04

LoginView

Transcrição

Este curso dá prosseguimento à primeira parte do curso de Xamarin Forms (https://www.alura.com.br/curso-online-xamarin-aplicativos-mobile-com-visual-studio-parte-1), e nele veremos algumas *features* bem interessantes do **Xamarin Forms**.

Nossa última jornada da parte 1 do curso termina com a aplicação para a empresa fictícia Aluracar, e uma listagem de veículos disponíveis para agendamento do *TestDrive*. Há uma seleção dos recursos opcionais do veículo escolhido, na qual iremos preencher os dados que serão enviados no fim.

Antes, aplicação se iniciava pelo carregamento da lista de veículos obtidos por meio de uma requisição HTTP GET no servidor da Aluracar.

Em geral, a aplicação deve ir direto ao ponto, mostrando a listagem dos veículos disponíveis para *TestDrive*. No entanto, este não é exatamente o fluxo que desejamos. Nós queremos que o aplicativo esteja disponível e acessível somente aos usuários cadastrados no sistema da Aluracar.

Queremos evitar que a aplicação seja aberta por usuários não logados, ou seja, anônimos.

Portanto, utilizaremos um mecanismo de autenticação do usuário, criando uma tela que receberá o login do usuário e sua senha. Desta forma protegeremos contra acessos indevidos de pessoas estranhas, ou caso tenham acesso ao seu smartphone, ou ainda em caso de roubo, extravio, ou mesmo alguém "xeretando" seus dados, coletando informações que não deveriam ser acessadas por terceiros.

Iniciaremos este mecanismo de autenticação pela tela inicial de login, que será a nova *view* inicial da aplicação, seu ponto de partida. Como faremos isto no Xamarin Forms? É muito simples.

Acessaremos o projeto *portable*, ou PCL (portátil, "TestDrive Portable"), em seguida, clicaremos em "Views" e adicionando um novo item (com o lado direito do mouse: "*Add* > *New Item*"). Na nova janela, selecionaremos "Forms Xaml Page" e criaremos um arquivo para esta nova *view* de login, chamada "LoginView".

Feito isto, precisamos descobrir como inseri-la ao fluxo de navegação do aplicativo para que seja um novo ponto de entrada da aplicação do Xamarin Forms.

No primeiro curso, vimos que para definirmos a *view* de entrada da aplicação, entramos na classe `App.xaml` e em `App.xaml.cs`, neste *code behind*, na linha:

```
MainPage = new NavigationPage(new ListagemView());
```

A propriedade `MainPage` da classe `App` definirá a *view* de entrada da aplicação, anteriormente definido como `NavigationPage`, uma estrutura ou "pilha de navegação" que começa pela *view* de listagem (`ListagemView`). O que faremos agora é definir uma nova instância de nossa nova *view*, de login, como `MainPage`, ou página principal da aplicação.

```
public partial class App : Application
{
  public App()
  {
    InitializeComponent();

    MainPage = new LoginView;
  }
//...
```

Talvez você esteja se perguntando por que simplesmente não trocamos o `ListagemView` por `LoginView`? Ou por que não mantivemos a pilha de navegação, trocando-a simplesmente por `LoginView`? O motivo é simples.

Imagine a situação em que fazemos o login, pela `LoginView`, e seguíssemos para a tela de listagem de veículos. Em seguida, se colocássemos tudo dentro de um `NavigationPage` (ou seja, uma pilha de navegação), quando navegarmos para outras páginas, como a listagem de veículos e, depois, nos detalhes do veículo selecionado. Se decidíssemos clicar no botão "Voltar" da navegação, retornaríamos à listagem. Ainda teríamos uma seta indicando que podemos voltar mais uma tela, chegando ao login.

Porém, o melhor é evitarmos o uso da tela de login como raiz da navegação, pois ao retornarmos das páginas seguintes, não queremos chegar até ao login. No máximo, poderíamos chegar na tela de listagem (`ListagemView`).

Por isto, a navegação inicial aponta diretamente para `LoginView`, e precisamos implementar a *view* de login. Recebemos uma especificação com o desenho da tela:

É uma tela simples, e a implementaremos utilizando o Xamarin Forms. Vamos abrir o arquivo `LoginView.xaml`, então, removeremos a tag `<Label>` que vem automaticamente quando criamos este item:

```
<Label Text="{Binding MainText}" VerticalOptions="Center" HorizontalOptions="Center"/>
```

No desenho recebido, há uma imagem com logotipo, uma caixa de texto para nome de usuário, outro para senha e um botão de "Entrar". Sendo a entrada de dados a parte mais simples, para "Usuário" e "Senha", utilizaremos alguns componentes do próprio Xamarin Forms, os quais permitem que digitemos.

No primeiro curso, vimos o controle chamado `EntryCell`, que nos permitia fazer uma digitação, ou entrada de dados, dentro de um outro controle, denominado `Table View`.

O `EntryCell` não está disponível entre os controles deste contexto de uma *Content Page*, portanto, deixaremos de usá-lo. Utilizaremos um controle similar fora do contexto do `Table View`, o `Entry`, que em inglês significa "Entrada". Dentro destas tags, incluiremos as informações do usuário, em seguida, adicionaremos um segundo par de tags para digitação da senha:

```
<!--imagem-->
<Entry></Entry>
<Entry></Entry>
<!--entrar-->
```

Podemos também acrescentar um *label*, ou rótulo, uma legenda que indique ao usuário o tipo de campo a ser preenchido naquele determinado espaço em branco. Quando colocamos a propriedade `label`, nenhuma opção é mostrada. Isto ocorre porque o controle `Entry`, diferentemente do `EntryCell`, não possui um *label*, ou seja, falta uma legenda indicando ao usuário qual campo deverá ser preenchido.

Para contornar esta situação, podemos colocar `StackLayout` na horizontal e, juntamente com o `Entry`, um campo `Label` indicando, por exemplo, "Usuário:":

```
<!--imagem-->
<StackLayout>
  <Label Text="Usuário:"></Label>
<Entry></Entry>
</StackLayout>
<!--entrar-->
```

No entanto, isto ficaria inapropriado para nossa tela. Se olharmos o wireframe enviado, o "Usuário" não é um *label* e a legenda exibida está dentro do próprio controle de entrada. Estamos impossibilitados de criar outro controle `Label` separadamente. A solução é colocar as legendas dentro do controle `Entry` usando a propriedade `Placeholder`:

```
<!--imagem-->
<Entry Placeholder="Usuário"></Entry>
<Entry Placeholder="Senha"></Entry>
<!--entrar-->
```

Enquanto estiver vazio, o texto definido como *placeholder* será exibido, e conforme os dados são digitados pelo usuário, o *placeholder* deixará de aparecer. Para organizarmos estes dois campos na tela, utilizaremos um recurso velho conhecido, o `StackLayout`. Este será responsável por "empilhar" os dois controles, ordenando-os verticalmente.

```
<StackLayout>
  <!--imagem-->
  <Entry Placeholder="Usuário"></Entry>
  <Entry Placeholder="Senha"></Entry>
  <!--entrar-->
</StackLayout>
```

Agora vamos rodar a aplicação e verificar que tudo isto é exibido. Temos "Usuário" e "Senha", os dois controles que acionamos. Para testarmos, digitaremos "fulano" como usuário e "fulano123" como senha.