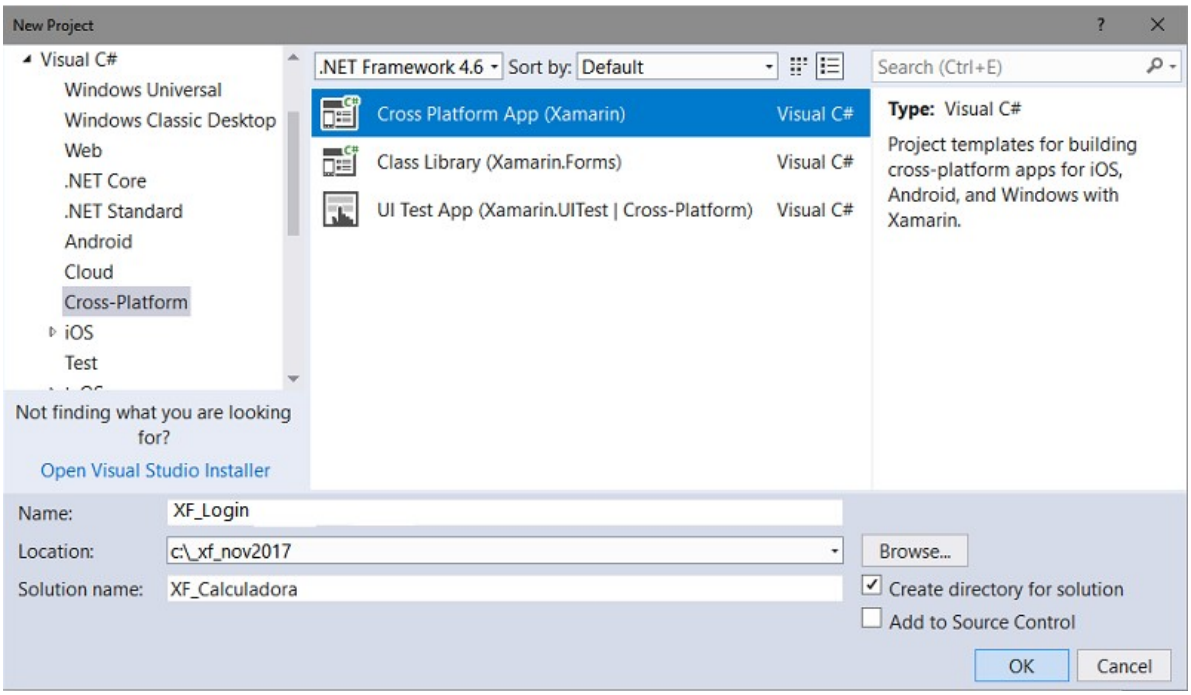
Xamarin Forms - Criando uma página de Login (MVVM)

Neste artigo vamos criar uma uma página de Login usando a abordagem MVVM em uma aplicação Xamarin Forms usando a linguagem C#.

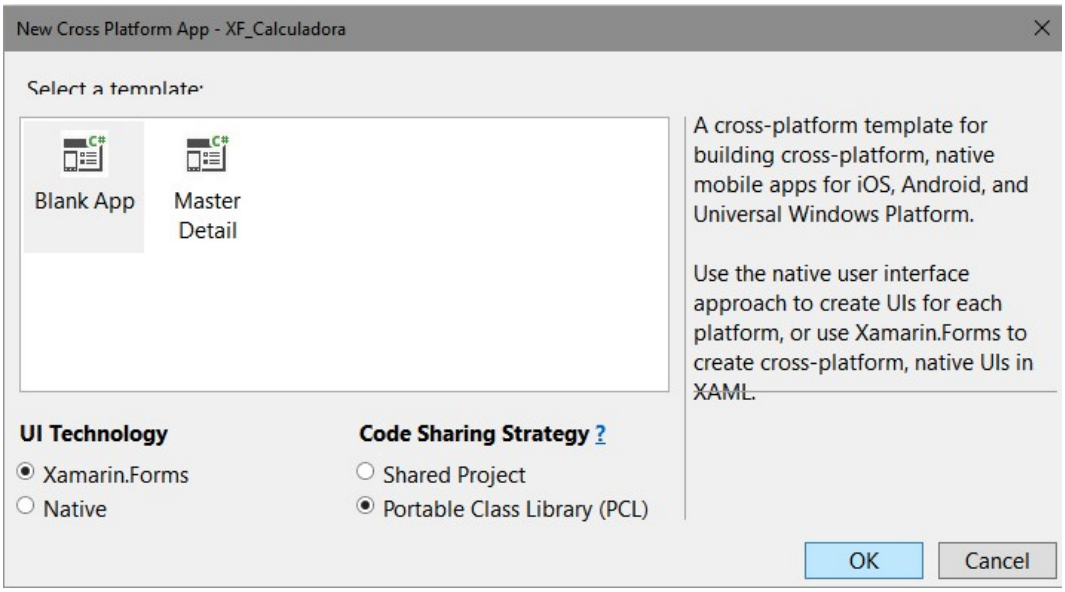
Esse artigo é um tutorial que mostra como criar uma página de login simples no Xamarin Forms usando a abordagem **MVVM.**

Abra o VS 2017 Community *(estou usando a versão 15.4.2)* e crie um projeto Xamarin Forms selecionando a opção **Visual C# -> Cross-Platform;**

Selecionando o template **Cross Platform App (Xamarin)** e clicando em OK;



A seguir marque **Xamarin Forms** e **Portable Class Library (PCL)** e clique em OK;



Definindo a ViewModel e a View

Vamos criar duas novas pastas em nosso projeto.

1. A pasta **ViewModels** onde vamos criar a classe LoginViewModel
2. A pasta **Pages** onde vamos criar a página de Login : LoginPage.xaml

No menu **Project** clique em **Add Folder** e informe **ViewModels**. Repita a operação e informe **Pages**.

Selecione a pasta **ViewModels** e no menu **Project** clique em **Add Class** informando o nome **LoginViewModel.cs.**

A seguir inclua o código abaixo neste arquivo:

```
using System;
using System.ComponentModel;
using System.Windows.Input;
using Xamarin.Forms;

namespace XF_Login.ViewModels
{
   public class LoginViewModel : INotifyPropertyChanged
   {
      public Action ExibirAvisoDeLoginInvalido;

      public event PropertyChangedEventHandler PropertyChanged = delegate { };

      private string email;
      public string Email
      {
         get { return email; }
         set
         {
            email = value;
            PropertyChanged(this, new PropertyChangedEventArgs("Email"));
         }
      }

      private string senha;
      public string Senha
      {
         get { return senha; }
         set
         {
```

```
            senha = value;
            PropertyChanged(this, new PropertyChangedEventArgs("Senha"));
        }
    }

    public ICommand SubmitCommand { protected set; get; }

    public LoginViewModel()
    {
        SubmitCommand = new Command(OnSubmit);
    }

    public void OnSubmit()
    {
        if (email != "macoratti@yahoo.com" || senha != "numsey")
        {
            ExibirAvisoDeLoginInvalido();
        }
    }

  }
}
```

No código definimos duas propriedades : **Email e Senha** e o comando **SubmitCommand**() que iremos usar no botão de comando Login da página de Login. Usamos a interface **ICommand** que permite definir e implementar um comando o que chamamos de **commanding**.

Eu não estou usando um banco de dados para validar o Email e a Senha; para simplificar estou fazendo uma validação no código bem simples.

A classe **LoginViewModel** implementa a interface **INotifyPropertyChanged** para que as alterações feitas nas propriedades sejam notificadas às Views.

Definimos também o método **ExibirAvisoDeLoginInvalido** usando um delegate **Action**, onde este método irá apenas exibir uma mensagem não retornando nenhum valor.

Com nossa **ViewModel** implementada podemos definir a nossa View.

Selecione a pasta **Pages** e no menu **Project** clique em **Add New Item** e selecione o template **Content Page** informando o nome **LoginPage.xaml.**

A seguir inclua o código abaixo no arquivo **LoginPage.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_Login.Pages.LoginPage"
             BackgroundImage="logo.jpg">
    <ContentPage.Content>
        <StackLayout Orientation="Vertical" Padding="30" Spacing="40">
            <BoxView HeightRequest="10"/>
            <Image HorizontalOptions="Center" WidthRequest="300" Source="maco.jpg"/>
            <Frame BackgroundColor="#BF043055" HasShadow="False">
                <StackLayout Orientation="Vertical" Spacing="10">
                    <Entry x:Name="Email" Text="{Binding Email}" Placeholder="Email"
                           PlaceholderColor="White" HeightRequest="40"
                           Keyboard="Email"
                           TextColor="White"/>
                    <Entry x:Name="Senha" Text="{Binding Senha}" Placeholder="Senha"
                           PlaceholderColor="White" HeightRequest="40"
                           IsPassword="True"
                           TextColor="White"/>
                </StackLayout>
```

```
        </Frame>
        <Button Command="{Binding SubmitCommand}" Text="Login" TextColor="White"
              FontAttributes="Bold" FontSize="Large" HorizontalOptions="FillAndExpand"
              BackgroundColor="#088da5" />
      </StackLayout>
   </ContentPage.Content>
</ContentPage>
```

No arquivo **MainPage.xaml.cs** temos o código a seguir onde definimos :

- A imagem de fundo da página : **BackgroundImage="logo.jpg"**
- A imagem centralizada acima do Frame : <Image HorizontalOptions="Center" WidthRequest="300" Source="maco.jpg"/>
- O campo para o usuário informar o Email usando a view Entry :

```
<Entry x:Name="Email" Text="{Binding Email}" Placeholder="Email"
                  PlaceholderColor="White" HeightRequest="40"
                  Keyboard="Email"
                  TextColor="White"/>
```

- O campo para o usuário informar a Senha usando a view Entry :

```
<Entry x:Name="Senha" Text="{Binding Senha}" Placeholder="Senha"
                  PlaceholderColor="White" HeightRequest="40"
                  IsPassword="True"
                  TextColor="White"/>
```

- O botão de comando :

```
<Button Command="{Binding SubmitCommand}" Text="Login" TextColor="White"
            FontAttributes="Bold" FontSize="Large" HorizontalOptions="FillAndExpand"
            BackgroundColor="#088da5" />
```

Usamos o databinding para vincular o valor das Views usadas com as propriedades **Email,Senha e SubmitCommand** definidas em **LoginViewModel**.

Mas como a nossa view vai se comunicar com a ViewModel ?

Fazemos isso no arquivo code-behind **LoginPage.xaml.cs** definindo o código a seguir:

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using XF_Login.ViewModels;

namespace XF_Login.Pages
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class LoginPage : ContentPage
   {
      public LoginPage()
      {
         var vm = new LoginViewModel();
         this.BindingContext = vm;

         vm.ExibirAvisoDeLoginInvalido += () => DisplayAlert("Erro", "Login Inválido, tente novamente", "OK");

         InitializeComponent();

         Email.Completed += (object sender, EventArgs e) =>
         {
```

```
            Senha.Focus();
        };

        Senha.Completed += (object sender, EventArgs e) =>
        {
            vm.SubmitCommand.Execute(null);
        };

    }
  }
}
```

Neste código estamos criando uma instância de **LoginViewModel** e fazendo a vinculação com a nossa view usando a propriedade **BindingContext**.

Implementamos também o método **ExibirAvisoDeLoginInvalido**() para exibir uma mensagem de erro.

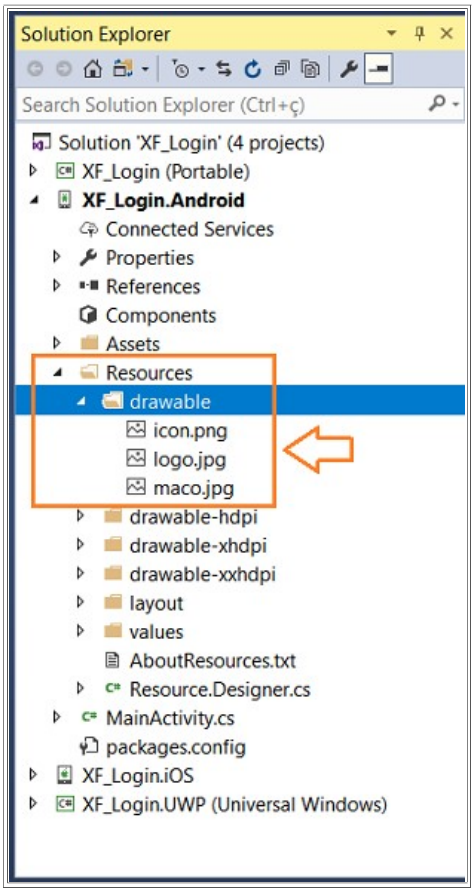
E concluímos tratando os evento **Completed** das Views **Entry** que ocorre quando o usuário finaliza o texto na view com a tecla Return.

Preparando as imagens usadas no projeto

As imagens usadas no projeto devem ser preparadas para serem exibidas em cada plataforma.

A imagem **logo.jpg e maco.jpg**, usadas no projeto possuem o nome e o tamanho apropriedado para serem exibidas no projeto Android que é onde eu vou testar o projeto.

Como estou testando o projeto usando o Android estou copiando as imagens para a pasta **Resources/drawable** do projeto Android:

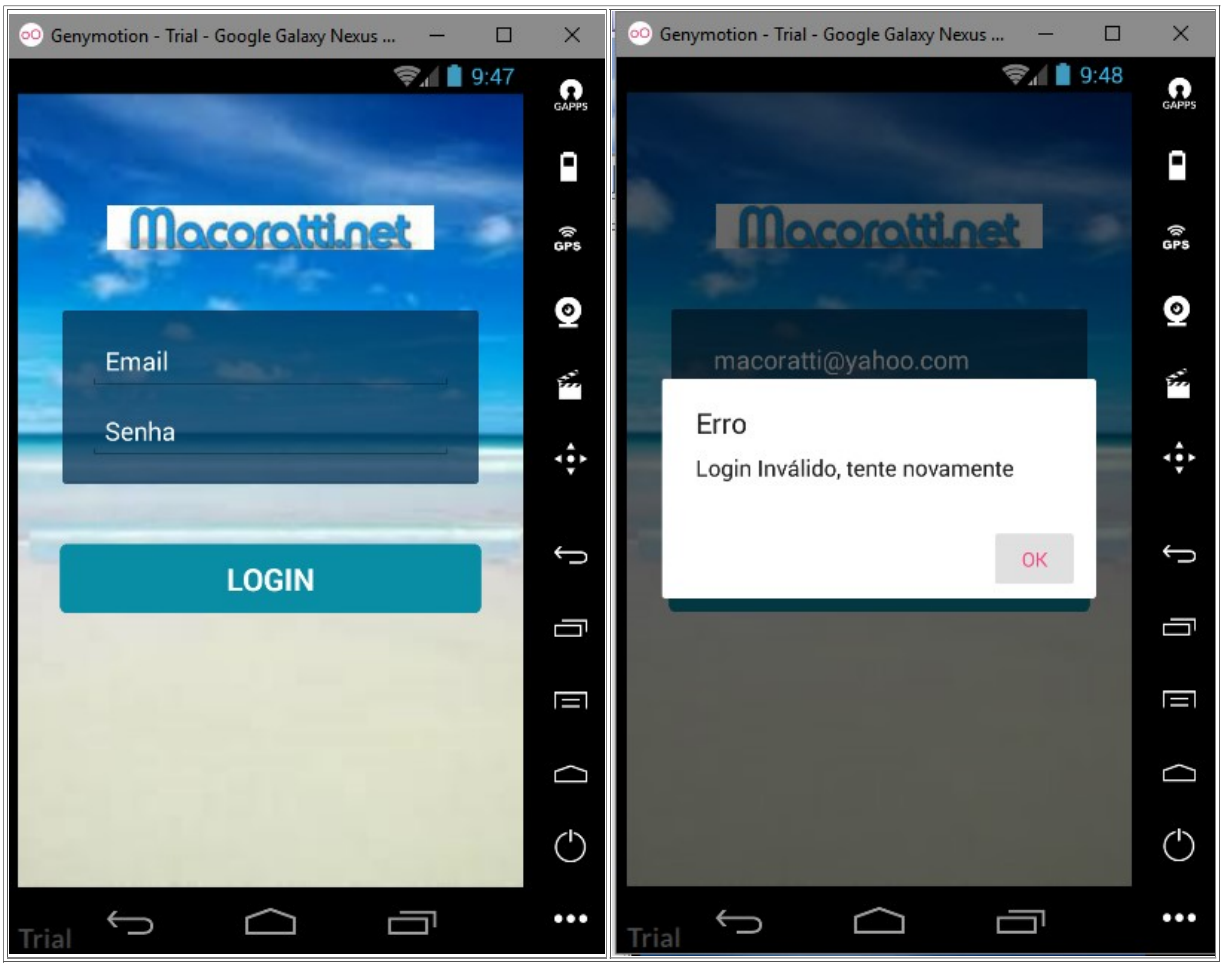


Agora basta alterar o arquivo **App.xaml.cs** do projeto PCL para exibir a página de Login:

```
using System.Text;
```

```
using Xamarin.Forms;

namespace XF_Login
{
   public partial class App : Application
   {
      public App()
      {
         InitializeComponent();

         MainPage = new XF_Login.Pages.LoginPage();
      }

      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
      }
   }
}
```

Executando o projeto iremos obter o seguinte resultado no emulador **Genymotion** para o Android :



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

"(Disse Jesus) - Quem ama a sua vida perdê-la-á, e quem neste mundo odeia a sua vida, guardá-la-á para a vida eterna."
João 12:25

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**
- 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.Widget.ListView/
- https://developer.xamarin.com/api/property/Android.Widget.ListView.Adapter/
- https://developer.xamarin.com/guides/xamarin-forms/user-interface/animation/

José Carlos Macoratti