06

DatePicker, TimePicker e ViewCell

Transcrição

Com os três campos de entrada destinados para "nome", "telefone" e "e-mail" do usuário definidos a partir de `EntryCell`, precisaremos de propriedades que armazenarão tais valores ao serem digitados. Vamos cria-las no *code behind* (`AgendamentoView.xaml.cs`), abrindo o arquivo e indo embaixo da propriedade `Veiculo`, em que incluiremos os novos códigos.

As propriedades `Nome`, `Fone` e `Email` serão `string`s que armazenarão os dados digitados pelo usuário na tela de agendamento.

```
public string Nome { get; set; }
public string Fone { get; set; }
public string Email { get; set; }
```

É necessário também utilizar o `Binding` no arquivo `AgendamentoView.xaml`, em cima de uma propriedade do componente `EntryCell`. Tal propriedade que armazenará o texto digitado se chama `Text`, do controle `EntryCell`, com `Binding` para as propriedades que acabamos de colocar no *code behind*:

```
<TableSection Title="Seus Dados">
    <EntryCell Label="Nome:" Text="{Binding Nome}"></EntryCell>
    <EntryCell Label="Fone:" Keyboard="Telephone" Text="{Binding Fone}"></EntryCell>
    <EntryCell Label="E-mail:" Keyboard="Email" Text="{Binding Email}"></EntryCell>
</TableSection>
```

Agora podemos rodar a aplicação e ver se nada foi quebrado. Iremos ao próximo passo, que tem a ver com o preenchimento dos dados relativos ao agendamento de *Test Drive*. Acrescentaremos algum componente que possibilite a entrada destes dados.

Poderíamos utilizar o `EntryCell`, porém ele não é formatado, sendo um componente que aceita qualquer texto, desde que se tenha o teclado adequado, por exemplo, para e-mail e telefone. Queremos algo mais específico, adequado à definição de data e hora.

Para isto, utilizaremos o componente específico do Xamarin Forms chamado `DatePicker`, [disponível em sua documentação (https://developer.xamarin.com/api/type/Xamarin.Forms.DatePicker/)](). O `DatePicker` pode ser customizado de acordo com o sistema operacional do *smartphone*.

Como podemos ver pelas propriedades públicas desta classe, existe um `DateTime`, portanto utilizaremos o `DatePicker` para a coleta da data escolhida pelo usuário.

```
<TableSection Title="Agendamento">
    <DatePicker></DatePicker>
</TableSection>
```

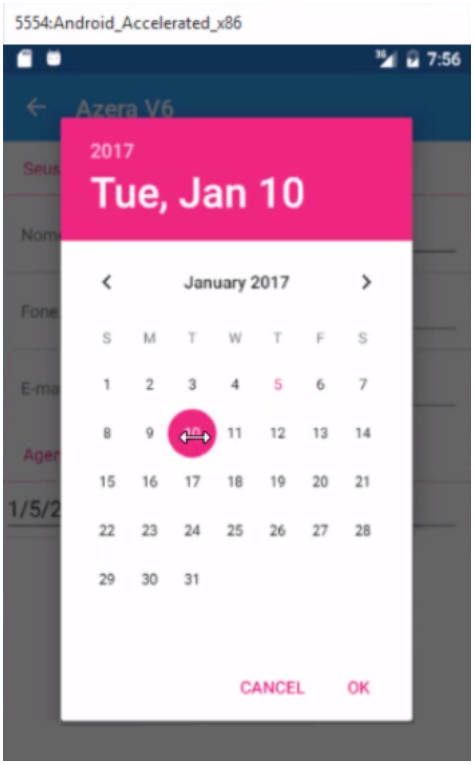
Após o qual rodaremos a aplicação para verificar seu funcionamento na tela. Apareceu um erro; o Visual Studio nos informa que o componente `DatePicker` não pode ser convertido para um tipo `Xamarin.Forms.Cell`.

Como vimos anteriormente, não podemos simplesmente acrescentar qualquer componente em um `TableView`, sendo aceitáveis apenas aqueles que têm `Cell`, o que não é o caso do `DatePicker` - não existe um `DatePickerCell`.

Para contornar este problema e conseguirmos utilizar o `DatePicker`, felizmente o Xamarin Forms permite que se utilize, em um `TableView`, um componente auxiliar denominado `ViewCell`, o qual possibilita o uso de componentes inicialmente não disponíveis ao `TableView`, como no caso do `DatePicker`. O código ficará assim:

```
<TableSection Title="Agendamento">
    <ViewCell>
        <DatePicker></DatePicker>
    </ViewCell>
</TableSection>
```

Vamos rodar a aplicação no emulador. Veremos que uma data é exibida (`1/5/2017`, cinco de janeiro de 2017) que, quando clicado, abre outra *view* para seleção de uma data:



O `DatePicker` é um componente nativo do sistema Android e, caso você rode esta aplicação em outro sistema operacional, ele irá obedecer seu padrão de exibição e seleção de data.

Agora colocaremos o selecionador de horas, assim como colocamos para a data, por meio de outro `ViewCell`, desta vez para o `TimePicker`:

```
    <ViewCell>
        <TimePicker></TimePicker>
    </ViewCell>
```

Vamos rodar a aplicação e verificar o que acontece. Selecionando a célula de hora, um relógio próprio do Android é exibido em outra *view*, podendo-se selecionar também entre `AM` e `PM`, ou seja, antes ou depois do meio dia.

Temos estes selecionadores de data e hora, mas falta um `Label`, algo que indique que eles são data e hora, para seguir o padrão do formulário. Desta forma, em `ViewCell` colocaremos também um `Label`, organizado por um container, o `StackLayout`:

```
<TableSection Title="Agendamento">
    <ViewCell>
        <StackLayout>
            <Label Text="Data:"></Label>
            <DatePicker></DatePicker>
        </StackLayout>
    </ViewCell>
    <ViewCell>
        <TimePicker></TimePicker>
    </ViewCell>
</TableSection>
```

Vamos rodar a aplicação e ver o que acontece? Existe algo errado, pois na página alterada é exibido "Data:", como esperado, porém o selecionador de data (`DatePicker`) está escondido, apesar de mesmo assim ser possível clicar nele.

Isto ocorre pois o `StackLayout` que colocamos está empilhando os componentes na vertical, porém cada linha em um `TableView` possui uma largura fixa. Então o `Label` é visível enquanto o `DatePicker` está escondido. Assim, é necessário fazer com que se empilhem os componentes na horizontal.

Voltando a `AgendamentoView.xaml`, modificaremos o `Orientation` do `StackLayout`, definindo-o como `Horizontal`:

```
<TableSection Title="Agendamento">
    <ViewCell>
        <StackLayout Orientation="Horizontal">
            <Label Text="Data:"></Label>
            <DatePicker></DatePicker>
        </StackLayout>
    </ViewCell>
    //...
</TableSection>
```

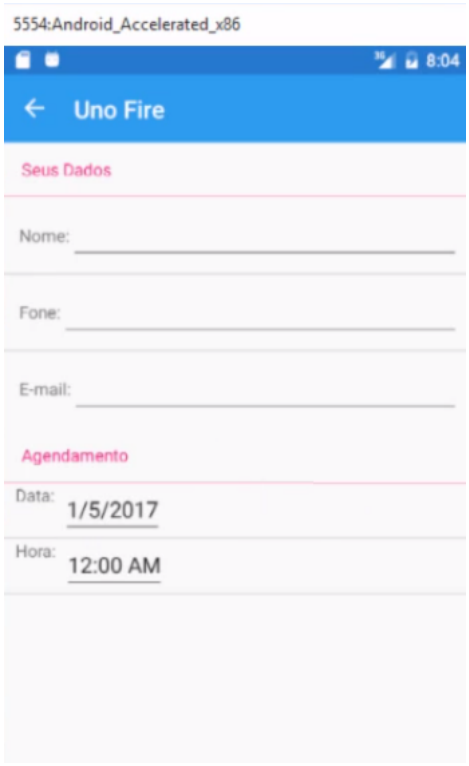
Feito isto, rodaremos a app, e veremos que a data e o título aparecem alinhados horizontalmente. Vamos fazer o mesmo com o selecionador de horas (`TimePicker`).

```
<TableSection Title="Agendamento">
    <ViewCell>
        <StackLayout Orientation="Horizontal">
            <Label Text="Data:"></Label>
            <DatePicker></DatePicker>
        </StackLayout>
    </ViewCell>
    <ViewCell>
        <StackLayout Orientation="Horizontal">
            <Label Text="Hora:"></Label>
            <TimePicker></TimePicker>
        </StackLayout>
    </ViewCell>
</TableSection>
```

Rodando a aplicação mais uma vez, fica faltando um pequeno ajuste de alinhamento dos primeiros campos de entrada em relação à parte "Agendamento". A seguir, vamos alterar a margem dos `StackLayout` s:

```
<TableSection Title="Agendamento">
    <ViewCell>
        <StackLayout Orientation="Horizontal" Margin="12,0,0,0">
            <Label Text="Data:"></Label>
            <DatePicker></DatePicker>
        </StackLayout>
    </ViewCell>
    <ViewCell>
        <StackLayout Orientation="Horizontal" Margin="12,0,0,0">
            <Label Text="Hora:"></Label>
            <TimePicker></TimePicker>
        </StackLayout>
    </ViewCell>
</TableSection>
```

Agora rodaremos a aplicação, obtendo-se um layout assim:



Com isto, vimos como fazer `Binding` de um `EntryCell` com as propriedades presentes no *code behind*, aprendemos como utilizar `DatePicker` e `TimePicker` para envio do agendamento e armazenamento dos dados no servidor. Ainda precisaremos incluir as propriedades do `Binding` do `DatePicker` e `TimePicker` , que veremos a seguir.