PAM-I

Programação de Aplicativos Mobile - I

Navegação por abas

Objetivos de aprendizagem

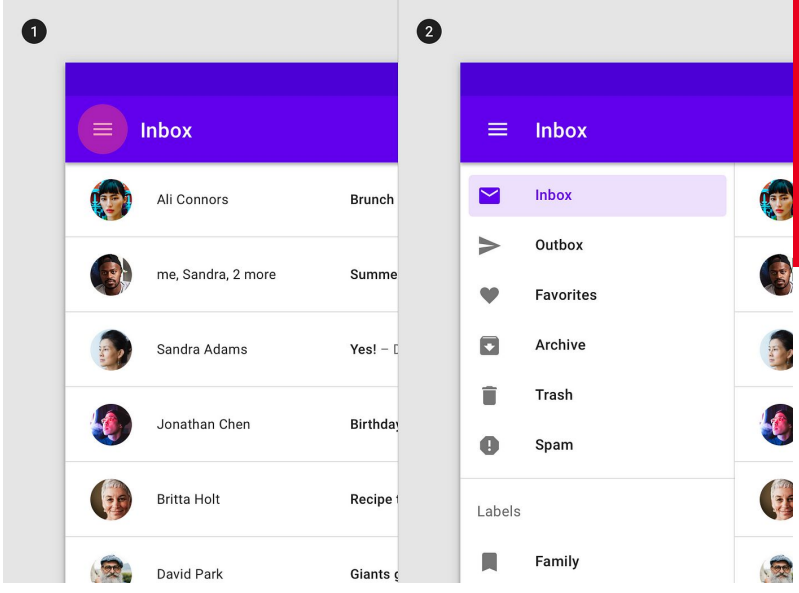
Implementar a navegação de submenu com o Shell do .NET MAUI

Implementar a navegação de guia com o Shell do .NET MAUI

Navegar entre as páginas nas páginas com guias

Implementando um submenu

A navegação por submenu é um tipo de navegação em que uma janela de itens de menu desliza ou voa para fora da tela (por isso o chamamos de flyout). Geralmente, ele é invocado tocando no que é chamado de menu "**hambúrguer**" ou um ícone com três linhas horizontais empilhadas umas sobre as outras.



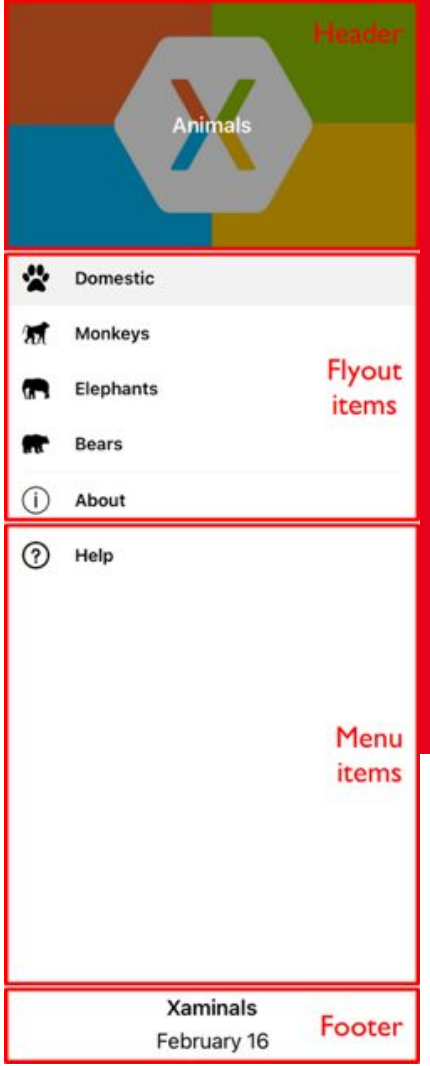
Implementando um submenu

A navegação de submenu exibe um menu que fornece um meio rápido de alternar o contexto em seu aplicativo.

O menu de submenu é composto por várias partes, o `Header`, `FlyoutItems`,`MenuItems` e `Footer`.

A imagem ao lado mostra um exemplo visual das partes do submenu.

Como o menu de submenu nem sempre está visível, ele é melhor utilizado para **alternar o contexto entre partes conceitualmente diferentes do seu aplicativo**.

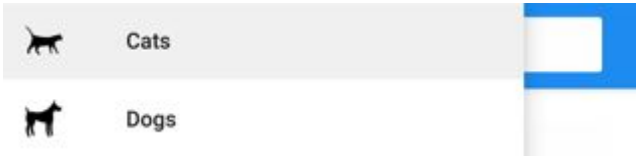


Implementando um submenu

A experiência de navegação fornecida pelo .NET MAUI Shell é baseada em submenus e guias.
O nível superior de navegação em um aplicativo Shell é um submenu ou uma barra de guias inferior, dependendo dos requisitos de navegação do aplicativo.

Um ou mais itens de submenu podem ser adicionados ao submenu e cada item de submenu é representado por um objeto `FlyoutItem`. Cada `FlyoutItem` objeto deve ser um filho do objeto `Shell`. Os itens de submenu aparecem na parte superior do submenu quando um cabeçalho de submenu não está presente.

Implementando um submenu



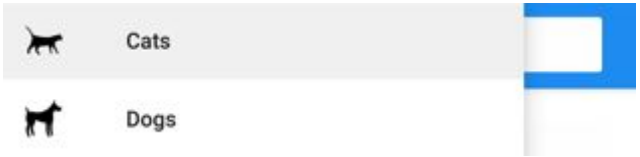

```
<Shell xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
     xmlns:controls="clr-namespace:Xaminals.Controls"
     xmlns:views="clr-namespace:Xaminals.Views"
     x:Class="Xaminals.AppShell">

     <FlyoutItem Title="Cats" Icon="cat.png">
     <Tab>
          <ShellContent ContentTemplate="{DataTemplate views:CatsPage}" />
     </Tab>
     </FlyoutItem>
     <FlyoutItem Title="Dogs" Icon="dog.png">
     <Tab>
          <ShellContent ContentTemplate="{DataTemplate views:DogsPage}" />
     </Tab>
     </FlyoutItem>
</Shell>
```

O mesmo código pode ser escrito de maneira simplificada porque um objeto `Shell` só pode conter objetos `FlyoutItem` ou um objeto `TabBar`, que só podem conter objetos Tab, que por sua vez só podem conter objetos `ShellContent`.

Esses operadores de conversão implícita podem ser usados para remover os objetos `FlyoutItem` e `Tab` do exemplo anterior:

Implementando um submenu




```
<Shell xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
     xmlns:controls="clr-namespace:Xaminals.Controls"
     xmlns:views="clr-namespace:Xaminals.Views"
     x:Class="Xaminals.AppShell">
   <ShellContent Title="Cats"
                 Icon="cat.png"
                 ContentTemplate="{DataTemplate views:CatsPage}" />
   <ShellContent Title="Dogs"
                 Icon="dog.png"
                 ContentTemplate="{DataTemplate views:DogsPage}" />
</Shell>
```

Adicionando items ao submenu

Os itens do menu podem ser opcionalmente adicionados ao submenu, e cada item do menu é representado por um objeto MenuItem. Os itens de menu são semelhantes aos botões em que tocar em um leva a uma ação a ocorrer em vez de uma página a ser exibida.

A posição dos objetos MenuItem no submento depende da ordem de declaração deles na hierarquia visual do Shell. Portanto, todos os objetos MenuItem declarados antes dos objetos `FlyoutItem` aparecerão antes dos objetos `FlyoutItem` no submenu, e todos os objetos `MenuItem` declarados após os objetos `FlyoutItem` aparecerão após os objetos `FlyoutItem` no submenu.

```
<Shell ...>
      ...
      <MenuItem Text="Help"
            IconImageSource="help.png"
            Command="{Binding HelpCommand}"
            CommandParameter="https://learn.microsoft.com/dotnet/maui/fundamentals/shell" />
</Shell>
```

Adicionando cabeçalho e rodapé

O cabeçalho e rodapé do submenu aparecem, opcionalmente, na parte superior e inferior do submenu respectivamente. Sua aparência sendo definida por um objeto que pode ser definido com a propriedade `Shell.FlyoutHeader`: e `Shell.FlyoutFooter`

```
<Shell ...>
  <Shell.FlyoutHeader>
    <Grid>
     <Image Source="header-image.png" />
    </Grid>
  </Shell.FlyoutHeader>
</Shell>
```

```
<Shell ...>
  <Shell.FlyoutFooter>
    <Grid>
      <Image Source="footer-image.png" />
    </Grid>
  </Shell.FlyoutFooter>
</Shell>
```

Mão na massa!
Projeto de Exemplo

Projeto Astronomia

Suponha que você esteja escrevendo um aplicativo para um planetário usando o .NET MAUI.

O objetivo do app é gerar interesse em astronomia apresentando fatos e conceitos astronômicos que sejam relevantes para a vida cotidiana. O aplicativo terá páginas que abrangem o nascer e o pôr-do-sol, as fases da Lua, os corpos astronômicos e uma página Sobre.

Seu trabalho é projetar e implementar um padrão de navegação intuitiva para essas páginas usando guias e entender quando esse padrão de navegação é mais apropriado como o personalizar.

1. Clone o repositório do exercício
2. Abra a solution Astronomy.sln
3. Na janela Solution Explorer, no projeto Astronomy, expanda a pasta Pages. Esta pasta contém as seguintes páginas:

   AboutPageView. Esta página exibe informações sobre o aplicativo.

   MoonPhaseView. Esta página exibe informações específicas sobre as fases da Lua vistas da Terra.

   SunriseView. Esta página exibe os horários do nascer e do pôr do sol para locais na Terra. Os dados são fornecidos pelo serviço web Sunrise Sunset.

4. Rodar o aplicativo.

Análise

De onde aparecem os nomes das fases da lua?

O que são setters?

Como acessar essas diferentes páginas?

1. No Solution Explorer, abra o AppShell.xaml
2. No Editor, procure o `<ShellContent>` e crie um `<FlyoutItem>` ao redor do mesmo. Depois defina a propriedade do Flyout como Fase Lunar.

```
<FlyoutItem Title="Moon Phase">
    <ShellContent
        ContentTemplate="{DataTemplate local:MoonPhaseView}"/>
</FlyoutItem>
```

3. Adicione uma propriedade FlyoutIcon ao Shell para apresentar uma imagem. A imagem padrão são três barras horizontais (O famoso menu Hamburguer) mas pode ser substituída por outra imagem.

```
<Shell
    x:Class="PAM_Astronomy.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:PAM_Astronomy.Views"
    Shell.FlyoutBehavior="Flyout"
    FlyoutIcon="moon.png"
    Title="PAM_Astronomy">
```

4. Agora, adicione mais opções de lista de opções.

Crie um novo `<FlyoutItem>` abaixo do que acabou de fazer e defina o Title como Sunrise. Faça o `ShellContent` apontar para a página `SunriseView`.

```
<FlyoutItem Title="Sunrise">
    <ShellContent
        ContentTemplate="{DataTemplate local:SunriseView}"/>
</FlyoutItem>

<FlyoutItem Title="About">
    <ShellContent
        ContentTemplate="{DataTemplate local:AboutView}"/>
</FlyoutItem>
```

Execute o código e depois faça as seguintes alterações:

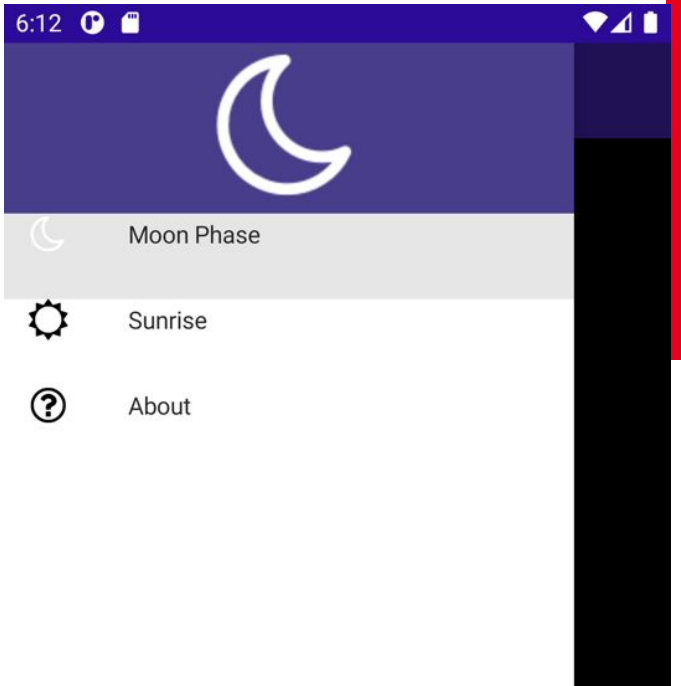
Altere os outros ícones das outras páginas para question.png e sun.png

Adicionando um cabeçalho a lista de opções

1. Adicione ao Shell o flyoutHeader como elemento filho.

Dentro dessa Shell.FlyoutHeader adicione um Grid e uma imagem.

```
<Shell.FlyoutHeader>
    <Grid HeightRequest="100"
BackgroundColor="DarkSlateBlue">
        <Image Source="moon.png" />
    </Grid>
</Shell.FlyoutHeader>
```



Adicionando as tabs

Adicionando um tabbar

1. Na janela do Gerenciador de Soluções, abra a página **AppShell.xaml**.
2. Na página de marcação XAML, exclua tudo que estiver dentro de <Shell>.
3. Crie um <TabBar> e um <Tab> vazio.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Shell
       x:Class="PAM_Astronomy.AppShell"
       xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:local="clr-namespace:PAM_Astronomy.Views"
       Shell.FlyoutBehavior="Flyout"
       FlyoutIcon="moon.png"
       Title="PAM_Astronomy">

       <TabBar>
       <Tab>

       </Tab>
       </TabBar>
</Shell>
```

Adicionando um tabbar

4. Adicione um `ShellContent` ao `Tab`. O conteúdo será o `MoonPhasePage`.

```
<TabBar>
    <Tab>
        <ShellContent ContentTemplate="{DataTemplate local:MoonPhaseView}" />
    </Tab>
</TabBar>
```

5. Dê à guia um título a ser exibido e um ícone usando as propriedades `Title` e `Icon`.

```
<Tab Title="Moon Phase" Icon="moon.png">
```

Adicionando um tabbar

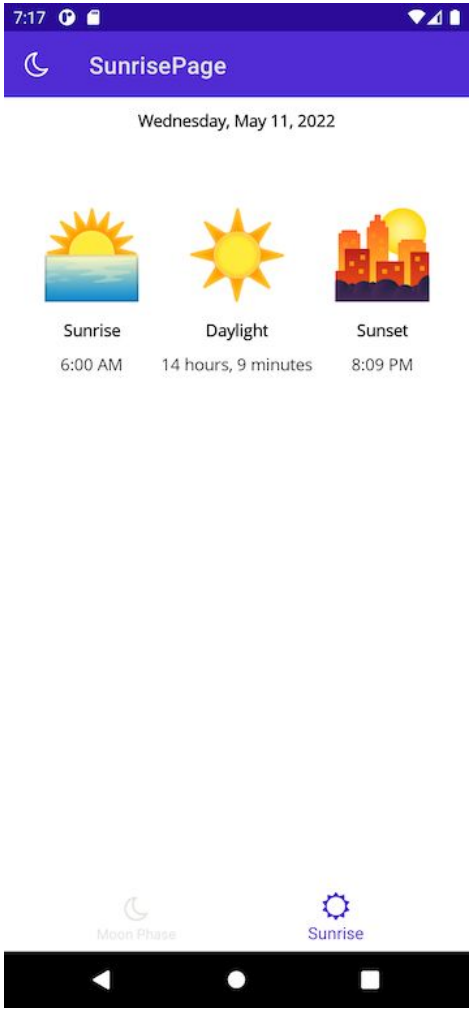
Adicione outro `Tab` para o `SunrisePage`. Defina seu `Title` como igual a **sunrise** e `Icon` como igual a **sun.png**. O XAML concluído terá esta aparência:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Shell
    x:Class="Astronomy.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:AstronomyView"
    FlyoutIcon="moon.png">

    <Shell.FlyoutHeader>
        <Grid HeightRequest="100" BackgroundColor="DarkSlateBlue">
            <Image Source="moon.png"/>
        </Grid>
    </Shell.FlyoutHeader>

    <TabBar>
        <Tab Title="Moon Phase" Icon="moon.png">
            <ShellContent ContentTemplate="{DataTemplate local:MoonPhaseView}" />
        </Tab>
        <Tab Title="Sunrise" Icon="sun.png">
            <ShellContent ContentTemplate="{DataTemplate local:SunriseView}"/>
        </Tab>
    </TabBar>
</Shell>
```

O Aplicativo deve estar com essa aparência



Adicionando o sub-menu novamente

Adicionando o submenu

1. Exclua o `TabBar` e todos os seus filhos que você acabou de criar.
2. No lugar, adicione um `<FlyoutItem>`. Defina a propriedade `Title` como Astronomy e o ícone como moon.png.

```
<FlyoutItem Title="Astronomy" Icon="moon.png">

</FlyoutItem>
```

3. Dentro de `<FlyoutItem>`, adicione um `<ShellContent>` que aponta para o `MoonPhaseView`. Defina sua propriedade `Title` como MoonPhase e a propriedade `Icon` como moon.png.

```
FlyoutItem Title="Astronomy" Icon="moon.png">
      <ShellContent Title="Moon" Icon="moon.png"
                  ContentTemplate="{DataTemplate local:MoonPhaseView}" />
</FlyoutItem>
```

Adicionando o submenu

4. Na mesma `<FlyoutItem>`, adicione outro `<ShellContent>` para apontar para o `SunriseView`. Defina a propriedade `Title` como Sunrise e a propriedade `Icon` como sun.png.

```
<FlyoutItem Title="Astronomy" Icon="moon.png">
    <ShellContent Title="Moon Phase" Icon="moon.png"
        ContentTemplate="{DataTemplate local:MoonPhaseView}"/>

     <ShellContent Title="Daylight" Icon="sun.png"
          ContentTemplate="{DataTemplate local:SunriseView}" />
</FlyoutItem>
```

5. Para criar um item de submenu que aponta para `AboutPage`, adicione um novo `<FlyoutItem>`. Defina sua propriedade `Title` como About e a propriedade `Icon` como question.png.

Adicionando o submenu

6. Dentro desse `<FlyoutItem>`, adicione um `<ShellContent>` que aponta para o `AboutPage`.

```
<FlyoutItem Title="About" Icon="question.png">
<ShellContent
Title="About"
ContentTemplate="{DataTemplate local:AboutPageView}"
Route="AboutPageView" />
</FlyoutItem>
```

7. Execute o aplicativo

Adicionando navegação através de rotas

1. Na janela Gerenciador de Soluções, abra a pasta Páginas. Além dos arquivos MoonPhaseView, SunriseView e AboutPageView, essa pasta contém duas páginas adicionais:
    a. AstronomicalBodiesView. Esta página contém quatro botões que permitem ao usuário selecionar os detalhes do Sol, da Lua, da Terra ou do Cometa Halley. A versão atual do aplicativo é simplesmente uma prova de conceito. No futuro, essa página permitirá que o usuário selecione de uma lista muito maior.
    b. AstronomicalBodyView. Esta página é usada para exibir as informações de um corpo astronômico

AstronomicalBodiesView.xaml

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
                xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                x:Class="PAM_Astronomy.Views.AstronomicalBodiesView"
                Title="AstronomicalBodiesView">
        <ContentPage.Resources>
        <ResourceDictionary>
                <Style TargetType="Button">
                <Setter Property="FontSize" Value="100" />
                <Setter Property="BorderColor" Value="DarkGray" />
                </Style>
                <Style TargetType="Label">
                <Setter Property="HorizontalOptions" Value="Center" />
                <Setter Property="VerticalOptions" Value="End" />
                <Setter Property="Margin" Value="5" />
                <Setter Property="FontAttributes" Value="Bold" />
                </Style>
        </ResourceDictionary>
        </ContentPage.Resources>

        <Grid Margin="8">
        <Grid.RowDefinitions>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Button x:Name="btnEarth" Grid.Row="0" Grid.Column="0" Text="   🌎" />
        <Button x:Name="btnMoon"  Grid.Row="0" Grid.Column="1" Text="   🌕" />
        <Button x:Name="btnSun"   Grid.Row="1" Grid.Column="0" Text="   ☀️" />
        <Button x:Name="btnComet" Grid.Row="1" Grid.Column="1" Text="   ☄️" />
        </Grid>
</ContentPage>
```

AstronomicalBodiesView.xaml.cs

```
public partial class AstronomicalBodiesView : ContentPage
{
	public AstronomicalBodiesView()
	{
	InitializeComponent();

	btnComet.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails?astroName=comet");
	btnEarth.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails?astroName=earth");
	btnMoon.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails?astroName=moon");
	btnSun.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails?astroName=sun");
	}
}
```

AstronomicalBodyView.xaml

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
          xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
          x:Class="PAM_Astronomy.Views.AstronomicalBodyView"
          Title="AstronomicalBodyView">
          <ContentPage.Resources>
          <ResourceDictionary>
          <Color x:Key="textColor">White</Color>
          <Style TargetType="Label">
                    <Setter Property="TextColor" Value="{StaticResource textColor}" />
          </Style>
          </ResourceDictionary>
          </ContentPage.Resources>

          <Shell.BackButtonBehavior>
          <BackButtonBehavior IconOverride="comet.png"/>
          </Shell.BackButtonBehavior>

          <Grid RowSpacing="10" ColumnSpacing="10">
          <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*" />
          <ColumnDefinition Width="*" />
          </Grid.ColumnDefinitions>

          <Grid.RowDefinitions>
          <RowDefinition Height="auto" />
          <RowDefinition Height="auto" />
          <RowDefinition Height="auto" />
          <RowDefinition Height="auto" />
          <RowDefinition Height="auto" />
          <RowDefinition Height="*" />
          </Grid.RowDefinitions>

          <Image Aspect="AspectFill" Grid.RowSpan="6" Grid.ColumnSpan="2" Source="starfield.png" />

          <Label x:Name="lblIcon" FontSize="96" Grid.Row="0" Grid.ColumnSpan="2" HorizontalOptions="Center" />
          <Label x:Name="lblName" FontSize="Medium" FontAttributes="Bold" Grid.Row="1" Grid.ColumnSpan="2" HorizontalOptions="Center" />

          <Label FontSize="Medium" FontAttributes="Bold" Grid.Row="2" HorizontalOptions="End" Text="Mass:" />
          <Label FontSize="Medium" FontAttributes="Bold" Grid.Row="3" HorizontalOptions="End" Text="Circumference:" />
          <Label FontSize="Medium" FontAttributes="Bold" Grid.Row="4" HorizontalOptions="End" Text="Age:" />

          <Label x:Name="lblMass" FontSize="Medium" Grid.Row="2" Grid.Column="1" />
          <Label x:Name="lblCircumference" FontSize="Medium" Grid.Row="3" Grid.Column="1" />
          <Label x:Name="lblAge" FontSize="Medium" Grid.Row="4" Grid.Column="1" />
          </Grid>
</ContentPage>
```

AstronomicalBodyView.xaml.cs

```
public partial class AstronomicalBodyView : ContentPage
{
	public AstronomicalBodyView()
	{
	InitializeComponent();
	}

	string astroName;
	public string AstroName
	{
	get => astroName;
	set
	{
		astroName = value;
		UpdateAstroBodyUI(astroName);
	}
	}

	void UpdateAstroBodyUI(string astroName)
	{
	AstronomicalBody body = FindAstroData(astroName);

	Title = body.Name;

	lblIcon.Text = body.EmojiIcon;
	lblName.Text = body.Name;
	lblMass.Text = body.Mass;
	lblCircumference.Text = body.Circumference;
	lblAge.Text = body.Age;
	}

	AstronomicalBody FindAstroData(string astronomicalBodyName)
	{
	return astronomicalBodyName switch
	{
		"comet" => SolarSystemData.HalleysComet,
		"earth" => SolarSystemData.Earth,
		"moon" => SolarSystemData.Moon,
		"sun" => SolarSystemData.Sun,
		_ => throw new ArgumentException()
	};
	}
```

2. Uma rota precisa ser configurada para navegar até AstronomicalBodyPage. Você pode fazer isso no construtor da classe AppShell:

```
public AppShell()
{
    InitializeComponent();
     Routing.RegisterRoute("astronomicalbodydetails", typeof(AstronomicalBodyView));
}
```

Na página de Detalhes

3. No AstronomicalBodiesPage.xaml.cs, crie manipuladores (*handlers)* de eventos de clique para cada `Button` na página:

```
public AstronomicalBodiesPage()
 {
      InitializeComponent();

      btnComet.Clicked += async (s, e) => awaitShell.Current.GoToAsync("astronomicalbodydetails");
      btnEarth.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails");
      btnMoon.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails");
      btnSun.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails");
}
```

AstronomicalBody

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace PAM_Astronomy.Models
8   {
9       public class AstronomicalBody
10      {
11          public string Name { get; set; }
12          public string Mass { get; set; }
13          public string Circumference { get; set; }
14          public string Age { get; set; }
15          public string EmojiIcon { get; set; }
16      }
17  }
```

```
namespace PAM_Astronomy.Models
{
    public static class SolarSystemData
    {
        public static AstronomicalBody Sun = new
AstronomicalBody()
        {
            Name = "The Sun (Sol)",
            Mass = "1.9855*10^30 kg",
            Circumference = "4,379,000 km",
            Age = "4.57 billion years",
            EmojiIcon = "☀️",
        };

        public static AstronomicalBody Earth = new
AstronomicalBody()
        {
            Name = "Earth",
            Mass = "5.97237*10^24 kg",
            Circumference = "40,075 km",
            Age = "4.54 billion years",
            EmojiIcon = "🌎",
        };

        public static AstronomicalBody Moon = new
AstronomicalBody()
        {
            Name = "Moon",
            Mass = "7.342*10^22 kg",
            Circumference = "10,921 km",
            Age = "4.53 billion years",
            EmojiIcon = "🌕",
        };
        public static AstronomicalBody HalleysComet =
new AstronomicalBody()
        {
            Name = "Halley's Comet",
            Mass = "22 * 10^14 kg",
            Circumference = "11 km",
            Age = "4.6 billion years",
            EmojiIcon = "☄️",
        };
    }
}
```

```
namespace PAM_Astronomy.Services
{
    public class SunriseService
    {
        const string SunriseSunsetServiceUrl = "https://api.sunrise-sunset.org";

        public async Task<(DateTime Sunrise, DateTime Sunset)> GetSunriseSunsetTimes(double
latitude, double longitude)
        {
            var query =
$"{SunriseSunsetServiceUrl}/json?lat={latitude}&lng={longitude}&date=today";

            var client = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            var json = await client.GetStringAsync(query);

            var options = new JsonSerializerOptions { PropertyNameCaseInsensitive = true };
            var data = JsonSerializer.Deserialize<SunriseSunsetData>(json, options);

            return (DateTime.Parse(data.Results.Sunrise), DateTime.Parse(data.Results.Sunset));
        }
    }
}
```

Sunrise Service

```
namespace PAM_Astronomy.Services
{
    public class SunriseService
    {
        const string SunriseSunsetServiceUrl =
"https://api.sunrise-sunset.org";

        public async Task<(DateTime Sunrise, DateTime Sunset)>
GetSunriseSunsetTimes(double latitude, double longitude)
        {
            var query =
$"{SunriseSunsetServiceUrl}/json?lat={latitude}&lng={longitude}&dat
e=today";

            var client = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept",
"application/json");
            var json = await client.GetStringAsync(query);

            var options = new JsonSerializerOptions {
PropertyNameCaseInsensitive = true };
            var data =
JsonSerializer.Deserialize<SunriseSunsetData>(json, options);

            return (DateTime.Parse(data.Results.Sunrise),
DateTime.Parse(data.Results.Sunset));
        }
    }
}
```

Análise

Cada botão vai para uma página diferente, ou não?

1. Para enviar dados para AstronomicalBodyPage, adicione uma cadeia de caracteres de parâmetro de consulta às informações de roteamento. Ela estará no formato `?astroName=astroBodyToDisplay`

```
btnComet.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails?astroName=comet");
btnEarth.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails?astroName=earth");
btnMoon.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails?astroName=moon");
btnSun.Clicked += async (s, e) => await Shell.Current.GoToAsync("astronomicalbodydetails?astroName=sun");
```

2. Para receber os dados em AstronomicalBodyPage, crie uma propriedade de nível de classe para armazenar os dados de entrada. Nomeie-o como `AstroName`.

```
string astroName;
public string AstroName
{
    get => astroName;
    set
    {
        astroName = value;

        // this is a custom function to update the UI immediately
        UpdateAstroBodyUI(astroName);
    }
}
```

3. Agora é necessário **decorar** a classe com uma anotação que mapeia o parâmetro de consulta de entrada para a propriedade que você acabou de criar.

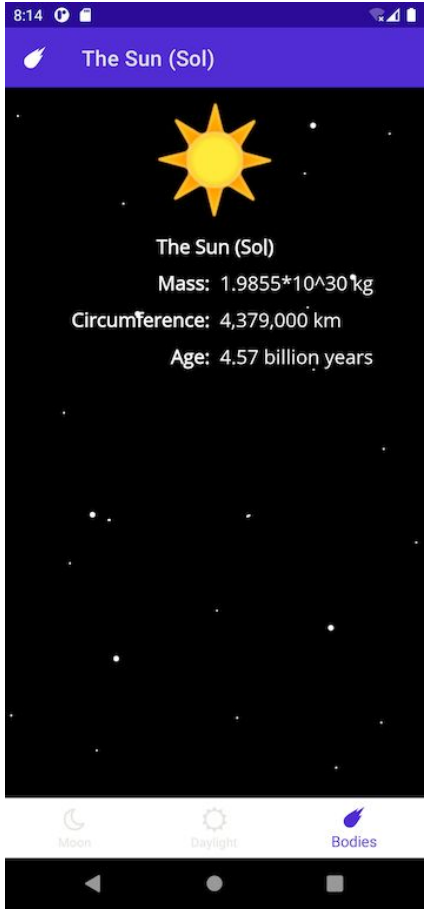
```
[QueryProperty(nameof(AstroName), "astroName")]

public partial class AstronomicalBodyPage

{ ...
```

Inicie o aplicativo e selecione a guia intitulada: Corpos.

Clique no botão Sol. Os detalhes do Sol devem aparecer. A barra de navegação deve conter uma *seta para voltar* que permite que o usuário retorne à lista de corpos. As guias restantes ainda estão visíveis e ativas:


8:14
The Sun (Sol)
The Sun (Sol)
Mass: 1.9855*10^30 kg
Circumference: 4,379,000 km
Age: 4.57 billion years
Moon
Daylight
Bodies

Adicionar dentro do Flyout Item

```
      <ShellContent Title="Bodies" Icon="comet.png"
            ContentTemplate="{DataTemplate local:AstronomicalBodiesPage}"/>
```

```
    <Shell.FlyoutHeader>
        <Grid HeightRequest="100" BackgroundColor="DarkSlateBlue">
            <Image Source="moon.png"/>
        </Grid>
    </Shell.FlyoutHeader>

    <FlyoutItem Title="Astronomy" Icon="moon.png">
        <ShellContent Title="Moon" Icon="moon.png"
                ContentTemplate="{DataTemplate local:MoonPhasePage}" />
        <ShellContent Title="Daylight" Icon="sun.png"
            ContentTemplate="{DataTemplate local:SunrisePage}" />
        <ShellContent Title="Bodies" Icon="comet.png"
            ContentTemplate="{DataTemplate local:AstronomicalBodiesPage}"/>
    </FlyoutItem>

    <FlyoutItem Title="About" Icon="question.png">
        <ShellContent
            ContentTemplate="{DataTemplate local:AboutPage}"/>
    </FlyoutItem>

</Shell>
```

Referências e ajudas

https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/shell/flyout

https://learn.microsoft.com/en-us/dotnet/maui/user-interface/pages/flyoutpage

https://www.youtube.com/watch?v=T3HPTy86rU4

https://medium.com/@cristian_lopes/net-maui-flyout-navigation-tab-navigation-ca62913c98fa

https://learn.microsoft.com/pt-br/training/modules/create-multi-page-apps/5-exercise-implement-tab-navigation