11

Revisão do curso

Transcrição

Estamos chegando ao fim da primeira parte do curso de Xamarin com Visual Studio, e faremos uma revisão dos conceitos vistos e aprendidos.

Partindo do requisito da empresa **Aluracar**, de criação de uma aplicação multiplataforma de agendamento de *Test Drive* de veículos, desenvolvemos uma página com a listagem dos veículos disponíveis. Após a seleção de algum deles, o cliente é redirecionado à página seguinte, com detalhes destes veículos e opções de acessórios que podem ser inclusos no serviço.

A tela seguinte consiste no agendamento propriamente dito, com cadastro de dados do usuário e definição de data e hora do agendamento.

Voltando à tela de listagem de veículos, mexemos em alguns componentes de controles do Xamarin Forms, sendo o `StackLayout` um deles. Ele permite que seu conteúdo seja organizado por meio de seu empilhamento (horizontal ou vertical).

Em seguida, vimos outro controle para organizar a tela, o `Grid`, que possibilita posicionar os controles usando colunas e linhas. Tentamos utilizá-lo na tela de listagem, porém ele não é adequado para tal.

Optamos então por outro componente, o `ListView`, que como o próprio nome indica, serve para visualizar uma lista. Por meio dele, fizemos o `Binding`, a amarração do componente visual do Xamarin Forms com os dados que se encontram por trás da aplicação, e que fornecem as informações necessárias para a listagem ser exibida na tela.

Aprendemos a utilizar o `Label`, que pode ser usado de maneira simples, como quando fazemos `Binding` da propriedade `Text`, ou mais complexa, no caso de utilizarmos outro componente chamado `Span`. Assim como na *web*, com o controle HTML `span`, ele é capaz de dividir o `Label` em várias partes.

O usuário pode visualizar as mensagens de alerta, erro e sucesso, a partir do componente visual do Xamarin Forms chamado `DisplayAlert`. No entanto, não queremos ficar interrompendo o uso e fluxo da navegação, então o ideal é que estes alertas só sejam utilizados em casos importantes, já que ele bloqueia a tela, impedindo o usuário de realizar qualquer outra ação.

Também aprendemos sobre o componente `TableView`, usado para a listagem das opções de acessórios à disposição. Com ele, conseguimos criar entradas de dados heterogêneos, que podem ser valores booleanos, com "verdadeiro" ou "falso", como no caso destas chaves que são ativadas ou desativadas.

É possível utilizar o `TableView` para acrescentar controles de entrada de dados como login e senha, por exemplo. Também se pode misturar estas configurações, ou fazer um controle *slider*, uma barra de progressão, por exemplo, uma barra de volume.

Vimos também sobre o `SwitchCell`, a célula de entrada de dados para ligar ou desligar alguma configuração, por trás do qual existe um `Binding` para algum valor, alguma propriedade que se encontra na nossa classe, no *code behind*.

Tal propriedade é do tipo booleano, cujo "verdadeiro" ou "falso" mapeamos para o `SwitchCell`. Em um *smartphone*, por exemplo, o `SwitchCell` possui a função de uma caixa de verficação (*checkbox*). Assim, vimos como criar uma tela de configurações, de acessórios que podem ser ativados ou desativados.

Indo à tela seguinte, de agendamento do *Test Drive*, pegamos o `TableView` para dividir em seções diferentes (usando-se a classe `TableSection`) para melhor organização e estruturação de layout. Aprendemos a trabalhar com entrada de dados por meio de `EntryCell` , ou célula de entrada de dados, como indica o nome.

O preenchimento de dados via teclado do dispositivo é realizado pelo `EntryCell` em um `TableView` . O `EntryCell` varia, como neste caso em que temos preenchimentos de nome (pelo teclado alfanumérico), telefone e e-mail. Ou seja, os teclados podem variar de acordo com a necessidade do usuário.

O `EntryCell` pode ser definido com teclado de tipo *URL*, que fornece uma tecla com o texto `.com` para facilitar o preenchimento de endereço de internet. Caso se queira digitar o número de telefone, por exemplo, existe o teclado específico para isto, e o mesmo ocorre para o campo de e-mail.

> Pode parecer besteira, mas atentar a estes detalhes melhora em muito a usabilidade do nosso aplicativo.

Mais adiante, aprendemos a selecionar data e hora utilizando-se `DatePicker` para a data, conforme as configurações do dispositivo em uso; é muito simples selecionar uma data, coletando-a. Para fazer o mesmo em relação à hora, há o `TimePicker` .

Vamos ficando por aqui. Espero que tenham gostado da primeira parte do curso, na parte dois (https://cursos.alura.com.br/course/xamarin-aplicativos-mobile-com-visual-studio-parte-2), veremos como utilizar **MVVM**, padrão de projeto que ajudará no desacoplamento da nossa aplicação. Nós aprenderemos a fazer **requisição HTTP Get de POST** para o agendamento do *Test Drive*, e também mostraremos como trocar mensagens entre componentes distintos do Xamarin Forms.

Façam os exercícios, pratiquem bastante, boas aulas, e até a próxima!