07

Grids

Transcrição

Com uma lista contendo os nomes dos veículos, é necessário exibir também seus preços. Incluiremos portanto outros `Label` s para isto:

```
<StackLayout>
    <Label Text="Azera V6" />
    <Label Text="60000" />
    <Label Text="Fiesta 2.0" />
    <Label Text="50000" />
    <Label Text="HB20 S" />
    <Label Text="40000" />
</StackLayout>
```

Vamos rodar a aplicação e verificar como os preços aparecerão na lista. Como já esperávamos, os preços apareceram um embaixo de cada nome de veículo.

Segundo a especificação de layout, queremos uma grade com duas colunas e várias linhas, cada uma contendo um veículo, sendo que a primeira coluna possui o nome, e a segunda, seu preço.

Da maneira como ele se encontra neste momento, o `StackLayout` está empilhando os nomes e os valores, não permitindo o formato em *grid*. Por isto, substituiremos pelo controle `Grid` , após o qual rodaremos a app para ver seu comportamento.

```
<Grid>
    <Label Text="Azera V6" />
    <Label Text="60000" />
    <Label Text="Fiesta 2.0" />
    <Label Text="50000" />
    <Label Text="HB20 S" />
    <Label Text="40000" />
</Grid>
```

O emulador nos mostra todos os textos sobrepostos no canto superior esquerdo da tela. O `Grid` não distribuiu os componentes em várias linhas e colunas, pois apesar de funcionar como uma grade, ele não define automaticamente o desenho da grade, e a quantidade de colunas e linhas.

Como ainda não definimos, entende-se que queremos apenas uma linha e uma coluna, e por isto um controle está se sobrepondo o outro. Neste caso, informaremos o `Grid` para que haja duas colunas, uma para o nome do veículo, e outra para o preço, e três linhas, uma para cada veículo:

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Label Text="Azera V6" />
```

```
    <Label Text="60000" />
    <Label Text="Fiesta 2.0" />
    <Label Text="50000" />
    <Label Text="HB20 S" />
    <Label Text="40000" />
</Grid>
```

Feito isto, rodaremos a aplicação novamente, e vemos que nada foi alterado, ainda temos apenas uma coluna. Apesar de termos definido que a grade terá duas colunas, não informamos quais componentes estarão na primeira, e quais estarão na segunda. Acrescentaremos em cada um destes `Label` s a coluna em que ele irá se encaixar, por meio da propriedade `Grid.Column` , definindo-se o índice da coluna, cujo valor começa do `0` .

Assim, para cada `Label` com o nome do veículo, o índice será `0` , e para cada preço, será `1` .

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Text="Azera V6" />
    <Label Grid.Column="1" Text="60000" />
    <Label Grid.Column="0" Text="Fiesta 2.0" />
    <Label Grid.Column="1" Text="50000" />
    <Label Grid.Column="0" Text="HB20 S" />
    <Label Grid.Column="1" Text="40000" />
</Grid>
```

Agora vamos rodar a app para verificar como isto é distribuído. Desta vez, a grade foi dividida em duas colunas, porém tudo ficou na mesma linha, ainda em sobreposição. Alteraremos isto com o `RowDefinitions` , acrescentando também a posição da linha para cada `Label` :

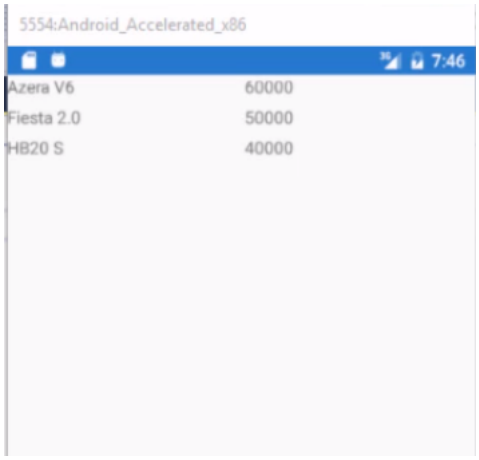
```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition></RowDefinition>
        <RowDefinition></RowDefinition>
        <RowDefinition></RowDefinition>
    </Grid.RowDefinitions>
    <Label Grid.Row="0" Grid.Column="0" Text="Azera V6" />
    <Label Grid.Row="0" Grid.Column="1" Text="60000" />
    <Label Grid.Row="1" Grid.Column="0" Text="Fiesta 2.0" />
    <Label Grid.Row="1" Grid.Column="1" Text="50000" />
    <Label Grid.Row="2" Grid.Column="0" Text="HB20 S" />
    <Label Grid.Row="2" Grid.Column="1" Text="40000" />
</Grid>
```

Ao rodarmos a app mais uma vez, verifica-se que as linhas e colunas estão melhor distribuídas na tela, por igual. Para diminuir o espaçamento entre as linhas, precisamos indicar à grade para que elas não sejam expandidas, pela

propriedade `VerticalOptions`, que por padrão se encontra com `Fill` (que significa "preencher", no caso, toda a tela). Iremos alterar pelo valor `Start`, que fará com que tudo fique no topo, sem expandir mais do que o necessário.

```
<Grid VerticalOptions="Start">
    <Grid.ColumnDefinitions>
        <ColumnDefinition></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition></RowDefinition>
        <RowDefinition></RowDefinition>
        <RowDefinition></RowDefinition>
    </Grid.RowDefinitions>
    <Label Grid.Row="0" Grid.Column="0" Text="Azera V6" />
    <Label Grid.Row="0" Grid.Column="1" Text="60000" />
    <Label Grid.Row="1" Grid.Column="0" Text="Fiesta 2.0" />
    <Label Grid.Row="1" Grid.Column="1" Text="50000" />
    <Label Grid.Row="2" Grid.Column="0" Text="HB20 S" />
    <Label Grid.Row="2" Grid.Column="1" Text="40000" />
</Grid>
```

Com isto, podemos rodar novamente a aplicação, e veremos que temos a grade como gostaríamos:



Ainda assim, existe um problema: no momento há três veículos, mas imagine que tenhamos dez, trinta ou duzentos veículos. Fica inviável trabalharmos com a grade desta forma, pois teremos que inserir muitas linhas manualmente. Em breve aprenderemos como contornar isto utilizando outro componente.