Macoratti.net Xamarin Android - Enviando mensagens SMS

Neste artigo vou mostrar com podemos fazer enviar mensagens SMS diretamente de uma aplicação Android usando o Xamarin Android, o Visual Studio 2015 e a linguagem C#.

Existem duas opções para enviar mensagens SMS no Android:

1- Utilizar a classe **SmsManager** para enviar mensagens em segundo plano. Para isso use uma Intent para enviar o usuário para o aplicativo SMS com um número predefinido e a mensagem.

***Nota**: A classe SmsManager gerencia operações SMS como envio de dados, texto e mensagens.*

Nesta abordagem você precisa adicionar a permissão **SEND_SMS** ao manifesto do Android : **[assembly: UsesPermission(Manifest.Permission.SendSms)] ,** e usar o método **SendTextMessage** da classe **SmsManager**.

A classe **SmsManager** esta incluida no namespace **Android.Telephony** fornece APIs para monitorar as informações básicas do telefone, como o tipo de rede e o estado da conexão, além de utilitários para manipulação de cadeias de números de telefone.

2 - Podemos também enviar SMS criando uma **Intent** com uma ação **ActionSendTo** e uma **Uri** que começa com **smsto:** . Abaixo temos um trecho de código que podemos usar nessa abordagem:

```
var smsUri = Android.Net.Uri.Parse("smsto:1234567890");
var smsIntent = new Intent (Intent.ActionSendto, smsUri);
smsIntent.PutExtra ("sms_body", "Enviando um SMS teste no Xamarin.Android");
StartActivity (smsIntent);
```

Neste artigo eu vou apresentar um exemplo que usa a primeira abordagem.

Antes de realizar a operação, podemos verificar se o dispositivo suporta o envio de SMS usando o método **HasSystemFeature**() em tempo de execução:

```
if (!PackageManager.HasSystemFeature(Android.Content.PM.PackageManager.FeatureTelephony))
{
   Toast.MakeText(this, "Este dispositivo não possui recursos para enviar SMS", ToastLength.Short);
   return;
}
```

Vamos então aplicar esses conceitos na prática.

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 VS Community 2015

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

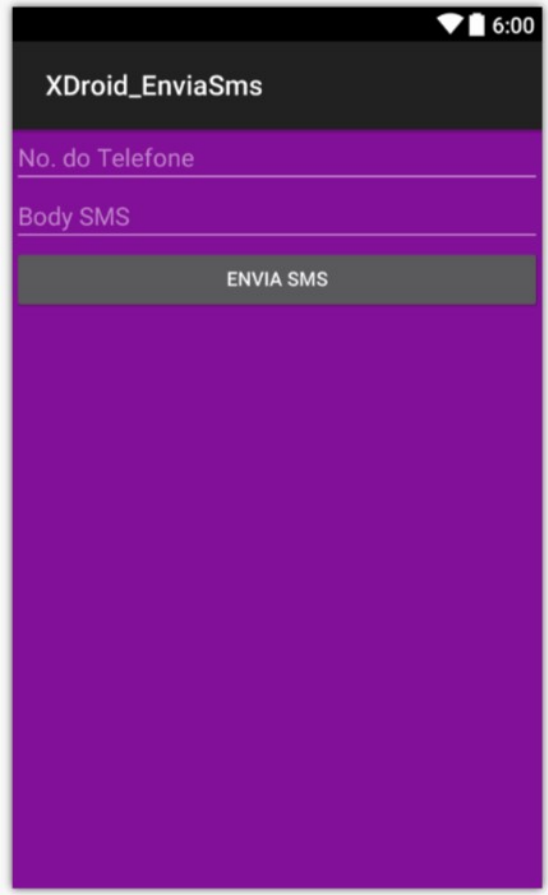
Selecione a linguagem Visual C# e o template **Android -> Blank App(Android)**

Informe o nome um nome adequado ao seu projeto, eu vou usar o nome **XDroid_EnviaSms**, e clique no botão **OK**;

Abra o arquivo **Main.axml** na pasta **Resources/layout** no modo **Designer** e a seguir inclua uma caixa de texto usando o controle **EditText** e um **Button**.

- 1 EditText - inputType="phone", hint="@string/foneTexto", id=**"@+id/foneTexto"**
- 1 EditText - hint="@string/editaTexto", id=**"@+id/editaTexto"**
- 1 Button - id=@**+id/enviaSms,** android:text="@string/enviaSms"

Abaixo vemos o leiaute no emulador do Xamarin 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:background="#831098"
   android:layout_height="match_parent">
   <EditText
      android:inputType="phone"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/foneTexto"
      android:phoneNumber="true"
      android:textColor="#95e640"
      android:hint="@string/foneTexto" />
   <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textColor="#95e640"
      android:id="@+id/editaTexto"
      android:hint="@string/editaTexto" />
   <Button
      android:text="@string/enviaSms"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/enviaSms" />
</LinearLayout>
```

A seguir abra o arquivo **strings.xml** na pasta **Resources/values** e inclua o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="ApplicationName">XDroid_EnviaSms</string>
   <string name="foneTexto">No. do Telefone</string>
   <string name="editaTexto">Body SMS</string>
   <string name="enviaSms">Envia SMS</string>
</resources>
```

Agora estamos pronto para definir o código da atividade principal.

Definindo o código da MainActivity

Abra o arquivo **MainActivity** e inclua o código abaixo :

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

namespace XDroid_EnviaSms
{
   [Activity(Label = "XDroid_EnviaSms", MainLauncher = true, Icon = "@drawable/icon")]
   public class MainActivity : Activity
   {
      private SmsSentReceiver receiver;
      protected override void OnCreate(Bundle bundle)
      {
         base.OnCreate(bundle);
         // Set our view from the "main" layout resource
         SetContentView(Resource.Layout.Main);
```

```
        EditText editaTexto = FindViewById<EditText>(Resource.Id.editaTexto);
        EditText foneTexto = FindViewById<EditText>(Resource.Id.foneTexto);
        Button enviaSms = FindViewById<Button>(Resource.Id.enviaSms);

        if (!PackageManager.HasSystemFeature(Android.Content.PM.PackageManager.FeatureTelephony))
        {
            Toast.MakeText(this, "Este dispositivo não possui recursos para enviar SMS", ToastLength.Short);
            return;
        }

        enviaSms.Click += delegate
        {
            var sentIntent = new Intent(SmsSentReceiver.SentIntent);
            var sent = PendingIntent.GetBroadcast(this, 0, sentIntent, 0);
            var deliveredIntent = new Intent(SmsSentReceiver.DeliveredIntent);
            var delivered = PendingIntent.GetBroadcast(this, 0, deliveredIntent, 0);

            var manager = SmsManager.Default; //obtém a instância de SmsManager
            manager.SendTextMessage(foneTexto.Text, null, editaTexto.Text, sent, delivered);
        };
    }

    protected override void OnResume()
    {
        base.OnResume();
        receiver = new SmsSentReceiver();

        // A classe RegisterReceiver faz o registro para receber intent transmitidas
        // para serem executadas no contexto do agendador.
        // Ela Isso permite que você imponha permissões sobre quem pode transmitir intenções
               //  para o seu receptor ou ter o receptor executado em um segmento diferente do thread
              //   principal do aplicativo.
        RegisterReceiver(receiver, new IntentFilter(SmsSentReceiver.SentIntent));
        RegisterReceiver(receiver, new IntentFilter(SmsSentReceiver.DeliveredIntent));
    }

    protected override void OnPause()
    {
        base.OnPause();
       // Desfaz o registro de um BroadcastReceiver() registrado
        UnregisterReceiver(receiver);
    }

    // A classe BroadcastReceiver é a classe base para receber Intents
    // que foram enviadas via sendBroadcast()
    private class SmsSentReceiver : BroadcastReceiver
    {
        public const string SentIntent = "SentSMS";
        public const string DeliveredIntent = "DeliveredSMS";

        public override void OnReceive(Context context, Intent intent)
        {
            if (ResultCode == Result.Ok)
            {

              // Intent.Action retorna a ação realizada
               if (intent.Action == SentIntent)
               {
                   Toast.MakeText(context, "SMS Enviado.", ToastLength.Short).Show();
               }
               else if (intent.Action == DeliveredIntent)
               {
                   Toast.MakeText(context, "SMS despachado.", ToastLength.Short).Show();
               }
            }
            else
```

```
            {
                if (intent.Action == SentIntent)
                {
                    Toast.MakeText(context, "Falha ao enviar SMS.", ToastLength.Short).Show();
                }
                else if (intent.Action == DeliveredIntent)
                {
                    Toast.MakeText(context, "Falha ao encaminhar SMS.", ToastLength.Short).Show();
                }
            }
        }
    }
  }
}
```

Vamos entender o código : (*os comentários já explicam detalhes das classes principais usadas*)

Para enviar mensagens SMS, primeiro precisamos solicitar a permissão **SendSms**. Isso é necessário porque enviar uma mensagem SMS envolve um custo para o usuário.

Assim que tivermos a permissão, podemos então obter uma instância da classe **SmsManager**. Em seguida, passamos o número de destino e a mensagem para o método **SendTextMessage().**

Para confirmar que uma mensagem foi entregue a um destinatário, uma **Intent** pode ser fornecida ao método **SendTextMessage**(). Quando a a mensagem for entregue, a intenção será transmitida.

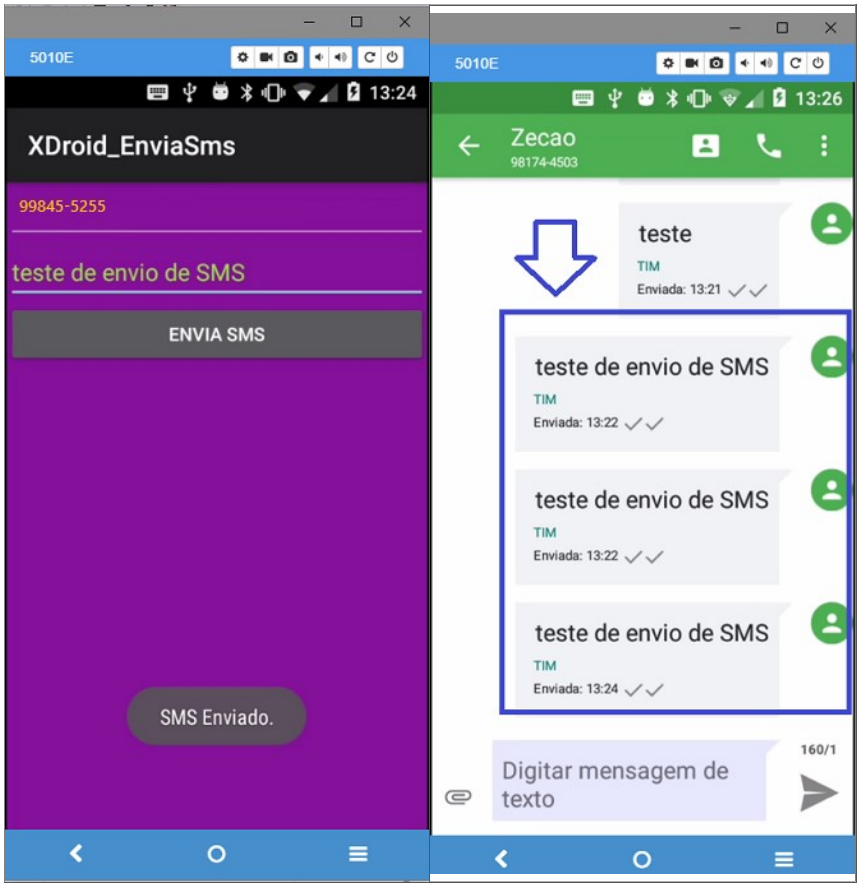
O penúltimo parâmetro do método **SendTextMessage**() é uma instância **PendingIntent** que será transmitida se a mensagem for enviada com êxito ou se houver um erro no envio.

O retorno de **ResultCode** será Ok se a operação for bem-sucedida, ou outro valor se houver um erro.

Para confirmar que a mensagem SMS foi entregue ao destinatário, fornecemos uma instância **PendingIntent** para o último parâmetro. Esta **Intent** será transmitida quando a mensagem for entregue. Se a mensagem for muito longa para uma única mensagem, podemos usar o método **DivideMessage**() na instância **SmsManager** para quebrar a mensagem em fragmentos.

Os fragmentos são enviados para um destinatário usando o método **SendMultipartTextMessage**(). Passamos a coleção de fragmentos juntamente com uma coleção de intenções pendentes que devem ser transmitidas como cada fragmento é enviado e entregue.

Executando o projeto usando o **Vysor** para emular um dispostivo físico - um ALCATEL PIXI 4 - iremos obter o seguinte resultado: *(o comportamento pode variar em outros dispositivos)*



Na primeira tela informamos o número de celular para onde vamos enviar a mensagem e a seguir o texto da mensagem.

A clicar no botão - **ENVIA SMS** - teremos a mensagem - **SMS Enviado**, indicando que a mensagem foi enviada. *(No exemplo estou usando um dispositivo físico sem créditos para enviar mensagem)*

Podemos verificar no dispostivo as mensagens enviadas *(que foram represadas por falta de crédito)*.

Assim vimos que enviar SMS é bem simples usando o Xamarin Anroid. Fique a vontade para implementar outros recursos para personalizar o projeto.

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

Mas, ó homem, quem és tu, que a Deus replicas? Porventura a coisa formada dirá ao que a formou: Por que me fizeste assim?
Ou não tem o oleiro poder sobre o barro, para da mesma massa fazer um vaso para honra e outro para desonra?
Romanos 9:20,21

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

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