O [Android SDK](http://developer.android.com/) provê diversos recursos interessantes para a criação de layouts que se adaptam às diversas telas e resoluções dos aparelhos. Seguem algumas dicas que costumamos utilizar em nossos projetos.

**Imagens .9.png**

Como aplicativos Android podem ser instalados em vários *smartphones* que possuem [telas diferentes](http://developer.android.com/guide/practices/screens_support.html), a gente acaba tendo que fazer várias imagens de tamanhos diferentes. Mas em alguns casos, nós podemos utilizar as imagens .9.png.

Mas o que é esse formato? Simples, ele permite que as imagens sejam esticadas, sem perder qualidade, vertical e horizontalmente (conforme determinado por você), dependendo da tela do aparelho. E como fazer? Basta utilizar um programinha disponibilizado pela Google quando você baixa o Android SDK chamado **draw9patch**. Para saber mais, [clique aqui](http://developer.android.com/guide/developing/tools/draw9patch.html).

Abaixo, um exemplo de imagem .9.png:

Imagem .9.png

O exemplo acima poderia gerar os seguintes resultados sem nenhum esforço:

Botão 1 

Aqui na empresa nós utilizamos muito esse artifício para botões que possuem o mesmo *background*, economizando o trabalho do designer e também o tamanho do aplicativo.

**Gradiente (Degradê)**

Quando queremos utilizar algum tipo de gradiente nos nossos layouts, podemos criar um arquivo xml em *drawable* da seguinte maneira:

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle">

<gradient android:startColor="#7F9BA4" android:endColor="#59707D"

android:angle="270" />

</shape>

O que ele faz nada mais é que determinar o formato do gradiente, no nosso caso é retangular, e as cores de início e fim. É bem simples e o resultado é igual ao de uma imagem feita do mesmo jeito. O mais legal é que você também economizará o trabalho do designer e o tamanho do aplicativo.

**Borda arredondada**

Abaixo, outra dica: implementar bordas arredondadas.

<?xml version="1.0" encoding="UTF-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<solid android:color="#00FFFF" />

<padding android:left="4dp" android:top="4dp"

android:right="4dp" android:bottom="4dp" />

<corners android:radius="4dp" />

</shape>

Basta você colocar na pasta *drawable* e chamá-lo no *background* de um *LinearLayout*, por exemplo. Os cantos ficarão arredondados!

**Diferentes estados de botão**

Quando temos um botão em nosso layout, é importante exibir os seus diferentes estados para o usuário saber o que está acontecendo. Para isso, precisamos criar um arquivo xml em *drawable*com o seguinte conteúdo, por exemplo:

<?xml version="1.0" encoding="utf-8"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<item android:state\_pressed="true" android:drawable="@drawable/botao\_selecionado" />

<item android:state\_focused="true" android:drawable="@drawable/botao\_com\_foco" />

<item android:drawable="@drawable/botao\_normal" />

</selector>

Para usá-lo, basta chamá-lo no *background* do botão em questão.

Espero que essas dicas ajudem a vocês como me ajudaram. E se tiver outra, deixe seu comentário! Até breve :)

**POSTS RELACIONADOS:**

* [**Criando lista com seções em Android**](http://www.mobits.com.br/2011/10/3/criando-lista-com-secoes-em-android)
* [**Como criar cabeçalho e rodapé no Android**](http://www.mobits.com.br/2011/5/27/como-criar-cabecalho-e-rodape-no-android)
* [**Personalizando fontes no Android**](http://www.mobits.com.br/2011/5/20/personalizando-fontes-no-android)
* [**Como fazer layouts para Android**](http://www.mobits.com.br/2010/6/10/como-fazer-layouts-para-android)

<TextView android:textSize="12dip" android:textAppearance="bold" // ? />

### Android - Shape

Olá Pessoal,  
  
  
Hoje vou apresentar alguns exemplo de utilização de shape esse que é o recurso responsável por criar efeitos gradientes, cantos arredondados entre outros em nosso componentes.  
  
Para criar um shape devemos primeiro criar a pasta drawable dentro da pasta res.  
  
Para esse projeto irei criar um shape para cada componente (LinearLayout, Button, TextView e TextView2).  
  
Abaixo segue um print da estrutura do projeto.

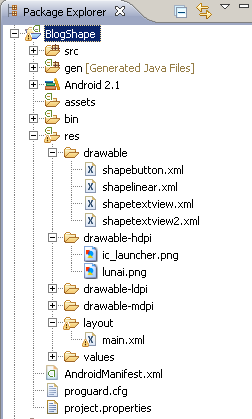
[](http://3.bp.blogspot.com/-tLU-zq0JKBM/T4TVHLIybLI/AAAAAAAAAV0/KdaG5Cjvlto/s1600/projeto.PNG)

Imagem utilizada de background do layout lunai.png:

[](http://2.bp.blogspot.com/-jLXD0s5E11c/T4TX4rxqF9I/AAAAAAAAAWE/Y9G20XGATxo/s1600/lunai.png)

Abaixo seguem todos os arquivos utilizados.  
  
Arquivo shapebutton.xml

[?](http://escoladeandroid.blogspot.com.br/2012/04/android-shape.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <?xml version="1.0" encoding="UTF-8"?>  <shape xmlns:android="<http://schemas.android.com/apk/res/android>">        <gradient          android:startColor="#FFFAFA"          android:endColor="#00008B"          android:angle="270" />        <stroke          android:width="2dp"          android:color="#ffffff" />        <corners          android:radius="50dp" />        <padding          android:left="15dp"          android:top="15dp"          android:right="15dp"          android:bottom="15dp" />    </shape> |

Arquivo shapelinear.xml

[?](http://escoladeandroid.blogspot.com.br/2012/04/android-shape.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?xml version="1.0" encoding="UTF-8"?>  <shape xmlns:android="<http://schemas.android.com/apk/res/android>">        <gradient           android:startColor="#DDff0000"           android:endColor="#DDFFA500"           android:angle="0"/>        <corners          android:bottomRightRadius="7dp"          android:bottomLeftRadius="7dp"          android:topLeftRadius="7dp"          android:topRightRadius="7dp"/>  </shape> |

Arquivo shapetextview.xml

[?](http://escoladeandroid.blogspot.com.br/2012/04/android-shape.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?xml version="1.0" encoding="UTF-8"?>  <shape xmlns:android="<http://schemas.android.com/apk/res/android>">        <solid android:color="#551A8B" />        <padding          android:bottom="4dp"          android:left="4dp"          android:right="4dp"          android:top="4dp" />        <corners          android:bottomRightRadius="7dp"          android:bottomLeftRadius="7dp"          android:topLeftRadius="7dp"          android:topRightRadius="7dp"/>  </shape> |

Arquivo shapetextview2.xml

[?](http://escoladeandroid.blogspot.com.br/2012/04/android-shape.html)

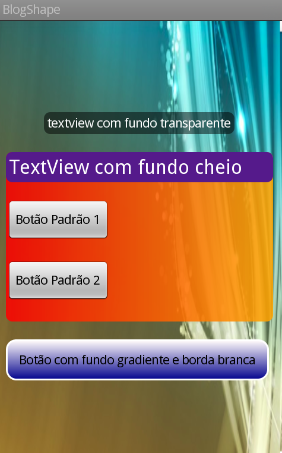
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?xml version="1.0" encoding="UTF-8"?>  <shape xmlns:android="<http://schemas.android.com/apk/res/android>">        <solid android:color="#80000000" />        <padding          android:bottom="4dp"          android:left="4dp"          android:right="4dp"          android:top="4dp" />        <corners          android:bottomRightRadius="7dp"          android:bottomLeftRadius="7dp"          android:topLeftRadius="7dp"          android:topRightRadius="7dp"/>  </shape> |

Arquivo main.xml

[?](http://escoladeandroid.blogspot.com.br/2012/04/android-shape.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout xmlns:android="<http://schemas.android.com/apk/res/android>"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      android:background="@drawable/lunai"      android:orientation="vertical" >        <LinearLayout          android:id="@+id/linearLayout1"          android:layout\_width="fill\_parent"          android:layout\_height="wrap\_content"          android:layout\_centerHorizontal="true"          android:layout\_centerVertical="true"          android:layout\_marginLeft="20dp"          android:background="@drawable/shapelinear"          android:orientation="vertical" >            <TextView              android:id="@+id/textView1"              android:layout\_width="fill\_parent"              android:layout\_height="wrap\_content"              android:text="TextView com fundo cheio"              android:background="@drawable/shapetextview"              android:textAppearance="?android:attr/textAppearanceLarge" />            <Button              android:id="@+id/button1"              android:layout\_width="wrap\_content"              android:layout\_height="wrap\_content"              android:layout\_marginTop="20dp"              android:text="Botão Padrão 1" />            <Button              android:id="@+id/button2"              android:layout\_width="wrap\_content"              android:layout\_height="wrap\_content"              android:layout\_marginTop="20dp"              android:layout\_marginBottom="20dp"              android:text="Botão Padrão 2" />        </LinearLayout>        <Button          android:id="@+id/button3"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:layout\_alignLeft="@+id/linearLayout1"          android:layout\_below="@+id/linearLayout1"          android:background="@drawable/shapebutton"          android:layout\_marginTop="20dp"          android:text="Botão com fundo gradiente e borda branca" />        <TextView          android:id="@+id/textView2"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:layout\_above="@+id/linearLayout1"          android:layout\_centerHorizontal="true"          android:layout\_marginBottom="20dp"          android:background="@drawable/shapetextview2"          android:text="textview com fundo transparente"          android:textColor="@android:color/white" />    </RelativeLayout> |

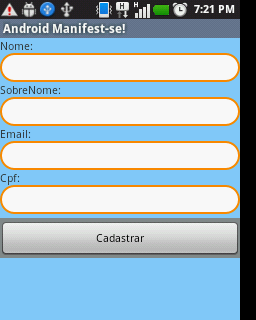
Abaixo segue uma imagem com o resultado do layout.

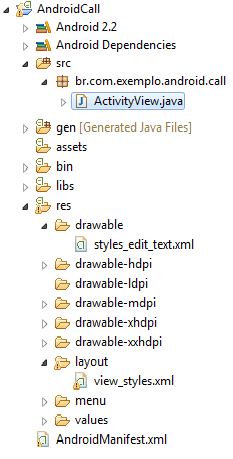
[](http://2.bp.blogspot.com/-lS1l4b3-XDg/T4TXiR_1-pI/AAAAAAAAAV8/3rkjtgGyZMw/s1600/shape.PNG)

Agora uma breve explicação sobre os principais atributos de um shape:  
  
Stroke = desenha uma borda sobre o componente  
Corners = define os canots arredondados  
Solid = define a cor de fundo do componente

### Cantos Arrendondados exemplo no widget Editext do android !

**O post a seguir tem o propósito de lhe mostrar como estilizar o componente editText com  xml.**  
**A nosssa tela terá essa imagem a seguir:**

[](http://2.bp.blogspot.com/-ymb6S0jh__E/UbUBFsntkiI/AAAAAAAAAFk/64Q7KMFmYzo/s1600/device-2013-06-09-192146.png)

[](http://3.bp.blogspot.com/-qBKL_FEFJz8/UbUOIXe3QSI/AAAAAAAAAF0/pYdkMEsagDg/s1600/xml.tela.png)

**Lembrando que você para fazer esse layout***view\_styles.xml***da tela você pode só clicar no componente e arrastar para o layout ganhando-se tempo na elaboração do projeto.**

1: <?xml version="1.0" encoding="utf-8"?>

2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

3: android:layout\_width="match\_parent"

4: android:layout\_height="match\_parent"

5: android:background="#87CEFA"

6: android:orientation="vertical" >

7: <!--

8: Em todos os componentes edittext existe o atributo

9: background e veja que busquei pasta drawable e o xml que estiliza

10: o componete -->

11: <TextView

12: android:layout\_width="wrap\_content"

13: android:layout\_height="wrap\_content"

14: android:text="Nome:" />

15: <EditText

16: android:id="@+id/editName"

17: android:layout\_width="match\_parent"

18: android:layout\_height="wrap\_content"

19: android:background="@drawable/styles\_edit\_text" >

20: <requestFocus />

21: </EditText>

22: <TextView

23: android:layout\_width="wrap\_content"

24: android:layout\_height="wrap\_content"

25: android:text="SobreNome:" />

26: <EditText

27: android:id="@+id/editSobre"

28: android:layout\_width="match\_parent"

29: android:layout\_height="wrap\_content"

30: android:background="@drawable/styles\_edit\_text" />

31: <TextView

32: android:layout\_width="wrap\_content"

33: android:layout\_height="wrap\_content"

34: android:text="Email:" />

35: <EditText

36: android:id="@+id/editEmail"

37: android:layout\_width="match\_parent"

38: android:layout\_height="wrap\_content"

39: android:background="@drawable/styles\_edit\_text" />

40: <TextView

41: android:layout\_width="wrap\_content"

42: android:layout\_height="wrap\_content"

43: android:text="Cpf:" />

44: <EditText

45: android:id="@+id/editCpf"

46: android:layout\_width="match\_parent"

47: android:layout\_height="wrap\_content"

48: android:background="@drawable/styles\_edit\_text" />

49: <!--

50: Esse linearlayout aborda o botão pra ter o controle do

51: componente,veja que adicionei a cor pra destacar o botão

52: -->

53: <LinearLayout

54: android:layout\_width="match\_parent"

55: android:layout\_height="wrap\_content"

56: android:layout\_marginTop="5dp"

57: android:background="#838B8B" >

58: <Button

59: android:id="@+id/button1"

60: android:layout\_width="fill\_parent"

61: android:layout\_height="wrap\_content"

62: android:layout\_marginTop="5dp"

63: android:text="Cadastrar" />

64: </LinearLayout>

65: </LinearLayout>

**Crie uma pasta chamado drawable na pasta RES do projeto depois crie esse arquivo abaixo***styles\_edit\_text.xml***nela,esse arquivo fará toda estilização do componente editText.**  
**O atributo***coners***fará os cantos do widget ficar arredondados ,***stroke* **cria uma borda e depois de criado a borda, o cursor fica fora do componente por isso coloca-se o componente** *padding***para ajustar. Querendo alterar a cor de fundo do widget editText o atributo***solid***faz essa alteração!**

1: <?xml version="1.0" encoding="utf-8"?>

2: <shape xmlns:android="http://schemas.android.com/apk/res/android" >

3: <corners android:radius="19dp" />

4: <stroke

5: android:width="2dp"

6: android:color="#FF8C00" />

7: <padding

8: android:bottom="7dp"

9: android:left="7dp"

10: android:right="7dp"

11: android:top="7dp" />

12: <solid android:color="#FFF" />

13: </shape>

**A nossa Activity fica assim:**

1: public class ActivityView extends Activity {

2: @Override

3: protected void onCreate(Bundle savedInstance

# [Alterar a cor da linha de um EditText](http://pt.stackoverflow.com/questions/44587/alterar-a-cor-da-linha-de-um-edittext)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| votar a favor2votar contra[favorita](http://pt.stackoverflow.com/questions/44587/alterar-a-cor-da-linha-de-um-edittext) | Tenho um Layout simulando uma tela de PIN (tipo uma lockscreen), onde eu tenho um EditText. Estou usando a AppCompat v21 para poder usar os elementos estilo Material. Porém, quero definir somente nessa tela de PIN a cor de destaque (accent) para o EditText, sem alterar todo o projeto.  Atualmente, esse é o meu arquivo styles.xml:  <?xml version="1.0" encoding="utf-8"?>  <resources>  <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">  <item name="colorPrimary">@color/indigo</item>  <item name="colorPrimaryDark">@color/indigo\_dark</item>  <item name="colorAccent">@android:color/white</item>  </style>  </resources>  Quero com que somente o EditText da tela de PIN use a cor branca no destaque. O resto do app quero que tenha outra cor, mas não estou conseguindo fazer isso. Existe alguma tag que preciso usar? Tem como fazer isso via código?  [android](http://pt.stackoverflow.com/questions/tagged/android) [android-layout](http://pt.stackoverflow.com/questions/tagged/android-layout)   |  |  |  | | --- | --- | --- | | [compartilhar](http://pt.stackoverflow.com/q/44587)[melhorar esta pergunta](http://pt.stackoverflow.com/posts/44587/edit) | [editada 19/12/14 às 19:19](http://pt.stackoverflow.com/posts/44587/revisions)  [[https://www.gravatar.com/avatar/64e7d51b09a244e9376384839fb17fc1?s=32&d=identicon&r=PG](http://pt.stackoverflow.com/users/357/piovezan)](http://pt.stackoverflow.com/users/357/piovezan)  [Piovezan](http://pt.stackoverflow.com/users/357/piovezan) **3.734**323 | perguntada 19/12/14 às 14:33  [[https://i.stack.imgur.com/hTXyv.jpg?s=32&g=1](http://pt.stackoverflow.com/users/7404/renan-lazarotto)](http://pt.stackoverflow.com/users/7404/renan-lazarotto)  [Renan Lazarotto](http://pt.stackoverflow.com/users/7404/renan-lazarotto) **468**111 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 1 |  | | Renan, pelo que eu vi, está previsto nos próximas versões do **AppCompat** o atributo theme para os widgets (atualmente só o Toolbar possui essa funcionalidade). No Android Lollipop isso já é suportado. Mais detalhes em: [chris.banes.me/2014/11/12/theme-vs-style](https://chris.banes.me/2014/11/12/theme-vs-style/) –  [Wakim](http://pt.stackoverflow.com/users/6436/wakim) [19/12/14 às 20:37](http://pt.stackoverflow.com/questions/44587/alterar-a-cor-da-linha-de-um-edittext#comment89157_44587) |   Comentar |

## 2 Respostas

[ativas](http://pt.stackoverflow.com/questions/44587/alterar-a-cor-da-linha-de-um-edittext?answertab=active#tab-top)[mais antigas](http://pt.stackoverflow.com/questions/44587/alterar-a-cor-da-linha-de-um-edittext?answertab=oldest#tab-top)[votos](http://pt.stackoverflow.com/questions/44587/alterar-a-cor-da-linha-de-um-edittext?answertab=votes#tab-top)

|  |  |  |  |
| --- | --- | --- | --- |
| votar a favor1votar contraaceitos | O gerador [Android Holo Colors](http://android-holo-colors.com/) pode te ajudar nesta tarefa, ou você pode criar seu próprio **drawable**para definir como fundo do seu EditText, mas envolve mais elementos como cor de foco, desabilitado e etc.  Por exemplo, se quiser apenas uma cor sólida da borda inferior, crie um arquivo **bg\_pin.xml** no diretório res/drawable assim:  <layer-list xmlns:android="http://schemas.android.com/apk/res/android" >  <!-- cor da borda -->  <item>  <shape>  <solid android:color="@android:color/white" />  </shape>  </item>  <!-- cor de fundo, tamanho somente da borda inferior -->  <item android:bottom="1dp">  <shape>  <solid android:color="@android:color/transparent" />  </shape>  </item>  </layer-list>  E então, no layout do seu XML onde quiser aplicar o estilo, algo assim:  <EditText  ....  android:background="@drawable/bg\_pin" />   |  |  | | --- | --- | | [compartilhar](http://pt.stackoverflow.com/a/44597)[melhorar esta resposta](http://pt.stackoverflow.com/posts/44597/edit) | respondida 19/12/14 às 16:48  [[https://www.gravatar.com/avatar/c75d118f0ca9325db96f1f8980e2244e?s=32&d=identicon&r=PG](http://pt.stackoverflow.com/users/4337/paulo-rodrigues)](http://pt.stackoverflow.com/users/4337/paulo-rodrigues)  [Paulo Rodrigues](http://pt.stackoverflow.com/users/4337/paulo-rodrigues) **4.561**2629 | |
|  | Comentar |
| votar a favor0votar contra | Acredito que a forma mais correta seria você criar uma derivação do tema principal do projeto para essa tela, pois se outras mudanças forem necessárias nela, você não tem que ficar criando arquivos específicos, mas apenas mudar no tema para essa tela.  Supondo que seu tema se chame AppTheme.  Eu criaria um tema para essa tela e nela eu definiria o colorAcent.  <style name="AppTheme.Pin" parent="AppTheme">  <item name="colorAccent">cor desejada</item>  </style>  Depois, basta ir no Androidmanifest.xml e especificar que para essa Activity, o tema será esse:  <activity android:name=".PinActivity"  android:theme="@style/AppTheme.Pin"/>  Se quiser, pode informar via código também, dentro do método onCreate:  setTheme(R.style.AppTheme\_Pin);   |  | | --- | | [compartilhar](http://pt.stackoverflow.com/a/44826)[melhorar esta resposta](http://pt.stackoverflow.com/posts/44826/edit) | |

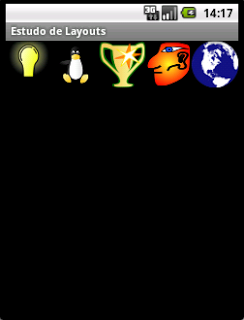
### Lição 5: Layout e interface com usuário

**Criação de layout**  
  
Um Layout é um arquivo XML que define quais ViewGroups existem e quais Views elas contém. Pontos importantes:  
Toda activity deve possuir pelo menos uma View para interagir com o usuário;  
Normalmente, usamos um arquivo de layout, contendo um ViewGroup e algumas Views;  
Todos os elementos de layout são referenciados na classe “R.java”;  
  
Baixe o projeto “http://layout-exercise.googlecode.com/svn/trunk/”:   
1. No eclipse, abra a view: “SVN repositories”;  
2. Adicione o repositório acima;  
3. Expanda o repositório;  
4. Selecione a pasta “Layouts” e faça “Checkout”;  
  
Examine o arquivo “res/layout/main.xml”:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2. android:orientation="vertical"
3. android:layout\_width="fill\_parent"
4. android:layout\_height="fill\_parent"
5. >
6. <TextView
7. style="@style/titulo\_link.vermelho"
8. android:layout\_width="fill\_parent"
9. android:layout\_height="wrap\_content"
10. android:text="@string/linear"
11. android:onClick="clicouLinear"
12. />
13. <TextView
14. style="@style/titulo\_link.verde"
15. android:layout\_width="fill\_parent"
16. android:layout\_height="wrap\_content"
17. android:text="@string/table"
18. android:onClick="clicouTable"
19. />
20. <TextView
21. style="@style/titulo\_link.azul"
22. android:layout\_width="fill\_parent"
23. android:layout\_height="wrap\_content"
24. android:text="@string/relative"
25. android:onClick="clicouRelative"
26. />
27. <TextView
28. style="@style/titulo\_link.lilas"
29. android:layout\_width="fill\_parent"
30. android:layout\_height="wrap\_content"
31. android:text="@string/with\_images"
32. android:onClick="clicouWithImages"
33. />
34. <TextView
35. style="@style/titulo\_link.laranja1"
36. android:layout\_width="fill\_parent"
37. android:layout\_height="wrap\_content"
38. android:text="@string/dontDoIt01"
39. android:onClick="clicouDont01"
40. />
41. <TextView
42. style="@style/titulo\_link.piscina"
43. android:layout\_width="fill\_parent"
44. android:layout\_height="wrap\_content"
45. android:text="@string/dontDoIt02"
46. android:onClick="clicouDont02"
47. />
48. <TextView
49. style="@style/titulo\_link.amarelo"
50. android:layout\_width="fill\_parent"
51. android:layout\_height="wrap\_content"
52. android:text="@string/listacombarra"
53. android:onClick="clicouLista"
54. />
55. </LinearLayout>

Todo layout é um arquivo ANDROID XML, cujo nome deve começar com letras minúsculas, e deve residir na pasta: “res/layout”. Ele deve possuir um e somente um ViewGroup principal, que, neste caso, é o LinearLayout. O LinearLayout coloca tudo em sequência. Execute a aplicação e selecione “LinearLayout”:

[](http://3.bp.blogspot.com/-AQaiTiv2QGY/TZygzdZvUkI/AAAAAAAAAEQ/ta3agIjEUuk/s1600/fig01.png)

Todas as views “filhas” são organizadas em sequência, enquando couberem na mesma linha. Voltando ao arquivo “main.xml”, note que temos 7 Views dentro do nosso ViewGroup principal e todas são do tipo “TextView”, Uma “TextView” serve para exibir textos.  
  
Agora, abra o seu primeiro projeto, cujo layout é mais simples, e veja o arquivo “res/main.xml”:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3. android:orientation="vertical"
4. android:layout\_width="fill\_parent"
5. android:layout\_height="fill\_parent"
6. >
7. <TextView
8. android:layout\_width="fill\_parent"
9. android:layout\_height="wrap\_content"
10. android:text="@string/hello"
11. />
12. </LinearLayout>

Temos uma única “TextView”, com três propriedades informadas:

**android:layout\_width**

Qual é a largura que a View prefere ter, dentro de seu “parent”, que, neste caso é o ViewGroup LinearLayout.

**android:layout\_height**

Qual é a altura que a View prefere ter dentro de seu “parent”.

**android:text**

Qual é o texto que será exibido dentro da View.

As propriedades “android:layout\_width” e “android:layout\_height” são muito importantes e são obrigatórias. Nós podemos especificar três valores:   
FILL\_PARENT A view quer ser tão grande naquela dimensão (altura ou largura) como o seu “parent”. Ela encherá o ViewGroup;  
WRAP\_CONTENT A view quer ser grande o suficiente para conter o texto (ou imagem) atribuído;  
Unidades Podem ser “dp” - density independent values, “sp” - scale independent, “pt” - 1/72 de polegada ou “px” -pixels;  
  
No nosso caso, estamos estabelecento que a TextView será da largura de seu “parent” e terá a altura suficiente para conter o texto que atribuímos (este é o padrão para “TextView”).  
  
Na propriedade “android:text” especificamos um string resource.   
  
Nossa aplicação pode ter vários tipos de recursos (resources): layouts, estilos, strings, imagens etc; os nomes dos recursos ficam “compilados” em uma tabela binária, chamada de “<projeto>.arsc”. Para identificar o recurso, usamos um número inteiro, que é o seu índice na tabela.  
  
Sempre que criamos alguma coisa na pasta “res”, o ADT altera a classe “R.java”, dentro da pasta “gen”. Ele cria uma constante dentro da “inner class” apropriada. Eis o “R.java” do nosso primeiro projeto: 

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. **package** br.com.thecodebakers.primeiro;
3. **public** **final** **class** R {
4. **public** **static** **final** **class** attr {
5. }
6. **public** **static** **final** **class** drawable {
7. **public** **static** **final** **int** icon=0x7f020000;
8. }
9. **public** **static** **final** **class** layout {
10. **public** **static** **final** **int** main=0x7f030000;
11. }
12. **public** **static** **final** **class** string {
13. **public** **static** **final** **int** app\_name=0x7f040001;
14. **public** **static** **final** **int** hello=0x7f040000;
15. }
16. }

Temos três “inner classes” que são constantes dentro desta classe: “attr”, que representa atributos (não estamos usando), “drawable”, que representa imagens, “layout”, que representa layouts de views, e “string”, que representa recursos de strings. Outro muito importante é o “id”, que representa os identificadores que criamos dentro de nossos layouts (veja a classe R.java do projeto “Layouts”).  
  
No layout de nossa activity (primeiro projeto), dissemos o seguinte:   
  
android:text="@string/hello"  
  
Logo, o texto que será exibido é o que está contido dentro do recurso de strings chamado “hello”. E onde fica ele? Dentro de um arquivo XML, que fica na pasta “res”:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. <?xml version="1.0" encoding="utf-8"?>
2. <resources>
3. <string name="hello">Hello World, Principal!</string>
4. <string name="app\_name">Primeiro Projeto</string>
5. </resources>

Por que isto? para facilitar a localização (L10N) de nossa aplicação.  
  
Finalmente, temos também os “ids”. Abra o projeto “BioMago”, que importamos na primeira aula. Execute o programa e cadastre um registro. Note que, ao cadastrar, o BioMago tem que pegar o nome e a data de nascimento. A atividade que faz o cadastramento é a “Edicao.java” e seu layout é “R.layout.edicao”, ou o arquivo: “edicao.xml”. Vamos primeiramente ver o arquivo de layout:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. <RelativeLayout
2. xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout\_width="fill\_parent"
4. android:layout\_height="fill\_parent">
6. <TextView
7. android:layout\_alignParentTop="true"
8. android:layout\_centerHorizontal="true"
9. android:text="@string/lblNome"
10. style="@style/titulo"
11. android:layout\_height="wrap\_content"
12. android:layout\_width="fill\_parent"
13. android:id="@+id/lblnome"/>
14. <EditText
15. android:layout\_below="@+id/lblnome"
16. android:layout\_centerHorizontal="true"
17. android:id="@+id/txtNome"
18. android:maxLines="1"
19. android:layout\_height="wrap\_content"
20. android:layout\_width="fill\_parent"/>
22. <TextView
23. android:layout\_below="@+id/txtNome"
24. android:layout\_centerHorizontal="true"
25. style="@style/titulo"
26. android:text="@string/lblNascimento"
27. android:layout\_height="wrap\_content"
28. android:layout\_width="fill\_parent"
29. android:id="@+id/lbldata"/>
31. <DatePicker
33. android:layout\_below="@+id/lbldata"
34. android:layout\_centerHorizontal="true"
35. android:id="@+id/dtData"
36. android:layout\_height="wrap\_content"
37. android:layout\_width="fill\_parent"
38. ></DatePicker>
40. <ImageView
41. android:layout\_alignParentBottom="true"
42. android:layout\_alignParentLeft="true"
43. android:src="@drawable/ok"
44. android:layout\_gravity="left"
45. android:id="@+id/btnSalvar"
46. android:layout\_height="wrap\_content"
47. android:layout\_width="wrap\_content"
48. android:onClick="salvar"
49. ></ImageView>
50. <ImageView
51. android:layout\_alignParentBottom="true"
52. android:layout\_alignParentRight="true"
53. android:src="@drawable/cancelar"
54. android:layout\_gravity="left"
55. android:id="@+id/btnCancelar"
56. android:layout\_height="wrap\_content"
57. android:layout\_width="wrap\_content"
58. android:onClick="cancelar"
59. ></ImageView>
60. </RelativeLayout>

Neste projeto, usamos outro tipo de ViewGroup, que é o “RelativeLayout”. Note a View “EditText”, que está em negrito:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. <EditText
2. android:layout\_below="@+id/lblnome"
3. android:layout\_centerHorizontal="true"
4. android:id="@+id/txtNome"
5. android:maxLines="1"
6. android:layout\_height="wrap\_content"
7. android:layout\_width="fill\_parent"/>

O que é isso? A propriedade “id” permite que nos referenciamos a uma View dentro do código-fonte Java, podendo obter o valor que o usuário digitou, por exemplo. O que informamos dentro da propriedade foi “@+id/txtNome”, que significa: se não existir o “id” “txtNome”, crie-o. Uma entrada foi criada dentro de R.java:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. **public** **final** **class** R {
2. **public** **static** **final** **class** attr {
3. ...
4. }
5. **public** **static** **final** **class** drawable {
6. ...
7. }
8. **public** **static** **final** **class** id {
9. ...
10. **public** **static** **final** **int** txtNome=0x7f07000a;
11. ...
12. }
13. **public** **static** **final** **class** layout {
14. ...
15. }
16. **public** **static** **final** **class** menu {
17. ...
18. }
19. **public** **static** **final** **class** string {
20. ...
21. }
22. **public** **static** **final** **class** style {
23. ...
24. }
25. }

Ou seja, criamos um novo identificador de View dentro de nosso projeto, e podemos usá-lo para nos referenciar à EditText que criamos. Para obter o nome do usuário, o código da Activity tem que declarar uma variável do tipo: android.widget.EditText:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. **public** **class** Edicao **extends** Activity {
2. ...
3. **private** EditText txtNome;
4. …
5. **public** **void** salvar (View view) {
6. txtNome = (EditText) **this**.findViewById(R.id.txtNome);
7. Elemento elemento = **new** Elemento();
8. elemento.setNome(txtNome.getText().toString());
9. …

Assim, podemos acessar a View e pegar o nome que o usuário informou.   
  
A EditText possui alguns atributos e propriedades muito interessantes, por exemplo, o atributo XML “android:inputType” nos permite determinar qual tipo de dado o usuário pode digitar, e o atributo: “android:password” substitui o valor digitado por um círculo, como se fosse um campo de senha.  
  
Podemos pegar o conteúdo de uma EditText com o método “getText()”, mas note que ele somente devolve um “Editable”. Se quisermos pegar o string, temos que usar o método: “getText().toString()”. E, se quisermos converter em um número: Long.valueOf(txt.getText().toString());  
  
Se informarmos “numberSigned” no atributo XML “android:inputType”, nossa EditText somente aceitará a digitacão de algarismos, com ou sem sinal.  
  
Para concluir esta parte, precisamos ter uma maneira de reagir aos eventos provocados pelo usuário, como o clique de um botão, por exemplo. Todo elemento “Clicável” possui o atributo XML: “android:onClick”, que nos permite nomear um método de “callback” dentro da classe da Activity. Por exemplo, abra o layout “main,xml” e a Activity “Principal.java”, do projeto “Layouts”:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. <TextView
2. style="@style/titulo\_link.vermelho"
3. android:layout\_width="fill\_parent"
4. android:layout\_height="wrap\_content"
5. android:text="@string/linear"
6. android:onClick="clicouLinear"
7. />

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. **public** **void** clicouLinear(View view) {
2. Intent i = **new** Intent (**this**.getApplicationContext(), LinearDemo.**class**);
3. **this**.startActivity(i);
4. }

Repare que no layout, a TextView está apontando o “android:onClick” para “clicouLinear”, e que na classe da Activity, temos um método com a assinatura: “public void clicouLinear (View view)”. O nome do método tem que ser o mesmo informado no atributo, e a assinatura tem que ter “void” como retorno, e uma instância de “View” como argumento.  
  
Este método será invocado quando aquela TextView for “clicada” pelo usuário.   
  
E o que o método faz? Ele invoca outra Activity!  
  
Ele cria uma instância de android.content.Intent, que serve para determinar o que será executado. Pode ser uma Actrivity do projeto ou um aplicativo externo, como uma página Web. Neste caso, estamos criando um “Intent” que chama a classe da atividade LinearDemo. O método “startActivity()” inicia a outra atividade.  
  
Também poderíamos mostrar uma caixa de diálogo, uma mensagem, tipo JOptionPane, só que usamos outra classe para isto:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. Resources res = **this**.getResources();
2. **new** AlertDialog.Builder(**this**)
3. .setMessage(res.getString(R.string.erro\_data\_posterior))
4. .setNeutralButton(res.getString(R.string.lblOK), **new**    DialogInterface.OnClickListener() {
5. **public** **void** onClick(DialogInterface dialog, **int** which) {
6. }
7. })
8. .show();

Já sei... Muita calma nessa hora!  
  
Para começar, apontamos para os recursos da aplicação usando a classe android.content.res.Resources. Ela serve para obtermos recursos, por exemplo “strings”, dentro do código-fonte Java. Nos arquivos XML usamos apenas “@string/<nome>”, mas no código Java precisamos usar: <instância de Resources>.getString(identificador). Lembrando que todo resource string está possui seu identificador dentro da classe “R.java”.  
  
Depois, criamos uma instância de android.app.AlertDialog.Builder e outra de AlertDialog;  
  
Poderíamos reescrever este código-fonte assim:

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. AlertDialog.Builder construtor = **new** AlertDialog.Builder(**this**);
2. construtor.setMessage(res.getString(R.string.erro\_data\_posterior))
3. .setPositiveButton("Ok", **new** DialogInterface.OnClickListener() {
4. **public** **void** onClick(DialogInterface dialog, **int** id) {
5. // comandos...
6. }
7. });
8. AlertDialog alert = construtor.create();
9. alert.show();

Dois pontos importantes:   
  
1 Todos os diálogos são MODELESS;  
2 Use sempre string resources e evite literais;  
  
Agora, você tem todos os elementos para fazer o laboratório 2.  
  
**Laboratório 2**  
  
Vamos criar uma aplicação que interage com o usuário. Ela receberá um número e fará sua fatoração (decomposição em números primos), exibindo um diálogo que mostrará o resultado.

**1**

Crie um novo projeto Android, conforme fizemos no Laboratório 1;

**2**

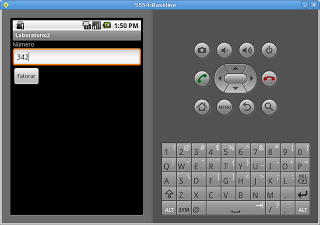
Edite o layout “main.xml” e acrescente:

|  |  |
| --- | --- |
| a | Uma TextView contendo o rótulo: “Número” (deverá estar dentro de um string resource); |
| b | Uma EditView que receba o número digitado (deve apenas receber números e deve ter “id”); |
| c | Um botão Button, que invoque o “callback” para fazer e mostrar a fatoração; |

**3**

O “callback” deverá decompor o número informado em fatores primos, e usar um diálogo para mostrar o resultado;

**O crivo de Eratóstenes**  
  
Para decompor em fatores primos, é necessário, primeiramente, obter uma lista de números primos! Segundo a Wikipedia:   
  
O Crivo de Eratóstenes é um algoritmo e um método simples e prático para encontrar números primos até um certo valor limite. Segundo a tradição, foi criado pelo matemático grego Eratóstenes (c. 285-194 a.C.), o terceiro bibliotecário-chefe da Biblioteca de Alexandria.  
  
Felizmente, para você, estávamos de bom humor e providenciamos uma implementação do “Crivo”, que pode ser encontrada ao final da lição.  
  
O resultado de sua aplicação deve ser como a figura:

[](http://2.bp.blogspot.com/-vDV9_0DNcuE/Tf988YKxmQI/AAAAAAAAANI/Tdl0w5xy-ew/s1600/lab2.png)

**Implementação do Crivo**

[view plainprint?](http://www.thecodebakers.org/p/licao-5-layout-e-interface-com-usuario.html)

1. **package** br.nom.sampaio.cleuton.factorwiz;
2. **import** java.util.ArrayList;
3. **import** java.util.Iterator;
4. **import** java.util.List;
5. **import** android.view.View;
7. **public** **class** FatoracaoBO {
8. **private** **static** List<Long> primos;
9. **private** **static** **long** sinalUltimo = 1;
10. **public** **static** String fatorar (**long** numero) {
11. String resultado = **null**;
13. **if** (numero == 0) {
14. resultado = "0";
15. }
16. **else** {
17. **if** (numero > 0 && numero < 3) {
18. resultado = Long.toString(numero);
19. }
20. **else** {
21. Long valor = numero;
22. **if** (numero < 0) {
23. sinalUltimo = -1;
24. valor = numero \* (-1);
25. }
27. primos = crivo(valor);
28. resultado = calcular(valor);
29. }
30. }
32. **return** resultado;
33. }
35. **private** **static** String calcular(Long numero) {
36. String resultado = "";
37. Long valor = numero;
39. Iterator<Long> it = primos.iterator();
40. Long primo = it.next();
42. **while** (valor > 1) {
43. **long** resto = valor % primo;
44. **if** (resto > 0) {
45. primo = it.next();
46. }
47. **else** {
48. **if** (resultado.length() > 0) {
49. resultado += "x";
50. }
51. valor = valor / primo;
52. **if** (valor == 1) {
53. primo = primo \* sinalUltimo;
54. }
55. resultado += Long.toString(primo);
56. }
57. }
59. **return** resultado;
60. }
62. **private** **static** List<Long> crivo (**long** n) {
63. List<Long> lprimos = **new** ArrayList<Long>();
65. **for** (**long** x = 2; x <= n; x++) {
66. lprimos.add(x);
67. }
68. **long** p = 2;
69. Iterator<Long> it = lprimos.iterator();
70. **while** (Math.pow((**double**)p,2) <= n) {
71. **int** m = 2;
72. **while** (m\*p <= n) {
73. it = lprimos.iterator();
74. **while** (it.hasNext()) {
75. Long valor = it.next();
76. **if** (valor==(m\*p)) {
77. it.remove();
78. **break**;
79. }
80. }
81. m++;
82. }
84. it = lprimos.iterator();
85. **while** (it.hasNext()) {
86. Long valor = it.next();
87. **if** (valor > p) {
88. p = valor;
89. **break**;
90. }
91. }
92. }
94. **return** lprimos;
96. }
97. }

### Criando Botão com Borda e Degrade

Crie sempre os arquivos XML de Layout dentro das pastas Drawable. Veja um exemplo de botão na **Figura 5**. Na **Listagem 5** vemos a codificação do mesmo e que serve para os botaonormal.XML|botaoselecionado.XML|botaofoco.XML.



**Figura 5.**Botão DevMedia

**Listagem 5.**botao.XML

<?xml version="1.0" encoding="utf-8"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android" >

<item android:state\_pressed="true" android:drawable="@drawable/botaoselecionado" />

<item android:state\_focused="true"

android:drawable="@drawable/botaofoco" />

<item android:drawable="@drawable/botaonormal"/>

</selector>

Para alterar as cores do degrade e o layout do botão basta usar o código das **Listagens 6 e 7.**

**Listagem 6.**Alteração de cores

<?xml version="1.0" encoding="UTF-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android" >

<gradient

android:angle="270"

android:startColor="#2e8fc2 "

android:endColor="#006599"/>

<stroke

android:width="3dp"

android:color="#00679e " />

<corners android:radius="10dp" />

<padding

android:bottom="10dp"

android:left="10dp"

android:right="10dp"

android:top="10dp" />

</shape>

**Listagem 7.** Objeto Button Layout

<Button

android:id="@+id/btDevmedia"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_margin="10dp"

android:layout\_weight="1"

android:background="@drawable/botao"

android:onClick="OnClick"

android:text="DevMedia"

android:textColor="#FFFFFF"

android:textSize="25sp"

android:textStyle="bold" />

# [Como funciona a adaptação a diferentes resoluções no Android?](http://pt.stackoverflow.com/questions/6279/como-funciona-a-adapta%c3%a7%c3%a3o-a-diferentes-resolu%c3%a7%c3%b5es-no-android)

|  |  |  |  |
| --- | --- | --- | --- |
| votar a favor5votar contra[favorita](http://pt.stackoverflow.com/questions/6279/como-funciona-a-adapta%C3%A7%C3%A3o-a-diferentes-resolu%C3%A7%C3%B5es-no-android)  1 | No html por exemplo, os objetos podem(as vezes sem nenhum código especial) se adaptar ao tamanho da janela, como por exemplo ao colocar duas imagens de tamanho fixo, se diminuir a janela uma vai para a linha de baixo.  Mas e em um App Android? Os elementos são fixos? Isso varia? Preciso fazer um app diferente para cada resolução ou tamanho de tela?  Ainda não programo para Android, mas gostaria de saber qual o conceito das Apps nesse sentido.  [android](http://pt.stackoverflow.com/questions/tagged/android) [resolução-de-tela](http://pt.stackoverflow.com/questions/tagged/resolu%c3%a7%c3%a3o-de-tela) [app](http://pt.stackoverflow.com/questions/tagged/app)   |  |  | | --- | --- | | [compartilhar](http://pt.stackoverflow.com/q/6279)[melhorar esta pergunta](http://pt.stackoverflow.com/posts/6279/edit) | perguntada 18/02/14 às 14:10  [[https://i.stack.imgur.com/jQBiM.jpg?s=32&g=1](http://pt.stackoverflow.com/users/60/joao-paulo)](http://pt.stackoverflow.com/users/60/joao-paulo)  [Joao Paulo](http://pt.stackoverflow.com/users/60/joao-paulo) **4.034**32071 | |
|  | comentar |

## 2 Respostas

[ativas](http://pt.stackoverflow.com/questions/6279/como-funciona-a-adapta%c3%a7%c3%a3o-a-diferentes-resolu%c3%a7%c3%b5es-no-android?answertab=active#tab-top)[mais antigas](http://pt.stackoverflow.com/questions/6279/como-funciona-a-adapta%c3%a7%c3%a3o-a-diferentes-resolu%c3%a7%c3%b5es-no-android?answertab=oldest#tab-top)[votos](http://pt.stackoverflow.com/questions/6279/como-funciona-a-adapta%c3%a7%c3%a3o-a-diferentes-resolu%c3%a7%c3%b5es-no-android?answertab=votes#tab-top)

|  |  |
| --- | --- |
| votar a favor5votar contraaceitos | Resposta: A adaptação para diferentes resoluções do [android](http://pt.stackoverflow.com/questions/tagged/android) consiste em: Valores relativos à um LinearLayout que seriam os Pesos(android:layout\_weight). Explicação: Você pode atribuir a propriedade android:layout\_weight em seus elementos XML, que seria como se fosse um peso(bem dizer porcentagem) que é distribuído de acordo com o valor desta propriedade em todos os elementos que estão relativos ao seu LinearLayout, seria uma soma de todos os Weight's que estão declarados em Elementos dentro do seu LinearLayout que por si só tem um WeightSum que é uma propriedade que diz qual é o valor máximo(100%) permitido aos seus filhos. Por padrão o valor do WeightSum é 1.  A propriedade em sí do WeightSum que é o padrão de peso total do LinearLayout seria android:weightSum=x onde x seria o valor do weight (que pode ser decimal).  A propriedade em sí do Layout Weight que é o valor do peso de um elemento filho do LinearLayoutseria android:layout\_weight=x onde x seria o valor do weight (que pode ser decimal). Exemplo: <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent"  android:paddingLeft="16dp"  android:paddingRight="16dp"  android:weightSum="1" <!-- aqui está o weightSum -->  android:orientation="vertical" >  <EditText  android:layout\_width="fill\_parent"  android:layout\_height="wrap\_content"  android:hint="@string/to" />  <EditText  android:layout\_width="fill\_parent"  android:layout\_height="wrap\_content"  android:hint="@string/subject" />  <EditText  android:layout\_width="fill\_parent"  android:layout\_height="0dp"  android:layout\_weight="1" <!-- aqui está o weight -->  android:gravity="top"  android:hint="@string/message" />  <Button  android:layout\_width="100dp"  android:layout\_height="wrap\_content"  android:layout\_gravity="right"  android:text="@string/send" />  </LinearLayout> Resultado: exemplo weight Observações: No EditText da Mensagem você não tem valores em dp absolutos para o height(altura), você só tem android:layout\_weight="1" que seria o peso do tal elemento.  Se você olhar em todos os elementos filhos do LinearLayout, vai perceber que o único elemento que contém weight(peso) seria o EditText da Mensagem, por isso a soma de todos os weight's seria 1. Porém se houvesse outro elemento com 1 de weight também, o weight total seria 2 e iria ultrapassar o limite padrão do WeightSum do LinearLayout que seria 1.  Porém se você tiver um WeightSum de 10, por exemplo, você poderia ter 10 elementos filhos contendo android:layout\_weight=1 cada um, sendo assim cada elemento teria 10% do tamanho total. Observação muito importante: Você pode estar se perguntando: Ok, mas onde que eu informo se o weight é pra altura ou pra largura??  A resposta é: Isso é estritamente vinculado ao Orientation do seu LinearLayout que seria esta propriedade:  android:orientation="vertical"  Da qual você pode notar que está declarada no LinearLayout de exemplo acima, porém se ele fosse **horizontal** os weight's seriam aplicados na horizontal apenas. |

Nine-Patch

A NinePatch é uma imagem PNG em que você pode definir regiões esticáveis ​​que o Android escalas quando o conteúdo dentro do Vista excede os limites de imagem normais. Você normalmente atribui este tipo de imagem como fundo de uma ideia que tem pelo menos uma dimensão definida como "wrap\_content", e quando o Vista cresce para acomodar o conteúdo, a imagem Nine-Patch também é dimensionado para corresponder ao tamanho do Vista . Um exemplo de uso de uma imagem de Nine-Patch é o fundo usado por padrão Button widget do Android, que deve esticar para acomodar o texto (ou imagem) dentro do botão.

Mesmo que com um bitmap normal, você pode fazer referência a um arquivo de Nine-Patch directamente ou a partir de um recurso definido pelo XML.

Para uma discussão completa sobre como criar um arquivo de Nine-Patch com regiões esticáveis, ver o documento Gráficos 2D.

Nine-Patch Arquivo

LOCAL DO ARQUIVO:

res / drawable / filename.9.png

O nome do arquivo é usado como o ID do recurso.

COMPILADA DATATYPE RECURSOS:

Ponteiro recurso a uma NinePatchDrawable.

RECURSO DE REFERÊNCIA:

Em Java: R.drawable.filename

Em XML: @ [pacote]: drawable / filename

EXEMPLO:

Com uma imagem salva no res / drawable / myninepatch.9.png, este XML disposição aplica-se o Nine-Patch para a View:

<Button

    android: layout\_height = "wrap\_content"

    android: layout\_width = "wrap\_content"

    android: Fundo = "@ drawable / myninepatch" />

VEJA TAMBÉM:

Gráficos 2D

NinePatchDrawable

XML Nine-Patch

Um XML Nine-Patch é um recurso definido em XML que aponta para um arquivo de Nine-Patch. O XML pode especificar composição de cores para a imagem.

LOCAL DO ARQUIVO:

res / drawable / filename.xml

O nome do arquivo é usado como o ID do recurso.

COMPILADA DATATYPE RECURSOS:

Ponteiro recurso a uma NinePatchDrawable.

RECURSO DE REFERÊNCIA:

Em Java: R.drawable.filename

Em XML: @ [pacote]: drawable / filename

Sintaxe:

<? xml version = "1.0" encoding = "utf-8"?>

<nove remendo

    xmlns: android = "http://schemas.android.com/apk/res/android"

    android: src = "@ [pacote]: drawable / drawable\_resource"

    android: dither = ["true" | "false"] />

ELEMENTOS:

<nove remendo>

Define a fonte Nine-Patch e suas propriedades.

atributos:

xmlns: android

Corda. Requeridos. Define o namespace XML, que deve ser "http://schemas.android.com/apk/res/android".

android: src

Drawable recurso. Requeridos. A referência a um arquivo de Nine-Patch.

android: dither

Booleano. Habilita ou desabilita a composição de cores do bitmap se o bitmap não tem a mesma configuração de pixel como a tela (por exemplo: um bitmap ARGB 8888 com uma tela RGB 565).

EXEMPLO:

<? xml version = "1.0" encoding = "utf-8"?>

<xmlns nove remendo: android = "http://schemas.android.com/apk/res/android"

    android: src = "@ drawable / myninepatch"

    android: dither = "false" />

Recursos Drawable

Veja também

Gráficos 2D

Um recurso drawable é um conceito geral para um gráfico que pode ser desenhado na tela e que você pode recuperar com APIs tais como getDrawable (int) ou aplicar a outro recurso XML com atributos como android: drawable e android: icon. Existem vários tipos diferentes de drawables:

Arquivo de bitmap

Um arquivo bitmap gráfico (.png, .jpg ou .gif). Cria um BitmapDrawable.

Nine-Patch Arquivo

Um arquivo PNG com regiões esticáveis ​​para permitir redimensionamento de imagens com base em conteúdo (.9.png). Cria um NinePatchDrawable.

Lista de camadas

A Drawable que gerencia uma série de outros Drawables. Estes são desenhados a fim de matriz, de modo que o elemento com o maior índice é ser desenhada em cima. Cria um LayerDrawable.

Lista de Estado

Um arquivo XML que faz referência a diferentes gráficos de bitmap para diferentes estados (por exemplo, para usar uma imagem diferente quando um botão é pressionado). Cria um StateListDrawable.

Lista nível

Um arquivo XML que define um drawable que administra uma série de Drawables alternados, cada atribuído um valor máximo numérica. Cria um LevelListDrawable.

Transição Drawable

Um arquivo XML que define um drawable que podem cruzar-fade entre dois recursos resgatáveis. Cria um TransitionDrawable.

Inset Drawable

Um arquivo XML que define um drawable que inserir outro drawable a uma determinada distância. Isso é útil quando a View precisa de um drawble fundo que é menor do que os limites reais da Veja.

Clipe Drawable

Um arquivo XML que define um drawable que clipes outro Drawable com base no valor atual nível deste Drawable. Cria um ClipDrawable.

Scale Drawable

Um arquivo XML que define um drawable que altera o tamanho de outro Drawable base no seu valor actual nível. Cria um ScaleDrawable

Forma Drawable

Um arquivo XML que define uma forma geométrica, incluindo cores e gradientes. Cria um ShapeDrawable.

Ver também o documento de Animação de recursos para como criar um AnimationDrawable.

Nota: Um recurso de cor pode também ser utilizada como um estirável em XML. Por exemplo, ao criar uma lista de estado drawable, você pode fazer referência a um recurso de cor para o android: atributo drawable (android: drawable = "@ cor / verde").

Bitmap

Uma imagem bitmap. Android suporta arquivos de bitmap em três formatos: .png (preferencial), .jpg (aceitável), .gif (desencorajado).

Você pode fazer referência um arquivo de bitmap diretamente, usando o nome do arquivo como a identificação do recurso, ou criar uma ID de recurso de alias em XML.

Nota: Os arquivos de bitmap podem ser automaticamente otimizada com compressão de imagem sem perda pela ferramenta AAPT durante o processo de criação. Por exemplo, uma verdadeira cor-PNG que não necessita de mais de 256 cores podem ser convertidos a uma 8 bits PNG com uma paleta de cores. Isto resultará numa imagem de igual qualidade, mas que requer menos memória. Então, estar ciente de que os binários de imagem colocados neste diretório pode mudar durante a compilação. Se você está pensando em ler uma imagem como um fluxo de bits, a fim de convertê-lo em um bitmap, colocar as suas imagens no res / raw / pasta em vez disso, onde não será otimizado.

Arquivo de bitmap

Um arquivo de bitmap é um .png, .jpg ou .gif. Android cria um recurso Drawable para qualquer um desses arquivos quando você salvá-los no res / drawable / diretório.

LOCAL DO ARQUIVO:

res / drawable / filename.png (.png, .jpg ou .gif)

O nome do arquivo é usado como o ID do recurso.

COMPILADA DATATYPE RECURSOS:

Ponteiro recurso a uma BitmapDrawable.

RECURSO DE REFERÊNCIA:

Em Java: R.drawable.filename

Em XML: @ [pacote]: drawable / filename

EXEMPLO:

Com uma imagem salva no res / drawable / myimage.png, este XML disposição aplica-se a imagem para um Vista:

<ImageView

    android: layout\_height = "wrap\_content"

    android: layout\_width = "wrap\_content"

    android: src = "@ drawable / myimage" />

O seguinte código de aplicativo recupera a imagem como um Drawable:

Recursos res = getResources ();

Drawable Drawable = res.getDrawable (R.drawable.myimage);

VEJA TAMBÉM:

Gráficos 2D

BitmapDrawable

Bitmap XML

Um bitmap XML é um recurso definido em XML que aponta para um arquivo de bitmap. O efeito é um apelido para um arquivo bitmap cru. O XML pode especificar propriedades adicionais para o bitmap, tais como composição de cores e ladrilhos.

Nota: Você pode usar um <bitmap> elemento como um filho de um elemento <item>. Por exemplo, ao criar uma lista de estado ou camada de lista, você pode excluir o android: atributo drawable de um elemento <item> e um ninho <bitmap> em seu interior que define o item drawable.

LOCAL DO ARQUIVO:

res / drawable / filename.xml

O nome do arquivo é usado como o ID do recurso.

COMPILADA DATATYPE RECURSOS:

Ponteiro recurso a uma BitmapDrawable.

RECURSO DE REFERÊNCIA:

Em Java: R.drawable.filename

Em XML: @ [pacote]: drawable / filename

Sintaxe:

<? xml version = "1.0" encoding = "utf-8"?>

<bitmap

    xmlns: android = "http://schemas.android.com/apk/res/android"

    android: src = "@ [pacote]: drawable / drawable\_resource"

    Android: antialias = ["true" | "false"]

    android: dither = ["true" | "false"]

    android: filter = ["true" | "false"]

    android: gravidade = ["top" | "bottom" | "esquerda" | "direito" | "center\_vertical" |

                      "fill\_vertical" | "center\_horizontal" | "fill\_horizontal" |

                      "center" | "encher" | "clip\_vertical" | "clip\_horizontal"]

    android: mipmap = ["true" | "false"]

    android: TILEMODE = ["desativado" | "clamp" | "repeat" | "espelho"] />

ELEMENTOS:

<bitmap>

Define a fonte bitmap e suas propriedades.

atributos:

xmlns: android

Corda. Define o namespace XML, que deve ser "http://schemas.android.com/apk/res/android". Isso é necessário apenas se o <bitmap> é a raiz elemento que não é necessário quando o <bitmap> está aninhado em um <item>.

android: src

Drawable recurso. Requeridos. Referência a um recurso amovível.

Android: antialias

Booleano. Ativa ou desativa o anti-serrilhamento.

android: dither

Booleano. Habilita ou desabilita a composição de cores do bitmap se o bitmap não tem a mesma configuração de pixel como a tela (por exemplo: um bitmap ARGB 8888 com uma tela RGB 565).

android: Filtro

Booleano. Habilita ou desabilita a filtragem de bitmap. A filtragem é usada quando o bitmap é reduzido ou esticado para suavizar sua aparência.

android: gravidade

Palavra-chave. Define a gravidade para o mapa de bits. A gravidade indica onde posicionar o amovível no seu recipiente, se o mapa de bits é menor do que o recipiente.

Deve ser um ou mais (separadas por '|') dos seguintes valores constantes:

Valor Descrição

topo Coloque o objeto na parte superior do recipiente, não mudar o seu tamanho.

inferior Coloque o objeto na parte inferior do seu recipiente, não mudar o seu tamanho.

deixou Coloque o objeto no lado esquerdo de seu contêiner, não mudar o seu tamanho.

direito Coloque o objeto na borda direita de seu recipiente, não mudar o seu tamanho.

center\_vertical Lugar objeto no centro vertical de seu recipiente, não mudar o seu tamanho.

fill\_vertical crescer o tamanho vertical do objeto, se necessário para que ele preenche completamente o seu recipiente.

center\_horizontal Lugar objeto no centro horizontal de seu recipiente, não mudar o seu tamanho.

fill\_horizontal aumentar o tamanho horizontal do objecto, se necessário para que ele preenche completamente o seu recipiente.

colocar o objecto de centro no centro do recipiente, tanto o eixo vertical e horizontal, sem alterar o seu tamanho.

preencher crescer o tamanho horizontal e vertical do objeto, se necessário para que ele preenche completamente o seu recipiente. Este é o padrão.

clip\_vertical opção adicional que pode ser configurado para ter a parte superior e / ou bordas inferiores da criança cortada para limites do seu contêiner. O clipe é baseado na gravidade verticais: Grampos uma gravidade superior a borda inferior, um clips de gravidade inferior da borda superior, e nem grampos ambas as bordas.

clip\_horizontal opção adicional que pode ser ajustado para ter à esquerda e / ou direito bordas da criança cortada para limites do seu contêiner. O clipe é baseado na gravidade horizontal: clipes de uma gravidade esquerda da borda direita, um clips de gravidade direito a borda esquerda, e nem grampos ambas as bordas.

android: mipmap

Booleano. Habilita ou desabilita a dica mipmap. Veja setHasMipMap () para mais informações. O valor padrão é falso.

android: TILEMODE

Palavra-chave. Define o modo de telha. Quando o modo de telha é habilitado, o bitmap é repetido. A gravidade é ignorado quando o modo de telha está habilitado.

Deve ser um dos seguintes valores constantes:

Valor Descrição

com deficiência não telha o bitmap. Esse é o valor padrão.

braçadeira Reproduz a cor da borda se o shader empates fora de seus limites originais

repeat Repete a imagem do shader horizontalmente e verticalmente.

espelho Repete a imagem do shader horizontalmente e verticalmente, alternando imagens de espelho para que as imagens adjacentes sempre costura.

EXEMPLO:

<? xml version = "1.0" encoding = "utf-8"?>

<xmlns bitmap: android = "http://schemas.android.com/apk/res/android"

    android: src = "@ drawable / ícone"

    android: TILEMODE = "repetir" />

VEJA TAMBÉM:

BitmapDrawable

Criação de recursos de alias

Eu ouvi recentemente sobre imagens de 9 de patch. Eu sei que é 9 azulejos e é extensível . Eu gostaria de saber mais sobre ele .

Como posso criar uma imagem de 9 patch?

Existe alguma ferramenta? Posso criar -lo do Android SDK ou código?

Principais vantagens do 9 -patch sobre png regular?

(é extensível dinamicamente / automaticamente de acordo com a tela? )

## Top 5 Responder

**1**[CommonsWare](http://stackoverflow.com/users/115145/" \t "_blank)@2013-06-15 14:40:17

Como posso criar uma imagem de 9 patch? Existe alguma ferramenta? Posso criar -lo do Android SDK ou código?

Quando você procura um grande motor de pesquisa android 9-patch tool, o primeiro hit é parahttp://developer.android.com/tools/help/draw9patch.html.

Principais vantagens do 9 -patch sobre png regular? (é extensível dinamicamente / automaticamente de acordo com a tela? )

A documentação para desenvolvedores Android contém outras páginas que http://developer.android.com/guide/topics/graphics/2d-graphics.html#nine-patch . Esta documentação inclui passagens como :

Um gráfico NinePatchDrawable é uma imagem bitmap extensível, que Android vai redimensionar automaticamente para acomodar o conteúdo do Vista em que você colocou -lo como plano de fundo. Um exemplo do uso de um NinePatch é os fundos utilizados por botões padrão do Android - botões devem esticar para acomodar cordas de vários comprimentos .

**2**[Petr Krejč&#237;](http://stackoverflow.com/users/2487168/" \t "_blank)@2013-06-16 22:42:55

Image 9 -patch Com você pode escolher qual parte da sua imagem pode ser esticada . Deve ser PNG imagem e nome deve terminar com .9.png ( something.9.png )

http://developer.android.com/tools/help/draw9patch.html

Ferramenta simples para todas as densidades :

http://android-ui-utils.googlecode.com/hg/asset-studio/dist/nine-patches.html

**3**[Nick Westgate](http://stackoverflow.com/users/313445/" \t "_blank)@2014-03-09 10:09:12

Hoje eu descobri nove imagens de patch. A documentação ligada a em outras respostas é ok eu suponho, mas eles realmente não têm em exemplos. http://tekeye.biz/2012/android-9patch-files tem alguns grandes exemplos do final que responder à segunda parte da sua pergunta, explicando como funciona a escala - e não apenas para os botões -, mas também quadros. O patch nove pode especificar áreas não contíguas à escala.

Apenas para ilustrar as respostas para a primeira parte da sua pergunta, os navios SDK com " draw9patch " (na pasta ferramentas ), que é um editor simples . http://weblookandfeel.com/nine-patch-editor/ um download que é bom, e ele tem uma imagem padrão simples, mas inteligente .

**4**[gnB](http://stackoverflow.com/users/2162226/" \t "_blank)@2014-03-31 18:29:35

A maioria dos exemplos falar sobre a criação de uma imagem de 9 -patch, mas os detalhes de implementação normalmente são deixados em um nível elevado .

http://stackoverflow.com/a/22280925/2162226 - com as http://tekeye.biz/2012/android-9patch-files que fornece um arquivo de download do projeto de trabalho, salvou o dia .

Aqui estão os principais detalhes de implementação que trabalharam para mim (uma vez que você tem uma imagem de 9 -patch pronto para ir ):

Referência o drawable com o nome, mas não incluem .9.png (auto -complete em eclipse vai cuidar do presente )

Certifique-se que você só tem 1 imagem sob a principal pasta drawable / ( não uma versão para cada pasta dpi)

A imagem deve ser especificado usando: fundo, não : src (isso me ficou preso por um tempo)

draw9patch

Verifique se a imagem e layout que o contém está usando:

android:background="@drawable/splash\_logo"

mipmap vs pastas drawable [duplicado]

up voto

147

voto para baixo

favorito

23

Esta questão já tem uma resposta aqui:

drawables mipmap para ícones 7 respostas

Eu estou trabalhando com android estúdio 1.1 Preview 1

E eu notei que quando eu criar um novo projeto que estou ficando próxima hierarquia:

entrar descrição da imagem aqui

pastas mipmap para diffrent DPIs, não há pastas drawable mais diffrent DPIs.

Devo colocar todos os meus recursos nas pastas mipmap, ou apenas o ícone do aplicativo?

android-estúdio mipmaps android-drawable android

shareimprove esta pergunta

editado 21 de janeiro às 11:08

perguntou 21 de janeiro às 10:48

David

6,551115285

marcado como duplicado por Selvin, Duggu, Código L ღ ver, Max, shkschneider 21 de janeiro às 16:35

Esta pergunta foi feita antes e já tem uma resposta. Se essas respostas não resolver completamente a sua pergunta, por favor, fazer uma nova pergunta.

3

Confira esta declaração a partir do google ligação local do blog abaixo: - "... porque eles são usados ​​em resoluções diferentes de densidade de corrente do dispositivo Por exemplo, um ícone de aplicação xxxhdpi pode ser usado no lançador para um dispositivo xxhdpi.". Isto diz que alguns dispositivos explicitamente tenta usar ícones de diferentes densidades e não o que a densidade de dispositivo atual especifica. por exemplo. algum fabricante pode gostar de usar ícones maiores na tela inicial do que a tela lançador, ou ícones menores para as notificações do que do que a especificação de costume. Devemos identificar, assim, tais recursos e colocá-los em pastas mipmap. - Rahul Thakur 21 de abril em 10:59

O comentário acima é apenas uma suposição sobre o que eu entendo e pode estar errado. - Rahul Thakur 21 de abril às 11:00

Adicione um comentário

1 Resposta

activeoldestvotes

up voto

212

voto para baixo

aceitaram

As pastas mipmap são para colocar seus ícones de aplicativos em apenas. Quaisquer outros ativos drawable que você usa deve ser colocado nas pastas drawable relevantes como antes.

De acordo com este blogpost Google:

É melhor prática para colocar seus ícones de aplicativos em pastas mipmap- (não as pastas drawable-), porque eles são usados ​​em resoluções diferentes de densidade de corrente do dispositivo.

Ao fazer referência as pastas mipmap- garantir que você está utilizando a seguinte referência:

android: icon = "@ mipmap / ic\_launcher"

shareimprove esta resposta

editado dia 17 de maio 20:08

David Passmore

2,87512143

respondeu 21 de janeiro às 11:06

Joel

4,5493923

13

para definir o ícone no android manifesto: icon = "@ mipmap / ic\_launcher" - asthme 05 de abril às 17:30

40

Para esclarecer, os "ícones de aplicativos" na seção citado nos referindo ao lançador de ícones, não necessariamente outros ícones. - CommonsWare 19 de maio em 00:35

2

"porque eles são usados ​​em resoluções diferentes de densidade de corrente do dispositivo", o que isso significa realmente? Essa é uma das frases mais enigmáticas nesta versão. - Southerton 26 de junho às 15:17

3

Quando você estiver usando-os em seu código se certificar de que você está usando (R.mipmap.icon\_name) não (R.drawable.icon\_name). - Sajidur Rahman 12 de julho em 7:11

3

southerton A razão que eles usam uma densidade diferente é que alguns lançadores de realmente mostrar os ícones maiores que se destinam. Devido a isso, eles usam o próximo tamanho máximo. - Richard Le Mesurier 21 de julho em 13:08

## [Personalizando fontes no Android](http://www.mobits.com.br/2011/5/20/personalizando-fontes-no-android)

* Autor AUTOR   **Afonso Junior**
* Tags TAGs   [**android sdk**](http://www.mobits.com.br/tags/android%20sdk),[**desenvolvimento**](http://www.mobits.com.br/tags/desenvolvimento), [**layout**](http://www.mobits.com.br/tags/layout)
* Comentários [**10**](http://www.mobits.com.br/2011/5/20/personalizando-fontes-no-android#comments)COMENTÁRIOS

Embora o pacote de fontes padrão do [Android](http://www.android.com/) sirva para a maioria dos propósitos, há vezes em que você precisa fazer telas mais incrementadas, e isso inclui mexer nas fontes. Por conta disso, você pode precisar alterar alguns atributos das fontes, ou até mesmo utilizar fontes personalizadas.

Veja abaixo algumas opções de personalizações de fontes:

### Typeface

O Android possui três tipos padrões de fonte, ou famílias: Droid Sans, Droid Sans Mono e Droid Serif. Para utilizar uma delas, basta informar no atributo **android:typeface** o tipo a ser utilizado. Veja no exemplo a seguir as três famílias (mais o tipo normal) sendo utilizadas:

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/texto\_fonte1" android:textSize="25sp"

android:text="Utilizando fonte Sans"

android:typeface="sans"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/texto\_fonte2" android:textSize="25sp"

android:text="Utilizando fonte Serif"

android:typeface="serif"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/texto\_fonte3" android:textSize="25sp"

android:text="Utilizando fonte Sans"

android:typeface="monospace"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

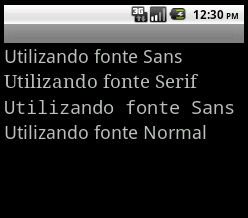
android:id="@+id/texto\_fonte4" android:textSize="25sp"

android:text="Utilizando fonte Normal"

android:typeface="normal"

/>

O resultado obtido:



### Style

O estilo da fonte serve para dar ênfase ao texto. Pode ser normal, **negrito** ou itálico.

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

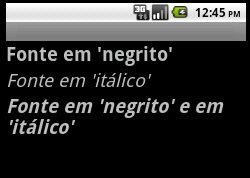
android:id="@+id/texto\_fonte1" android:textSize="25sp"

android:text="Fonte em 'negrito'"

android:textStyle="bold"

/>

Para aplicar negrito e itálico ao mesmo tempo, basta utilizar "bold|italic".



### Tamanho da fonte

O atributo **android:textsize** especifica o tamanho da fonte. Seu valor é constituído de duas partes: um número de ponto flutuante e sua unidade de medida. A unidade pode ser em **sp** (scaled pixels, ou escala de pixels), **px** (pixels), **dp** (density-independent pixels, ou pixels de densidade independente),**in** (inches, polegadas) e **mm** (milímetros). O mais aconselhável é utilizar o **sp**, assim o tamanho da fonte é redimensionada de acordo com as propriedades de visualização do usuário.

<TextView

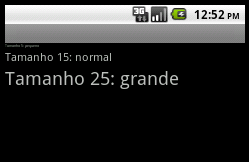
android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/texto\_fonte2" android:textSize="15sp"

android:text="Tamanho 15: normal"

/>



### Sombra

Abaixo, exemplo de como aplicar sombra:

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/texto\_fonte1" android:textSize="25sp"

android:text="Texto com sombra verde"

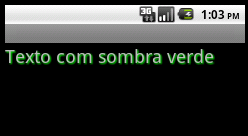
android:shadowColor="#00FF00"

android:shadowDx="1"

android:shadowDy="1"

android:shadowRadius="1.2"

/>



Você pode usar quatro diferentes atributos para configurar a sombra de uma fonte:

* **android:shadowColor**: cor da sombra
* **android:shadowRadius**: o raio da sombra
* **android:shadowDx**: o distanciamento horizontal da sombra em relação ao texto
* **android:shadowDy**: o distanciamento vertical da sombra em relação ao texto

### Usando outras fontes

Vamos utilizar como exemplo a fonte [Mail Ray Stuff](http://www.1001freefonts.com/MailRayStuff.php). Salve ela na pasta **assets** do seu projeto (se a pasta não existir, crie). A [TextView](http://developer.android.com/reference/android/widget/TextView.html) deve ter o seguinte conteúdo:

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/hello"

android:id="@+id/texto\_fonte"

android:textSize="25sp"/>

</TextView>

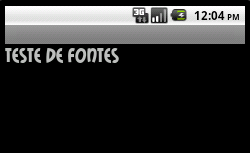
No método **onCreate()**, coloque o seguinte código:

TextView txt = (TextView) findViewById(R.id.texto\_fonte);

Typeface font = Typeface.createFromAsset(getAssets(), "mailrays.ttf");

txt.setTypeface(font);

O código é bem simples de se entender: a classe TypeFace tem o método **createFromAsset**. Ele recebe como parâmetro a pasta default dos **assets** e o nome do arquivo de fonte a ser utilizado.



Vale ressaltar que quando se fala de fontes customizáveis, tenha sempre em mente que elas podem não funcionar corretamente, principalmente se ela não possuir todos os caracteres especiais que a localização pode vir a exigir.