ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

PROGRAMAÇÃO DE APLICATIVOS MOBILE – II – 2024/1

Luiz Fernando Souza

Command – Continuação

1. Crie o método **CountCharacters** na classe **UsuarioViewModel** conforme abaixo

```
public async Task CountCharacters()
{
    string nameLenght =
        string.Format("Seu nome tem {0} Letras", name.Length);

    await Application.Current
        .MainPage.DisplayAlert("Informação", nameLenght, "Ok");
}
```

2. Declare um *ICommand* chamado **CountCommand**

```
public ICommand CountCommand { get; }
```

3. Faça a vinculação entre o comando e o método criado:

```
public UsuarioViewModel()
{
    ShowMessageCommand = new Command(ShowMessage);
    CountCommand = new Command(async() => await CountCharacters());
}
```

4. Insira um Button na view UsuarioView.xaml, realizando a referência ao Command da classe ViewModel

```
<Button Command="{Binding CountCommand}" Text="Contar Caracteres" />
```

Luiz Fernando Souza

Confirmação do usuário

5. Crie o método abaixo para exibir um alert ao usuário que a resposta sendo positiva, limpará os campos da tela. Faça o using de System.Threading.Tasks.

```
public async Task CleanConfirmation()
{
    if (await Application.Current.MainPage
        .DisplayAlert("Confirmação", "Confirma limpeza dos dados?", "Yes", "No"))
    {
        Name = string.Empty;
        DisplayMessage = string.Empty;
        OnPropertyChanged(Name);
        OnPropertyChanged(DisplayMessage);

        await Application.Current.MainPage
                .DisplayAlert("Informação", "Limpeza realizada com sucesso", "Ok");
    }
}
```

6. Declare um *ICommand* conforme a seguir:

```
public ICommand CleanCommand { get; }
```

7. Vincule o comando a uns dos métodos criados anteriormente no construtor da *ViewModel*

```
public UsuarioViewModel()
{
    ShowMessageCommand = new Command(ShowMessage);
    CountCommand = new Command(async() => await CountCharacters());
    CleanCommand = new Command(async () => await CleanConfirmation());
}
```

8. Arraste um Button para a *View* referenciando o método *CleanCommand* na propriedade *Command*

```
<Button Command="{Binding CleanCommand}"  Text="Limpar Campos" />
```

Opções através dos Commands

9. Crie o método **ShowOptions** conforme abaixo

```
public async Task ShowOptions()
{
    string result = await Application.Current.MainPage
        .DisplayActionSheet("Selecione uma opção: ", "",
        "Cancelar", "Limpar", "Contar Caracteres", "Exibir Saudação");

    if (result != null)
    {
        if(result.Equals("Limpar"))
            await CleanConfirmation();
        if (result.Equals("Contar Caracteres"))
            await CountCharacters();
        if (result.Equals("Exibir Saudação"))
            ShowMessage();
    }
}
```

10. Declare o *ICommand* **OptionCommand**

```
public ICommand OptionCommand { get; }
```

11. Vincule o Comando ao método

```
public UsuarioViewModel()
{
    ShowMessageCommand = new Command(ShowMessage);
    CountCommand = new Command(async() => await CountCharacters());
    CleanCommand = new Command(async () => await CleanConfirmation());
    OptionCommand = new Command(async () => await ShowOptions());
}
```

12. Arraste um Button para a *View* referenciando o método ***OptionCommand*** na propriedade *Command*

```
<Button Command="{Binding OptionCommand}" Text="Exibir Opções"  />
```

- Execute o App para testar a programação

13. Na parte de código da *View*, faremos uma modificação no construtor para permitir que a *ViewModel* possa ser acessada em outras partes do código. Para isso iremos declarar de forma global a *ViewModel*, instanciaremos ela dentro do construtor e atribuiremos ao contexto da *View*.

```
private UsuarioViewModel viewModel;
0 references
public UsuarioView()
{
    InitializeComponent();
    viewModel = new UsuarioViewModel();
    BindingContext = viewModel;
}
```

Com isso fechamos o entendimento do ciclo de vida da aplicação e o uso inicial do MVVM. A partir das próximas aulas aprofundaremos esses conhecimentos na construção de aplicações mais completas.