Xamarin Android - Exibindo diálogos de Alerta com AlertDialog e avisos com Toast

Neste artigo vou mostrar como exibir uma janela de alerta e avisos em uma aplicação **Android** usando Visual Studio com **Xamarin**.

Uma diálogo de alerta é uma parte importante de uma aplicação. Podemos usar os diálogos de alerta para transmitir informações diversas, mensagens de erro e até mesmo para a solicitar confirmações ao usuário.

O **Xamarin** fornece a sua própria maneira de mostrar o alerta e transmitir mensagens para o usuário.

Alertas no Xamarin são objetos da classe **AlertDialog.Builder** onde :

- **AlertDialog** é uma subclasse da classe **Dialog**() que pode exibir até 3 botões. *(Se você só quer exibir uma String nesta caixa de diálogo, use o método **setMessage**()).*
- **Builder** é um método da classe **AlertDialog**() que cria um alerta de diálogo para exibir.

Podemos adicionar múltiplos botões em um alerta de diálogo e o construtor toma o contexto atual (**Activity**) e exibe uma caixa de alerta na thread principal. **(UI Thread).**

O código abaixo mostra um exemplo onde estamos criando uma instância de **AlertDialog**() e a seguir criando uma alerta usando o método **Create**().

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
AlertDialog alerta = builder.Create();
```

Outro recurso para exibir mensagens ao usuário é a classe **Toast**.

A classe **Toast** permite exibir mensagens ao usuário de maneira simples e rápida em uma janela usando uma pequena view.

A janela usa a quantidade de espaço necessária para exibir a mensagem e permanece visível por um tempo determinado.

A mensagem exibida não permite interação com o usuário, seu propósito é apenas emitir uma informação rápida ao usuário.

Abaixo vemos um exemplo onde estamos usando a classe **Toast** e o método **MakeText** :

```
Toast.MakeText(this, 'Macoratti .net' , ToastLength.Long).Show();
```

- **this** : representa o contexto
- A seguir vem o texto a ser exibido : **'Macoratti .net'**
- A enumeração **ToastLength** define o tempo de exibição que pode ser : Long ou Short.

Vejamos isso na prática usando o Visual Studio com Xamarin.

Recursos usados:

- **Visual Studio Community 2015 ou Xamarin Studio**
- **Xamarin**
- **Emulador Android virtual ou físico (veja como emular usando o Vysor)**

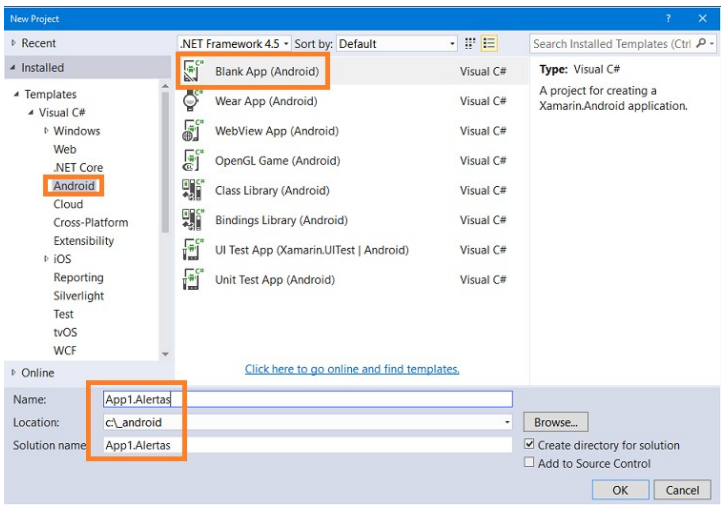
Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.

Criando o projeto no Visual Studio 2015 Community

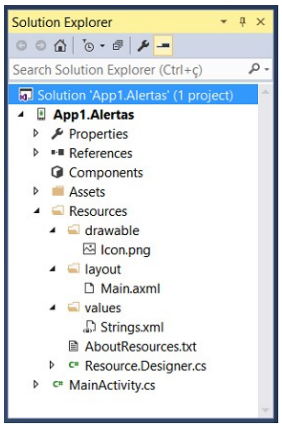
Abra o VS 2015 Community e clique em **New Project;**

Selecione a linguagem Visual C# e o template **Android -> Blank App(Android)**

Informe o nome **App1.Alertas** e clique no botão **OK**;



Será criada uma solução com a seguinte estrutura:



- **Properties** - Contém o arquivo AndroidManifest.xml que descreve as funcionalidades e requisitos da sua aplicação Android, e o arquivo AssemblyInfo.cs contém informação sobre o projeto como número de versão e build.

- **References** - Contém as bibliotecas Mono.Android, System.Core e todas as bibliotecas usadas no seu projeto;

- **Components** - Contém componentes de terceiros ou desenvolvidos por você usados no seu projeto.

A maioria dos componentes está disponíveis diretamente do **Xamarin Component Store** e são **free** *(não todos)* e prontos para serem usados; (Para incluir um componente clique com o botão direito sobre **Components** e a seguir em Get More Components);

- **Assets e Resources** - Contém arquivos que não são código, como imagens, sons, arquivos XML e qualquer outro recurso que sua aplicação for usar. Os arquivos externos colocados na pasta **Assets** são facilmente acessíveis em tempo de execução através do Asset Manager.

Já os arquivos colocados na pasta **Resources** precisam ser declarados e mantidos em uma lista com os IDs dos recursos que você desejar usar em tempo de execução.

De forma geral, todas a imagens, ícones, sons e outros arquivos externos são colocados na pasta **Resources** enquanto que dicionários e arquivos XML são postos na pasta **Assets**;

Na subpasta **layout** temos os arquivos .axml que definem as views usadas no projeto;

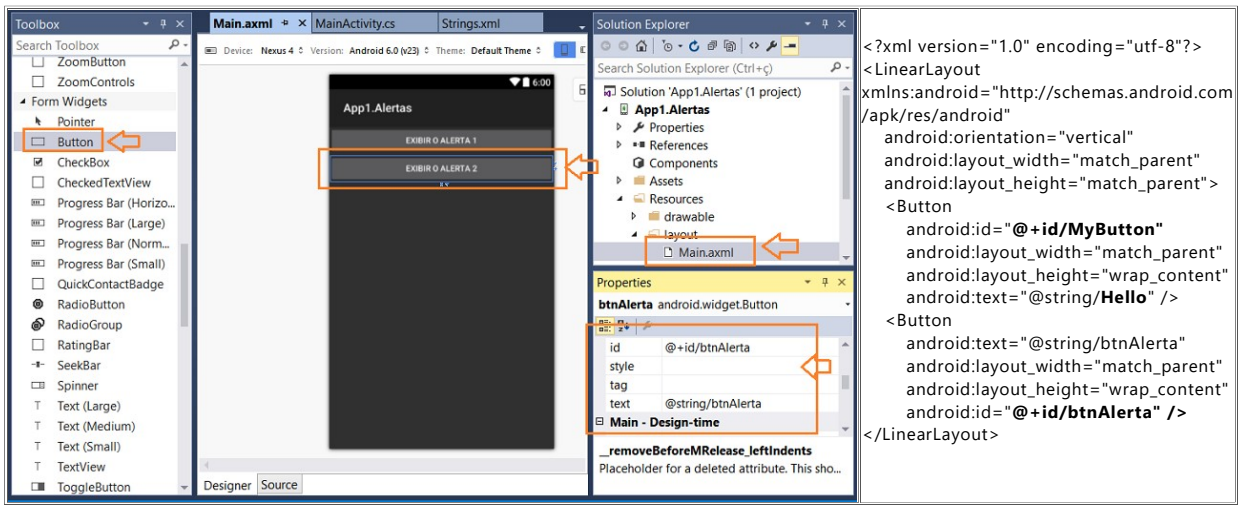
Na subpasta **values** temos o arquivo Strings.xml onde definimos as strings usadas no projeto;

Nota : A pasta **Drawable** contém recursos como imagens png, jpg, etc., usadas no aplicativo. Ela contém múltiplas pastas especificas para cada resolução possível em uma aplicação Android. Numa aplicação típica Android você vai acabar encontrando as pastas: Drawable-LDPI, Drawable-mdpi, Drawable-hdpi, Drawable-xhdpi, Drawable-xxhdpi, etc.

1- Vamos Abrir o arquivo **Main.axml** na pasta **Resources/layout** e no modo **Designer** incluir um novo controle **Button** a partir da **ToolBox** e definir a seguintes propriedades:

- **id = @+id/btnAlerta**
- **text = @string/btnAlerta**

Abaixo vemos o leiaute no emulador do Xamarin exibindo a tela e ao lado o respectivo código XML gerado :




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com
/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <Button
    android:id="@+id/MyButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/Hello" />
  <Button
    android:text="@string/btnAlerta"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/btnAlerta" />
</LinearLayout>
```

Vamos agora abrir o arquivo **Strings.xml** na pasta **Resources/values** e definir o texto que será exibido nos botões de comando:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="Hello">Exibir o Alerta 1 </string>
  <string name="btnAlerta">Exibir o Alerta 2</string>
  <string name="ApplicationName">App1.Alertas</string>
</resources>
```

Agora podemos iniciar a implementação do código para criar as janelas de alertas no arquivo **MainActivity.cs**.

Veja como deve ficar o código do arquivo **MainActivity.cs**: *(O código em azul foi o que incluímos)*

```
using Android.App;
using Android.OS;
using Android.Widget;

namespace App1.Alertas
{
  [Activity(Label = "App1.Alertas", MainLauncher = true, Icon = "@drawable/icon")]
  public class MainActivity : Activity
  {
    int count = 1;

    protected override void OnCreate(Bundle bundle)
    {
      base.OnCreate(bundle);

      SetContentView(Resource.Layout.Main);

      Button button = FindViewById<Button>(Resource.Id.MyButton);
      button.Click += delegate
```

```
        {
            button.Text = string.Format("{0} cliques !", count++);

            //define o alerta para executar a tarefa
            AlertDialog.Builder builder = new AlertDialog.Builder(this);

            AlertDialog alerta = builder.Create();

            //Define o Titulo
            alerta.SetTitle("Macoratti .net - Deseja Continuar");
            alerta.SetIcon(Android.Resource.Drawable.IcDialogAlert);
            alerta.SetMessage("Macoratti .net Quase tudo para a Plataforma .Net");

            alerta.SetButton("OK", (s, ev) =>
            {
                Toast.MakeText(this, "Legal, vamos continuar... !", ToastLength.Short).Show();
            });
            alerta.Show();
        };

        Button btnAlerta = FindViewById<Button>(Resource.Id.btnAlerta);
        btnAlerta.Click += delegate
        {
            //define o alerta para executar a tarefa
            AlertDialog.Builder alerta = new AlertDialog.Builder(this);
            //define o titulo e o ícone a exibir no dialogo
            alerta.SetTitle("Deseja Salvar o trabalho ?");
            alerta.SetIcon(Android.Resource.Drawable.IcInputAdd);
            //define a mensagem
            alerta.SetMessage("Macoratti .net - Quase tudo para a plataforma .NET");
            //define o botão positivo
            alerta.SetPositiveButton("Salvar", (senderAlert, args) =>
            {
                Toast.MakeText(this, "Salvo com sucesso!", ToastLength.Short).Show();
            });
            //define o botão negativo
            alerta.SetNegativeButton("Cancelar", (senderAlert, args) =>
            {
                Toast.MakeText(this, "Cancelado !", ToastLength.Short).Show();
            });
            //cria o alerta e exibe
            Dialog dialog = alerta.Create();
            dialog.Show();
        };
    }
  }
}
```

Perceba no código que para cada Button definimos um evento **Click** e em cada evento criamos o código para exibir as janelas de alerta. *(Eu apresentei duas maneiras diferentes de criar e definir os alertas.)*

Assim, em cada evento **Click** dos botões **MyButton e btnAlerta** definimos o código :

1- Para criar uma instância da classe **AlertDialog.Builder**

AlertDialog.Builder builder = new AlertDialog.Builder(this); e AlertDialog.Builder alerta = new AlertDialog.Builder(this);

3- Definimos o título usando o método **SetTittle**() e os ícones : **IcDialogAlert e IcInputAdd,** usando o método **SetIcon()**;

4- Definimos a mensagem usando o método **SetMessage**()

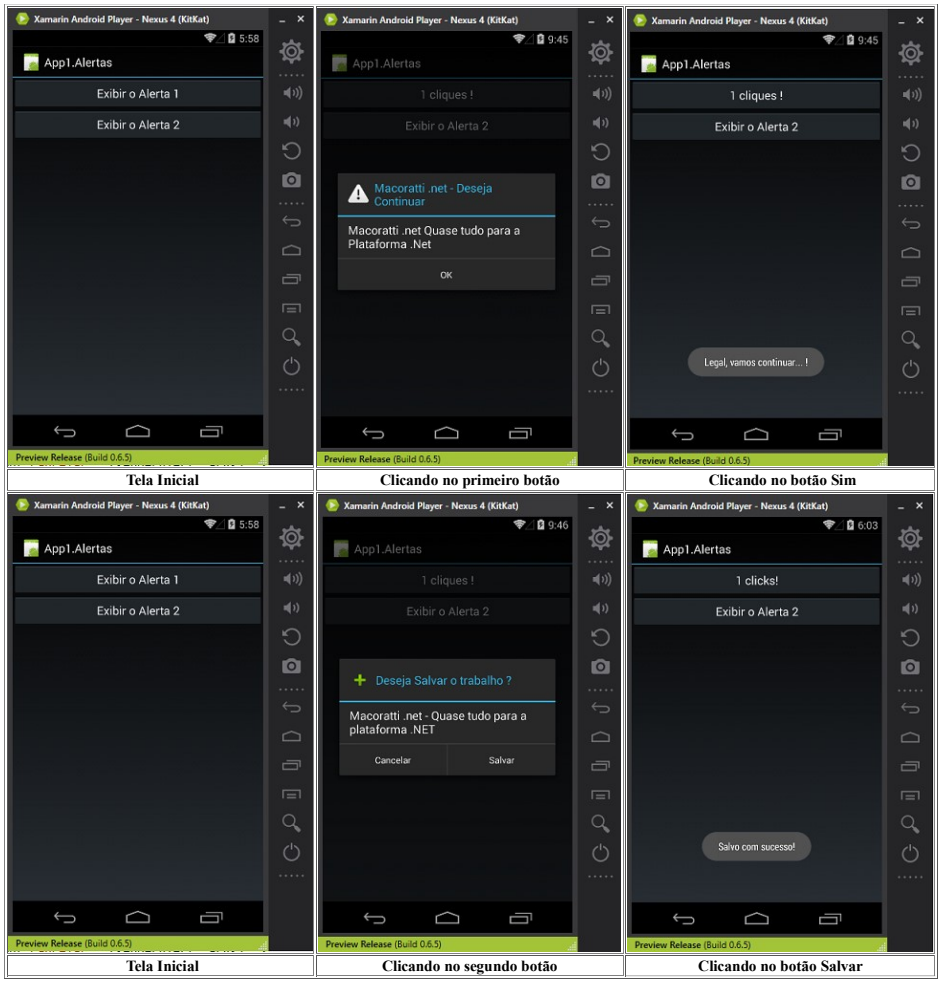
5- Para o segundo botão definimos o primeiro botão usando - **SetPositiveButton**() e o segundo botão usando - **SetNegativeButton**()

6- Exibimos o alerta : **alerta.Show**

Estamos usando também o recurso do método **Toast.MakeText** que cria uma pequena mensagem de retorno da operação exibindo um texto em um pequeno *pop up* durante um certo tempo.

Ex: **Toast.MakeText(this, "Cancelado !", ToastLength.Short).Show();**

Executando o projeto iremos obter o seguinte resultado:

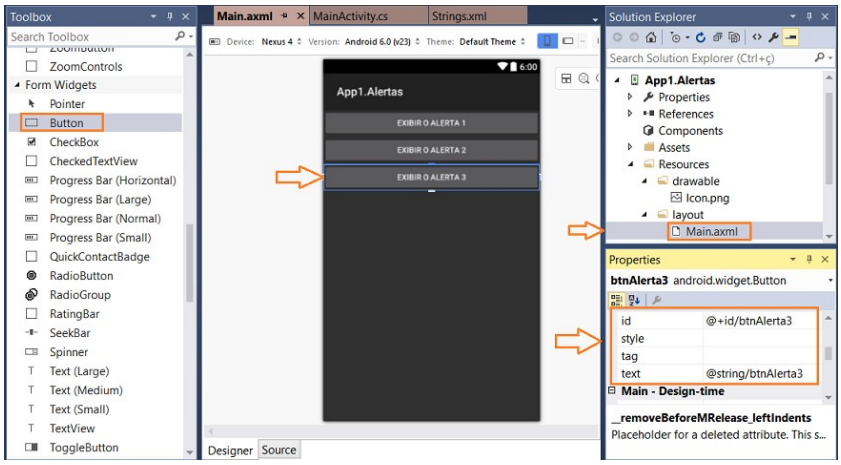


Como eu já mencionei podemos definir até 3 botões no diálogo de alerta então vamos incluir mais um Button a partir da **ToolBox** no arquivo **Main.axml** :

Vamos Abrir o arquivo **Main.axml** na pasta **Resources/layout** e no modo **Designer** incluir um novo controle **Button** a partir da **ToolBox** e definir a seguintes propriedades:

- **id = @+id/btnAlerta3**
- **text = @string/btnAlerta3**

Abaixo vemos o leiaute no emulador do Xamarin exibindo a tela e ao lado o respectivo código XML gerado :




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <Button
    android:id="@+id/MyButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/Hello" />
  <Button
    android:text="@string/btnAlerta"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/btnAlerta" />
  <Button
    android:text="@string/btnAlerta3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/btnAlerta3" />
</LinearLayout>
```

Vamos agora abrir o arquivo **Strings.xml** na pasta **Resources/values** e definir o texto que será exibido no terceiro botão :

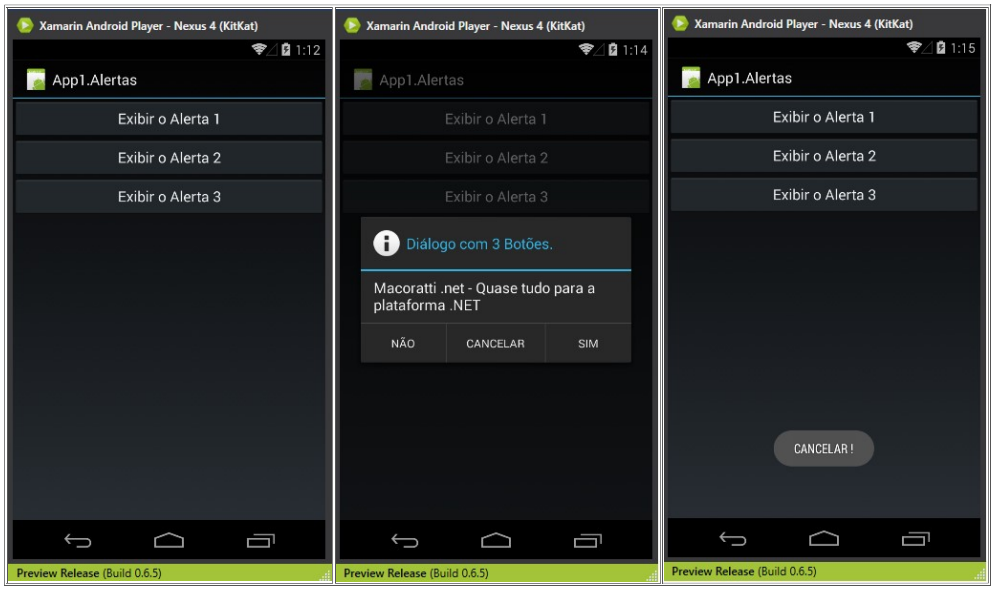
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="Hello">Exibir o Alerta 1 </string>
  <string name="btnAlerta">Exibir o Alerta 2</string>
  <string name="btnAlerta3">Exibir o Alerta 3</string>
  <string name="ApplicationName">App1.Alertas</string>
</resources>
```

Agora podemos vamos incluir o seguinte código no arquivo **MainActivity.cs** no interior do método **OnCreate():**

```
Button btnAlerta3 = FindViewById<Button>(Resource.Id.btnAlerta3);
btnAlerta3.Click += delegate
{
   //define o alerta para executar a tarefa
   AlertDialog.Builder builder = new AlertDialog.Builder(this);
   AlertDialog alerta = builder.Create();
   alerta.SetCancelable(true);
   alerta.SetIcon(Android.Resource.Drawable.IcDialogInfo);
   //define o titulo
   alerta.SetTitle("Diálogo com 3 Botões.");

   //define a mensagem
   alerta.SetMessage("Macoratti .net - Quase tudo para a plataforma .NET");

   //define os 3 botões
   alerta.SetButton("SIM", (s, ev) =>
   {
      Toast.MakeText(this, "SIM !", ToastLength.Short).Show();
   });
   alerta.SetButton2("NÃO", (s, ev) =>
   {
      Toast.MakeText(this, "NÃO !", ToastLength.Short).Show();
   });
   alerta.SetButton3("CANCELAR", (s, ev) =>
   {
      Toast.MakeText(this, "CANCELAR !", ToastLength.Short).Show();
   });

   alerta.Show();
};
```

Nota: **SetCancelable(true**) define se o diálogo pode ser cancelado pela tecla Android.Views.KeyEvent.KEYCODE_BACK.

Executando novamente o projeto e clicando no terceiro botão e depois no botão cancelar do diálogo de alerta teremos :



Aguarde mais artigos sobre a criação de aplicações com Xamarin.Android.

Pegue o projeto completo aqui : **App1.Alertas.zip** (*sem as referências*)

Segui a paz com todos, e a santificação, sem a qual ninguém verá o Senhor;
Hebreus 12:14

- **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

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.App.AlertDialog/

José Carlos Macoratti