Ministério da Educação

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO **CAMPUS CUBATÃO**

Matéria: PDMI – ADS 671 – Professor: Wellington Tuler Moraes

Nome:_____ Data: __/__/_____

Nome:_____ Data: __/__/_____

Trabalho Prático 02

Utilizando Visual Studio XAMARIN – Projeto Cross-Plataform, criar uma aplicação com as seguintes características:

1 - Página principal com um gerenciador de Layout *StackLayout* apresentando as seguintes opções de menu:

Figura 1: Menu principal da aplicação Mobile.

Cada item em cinza é um botão que aciona uma tela específica com as seguintes características (nem todos os botões serão implementados):

Style Resources:

A – Estilo Padrão:

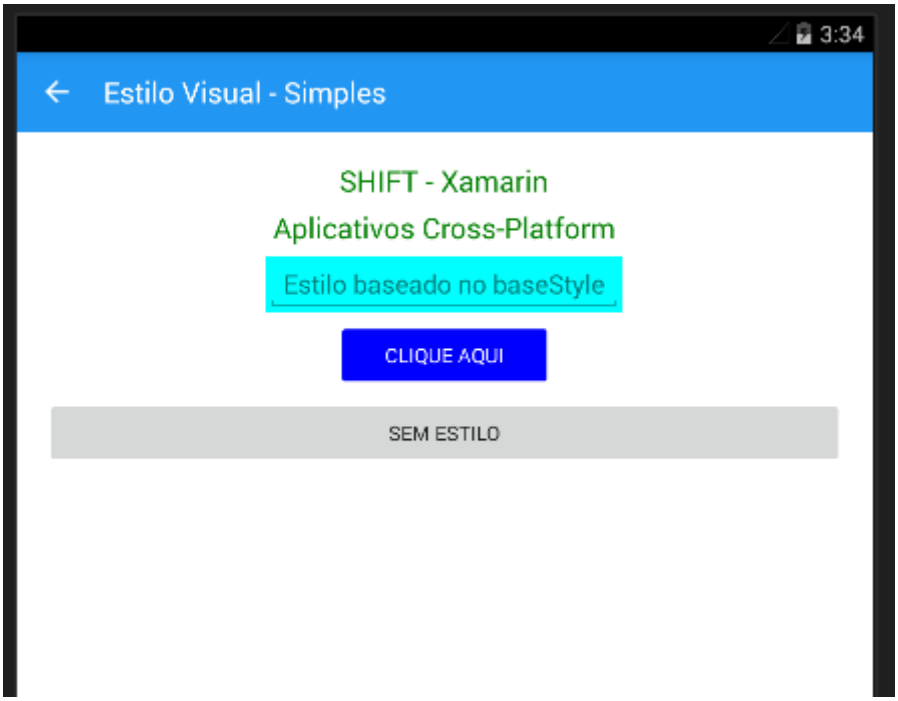

Figura 2: Primeira tela da aplicação

PS: Criar um layout de tela livre, com diversos estilos, Botão voltar deve permitir a aplicação voltar a tela menu inicial.

B - Estilo Dinâmico:

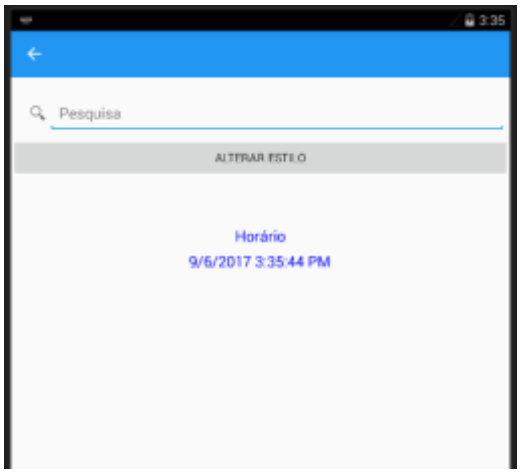

Figura 3: Aplicação iniciada no segundo botão usando MVVM para relógio dinâmico.

PS: Deve conter um relógio que se atualiza dinamicamente (Pode-se empregar MVVM).

C – Estilo Event Triggers

Elaborar o seguinte código para ver o resultado na página (TriggersView), observe o nome do projeto e namespace poderá mudar de acordo com o nome que você escolher.

Necessário criar a seguinte classe antes de usar o XAML:

```
namespace XF.Recursos.Global
{
    0 referências
    public class EntryNumerico : TriggerAction<Entry>
    {
        0 referências
        protected override void Invoke(Entry sender)
        {
            int parsed;
            bool validar = int.TryParse(sender.Text, out parsed);
            if (!validar)
            {
                sender.TextColor = Color.Red;
            }
            else
            {
                sender.TextColor = Color.Blue;
            }
        }
    }
}
```

Figura 4: Código da Trigger EntryNumerico

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:global="clr-namespace:XF.Recursos.Global"
             x:Class="XF.Recursos.Estilo.TriggersView">

    <ContentPage.Resources>
        <ResourceDictionary>
            <Style TargetType="Entry">
                <Style.Triggers>
                    <Trigger TargetType="Entry" Property="IsFocused"
Value="True">
                        <Setter Property="BackgroundColor" Value="Yellow" />
                        <Setter Property="TextColor" Value="Black" />
                    </Trigger>
                </Style.Triggers>
            </Style>
        </ResourceDictionary>
    </ContentPage.Resources>

    <ContentPage.Content>
        <StackLayout Padding="30">
            <Label Text="Property Trigger" Font="15" />
            <Entry Placeholder="Nome" />
            <Entry Placeholder="RM" />
            <Label Text="Data Trigger" Font="15" />
            <Entry x:Name="txtEmail" Placeholder="E-mail" />
            <Button x:Name="btnEnviar" Text="Enviar e-mail">
                <Button.Triggers>
                    <DataTrigger TargetType="Button"
                       Binding="{Binding Source={x:Reference txtEmail},
Path=Text.Length}" Value="0">
                        <Setter Property="IsEnabled" Value="False" />
                    </DataTrigger>
                </Button.Triggers>
            </Button>
            <Label Text="Event Trigger" Font="15" />
            <Entry Placeholder="Idade" Keyboard="Numeric">
                <Entry.Triggers>
                    <EventTrigger Event="TextChanged">
                        <global:EntryNumerico />
                    </EventTrigger>
                </Entry.Triggers>
            </Entry>
        </StackLayout>
    </ContentPage.Content>

</ContentPage>
```

Figura 4.1: Código disponível no terceiro botão.

D – ListView Simples

Figura 5: Implementação do quarto botão com chamada da View Simples.

Passar Parâmetros:

Implementar apenas o botão HOME:

E – Passagem de Parâmetros:

E1 – Criar a seguinte classe:

```
public class Contato
{
    3 referências
    public string Nome { get; set; }

    2 referências
    public int Idade { get; set; }

    2 referências
    public string Profissao { get; set; }

    2 referências
    public string Pais { get; set; }

    0 referências
    public override string ToString()
    {
        return Nome;
    }
}
```

Figura 6: Código da classe Contato.

E2 - Criar um formulário que receba os valores da classe Contato (Nome, idade, profissão e País) em campos para instanciar um objeto, conforme o exemplo a seguir:

```
private async void btnDetalhe_Clicked(object sender, EventArgs e)
{
    var contato = new Contato
    {
        Nome = "Flavio Mariotti",
        Idade = 32,
        Profissao = "Developer",
        Pais = "Brasil"
    };

    var detalhePage = new DetalheView();
    detalhePage.BindingContext = contato;
    await Navigation.PushAsync(detalhePage);
}
```

Figura 7: Código do botão acionado e que passa os valores para criar o objeto.

OBS: Os valores fixos no código exemplo acima são apenas exemplo, devemos passar o conteúdo de cada TextBox para instanciar o objeto.

E3 – Criar um botão que passe os parâmetros para uma nova página que conterá uma lista e apresentar os dados recebidos na tela, conforme exemplo:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="XF.Recursos.PassParameter.DetalheView"
             Title="Detalhes do contato">

    <ContentPage.Content>
        <StackLayout HorizontalOptions="Center" VerticalOptions="Center">
            <StackLayout Orientation="Horizontal">
                <Label Text="Nome:" FontSize="Medium"
HorizontalOptions="FillAndExpand" />
                <Label Text="{Binding Nome}" FontSize="Medium"
FontAttributes="Bold" />
            </StackLayout>
            <StackLayout Orientation="Horizontal">
                <Label Text="Idade:" FontSize="Medium"
HorizontalOptions="FillAndExpand" />
                <Label Text="{Binding Idade}" FontSize="Medium"
FontAttributes="Bold" />
            </StackLayout>
            <StackLayout Orientation="Horizontal">
                <Label Text="Profissão:" FontSize="Medium"
HorizontalOptions="FillAndExpand" />
                <Label Text="{Binding Profissao}" FontSize="Medium"
FontAttributes="Bold" />
            </StackLayout>
            <StackLayout Orientation="Horizontal">
                <Label Text="País:" FontSize="Medium"
HorizontalOptions="FillAndExpand" />
                <Label Text="{Binding Pais}" FontSize="Medium"
FontAttributes="Bold" />
            </StackLayout>
            <Button x:Name="btnVoltar" Text="Home" Clicked="btnVoltar_Clicked"
/>
        </StackLayout>
    </ContentPage.Content>

</ContentPage>
```

Figura 8: Código exemplo para exibição dos valores recebido pela página View.

List View

Nesta sessão implementaremos apenas a chamada do botão “Produto”.

F- Criar formulários para receber os valores digitados pelos usuários através do formulário de produtos e passar as informações para outro formulário, através de objeto instanciado com valores digitados no formulário inicial.

A classe produto, que servirá como base, é apresentada a seguir:

Matéria: PDMI – ADS 671 – Professor: Wellington Tuler Moraes

Nome:_____ Data: __/__/_____

Nome:_____ Data: __/__/_____

```
4 referências
public class Produto
{
    2 referências
    public int Id { get; set; }
    2 referências
    public string Descricao { get; set; }
    1 referência
    public string Categoria { get; set; }
    1 referência
    public int Quantidade { get; set; }
    1 referência
    public decimal Preco { get; set; }
}
```