06

Entry

Transcrição

IMPORTANTE: Para começar este vídeo, implemente as classes `Usuario` e `LoginResult`, conforme abaixo:

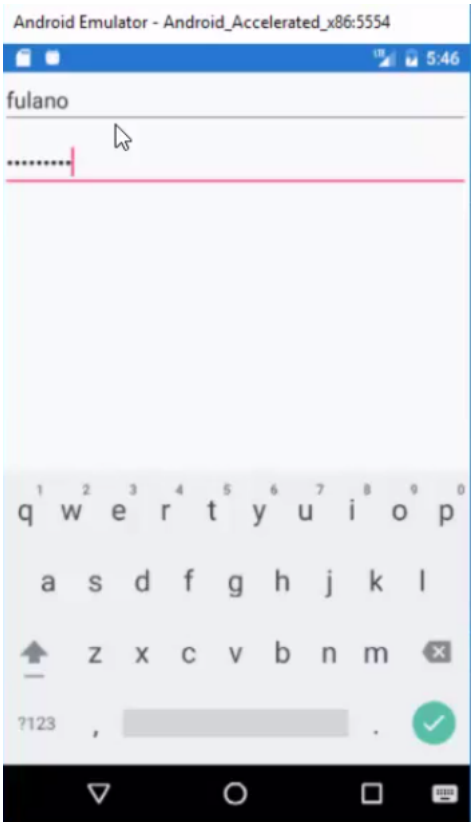
```
public class Usuario
{
    public int id { get; set; }
    public string nome { get; set; }
    public string dataNascimento { get; set; }
    public string telefone { get; set; }
    public string email { get; set; }
}

public class LoginResult
{
    public Usuario usuario { get; set; }
}
```

Quando o usuário preenche o campo de "Senha" com "fulano123", por exemplo, acontece algo estranho: será usado um padrão diferente de preenchimento de senhas, que geralmente envolve uso de caracteres especiais e ocultos. É melhor evitarmos a exibição de senhas por questões de segurança e privacidade.

Felizmente o Xamarin Forms tem uma maneira elegante e prática de resolver este problema, por meio da propriedade `IsPassword`, que o controle `Entry` possui, e que determina se é uma senha que está sendo digitada ou não.

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

Veremos como um campo determinado como senha se comporta rodando-se a aplicação. No emulador, digitamos "fulano" como "Usuário" e "fulano123" como "Senha". Conseguimos ver cada letra durante `1` ou `2s`, porém, no fim da digitação, todos os caracteres estarão ocultos, sendo substituídos por caracteres especiais.



Ainda precisamos acertar o layout desta tela, ainda muito diferente do que recebemos na especificação (mais agradável, alinhado no centro). Sendo assim, modificaremos a *view* para obtermos espaçamentos melhores.

Alteraremos o `StackLayout` para centralizarmos os elementos por meio da propriedade `VerticalOptions`, que determina como o `StackLayout`, que armazena os controles internos, será exibido verticalmente. Acrescentaremos também uma margem de `64`, um número aleatório.

```
<StackLayout VerticalOptions="Center" Margin="64">
    <!--imagem-->
    <Entry Placeholder="Usuário"></Entry>
    <Entry Placeholder="Senha" IsPassword="True"></Entry>
    <!--entrar-->
</StackLayout>
```

Rodando a aplicação, verificaremos como se exibe o layout:

Agora, a tela de login está "tomando corpo", vamos continuar inserindo o logotipo da Aluracar, enviado pela equipe de design da empresa.

> Para fazer esse exercício, abra este link (https://raw.githubusercontent.com/alura-cursos/xamarin-crie-aplicacoes-mobile-parte-2/master/TestDrive/TestDrive.Droid/Resources/drawable/aluracar.png), clicando com o botão direito na imagem, e em seguida em "Salvar como" para baixá-la.

Para adicionar esta imagem ao projeto, pausamos a aplicação, abriremos `TestDrive.Droid` , indo em "Resources > drawable". Este é o diretório padrão para inserção de imagens, e é onde salvaremos o logotipo recebido pela Aluracar.

A imagem não aparece imediatamente na pasta pois não a adicionamos ao projeto. Para tal, clicaremos no botão "*Show All Files*" com o lado direito do mouse, selecionando em seguida "*Include In Project*", sinalizando que ele pode ser utilizado no projeto Xamarin.

O próximo passo consiste em consumirmos esta imagem na aplicação. Como exibimos uma imagem em um projeto Xamarin Forms? É muito simples: substituiremos `<!--imagem-->` pelo componente de imagem `Image` . Porém, precisamos definir o arquivo de imagem a ser utilizado, sua propriedade, origem ou fonte, que serão simplesmente o nome e extensão da imagem.

```
<StackLayout VerticalOptions="Center" Margin="64">
    <Image Source="aluracar.png"></Image>
    <Entry Placeholder="Usuário"></Entry>
    <Entry Placeholder="Senha" IsPassword="True"></Entry>
    <!--entrar-->
</StackLayout>
```

Feito isto, a imagem aparecerá como é exibido abaixo:

Falta adicionarmos o botão de login, "Entrar", para realizar autenticação da tela de login. O componente `Button` fará isto, e definiremos como texto o que queremos que seja exibido nele:

```
<StackLayout VerticalOptions="Center" Margin="64">
    <Image Source="aluracar.png"></Image>
    <Entry Placeholder="Usuário"></Entry>
    <Entry Placeholder="Senha" IsPassword="True"></Entry>
    <Button Text="Entrar"></Button>
</StackLayout>
```

Vamos rodar a aplicação mais uma vez, verificando-se como ficou o layout:

Neste momento, quando a tela de login é exibida e preenchemos os campos disponíveis, quando clicamos em "Entrar", nada acontecerá. Pensaremos agora em como navegar às próximas telas da app. Considerando que a navegação é mais simples do que a autenticação, ou seja, do que a verificação de usuário e senha no banco de dados do servidor, vamos começar por ela.

Programaremos o evento do botão acionado assim que for clicado pelo usuário, o `Clicked`, apertando o "TAB" para adição do método `Button_Clicked`, o qual aparecerá no *code behind* da *view* de login (`LoginView`).

```
<StackLayout VerticalOptions="Center" Margin="64">
    <Image Source="aluracar.png"></Image>
    <Entry Placeholder="Usuário"></Entry>
    <Entry Placeholder="Senha" IsPassword="True"></Entry>
    <Button Text="Entrar" Clicked="Button_Clicked"></Button>
</StackLayout>
```

No lado direito da tela do Visual Studio, acessaremos "Views > LoginView.xaml > LoginView.xaml.cs". Quando abrimos este arquivo, veremos que foi criado o método `Button_Clicked`. Em seguida, iremos à tela de listagem de veículos (`ListagemView.xaml`) para que, ao clicarmos no botão, isto ocorrerá sem que haja necessidade de autenticação.

No entanto, é insuficiente que o usuário navegue pela aplicação. Queremos criar a mesma pilha de navegação que tínhamos antes. Neste ponto, estabeleceremos que o novo `MainPage` da app, ou seja, sua página principal, seja a `ListagemView`. Vimos que ela é definida em `App.xaml.cs`, é nela que devemos alterar o `MainPage` (e não em `LoginView.xaml.cs`).

Precisamos acessar a classe `App` a partir de `LoginView`. Devemos utilizar a técnica de troca de mensagens entre componentes vistos no primeiro curso de Xamarin (https://www.alura.com.br/curso-online-xamarin-aplicativos-mobile-com-visual-studio-parte-1), por meio da classe `MessagingCenter` do Xamarin Forms.

A partir de `LoginView`, a classe `App.xaml.cs` será "avisada" sobre uma nova `MainPage` a ser definida para a aplicação. Desta forma, no evento `Button_Clicked` do `LoginView.xaml.cs`, lançaremos uma mensagem:

```
private void Button_Clicked(object sender, EventArgs e)
{
  MessagingCenter.Send<Usuario>(new Usuario(), "SucessoLogin");
}
```

Também precisamos definir o tipo de mensagem. Neste caso será `Usuario`, que a enviará a partir de `LoginView`, e será recebida na classe `App`, passando por parâmetro o novo usuário. O nome desta mensagem será do tipo `SucessoLogin`, e inicialmente todo login será bem-sucedido pois ainda falta criar um mecanismo de autenticação.

No evento `OnStart()` do arquivo `App.xaml.cs`, ao inicializarmos, rodaremos `MessagingCenter.Subscribe` para a captura da mensagem, então substituiremos a linha `// Handle when your app starts`, como é mostrado abaixo:

```
protected override void OnStart()
{
  MessagingCenter.Subscribe<Usuario>(this, "SucessoLogin",
  (usuario) =>
  {
    MainPage = new NavigationPage(new ListagemView());
  });
}
```

No código acima, `Usuario` é o tipo de objeto que está sendo passado na mensagem, e o objeto responsável por capturá-laé `this`, que referencia o próprio `App`. O nome da mensagem é `SucessoLogin`, como já havíamos determinado. Colocamos também um método anônimo (uma expressão lambda) que receberá o procedimento a ser executado quando a mensagem for capturada. Nele, deve ocorrer a alteração do `MainPage`. Tínhamos um `NavigationPage` no início do curso, para o qual se passava a instância nova do `ListagemView` como raiz desta pilha da navegação.

Feitas estas alterações, rodaremos a aplicação para ver se o mecanismo de navegação funciona. Faremos um teste preenchendo os campos com palavras quaisquer e, com isto, acabamos de navegar para a página de `ListagemView`, a página inicial da nossa navegação.

Conforme vamos navegando, conseguimos voltar pela seta que indica o sentido à esquerda (localizado no topo do layout). Assim, chegamos até a página de listagem de veículos, porém, sem a possibilidade de voltar mais do que isso, porque a tela inicial (a raiz de navegação) é `ListagemView`.