



TREINAMENTOS

Desenvolvimento Mobile com Android

Android

21 de março de 2016

As apostilas atualizadas estão disponíveis em www.k19.com.br

Sumário	i
1 Introdução	1
1.1 O que é o Android?	1
1.2 O que é necessário para desenvolver um aplicativo Android?	1
1.3 Exercícios de Fixação	1
2 Visão Geral	9
2.1 Hello World	9
2.2 Exercícios de Fixação	9
3 Interfaces	21
3.1 Layouts	21
3.2 Views	21
3.3 Exercícios de Fixação	32
3.4 ViewGroups	35
3.5 Exercícios de Fixação	40
3.6 Strings	50
3.7 Exercícios de Fixação	51
3.8 Dimensões	55
3.9 Exercícios de Fixação	56
3.10 Cores	58
3.11 Exercícios de Fixação	59
3.12 Dialogs	61
3.13 Exercícios de Fixação	62
4 Usando Intents	65
4.1 O que é um <i>Intent</i>	65
4.2 Usando o <i>Intent</i> com <i>Activities</i>	65
4.3 Exercícios de Fixação	65
4.4 Usando <i>Intents</i> para passar dados	67
4.5 Exercícios de Fixação	68
4.6 Abrindo outros aplicativos	70

4.7	Exercícios de Fixação	71
5	Persistência de dados	73
5.1	Persistindo informação	73
5.2	Usando o <i>SharedPreferences</i>	73
5.3	Exercícios de Fixação	74
5.4	Usando o <i>SQLite</i>	77
5.5	Exercícios de Fixação	77
6	HTTP e JSON	83
6.1	HTTP	83
6.2	JSON	83
6.3	Exercícios de Fixação	84
7	Threads e AsyncTasks	89
7.1	<i>Threads</i> e <i>Handlers</i>	89
7.2	Exercícios de Fixação	89
7.3	<i>AsyncTasks</i>	92
7.4	Exercícios de Fixação	92
8	Services e BroadcastReceiver	97
8.1	Serviços	97
8.2	Exercícios de Fixação	100
8.3	<i>Broadcast Receivers</i>	104
8.4	Exercícios de Fixação	104
9	Notificações	107
9.1	<i>Dialogs</i>	107
9.2	Exercícios de Fixação	107
9.3	<i>Notifications</i>	110
9.4	Exercícios de Fixação	110
10	Mapas e GPS	113
10.1	Utilizando o GPS	113
10.2	Exercícios de Fixação	113
10.3	Usando o <i>MapView</i>	116
10.4	Exercícios de Fixação	117
11	Apêndice - Multimedia	121
11.1	Introdução	121
11.2	Reprodução de Mídia	121
11.3	Exercícios de Fixação	127
11.4	Captura de Áudio	130
11.5	Exercícios de Fixação	132
11.6	Captura de Vídeo	136
11.7	Exercícios de Fixação	145
12	Apêndice - AppWidgets	149
12.1	Introdução	149
12.2	Principais classes	149
12.3	Manifest	149

12.4	Configurando o AppWidgetProviderInfo	150
12.5	Definindo o Layout	150
12.6	Exercícios de Fixação	151
13	Apêndice - Publicando no Google Play	155
13.1	Como gerar um aplicativo	155
13.2	Exercícios de Fixação	155
14	Apêndice - Sensores	161
14.1	O Básico	161
14.2	Sensores de Movimento	163
14.3	Sensores de Posição	166
14.4	Sensores de Ambiente	167
14.5	Exercícios de Fixação	168
15	Apêndice - Web Apps com WebView	171
15.1	Introdução	171
15.2	Manifest	171
15.3	Layout	171
15.4	Carregando uma página	171
15.5	Controlando a Navegação	172
15.6	Associando código JavaScript a código Android	173
15.7	Exercícios de Fixação	174
16	Apêndice - Bluetooth	177
16.1	Classes utilizadas	177
16.2	Permissões	177
16.3	Usando o Bluetooth	177
16.4	Listando dispositivos pareados	178
16.5	Descobrimos dispositivos	178
16.6	Usando conexões	178
17	Apêndice - Animações	183
17.1	Exercícios de Fixação	183
18	Apêndice - Monetizando com Ads	187
18.1	Introdução	187
18.2	Conta de Veículo	187
18.3	SDK do serviço de Ad	187
18.4	Manifest	188
18.5	Layout	189
18.6	Inicializando o Ad	189





O que é o Android?

Atualmente, o Android é o sistema operacional mais utilizado em smartphones e tablets. Também é utilizado em televisores, carros, relógios e outros dispositivos. Ele é desenvolvido pela Google e é baseado no núcleo do Linux.

Android pode ser visto também como uma plataforma que permite a criação de aplicativos para uma grande variedade de dispositivos, principalmente, smartphones e tablets.



O que é necessário para desenvolver um aplicativo Android?

As ferramentas e programas necessários para desenvolver um aplicativo Android são todos gratuitos e disponíveis para os sistemas operacionais OS X, Windows e Linux.

Para começar a programar para o Android, é necessário conhecer a linguagem Java, pois essa é a linguagem utilizada para desenvolver aplicativos.

Entre as ferramentas que você precisa instalar estão:

JDK 7 - Java Development Kit 7

Android SDK - Contém as bibliotecas da plataforma Android e diversas ferramentas para o desenvolvimento de aplicativos

Android Studio - Principal IDE utilizada para o desenvolvimento de aplicativos Android

Você não é obrigado a ter um aparelho com Android pois o SDK inclui um emulador que permite executar os aplicativos desenvolvidos. Porém, é altamente recomendado executar esses aplicativos em um aparelho real.



Mais Sobre

Os aplicativos Android são empacotados em arquivos APK. Basicamente, um arquivo APK contém o código compilado e os demais recursos como XMLs e imagens utilizados em um aplicativo.

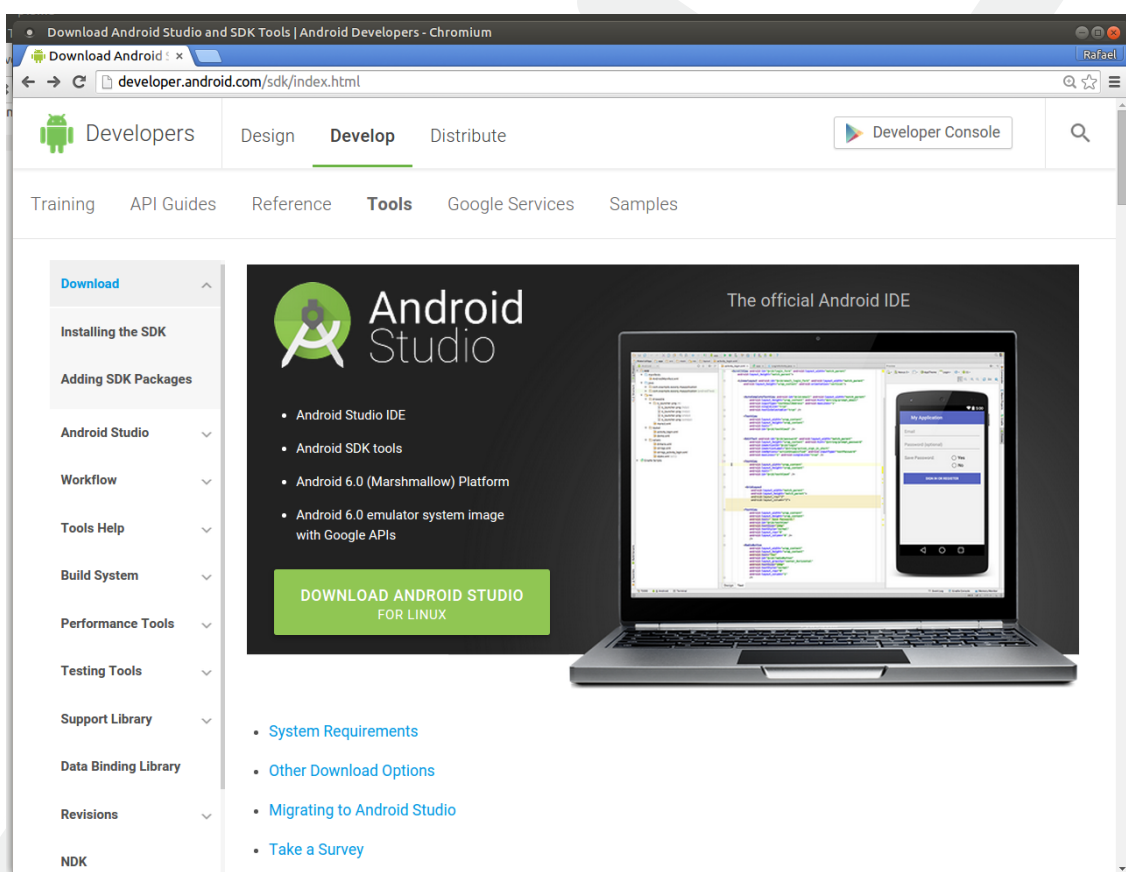


Exercícios de Fixação

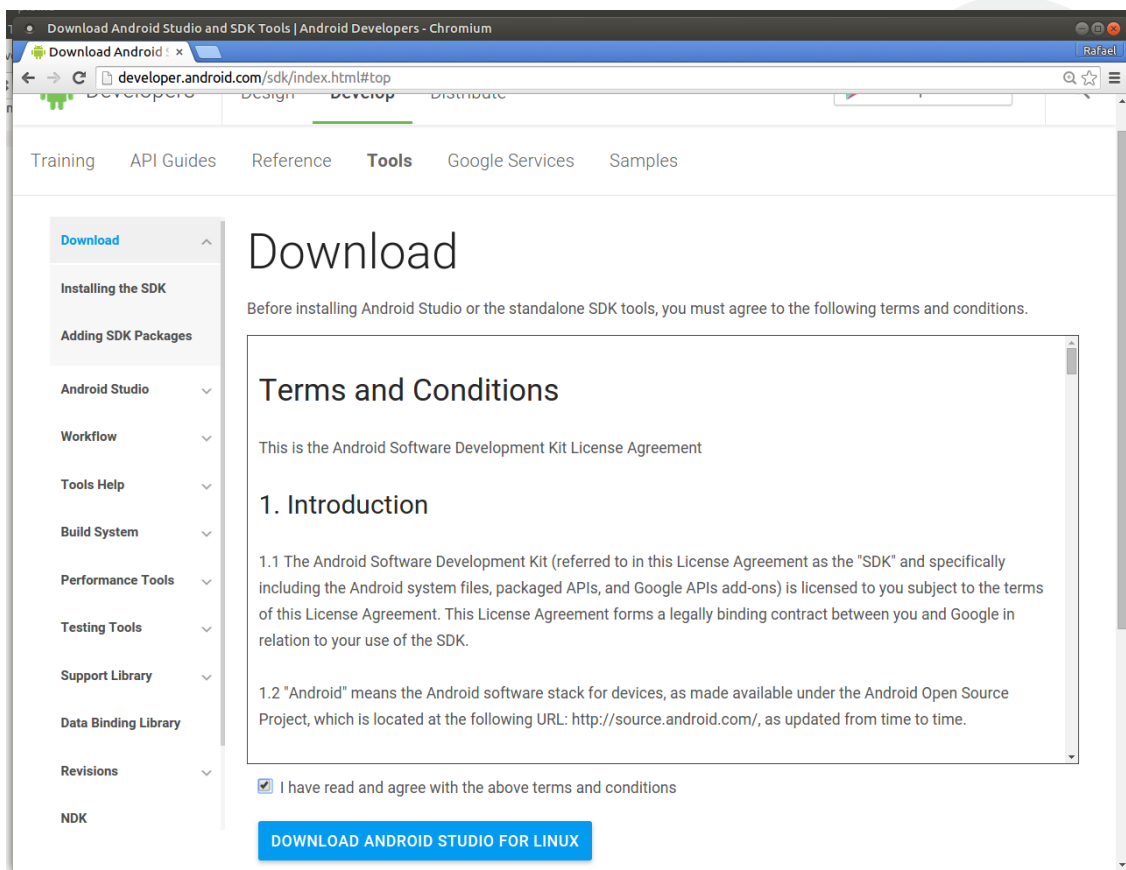
- 1 Para instalar o Android Studio e o Android SDK, acesse o endereço a seguir

<http://developer.android.com/sdk/index.html>

Clique no botão para realizar o download correspondente ao seu sistema operacional.

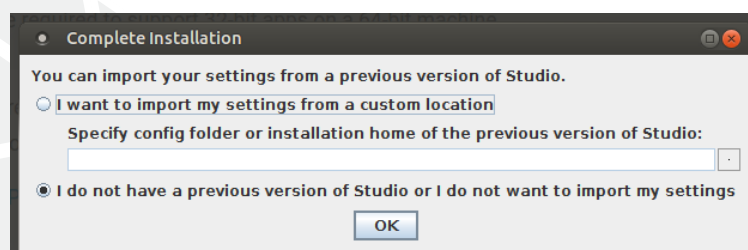


Aceite os termos e as condições e clique no botão para iniciar o download correspondente ao seu sistema operacional.

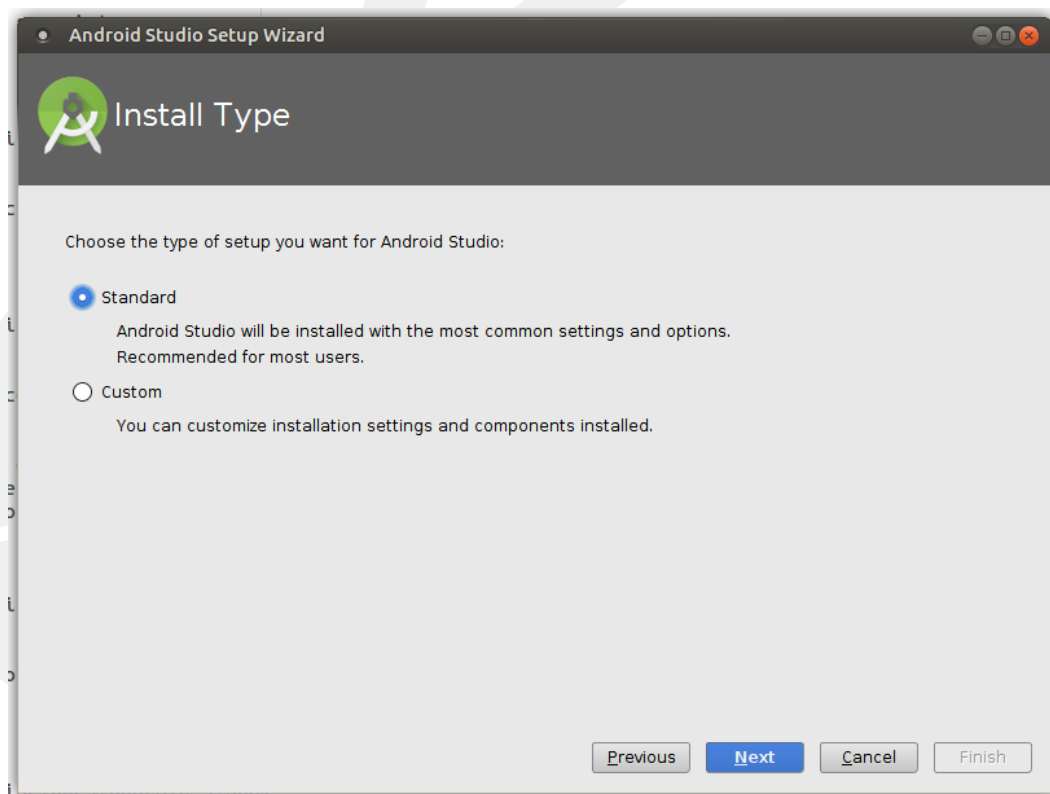
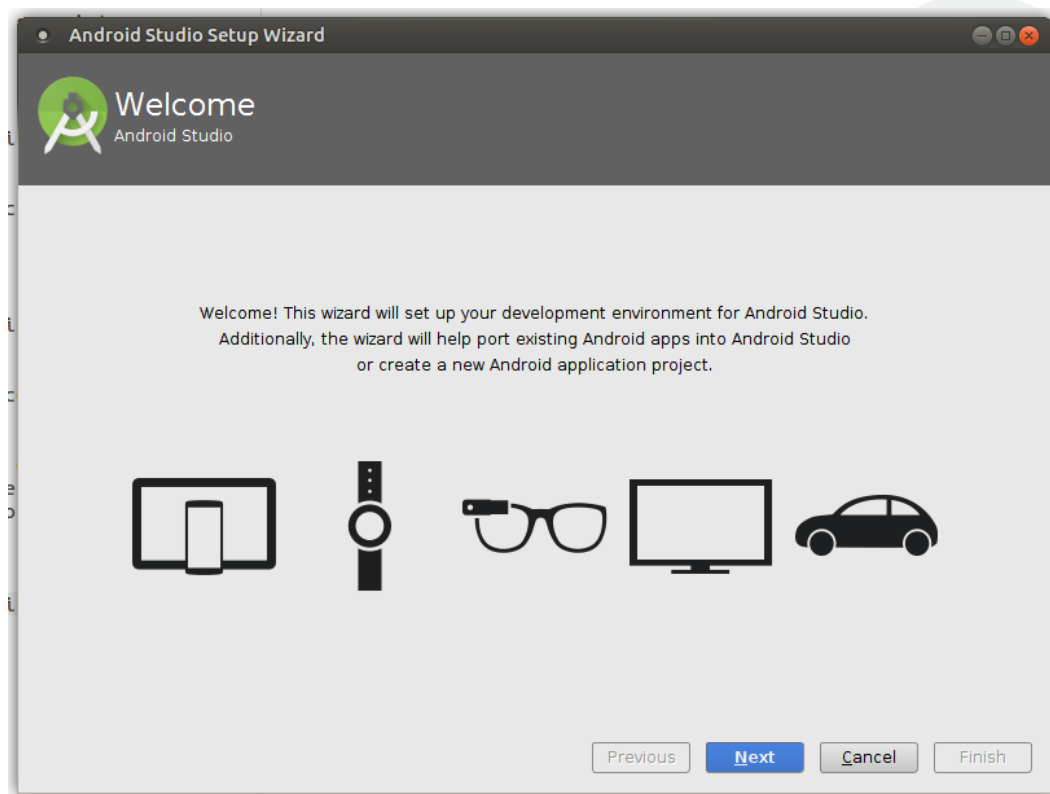


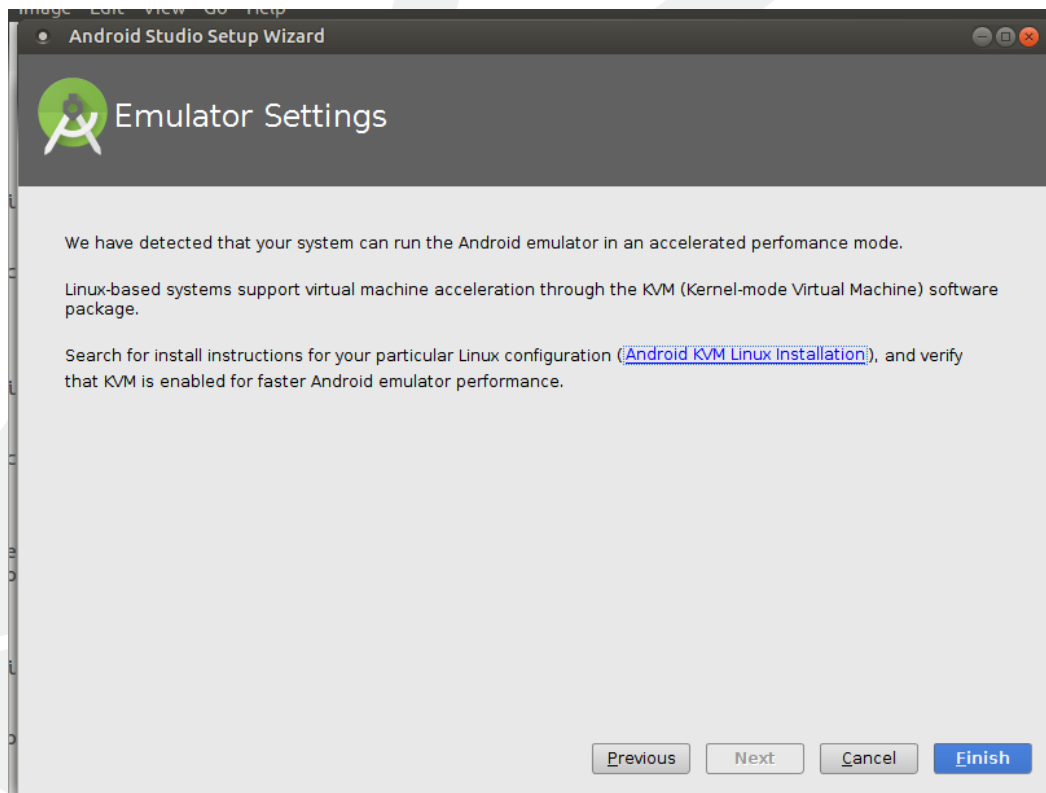
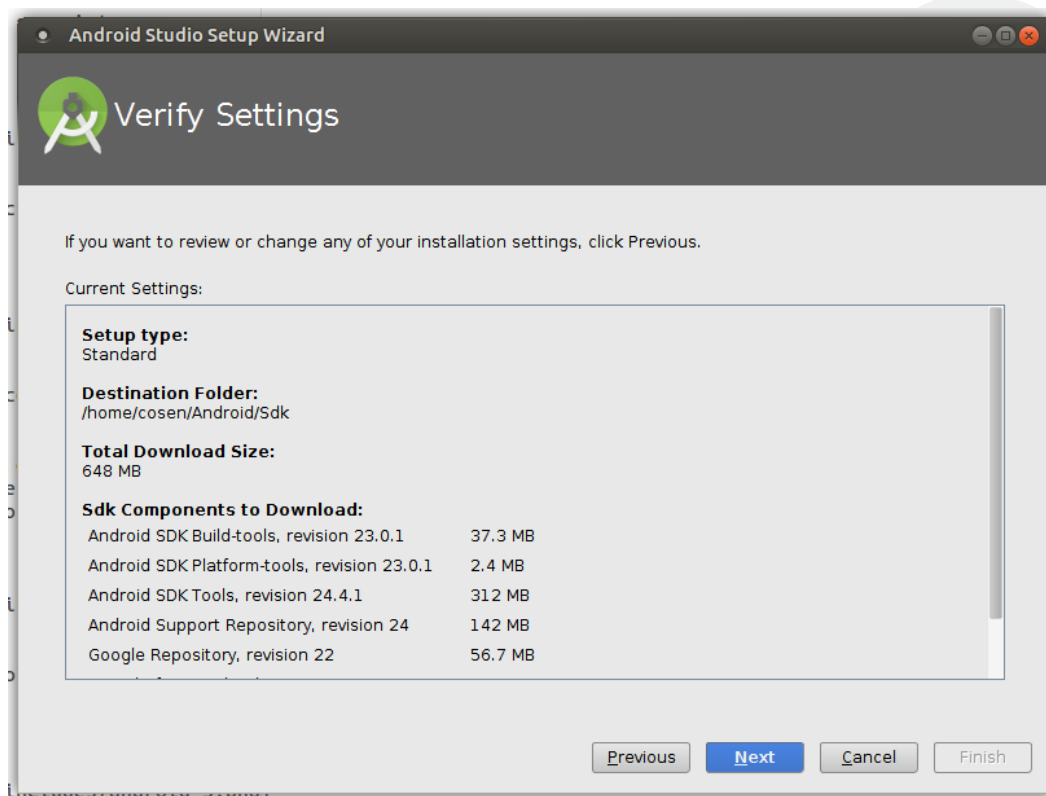
Ao término do download, descompacte o arquivo obtido se necessário, execute o Android Studio e aguarde a inicialização dessa IDE.

Se você deseja importar configurações de uma versão antiga do Android Studio selecione a primeira opção, indique a pasta de instalação e clique no botão OK. Caso contrário, selecione a segunda opção e clique no botão OK.

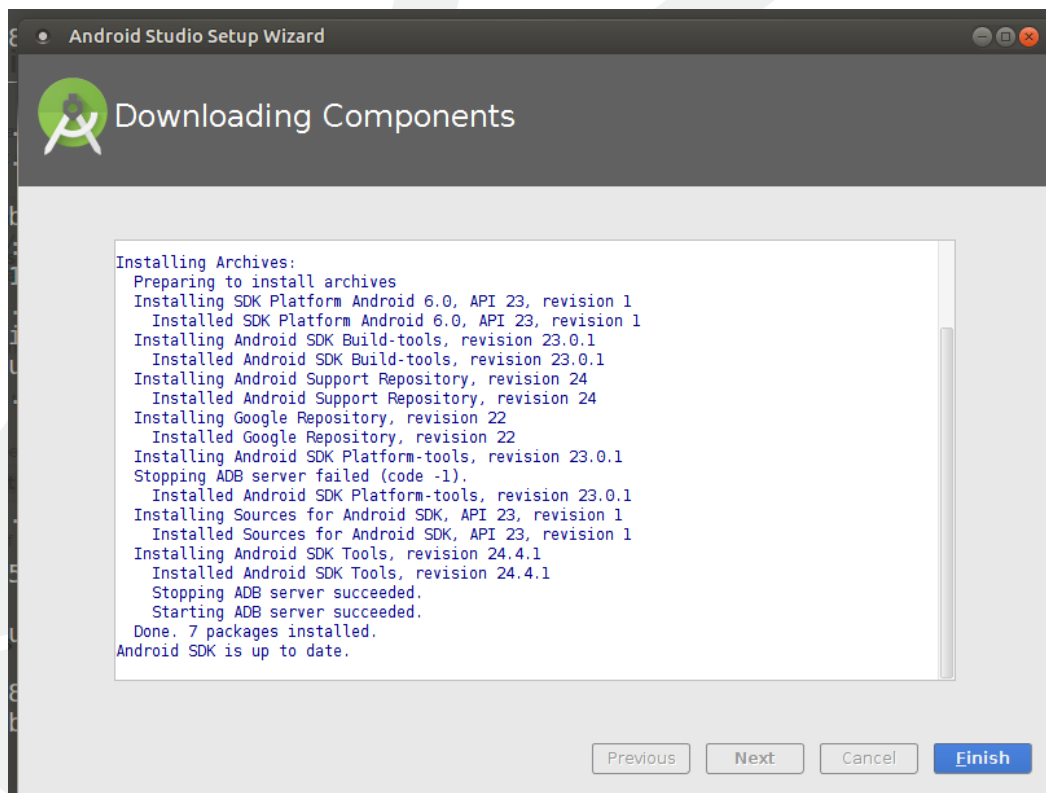
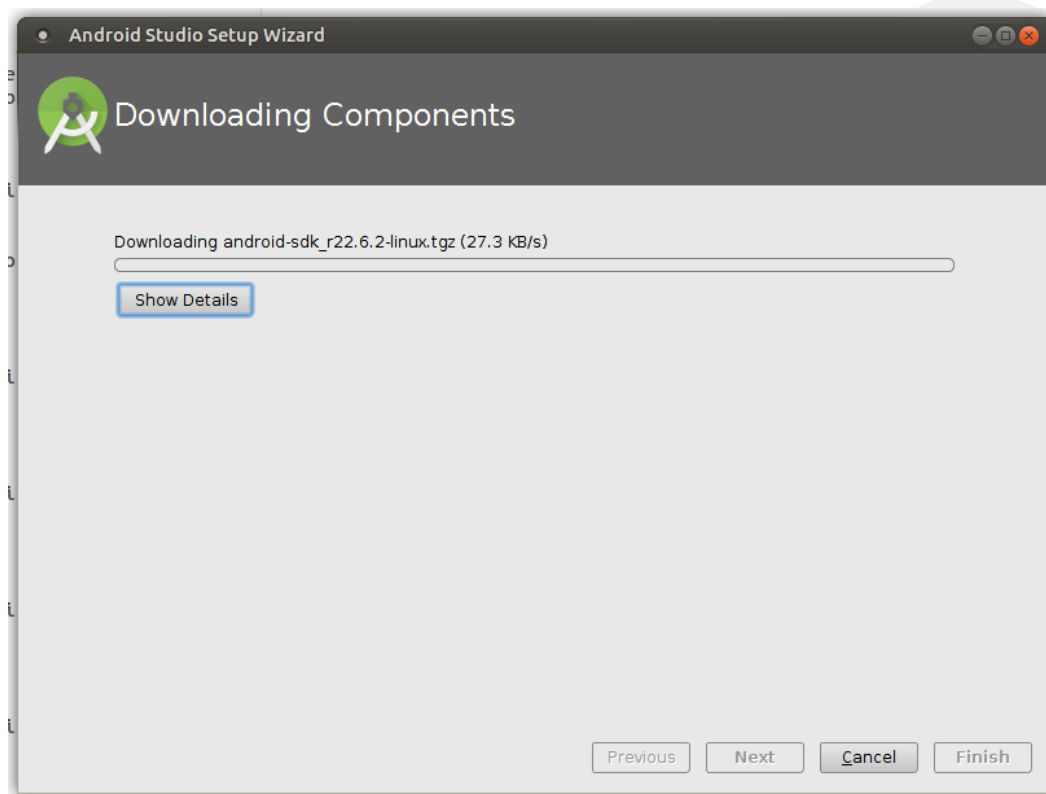


Siga os passos abaixo para configurar o ambiente de desenvolvimento para o Android Studio.

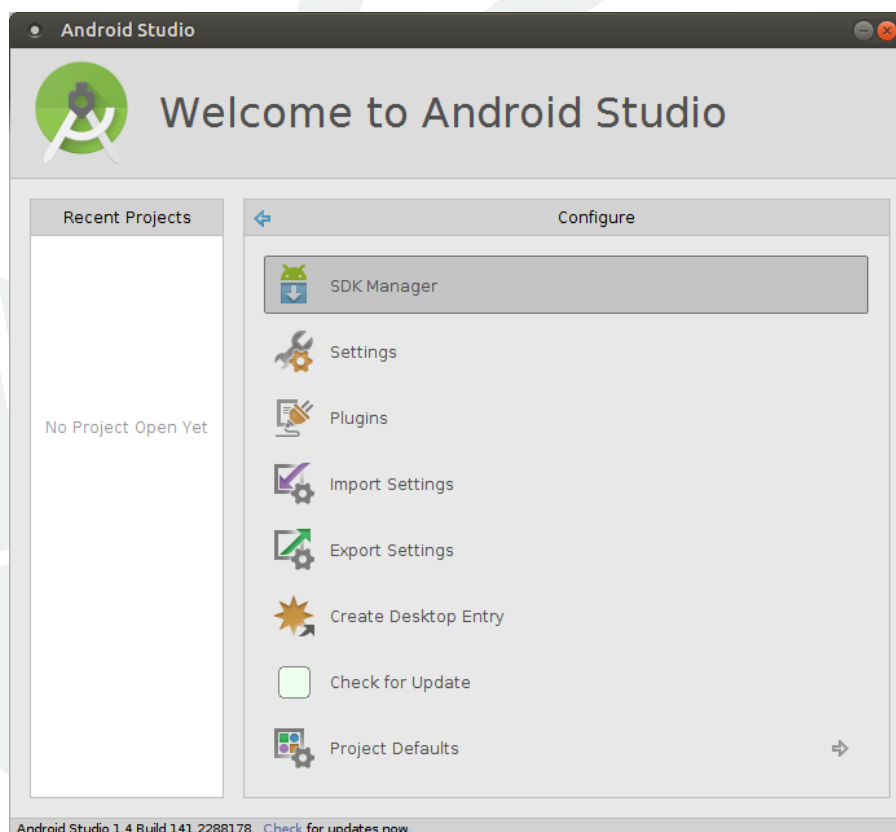
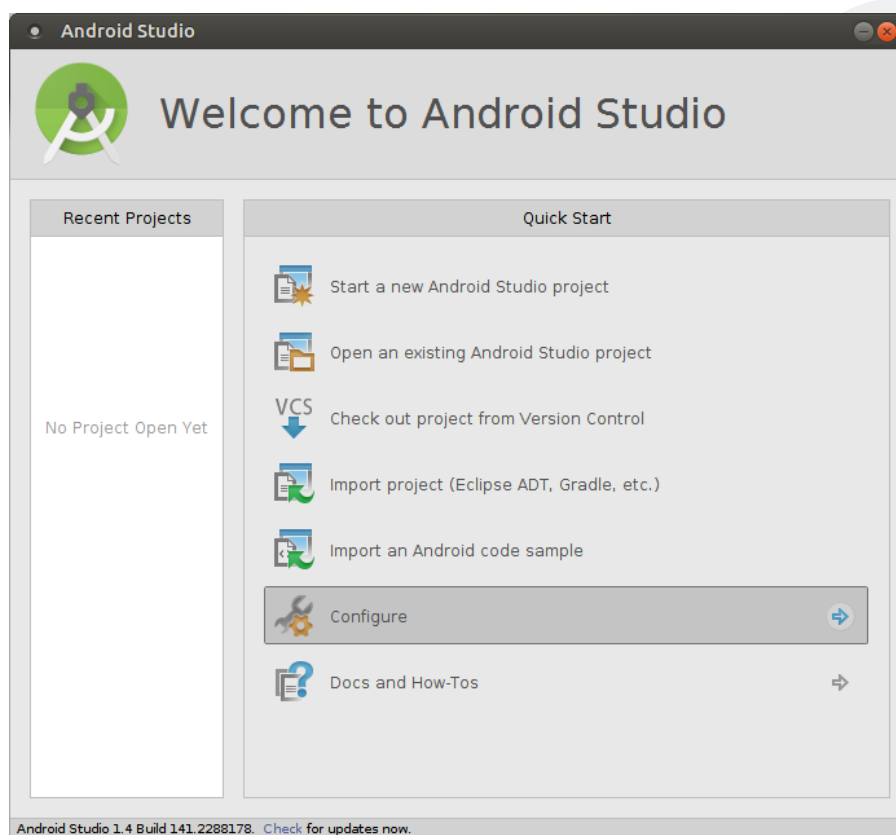




Aguarde o download do Android SDK e o término da instalação.



O próximo passo é configurar o Android SDK. Alguns pacotes devem ser instalados.



Em **SDK Platforms** selecione os pacotes:

- Android 6.0 platform
- Google APIs (API Level 23)
- Google APIs Intel x86 Atom System Image
- Sources for Android 23
- Android 4.0.3 Platform
- Intel x86 Atom System Image
- Google APIs (API Level 15)
- Sources for Android 15

Em **SDK Tools** selecione os pacotes:

- Android SDK Build Tools
- Android SDK Tools (última versão)
- Android SDK Platform-Tools (última versão)
- Documentation for Android SDK
- Android Support Repository (última versão)
- Android Support Library (última versão)
- Google Play services (última versão)
- Google Repository (última versão)
- Intel x86 Emulator Accelerator (HAXM installer) (última versão)



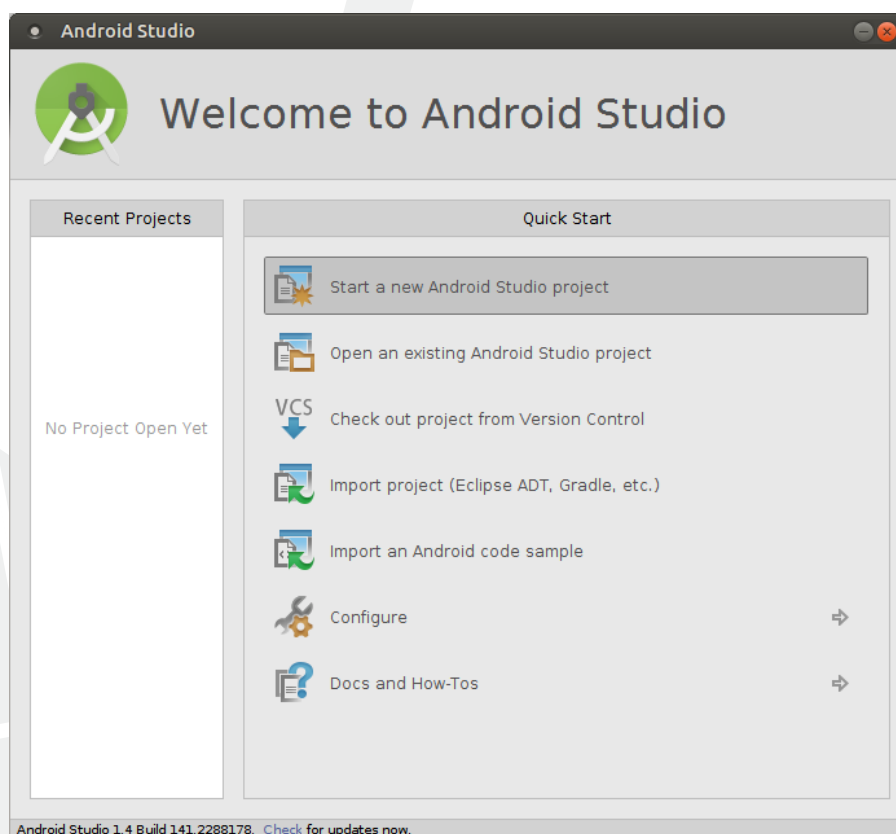
Hello World

Vamos testar o ambiente de desenvolvimento criando um aplicativo simples que apenas mostra uma mensagem na tela.



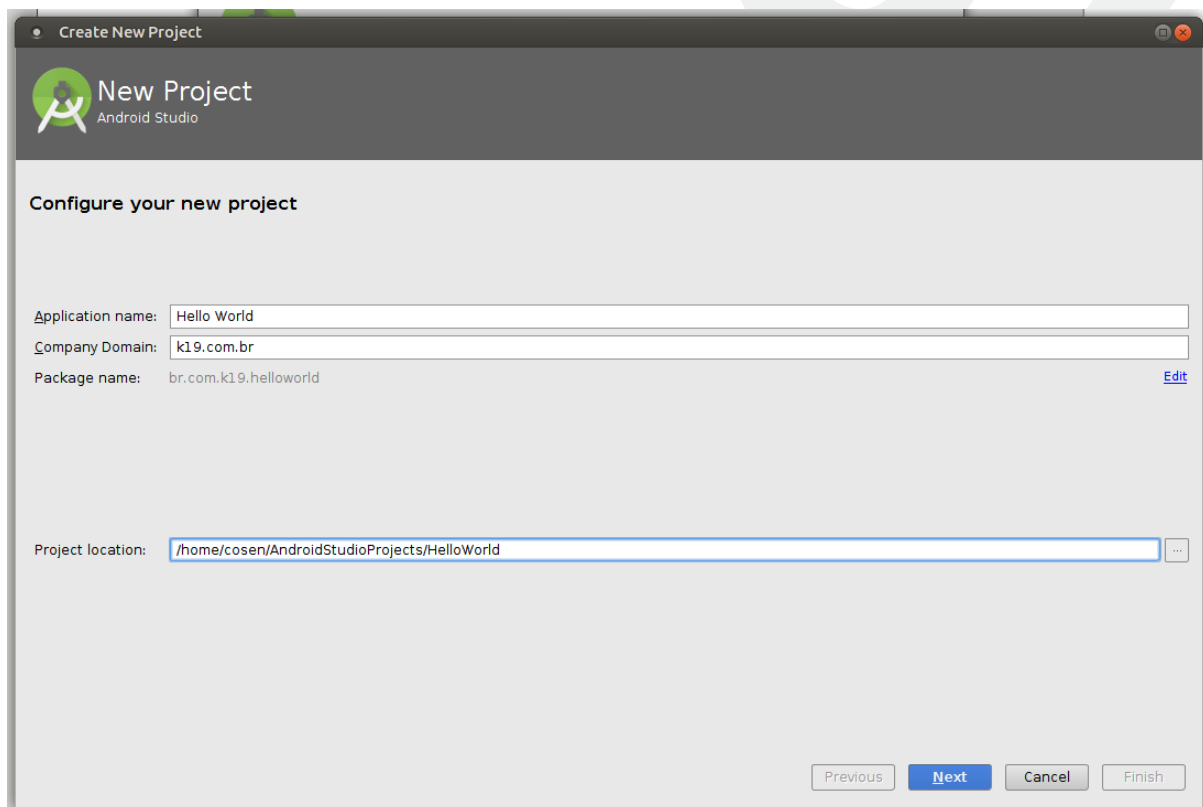
Exercícios de Fixação

- 1 Abra o Android Studio e crie um projeto para desenvolver um aplicativo. Siga as imagens abaixo.

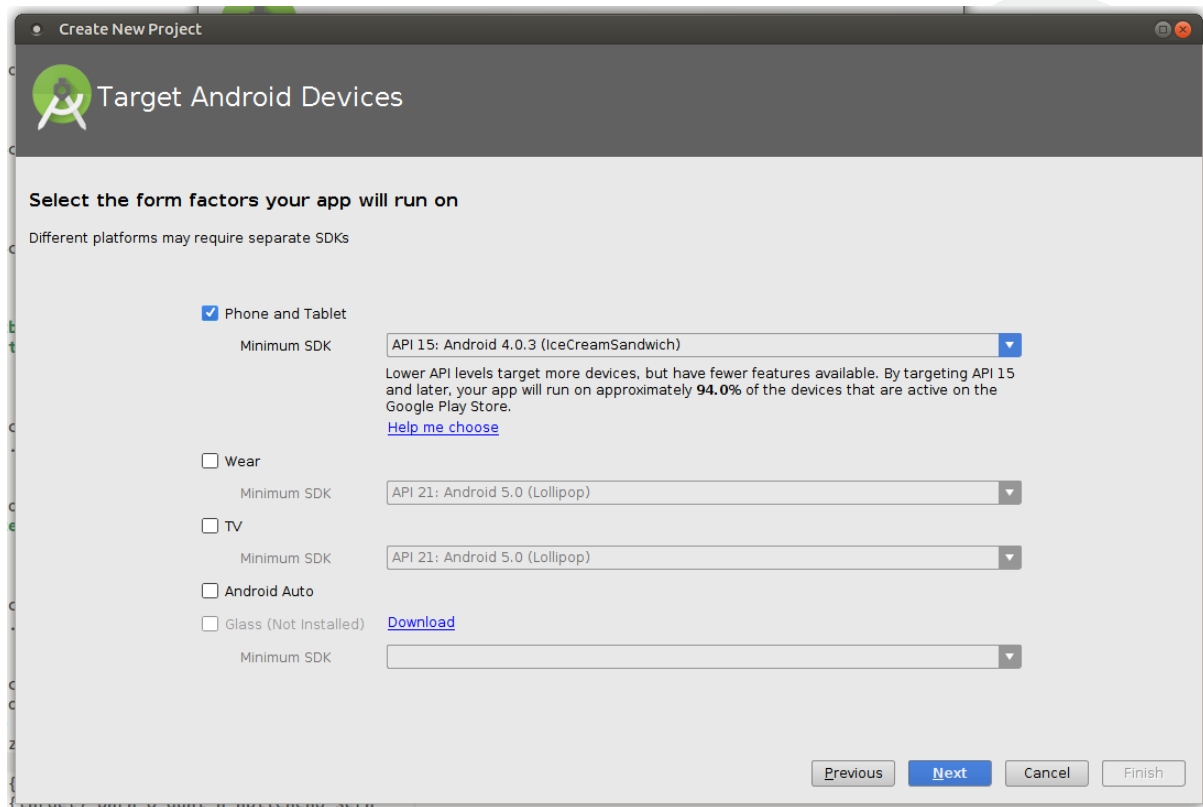


Na próxima tela, defina o conteúdo dos campos “Application Name” e “Company Domain”. Por padrão, o conteúdo do campo “Application Name” é utilizado como legenda do ícone de inicialização do aplicativo (launcher). Além disso, esse conteúdo em conjunto com o conteúdo do campo

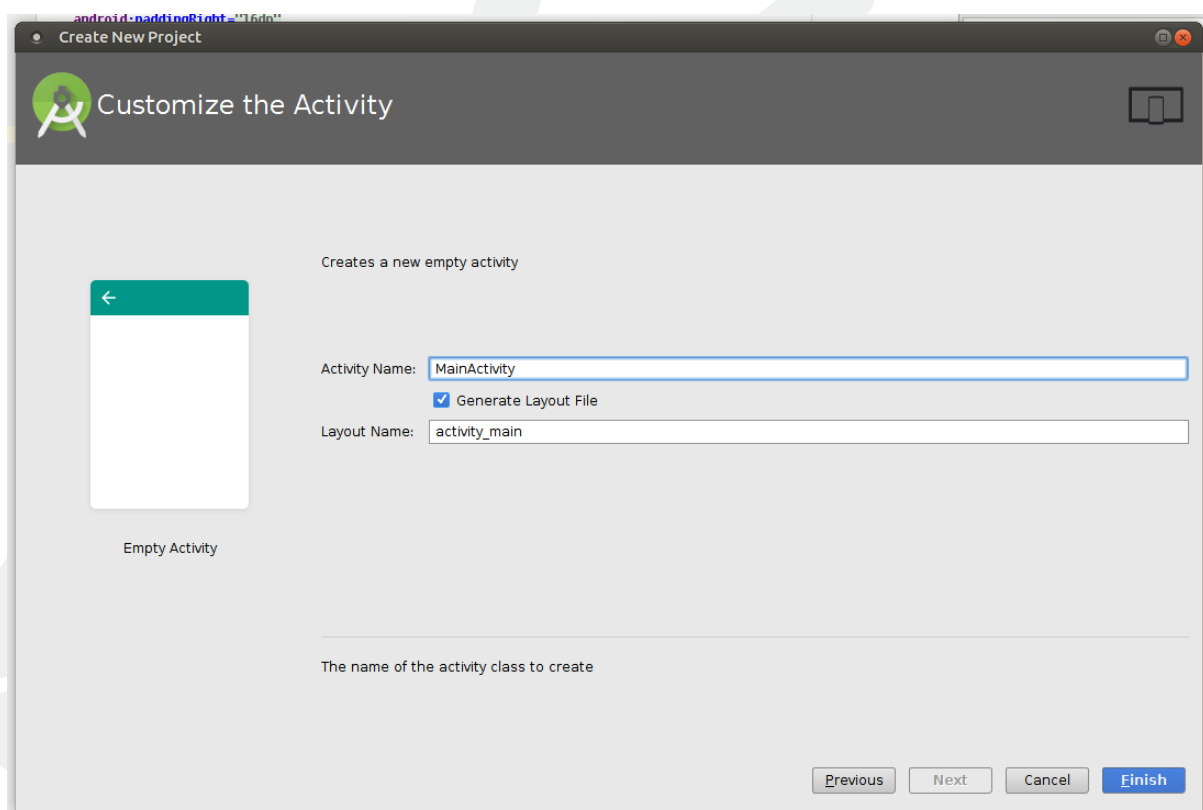
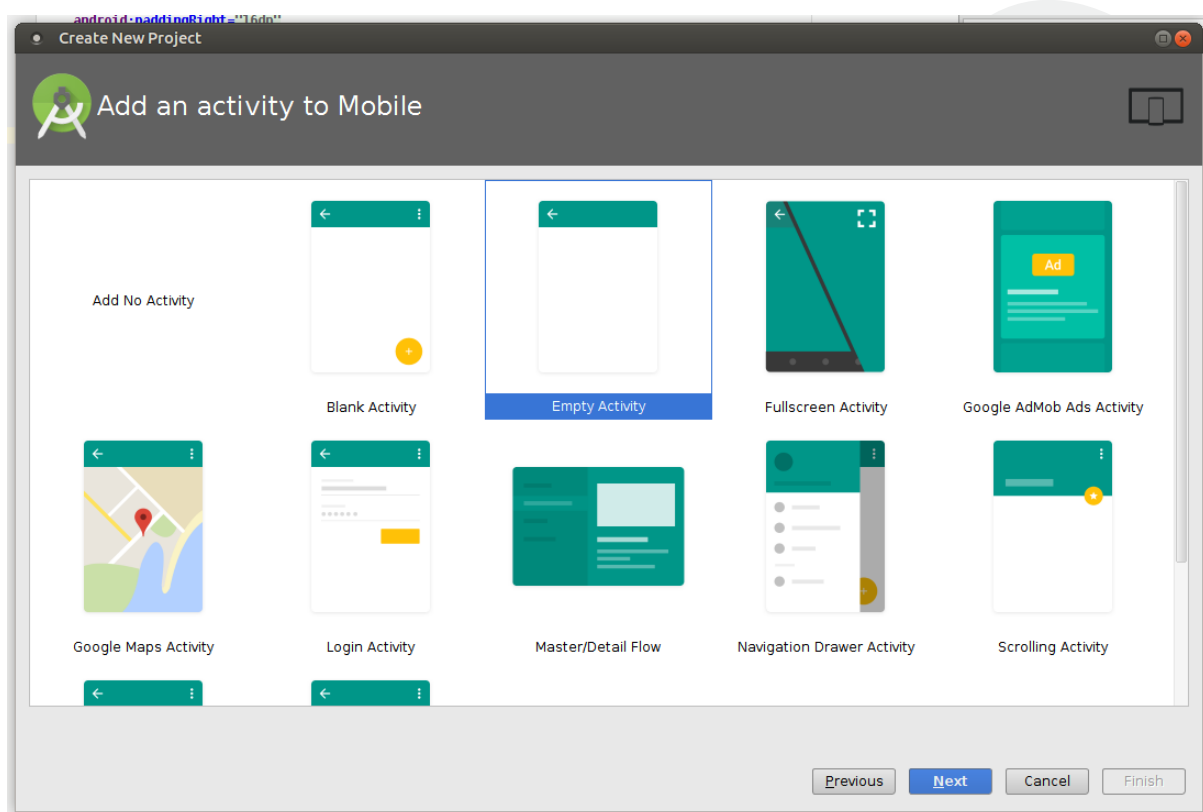
“Company Domain” são utilizados para definir o “Package Name”. O Package Name identifica unicamente um aplicativo dentro da Play Store (loja oficial de aplicativos). Em outras palavras, dois aplicativos na Play Store não podem possuir o mesmo Package Name.



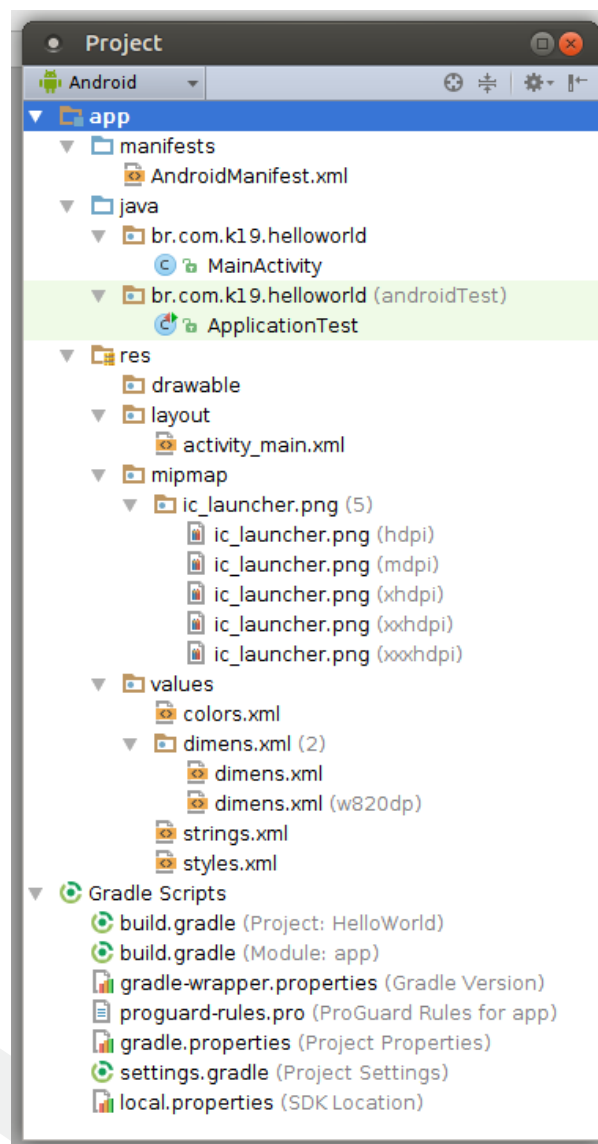
Na tela seguinte, é necessário selecionar para quais tipos de dispositivos e versões da plataforma Android o aplicativo será desenvolvido. Selecionar versões mais antigas da plataforma Android permite atender um maior número de usuários. Contudo, algumas funcionalidades específicas das versões mais novas não estarão disponíveis. O Android Studio informa a porcentagem de dispositivos ativos na Play Store que poderão executar o aplicativo de acordo com a versão selecionada da plataforma Android.



Nas próximas telas, para que o aplicativo tenha pelo menos uma tela, adicione e configure uma activity. Selecione o template “Empty Activity”. Por enquanto, considere que uma activity é uma tela do aplicativo.



- 2 Observe a estrutura do projeto criado com o Android Studio.



AndroidManifest.xml é o arquivo que guarda as configurações principais do aplicativo. Veremos mais a fundo quais são as opções de configuração no decorrer do curso.

java é a pasta onde fica o código fonte java do aplicativo.

res é a pasta onde os recursos do aplicativo são armazenados. Há vários tipos de recursos. Por isso, essa pasta é subdividida em várias pastas.

res/drawable é a pasta destinada a armazenar a maioria das imagens que são usadas no aplicativo.

res/layout é a pasta utilizada para armazenar os arquivos XML que definem o layout das telas do aplicativo.

res/menu é a pasta utilizada para armazenar os arquivos XML que definem o comportamento e itens dos menus do aplicativo.

res/mipmap é a pasta utilizada para armazenar o ícone principal do aplicativo.

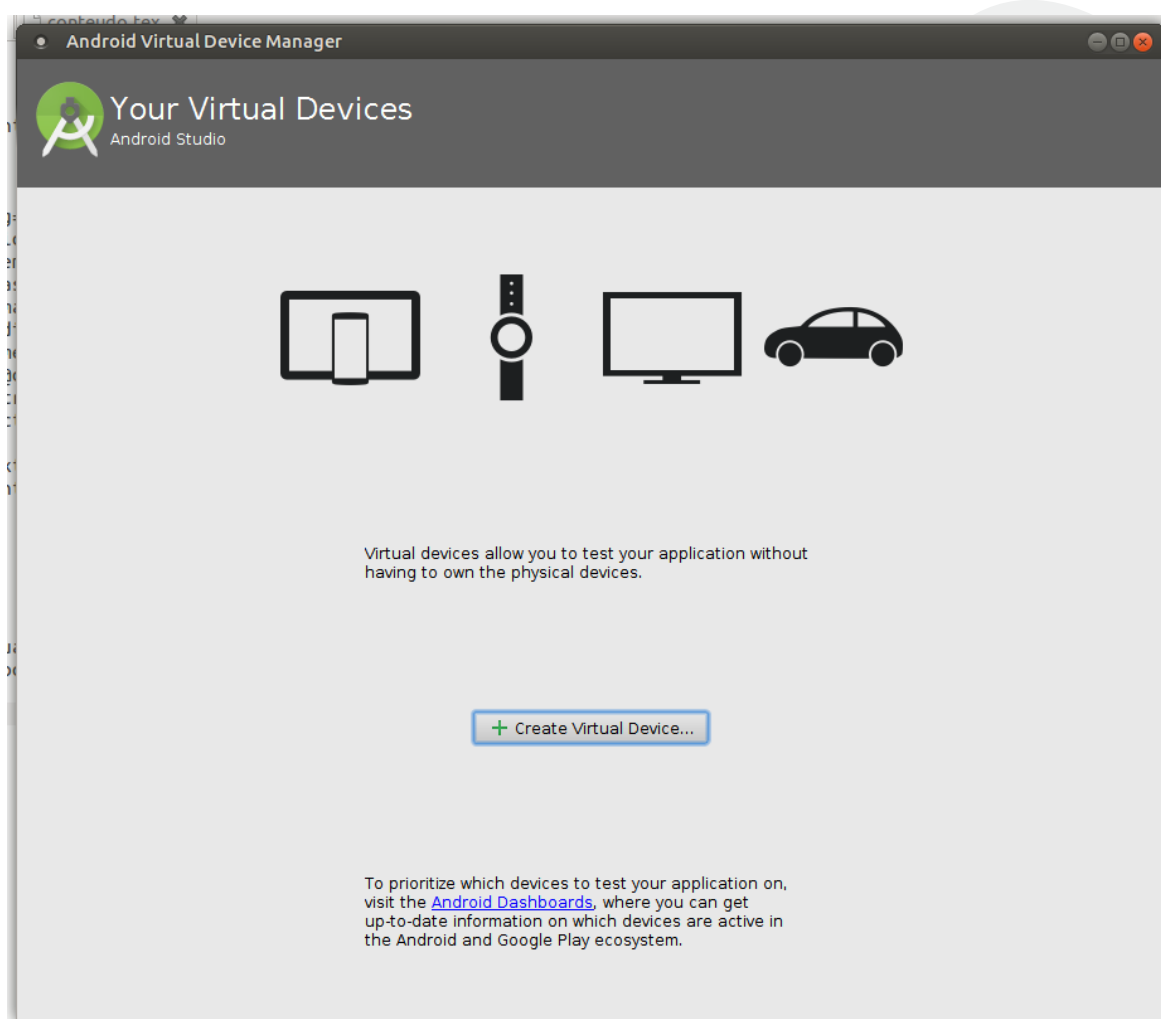
res/values é a pasta onde são armazenados os arquivos XML que serão usados para definir temas, dicionários de palavras em um ou mais idiomas, paletas de cores e guias de medidas, por exemplo.

- 3 Altere o arquivo **activity_main.xml**. Esse arquivo está na pasta **res/layout**.

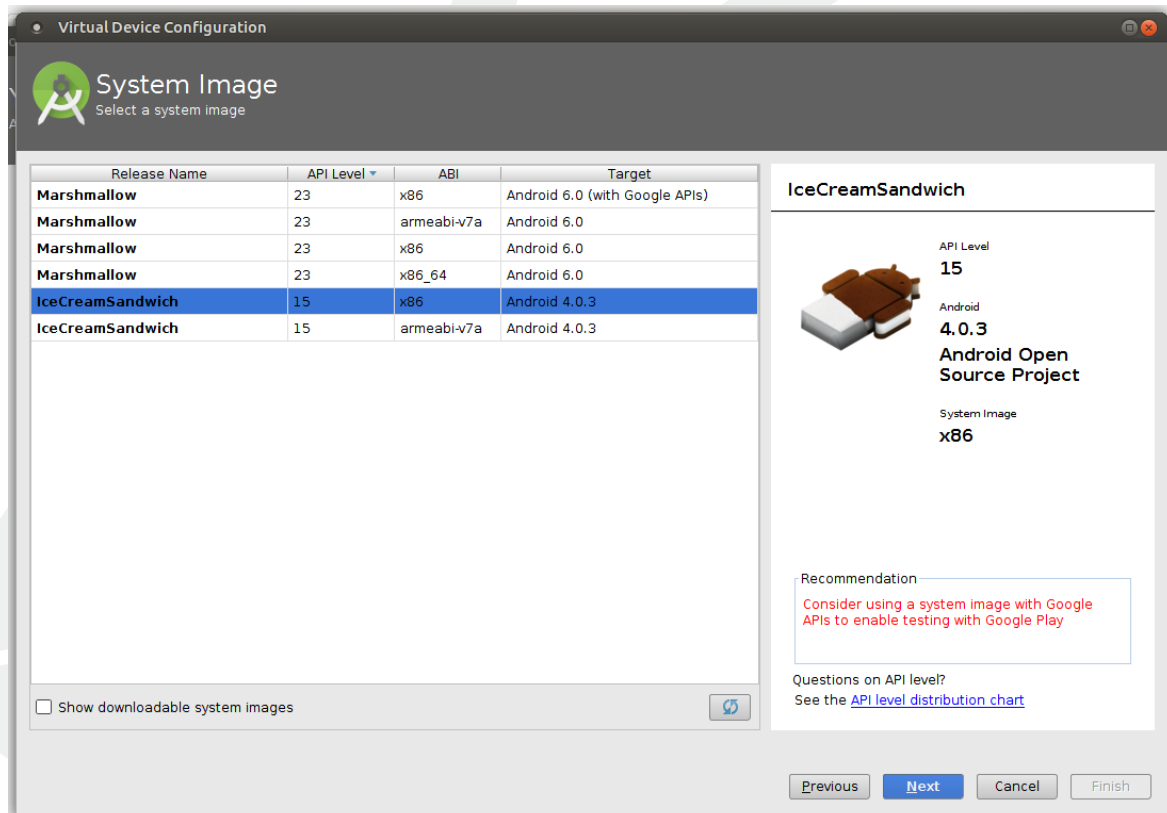
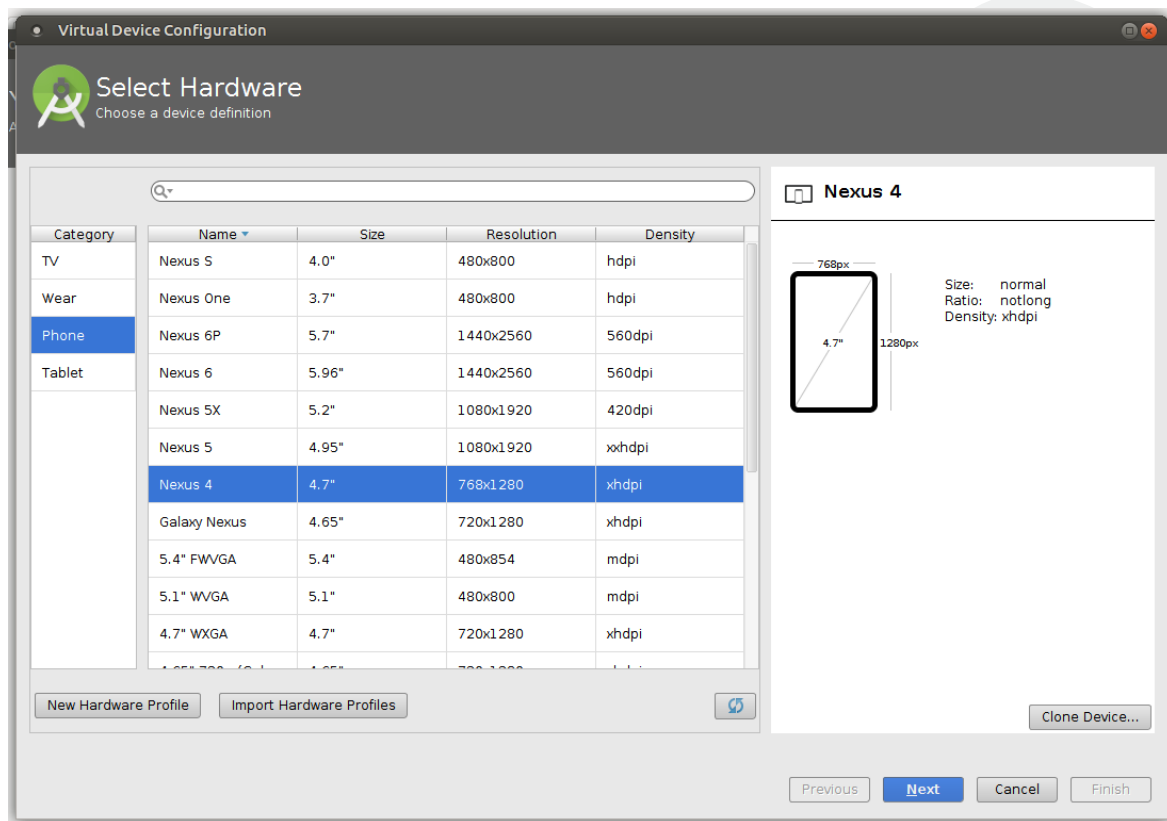
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:paddingBottom="@dimen/activity_vertical_margin"
8     android:paddingLeft="@dimen/activity_horizontal_margin"
9     android:paddingRight="@dimen/activity_horizontal_margin"
10    android:paddingTop="@dimen/activity_vertical_margin"
11    tools:context=".MainActivity">
12
13    <TextView
14        android:layout_width="wrap_content"
15        android:layout_height="wrap_content"
16        android:text="Hello World K19!" />
17 </RelativeLayout>
```

Código XML 2.1: activity_main.xml

- 4 Agora, crie um Android Virtual Device (AVD) para executar e testar o aplicativo. Selecione a opção **Tools->Android->AVD Manager** e siga os passos abaixo.

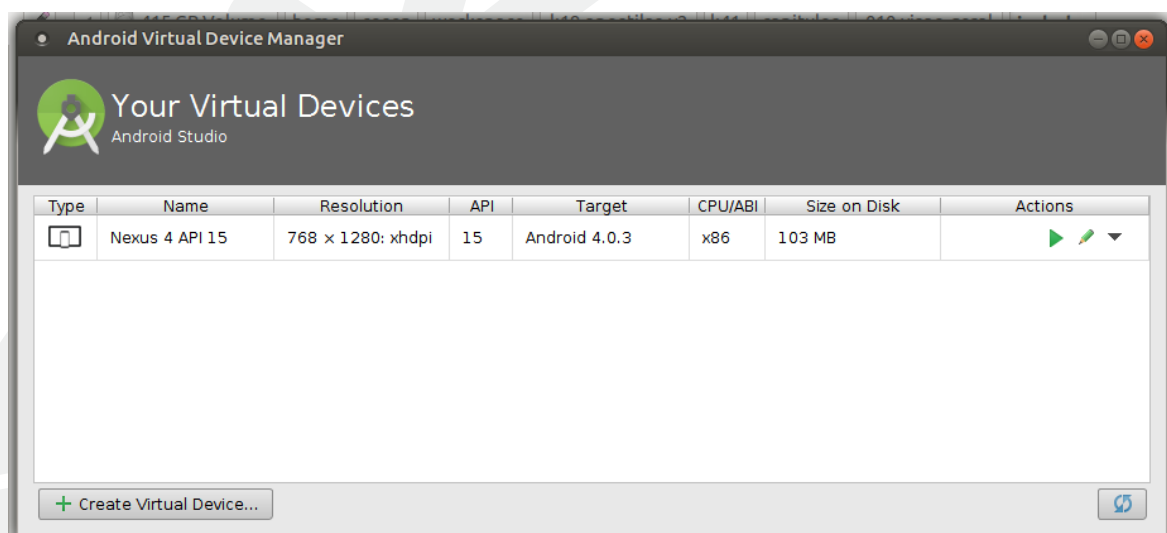
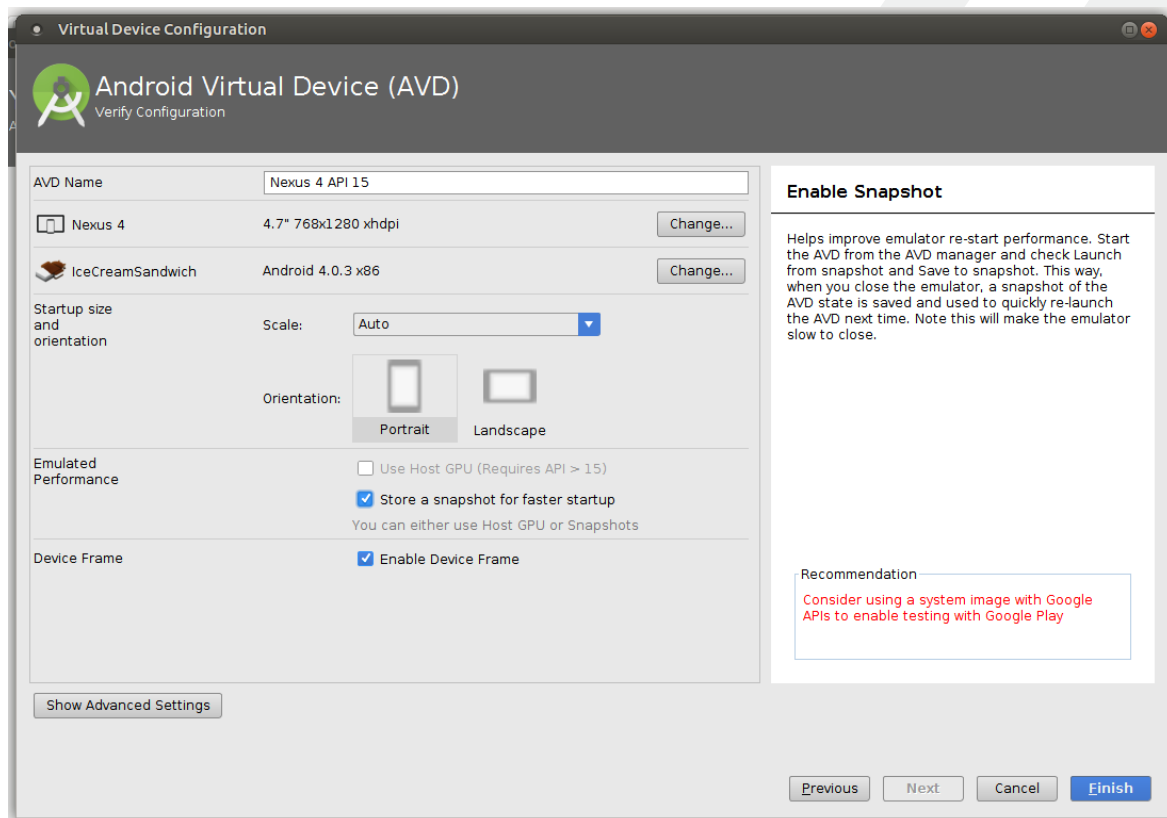


Nas próximas telas, podemos seleccionar uma definição de dispositivo e uma imagem da plataforma Android para o AVD.

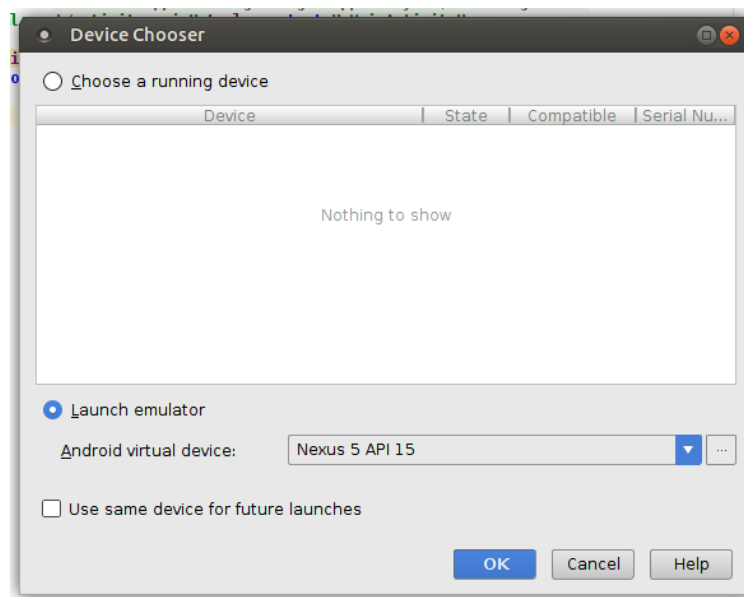


Na tela seguinte, podemos configurar diversas características do AVD. Marque a opção "Store a

snapshot for faster startup” para diminuir o tempo de inicialização do AVD.



- 5 Execute o aplicativo. Para isso, selecione a opção **Run->Run 'app'**. Depois, marque o AVD criado anteriormente e clique no botão OK. Aguarde a inicialização completa do AVD e do aplicativo.



Vamos entender, passo a passo, o que aconteceu ao executar o projeto com o Android Studio.

1. O código java compilado e os demais recursos do aplicativo são empacotados em um arquivo APK. Esse arquivo APK é enviado para o dispositivo selecionado.
2. O aplicativo é instalado no dispositivo a partir do arquivo APK. Nessa instalação, o dispositivo processa o arquivo `AndroidManifest.xml` para obter algumas informações do aplicativo como o Package Name, ícone, rótulo e tema. Observe a seguir as linhas 4, 8, 9 e 11 do arquivo `AndroidManifest.xml`.
3. O aplicativo é executado automaticamente. Observe a seguir o trecho da linha 12 até a linha 19 do arquivo `AndroidManifest.xml`. Note que a `MainActivity` foi registrada como activity principal do aplicativo pois ela foi associada a um intent filter com ação `MAIN` e categoria `LAUNCHER`. Quando um aplicativo é executado, a activity principal desse aplicativo é processada.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     package="br.com.k19.helloworld" >
5
6     <application
7         android:allowBackup="true"
8         android:icon="@mipmap/ic_launcher"
9         android:label="@string/app_name"
10        android:supportRtl="true"
11        android:theme="@style/AppTheme" >
12        <activity
13            android:name=".MainActivity" >
14            <intent-filter>
15                <action android:name="android.intent.action.MAIN" />
16
17                <category android:name="android.intent.category.LAUNCHER" />
18            </intent-filter>
19        </activity>
20    </application>
21 </manifest>

```

Código XML 2.2: AndroidManifest.xml

4. A `MainActivity` é processada e o seu método `onCreate` é executado. Observe a seguir a linha 6 do arquivo `MainActivity.java`. O método `setContentView` utilizado nessa linha define que o layout da `MainActivity` é o `activity_main`.

```

1 public class MainActivity extends AppCompatActivity {
2
3     @Override
4     protected void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         setContentView(R.layout.activity_main);
7     }
8 }

```

Código Java 2.1: MainActivity.java

5. O layout da `MainActivity` é processado (inflado). Observe as linhas 12, 13, 14 e 15 do arquivo `activity_main.xml`. Um `TextView` é utilizado para inserir o texto "Hello World K19!" na tela da `MainActivity`.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

```

```
3   xmlns:tools="http://schemas.android.com/tools"
4   android:layout_width="match_parent"
5   android:layout_height="match_parent"
6   android:paddingBottom="@dimen/activity_vertical_margin"
7   android:paddingLeft="@dimen/activity_horizontal_margin"
8   android:paddingRight="@dimen/activity_horizontal_margin"
9   android:paddingTop="@dimen/activity_vertical_margin"
10  tools:context=".MainActivity">
11
12  <TextView
13      android:layout_width="wrap_content"
14      android:layout_height="wrap_content"
15      android:text="Hello World K19!" />
16</RelativeLayout>
```

Código XML 2.3: activity_main.xml

6. A tela da MainActivity obtida com o layout activity_main.xml é exibida no dispositivo.

Nesse capítulo, mostraremos como a interface de um aplicativo Android é definida.



Layouts

Geralmente, as interfaces dos aplicativos são definidas em arquivos XML armazenados na pasta `res/layout`. Todos os arquivos de layout devem ficar nesta pasta. Você não pode agrupá-los em sub-pastas. Essas interfaces também podem ser definidas com código Java. Porém, essa abordagem não é muito prática para criar interfaces extensas.

Muitas vezes, os layouts devem ser acessados no código Java. Por isso, eles são registrados automaticamente em uma classe especial chamada `R`. Essa classe é gerada automaticamente pelo Android Studio e não deve ser alterada. Além dos layouts, qualquer outro tipo de recurso é registrado nessa classe.

Considere um arquivo XML chamado `main.xml` armazenado na pasta `res/layout`. Esse arquivo define um layout. Para acessá-lo no código Java do aplicativo, podemos utilizar o atributo `R.layout.main`.

No exemplo abaixo, o atributo `R.layout.main` foi utilizado para associar o layout `main.xml` a uma activity através do método `setContentView()`.

```
1 @Override
2 public void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.main);
5 }
```

Nos layouts podemos utilizar dois tipos de elementos: Views ou ViewGroups. Um elemento do tipo View exibe algum conteúdo na tela do dispositivo e oferece algum tipo de interação para o usuário. Um elemento do tipo ViewGroup é utilizado para agrupar outros elementos, Views ou ViewGroups. Ele também determina como os elementos contidos nele serão apresentados na tela do dispositivo.



Views

Há diversos tipos de Views nas bibliotecas da plataforma Android. Veremos a seguir diversos desses tipos.

TextView

Um TextView é utilizado para exibir algum texto para o usuário. O valor do atributo text é o texto que será exibido para o usuário.

```
1 <TextView
2   android:layout_width="wrap_content"
3   android:layout_height="wrap_content"
4   android:id="@+id/texto"
5   android:text="Texto"/>
```



Importante

Os atributos layout_width e layout_height são obrigatórios em todos os tipos de Views e ViewGroups. Eles são utilizados para definir a largura e a altura dos elementos. Para esses atributos podemos utilizar um dos seguintes valores:

match_parent - Esse valor faz o elemento ter a mesma largura ou altura do elemento pai.

wrap_content - Esse valor faz o elemento ter a largura ou a altura suficiente englobar o seu conteúdo.

fill_parent - Mesmo que match_parent. Recomenda-se utilizar o valor match_parent a partir da API Level 8.

valor - Uma dimensão, ou seja, um valor numérico utilizando uma das unidades de medida suportadas na plataforma Android.



Importante

O atributo id é utilizado para identificar os elementos unicamente. Apesar de não ser obrigatório definir este atributo explicitamente, é muito comum adicioná-lo nos elementos. Esse atributo é muito importante para recuperar os elementos no código Java ou para organizar os elementos dentro de um RelativeLayout que é um tipo específico de ViewGroup.

Por serem considerados recursos, os ids são registrados na classe especial R. Por exemplo, o id do TextView acima pode ser acessado com o atributo R.id.texto no código Java.

O valor do atributo id deve:

1. Começar com o caractere arroba (@).
2. O sinal de mais (+) pode aparecer depois do arroba (@). Esse sinal indica que estamos criando um novo id. Contudo, podemos utilizá-lo mesmo quando queremos referenciar um id existente. Por isso, normalmente, os desenvolvedores utilizam esse sinal sempre.
3. Em seguida, aparecem os caracteres "id/".
4. Por fim, aparece o id propriamente.

A quantidade máxima de linhas de um `TextView` pode ser definida através do atributo `maxLines`. O valor desse atributo deve ser um número inteiro maior do que zero.

Se o texto for maior do que o espaço de um `TextView`, podemos utilizar o atributo `ellipsize` para abreviá-lo com o uso de reticências (...). Os principais valores que podem ser associados a esse atributo são: `start`, `middle` e `end`. A diferença entre esses valores é o posicionamento da reticências.

```
1 <TextView
2   android:layout_width="60dp"
3   android:layout_height="wrap_content"
4   android:ellipsize="end"
5   android:maxLines="1"
6   android:text="Texto muito longo"/>
```



Mais Sobre

Para maiores detalhes sobre o `TextView` consulte a documentação:

<http://developer.android.com/reference/android/widget/TextView.html>

Button

Um `Button` é utilizado para exibir um botão para o usuário.

É possível adicionar uma imagem nos botões com os atributos `drawableLeft`, `drawableRight`, `drawableTop` ou `drawableBottom`.

Também é possível definir uma imagem ou uma cor como background de um botão utilizando o atributo `background`. Esse atributo pode ser utilizado nos demais elementos.

```
1 <Button
2   android:id="@+id/ok"
3   android:layout_width="match_parent"
4   android:layout_height="wrap_content"
5   android:background="#ffff00"
6   android:drawableLeft="@drawable/confirma"
7   android:text="Ok" />
```

No código Java de uma activity, é possível definir a lógica que deve ser executada quando um determinado botão for pressionado.

Para isso, podemos recuperar o botão através do método `findViewById()`. O id do botão deve ser passado como parâmetro para esse método. Lembre-se que, no código Java, os ids dos elementos são recuperados através da classe especial `R`.

Após recuperar o botão, o método `setOnClickListener()` pode ser utilizado para associar esse botão a um `View.OnClickListener`. O método `onClick` desse `View.OnClickListener` será executado toda vez que o botão for pressionado.

```
1 Button botaoOk = (Button) findViewById(R.id.ok);
2
3 botaoOk.setOnClickListener(new View.OnClickListener() {
4     @Override
5     public void onClick(View v) {
6         Toast toast = Toast.makeText(getApplicationContext(),
7             "Processando...", Toast.LENGTH_SHORT);
```

```
8 toast.show();
9 }
10 });
```

Há outra forma para definir a lógica que deve ser executada quando um determinado botão é pressionado. Podemos criar um método público na classe da activity com um parâmetro do tipo View e com retorno void. Depois, associar esse método ao botão através do atributo onClick. Toda vez que o botão for pressionado esse método será executado.

```
1 public void onClick(View view) {
2     Toast toast = Toast.makeText(getApplicationContext(),
3         "Processando...", Toast.LENGTH_SHORT);
4     toast.show();
5 }
```

```
1 <Button
2     android:id="@+id/ok"
3     android:layout_width="match_parent"
4     android:layout_height="wrap_content"
5     android:text="Ok"
6     android:onClick="onClick"/>
```



Mais Sobre

Para maiores detalhes sobre o Button consulte a documentação:

<http://developer.android.com/reference/android/widget/Button.html>

RadioButton

Os RadioButtons permitem que o usuário selecione uma opção de um conjunto de opções. O RadioGroup é utilizado para definir conjuntos de opções mutuamente exclusivas.

O atributo orientation de um RadioGroup determina se as opções do conjunto correspondente aparecerão na horizontal ou na vertical.

O atributo checked pode ser utilizado para indicar se um RadioButton está selecionado ou não. Dois ou mais RadioButtons em um mesmo RadioGroup não podem estar selecionados ao mesmo tempo.

```
1 <RadioGroup
2     android:id="@+id/sexo"
3     android:layout_width="match_parent"
4     android:layout_height="wrap_content"
5     android:orientation="horizontal" >
6
7     <RadioButton
8         android:checked="true"
9         android:id="@+id/feminino"
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="Feminino" />
13
14    <RadioButton
15        android:id="@+id/masculino"
16        android:layout_width="wrap_content"
17        android:layout_height="wrap_content"
18        android:text="Masculino" />
```

```
19
20 </RadioGroup>
```

No código Java da activity, para recuperar a opção selecionada pelo usuário, podemos utilizar o método `findViewById()` para obter o `RadioGroup` que define o conjunto de opções. Em seguida, podemos utilizar o método `getCheckedRadioButtonId()` para descobrir o id do `RadioButton` selecionado nesse `RadioGroup`. Esse método devolve -1 se nenhum `RadioButton` estiver selecionado. Por fim, novamente, podemos utilizar o método `findViewById()` para recuperar esse `RadioButton`.

```
1 RadioGroup radioGroup = (RadioGroup) findViewById(R.id.sexos);
2
3 int radioButtonId = radioGroup.getCheckedRadioButtonId();
4
5 RadioButton radioButton = (RadioButton) findViewById(radioButtonId);
6
7 Toast toast = Toast.makeText(getApplicationContext(),
8     radioButton.getText(), Toast.LENGTH_SHORT);
9 toast.show();
```



Mais Sobre

Para maiores detalhes sobre o `RadioButton` consulte a documentação:

<http://developer.android.com/reference/android/widget/RadioButton.html>

CheckBox

Os `CheckBox`s permitem que o usuário selecione uma ou mais opções de um conjunto de opções.

O atributo `checked` pode ser utilizado para indicar se um `CheckBox` está selecionado ou não.

```
1 <CheckBox
2     android:checked="true"
3     android:id="@+id/java"
4     android:layout_width="match_parent"
5     android:layout_height="wrap_content"
6     android:text="Java" />
7
8 <CheckBox
9     android:id="@+id/csharp"
10    android:layout_width="match_parent"
11    android:layout_height="wrap_content"
12    android:text="C#" />
13
14 <CheckBox
15    android:checked="true"
16    android:id="@+id/php"
17    android:layout_width="match_parent"
18    android:layout_height="wrap_content"
19    android:text="PHP" />
```

No código Java, para saber o estado de um determinado `Checkbox`, podemos utilizar o método `findViewById()` para recuperá-lo e o método `isChecked()` para verificar o seu estado.

```
1 CheckBox checkBox = (CheckBox) findViewById(R.id.java);
2
3 boolean checked = checkBox.isChecked();
4
5 Toast toast = Toast.makeText(getApplicationContext(),
```

```
6 checkBox.getText() + ": " + checked, Toast.LENGTH_SHORT);
7 toast.show();
```



Mais Sobre

Para maiores detalhes sobre o CheckBox consulte a documentação:

<http://developer.android.com/reference/android/widget/CheckBox.html>

Switch

Um Switch permite que o usuário selecione uma entre duas opções (off e on).

Os atributos textOff e textOn permitem alterar os textos das opções off e on respectivamente.

```
1 <Switch
2   android:id="@+id/musica"
3   android:layout_width="match_parent"
4   android:layout_height="wrap_content"
5   android:text="Música"
6   android:textOff="Desligada"
7   android:textOn="Ligada" />
```

No código Java, para saber o estado de um determinado Switch, podemos utilizar o método findViewById() para recuperá-lo e o método isChecked() para verificar o seu estado.

```
1 Switch s = (Switch) findViewById(R.id.musica);
2
3 boolean checked = s.isChecked();
4
5 Toast toast = Toast.makeText(getApplicationContext(),
6   s.getText() + ": " + checked, Toast.LENGTH_SHORT);
7 toast.show();
```



Mais Sobre

Para maiores detalhes sobre o Switch consulte a documentação:

<http://developer.android.com/reference/android/widget/Switch.html>

ImageButton

Um ImageButton é análogo a um Button. Contudo, um ImageButton possui apenas imagem.

```
1 <ImageButton
2   android:layout_width="match_parent"
3   android:layout_height="wrap_content"
4   android:src="@drawable/confirmar" />
```



Mais Sobre

Para maiores detalhes sobre o ImageButton consulte a documentação:

<http://developer.android.com/reference/android/widget/ImageButton.html>

ImageView

Um ImageView é utilizado para exibir para o usuário uma imagem.

O atributo `src` determina o recurso correspondente à imagem que deve ser exibida por um ImageView.

O atributo `scaleType` determina como a imagem deve se adaptar à área do ImageView. Para esse atributo, podemos utilizar os seguintes valores:

center - A imagem fica centralizada na área do ImageView. O tamanho original da imagem não sofre alteração. Se a imagem for maior que a área do ImageView parte dela não aparecerá.

centerCrop - A imagem fica centralizada na área do ImageView. O seu tamanho aumenta ou diminui sem perder a proporção original de tal forma que a imagem ocupe toda a área do ImageView. Além disso, pelo menos uma das dimensões (altura ou largura) da imagem será exatamente igual à dimensão correspondente na área do ImageView. Normalmente, nesse processo, a imagem é “cortada”.

centerInside - A imagem fica centralizada na área do ImageView. Se a imagem é menor do que a área do ImageView, o tamanho da imagem não sofre alteração. Caso contrário, a imagem diminui sem perder a proporção original até que as suas dimensões sejam menores ou iguais às dimensões correspondentes na área do ImageView. Nesse processo, a imagem sempre aparecerá sem “cortes”.

fitCenter - A imagem fica centralizada na área do ImageView. O seu tamanho aumenta ou diminui sem perder a proporção original e sem exceder as dimensões do ImageView. Além disso, a imagem terá o maior tamanho possível. Nesse processo, a imagem sempre aparecerá sem “cortes”.

fitEnd - Semelhante ao `fitCenter`. Contudo, a imagem é acomodada no canto inferior direito da área do ImageView.

fitStart - Semelhante ao `fitCenter`. Contudo, a imagem é acomodada no canto superior esquerdo da área do ImageView.

fitXY - As dimensões da imagem serão iguais às dimensões do ImageView. Nesse processo, a proporção original da imagem pode não ser preservada.

matrix - Permite que o tamanho da imagem aumente ou diminua com proporções personalizadas.

```
1 <ImageView
2   android:id="@+id/logo"
3   android:layout_width="200dp"
4   android:layout_height="80dp"
5   android:scaleType="center"
6   android:src="@drawable/k19_logo" />
```



Mais Sobre

Para maiores detalhes sobre o ImageView consulte a documentação:

<http://developer.android.com/reference/android/widget/ImageView.html>

ProgressBar

Um ProgressBar é utilizado para indicar visualmente o progresso de uma operação.

Por padrão, um ProgressBar é uma roda girando (spinning wheel). O atributo `style` pode ser utilizado para alterar o estilo de um ProgressBar. A plataforma Android oferece os seguintes estilos:

Widget.ProgressBar - Uma roda média girando.

Widget.ProgressBar.Horizontal - Uma barra horizontal.

Widget.ProgressBar.Small - Uma roda pequena girando.

Widget.ProgressBar.Large - Uma roda grande girando.

Widget.ProgressBar.Inverse - Uma roda média girando com esquema invertido de cores.

Widget.ProgressBar.Small.Inverse - Uma roda pequena girando com esquema invertido de cores.

Widget.ProgressBar.Large.Inverse - Uma roda grande girando com esquema invertido de cores.

Os estilos “invertidos” são úteis para temas com cores claras.

Quando o estilo `Widget.ProgressBar.Horizontal` é utilizado, faz sentido também utilizar os atributos `max` e `progress`. O atributo `max` determina o valor que será atingido quando a operação do ProgressBar estiver concluída. Por padrão, o valor desse atributo é 100. O atributo `progress` indica quanto já foi realizado da operação do ProgressBar.

```
1 <ProgressBar
2   android:id="@+id/progresso"
3   style="@android:style/Widget.ProgressBar.Horizontal"
4   android:layout_width="wrap_content"
5   android:layout_height="wrap_content"
6   android:max="200"
7   android:progress="75" />
```

No código Java, podemos alterar o progresso de um ProgressBar. Para isso, podemos utilizar o método `incrementProgressBy()`.

```
1 ProgressBar progressBar = (ProgressBar) findViewById(R.id.progresso);
2 progressBar.incrementProgressBy(5);
```

Também podemos utilizar o método `setProgress()` para alterar o progresso de um ProgressBar.

```
1 ProgressBar progressBar = (ProgressBar) findViewById(R.id.progresso);
2 progressBar.setProgress(55);
```



Mais Sobre

Para maiores detalhes sobre o ProgressBar consulte a documentação:

<http://developer.android.com/reference/android/widget/ProgressBar.html>

SeekBar

Um SeekBar é um tipo específico de ProgressBar que permite que o usuário modifique o nível de progresso arrastando o indicador para esquerda ou para direita.

```
1 <SeekBar
2   android:id="@+id/progresso"
3   android:layout_width="match_parent"
4   android:layout_height="wrap_content"
5   android:max="200"
6   android:progress="75" />
```



Mais Sobre

Para maiores detalhes sobre o SeekBar consulte a documentação:

<http://developer.android.com/reference/android/widget/SeekBar.html>

Spinner

Um Spinner permite que o usuário selecione uma opção de um conjunto de opções.

```
1 <Spinner
2   android:id="@+id/cidades"
3   android:layout_width="match_parent"
4   android:layout_height="wrap_content" />
```

As opções podem ser inseridas em um Spinner de forma estática. Para isso, podemos definir um array de strings no arquivo strings.xml utilizando o elemento string-array. Observe o código abaixo. Um nome foi definido para o array e as opções foram definidas com o elemento item.

```
1 <resources>
2   ...
3   <string-array name="cidades">
4     <item>São Paulo</item>
5     <item>Mogi das Cruzes</item>
6     <item>Campinas</item>
7     <item>Pindamonhangaba</item>
8   </string-array>
9 </resources>
```

As opções definidas em um array de strings no arquivo strings.xml podem ser associadas a um Spinner através do atributo entries. O array é acessado com o prefixo "@array/" seguido do seu nome.

```
1 <Spinner
2   android:id="@+id/cidades"
3   android:layout_width="match_parent"
4   android:layout_height="wrap_content"
5   android:entries="@array/cidades" />
```

Outra possibilidade é definir as opções de um Spinner dinamicamente. Por exemplo, com dados obtidos de um web service. Para isso, é necessário criar um Adapter para fornecer as opções para o Spinner.

No código Java abaixo, um Spinner foi recuperado através do método findViewById(). Uma

lista de strings foi criada e populada com nomes de algumas cidades. Um `ArrayAdapter` de string foi criado a partir dessa lista. O segundo parâmetro do construtor utilizado da classe `ArrayAdapter` é o layout que será aplicado a cada opção do `Spinner`. Nesse exemplo, utilizamos um layout fornecido pela plataforma Android, o `R.layout.simple_spinner_dropdown_item`. Por fim, o `Adapter` foi associado ao `Spinner`.

```
1 Spinner spinner = (Spinner) findViewById(R.id.cidades);
2
3 ArrayList<String> cidades = new ArrayList<>();
4 cidades.add("São Paulo");
5 cidades.add("Mogi das Cruzes");
6 cidades.add("Campinas");
7 cidades.add("Pindamonhangaba");
8
9 ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
10     android.R.layout.simple_spinner_dropdown_item, cidades);
11
12 spinner.setAdapter(adapter);
```

A opção selecionada em um `Spinner` pode ser recuperada através do método `getSelectedItem()`.

```
1 Spinner cidades = (Spinner) findViewById (R.id.cidades);
2 String cidade = cidades.getSelectedItem().toString();
```



Mais Sobre

Para maiores detalhes sobre o `Spinner` consulte a documentação:

<http://developer.android.com/reference/android/widget/Spinner.html>

EditText

Um `EditText` é um tipo específico de `TextView` que permite que o usuário modifique o seu texto.

O atributo `hint` é utilizado para definir uma dica para o usuário saber o que deve ser preenchido em um `EditText`.

O atributo `inputType` determina o tipo de texto de um `EditText`. O teclado oferecido ao usuário para modificar o texto de um `EditText` depende do valor desse atributo. Alguns valores que podem ser utilizados nesse atributo são:

text - texto normal

textEmailAddress - email

textPassword - senha

textAutoCorrect - texto normal com correção ortográfica

textCapSentences - texto normal com a primeira letra de cada sentença maiúscula

Dois ou mais valores podem ser utilizados ao mesmo tempo no atributo `inputType`. Para isso, basta concatená-los com o caractere barra vertical (`|`).

```

1 <EditText
2   android:id="@+id/mensagem"
3   android:layout_width="match_parent"
4   android:layout_height="wrap_content"
5   android:hint="Escreva uma mensagem"
6   android:inputType="textAutoCorrect|textCapSentences" />

```

Para recuperar o texto de um `EditText`, podemos utilizar o método `getEditableText()`. Esse método devolve a referência de um `Editable`. O texto pode ser extraído através do método `toString()`.

```

1 EditText nome = (EditText) findViewById(R.id.mensagem);
2 String texto = nome.getEditableText().toString();

```

Uma ação pode ser definida no teclado correspondente a um `EditText` através do atributo `imeOptions`. Os valores `actionNext`, `actionDone`, `actionSend` e `actionSearch` são exemplos de valores aceitos por esse atributo.

```

1 <EditText
2   android:id="@+id/mensagem"
3   android:layout_width="match_parent"
4   android:layout_height="wrap_content"
5   android:hint="Escreva uma mensagem"
6   android:imeOptions="actionSend"
7   android:inputType="textAutoCorrect|textCapSentences" />

```

No código Java da activity, podemos definir a lógica que deve ser executada quando a ação definida no teclado de um `EditText` é disparada. Note no código abaixo que um objeto da interface `TextView.OnEditorActionListener` foi associado a um `EditText`. O método `onEditorAction()` é chamado toda vez que a ação do `EditText` é disparada.

```

1 EditText editText = (EditText) findViewById(R.id.mensagem);
2
3 editText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
4   @Override
5   public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
6     boolean handled = false;
7     if (actionId == EditorInfo.IME_ACTION_SEND) {
8       Toast toast = Toast.makeText(getApplicationContext(),
9         "Enviando...", Toast.LENGTH_SHORT);
10      toast.show();
11    }
12    return handled;
13  }
14 });

```



Mais Sobre

Para maiores detalhes sobre o `EditText` consulte a documentação:

<http://developer.android.com/reference/android/widget/EditText.html>

DatePicker

Um `DatePicker` permite que o usuário selecione uma data ou horário.

```

1 <DatePicker
2   android:id="@+id/nascimento"

```

```
3 android:layout_width="match_parent"
4 android:layout_height="wrap_content"
5 android:calendarViewShown="false" />
```



Mais Sobre

Para maiores detalhes sobre o DatePicker consulte a documentação:

<http://developer.android.com/reference/android/widget/DatePicker.html>



Exercícios de Fixação

- 1 No Android Studio, crie um projeto chamado **Interfaces** seguindo os passos vistos no 2.
- 2 Copie o arquivo **k19-logo.png** da pasta **K19-Arquivos/imagens** para a pasta **drawable** do projeto **Interfaces**. Como os nomes dos recursos de um aplicativo Android não podem possuir o caractere hífen (-), renomeie o arquivo **k19-logo.png** substituindo o caractere hífen por underscore (_).



Importante

Você também pode obter esse arquivo através do site da K19: www.k19.com.br/arquivos.

- 3 Altere o arquivo **activity_main.xml** de acordo com o código abaixo.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ScrollView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">
8
9     <LinearLayout
10         android:layout_width="match_parent"
11         android:layout_height="wrap_content"
12         android:orientation="vertical"
13         android:paddingBottom="@dimen/activity_vertical_margin"
14         android:paddingLeft="@dimen/activity_horizontal_margin"
15         android:paddingRight="@dimen/activity_horizontal_margin"
16         android:paddingTop="@dimen/activity_vertical_margin">
17
18         <ImageView
19             android:layout_width="match_parent"
20             android:layout_height="100dp"
21             android:scaleType="centerInside"
22             android:src="@drawable/k19_logo" />
23
24         <TextView
25             android:layout_width="wrap_content"
26             android:layout_height="wrap_content"
27             android:text="Nome: " />
```

```

28
29 <EditText
30     android:id="@+id/nome"
31     android:layout_width="match_parent"
32     android:layout_height="wrap_content"
33     android:layout_marginBottom="16dp"
34     android:hint="Ex: Rafael Cosentino"
35     android:imeOptions="actionNext"
36     android:inputType="textPersonName" />
37
38 <TextView
39     android:layout_width="wrap_content"
40     android:layout_height="wrap_content"
41     android:text="Email: " />
42
43 <EditText
44     android:id="@+id/email"
45     android:layout_width="match_parent"
46     android:layout_height="wrap_content"
47     android:layout_marginBottom="16dp"
48     android:hint="Ex: contato@k19.com.br"
49     android:imeOptions="actionNext"
50     android:inputType="textEmailAddress" />
51
52 <TextView
53     android:layout_width="wrap_content"
54     android:layout_height="wrap_content"
55     android:text="Cidade: " />
56
57 <Spinner
58     android:id="@+id/cidades"
59     android:layout_width="match_parent"
60     android:layout_height="wrap_content"
61     android:layout_marginBottom="16dp" />
62
63 <CheckBox
64     android:id="@+id/noticias"
65     android:layout_width="wrap_content"
66     android:layout_height="wrap_content"
67     android:layout_marginBottom="16dp"
68     android:checked="true"
69     android:text="Deseja receber notícias por email?" />
70
71 <TextView
72     android:layout_width="match_parent"
73     android:layout_height="wrap_content"
74     android:text="Sexo: " />
75
76 <RadioGroup
77     android:id="@+id/sexo"
78     android:layout_width="match_parent"
79     android:layout_height="wrap_content"
80     android:layout_marginBottom="16dp"
81     android:orientation="horizontal">
82
83     <RadioButton
84         android:id="@+id/masculino"
85         android:layout_width="wrap_content"
86         android:layout_height="wrap_content"
87         android:checked="true"
88         android:text="Masculino" />
89
90     <RadioButton
91         android:id="@+id/feminino"
92         android:layout_width="wrap_content"
93         android:layout_height="wrap_content"
94         android:text="Feminino" />
95
96 </RadioGroup>
97

```

```
98     <Button
99         android:id="@+id/cadastrar"
100         android:layout_width="match_parent"
101         android:layout_height="wrap_content"
102         android:background="#064e83"
103         android:text="Cadastrar"
104         android:textColor="#ffffff" />
105     </LinearLayout>
106
107 </ScrollView>
```

Código XML 3.18: activity_main.xml

- 4 Altere o arquivo **MainActivity.java** de acordo com o código abaixo.

```
1 package br.com.k19.interfaces;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.widget.AdapterView;
7 import android.widget.AdapterView.OnItemClickListener;
8
9 import java.util.ArrayList;
10
11 public class MainActivity extends AppCompatActivity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17
18         initSpinner();
19     }
20
21     private void initSpinner() {
22         Spinner spinner = (Spinner) findViewById(R.id.cidades);
23
24         if (spinner == null) {
25             return;
26         }
27
28         ArrayList<String> cidades = new ArrayList<>();
29         cidades.add("São Paulo");
30         cidades.add("Mogi das Cruzes");
31         cidades.add("Campinas");
32         cidades.add("Pindamonhangaba");
33
34         ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
35             android.R.layout.simple_dropdown_item_1line, cidades);
36
37         spinner.setAdapter(adapter);
38     }
39 }
```

Código Java 3.13: MainActivity.java

- 5 Execute o aplicativo. Para isso, selecione a opção **Run->Run 'app'**. Depois, marque o AVD criado anteriormente e clique no botão OK. Aguarde a inicialização completa do AVD e do aplicativo.

- 6 Acrescente o método **initButton()** na classe **MainActivity**.


```

1 private void initButton() {
2     final EditText nome = (EditText) findViewById(R.id.nome);
3     final EditText email = (EditText) findViewById(R.id.email);
4     final Spinner cidades = (Spinner) findViewById(R.id.cidades);
5     final CheckBox noticias = (CheckBox) findViewById(R.id.noticias);
6     final RadioGroup sexo = (RadioGroup) findViewById(R.id.sexo);
7
8     Button button = (Button) findViewById(R.id.cadastrar);
9
10    if(button == null) {
11        return;
12    }
13
14    button.setOnClickListener(new View.OnClickListener() {
15        @Override
16        public void onClick(View v) {
17            StringBuilder builder = new StringBuilder();
18            builder.append("Nome: " + nome.getText().toString() + "\n");
19            builder.append("Email: " + email.getText().toString() + "\n");
20            builder.append("Cidade: " + cidades.getSelectedItem() + "\n");
21            builder.append("Receber notícia? " + noticias.isChecked() + "\n");
22
23            RadioButton sexoSelecionado =
24                (RadioButton) findViewById(sexo.getCheckedRadioButtonId());
25
26            builder.append("Sexo: " + sexoSelecionado.getText());
27
28            Toast toast = Toast.makeText(MainActivity.this,
29                builder.toString(), Toast.LENGTH_LONG);
30            toast.show();
31        }
32    });
33 }

```

Código Java 3.14: MainActivity.java

7 Altere o método onCreate() na classe MainActivity.

```

1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.activity_main);
5
6     initSpinner();
7     initButton();
8 }

```

Código Java 3.15: MainActivity.java



ViewGroups

Há diversos tipos de ViewGroups nas bibliotecas da plataforma Android. Veremos a seguir alguns desses tipos.

LinearLayout

Os elementos contidos em um LinearLayout são organizados em uma única coluna um embaixo do outro ou em uma única linha um do lado do outro. O atributo orientation determina se a disposição dos elementos será horizontal ou vertical.

```
1 <LinearLayout
2   android:layout_width="match_parent"
3   android:layout_height="match_parent"
4   android:orientation="vertical">
5
6   . . .
7
8 </LinearLayout>
```

O atributo `gravity` permite ajustar a posição dos elementos contidos em um `LinearLayout`. Veja alguns valores que podem ser utilizados com esse atributo:

center - os elementos são centralizados na horizontal e na vertical.

center_horizontal - os elementos são centralizados na horizontal.

center_vertical - os elementos são centralizados na vertical.

left - empurra os elementos para a esquerda.

right - empurra os elementos para a direita.

top - empurra os elementos para cima.

bottom - empurra os elementos para baixo.



Mais Sobre

Para maiores detalhes sobre o `LinearLayout` consulte a documentação:

<http://developer.android.com/reference/android/widget/LinearLayout.html>

RelativeLayout

Os elementos contidos em um `RelativeLayout` podem ser posicionados um em relação ao outro ou em relação ao próprio `RelativeLayout`.

No exemplo abaixo, os atributos `layout_alignParentRight` e `layout_centerVertical` foram utilizados para posicionar o elemento `R.id.verde` no centro e a direita dentro do `RelativeLayout`.

O elemento `R.id.vermelho` foi posicionado abaixo do elemento `R.id.verde` com a utilização do atributo `layout_below`. Além disso, com a utilização do atributo `layout_alignLeft`, o lado esquerdo do elemento `R.id.vermelho` foi alinhado ao lado esquerdo do elemento `R.id.verde`.

O elemento `R.id.azul` foi posicionado acima do elemento `R.id.vermelho` com a utilização do atributo `layout_above`. Além disso, com a utilização do atributo `layout_toLeftOf`, o elemento `R.id.azul` foi posicionado a esquerda do elemento `R.id.verde`.

```
1 <RelativeLayout
2   android:layout_width="match_parent"
3   android:layout_height="match_parent">
4
5   <TextView
6     android:id="@+id/verde"
7     android:layout_width="100dp"
```

```

8     android:layout_height="100dp"
9     android:layout_alignParentRight="true"
10    android:layout_centerVertical="true"
11    android:background="#00ff00" />
12
13    <TextView
14        android:id="@+id/vermelho"
15        android:layout_width="100dp"
16        android:layout_height="100dp"
17        android:layout_alignLeft="@id/verde"
18        android:layout_below="@id/verde"
19        android:background="#ff0000" />
20
21    <TextView
22        android:id="@+id/azul"
23        android:layout_width="100dp"
24        android:layout_height="100dp"
25        android:layout_above="@id/verde"
26        android:layout_toLeftOf="@id/verde"
27        android:background="#0000ff" />
28
29 </RelativeLayout>

```



Mais Sobre

Para maiores detalhes sobre o RelativeLayout consulte a documentação:

<http://developer.android.com/reference/android/widget/RelativeLayout.html>

ListView

Um ListView é utilizado para organizar elementos em uma lista com rolagem vertical. Por exemplo, considere uma lista de filmes. Utilizaremos um ListView para exibir essa lista. Esses filmes serão modelados pela classe a seguir

```

1 public class Movie {
2     private String name;
3     private int rating;
4     private int year;
5
6     // GETTERS E SETTERS
7 }

```

Código Java 3.16: Movie.java

Devemos criar um layout para definir como cada filme deve ser exibido em uma lista. O layout abaixo utiliza um RelativeLayout para definir como exibir para o usuário o nome, o ano e a avaliação de um único filme.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:gravity="center_vertical"
7     android:padding="16dp">
8
9     <TextView
10         android:id="@+id/name"
11         android:layout_width="wrap_content"

```

```

12     android:layout_height="wrap_content"
13     android:layout_alignParentLeft="true"
14     android:layout_toLeftOf="@+id/rating"
15     android:ellipsize="end"
16     android:lines="1"
17     android:textColor="#333333"
18     android:textSize="18sp" />
19
20     <TextView
21         android:id="@+id/year"
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         android:layout_alignParentLeft="true"
25         android:layout_below="@+id/name"
26         android:lines="1"
27         android:textColor="#666666"
28         android:textSize="14sp" />
29
30     <TextView
31         android:id="@+id/rating"
32         android:layout_width="50dp"
33         android:layout_height="50dp"
34         android:layout_alignParentRight="true"
35         android:layout_marginLeft="16dp"
36         android:background="#666666"
37         android:gravity="center"
38         android:lines="1"
39         android:textColor="ffffff"
40         android:textSize="30sp" />
41
42 </RelativeLayout>

```

Código XML 3.21: list_view_item_movie.xml

Um `ListAdapter` deve ser implementado para montar os itens do `ListView`. A classe abaixo implementará o nosso `ListAdapter`. Observe que ela herda da classe `ArrayAdapter`. Essa herança facilita o procedimento de criação de um `ListAdapter`.

```

1 public class MovieAdapter extends ArrayAdapter<Movie> {
2     ...
3 }

```

Código Java 3.17: MovieAdapter.java

Vamos adicionar um construtor na classe `MovieAdapter` com três parâmetros. O primeiro é o contexto que será associado ao `ListAdapter`. O segundo é o id do layout que deve ser utilizado para cada filme da lista. O terceiro é uma lista de filmes. Esses parâmetros devem ser repassados para a classe mãe através do construtor com os mesmos parâmetros. O id do layout deve ser armazenado em um atributo.

```

1 public class MovieAdapter extends ArrayAdapter<Movie> {
2
3     private int layout;
4
5     public MovieAdapter(Context context, int layout, List<Movie> movies) {
6         super(context, layout, movies);
7         this.layout = layout;
8     }
9
10    ...
11 }

```

Código Java 3.18: MovieAdapter.java

O método `getView()` deve ser reescrito na classe `MovieAdapter`. Esse método deve montar uma `View` para exibir um determinado filme. O primeiro parâmetro desse método é a posição do filme na lista de filmes recebida no construtor. O segundo parâmetro é uma referência da `View` que deve ser reutilizada se possível. O terceiro parâmetro é o `ViewGroup` no qual os filmes devem ser apresentados.

Para processar o layout que define como cada filme deve ser apresentado, podemos utilizar um `LayoutInflater`. Uma referência de um objeto desse tipo pode ser obtida através do `getSystemService()`. O método `inflate()` processa um layout e gera uma árvore com objetos correspondentes aos elementos da árvore de elementos definidos nesse layout. Uma referência do objeto raiz da árvore gerada é devolvida pelo método `inflate()`. Com essa referência podemos recuperar os demais objetos da árvore gerada. Os dados do filme devem ser inseridos nesses objetos. O método `getItem()` pode ser utilizado para recuperar uma referência do filme da posição desejada.

```

1 public class MovieAdapter extends ArrayAdapter<Movie> {
2
3     ...
4
5     @Override
6     public View getView(int position, View view, ViewGroup parent) {
7         if (view == null) {
8             LayoutInflater inflater = (LayoutInflater) getContext().
9                 getSystemService(Context.LAYOUT_INFLATER_SERVICE);
10            view = inflater.inflate(this.layout, parent, false);
11        }
12
13        TextView name = (TextView) view.findViewById(R.id.name);
14        TextView year = (TextView) view.findViewById(R.id.year);
15        TextView rating = (TextView) view.findViewById(R.id.rating);
16
17        Movie movie = getItem(position);
18
19        name.setText(movie.getName());
20        year.setText("Year: " + movie.getYear());
21        rating.setText(String.valueOf(movie.getRating()));
22
23        return view;
24    }
25 }

```

Código Java 3.19: MovieAdapter.java

O `ListView` deve ser adicionado em um layout.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ListView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/movies"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent" />

```

Código XML 3.22: activity_main.xml

No código Java, podemos recuperar o `ListView` e associá-lo a um `MovieAdapter`. A lista de filmes que deve ser passada como parâmetro do construtor da classe `MovieAdapter` pode ser obtida de um web service por exemplo.

```

1 public class MainActivity extends AppCompatActivity {
2
3     @Override
4     protected void onCreate(Bundle savedInstanceState) {

```

```
5     super.onCreate(savedInstanceState);
6     setContentView(R.layout.activity_main);
7
8     ArrayList<Movie> movies = ...
9
10    MovieAdapter adapter = new MovieAdapter(this,
11        R.layout.list_view_item_movie, movies);
12
13    ListView listView = (ListView) findViewById(R.id.movies);
14
15    listView.setAdapter(adapter);
16 }
17 }
```

Código Java 3.20: MainActivity.java

GridView

Um GridView é semelhante a um ListView. Basicamente, a diferença entre eles é que um GridView pode ter várias colunas.

O atributo `numColumns` determina a quantidade de colunas.

```
1 <GridView
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:id="@+id/movies"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:numColumns="2"/>
```



Exercícios de Fixação

8 No projeto **Interfaces**, crie um arquivo XML chamado **activity_linear_layout.xml** na pasta **layout**.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical">
7
8     <TextView
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:background="#ff0000"
12        android:text="Texto 1"
13        android:textColor="#ffffff" />
14
15    <TextView
16        android:layout_width="wrap_content"
17        android:layout_height="wrap_content"
18        android:background="#00ff00"
19        android:text="Texto 2"
20        android:textColor="#ffffff" />
21
22    <TextView
23        android:layout_width="wrap_content"
24        android:layout_height="wrap_content"
25        android:background="#0000ff">
```

```

26     android:text="Texto 3"
27     android:textColor="#ffffff" />
28
29 </LinearLayout>

```

Código XML 3.24: activity_linear_layout.xml

9 Altere o método **onCreate()** na classe **MainActivity**.

```

1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.activity_linear_layout);
5     ...
6 }

```

Código Java 3.21: MainActivity.java

10 Execute o aplicativo e observe o resultado.

11 Crie um arquivo XML chamado **activity_relative_layout.xml** na pasta **layout**.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent">
6
7     <TextView
8         android:id="@+id/texto1"
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:layout_alignParentLeft="true"
12        android:layout_alignParentTop="true"
13        android:layout_marginLeft="40dp"
14        android:layout_marginTop="20dp"
15        android:background="#ff0000"
16        android:text="Texto 1"
17        android:textColor="#ffffff" />
18
19     <TextView
20         android:id="@+id/texto2"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content"
23         android:layout_below="@id/texto1"
24         android:layout_toRightOf="@id/texto1"
25         android:background="#00ff00"
26         android:text="Texto 2"
27         android:textColor="#ffffff" />
28
29     <TextView
30         android:id="@+id/texto3"
31         android:layout_width="wrap_content"
32         android:layout_height="wrap_content"
33         android:layout_alignParentBottom="true"
34         android:layout_toLeftOf="@id/texto2"
35         android:background="#0000ff"
36         android:text="Texto 3"
37         android:textColor="#ffffff" />
38
39 </RelativeLayout>

```

Código XML 3.25: activity_relative_layout.xml

- 12 Altere o método **onCreate()** na classe **MainActivity**.

```
1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.activity_relative_layout);
5     ...
6 }
```

Código Java 3.22: MainActivity.java

- 13 Execute o aplicativo e observe o resultado.

- 14 Crie um arquivo XML chamado **activity_frame_layout.xml** na pasta **layout**.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <FrameLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent">
6
7     <TextView
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"
10        android:background="#ff0000"
11        android:text="Texto 1"
12        android:textColor="#ffffff" />
13
14    <TextView
15        android:layout_width="wrap_content"
16        android:layout_height="wrap_content"
17        android:background="#00ff00"
18        android:text="Texto 2"
19        android:layout_marginTop="10dp"
20        android:layout_marginLeft="20dp"
21        android:textColor="#ffffff" />
22
23    <TextView
24        android:layout_width="wrap_content"
25        android:layout_height="wrap_content"
26        android:background="#0000ff"
27        android:text="Texto 3"
28        android:layout_marginTop="20dp"
29        android:layout_marginLeft="40dp"
30        android:textColor="#ffffff" />
31
32 </FrameLayout>
```

Código XML 3.26: activity_frame_layout.xml

- 15 Altere o método **onCreate()** na classe **MainActivity**.

```
1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
```



```

4 setContentView(R.layout.activity_frame_layout);
5 ...
6 }

```

Código Java 3.23: MainActivity.java

16 Execute o aplicativo e observe o resultado.

17 Crie um arquivo XML chamado **activity_list_view.xml** na pasta **layout**.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ListView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/list_view"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent" />

```

Código XML 3.27: activity_list_view.xml

18 Na classe **MainActivity**, acrescente os métodos **createItemsList()** e **initListView()**.

```

1 private ArrayList<String> createItemsList() {
2     ArrayList<String> list = new ArrayList<String>();
3
4     for (int i = 1; i <= 60; i++) {
5         list.add("Item " + i);
6     }
7     return list;
8 }
9
10 private void initListView() {
11     ListView listView = (ListView) findViewById(R.id.list_view);
12
13     if (listView == null) {
14         return;
15     }
16
17     ArrayList<String> list = createItemsList();
18
19     ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
20         android.R.layout.simple_list_item_1, list);
21
22     listView.setAdapter(adapter);
23 }

```

Código Java 3.24: MainActivity.java

19 Ainda na classe **MainActivity**, altere o método **onCreate()**.

```

1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.activity_list_view);
5
6     initSpinner();
7     initButton();
8     initListView();
9 }

```

Código Java 3.25: MainActivity.java

- 20 Execute o aplicativo e observe o resultado.
- 21 Crie um arquivo XML chamado **activity_advanced_list_view.xml** na pasta **layout**.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ListView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/advanced_list_view"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent" />
```

Código XML 3.28: activity_advanced_list_view.xml

- 22 Crie também um arquivo XML chamado **list_view_item.xml** na pasta **layout**.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:gravity="center_vertical"
7     android:padding="16dp">
8
9     <TextView
10         android:id="@+id/name"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:layout_alignParentLeft="true"
14         android:layout_toLeftOf="@+id/rating"
15         android:ellipsize="end"
16         android:lines="1"
17         android:textColor="#333333"
18         android:textSize="18sp" />
19
20     <TextView
21         android:id="@+id/year"
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         android:layout_alignParentLeft="true"
25         android:layout_below="@+id/name"
26         android:lines="1"
27         android:textColor="#666666"
28         android:textSize="14sp" />
29
30     <TextView
31         android:id="@+id/rating"
32         android:layout_width="50dp"
33         android:layout_height="50dp"
34         android:layout_alignParentRight="true"
35         android:layout_marginLeft="16dp"
36         android:background="#666666"
37         android:gravity="center"
38         android:lines="1"
39         android:textColor="ffffff"
40         android:textSize="30sp" />
41
42 </RelativeLayout>
```

Código XML 3.29: list_view_item.xml

23 No pacote **br.com.k19.interfaces**, crie uma classe chamada **Movie**.

```
1 package br.com.k19.interfaces;
2
3 import java.util.Random;
4
5 public class Movie {
6     private String name;
7     private int rating;
8     private int year;
9
10    private static String[] names = new String[]{
11        "Back to the Future", "Star Wars", "Lord of the Rings",
12        "The Hitchhiker's Guide to the Galaxy", "Tron", "Blade Runner",
13        "Harry Potter and the Sorcerer's Stone", "Raiders of the Lost Ark"};
14
15    public static Movie newInstance() {
16        Random random = new Random();
17
18        Movie movie = new Movie();
19        movie.setName(names[random.nextInt(names.length)]);
20        movie.setRating(random.nextInt(5) + 1);
21        movie.setYear(1980 + random.nextInt(35));
22
23        return movie;
24    }
25
26    public String getName() {
27        return name;
28    }
29
30    public void setName(String name) {
31        this.name = name;
32    }
33
34    public int getRating() {
35        return rating;
36    }
37
38    public void setRating(int rating) {
39        this.rating = rating;
40    }
41
42    public int getYear() {
43        return year;
44    }
45
46    public void setYear(int year) {
47        this.year = year;
48    }
49 }
```

Código Java 3.26: Movie.java

24 Agora, crie uma classe chamada **MovieAdapter** no mesmo pacote.

```
1 package br.com.k19.interfaces;
2
3 import android.content.Context;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.ArrayAdapter;
8 import android.widget.ImageView;
9 import android.widget.TextView;
10
11 import java.util.List;
```

```
12
13 public class MovieAdapter extends ArrayAdapter<Movie> {
14
15     private int resource;
16
17     public MovieAdapter(Context context, int resource, List<Movie> objects) {
18         super(context, resource, objects);
19         this.resource = resource;
20     }
21
22     @Override
23     public View getView(int position, View convertView, ViewGroup parent) {
24         View row = convertView;
25
26         if (row == null) {
27             LayoutInflater inflater = (LayoutInflater) getContext()
28                 .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
29
30             row = inflater.inflate(this.resource, parent, false);
31         }
32
33         Movie movie = getItem(position);
34
35         TextView name = (TextView) row.findViewById(R.id.name);
36         name.setText(movie.getName());
37
38         TextView year = (TextView) row.findViewById(R.id.year);
39         year.setText("Year: " + movie.getYear());
40
41         TextView rating = (TextView) row.findViewById(R.id.rating);
42         rating.setText(String.valueOf(movie.getRating()));
43
44         return row;
45     }
46 }
```

Código Java 3.27: MovieAdapter.java

- 25 Na classe **MainActivity**, acrescente os métodos **createMoviesList()** e **initAdvancedListView()**.

```
1 private ArrayList<Movie> createMoviesList() {
2     ArrayList<Movie> list = new ArrayList<Movie>();
3
4     for (int i = 1; i <= 60; i++) {
5         list.add(Movie.newInstance());
6     }
7
8     return list;
9 }
10
11 private void initAdvancedListView() {
12     ListView listView = (ListView) findViewById(R.id.advanced_list_view);
13
14     if (listView == null) {
15         return;
16     }
17
18     ArrayList<Movie> list = createMoviesList();
19
20     MovieAdapter adapter = new MovieAdapter(this, R.layout.list_view_item, list);
21     listView.setAdapter(adapter);
22 }
```

Código Java 3.28: MainActivity.java

- 26 Ainda na classe **MainActivity**, altere o método **onCreate()**.

```

1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.activity_advanced_list_view);
5
6     initSpinner();
7     initButton();
8     initListView();
9     initAdvancedListView();
10 }

```

Código Java 3.29: MainActivity.java

- 27 Execute o aplicativo e observe o resultado.

- 28 Crie um arquivo XML chamado **activity_grid_view.xml** na pasta **layout**.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <GridView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/grid_view"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:horizontalSpacing="16dp"
8     android:numColumns="auto_fit"
9     android:padding="16dp"
10    android:verticalSpacing="16dp" />

```

Código XML 3.30: activity_grid_view.xml

- 29 Na classe **MainActivity**, acrescente o método **initGridView()**.

```

1 private void initGridView() {
2     GridView gridView = (GridView) findViewById(R.id.grid_view);
3
4     if (gridView == null) {
5         return;
6     }
7
8     ArrayList<String> list = createItemsList();
9
10    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
11        android.R.layout.simple_list_item_1, list);
12    gridView.setAdapter(adapter);
13 }

```

Código Java 3.30: MainActivity.java

- 30 Ainda na classe **MainActivity**, altere o método **onCreate()**.

```

1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.activity_grid_view);
5
6     initSpinner();
7     initButton();

```

```
8     initListView();
9     initAdvancedListView();
10    initGridView();
11 }
```

Código Java 3.31: MainActivity.java

31 Execute o aplicativo e observe o resultado.

32 Crie um arquivo XML chamado **activity_advanced_grid_view.xml** na pasta **layout**.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <GridView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/advanced_grid_view"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:numColumns="auto_fit" />
```

Código XML 3.31: activity_advanced_grid_view.xml

33 Crie também um arquivo XML chamado **grid_view_item.xml** na pasta **layout**.

```
1 <LinearLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:padding="8dp"
6     android:orientation="vertical">
7
8     <LinearLayout
9         android:layout_width="match_parent"
10        android:layout_height="wrap_content"
11        android:background="#ffffff"
12        android:orientation="vertical"
13        android:padding="8dp">
14
15        <TextView
16            android:id="@+id/name"
17            android:layout_width="match_parent"
18            android:layout_height="wrap_content"
19            android:ellipsize="end"
20            android:lines="2"
21            android:textColor="#333333"
22            android:textSize="14sp" />
23
24        <TextView
25            android:id="@+id/year"
26            android:layout_width="match_parent"
27            android:layout_height="wrap_content"
28            android:ellipsize="end"
29            android:lines="1"
30            android:textColor="#333333"
31            android:textSize="10sp" />
32    </LinearLayout>
33
34    <TextView
35        android:id="@+id/rating"
36        android:layout_width="match_parent"
37        android:layout_height="wrap_content"
38        android:background="#666666"
39        android:ellipsize="end"
```

```

40     android:gravity="center"
41     android:lines="1"
42     android:padding="8dp"
43     android:textColor="#ffffff"
44     android:textSize="10sp" />
45
46 </LinearLayout>

```

Código XML 3.32: grid_view_item.xml

- 34 Na classe **MainActivity**, acrescente o método **initAdvancedGridView()**.

```

1 private void initAdvancedGridView() {
2     GridView gridView = (GridView) findViewById(R.id.advanced_grid_view);
3
4     if (gridView == null) {
5         return;
6     }
7
8     ArrayList<Movie> list = createMoviesList();
9
10    MovieAdapter adapter = new MovieAdapter(this, R.layout.grid_view_item, list);
11    gridView.setAdapter(adapter);
12    gridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
13        @Override
14        public void onItemClick(AdapterView<?> parent, View view,
15            int position, long id) {
16
17            Movie movie = (Movie) parent.getAdapter().getItem(position);
18
19            Toast toast = Toast.makeText(MainActivity.this, movie.getName(),
20                Toast.LENGTH_LONG);
21
22            toast.show();
23        }
24    });
25 }

```

Código Java 3.32: MainActivity.java

- 35 Ainda na classe **MainActivity**, altere o método **onCreate()**.

```

1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.activity_advanced_grid_view);
5
6     initSpinner();
7     initButton();
8     initListView();
9     initAdvancedListView();
10    initGridView();
11    initAdvancedGridView();
12 }

```

Código Java 3.33: MainActivity.java

- 36 Execute o aplicativo e observe o resultado.



Strings

Textos aparecem frequentemente nas telas de um aplicativo. Aparecem na barra de título, nos botões e no menu de opções, por exemplo.

Suponha um aplicativo de e-commerce. Nesse aplicativo, imagine quantas vezes um botão com o texto “Comprar” pode aparecer. Ele pode aparecer na tela principal, na tela dos detalhes de um produto ou na tela de listagem de produtos. Enfim, esse botão pode aparecer em diversos pontos.

Em qualquer uma dessas telas, poderíamos escrever o seguinte código para apresentarmos tal botão:

```
1 ...
2 <Button
3     android:layout_width="wrap_content"
4     android:layout_height="wrap_content"
5     android:text="Comprar" />
6 ...
```

Código XML 3.33: Exemplo para o botão “Comprar”

Observando o exemplo acima, fica evidente que se algum dia mudarmos de idéia e resolvermos trocar o texto Comprar por Compre agora, deveremos procurar em todo o projeto do aplicativo botões com o atributo `android:text="Comprar"` e substituímos o texto.

Essa inconveniência ocorre pelo fato de não termos definido os textos (*strings*) de maneira centralizada para podermos reaproveitá-los ao longo do nosso aplicativo. Para fazermos isso, devemos criar um arquivo XML na pasta `res/values` que, geralmente, recebe o nome de `strings.xml`.

Nesse arquivo devemos definir os textos da seguinte forma:

```
1 <resources>
2     <string name="buyButton">Comprar</string>
3 </resources>
```

Código XML 3.34: res/values/strings.xml

A instrução da linha 3 está registrando um recurso do tipo `string` cujo nome e valor serão, respectivamente, `buyButton` e `Comprar`.

Em seguida, devemos alterar o atributo `android:text` de todos os botões que possuem o texto `Comprar` da seguinte forma:

```
1 ...
2 <Button
3     android:layout_width="wrap_content"
4     android:layout_height="wrap_content"
5     android:text="@string/buyButton" />
6 ...
```

Código XML 3.35: Refatorando o botão Comprar

Repare que na linha 4 substituímos o valor `Comprar` por `@string/buyButton`. O símbolo `@string/` nos permite acessar qualquer recurso do tipo `string` registrado em nossa aplicação.

No momento em que o XML do botão for processado, o valor `@string/buyButton` será resolvido para o texto Comprar.

Com isso, se algum dia o texto dos botões tiver que ser alterado, devemos alterar apenas a definição da string no arquivo `res/values/strings.xml`.



Mais Sobre

Centralizar as definições das strings do aplicativo é interessante para facilitar a alteração das mesmas por outros valores. Porém, esse não é o único benefício da centralização.

Quando desejamos disponibilizar o nosso aplicativo em diversos idiomas, devemos tomar o cuidado de traduzir os textos de acordo com o idioma escolhido pelo usuário e a centralização das definições de strings será muito útil nesse processo.

No sistema Android, possuímos um mecanismo interessante que nos permite facilmente substituir as definições de strings por outras. Na realidade, esse mecanismo nos permite substituir um ou mais recursos registrados no aplicativo.

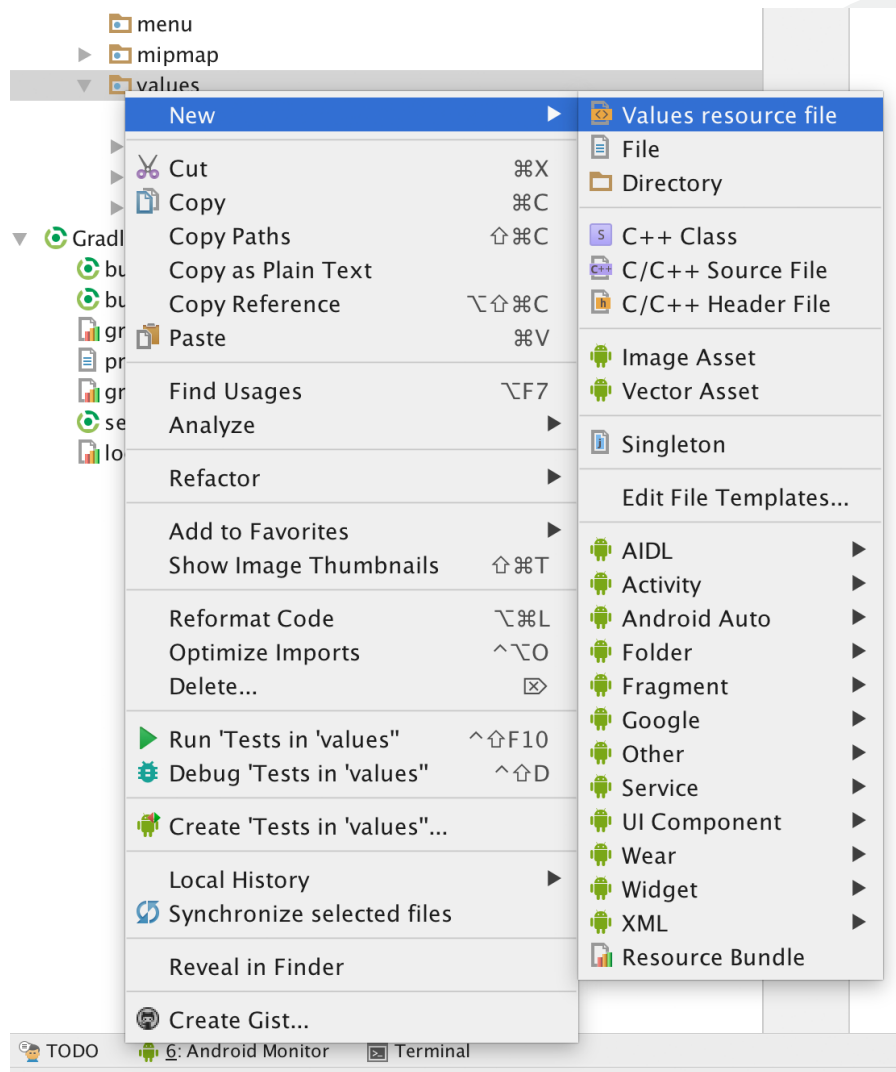
O mecanismo se chama “Localização” e maiores detalhes podem ser encontrados no seguinte endereço:

<http://developer.android.com/guide/topics/resources/localization.html>



Exercícios de Fixação

37 No projeto **Interfaces**, crie um novo arquivo de recursos chamado `strings.xml` e salve o mesmo no diretório `res/values`. Caso o arquivo já exista, pule para o próximo exercício.



- 38 Abra o arquivo `strings.xml` previamente criado e insira as linhas destacadas:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <string name="app_name">Interfaces</string>
4   <string name="name">Name:</string>
5   <string name="email">Email:</string>
6   <string name="label_city">City:</string>
7   <string name="label_newsletter">Do you want to receive news by email?</string>
8   <string name="label_gender">Gender:</string>
9   <string name="label_gender_male">Male</string>
10  <string name="label_gender_female">Female</string>
11  <string name="label_signup">Sign Up</string>
12 </resources>

```

Código XML 3.36: `strings.xml`

- 39 Agora, repita o processo e crie outro arquivo de recurso chamado `strings.xml`. Porém, desta vez, crie o arquivo no diretório `res/values-pt-rBR` e insira o seguinte conteúdo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="name">Nome:</string>
4     <string name="email">Email:</string>
5     <string name="label_city">Cidade:</string>
6     <string name="label_newsletter">Deseja receber noticias por email?</string>
7     <string name="label_gender">Sexo:</string>
8     <string name="label_gender_male">Masculino</string>
9     <string name="label_gender_female">Feminino</string>
10    <string name="label_signup">Cadastrar</string>
11 </resources>

```

Código XML 3.37: strings.xml

40 Abra o arquivo activity_main.xml e altere as linhas destacadas:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ScrollView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">
8
9     <LinearLayout
10        android:layout_width="match_parent"
11        android:layout_height="wrap_content"
12        android:orientation="vertical"
13        android:paddingBottom="@dimen/activity_vertical_margin"
14        android:paddingLeft="@dimen/activity_horizontal_margin"
15        android:paddingRight="@dimen/activity_horizontal_margin"
16        android:paddingTop="@dimen/activity_vertical_margin">
17
18        <ImageView
19            android:layout_width="match_parent"
20            android:layout_height="100dp"
21            android:scaleType="centerInside"
22            android:src="@drawable/k19_logo" />
23
24        <TextView
25            android:layout_width="wrap_content"
26            android:layout_height="wrap_content"
27            android:text="@string/name" />
28
29        <EditText
30            android:id="@+id/nome"
31            android:layout_width="match_parent"
32            android:layout_height="wrap_content"
33            android:layout_marginBottom="16dp"
34            android:hint="Ex: Rafael Cosentino"
35            android:imeOptions="actionNext"
36            android:inputType="textPersonName" />
37
38        <TextView
39            android:layout_width="wrap_content"
40            android:layout_height="wrap_content"
41            android:text="@string/email" />
42
43        <EditText
44            android:id="@+id/email"
45            android:layout_width="match_parent"
46            android:layout_height="wrap_content"
47            android:layout_marginBottom="16dp"
48            android:hint="Ex: contato@k19.com.br"
49            android:imeOptions="actionNext"
50            android:inputType="textEmailAddress" />
51

```

```

52     <TextView
53         android:layout_width="wrap_content"
54         android:layout_height="wrap_content"
55         android:text="@string/label_city" />
56
57     <Spinner
58         android:id="@+id/cidades"
59         android:layout_width="match_parent"
60         android:layout_height="wrap_content"
61         android:layout_marginBottom="16dp" />
62
63     <CheckBox
64         android:id="@+id/noticias"
65         android:layout_width="wrap_content"
66         android:layout_height="wrap_content"
67         android:layout_marginBottom="16dp"
68         android:checked="true"
69         android:text="@string/label_newsletter" />
70
71     <TextView
72         android:layout_width="match_parent"
73         android:layout_height="wrap_content"
74         android:text="@string/label_gender" />
75
76     <RadioGroup
77         android:id="@+id/sexo"
78         android:layout_width="match_parent"
79         android:layout_height="wrap_content"
80         android:layout_marginBottom="16dp"
81         android:orientation="horizontal">
82
83         <RadioButton
84             android:id="@+id/masculino"
85             android:layout_width="wrap_content"
86             android:layout_height="wrap_content"
87             android:checked="true"
88             android:text="@string/label_gender_male"/>
89
90         <RadioButton
91             android:id="@+id/feminino"
92             android:layout_width="wrap_content"
93             android:layout_height="wrap_content"
94             android:text="@string/label_gender_female"/>
95
96     </RadioGroup>
97
98     <Button
99         android:id="@+id/cadastrar"
100         android:layout_width="match_parent"
101         android:layout_height="wrap_content"
102         android:background="#064e83"
103         android:text="@string/label_signup"
104         android:textColor="#ffffff" />
105 </LinearLayout>
106
107 </ScrollView>

```

Código XML 3.38: activity_main.xml

- 41 Altere também o arquivo MainActivity.java conforme as linhas destacadas abaixo:

```

1  ...
2  @Override
3  protected void onCreate(Bundle savedInstanceState) {
4      super.onCreate(savedInstanceState);
5      setContentView(R.layout.activity_main);
6  }

```

```

7     initSpinner();
8     initButton();
9     initListView();
10    initAdvancedListView();
11    initGridView();
12    initAdvancedGridView();
13    addViewToLinearLayout();
14 }
15 ...

```

Código Java 3.34: MainActivity.java

- 42 Execute o aplicativo e verifique os textos dos rótulos dos campos exibidos na tela. Repare que, se o dispositivo estiver configurado para o idioma português do Brasil, os textos aparecerão em português do Brasil. Porém, se estiver configurado para qualquer outro idioma, os textos aparecerão em inglês.



Dimensões

Conforme desenvolvemos um aplicativo, definimos diversas medidas para os mais diferentes tipos de elementos como, por exemplo, textos, botões, barras de título, caixas de diálogo e etc.

No início do desenvolvimento pode não ficar tão evidente, porém após a criação de algumas telas percebemos que muitas dessas medidas começam a se repetir.

Suponha que tenhamos decidido que em nosso aplicativo a fonte utilizada nos textos deva ter 18px (pixels) de altura. Para isso, poderíamos ter criado os textos da seguinte maneira:

```

1 <TextView
2     android:layout_width="match_parent"
3     android:layout_height="wrap_content"
4     android:textSize="18px"
5     android:text="@string/textoDeExemplo" />

```

Código XML 3.39: Exemplo de texto com altura de 18px

Na quarta linha do exemplo acima definimos a medida da altura do texto. Para facilitarmos a manutenção, podemos centralizar a definição das medidas utilizadas no aplicativo em um arquivo XML na pasta `res/values`. Apesar de não existir um nome obrigatório para esse arquivo, geralmente, utilizamos o nome `dimens.xml`. Veja um exemplo:

```

1 <resources>
2     <dimen name="textoPadrao">18px</dimen>
3     <dimen name="alturaDaBarraDeTitulo">40px</dimen>
4 </resources>

```

Código XML 3.40: Exemplo do arquivo `res/values/dimens.xml`

No linha 2 do exemplo acima definimos a dimensão de 18px e atribuímos a ela o nome `textoPadrao`. Observe que a dimensão também é tratada como um recurso e podemos referenciá-la nos XMLs das nossas telas, assim como fizemos com as strings:

```

1 <TextView
2     android:layout_width="match_parent"

```

```

3     android:layout_height="wrap_content"
4     android:textSize="@dimen/textoPadrao"
5     android:text="@string/textoDeExemplo" />

```

Código XML 3.41: Exemplo de texto com altura definida no `dimens.xml`



Exercícios de Fixação

43 No projeto **Interfaces**, crie um novo arquivo de recursos chamado `dimens.xml` e salve o mesmo no diretório `res/values`. Caso o arquivo já exista, pule para o próximo exercício.

44 Altere o arquivo `dimens.xml` conforme o código abaixo:

```

1 <resources>
2     <dimen name="activity_horizontal_margin">16dp</dimen>
3     <dimen name="activity_vertical_margin">16dp</dimen>
4     <dimen name="campo_margem_inferior">16dp</dimen>
5 </resources>

```

Código XML 3.42: `dimens.xml`

45 Altere o arquivo `activity_main.xml` conforme as linhas destacadas abaixo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ScrollView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">
8
9     <LinearLayout
10        android:layout_width="match_parent"
11        android:layout_height="wrap_content"
12        android:orientation="vertical"
13        android:paddingBottom="@dimen/activity_vertical_margin"
14        android:paddingLeft="@dimen/activity_horizontal_margin"
15        android:paddingRight="@dimen/activity_horizontal_margin"
16        android:paddingTop="@dimen/activity_vertical_margin">
17
18        <ImageView
19            android:layout_width="match_parent"
20            android:layout_height="100dp"
21            android:scaleType="centerInside"
22            android:src="@drawable/k19_logo" />
23
24        <TextView
25            android:layout_width="wrap_content"
26            android:layout_height="wrap_content"
27            android:text="@string/name" />
28
29        <EditText
30            android:id="@+id/nome"
31            android:layout_width="match_parent"
32            android:layout_height="wrap_content"
33            android:layout_marginBottom="@dimen/campo_margem_inferior"
34            android:hint="Ex: Rafael Cosentino"
35            android:imeOptions="actionNext"

```

```

36         android:inputType="textPersonName" />
37
38     <TextView
39         android:layout_width="wrap_content"
40         android:layout_height="wrap_content"
41         android:text="@string/email" />
42
43     <EditText
44         android:id="@+id/email"
45         android:layout_width="match_parent"
46         android:layout_height="wrap_content"
47         android:layout_marginBottom="@dimen/campo_margem_inferior"
48         android:hint="Ex: contato@k19.com.br"
49         android:imeOptions="actionNext"
50         android:inputType="textEmailAddress" />
51
52     <TextView
53         android:layout_width="wrap_content"
54         android:layout_height="wrap_content"
55         android:text="@string/label_city" />
56
57     <Spinner
58         android:id="@+id/cidades"
59         android:layout_width="match_parent"
60         android:layout_height="wrap_content"
61         android:layout_marginBottom="@dimen/campo_margem_inferior" />
62
63     <CheckBox
64         android:id="@+id/noticias"
65         android:layout_width="wrap_content"
66         android:layout_height="wrap_content"
67         android:layout_marginBottom="@dimen/campo_margem_inferior"
68         android:checked="true"
69         android:text="@string/label_newsletter" />
70
71     <TextView
72         android:layout_width="match_parent"
73         android:layout_height="wrap_content"
74         android:text="@string/label_gender" />
75
76     <RadioGroup
77         android:id="@+id/sexo"
78         android:layout_width="match_parent"
79         android:layout_height="wrap_content"
80         android:layout_marginBottom="@dimen/campo_margem_inferior"
81         android:orientation="horizontal">
82
83         <RadioButton
84             android:id="@+id/masculino"
85             android:layout_width="wrap_content"
86             android:layout_height="wrap_content"
87             android:checked="true"
88             android:text="@string/label_gender_male" />
89
90         <RadioButton
91             android:id="@+id/feminino"
92             android:layout_width="wrap_content"
93             android:layout_height="wrap_content"
94             android:text="@string/label_gender_female" />
95
96     </RadioGroup>
97
98     <Button
99         android:id="@+id/cadastrar"
100         android:layout_width="match_parent"
101         android:layout_height="wrap_content"
102         android:background="#064e83"
103         android:text="@string/label_signup"
104         android:textColor="#ffffff" />
105 </LinearLayout>

```

```
106  
107 </ScrollView>
```

Código XML 3.43: activity_main.xml

- 46 Execute o aplicativo e repare que as medidas definidas no arquivo `dimens.xml` foram aplicadas corretamente na tela.



Cores

Quando baixamos em nossos celulares alguns aplicativos de lojas como a Play Store do Google, muitas vezes nos impressionamos com o visual do mesmo. Às vezes pela excelente diagramação e organização das informações, às vezes pela beleza dos recursos gráficos como ícones e imagens. Independentemente do motivo, é muito provável que tudo isso não nos causaria a mesma impressão se o responsável pelo design do aplicativo tivesse escolhido as cores “erradas”.

Existem diversos estudos sobre a aplicação das cores (isoladas ou combinadas) e o impacto que elas causam no sentimento humano. Não é à toa que os designers se preocupam tanto em encontrar a combinação perfeita e estabelecer a paleta de cores de um aplicativo e, com isso, provocar o sentimento desejado nas pessoas.

As cores em um aplicativo Android devem ser definidas seguindo o formato RGB (Red, Green, Blue - Vermelho, Verde, Azul) ou ARGB (Alpha channel, Red, Green, Blue - Canal alpha, Vermelho, Verde, Azul).

O formato RGB pode ser representado por três ou seis dígitos hexadecimais prefixados pelo símbolo #. Por exemplo, a cor vermelha pode ser representada por `#f00` ou `#ff0000`. A diferença entre o formato de três e seis dígitos está na quantidade de cores que cada um pode representar. O formato de três dígitos nos permite definir até 4.096 cores. Já o de seis dígitos, 16.777.216 cores.

Assim como o RGB, o formato ARGB também pode ser representado por dígitos hexadecimais prefixados pelo símbolo #, porém com quatro ou oito dígitos. Neste formato, o canal alpha representa a intensidade da transparência da cor. Portanto, se quisermos representar a cor vermelha com aproximadamente 50% de opacidade, podemos escrever `#8f00` ou `#88ff0000`.



Mais Sobre

Para maiores detalhes sobre a representação hexadecimal do formato RGB e ARGB acesse:

RGB - https://en.wikipedia.org/wiki/Web_colors

ARGB - https://en.wikipedia.org/wiki/RGBA_color_space

Para centralizarmos as definições das cores em um aplicativo Android, devemos criar um arquivo de recurso no diretório `res/values`. Apesar de não existir um nome obrigatório para esse arquivo, geralmente, utilizamos o nome `colors.xml`. Veja um exemplo:

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



```

2 <resources>
3   <color name="colorPrimary">#3F51B5</color>
4   <color name="colorPrimaryDark">#303F9F</color>
5   <color name="colorAccent">#FF4081</color>
6 </resources>

```

Código XML 3.44: Exemplo do arquivo res/values/colors.xml

No linha 3 do exemplo acima definimos a cor `colorPrimary` e atribuímos a ela o valor `#3F51B5`. Observe que a cor também é tratada como um recurso e podemos referenciá-la nos XMLs das nossas telas, assim como fizemos com as strings e dimensões:

```

1 <TextView
2   android:layout_width="match_parent"
3   android:layout_height="wrap_content"
4   android:textColor="@color/colorPrimary"
5   android:textSize="@dimen/textoPadrao"
6   android:text="@string/textoDeExemplo" />

```

Código XML 3.45: Exemplo de texto com a cor definida no colors.xml



Exercícios de Fixação

- 47 No projeto **Interfaces**, crie um novo arquivo de recursos chamado `colors.xml` e salve o mesmo no diretório `res/values`. Caso o arquivo já exista, pule para o próximo exercício.
- 48 Altere o arquivo `colors.xml` conforme o código abaixo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <color name="colorPrimary">#3F51B5</color>
4   <color name="colorPrimaryDark">#303F9F</color>
5   <color name="colorAccent">#FF4081</color>
6   <color name="botao">#064e83</color>
7 </resources>

```

Código XML 3.46: colors.xml

- 49 Altere o arquivo `activity_main.xml` conforme as linhas destacadas abaixo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ScrollView
3   xmlns:android="http://schemas.android.com/apk/res/android"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   tools:context=".MainActivity">
8
9   <LinearLayout
10    android:layout_width="match_parent"
11    android:layout_height="wrap_content"
12    android:orientation="vertical"
13    android:paddingBottom="@dimen/activity_vertical_margin"
14    android:paddingLeft="@dimen/activity_horizontal_margin"
15    android:paddingRight="@dimen/activity_horizontal_margin"
16    android:paddingTop="@dimen/activity_vertical_margin">

```

```

17
18     <ImageView
19         android:layout_width="match_parent"
20         android:layout_height="100dp"
21         android:scaleType="centerInside"
22         android:src="@drawable/k19_logo" />
23
24     <TextView
25         android:layout_width="wrap_content"
26         android:layout_height="wrap_content"
27         android:text="@string/name" />
28
29     <EditText
30         android:id="@+id/nome"
31         android:layout_width="match_parent"
32         android:layout_height="wrap_content"
33         android:layout_marginBottom="@dimen/campo_margem_inferior"
34         android:hint="Ex: Rafael Cosentino"
35         android:imeOptions="actionNext"
36         android:inputType="textPersonName" />
37
38     <TextView
39         android:layout_width="wrap_content"
40         android:layout_height="wrap_content"
41         android:text="@string/email" />
42
43     <EditText
44         android:id="@+id/email"
45         android:layout_width="match_parent"
46         android:layout_height="wrap_content"
47         android:layout_marginBottom="@dimen/campo_margem_inferior"
48         android:hint="Ex: contato@k19.com.br"
49         android:imeOptions="actionNext"
50         android:inputType="textEmailAddress" />
51
52     <TextView
53         android:layout_width="wrap_content"
54         android:layout_height="wrap_content"
55         android:text="@string/label_city" />
56
57     <Spinner
58         android:id="@+id/cidades"
59         android:layout_width="match_parent"
60         android:layout_height="wrap_content"
61         android:layout_marginBottom="@dimen/campo_margem_inferior" />
62
63     <CheckBox
64         android:id="@+id/noticias"
65         android:layout_width="wrap_content"
66         android:layout_height="wrap_content"
67         android:layout_marginBottom="@dimen/campo_margem_inferior"
68         android:checked="true"
69         android:text="@string/label_newsletter" />
70
71     <TextView
72         android:layout_width="match_parent"
73         android:layout_height="wrap_content"
74         android:text="@string/label_gender" />
75
76     <RadioGroup
77         android:id="@+id/sexo"
78         android:layout_width="match_parent"
79         android:layout_height="wrap_content"
80         android:layout_marginBottom="@dimen/campo_margem_inferior"
81         android:orientation="horizontal">
82
83         <RadioButton
84             android:id="@+id/masculino"
85             android:layout_width="wrap_content"
86             android:layout_height="wrap_content"

```

```

87         android:checked="true"
88         android:text="@string/label_gender_male" />
89
90     <RadioButton
91         android:id="@+id/feminino"
92         android:layout_width="wrap_content"
93         android:layout_height="wrap_content"
94         android:text="@string/label_gender_female" />
95
96 </RadioGroup>
97
98 <Button
99     android:id="@+id/cadastrar"
100    android:layout_width="match_parent"
101    android:layout_height="wrap_content"
102    android:background="@color/botao"
103    android:text="@string/label_signup"
104    android:textColor="#ffffff" />
105 </LinearLayout>
106
107 </ScrollView>

```

Código XML 3.47: activity_main.xml

- 50 Execute o aplicativo e repare que a cor definida no arquivo `colors.xml` foi aplicada corretamente no botão.



Dialogs

As caixas de diálogo são pequenas janelas que “flutuam” sobre uma tela (activity) e muitos aplicativos utilizam esse recurso para que o usuário possa ser notificado de algo ou para que ele possa tomar uma decisão antes de dar continuidade no que estava fazendo na tela.

A API do Android nos disponibiliza uma série de tipos de caixas de diálogo. Um desses tipos é o `AlertDialog`.

Para criarmos um `AlertDialog` dentro de uma activity, podemos utilizar a classe `AlertDialog.Builder`. Veja o exemplo:

```

1 AlertDialog.Builder builder = new AlertDialog.Builder(this);
2 builder.setMessage("Mensagem de teste");
3 builder.setTitle("Título da caixa");
4
5 AlertDialog dialog = builder.create();

```

Código Java 3.35: Exemplo do AlertDialog

No exemplo acima, criamos uma caixa de diálogo com um título e uma mensagem. Nenhum botão foi adicionado e, para fechar a caixa, o usuário deverá pressionar a tecla “back” do aparelho. Em algumas situações isso pode causar um entendimento errado por parte do usuário, pois esse poderá ficar sem saber o que fazer. Para evitarmos esse tipo de situação, podemos acrescentar até três botões à uma caixa de diálogo do tipo `AlertDialog`:

```

1 AlertDialog.Builder builder = new AlertDialog.Builder(this);
2 builder.setMessage("Mensagem de teste");
3 builder.setTitle("Título da caixa");
4

```

```

5 builder.setPositiveButton("Botão 1", new DialogInterface.OnClickListener() {
6     public void onClick(DialogInterface dialog, int id) {
7         // Método executado ao pressionar o botão
8     }
9 });
10 builder.setNegativeButton("Botão 2", new DialogInterface.OnClickListener() {
11     public void onClick(DialogInterface dialog, int id) {
12         // Método executado ao pressionar o botão
13     }
14 });
15 builder.setNeutralButton("Botão 3", new DialogInterface.OnClickListener() {
16     public void onClick(DialogInterface dialog, int id) {
17         // Método executado ao pressionar o botão
18     }
19 });
20
21 AlertDialog dialog = builder.create();

```

Código Java 3.36: Exemplo de botões no AlertDialog



Exercícios de Fixação

- 51 No projeto **Interfaces**, crie um novo arquivo XML chamado **activity_dialog.xml** na pasta **layout**.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:id="@+id/linear_layout_dialog"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical">
7
8     <Button
9         android:id="@+id/show_hide_dialog_btn"
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="Exibir caixa de diálogo" />
13
14 </LinearLayout>

```

Código XML 3.48: activity_dialog.xml

- 52 Acrescente o método **initAlertDialog()** na classe **MainActivity**.

```

1 private void initAlertDialog() {
2     Button button = (Button) findViewById(R.id.show_hide_dialog_btn);
3     final LinearLayout linearLayout =
4         (LinearLayout) findViewById(R.id.linear_layout_dialog);
5
6     if(button == null || linearLayout == null) {
7         return;
8     }
9
10    AlertDialog.Builder builder = new AlertDialog.Builder(this);
11    builder.setMessage("Mensagem de teste");
12    builder.setTitle("Título da caixa");
13
14    builder.setPositiveButton("Botão 1", new DialogInterface.OnClickListener() {
15        public void onClick(DialogInterface dialog, int id) {
16            TextView message = new TextView(MainActivity.this);
17            message.setText("Botão 1 clicado");
18            linearLayout.addView(message);
19        }
20    });
21 }

```

```

19     }
20   });
21   builder.setNegativeButton("Botão 2", new DialogInterface.OnClickListener() {
22     public void onClick(DialogInterface dialog, int id) {
23       TextView message = new TextView(MainActivity.this);
24       message.setText("Botão 2 clicado");
25       linearLayout.addView(message);
26     }
27   });
28   builder.setNeutralButton("Botão 3", new DialogInterface.OnClickListener() {
29     public void onClick(DialogInterface dialog, int id) {
30       TextView message = new TextView(MainActivity.this);
31       message.setText("Botão 3 clicado");
32       linearLayout.addView(message);
33     }
34   });
35
36   final AlertDialog dialog = builder.create();
37
38   button.setOnClickListener(new View.OnClickListener() {
39     @Override
40     public void onClick(View v) {
41       dialog.show();
42     }
43   });
44 }

```

Código Java 3.37: MainActivity.java

- 53 Altere o método **onCreate()** da classe **MainActivity** de acordo com as linhas destacadas abaixo:

```

1  @Override
2  protected void onCreate(Bundle savedInstanceState) {
3      super.onCreate(savedInstanceState);
4      setContentView(R.layout.activity_dialog);
5
6      initSpinner();
7      initButton();
8      initListView();
9      initAdvancedListView();
10     initGridView();
11     initAdvancedGridView();
12     addViewToLinearLayout();
13     initAlertDialog();
14 }

```

Código Java 3.38: MainActivity.java

- 54 Execute o aplicativo e observe o resultado.





O que é um *Intent*

Intents são objetos responsáveis por passar informações, como se fossem mensagens, para os principais componentes da API do Android, como as *Activities*, *Services* e *BroadCast Receivers*.

Para que um destes componentes seja instanciado, é necessário que seja criado um *Intent*, mesmo quando não temos nenhuma informação para passar para o componente criado.

Quando usado em conjunto com *Intent Filters* podemos até iniciar uma *Activity* de outros aplicativo, ou o inverso, deixar que um outro aplicativo inicie uma das nossas *Activities*.



Usando o *Intent* com *Activities*

Para iniciar uma nova *Activity* é necessário usar o método **startActivity()** presente no objeto **Context**, ou na *Activity*.

```
1 Intent intent = new Intent(this, NewActivity.class);  
2 startActivity(intent);
```

Código Java 4.1: Exemplo.java

No exemplo acima, estamos iniciando uma *Activity* cujo nome é *NewActivity*. O primeiro parâmetro que passamos para o construtor do *Intent* é o contexto, no caso **this** se refere à própria *Activity* atual, que está chamando a próxima.



Exercícios de Fixação

- 1 Crie um novo projeto Android, da mesma forma que foi criado no capítulo anterior. Use como nome para o projeto **Intents**. O nome do pacote deve ser **br.com.k19.android.cap04**, e o nome da *activity* deve ser **MainActivity**.
- 2 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```
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:orientation="vertical" >
```

```

6
7     <Button
8         android:id="@+id/main_button"
9         android:layout_width="match_parent"
10        android:layout_height="wrap_content"
11        android:text="@string/next_screen"/>
12
13 </LinearLayout>

```

Código XML 4.1: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="app_name">Intents</string>
4     <string name="next_screen">Próxima tela</string>
5 </resources>

```

Código XML 4.2: strings.xml

A seguir edite o arquivo **MainActivity.java** para que ele fique com o seguinte conteúdo:

```

1 package br.com.k19.android.cap04;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.view.View.OnClickListener;
8 import android.widget.Button;
9
10 public class MainActivity extends Activity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.main);
16         Button button = (Button) findViewById(R.id.main_button);
17
18         button.setOnClickListener(new OnClickListener() {
19
20             public void onClick(View v) {
21                 Intent intent = new Intent(MainActivity.this, SecondActivity.class);
22                 startActivity(intent);
23             }
24         });
25     }
26 }

```

Código Java 4.2: MainActivity.java

3 Crie um novo arquivo XML na pasta de layouts chamado **second.xml** com o conteúdo abaixo:

```

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:orientation="vertical" >
6
7     <TextView
8         android:layout_width="match_parent"
9         android:layout_height="wrap_content"

```



```

10     android:text="@string/second_screen" />
11
12 </LinearLayout>

```

Código XML 4.3: second.xml

Adicione uma nova entrada no arquivo **strings.xml**:

```

1 <string name="second_screen">Nova tela</string>

```

Código XML 4.4: strings.xml

Crie uma nova classe chamada **SecondActivity** que herda *Activity* e possui o conteúdo abaixo:

```

1 package br.com.k19.android.cap04;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.widget.EditText;
6
7 public class SecondActivity extends Activity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.second);
13    }
14 }

```

Código Java 4.3: SecondActivity.java

Se lembre de adicionar os nomes das *activities* no **AndroidManifest.xml**. Para fazer isso basta adicionar a seguinte *tag* dentro da *tag application*:

```

1 <activity android:name=".SecondActivity" />

```

Código XML 4.5: AndroidManifest.xml

Após isso, rode a aplicação e veja o resultado. Você deve ter um botão que abre uma nova tela, e apertando *back* você volta a tela anterior.



Usando *Intents* para passar dados

No exercício anterior, foi instanciada uma nova *Activity*, mas não foi passada nenhuma informação para ela. Isso pode ser feito utilizando o método `putExtra` do *Intent*.

```

1 Intent intent = new Intent(this, NewActivity.class);
2 intent.putExtra("curso", "Android");
3 intent.putExtra("sigla", "k41");
4 intent.putExtra("total", 25);

```

Código Java 4.4: Exemplo

Este método tem *overloading*, logo podemos passar diferentes tipos. Podemos passar tipos primitivos e Strings. O primeiro parâmetro indica qual a chave que estamos usando para a informação. O segundo é o valor da informação que estamos passando.

Do outro lado, na *Activity* que está sendo criada, podemos obter os valores através do *Bundle* que pode ser obtido a partir do método *getExtras* presente no *Intent*:

```

1  protected void onCreate(Bundle savedInstanceState) {
2      super.onCreate(savedInstanceState);
3      setContentView(R.layout.some_layout);
4
5      Bundle extras = getIntent().getExtras();
6      String curso = extras.getString("curso");
7      String sigla = extras.getString("sigla");
8      int total = extras.getInt("total");
9
10     ...
11 }

```

Código Java 4.5: Exemplo



Exercícios de Fixação

- 4 Edite o conteúdo do arquivo **second.xml**, para o exemplo abaixo:

```

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:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/name_label"
9          android:layout_width="match_parent"
10         android:layout_height="wrap_content"
11         android:text="@string/name" />
12
13     <EditText
14         android:id="@+id/name_edit_text"
15         android:layout_width="match_parent"
16         android:layout_height="wrap_content"
17         android:inputType="textCapWords" />
18
19     <TextView
20         android:id="@+id/age_label"
21         android:layout_width="match_parent"
22         android:layout_height="wrap_content"
23         android:text="@string/age" />
24
25     <EditText
26         android:id="@+id/age_edit_text"
27         android:layout_width="match_parent"
28         android:layout_height="wrap_content"
29         android:inputType="number" />
30
31     <Button
32         android:id="@+id/next_button"
33         android:layout_width="match_parent"
34         android:layout_height="wrap_content"
35         android:text="@string/next_screen" />
36
37 </LinearLayout>

```

Código XML 4.6: second.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="app_name">Intents</string>
4     <string name="next_screen">Próxima tela</string>
5     <string name="name">Nome</string>
6     <string name="age">Idade</string>
7     <string name="user_name">Nome: %1s</string>
8     <string name="user_age">Idade: %1s</string>
9 </resources>

```

Código XML 4.7: strings.xml

A seguir edite o arquivo **SecondActivity.java** para que ele fique com o seguinte conteúdo:

```

1 package br.com.k19.android.cap04;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.view.View.OnClickListener;
8 import android.widget.Button;
9 import android.widget.EditText;
10
11 public class SecondActivity extends Activity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.second);
17
18         final EditText nameEditText = (EditText) findViewById(R.id.name_edit_text);
19         final EditText ageEditText = (EditText) findViewById(R.id.age_edit_text);
20         Button button = (Button) findViewById(R.id.next_button);
21
22         button.setOnClickListener(new OnClickListener() {
23
24             public void onClick(View v) {
25                 String name = nameEditText.getText().toString();
26                 String age = ageEditText.getText().toString();
27
28                 Intent intent = new Intent(SecondActivity.this, ThirdActivity.class);
29                 intent.putExtra("name", name);
30                 intent.putExtra("age", age);
31                 startActivity(intent);
32             }
33         });
34
35     }
36
37 }

```

Código Java 4.6: SecondActivity.java

- 5 Crie um novo arquivo XML na pasta de layouts chamado **third.xml** com o conteúdo abaixo:

```

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:orientation="vertical" >
6
7
8     <TextView
9         android:id="@+id/name"

```

```

10     android:layout_width="match_parent"
11     android:layout_height="wrap_content" />
12
13     <TextView
14         android:id="@+id/age"
15         android:layout_width="match_parent"
16         android:layout_height="wrap_content" />
17
18 </LinearLayout>

```

Código XML 4.8: third.xml

Adicione uma nova entrada no arquivo **strings.xml**:

```

1 <string name="second_screen">Nova tela</string>

```

Código XML 4.9: strings.xml

Crie uma nova classe chamada **ThirdActivity** que herda *Activity* e possui o conteúdo abaixo:

```

1 package br.com.k19.android.cap04;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.widget.TextView;
6
7 public class ThirdActivity extends Activity {
8
9     @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.third);
13
14         Bundle extras = getIntent().getExtras();
15         String name = extras.getString("name");
16         String age = extras.getString("age");
17
18         TextView nameTextView = (TextView) findViewById(R.id.name);
19         TextView ageTextView = (TextView) findViewById(R.id.age);
20
21         nameTextView.setText(getString(R.string.user_name, name));
22         ageTextView.setText(getString(R.string.user_age, age));
23     }
24 }

```

Código Java 4.7: ThirdActivity.java

Adicione a nova *Activity* ao **AndroidManifest.xml**.

```

1 <activity android:name=".ThirdActivity" />

```

Código XML 4.10: AndroidManifest.xml

Após isso, rode a aplicação e veja o resultado. Faça alguns testes, clique nos botões, digite nome e idade, e veja se está funcionando como o esperado.



Abrindo outros aplicativos

É possível abrir outros aplicativos utilizando intents. Para isso, é necessário passar uma *flag* que chamamos de *action*. Dependendo do tipo de *action* que passarmos, um novo aplicativo será aberto

para executar a ação. Este tipo de intent é chamado de implícito, porque não é especificado qual a activity que será aberta. Apenas passamos uma ação, e o sistema irá decidir qual activity deverá ser utilizada nesse caso.



Exercícios de Fixação

6 Crie um novo projeto Android, da mesma forma que foi criado no capítulo anterior. Use como nome para o projeto **IntentActions**. O nome do pacote deve ser **br.com.k19.android.cap04_02**, e o nome da *activity* deve ser **MainActivity**.

7 Edite o conteúdo do arquivo **main.xml**, para o exemplo abaixo:

```

1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:orientation="vertical" >
6
7   <Button
8     android:id="@+id/view_site_button"
9     android:layout_width="match_parent"
10    android:layout_height="wrap_content"
11    android:text="@string/view_site_label" />
12
13   <Button
14     android:id="@+id/send_email_button"
15     android:layout_width="match_parent"
16     android:layout_height="wrap_content"
17     android:text="@string/send_email_label" />
18
19   <Button
20     android:id="@+id/make_call_button"
21     android:layout_width="match_parent"
22     android:layout_height="wrap_content"
23     android:text="@string/make_call_label" />
24
25 </LinearLayout>

```

Código XML 4.11: *second.xml*

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <resources>
2
3   <string name="app_name">IntentActions</string>
4   <string name="hello_world">Hello world!</string>
5   <string name="menu_settings">Settings</string>
6   <string name="title_activity_main">MainActivity</string>
7   <string name="view_site_label">Ver site da K19</string>
8   <string name="send_email_label">Enviar email de contato</string>
9   <string name="make_call_label">Fazer ligação</string>
10
11 </resources>

```

Código XML 4.12: *strings.xml*

A seguir edite o arquivo **MainActivity.java** para que ele fique com o seguinte conteúdo:

```
1 package br.com.k19.android.cap04;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.net.Uri;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.view.View.OnClickListener;
9 import android.widget.Button;
10
11 public class MainActivity extends Activity {
12
13     @Override
14     public void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.main);
17
18         Button viewSiteButton = (Button) findViewById(R.id.view_site_button);
19         Button sendEmailButton = (Button) findViewById(R.id.send_email_button);
20         Button makeCallButton = (Button) findViewById(R.id.make_call_button);
21
22         viewSiteButton.setOnClickListener(new OnClickListener() {
23
24             @Override
25             public void onClick(View v) {
26                 Intent intent = new Intent(Intent.ACTION_VIEW, Uri
27                     .parse("http://k19.com.br"));
28                 startActivity(intent);
29             }
30         });
31
32         sendEmailButton.setOnClickListener(new OnClickListener() {
33
34             @Override
35             public void onClick(View v) {
36                 Intent intent = new Intent(Intent.ACTION_SEND);
37                 intent.setType("plain/text");
38                 intent.putExtra(Intent.EXTRA_EMAIL,
39                     new String[] { "contato@k19.com.br" });
40                 startActivity(Intent.createChooser(intent, "Enviar email"));
41             }
42         });
43
44         makeCallButton.setOnClickListener(new OnClickListener() {
45
46             @Override
47             public void onClick(View v) {
48                 Intent intent = new Intent(Intent.ACTION_DIAL, Uri
49                     .parse("tel:2387-3791"));
50                 startActivity(intent);
51             }
52         });
53     }
54 }
```

Código Java 4.8: MainActivity.java

Após isso, adicione a permissão para fazer ligações no *AndroidManifest.xml*:

```
1 <uses-permission android:name="android.permission.CALL_PHONE" />
```

Código XML 4.13: AndroidManifest.xml

A tag deve ser adicionada dentro da tag <manifest> e fora da tag <application>.



Persistindo informação

A API do Android oferece diferentes opções quando se trata de salvar dados para serem usados posteriormente. Qual a opção é mais apropriada depende do tipo de informação que será salva e da disponibilidade que queremos que ela tenha.

Existem 4 tipos de armazenamento possíveis:

Shared Preferences - é um tipo de armazenamento que utiliza chave/valor indicado principalmente para configurações e dados isolados.

SQLite - banco de dados privado que pode ser utilizado pelo seu aplicativo. É o mais indicado quando temos várias informações com a mesma estrutura, que podem ser organizadas em tabelas e serem consultadas.

Internal Storage - armazena na memória interna do aparelho, os dados aqui armazenados são privados da sua aplicação e não podem ser acessados por outros aplicativos ou pelo usuário.

External Storage - armazena em um SD, que pode ser externo ou interno do aparelho. Os arquivos armazenados no SD são visíveis para todos, e o usuário pode alterá-los quando conecta o USB a um computador.

Neste capítulo, iremos ver os dois primeiros tipos de armazenamento. O internal e external serão vistos em capítulos mais a frente.



Usando o *SharedPreferences*

Abaixo está um exemplo de como utilizar o *SharedPreferences* para ler informações:

```
1 SharedPreferences prefs = getSharedPreferences(nome, modo);  
2 String someString = prefs.getString(chave, null);  
3 int someInt = prefs.getInt(outraChave, 0);
```

Código Java 5.1: *Exemplo.java*

Você obtém o *SharedPreferences* chamando o método *getSharedPreferences*, passando para ele uma string, que será a chave para indicar o *SharedPreferences*. Você pode utilizar vários *SharedPreferences* por aplicação se achar necessário. O modo indica a permissão do *SharedPreferences*. Se passar 0, indica modo privado.

Para ler os dados, é só chamar o método `get` correspondente ao tipo de informação que você quer. Lembre-se que no *SharedPreferences* você só pode armazenar informações de tipo primitivo e Strings. O segundo parâmetro indica um valor padrão caso a chave não seja encontrada.

Para salvar os dados no *SharedPreferences* é necessário usar um *editor*. Veja o exemplo abaixo:

```
1 SharedPreferences prefs = getSharedPreferences(nome, modo);
2 Editor editor = prefs.edit();
3 editor.putString("curso", "k41");
4 editor.commit();
```

Código Java 5.2: Exemplo.java

Obtemos o editor chamando o método `edit()`. Adicionamos informações chamando o método `put()` correspondente ao tipo que estamos armazenando. É necessário chamar o método `commit()` no final, senão as alterações não serão salvas.



Exercícios de Fixação

- 1 Crie um novo projeto Android. Use como nome para o projeto **SharedPrefs**. O nome do pacote deve ser **br.com.k19.android.cap05**, e o nome da *activity* deve ser **MainActivity**.
- 2 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent" >
5
6   <TextView
7       android:id="@+id/welcome_message"
8       android:layout_width="wrap_content"
9       android:layout_height="wrap_content"
10      android:layout_centerHorizontal="true"
11      android:layout_centerVertical="true" />
12
13   <Button
14       android:id="@+id/add_name_button"
15       android:layout_width="wrap_content"
16       android:layout_height="wrap_content"
17       android:layout_below="@id/welcome_message"
18       android:layout_centerHorizontal="true"/>
19
20 </RelativeLayout>
```

Código XML 5.1: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```
1 <resources>
2
3   <string name="app_name" >SharedPrefs</string>
4   <string name="hello_world" >Hello world!</string>
5   <string name="menu_settings" >Settings</string>
6   <string name="title_activity_main" >MainActivity</string>
7   <string name="type_your_name" >Digite seu nome</string>
8   <string name="save" >Salvar</string>
```



```

9
10 </resources>

```

Código XML 5.2: strings.xml

A seguir edite o arquivo **MainActivity.java** para que ele fique com o seguinte conteúdo:

```

1 package br.com.k19.android.cap05;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.content.SharedPreferences;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.view.View.OnClickListener;
9 import android.widget.Button;
10 import android.widget.TextView;
11
12 public class MainActivity extends Activity {
13     final static String APP_PREFS = "app_prefs";
14     final static String USERNAME_KEY = "username";
15
16     @Override
17     public void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.main);
20     }
21
22     @Override
23     protected void onResume() {
24         super.onResume();
25
26         SharedPreferences prefs = getSharedPreferences(APP_PREFS, MODE_PRIVATE);
27         String username = prefs.getString(USERNAME_KEY, null);
28
29         TextView message = (TextView) findViewById(R.id.welcome_message);
30         Button addNameButton = (Button) findViewById(R.id.add_name_button);
31
32         if (username != null) {
33             message.setText("Bem vindo, " + username + "!");
34             addNameButton.setText("Trocar de nome");
35         } else {
36             message.setText("Você não cadastrou seu nome...");
37             addNameButton.setText("Adicionar nome");
38         }
39
40         addNameButton.setOnClickListener(new OnClickListener() {
41
42             @Override
43             public void onClick(View v) {
44                 Intent intent = new Intent(MainActivity.this,
45                     AddNameActivity.class);
46                 startActivity(intent);
47             }
48         });
49     }
50 }

```

Código Java 5.3: MainActivity.java

- 3 Crie um novo arquivo XML na pasta de layouts chamado **add_name.xml** com o conteúdo abaixo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent" >

```

```

5
6
7     <EditText
8         android:id="@+id/name_edit_text"
9         android:layout_width="match_parent"
10        android:layout_height="wrap_content"
11        android:inputType="textCapWords"
12        android:hint="@string/type_your_name"
13        android:layout_centerHorizontal="true"
14        android:layout_marginTop="20dp"/>
15
16    <Button
17        android:id="@+id/add_name_button"
18        android:layout_width="wrap_content"
19        android:layout_height="wrap_content"
20        android:layout_below="@id/name_edit_text"
21        android:layout_centerHorizontal="true"
22        android:layout_marginTop="20dp"
23        android:text="@string/save"/>
24</RelativeLayout>

```

Código XML 5.3: add_name.xml

Crie uma nova classe chamada **AddNameActivity** que herda *Activity* e possui o conteúdo abaixo:

```

1 package br.com.k19.android.cap05;
2
3 import android.app.Activity;
4 import android.content.SharedPreferences;
5 import android.content.SharedPreferences.Editor;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.view.View.OnClickListener;
9 import android.widget.Button;
10 import android.widget.EditText;
11
12 public class AddNameActivity extends Activity {
13
14     private SharedPreferences prefs;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.add_name);
20
21         prefs = getSharedPreferences(MainActivity.APP_PREFS, MODE_PRIVATE);
22
23         final EditText name = (EditText) findViewById(R.id.name_edit_text);
24         Button saveButton = (Button) findViewById(R.id.add_name_button);
25
26         saveButton.setOnClickListener(new OnClickListener() {
27
28             @Override
29             public void onClick(View v) {
30                 String username = name.getText().toString();
31                 Editor editor = prefs.edit();
32                 editor.putString(MainActivity.USERNAME_KEY, username);
33                 editor.commit();
34                 finish();
35             }
36         });
37     }
38 }

```

Código Java 5.4: Main.java

Se lembre de adicionar os nomes das *activities* no **AndroidManifest.xml**. Após isso, rode a apli-

cação e veja o resultado. Feche a aplicação e abra novamente, para ver se o nome está salva.



Usando o SQLite

O *SQLite* é um banco de dados bem simples que consome poucos recursos, bastante usado em dispositivos embarcados. Para utilizar o SQLite, é necessário que você crie uma subclasse de *SQLiteOpenHelper*. Em seguida é necessário sobrescrever os métodos **onCreate()** e **onUpgrade()**. O primeiro é chamado quando ainda não existe um banco de dados, nele você deve incluir os comandos para criar tabelas e inicializar qualquer tipo de dados, se preciso. O segundo é chamado quando a versão da base de dados é alterada, e nele você deve incluir quaisquer comandos relacionados à alteração do esquema, como alterações em tabelas e colunas.

O *SQLiteOpenHelper* oferece dois métodos que serão muito usados, o **getWritableDatabase()** e **getReadableDatabase()**. Como o nome indica, estes métodos servem para obter uma instância da base de dados. Estes métodos retornam uma instância de *SQLiteDatabase*, que é utilizada para fazer consultas aos dados. Os métodos que são usados com esse propósito são o **insert()**, **update()** e **delete()**. Também são usados os métodos **query()** e **rawQuery()**. O primeiro oferece uma interface para criar consultas, enquanto o segundo permite utilizar SQL diretamente.

O resultado de uma consulta é um objeto do tipo *Cursor*, que permite iterar sobre os dados.



Exercícios de Fixação

- 4 Crie um novo projeto Android. Use como nome para o projeto **SQLite**. O nome do pacote deve ser **br.com.k19.android.cap05_02**, e o nome da *activity* deve ser **MainActivity**.
- 5 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```
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:orientation="vertical" >
6
7     <ListView
8         android:id="@android:id/list"
9         android:layout_width="match_parent"
10        android:layout_height="wrap_content" />
11
12    <TextView
13        android:id="@android:id/empty"
14        android:layout_width="match_parent"
15        android:layout_height="wrap_content"
16        android:text="@string/no_notes" />
17
18 </LinearLayout>
```

Código XML 5.4: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```
1 <resources>
2
3   <string name="app_name">SQLite</string>
4   <string name="title_activity_main">MainActivity</string>
5   <string name="add">Adicionar</string>
6   <string name="no_notes">Nenhuma anotação</string>
7   <string name="write_a_note">Escreva uma anotação</string>
8   <string name="save">Salvar</string>
9
10 </resources>
```

Código XML 5.5: strings.xml

Crie um novo arquivo java chamado **CustomSQLiteOpenHelper** com o seguinte conteúdo:

```
1 package br.com.k19.android.cap05_02;
2
3 import android.content.Context;
4 import android.database.sqlite.SQLiteDatabase;
5 import android.database.sqlite.SQLiteOpenHelper;
6
7 public class CustomSQLiteOpenHelper extends SQLiteOpenHelper {
8
9     public static final String TABLE_NOTES = "notes";
10    public static final String COLUMN_ID = "_id";
11    public static final String COLUMN_NOTES = "note";
12
13    private static final String DATABASE_NAME = "notes.db";
14    private static final int DATABASE_VERSION = 1;
15
16    // Database creation sql statement
17    private static final String DATABASE_CREATE = "create table "
18        + TABLE_NOTES + "(" + COLUMN_ID
19        + " integer primary key autoincrement, " + COLUMN_NOTES
20        + " text not null)";
21
22    public CustomSQLiteOpenHelper(Context context) {
23        super(context, DATABASE_NAME, null, DATABASE_VERSION);
24    }
25
26    @Override
27    public void onCreate(SQLiteDatabase database) {
28        database.execSQL(DATABASE_CREATE);
29    }
30
31    @Override
32    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
33        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NOTES);
34        onCreate(db);
35    }
36
37 }
```

Código Java 5.5: CustomSQLiteOpenHelper.java

Crie um arquivo java chamado **Note**, com o seguinte conteúdo:

```
1 package br.com.k19.android.cap05_02;
2
3 public class Note {
4
5     private long id;
6     private String note;
7
8     @Override
9     public String toString() {
```

```

10     return note;
11 }
12
13 public long getId() {
14     return id;
15 }
16
17 public void setId(long id) {
18     this.id = id;
19 }
20
21 public String getNote() {
22     return note;
23 }
24
25 public void setNote(String note) {
26     this.note = note;
27 }
28 }

```

Código Java 5.6: Note.java

Crie um arquivo chamado **NotesDao**, com o seguinte conteúdo:

```

1 package br.com.k19.android.cap05_02;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import android.content.ContentValues;
7 import android.content.Context;
8 import android.database.Cursor;
9 import android.database.SQLException;
10 import android.database.sqlite.SQLiteDatabase;
11
12 public class NotesDao {
13
14     private SQLiteDatabase database;
15     private String[] columns = { CustomSQLiteOpenHelper.COLUMN_ID,
16         CustomSQLiteOpenHelper.COLUMN_NOTES };
17     private CustomSQLiteOpenHelper sqliteOpenHelper;
18
19     public NotesDao(Context context) {
20         sqliteOpenHelper = new CustomSQLiteOpenHelper(context);
21     }
22
23     public void open() throws SQLException {
24         database = sqliteOpenHelper.getWritableDatabase();
25     }
26
27     public void close() {
28         sqliteOpenHelper.close();
29     }
30
31     public Note create(String note) {
32         ContentValues values = new ContentValues();
33         values.put(CustomSQLiteOpenHelper.COLUMN_NOTES, note);
34         long insertId = database.insert(CustomSQLiteOpenHelper.TABLE_NOTES, null,
35             values);
36         Cursor cursor = database.query(CustomSQLiteOpenHelper.TABLE_NOTES,
37             columns, CustomSQLiteOpenHelper.COLUMN_ID + " = " + insertId, null,
38             null, null, null);
39         cursor.moveToFirst();
40         Note newNote = new Note();
41         newNote.setId(cursor.getLong(0));
42         newNote.setNote(cursor.getString(1));
43         cursor.close();
44         return newNote;
45     }

```

```

46
47 public void delete(Note note) {
48     long id = note.getId();
49     database.delete(CustomSQLiteOpenHelper.TABLE_NOTES, CustomSQLiteOpenHelper.↵
        COLUMN_ID
50     + " = " + id, null);
51 }
52
53 public List<Note> getAll() {
54     List<Note> notes = new ArrayList<Note>();
55
56     Cursor cursor = database.query(CustomSQLiteOpenHelper.TABLE_NOTES,
57         columns, null, null, null, null, null);
58
59     cursor.moveToFirst();
60     while (!cursor.isAfterLast()) {
61         Note note = new Note();
62         note.setId(cursor.getLong(0));
63         note.setNote(cursor.getString(1));
64         notes.add(note);
65         cursor.moveToNext();
66     }
67     cursor.close();
68     return notes;
69 }
70 }

```

Código Java 5.7: NotesDao.java

Edite o arquivo *res/menu/main.xml* (se o arquivo não existir, você deve criá-lo). Deixe-o com o seguinte conteúdo:

```

1 <menu xmlns:android="http://schemas.android.com/apk/res/android">
2
3     <item android:id="@+id/add_note"
4         android:title="@string/add"
5         android:showAsAction="ifRoom" />
6
7 </menu>

```

Código XML 5.6: main.xml

A seguir edite o arquivo **MainActivity.java** para que ele fique com o seguinte conteúdo:

```

1 package br.com.k19.android.cap05_02;
2
3 import java.util.List;
4
5 import android.app.ListActivity;
6 import android.content.Intent;
7 import android.os.Bundle;
8 import android.view.Menu;
9 import android.view.MenuItem;
10 import android.widget.ArrayAdapter;
11
12 public class MainActivity extends ListActivity {
13
14     private NotesDao dao;
15
16     @Override
17     public void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.main);
20
21         dao = new NotesDao(this);
22         dao.open();
23     }

```

```

24
25 @Override
26 protected void onResume() {
27     dao.open();
28     super.onResume();
29
30     List<Note> notes = dao.getAll();
31
32     ArrayAdapter<Note> adapter = new ArrayAdapter<Note>(this,
33         android.R.layout.simple_list_item_1, notes);
34     setListAdapter(adapter);
35 }
36
37 @Override
38 protected void onPause() {
39     dao.close();
40     super.onPause();
41 }
42
43 @Override
44 public boolean onCreateOptionsMenu(Menu menu) {
45     getMenuInflater().inflate(R.menu.main, menu);
46     return true;
47 }
48
49 @Override
50 public boolean onOptionsItemSelected(MenuItem item) {
51     if (item.getItemId() == R.id.add_note) {
52         Intent intent = new Intent(this, AddNoteActivity.class);
53         startActivity(intent);
54     }
55     return super.onOptionsItemSelected(item);
56 }
57 }

```

Código Java 5.8: MainActivity.java

6 Crie um novo arquivo XML na pasta de layouts chamado **add_note.xml** com o conteúdo abaixo:

```

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:orientation="vertical" >
6
7     <EditText
8         android:id="@+id/note_text"
9         android:layout_width="match_parent"
10        android:layout_height="wrap_content"
11        android:inputType="textMultiLine"
12        android:hint="@string/write_a_note" />
13
14    <Button
15        android:id="@+id/save_note_button"
16        android:layout_width="match_parent"
17        android:layout_height="wrap_content"
18        android:hint="@string/save" />
19
20 </LinearLayout>

```

Código XML 5.7: add_name.xml

Crie uma nova classe chamada **AddNoteActivity** que herda *Activity* e possui o conteúdo abaixo:

```

1 package br.com.k19.android.cap05_02;
2

```

```
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.view.View.OnClickListener;
7 import android.widget.Button;
8 import android.widget.EditText;
9
10 public class AddNoteActivity extends Activity {
11
12     private NotesDao dao;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.add_note);
18
19         dao = new NotesDao(this);
20         dao.open();
21
22         Button saveButton = (Button) findViewById(R.id.save_note_button);
23         final EditText noteText = (EditText) findViewById(R.id.note_text);
24
25         saveButton.setOnClickListener(new OnClickListener() {
26
27             @Override
28             public void onClick(View v) {
29                 String note = noteText.getText().toString();
30                 dao.create(note);
31                 finish();
32             }
33         });
34     }
35
36     @Override
37     protected void onResume() {
38         dao.open();
39         super.onResume();
40     }
41
42     @Override
43     protected void onPause() {
44         dao.close();
45         super.onPause();
46     }
47 }
```

Código Java 5.9: Main.java

Se lembre de adicionar os nomes das *activities* no **AndroidManifest.xml**. Após isso, rode a aplicação e veja o resultado. Feche a aplicação e abra novamente, para ver se o nome está salva.

É muito comum um aplicativo fazer requisições HTTP para fazer consultas a *webservices*. Dessa forma, seu aplicativo pode integrar até diferentes serviços em uma única interface.



HTTP

Para fazer requisições, a API do Android oferece duas alternativas. A primeira é utilizando a classe **DefaultHttpClient**, do projeto Apache. Também existe a classe **AndroidHttpClient** que é um subtipo do **DefaultHttpClient** já configurado para valores otimizados no Android. Hoje em dia não é mais recomendado utilizar estas classes, porque a equipe do Google não dá manutenção a essa implementação.

O método recomendado é utilizar a classe **URLConnection**, que é desenvolvido e suportado pelo Google. Veja um exemplo abaixo de como utilizá-la:

```
1 URL url = new URL("http://www.android.com/");
2 HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
3 try {
4     InputStream in = new BufferedInputStream(urlConnection.getInputStream());
5     // lê os dados do InputStream
6 } finally {
7     urlConnection.disconnect();
8 }
```

Código Java 6.1: Exemplo.java



JSON

JSON ganhou muita força nos últimos anos como o formato mais utilizado no retorno de *web-services*, devido a sua simplicidade em comparação com XML. O Android possui bibliotecas padrão para lidar com JSON. Existem duas classes que são utilizadas com este propósito, **JSONObject** e **JSONArray**. A primeira serve para lidar com um objeto em JSON, enquanto a segunda é usada em arrays de objetos JSON. Veja abaixo um exemplo de uso:

```
1 JSONObject json = new JSONObject(jsonString);
2 try {
3     String campo1 = json.getString("campoObrigatorio");
4     String campo2 = json.optString("campoOpcional", null);
5     JSONObject objeto = json.getJSONObject("objetoAninhado");
6 } catch (JSONException e) {
7     e.printStackTrace();
8 }
```

Código Java 6.2: Exemplo.java



Exercícios de Fixação

- 1 Crie um novo projeto Android. Use como nome para o projeto **HttpAndJson**. O nome do pacote deve ser **br.com.k19.android.cap06**, e o nome da *activity* deve ser **MainActivity**.
- 2 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:orientation="vertical"
6   android:padding="16dp"
7   android:background="#EEEEEE" >
8
9   <TextView
10      android:id="@+id/name_text"
11      android:layout_width="wrap_content"
12      android:layout_height="wrap_content"
13      android:textSize="20dp"
14      android:textColor="#064E83"
15      android:paddingBottom="8dp"
16      android:textStyle="bold" />
17
18   <TextView
19      android:id="@+id/address_text"
20      android:layout_width="wrap_content"
21      android:layout_height="wrap_content"
22      android:textColor="#6C6C6C" />
23
24   <TextView
25      android:id="@+id/city_text"
26      android:layout_width="wrap_content"
27      android:layout_height="wrap_content"
28      android:textColor="#6C6C6C" />
29
30   <TextView
31      android:id="@+id/phone_text"
32      android:layout_width="wrap_content"
33      android:layout_height="wrap_content"
34      android:textColor="#6C6C6C" />
35
36   <TextView
37      android:id="@+id/likes_text"
38      android:layout_width="wrap_content"
39      android:layout_height="wrap_content"
40      android:textColor="#6C6C6C" />
41
42 </LinearLayout>
```

Código XML 6.1: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```
1 <resources>
2
3   <string name="app_name">HttpAndJson</string>
4   <string name="hello_world">Hello world!</string>
5   <string name="menu_settings">Settings</string>
6   <string name="title_activity_main">HttpAndJson</string>
7   <string name="phone_label">Telefone: %1$s</string>
```

```

8   <string name="address_label">Endereço: %1$s</string>
9   <string name="city_label">Cidade: %1$s</string>
10  <string name="likes_label">Total de likes: %1$d</string>
11
12 </resources>

```

Código XML 6.2: strings.xml

Crie um novo arquivo java chamado **MainActivity.java** com o seguinte conteúdo:

```

1 package br.com.k19.android.cap06;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStream;
6 import java.io.InputStreamReader;
7 import java.net.HttpURLConnection;
8 import java.net.URL;
9
10 import org.json.JSONException;
11 import org.json.JSONObject;
12
13 import android.app.Activity;
14 import android.os.Bundle;
15 import android.os.StrictMode;
16 import android.view.Menu;
17 import android.widget.TextView;
18
19 public class MainActivity extends Activity {
20
21     @Override
22     public void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.main);
25
26         TextView nameText = (TextView) findViewById(R.id.name_text);
27         TextView phoneText = (TextView) findViewById(R.id.phone_text);
28         TextView addressText = (TextView) findViewById(R.id.address_text);
29         TextView cityText = (TextView) findViewById(R.id.city_text);
30         TextView likesText = (TextView) findViewById(R.id.likes_text);
31
32         StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll()
33             .build();
34         StrictMode.setThreadPolicy(policy);
35
36         String response = makeRequest("http://graph.facebook.com/k19treinamentos");
37
38         try {
39             JSONObject json = new JSONObject(response);
40             String name = json.getString("name");
41             String phone = json.getString("phone");
42             int likes = json.getInt("likes");
43             String address = json.getJSONObject("location").getString("street");
44             String city = json.getJSONObject("location").getString("city");
45
46             nameText.setText(name);
47             phoneText.setText(getString(R.string.phone_label, phone));
48             addressText.setText(getString(R.string.address_label, address));
49             cityText.setText(getString(R.string.city_label, city));
50             likesText.setText(getString(R.string.likes_label, likes));
51
52         } catch (JSONException e) {
53             e.printStackTrace();
54         }
55
56         private String makeRequest(String urlAddress) {
57             HttpURLConnection con = null;
58             URL url = null;

```

```

59     String response = null;
60     try {
61         url = new URL(urlAddress);
62         con = (HttpURLConnection) url.openConnection();
63         response = readStream(con.getInputStream());
64     } catch (Exception e) {
65         e.printStackTrace();
66     } finally {
67         con.disconnect();
68     }
69     return response;
70 }
71
72
73 private String readStream(InputStream in) {
74     BufferedReader reader = null;
75     StringBuilder builder = new StringBuilder();
76     try {
77         reader = new BufferedReader(new InputStreamReader(in));
78         String line = null;
79         while ((line = reader.readLine()) != null) {
80             builder.append(line + "\n");
81         }
82     } catch (IOException e) {
83         e.printStackTrace();
84     } finally {
85         if (reader != null) {
86             try {
87                 reader.close();
88             } catch (IOException e) {
89                 e.printStackTrace();
90             }
91         }
92     }
93     return builder.toString();
94 }
95
96 }

```

Código Java 6.3: MainActivity.java

Após isso é necessário adicionar a permissão para acessar a Internet. Adicione o seguinte trecho de código ao arquivo **AndroidManifest.xml**.

```

1 <uses-permission android:name="android.permission.INTERNET" />

```

Código XML 6.3: AndroidManifest.xml

Após isso, rode a aplicação e veja o resultado.

3 Na pasta **res/values** crie um arquivo chamado **colors.xml** com o conteúdo abaixo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3
4     <color name="light_gray">#EEEEEE</color>
5     <color name="blue">#064E83</color>
6     <color name="gray">#6C6C6C</color>
7
8 </resources>

```

Código XML 6.4: colors.xml

Na pasta **res/values** crie um arquivo chamado **dimens.xml** com o conteúdo abaixo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3
4     <dimen name="padding_small">8dp</dimen>
5     <dimen name="padding_medium">8dp</dimen>
6     <dimen name="padding_large">16dp</dimen>
7     <dimen name="title_size">20dp</dimen>
8
9 </resources>

```

Código XML 6.5: colors.xml

Edite novamente o arquivo **main.xml**, deixando igual ao exemplo abaixo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical"
7     android:padding="@dimen/padding_large"
8     android:background="@color/light_gray" >
9
10    <TextView
11        android:id="@+id/name_text"
12        android:layout_width="wrap_content"
13        android:layout_height="wrap_content"
14        android:textSize="@dimen/title_size"
15        android:textColor="@color/blue"
16        android:paddingBottom="@dimen/padding_medium"
17        android:textStyle="bold" />
18
19    <TextView
20        android:id="@+id/address_text"
21        android:layout_width="wrap_content"
22        android:layout_height="wrap_content"
23        android:textColor="@color/gray" />
24
25    <TextView
26        android:id="@+id/city_text"
27        android:layout_width="wrap_content"
28        android:layout_height="wrap_content"
29        android:textColor="@color/gray" />
30
31    <TextView
32        android:id="@+id/phone_text"
33        android:layout_width="wrap_content"
34        android:layout_height="wrap_content"
35        android:textColor="@color/gray" />
36
37    <TextView
38        android:id="@+id/likes_text"
39        android:layout_width="wrap_content"
40        android:layout_height="wrap_content"
41        android:textColor="@color/gray" />
42
43 </LinearLayout>

```

Código XML 6.6: main.xml

Rode novamente a aplicação e veja se está igual ao que era antes.



THREADS E ASYNCTASKS

No Android, existe uma *thread* principal que é responsável por desenhar a tela e lidar com os eventos de toque na tela. Esta *thread* é conhecida como *UI thread* (*User Interface Thread*), ou também como *main thread*. Se o desenvolvedor não utilizar nenhum tipo de concorrência, todo o código que escrever irá rodar nesta *thread* principal. Isso se torna um problema para tarefas que levam muito tempo a serem executadas, pois enquanto a tarefa está sendo executada, a interface para de responder a eventos, como toques feito pelo usuário.

Se houver qualquer processamento que ocupe a *UI thread* por mais de 5 segundos, a aplicação irá receber automaticamente um ANR (*Application not responding*), e o sistema irá fechar a aplicação. Por isso, qualquer processamento mais lento deve ser feito em outras *threads* para não ocupar a *UI thread*.



Threads e Handlers

No Android é suportado o mesmo tipo de concorrência dos demais aplicativos Java. Podemos utilizar *threads*, que executam objetos do tipo *Runnable*. O único porém, é que não podemos alterar nada relativo a UI dentro destas *threads* que rodam em *background*. Apenas a *UI thread* é que pode alterar a UI. Para contornar esse problema podemos utilizar *Handlers*. Um **Handler** é um objeto que possui o método **post(Runnable)**. O *Runnable* que é passado ao método *post* é executado posteriormente dentro da *main thread* e por isso pode realizar alterações na interface da aplicação.

Outra alternativa que não envolve criar um **Handler** é utilizar o método **runOnUiThread(Runnable)**, que pertence a *Activity*. O *Runnable* que é passado a este método também é executado dentro da *main thread*.



Exercícios de Fixação

- 1 Crie um novo projeto Android. Use como nome para o projeto **Threads**. O nome do pacote deve ser **br.com.k19.android.cap07**, e o nome da *activity* deve ser **MainActivity**.
- 2 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent" >
5
6     <ProgressBar
7         android:id="@+id/progress_bar"
8         style="?android:attr/progressBarStyleHorizontal"
```

```

9      android:layout_width="match_parent"
10     android:layout_height="wrap_content"
11     android:layout_centerInParent="true"
12     android:indeterminate="false"
13     android:max="10"
14     android:padding="8dp" >
15 </ProgressBar>
16
17 <Button
18     android:id="@+id/start_button"
19     android:layout_width="wrap_content"
20     android:layout_height="wrap_content"
21     android:layout_below="@id/progress_bar"
22     android:layout_centerHorizontal="true"
23     android:text="@string/start" >
24 </Button>
25
26 </RelativeLayout>

```

Código XML 7.1: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <resources>
2
3     <string name="app_name">Threads</string>
4     <string name="menu_settings">Settings</string>
5     <string name="title_activity_main">MainActivity</string>
6     <string name="start">Iniciar</string>
7
8 </resources>

```

Código XML 7.2: strings.xml

Edite o arquivo **MainActivity.java** com o seguinte conteúdo:

```

1 package br.com.k19.cap07;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.os.Handler;
6 import android.view.View;
7 import android.view.View.OnClickListener;
8 import android.widget.Button;
9 import android.widget.ProgressBar;
10
11 public class MainActivity extends Activity {
12     private Handler handler;
13     private ProgressBar progress;
14     private Button startButton;
15
16     @Override
17     public void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.main);
20         progress = (ProgressBar) findViewById(R.id.progress_bar);
21         startButton = (Button) findViewById(R.id.start_button);
22         handler = new Handler();
23
24         startButton.setOnClickListener(new OnClickListener() {
25
26             @Override
27             public void onClick(View v) {
28                 Runnable runnable = new Runnable() {
29                     @Override
30                     public void run() {
31                         for (int i = 1; i <= 10; i++) {

```



```

32         final int value = i;
33         try {
34             Thread.sleep(1000);
35         } catch (InterruptedException e) {
36             e.printStackTrace();
37         }
38         handler.post(new Runnable() {
39             @Override
40             public void run() {
41                 progress.setProgress(value);
42             }
43         });
44     }
45 }
46 };
47
48 new Thread(runnable).start();
49 }
50 });
51 }
52 }

```

Código Java 7.1: MainActivity.java

Após isso, rode a aplicação e veja o resultado. Clique no botão para iniciar o progresso.

3 Edite novamente o arquivo **MainActivity.java** e deixe-o igual ao exemplo abaixo:

```

1 package br.com.k19.cap07;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.view.View.OnClickListener;
7 import android.widget.Button;
8 import android.widget.ProgressBar;
9
10 public class MainActivity extends Activity {
11     private ProgressBar progress;
12     private Button startButton;
13
14     @Override
15     public void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.main);
18         progress = (ProgressBar) findViewById(R.id.progress_bar);
19         startButton = (Button) findViewById(R.id.start_button);
20
21         startButton.setOnClickListener(new OnClickListener() {
22
23             @Override
24             public void onClick(View v) {
25                 Runnable runnable = new Runnable() {
26                     @Override
27                     public void run() {
28                         for (int i = 1; i <= 10; i++) {
29                             final int value = i;
30                             try {
31                                 Thread.sleep(1000);
32                             } catch (InterruptedException e) {
33                                 e.printStackTrace();
34                             }
35                             runOnUiThread(new Runnable() {
36                                 @Override
37                                 public void run() {
38                                     progress.setProgress(value);
39                                 }

```

```
40         });
41     }
42 }
43 };
44
45     new Thread(runnable).start();
46 }
47 });
48 }
49 }
```

Código Java 7.2: MainActivity.java

Rode novamente a aplicação e veja se funciona como o esperado.



AsyncTasks

Outra alternativa para utilizar concorrência no Android é utilizar *AsyncTasks*. Um *AsyncTask* é um objeto que encapsula em uma interface simples o uso de *threads*. Uma *AsyncTask* deve implementar obrigatoriamente o método **doInBackground()**, que exatamente a tarefa que está sendo executada em background. Caso seja necessário alguma atualização na interface, é só sobrescrever o método **onPostExecute()**. Tudo que estiver dentro deste método é executado na UI *thread*. Outro método interessante que pode ser sobrescrito é o método **onPreExecute()** que é executado antes do **doInBackground()** e que também é executado na UI *thread*.



Exercícios de Fixação

- 4 Crie um novo projeto Android. Use como nome para o projeto **AsyncTask**. O nome do pacote deve ser **br.com.k19.android.cap07_02**, e o nome da *activity* deve ser **MainActivity**.
- 5 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```
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:orientation="vertical" >
6
7     <Button
8         android:id="@+id/start_button"
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:onClick="downloadPicture"
12        android:text="@string/start_image_download" >
13    </Button>
14
15    <ImageView
16        android:id="@+id/image_view"
17        android:layout_width="match_parent"
18        android:layout_height="match_parent" >
19    </ImageView>
20
21 </LinearLayout>
```

Código XML 7.3: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <resources>
2
3   <string name="app_name">AsyncTask</string>
4   <string name="title_activity_main">MainActivity</string>
5   <string name="start_image_download">Iniciar download da imagem</string>
6   <string name="download">Download</string>
7   <string name="downloading">downloading</string>
8
9 </resources>

```

Código XML 7.4: strings.xml

Edite o arquivo **MainActivity.java** com o seguinte conteúdo:

```

1 package br.com.k19.android.cap07_02;
2
3 import java.io.IOException;
4 import java.io.InputStream;
5 import java.net.HttpURLConnection;
6 import java.net.MalformedURLException;
7 import java.net.URL;
8
9 import android.app.Activity;
10 import android.app.ProgressDialog;
11 import android.graphics.Bitmap;
12 import android.graphics.BitmapFactory;
13 import android.os.AsyncTask;
14 import android.os.Bundle;
15 import android.view.View;
16 import android.view.View.OnClickListener;
17 import android.widget.Button;
18 import android.widget.ImageView;
19
20 public class MainActivity extends Activity {
21
22     private ProgressDialog dialog;
23     private Button startButton;
24     private ImageView imageView;
25     private DownloadImageTask task;
26
27     @Override
28     public void onCreate(Bundle savedInstanceState) {
29         super.onCreate(savedInstanceState);
30         setContentView(R.layout.main);
31
32         imageView = (ImageView) findViewById(R.id.image_view);
33         startButton = (Button) findViewById(R.id.start_button);
34
35         startButton.setOnClickListener(new OnClickListener() {
36
37             @Override
38             public void onClick(View v) {
39                 dialog = ProgressDialog.show(MainActivity.this,
40                     getString(R.string.download),
41                     getString(R.string.downloading));
42                 task = new DownloadImageTask();
43                 task.execute("http://k19.com.br/css/img/main-header-logo.png");
44             }
45         });
46
47     }

```

```

48
49 @Override
50 protected void onDestroy() {
51     if (dialog != null && dialog.isShowing()) {
52         dialog.dismiss();
53         dialog = null;
54     }
55     if (task != null) {
56         task.cancel(true);
57     }
58     super.onDestroy();
59 }
60
61 private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
62
63     @Override
64     protected Bitmap doInBackground(String... params) {
65         try {
66             return downloadBitmap(params[0]);
67         } catch (IOException e) {
68             e.printStackTrace();
69         }
70         return null;
71     }
72
73     @Override
74     protected void onPreExecute() {
75         super.onPreExecute();
76         dialog.show();
77     }
78
79     @Override
80     protected void onPostExecute(Bitmap result) {
81         super.onPostExecute(result);
82         dialog.dismiss();
83         if (result != null) {
84             imageView.setImageBitmap(result);
85         }
86     }
87
88     private Bitmap downloadBitmap(String url) throws IOException {
89         URL imageUrl = null;
90         try {
91             imageUrl = new URL(url);
92         } catch (MalformedURLException e) {
93             e.printStackTrace();
94             return null;
95         }
96         Bitmap bitmapImage = null;
97         try {
98             HttpURLConnection conn = (HttpURLConnection) imageUrl
99                 .openConnection();
100             conn.setDoInput(true);
101             conn.connect();
102             InputStream is = conn.getInputStream();
103
104             bitmapImage = BitmapFactory.decodeStream(is);
105         } catch (IOException e) {
106             e.printStackTrace();
107         }
108         return bitmapImage;
109     }
110 }
111 }
112 }

```

Código Java 7.3: MainActivity.java

Adicione a permissão para internet no **AndroidManifest.xml**. Basta adicionar a seguinte linha:

```
1 <uses-permission android:name="android.permission.INTERNET" />
```

Código XML 7.5: AndroidManifest.xml

Após isso, rode a aplicação e veja o resultado.





Serviços

Serviços são aplicações que executam, em geral, processos longos em background desprovidos de interface. Uma aplicação, por exemplo, pode requisitar a um serviço para fazer um download ou mesmo executar uma música enquanto o usuário interage com a interface ou mesmo sai da aplicação host. A aplicação e o Serviço podem ainda se comunicar entre si.

Por padrão, um serviço sempre é executado na Thread principal da aplicação host. Porém, isto pode ser configurado para que o serviço inicie outras threads quando é chamado evitando assim que a interface trave durante uma execução que consuma muito processamento.

Manifest

Para criar um serviço é preciso declarar o nome da classe no Manifest.

```
1 <manifest ... >
2   ...
3   <application ... >
4     <service android:name=".ExampleService" />
5     ...
6   </application>
7 </manifest>
```

Código XML 8.1: *android:name* é o único atributo obrigatório

O serviço pode ser utilizado por qualquer aplicação através de um Intent. Se o serviço a ser implementado for apenas útil para a aplicação que o contém, então é preciso explicitar que o serviço é *privado* no Manifest.

```
1 <manifest ... >
2   ...
3   <application ... >
4     <service
5       android:name=".ExampleService"
6       android:exported="false"
7     />
8     ...
9   </application>
10 </manifest>
```

Código XML 8.2: Tornando o serviço local

Classe Service

Para criar um serviço é preciso implementar uma extensão da classe Service e sobrescrever alguns métodos de callback.

onStartCommand() - Método que inicia um serviço indefinidamente. O serviço apenas termina quando o método `stopSelf()` é executado a partir do próprio serviço ou quando o método `stopService()` é executado a partir de outra aplicação.

onBind() - Método que é chamado pelo sistema para associar o serviço a uma aplicação. Ele deve prover uma interface de comunicação entre ambos. Este método deve ser implementado obrigatoriamente, logo, se o serviço não for desenhado para suportar Bind então o método `onBind` deve devolver `null`.

onCreate() - Método chamado pelo sistema no momento da criação do serviço e pode ser utilizado para realizar pré configurações.

onDestroy() - Método chamado pelo sistema quando o serviço for destruído e pode ser utilizado para liberar recursos utilizados.

Abaixo temos uma implementação simples de um serviço.

```
1 public class ExampleService extends Service {
2
3     @Override
4     public void onCreate() {
5         // metodo executado no momento em que o servico e criado
6     }
7
8     @Override
9     public int onStartCommand(Intent intent, int flags, int startId) {
10        // execucao do servico
11        return START_STICKY;
12    }
13
14    @Override
15    public IBinder onBind(Intent intent) {
16        // sem suporte a Binding
17        return null;
18    }
19
20    @Override
21    public void onDestroy() {
22        // metodo executado no momento em que o servico e destruido
23    }
24 }
```

Código Java 8.1: Extendendo a classe Service

Observe que o método `onStartCommand()` devolve um inteiro. Este valor indica como o sistema deve continuar o serviço caso o sistema o mate. Existem 3 valores possíveis:

START_NOT_STICKY - Não reinicia o serviço a menos que hajam Intents a serem entregues;

START_STICKY - Reinicia o serviço mas não continua a partir do Intent que estava em execução mas apenas para os que estavam pendentes;

START_REDELIVER_INTENT - Reinicia o serviço retomando a partir do Intent que estava em execução.

Lembre-se de que a thread que executa o serviço é a thread principal da aplicação host. Caso o serviço ocupe muito processamento é preciso que o serviço utilize uma nova thread evitando assim

travamentos na interface. No exemplo abaixo, a classe do serviço foi modificada para executar sua tarefa em uma thread separada.

```

1 public class ExampleService extends Service {
2
3     private Looper mServiceLooper;
4     private ServiceHandler mServiceHandler;
5
6     // Handler que executa de fato a tarefa do serviço em uma thread separada
7     private final class ServiceHandler extends Handler {
8         public ServiceHandler(Looper looper) {
9             super(looper);
10        }
11
12        @Override
13        public void handleMessage(Message msg) {
14            // Implementação da tarefa do serviço
15            ...
16
17            // Parando explicitamente o serviço
18            stopSelf(msg.arg1);
19        }
20    }
21
22    @Override
23    public void onCreate() {
24        // Criando a thread responsável pela execução da tarefa
25        HandlerThread thread = new HandlerThread("ServiceStartArguments",
26            Process.THREAD_PRIORITY_BACKGROUND);
27        thread.start();
28
29        // Obtendo o Looper da thread e passando como parametro para o Handler
30        mServiceLooper = thread.getLooper();
31        mServiceHandler = new ServiceHandler(mServiceLooper);
32    }
33
34    @Override
35    public int onStartCommand(Intent intent, int flags, int startId) {
36        Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show();
37
38        // Para cada chamada ao serviço enfileiramos uma tarefa no Handler
39        Message msg = mServiceHandler.obtainMessage();
40        msg.arg1 = startId;
41        mServiceHandler.sendMessage(msg);
42
43        // Se o serviço morrer a partir deste ponto, reiniciar
44        return START_STICKY;
45    }
46
47    @Override
48    public IBinder onBind(Intent intent) {
49        // sem suporte a Binding
50        return null;
51    }
52
53    @Override
54    public void onDestroy() {
55        Toast.makeText(this, "service done", Toast.LENGTH_SHORT).show();
56    }
57 }

```

Código Java 8.2: Executando o serviço em uma thread a parte

Um detalhe importante é que o método `onStartCommand()` pode ser chamado pelo sistema diversas vezes, uma para cada requisição da aplicação. Cada uma delas vem acompanhada de um id (`startId`). Se ao final de uma execução o método `stopSelf()` for chamado enquanto uma outra requisição está sendo executada, o serviço terminaria sem completar a segunda execução. Para evitar isto,

o método `stopSelf()` pode receber um inteiro que representa o id da requisição que terminou. Se o id for igual ao da última requisição o serviço termina, caso contrário ele continua a executar até que as requisições acabem.

É muito comum implementar serviços que utilizem sua própria thread para executar as tarefas requisitadas, desta forma, o framework fornece uma extensão da classe `Service` que simplifica a criação de serviços como o mostrado no último exemplo. O código abaixo implementa um serviço que se comporta como o exemplo anterior utilizando a classe `IntentService`.

```
1 public class ExampleService extends IntentService {
2
3     /**
4      * O construtor é obrigatório e deve chamar o construtor da super classe
5      * passando o nome da Thread worker
6      */
7     public ExampleService() {
8         super("ExampleService");
9     }
10
11     /**
12      * Este método é chamado pela IntentService a partir de um worker Thread e recebe o
13      * Intent que iniciou o serviço. Quando o método termina o IntentService para o
14      * serviço.
15      */
16     @Override
17     protected void onHandleIntent(Intent intent) {
18         // Implementação da tarefa do serviço
19     }
20 }
```

Código Java 8.3: Utilizando a classe `IntentService`

Quando utilizamos a classe `IntentService`, não é necessário se preocupar em parar o serviço.

Iniciando um serviço

Para dar início a um serviço basta criar um intent e passá-lo como parâmetro ao método `startService()` como no exemplo a seguir:

```
1 Intent intent = new Intent(this, ExampleService.class);
2 startService(intent);
```

Código Java 8.4: Iniciando um serviço



Exercícios de Fixação

- 1 Crie um novo projeto Android. Use como nome para o projeto **Services**. O nome do pacote deve ser **br.com.k19.android.cap08**, e o nome da *activity* deve ser **MainActivity**.
- 2 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```
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:orientation="vertical" >
6
7     <Button
8         android:id="@+id/start_button"
9         android:layout_width="match_parent"
10        android:layout_height="wrap_content"
11        android:text="@string/start_downloads"
12        android:layout_gravity="center" />
13
14 </LinearLayout>

```

Código XML 8.3: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <resources>
2
3     <string name="app_name">Services</string>
4     <string name="hello_world">Hello world!</string>
5     <string name="menu_settings">Settings</string>
6     <string name="title_activity_main">MainActivity</string>
7     <string name="start_downloads">Iniciar downloads</string>
8     <string name="download_error">Erro ao fazer download.</string>
9     <string name="download_success">Download feito com sucesso %1$s.</string>
10
11 </resources>

```

Código XML 8.4: strings.xml

Crie um arquivo chamado **DownloadService.java** com o seguinte conteúdo:

```

1 package br.com.k19.android.cap08;
2
3 import java.io.File;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.InputStream;
7 import java.io.InputStreamReader;
8 import java.net.URL;
9
10 import android.app.Activity;
11 import android.app.IntentService;
12 import android.content.Intent;
13 import android.net.Uri;
14 import android.os.Bundle;
15 import android.os.Environment;
16 import android.os.Message;
17 import android.os.Messenger;
18 import android.util.Log;
19
20 public class DownloadService extends IntentService {
21
22     private int result = Activity.RESULT_CANCELED;
23
24     public DownloadService() {
25         super("DownloadService");
26     }
27
28     @Override
29     protected void onHandleIntent(Intent intent) {
30         Uri data = intent.getData();
31         String urlPath = intent.getStringExtra("urlPath");
32         String fileName = data.getPath();
33         File output = new File(Environment.getExternalStorageDirectory(),
34             fileName);
35         if (output.exists()) {
36             output.delete();

```

```

37     }
38
39     InputStream stream = null;
40     FileOutputStream fos = null;
41     try {
42
43         URL url = new URL(urlPath);
44         stream = url.openConnection().getInputStream();
45         InputStreamReader reader = new InputStreamReader(stream);
46         fos = new FileOutputStream(output.getPath());
47         int next = -1;
48         while ((next = reader.read()) != -1) {
49             fos.write(next);
50         }
51         result = Activity.RESULT_OK;
52
53     } catch (Exception e) {
54         e.printStackTrace();
55     } finally {
56         if (stream != null) {
57             try {
58                 stream.close();
59             } catch (IOException e) {
60                 e.printStackTrace();
61             }
62         }
63         if (fos != null) {
64             try {
65                 fos.close();
66             } catch (IOException e) {
67                 e.printStackTrace();
68             }
69         }
70     }
71
72     Bundle extras = intent.getExtras();
73     if (extras != null) {
74         Messenger messenger = (Messenger) extras.get("messenger");
75         Message msg = Message.obtain();
76         msg.arg1 = result;
77         msg.obj = output.getAbsolutePath();
78         try {
79             messenger.send(msg);
80         } catch (android.os.RemoteException e1) {
81             Log.e("DownloadService", "Erro ao enviar mensagem", e1);
82         }
83     }
84 }
85 }
86 }

```

Código Java 8.5: DownloadService.java

Edite o arquivo **MainActivity.java** com o seguinte conteúdo:

```

1 package br.com.k19.android.cap08;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.net.Uri;
6 import android.os.Bundle;
7 import android.os.Handler;
8 import android.os.Message;
9 import android.os.Messenger;
10 import android.view.View;
11 import android.view.View.OnClickListener;
12 import android.widget.Button;
13 import android.widget.Toast;
14

```

```

15 public class MainActivity extends Activity {
16
17     private Handler handler = new Handler() {
18         public void handleMessage(Message message) {
19             Object path = message.obj;
20             if (message.arg1 == RESULT_OK && path != null) {
21                 Toast.makeText(MainActivity.this,
22                     getString(R.string.download_success, path.toString()),
23                     Toast.LENGTH_LONG).show();
24             } else {
25                 Toast.makeText(MainActivity.this,
26                     getString(R.string.download_error), Toast.LENGTH_LONG)
27                     .show();
28             }
29         }
30     };
31 };
32
33 @Override
34 public void onCreate(Bundle savedInstanceState) {
35     super.onCreate(savedInstanceState);
36     setContentView(R.layout.main);
37
38     Button startButton = (Button) findViewById(R.id.start_button);
39     startButton.setOnClickListener(new OnClickListener() {
40
41         @Override
42         public void onClick(View v) {
43             Intent intent = new Intent(MainActivity.this,
44                 DownloadService.class);
45             Messenger messenger = new Messenger(handler);
46             intent.putExtra("messenger", messenger);
47             intent.setData(Uri.parse("cursos.html"));
48             intent.putExtra("urlPath", "http://k19.com.br/cursos");
49             startService(intent);
50         }
51     });
52 }
53 }

```

Código Java 8.6: MainActivity.java

Edite o arquivo **AndroidManifest.xml**, igual ao exemplo abaixo:

```

1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2     package="br.com.k19.android.cap08"
3     android:versionCode="1"
4     android:versionName="1.0" >
5
6     <uses-sdk
7         android:minSdkVersion="8"
8         android:targetSdkVersion="15" />
9     <uses-permission android:name="android.permission.INTERNET"/>
10    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
11
12    <application
13        android:icon="@drawable/ic_launcher"
14        android:label="@string/app_name"
15        android:theme="@style/AppTheme" >
16        <activity
17            android:name=".MainActivity"
18            android:label="@string/app_name" >
19            <intent-filter>
20                <action android:name="android.intent.action.MAIN" />
21                <category android:name="android.intent.category.LAUNCHER" />
22            </intent-filter>
23        </activity>
24        <service android:name=".DownloadService"></service>
25    </application>

```

```

26
27 </manifest>

```

Código XML 8.5: AndroidManifest.xml

Após isso, rode a aplicação e veja o resultado. Clique no botão para iniciar o download.



Broadcast Receivers

Um *Broadcast Receiver* é um objeto que herda **BroadcastReceiver**, e que implementa o método **onReceive()**. Eles devem ser registrados no *AndroidManifest.xml*. Um *receiver* em geral deve ser usado para receber alguma notificação do sistema, e executar uma tarefa dependendo do tipo de notificação recebido. Por exemplo, se sua aplicação recebeu uma notificação de que a bateria está baixa, ele pode forçadamente parar serviços ou tarefas que estejam consumindo muito processamento, ou até fechar o aplicativo.



Exercícios de Fixação

- 3 Crie um novo projeto Android. Use como nome para o projeto **Reveiver**. O nome do pacote deve ser **br.com.k19.android.cap08_02**, e o nome da *activity* deve ser **MainActivity**.
- 4 Crie um arquivo chamado **PhoneReceiver.java** com o seguinte conteúdo:

```

1 package br.com.k19.android.cap08_02;
2
3 import android.content.BroadcastReceiver;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.os.Bundle;
7 import android.telephony.TelephonyManager;
8 import android.util.Log;
9
10 public class PhoneReceiver extends BroadcastReceiver {
11     private static final String TAG = "PhoneReceiver";
12
13     @Override
14     public void onReceive(Context context, Intent intent) {
15         Bundle extras = intent.getExtras();
16         if (extras != null) {
17             String state = extras.getString(TelephonyManager.EXTRA_STATE);
18             Log.w(TAG, state);
19             if (state.equals(TelephonyManager.EXTRA_STATE_RINGING)) {
20                 String phoneNumber = extras
21                     .getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
22                 Log.w(TAG, phoneNumber);
23             }
24         }
25     }
26 }

```

Código Java 8.7: PhoneReceiver.java

Edite o arquivo **AndroidManifest.xml**, e deixe-o igual ao exemplo abaixo:

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2   package="br.com.k19.android.cap08_02"
3   android:versionCode="1"
4   android:versionName="1.0" >
5
6   <uses-sdk
7     android:minSdkVersion="8"
8     android:targetSdkVersion="15" />
9
10  <uses-permission android:name="android.permission.READ_PHONE_STATE" />
11
12  <application
13    android:icon="@drawable/ic_launcher"
14    android:label="@string/app_name"
15    android:theme="@style/AppTheme" >
16    <activity
17      android:name=".MainActivity"
18      android:label="@string/title_activity_main" >
19      <intent-filter>
20        <action android:name="android.intent.action.MAIN" />
21
22        <category android:name="android.intent.category.LAUNCHER" />
23      </intent-filter>
24    </activity>
25
26    <receiver android:name=".PhoneReceiver" >
27      <intent-filter>
28        <action android:name="android.intent.action.PHONE_STATE" >
29        </action>
30      </intent-filter>
31    </receiver>
32  </application>
33
34 </manifest>
```

Código XML 8.6: AndroidManifest.xml

Após isso, rode a aplicação. Para testar, você deve simular uma ligação no emulador. Para fazer isso, mude a perspectiva para DDMS, e encontre a aba **tEmulator Control**. Nesta aba, basta preencher um número e pressionar o botão **Call**. Após isso, veja no LogCat se deu certo.





Dialogs

Quando você precisa mostrar avisos ao usuário, o mais indicado é utilizar *dialogs*. Para criar um *dialog* no Android, o indicado é herdar a classe *DialogFragment*. Ao criar uma subclasse, você deve sobrescrever os métodos **onCreate()** e **onCreateView()** para criar o seu diálogo customizado. Outra opção caso você queira um diálogo simples, como um *AlertDialog* por exemplo, você pode sobrescrever o método **onCreateDialog()**.



Exercícios de Fixação

- 1 Crie um novo projeto Android. Use como nome para o projeto **Dialogs**. O nome do pacote deve ser **br.com.k19.android.cap09**, e o nome da *activity* deve ser **MainActivity**.
- 2 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```

1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:orientation="vertical" >
6
7   <Button
8     android:id="@+id/show_progress_dialog_button"
9     android:layout_width="match_parent"
10    android:layout_height="wrap_content"
11    android:padding="5dp"
12    android:layout_margin="8dp"
13    android:text="@string/show_progress_dialog" />
14
15   <Button
16     android:id="@+id/show_alert_dialog_button"
17     android:layout_width="match_parent"
18     android:layout_height="wrap_content"
19     android:padding="5dp"
20     android:layout_margin="8dp"
21     android:text="@string/show_alert_dialog" />
22
23   <Button
24     android:id="@+id/show_custom_dialog_button"
25     android:layout_width="match_parent"
26     android:layout_height="wrap_content"
27     android:padding="5dp"
28     android:layout_margin="8dp"
29     android:text="@string/show_custom_dialog" />
30
31 </LinearLayout>

```

Código XML 9.1: main.xml

Copie o logo do site da K19 para a pasta *drawable-hdpi*. O arquivo deve-se chamar **k19_logo.png**.

Na pasta **res/layouts** crie um arquivo chamado **custom_dialog.xml**. Ele deve conter o seguinte conteúdo:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="wrap_content"
4     android:layout_height="wrap_content"
5     android:orientation="vertical" >
6
7     <ImageView
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"
10        android:src="@drawable/k19_logo"
11        android:layout_gravity="center_horizontal"
12        android:padding="8dp"
13        android:contentDescription="@string/logo" />
14
15 </LinearLayout>
```

Código XML 9.2: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```
1 <resources>
2
3     <string name="app_name">Notifications</string>
4     <string name="title_activity_main">MainActivity</string>
5     <string name="attention">Atenção</string>
6     <string name="which_button_gonna_press">Qual botão você irá apertar?</string>
7     <string name="yes">Sim</string>
8     <string name="pressed_yes">Pressionou sim</string>
9     <string name="no">Não</string>
10    <string name="pressed_no">Pressionou não</string>
11    <string name="wait">Aguarde...</string>
12    <string name="k19_training">K19 Treinamentos</string>
13    <string name="show_progress_dialog">Mostrar Progress Dialog</string>
14    <string name="show_alert_dialog">Mostrar Alert Dialog</string>
15    <string name="show_custom_dialog">Mostrar Custom Dialog</string>
16    <string name="logo">logo</string>
17
18 </resources>
```

Código XML 9.3: strings.xml

Edite o arquivo **MainActivity.java** com o seguinte conteúdo:

```
1 package br.com.k19.android.cap09;
2
3 import android.app.AlertDialog;
4 import android.app.Dialog;
5 import android.app.ProgressDialog;
6 import android.content.DialogInterface;
7 import android.os.Bundle;
8 import android.support.v4.app.DialogFragment;
9 import android.support.v4.app.FragmentActivity;
10 import android.view.LayoutInflater;
11 import android.view.View;
12 import android.view.View.OnClickListener;
13 import android.view.ViewGroup;
```

```

14 import android.widget.Button;
15 import android.widget.Toast;
16
17 public class MainActivity extends FragmentActivity {
18
19     @Override
20     public void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22         setContentView(R.layout.main);
23
24         Button progressButton = (Button) findViewById(R.id.show_progress_dialog_button);
25         Button alertButton = (Button) findViewById(R.id.show_alert_dialog_button);
26         Button customButton = (Button) findViewById(R.id.show_custom_dialog_button);
27
28         progressButton.setOnClickListener(new OnClickListener() {
29
30             @Override
31             public void onClick(View v) {
32                 DialogFragment dialog = ProgressDialogFragment.newInstance();
33                 dialog.show(getSupportFragmentManager(), "progress");
34             }
35         });
36
37         alertButton.setOnClickListener(new OnClickListener() {
38
39             @Override
40             public void onClick(View v) {
41                 DialogFragment dialog = AlertDialogFragment.newInstance();
42                 dialog.show(getSupportFragmentManager(), "alert");
43             }
44         });
45
46         customButton.setOnClickListener(new OnClickListener() {
47
48             @Override
49             public void onClick(View v) {
50                 DialogFragment dialog = CustomDialogFragment.newInstance();
51                 dialog.show(getSupportFragmentManager(), "custom");
52             }
53         });
54     }
55
56     public static class AlertDialogFragment extends DialogFragment {
57         public static AlertDialogFragment newInstance() {
58             AlertDialogFragment frag = new AlertDialogFragment();
59             return frag;
60         }
61
62         @Override
63         public Dialog onCreateDialog(Bundle savedInstanceState) {
64             AlertDialog dialog = new AlertDialog.Builder(getActivity())
65                 .create();
66             dialog.setTitle(getActivity().getString(R.string.attention));
67             dialog.setMessage(getActivity().getString(R.string.which_button_gonna_press));
68             dialog.setButton(DialogInterface.BUTTON_POSITIVE, getActivity().getString(R.string.yes),
69                 new DialogInterface.OnClickListener() {
70                     @Override
71                     public void onClick(DialogInterface dialog, int which) {
72                         Toast.makeText(getActivity(), R.string.pressed_yes,
73                             Toast.LENGTH_SHORT).show();
74                     }
75                 });
76             dialog.setButton(DialogInterface.BUTTON_NEGATIVE, getActivity().getString(R.string.no),
77                 new DialogInterface.OnClickListener() {
78                     @Override
79                     public void onClick(DialogInterface dialog, int which) {
80                         Toast.makeText(getActivity(), R.string.pressed_no,
81                             Toast.LENGTH_SHORT).show();

```

```
82         }
83     });
84     return dialog;
85 }
86 }
87
88 public static class ProgressDialogFragment extends DialogFragment {
89     public static ProgressDialogFragment newInstance() {
90         ProgressDialogFragment frag = new ProgressDialogFragment();
91         return frag;
92     }
93
94     @Override
95     public Dialog onCreateDialog(Bundle savedInstanceState) {
96         Dialog dialog = new ProgressDialog(getActivity());
97         dialog.setTitle(R.string.wait);
98         return dialog;
99     }
100 }
101
102 public static class CustomDialogFragment extends DialogFragment {
103
104     public static CustomDialogFragment newInstance() {
105         CustomDialogFragment frag = new CustomDialogFragment();
106         return frag;
107     }
108
109     @Override
110     public void onCreate(Bundle savedInstanceState) {
111         super.onCreate(savedInstanceState);
112     }
113
114     @Override
115     public View onCreateView(LayoutInflater inflater, ViewGroup container,
116         Bundle savedInstanceState) {
117         View v = inflater.inflate(R.layout.custom_dialog, container,
118             false);
119         getDialog().setTitle(R.string.k19_training);
120         return v;
121     }
122 }
123 }
```

Código Java 9.1: MainActivity.java

Após isso, rode a aplicação e veja o resultado. Clique nos botões para ver o que acontece.



Notifications

Outro estilo de notificação é o conhecido como *Status Bar Notifications*, que são aqueles alertas que aparecem na barra de status. Existem diferentes tipos de alertas que podem ser criados. Em todos os casos, você deve utilizar a classe *NotificationManager* para enviar as notificações para o sistema. Para construir uma notificação, é utilizado o **Notification.Builder()** que possui diferentes métodos que customizam o conteúdo e aparência da notificação.



Exercícios de Fixação

- 3 Crie um novo projeto Android. Use como nome para o projeto **Notifications**. O nome do pacote

deve ser **br.com.k19.android.cap09_02**, e o nome da *activity* deve ser **MainActivity**.

- 4 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent" >
5
6     <Button
7         android:id="@+id/create_notification_button"
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"
10        android:text="@string/create_notification"
11        android:layout_centerInParent="true" />
12
13 </RelativeLayout>

```

Código XML 9.4: main.xml

Na pasta **res/layouts** crie um arquivo chamado **notification.xml**. Ele deve conter o seguinte conteúdo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent" >
5
6     <TextView
7         android:layout_width="match_parent"
8         android:layout_height="wrap_content"
9         android:layout_centerInParent="true"
10        android:text="@string/notification_activity_description"
11        android:gravity="center_horizontal" />
12
13 </RelativeLayout>

```

Código XML 9.5: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <resources>
2
3     <string name="app_name">Notifications</string>
4     <string name="menu_settings">Settings</string>
5     <string name="title_activity_main">MainActivity</string>
6     <string name="new_notification">Nova notificação</string>
7     <string name="notification_content">Conteúdo descrevendo a notificação.</string>
8     <string name="create_notification">Criar notificação</string>
9     <string name="notification_activity_description">Esta tela foi aberta a partir da
10         notificação.</string>
11 </resources>

```

Código XML 9.6: strings.xml

Crie um arquivo chamado **NotificationActivity.java** com o seguinte conteúdo:

```

1 package br.com.k19.android.cap09_02;
2
3 import br.com.k19.android.cap09_02.R;
4
5 import android.app.Activity;

```

```
6 import android.os.Bundle;
7
8 public class NotificationActivity extends Activity {
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.notification);
13    }
14 }
```

Código Java 9.2: NotificationActivity.java

Edite o arquivo **MainActivity.java** com o seguinte conteúdo:

```
1 package br.com.k19.android.cap09_02;
2
3 import android.app.Activity;
4 import android.app.Notification;
5 import android.app.NotificationManager;
6 import android.app.PendingIntent;
7 import android.content.Intent;
8 import android.os.Bundle;
9 import android.view.View;
10 import android.view.View.OnClickListener;
11 import android.widget.Button;
12
13
14 public class MainActivity extends Activity {
15
16     @Override
17     public void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.main);
20
21         Button createNotification = (Button) findViewById(R.id.create_notification_button);
22         createNotification.setOnClickListener(new OnClickListener() {
23
24             @Override
25             public void onClick(View v) {
26                 Intent intent = new Intent(MainActivity.this, NotificationActivity.class);
27                 PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this, 0, intent, 0);
28
29                 Notification notification = new NotificationCompat.Builder(MainActivity.this)
30                     .setContentTitle(getString(R.string.new_notification))
31                     .setContentText(getString(R.string.notification_content))
32                     .setSmallIcon(R.drawable.ic_launcher)
33                     .setAutoCancel(true)
34                     .setContentIntent(pendingIntent)
35                     .build();
36
37                 NotificationManager notificationManager = (NotificationManager)
38                     getSystemService(NOTIFICATION_SERVICE);
39                 notificationManager.notify(0, notification);
40             }
41         });
42     }
43 }
```

Código Java 9.3: MainActivity.java

Após isso, rode a aplicação e veja o resultado. Clique no botão, e logo em seguida na notificação que deve aparecer.

Uma das vantagens de se desenvolver aplicações para dispositivos móveis é que podemos tirar proveito da geolocalização e criar aplicativos úteis para o usuário, que levam em consideração a sua localização.



Utilizando o GPS

Para utilizar o GPS, é necessário utilizar a classe *LocationManager*. Esta classe permite obter a posição do usuário, registrar *listeners* de localização, etc. Para obter uma posição, é necessário escolher um LOCATION PROVIDER. O *provider* define como a posição do usuário será obtida:

network - utiliza as antenas de rede e Wi-Fi para determinar a posição do usuário. É mais rápido em geral do que o GPS, e funciona melhor em ambientes fechados também.

gps - utiliza o sistema de GPS do aparelho para determinar a posição. A precisão nesse caso é melhor do que usando *network*.

Para utilizar o GPS, é necessário incluir a permissão ACCESS_FINE_LOCATION. Para usar apenas a rede é necessário incluir a permissão ACCESS_COARSE_LOCATION.

É possível usar um objeto do tipo **Criteria** para auxiliar que tipo de *provider* será utilizado. Com este objeto definimos parâmetros como precisão, velocidade de resposta, etc, e ele se encarrega de escolher o *provider* mais adequado.



Exercícios de Fixação

1 Crie um novo projeto Android. Use como nome para o projeto **LocationApi**. O nome do pacote deve ser **br.com.k19.android.cap10**, e o nome da *activity* deve ser **MainActivity**.

2

Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```
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:orientation="vertical" >
6
7     <LinearLayout
8         android:layout_width="match_parent"
9         android:layout_height="wrap_content"
10        android:layout_marginTop="40dp"
11        android:orientation="horizontal" >
12
13        <TextView
14            android:layout_width="wrap_content"
15            android:layout_height="wrap_content"
16            android:layout_marginLeft="10dp"
17            android:layout_marginRight="5dp"
18            android:text="@string/latitude_label"
19            android:textSize="20dp" >
20        </TextView>
21
22        <TextView
23            android:id="@+id/latitude_text"
24            android:layout_width="wrap_content"
25            android:layout_height="wrap_content"
26            android:text="@string/unknown"
27            android:textSize="20dp" >
28        </TextView>
29    </LinearLayout>
30
31    <LinearLayout
32        android:layout_width="match_parent"
33        android:layout_height="wrap_content" >
34
35        <TextView
36            android:layout_width="wrap_content"
37            android:layout_height="wrap_content"
38            android:layout_marginLeft="10dp"
39            android:layout_marginRight="5dp"
40            android:text="@string/longitude_label"
41            android:textSize="20dp" >
42        </TextView>
43
44        <TextView
45            android:id="@+id/longitude_text"
46            android:layout_width="wrap_content"
47            android:layout_height="wrap_content"
48            android:text="@string/unknown"
49            android:textSize="20dp" >
50        </TextView>
51    </LinearLayout>
52
53 </LinearLayout>

```

Código XML 10.1: main.xml

Edite o arquivo *AndroidManifest.xml*. Ele deve conter o seguinte conteúdo:

```

1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2     package="com.example.locationapi"
3     android:versionCode="1"
4     android:versionName="1.0" >
5
6     <uses-sdk
7         android:minSdkVersion="8"
8         android:targetSdkVersion="15" />
9
10    <uses-permission android:name="android.permission.INTERNET" />
11    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
12    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
13
14    <application
15        android:icon="@drawable/ic_launcher"

```



```

16     android:label="@string/app_name"
17     android:theme="@style/AppTheme" >
18     <activity
19         android:name=".MainActivity"
20         android:label="@string/app_name" >
21         <intent-filter>
22             <action android:name="android.intent.action.MAIN" />
23
24             <category android:name="android.intent.category.LAUNCHER" />
25         </intent-filter>
26     </activity>
27 </application>
28
29 </manifest>

```

Código XML 10.2: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <resources>
2
3     <string name="app_name">LocationApi</string>
4     <string name="menu_settings">Settings</string>
5     <string name="point_label">%.4f</string>
6     <string name="location_not_available">Local não disponível</string>
7     <string name="latitude_label">"Latitude: "</string>
8     <string name="longitude_label">"Longitude: "</string>
9     <string name="unknown">desconhecido</string>
10
11 </resources>

```

Código XML 10.3: strings.xml

Edite o arquivo **MainActivity.java** com o seguinte conteúdo:

```

1 package com.example.locationapi;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.location.Criteria;
6 import android.location.Location;
7 import android.location.LocationListener;
8 import android.location.LocationManager;
9 import android.os.Bundle;
10 import android.util.Log;
11 import android.widget.TextView;
12 import android.widget.Toast;
13
14 public class MainActivity extends Activity implements LocationListener {
15     private static final String TAG = "MainActivity";
16
17     private TextView latitudeText;
18     private TextView longitudeText;
19     private LocationManager locationManager;
20     private String provider;
21
22     @Override
23     public void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.main);
26
27         latitudeText = (TextView) findViewById(R.id.latitude_text);
28         longitudeText = (TextView) findViewById(R.id.longitude_text);
29
30         locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
31
32         Criteria criteria = new Criteria();

```

```

33     provider = locationManager.getBestProvider(criteria, false);
34
35     Location location = locationManager.getLastKnownLocation(provider);
36
37     if (location != null) {
38         Log.d(TAG, "Provider " + provider + " foi selecionado.");
39         onLocationChanged(location);
40     } else {
41         latitudeText.setText(R.string.location_not_available);
42         longitudeText.setText(R.string.location_not_available);
43     }
44 }
45
46 @Override
47 protected void onResume() {
48     super.onResume();
49     locationManager.requestLocationUpdates(provider, 400, 1, this);
50 }
51
52 @Override
53 protected void onPause() {
54     super.onPause();
55     locationManager.removeUpdates(this);
56 }
57
58 @Override
59 public void onLocationChanged(Location location) {
60     double lat = location.getLatitude();
61     double lng = location.getLongitude();
62     latitudeText.setText(getString(R.string.point_label, lat));
63     longitudeText.setText(getString(R.string.point_label, lng));
64 }
65
66 @Override
67 public void onStatusChanged(String provider, int status, Bundle extras) {
68 }
69
70
71 @Override
72 public void onProviderEnabled(String provider) {
73     Toast.makeText(this, "Novo provider " + provider,
74         Toast.LENGTH_SHORT).show();
75 }
76
77
78 @Override
79 public void onProviderDisabled(String provider) {
80     Toast.makeText(this, "Provider desabilitado " + provider,
81         Toast.LENGTH_SHORT).show();
82 }
83 }

```

Código Java 10.1: MainActivity.java

Rode novamente a aplicação e veja se funciona como o esperado.



Usando o MapView

É possível mostrar mapas na aplicação utilizando o **MapView**. Para isso é necessário utilizar um *target* com suporte à Google APIs, e incluir a seguinte *tag* no *AndroidManifest.xml*:

```
1 <uses-permission android:name="android.permission.INTERNET" />
```

Código XML 10.4: AndroidManifest.xml

É necessário também ter uma chave cadastrada no Google Maps API, para que a nossa aplicação possa ter acesso ao Maps API do Google.



Mais Sobre

Para descobrir como gerar a sua chave para uso (que será necessária no próximo exercício) acesse a url abaixo:

<https://developers.google.com/maps/documentation/android/mapkey>



Exercícios de Fixação

3 Crie um novo projeto Android. Use como nome para o projeto **MapsExample**. O nome do pacote deve ser **br.com.k19.android.cap10_02**, e o nome da *activity* deve ser **MainActivity**.

4 Edite o arquivo **AndroidManifest.xml** e deixe-o com o seguinte conteúdo:

```

1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2   package="br.com.k19.android.cap10_02"
3   android:versionCode="1"
4   android:versionName="1.0" >
5
6   <uses-sdk
7     android:minSdkVersion="8"
8     android:targetSdkVersion="15" />
9
10  <uses-permission android:name="android.permission.INTERNET" />
11
12  <application
13    android:icon="@drawable/ic_launcher"
14    android:label="@string/app_name"
15    android:theme="@style/AppTheme" >
16
17    <uses-library android:name="com.google.android.maps" />
18
19    <activity
20      android:name=".MainActivity"
21      android:label="@string/title_activity_main" >
22      <intent-filter>
23        <action android:name="android.intent.action.MAIN" />
24
25        <category android:name="android.intent.category.LAUNCHER" />
26      </intent-filter>
27    </activity>
28  </application>
29
30 </manifest>

```

Código XML 10.5: AndroidManifest.xml

Crie um arquivo chamado **CustomItemizedOverlay** com o seguinte conteúdo:

```

1 package br.com.k19.android.cap10_02;
2
3 import java.util.ArrayList;
4

```

```

5 import android.app.AlertDialog;
6 import android.app.AlertDialog.Builder;
7 import android.content.Context;
8 import android.content.DialogInterface;
9 import android.graphics.drawable.Drawable;
10 import android.widget.Toast;
11
12 import com.google.android.maps.ItemizedOverlay;
13 import com.google.android.maps.OverlayItem;
14
15 public class CustomItemizedOverlay extends ItemizedOverlay<OverlayItem> {
16
17     private ArrayList<OverlayItem> mOverlays = new ArrayList<OverlayItem>();
18     private Context context;
19
20     public CustomItemizedOverlay(Context context, Drawable defaultMarker) {
21         super(boundCenterBottom(defaultMarker));
22         this.context = context;
23     }
24
25     public void addOverlay(OverlayItem overlay) {
26         mOverlays.add(overlay);
27         populate();
28     }
29
30     @Override
31     protected OverlayItem createItem(int i) {
32         return mOverlays.get(i);
33     }
34
35     @Override
36     public int size() {
37         return mOverlays.size();
38     }
39
40     protected boolean onTap(int index) {
41         OverlayItem item = mOverlays.get(index);
42         AlertDialog.Builder dialog = new AlertDialog.Builder(context);
43         dialog.setTitle(item.getTitle());
44         dialog.setMessage(item.getSnippet());
45         dialog.show();
46         return true;
47     };
48 }

```

Código Java 10.2: CustomItemizedOverlay.java

Baixe a imagem do logo no site da K19 (<http://k19.com.br>) e salve na pasta *res/drawable-hdpi* com o nome de **k19_logo.png**.

Edite o arquivo **main.xml** e deixe-o igual ao exemplo abaixo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <com.google.android.maps.MapView xmlns:android="http://schemas.android.com/apk/res/↵
   android"
3     android:id="@+id/mapview"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:apiKey="API KEY"
7     android:clickable="true" />

```

Código XML 10.6: main.xml

Edite o arquivo **MainActivity.java** com o seguinte conteúdo:

```

1 package br.com.k19.android.cap10_02;

```

```

2
3 import java.util.List;
4
5 import android.content.Context;
6 import android.graphics.drawable.Drawable;
7 import android.location.Location;
8 import android.location.LocationListener;
9 import android.location.LocationManager;
10 import android.os.Bundle;
11
12 import com.google.android.maps.GeoPoint;
13 import com.google.android.maps.MapActivity;
14 import com.google.android.maps.MapController;
15 import com.google.android.maps.MapView;
16 import com.google.android.maps.MyLocationOverlay;
17 import com.google.android.maps.Overlay;
18 import com.google.android.maps.OverlayItem;
19
20 public class MainActivity extends MapActivity {
21
22     private MapController mapController;
23     private MapView mapView;
24     private CustomItemizedOverlay itemizedOverlay;
25     private MyLocationOverlay myLocationOverlay;
26
27     public void onCreate(Bundle bundle) {
28         super.onCreate(bundle);
29         setContentView(R.layout.main);
30
31         mapView = (MapView) findViewById(R.id.mapview);
32         mapView.setBuiltInZoomControls(true);
33
34         mapView.setSatellite(false);
35         mapController = mapView.getController();
36         mapController.setZoom(14);
37
38         myLocationOverlay = new MyLocationOverlay(this, mapView);
39         mapView.getOverlays().add(myLocationOverlay);
40
41
42         List<Overlay> mapOverlays = mapView.getOverlays();
43         Drawable drawable = this.getResources().getDrawable(R.drawable.k19_logo);
44         itemizedOverlay = new CustomItemizedOverlay(this, drawable);
45
46         GeoPoint point = new GeoPoint(-23570794, -46690747);
47         OverlayItem overlayitem = new OverlayItem(point, "K19", "Cursos e Treinamentos");
48
49         itemizedOverlay.addOverlay(overlayitem);
50         mapOverlays.add(itemizedOverlay);
51     }
52
53     @Override
54     protected boolean isRouteDisplayed() {
55         return false;
56     }
57 }

```

Código Java 10.3: MainActivity.java

Rode a aplicação e veja o resultado.





Introdução

O Android provê uma API robusta para desenvolvimento de aplicações com suporte a reprodução e gravação de áudio e imagem. Neste capítulo vamos entender como funcionam as principais funções de manipulação de mídia.



Reprodução de Mídia

O framework de multimedia do Android é capaz reproduzir os tipos mais comuns de mídia. Com ele é possível reproduzir áudio de vídeo puros ou codificados a partir do sistema de arquivos ou mesmo através da internet.

Classes importantes

A reprodução de mídia é uma tarefa relativamente simples e para adicionarmos esta funcionalidade em uma aplicação Android vamos utilizar duas das classes mais importantes do framework.

MediaPlayer - Principal classe para execução de som e video

AudioManager - Esta classe manipula as entradas e saídas de áudio do dispositivo.

Manifest

Dependendo da funcionalidade que a aplicação irá utilizar do framework será preciso configurar o manifesto com as permissões necessárias:

Caso seja necessário utilizar o MediaPlayer como player de um stream através da internet então temos que adicionar a seguinte permissão:

```
1 <uses-permission android:name="android.permission.INTERNET" />
```

Código XML 11.1: Internet Permission

Caso o seu player precise manter a tela sem esmaecer ou manter o processador sem entrar em modo de economia de energia então é preciso adicionar a seguinte permissão para que os métodos MediaPlayer.setScreenOnWhilePlaying() or MediaPlayer.setWakeMode() sejam executados.

```
1 <uses-permission android:name="android.permission.WAKE_LOCK" />
```

Código XML 11.2: Wake Lock Permission

Utilizando o MediaPlayer

O MediaPlayer é um dos componentes de mídia mais importantes do framework do Android. Uma instancia desta classe é capaz de buscar, decodificar e reproduzir conteúdos de áudio e vídeo exigindo pouca configuração.

Principais funções da classe MediaPlayer:

MediaPlayer.setDataSource() - Seleciona a mídia (caminho do arquivo local ou remoto) a ser reproduzida;

MediaPlayer.prepare() - Prepara (decodifica, pré-armazena, etc) a mídia transformando-a em áudio puro pronto para ser reproduzido;

MediaPlayer.start() - Inicia a reprodução do áudio;

MediaPlayer.pause() - Pausa a reprodução do áudio;

MediaPlayer.stop() - Encerra a reprodução do áudio;

mediaPlayer.setAudioStreamType() - Define o tipo de mídia que será reproduzido. Para arquivos de música vamos passar como parâmetro a constante AudioManager.STREAM_MUSIC.

Abaixo temos um trecho de código que reproduz um arquivo de áudio puro localizado em res/raw/. Neste caso estamos utilizando como entrada o recurso de áudio da própria aplicação.

```
1 // criando a instancia do MediaPlayer
2 MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.arquivo_som_puro);
3 // iniciando a reprodução
4 mediaPlayer.start();
```

Código Java 11.1: Reprodução de áudio puro

No caso acima, como o arquivo contém som puro, não existe a necessidade de nenhuma preparação (decodificação) para ser reproduzido. Contudo, serão raras as vezes em que a mídia a ser reproduzida não estará codificada. Por esta razão a classe MediaPlayer suporta os tipos mais comuns de formatos e podemos utilizar o método MediaPlayer.prepare() para reproduzi-los como mostra o código a seguir.

```
1 // localização do arquivo local de mídia
2 Uri myUri = ....;
3 // criando um player
4 MediaPlayer mediaPlayer = new MediaPlayer();
5 // definindo que o tipo de stream é arquivo de música
6 mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
7 // fazendo com que o player encontre o arquivo de entrada
8 mediaPlayer.setDataSource(getApplicationContext(), myUri);
9 // preparando a música para ser reproduzida
10 mediaPlayer.prepare();
11 // iniciando a reprodução
12 mediaPlayer.start();
```

Código Java 11.2: Arquivo de mídia local

```
1 // localização do arquivo remoto de mídia
2 String url = "http://.....";
3 // criando um player
```



```

4 MediaPlayer mediaPlayer = new MediaPlayer();
5 // definindo que o tipo de stream é arquivo de música
6 mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
7 // fazendo com que o player encontre o arquivo de entrada
8 mediaPlayer.setDataSource(url);
9 // preparando a música para ser reproduzida
10 mediaPlayer.prepare();
11 // iniciando a reprodução
12 mediaPlayer.start();

```

Código Java 11.3: Mídia transmitida pela internet



Mais Sobre

Para saber mais sobre os tipos de mídia suportados pelo MediaPlayer acesse <http://developer.android.com/guide/appendix/media-formats.html>

Preparação Assíncrona

Com o que foi visto até este ponto podemos desenvolver uma aplicação que reproduz áudio sem muito esforço. Porém, quando se trata de uma mídia codificada em algum formato específico, então a chamada do método `MediaPlayer.prepare()` pode levar muito tempo para busca, decodificação e pré-carregamento do áudio e isto pode acarretar em uma lentidão sensível da nossa aplicação.

Para evitar este tipo de problema não podemos executar este procedimento utilizando a thread da interface, mas sim, iniciar um processo assíncrono que realizará estas atividades custosas e em seguida comunicará a thread principal que a tarefa foi concluída. Este tipo de abordagem é tão comum que a classe `MediaPlayer` já fornece suporte a carregamento assíncrono através do método `prepareAsync()`.

Basta associar ao player um objeto que implementa a interface `MediaPlayer.OnPreparedListener` e sobrescrever o método `MediaPlayer.OnPreparedListener.onPrepared()`. Uma vez que o método `prepareAsync()` for chamado e terminar de preparar a mídia o método `onPrepared()` será executado e o player pode iniciar a reprodução da mídia.

O código abaixo é um exemplo do uso assíncrono do método `prepare()`.

```

1 public class AudioPlaybackTest extends Activity implements MediaPlayer.OnPreparedListener {
2     MediaPlayer mMediaPlayer = null;
3     ...
4
5     @Override
6     public void onCreate(Bundle icle) {
7         super.onCreate(icle);
8         ...
9
10        mMediaPlayer = ... // inicialização do MediaPlayer
11        mMediaPlayer.setOnPreparedListener(this);
12        mMediaPlayer.prepareAsync();
13    }
14
15    /** Chamado quando o MediaPlayer estiver pronto */
16    @Override
17    public void onPrepared(MediaPlayer player) {
18        player.start();
19    }
20 }

```

Código Java 11.4: Exemplo de Activity com preparação assíncrona

Estados de execução

Quando estamos utilizando o MediaPlayer devemos levar em consideração o estado em que ele está antes de executar um comando. Cada comando só pode ser utilizado em alguns estados e se isto não for respeitado a aplicação pode se comportar de maneira inesperada e até ser finalizada com erro.

Quando criamos um objeto do tipo MediaPlayer, ele está no estado Idle. Ao executar o método `setDataSource` ele passa para o estado `Initialized`. Depois da execução do método `prepare` (ou `prepareAsync`) o objeto passa a assumir o estado `Prepared`. Os métodos `start` e `pause` alternam o estado do player entre `Started` e `Paused`. Quando a execução termina o estado é `PlaybackCompleted` e quando o método `stop` é executado o estado do player é `Stopped`.

Um exemplo de comando inválido seria chamar o método `start` com o player no estado `Stopped`. Neste estado é preciso chamar novamente o método `prepare()` se desejar reproduzir a mídia novamente.

Veja abaixo o mapa de estados completo do MediaPlayer.

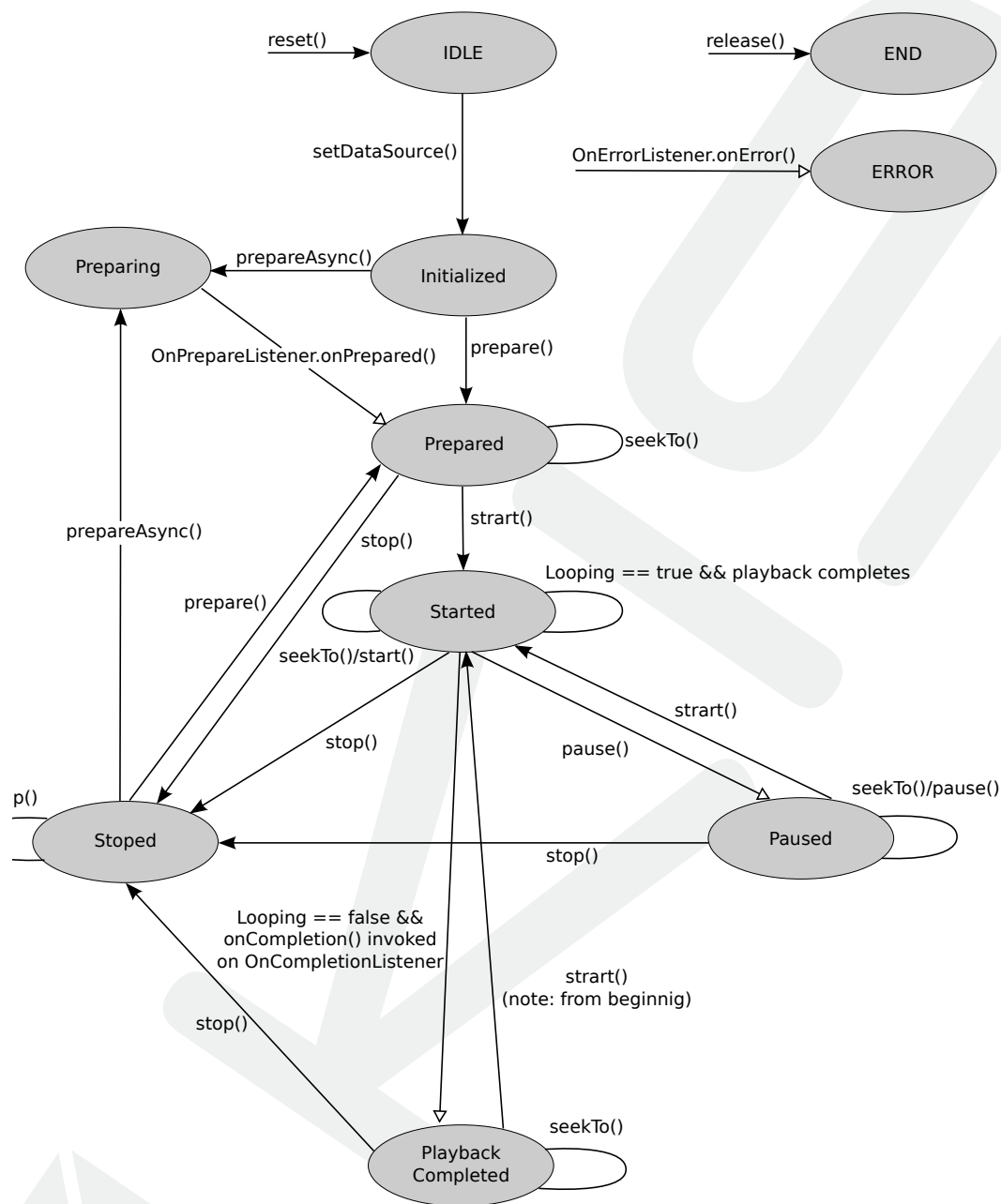


Figura 11.1: Resultado da tela.

Desalocando o recurso do MediaPlayer

É preciso sempre ter em mente que o MediaPlayer consome muito recurso do dispositivo e portanto deve-se ter certeza de que não estamos segurando recursos por tempo além do que realmente é necessário.

Observe abaixo o código que pode ser utilizado para liberar o recurso do MediaPlayer.

```

1 // liberando o recurso alocado
2 mediaPlayer.release();
3 // perdendo a referência do MediaPlayer
4 mediaPlayer = null;

```

Código Java 11.5: Liberação de recurso alocado pelo MediaPlayer

Se a intenção é desenvolver uma aplicação que não mantenha a reprodução em background (vamos explicar a seguir) então é interessante liberar o recurso do MediaPlayer no momento da chamada onStop() da activity.

Reproduzindo mídia em Background

Para desenvolver uma aplicação que reproduza mídia em background, ou seja, continue executando a mídia mesmo que outra aplicação esteja na tela, então será preciso iniciar um Serviço (visto em capítulos anteriores) e controlar o MediaPlayer através dele. Assim como recomendado em uma Activity normal, o Service deve executar as tarefas assincronamente. Observe o código de exemplo abaixo:

```

1 public class MyService extends Service implements MediaPlayer.OnPreparedListener {
2     private static final ACTION_PLAY = "br.com.k19.action.PLAY";
3     MediaPlayer mMediaPlayer = null;
4
5     public int onStartCommand(Intent intent, int flags, int startId) {
6         ...
7         if (intent.getAction().equals(ACTION_PLAY)) {
8             mMediaPlayer = ... // inicialização do player
9             mMediaPlayer.setOnPreparedListener(this);
10            mMediaPlayer.prepareAsync(); // preparando a mídia assincronamente
11        }
12    }
13
14    /** Chamado quando o MediaPlayer estiver pronto */
15    @Override
16    public void onPrepared(MediaPlayer player) {
17        player.start();
18    }
19 }

```

Código Java 11.6: Reprodução em background

Podemos ainda melhorar a implementação acima adicionando tratamento de erros de operações assíncronas. Bastando, para isto, associar ao player um objeto que implementa MediaPlayer.OnErrorListener utilizando o método setOnErrorListener() e sobrescrever o método MediaPlayer.OnErrorListener.onError(). Se ocorrer algum erro fora da thread principal o método onError() será executado e o erro poderá ser tratado apropriadamente.

Por último, lembre-se de que o MediaPlayer consome muito recurso e é recomendado chamar explicitamente o método MediaPlayer.release() já que o Garbage Collector pode demorar muito até liberar os recursos alocados. Para garantir este comportamento, sobrescreva o método Service.onDestroy().

```

1 public class MyService extends Service implements MediaPlayer.OnPreparedListener, ↵
2     MediaPlayer.OnErrorListener {
3     private static final ACTION_PLAY = "br.com.k19.action.PLAY";
4     MediaPlayer mMediaPlayer = null;
5
6     public int onStartCommand(Intent intent, int flags, int startId) {
7         ...
8         if (intent.getAction().equals(ACTION_PLAY)) {
9             mMediaPlayer = ... // inicialização do player
10            mMediaPlayer.setOnPreparedListener(this);
11            mMediaPlayer.setOnErrorListener(this);
12            mMediaPlayer.prepareAsync(); // preparando a mídia assincronamente

```

```

12     }
13 }
14
15 @Override
16 public void onPrepared(MediaPlayer player) {
17     player.start();
18 }
19
20 @Override
21 public boolean onError(MediaPlayer mp, int what, int extra) {
22     // Tratamento do erro
23 }
24
25 @Override
26 public void onDestroy() {
27     if (mMediaPlayer != null) mMediaPlayer.release();
28 }
29
30 }

```

Código Java 11.7: Versão melhorada

É importante lembrar que se ocorrer algum erro no MediaPlayer o seu estado passará para Error e deverá ser resetado utilizando o método `reset()` para ser útil novamente.



Exercícios de Fixação

- 1 Crie um novo projeto Android. Use como nome para o projeto **MediaPlayer**. O nome do pacote deve ser **br.com.k19.android.cap11**, e o nome da *activity* deve ser **MainActivity**.
- 2 Edite o arquivo **AndroidManifest.xml** e deixe-o com o seguinte conteúdo:

```

1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2     package="br.com.k19.android.cap11"
3     android:versionCode="1"
4     android:versionName="1.0" >
5
6     <uses-sdk
7         android:minSdkVersion="8"
8         android:targetSdkVersion="15" />
9
10    <application
11        android:icon="@drawable/ic_launcher"
12        android:label="@string/app_name"
13        android:theme="@style/AppTheme" >
14        <activity
15            android:name=".MainActivity"
16            android:label="@string/title_activity_main" >
17            <intent-filter>
18                <action android:name="android.intent.action.MAIN" />
19
20                <category android:name="android.intent.category.LAUNCHER" />
21            </intent-filter>
22        </activity>
23
24        <service android:name=".MediaPlayerService">
25            </service>
26    </application>
27
28 </manifest>

```

Código XML 11.3: Manifest.xml

3 Crie uma pasta chamada raw dentro da pasta res. Coloque um arquivo mp3 chamado sample.mp3 dentro de raw.

4 Crie um arquivo na pasta src chamado MediaPlayerService.java e crie nela uma classe chamada MediaPlayerService que herda de Service e implementa as interfaces OnPreparedListener e OnErrorListener. O código deve ficar como o exemplo abaixo:

```
1 package br.com.k19.android.cap11;
2
3 import java.io.IOException;
4
5 import android.app.Service;
6 import android.content.Intent;
7 import android.media.MediaPlayer;
8 import android.media.MediaPlayer.OnErrorListener;
9 import android.media.MediaPlayer.OnPreparedListener;
10 import android.net.Uri;
11 import android.os.IBinder;
12 import android.widget.Toast;
13
14 public class MediaPlayerService extends Service implements OnPreparedListener, ↵
15     OnErrorListener {
16
17     private MediaPlayer mMediaPlayer;
18
19     @Override
20     public int onStartCommand(Intent intent, int flags, int startId) {
21         Toast.makeText(this, "Iniciando o Servico", Toast.LENGTH_SHORT).show();
22
23         try {
24             mMediaPlayer = new MediaPlayer();
25             Uri path = Uri.parse("android.resource://br.com.k19.android.cap11/" + R.raw.↵
26                 sample);
27             mMediaPlayer.setDataSource(getApplicationContext(), path);
28             mMediaPlayer.setOnPreparedListener(this);
29             mMediaPlayer.prepareAsync();
30         } catch (IOException e) {
31             // recurso nao encontrado
32         }
33
34         return super.onStartCommand(intent, flags, startId);
35     }
36
37     @Override
38     public void onDestroy() {
39         super.onDestroy();
40         Toast.makeText(this, "Terminando o Servico", Toast.LENGTH_SHORT).show();
41         if (mMediaPlayer != null) {
42             mMediaPlayer.release();
43             mMediaPlayer = null;
44         }
45     }
46
47     @Override
48     public IBinder onBind(Intent intent) {
49         return null;
50     }
51
52     public void onPrepared(MediaPlayer mp) {
```

```

52     mMediaPlayer.start();
53 }
54
55 public boolean onError(MediaPlayer arg0, int arg1, int arg2) {
56     // tratamento de erros
57     return false;
58 }
59 }

```

Código Java 11.8: MediaPlayer

- 5 O arquivo res/values/strings.xml deve ficar com o seguinte conteúdo:

```

1 <resources>
2   <string name="app_name">MediaPlayer</string>
3   <string name="hello_world">Now playing!</string>
4   <string name="menu_settings">Settings</string>
5   <string name="title_activity_main">MainActivity</string>
6 </resources>

```

Código XML 11.4: string.xml

- 6 A classe MainActivity deve ficar parecida com o código a seguir:

```

1 package br.com.k19.android.cap11;
2
3 import br.com.k19.android.cap11.R;
4 import android.os.Bundle;
5 import android.app.Activity;
6 import android.content.Intent;
7 import android.view.Menu;
8
9 public class MainActivity extends Activity {
10
11     private Intent intent;
12
13     @Override
14     public void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17
18         intent = new Intent(this, ExampleService.class);
19         startService(intent);
20     }
21
22     @Override
23     protected void onDestroy() {
24         super.onDestroy();
25         stopService(intent);
26     };
27
28     @Override
29     public boolean onCreateOptionsMenu(Menu menu) {
30         getMenuInflater().inflate(R.menu.activity_main, menu);
31         return true;
32     }
33 }

```

Código Java 11.9: Activity

- 7 Rode a aplicação.



Captura de Áudio

O framework do Android suporta também gravação de áudio nos formatos mais comuns tornando a aplicação bastante versátil. A classe mais importante para esta tarefa é a `MediaRecorder`.

Manifest

É preciso explicitar no arquivo de manifesto o pedido de permissão para gravar áudio.

```
1 <uses-permission android:name="android.permission.RECORD_AUDIO" />
```

Código XML 11.5: Permissão de gravação de áudio

Gravando áudio pelo microfone

Para iniciar uma gravação de áudio pelo microfone basta seguir alguns passos:

1. Criar uma instância de `android.media.MediaRecorder`.
2. Configurar a fonte de áudio, que neste caso é o microfone, utilizando o método `MediaRecorder.setAudioSource()` passando a constante `MediaRecorder.AudioSource.MIC`.
3. Em seguida configure o formato de saída `MediaRecorder.setOutputFormat()` passando o formato desejado. Veja abaixo as opções:

AAC_ADTS - AAC ADTS file format

AMR_NB - AMR NB file format

AMR_WB - AMR WB file format

DEFAULT - Tipo padrão

MPEG_4 - MPEG4 media file format

THREE_GPP - 3GPP media file format

4. Dê um nome ao arquivo de saída através do método `MediaRecorder.setOutputFile()` passando um file descriptor (String) que define o arquivo de saída. Exemplo:

```
1 mRecorder.setOutputFile(fileName);
```

Código Java 11.10: Definindo arquivo de saída

5. O próximo passo é definir o encoder usando o método `MediaRecorder.setAudioEncoder()`. Veja abaixo as opções:

AAC - AAC Low Complexity (AAC-LC) audio codec

AAC_ELD - Enhanced Low Delay AAC (AAC-ELD) audio codec

AMR_NB - AMR (Narrowband) audio codec

AMR_WB - AMR (Wideband) audio codec

DEFAULT - Tipo padrão

HE_AAC - High Efficiency AAC (HE-AAC) audio codec

6. Com estas informações definidas podemos chamar o método `MediaRecorder.prepare()`.
7. Quando o recorder estiver preparado podemos iniciar a gravação com o método `MediaRecorder.start()`.
8. Para terminar de gravar basta chamar o método `MediaRecorder.stop()`.
9. Depois de concluída a gravação, lembre-se de chamar o método `MediaRecorder.release()` para liberar os recursos alocados pelo gravador.

Observe abaixo um trecho de código de um exemplo simples de gravação a partir do microfone.

```
1 MediaRecorder recorder = new MediaRecorder();
2 recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
3 recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
4 recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
5 recorder.setOutputFile(PATH_NAME);
6 recorder.prepare();
7 recorder.start(); // iniciando a gravação
8 ...
9 recorder.stop();
10 recorder.release(); // Liberando o recurso alocado
```

Código Java 11.11: Gravando a partir do microfone

Estados de execução

Assim como o `MediaPlayer`, o `MediaRecorder` pode assumir alguns estados dependendo da etapa de gravação que a aplicação estiver realizando. Novamente, dependendo do estado que o recorder estiver, alguns comandos podem ou não ser executados.

Veja abaixo o mapa completo de possíveis estados do `MediaRecorder`.

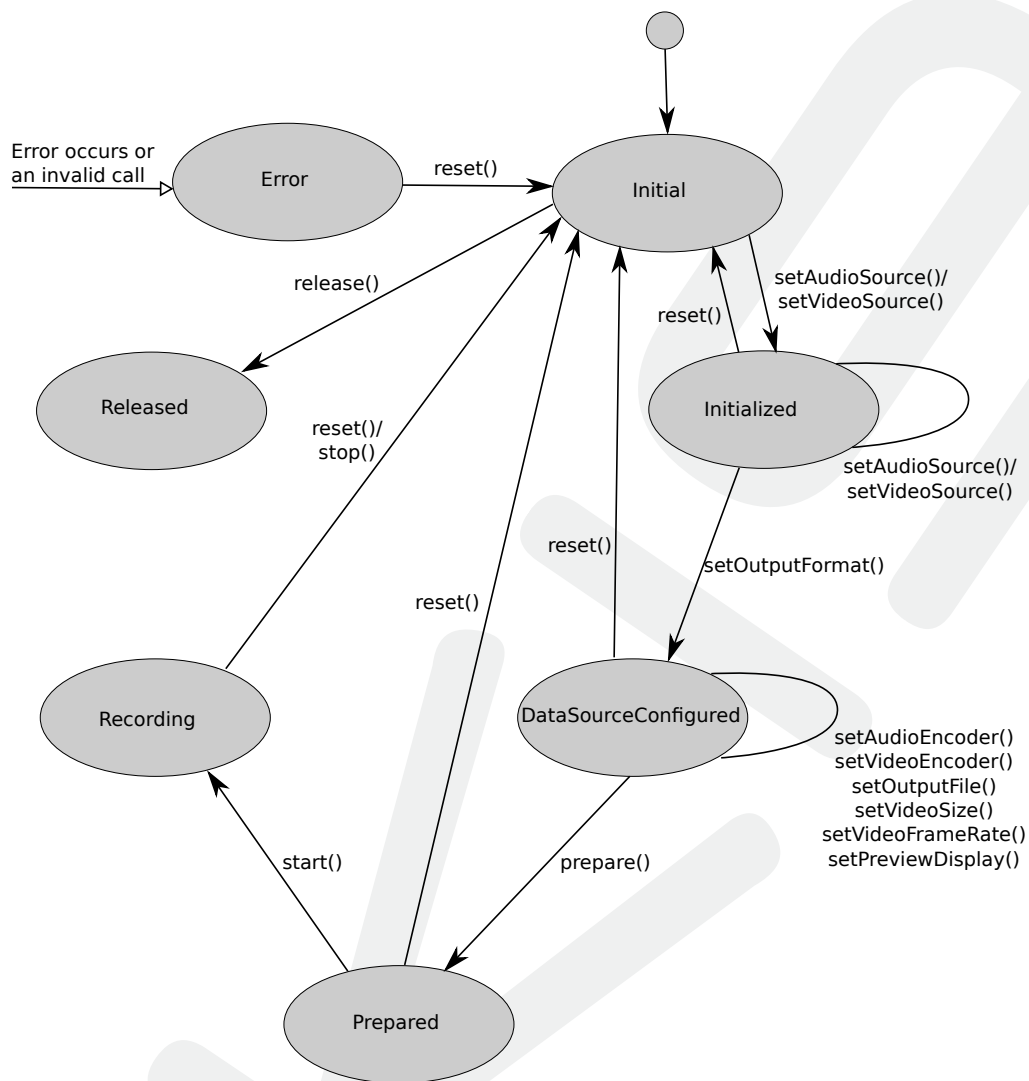


Figura 11.2: Resultado da tela



Exercícios de Fixação

- 8 Crie um novo projeto Android. Use como nome para o projeto **MediaRecorder**. O nome do pacote deve ser **br.com.k19.android.cap11**, e o nome da *activity* deve ser **MainActivity**.
- 9 Na pasta **res/layouts** crie um arquivo chamado **activity_main.xml**. Ele deve conter o seguinte conteúdo:

```

1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent" >
  
```

```

5
6      <TextView
7          android:id="@+id/textView1"
8          android:layout_width="wrap_content"
9          android:layout_height="wrap_content"
10         android:layout_centerHorizontal="true"
11         android:layout_centerVertical="true"
12         android:text="@string/hello_world"
13         tools:context=".MainActivity" />
14
15      <Button
16          android:id="@+id/button1"
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content"
19          android:layout_alignParentLeft="true"
20          android:layout_alignParentTop="true"
21          android:text="@string/gravar_button_label" />
22
23      <Button
24          android:id="@+id/button2"
25          android:layout_width="wrap_content"
26          android:layout_height="wrap_content"
27          android:layout_alignParentRight="true"
28          android:layout_alignParentTop="true"
29          android:text="@string/reproduzir_button_label" />
30
31 </RelativeLayout>

```

Código XML 11.6: activity_main.xml

- 10 O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <resources>
2     <string name="app_name">MediaRecorder</string>
3     <string name="hello_world">Recording Audio!</string>
4     <string name="menu_settings">Settings</string>
5     <string name="title_activity_main">MainActivity</string>
6     <string name="gravar_button_label">Iniciar gravacao</string>
7     <string name="reproduzir_button_label">Iniciar reproducao</string>
8 </resources>

```

Código XML 11.7: strings.xml

- 11 O arquivo **AndroidManifest.xml** deve ficar com o seguinte conteúdo:

```

1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2     package="br.com.k19.android.cap11"
3     android:versionCode="1"
4     android:versionName="1.0" >
5
6     <uses-sdk
7         android:minSdkVersion="8"
8         android:targetSdkVersion="15" />
9
10    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
11    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
12
13    <application
14        android:icon="@drawable/ic_launcher"
15        android:label="@string/app_name"
16        android:theme="@style/AppTheme" >
17        <activity
18            android:name=".MainActivity"
19            android:label="@string/title_activity_main" >

```

```
20         <intent-filter>
21             <action android:name="android.intent.action.MAIN" />
22
23             <category android:name="android.intent.category.LAUNCHER" />
24         </intent-filter>
25     </activity>
26 </application>
27
28 </manifest>
```

Código XML 11.8: AndroidManifest.xml

12 Edite o arquivo **MainActivity.java** com o seguinte conteúdo:

```
1 package br.com.k19.android.cap11;
2
3 import java.io.File;
4 import java.io.IOException;
5
6 import android.app.Activity;
7 import android.media.MediaPlayer;
8 import android.media.MediaPlayer.OnCompletionListener;
9 import android.media.MediaRecorder;
10 import android.os.Bundle;
11 import android.os.Environment;
12 import android.util.Log;
13 import android.view.MenuItem;
14 import android.view.View;
15 import android.view.View.OnClickListener;
16 import android.widget.Button;
17
18 public class MainActivity extends Activity
19 {
20     private final String LOG_TAG = "MainActivity";
21
22     private MediaRecorder mRecorder;
23     private boolean recording;
24     Button gravarButton;
25
26     private MediaPlayer mPlayer;
27     private boolean playing;
28     Button reproduzirButton;
29
30     private String mFileName;
31
32     private void onPlay() {
33         if (playing) {
34             reproduzirButton.setText("Iniciar execucao");
35             mPlayer.stop();
36             mPlayer.release();
37             mPlayer = null;
38         }
39         else {
40             try{
41                 reproduzirButton.setText("Parar execucao");
42                 mPlayer = new MediaPlayer();
43
44                 mPlayer.setOnCompletionListener(new OnCompletionListener() {
45                     public void onCompletion(MediaPlayer mp) {
46                         playing = true;
47                         onPlay();
48                     }
49                 });
50
51                 mPlayer.setDataSource(mFileName);
52                 mPlayer.prepare();
53                 mPlayer.start();
```

```

54     }
55     catch (Exception e){
56         Log.e(LOG_TAG, e.getMessage());
57     }
58 }
59 playing = !playing;
60 }
61
62 private void onRecord() {
63     if (recording) {
64         gravarButton.setText("Iniciar gravacao");
65         mRecorder.stop();
66         mRecorder.release();
67         mRecorder = null;
68     }
69     else {
70         try {
71             mRecorder = new MediaRecorder();
72             mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
73             mRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
74             mRecorder.setOutputFile(mFileName);
75             mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
76             mRecorder.prepare();
77             mRecorder.start();
78             gravarButton.setText("Parar gravacao");
79
80         } catch (IOException e) {
81             Log.e(LOG_TAG, e.getMessage());
82         }
83     }
84     recording = !recording;
85 }
86
87 @Override
88 protected void onCreate(Bundle savedInstanceState) {
89     super.onCreate(savedInstanceState);
90
91     String filepath = Environment.getExternalStorageDirectory().getPath();
92     File file = new File(filepath, "RecordFolder");
93     if (!file.exists()){
94         file.mkdirs();
95     }
96     mFileName = file.getAbsolutePath() + "/lastRecordedFile.3gp";
97
98     recording = false;
99
100    setContentView(R.layout.activity_main);
101
102    gravarButton = (Button) findViewById(R.id.button1);
103    reproduzirButton = (Button) findViewById(R.id.button2);
104
105    gravarButton.setOnClickListener(new OnClickListener(){
106        public void onClick(View v) {
107            onRecord();
108        }
109    });
110
111    reproduzirButton.setOnClickListener(new OnClickListener(){
112        public void onClick(View v) {
113            onPlay();
114        }
115    });
116 }
117
118 @Override
119 public boolean onOptionsItemSelected(int featureId, MenuItem item) {
120     return super.onOptionsItemSelected(featureId, item);
121 }
122 }

```

Código Java 11.12: MainActivity.java

- 13 Execute a aplicação. Note que por uma questão de simplicidade não tomamos todos cuidados com relação a concorrência entre gravação e reprodução, bem como acessos a arquivos inexistentes.



Captura de Vídeo

Objetivo

Nesta seção vamos aprender a manipular a câmera produzindo imagens e videos com áudio. O android expõe dos recursos da câmera, quando presente, de duas maneiras: Via intent ou diretamente via API. Veremos a seguir que cada modo deve ser utilizado de acordo com a necessidade.

Manifest

Antes de utilizar o recurso de captação de imagens em nossa aplicação devemos considerar se a câmera e suas funções são ou não obrigatórias para um bom funcionamento do seu aplicativo. Se sim é preciso declarar no Manifesto o uso da câmera e desta maneira não será permitida a instalação do aplicativo em dispositivos sem câmera.

Sua aplicação deve requisitar permissão para uso da câmera.

```
1 <uses-permission android:name="android.permission.CAMERA" />
```

Código XML 11.9: Permissão para utilização de câmera.

Sua aplicação deve também requisitar acesso à algumas funções da câmera:

```
1 <uses-feature android:name="android.hardware.camera" />
```

Código XML 11.10: Permissão para utilização de recursos da câmera.

A declaração de utilização de funções é interessante para que o Google Play evite que sua aplicação seja instalada em dispositivos que não tenham a função específica. Veja outras funções:

```
1 <uses-feature android.hardware.camera.autofocus />
2 <uses-feature android.hardware.camera.flash />
3 <uses-feature android.hardware.camera.front />
```

Código XML 11.11: Mais tipos de permissões para acesso a recursos da câmera.

Se a aplicação utilizar dos recursos da câmera não sendo eles obrigatórios então é preciso adicionar um atributo na declaração indicando que ela é opcional mas desejável. Exemplo:

```
1 <uses-feature android:name="android.hardware.camera" android:required="false" />
```

Código XML 11.12: Recurso desejável

Existem outros tipos de permissão que podem ser úteis em nossa aplicação com câmera:

Permissão de armazenamento - Se sua aplicação salva imagens ou vídeos em dispositivos externos de armazenamento.

```
1 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Código XML 11.13: Permissão de armazenamento

Permissão de gravação de áudio - Para gravar áudio durante a captura de vídeo.

```
1 <uses-permission android:name="android.permission.RECORD_AUDIO" />
```

Código XML 11.14: Permissão de gravação de áudio

Permissão de localização - Se sua aplicação utiliza informação de localização para classificar imagens.

```
1 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Código XML 11.15: Permissão de localização

Utilizando a câmera via intent

Quando queremos apenas manipular fotos e vídeos sem ter que escrever muito código podemos requisitar a captura da imagem para uma outra aplicação via intent. Esta outra aplicação realiza a captura da imagem ou vídeo e devolve o controle para a sua aplicação poder manipular o item recém capturado.

O primeiro passo é criar um intent que irá requisitar a imagem ou vídeo. Este intent pode ser de dois tipos:

MediaStore.ACTION_IMAGE_CAPTURE - Intent utilizado para requisitar imagens de uma aplicação de câmera existente.

MediaStore.ACTION_VIDEO_CAPTURE - Intent utilizado para requisitar vídeos de uma aplicação de câmera existente.

Em seguida precisamos inicializar o intent. Neste passo vamos utilizar o método `startActivityForResult()` para executar o intent e então o é transeferido para a aplicação de manipulação de câmera e o usuário poderá tirar a sua foto ou gravar o seu vídeo. Observe abaixo dois exemplos de implementação de aplicações. Uma que captura imagens e outra que captura vídeos.

Capturando imagens

No código abaixo podemos ver como podem ser feitos os dois primeiros passos descritos acima. Note ainda que adicionamos um atributo extra ao intent definindo onde o arquivo de saída será gravado. Para isto utilizamos o método `Intent.putExtra()`.

```
1 private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
2 private Uri fileUri;
3
4 @Override
5 public void onCreate(Bundle savedInstanceState) {
6     super.onCreate(savedInstanceState);
```

```
7   setContentView(R.layout.main);
8
9   // criando o Intent
10  Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
11
12  // configurando onde o arquivo de saída será gravado
13  fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE);
14  intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
15
16  // iniciando o Intent
17  startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
18 }
```

Código Java 11.13: Capturando imagens

Perceba que além do intent, o método `startActivityResult` ainda recebe um inteiro que será utilizado, mais adiante, para identificar o intent.

Capturando vídeos

De uma maneira muito similar ao exemplo anterior podemos criar um intent para captura de vídeo e executá-lo da maneira abaixo. Note que além do atributo `MediaStore.EXTRA_OUTPUT` que define o local onde o vídeo será armazenado estamos também configurando a qualidade de gravação de vídeo colocando o valor 1 na propriedade `MediaStore.EXTRA_VIDEO_QUALITY` (o valor 0 indica baixa qualidade de captura).

```
1 private static final int CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE = 200;
2 private Uri fileUri;
3
4 @Override
5 public void onCreate(Bundle savedInstanceState) {
6     super.onCreate(savedInstanceState);
7     setContentView(R.layout.main);
8
9     // criando o intent
10    Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
11
12    // configurando onde o arquivo de saída será gravado
13    fileUri = getOutputMediaFileUri(MEDIA_TYPE_VIDEO);
14    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
15
16    // configurando a qualidade do vídeo
17    intent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, 1);
18
19    // iniciando o Intent
20    startActivityForResult(intent, CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE);
21 }
```

Código Java 11.14: Capturando vídeos

Podemos também definir o tamanho máximo do vídeo, em segundos, configurando a propriedade `MediaStore.EXTRA_DURATION_LIMIT`, ou limitar o tamanho máximo do vídeo, em bytes, utilizando a propriedade `MediaStore.EXTRA_SIZE_LIMIT`.

Recuperando os dados gravados

Por último precisamos recuperar os dados capturados no passo anterior. Para recebermos o resultado da ação do usuário no aplicativo de câmera devemos sobrescrever o método `onActivityResult(int requestCode, int resultCode, Intent data)` da atividade que iniciou o intent. Uma vez que o

usuário termine a ação no aplicativo de câmera este método será chamado (callback) e receberemos a imagem ou o vídeo para que a nossa aplicação os manipule.

Vamos utilizar o parâmetro requestCode para descobrir qual intent gerou aquela resposta (foto ou vídeo). Em seguida testamos a variável resultCode para verificar se tudo correu bem com a captura ou o usuário cancelou a aplicação sem tirar uma foto ou gravar um vídeo. No caso de sucesso, a imagem ou o vídeo estará disponível como resultado da chamada data.getData().

Observe o trecho abaixo com um exemplo de implementação do callback.

```

1 private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
2 private static final int CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE = 200;
3
4 @Override
5 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
6     // verificando o tipo da respost
7     if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
8
9         if (resultCode == RESULT_OK) {
10             Toast.makeText(this, "Imagem salva em:\n" + data.getData(), Toast.LENGTH_LONG).show();
11         } else if (resultCode == RESULT_CANCELED) {
12             // Cancelamento da operacao
13         } else {
14             // Ocorreu algo inesperado
15         }
16     }
17     // verificando o tipo da respost
18     if (requestCode == CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE) {
19
20         if (resultCode == RESULT_OK) {
21             Toast.makeText(this, "Video salvo em:\n" +
22                 data.getData(), Toast.LENGTH_LONG).show();
23         } else if (resultCode == RESULT_CANCELED) {
24
25         } else {
26
27         }
28     }
29 }

```

Código Java 11.15: Capturando vídeos

Utilizando a câmera via API

Quando queremos desenvolver uma versão customizada de aplicação de câmera proporcionando uma experiência de uso mais interessante e complexa com funções especiais, então ao invés de utilizar o recurso via Intent (utilizando outro aplicativo) podemos acessar a câmera diretamente via API.

Para possibilitar a captura de imagens ou vídeo precisamos seguir alguns passos:

Detecção e acesso à câmera

Caso a sua aplicação não necessite da câmera obrigatoriamente podemos definir um método para verificação da existência do recurso no dispositivo. Uma instância do PackageManager que está no contexto da aplicação permite que o método PackageManager.hasSystemFeature() seja executado. Este método retorna um valor booleano indicando a presença ou não de uma feature.

```

1 private boolean checkCameraHardware(Context context) {

```

```

2 // consultando a presença da câmera no dispositivo
3 if (context.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA)){
4     return true;
5 } else {
6     return false;
7 }
8 }

```

Código Java 11.16: verificando o hardware

Uma maneira segura de obter o recurso da câmera é a seguinte:

```

1 public static Camera getCameraInstance(){
2     Camera c = null;
3     try {
4         c = Camera.open();
5     }
6     catch (Exception e){
7         // o recurso pode estar indisponível ou sendo utilizado
8         // é preciso prever estas situações
9     }
10    return c;
11 }

```

Código Java 11.17: Criando a instância da Camera

Criação de uma classe para a visualização da câmera

Vamos criar uma classe (View) que estende SurfaceView e implementa a interface SurfaceHolder para ser utilizada na visualização das imagens geradas pela câmera.

```

1 /** A basic Camera preview class */
2 public class CameraPreview extends SurfaceView implements SurfaceHolder.Callback {
3     private SurfaceHolder mHolder;
4     private Camera mCamera;
5
6     public CameraPreview(Context context, Camera camera) {
7         super(context);
8         mCamera = camera;
9
10        // Adicionando um SurfaceHolder.Callback assim receberemos uma notificação
11        // assim que o Surface é criado.
12        mHolder = getHolder();
13        mHolder.addCallback(this);
14        // configuracao obsoleta, necessaria apenas nas versoes anteriores ao Android 3.0
15        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
16    }
17
18    public void surfaceCreated(SurfaceHolder holder) {
19        // Com o Surface criado é preciso dizer onde desenhar a imagem prévia
20        try {
21            mCamera.setPreviewDisplay(holder);
22            mCamera.startPreview();
23        } catch (IOException e) {
24            Log.d(TAG, "Erro ao associar o preview: " + e.getMessage());
25        }
26    }
27
28    public void surfaceDestroyed(SurfaceHolder holder) {
29    }
30
31    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
32        // E preciso acompanhar as rotacoes da tela e demais mudancas no preview
33
34        if (mHolder.getSurface() == null){

```

```

35         return;
36     }
37
38     // parando o preview antes de modifica-lo
39     try {
40         mCamera.stopPreview();
41     } catch (Exception e){
42         Log.d(TAG, "Problemas para parar o preview: " + e.getMessage());
43     }
44
45     // ajustar o preview
46
47     // reiniciar o preview
48     try {
49         mCamera.setPreviewDisplay(mHolder);
50         mCamera.startPreview();
51
52     } catch (Exception e){
53         Log.d(TAG, "Problemas para iniciar o preview: " + e.getMessage());
54     }
55 }
56 }

```

Código Java 11.18: Criando a View

Criação de um layout de visualização

Posteriormente vamos também associar a classe de visualização a um layout. É neste passo que vamos definir a tela que o usuário vai utilizar para capturar imagens/videos. A tela vai conter apenas o frame que vai comportar o vídeo e um botão de início de captura.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="horizontal"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6  >
7      <FrameLayout
8          android:id="@+id/camera_preview"
9          android:layout_width="fill_parent"
10         android:layout_height="fill_parent"
11         android:layout_weight="1"
12     />
13
14     <Button
15         android:id="@+id/button_capture"
16         android:text="Capture"
17         android:layout_width="wrap_content"
18         android:layout_height="wrap_content"
19         android:layout_gravity="center"
20     />
21 </LinearLayout>

```

Código Java 11.19: Criando o Layout

É interessante fixar a orientação da tela. Por uma questão de simplicidade vamos fazer isso adicionando as configuração a seguir no manifesto.

```

1  <activity android:name=".CameraActivity"
2      android:label="@string/app_name"
3
4      android:screenOrientation="landscape">
5      <!-- fixando a orientacao em retrato -->
6
7      <intent-filter>

```

```

8      <action android:name="android.intent.action.MAIN" />
9      <category android:name="android.intent.category.LAUNCHER" />
10     </intent-filter>
11 </activity>

```

Código Java 11.20: Ajustando a orientação da tela

Liberação de recurso

Depois de utilizar a câmera é preciso liberar o recurso para que outras aplicações possam utilizá-lo. Isto pode ser feito chamando o método `Camera.release()`. Observe o código abaixo:

```

1 public class CameraActivity extends Activity {
2     private Camera mCamera;
3     private SurfaceView mPreview;
4     private MediaRecorder mMediaRecorder;
5
6     ...
7
8     @Override
9     protected void onPause() {
10         super.onPause();
11         releaseMediaRecorder();
12         releaseCamera();
13     }
14
15     private void releaseMediaRecorder(){
16         if (mMediaRecorder != null) {
17             mMediaRecorder.reset();
18             mMediaRecorder.release();
19             mMediaRecorder = null;
20             mCamera.lock();
21         }
22     }
23
24     private void releaseCamera(){
25         if (mCamera != null){
26             mCamera.release();
27             mCamera = null;
28         }
29     }
30 }

```

Código Java 11.21: Liberando o recurso

Capturando imagem

Com a View e o Layout definidos estamos prontos para codificar a aplicação que captura imagem. Para isto vamos implementar a interface `Camera.PictureCallback` e sobrescrever o método `onPictureTaken()`. Quando a foto for tomada é este o método que será chamado. No exemplo abaixo, quando o método é chamado a imagem será armazenada.

```

1 private PictureCallback mPicture = new PictureCallback() {
2
3     @Override
4     public void onPictureTaken(byte[] data, Camera camera) {
5
6         File pictureFile = getOutputMediaFile(MEDIA_TYPE_IMAGE);
7         if (pictureFile == null){
8             Log.d(TAG, "Problemas na criação da mídia: " +
9                 e.getMessage());
10             return;
11         }
12     }
13 }

```

```

13     try {
14         FileOutputStream fos = new FileOutputStream(imageFile);
15         fos.write(data);
16         fos.close();
17     } catch (FileNotFoundException e) {
18         Log.d(TAG, "Arquivo não encontrado: " + e.getMessage());
19     } catch (IOException e) {
20         Log.d(TAG, "Problemas no acesso ao arquivo: " + e.getMessage());
21     }
22 }
23 };

```

Código Java 11.22: Definindo o callback

Em seguida temos que associar o botão definido no layout ao método `Camera.takePicture()`. É este método que de fato tira a foto, ele recebe como parâmetro o callback definido anteriormente.

```

1 // Adicionando um listener ao botão de captura
2 Button captureButton = (Button) findViewById(id.button_capture);
3 captureButton.setOnClickListener(
4     new View.OnClickListener() {
5         @Override
6         public void onClick(View v) {
7             // obtendo imagem da camera
8             mCamera.takePicture(null, null, mPicture);
9         }
10    }
11 );

```

Código Java 11.23: Adicionando o Listener

Capturando vídeo

Para realizar a captura de vídeo o processo é um pouco mais delicado e exige que cada passo seja feito em ordem. Nesta tarefa vamos utilizar, além da classe Câmera, a classe Media Recorder para armazenar o vídeo produzido.

Configurando o MediaRecorder Para armazenar o vídeo precisamos de um objeto MediaRecorder configurado apropriadamente. Esta configuração deve ser feita em uma ordem específica. Uma vez definidas as propriedades vamos chamar o método `MediaRecorder.prepare()` que fará uma validação da configuração e em seguida vai preparar o nosso recorder para gravação. Veja o exemplo abaixo:

```

1 private boolean prepareVideoRecorder(){
2
3     mCamera = getCameraInstance();
4     mMediaRecorder = new MediaRecorder();
5
6     mCamera.unlock();
7     mMediaRecorder.setCamera(mCamera);
8
9     mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.CAMCORDER);
10    mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
11
12    mMediaRecorder.setProfile(CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH));
13
14    mMediaRecorder.setOutputFile(getOutputMediaFile(MEDIA_TYPE_VIDEO).toString());
15
16    mMediaRecorder.setPreviewDisplay(mPreview.getHolder().getSurface());
17
18    try {
19        mMediaRecorder.prepare();
20    } catch (IllegalStateException e) {
21        Log.d(TAG, "IllegalStateException: " + e.getMessage());
22    }
23 }

```

```

22     releaseMediaRecorder();
23     return false;
24 } catch (IOException e) {
25     Log.d(TAG, "IOException: " + e.getMessage());
26     releaseMediaRecorder();
27     return false;
28 }
29 return true;
30 }

```

Código Java 11.24: Preparando o gravador de vídeo

Em seguida vamos associar um callback ao botão definido no layout. Novamente toda vez que a gravação começar ou terminar devemos executar os comandos em uma ordem específica como vemos a seguir:

START

- 1 - Desbloqueie a câmera Camera.unlock()
- 2 - Configure a uma instancia de MediaRecorder
- 3 - inicie a gravação MediaRecorder.start()
- 4 - Grave o vídeo

STOP

- 5 - Para a gravação usando MediaRecorder.stop()
- 6 - Libere o recurso do gravador MediaRecorder.release()
- 7 - Bloqueie a câmera usando Camera.lock()

Abaixo temos um exemplo de código que ilustra exatamente a ordem especificada.

```

1 private boolean isRecording = false;
2
3 // Adicionando um listener ao botao de captura
4 Button captureButton = (Button) findViewById(id.button_capture);
5 captureButton.setOnClickListener(
6     new View.OnClickListener() {
7         @Override
8         public void onClick(View v) {
9             if (isRecording) {
10                 mMediaRecorder.stop();
11                 releaseMediaRecorder();
12                 mCamera.lock();
13
14                 setCaptureButtonText("Capture");
15                 isRecording = false;
16             } else {
17                 if (prepareVideoRecorder()) {
18                     mMediaRecorder.start();
19
20                     setCaptureButtonText("Stop");
21                     isRecording = true;
22                 } else {
23                     releaseMediaRecorder();
24                 }
25             }
26         }
27     }
28 );

```

```

25         }
26     }
27 }
28 );

```

Código Java 11.25: Adicionando o listener ao botão de captura



Exercícios de Fixação

14 Crie um novo projeto Android. Use como nome para o projeto **Camera**. O nome do pacote deve ser **br.com.k19.android.cap11**, e o nome da *activity* deve ser **MainActivity**.

15 Modifique o arquivo **AndroidManifest.xml** seguindo o código a seguir:

```

1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2     package="br.com.k19.android.cap11"
3     android:versionCode="1"
4     android:versionName="1.0" >
5
6     <uses-sdk
7         android:minSdkVersion="8"
8         android:targetSdkVersion="15" />
9
10    <uses-permission android:name="android.permission.CAMERA" />
11    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
12
13    <application
14        android:icon="@drawable/ic_launcher"
15        android:label="@string/app_name"
16        android:theme="@style/AppTheme" >
17        <activity
18            android:name=".MainActivity"
19            android:label="@string/title_activity_main"
20            android:screenOrientation="landscape">
21            <intent-filter>
22                <action android:name="android.intent.action.MAIN" />
23
24                <category android:name="android.intent.category.LAUNCHER" />
25            </intent-filter>
26        </activity>
27    </application>
28
29 </manifest>

```

Código XML 11.16: AndroidManifest.xml

16 Modifique o arquivo **activity_main.xml** seguindo o código a seguir:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="horizontal"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7     <Button
8         android:id="@+id/button_photo"
9         android:text="@string/photo_button_label"
10        android:layout_width="wrap_content"

```

```

11     android:layout_height="wrap_content"
12     android:layout_gravity="center"
13     />
14
15     <Button
16     android:id="@+id/button_video"
17     android:text="@string/video_button_label"
18     android:layout_width="wrap_content"
19     android:layout_height="wrap_content"
20     android:layout_gravity="center"
21     />
22 </LinearLayout>

```

Código XML 11.17: activity_main.xml

17 Modifique o arquivo strings.xml seguindo o código a seguir:

```

1 <resources>
2
3     <string name="app_name">Camera</string>
4     <string name="hello_world">Hello world!</string>
5     <string name="menu_settings">Settings</string>
6     <string name="title_activity_main">MainActivity</string>
7     <string name="video_button_label">Capturar Video</string>
8     <string name="photo_button_label">Capturar Photo</string>
9
10 </resources>

```

Código XML 11.18: strings.xml

18 Modifique o arquivo MainActivity.java seguindo o código a seguir:

```

1 package br.com.k19.android.cap11;
2
3 import java.io.File;
4 import java.text.SimpleDateFormat;
5 import java.util.Date;
6
7 import br.com.k19.android.cap11.R.id;
8
9 import android.net.Uri;
10 import android.os.Bundle;
11 import android.os.Environment;
12 import android.provider.MediaStore;
13 import android.app.Activity;
14 import android.content.Intent;
15 import android.util.Log;
16 import android.view.Menu;
17 import android.view.View;
18 import android.widget.Button;
19 import android.widget.Toast;
20
21 public class MainActivity extends Activity {
22     public static final int MEDIA_TYPE_IMAGE = 1;
23     public static final int MEDIA_TYPE_VIDEO = 2;
24     private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
25     private static final int CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE = 200;
26     private Uri fileUri;
27
28     private static Uri getUri(int type){
29
30         File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
31             Environment.DIRECTORY_PICTURES), "CameraOutput");

```



```

32
33     if (! mediaStorageDir.exists()){
34         if (! mediaStorageDir.mkdirs()){
35             Log.d("CameraOutput", "nao foi possivel criar o diretorio");
36             return null;
37         }
38     }
39
40     String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
41     File mediaFile;
42     if (type == MEDIA_TYPE_IMAGE){
43         mediaFile = new File(mediaStorageDir.getPath() + File.separator +
44             "IMG_" + timeStamp + ".jpg");
45     } else if(type == MEDIA_TYPE_VIDEO) {
46         mediaFile = new File(mediaStorageDir.getPath() + File.separator +
47             "VID_" + timeStamp + ".mp4");
48     } else {
49         return null;
50     }
51
52     return Uri.fromFile(mediaFile);
53 }
54
55
56 @Override
57 public void onCreate(Bundle savedInstanceState) {
58     super.onCreate(savedInstanceState);
59     setContentView(R.layout.activity_main);
60
61     Button captureButton = (Button) findViewById(id.button_photo);
62     captureButton.setOnClickListener(
63         new View.OnClickListener() {
64             public void onClick(View v) {
65                 // Cria o Intent
66                 Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
67
68                 fileUri = getUri(MEDIA_TYPE_IMAGE);
69                 intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
70
71                 // Inicia o Intent
72                 startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
73             }
74         }
75     );
76
77     captureButton = (Button) findViewById(id.button_video);
78     captureButton.setOnClickListener(
79         new View.OnClickListener() {
80             public void onClick(View v) {
81                 // Cria o Intent
82                 Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
83
84                 fileUri = getUri(MEDIA_TYPE_VIDEO);
85                 intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
86                 intent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, 1);
87
88                 // Inicia o Intent
89                 startActivityForResult(intent, CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE);
90             }
91         }
92     );
93 }
94
95 @Override
96 public boolean onCreateOptionsMenu(Menu menu) {
97     getMenuInflater().inflate(R.menu.activity_main, menu);
98     return true;
99 }
100

```

```
101     @Override
102     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
103         if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
104             if (resultCode == RESULT_OK) {
105                 Toast.makeText(this, "Imagem salva em:\n" + fileUri.getPath(), Toast.LENGTH_LONG).show();
106             } else if (resultCode == RESULT_CANCELED) {
107                 // Cancelou a captura
108             } else {
109                 // Ocorreu algum problema nao esperado
110             }
111         }
112
113         if (requestCode == CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE) {
114             if (resultCode == RESULT_OK) {
115                 Toast.makeText(this, "Imagem salva em:\n" + fileUri.getPath(), Toast.LENGTH_LONG).show();
116             } else if (resultCode == RESULT_CANCELED) {
117                 // Cancelou a captura
118             } else {
119                 // Ocorreu algum problema nao esperado
120             }
121         }
122     }
123 }
```

Código Java 11.26: MainActivity.java

- 19 Rode o programa e teste a capture de imagem e vídeo.



Introdução

AppWidgets são pequenas visualizações de aplicações que podem ser inseridas em outras aplicações. Estas porções de aplicações são publicadas utilizando um provedor de App Widget. Vamos entrar no detalhe deste processo mais adiante.



Principais classes

Para criar um AppWidget vamos precisar de um objeto do tipo `AppWidgetProviderInfo`. Este objeto descreve os metadados do nosso AppWidget tais como: layout, update frequency, provedor, etc.

Outro objeto importante é uma implementação de `AppWidgetProvider` que vai definir os métodos básicos que farão interface, por meio de eventos, com o nosso Widget. É por ele que vamos receber mensagens informando quando o Widget foi atualizado, habilitado, deletado, etc.

Por último vamos precisar também de um layout, que é como vai parecer o nosso Widget.



Manifest

No Manifest vamos definir um elemento chamado receiver. É nele que vamos definir o provedor utilizando o atributo `android:name`.

```
1 <receiver android:name="ExampleAppWidgetProvider" >
2 </receiver>
```

Código Java 12.1: Definindo o receiver

Dentro do receiver vamos especificar duas informações: - Quais eventos o nosso Widget irá atender usando o elemento `intent-filter`; - Qual o nome e onde está o nosso `AppWidgetProviderInfo`

Observe abaixo como ficou nosso Manifest.

```
1 <receiver android:name="ExampleAppWidgetProvider" >
2   <intent-filter>
3     <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
4   </intent-filter>
5   <meta-data android:name="android.appwidget.provider"
6             android:resource="@xml/example_appwidget_info" />
7 </receiver>
```

Código Java 12.2: exemplo de Manifest

Note que o elemento `intent-filter` contém apenas a `action android.appwidget.action.APPWIDGET_UPDATE`, o que significa que o Widget só irá atender a eventos do tipo `update`. O Widget pode ser sensível a quatro tipos de eventos:

- `android.appwidget.action.ACTION_APPWIDGET_UPDATE`
- `android.appwidget.action.ACTION_APPWIDGET_DELETED`
- `android.appwidget.action.ACTION_APPWIDGET_ENABLED`
- `android.appwidget.action.ACTION_APPWIDGET_DISABLED`



Configurando o AppWidgetProviderInfo

Como vimos anteriormente, este objeto é responsável por definir as características do Widget. Para configurá-lo vamos criar um recurso do tipo `xml` e salvá-lo em `res/xml/`. Este `xml` possui apenas elemento `<appwidget-provider>`.

É dentro deste elemento que vamos adicionar as características que desejamos por meio de alguns atributos. Abaixo podemos observar algumas das propriedades mais importantes:

android:minWidth - Largura mínima que o Widget vai ocupar por padrão;

android:minHeight - Altura mínima que o Widget vai ocupar por padrão;

android:minResizeWidth - Largura mínima que o aplicativo deve ter sem comprometer a usabilidade;

android:minResizeHeight - Altura mínima que o aplicativo deve ter sem comprometer a usabilidade;

android:updatePeriodMillis - Frequência com que o Widget irá consultar o `AppWidgetProvider` para atualizações;

android:initialLayout - Atributo que faz referência ao recurso que define o layout do Widget;

android:configure - Atributo que define qual aplicação de configuração do Widget será aberta em sua primeira execução (opcional);

android:previewImage - Imagem do Widget sugerindo sua aparência depois de configurado. Se este atributo não for utilizado a imagem padrão será o ícone do Widget.

android:resizeMode - Define como o Widget pode ser redimensionado. O atributo pode assumir os seguintes valores: `"horizontal"`, `"vertical"`, `"horizontal|vertical"` ou `"none"`.



Definindo o Layout

Uma AppWidget é mais limitada quanto a quais *views* podem ser usadas. Os layouts disponíveis são `FrameLayout`, `LinearLayout` e `RelativeLayout`. As *views* disponíveis são `AnalogClock`, `Button`, `Chronometer`, `ImageButton`, `ImageView`, `ProgressBar` e `TextView`. A partir do Android 3.0 foram adicionadas as *views* `GridView`, `ListView`, `StackView`, `ViewFlipper` e `AdapterViewFlipper`.

A única interação possível com um Widget é através de um `OnClickListener`.



Exercícios de Fixação

- 1 Crie um novo projeto Android. Use como nome para o projeto **AppWidget**. O nome do pacote deve ser **br.com.k19.android.cap12**. Você pode criar o projeto sem nenhuma activity por padrão.
- 2 Na pasta **res/drawable** (senão existir, pode criar) crie um arquivo chamado **shape.xml**. Ele deve conter o seguinte conteúdo:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <shape xmlns:android="http://schemas.android.com/apk/res/android"
3     android:shape="rectangle" >
4
5     <stroke
6         android:width="2dp"
7         android:color="#FFFFFF" />
8
9     <gradient
10        android:angle="225"
11        android:endColor="#DD2ECCFA"
12        android:startColor="#DD000000" />
13
14     <corners
15        android:bottomLeftRadius="7dp"
16        android:bottomRightRadius="7dp"
17        android:topLeftRadius="7dp"
18        android:topRightRadius="7dp" />
19
20 </shape>

```

Código XML 12.1: shape.xml

Na pasta **res/layouts** crie um arquivo chamado **widget_layout.xml**. Ele deve conter o seguinte conteúdo:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:id="@+id/layout"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:layout_margin="8dp"
7     android:background="@drawable/shape" >
8
9     <TextView
10        android:id="@+id/update"
11        style="@android:style/TextAppearance.Large"
12        android:layout_width="match_parent"
13        android:layout_height="match_parent"
14        android:layout_gravity="center"
15        android:gravity="center_horizontal|center_vertical"
16        android:layout_margin="4dp" >
17
18 </TextView>

```

```
18
19 </LinearLayout>
```

Código XML 12.2: widget_layout.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```
1 <resources>
2
3     <string name="app_name">AppWidget</string>
4     <string name="number_label">Sorteado: %d</string>
5
6 </resources>
```

Código XML 12.3: strings.xml

Dentro da pasta **res/xml** (senão existir, pode criar) crie um arquivo chamado **widget_info.xml**, com o seguinte conteúdo:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
3     android:initialLayout="@layout/widget_layout"
4     android:minHeight="72dp"
5     android:minWidth="300dp"
6     android:updatePeriodMillis="300000" >
7
8 </appwidget-provider>
```

Código XML 12.4: widget_info.xml

Crie um arquivo **WidgetProvider.java** com o seguinte conteúdo:

```
1 package br.com.k19.android.cap12;
2
3 import java.util.Random;
4
5 import android.app.PendingIntent;
6 import android.appwidget.AppWidgetManager;
7 import android.appwidget.AppWidgetProvider;
8 import android.content.ComponentName;
9 import android.content.Context;
10 import android.content.Intent;
11 import android.widget.RemoteViews;
12
13 public class WidgetProvider extends AppWidgetProvider {
14
15     private static final String ACTION_CLICK = "ACTION_CLICK";
16
17     @Override
18     public void onUpdate(Context context, AppWidgetManager appWidgetManager,
19         int[] appWidgetIds) {
20
21         ComponentName thisWidget = new ComponentName(context,
22             WidgetProvider.class);
23         int[] allWidgetIds = appWidgetManager.getAppWidgetIds(thisWidget);
24
25         for (int widgetId : allWidgetIds) {
26
27             int number = (new Random().nextInt(100));
28
29             RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
30                 R.layout.widget_layout);
31
32             remoteViews.setTextViewText(R.id.update, // String.valueOf(number));
33                 context.getString(R.string.number_label, number));
```

```
34
35     Intent intent = new Intent(context, WidgetProvider.class);
36
37     intent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
38     intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, appWidgetIds);
39
40     PendingIntent pendingIntent = PendingIntent.getBroadcast(context,
41         0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
42     remoteViews.setOnClickPendingIntent(R.id.update, pendingIntent);
43     appWidgetManager.updateAppWidget(widgetId, remoteViews);
44 }
45 }
46 }
```

Código Java 12.3: MainActivity.java

Edite o **AndroidManifest.xml** e deixe-o igual ao exemplo abaixo:

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2     package="br.com.k19.android.cap12"
3     android:versionCode="1"
4     android:versionName="1.0" >
5
6     <uses-sdk
7         android:minSdkVersion="15"
8         android:targetSdkVersion="15" />
9
10    <application
11        android:icon="@drawable/ic_launcher"
12        android:label="@string/app_name"
13        android:theme="@style/AppTheme" >
14        <receiver android:name=".WidgetProvider" >
15            <intent-filter>
16                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
17            </intent-filter>
18
19            <meta-data
20                android:name="android.appwidget.provider"
21                android:resource="@xml/widget_info" />
22        </receiver>
23    </application>
24
25 </manifest>
```

Código XML 12.5: AndroidManifest.xml

Após isso, rode a aplicação. Você irá precisar instalar o widget na Home Screen para poder ver o resultado.



APÊNDICE - PUBLICANDO NO *Google Play*

Após fazer um aplicativo, você pode disponibilizá-lo na loja de aplicativos oficial do Android, a *Google Play*. A partir de lá outros usuários podem baixar o seu aplicativo. Para isso, antes é necessário que o desenvolvedor se cadastre para poder publicar aplicativos. Ao se inscrever, é necessário pagar uma taxa de US \$15,00 e ter (ou criar) uma *google account*.



Mais Sobre

Caso você queira fazer a inscrição, acesse o endereço abaixo e siga os passos.

<https://play.google.com/apps/publish>



Como gerar um aplicativo

Para subir um aplicativo no *Play* é necessário antes gerar um pacote do nosso aplicativo. O pacote é um arquivo com a extensão **.apk**. É possível gerar um pacote assinado ou não assinado. Um pacote não assinado é um pacote cuja autoria do desenvolvedor não pode ser identificada. Um pacote assinado é um pacote que inclui uma assinatura no pacote, de acordo com um certificado que apenas o desenvolvedor possui. Esse certificado não deve ser compartilhado com mais ninguém que não seja o desenvolvedor, senão outras pessoas podem assinar aplicativos como se fosse o próprio desenvolvedor e utilizar isto para fins maliciosos.

Por padrão, o Eclipse usa um certificado para debug (a *debug.keystore*) que é utilizado durante o processo de desenvolvimento para diminuir o trabalho de ter que gerar sempre um APK assinado. Quando for criar um APK para distribuir o seu aplicativo, é recomendado que seja utilizado um certificado diferente do que é utilizado durante o desenvolvimento.



Exercícios de Fixação

- 1 Vamos criar um pacote assinado. Selecione qualquer um dos projetos Android que estejam finalizados no Eclipse. Clique com o botão direito sobre o projeto, e selecione a opção **Android Tools** -> **Export Signed Application Package**. Ao selecionar, irá aparecer a tela abaixo:

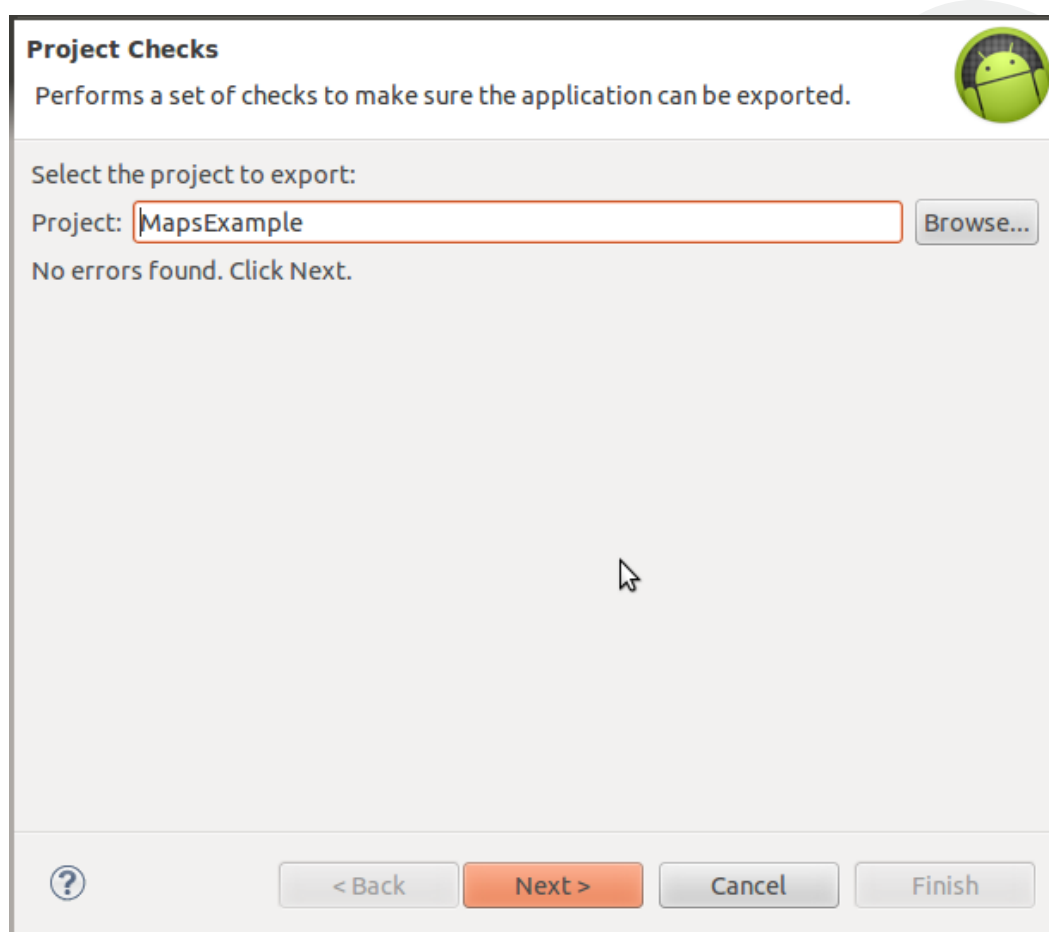
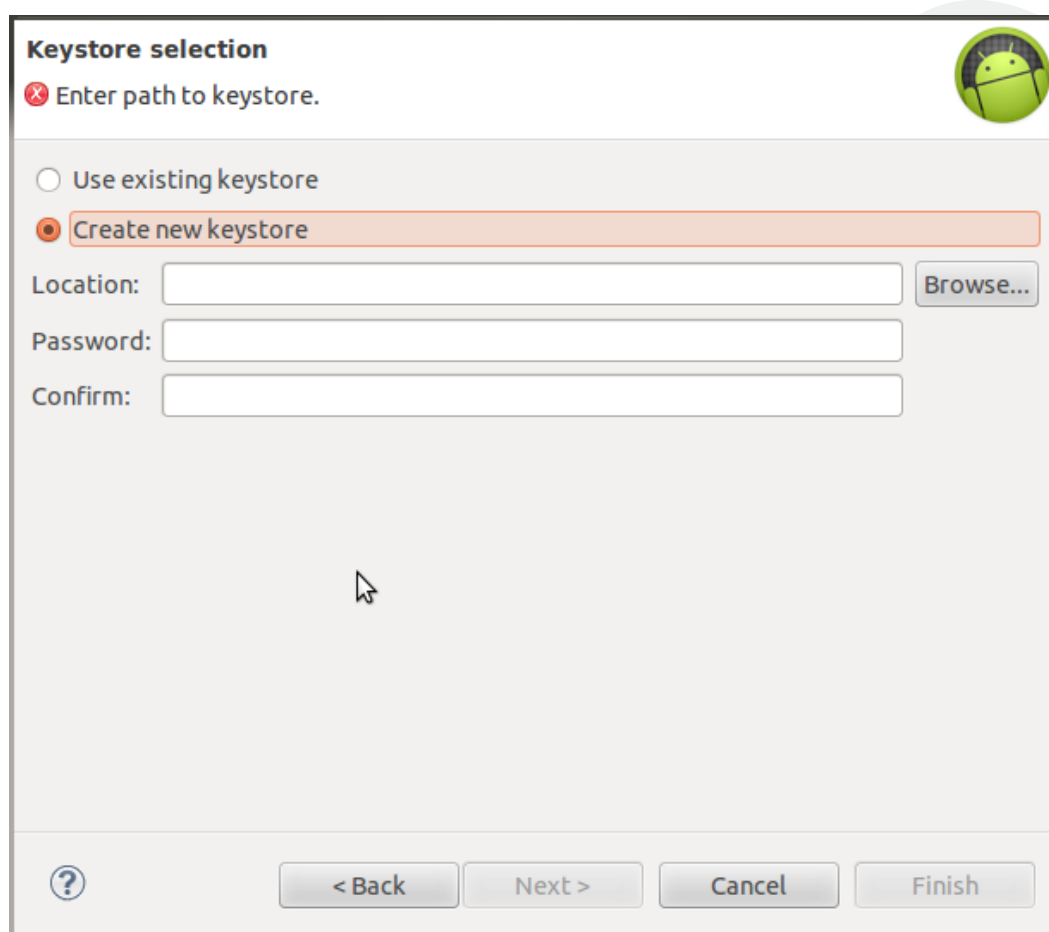


Figura 13.1: Wizard para criar pacotes assinados.

No momento que você gera um apk assinado, o Eclipse irá rodar o Android Lint, que irá verificar por erros na sua aplicação. Vários *warnings* durante o desenvolvimento são considerados erros neste passo. Imagens faltando na pasta *drawable*, strings de tradução faltando em alguma das versões do *strings.xml* são exemplos de erros comuns. Caso seja acusado algum erro, você deve arrumá-los antes de gerar o apk.

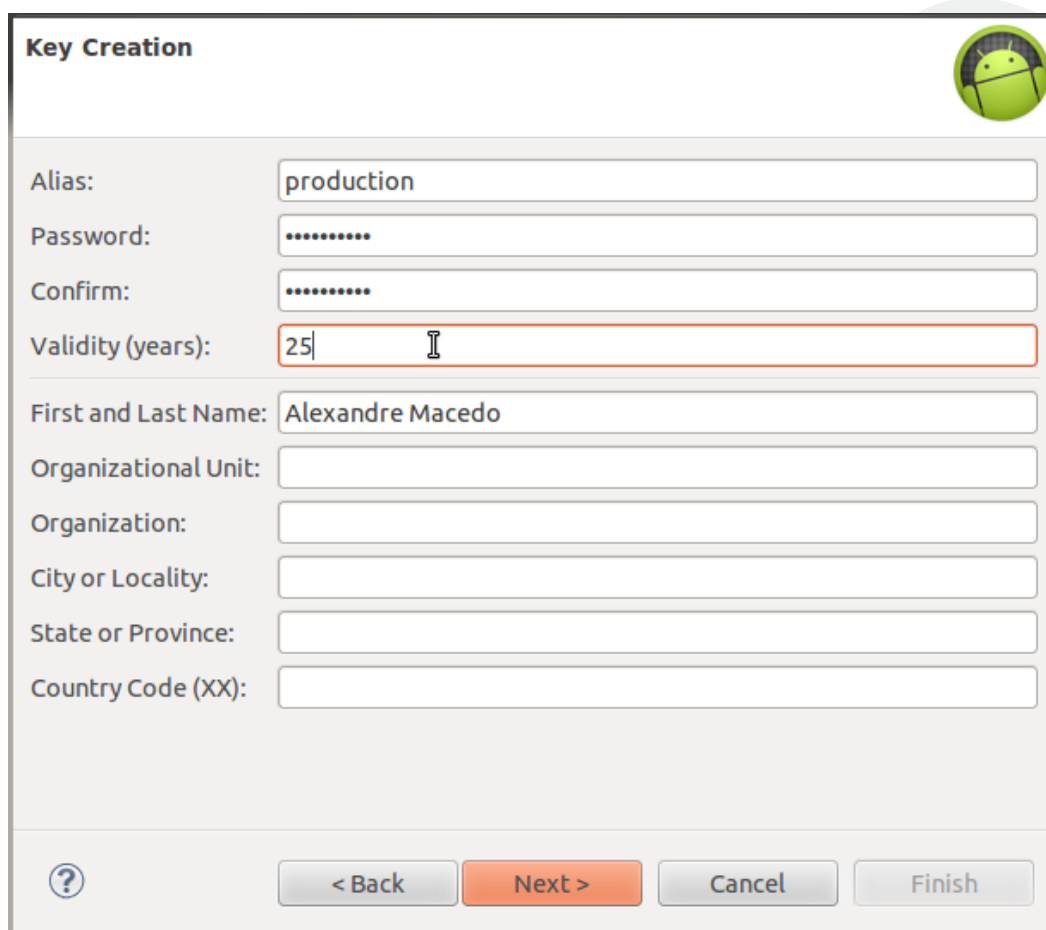
Depois que você corrigir todos os problemas, ou caso não tenha nenhum problema, você pode avançar para a tela seguinte, igual ao exemplo abaixo.



The image shows a 'Keystore selection' dialog box with an Android logo in the top right corner. At the top, there is a red 'X' icon and the text 'Enter path to keystore.'. Below this, there are two radio buttons: 'Use existing keystore' (unselected) and 'Create new keystore' (selected). The 'Create new keystore' option is highlighted with an orange border. Below the radio buttons, there are three text input fields labeled 'Location:', 'Password:', and 'Confirm:'. To the right of the 'Location:' field is a 'Browse...' button. At the bottom of the dialog, there is a row of four buttons: a help button (question mark icon), '< Back', 'Next >', 'Cancel', and 'Finish'. A mouse cursor is visible over the 'Next >' button.

Figura 13.2: Wizard para criar pacotes assinados.

Você deve criar uma nova *keystore* e selecionar uma senha. Escolha um nome e local para salvar a *keystore* e digite uma senha segura, que não seja fácil de adivinhar. Depois disso, continue e clique em *Next*.



The image shows a 'Key Creation' wizard dialog box. It has a title bar with the text 'Key Creation' and an Android robot icon. The dialog contains several input fields: 'Alias' with the value 'production', 'Password' and 'Confirm' fields with masked characters (dots), 'Validity (years)' with the value '25', 'First and Last Name' with the value 'Alexandre Macedo', and empty fields for 'Organizational Unit', 'Organization', 'City or Locality', 'State or Province', and 'Country Code (XX)'. At the bottom, there is a help icon (question mark) and four buttons: '< Back', 'Next >' (highlighted in orange), 'Cancel', and 'Finish'.

Figura 13.3: Wizard para criar pacotes assinados.

Nesta tela você deve escolher um nome para a chave (uma *keystore* pode armazenar mais de uma chave), e uma senha de acesso só para esta chave. Novamente, escolha um nome que você irá se lembrar depois, e digite uma senha segura. Você deve escolher também por quantos anos esta chave será válida antes de expirar. O recomendado é 25 anos. Por último, você deve preencher as informações do desenvolvedor (pelo menos o nome é obrigatório). Depois de preenchidos, pode continuar.

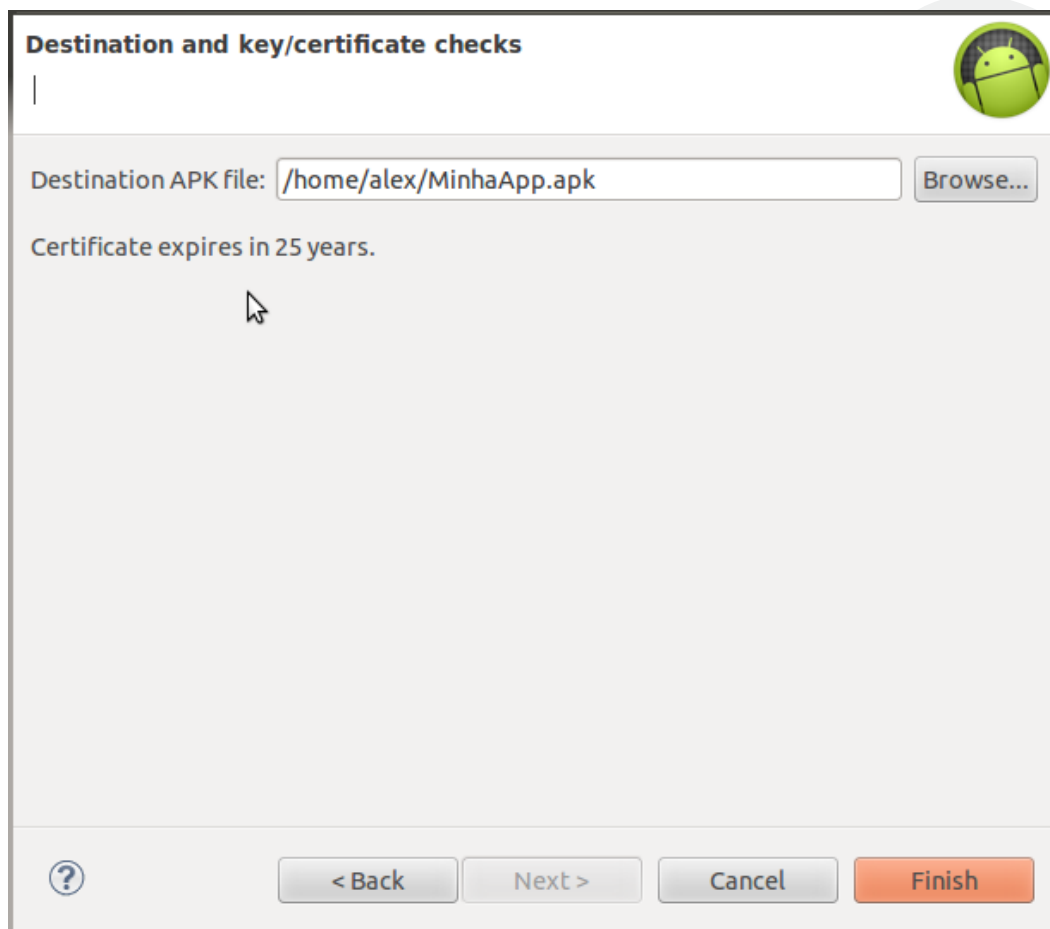


Figura 13.4: Wizard para criar pacotes assinados.

Por último deve ser escolhido o local onde o apk será salvo. Escolhar uma pasta de sua preferência e clique para concluir.



Mais Sobre

Agora que foi criado uma *keystore* e uma chave, você pode utilizá-las nas próximas vezes que for gerar um novo apk assinado. Neste caso, basta apenas deixar marcado a opção *Use existing keystore* quando for gerar um novo pacote.



APÊNDICE - Sensores



O Básico

O framework de sensores do Android fornece acesso a uma série de leitores disponibilizando ferramentas de medições para sensores baseados em hardware e software. É preciso lembrar que nem todo o dispositivo possui todos os tipos de sensores e dependendo do que for utilizado pode restringir muito o número de dispositivos candidatos a instalar o aplicativo. Outro detalhe importante é que alguns dispositivos podem ter até mais do que apenas um sensor do mesmo tipo.

Tipos de sensores:

ACCELEROMETER - Sensor de Aceleração (inclui gravidade).

AMBIENT_TEMPERATURE - Sensor de temperatura.

GRAVITY - Sensor de gravidade.

GYROSCOPE - Sensor de Aceleração (não inclui a gravidade).

LIGHT - Sensor de LUZ.

LINEAR_ACCELERATION - Sensor de aceleração linear.

MAGNETIC_FIELD - Sensor de campo magnético.

ORIENTATION - Sensor de orientação.

PRESSURE - Sensor de pressão.

PROXIMITY - Sensor de proximidade

RELATIVE_HUMIDITY - Sensor de umidade relativa.

ROTATION_VECTOR - Sensor de rotação.

Classes importantes

Sensor - Classe que representa um sensor;

SensorManager - Classe que dá acesso aos sensores do dispositivo, associa listeners a eventos além de facilitar contatos fornecendo uma série de constantes, por exemplo: a gravidade da Terra, de Marte, a luminosidade em uma noite de lua cheia, o campo magnético da Terra, pressão atmosférica no nível do mar entre outros.

SensorEvent - Classe que representa a leitura de um sensor, fornecendo, por exemplo, a precisão, o timestamp do registro, o tipo do sensor além da própria medição registrada.

SensorEventListener - Classe que é notificada das mudanças nos valores de leitura e precisão dos sensores

Obtendo a instância de um sensor

Em primeiro lugar é preciso obter o `SensorManager`. Isto pode ser feito chamando o método `getSystemService()` passando o parâmetro `Context.SENSOR_SERVICE`. Em seguida podemos, através do `SensorManager`, obter uma instância de `Sensor` executando `SensorManager.getDefaultSensor()` passando o tipo do sensor desejado, por exemplo, `Sensor.TYPE_ACCELEROMETER`.

Veja o trecho de código que obtém a instância de um `Sensor`

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
4 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Código Java 14.1: Obtenda instância de um Sensor

Recebendo os eventos do Sensor

Usando o `SensorManager`, vamos associar uma instância que implemente `SensorEventListener` ao sensor que se queira receber leituras incluindo também no parâmetro a frequência de recebimento das informações. Esta associação é feita executando o método `SensorManager.registerListener()`.

O listener deve implementar dois métodos:

SensorEventListener.onSensorChanged() - chamado quando houve uma mudança nos valores do sensor.

SensorEventListener.onAccuracyChanged() - chamado quando houve uma alteração na precisão do sensor.

No exemplo abaixo o listener é a própria atividade.

```
1 public class SensorTest extends Activity implements SensorEventListener {  
2  
3     private SensorManager mSensorManager;  
4     private Sensor mSensor;  
5  
6     protected void onCreate() {  
7         super.onCreate();  
8         ...  
9  
10        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
11        mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);  
12        mSensorManager.registerListener(this, mSensor, SensorManager.SENSOR_DELAY_NORMAL);  
13    }  
14  
15    public void onSensorChanged(SensorEvent event){  
16        // utilização dos dados lidos do sensor  
17    }  
18  
19    public void onAccuracyChanged(SensorEvent event, int accuracy){  
20        // resposta à modificação de precisão  
21    }  
22 }
```


Código Java 14.2: Associando eventos a listeners

O Parâmetro que define a periodicidade de recebimento de eventos do sensor, em milisegundos, é um inteiro que serve apenas como referência para o sistema. Os eventos podem acontecer antes ou depois do especificado. Para facilitar o framework possui intervalos pré-definidos em constantes: `SENSOR_DELAY_NORMAL`, `SENSOR_DELAY_UI`, `SENSOR_DELAY_GAME`, ou `SENSOR_DELAY_FASTEST`.



Sensores de Movimento

A API do Android fornece suporte a vários tipos de sensores de movimento do dispositivo. Os tipos são classificados em dois grupos: sensores baseados em hardware e sensores baseados em software. O acelerômetro e o giroscópio são sempre baseados em hardware enquanto os sensores de gravidade, aceleração linear e de vetores de rotação podem ser tanto por hardware quanto por software.

Os sensores por software baseiam seus cálculos utilizando os sensores de hardware disponíveis, portanto, sua disponibilidade é fortemente dependente dos recursos de hardware presentes do dispositivo.

Formato dos valores

Todos os sensores de movimento fornecem valores multidimensionais, ou seja, fazendo a leitura do acelerômetro receberemos um array de float com três valores, um para cada eixo (x, y e z). No caso do sensor de rotação um quarto componente com um valor escalar compõe a resposta da leitura.

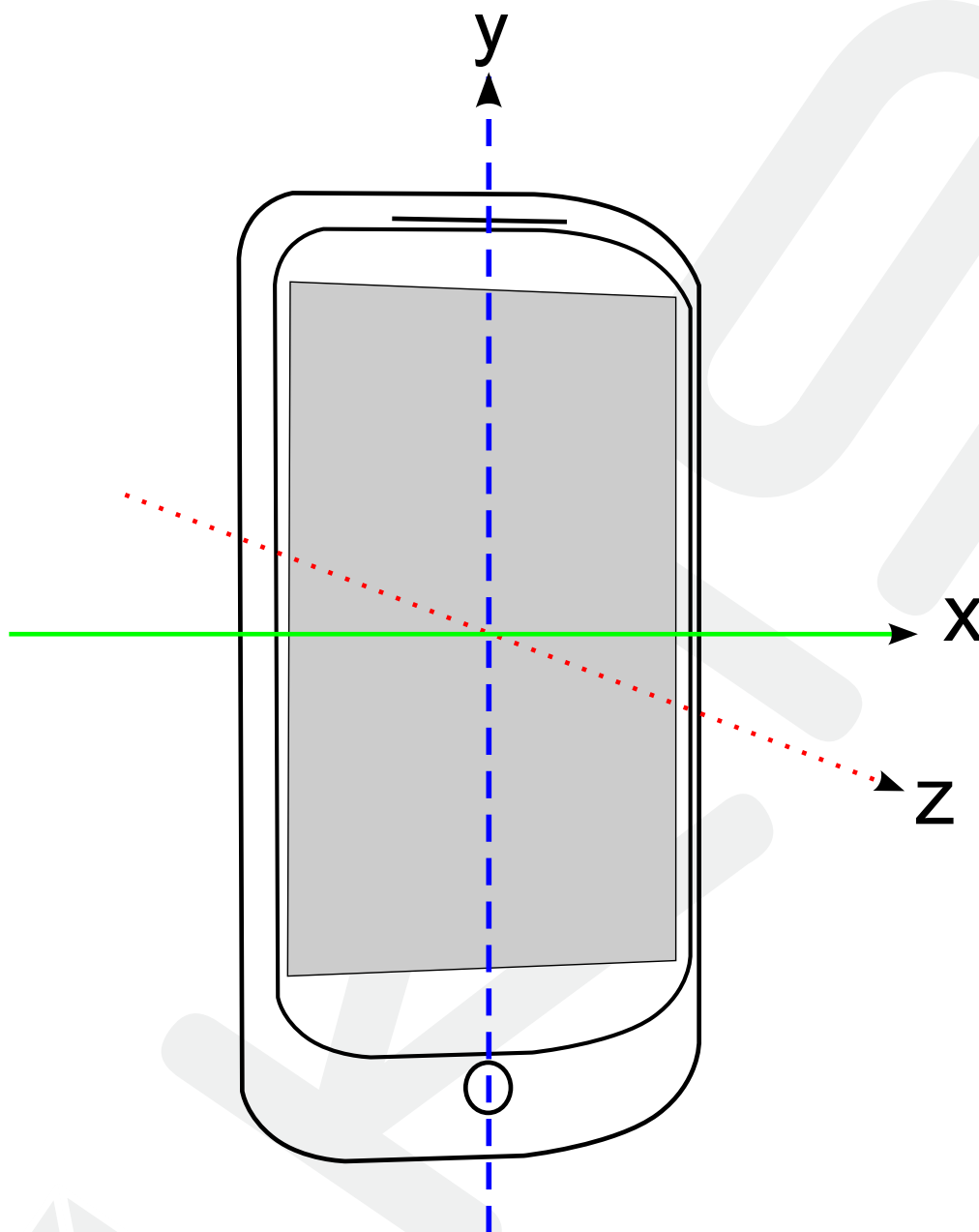


Figura 14.1: Resultado da tela.



Mais Sobre

Para saber mais sobre os tipos de sensores de movimento suportados e quais os valores retornados. Acesse: http://developer.android.com/guide/topics/sensors/sensors_motion.html

Acelerômetro

O Acelerômetro, como pode-se deduzir do próprio nome, mede a aceleração aplicada no dispositivo, incluindo a força da gravidade. Quando o celular está em repouso em uma mesa, por exemplo, a aceleração medida é a da gravidade e quando o dispositivo está em queda livre a aceleração medida é zero. A unidade de medida é m/s^2 . Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Código Java 14.3: Exemplo de uso do Acelerômetro

Sensor de Gravidade

O sensor de gravidade fornece um vetor tridimensional contendo a direção e a amplitude da gravidade. Os valores são medidos em m/s^2 . Quando o dispositivo está em repouso, o sensor de gravidade faz a mesma leitura que o acelerômetro. Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY);
```

Código Java 14.4: Exemplo de uso do Sensor de Gravidade

Giroscópio

O Giroscópio mede a rotação em rad/s em torno dos eixos x, y e z do dispositivo. Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
```

Código Java 14.5: Exemplo de uso do Giroscópio

Acelerômetro Linear

Este sensor fornece uma medida, em m/s^2 , da aceleração em torno de cada eixo (x, y, e z) excluindo-se a aceleração da gravidade. Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
```

Código Java 14.6: Exemplo de uso do Acelerômetro Linear

Sensor de Vetor de Rotação

Um evento deste sensor representa a orientação do dispositivo como combinação de um ângulo e eu eixo. Os elementos de um vetor de rotação não possuem unidade. São quatro informações resultantes de uma leitura: um vetor tridimensional (x, y e z) e um valor representando a magnitude. Exemplo de utilização:

```
1 private SensorManager mSensorManager;
```

```
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);
```

Código Java 14.7: Exemplo de uso do Sensor de Rotação



Sensores de Posição

A API do Android provê também dois tipos de sensores de posicionamento: o sensor de campo geomagnético e o de orientação. Além destes, existe um sensor que determina distância chamado sensor de proximidade. O sensor de orientação é baseado em software e está em desuso desde o Android 2.2.

Assim como os sensores de movimento, os sensores geomagnético e de orientação devolvem um vetor tridimensional, ou seja, são floats representando as três dimensões: x, y e z. Já o sensor de proximidade o retorno é apenas um valor representando a distância.



Mais Sobre

Para saber mais sobre os tipos de sensores de posicionamento suportados e quais os valores retornados. Acesse: http://developer.android.com/guide/topics/sensors/sensors_position.htm

Sensor de Orientação

Este sensor fornece a posição do dispositivo relativo ao norte magnético da Terra. O tipo do retorno é em graus. Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);
```

Código Java 14.8: Exemplo de uso do Sensor de Orientação

Sensor de Campo Geomagnético

Permite acompanhar as mudanças no campo magnético da Terra. Em geral a informação deste sensor deve ser utilizada em conjunto com os sensores de rotação ou o acelerômetro. O tipo de dado retornado pelo sensor é μT . Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
```

Código Java 14.9: Exemplo de uso do Sensor de Campo Geomagnético

Sensor de Proximidade

O sensor de proximidade determina o quão distante está um objeto do dispositivo. A informação da distância é retornada em um float em centímetros. Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
```

Código Java 14.10: Exemplo de uso do Sensor de Proximidade

Alguns sensores de proximidade podem devolver valores binários indicando apenas se o objeto está longe ou perto.



Sensores de Ambiente

A API do Android ainda fornece suporte a sensores de monitoração de ambiente tais como: temperatura, pressão, luminância, umidade. Este tipo de sensor é o de maior facilidade de manipulação pois não requer ajustes de calibração, modificações e seus dados podem ser utilizados diretamente. Observe abaixo a lista dos sensores e os tipos de dados que eles fornecem.



Mais Sobre

Para saber mais sobre os tipos de sensores de ambiente suportados e quais os valores retornados. Acesse: http://developer.android.com/guide/topics/sensors/sensors_environment.html

Sensor de Luz

Obtém do ambiente a luminância atual. A informação é medida em lux (lx). Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Código Java 14.11: Exemplo de uso do Sensor de Luz

Sensor de Pressão

Obtém do ambiente a pressão atmosférica. A informação é medida em hectopascal (hPa). Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE);
```

Código Java 14.12: Exemplo de uso do Sensor de Pressão

Sensor de Temperatura

Obtém a temperatura ambiente em graus Celcius (°C). Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_AMBIENT_TEMPERATURE);
```

Código Java 14.13: Exemplo de uso do Sensor de Temperatura

Existe um tipo de sensor de temperatura (TYPE_TEMPERATURE) que mede a temperatura do dispositivo mas entrou em desuso com o Android 4.0.

Sensor de Umidade Relativa

Obtém a umidade relativa do ambiente em porcentagem (temperatura ambiente é possível, por exemplo, calcular a umidade absoluta. Exemplo de utilização:

```
1 private SensorManager mSensorManager;  
2 private Sensor mSensor;  
3 ...  
4 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
5 mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_RELATIVE_HUMIDITY);
```

Código Java 14.14: Exemplo de uso do Sensor de Umidade Relativa



Exercícios de Fixação

- 1 Crie um novo projeto Android. Use como nome para o projeto **Sensores**. O nome do pacote deve ser **br.com.k19.android.animation**, e o nome da *activity* deve ser **MainActivity**.
- 2 Modifique o arquivo `activity_main.xml` seguindo o código a seguir:

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
2   xmlns:tools="http://schemas.android.com/tools"  
3   android:layout_width="match_parent"  
4   android:layout_height="match_parent" >  
5  
6       <TextView  
7         android:layout_width="wrap_content"  
8         android:layout_height="wrap_content"  
9         android:layout_marginLeft="100dp"  
10        android:layout_marginTop="80dp"  
11        android:text="X: "  
12        tools:context=".MainActivity" />  
13  
14       <TextView  
15         android:layout_width="wrap_content"  
16         android:layout_height="wrap_content"  
17         android:layout_marginLeft="100dp"  
18         android:layout_marginTop="110dp"  
19         android:text="Y: "  
20         tools:context=".MainActivity" />  
21  
22       <TextView
```

```

23     android:layout_width="wrap_content"
24     android:layout_height="wrap_content"
25     android:layout_marginLeft="100dp"
26     android:layout_marginTop="140dp"
27     android:text="Z: "
28     tools:context=".MainActivity" />
29
30     <TextView
31         android:id="@+id/textView1"
32         android:layout_width="wrap_content"
33         android:layout_height="wrap_content"
34         android:layout_centerHorizontal="true"
35         android:layout_marginTop="80dp"
36         android:text="@string/acc_value1"
37         tools:context=".MainActivity" />
38
39     <TextView
40         android:id="@+id/textView2"
41         android:layout_width="wrap_content"
42         android:layout_height="wrap_content"
43         android:layout_centerHorizontal="true"
44         android:layout_marginTop="110dp"
45         android:text="@string/acc_value2"
46         tools:context=".MainActivity" />
47
48     <TextView
49         android:id="@+id/textView3"
50         android:layout_width="wrap_content"
51         android:layout_height="wrap_content"
52         android:layout_centerHorizontal="true"
53         android:layout_marginTop="140dp"
54         android:text="@string/acc_value3"
55         tools:context=".MainActivity" />
56
57 </RelativeLayout>

```

Código XML 14.1: activity_main.xml

3 Modifique o arquivo strings.xml seguindo o código a seguir:

```

1 <resources>
2     <string name="app_name">Sensores</string>
3     <string name="acc_value1">0</string>
4     <string name="acc_value2">0</string>
5     <string name="acc_value3">0</string>
6     <string name="menu_settings">Settings</string>
7     <string name="title_activity_main">MainActivity</string>
8 </resources>

```

Código XML 14.2: strings.xml

4 Modifique o arquivo MainActivity.java seguindo o código a seguir:

```

1 package br.com.k19.android.sensores;
2
3 import android.hardware.Sensor;
4 import android.hardware.SensorEvent;
5 import android.hardware.SensorEventListener;
6 import android.hardware.SensorManager;
7 import android.os.Bundle;
8 import android.app.Activity;
9 import android.content.Context;
10 import android.view.Menu;
11 import android.widget.TextView;

```

```
12
13 public class MainActivity extends Activity implements SensorEventListener {
14
15     private SensorManager mSensorManager;
16     private Sensor mSensor;
17
18     @Override
19     public void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22
23         mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
24         mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
25         mSensorManager.registerListener((SensorEventListener) this, mSensor, ←
26             SensorManager.SENSOR_DELAY_NORMAL);
27     }
28
29     @Override
30     public boolean onCreateOptionsMenu(Menu menu) {
31         getMenuInflater().inflate(R.menu.activity_main, menu);
32         return true;
33     }
34
35     public void onAccuracyChanged(Sensor arg0, int arg1) {
36         // TODO Auto-generated method stub
37     }
38
39     public void onSensorChanged(SensorEvent ev) {
40
41         TextView x = (TextView) findViewById(R.id.textView1);
42         TextView y = (TextView) findViewById(R.id.textView2);
43         TextView z = (TextView) findViewById(R.id.textView3);
44
45         x.setText(String.valueOf(ev.values[0]));
46         y.setText(String.valueOf(ev.values[1]));
47         z.setText(String.valueOf(ev.values[2]));
48     }
49 }
50 }
```

Código Java 14.15: MainActivity.java

- 5 Rode a aplicação e movimente o dispositivo.

APÊNDICE - WEB APPS COM WEBVIEW



Introdução

WebView é uma maneira de disponibilizar conteúdo web dentro da aplicação client. A classe WebView estende a classe View do Android e permite que uma página seja mostrada como parte do layout da atividade. É importante lembrar que esta visualização de páginas web não inclui os controles de navegação e barra de endereço como em um browser padrão.

Uma utilização comum para WebView poderia ser, por exemplo, incluir na aplicação conteúdo variável como licenças de uso, guias de usuário, change logs e etc. Nestes casos pode ser mais interessante desenvolver uma página web em formato compatível com dispositivos mobile ao invés de obter o conteúdo via serviço, parseá-lo e ajustá-lo ao layout do client Android.



Manifest

Para utilizar conteúdo web na aplicação é preciso explicitar o pedido de permissão de acesso no arquivo Manifest. Para isto adiciona a linha abaixo:

```
1 <uses-permission android:name="android.permission.INTERNET" />
```

Código XML 15.1: Permissão de acesso



Layout

Para adicionar uma WebView na aplicação é preciso incluir no layout da activity um elemento do tipo WebView. No exemplo abaixo a página web vai ocupar a tela inteira.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <WebView xmlns:android="http://schemas.android.com/apk/res/android"
3     android:id="@+id/webview"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6 />
```

Código XML 15.2: Elemento WebView



Carregando uma página

Depois de adicionado o elemento no layout da aplicação, carregar a página no WebView se torna uma tarefa muito simples. Basta executar o método `WebView.loadUrl()` passando o endereço da página a ser carregada. Como no código abaixo:

```
1 WebView myWebView = (WebView) findViewById(R.id.webview);
2 myWebView.loadUrl("http://www.k19.com.br");
```

Código Java 15.1: Carregando a url



Controlando a Navegação

Em geral, as páginas disponibilizadas via WebView não deveriam ter muitos links restringindo assim a navegação do usuário, afinal, apresentar páginas web na aplicação não faz dela um browser. Contudo, é possível que o usuário clique em um link que o levará para uma outra página. Quando isto acontece, o comportamento esperado do Android é abrir o browser padrão para exibir a página.

Para evitar que isto aconteça, podemos alterar este comportamento e manter a navegação dentro do WebView. Isto pode ser feito definindo o WebViewClient do WebView como demonstrado no trecho de código abaixo:

```
1 WebView myWebView = (WebView) findViewById(R.id.webview);
2 myWebView.setWebViewClient(new WebViewClient());
```

Código Java 15.2: Direcionando a navegação para o próprio WebView

É possível ainda decidir se o nosso WebView vai abrir o link ou vai delegar a tarefa ao browser padrão dada a url. Isto é especialmente interessante quando queremos abrir as páginas do próprio domínio na WebView, já que estariam em um formato para exibição apropriado para dispositivos móveis, e abrir páginas de domínios externos no web browser padrão. Para isto, basta estender a classe WebViewClient e sobrescrever o método WebViewClient.shouldOverrideUrlLoading(). O código abaixo é um exemplo de aplicação:

```
1 private class MyWebViewClient extends WebViewClient {
2     @Override
3     public boolean shouldOverrideUrlLoading(WebView view, String url) {
4         if (Uri.parse(url).getHost().equals("www.k19.com.br")) {
5             // Se o domínio for interno
6             return false;
7         }
8         // Se o domínio for externo então vamos direcionar para o browser padrão
9         Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
10        startActivity(intent);
11        return true;
12    }
13 }
14
15 Em seguida associe esta nova implementação como WebViewClient do WebView.
16
17 \begin{java}{Associando o WebViewClient customizado}
18 WebView myWebView = (WebView) findViewById(R.id.webview);
19 myWebView.setWebViewClient(new MyWebViewClient());
```

Código Java 15.3: Sobrescrevendo a url

O WebView ainda suporta a recuperação do histórico de navegação. Isto só é possível se mais de uma página foi visualizada através dele. Deste modo, é possível adicionar funcionalidades "Back" e "Forward" como as de um navegador padrão. Existem 4 métodos que auxiliam nestas funções:

WebView.goBack() - volta para a página anterior.

WebView.goForward() - avança para a próxima página.

WebView.canGoBack() - verifica se existe histórico anterior à página corrente.

WebView.canGoForward() - verifica se existe histórico posterior à página corrente.

O exemplo a seguir utiliza estes controles para voltar a navegação caso o usuário aperte o botão Back do dispositivo. Se não houver páginas no histórico então a aplicação fecha.

```
1 @Override
2 public boolean onKeyDown(int keyCode, KeyEvent event) {
3     // Checando se o botão apertado é o Back
4     if ((keyCode == KeyEvent.KEYCODE_BACK) && myWebView.canGoBack() {
5         myWebView.goBack();
6         return true;
7     }
8     // Caso contrário manter o comportamento padrão do botão
9     return super.onKeyDown(keyCode, event);
10 }
```

Código Java 15.4: Navegando pelo histórico



Associando código JavaScript a código Android

A API do Android também permite que um código do cliente Android seja chamado a partir de um código JavaScript. Isto é possível implementando o código que será chamado em uma classe que é adicionada ao contexto de execução do JavaScript. Chamamos esta classe de Interface.

No código abaixo vamos definir um método chamado showToasts que, quando executado, mostra uma janela contendo uma mensagem. Este método faz parte de uma classe cuja instância será adicionada ao contexto de execução do JavaScript. A ideia neste exemplo é substituir a função alert() do JavaScript pelo método showToasts();

```
1 public class JavaScriptInterface {
2     Context mContext;
3
4     /* Cria a interface guardando nela o contexto */
5     JavaScriptInterface(Context c) {
6         mContext = c;
7     }
8
9     /* Mostra a notificação */
10    public void showToast(String toast) {
11        Toast.makeText(mContext, toast, Toast.LENGTH_SHORT).show();
12    }
13 }
```

Código Java 15.5: Classe que implementa o método a ser chamado via JS

Note que, por conveniência, adicionamos o Context no construtor da classe para utilizá-lo durante a chamada do método.

Feita a implementação podemos associar uma instância de JavaScriptInterface ao WebView chamando o método WebView.addJavascriptInterface(). Este método recebe dois parâmetros: a instância e o nome da Interface como mostra o trecho de código abaixo:

```

1 WebView webView = (WebView) findViewById(R.id.webview);
2 webView.addJavascriptInterface(new JavaScriptInterface(this), "Android");

```

Código Java 15.6: Associação da Interface ao WebView

Com a Interface criada podemos acessar o método através do nome da Interface ("Android" neste caso) a partir do código JavaScript. O código a seguir executa uma chamada ao método `Android.showToast()` em resposta ao evento click do botão "Say hello".

```

1 <input type="button" value="Say hello" onClick="showAndroidToast('Hello Android!')" />
2
3 <script type="text/javascript">
4     function showAndroidToast(toast) {
5         Android.showToast(toast);
6     }
7 </script>

```

Código HTML 15.1: Código HTML e JS



Exercícios de Fixação

- 1 Crie um novo projeto Android. Use como nome para o projeto **WebView**. O nome do pacote deve ser **br.com.k19.android.cap12**, e o nome da *activity* deve ser **MainActivity**.
- 2 Na pasta **res/layouts** crie um arquivo chamado **main.xml**. Ele deve conter o seguinte conteúdo:

```

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:orientation="vertical" >
6
7     <WebView
8         android:id="@+id/webview"
9         android:layout_width="match_parent"
10        android:layout_height="match_parent" />
11
12 </LinearLayout>

```

Código XML 15.3: main.xml

O arquivo **res/values/strings.xml** deve ficar com o seguinte conteúdo:

```

1 <resources>
2
3     <string name="app_name">WebView</string>
4     <string name="menu_settings">Settings</string>
5     <string name="title_activity_main">Site K19</string>
6
7 </resources>

```

Código XML 15.4: strings.xml

Edite o arquivo **MainActivity.java** com o seguinte conteúdo:

```
1 package br.com.k19.android.cap12;  
2  
3 import android.app.Activity;  
4 import android.os.Bundle;  
5 import android.webkit.WebSettings;  
6 import android.webkit.WebView;  
7  
8 public class MainActivity extends Activity {  
9  
10     @Override  
11     public void onCreate(Bundle savedInstanceState) {  
12         super.onCreate(savedInstanceState);  
13         setContentView(R.layout.main);  
14  
15         WebView webView = (WebView) findViewById(R.id.webview);  
16         WebSettings webSettings = webView.getSettings();  
17         webSettings.setJavaScriptEnabled(true);  
18  
19         webView.loadUrl("http://k19.com.br");  
20     }  
21 }
```

Código Java 15.7: MainActivity.java

Adicione a permissão para internet no **AndroidManifest.xml**. Basta adicionar a seguinte linha:

```
1 <uses-permission android:name="android.permission.INTERNET" />
```

Código XML 15.5: AndroidManifest.xml

Após isso, rode a aplicação e veja o resultado.



APÊNDICE - BLUETOOTH

A plataforma do Android oferece suporte à Bluetooth, permitindo que dispositivos se comuniquem e troquem dados sem utilizar fios, apenas através do Bluetooth.



Classes utilizadas

Toda a API referente a Bluetooth se encontra no *package* **android.bluetooth**. Entre as classes importantes encontradas neste package temos:

BluetoothAdapter - representa um adapter, que é o ponto de partida para várias ações, como descobrir aparelhos, parear e transmitir dados.

BluetoothDevice - representa um aparelho conectado.

BluetoothSocket - representa um socket, que é um canal de comunicação entre os aparelhos.

BluetoothServerSocket - representa um socket do tipo servidor, que recebe diferentes requisições, necessário se quiser conectar com mais de um aparelho android.



Permissões

Existem duas permissões com relação a Bluetooth. A **BLUETOOTH** e a **BLUETOOTH_ADMIN**. A primeira é usada para realizar uma conexão por bluetooth, como iniciar uma conexão, aceitar uma conexão ou transferir dados. A segunda é usada para buscar por aparelhos e para alterar as configurações de bluetooth do aparelho.

```
1 <uses-permission android:name="android.permission.BLUETOOTH" />
2 <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Código XML 16.1: Exemplo



Usando o Bluetooth

É necessário antes ver se o aparelho possui suporte a bluetooth, e caso positivo, se o bluetooth está habilitado. Caso não esteja habilitado, é possível enviar o usuário para configurar e habilitar o bluetooth.

```
1 BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
2 if (mBluetoothAdapter != null) {
```

```

3 // verifica se o bluetooth está habilitado
4 if (!mBluetoothAdapter.isEnabled()) {
5     // envia o usuário para ativar o bluetooth
6     Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
7     startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
8 }
9 // restante do código
10 } else {
11     // sem suporte a bluetooth
12 }

```

Código Java 16.1: Exemplo



Listando dispositivos pareados

```

1 // retorna os dispositivos pareados
2 Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
3
4 if (pairedDevices.size() > 0) {
5     // iteramos nos dispositivos
6     for (BluetoothDevice device : pairedDevices) {
7         // adiciona em um adapter para mostrar em uma ListView, como exemplo
8         adapter.add(device.getName() + "\n" + device.getAddress());
9     }
10 }

```

Código Java 16.2: Exemplo



Descobrimos dispositivos

Para descobrir dispositivos, é necessário chamar o método **startDiscovery()** no *BluetoothAdapter*. O método é assíncrono, por isso é necessário registrar um *receiver* como no exemplo abaixo:

```

1 private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
2     public void onReceive(Context context, Intent intent) {
3         String action = intent.getAction();
4         // quando um aparelho é encontrado
5         if (BluetoothDevice.ACTION_FOUND.equals(action)) {
6             // obter o dispositivo do intent
7             BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
8             // adiciona a um adapter para mostrar em uma ListView
9             adapter.add(device.getName() + "\n" + device.getAddress());
10        }
11    }
12 };
13 // registra o BroadcastReceiver
14 IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
15 registerReceiver(mReceiver, filter);
16 // lembre-se de remover o registro no onDestroy

```

Código Java 16.3: Exemplo



Usando conexões

Abaixo está um exemplo de uma thread usada para conectar com outros aparelhos, sendo que o dispositivo funciona como o servidor.


```

1 private class AcceptThread extends Thread {
2     private final BluetoothServerSocket mmServerSocket;
3
4     public AcceptThread() {
5         BluetoothServerSocket tmp = null;
6         try {
7             // UUID é o identificador do aparelho, também usado pelo cliente
8             tmp = mBluetoothAdapter.listenUsingRfcommWithServiceRecord(NAME, MY_UUID);
9         } catch (IOException e) { }
10        mmServerSocket = tmp;
11    }
12
13    public void run() {
14        BluetoothSocket socket = null;
15        // continua tentando até encontra um socket ou uma exception
16        while (true) {
17            try {
18                socket = mmServerSocket.accept();
19            } catch (IOException e) {
20                break;
21            }
22            // conexão aceita
23            if (socket != null) {
24                // Faz algo com o socket em uma thread separada
25                manageConnectedSocket(socket);
26                mmServerSocket.close();
27                break;
28            }
29        }
30    }
31
32    public void cancel() {
33        try {
34            mmServerSocket.close();
35        } catch (IOException e) { }
36    }
37 }

```

Código Java 16.4: Exemplo

Abaixo está um exemplo de como conectar como um cliente a um servidor:

```

1 private class ConnectThread extends Thread {
2     private final BluetoothSocket mmSocket;
3     private final BluetoothDevice mmDevice;
4
5     public ConnectThread(BluetoothDevice device) {
6         BluetoothSocket tmp = null;
7         mmDevice = device;
8
9         // obtem um socket para conectar com um dispositivo
10        try {
11            // MY_UUID is the app's UUID string, also used by the server code
12            tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
13        } catch (IOException e) { }
14        mmSocket = tmp;
15    }
16
17    public void run() {
18
19        mBluetoothAdapter.cancelDiscovery();
20
21        try {
22            mmSocket.connect();
23        } catch (IOException connectException) {
24            // erro ao conectar
25            try {

```

```

26         mmSocket.close();
27     } catch (IOException closeException) { }
28     return;
29 }
30
31 // faz alguma coisa com o socket em uma thread separada
32 manageConnectedSocket(mmSocket);
33 }
34
35 public void cancel() {
36     try {
37         mmSocket.close();
38     } catch (IOException e) { }
39 }
40 }

```

Código Java 16.5: Exemplo

Quando você tem um ou mais aparelhos conectados, você pode começar a transferir dados entre eles através de um *socket*. Toda a transmissão de dados é feita utilizando a classe *InputStream* e *OutputStream*. Veja abaixo um exemplo de como transferir dados em uma thread.

```

1 private class ConnectedThread extends Thread {
2     private final BluetoothSocket mmSocket;
3     private final InputStream mmInStream;
4     private final OutputStream mmOutStream;
5
6     public ConnectedThread(BluetoothSocket socket) {
7         mmSocket = socket;
8         InputStream tmpIn = null;
9         OutputStream tmpOut = null;
10
11         try {
12             tmpIn = socket.getInputStream();
13             tmpOut = socket.getOutputStream();
14         } catch (IOException e) { }
15
16         mmInStream = tmpIn;
17         mmOutStream = tmpOut;
18     }
19
20     public void run() {
21         byte[] buffer = new byte[1024];
22         int bytes; // bytes returned from read()
23
24         // continua lendo o inputStream até ocorrer um erro
25         while (true) {
26             try {
27                 bytes = mmInStream.read(buffer);
28                 // envia o dado obtido para alguma activity via handler
29                 mHandler.obtainMessage(MESSAGE_READ, bytes, -1, buffer)
30                     .sendToTarget();
31             } catch (IOException e) {
32                 break;
33             }
34         }
35     }
36
37     public void write(byte[] bytes) {
38         try {
39             mmOutStream.write(bytes);
40         } catch (IOException e) { }
41     }
42
43     public void cancel() {
44         try {
45             mmSocket.close();
46         } catch (IOException e) { }

```

```
47     }  
48 }
```

Código Java 16.6: Exemplo



A partir do Android 3.0, a API Properties Animation foi introduzida e provê suporte para animações.

Basicamente a API permite que propriedades de um objeto sejam alteradas com o tempo. A classe básica da API é a Animator mas normalmente usa-se a classe ObjectAnimator. Outra classe importante é a classe AnimatorListener que pode ser utilizada para executar ações antes e depois de uma animação.

As animações podem ser utilizadas em Views ou em transições entre Activities.

Neste capítulo vamos implementar um exercício que mostra um exemplo de utilização da classe ObjectAnimator.



Exercícios de Fixação

1

Crie um novo projeto Android. Use como nome para o projeto **Animacao**. O nome do pacote deve ser **br.com.k19.android.animation**, e o nome da *activity* deve ser **MainActivity**. Não se esqueça que, para este app, a versão mínima do SDK é 11.

2

Modifique o arquivo AndroidManifest.xml seguindo o código a seguir:

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2   package="br.com.k19.android.animacao"
3   android:versionCode="1"
4   android:versionName="1.0" >
5
6   <uses-sdk
7     android:minSdkVersion="11"
8     android:targetSdkVersion="15" />
9
10  <application
11    android:icon="@drawable/ic_launcher"
12    android:label="@string/app_name"
13    android:theme="@style/AppTheme" >
14    <activity
15      android:name=".MainActivity"
16      android:label="@string/title_activity_main" >
17      <intent-filter>
18        <action android:name="android.intent.action.MAIN" />
19
20        <category android:name="android.intent.category.LAUNCHER" />
21      </intent-filter>
```

```
22     </activity>
23   </application>
24
25 </manifest>
```

Código XML 17.1: AndroidManifest.xml

3 Modifique o layout principal seguindo o código abaixo.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:id="@+id/layout"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical" >
7
8     <LinearLayout
9         android:id="@+id/test"
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content" >
12
13        <Button
14            android:id="@+id/Button01"
15            android:layout_width="wrap_content"
16            android:layout_height="wrap_content"
17            android:onClick="startAnimation"
18            android:text="Rotate" />
19
20    </LinearLayout>
21
22    <ImageView
23        android:id="@+id/imageView1"
24        android:layout_width="wrap_content"
25        android:layout_height="wrap_content"
26        android:layout_centerHorizontal="true"
27        android:layout_centerVertical="true"
28        android:src="@drawable/ic_launcher" />
29
30 </RelativeLayout>
```

Código XML 17.2: activity_main.xml

4 Altere a MainActivity para que fique como o código a seguir:

```
1 package br.com.k1.android.animacao;
2
3 import android.animation.ObjectAnimator;
4 import android.app.Activity;
5 import android.os.Bundle;
6 import android.view.Menu;
7 import android.view.View;
8 import android.widget.ImageView;
9
10 public class MainActivity extends Activity {
11
12     /* Chamado quando a Activity e criada. */
13     @Override
14     public void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17
18     }
19
20     public void startAnimation(View view) {
```

```
21     float dest = 0;
22     ImageView aniView = (ImageView) findViewById(R.id.imageView1);
23     if (view.getId() == R.id.Button01){
24         dest = 360;
25         if (aniView.getRotation() == 360) {
26             System.out.println(aniView.getAlpha());
27             dest = 0;
28         }
29         ObjectAnimator animation1 = ObjectAnimator.ofFloat(aniView,
30             "rotation", dest);
31         animation1.setDuration(2000);
32         animation1.start();
33     }
34
35 }
36
37 @Override
38 public boolean onCreateOptionsMenu(Menu menu) {
39     return super.onCreateOptionsMenu(menu);
40 }
41 }
```

Código Java 17.1: MainActivity.java

Rode a aplicação e clique no botão para ver o ícone da aplicação rotacionando.





Introdução

Uma das possibilidades para monetizar através de uma aplicação além de vendê-la é fazendo uso de publicidade. Neste capítulo vamos aprender sobre como inserir banners publicitários na aplicação.



Conta de Veículo

Para que os cliques no banner sejam contabilizados corretamente e associados a um veículo, é preciso abrir uma conta em uma rede de publicidade móvel. Neste capítulo, vamos utilizar um exemplo com AdMob, que é a rede de publicidade do Google.



SDK do serviço de Ad

Aberta a conta, o serviço da sua escolha deve ainda fornecer as bibliotecas que irão lhe ajudar no desenvolvimento da sua aplicação com publicidade. O Google fornece um lib para facilitar a inclusão de banners na aplicação. Lembre-se que os jar devem ser colocados na pasta **lib** e incluídos no Build-Path do Eclipse.

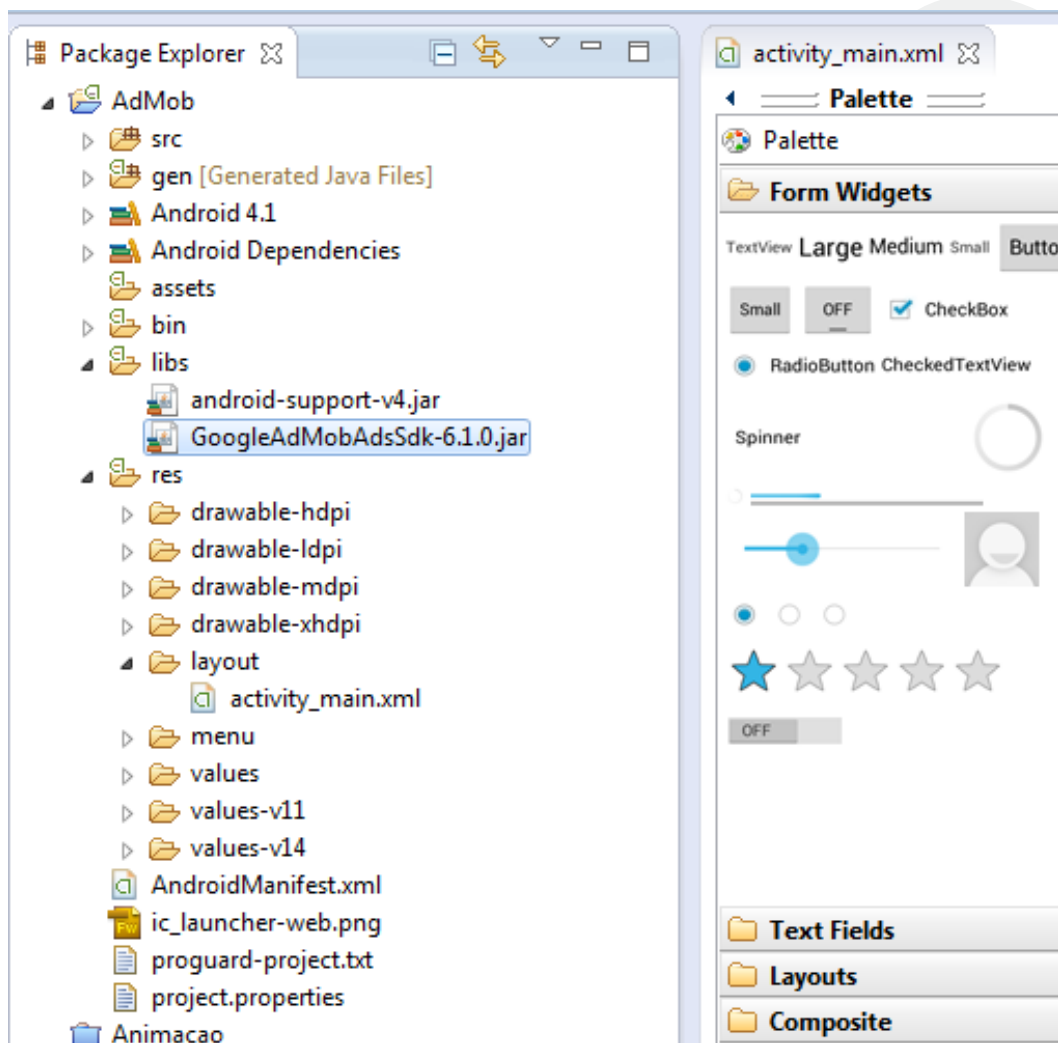


Figura 18.1: SDK do AdMob do Google.



Mais Sobre

Para saber mais sobre o AdMob e como fazer o download do SDK acesse: <http://developer.android.com/guide/appendix/media-formats.html>

Manifest

Os banners são preenchidos com conteúdo requisitado na web e para isso é preciso explicitar que o seu aplicativo irá acessar a internet.

```

1 </manifest>
2   <uses-permission android:name="android.permission.INTERNET" />
3   ...
4   <application>...</application>
5 </manifest>

```

Código Java 18.1: AndroidManifest.xml



Layout

Para posicionar o banner no aplicativo, basta inserir no layout um elemento do tipo `com.google.ads.AdView`. Observe o layout abaixo:

```

1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     android:id="@+id/ad_catalog_layout"
3     android:orientation="vertical"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent" >
6     <com.google.ads.AdView
7         xmlns:googleads="http://schemas.android.com/apk/lib/com.google.ads"
8         android:id="@+id/ad"
9         android:layout_width="fill_parent"
10        android:layout_height="wrap_content"
11        googleads:adSize="BANNER"
12        googleads:adUnitId="@string/admob_id" />
13     <TextView android:id="@+id/title"
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:text="@string/banner_top" />
17     <TextView android:id="@+id/status"
18         android:layout_width="match_parent"
19         android:layout_height="wrap_content" />
20 </LinearLayout>

```

Código XML 18.1: Layout.xml

Lembre-se de que estas configurações são específicas do AdMob e podem variar dependendo da rede de publicidade que está sendo utilizada.



Inicializando o Ad

Dependendo da arquitetura que foi utilizada na aplicação podemos inicializar o Ad no `Activity.onCreate` ou no `Fragment.onCreateView`. O código abaixo ilustra como o Ad pode ser inicializado:

```

1 public View onCreateView(LayoutInflater inflater, ViewGroup container,
2     Bundle savedInstanceState) {
3     ...
4     View v = inflater.inflate(R.layout.main, container, false);
5     mAdStatus = (TextView) v.findViewById(R.id.status);
6     mAdView = (AdView) v.findViewById(R.id.ad);
7     mAdView.setAdListener(new MyAdListener());
8
9     AdRequest adRequest = new AdRequest();
10    adRequest.addKeyword("sporting goods");
11    mAdView.loadAd(adRequest);
12    return v;
13 }

```

Código Java 18.2: Fragment.onCreateView()

Note que um listener foi associado à view. Quando disponível, o banner pode reportar alguns eventos, tais como, sucesso ou erro no carregamento. O listener pode ajudar quando precisamos tomar decisões frente a estes eventos. Um bom exemplo seria carregar um banner padrão caso houvesse um erro no carregamento a partir do servidor do Ad's. O código abaixo implementa um exemplo de listener:

```

1 private class MyAdListener implements AdListener {
2     ...

```

```
3
4  @Override
5  public void onFailedToReceiveAd(Ad ad, ErrorCode errorCode) {
6      mAdStatus.setText(R.string.error_receive_ad);
7  }
8
9  @Override
10 public void onReceiveAd(Ad ad) {
11     mAdStatus.setText("");
12 }
13 }
```

Código Java 18.3: Listener