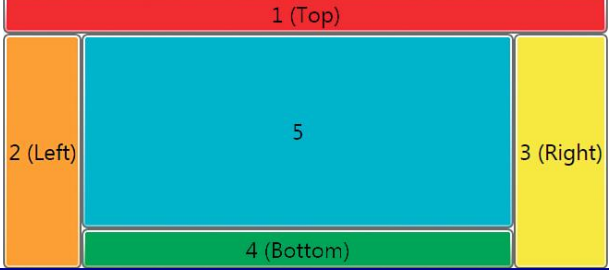

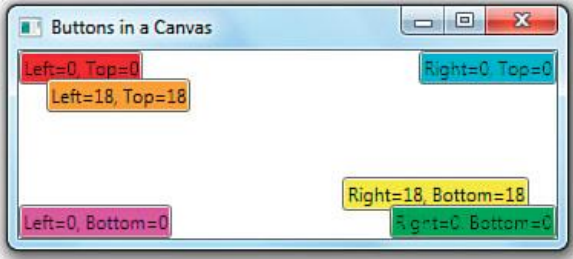
Layout	Exemplo	Organização
Grid		Organiza os elementos em linhas e colunas
Stack Panel		Empilha os elementos horizontalmente ou verticalmente
Dock Panel		Organizar os conteúdos colando-os verticalmente ou horizontalmente uns aos outros.
Wrap Panel		Organiza os elementos da esquerda para a direita até ao limite do elemento onde está definido.
Canvas Panel		Organiza os elementos pelas suas coordenadas (x,y)

Tabela 2-1: Principais *layouts* disponíveis em (fonte: WPF Unleashed - Adam Nathan)

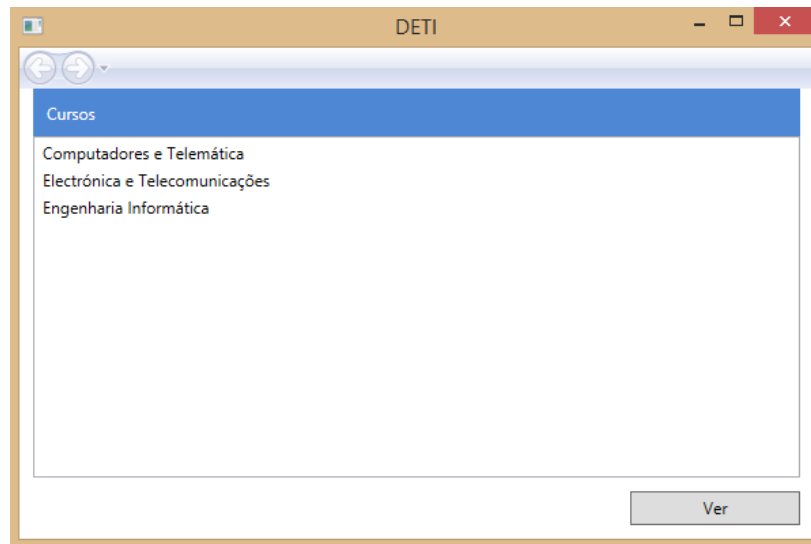
Adicione um *layout* do tipo grelha (*grid*) com uma só coluna e três linhas na janela DETI-Home. Pode definir a grelha usando a *toolbox*, ou adicionando o código seguinte no ficheiro xaml. Note que a primeira e a última linha são formatadas em função do conteúdo (opção *auto*).

```
<Grid Margin="10,0,10,10">
  <Grid.ColumnDefinitions>
    <ColumnDefinition />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition />
    <RowDefinition Height="Auto"/>
  </Grid.RowDefinitions>
</Grid>
```

Adicione três controlos na grelha utilizando as palavras chaves *Grid.Row* e *Grid.Column* para especificar a posição na grelha. A interface deve ter um *label* (com o nome *Cursos*), uma *listbox* com o nome *cursosListBox* (adicione 3 *<ListBoxItem>* para os três cursos do DETI) e um botão para ver o curso em questão. No final o resultado deve ser semelhante a janela seguinte. Coloque o *label* “cursos” num *border* com as características seguintes para ter o aspeto visual pretendido:

```
<Border Grid.Column="0" Grid.Row="0" Height="35"
  Padding="5" Background="#4E87D4">

</Border>
```



2.4. Adição de uma imagem

Adicione a imagem fornecida (Logo_UA.jpg) ao projeto no visual Studio. Adicione na grelha mais uma coluna a esquerda com tamanho fixo de 200, e mais uma linha.

Use os atributos *Grid.Column* e *Grid.Row* para colocar os controlos na posição correta para obter a figura seguinte.

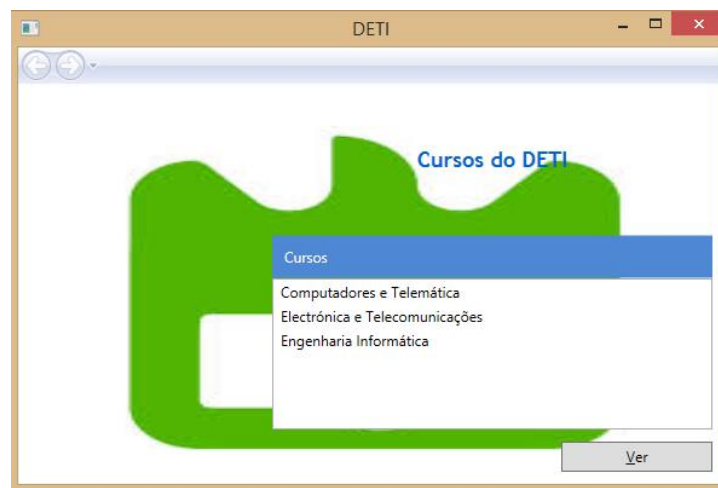
Adicione ainda um título geral a janela usando o código seguinte:

```
<Label Grid.Column="1" VerticalAlignment="Center"
      FontFamily="Trebuchet MS" FontWeight="Bold" FontSize="18"
      Foreground="#0066cc">
    Cursos do DETI
</Label>
```

Utilize o logotipo como imagem de fundo usando o código seguinte:

```
<Grid.Background>
    <ImageBrush ImageSource="Logo_UA.jpg"/>
</Grid.Background>
```

A janela deve ficar com um aspeto semelhante a figura seguinte:



2.5. Navegação entre páginas

Adicione um evento ao botão “ver” e coloque o seguinte necessário para visualizar outra página:

```
DETI_Cursos cursosPage = new DETI_Cursos();
this.NavigationService.Navigate(cursosPage);
```

Modifique o botão ver para permitir ativar o mesmo através do acelerador alt+v (basta colocar um *underscore* antes da letra de ativação no nome).

Aproveitando ao máximo o código da página criada anteriormente, modifique a página DETI-Cursos para obter algo semelhante com a janela seguinte a clicar no botão ver.



2.6. Acesso aos controlos

Faça as alterações necessárias para que, em frente do título nome, apareça o nome do curso selecionado na `listbox` na página DETI-Home quando é aberta a página cursos. Pode usar o código seguinte para aceder e converter para `String` o conteúdo selecionado numa `listbox`:

```
((ListBoxItem)cursosListBox.SelectedValue).Content.ToString()
```

Modifique o código para que o duplo clique num elemento da `listbox` na página inicial tenha o mesmo efeito que carregar no botão ver.

Altere ainda o código para verificar se algum item está selecionado (o item ser não nulo) e nesse caso dar uma mensagem de erro usando o código seguinte. Ao que correspondem os vários parâmetros?

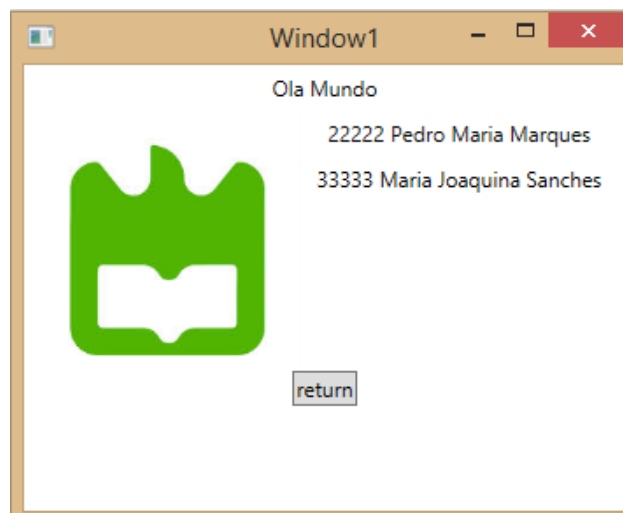
```
MessageBox.Show("Selecione um curso", "Erro", MessageBoxButton.OK);
```

2.7. Aplicação tipo janela *Standalone*

Crie um novo Projeto “WPF application” no Visual Studio em C# com o nome DETI_windows. Pode criar este novo projeto na solução já criada tendo assim acesso aos dois projetos na mesma solução. Adicione 2 botões com os nomes janela1 e janela 2. Crie uma nova janela no projeto anterior com o nome exemplo 1. No código da janela principal utilize o botão janela 1 para invocar uma janela exemplo 1 com o método `showDialog` e o botão 2 para invocar a mesma janela com o método `Show`. Qual a diferença?

Adicione uma `listbox` na `grid` da janela exemplo 1 e utilizando como base a estrutura seguinte, crie uma janela com um `layout` semelhante ao exemplo na figura.

```
<Grid>
    <StackPanel>
        <Label />
        <WrapPanel>
            <Image Source="Logo_UA.jpg" />
            <StackPanel >
                <Label />
                <Label />
            </StackPanel>
        </WrapPanel>
        <Button/>
    </StackPanel>
</Grid>
```



Adicione o código necessário para fechar a janela ao clicar no botão `return`.
 Altere o layout `wrapPanel` para outros layouts e veja o resultado.

2.8. Exemplo de Estilos e Triggers

O WPF disponibiliza estilos que permitem modificar a aparência ou comportamento de um controlo. Imagine por exemplo que quer alterar a aparência e comportamento das caixas de texto do exemplo anterior.

Define um estilo nas `Application resources` (para que o mesmo esteja disponível para toda a aplicação) usando o código seguinte:

```
<Style x:Key="LabelStyle" TargetType="Label">
  <Setter Property="Height" Value="25" />
  <Setter Property="Width" Value="180" />
  <Setter Property="HorizontalContentAlignment" Value="Center" />
</Style>
```

Modifique o seu programa para que todos os `labels` da aplicação use este estilo (use o código: `Style="{StaticResource LabelStyle}"`). Note que se definir propriedades locais para um controlo as mesmas terão prevalência sobre o estilo. Mude o estilo para um `background LightCoral` e veja o resultado. Pode ainda alterar outras propriedades do `label`.

É ainda possível em wpf utilizar `triggers` para modificar o comportamento em função de condições associadas a propriedades, eventos, alterações de dados ou até a combinação lógica de vários `triggers`. Adicione o bloco de código seguinte no estilo do `label` criado e observe o resultado.

```
<Style.Triggers>
  <Trigger Property="IsMouseOver" Value="True">
    <Setter Property="Background" Value="Red"/>
  </Trigger>
</Style.Triggers>
```