Xamarin.Forms - Usando XAML e C# para criar Interfaces com o usuário

No artigo de hoje vou apresentar um resumo básico sobre a anatomia de uma aplicação **Xamarins.Forms** e como podemos usar XAML e C# para criar interfaces com o usuário. (**Artigo Anterior**)

Nota: Quando este artigo foi escrito o Xamarin era uma ferramenta paga, mas em março de 2016 ela foi adquirida pela Microsoft e agora esta integrada ao Visual Studio de forma gratuita.
Você pode fazer o download do Xamarin neste link : https://www.xamarin.com/download

O **Xamarin.Forms,** lançado em 2014**,** é uma plataforma que roda no **Xamarin** e foi originalmente criada como um **kit de ferramentas UI** que permite aos desenvolvedores criar facilmente interfaces de usuário que podem ser compartilhadas entre Android, iOS e Windows Phone.

O **Xamarin.Forms** foi escrito na linguagem C#, e, permite a prototipagem rápida de aplicações que podem evoluir ao longo do tempo para aplicações complexas. Como os aplicativos **Xamarin.Forms** são aplicativos nativos, eles não têm as limitações de outros kits de ferramentas.

A interface de usuário (**UI**) moderna é construída a partir de objetos visuais de vários tipos. Dependendo do sistema operacional, estes objetos visuais podem vir com nomes diferentes : controles, elementos, views, widgets,etc., mas todos eles são dedicados aos trabalhos de apresentação ou interação com o usuário.

No **Xamarin.Forms**, os objetos que aparecem na tela são chamados coletivamente de elementos visuais e podem ser classificados em quatro categorias principais:

1. **View** - referem-se a controles ou widgets em outras plataformas. Correspondem a elementos de interface do usuário como labels, buttons, campos textos, etc.;
2. **Page** - o Xamarin.Forms representa uma única tela em sua aplicação. São semelhantes ao conceito de **Activity** do Android, ou a uma **Page** no Windows Phone, ou a um controle **View** no iOS;
3. **Layout** - É um subtipo especializado de View e atua como um container para outros Layouts e Views. Os subtipos de leiautes contém lógica que é especifica para organizar as views filhas de certa forma;
4. **Cell** - Esta classe é um elemento especializado que é usado para itens em uma lista ou tabela. Ele descreve como cada item na lista deverá ser desenhado;

Estes conceitos não são abstratos. API do Xamarin.Forms define classes chamadas : **VisualElement, Page, Layout, e View**. Essas classes e seus descendentes formam a espinha dorsal da interface de usuário do Xamarin.Forms. A classe **VisualElement** é uma classe extremamente importante no Xamarin.Forms. Um objeto **VisualElement** é algo que ocupa espaço na tela. Uma aplicação Xamarin.Forms consiste em uma ou mais páginas. Uma página normalmente ocupa toda, ou pelo menos uma grande área, da tela.

Algumas aplicações são compostas por apenas uma única página (**ContentPage**), enquanto outros permitem a navegação entre múltiplas páginas. Em cada página, os elementos visuais são organizados em uma hierarquia pai-filho. O filho de uma **ContentPage** é geralmente um layout de algum tipo para organizar os elementos visuais.

Alguns **layouts** têm um único filho, mas muitos layouts possuem vários filhos que o layout organiza dentro de si. Estes filhos podem ser outros layouts ou views. Diferentes tipos de layouts organizam os filhos em uma pilha, numa grade bidimensional, ou de um modo mais livre. Inicialmente vamos tratar com layouts com um único filho.

O termo **view** no Xamarin.Forms indica tipos familiares de objetos de apresentação e de interação : *text, bitmaps, buttons, campos de entrada de texto, controles deslizantes, interruptores, barras de progresso, data e hora, e outros* que você pode criar. Estes são muitas vezes chamados de controles ou widgets em outros ambientes de programação. Vamos chamá-los de **views ou elements.**

A tabela abaixo lista alguns dos controles mais usados:

Controle Xamarin.Forms	Descrição
Label	Exibe texto somente para leitura.
Entry	Representa um controle de texto de uma única linha.
Button	São usados para iniciar comandos.
Image	Usado para exibir imagens bitmap.
ListView	Apresenta uma lista de itens onde os itens dentro da lista são conhecidos como **Cells**.

Os controles são hospedados dentro de um **Layout** e o Xamarin.Forms possui duas categorias diferentes de leiautes que arranjam os controles em diferentes modos:

- **Managed Layouts** (*layout gerenciado*) - São leiautes que tomam conta do posicionamento e numeração dos controles filhos na tela e seguem o modelo **CSS box.** Aplicativos não devem tentar definir diretamente o tamanho ou a posição de controles filho. Um exemplo comum desse leiaute é o **StackLayout**.
- **Unmanaged Layouts** (*layout não gerenciado*) - Neste modo não existe a organização ou posicionamento de seus filhos na tela. O usuário irá especificar o tamanho e localização do controle filho quando ele for adicionado no layout. Como exemplo deste layout temos o **AbsoluteLayout**.

StackLayout

Este layout é muito comum e simplifica muito o desenvolvimento **cross-plataform** pois arranja automaticamente os controles na tela sem considerar o seu tamanho. Cada elemento filho é posicionado um depois do outro, que seja horizontalmente quer seja verticalmente na ordem que foram incluídos. O espaço que este layout irá usar vai depender em como as propriedades **HorizontalOptions e LayoutOptions** forem definidas, mas por padrão ele vai tentar usar a tela inteira.

A seguir temos um exemplo de utilização de **StackLayout** para organizar 3 controles Labels na tela usando **C# e XAML** :

```
public class StackLayoutExemplo: ContentPage

{
   public StackLayoutExample()
   {
      Padding = new Thickness(20);
      var red = new Label
      {
         Text = "Pare",
         BackgroundColor = Color.Red,
         FontSize = 20
      };
      var yellow = new Label
      {
         Text = "Devagar",
         BackgroundColor = Color.Yellow,
         FontSize = 20
      };
      var green = new Label
      {
         Text = "Siga",
         BackgroundColor = Color.Green,
         FontSize = 20
      };

      Content = new StackLayout
      {
         Spacing = 10,
         Children = { red, yellow, green }
      };
   }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2015/forms"
                xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                x:Class="AloMundo.StackLayoutExemplo"
          Padding="20">

 <StackLayout Spacing="10">

  <Label Text="Pare"
       BackgroundColor="Red"
       Font="20" />

  <Label Text="Devagar"
       BackgroundColor="Yellow"
       Font="20" />

  <Label Text="Siga"
       BackgroundColor="Green"
       Font="20" />

 </StackLayout>
</ContentPage>
```

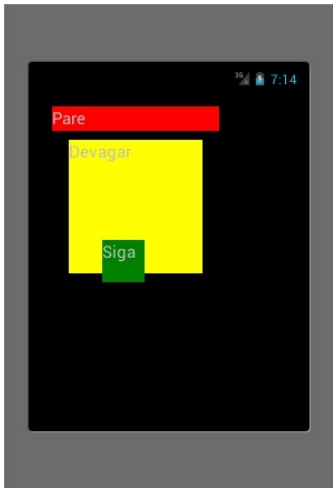
AbsoluteLayout

Neste leiaute cada controle precisa explicitamente posicionado dentro do layout lembrando o estilo de posicionamento Windows Forms. Enquanto este modo permite um posicionamento muito preciso dos controles, ele requer mais testes para diferentes tipos de tamanho de telas.

A seguir temos um exemplo de utilização de **AbsoluteLayout** para organizar 3 controles Labels na tela usando **C# e XAML** e o resultado obtido no emulador Android :

```
public class MyAbsoluteLayoutPage : ContentPage
{
   public MyAbsoluteLayoutPage()
   {
      var red = new Label
      {
         Text = "Pare",
         BackgroundColor = Color.Red,
         FontSize = 20,
         WidthRequest = 200,
         HeightRequest = 30
      };
      var yellow = new Label
      {
         Text = "Devagar",
         BackgroundColor = Color.Yellow,
         FontSize = 20,
         WidthRequest = 160,
         HeightRequest = 160
      };
      var green = new Label
      {
         Text = "Siga",
         BackgroundColor = Color.Green,
         FontSize = 20,
         WidthRequest = 50,
         HeightRequest = 50

      };
      var absLayout = new AbsoluteLayout();
      absLayout.Children.Add(red, new Point(20,20));
      absLayout.Children.Add(yellow, new Point(40,60));
      absLayout.Children.Add(green, new Point(80,180));

      Content = absLayout;
   }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2015/forms"
                xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                x:Class="HelloXamarinFormsWorldXaml.AbsoluteLayoutExample"
          Padding="20">

 <AbsoluteLayout>

  <Label Text="Pare"
       BackgroundColor="Red"
       Font="20"
       AbsoluteLayout.LayoutBounds="20,20,200,30" />

  <Label Text="Devagar"
       BackgroundColor="Yellow"
       Font="20"
       AbsoluteLayout.LayoutBounds="40,60,160,160" />

  <Label Text="Siga"
       BackgroundColor="Green"
       Font="20"
       AbsoluteLayout.LayoutBounds="80,180,50,50" />

 </AbsoluteLayout>

</ContentPage>
```



Listas no Xamarin.Forms - ListView

O controle **ListView** é muito comum em aplicações mobile, sendo responsável por exibir uma coleção de itens na tela, onde cada item está contido em uma única célula. Por padrão um ListView irá usar o template **TextCell** e irá renderizar uma única linha de texto.

Abaixo temos um trecho de código que é um exemplo simples de como usar o controle **ListView** onde criamos um arquivo **TestePage1.xaml.**

Neste arquivo usamos o código XAML onde definimos uma Label com o texto Macoratti .net alinhado ao centro com rotação de -15 e outros atributos.

Definimos também um **ListView** chamado **ListaCliente** com cor de fundo igual a **White**:

```
<?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="StackLayoutExemplo.TestePage1">
  <ContentPage.Content>
    <StackLayout Padding="20" Spacing="20">
     <Label Text="Macoratti .net"
        VerticalOptions="Start"
        XAlign="Center"
        Rotation="-15"
        IsVisible="true"
        FontSize="Large"
        FontAttributes="Bold"
        TextColor="Aqua" />
      <ListView x:Name="ListaCliente" BackgroundColor="White" />
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

Para atribuir itens ao ListView usamos sua propriedade **ItemSource** no arquivo code-behind **TestePage1.xaml.cs** :

```
using Xamarin.Forms;

namespace StackLayoutExemplo
{
   public partial class TestePage1 : ContentPage
   {
      public TestePage1()
      {
         InitializeComponent();
         this.ListaCliente.ItemsSource = new string[]
          {
                        "Comprar Peras",
                       "Comprar Laranjas",
                       "Comprar Mangas",
                       "Comprar Maças",
                      "Comprar bananas"
          };
      }
   }
}
```

A figura ao lado mostra o resultado visto em um emulador Android.

Podemos também exibir objetos customizados em um **ListView** usando o template **TextCell** e vinculando a fonte de dados à propriedade **ItemsSource**.

No código abaixo temos primeiro a definição da classe **TodoItem** a seguir a vinculação dos dados no ListView através da propriedade **ItemsSource** e a definição do controle cuja propriedade será exibida na lista(**TextCell**) no caso a propriedade **Name**:

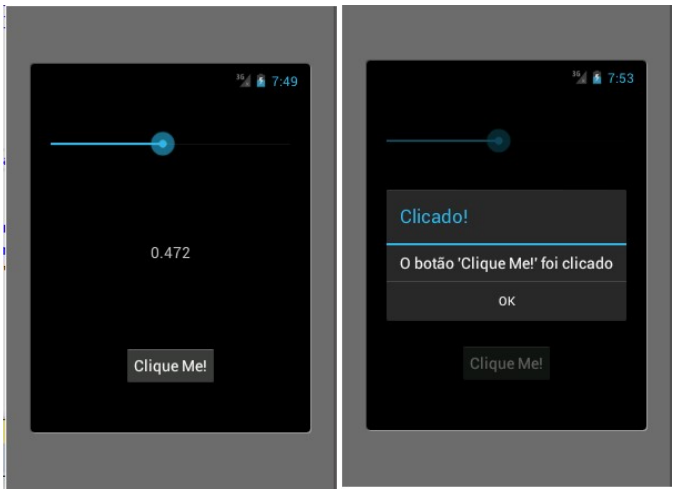
```
public class TodoItem {
   public string Name { get; set; }
   public bool Done { get; set; }
}

listView.ItemsSource = new TodoItem [] {
   new TodoItem {Name = "Comprar Peras"},
   new TodoItem {Name = "Comprar Laranjas", Done=true},
   new TodoItem {Name = "Comprar Magas"},
   new TodoItem {Name = "Comprar Maças", Done=true},
   new TodoItem {Name = "Comprar Bananas", Done=true}
};

listView.ItemTemplate = new DataTemplate(typeof(TextCell));
listView.ItemTemplate.SetBinding(TextCell.TextProperty, "Name");
```

Em outro exemplo temos o arquivo **Page1.xaml** onde definimos um Slider com o evento **ValueChanged**, uma Label e um Button com o evento **Clicked** usando o código XAML:

```
<?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="XAML_Eventos.Page1">

  <StackLayout>
    <Slider VerticalOptions="CenterAndExpand"
            ValueChanged="OnSliderValueChanged" />

    <Label x:Name="valueLabel"
      Text="Uma label"
      Font="Large"
      HorizontalOptions="Center"
      VerticalOptions="CenterAndExpand" />

    <Button Text="Clique Me!"
            HorizontalOptions="Center"
            VerticalOptions="CenterAndExpand"
            Clicked="OnButtonClicked" />
  </StackLayout>

</ContentPage>
```

No arquivo code-behind **Page1.xaml.cs** definimos o código que trata os eventos **OnSliderValueChanged** e **OnButtonClicked** definidos no código XAML:

```
using System;

using Xamarin.Forms;

namespace XAML_Eventos
{
   public partial class Page1 : ContentPage
   {
      public Page1()
      {
         InitializeComponent();
      }

      void OnSliderValueChanged(object sender,
                        ValueChangedEventArgs args)
      {
         valueLabel.Text = ((Slider)sender).Value.ToString("F3");
      }

      async void OnButtonClicked(object sender, EventArgs args)
      {
         Button button = (Button)sender;
         await DisplayAlert("Clicado!","O botão '" + button.Text + "' foi clicado","OK");
      }
   }
}
```



Dessa forma podemos conjugar código XAML com código C# ou se preferir somente código C# para definir a interface do usuário em aplicações Xamarin.Forms.

No próximo artigo vou mostrar como criar a sua primeira aplicação Xamarin.Forms.

Quer aprender os conceitos da Programação Orientada a objetos ?

- **Curso Fundamentos da Programação Orientada a Objetos com VB .NET** NEW

Quer aprender o gerar relatórios com o ReportViewer no VS 2013 ?

- **Curso - Gerando Relatórios com o ReportViewer no VS 2013 - Vídeo Aulas** NEW

Gostou ? Compartilhe no Facebook Compartilhe no Twitter

Referências:

- Seção VB .NET do Site Macoratti.net
- Super DVD .NET - A sua porta de entrada na plataforma .NET
- Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#
- Super DVD C# - Recursos de aprendizagens e vídeo aulas para C#
- Seção C# do site Macoratti.net
- Seção ASP .NET do site Macoratti .net
- Curso Básico VB .NET - Vídeo Aulas
- Curso C# Básico - Vídeo Aulas
- Curso Fundamentos da Programação Orientada a Objetos com VB .NET NEW
- **Macoratti .net | Facebook**
- **macoratti - YouTube**
- **Jose C Macoratti (@macorati) | Twitter**
- **An Introduction to Xamarin.Forms - Xamarin**
- **https://www.visualstudio.com/pt-br/features/xamarin-vs.aspx**
- **https://xamarin.com/starter**
- **Xamarim Studio - Desenvolvimento Multiplataforma com C# (Android, iOS e Windows)**
- **Xamarim - Criando Apps com o Visual Studio e C#** **(vídeo aula)**

José Carlos Macoratti