01

ListagemView, Navigation, PushAsync, NavigationPage e Title

Transcrição

Conseguimos implementar com sucesso uma listagem com os veículos disponíveis ao usuário no momento de agendamento do *Test Drive*. Desta forma, sempre que o usuário clicar em um dos itens, uma mensagem será exibida na tela dizendo que ele tocou em um determinado veículo com respectivo valor.

Nosso próximo objetivo será criar a navegação para a tela seguinte a partir da seleção de um dos veículos. Esta nova tela exibirá acessórios e o total do preço a ser pago, cujos cliques levarão a outras telas, e assim por diante.

Trata-se de uma sequência de telas, ou *views*, que serão acessadas pelo usuário para que se complete o fluxo da aplicação. Implementaremos esta navegação entre as telas, e o primeiro passo é modificar o projeto para que se comportem novas telas.

Para isto, queremos organizar melhor os itens do nosso projeto. Estamos trabalhando com a página de listagem que em nosso projeto é chamado de `MainPage.xaml` (em inglês remete a "página principal"). No entanto, daremos um nome mais preciso e claro a ela. Vamos renomeá-la de "ListagemView".

Além disto, é preciso alterar o nome da classe que se encontra no XAML. Substituiremos `MainPage` por `ListagemView` na tag `<ContentPage>`, algo que precisa ser feito no *code behind* também:

```
public partial class ListagemView : ContentPage
```

Agora vamos rodar a aplicação para verificar se nada foi quebrado. Funcionou! O próximo passo consiste em deixarmos esta e as próximas *views* em uma pasta específica, chamada "Views".

Para isto, clicaremos com o lado direito do mouse em "TestDrive (Portable)", e depois em "*Add* > *New Folder*", nomeando esta nova pasta como "Views", para a qual moveremos `ListagemView.xaml`. Precisaremos acertar o *namespace* neste arquivo, alterando `TestDrive.ListagemView` por `TestDrive.Views.ListagemView`.

Abriremos `ListagemView.xaml.cs` alterando o *namespace* de acordo com a estrutura do nosso projeto, deixando `namespace TestDrive.Views`, e não apenas `namespace TestDrive`.

Rodaremos novamente a aplicação, e descobriremos um erro: o tipo ou *namespace* `ListagemView` não existe no *namespace* `TestDrive`. O erro está no arquivo denominado `App.xaml.cs`, e o programa não reconhece o *namespace* da nossa *view*. Trocaremos a referência para `ListagemView`:

```
public App()
{
    InitializeComponent();

    MainPage = new ListagemView();
}
```

A partir disto, podemos rodar novamente a aplicação, o que ocorre perfeitamente. Agora criaremos uma nova *view* que representará a segunda página da nossa aplicação, que conterá detalhes do modelo do veículo disponibilizado para o *Test Drive*.

Por conta disto, chamaremos esta nova página de "DetalheView". Clicaremos com o lado direito em `Views`, no menu lateral do programa, e em "*Add* > *New Item...*". Na nova janela, selecionaremos "Forms Xaml Page", apertando "*Add*" em seguida.

O arquivo `DetalheView.xaml` foi criado com `ContentPage` em formato *default* (padrão), com um `Label` e nenhum outro controle. Para permitir que o usuário possa clicar na listagem que já existe, navegando a nova página de detalhes, teremos que modificar o evento de `ItemTapped` que criamos na página de listagem.

Então, em `ListagemView.xaml.cs`, removeremos o `DisplayAlert`, substituindo-o por um objeto de navegação do Xamarin chamado `Navigation`, com uma série de comandos ou métodos que podemos utilizar para a navegação, o `PushAsync`, o qual pegará uma nova página, empilhando-a para cima conforme a navegação.

Aqueles que estão acostumados a usar *smartphones* sabem que é possível seguir um fluxo de navegação que permite que se volte a uma página visitada anteriormente. No Xamarin, temos *Navigation Stack* (ou "pilha de navegação") que, conforme a navegação, permite "desempilhar" estas páginas para voltarmos às anteriores.

É exatamente isto que faremos com o comando `PushAsync`, método assíncrono que funcionará quando a aplicação estiver disponível para tal, retornando o controle para ela imediatamente a partir da execução deste comando, sem que ela seja bloqueada.

Dentro deste método, passaremos como parâmetro a nova página, aquela para a qual precisaremos navegar.

```
Navigation.PushAsync(new DetalheView());
```

Vamos rodar novamente a aplicação e ver o que acontece. Assim que a listagem é mostrada, clicaremos no veículo "Fiesta 2.0", para confirmar o direcionamento à tela seguinte. Deu erro! A mensagem nos diz para que devemos usar um `NavigationPage`, uma estratégia de navegação do Xamarin Forms nos permite utilizar o "empilhamento" de páginas na navegação.

Para que isto funcione, modificaremos o nosso arquivo `App.xaml.cs`, pois é ele que inicia a navegação, sendo aquele que chama a primeira página da app, a `ListagemView`.

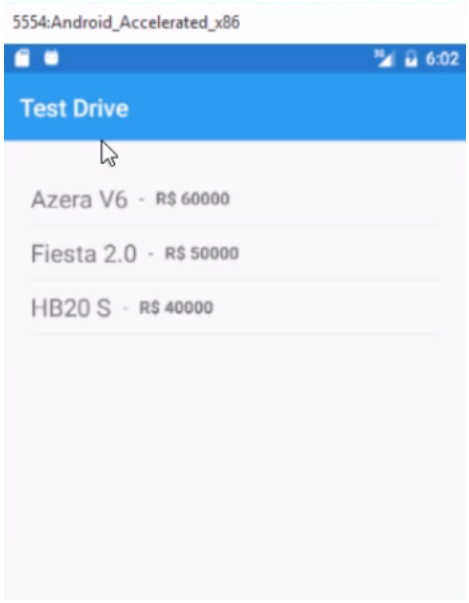
```
public App()
{
    InitializeComponent();

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

Desta forma, veremos se a navegação funciona ao rodarmos novamente a app mais uma vez. No topo do layout, há uma faixa azul adicionada recentemente, indicando que podemos colocar um título ali - uma vez que teremos várias páginas na aplicação. Precisaremos identificá-las corretamente para não prejudicarmos a navegação do usuário.

Abriremos `ListagemView.xaml` e iremos ao `ContentPage`, componente raiz da página que ainda não possui nenhum título associado, nenhuma propriedade `Title` definida. Daremos o título "Test Drive" à página inicial.

```
<ContentPage //...
    Title="Test Drive"
    //...>
</ContentPage>
```

Rodando a aplicação, teremos o seguinte layout:



Vamos testar a navegação clicando em algum dos itens. Somos direcionados a outra página que, apesar de não exibir nenhum título, sabemos que é a `DetalheView`, nossa primeira página, pois assim configuramos a navegação.