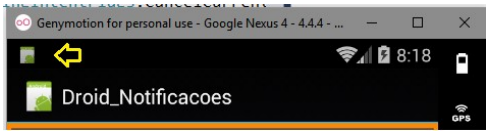
Macoratti.net Xamarin Android - Enviando notificações Locais

Este artigo mostra como podemos enviar notificações locais no Xamarin Android usando a linguagem C# e o Visual Studio 2017.

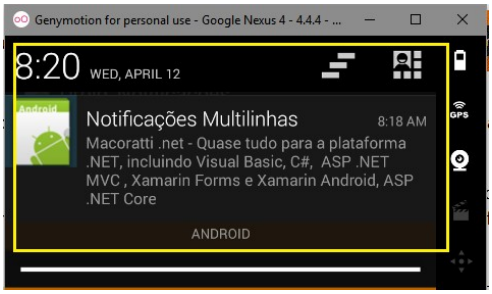
Aplicativos móveis usam notificações como uma maneira discreta de informar o usuário que algum evento específico do aplicativo aconteceu. As notificações são normalmente utilizadas para notificar o usuário do estado de um processo de aplicação em execução em segundo plano.

Um exemplo disso pode ser o download de um arquivo grande. Este arquivo pode levar muito tempo para ser baixado, portanto, esta atividade deve ocorrer em segundo plano. Quando o download é concluído, o usuário é informado do fato por uma notificação. Além disso, as notificações não se limitam apenas a aplicativos locais. Também é possível para aplicativos de servidor publicar notificações para aplicativos móveis.

O Android fornece duas áreas controladas pelo sistema para exibir ícones de notificação e informações de notificação ao usuário. Quando uma notificação é publicada pela primeira vez, seu ícone é exibido na área de notificação, conforme mostrado na seguinte captura de tela:



Para obter detalhes sobre a notificação, o usuário pode abrir a gaveta de notificações (*que expande cada ícone de notificação para revelar o conteúdo da notificação*) e executar quaisquer ações associadas às notificações. A captura de tela a seguir mostra uma gaveta de notificação que corresponde à área de notificação exibida acima:



As notificações do Android utilizam dois tipos de layouts:

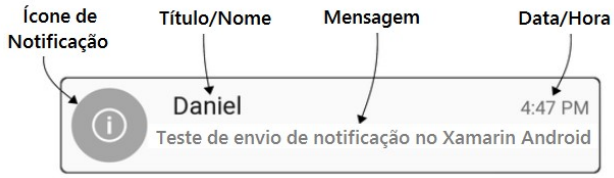
1. **Layout básico** - um formato de apresentação compacto e fixo.
2. **Layout expandido** - um formato de apresentação que pode se expandir para um tamanho maior para revelar mais informações.

1 - Layout básico

Todas as notificações do Android são construídas no formato de layout básico, que inclui, no mínimo, os seguintes elementos:

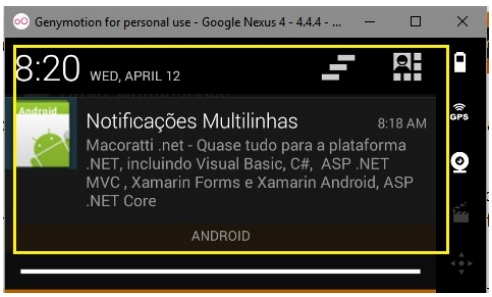
- Um ícone de notificação, que representa o aplicativo de origem ou o tipo de notificação, se o aplicativo oferecer suporte a diferentes tipos de notificações.
- O título da notificação ou o nome do remetente se a notificação for uma mensagem pessoal.
- A mensagem de notificação.
- Um carimbo de data/hora.

Esses elementos são exibidos conforme ilustrado no diagrama a seguir:



2 - Layout Expandido

A partir do Android 4.1, as notificações podem ser configuradas com estilos de layout expandidos que permitem ao usuário expandir a altura da notificação para exibir mais conteúdo. Quando a notificação é expandida ela revela a mensagem completa:



O Android suporta três estilos de layout expandidos para notificações de evento único:

1. **Big Text** - No modo retraído, exibe um trecho da primeira linha da mensagem seguido por dois pontos. No modo expandido, exibe toda a mensagem *(como visto no exemplo acima);*
2. **Inbox** - No modo retraído, exibe o número de novas mensagens. No modo expandido, exibe a primeira mensagem de e-mail ou uma lista das mensagens na caixa de entrada;
3. **Image** - No modo retraído, exibe apenas o texto da mensagem. No modo expandido, exibe o texto e uma imagem;

Criando notificações

Para criar uma notificação no Android, podemos uasr a classe **Notification.Builder**. Essa classe foi introduzido no Android 3.0 para simplificar a criação de objetos de notificação. Para criar notificações que sejam compatíveis com versões anteriores do Android, você pode usar a classe **NotificationCompat.Builder** em vez de **Notification.Builder.** ***(****Iremos usar essa abordagem no exemplo deste artigo.)*

A classe **Notification.Builder** fornece métodos para definir as várias opções em uma notificação, como:

- O conteúdo, incluindo o título, o texto da mensagem e o ícone de notificação;
- O estilo da notificação, como texto grande, caixa de entrada ou estilo de imagem;
- A prioridade da notificação: mínimo , baixo, padrão, alto ou máximo;
- A visibilidade da notificação no lockscreen: pública, privada ou secreta;
- A Categoria de metadados que ajudam o Android a classificar e filtrar a notificação;
- Uma intent opcional que indica uma atividade a ser iniciada quando a notificação é tocada;

Para gerar uma notificação no Android, vamos realizar as seguintes tarefas:

- Instanciar um objeto Notification.Builder (ou **NotificationCompat.Builder**);
- Chamar vários métodos sobre o objeto Notification.Builder para definir opções de notificação;
- Chamar o método Build do objeto Notification.Builder para instanciar um objeto de notificação;
- Chamar o método Notify do gerenciador de notificações para publicar a notificação;

Devemos fornecer pelo menos as seguintes informações para cada notificação:

- Um ícone pequeno (24x24 dp em tamanho)
- Um título curto
- O texto da notificação

Depois de definir essas opções no **builder**, geramos um objeto de notificação que contém as configurações. Para publicar a notificação, passamos esse objeto de notificação para o gerenciador de notificações. A classe **NotificationManager** é responsável por publicar notificações e exibi-las para o usuário. Uma referência a esta classe pode ser obtida de qualquer contexto, tal como uma atividade ou um serviço.

Se você estiver criando um aplicativo que também será executado em versões anteriores do Android *(desde o API nível 4)*, você deve usar a classe **NotificationCompat.Builder** em vez de da classe **Notification.Builder**.

Quando você cria notificações com o **NotificationCompat.Builder**, o Android garante que o conteúdo básico de notificação seja exibido corretamente em dispositivos mais antigos. No entanto, como alguns recursos avançados de notificação não estão disponíveis em versões mais antigas do Android, o código deve manipular explicitamente problemas de compatibilidade para estilos de notificação expandidos, categorias e níveis de visibilidade.

No exemplo deste artigo vamos usar a classe **NotificationCompat.Builder** para manter a compatibilidade com versões anteriores do Android. E para isso vamos incluir o componente **Android Support Library v4** em nosso projeto via Nuget.

Recursos usados:

- **Visual Studio Community 2017 ou Xamarin Studio**
- **Xamarin**
- **Emulador Android virtual ou físico (veja como emular usando o Vysor)**

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

Criando o projeto Android no VS Community 2017

Abra o VS 2017 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 **Droid_Notificacoes**, e clique no botão **OK**;

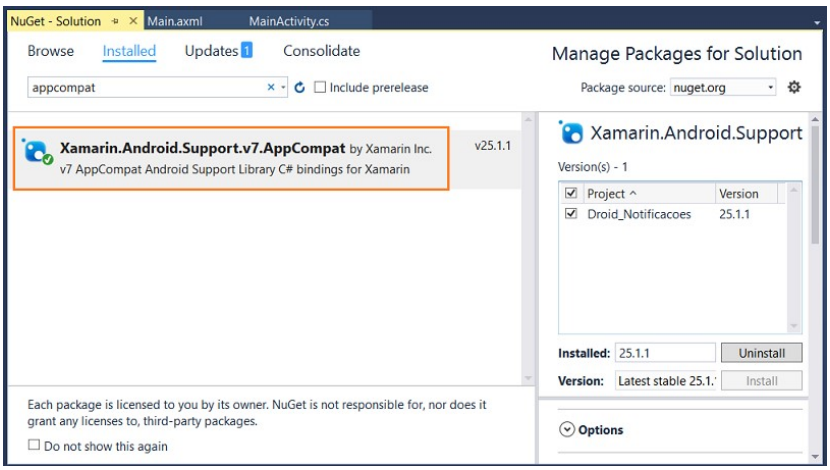
Abra o arquivo **Main.axml** na pasta **Resources/layout** no modo **Source** e a seguir inclua o código abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:background="@android:color/holo_orange_dark"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/btnNotifica"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Exibir Notificação" />
</LinearLayout>
```

Este código apenas inclui um **Button** com o texto - **Exibir Notificação.**

Antes de prosseguir vamos incluir no projeto o pacote de compatibilidade. No menu **Tools** clique em **Nuget Package Manager -> Manage Nuget Packages for Solution;**

A seguir selecione o pacote **Xamarin.Android.Support.v7.AppCompat** e instale o no projeto clicando no botão **Install**:



Agora podemos iniciar a implementação do código para exibir enviar a notificação no arquivo **MainActivity.cs.**

Vamos começar definindo os namespaces usados no projeto:

```
using Android.App;
using Android.Widget;
using Android.OS;
using Android.Content;
using Android.Support.V4.App;
```

A seguir a declaração da **Activity** como sendo a principal e a definição do ícone da aplicação:

```
[Activity(Label = "Droid_Notificacoes", MainLauncher = true, Icon = "@drawable/icon")]
```

A seguir defina o seguinte código na classe

```
public class MainActivity : Activity
{
    int iNotificacao = 50;

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

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

        Button btnNotifica = FindViewById<Button>(Resource.Id.btnNotifica);

        btnNotifica.Click += delegate
        {
            Intent novaIntent = new Intent(this, typeof(MainActivity));
            PendingIntent resultPendingIntent = PendingIntent.GetActivity(this, 0, novaIntent, PendingIntentFlags.CancelCurrent);

            string conteudo = "Macoratti .net - Quase tudo para a plataforma .NET, incluindo Visual Basic, C#,  ASP .NET MVC , Xamarin Forms e Xamarin Android, ASP .NET Core";
```

```
            NotificationCompat.Builder builder = new NotificationCompat.Builder(this)
            .SetAutoCancel(true)
            .SetContentIntent(resultPendingIntent)
            .SetContentTitle("Notificacão do Macoratti")
            .SetSmallIcon(Resource.Drawable.Icon)
            .SetStyle(new NotificationCompat.BigTextStyle().BigText(conteudo))
            .SetContentText(conteudo);

            NotificationManager gerenciaNotificacao = (NotificationManager)GetSystemService(Context.NotificationService);
            gerenciaNotificacao.Notify(iNotificacao, builder.Build());

        };
    }
}
```

Vamos entender o código:

Iniciamos criando uma **Intent** para a Activity principal e a seguir declaramos uma **PendingIntent** que descreve uma **Intent** e a ação que vamos realizar com ela.

Ao conceder um **PendingIntent** a um outro aplicativo, estamos concedendo o direito de executar a operação especificada como se o outro aplicativo fosse você mesmo *(com as mesmas permissões e identidade)*.

A seguir usamos a classe **NotificationCompat.Builder** para gerar uma notificação básica. Observe que métodos **NotificationCompat.Builder** suportam o encadeamento de método; Ou seja, cada método retorna o objeto construtor para que você possa usar o resultado da última chamada de método para chamar a próxima chamada de método:

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(this)
        .SetAutoCancel(true)
        .SetContentIntent(resultPendingIntent)
        .SetContentTitle("Notificacão do Macoratti")
        .SetSmallIcon(Resource.Drawable.Icon)
        .SetStyle(new NotificationCompat.BigTextStyle().BigText(conteudo))
        .SetContentText(conteudo);
```

Neste exemplo, um novo objeto **NotificationCompact.Builder** chamado **builder** é instanciado; o título e o texto da notificação são definidos e o ícone de notificação é carregado a partir de **Resources/ drawable/Icon.** Além disso definimos o estilo do layout como **BigText.**

A chamada para o método **Build** do construtor de notificações cria um objeto de notificação com essas configurações. O próximo passo é chamar o método **Notify** do gerenciador de notificações. Para localizar o gerenciador de notificações, fizemos uma chamada a **GetSystemService**.

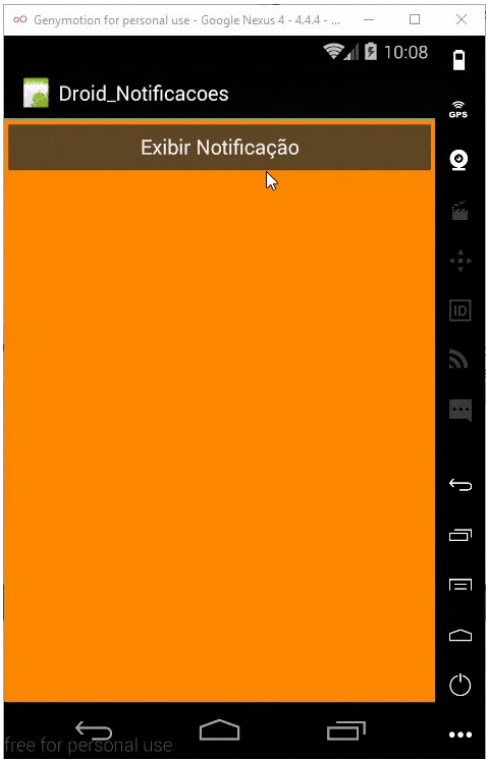
O método **Notify** aceita dois parâmetros:

1. O identificador de notificação;
2. O objeto de notificação;

```
gerenciaNotificacao.Notify(iNotificacao, builder.Build());
```

O identificador de notificação é um número inteiro exclusivo que identifica a notificação para o seu aplicativo. Neste exemplo, o identificador de notificação é definido como o valor 50; No entanto, em um aplicativo de produção, você deseja dar a cada notificação um identificador exclusivo. A reutilização do valor do identificador anterior numa chamada para o **Notify** faz com que a última notificação seja sobrescrita.

Executando o projeto iremos obter o seguinte resultado:



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

"Porque a lei foi dada por Moisés; a graça e a verdade vieram por Jesus Cristo." João 1:17

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/android/application_fundamentals/notifications/local_notifications_in_android/#overview
- https://developer.xamarin.com/guides/android/application_fundamentals/notifications/local_notifications_in_android/#compatibility

---

José Carlos Macoratti