ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

PROGRAMAÇÃO DE APLICATIVOS MOBILE – II – 2024/1

Luiz Fernando Souza

Pesquisa de Personagens para as Disputas – Controle Search Bar

1. Crie a classe Disputa dentro da pasta Models com as propriedades a seguir

```
public int Id { get; set; }
0 references
public DateTime? DataDisputa { get; set; }
0 references
public int AtacanteId { get; set; }
0 references
public int OponenteId { get; set; }
0 references
public string Narracao { get; set; }
0 references
public int HabilidadeId { get; set; }
0 references
public List<int> ListaIdPersonagens { get; set; } = new List<int>();
0 references
public List<string> Resultados { get; set; } = new List<string>();
```

2. Abra a classe **PersonagemService** e adicione um método para realizar pesquisar pelo nome aproximado, conforme abaixo

```
public async Task<ObservableCollection<Personagem>> GetByNomeAproximadoAsync(string busca)
{
    string urlComplementar = $"/GetByNomeAproximado/{busca}";

    ObservableCollection<Models.Personagem> listaPersonagens = await
    _request.GetAsync<ObservableCollection<Models.Personagem>>(_apiUrlBase + urlComplementar, _token);

    return listaPersonagens;
}
```

3. Crie uma pasta chamada Disputas dentro da pasta ViewModels e dentro da pasta Disputas crie uma classe chamada DisputaViewModel, deixando-a pública e fazendo herança com a classe BaseViewModel

4. Declare os objetos que serão utilizados, e os dados de inicialização no construtor. Usings para AppRpgEtec.Model, AppRpgEtec.Services.Personagens e System.Collections.ObjectModel

```
private PersonagemService pService;
1 reference
public ObservableCollection<Personagem> PersonagensEncontrados { get; set; }
1 reference
public Personagem Atacante { get; set; }
1 reference
public Personagem Oponente { get; set; }
0 references
public DisputaViewModel()
{
    string token = Preferences.Get("UsuarioToken", string.Empty);
    pService = new PersonagemService(token);

    Atacante = new Personagem();
    Oponente = new Personagem();

    PersonagensEncontrados = new ObservableCollection<Personagem>();
}
```

5. Programe o método que executará a pesquisa.

```
public async Task PesquisarPersonagens(string textoPesquisaPersonagem)
{
    try
    {
        PersonagensEncontrados = await pService.GetByNomeAproximadoAsync(textoPesquisaPersonagem);
        OnPropertyChanged(nameof(PersonagensEncontrados));
    }
    catch (Exception ex)
    {

        await Application.Current.MainPage
            .DisplayAlert("Ops", ex.Message + " Detalhes: " + ex.InnerException, "Ok");
    }
}
```

6. Declare um ICommand e o vincule ao método de pesquisa. Faça o using para System.Windows.Input

```
public DisputaViewModel()
{
    string token = Preferences.Get("UsuarioToken", string.Empty);
    pService = new PersonagemService(token);

    Atacante = new Personagem();
    Oponente = new Personagem();

    PersonagensEncontrados = new ObservableCollection<Personagem>();

2   PesquisarPersonagensCommand =
        new Command<string>(async (string pesquisa) => { await PesquisarPersonagens(pesquisa); });
}
1 reference
1 public ICommand PesquisarPersonagensCommand { get; set; }
```

7. Programe duas propriedades que retornarão o nome do atacante e do oponente

```
public string DescricaoPersonagemAtacante
{
    get => Atacante.Nome;
}
0 references
public string DescricaoPersonagemOponente
{
    get => Oponente.Nome;
}
```

8. Programe o método que receberá o personagem selecionado

```
public async void SelecionarPersonagem(Personagem p)
{
    try
    {
        string tipoCombatente = await Application.Current.MainPage
            .DisplayActionSheet("Atacante ou Oponente?", "Cancelar", "", "Atacante", "Oponente");


        if(tipoCombatente == "Atacante")
        {
            Atacante = p;
            OnPropertyChanged(nameof(DescricaoPersonagemAtacante));
        }
        else if (tipoCombatente == "Oponente")
        {
            Oponente = p;
            OnPropertyChanged(nameof(DescricaoPersonagemOponente));
        }

    }
    catch (Exception ex)
    {

        await Application.Current.MainPage
            .DisplayAlert("Ops", ex.Message + " Detalhes: " + ex.InnerException, "Ok");
    }
}
```

9. Crie o atributo/propriedade que armazenará o personagem selecionado para executar o método criado na etapa anterior

```
private Personagem personagemSelecionado;
0 references
public Personagem PersonagemSelecionado
{
    set
    {
        if(value != null)
        {
            personagemSelecionado = value;
            SelecionarPersonagem(personagemSelecionado);
            OnPropertyChanged();
            PersonagensEncontrados.Clear();
        }
    }
}
```

10. Abra a content page ListagemView na pasta Views/Disputas e na parte de código declare um objeto ViewModel de Disputa e inicialize-o no construtor, vinculando-o à view.

```
public partial class ListagemView : ContentPage
{
    DisputaViewModel viewModel;
    0 references
    public ListagemView()
    {
        InitializeComponent();

        viewModel = new DisputaViewModel();
        BindingContext = viewModel;
    }
}
```

11. Insira o layout no design da View dentro do VerticalStackLayout que já existia.

```
<SearchBar x:Name="searchBar" HeightRequest="25" Placeholder="Digite o nome do Personagem..."
                      TextColor="Black" SearchCommand="{Binding PesquisarPersonagensCommand}"
                      SearchCommandParameter="{Binding Source={x:Reference searchBar}, Path=Text}"/>

            <ListView x:Name="lvPersonagens" HasUnevenRows="True" ItemsSource="{Binding
PersonagensEncontrados}"
                      SelectedItem="{Binding PersonagemSelecionado}"
                      Margin="10, 5, 0, 0">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <StackLayout Orientation="Vertical">
                                <Label Text="{Binding Id}" TextColor="Blue" IsVisible="False"/>
                                <Label Text="{Binding Nome}" TextColor="Blue" FontSize="18"/>
                                <Label Text="{Binding Classe}" TextColor="Blue" FontSize="14"/>
                            </StackLayout>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>

            <Grid HorizontalOptions="Fill" Margin="5, 5, 0, 0" Padding="10, 10, 0, 0">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                </Grid.RowDefinitions>
                <Label Text="Atacante: " Grid.Column="0" Grid.Row="0" FontAttributes="Bold" />
                <Label Text="{Binding DescricaoPersonagemAtacante}" Grid.Column="1" Grid.Row="0"
TextColor="Blue" />

                <Label Text="Oponente: " Grid.Column="0" Grid.Row="1" FontAttributes="Bold" />
                <Label Text="{Binding DescricaoPersonagemOponente}" Grid.Column="1" Grid.Row="1"
TextColor="Blue" />
                </Grid>
```

- Execute o aplicativo para certificar que o campo de pesquisa está sendo exibido e ao digitar o nome aproximado e clicar na lupa ou no botão de checagem do teclado os personagens vão aparecer. Ao clicar no personagem escolhido, você deverá informar se é um Atacante ou Oponente e os dados do personagem devem aparecer na View.

Busca com modificação imediata após a digitação

12. Insira um campo *Entry* abaixo da *SearchBar*

```
<SearchBar x:Name="searchBar" HeightRequest="25" Placeholder="Digite o nome do Personagem..."
           TextColor="Black" SearchCommand="{Binding PesquisarPersonagensCommand}"
           SearchCommandParameter="{Binding Source={x:Reference searchBar}, Path=Text}"/>

<Entry Placeholder="Digite o nome do Personagem..." Text="{Binding TextoBuscaDigitado}">
</Entry>

<ListView x:Name="lvPersonagens" HasUnevenRows="True" ItemsSource="{Binding PersonagensEncontrados}"
```

13. Retorne para a classe *DisputaViewModel* e crie um atributo propriedade com as características abaixo

```
private string textoBuscaDigitado = string.Empty;
0 references
public string TextoBuscaDigitado
{
    get { return textoBuscaDigitado; }
    set
    {
        //Verifica se não é nulo, se não é vazio e se o tamanho do texto é maior que zero.
        if ((value != null && !string.IsNullOrEmpty(value) && value.Length > 0))
        {
            textoBuscaDigitado = value;
            _ = PesquisarPersonagens(textoBuscaDigitado);
        }
        else
        {
            //Limpa o list view que exibe o resultado da pesquisa
            PersonagensEncontrados.Clear();
        }
    }
}
```

- Agora você duas maneiras de realizar uma busca, use a que achar melhor.
- A próxima etapa será programar o ataque com armas ou habilidades.