Macoratti.net Xamarin Forms - Criando uma página de pequisa

Neste artigo vou mostrar como criar uma página de pequisa usando um SearchBar e um ListView em aplicações Xamarin Forms usando o Visual Studio 2015 e a linguagem C#.

Esse artigo será essencialmente prático e vai usar os conceitos já abordados sobre as views ListView e SearchBar no Xamarin. Forms. Assim se você esta chegando agora recomendo que leia os artigos onde eu já tratei desse assunto:

- **Xamarin.Forms - Trabalhando com ListView - Macoratti**
- **Xamarin Forms - Acessando a base de filmes do NetFlix - Macoratti**

No exemplo usado no artigo vamos criar um projeto Xamarin Forms e definir uma página de **Cadastro de Produtos** onde precisamos fazer a busca das categorias cadastradas.

Vamos implementar assim uma página de busca usando o SearchBar e o ListView que vai retornar o código e o nome de uma categoria selecionada.

Então vamos à parte prática...

Recursos usados:

- **Visual Studio Community 2015 ou Xamarin Studio**
- **Xamarin**

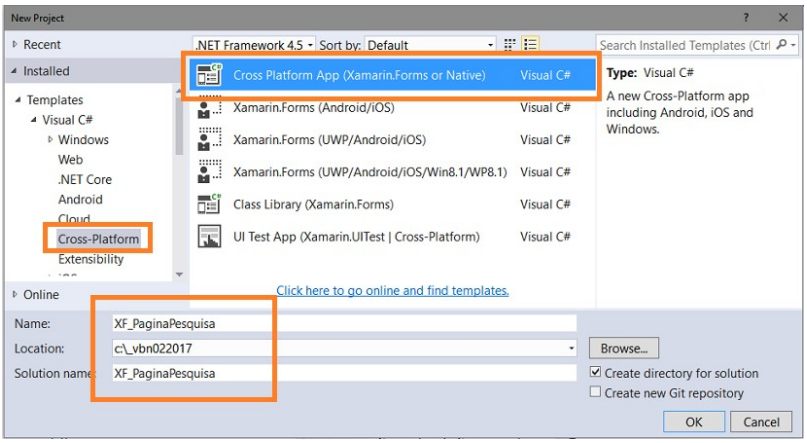
Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.

Criando o projeto no Visual Studio 2015 Community com a nova versão do Xamarin Forms

Abra o Visual Studio Community 2015 e clique em **New Project;**

Selecione Visual C#, o template **Cross Plataform** e a seguir **Cross Plataform App(Xamarin.Forms or Native);**

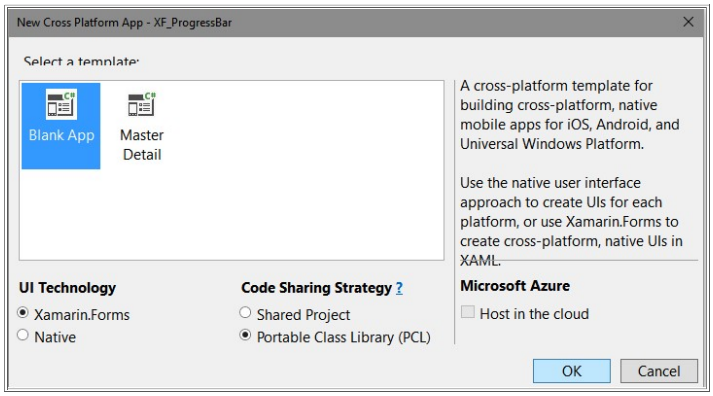
Informe o nome **XF_PaginaPesquisa** e clique no botão OK;



Observe que a nova versão do Xamarin Forms disponbilizada em **Fevereiro/2017** trás novas opções de templates de projeto.

A seguir selecione **Blank App** e marque as opções - **Xamarin.Forms** e **Portable Class Library (PCL**) e clique em OK;

Será apresentada a janela abaixo, outra diferença da nova versão, onde você pode escolhar os templates **Blank App** e **Master Detail** e a seguir a tecnologia e o tipo de projeto.



Marque as opções Blank App, Xamarin.Forms e Portable Class Library(PCL) e clique em OK.

Será criado uma solução contendo no projeto **Portable** as páginas **App.xaml e MainPage.xaml.** Essa é outra diferença que a nova versão trouxe.

No code-behind do arquivo **App.xaml** temos a classe **App.cs** que irá conter o código compartilhado e que vamos usar neste artigo.

Iremos usar a página **MainPage.xaml** como página principal da nossa aplicação.

Assim se você abrir o arquivo **App.cs** verá alterar o código já definido conforme o mostrado abaixo que abre a página **MainPage** :

```
using Xamarin.Forms;

namespace XF_Bindable_Spinner
{
   public class App : Application
   {
      public App()
      {
       MainPage = new NavigationPage(new XF_PaginaPesquisa.MainPage());
      }

      protected override void OnStart()
      {
         // Handle when your app starts
      }

      protected override void OnSleep()
      {
         // Handle when your app sleeps
      }

      protected override void OnResume()
      {
         // Handle when your app resumes
      }
   }
}
```

Definindo a página de cadastro de produtos na MainPage

Abra o arquivo **MainPage.xaml** e altere o seu código conforme mostrado a seguir: *(Ao lado você verá o layout da página que será exibida ao cliente)*

```
<?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:local="clr-namespace:XF_PaginaPesquisa"
             BackgroundColor="White"
             x:Class="XF_PaginaPesquisa.MainPage"
             Title="Cadastro de Produto">

        <StackLayout>
          <StackLayout Orientation="Horizontal" Padding="5">
          <Label x:Name="CategoriaId" Text=">>" FontSize="Medium" HorizontalOptions="End" />
          <Label x:Name="CategoriaNome" Text="Selecione a Categoria" FontSize="Medium" HorizontalOptions="Start"/>
          <Label.GestureRecognizers>
             <TapGestureRecognizer Tapped="OnTapCategorias" NumberOfTapsRequired="1"/>
          </Label.GestureRecognizers>

        </StackLayout>
          <Entry Placeholder="Informe o nome do Produto" />
          <Entry Placeholder="Informe o preço do Produto" />
          <Entry Placeholder="Informe o estoque do Produto" />
         <StackLayout Orientation="Horizontal" VerticalOptions="Center">
             <Button Text="Gravar" HorizontalOptions="StartAndExpand"/>
             <Button Text="Cancelar" HorizontalOptions="EndAndExpand"/>
          </StackLayout>
      </StackLayout>
</ContentPage>
```

Neste código estou apenas definindo uma página de cadastro de produtos apenas para mostrar como criar a página de pesquisa.

No código em destaque eu defini duas Labels para exibir o **Id** e o **Nome da categoria.**

Agora observe que eu defini também para as Labels o evento **Tapped** usando os recursos da classe **GestureRecognizer**. Dessa forma eu tornei as Labels clicáveis e defini na propriedade **NumberOfTapsRequired** que basta clicar uma vez para que o evento fosse disparado.

Para tornar um elemento de interface do usuário clicável com o gesto de toque fazemos assim:

1. Criamos uma instância da classe **TapGestureRecognizer;**
2. Manipulamos o evento **Tapped** ;
3. Adicionamos o novo reconhecedor de gesto para a coleção de **GestureRecognizers** sobre o elemento de interface do usuário;

Então vejamos a seguir o código C# do evento **OnTapCategorias** no arquivo **MainPage.xaml.cs** :

```
      private async void OnTapCategorias(object sender, EventArgs e)
      {
         await Navigation.PushAsync(new CategoriasPage(CategoriaNome, CategoriaId));
      }
```

Neste código estamos navegando para a página **CategoriasPage**() passando as Labels **CategoriaNome** e **CategoriaId**.

Definindo o modelo de domínio e a classe de serviço

Antes de criar a página de pesquisa precisamos definir um modelo de domínio que será a classe **Categoria** e também precisamos criar um serviço, que será a classe **CategoriaService**, que forneça alguns dados para trabalharmos, visto que não vou usar um banco de dados para tornar o projeto mais simples. Vamos criar também a pasta Views para conter a página de pesquisa.

Para melhor organizar o código no projeto Portable vamos criar três pastas no projeto :

- **Model** - pasta onde iremos criar a classe **Categoria**;
- **Service** - pasta onde iremos criar a classe **CategoriaService**;
- **Views** - pasta onde iremos criar a página de pesquisa CategoriasPage;

Para criar as pastas selecione o projeto e no menu **Project** clique em **New Folder** e informe o nome da pasta.

Criando o modelo de domínio

Selecione a pasta **Model** e no menu **Project** clique em **Add Class** e informe o nome **Categoria**; a seguir digite o código abaixo neste arquivo:

```
namespace XF_PaginaPesquisa.Model
{
    public class Categoria
    {
        public long? CategoriaId { get; set; }
        public string Nome { get; set; }
    }
}
```

Criando o serviço

Da mesma forma selecione a pasta **Model** e crie a classe **CategoriaService** contendo o seguinte código:

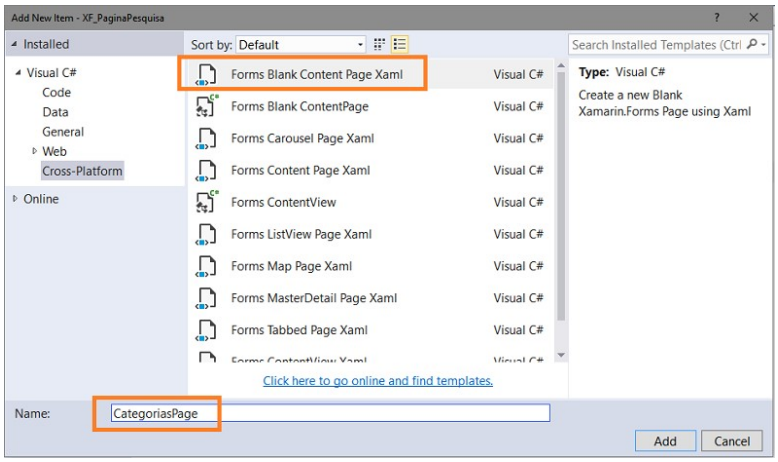
```
using System.Collections.ObjectModel;
using XF_PaginaPesquisa.Model;

namespace XF_PaginaPesquisa.Service
{
    public class CategoriaService
    {
        private ObservableCollection<Categoria> categorias = new ObservableCollection<Categoria>();
        public CategoriaService()
        { }

        public ObservableCollection<Categoria> GetCategorias()
        {
            categorias.Add(new Categoria() { CategoriaId = 1, Nome = "Equipamentos" });
            categorias.Add(new Categoria() { CategoriaId = 2, Nome = "Acessórios" });
            categorias.Add(new Categoria() { CategoriaId = 3, Nome = "Notebooks" });
            categorias.Add(new Categoria() { CategoriaId = 4, Nome = "Impressoras" });
            categorias.Add(new Categoria() { CategoriaId = 5, Nome = "SmartPhones" });
            categorias.Add(new Categoria() { CategoriaId = 6, Nome = "Tablets" });
            categorias.Add(new Categoria() { CategoriaId = 7, Nome = "Toners" });
            categorias.Add(new Categoria() { CategoriaId = 8, Nome = "TVs" });

            return categorias;
        }

        public void Add(Categoria categoria)
        {
            categorias.Add(categoria);
        }
    }
}
```

Criando a página de pesquisa

Agora que já temos o domínio e o serviço que fornece alguns dados vamos criar página de pesquisa.

Selecione a pasta **View** e no menu **Project** clique em **Add New Item** e selecione o item **Cross Platform** e o template **Forms Blank Content Page Xaml** (*esta é outra novidade da nova versão*).

Informe o nome **CategoriasPage** e clique em **Add**;



A seguir defina o código XAML abaixo no arquivo .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="XF_PaginaPesquisa.Views.CategoriasPage"
         Title="Seleção de Categoria">

   <StackLayout Orientation="Vertical" HorizontalOptions="FillAndExpand" Padding="5,20,5,0">

      <SearchBar Placeholder="Digite o nome da categoria..." TextColor="Black" TextChanged="OnTextChanged" />

      <ListView x:Name="lvwCategorias" HasUnevenRows="True" ItemTapped="lvwCategorias_OnItemTapped" >
         <ListView.ItemTemplate>
            <DataTemplate>
               <ViewCell>
                  <StackLayout Orientation="Horizontal">
                     <Label Text="{Binding CategoriaId}" TextColor="Blue" FontSize="0" HorizontalOptions="Start"/>
                     <Label Text="{Binding Nome}" TextColor="Blue" FontSize="Large" HorizontalOptions="End"/>
                  </StackLayout>
               </ViewCell>
            </DataTemplate>
         </ListView.ItemTemplate>
      </ListView>
   </StackLayout>

</ContentPage>
```

Vamos entender o código :

Estamos usando um **SearchBar** que fornece uma caixa de busca e definimos o evento **TextChanged** - **OnTextChanged -** nesta view para tratar a digitação do usuário;

A ListView suporta seleção de um item de cada vez, sendo que a seleção esta ativada por padrão. Assim, quando um usuário toca em um item da lista, dois eventos são disparados:

1. **ItemTapped**
2. **ItemSelected**

Assim definimos o evento **ItemTapped** - **lvwCategorias_OnItemTapped**.

Estamos usando um **DataTemplate** para exibir dados de uma coleção de objetos em um ListView.

A aparência dos dados em cada célula em um ListView pode ser gerenciada pela definição da propriedade **ItemTemplate** para um **DataTemplate**, onde os elementos especificados no DataTemplate definem a aparência de cada célula.

Lembrando que um filho de um **DataTemplate** tem que ser do tipo ou derivar de um **ViewCell** onde estamos usando o databinding para exibir o Id e o nome da categoria.

Código dos eventos da página de pesquisa

Vamos agora definir o código dos eventos da página de pesquisa.

Abra o arquivo **CategoriaPage.xaml.cs** e informe o código a seguir :

```
using System.Collections.Generic;
using System.Linq;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using XF_PaginaPesquisa.Model;
using XF_PaginaPesquisa.Service;

namespace XF_PaginaPesquisa.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class CategoriasPage : ContentPage
   {
```

```
        private Label nome;
        private Label valor;
        private IEnumerable<Categoria> itens;
        CategoriaService categ = new CategoriaService();

        public CategoriasPage(Label nome, Label valor)
        {
            InitializeComponent();
            itens = categ.GetCategorias();
            this.valor = valor;
            this.nome = nome;
        }

        private void OnTextChanged(object sender, TextChangedEventArgs e)
        {
            lvwCategorias.BeginRefresh();

            if (string.IsNullOrWhiteSpace(e.NewTextValue))
            {
                lvwCategorias.ItemsSource = itens;
            }
            else
            {
                lvwCategorias.ItemsSource = itens.Where(i => i.Nome.Contains(e.NewTextValue));
            }
            lvwCategorias.EndRefresh();
        }

        private async void lvwCategorias_OnItemTapped(object sender, ItemTappedEventArgs e)
        {
            var item = (sender as ListView).SelectedItem as Categoria;
            this.valor.Text = item.CategoriaId.ToString();
            this.nome.Text = item.Nome;
            await Navigation.PopAsync();
        }
    }
}
```

Definimos as variáveis **nome e valor** do tipo Label que receberão os valores selecionados e definimos a variável **itens** que representa uma coleção de categorias a exibir, e criamos uma instância da classe **CategoriaService**:

```
        private Label nome;
        private Label valor;
        private IEnumerable<Categoria> itens;
        CategoriaService categ = new CategoriaService();
```

No construtor da classe atribuimos os valores às variáveis. Onde para obter os itens chamamos o método **GetCategorias**() da classe **CategoriaService** que retorna todas as categorais:

```
        public CategoriasPage(Label nome, Label valor)
        {
            InitializeComponent();
            itens = categ.GetCategorias();
            this.valor = valor;
            this.nome = nome;
        }
```

No evento **OnTextChanged** que será disparado quando o usuário digitar algo na **SearchBar** temos o código abaixo:

```
        private void OnTextChanged(object sender, TextChangedEventArgs e)
        {
            lvwCategorias.BeginRefresh();

            if (string.IsNullOrWhiteSpace(e.NewTextValue))
            {
                lvwCategorias.ItemsSource = itens;
            }
            else
            {
                lvwCategorias.ItemsSource = itens.Where(i => i.Nome.Contains(e.NewTextValue));
            }
            lvwCategorias.EndRefresh();
        }
```

Neste código, se o valor digitado for vazio ou nulo atribuimos a coleção de itens ao ListView. Se o cliente digitar algum texto então filtramos o resultado usando uma consulta LINQ.

O código do evento **OnItemTapped** do Listview mostrado abaixo será disparado quando um item do ListView for selecionado:

```
        private async void lvwCategorias_OnItemTapped(object sender, ItemTappedEventArgs e)
        {
            var item = (sender as ListView).SelectedItem as Categoria;
            this.valor.Text = item.CategoriaId.ToString();
            this.nome.Text = item.Nome;
            await Navigation.PopAsync();
        }
```

Aqui atribuímos os valores do Id e do nome da categoria às Labels para serem exibidos na página **MainPage**.

Executando o projeto iremos obter o seguinte resultado:

Pegue o projeto aqui : **XF_PaginaPesquisa.zip** (sem as referências)

(Disse Jesus) - 'Aquele que tem os meus mandamentos e os guarda esse é o que me ama; e aquele que me ama será amado de meu Pai, e eu o amarei, e me manifestarei a ele.'
João 14:21

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**
- VB.NET 2005 - Controles - Macoratti.net
- Seção de Jogos do site Macoratti .net
- Xamarim - Desenvolvimento Multiplataforma com C# ... - Macoratti.net
- Xamarin - Apresentando Xamarin.Forms - Macoratti.net
- Xamarin.Forms - Olá Mundo - Criando sua primeira ... - Macoratti.net
- Xamarin.Forms - Olá Mundo - Anatomia da aplicação - Macoratti.net
- https://developer.xamarin.com/api/type/Android.App.AlertDialog/
- Xamarin Android - Tratando eventos de forma declarativa
- Seção Mobile/Xamarin do site Macoratti .net
- https://developer.android.com/reference/android/app/Activity.html
- https://developer.xamarin.com/api/type/Android.Widget.ProgressBar/

José Carlos Macoratti