01

Padding, TableView, Intent, Section e SwitchCell on

Transcrição

Veremos como desenhar a tela de detalhe do veículo selecionado pelo usuário. Os veículos serão utilizados posteriormente no agendamento do *Test Drive*.

Na tela de detalhe, atualmente, temos o título com o nome do veículo selecionado pelo usuário, e o botão "Próximo", que levará à navegação para a página seguinte: de agendamento.

Implementaremos o "miolo" a seguir com uma seção chamada "Acessórios", contendo três chaves de seleção, de "liga e desliga", e "Total", com o valor final a ser cobrado conforme as escolhas do usuário:

Como traduzimos esta interface para os controles e componentes do Xamarin Forms? O que temos no meio desta tela é o que chamamos de `TableView`, uma visualização de tabela. Nele, teremos uma sequência de campos, geralmente de formulário. Neste caso, porém, serão campos de configuração.

Se você observar seu aparelho Android, ou mesmo um iPhone, verá que algumas telas possuem características diferentes, com aparência de tela de configuração, em que é possível ativar ou desativar o Bluetooth ou o Wi-Fi, por exemplo. Veremos como implementar uma tela destas, configurando quais acessórios estarão disponíveis no veículo para *Test Drive*.

Antes de incluirmos este novo controle com os acessórios possíveis de serem ligados ou desligados, teremos que acrescentar um controle que aja como um container com configurações, além do botão.

Como visto anteriormente, poderemos utilizar um componente que nos auxiliará a empilhar estes controles na tela, o `StackLayout`, que deixaremos com algumas margens usando o `Padding`, pois queremos que ele não fique tão próximo

das bordas. No caso com valor `25`.

Antes do botão, colocaremos a tabela de configurações (`TableView`) com um componente interno, raiz, chamado `TableRoot`, dentro do qual haverá uma seção. Pode-se ter uma série de configurações, as quais podem ser quebradas e agrupadas, como ocorre no seu celular: configurações de conexão, bateria, usabilidade, opções do usuário, tela, esquema de cores, e por aí vai.

Ou seja, é possível agrupar configurações diferentes dentro de um grupo lógico, que faça sentido. Outra característica importante de se definir no `TableView` é sua intenção em relação a ele, como o tipo de informação que ele irá conter, e que será coletado.

Existem algumas opções de intenções de `TableView` que serão acessadas por meio da propriedade `Intent`. Com ela, podemos especificar se informaremos dados, no caso de trabalharmos com um formulário a ser preenchido com texto ou data; se trabalharmos com um menu, será uma lista simples que selecionaremos e ele executará a ação; ou se for um `Settings` (em inglês significa "configurações"), que dará a nossa tela a aparência de configuração.

Esta última será a nossa intenção, portanto escolheremos `Settings`. Dentro do `TableRoot` faremos o agrupamento da seção, com `TableSection`, para o qual definiremos um título, texto que aparecerá no topo da tela, neste agrupamento de configurações.

```
<StackLayout Padding="25">
    <TableView>
        <TableRoot>
            <TableSection Title="Acessórios">

            </TableSection>
        </TableRoot>
    </TableView>

    <Button x:Name="buttonProximo" Text="Próximo" Clicked="buttonProximo_Clicked" VerticalOptio
</StackLayout>
```

Após estas alterações, vamos rodar a aplicação para verificar a aparência do `TableView` inicial. Iremos à página de detalhes clicando em um dos veículos, e conseguimos ver a `TableView`, ainda incompleta, por não termos terminado de configurá-la.

Agora começaremos a implementar cada uma das chaves que serão ativadas ou desativadas conforme a necessidade do usuário, em `TableSection`. É importante sabermos que em um `TableView` não podemos colocar qualquer controle, como `Label` ou `Button`.

Existem alguns controles próprios para serem utilizados dentro de um `TableView`. Cada controle destes é um tipo de célula (inclusive seus nomes terminam com "*Cell*"). Em inglês, qual o nome do controle que liga ou desliga algo? "*Switch*", certo? Usaremos uma célula de `Switch` para ativar ou desativar uma configuração, que neste caso será um acessório de veículos.

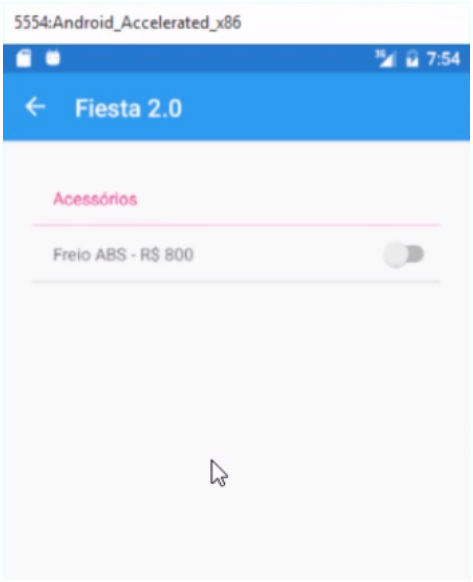
Teremos que definir o texto para este `Switch`, acompanhado por seu preço, a ser somado no cálculo final do valor.

```
<StackLayout Padding="25">
    <TableView>
        <TableRoot>
```

```
            <TableSection Title="Acessórios">
                <SwitchCell Text="Freio ABS - R$ 800"></SwitchCell>
            </TableSection>
        </TableRoot>
    </TableView>

    <Button x:Name="buttonProximo" Text="Próximo" Clicked="buttonProximo_Clicked" VerticalOptio
</StackLayout>
```

Vamos rodar a aplicação e ver o resultado obtido:



O `SwitchCell` também funciona como um *checkbox*, pois ao observar um aparelho Android, por exemplo, ele não terá configurações marcáveis em um *checkbox*. O `SwitchCell` possui função binária, armazenando valores "sim" ou "não", "verdadeiro" ou "falso", "ligado" ou "desligado".

Agora, vamos trazer o valor do acessório freio ABS ativado. Como faremos isto? Queremos que, ao abrir a tela, o usuário já tenha o freio ABS selecionado. Faremos isto por meio da propriedade do `SwitchCell` denominada `On`, que em inglês significa "ligado":

```
<SwitchCell Text="Freio ABS - R$ 800" On="True"></SwitchCell>
```

Rodando a aplicação mais uma vez, veremos que aprendemos como utilizar um `SwitchCell` em um `TableView`, trazendo seu valor previamente ativado, por padrão.