Macoratti.net Xamarin Android - Usando o serviço de Alarme

Neste artigo vou mostrar como usar o serviço de Alarme do Android em uma aplicação Xamarin Android usando o VS 2017 e a linguagem C#.

A classe **AlarmManager** fornece acesso aos serviços de alarme do sistema. Estes serviços permitem que você agende sua aplicação para ser executada em algum ponto no futuro. Quando um alarme dispara, a Intent que tinha sido registrada para ele é transmitida pelo sistema, iniciando automaticamente o aplicativo de destino se ele ainda não estiver em execução.

Os alarmes registrados são mantidos enquanto o dispositivo está dormindo *(e pode, opcionalmente, despertar o dispositivo se eles se apagam durante esse tempo)*, mas será desmarcado se estiver desligado e reiniciado.

O **Alarm Manager** ou *Gerenciador de Alarmes* mantém um bloqueio de vigília da CPU enquanto o método **onReceive**() do receptor de alarme está sendo executado. Isso garante que o telefone não irá dormir até que você tenha terminado de lidar com a transmissão. Uma vez que **onReceive**() retorna, o **Alarm Manager** lança este bloqueio de vigília. Isso significa que o telefone em alguns casos irá dormir assim que o método onReceive() for concluído.

Além da classe **AlarmManager** vamos associar o serviço de alarme com o componente **BroadCast Receiver** do android, assim o serviço irá invocar este receiver na hora agendada.

Um **broadcast receiver** ou receptor de difusão é um componente do Android que permite que um aplicativo responda a mensagens *(uma Intenção do Android)* que são transmitidas pelo sistema operacional Android ou por um aplicativo. As transmissões seguem um modelo de *publicação-inscrição* - um evento faz com que uma transmissão seja publicada e recebida pelos componentes que estão interessados no evento.

Estamos usando **Intents** que é um conceito abstrato para algum tipo de operação que deverá ser executada no sistema operacional Android. **Intents** ou **Intenções** no Android, são estruturas de dados que são objetos de mensagens. Intenções podem solicitar uma operação a ser realizada por algum outro componente no Android e são geralmente usadas para iniciar **Atividades** e **Serviços.**

Também estamos o conceito de **PendingIntent** que é um token que você dá a um aplicativo externo (*por exemplo, NotificationManager, AlarmManager,etc.*), o que permite que o aplicativo use as permissões de seu aplicativo para executar um trecho de código predefinido.

Neste artigo vamos criar um exemplo bem simples onde vamos agendar Notificações e Mensagens usando o **Alarm Manager e o BroadCast.**

Recursos usados:

- **Visual Studio Community 2017 ou Xamarin Studio**
- **Xamarin**

Nota: Baixe e use a versão Community 2017 do VS ela é grátis e é equivalente a versão Professional.

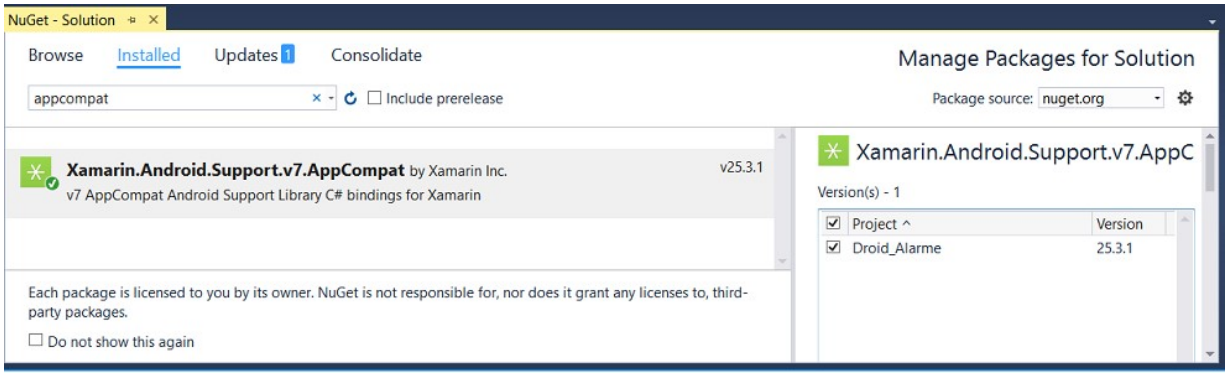
Criando o projeto no Visual Studio 2017 Community

Abra o Visual Studio Community 2017 e clique em **New Project;**

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

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

A seguir vamos Incluir o **pacote Xamarin.Android.Support.v7.app.AppCompat** no projeto via menu **Tools ->Nuget Package Manager -> Manage Nuget Packages for Solution;**

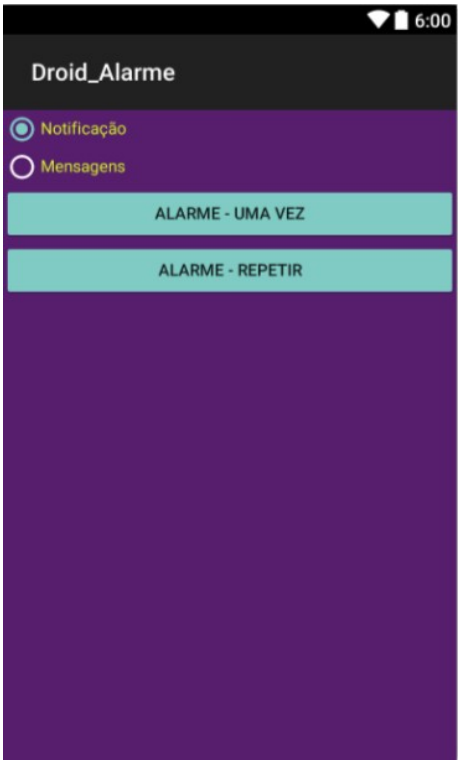


Após isso, abra o arquivo **Main.axml** na pasta **Resources/layout** e no modo **Designer.**

A seguir inclua os seguintes controles a partir da **ToolBox**:

- **1 TextView - id = txtvTitulo**
- **1 CheckBox - id = chkAtivar**
- **1 Button - id = btnInicia**
- **1 Button - id = btnCancela**
- **1 TextView - id = txtvContador**

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




```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:background="#561e6d"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <RadioGroup
      android:id="@+id/rdbGrupo"
      android:layout_width="match_parent"
      android:layout_height="wrap_content">
      <RadioButton
         android:id="@+id/rdbNotificacao"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:checked="true"
         android:textColor="#daff22"
         android:text="Notificação" />
      <RadioButton
         android:id="@+id/rdbToast"
         android:textColor="#daff22"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Mensagens" />
   </RadioGroup>
   <Button
      android:layout_below="@id/rdbGrupo"
      android:id="@+id/btnUmaVez"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Alarme - Uma vez"
      style="@style/Widget.AppCompat.Button.Colored"
      />
   <Button
      android:layout_below="@id/btnUmaVez"
      android:id="@+id/btnRepetir"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Alarme - Repetir"
      style="@style/Widget.AppCompat.Button.Colored"
      />
</RelativeLayout>
```

A seguir vamos criar uma pasta no projeto chamada **BroadCast** onde vamos criar duas classes:

1. **AlarmeMensagemReceiver** - Envia Mensagens usando a classe **Toast**;
2. **AlarmeNotificacaoReceiver** - Envia Mensagens do tipo Notificações;

Esta classes herdam da classe **BroadCastReceiver** e implementam o método **OnReceive**().

Selecione o nome do projeto e no menu **Project** clique em **New Folder** e informe o nome **BroadCast**.

A seguir clique com o botão direito do mouse sobre a pasta **BroadCast** e a seguir clique em **Add -> Class,** e informe o nome **AlarmeMensagemReceiver**.

A seguir inclua o código abaixo nesta classe que irá criar uma mensagem usando a classe **Toast**:

```
using Android.Content;
using Android.Widget;

namespace Droid_Alarme.BroadCast
{
```

```
    [BroadcastReceiver(Enabled = true)]
    public class AlarmeMensagemReceiver : BroadcastReceiver
    {
        public override void OnReceive(Context context, Intent intent)
        {
            Toast.MakeText(context, "Alarme do Macoratti", ToastLength.Long).Show();
        }
    }
}
```

Repita o procedimento acima e crie a classe **AlarmeNotificacaoReceiver** na mesma pasta e a seguir inclua o código abaixo nesta classe que cria uma notificação :

```
using Android.App;
using Android.Content;
using Android.Support.V7.App;

namespace Droid_Alarme.BroadCast
{
    [BroadcastReceiver(Enabled = true)]
    class AlarmeNotificacaoReceiver : BroadcastReceiver
    {
        public override void OnReceive(Context context, Intent intent)
        {
            //cria uma notificação
            NotificationCompat.Builder builder = new NotificationCompat.Builder(context);

            //configura as propriedades do objeto notification
            builder.SetAutoCancel(true)
                .SetDefaults((int)NotificationDefaults.All)         //define quais opções padrão de notificação serão usadas
                .SetSmallIcon(Resource.Drawable.Icon)              //define o ícone small para ser usado na notificação
                .SetContentTitle("Alarme Ativado")                 //define a primeira linha de texto da notificação
                .SetContentText("Esta é minha notificação")    // define a segunda linha de texto da notificação
                .SetContentInfo("Info");

            NotificationManager manager = (NotificationManager)context.GetSystemService(Context.NotificationService);
            //exibe a notificação na barra
            manager.Notify(1, builder.Build());
        }
    }
}
```

Agora vamos definir o código no arquivo **MainActivity.cs** vinculado a nossa view **Main.axml**.

Abra o arquivo **MainActivity.cs** e altere o código desse arquivo conforme abaixo:

```
using Android.App;
using Android.Content;
using Android.OS;
using Android.Widget;
using Droid_Alarme.BroadCast;

namespace Droid_Alarme
{
    [Activity(Label = "Droid_Alarme", MainLauncher = true, Icon = "@drawable/icon", Theme ="@style/Theme.AppCompat.Light.NoActionBar")]
    public class MainActivity : Activity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            // Set our view from the "main" layout resource
            SetContentView (Resource.Layout.Main);

            var rdbNotificacao = FindViewById<RadioButton>(Resource.Id.rdbNotificacao);
            var rdbMensagem = FindViewById<RadioButton>(Resource.Id.rdbToast);

            var btnUmaVez = FindViewById<Button>(Resource.Id.btnUmaVez);
            var btnRepetir = FindViewById<Button>(Resource.Id.btnRepetir);

            btnUmaVez.Click += delegate
            {
                if (rdbNotificacao.Checked == true)
                    IniciarAlarme(true, false);
```

```
            else
                IniciarAlarme(false, false);
        };

        btnRepetir.Click += delegate
        {
            if (rdbNotificacao.Checked == true)
                IniciarAlarme(true, true);
            else
                IniciarAlarme(false, true);
        };
    }

    private void IniciarAlarme(bool isNotificacao, bool isRepetirAlarme)
    {
        AlarmManager manager = (AlarmManager)GetSystemService(Android.Content.Context.AlarmService);
        Intent minhaIntent;
        PendingIntent pendingIntent;

        if(!isNotificacao)
        {
            minhaIntent = new Intent(this, typeof(AlarmeMensagemReceiver));
            pendingIntent = PendingIntent.GetBroadcast(this, 0, minhaIntent, 0);
        }
        else
        {
            minhaIntent = new Intent(this, typeof(AlarmeNotificacaoReceiver));
            pendingIntent = PendingIntent.GetBroadcast(this, 0, minhaIntent, 0);
        }

        if(!isRepetirAlarme)
        {
            manager.Set(AlarmType.RtcWakeup, SystemClock.ElapsedRealtime() + 3000, pendingIntent);
        }
        else
        {
            manager.SetRepeating(AlarmType.RtcWakeup, SystemClock.ElapsedRealtime() + 3000, 60 * 1000, pendingIntent);
        }
    }
  }
}
```

Vamos entender o código :

1- Aplicamos um **Theme** no arquivo **MainActivity** para definir uma aparência visual aos controles padrões usados nas páginas;

```
[Activity(Label = "Droid_Alarme", MainLauncher = true, Icon = "@drawable/icon", Theme
="@style/Theme.AppCompat.Light.NoActionBar")]
```

2- No evento **Click** dos botões de comando - **btnUmaVez/btnRepetir** - definimos o código que vai verificar se o a mensagem será enviada uma vez ou será repetida e chamamos o método **IniciarAlarme**() com os parâmetros pertinentes:

```
btnUmaVez.Click += delegate
{
    if (rdbNotificacao.Checked == true)
        IniciarAlarme(true, false);
    else
        IniciarAlarme(false, false);
};

btnRepetir.Click += delegate
{
    if (rdbNotificacao.Checked == true)
        IniciarAlarme(true, true);
    else
        IniciarAlarme(false, true);
};
```

3 - No código do método **IniciarAlarme()** cria um gerenciador de alarme usando o serviço de Alarme do dispositivo e define se vamos chamar o método **AlarmeMensagemReceiver** ou o método **AlarmeNotificacaoReceiver** usando o método **GetBroadCast** que retorna um **PendingIntent** que irá realizar a difusão da mensagem.

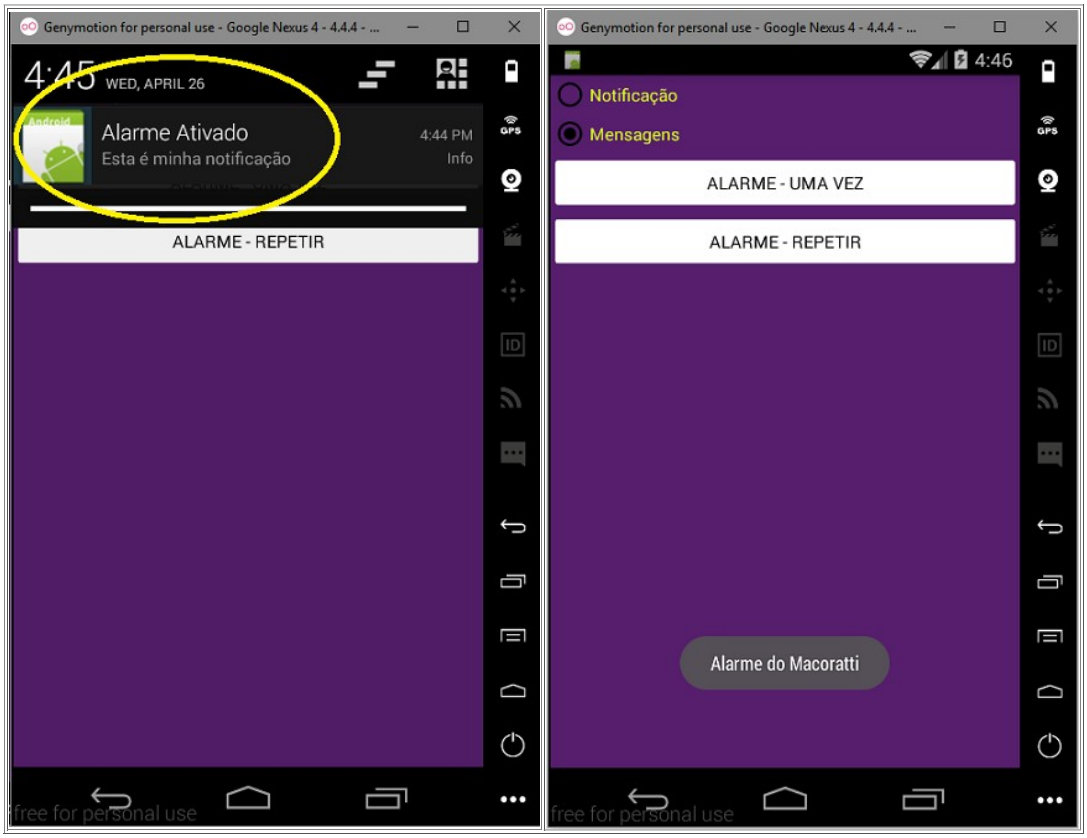
```
AlarmManager manager = (AlarmManager)GetSystemService(Android.Content.Context.AlarmService);
 Intent minhaIntent;
 PendingIntent pendingIntent;

 if(!isNotificacao)
 {
    minhaIntent = new Intent(this, typeof(AlarmeMensagemReceiver));
    pendingIntent = PendingIntent.GetBroadcast(this, 0, minhaIntent, 0);
 }
 else
 {
    minhaIntent = new Intent(this, typeof(AlarmeNotificacaoReceiver));
    pendingIntent = PendingIntent.GetBroadcast(this, 0, minhaIntent, 0);
 }
```

Depois verificamos se o alarme será enviado apenas uma vez ou será repetido e definimos o tipo de alarme:

```
if(!isRepetirAlarme)
 {
    manager.Set(AlarmType.RtcWakeup, SystemClock.ElapsedRealtime() + 3000, pendingIntent);
 }
 else
 {
    manager.SetRepeating(AlarmType.RtcWakeup, SystemClock.ElapsedRealtime() + 3000, 60 * 1000, pendingIntent);
 }
```

Executando o projeto usando o emulador do **Xamarin Android Player** e emulando o **Genymotion** iremos obter o seguinte resultado:

Na primeira figura ativamos o envido da mensagem de notificação e na segunda a mensagem do tipo Toast:



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

Sabendo, amados irmãos, que a vossa eleição é de Deus;Porque o nosso evangelho não foi a vós somente em palavras, mas também em poder, e no Espírito Santo, e em muita certeza, como bem sabeis quais fomos entre vós, por amor de vós.
1 Tessalonicenses 1:4,5

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