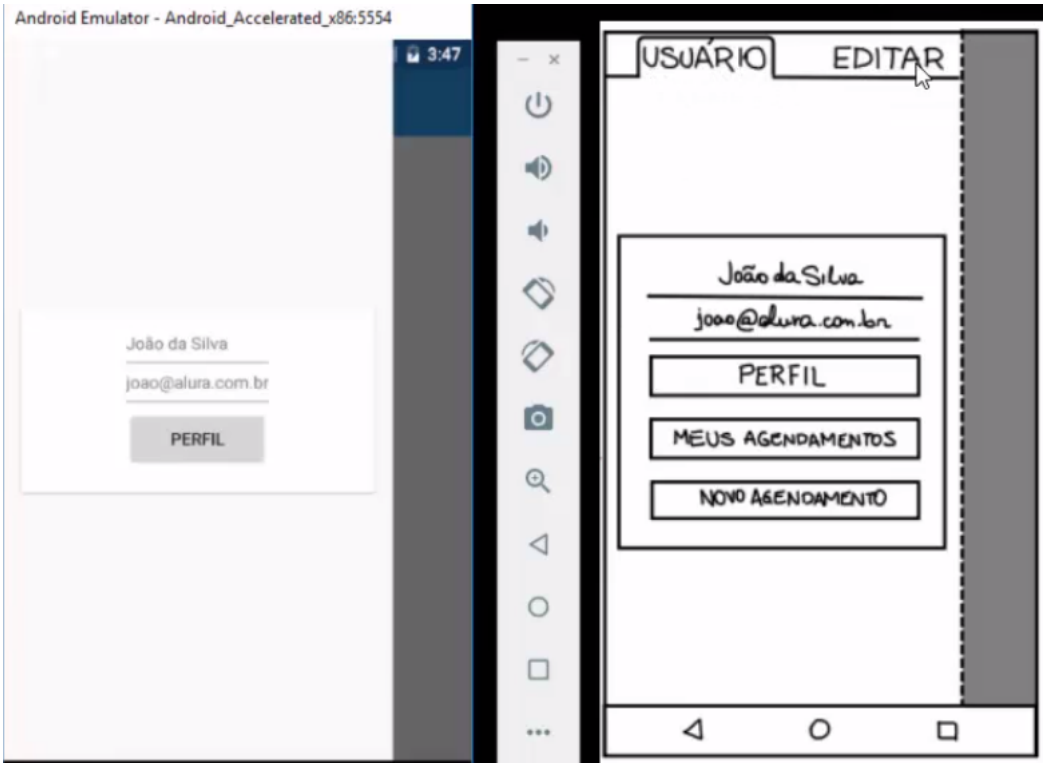
04

MasterView

Transcrição

Com os dados do usuário a partir da autenticação no servidor da Aluracar, temos na tela de perfil um layout ainda muito simples. Vamos deixá-lo mais parecido com a especificação que recebemos, com uma aba de "Usuário" e outra de "Editar", lado a lado:



Ainda faltam alguns botões, de "Perfil", "Meus agendamentos" e "Novo agendamento", porém trataremos disto mais para a frente. Note que na *view* (`MainView`) não há nenhuma aba. Precisaremos encontrar no Xamarin Forms algum tipo de *view* que as permita.

Dentre as opções disponíveis, existe a `ContentPage` , a mais simples, sem nenhum recurso de navegação, a `MasterDetailPage` , a que já estamos utilizando para "quebrar" a visualização entre perfil do usuário e listagem de veículos, `NavigationPage` , que permite a navegação para outras páginas, retornando posteriormente àquelas visitadas anteriormente. Há a `CarouselPage` , em que é possível visualizar o conteúdo como se fosse um "carrossel de páginas", mesmo. Existe também a `TabbedPage` , cujo nome indica que trata-se de uma página com abas.

Nela, tem-se a visualização de apenas uma página por vez, com a indicação de qual aba está selecionada no momento. Conforme selecionamos outra aba, a página que está sendo visualizada também é trocada. Conseguimos "chavear", trocar a página conforme a seleção das abas.

Neste caso, a aba "Usuário" nos levará àquela página de perfil de usuário que já conhecemos. Ao clicarmos em "Editar", teremos outra visualização, com campos abertos para modificação de dados. Em uma, apenas visualizamos, visualizamos e editamos na outra. Para transformarmos isto em um controle do Xamarin Forms, precisaremos utilizar o componente `TabbedPage`.

Para isto, poderíamos simplesmente criar uma nova página, ou *view*, chamando-a de `TabbedPage`. Podemos, no entanto, simplesmente trocar o tipo, de `MasterView`, por `TabbedPage`, da mesma maneira como já fizemos ao transformarmos um `ContentPage` em um `MasterDetailView`. Substituiremos `ContentPage` por `TabbedPage`, em `MasterView.xaml` (lembrando de fazer o mesmo na tag de fechamento correspondente).

Além disto, precisaremos trocar o tipo da página `MasterView` no *code behind*, pois se temos o `TabbedPage`, também precisamos tê-lo declarado no código C# em `MasterView.xaml.cs`. Onde temos `ContentPage`, substituiremos por `TabbedPage`:

```
namespace TestDrive.Views
{
  public partial class MasterView : TabbedPage
  {
    //...
  }
}
```

Aproveitando que já estamos neste código C#, temos o `ViewModel` sendo utilizado apenas uma vez; removeremos a linha `public MasterViewModel ViewModel { get; set; }`, cuja propriedade estamos declarando, passando o `BindingContext` recortando `new MasterViewModel(usuario);` e passando-o para a linha de baixo, deixando o código mais simples e limpo:

```
public MasterView (Usuario usuario)
{
  InitializeComponent();
  this.BindingContext = new MasterViewModel(usuario);
}
```

Voltando à `TabbedPage`, do que necessitamos para trabalhar com ela? Será que basta rodarmos a aplicação? Vamos testar fazendo isto e verificando sua exibição no emulador. Será exibido um erro: "objeto do tipo `Xamarin.Forms.Frame` não pôde ser convertido para o tipo `Xamarin.Forms.Page`", isto significa que o programa está tentando converter um `Frame` em uma página.

Ao verificarmos o código XAML (`MasterView.xaml`), o `Frame` se localiza onde deveria estar uma página. Isto ocorre quando temos um `TabbedPage`, temos abas e páginas, portanto precisamos declarar dentro dele suas páginas-filhas, aquelas que serão selecionadas para troca de páginas. Neste momento, estamos apenas colocando um `Frame`, quando necessitamos declará-las a partir da propriedade `Children` ("filhos" em inglês) que está dentro da classe `TabbedPage`.

Ao declararmos páginas *inline* anteriormente, com o `MasterDetail`, criamos uma tag chamada `ContentPage`, que é o tipo de página mais simples. Como neste caso são duas páginas, criaremos duas `ContentPage`s, uma para cada aba. O `Frame` referente a nome e e-mail acabou ficando meio solto no layout, então o copiaremos e colaremos dentro da primeira tag `ContentPage`. Ao fazermos isto, o `Frame` passa a fazer parte desta `ContentPage`, ou seja, ele passa a ser o conteúdo da primeira aba da `TabbedPage`.

```
<TabbedPage.Children>
    <ContentPage>
      <Frame OutlineColor="Silver" VerticalOptions="CenterAndExpand" Margin="15">
        <StackLayout VerticalOptions="Center" HorizontalOptions="Center">
          <Label Text="{Binding Nome}"></Label>
          <BoxView Color="Gray" HeightRequest="1" HorizontalOptions="Fill"></BoxView>
          <Label Text="{Binding Email}"></Label>
          <BoxView Color="Gray" HeightRequest="1" HorizontalOptions="Fill"></BoxView>
          <Button Text="Perfil"></Button>
        </StackLayout>
      </Frame>
    </ContentPage>

    <ContentPage>
    </ContentPage>
</TabbedPage.Children>
```

Podemos rodar a aplicação para ver como o Xamarin exibirá este `ContentPage`. Vemos uma linha acima do restante onde devem estar as abas (na parte superior do layout) e, ao clicarmos nelas, perceberemos que se movem, ou seja, realmente temos duas abas. No desenho do Xamarin Forms, as abas ficam embaixo, porém por padrão no Android, elas ficam em cima.

Além de gerar o código nativo, o Xamarin respeita o padrão de interface do sistema operacional em uso.

As abas implementadas ainda não possuem nenhum texto ou informação além daquelas que já vimos antes, não se sabe o que cada uma delas faz. Para definirmos um título para elas, voltamos a `MasterView.xaml` e, na primeira `ContentPage` contida em `TabbedPage.Children`, definimos o título "Usuário". Faremos o mesmo com a aba "Editar".

```
<TabbedPage.Children>
    <ContentPage Title="Usuário">
      <Frame OutlineColor="Silver" VerticalOptions="CenterAndExpand" Margin="15">
        <StackLayout VerticalOptions="Center" HorizontalOptions="Center">
          <Label Text="{Binding Nome}"></Label>
          <BoxView Color="Gray" HeightRequest="1" HorizontalOptions="Fill"></BoxView>
          <Label Text="{Binding Email}"></Label>
          <BoxView Color="Gray" HeightRequest="1" HorizontalOptions="Fill"></BoxView>
          <Button Text="Perfil"></Button>
        </StackLayout>
      </Frame>
    </ContentPage Title="Editar">

    <ContentPage>
    </ContentPage>
</TabbedPage.Children>
```

Rodaremos a aplicação repetindo os procedimentos de login.

Conseguimos implementar as abas corretamente e, ao trocarmos de aba, a página a ser exibida também mudará. A aba "Usuário", que era a antiga `MasterView` agora faz parte da página da primeira aba, e "Editar" por ora é uma página em branco que começaremos a implementar daqui a pouco.