Luiz Fernando Souza

Aula 14 - Galeria e câmera do App para salvamento de imagens na API

1. Abra a classe Services/Usuarios/**UsuarioService** e adicione um método para atualizar a foto do usuário na API

```
public async Task<int> PutFotoUsuarioAsync(Usuario u)
        {
            string urlComplementar = "/AtualizarFoto";
            var result = await _request.PutAsync(apiUrlBase + urlComplementar, u, _token);
            return result;
            }
```

- Insira também o método responsável por fazer a consulta de um usuário.

```
public async Task<Usuario> GetUsuarioAsync(int usuarioId)
        {
            string urlComplementar = string.Format("/{0}", usuarioId);
            var usuario = await
                _request.GetAsync<Models.Usuario>(apiUrlBase + urlComplementar, _token);
            return usuario;
               }
```

2. Crie uma classe dentro da pasta ViewModels/Usuarios chamada **ImagemUsuarioViewModel.cs**. Herdando de BaseViewModel. Será necessário o using para AppRpgEtec.Services.Usuarios

```
3 references
public class ImagemUsuarioViewModel : BaseViewModel
{
    private UsuarioService uService;
    1 reference
    public ImagemUsuarioViewModel()
    {
        string token = Preferences.Get("UsuarioToken", string.Empty);
        uService = new UsuarioService(token);
    }

}
```

3. Adicione o atributo/propriedade *fonteImagem* para armazenar a imagem para exibir na view e *Foto* para atribuir ao futuro objeto do tipo Usuário.

```
private ImageSource fonteImagem;
1 reference
public ImageSource FonteImagem
{
    get { return fonteImagem; }
    set
    {
        fonteImagem = value;
        OnPropertyChanged();
    }
}
```

```
private byte[] foto; //CTRL + R,E
1 reference
public byte[] Foto
{
    get => foto;
    set
    {
        foto = value;
        OnPropertyChanged();
    }
}
```

4. Clique com o botão direito na raiz da solution, selecione "*Manage Nuget packeges for Solution*" e o procure em "*browse*" por "Xam.Plugin.Media" by James Montemagno versão 5.0.1. e instale este pacote no projeto.

5. Crie o método para fotografar com a estrutura do try/catch. Use o atalho try + TAB + TAB

```
public async void Fotografar()
{
    try
    {
        //Codificação aqui
    }
    catch (Exception ex)
    {
        await Application.Current.MainPage
            .DisplayAlert("Ops", ex.Message + " Detalhes: " + ex.InnerException, "Ok");
    }
}
```

6. Faça a programação dentro do bloco try do método criado no item anterior

```
await CrossMedia.Current.Initialize();

if (!CrossMedia.Current.IsCameraAvailable ||
!CrossMedia.Current.IsTakePhotoSupported)
{
    await Application.Current.MainPage.DisplayAlert("Sem Câmera", "A câmera não está disponível.", "OK");
    await Task.FromResult(false); //using System.Threading.Tasks;
}

string fileName = String.Format("{0:ddMMyyy_HHmmss}", DateTime.Now) + ".jpg";

var file = await CrossMedia.Current.TakePhotoAsync
(new Plugin.Media.Abstractions.StoreCameraMediaOptions
{
    Directory = "Fotos",
    PhotoSize = Plugin.Media.Abstractions.PhotoSize.Small,
    Name = fileName
});

if (file == null)
    await Task.FromResult(false);

MemoryStream ms = null;
using (ms = new MemoryStream())
{
    var stream = file.GetStream();
    stream.CopyTo(ms);
}
FonteImagem = ImageSource.FromStream(() => file.GetStream());
Foto = ms.ToArray();
```

7. Crie conforme abaixo o método que vai ler as propriedades da viewModel, atribuir ao objeto do tipo Usuario para que a classe de serviço envie para a API as informações.

```
public async void SalvarImagem()
{
    try
    {
        Usuario u = new Usuario();
        u.Foto = foto;
        u.Id = Preferences.Get("UsuarioId", 0);

        if (await uService.PutFotoUsuarioAsync(u) != 0)
        {
            await Application.Current.MainPage.DisplayAlert("Mensagem", "Dados salvos com sucesso!", "Ok");
            await App.Current.MainPage.Navigation.PopAsync();
        }
        else { throw new Exception("Erro ao tentar atualizar imagem"); }
    }
    catch (Exception ex)
    {
        await Application.Current.MainPage
            .DisplayAlert("Ops", ex.Message + " Detalhes: " + ex.InnerException, "Ok");
    }
}
```

8. Declare os ICommands (1) e inicialize-os no construtor (2), vinculando os mesmos aos métodos criados anteriormente. Será necessário o using de System.Windows.Input.

```
public ImagemUsuarioViewModel()
{
    string token = Preferences.Get("UsuarioToken", string.Empty);
    uService = new UsuarioService(token);

    AbrirGaleriaCommand = new Command(AbrirGaleria);
    SalvarImagemCommand = new Command(SalvarImagem);
    FotografarCommand = new Command(Fotografar);
}
```

2

```
1 reference
public ICommand AbrirGaleriaCommand { get; }
1 reference
public ICommand SalvarImagemCommand { get; }
1 reference
public ICommand FotografarCommand { get; }
```

1

9. Na pasta Views/Usuarios, crie uma *Content Page* (.Net Maui) chamada **ImagemUsuarioView.xaml**. Insira as tags abaixo dentro da tag ContentPage.Content.

```
<ScrollView>
      <VerticalStackLayout>

       </VerticalStackLayout>
</ScrollView>
```

10. Insira o design abaixo dentro da tag VerticalStackLayout que foi inserido na etapa anterior

```
<ScrollView>
                    <VerticalStackLayout HorizontalOptions="FillAndExpand"
VerticalOptions="Start">
                        <Grid HorizontalOptions="Fill" Margin="5, 5, 0, 0" Padding="10, 10, 0, 0">
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="33*"/>
                                <ColumnDefinition Width="34*"/>
                                <ColumnDefinition Width="33*"/>
                            </Grid.ColumnDefinitions>
                            <Grid.RowDefinitions>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>
                            <Button Text="Câmera" HorizontalOptions="Center" Grid.Column="0"
Grid.Row="0" Command="{Binding FotografarCommand}"/>
                            <Button Text="Álbum" HorizontalOptions="Center" Grid.Column="1"
Grid.Row="0" Command="{Binding AbrirGaleriaCommand}"/>
                            <Button Text="Gravar" HorizontalOptions="Center" Grid.Column="2"
Grid.Row="0" Command="{Binding SalvarImagemCommand}" />
                        </Grid>
                    </VerticalStackLayout>
                </ScrollView>
                    <Image Source="{Binding FonteImagem}" Margin="10"/>
```

11. Na parte de Código da view iremos fazer a vinculação entre a *ViewModel* e a *View*. Necessário o using de AppRpgEtec.ViewModels.Usuarios

```
ImagemUsuarioViewModel viewModel;
0 references
public ImagemUsuarioView()
{
    InitializeComponent();

    viewModel = new ImagemUsuarioViewModel();
    Title = "Imagem do Usuário";
    BindingContext = viewModel;
}
```

12. Abra a view **AppShell.xaml** e insira um item de menu para chamar a view ImagemUsuarioView

```
<Tab Title="Info" Route="info" Icon="info.svg">
    <ShellContent Title="Usuário" Route="usuario"
                  ContentTemplate="{DataTemplate viewsUsuarios:ImagemUsuarioView}" />

    <ShellContent Title="Sobre" Route="sobre"
                  ContentTemplate="{DataTemplate local:AboutView}" />

    <ShellContent Title="Descrição" Route="desc"
                  ContentTemplate="{DataTemplate local:DescriptionView}" />
</Tab>
```

13. Abra o arquivo **AndroidManifest.xml** que fica na pasta *Plataforms/Android* (pasta Properties) e adicione o código abaixo dentro da tag application, que permitirá o acionamento da câmera do dispositivo.

```
<provider android:name="androidx.core.content.FileProvider"
                          android:authorities="${applicationId}.fileprovider"
                          android:exported="false"
                          android:grantUriPermissions="true">
                    <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
                                       android:resource="@xml/file_paths"></meta-data>
    </provider>
```

14. Ainda no AndroidManifest, adicione as permissões abaixo de câmera, leitura e escrita de arquivos externos

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
```

15. Abra a classe **Plataforms/Android/Resources/MainActivity.cs** e adicione uma referência para inicialização da câmera. O pacote *CrossMedia* pertence ao Xamarin, por isso a referência a ele.

```
public class MainActivity : MauiAppCompatActivity
{
    0 references
    protected override void OnCreate(Bundle savedInstanceState)
    {
        Xamarin.Essentials.Platform.Init(this, savedInstanceState);
        base.OnCreate(savedInstanceState);
    }
}
```

16. Clique com o direito na pasta *Resources*, crie uma pasta chamada **xml**, clique com o direito → Add new Item para criar um arquivo do tipo xml (digite xml no campo Search para filtrar o template) com o nome **file_paths.xml**, remova as linhas que existia neste arquivo e insira a codificação abaixo. Esta ação vai definir o nome padrão das pastas de imagens e vídeo do aplicativo. Clique em cima do arquivo e em F4 (Propriedades) alterando a propriedade BuildAction para AndroidResource

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-files-path name="my_images" path="Pictures" />
    <external-files-path name="my_movies" path="Movies" />
</paths>
```

- Execute o aplicativo para testar o funcionamento da câmera e o salvamento da imagem na API.

17. Volte à viewModel **ImagemUsuarioViewModel.cs** e programe um método para selecionar uma foto do álbum. Primeiramente deixe montada a estrutura do try/catch

```
public async void AbrirGaleria()
{
    try
    {
        //Codificação aqui
    }
    catch (Exception ex)
    {
        await Application.Current.MainPage
            .DisplayAlert("Ops", ex.Message + " Detalhes: " + ex.InnerException, "Ok");
    }
}
```

18. Realize a codificação dentro do bloco try do método criado no item anterior. Execute o app para selecionar uma imagem da galeria e salvar.

```
await CrossMedia.Current.Initialize();
if (!CrossMedia.Current.IsPickPhotoSupported)
{
    await Application.Current.MainPage.DisplayAlert("Galeria não suportada",
        "Não existe permissão para acessar a galeria.", "OK");
    return;
}
var file = await CrossMedia.Current.PickPhotoAsync();
if (file == null)
    return;

MemoryStream ms = null;
using (ms = new MemoryStream())
{
    var stream = file.GetStream();
    stream.CopyTo(ms);
}
FonteImagem = ImageSource.FromStream(() => file.GetStream());
Foto = ms.ToArray();
return;
```