Apostila de

Programando Passo a Passo
3ª Edição

De : Luciano Alves da Silva (lucianopascal@yahoo.com.br)

Apresentação

Android é uma plataforma aberta voltada para dispositivos móveis desenvolvida pela Google e atualmente é mantida pela Open Handset Alliance (OHA). Todas as aplicações desenvolvidas para essa plataforma utilizam à linguagem Java, o que facilita muitos programadores com conhecimentos em Java a desenvolver aplicações para essa plataforma.

Este material tem por objetivo mostrar de modo fácil como programar na plataforma para dispositivos móveis da Google (Android) usando o eclipse. Neste material vamos conhecer um pouco do histórico do Android, como surgiu, quais dispositivos suportam esse sistema operacional, como é a sua estrutura e como desenvolver diversos tipos de aplicações para Android por meio de vários programas e exemplos bem explicados.

Índice analítico

1) Introdução

O Android é uma plataforma desenvolvida pela Google voltada para dispositivos móveis. Em 5 de novembro de 2007, a empresa tornou pública a primeira plataforma Open Source de desenvolvimento para dispositivos moveis baseada na plataforma Java com sistema operacional Linux, na qual foi chamada de Android. Essa plataforma é mantida pela OHA (Open Handset Alliance), um grupo formado por mais de 40 empresas as quais se uniram para inovar e acelerar o desenvolvimento de aplicações, serviços, trazendo aos consumidores uma experiência mais rica em termos de recursos, menos dispendiosa em ternos financeiros para o mercado móvel. Pode-se dizer que a plataforma Android é a primeira plataforma móvel completa, aberta e livre.

Um dos SmartPhones que ofereceu suporte a esse sistema operacional foi o G1 da empresa T-Mobile. Veja a figura dele abaixo:

(G1 da T-Mobile)

Não demorou muito para que o Android chegasse aqui no Brasil. Hoje já contamos com operadoras como Claro, TIM e Vivo que já oferecem suporte a essa plataforma.

Os SmartPhones disponíveis aqui no Brasil, oferecidos por algumas dessas operadoras, que suportam o sistema Android é o Samgung Galaxy e o Motorola Milestone. Veja a figura desses SmartPhones abaixo:

(Samsung Galaxy)

(Motorola MileStone)

2) A estrutura geral da plataforma Android

Android é a plataforma open source para dispositivos móveis da Open Handset Alliance (OHA). O Android SDK é o kit de desenvolvimento que disponibiliza as ferramentas e APIs necessárias para desenvolver aplicações para a plataforma Android, utilizando a linguagem Java. Recursos :

- **Application framework** proporciona a reutilização e substituição de componentes

- **Dalvik virtual machine** otimizada para dispositivos móveis

- **Browser Integrado** baseado no webkit engine

- **Gráficos Otimizados** possui uma biblioteca 2D; e 3D baseada na especificação OpenGL ES 1.0 (aceleração de hardware é opcional)

- **SQLite** para guardar dados estruturados

- **Suporte multimídia** para áudio, vídeo e formatos de imagem (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

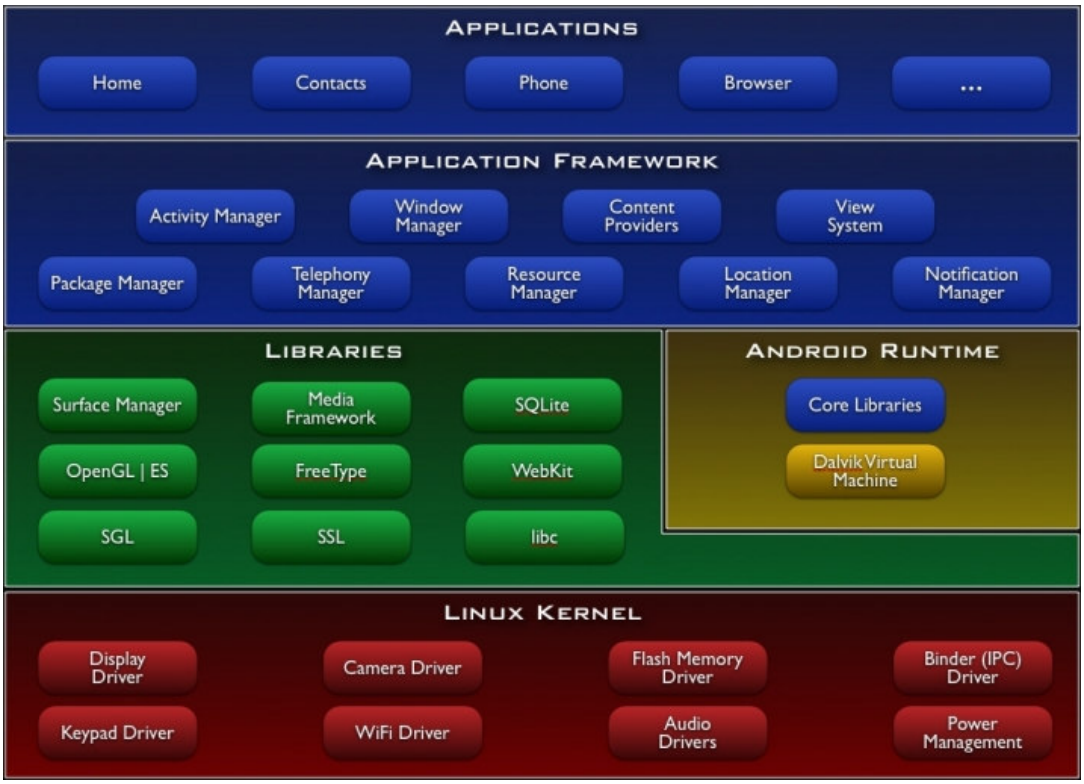
- **Telefonia GSM** (dependente de hardware)

- **Bluetooth, EDGE, 3G, e WiFi** (dependente de hardware)

- **Câmera, GPS, compasso, e acelerômetro** (dependente de hardware)

- **Rico ambiente de desenvolvimento** , incluindo um emulador de dispositivo, ferramentas de depuração, memória, performance e um plugin para o Eclipse (ADT)

2.1) A arquitetura do Android



(Arquitetura geral da plataforma Android)

2.2) Aplicações

Junto com o Android vem um conjunto de aplicações fundamentais. São elas:

- um cliente de e-mail;
- programa de SMS;
- agenda;
- mapas;
- navegador;
- contatos entre outros.

Todos os aplicativos implementados foram desenvolvidos na linguagem de programação Java.

2.3) Bibliotecas

O Android inclui um conjunto de bibliotecas C/C++ utilizadas por vários componentes do sistema. Estas capacidades são expostas para os desenvolvedores através do Framework. Abaixo, algumas das principais bibliotecas:

- **System C library** – uma implementação derivada da biblioteca C padrão sistema (libc) do BSD sintonizada para dispositivos rodando Linux.

- **Media Libraries** – baseado no PacketVideo's OpenCORE; as bibliotecas suportam os mais populares formatos de áudio e vídeo, bem como imagens estáticas.

- **Surface Manager** – gere o acesso ao subsistema de exibição bem como as múltiplas camadas de aplicações 2D e 3D;

- **LibWebCore** – um web browser engine utilizado tanto no Android Browser quanto para exibições web.

- **SGL – o engine de gráficos 2D**
- **3D libraries** – uma implementação baseada no OpenGL ES 1.0 APIs; as bibliotecas utilizam aceleração 3D via hardware (quando disponível) ou o software de renderização 3D altamente otimizado incluído no Android.

- **FreeType** – renderização de fontes bitmap e vector

- **SQLite** – um poderoso e leve engine de banco de dados relacional disponível para todas as aplicações

2.4) Android Runtime

O Android inclui um grupo de bibliotecas que fornece a maioria das funcionalidades disponíveis nas principais bibliotecas da linguagem Java.

Toda aplicação Android roda em seu próprio processo, com sua própria instância da máquina virtual Dalvik. O Dalvik foi escrito de forma a executar várias VMs eficientemente. Ele executa arquivos .dex, que é otimizado para consumo mínimo de memória. A VM é baseada em registros e roda classes compiladas pela linguagem Java que foram transformadas em arquivos .dex, através da ferramenta "dx" incluída no SDK.

O Dalvik VM baseia-se no kernel do Linux para funcionalidades subjacentes como o encadeamento e a gestão de baixo nível de memória.

2.5) Linux Kernel

Utiliza a versão 2,6 do kernel do Linux para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos, etc. O kernel também atua como uma camada de abstração entre o hardware e o resto do software.

3) Instalando o Eclipse e o Android

Para a elaboração desse material, eu fiz o uso do Eclipse Galileo (Eclipse 3.5.1 para Windows) e o SDK do Android Revisão 5 e o plugin do Android para o Eclipse ADT-0.9.6. Qualquer versão (de preferência superior) dos programas citados acima serve. Claro , para que toda essa aplicação funcione é necessário que você tenha instalado antes de tudo, a Máquina Virtual Java (de preferência a versão 5 ou posterior). Bom, mãos a obra.

Para saber se você possui uma Maquina virtual Java, entre no prompt de comando de digite a seguinte linha:

```
java  –version
```

Se mostrar algo parecido como mostra o código abaixo:

```
java version "1.6.0_07"
Java(TM) SE Runtime Environment (build 1.6.0_07-b06)
Java HotSpot(TM) Client VM (build 10.0-b23, mixed mode, sharing)
```

Beleza, você possui uma máquina virtual Java instalada no seu computador, caso contrário, instale o JDK. Você pode fazer o download do JDK pelo link abaixo:

http://java.sun.com/javase/downloads/index.jsp

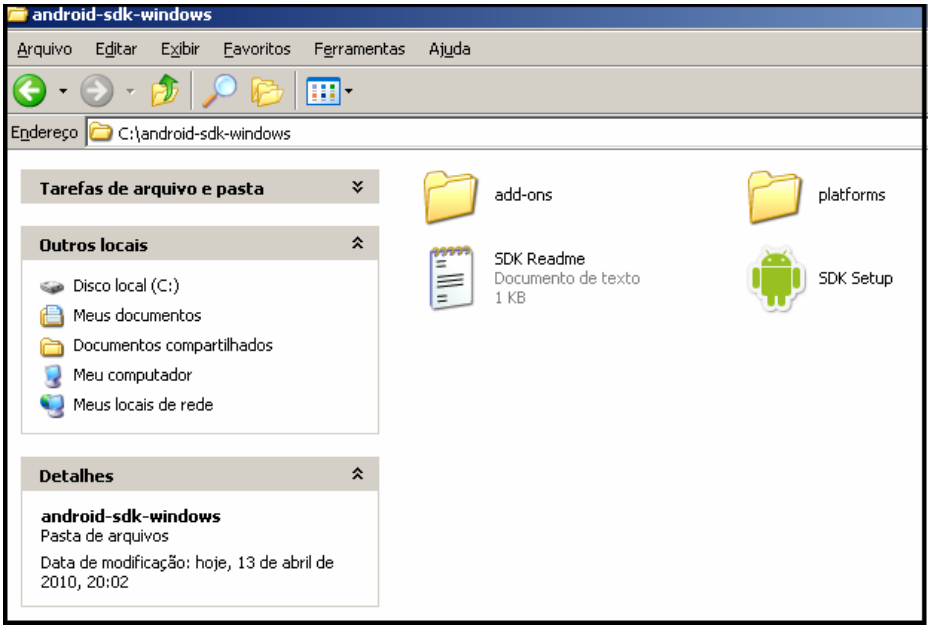
Se você já possui a máquina virtual Java instalada em seu computador, basta agora você fazer o download do Eclipse, que pode ser feita pelo link abaixo:

http://www.eclipse.org/downloads/

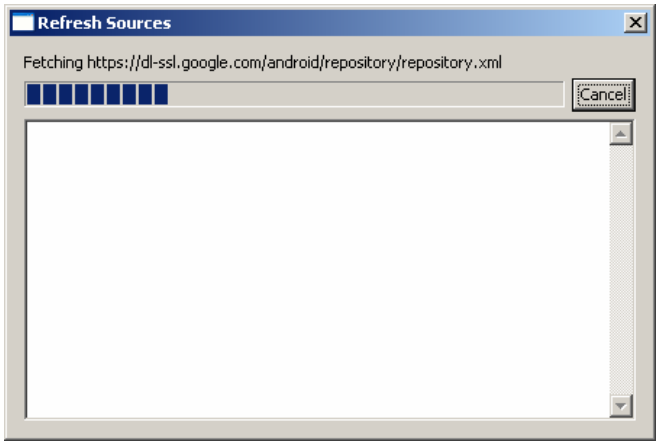
Para fazer o download do Android SDK e do seu plugin , faça pelo link abaixo:

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

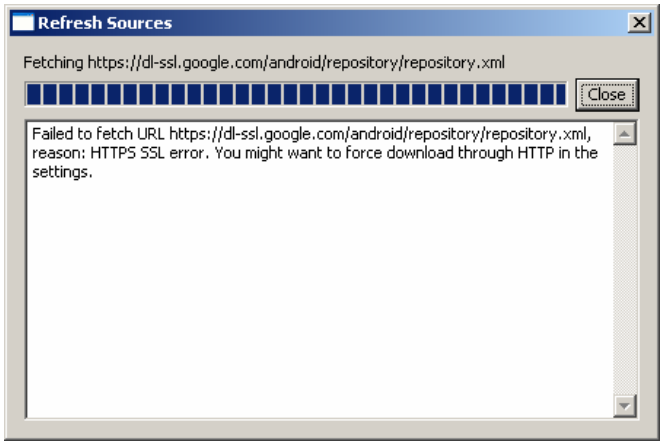
Depois de obter os programas e plugins citados acima, vamos fazer agora as devidas configurações. Primeiramente, você irá descompactar o arquivo “android-sdk_r05-windows.zip”, de preferência no diretório raiz “C:\”. Depois de descompactar, execute o utilitário “SDK Setup”, que se encontra dentro da pasta descompactada, conforme é mostrado na figura abaixo:



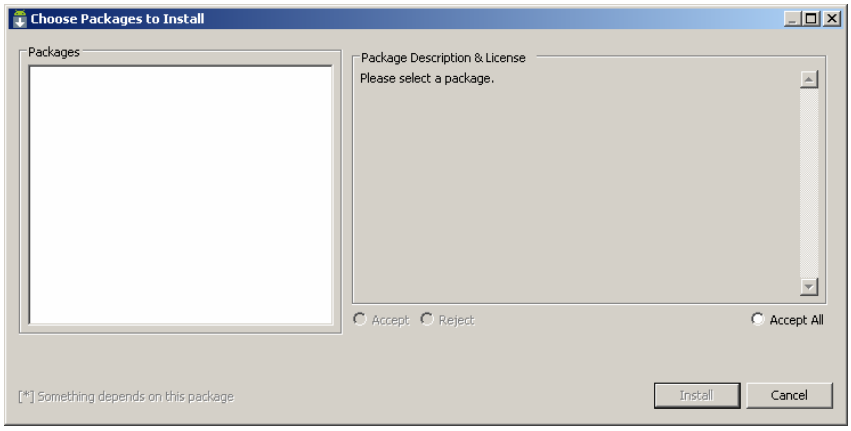
Ao executá-lo, ele irá atualizar as suas fontes , conforme mostra a figura abaixo:



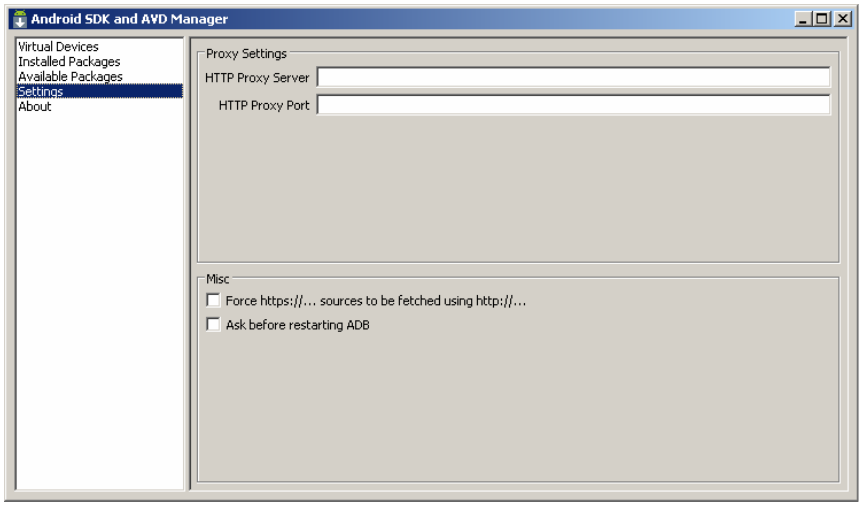
Se durante a atualização das fontes, a aplicação apresentar um erro, conforme mostra a figura abaixo:



Calma, não se desespere! Você vai fechar essa caixa de diálogo clicando no botão “Close”, e será mostrada uma caixa de diálogo, conforme mostra a figura abaixo, simplesmente feche-a, clicando no botão “Cancel”.

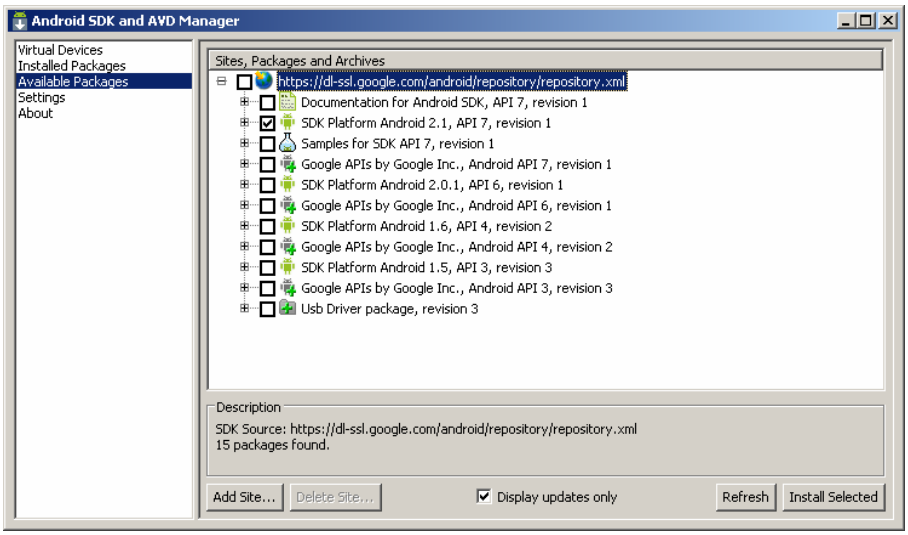


Agora você vai na seção “Settings”, conforme mostra a figura abaixo:

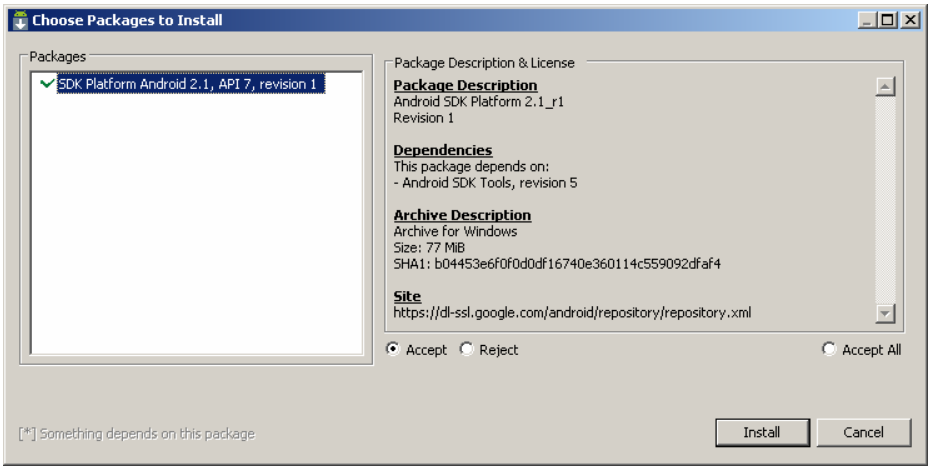


Agora , clique na opção “Force https://... Sources to be fetched using http://...”, e será novamente mostrada a caixa de diálogo de atualização das fontes, que fará a atualização desta vez, com sucesso.

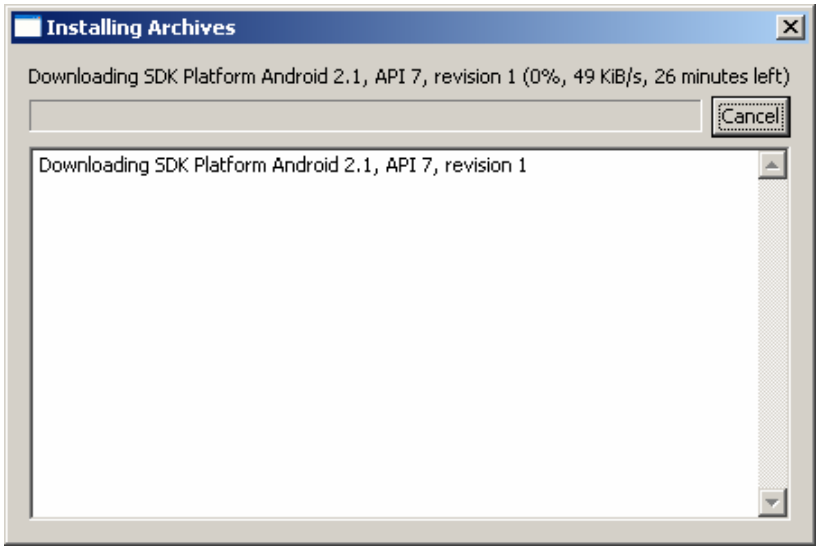
Depois de feito a atualização, vá na seção “Available Packages” e expanda o item ao lado e marque a opção “SDK Platform Android 2.1, API7, revision 1””, como demostra a figura abaixo:



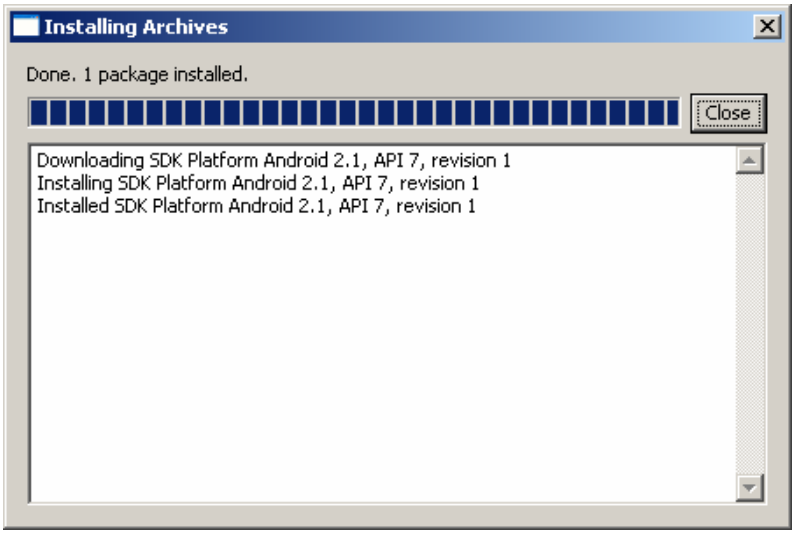
Agora clique no botão “Install Selected” e será mostrada uma nova tela, conforme a figura abaixo:



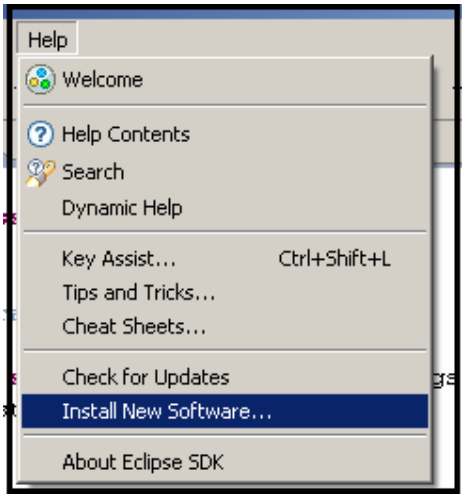
Agora simplesmente clique no botão “Install” e a instalação será efetuada, conforme demonstra a figura abaixo:



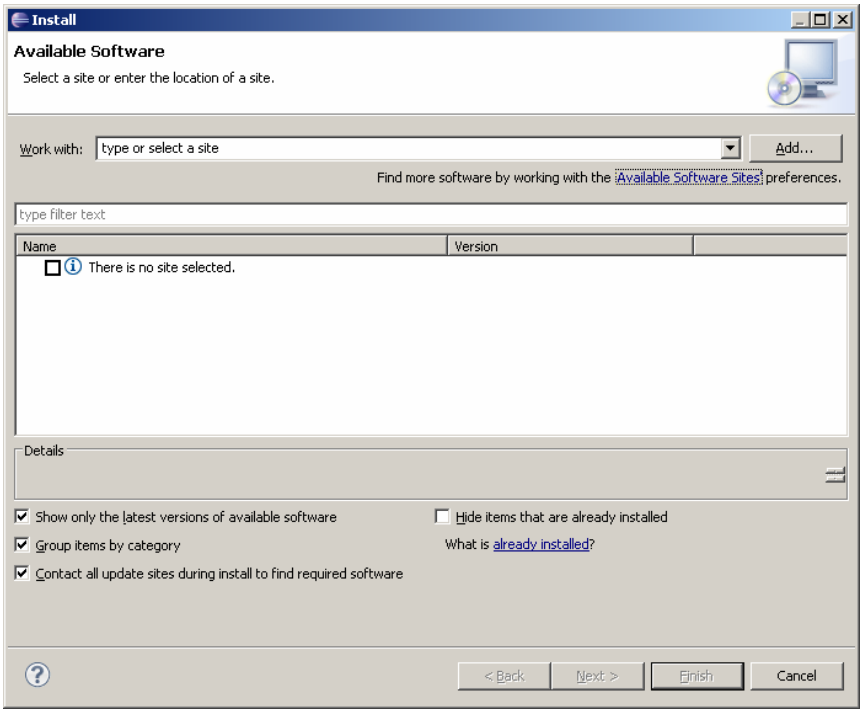
A instalação leva alguns minutos para ser feita. Quando a instalação for concluída, será exibida algumas mensagens, conforme mostra a figura abaixo:



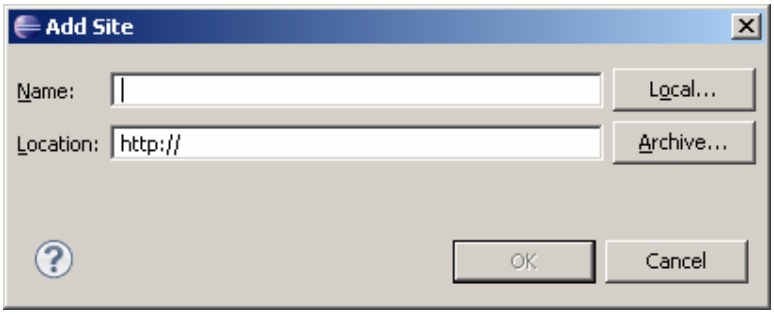
Pronto, uma etapa concluída. Agora vamos instalar o Eclipse com o plugin do Android. Para instalar o eclipse simplesmente descompacte em um local apropriado, de preferência no drive “C:\”. Depois disso copie para o drive “C:\” o plugin do Android “ADT-0.9.6.zip”. Feito isso vamos executar o eclipse. Com o eclipse aberto na no menu “help” -> “Install New Software”, como mostra a figura abaixo:



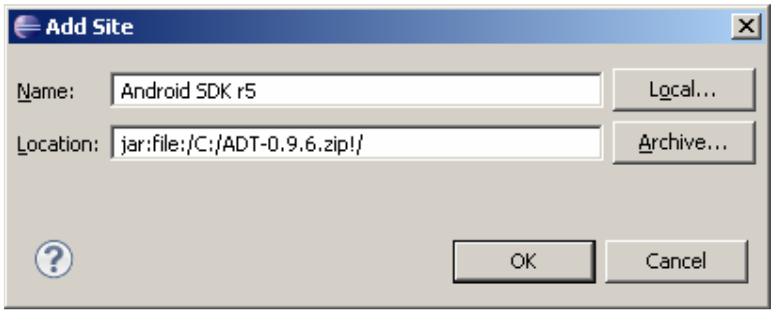
Ao fazer esse procedimento será aberta uma tela conforme mostra a figura abaixo:



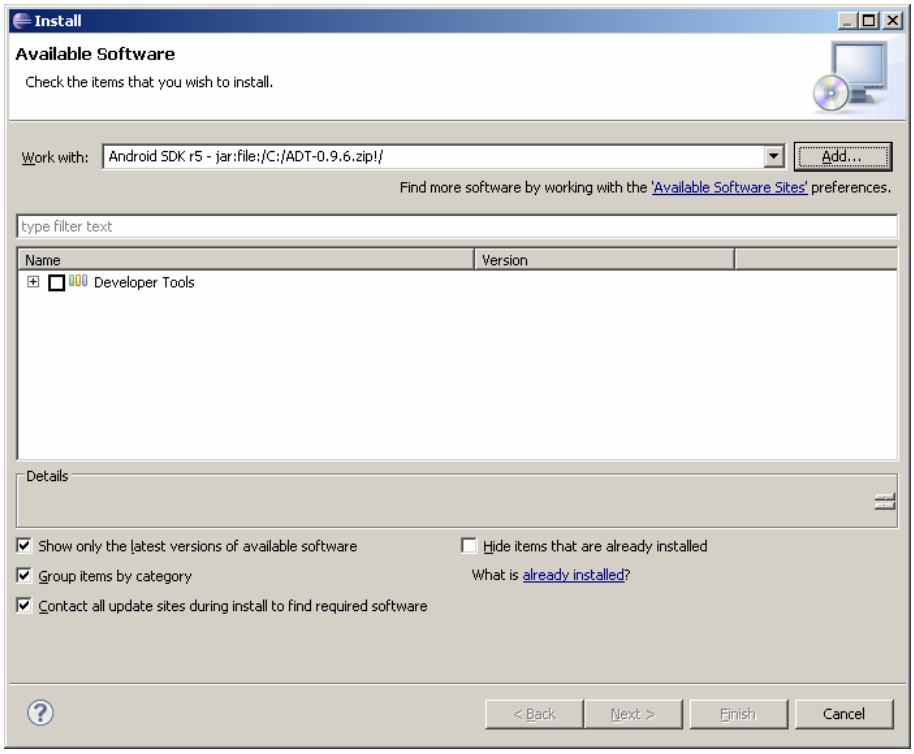
Para instalarmos o plugin do android, clique no botão “Add”, e será mostrada uma caixa de diálogo, conforme mostra a figura abaixo:



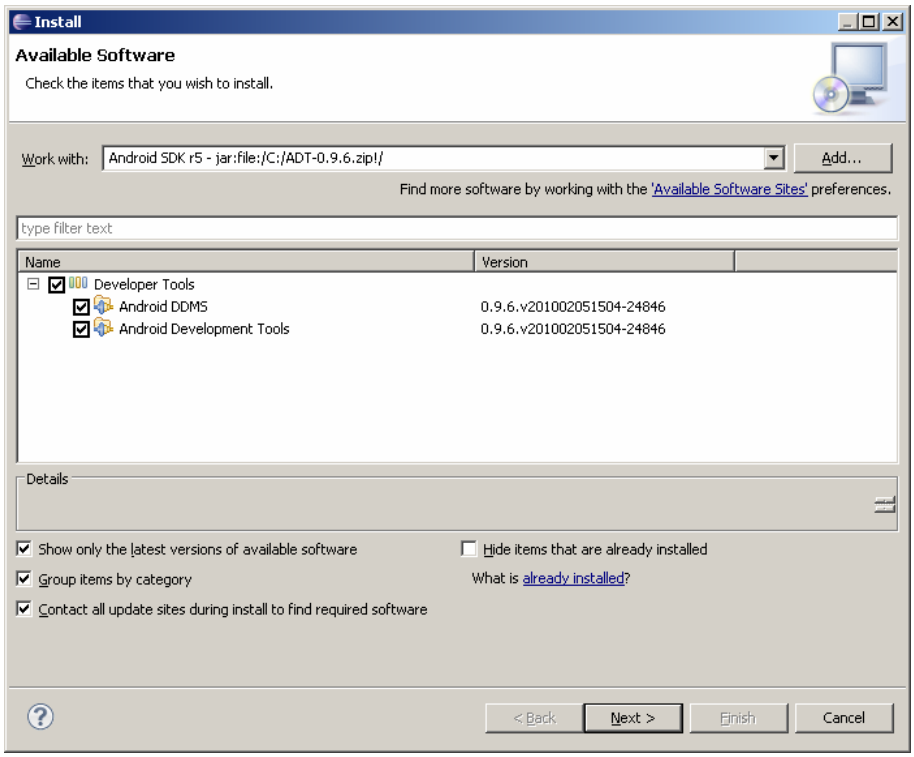
Agora vamos clicar no botão “Archive” e iremos procurar e selecionar o plugin do Android “A.D.T-0.9.6.zip”. Preencha o campo “Name” como mostra a figura abaixo:



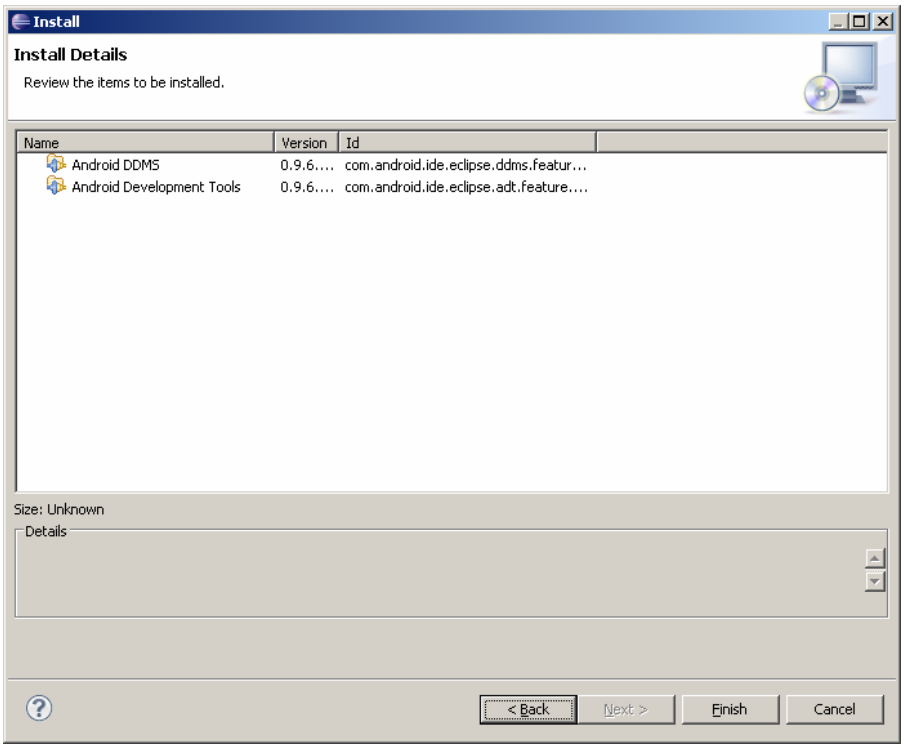
Ao clicar em “OK” será mostrada uma tela, conforme demonstra a figura abaixo:



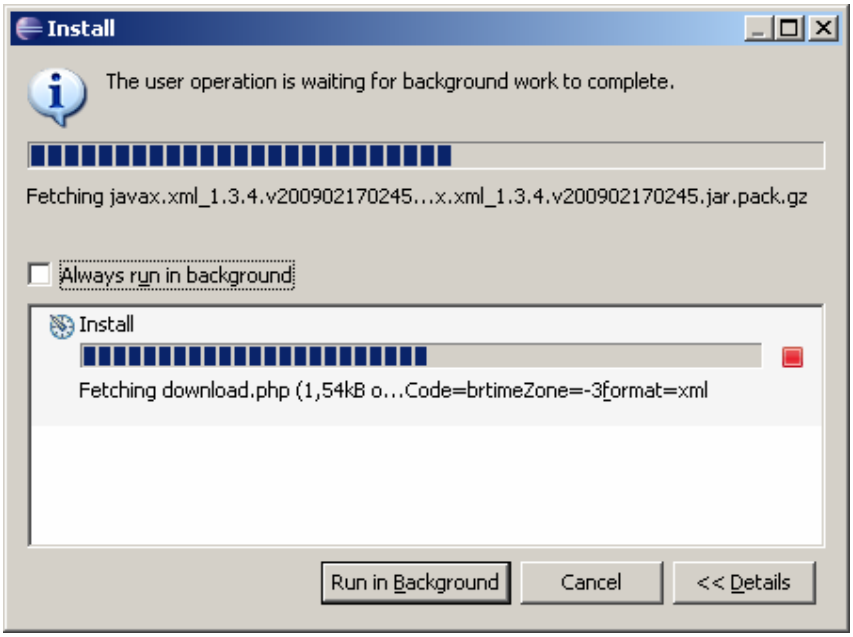
Agora expanda o item “Developer Tools” e marque todas as opções, conforme mostra a figura abaixo:



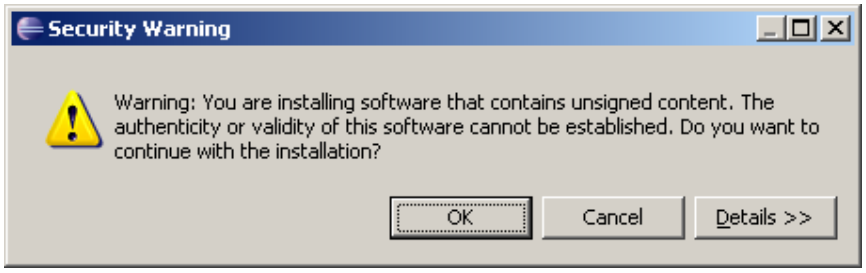
Após fazer isso clique no botão “Next”, e em seguida será mostrada a próxima tela, conforme demostra a figura abaixo:



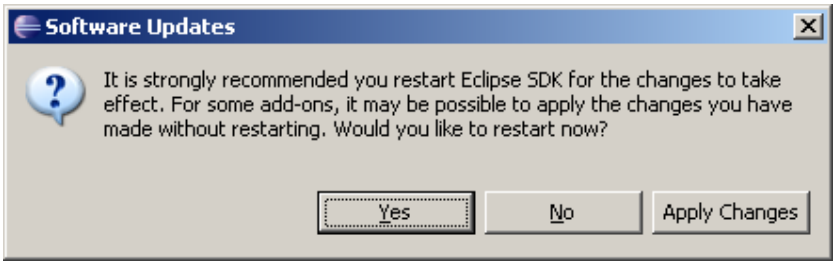
Agora, clique no botão “Finish”. Após isso ocorrerá alguns processos, como demonstra a figura abaixo, aguarde até terminar.



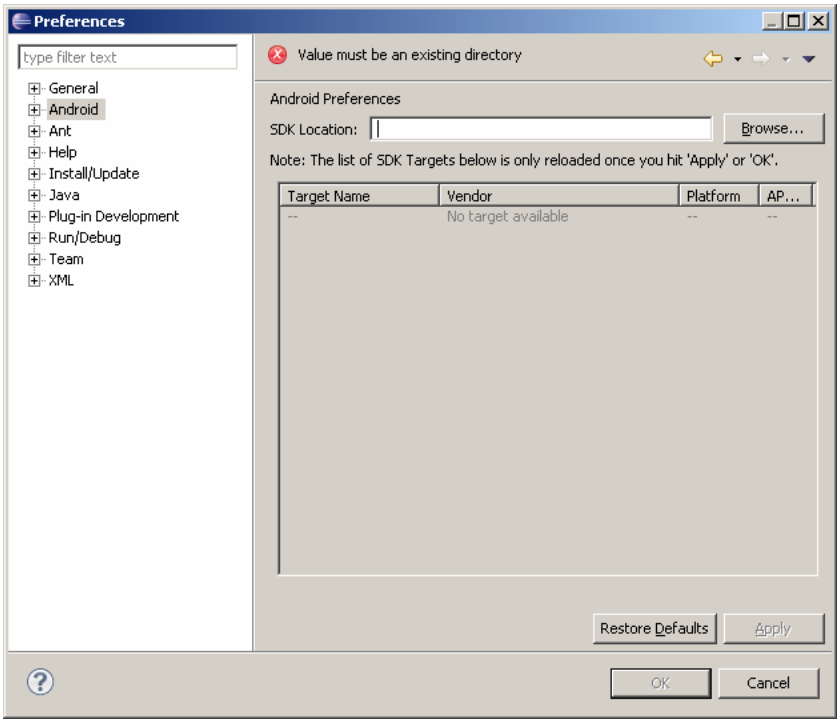
Se em algum momento durante o processo for exibida a figura abaixo:



Pode clicar em “OK” sem problemas, e o processo se completará. Após o termino do processo você deve reiniciar o Eclipse, clicando em “Yes”, na mensagem abaixo:

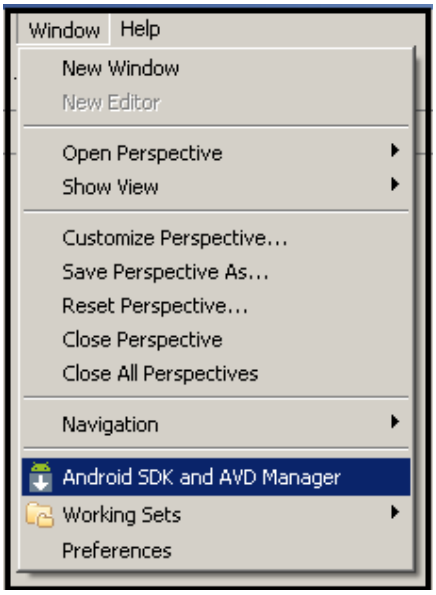


Após o eclipse ter reiniciado, vamos fazer agora as configurações para fazer conexão com o emulador do Android. Vamos no menu “Window” / “Preferences”. Aberta a caixa de diálogo, selecione o item “Android” e será mostrada uma tela, conforme demonstra a figura abaixo:

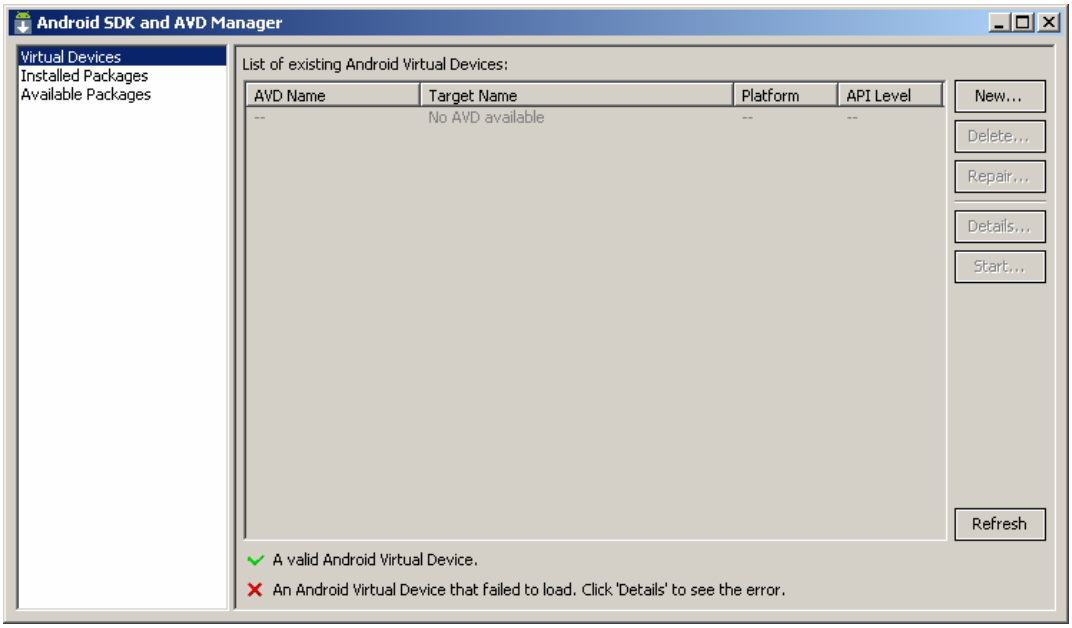


Agora você vai selecionar o diretório onde se encontra o Android, que aqui no meu computador, o android se encontra instalado em “C:\android-sdk-windows”, logo, terei que selecionar essa pasta. Feito isso basta clicar em “OK”.

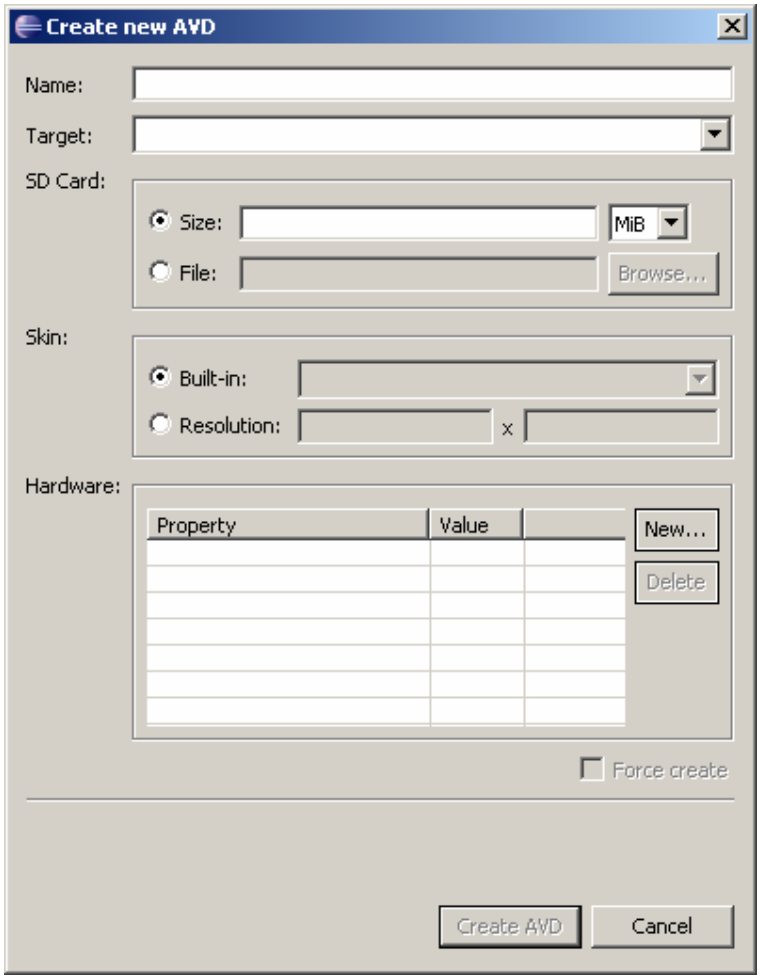
Para finalizar vamos definir um dispositivo virtual, conhecido como AVD (Android Virtual Device), onde nossas aplicações daqui para frente serão executadas. Para isso, vá no menu “Windows” / “Android SDK and AVD Manager”, conforme mostra a figura abaixo:



Feito o procedimento acima, será aberta uma tela conforme mostra a figura abaixo:



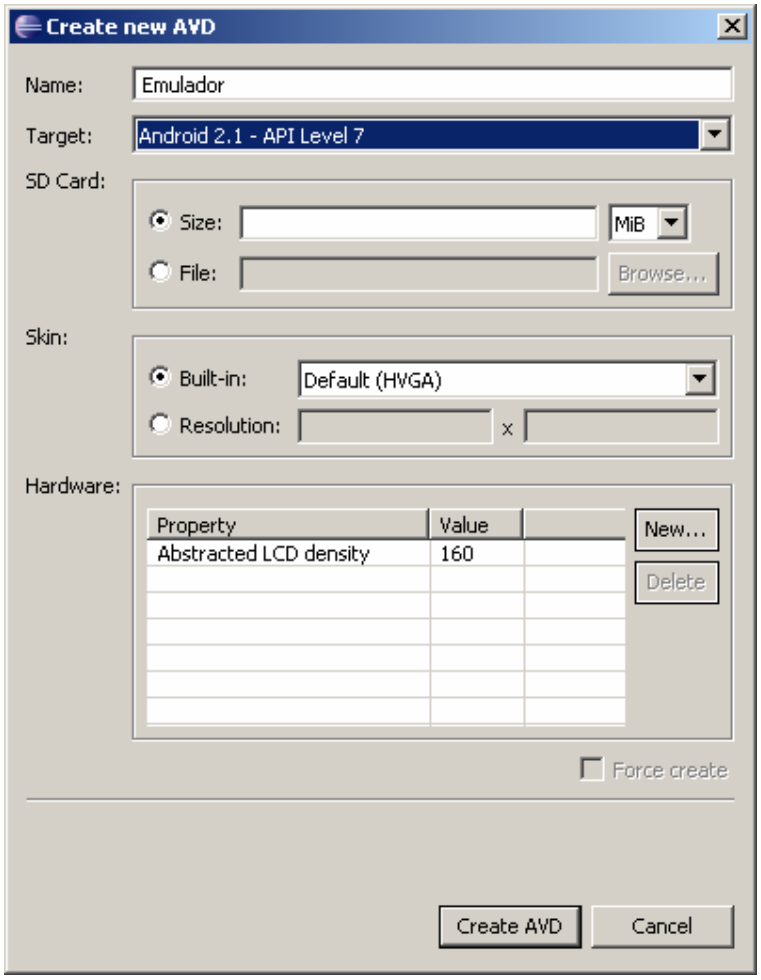
Para criarmos um dispositivo virtual clique no botão “New”, e será aberta uma tela conforme mostra a figura abaixo:



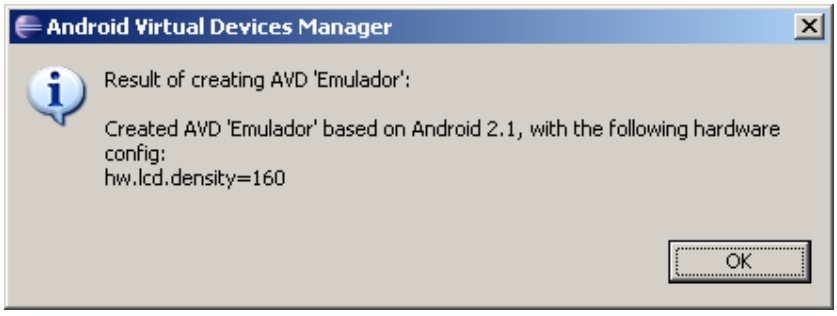
De inicio, vamos configurar o básico pra executarmos a nossa aplicação. Em “Name” você define o nome do AVD, vamos chamá-lo de “Emulador”.

Em “Target” definirmos a plataforma-alvo a ser executada, neste caso só temos uma o “Android 2.1 - API Level 7”. Vamos selecioná-la.

Depois de preencher todos os campos, a tela de criação do ADV deve estar de acordo com a figura abaixo:



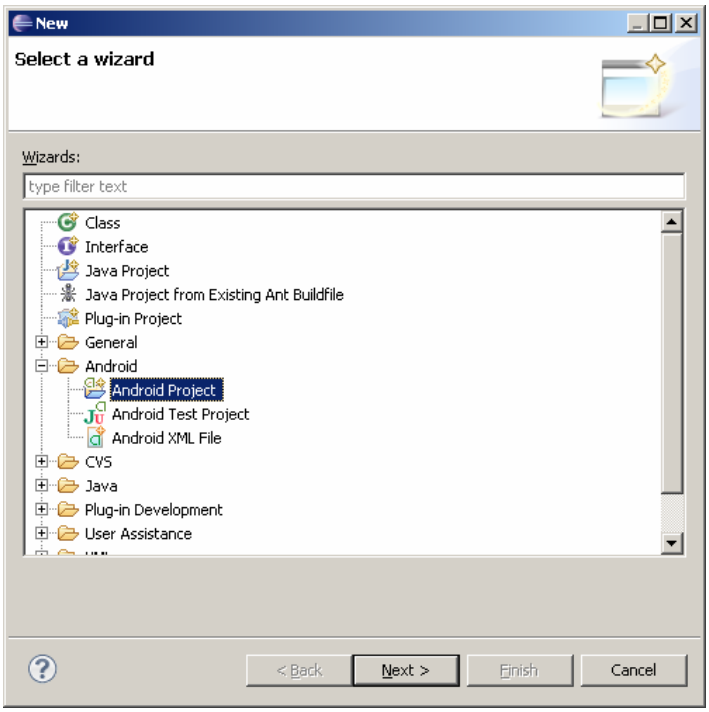
Para criarmos nosso AVD, clique no botão “Create AVD” e pronto. Após criarmos nosso AVD, será mostrada a seguinte mensagem , conforme mostra a figura abaixo:



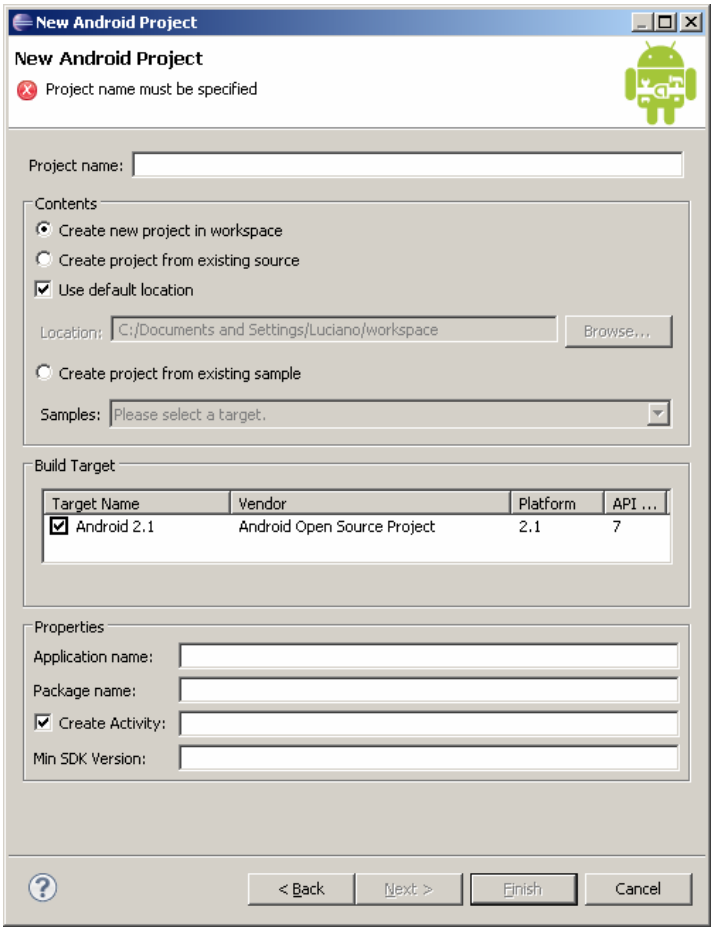
Depois disso, clique em “OK” na mensagem mostrada na figura acima e feche a janela do “Android SDK and AVD Manager”.

4) Criando a nossa primeira aplicação em Android

Agora vamos criar um novo projeto Android indo no menu “File” / “New” / “Other”. Selecione o projeto Android conforme figura abaixo. Depois de selecionar, clique em “Next”:



Após clicar em “Next” na figura acima, será exibida uma tela conforme figura abaixo:



Vamos preencher os campos citados abaixo:

Project name : HelloWorldAndroid

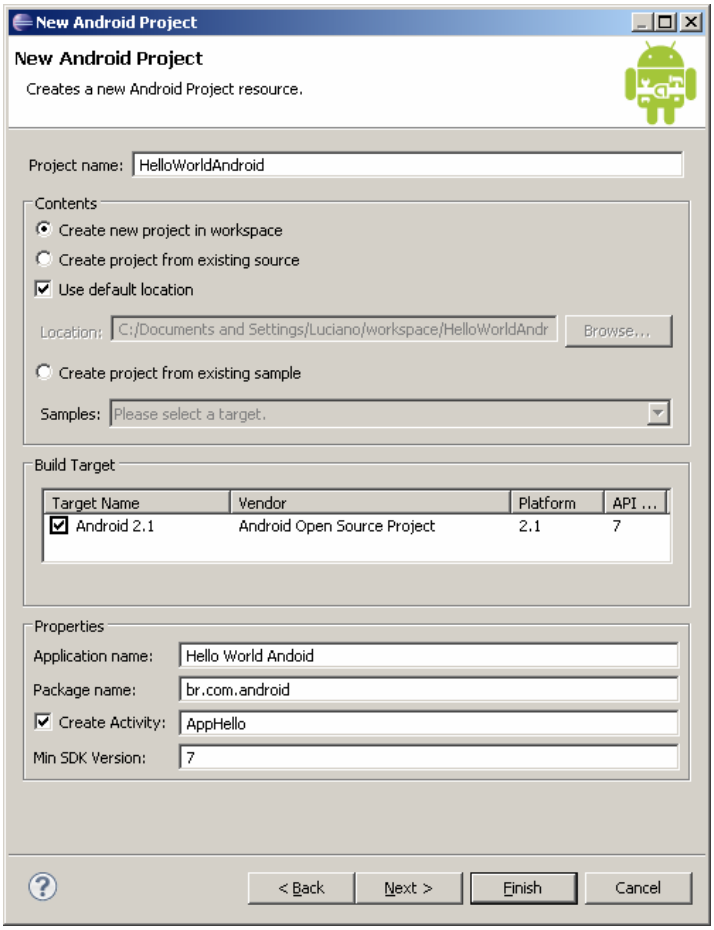
Application name : Hello World Android

Package name: br.com.android

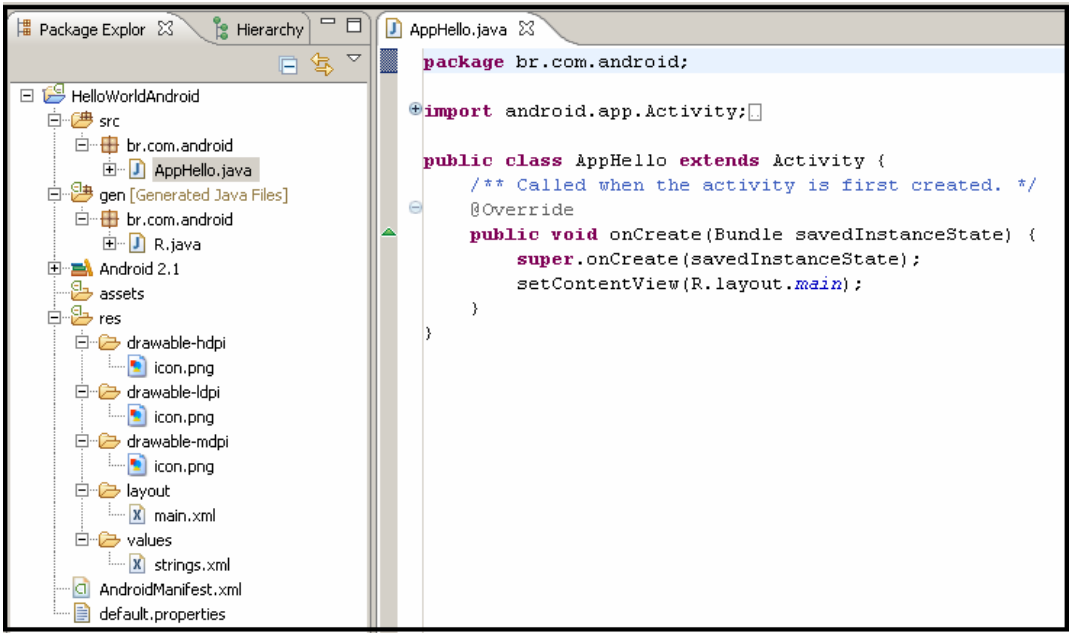
Create Activity : AppHello

Min SDK Version : 7

Os campos preenchidos acima devem estar de acordo com a figura abaixo:



Depois de tudo preenchido basta clicar no botão “Finish” e pronto, nosso projeto foi criado. Em Package Explorer, vamos dar uma olhada na estrutura do Projeto, simplesmente clicando no botão “+”. É só seguir a figura abaixo, aproveite e abra o arquivo AppHello.java , conforme figura abaixo:



Bom, agora irei descrever a estrutura de um projeto Android. Observem que dentro da pasta "HelloWorldAndroid" existe uma pasta chamada "src" e dentro dela é que ficam os códigos fonte java das aplicações. Observem que o arquivo "AppHello.java" se encontra dentro do pacote "br.com.android" (Esse pacote também é uma pasta). Esse arquivo é a nossa aplicação Android. Vou descrever em detalhes o arquivo "AppHello.java" (Veja o código abaixo):

```
package br.com.android;

import android.app.Activity;
import android.os.Bundle;

public class AppHello extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
  }
   }
```

Diferentemente das aplicações comuns de Java, toda classe para aplicação Android deve ser derivada da classe Activity (Atividade) e possui como método principal, o método onCreate. Dentro desse método ele invoca o método onCreate da super classe passando mesmo parâmetro (o *savedInstanceState)*, logo após esse método, vem o método setContentView, responsável por exibir a tela da minha aplicação , baseado nos layouts xml. Por padrão ele chama o arquivo "main.xml".

Dentro da pasta "HelloWorldAndroid" existe um diretório chamado "res", onde ficam armazenados todos os recursos utilizados pela aplicação. Dentro

do diretório “res” existem cinco diretórios, cada um deles com uma finalidade, que descreverei agora:

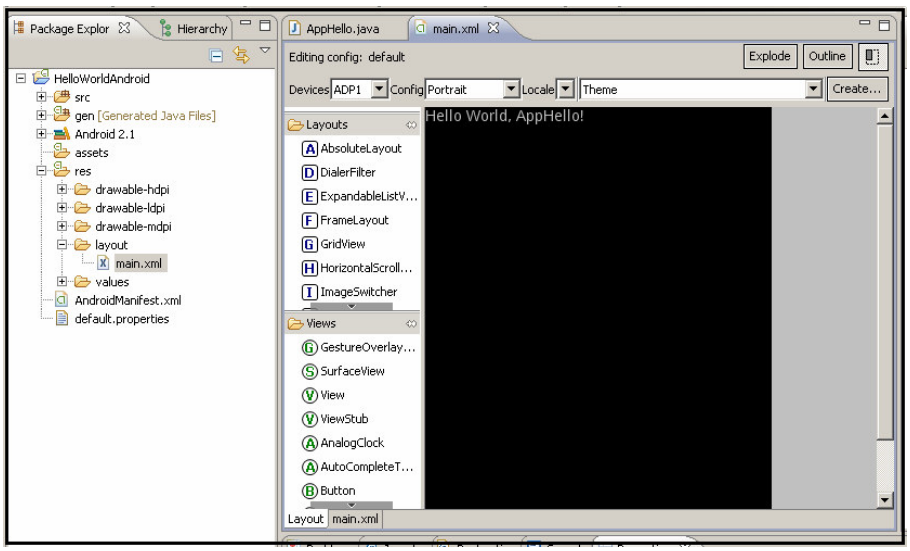
Os diretórios “drawable”

Diferente de algumas versões antigas do Android SDK, como o revision 1, que trabalhei na segunda edição desta apostila, esta versão do SDK trabalha com três diretórios “drawables”, drawable-hdpi,drawable-ldpi, drawable-mdpi.

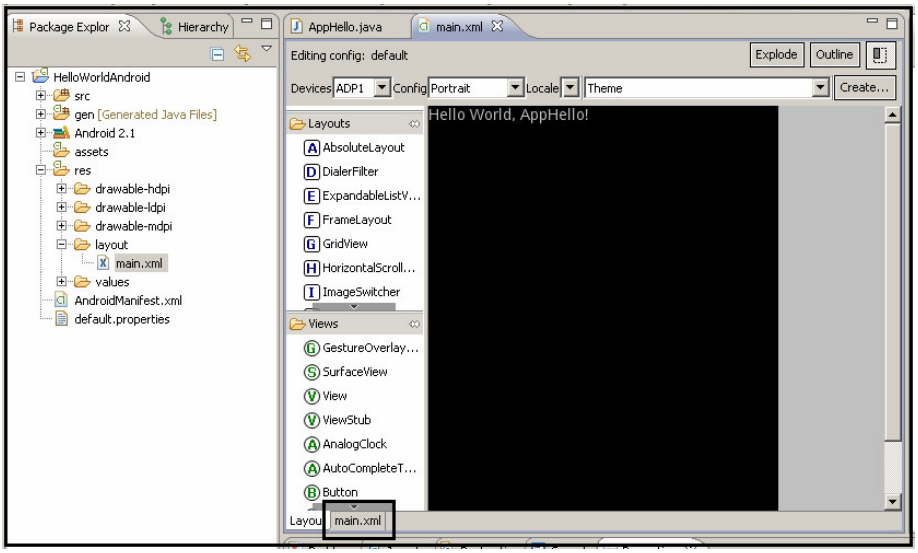
Todos os três armazenam somente imagens, mas qual a diferença de um para outro ? Cada um desses diretórios só será utilizado de acordo com a resolução do Android que você está utilizando, ou seja, qual modelo de emulador de você estiver usando. Por exemplo, quando você usa uma resolução de 480x800 no seu emulador, é utilizado o diretório “drawable-hdpi” para buscar a imagem que vai representar o ícone da sua aplicação Android. Se você for usar uma resolução 320x480 (que é a resolução padrão do emulador Android), é utilizado o diretório “drawable-mdpi”. Se você usar uma resolução 240x400, é utilizado o diretório “drawable-ldpi”.

- O diretório “layout” armazena todas os layouts da aplicação Android, que normalmente são arquivos “.xml”. Para quem conhece a combinação HTML + JavaScript, o Android é similar, é a combinação de XML + Java, logo todos os nosso componentes vão ser adicionados usando tags XML. Por padrão, o arquivo de layout é o main.xml.

Uma coisa interessante que existe nessa versão (e alguma das anteriores) é a capacidade de você ter um “preview” de como ficara a sua aplicação antes mesmo de você rodar o emulador Android, para confirmarmos isso, simplesmente vá no diretório “res/layout”,e de um duplo clique no arquivo “main.xml”, e você verá o seu preview, conforme demonstra a figura abaixo:



Para visualizarmos o código do arquivo main.xml, simplesmente clique na guia “main.xml”, que se encontra abaixo da seção “Views”, como demonstra a figura abaixo:



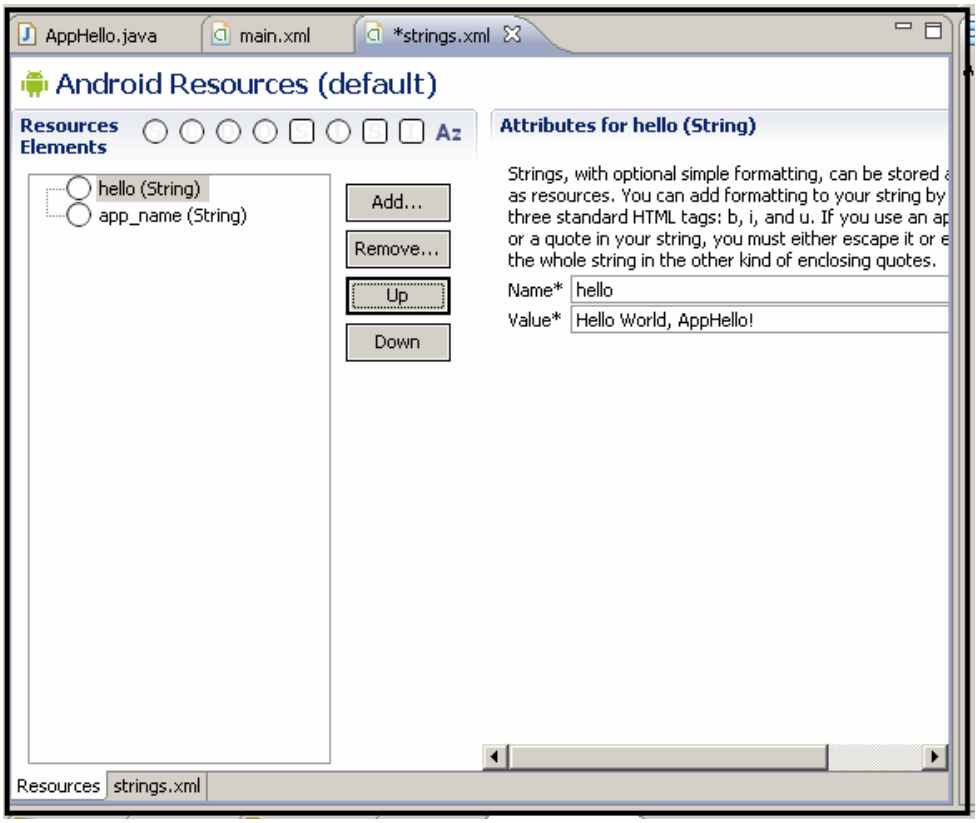
Veja o seu código abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

Observe que após a primeira linha (prólogo xml), existe uma tag chamada LinearLayout, responsável por organizar os componentes exibidos na tela, por padrão os componentes são distribuídos na vertical pelo atributo *android:orientation="vertical"*.

Dentro desta tag, existe uma um componente chamado TextView, que representa um texto a ser exibido na tela , por padrão, ele irá exibir "Hello World, AppHello" através do atributo *android:text="@string/hello"*, onde o valor *"@string/hello"* equivale a uma constante, que está definida no arquivo strings.xml, que se encontra no diretório "values", que iremos descreve-lo agora.

- O diretório "values" armazena valores estáticos que podem ser utilizados por um arquivo ".XML". Normalmente esses valores estáticos devem ser armazenados no arquivo "strings.xml". Vá no diretório "res/values" e de um duplo clique no arquivo "strings.xml", e será mostra o gerenciador desse arquivo, conforme mostra a figura abaixo:

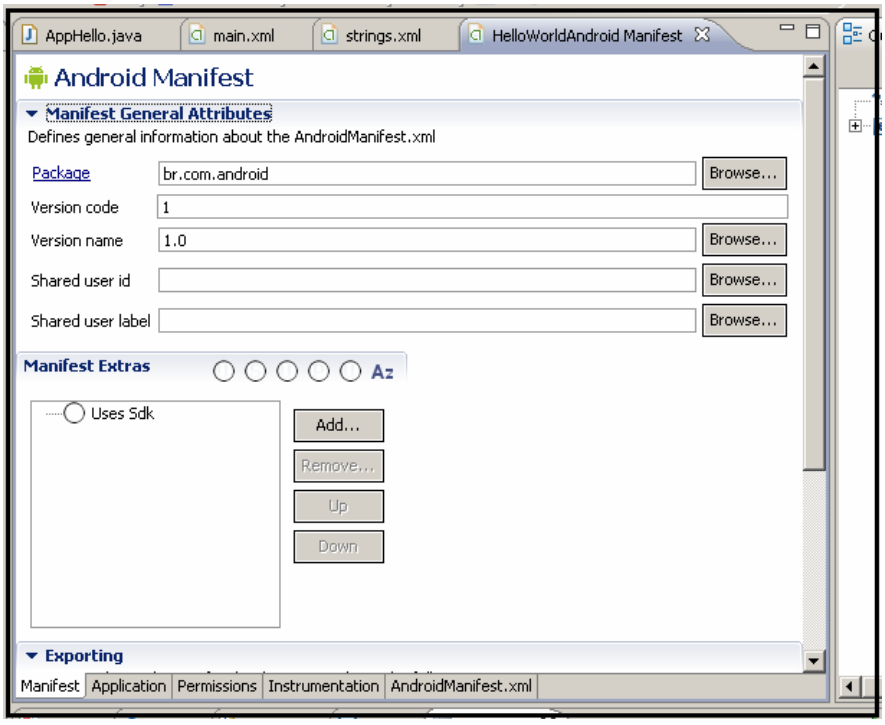


Observe que nas propriedades do atributo “hello”, está atribuído um valor a ela,que é o valor “Hello World, AppHello!”, isso quer dizer que lá no arquivo XML, no componente TextView, tem uma propriedade chama “android:text”, com o valor *“@string/hello”* , que equivale a na verdade a string “Hello World, AppHello!”. Para ver a sua estrutura , clique na guia “strings.xml”. O código desse arquivo é igual ao que demonstra o código abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, AppHello!</string>
    <string name="app_name">Hello World Android</string>
</resources>
```

Observem que dentro desse arquivo eu declaro um valor estático chamado app_name, cujo valor é “Hello World Android”.

Dentro da pasta HelloWorldAndroid existe um arquivo chamado “AndroidManifest.xml” Esse arquivo é o sistema nervoso de uma aplicação Android. É nele que ficam as definições referentes à aplicação. De um duplo clique nesse arquivo para abri-lo, feito isso será mostrado o seu gerenciador, conforme mostra a figura abaixo:



Bom, o que nos interessa aqui é o código. Para visualizarmos seu código, clique na seção “AndroidManifest.xml”. Veja seu código abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="br.com.android"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".AppHello"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>


</manifest>
```

Observem algumas tags interessantes. A tag *<application>* possui o atributo *android:icon*, no qual especifico o ícone da aplicação. Como havia citado anteriormente, todas as imagens ficam no diretório drawable e nesse diretório existe um arquivo de chamado “icon.png” que será o ícone da minha aplicação. Logo, para usar esse ícone neste atributo, deve-se passar o valor

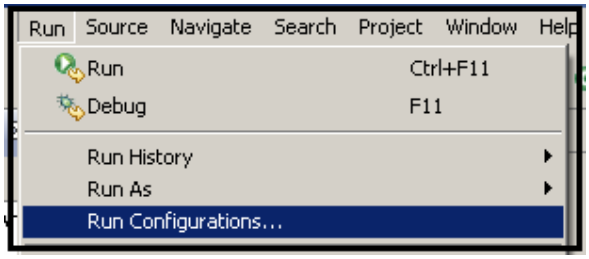
@drawable/icon .Observem que quando informamos o ícone, ele deve ser informado sem a extensão (nesse caso, PNG).

Observem agora a tag *<activity>*, ela define uma atividade (Activity).. Dentro desta tag, eu possuo o atributo chamado *android:label* que define o título da minha aplicação. O título que será exibido é o valor que está armazenado no valor estático app_name. Isso é obtido pela atribuição *android:label="@string/app_name"*.

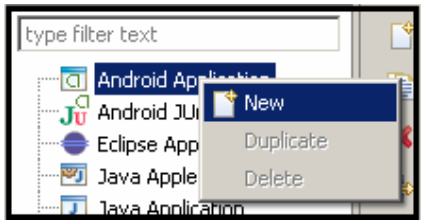
Como havia falado a aplicação Android nada mais é do que a combinação Java + XML. Agora, como um código Java vai acessar um componente que está escrito em XML ? Ah, essa é a finalidade do arquivo R.java (que fica dentro do pacote "gen" , situado no projeto), ele funciona como uma "interface" entre o código Java e o código XML, logo, se eu quiser manipular em tempo de execução um componente via Java, tenho que fazer interface com esse arquivo. Em breve vamos ver como.

OBS: O arquivo R.java não pode ser modificado manualmente. Ele é modificado automaticamente de acordo com as mudanças feitas no projeto.

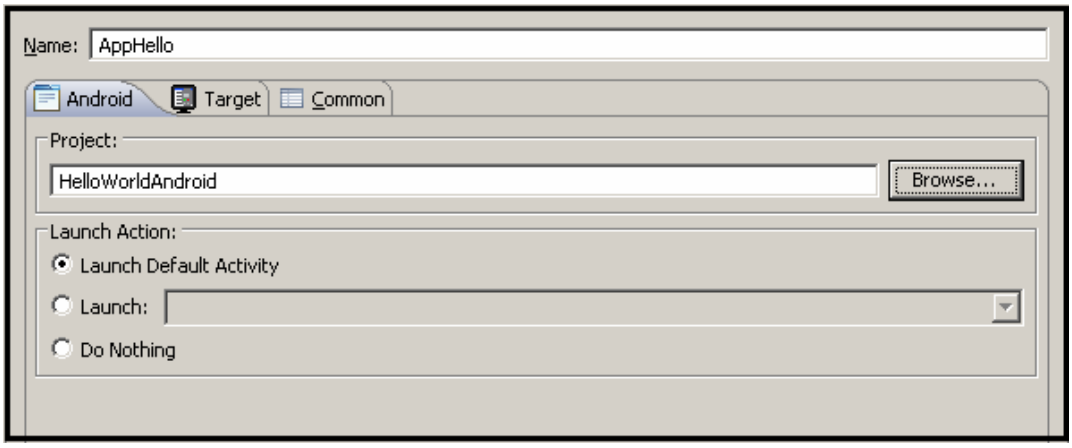
Agora iremos executar nossa aplicação. Vamos no menu "Run / Run Configurations", conforme mostra a figura abaixo:



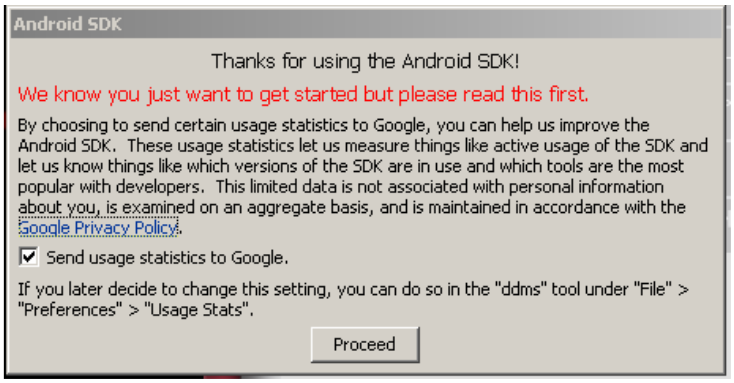
Feito isso, será aberta uma caixa de diálogo com vários itens. Clique com o botão direito do mouse no item "Android Application" e selecione a opção New, conforme a figura abaixo:



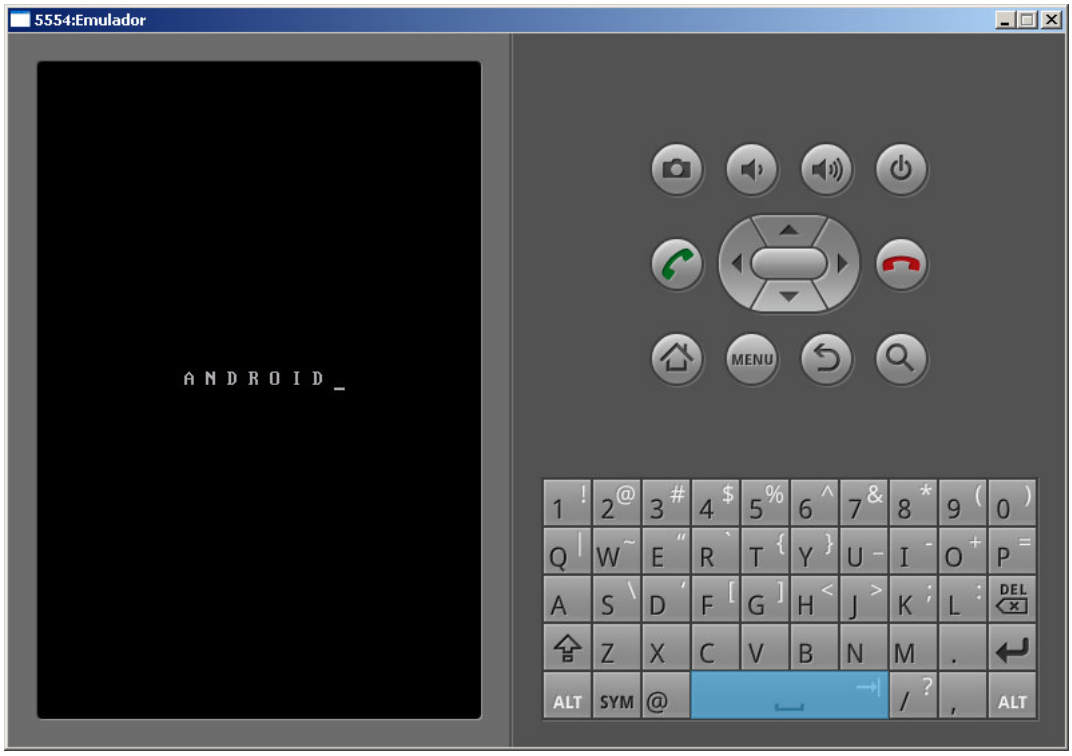
Feito isso, na propriedade "Name" ao lado digite "AppHello". Em Project selecione o projeto que criamos em clicando no botão Browse, com o nome de HelloWorldAndroid. E por último, em "Launch Action", deixe marcada a opção "Lauch Default Activity". Qualquer dúvida siga a figura abaixo:



Agora é só clicar em “Run” e rodar a aplicação. Quando o emulador Android é executado, possivelmente, poderá abrir junto com ele uma caixa de dialogo, conforme a figura abaixo. Normalmente, eu desmarco a opção “Send usage statistics to Google” e clico em “Proceed”.

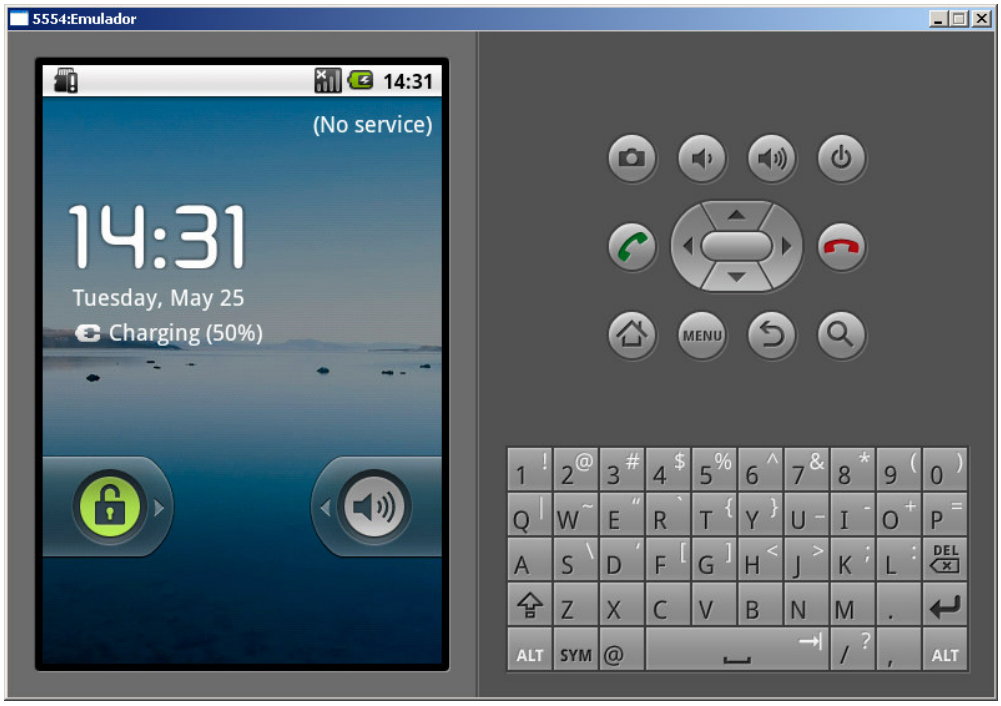


Pra você que está executando pela primeira vez o emulador do Android, vê que o emulador é uma espécie de IPhone. Lado esquerdo nós temos a tela do dispositivo e no lado direito temos o teclado com suas funções, conforme mostra a figura abaixo:



No início da execução do emulador mostra o título Android, conforme você vê na figura acima. Depois, vem um outro título escrito “Android”, com um titulo cinza em animação. Isso demora em torno de 2 a 10 minutos (dependendo da sua máquina. É recomendável que você tenha no mínimo 512 MB de memória e um processador bem rápido para um bom desempenho da execução) para a aplicação ser exibida, mesmo sendo essa aplicação algo muito simples.

Passado o tempo que citei acima, será mostrada a nossa aplicação e também algumas mensagens, é só cancela-las. Quando o emulador chegar nessa tela abaixo:



Clique no botão redondo com o titulo “MENU” para desbloquear a tela e a aplicação continuará a processar até ser carregada com sucesso, conforme mostra a figura abaixo:

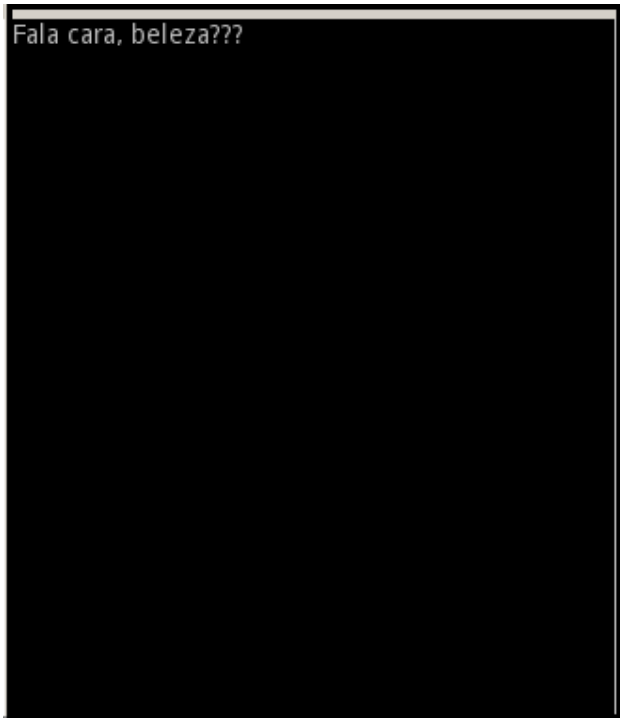
Esse emulador já vem com uma série de recursos como Navegador, Aplicações de demonstração, Mapas, Lista de contatos e etc.

Se você neste exato momento fechou o emulador após a execução da aplicação, vou te dizer uma coisa: “Não era para você ter feito isso”. Se você esperou muito tempo para ver essa aplicação em execução, ao executar novamente a aplicação, você vai esperar o mesmo tempo. Nessa situação, ao executar pela primeira vez o emulador, e caso vá executar outros programas, minimize o emulador ao invés de fechar, pois se você esperou muito tempo para executar esse programa, com ele minimizado, ao executar um outro programa, o eclipse vai fazer uso do emulador já aberto em vez de abrir outro, com isso, a aplicação levará em torno de 7 a 12 segundos em média para ser executada. Nunca esqueça isso!

Vamos modificar essa aplicação. Minimize o emulador e vamos abrir o arquivo “main.xml”. Na tag TextView que já havia explicado a vocês, possui um atributo chamado *android:text*, onde nele defino o título que será exibido, modifique agora essa propriedade com o seguinte valor (título), conforme o código abaixo:

```
android:text="Fala cara, beleza ???"
```

Feito isso, salve a aplicação e veja seu “preview”, clicando na seção “layout”. Veja seu preview abaixo:



Vamos fazer mais uma outra modificação na nossa aplicação. Abra novamente o arquivo main.xml, observe que ele possui um TextView certo ? Vamos colocar mais duas TextViews, a primeira TextView, no atributo android:text terá o título “Primeira frase”, o segundo TextView terá o título “Segunda Frase” e assim sucessivamente. Veja como ficará o código do arquivo “main.xml” :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Primeira Frase."
    />

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Segunda Frase"
    />

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Terceira Frase???"
    />
</LinearLayout>
```

Feito isso, salve o arquivo e veja seu “prevêem”, como demonstra a figura abaixo:

E aeh, tá entendo aos poucos como se faz aplicações Android ? Com certeza que sim!

Como podemos ver nessa versão do Android, ele já oferece um utilitário que permite a criação de aplicações de forma rápida, simplesmente arrastando e soltando os componentes. Isso acelera o processo de desenvolvimento de aplicações.

Nesta material, vamos trabalhar no Android usando esse utilitário que acelera o processo de desenvolvimento de aplicações, mas, em algumas ocasiões, faremos do modo tradicional, ou seja, digitar o código.

Agora vamos aprofundar um pouco e fazer aplicações mais interessantes com o uso dos Widgets (componentes) existentes na plataforma Android.

5) Usando Widgets

Toda aplicação Android é constituída por widgets, que são componentes gráficos que constituem uma aplicação Android. A partir de agora iremos conhecer os widgets básicos que constituem a plataforma android, para o desenvolvimento das aplicações. De acordo com alguns widgets que fomos conhecendo, vamos desenvolver aplicações que demonstrem o uso deles.

5.1) A widget TextView

A widget **TextView** funciona como se fosse uma Label (“rotulo”), onde nele podemos mostrar alguma informação, mensagem e etc. Na nossa primeira aplicação, tivemos a oportunidade de usarmos esse componente.

5.2) A widget EditText

A widget **EditText** funciona como se fosse caixa onde podemos digitar nela dados do teclado.

5.3) A widget Button

A widget **Button** nada mais é do que um Botão de comando , que quando clicado, dispara uma ação, um evento.

5.4) Desenvolvendo uma aplicação que soma números

Com os componentes até agora vistos, já é possível desenvolvermos uma aplicação. Vamos criar agora uma aplicação que faça uso de um desses widgets. Crie um novo projeto Android com os seguintes dados:

Project Name: SomaNumeros

Package Name: br.com.android

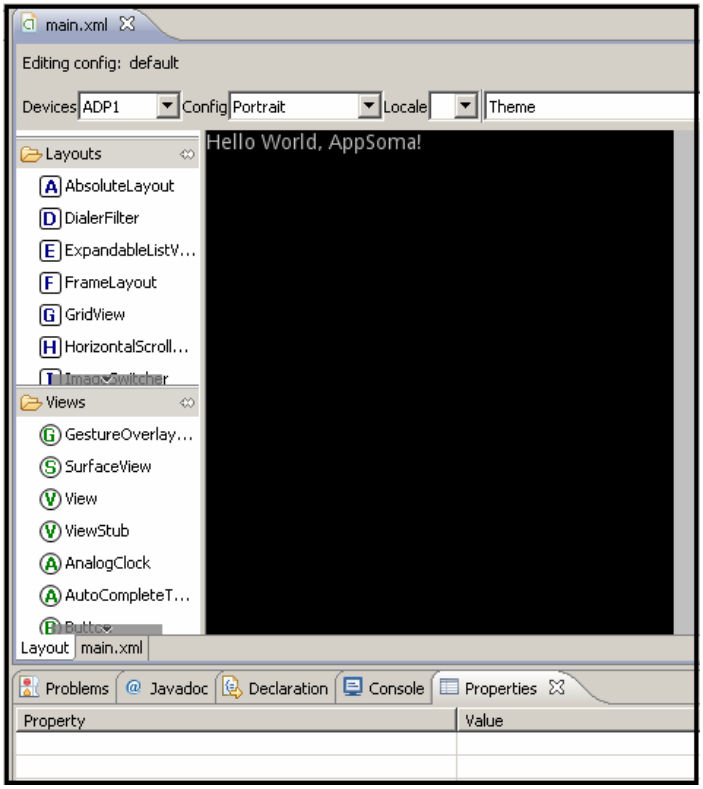
Create Activity: AppSoma

Application Name: Soma Números

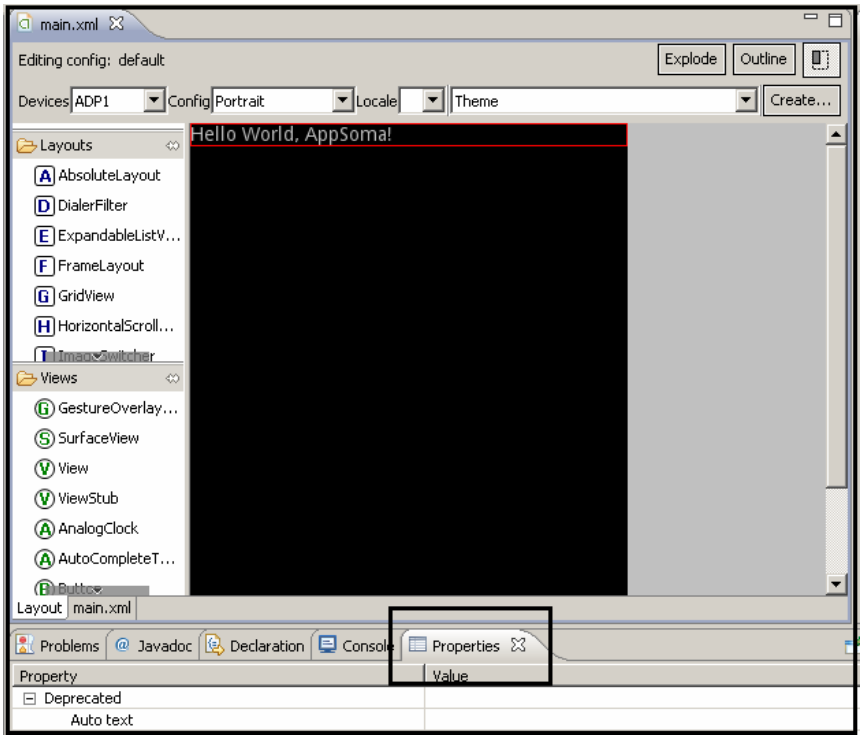
Min SDK Version: 7

Criado o projeto, vamos no arquivo “main.xml” desse projeto e como havia mencionado, vamos agora fazer uso do utilitário que ira nos ajudar a construir a nossa aplicação da forma rápida. Pelo nome do projeto, podemos ver que essa aplicação é uma aplicação de calculo. Essa aplicação vai ler dois números inteiros e no final, irá mostrar a soma deles, simples.

Vamos no arquivo “main.xml” desse projeto e vamos fazer as seguintes modificações. Observe que logo de início, ele mostra a frase “Hello World,AppSoma!” na widget TextView, como mostra a figura abaixo:

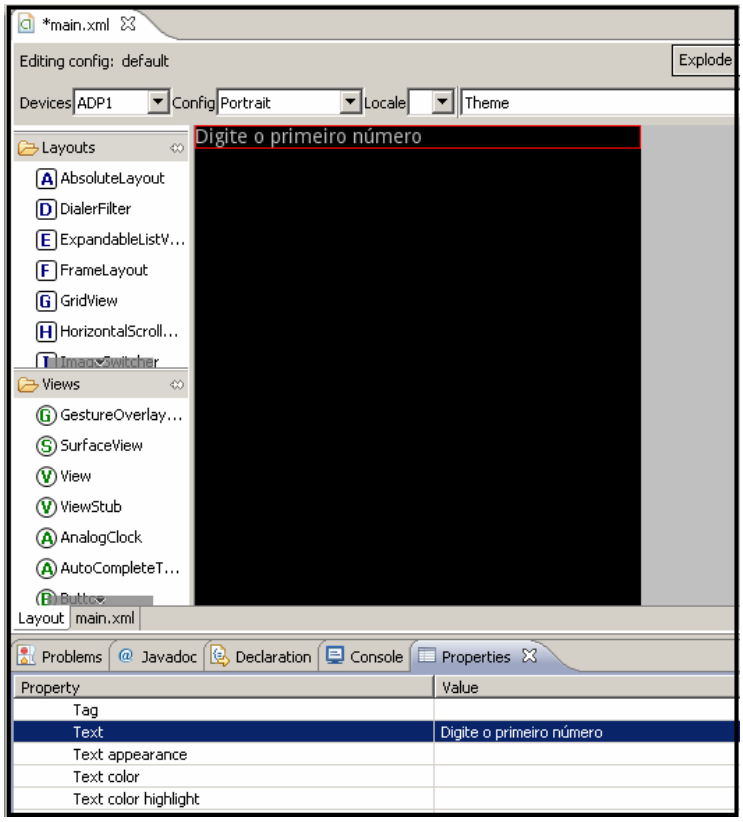


Bom, clique na frase para selecioná-la pois, iremos modificar seu conteúdo. Se você observar abaixo existe uma guia chamada “Properties”, que indica a propriedade de um componente devidamente em edição, conforme mostra a figura abaixo:



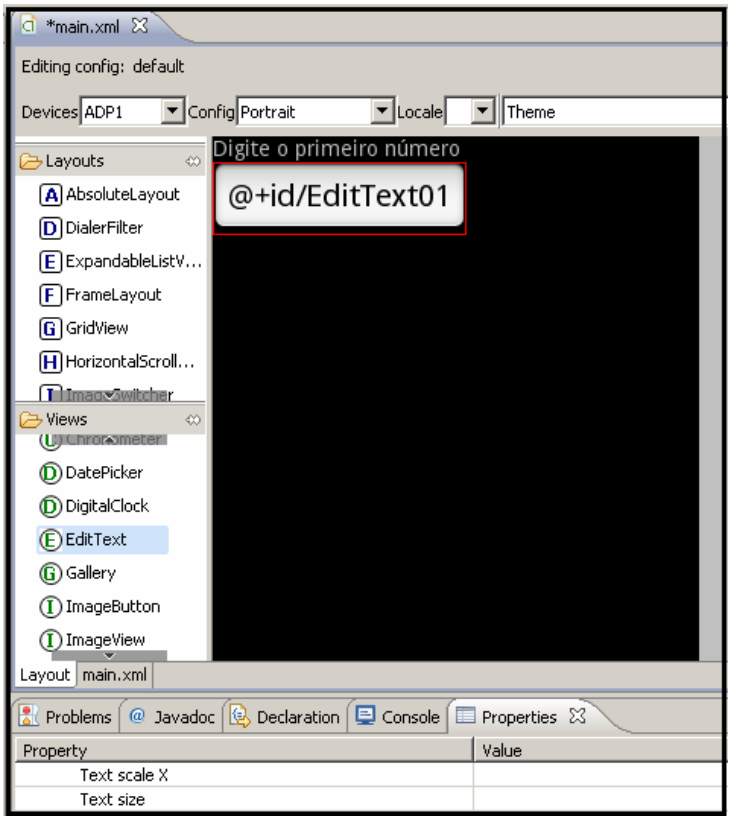
Se você notou, quando clicamos no componente mostrou uma série de valores na propriedade, isso indica os atributos daquele componente.

Agora vamos na guia “Properties” encontrar uma propriedade chamada “Text”, que indica o conteúdo assumido pelo componente TextView, que no caso é a frase “Hello World,AppHello!”. Depois de encontrar a propriedade Text, substitua o valor corrente pela frase “Digite o primeiro número” e depois disso, de ENTER. O resultado você confere na figura abaixo:



Ótimo, agora vamos inserir a widget EditText, que funciona como um campo para preenchermos com valores número ou alfanuméricos. Como vamos adicionar esse componente? Se você observar na figura acima, existe uma seção chamada “Views”, é nela onde ficam os componentes que constituem uma aplicação Android.

Primeiramente, encontre o componente (widget) EditText, depois de encontra-lo, simplesmente você vai clicar sobre ele e arrastar até a tela do dispositivo, para adiciona-lo. O resultado você confere na figura abaixo:



Bom, agora vamos modificar duas propriedades desse componente.

Encontre a propriedade "id" do componente EditText e nela você vai inserir o valor *"@+id/numero1"*. Essa propriedade serve para darmos um nome ao componente, caso ele seja trabalhado no código Java.

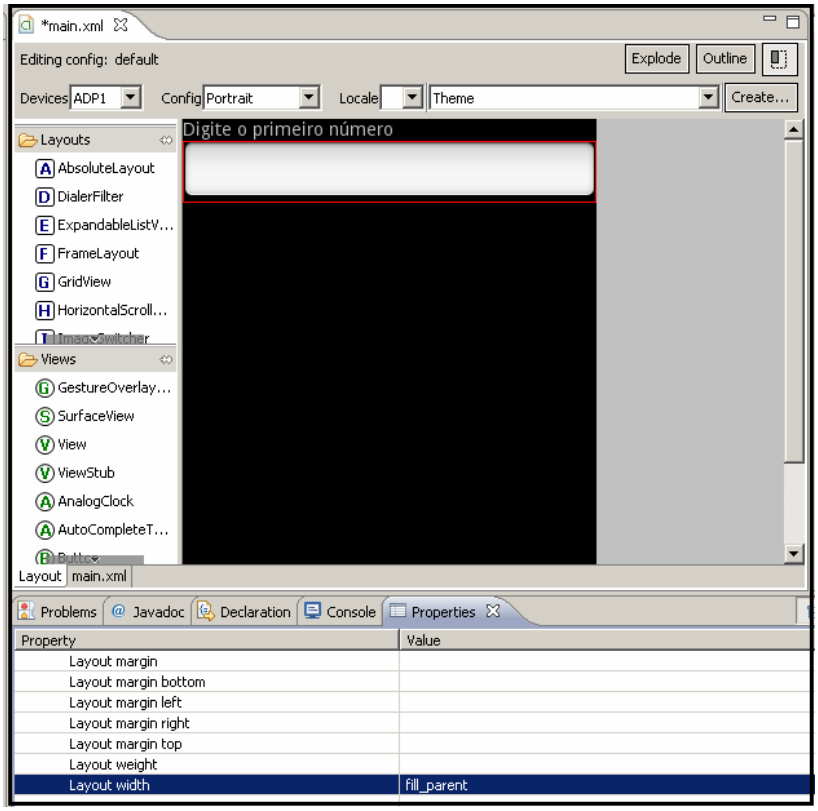
O nome de um widget deve estar nesse formato:

"@+<nome>/<nome>"

Se observamos, o valor desse atributo é : *"@+id/numero1"*. É como se "id" representasse um grupo e "numero1" representasse o nome do componente. Você irá entender essa notação mais a frente.

Depois disso, vamos na propriedade "Layout width", que define a largura de um componente e iremos definir o valor *"fill_parent"*, que indica que o componente irá ocupar toda a largura do dispositivo.

Para finalizar, vamos agora modificar a propriedade "Text", deixando seu conteúdo em branco. O resultado você confere na figura abaixo:



Agora você vai inserir, NA SEQUÊNCIA, os componentes TextView e EditText.

Na segunda TextView, vamos inserir na propriedade Text a frase “Digite o segundo número”.

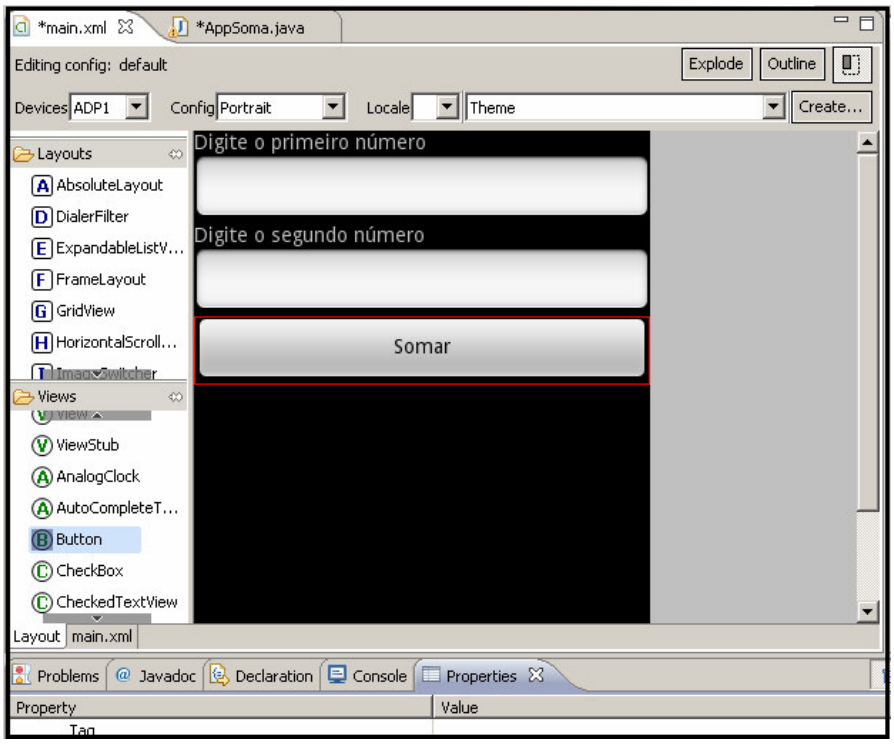
Já na segunda EditText, vamos repetir os mesmos procedimentos que fizemos na primeira EditText, sabendo-se que a diferença vai estar na propriedade “id” que assumirá o valor *“@+id/numero2”*. Somente essa é a diferença, o resto é tudo igual.

Agora vamos inserir um componente chamado Button na nossa aplicação. Depois de inserido, modifique as propriedades abaixo com os seus respectivos valores:

Button

Propriedade	Valor
Layout width	fill_parent
Id	@+id/btsomar
Text	Somar

Depois de feito todas as alterações, o layout da aplicação deve estar de acordo com a figura abaixo:



Beleza, agora vamos abrir o código do arquivo “AppSoma.java”, para acessar via código Java os componentes que adicionamos via XML. Siga os passos aqui descritos para você ver como esse processo é feito. Após a linha

```
import android.os.Bundle;
```

Digite:

```
import android.widget.*;
import android.view.*;
import android.app.*;
```

Antes da linha:

```
@Override
```

Digite:

```
EditText ednumero1,ednumero2;
```

Agora vamos à explicação do código acima. Como você pode ver , os widgets também podem ser usados no nosso código Java. Se no código XML eu possuir um widget do tipo EditText, para acessar esse componente pelo Java, é preciso fazer uso da classe EditText. Cada widget no XML possui o seu

respectivo em classe Java, logo, se possui um widget Button, para acessá-lo devo fazer uso da classe Button e assim vai.

Agora, após a linha:

```
setContentView(R.layout.main);
```

Digite as seguintes linhas de código:

```
ednumero1 = (EditText) findViewById(R.id.numero1);
ednumero2 = (EditText) findViewById(R.id.numero2);

Button btsomar = (Button) findViewById(R.id.btsomar);
```

Agora vou explicar as linhas acima. A linha:

```
ednumero1 = (EditText) findViewById(R.id.numero1);
```

Faz referência ao primeiro EditText, através do método findViewById com o parâmetro *"R.id.numero1"*.

Ah, se lembra o nome da primeira EditText que está no código XML? Ela se chama *"@+id/numero1"*.

Vamos entender, observe que para fazer referência ao EditText pelo método findViewById eu passei o parâmetro *R.campo.num1*.

Já na segunda linha, para fazer a referência à segunda EditText, cujo nome é *"@+id/numero2"*, pelo método findViewById, passei o parâmetro *R.id.numero2*.

Como você pode ver, estou fazendo uso da classe R, que funciona como interface entre o código Java e o arquivo XML.

O procedimento é o mesmo para o Button.

Agora iremos adicionar um evento *Click*, no nosso Button, pois quando eu clicar no botão, ele deverá mostrar a soma dos números. Então, logo após a linha:

```
Button btsomar = (Button) findViewById(R.id.btsomar);
```

Digite:

```
btsomar.setOnClickListener(new View.OnClickListener(){

 public void onClick(View arg0) {
```

```
      double num1 =
Double.parseDouble(ednumero1.getText().toString());
      double num2 =
Double.parseDouble(ednumero2.getText().toString());
      double res = num1 + num2;

      AlertDialog.Builder dialogo = new
AlertDialog.Builder(AppSoma.this);
      dialogo.setTitle("Aviso");
      dialogo.setMessage("Soma:" + res);
      dialogo.setNeutralButton("OK", null);
      dialogo.show();

                  }

            });
```

Toda vez que eu clicar no botão, ele irá mostrar o resultado da soma na tela através de uma caixa de mensagem. Ótimo! Vamos executar a nossa aplicação? Para executar faça os mesmos procedimentos que já mostrei. O resultado da execução dessa aplicação você vê na figura abaixo:

(Aplicação que soma números)

OBS: Provavelmente durante a execução da aplicação ao entrar com um número, deve ter surgido no dispositivo um teclado virtual (como mostra a figura acima), para ocultar ele é só pressionar ESC.

Irei descrever o código do evento Click. O método setOnClickLisneter serve para definir um evento de Click a um componente. Como parâmetro, criamos uma instância de OnClickListener e dentro dessa instância existe o método chamado onClick, que será disparado toda vez que o botão for clicado.

A linha:

```
double num1 =  Double.parseDouble(ednumero1.getText().toString());
```

Cria uma variável chamada *num1* e atribui a ela o valor que está contido em *num1*. Eu faço uso do método parseDouble da classe Double pois o conteúdo é uma String. Observem que chamo o método getText de *ednumero1* para retornar o conteúdo. Diferente de muitos métodos de retorno String, esse método getText não retorna uma String, mais sim um tipo chamado *Editable*. Por isso, chamei o método toString de getText para que me retornasse uma string. A descrição da próxima linha e a mesma.

O código abaixo:

```
AlertDialog.Builder dialogo = new AlertDialog.Builder(AppSoma.this);
      dialogo.setTitle("Aviso");
      dialogo.setMessage("Soma:" + res);
      dialogo.setNeutralButton("OK", null);
      dialogo.show();
```

É responsável por mostrar a soma na tela , através da classe AlertDialog.Builder, responsável por criar caixas de diálogo e exibi-las.

Beleza! Com esse conhecimento obtido até agora, você já tem capacidade para fazer uma aplicação básica em Android.

5.5) A widget CheckBox

A widget **CheckBox** funciona como um componente que pode ser marcado e desmarcado, e que possui também um rótulo.

5.6) Desenvolvendo uma aplicação simples de compras

Agora vamos fazer uma outra aplicação Android que vai fazer uso da widget CheckBox, que acabamos de conhecer acima. Nossa aplicação consiste em um simples sistemas de compras onde possuo cinco produtos, Arroz (R$ 2,69) , Leite (R$ 5,00) , Carne (R$ 10,00), Feijão (R$ 2,30) e Refrigerante coca-cola (R$ 2,00). Nessa aplicação eu marco os itens que quero comprar e no final o sistema mostra o valor total das compras.

Bom, vamos criar um novo projeto chamado SistemaDeCompras. Siga os dados do projeto abaixo:

Project Name: SistemaDeCompras

Package Name : br.com.android

Create Activity: AppCompra

Application Name: Sistema de Compras

Min SDK Version: 7

Vamos no arquivo “main.xml” desse projeto para carregarmos o utilitário. Depois de carregado, modifique o valor da propriedade Text da TextView com a frase “Digite o seu produto”. Feito isso, adicione os seguintes componentes, na seqüência:

CheckBox

Propriedade	Valor
Text	Arroz (R$ 2,69)
Id	@+id/chkarroz

CheckBox

Propriedade	Valor
Text	Leite (R$ 5,00)
Id	@+id/chkleite

CheckBox

Propriedade	Valor
Text	Carne (R$ 9,70)
Id	@+id/chkcarne

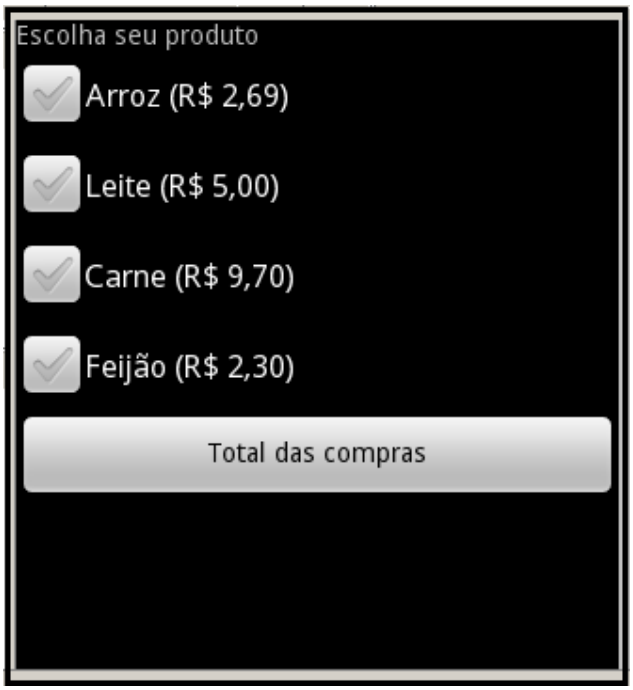
CheckBox

Propriedade	Valor
Text	Feijão (R$ 2,30)
Id	@+id/chkfeijao

Button

Propriedade	Valor
Text	Total das compras
Id	@+id/btotal
Layout_width	fill_parent

Ao final, o layout da nossa aplicação deve estar de acordo com a figura abaixo:



Agora vamos modificar o arquivo "AppCompra.java". O código desse arquivo será como o código que é exibido abaixo:

```
package br.com.android;

import android.app.AlertDialog;
import android.os.Bundle;
import android.widget.*;
import android.view.*;
import android.app.*;


public class AppCompra extends Activity {



    CheckBox chkarroz,chkleite,chkcarne,chkfeijao;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        chkarroz = (CheckBox) findViewById(R.id.chkarroz);
        chkleite = (CheckBox) findViewById(R.id.chkleite);
        chkcarne = (CheckBox) findViewById(R.id.chkcarne);
        chkfeijao = (CheckBox) findViewById(R.id.chkfeijao);


        Button bttotal = (Button) findViewById(R.id.bttotal);

        bttotal.setOnClickListener(new View.OnClickListener(){
```

```
                public void onClick(View arg0) {

                      double total =0;

                      if(chkarroz.isChecked())
                            total += 2.69;

                      if(chkleite.isChecked())
                            total += 5.00;
                      if(chkcarne.isChecked())
                            total += 9.7;

                      if(chkfeijao.isChecked())
                            total += 2.30;



                      AlertDialog.Builder dialogo = new
AlertDialog.Builder(AppCompra.this);
                       dialogo.setTitle("Aviso");  //Defino o título
                      dialogo.setMessage("Valor total da compra :" +
String.valueOf(total)); //colocando a mensagem que vai ter dentro do
Dialog
                dialogo.setNeutralButton("OK", null); //adicionando o
botão de OK

                dialogo.show(); //mostrando o Dialog

                }

        });

    }
}
```

Descrevendo o código do evento onClick : Dentro do evento eu crio uma variável chamada *total* que armazena o valor total da compra. Observe que eu tenho quatro estruturas *if's* onde cada uma verifica se um determinado item foi marcado, se foi, incrementa o valor do item com o valor da variável *total*. No final mostro valor total das compras na tela.

Vamos roda nossa aplicação? O resultado você confere na figura abaixo:

(Aplicação simples de compras)

5.7) A widget RadioButton

A widget **RadioButton** é um componente muito utilizado em opções de múltipla escolha, onde somente uma única opção pode ser selecionada.

5.8) Desenvolvendo uma aplicação de cálculo de salário (Com RadioButton)

Bom, agora vamos fazer uma outra aplicação. Essa aplicação que vamos desenvolver agora consiste em um sistema que vai ler o salário de um funcionário e vai permitir que você escolha o seu percentual de aumento que pode ser de 40% , 45% e 50% e no final o sistema irá mostrar o salário reajustado com o novo aumento.

Bom, vamos lá! Crie um novo projeto Android com os seguintes dados:

Project Name: CalculoDeSalario

Package Name : br.com.android

Create Activity: AppSalario

Application Name: Cálculo do salário

Min SDK Version: 7

Nessa primeira versão da aplicação, como havia falado, vamos fazer uso da widget **RadioButton**.

Carregado o arquivo "main.xml", modifique a propriedade Text da TextView com a frase "Digite seu salário (R$)". Em seguida adicione os seguintes componentes, na seqüência :

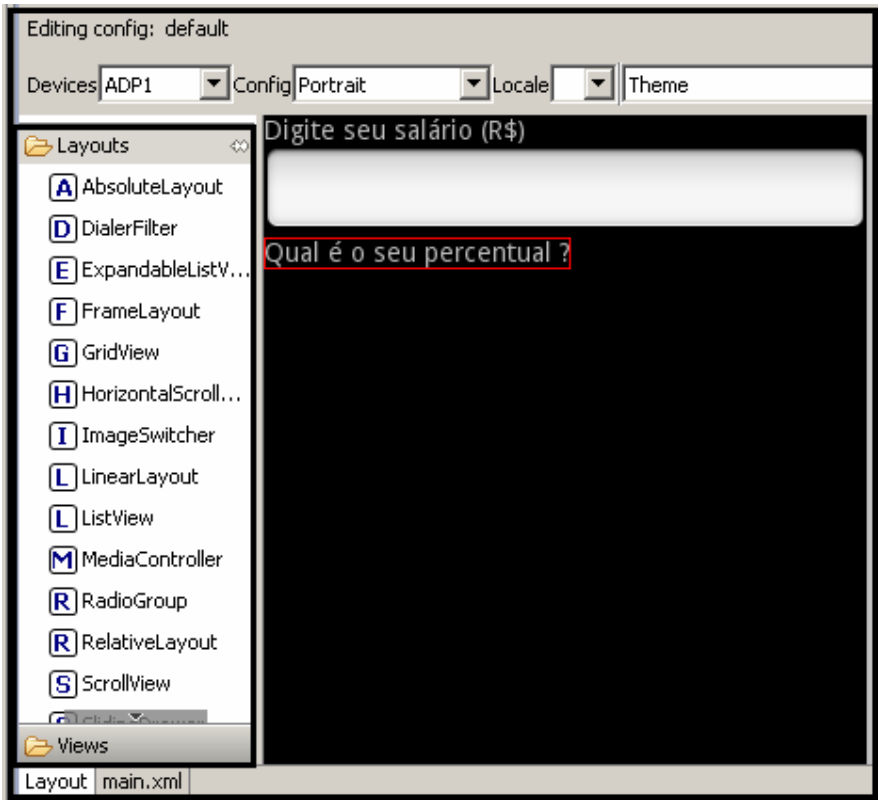
EditText

Propriedade	Valor
Text	
Id	@+id/edsalario
Layout_width	fill_parent

TextView

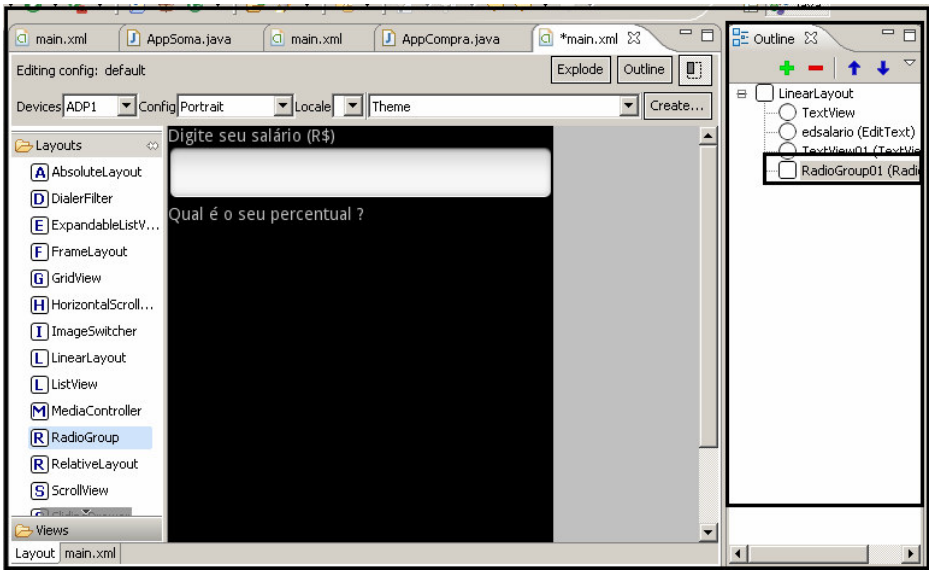
Propriedade	Valor
Text	Qual é o seu percentual ?

Bom, agora vamos adicionar um componente , ou melhor, uma estrutura, que será responsável por agrupar as RadioButtons dentro dela.O nome dessa estrutura se chama "RadioGroup" e ela se encontra dentro da seção "Layouts", conforme você confere na figura abaixo:



Para adicionar este componente no dispositivo simplesmente clique e arraste ele até a tela do dispositivo. Se observar, não mostra nenhum

componente selecionado no dispositivo mas, se você observar a direita existe uma seção chamada “Outline”, que todos os componentes situados na tela do dispositivo, visíveis ou não, conforme você confere na figura abaixo:

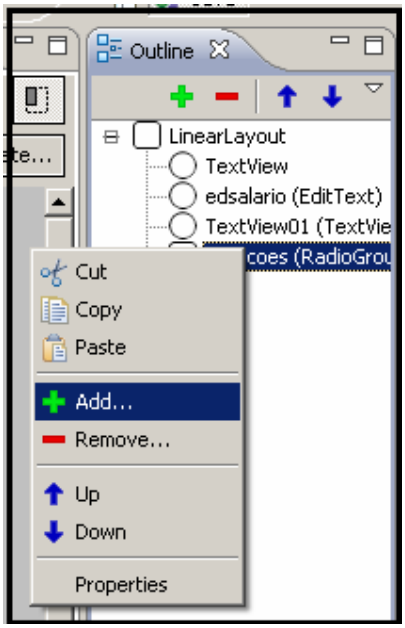


Se você observar na figura acima, mesmo não mostrando nenhum componente selecionado no dispositivo, o componente corrente em edição é o RadioGroup, pelo fato de ele estar vazio, sem nenhuma RadioButton.

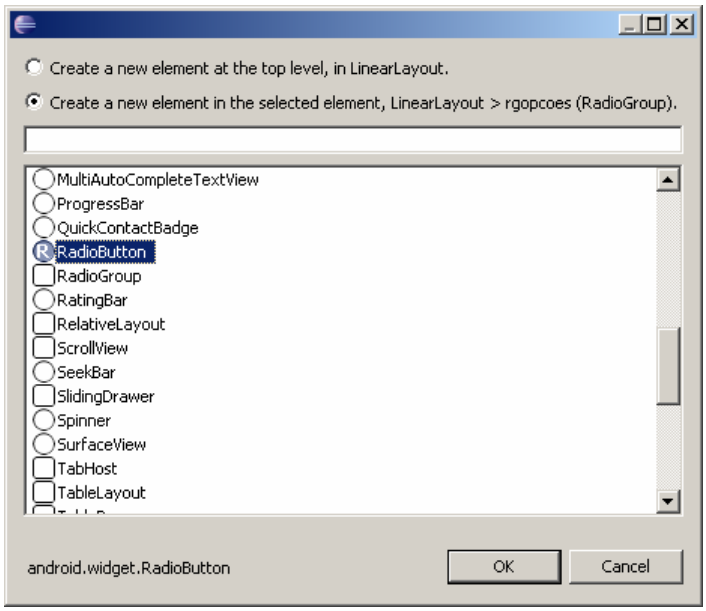
Com o RadioGroup selecionado, modifique as propriedades abaixo:

Propriedade	Valor
Orientation	vertical
Id	@+id/rgopcoes

Agora, vamos inserir as RadioButtons dentro dele. Como faremos isso? Na seção “Outline”, clique com o botão direito do sobre componente RadioGroup “rgopcoes” e surgirá um menu, selecione a opção “Add”, conforme mostra a figura abaixo:



Após selecionar a opção “Add’, será aberta uma caixa de dialogo com uma lista completa de componentes para você poder adicionar sobre a RadioGroup. No momento, só iremos adicionar RadioButtons nele logo, selecione o componente RadioButton, conforme mostra a figura abaixo:



Depois disso é só clicar em “OK” para que o componente seja inserido na RadioGroup. Com o RadioButton selecionado, modifique as seguintes propriedades abaixo:

Propriedade	Valor
Text	40%
Id	@+id/rb40

Agora vamos adicionar mais duas RadioButtons dentro da nossa RadioGroup “rgopcoes”, para isso repita os mesmos procedimentos acima. Depois de adicionados as RadioButtons, modifique as propriedades deles, conforme abaixo:

RadioButton1

Propriedade	Valor
Text	45%
Id	@+id/rb45

RadioButton2

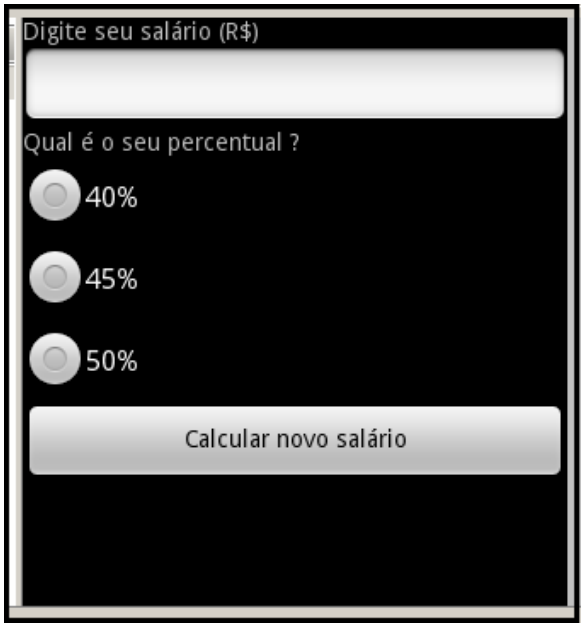
Propriedade	Valor
Text	50%
Id	@+id/rb50

Agora, vamos adicionar uma Button, simplesmente clicando e arrastando o componente na tela. Agora um detalhe, é para colocar esse componente na tela do dispositivo mas FORA da área do RadioGroup.

Depois de colocar o Button, modifique as propriedades abaixo:

Propriedade	Valor
Text	Calcular novo salário
Id	@+id/btcalcular
Layout_width	fill_parent

Depois de inserir todos os componentes citados, o layout da aplicação deve ficar de acordo com a figura abaixo:



No arquivo “AppSalario.java”, coloque o código abaixo:

```
package br.com.android;


import android.app.Activity;
import android.os.Bundle;
import android.widget.*;
import android.view.*;
import android.app.*;

public class AppSalario extends Activity {



    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btcalcular = (Button) findViewById(R.id.btcalcular);

        btcalcular.setOnClickListener(new View.OnClickListener(){

                  public void onClick(View arg0) {

                        double salario, novo_sal;

                        EditText edsalario = (EditText)
findViewById(R.id.edsalario);



                        salario =
Double.parseDouble(edsalario.getText().toString());

                        RadioGroup rg = (RadioGroup)
findViewById(R.id.rgopcoes);


                        int op = rg.getCheckedRadioButtonId();


                        if(op==R.id.rb40)
                              novo_sal = salario + (salario * 0.4);
                        else
                              if(op==R.id.rb45)
                                    novo_sal = salario + (salario *
0.45);
                              else
                                    novo_sal = salario + (salario *
0.5);


                        AlertDialog.Builder dialog = new
AlertDialog.Builder(AppSalario.this);
                      dialog.setTitle("Novo salário");
```

```
                        dialog.setMessage("Seu novo salário é : R$" +
String.valueOf(novo_sal));
                        dialog.setNeutralButton("OK", null);
                        dialog.show();




                    }


        });
    }
     }
```

Vamos à explicação de alguns códigos interessantes. Dentro do evento Click, eu realizo o cálculo do novo salário do funcionário. Os primeiros códigos do evento são similares de programas anteriores que já foram devidamente explicados. A linha:

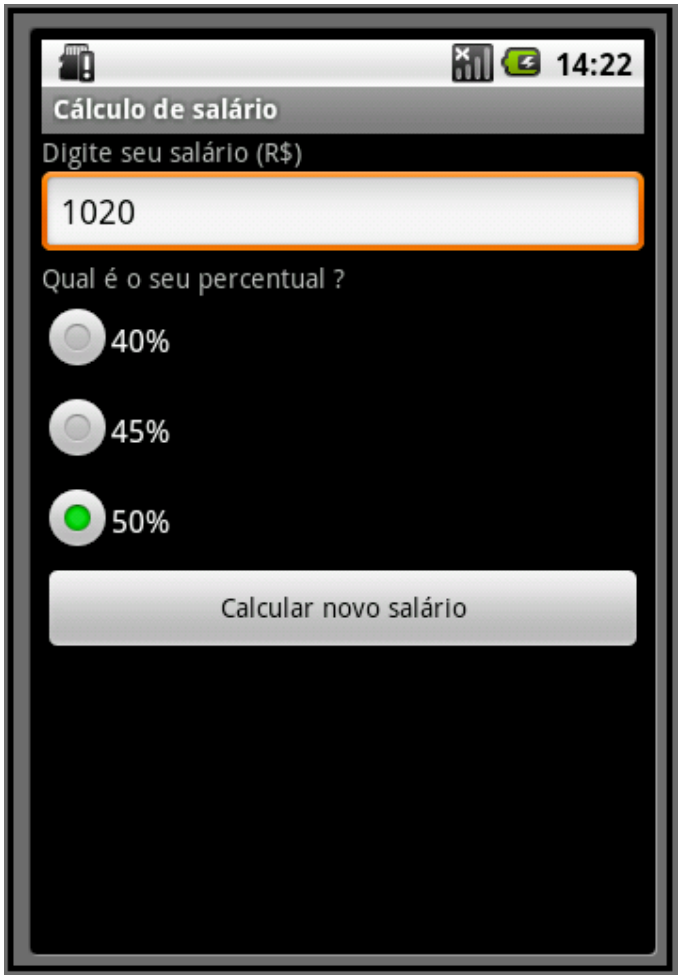
```
int op = rg.getCheckedRadioButtonId();
```

Cria uma variável *op* e retorna para ela o Id da opção selecionada, ou seja, qual RadioButton foi selecionada.

Agora na condição:

```
if(op==R.id.rb40)
```

Verifico se a opção de 40% foi selecionada, se for, realize o cálculo do salário com o reajuste de 40%. A mesma explicação e válida para o cálculo dos outros reajustes. Agora vamos executar a nossa aplicação. O resultado você vê na figura abaixo:



(Aplicação de cálculo de salário)

5.9) A widget Spinner

A widget **Spinner** é um componente do tipo caixa de combinação (“ComboBox”) onde nele é armazenado vários itens a serem selecionados. Para que um componente possa ser selecionado, é preciso clicarmos na seta , para que os itens possam ser mostrados e , por conseqüência, serem selecionados.

5.10) Desenvolvendo uma aplicação de cálculo de salário (Com Spinner)

Bom, agora vamos criar a nossa segunda versão do aplicativo acima, usando agora o componente Spinner. Crie um novo projeto Android com os seguintes dados:

Project Name: CalculoDeSalarioSpinner

Package Name : br.com.android

Create Activity: AppSalario

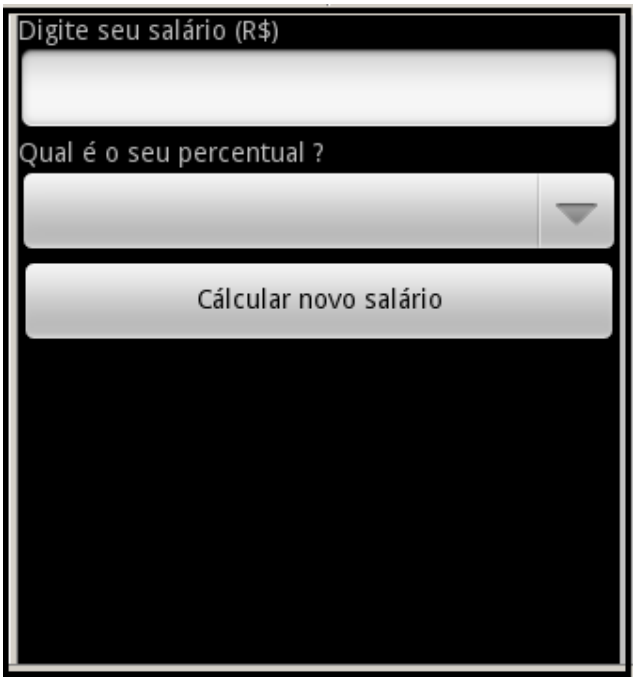
Application Name: Cálculo do salário

Min SDK Version: 7

Nessa segunda versão da aplicação, vamos fazer uso da widget Spinner. Carregue o arquivo "main.xml" e faça os mesmos procedimentos do programa anterior, só que ao invés de adicionar a RadioGroup com os RadioButtons, você vai inserir somente um componente Spinner. Segue abaixo as propriedades que você precisa modificar:

Propriedade	Valor
Id	@+id/spnopcoes
Layout_width	fill_parent

Seguindo os passos, a aplicação deve estar de acordo com a figura abaixo:



Agora no arquivo "AppSalario.java", coloque o seguinte código:

```
package br.com.android;

import android.app.Activity;
```

```
import android.os.Bundle;
import android.widget.*;
import android.view.*;
import android.app.*;

public class AppSalario extends Activity {



      private  static final String[] percentual = {"De 40%","De
45%","De 50%"};
      ArrayAdapter<String> aPercentual;
      Spinner spnsal;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btmostrar = (Button) findViewById(R.id.btcalcular);

        aPercentual = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,
percentual);

       spnsal = (Spinner) findViewById(R.id.spnopcoes);

       spnsal.setAdapter(aPercentual);



    btmostrar.setOnClickListener(new View.OnClickListener(){

    public void onClick(View arg0) {

      double salario=0, novo_sal = 0;

       EditText edsalario = (EditText) findViewById(R.id.edsalario);



      salario = Double.parseDouble(edsalario.getText().toString());

      switch(spnsal.getSelectedItemPosition()) {
            case 0: novo_sal = salario + (salario * 0.4); break;
            case 1: novo_sal = salario + (salario * 0.45); break;
            case 2: novo_sal = salario + (salario * 0.5); break; }

      AlertDialog.Builder dialogo = new
AlertDialog.Builder(AppSalario.this);
    dialogo.setTitle("Novo salário");
      dialogo.setMessage("Seu novo salário é : R$" +
String.valueOf(novo_sal));
      dialogo.setNeutralButton("OK", null);
      dialogo.show();


                }
```

```
        });
    }


    }
```

Observando o código do programa acima, podemos ver que ele é similar o da primeira versão do aplicativo, porém, quero comentar alguns códigos interessantes desta aplicação. Observe que foi necessário declarar um "array" de "String" chamado *percentual*, conforme mostra o código abaixo:

```
    private  static final String[] percentual = {"De 40%","De 45%","De
50%"};
```

Este "array" possui três elementos, correspondentes ao percentual do aumento do salário. Também foi necessário declarar um objeto do tipo ArrayAdapter chamado *aPercentual*. Esse objeto serve para fazer referencia ao "array" *percentual*. Dentro do método **OnCreate**, existe uma linha de código abaixo:

```
    aPercentual = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,
percentual);
```

Que cria uma instância da classe ArrayAdapter e atribuo essa instância ao objeto *aPercentual,* onde carrego nele o "array" de Strings *percentual*. Logo depois, vem a instrução:

```
     spnsal.setAdapter(aPercentual);
```

Onde carrego no objeto do tipo Spinner uma lista de opções de percentual.

Vamos agora dentro do evento OnClick do objeto Button. Dentro existe o código mostrado abaixo:

```
    switch(spnsal.getSelectedItemPosition()) {
           case 0: novo_sal = salario + (salario * 0.4); break;
           case 1: novo_sal = salario + (salario * 0.45); break;
           case 2: novo_sal = salario + (salario * 0.5); break; }
```

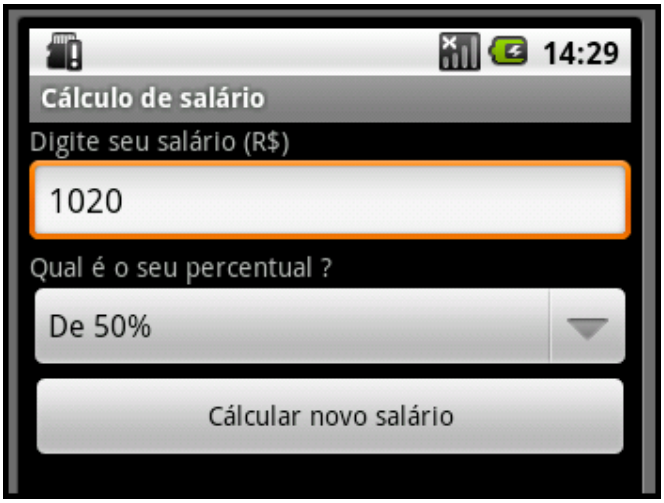
Que verifica qual será o novo salário , de acordo com a opção selecionada no objeto Spinner. Vamos entender esse código.

Observe que o objeto *spnsal*, possui um método chamado getSelectedItemPosition, que é responsável por retornar o índice do item selecionado, sabendo se que o primeiro item possui índice zero, o segundo possui índice um e assim por diante. Observe que dentro dessa estrutura eu

verifico a opção selecionada, se for a primeira, o novo salário terá aumento de 40%, se for a segunda , o aumento será de 45% senão, o aumento será de 50%.

Logo após o cálculo do novo salário, é exibido na tela o novo salário.

Vamos executar a nossa aplicação. O resultado da execução você confere na figura abaixo:



(Aplicação de calculo de salário usando o Spinner)

5.11) A widget ListView

A Widget **ListView** é um componente que possui vários itens a serem selecionados, similar ao componente Spinner. A única diferença entre o ListView e o Spinner é que no componente ListView, os itens já são mostrados sem nenhuma necessidade de se clicar em alguma parte dele para que os mesmos possam ser mostrados.

5.4) Desenvolvendo uma aplicação de lista telefônica

Agora vamos fazer uma nova aplicação em Android. Essa aplicação consiste em uma lista telefônica já pronta com contatos. Quando selecionamos um contato, ele mostra na tela uma mensagem com o nome selecionado. A nossa aplicação vai fazer uso do widget chamado ListView, que exiba uma lista contendo valores que podem ser selecionados.

Bom, vamos criar um novo projeto. Siga os dados abaixo:

Project Name: ListaTelefonica

Package Name : br.com.android

Create Activity: AppLista

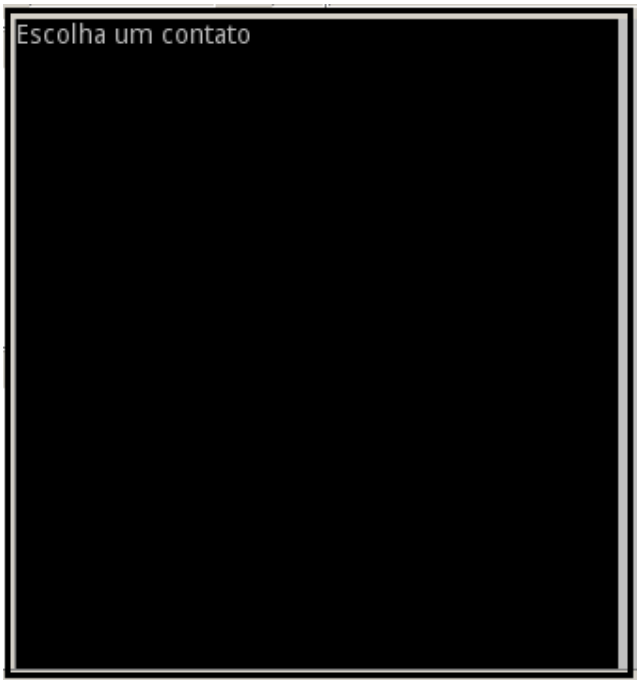
Application Name: Lista de contatos

Min SDK Version: 7

Vamos no layout do nosso arquivo main.xml e vamos modificar o conteúdo na TextView, com a frase : “Escolha um contato”. Depois disso, vamos inserir um ListView (que se encontra na guia “Layouts”) e depois modificar suas propriedades, conforme abaixo:

Propriedade	Valor
Id	@+id/lstcontatos
Layout_width	fill_parent

A aplicação depois de feito todos os passos acima, deve estar de acordo com a figura abaixo:



Pelo fato do ListView ser uma estrutura de layout não um componente, ele , como aconteceu com o RadioGroup, está vazio e no dispositivo não mostra ele selecionado.

No arquivo AppList.java, coloque o seguinte código:

```
package br.com.android;
```

```
import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.widget.*;
import android.widget.AdapterView.OnItemClickListener;
import android.view.*;

public class AppLista extends Activity {

    public ListView lista;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,
contatos);
        lista = (ListView) findViewById(R.id.lstcontatos);
        lista.setAdapter(adapter);



        lista.setOnItemClickListener(new OnItemClickListener(){

                public void onItemClick(AdapterView arg0, View arg1,
int arg2,
                                long arg3) {

         if(lista.getSelectedItem()!=null) {


          AlertDialog.Builder dialogo = new
AlertDialog.Builder(AppLista.this);
          dialogo.setTitle("Contato selecionado");
            dialogo.setMessage(lista.getSelectedItem().toString());
            dialogo.setNeutralButton("OK", null);
            dialogo.show();




         }

                }

        });


    }

    static final String[] contatos = new String[] {
        "Alline","Lucas","Rafael","Gabriela","Silvana"
    };



}
```

Vamos analisar alguns códigos acima. A linha:

```
static final String[] contatos = new String[] {
    "Alline","Lucas","Rafael","Gabriela","Silvana"
};
```

Cria uma constante chamada contatos, onde nela coloco alguns nomes. Essa constante vai ser utilizada pela nossa lista. Para que eu possa carregar dos dados em uma ListView, preciso fazer uso da classe ArrayAdapter, como mostra a instrução abaixo:

```
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout. simple_list_item_1,
contatos);
```

A instrução mostrada acima cria uma instância da classe ArrayAdapter chamada *adapter* onde carrego nela o vetor de Strings da constante *contatos*. A instrução:

```
lista.setAdapter(adapter);
```

Carrega os valores para a ListView, que está contido o objeto *adapter*.

Como havia falado, quando se clica em um item, o sistema mostraria uma mensagem do item selecionado. Isso é conseguido fazendo uso da interface OnItemClickListener, como mostra a instrução abaixo:

```
lista.setOnItemClickListener(new OnItemClickListener(){

    public void onItemClick(AdapterView arg0, View arg1, int
arg2,long arg3) {

if(lista.getSelectedItem()!=null)

  AlertDialog.Builder dialogo = new
AlertDialog.Builder(AppLista.this);
  dialogo.setTitle("Contato selecionado");
  dialogo.setMessage(lista.getSelectedItem().toString());
  dialogo.setNeutralButton("OK", null);
  dialogo.show();

                }

    });
```

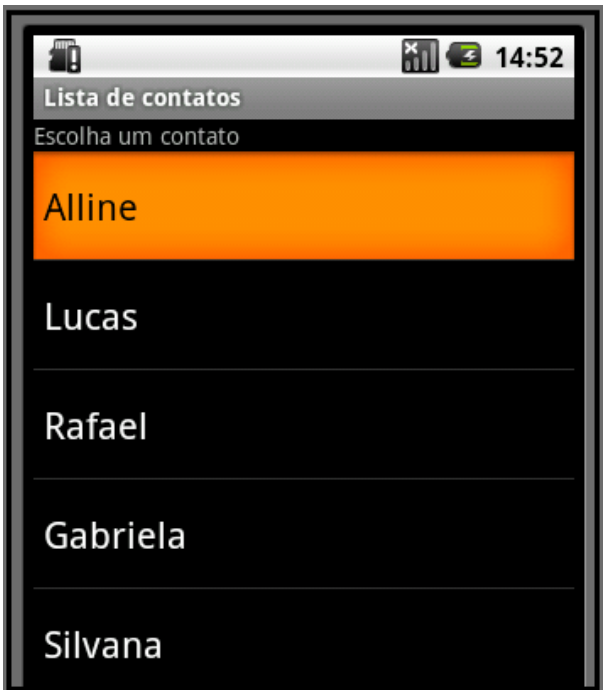
Toda vez que clicarmos em um item da lista, o método onItemClick será disparado e será executado o comando abaixo:

```
if(lista.getSelectedItem()!=null) {
    AlertDialog.Builder dialogo = new
AlertDialog.Builder(AppLista.this);
    dialogo.setTitle("Contato selecionado");
```

```
    dialogo.setMessage(lista.getSelectedItem().toString());
    dialogo.setNeutralButton("OK", null);
    dialogo.show();

}
```

Que exibe o nome do item selecionado, se ele estiver selecionado. A obtenção do item clicado e feita chamando o método getSelectedItem() . Porém, como ele retorna um tipo Object, preciso converter para String o item clicado, através do método toString.

Vamos executar a aplicação. O resultado você vê na figura abaixo:



(Aplicação de lista de contatos)

5.12) A widget Imageview

A widget **ImageView** é um componente que permite que visualizemos imagens dentro dele. As imagens suportadas por esse componente são imagens no formato JPEG, GIF e PNG.

5.13) Desenvolvendo uma aplicação que visualiza imagens (Com ImageView)

Agora vamos desenvolver uma aplicação que visualiza imagens, usando o componente ImageView.

Agora crie um novo projeto conforme os dados abaixo:

Project Name: VisualizadorDeImagens

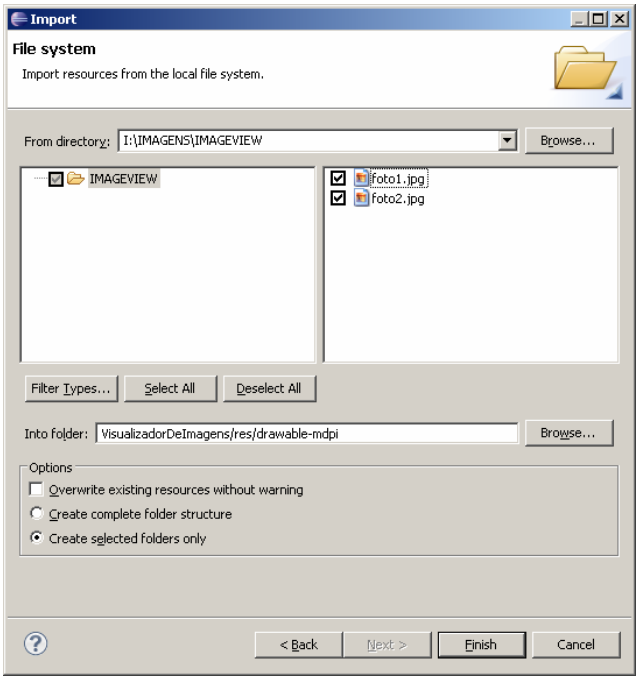
Package Name : br.com.android

Create Activity Name: AppImagem

Application Name: Visualizador de Imagens

Min SDK Version: 7

Antes de iniciarmos a codificação do programa, quero que você coloque duas imagens JPEG (com a extensão .jpg) que acompanham este material e que se encontram da pasta “IMAGEVIEW” , dentro da pasta “res/drawable-mdpi”. Para importar um arquivo, clique com o botão direito do mouse sobre a pasta “res/drawable-mdpi” e selecione “Import”, depois selecione File System (Que se encontra dentro da pasta “General”). Clique no botão browser para selecionar o diretório onde se encontram as imagens, depois de selecionado, marque os dois arquivos (imagens) para que eles sejam importados para a pasta “res/drawable-mdpi” . Veja a figura abaixo:

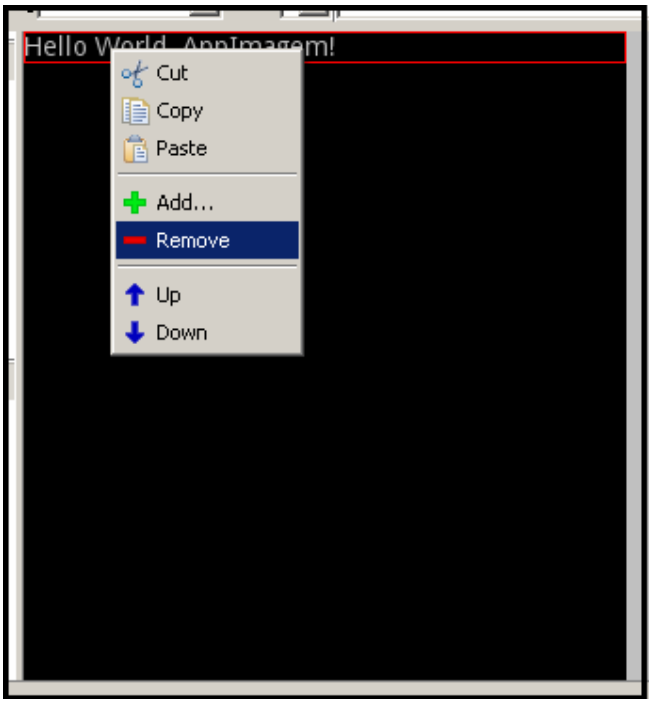


(Importando imagens para aplicação)

Depois disso, é só clicar em “Finish”.

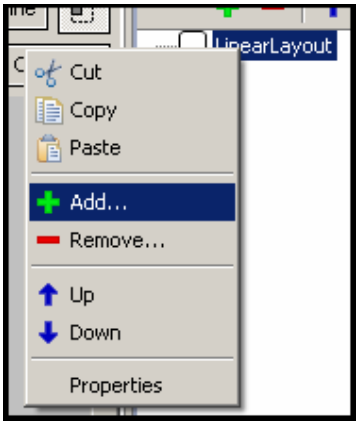
Agora no layout do arquivo “main.xml”, apague o componente TextView que se encontra na tela do dispositivo. Se você acha que para deletar um componente do layout do dispositivo é simplesmente selecionando ele e pressionar DELETE ? Se enganou. Para apagar o componente selecione e

clique com o botão direito do mouse sobre ele e escolha a opção “Remove”,conforme figura abaixo:

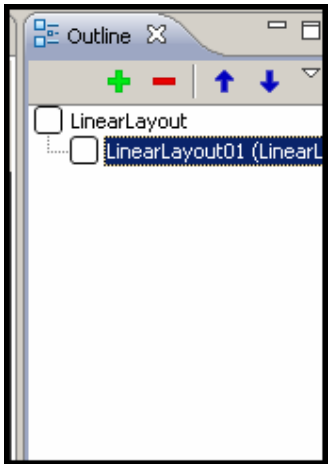


Agora siga os passos abaixo para construirmos a nossa aplicação.

Vá na seção “OutLine” e clique com o botão direito sobre o LinearLayout e selecione a opção “Add”, conforme figura abaixo:



Agora vamos adicionar um outro “LinearLayout” dentro dele. O resultado você confere na seção Outline, da figura abaixo:



Agora nessa estrutura “LinearLayout’ que inserimos, vamos modificar a seguinte propriedade abaixo:

Propriedade	Valor
Orientation	horizontal

Agora dentro da estrutura “LinearLayout” que configuramos acima, vamos inserir os seguintes componentes, na seqüência (use o mesmo procedimento que fiz para inserir a segunda estrutura “LinearLayout”):

ImageView

Propriedade	Valor
Id	@+id/imagem
Src	@drawable/foto1

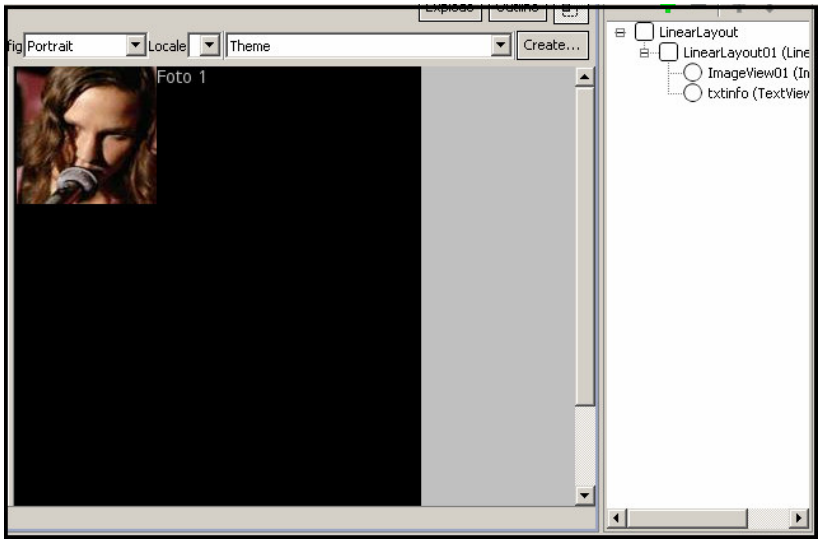
Bom antes de prosseguir, quero explicar a propriedade “Src”. Nessa propriedade definimos a imagem corrente que irá aparecer na tela que é especificada pela notação *“@drawable/foto1”*, irei explicar essa notação.

Se você notou, quando importamos as duas imagens que seriam utilizadas pelo nosso programa, essas imagens ficaram dentro do diretório “drawable-mdpi” certo ? Porém, quando especificamos pela propriedade “Src” o nome do diretório das imagens sempre será @drawable. Outro detalhe: Quando especificamos o nome do arquivo de imagem, o nome do arquivo não pode ter a extensão dele, isso é regra.

TextView

Propriedade	Valor
Id	@+id/txtinfo
Text	Foto 1

Seguindo os passos acima, o resultado do layout deve ficar de acordo com a figura abaixo:



Agora vamos colocar na seqüência dois buttons, só que esses dois componentes vão estar dentro da primeira estrutura LinearLayout, ou seja , da estrutura principal. Segue abaixo as propriedades que precisam ser modificadas:

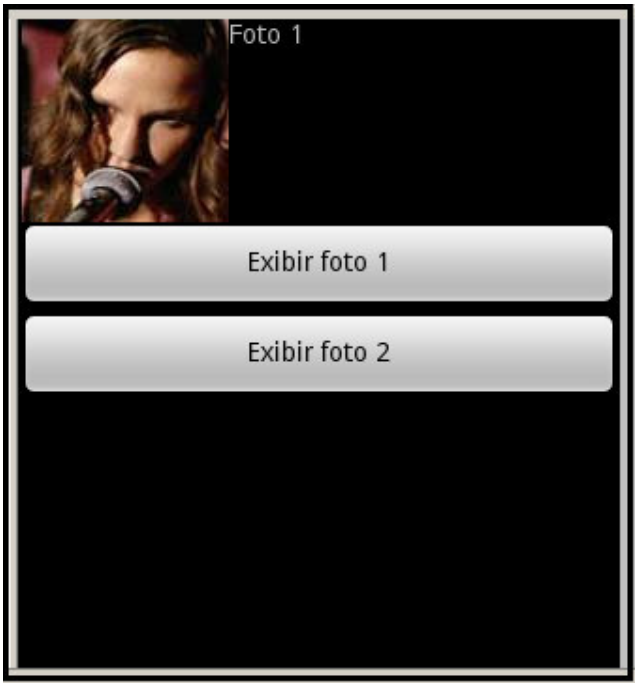
Button1

Propriedade	Valor
Id	@+id/btimagem1
Text	Exibir foto 1
Layout_width	fill_parent

Button2

Propriedade	Valor
Id	@+id/btimagem2
Text	Exibir foto 2
Layout_width	fill_parent

Depois de seguir todos os passos descritos acima, a aplicação tem que estar de acordo com a figura abaixo:



Agora no arquivo AppImage.java coloque o código abaixo:

```
package br.com.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.*;
import android.widget.*;


public class AppImagem extends Activity {

      ImageView imagem;
      TextView txt;
    @Override

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btfoto1 = (Button) findViewById(R.id.btimagem1);
        Button btfoto2 = (Button) findViewById(R.id.btimagem2);
        imagem = (ImageView) findViewById(R.id.imagem);
        txt = (TextView) findViewById(R.id.txtinfo);
         btfoto1.setOnClickListener(new View.OnClickListener(){

                  public void onClick(View arg0) {

                        imagem.setImageResource(R.drawable.foto1);
                        txt.setText("Foto 1");

                  }
              });
```

```
        btfoto2.setOnClickListener(new View.OnClickListener(){

                public void onClick(View arg0) {

                        imagem.setImageResource(R.drawable.foto2);
                        txt.setText("Foto 2");

                }

        });

    }
}
```

Agora vamos analisar alguns trechos de códigos. Vamos no evento Click referente a abertura da primeira imagem. O código:

```
imagem.setImageResource(R.drawable.foto1);
```

É responsável por abrir a imagem "foto1.jpg" e exibi-la no componente. Observe que foi passado o parâmetro *"R.drawable.foto1"* onde *"drawable"* corresponde a pasta e "*foto1"* corresponde ao arquivo *"foto1.jpg"*. Logo após vem o código:

```
txt.setText("Foto 1");
```

Cuja finalidade é mudar o título da TextView , de acordo com a String passada como parâmetro.

O comentário acima é o mesmo para o segundo botão referente a abertura da segunda imagem.

Vamos executar a nossa aplicação. O resultado você vê nas imagens abaixo:

(Aplicação com a primeira foto em exibição)

(Aplicação com a segunda foto em exibição)

5.14) A widget Gallery

Na aplicação anterior, fizemos uso do widget ImageView. Note que usamos a mesma widget para visualizar duas imagens distintas. Agora nesta segunda aplicação vamos fazer uso da widget **Gallery**. Essa widget funciona como um grupo do ImageViews onde cada foto pode ser visualizada simplesmente arrastando o mouse ou clicando nas setas direitas ou esquerda do SmartPhone . Veja na figura abaixo uma aplicação que faz uso dessa estrutura:

5.15) Desenvolvendo uma aplicação que visualiza imagens (Com Gallery)

Bom, vamos a nossa aplicação. Crie um novo projeto de acordo com os dados abaixo:

Project Name: GaleriaDeImagens

Package Name : br.com.android

Create Activity: AppGallery

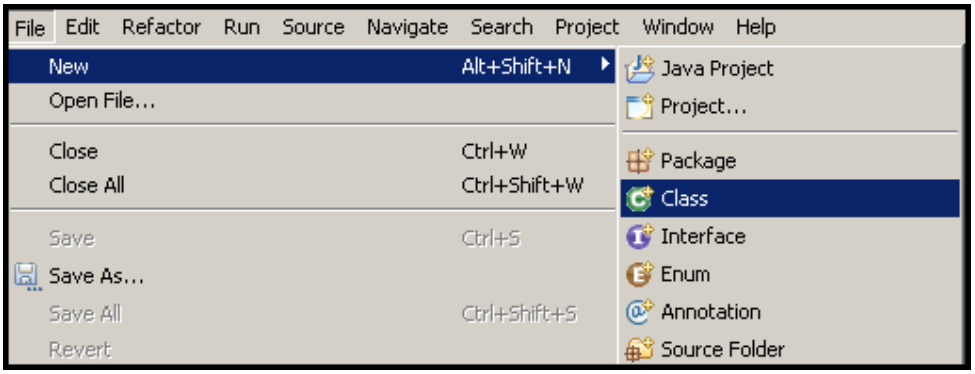
Application Name: Galeria de Imagens

Min SDK Version: 7

Depois de criado o projeto, coloque na pasta de imagens do android (res/drawable-mdpi), três imagens que se encontram dentro da pasta "GALLERY", que acompanham este material. O nome dos arquivos são "imagem1.jpg", "imagem2.jpg" e "imagem3.jpg".

Agora vamos criar uma classe chamada "ImageAdapter", que será uma classe iremos utilizar em nossa aplicação e essencial para o funcionamento do componente Gallery. Siga os passos abaixo:

Vá no menu File/New/Class, conforme mostra a figura abaixo:



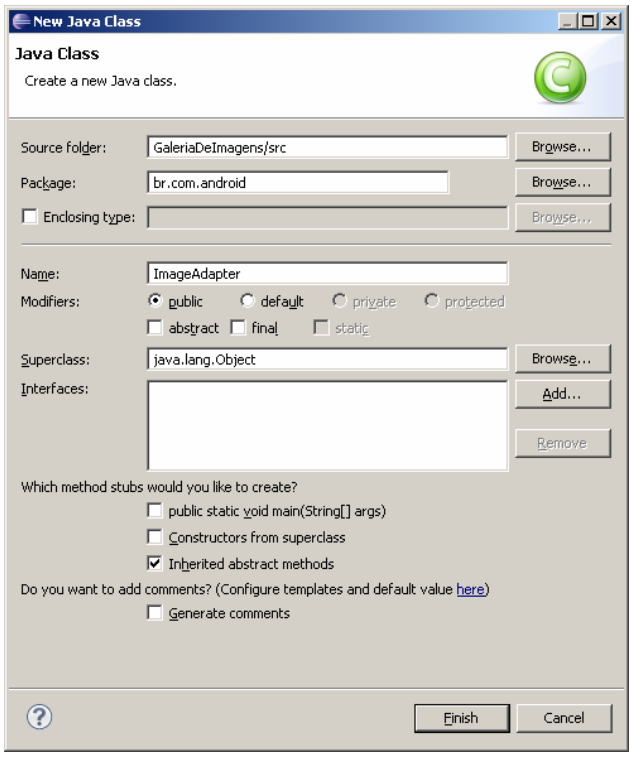
Agora, preencha os dados abaixo:

Source folder : GaleriaDeImagens/src

Package: br.com.android

Name: ImageAdapter

Seguindo os passos acima, os dados devem estar de acordo com a figura abaixo:



Se estiver tudo OK, é só pressionar o botão “Finish”, para que a classe possa ser criada e logo em seguida coloque o código abaixo:

```
package br.com.android;


import android.content.Context;
import android.view.*;
import android.widget.*;

public class ImageAdapter extends BaseAdapter {



    private Context myContext;

    /* Neste array são colocadas as imagens a serem exidas
      no componente Gallery .*/

    private int[] myImageIds = {
           R.drawable.imagem1,
           R.drawable.imagem2,
           R.drawable.imagem3,
           };

    public ImageAdapter(Context c) { this.myContext = c; }

    public int getCount() { return this.myImageIds.length; }
```

```
    public Object getItem(int position) { return position; }
    public long getItemId(int position) { return position; }

    public View getView(int position, View convertView, ViewGroup
parent) {
        ImageView i = new ImageView(this.myContext);

        i.setImageResource(this.myImageIds[position]);

        i.setScaleType(ImageView.ScaleType.FIT_XY);

        i.setLayoutParams(new Gallery.LayoutParams(150, 150));
        return i;
    }

    public float getScale(boolean focused, int offset) {
        return Math.max(0, 1.0f / (float)Math.pow(2,
Math.abs(offset)));
    }

}
```

Observem que dentro desta classe existe um array chamado *"myImageIds",* onde eu armazeno as imagens a serem visualizadas no componente.

Agora , carregue o arquivo main.xml e modifique o conteúdo da TextView com a frase: "Visualização de Imagens" e em seguida, insira no dispositivo o componente Gallery. Modifique as propriedades do componente Gallery conforme é mostrado abaixo:

Propriedade	Valor
Id	@+id/gallery
Layout_width	fill_parent

Feito isso, vamos no arquivo "AppGallery.java" , e vamos modifica-lo com o seguinte código abaixo:

```
package br.com.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.*;

public class AppGallery extends Activity {
       public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.main);

              ((Gallery) findViewById(R.id.gallery))
                     .setAdapter(new ImageAdapter(this));

    }
```

```
}
```

Agora vamos executar a nossa aplicação. O resultado da execução você confere nas figuras abaixo:

(Aplicação com a primeira foto em exibição)

(Aplicação com a segunda foto em exibição)

(Aplicação com a terceira foto em exibição)

Agora vamos tornar essa aplicação mais interessante. Vamos colocar nessa aplicação um ImageView, que irá armazena a imagem seleciona no componente Gallery. Bom, carregue novamente o arquivo main.xml e em seguida, coloque os seguintes componentes, na seqüência:

TextView

Propriedade	Valor
Text	Imagem selecionada

ImageView

Propriedade	Valor
Id	@+id/imagem
Layout_width	fill_parent

Agora no arquivo "AppGallery.java" vamos substituir o código recente pelo novo código abaixo:

```
package br.com.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.*;

import android.widget.*;

public class AppGallery extends Activity {

      Gallery g;
      ImageView imagem;

       private int[] myImageIds = {
                 R.drawable.imagem1,
                 R.drawable.imagem2,
                 R.drawable.imagem3,
                 };
   public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);




          g = (Gallery) findViewById(R.id.gallery);
          g.setAdapter(new ImageAdapter(this));

          imagem = (ImageView) findViewById(R.id.imagem);

            g.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
```

```
                public void onItemClick(AdapterView<?> arg0, View
arg1,
                        int arg2, long arg3) {

                    imagem.setImageResource(myImageIds[arg2]);

                    Toast.makeText(getBaseContext(),
                         "Figura " + (arg2 + 1) + " selecionada",
                         Toast.LENGTH_SHORT).show();


                }


            });

    }
}
```

Quais foram as modificações desse programa? Nesse programa foi adicionado um componente chamado ImageView, que faz referência ao componente ImageView no arquivo XML, como mostra o código abaixo:

```
    ImageView imagem;
```

Também na aplicação adicionamos o array que contém todas as referências das imagens contidas no projeto, conforme mostra o código abaixo:

```
    private int[] myImageIds = {
                R.drawable.imagem1,
                R.drawable.imagem2,
                R.drawable.imagem3,
                };
```

Dentro do método onCreate foi feito uma referência ao componente ImageView contido no XML e definimos o evento OnItemClickListener , do componente Gallery. Vamos analisar seu código abaixo:

```
    imagem.setImageResource(myImageIds[arg2]);

     Toast.makeText(getBaseContext(),
"Figura " + (arg2 + 1) + " selecionada",
      Toast.LENGTH_SHORT).show();
```

A primeira instrução carrega a imagem selecionada no componente Gallery no ImageView, através do método setImageResource, cujo parâmetro é o valor do índice do vetor.

A segunda instrução fazendo uso do método makeText, da classe Toast, cuja finalidade é mostrar uma pequena mensagem na tela em um tempo curto.

No primeiro parâmetro desse método sempre passamos o valor getBaseContext(). No segundo parâmetro, passamos o conteúdo a ser exibido na tela. No terceiro parâmetro, definimos o tempo de exibição da mensagem na tela.

Execute a aplicação. O resultado você vê na figura abaixo:

(Aplicação da imagens otimizada)

5.16) A widget ProgressBar

Agora será mostrado uma widget do Android que consiste em uma ProgressBar (Barra de progresso). Ela é muito utilizada quando queremos indicar algum andamento em processo. Por exemplo, quando ocorre a instalação de um programa ou quando se faz um download de um arquivo normalmente é mostrado aquela barra de porcentagem que indica em porcentagem, o andamento daquele processo. Aquilo é uma barra de progresso.

5.17) Desenvolvendo uma aplicação que simula um download

Para demonstrar o uso do componente ProgressBar, criaremos uma aplicação que vai simular um download, onde o processo desse download será

feito por esse componente. Vamos criar um novo projeto em Android, com os seguintes dados:

Project Name: ExemploProgressBar

Package Name : br.com.android

Create Activity: AppProgressBar

Application Name: Exemplo com ProgressBar

Min SDK Version: 7

Agora no arquivo de layout modifique o conteúdo da TextView com a seguinte frase: “Status download”.

Em seguida, adicione os seguintes componentes na seqüência:

ProgressBar

Propriedade	Valor
Id	@+id/progresso
Layout_width	fill_parent
Style	?android:attr/progressBarStyleHorizontal
Max	100

Button

Propriedade	Valor
Id	@+id/progresso
Layout_width	fill_parent
Text	Efetuar download

Seguindo os passos acima, a aplicação deve estar de acordo com a aplicação da figura abaixo:

Vendo o layout acima, talvez você dever estar se perguntando: Espere aí, nos não definimos uma barra de progresso horizontal? Sim, definimos sim. Porém, em tempo de projeto, a imagem PADRÃO do componente ProgressBar é um anel. Em tempo de projeto, quando modificamos o estilo dela, ela não sofreu nenhuma mudança quanto ao desenho dela, isso é normal. Quando executarmos, você verá que o estilo da ProgressBar estará de acordo com o estilo que selecionamos.

Agora vá no arquivo “AppProgressBar.java” e coloque o seguinte código abaixo:

```
package br.com.android;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.*;

public class AppProgressBar extends Activity implements Runnable{

      ProgressBar p;
      Button b;
      Thread t;
      Handler h;
      int i;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```
        p = (ProgressBar) findViewById(R.id.progresso);
        b = (Button) findViewById(R.id.btdownload);


        b.setOnClickListener(new View.OnClickListener(){

                public void onClick(View v) {
                      h = new Handler();
                      t = new Thread(AppProgressBar.this);
                      t.start();

                }

        });


    }
      public void run() {
            i=1;
            try {
            while(i<=100) {
                  Thread.sleep(100);
                  h.post(new Runnable(){

                        public void run() {
                              p.setProgress(i++);


                        }

                  });
            }
            }catch(Exception e) {}
      }
}
```

Agora vou explicar o código desse programa. Observe a declaração da classe abaixo:

```
public class AppProgressBar extends Activity implements Runnable{
```

Além de derivar da classe Activity, eu implemento também a interface Runnable. Essa interface possui um método abstrato chamado “run”, que é bastante utilizado por “Threads”. Uma Thread consiste em um processo que é executado em “paralelo” com o programa que o invocou, ou seja, quando uma Thread é executada ,não é preciso que a próxima instrução (ou instruções) após a Thread espere ela ser finalizada, elas podem ser executadas em paralelo.

Veja as declarações abaixo:

```
      Thread t;
```

```
Handler h;
```

Aqui é declarado uma variável do tipo Thread, e outra do tipo Handler (que consiste também em uma Thread). A variável do tipo Handler é utilizada quando queremos manipular os widgets em tempo de execução dentro de uma Thread.

Vamos para o evento onClick do botão. Veja o código abaixo:

```
public void onClick(View v) {
                h = new Handler();
                t = new Thread(AppProgressBar.this);
                t.start();

            }
```

No evento onClick, eu crio uma instância da classe Handler e atribuo essa instância à variável *"h"*. Logo após, eu crio uma instância da classe Thread e atribuo essa instância à variável "t" e depois, eu disparo a Thread pelo método start, executando assim o método run. Vamos ao código do método **run**.

```
public void run() {
   i=1;
   try {
   while(i<=100) {
    Thread.sleep(100);
    h.post(new Runnable(){
    public void run() {
      p.setProgress(i++);

                    }

        });
        }}catch(Exception e) {}
   }
```

Vamos analisar o código dentro do "loop". A instrução:

```
Thread.sleep(100);
```

Gera um atraso de 100 milissegundos. Logo após vem a instrução:

```
h.post(new Runnable(){
  public void run() {
    p.setProgress(i++);

                    }

        });
```

A instrução acima, é responsável por incrementar o valor de progresso da widget *ProgressBar*. Quando usamos uma Thread e queremos modificar o valor de algum componente (widget), precisamos disparar um método post e dentro dela, criar uma instância da interface Runnable, com o método run onde nele, é executado a instrução responsável por modificar o valor de um componente.

Vamos executar a nossa aplicação . O resultado você vê na figura abaixo:

(Aplicação que simula um download)

5.18) A widget DatePicker

O componente **DatePicker** funciona como se fosse um calendário onde especificamos ou consultamos uma determinada data.

5.19) Desenvolvendo uma aplicação de calendário

Crie um novo projeto com os seguintes dados abaixo:

Project Name: Calendario

Package Name : br.com.android

Create Activity: AppCalendario

Application Name: Calendário

Min SDK Version : 7

Depois de carregar o layout do arquivo “main.xml”, modifique o conteúdo da TextView com a frase : Selecione a data.

Agora, insira os seguintes componentes na seqüência:

DatePicker

Propriedade	Valor
Id	@+id/data
Layout_width	fill_parent

Button

Propriedade	Valor
Id	@+id/btdata
Layout_width	fill_parent
Text	Mostrar data

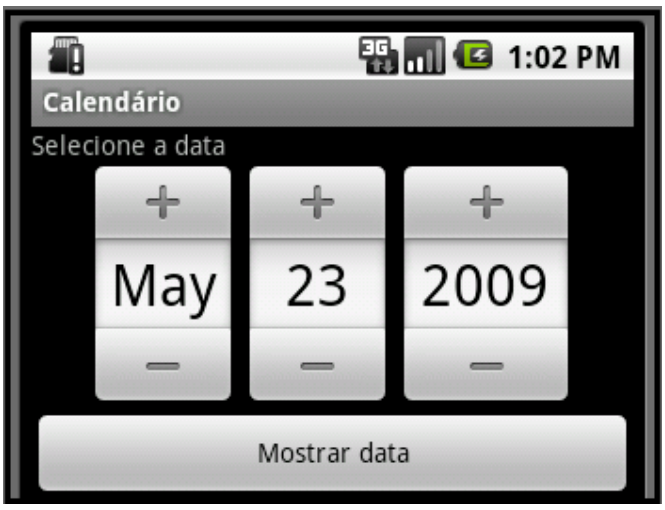
Seguindo os passos acima, a aplicação deve estar de acordo com a figura abaixo:

E por último, vá no arquivo “AppData.java” e coloque o seguinte código abaixo:

```
package br.com.android;


import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.*;


public class AppCalendario extends Activity {

      DatePicker dp;
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button b = (Button) findViewById(R.id.btdata);


        dp = (DatePicker) findViewById(R.id.data);

        dp.updateDate(2009, 04, 23);

        b.setOnClickListener(new View.OnClickListener(){

                public void onClick(View v) {

                        AlertDialog.Builder dialogo = new
AlertDialog.Builder(AppCalendario.this);
                        dialogo.setMessage("Data selecionada : "  +
String.valueOf(dp.getDayOfMonth()) +
                                        "/" + String.valueOf(dp.getMonth()
+ 1) + "/" + String.valueOf(dp.getYear()));
                        dialogo.setNeutralButton("OK", null);
                        dialogo.setTitle("Data");
                        dialogo.show();

                }

        });


            }
}
```

Vamos executar a nossa aplicação. O resultado você vê na figura abaixo:



(Aplicação de calendário)

5.20) A widget TimePicker

O componente **TimePicker** funciona como se fosse um **DatePicker** ,só que ao invés de trabalhar com datas, você trabalha com horas.

5.21) Desenvolvendo uma aplicação que faz uso do TimePicker

Crie um novo projeto com os seguintes dados abaixo:

Project Name: ExemploTimePicker

Package Name : br.com.android

Create Activity: AppTime

Application Name: Exemplo com TimePicker

Min SDK Version: 7

Agora carregue o arquivo de layout “main.xml” e modifique o conteúdo da TextView com a seguinte frase : “Selecione a hora:”. Agora, adicione os componentes na seqüência:

TimePicker

Propriedade	Valor
Id	@+id/hora
Layout_width	fill_parent

Button

Propriedade	Valor
Id	@+id/btmostrar
Layout_width	fill_parent
Text	Mostrar hora

Segundo os passos acima, a aplicação deve estar de acordo com o layout abaixo:

E por último, vá no arquivo “AppTime.java” e coloque o seguinte código abaixo:

```
package br.com.android;


import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.*;



public class AppTime extends Activity {

      TimePicker tp;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btmostrar = (Button) findViewById(R.id.btmostrar);


        tp = (TimePicker) findViewById(R.id.hora);


        btmostrar.setOnClickListener(new View.OnClickListener(){

                  public void onClick(View v) {
```

```
                        AlertDialog.Builder dialogo = new
AlertDialog.Builder(AppTime.this);
                        dialogo.setMessage("Hora selecionada : "  +
                                        String.valueOf(tp.getCurrentHour()
+ 1) + ":" + String.valueOf(tp.getCurrentMinute()));
                        dialogo.setNeutralButton("OK", null);
                        dialogo.setTitle("Hora");
                        dialogo.show();

                    }

        });


            }
}
```

Vamos executar a nossa aplicação . O resultado você vê na figura abaixo:

(Aplicação que demonstra o uso do TimePicker)

6) Mudando de layouts

Até agora fizemos aplicações em Android que utilizassem um único layout. Agora vamos fazer aplicações Android que utilizem mais de um layout.

Vamos criar um novo projeto de demonstração chamado TrocaDeLayouts, conforme os dados abaixo:

Project Name: TrocaDeLayouts

Package Name : br.com.android

Create Activity: AppLayout

Application Name: Mudando de layouts

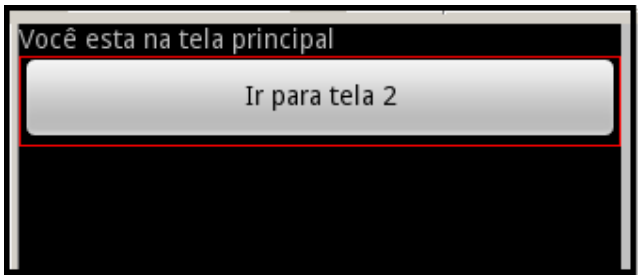
Min SDK Version: 7

Carregue o arquivo de layout “main.xml” e modifique o conteúdo da TextView, com a seguinte frase : “Você está na tela principal”. Agora adicione um Button e modifique as seguintes propriedades:

Button

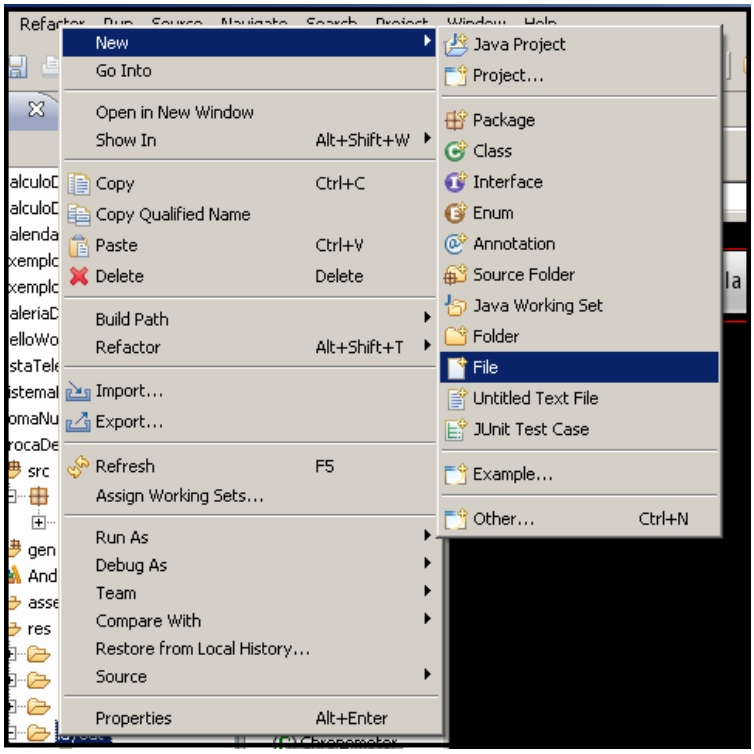
Propriedade	Valor
Id	@+id/bttela2
Layout width	fill_parent
Text	Ir pata tela 2

Seguindo os passos acima, a aplicação deve estar de acordo com a figura abaixo:

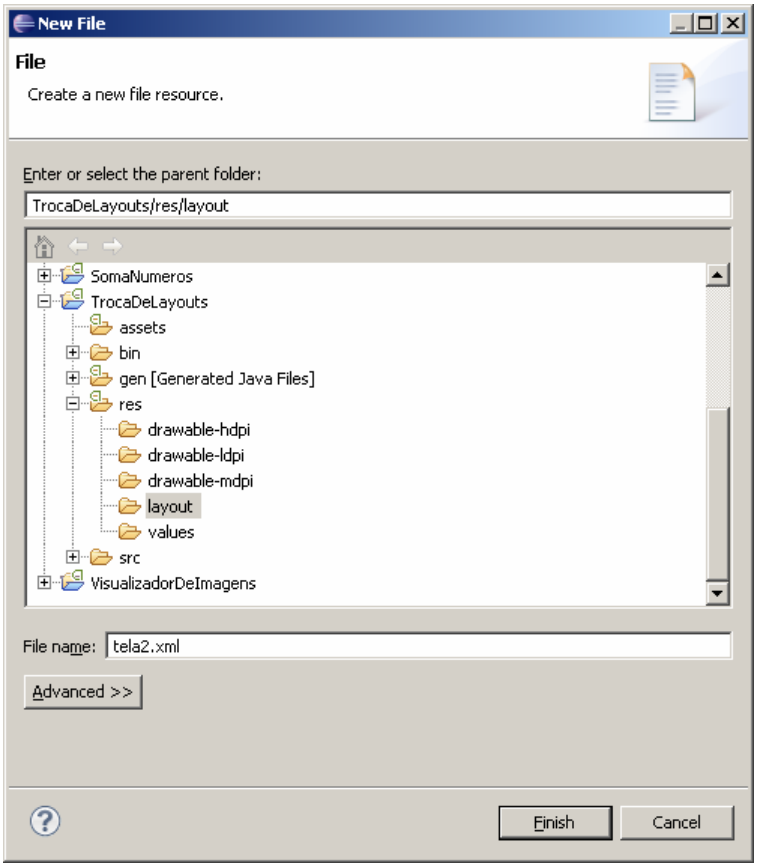


Ótimo! Agora vamos criar um novo arquivo chamado “tela2.xml”, que vai estar também dentro da pasta “layout”. Siga os passos abaixo:

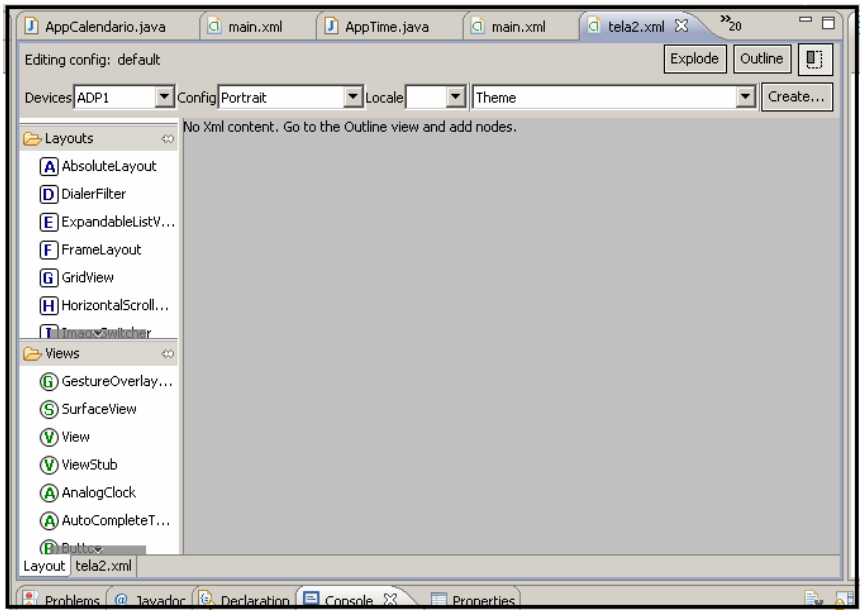
Clique com o botão direito sobre a pasta “layout” e selecione a opção “New/File”, conforme mostra a figura abaixo:



Agora no campo “File name” digite “tela2.xml”, conforme figura abaixo e logo depois, clique em “Finish”.



Ao carreguar o arquivo “tela2.xml”, será mostrado a seguinte tela abaixo:



Calma! Isso aconteceu pelo fato de o arquivo xml estar vazio. Agora, siga os passos abaixo:

Arraste para o meio da tela (região cinza da tela) a estrutura LinearLayout (se encontra na guia “Layouts”). E depois mude a seguinte propriedades dela:

Propriedade	Valor
Orientation	vertical

Logo em seguida, adicione os seguintes componentes, na seqüência:

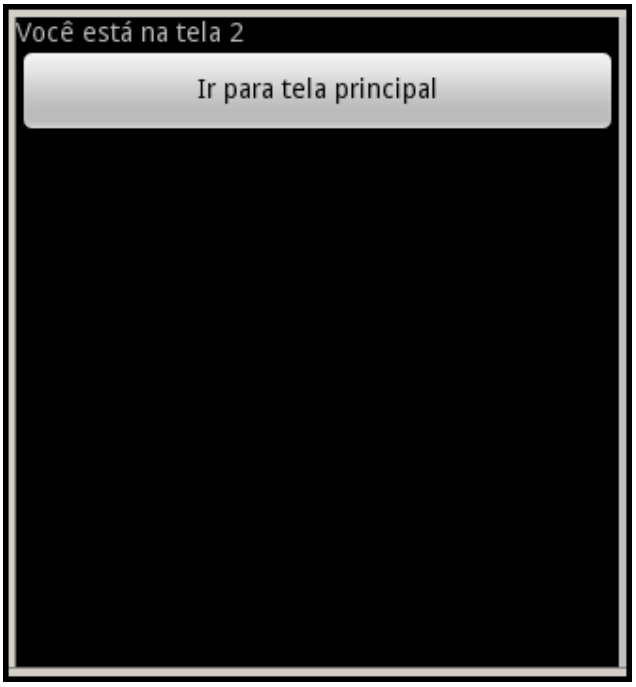
TextView

Propriedade	Valor
Text	Você está na tela 2

Button

Propriedade	Valor
Id	@+id/bttelaprincipal
Layout width	fill_parent
Text	Ir pata tela principal

Seguindo os passos acima, o layout do arquivo “tela2.xml” deve estar de acordo com a figura abaixo:

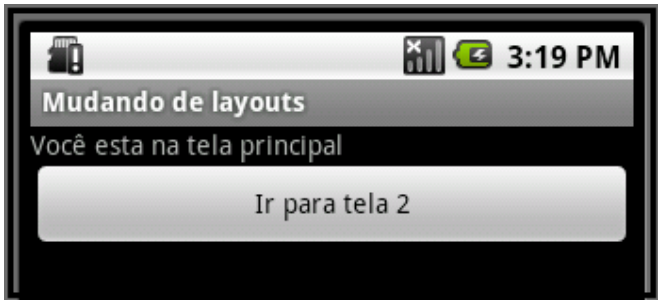


Depois disso, modifique o arquivo “AppLayout.java”, conforme o código abaixo:

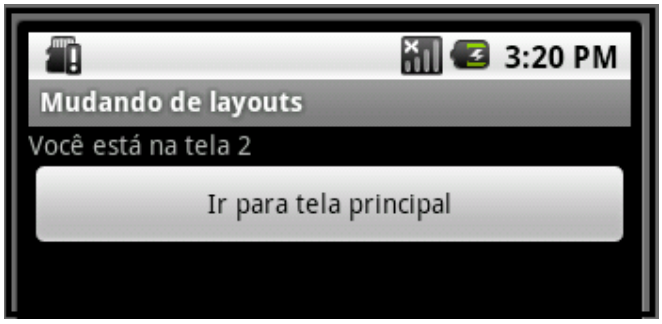
```
package br.com.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.*;
import android.widget.*;

public class AppLayout extends Activity {


      public void CarregaTelaPrincipal() {
      setContentView(R.layout.main);

      Button bttela2 = (Button) findViewById(R.id.bttela2);

      bttela2.setOnClickListener(new View.OnClickListener(){

                  public void onClick(View arg0) {

                        CarregaTela2();

                  }

      });

    }

    public void CarregaTela2() {
      setContentView(R.layout.tela2);
```

```
      Button bttelaprincipal = (Button)
findViewById(R.id.bttelaprincipal);

      bttelaprincipal.setOnClickListener(new View.OnClickListener(){

                  public void onClick(View arg0) {

                        CarregaTelaPrincipal();


                  }
                  });


    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        CarregaTelaPrincipal();

    }
}
```

Observem que nesta classe eu criei dois métodos : CarregaTelaPrincipal e CarregaTela2. Toda aplicação que utilize mais de um layout, o carregamento dos layouts e de seus respectivos widgets devem estar separados em funções desenvolvidas para esse propósito. Logo, o método CarregaTelaPrincipal carrega o layout principal e seus respectivos componentes, o mesmo válido para o método CarregaTela2, que carrega o layout da tela 2 e seus respectivos componentes. Feito isso, execute a aplicação. Veja o resultado abaixo:



(Aplicação no layout principal)



(Aplicação no segundo layout)

6.1) Desenvolvendo uma aplicação de cadastro

Agora vamos desenvolver uma aplicação de cadastro. Essa aplicação consiste em um cadastro de pessoas onde posso cadastrar dados como: Nome, Profissão e Idade. Essa aplicação vai fazer uso de três layouts:

O layout principal: Esse layout dará acesso ao *layout de cadastro* e o *layout de visualização de dados*.

O layout de cadastro: Nesse layout é onde será efetuado o cadastro dos dados da pessoa.

O layout de visualização de dados: Nesse layout é onde serão visualizados os dados cadastrados. Se nenhum dado foi cadastrado, será exibida uma mensagem informando essa situação.

Nessa aplicação, para armazenar os dados, eu faço um de uma estrutura de dados FIFO ou Fila. Nessa estrutura, os dados são armazenados em seqüência, e acessados em seqüência , ou seja, o primeiro dado a entrar será o primeiro a ser exibido.

Bom, vamos construir a nossa aplicação. Crie um novo projeto com os dados abaixo:

Project Name: AplicacaoDeCadastro

Package Name : br.com.android

Create Activity: AppCadastro

Application Name: Aplicação de Cadastro

Min SDK Version: 7

Dentro da pasta “res/drawable-mdpi”, coloque uma figura , que se encontra dentro da pasta “ICONE” que acompanha este material. Ela se chama “profile.png” e representa o ícone da nossa aplicação.

Agora carregue o arquivo “main.xml” e APAGUE a TextView que se encontra na tela do layout do dispositivo. O procedimento já foi mostrado.

Agora coloque os seguintes componentes, na seqüência:

ImageView

Propriedade	Valor
Src	@drawable/profile

TextView

Propriedade	Valor
Text	Bem vindo a Aplicação de Cadastro de Pessoas.Este é um pequeno programa de demonstração de cadastro.Selecione uma das opções abaixo:

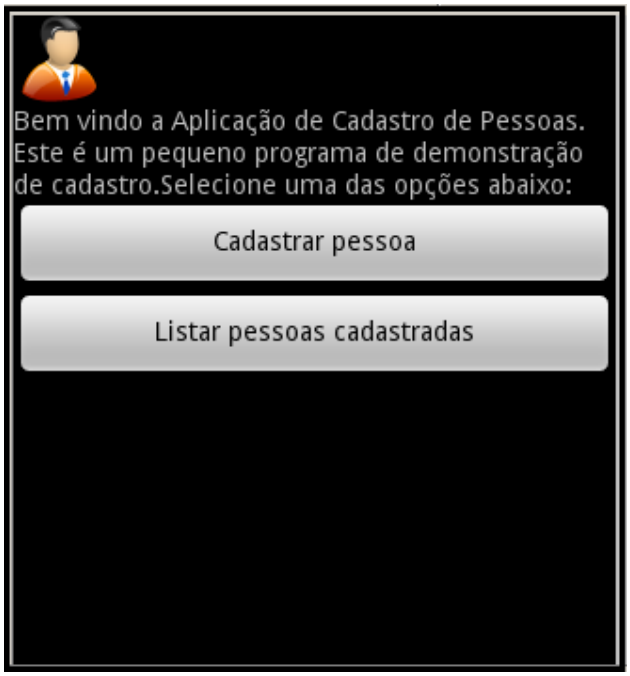
Button

Propriedade	Valor
Id	@+principal/btcadastrarpessoas
Layout_width	fill_parent
Text	Cadastrar pessoa

Button

Propriedade	Valor
Id	@+principal/btlistarpessoas
Layout_width	fill_parent
Text	Listar pessoas cadastradas

Seguindo os passos acima, a aplicação deve estar de acordo com a figura abaixo:



Agora vamos criar um arquivo chamado cadastro.xml e vamos coloca-lo dentro da pasta "res/layout" do nosso projeto.

No início, o arquivo está vazio. Agora você vai adicionar os seguintes componentes na seqüência:

LinearLayout

Propriedade	Valor
Orientation	vertical

ImageView

Propriedade	Valor
Src	@drawable/profile

TextView

Propriedade	Valor
Text	Módulo de cadastro.digite seus dados abaixo:

TextView

Propriedade	Valor
Text	Nome:

EditText

Propriedade	Valor
Id	@+cadastro/ednome
Layout_width	fill_parent
Text	

TextView

Propriedade	Valor
Text	Profissão:

EditText

Propriedade	Valor
Id	@+cadastro/edprofissao
Layout_width	fill_parent
Text	

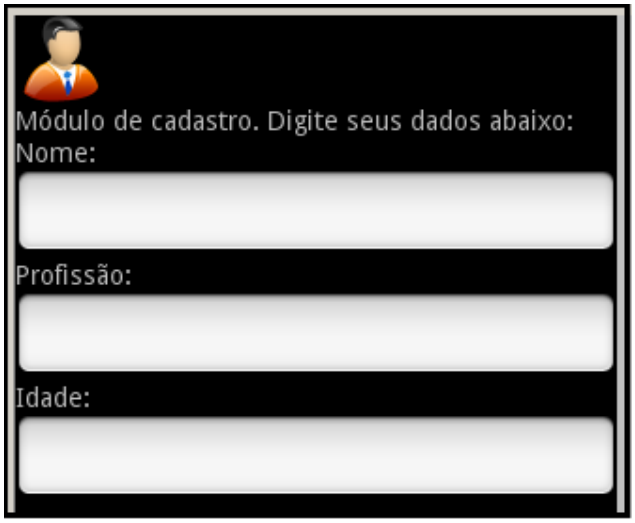
TextView

Propriedade	Valor
Text	Idade:

EditText

Propriedade	Valor
Id	@+cadastro/edidade
Layout_width	fill_parent
Text	

Seguindo os passos acima, a aplicação deve estar de acordo com a figura abaixo:



Bom, ainda não acabou. Agora vamos na guia “Outline” e vamos adicionar uma estrutura LinearLayout, que será responsável por organizar os botões de forma horizontal. Clique com o botão direito do mouse sobre a estrutura chamada LinearLayout01 (do tipo LinearLayout, óbvio) e selecione “Add”. Será aberta uma caixa de diálogo de componentes e estruturas e você vai adicionar uma LinearLayout para adicionar nessa estrutura.

A estrutura adicionada será indicada por LinearLayout02. Agora vamos modificar a seguintes propriedades dela abaixo:

Propriedade	Valor
Orientation	horizontal
Layout	fill_parent
Gravity	center

A propriedade “Gravity” , similar a propriedade “Orientation”, determina o alinhamento dos componentes dentro da estrutura, que no caso acima está alinhando os componentes de forma centralizada. Ou seja, os componentes vão estar dispostos de forma horizontal (um ao lado do outro) e todos esses componentes estarão alinhados de forma centralizada.

Seguindo o mesmo procedimento acima, vamos adicionar dois Buttons dentro dessa estrutura, e mude as seguintes propriedades citadas abaixo.

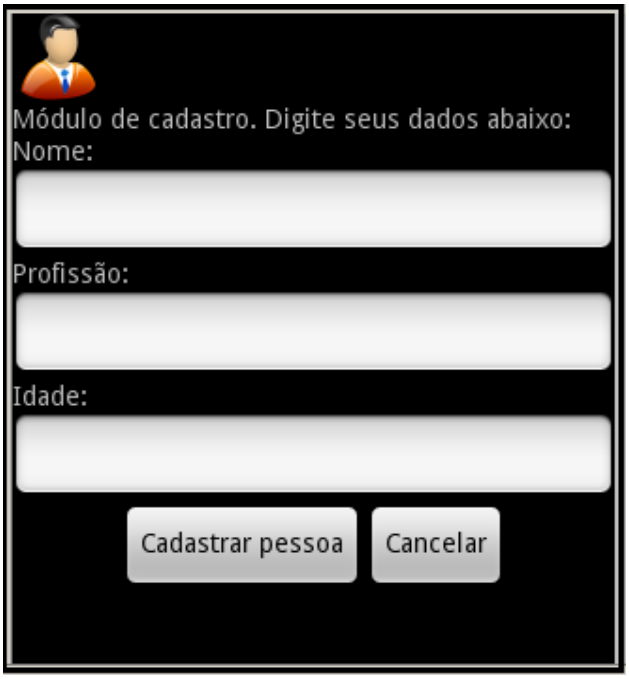
Button

Propriedade	Valor
Id	@+cadastro/btcadastrar
Text	Cadastrar pessoa

Button

Propriedade	Valor
Id	@+cadastro/btcancelar
Text	Cancelar

Seguindo os passos acima, o Layout de nossa aplicação deve estar de acordo com a figura abaixo:



Agora dentro da pasta “res/layout” vamos criar um arquivo chamado “listacadastrados.xml” que será responsável por mostrar todos usuários devidamente cadastrados. Depois de criar o arquivo, vamos adicionar os componentes abaixo, na seqüência:

LinearLayout

Propriedade	Valor
Orientation	vertical
Id	@+id/layoutPrincipal

ImageView

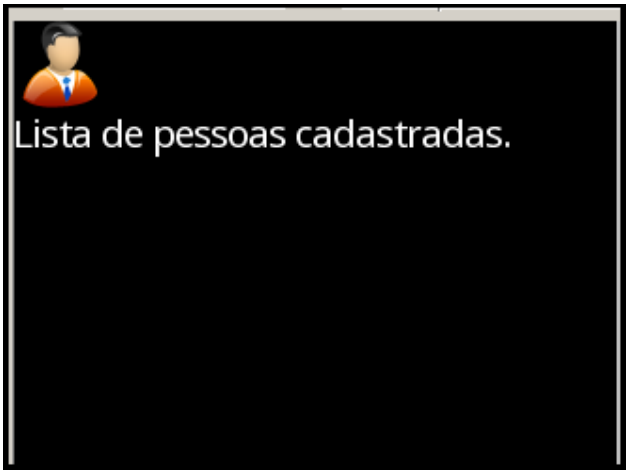
Propriedade	Valor
Src	@drawable/profile

TextView

Propriedade	Valor
Text	Lista de pessoas cadastradas.
Text color	#ffffff
Text size	20sp

Se você observar acima, estou trabalhando com duas propriedades pertencentes também ao componente TextView, uma chamada “Text color”, responsável por definir a cor do texto que neste caso, o valor da cor precisa estar no formato hexadecimal precedido do símbolo # ,e “Text size” responsável por definir o tamanho do texto.

Seguindo os passos acima, o layout da aplicação deve estar de acordo com a figura abaixo:



Bom, ainda não acabou. Se você observou, damos um nome a nossa estrutura LinearLayout, nós a chamados de “layoutPrincipal”. Bom, seguindo os procedimentos que foram explicados acima, vamos adicionar dentro da estrutura “layoutPrincipal” uma nova estrutura LinearLayout dentro dela. Depois de adicionar, vamos mudar as seguintes propriedades dela abaixo:

LinearLayout

Propriedade	Valor
Orientation	Horizontal
Id	@+id/layoutNome

Bom, agora dentro da estrutura “layoutNome” vamos adicionar os seguintes componentes na seqüência:

TextView

Propriedade	Valor
Text	Nome.
Text color	#ffff00
Text size	20sp

TextView

Propriedade	Valor
Text	
Text color	#ffffff
Text size	20sp
Id	@+lista/txtnome

Bom agora dentro da estrutura “layoutPrincipal”, vamos adicionar uma nova estrutura LinearLayout, como já foi mostrado acima. E depois, modifique as seguintes propriedades abaixo:

Propriedade	Valor
Orientation	Horizontal
Id	@+id/layoutProfissao

Bom, agora dentro da estrutura “layoutProfissao” vamos adicionar os seguintes componentes na seqüência:

TextView

Propriedade	Valor
Text	Profissão.
Text color	#ffff00
Text size	20sp

TextView

Propriedade	Valor
Text	
Text color	#ffffff
Text size	20sp
Id	@+lista/txtprofissao

Bom agora dentro da estrutura “layoutPrincipal”, vamos adicionar uma nova estrutura LinearLayout, como já foi mostrado acima. E depois, modifique as seguintes propriedades abaixo:

Propriedade	Valor
Orientation	Horizontal
Id	@+id/layoutIdade

Bom, agora dentro da estrutura “layoutProfissao” vamos adicionar os seguintes componentes na seqüência:

TextView

Propriedade	Valor
Text	Idade.
Text color	#ffff00
Text size	20sp

TextView

Propriedade	Valor
Text	
Text color	#ffffff
Text size	20sp
Id	@+lista/txtidade

Bom agora dentro da estrutura “layoutPrincipal”, vamos adicionar uma nova estrutura LinearLayout, como já foi mostrado acima. E depois, modifique as seguintes propriedades abaixo:

Propriedade	Valor
Orientation	Horizontal
Id	@+id/layoutBotoes
Layout width	fill_parent
Gravity	Center

Bom, agora dentro da estrutura “layoutProfissao” vamos adicionar os seguintes componentes na seqüência:

Button

Propriedade	Valor
Id	@+lista/btvoltar
Text	Voltar

Button

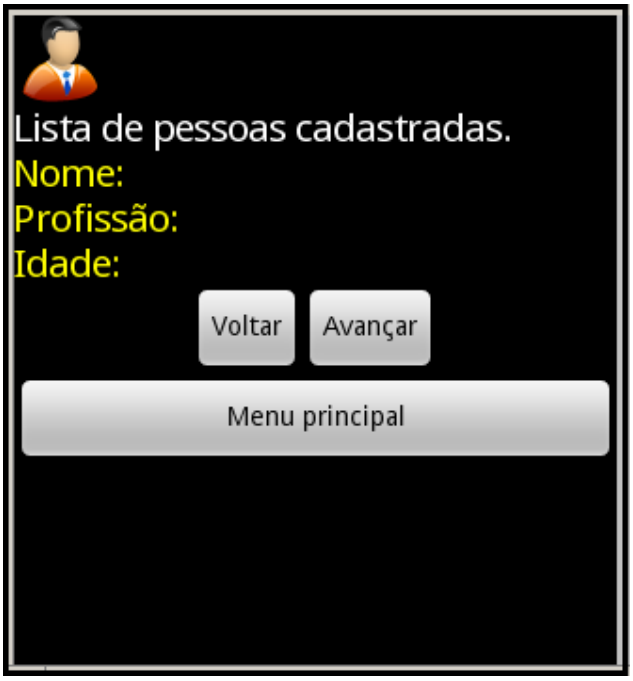
Propriedade	Valor
Id	@+lista/btavancar
Text	Avançar

Agora vamos adicionar na estrutura “layoutPrincipal’ um Button com as seguintes propriedades abaixo:

Button

Propriedade	Valor
Id	@+lista/btmenu
Text	Menu principal
Layout width	fill_parent

Seguindo todos os passos acima, o layout da aplicação deve estar de acordo com a figura abaixo:



Agora no arquivo “AppCadastro.java”, coloque o seguinte código:

```
package br.com.android;


import android.app.Activity;
import android.os.Bundle;
import android.widget.*;
import android.view.*;
import android.app.*;




public class AppCadastro extends Activity {

      Registro pri,reg,ult,aux;

    EditText ednome,edprof,edidade;
```

```
    int numreg,pos;


    void CarregaTelaPrincipal () {

      setContentView(R.layout.main);


         Button btcadpess = (Button)
findViewById(R.principal.btcadastrarpessoa);
         Button btlistapess = (Button)
findViewById(R.principal.btlistarpessoas);


        btcadpess.setOnClickListener(new View.OnClickListener(){
            public void onClick(View arg0){

                  CarregaTelaCadastro();

            }});


        btlistapess.setOnClickListener(new View.OnClickListener(){
            public void onClick(View arg0){

                  CarregaListaPessoas();

            }});



   }


    void CarregaTelaCadastro() {
      setContentView(R.layout.cadastro);


      Button btcadastrar = (Button)
findViewById(R.cadastro.btcadastrar);
      Button btcancelar = (Button)
findViewById(R.cadastro.btcancelar);

         btcadastrar.setOnClickListener(new View.OnClickListener(){
        public void onClick(View arg0){
            try {

            reg = new Registro();

            ednome = (EditText)findViewById(R.cadastro.ednome);
            edprof = (EditText)findViewById(R.cadastro.edprofissao);
            edidade = (EditText)findViewById(R.cadastro.edidade);


            reg.nome = ednome.getText().toString();
            reg.profissao = edprof.getText().toString();
            reg.idade = edidade.getText().toString();
```

```
            if(pri==null)
                  pri=reg;

            reg.Ant = ult;
            if(ult==null)
                  ult=reg;
            else {
                  ult.Prox = reg;
                  ult=reg;
            }

            numreg++;

            showMessage("Cadastro efetuado com sucesso", "Aviso");
            CarregaTelaPrincipal();

         }
         catch(Exception e) {

            showMessage("Erro ao cadastrar", "Erro");

         }}
    });

         btcancelar.setOnClickListener(new View.OnClickListener(){
              public void onClick(View arg0){
                  CarregaTelaPrincipal();
              }
              });
    }


    void CarregaListaPessoas() {

      if(numreg==0) {

            showMessage("Nenhum registro cadastrado", "Aviso");

            CarregaTelaPrincipal();
            return;


      }
      setContentView(R.layout.listacadastrados);
      pos=1;
      aux=pri;
      TextView txtnome = (TextView)findViewById(R.lista.txtnome);
      TextView txtidade = (TextView)findViewById(R.lista.txtidade);
      TextView txtprofissao =
(TextView)findViewById(R.lista.txtprofissao);

      Button btmenu = (Button) findViewById(R.lista.btmenu);
      Button btavancar = (Button) findViewById(R.lista.btavancar);
      Button btvoltar = (Button) findViewById(R.lista.btvoltar);

      txtnome.setText(aux.nome);
      txtidade.setText(aux.idade);
      txtprofissao.setText(aux.profissao);

      btmenu.setOnClickListener(new View.OnClickListener(){
```

```
                public void onClick(View arg0){
                    CarregaTelaPrincipal();
                }
                });


        btvoltar.setOnClickListener(new View.OnClickListener(){
                public void onClick(View arg0){
                    if(pos==1)
                          return;

                    pos--;
                    aux=aux.Ant;
                    TextView txtnome =
(TextView)findViewById(R.lista.txtnome);
                        TextView txtidade =
(TextView)findViewById(R.lista.txtidade);
                    TextView txtprofissao =
(TextView)findViewById(R.lista.txtprofissao);
                    txtnome.setText(aux.nome);
                    txtidade.setText(aux.idade);
                    txtprofissao.setText(aux.profissao);

                }
                });

        btavancar.setOnClickListener(new View.OnClickListener(){
                public void onClick(View arg0){
                    if(pos==numreg)
                          return;

                    pos++;
                    aux=aux.Prox;
                    TextView txtnome =
(TextView)findViewById(R.lista.txtnome);
                        TextView txtidade =
(TextView)findViewById(R.lista.txtidade);
                    TextView txtprofissao =
(TextView)findViewById(R.lista.txtprofissao);
                    txtnome.setText(aux.nome);
                    txtidade.setText(aux.idade);
                    txtprofissao.setText(aux.profissao);

                }
                });

    }
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        numreg=0;
        pri=ult=null;
        CarregaTelaPrincipal();



}

public void showMessage(String Caption,String Title) {
```

```
    AlertDialog.Builder dialogo = new
AlertDialog.Builder(AppCadastro.this);
    dialogo.setTitle(Title);
      dialogo.setMessage(Caption);
      dialogo.setNeutralButton("OK", null);
      dialogo.show();

    }

}
```

Agora no mesmo local onde se encontra o arquivo “AppCadastro.java” (no pacote “br.com.android”), você vai criar uma classe chamada Registro. Depois de criar a classe, coloque o código que é exibido abaixo:

```
package br.com.android;

public class Registro {
    String nome;
    String profissao;
    String idade;
    Registro Prox;
    Registro Ant;
}
```

Agora vamos analisar aos poucos os códigos dessa aplicação. Observe que nessa aplicação ou possuo três métodos: um método chamado CarregaTelaPrincipal , responsável por carregar o layout da tela principal. O método CarregaTelaCadastro é responsável por carregar a tela de cadastro.

Vamos analisar alguns códigos do método CarregaTelaCadastro. Se você observar nessa aplicação, que eu declarei quatro variáveis chamadas *pri,ult* e *reg* e *aux* do tipo Registro. A variável *pri* serve para apontar para o endereço do primeiro registro. A variável *ult* aponta para o endereço do último registro. A variável *reg* armazena os dados do registro corrente e a variável *aux* funciona como uma variável auxiliar.

É com a utilização essas variáveis que faço o cadastro dos dados das pessoas. Vamos para o evento *click* do botão btcadastrar situado dentro do método CarregaTelaCadastro e analisar algumas linhas de código. A linha:

```
reg = new Registro();
```

Cria uma nova instância da classe da classe Registro e coloca em *."reg"*. As linhas:

```
reg.nome = ednome.getText().toString();
 reg.profissao = edprof.getText().toString();
```

```
reg.idade = edidade.getText().toString();
```

Gravam os dados dos campos no objeto *"reg"*. Já as linhas abaixo:

```
if(pri==null)
   pri=reg;
   reg.Ant = ult;
   if(ult==null)
     ult=reg;
   else {
     ult.Prox = reg;
     ult=reg;
       }
```

Fazem todo o processo de armazenamento dos dados.

Agora vamos para o método CarregaListaPessoas. Quando esse método é chamado, é feita uma verificação se há dados cadastrados. Se não houver dados cadastrados, será exibida uma mensagem indicando essa situação e você será retornado a tela principal. Vou comentar algumas linhas. A linha:

```
aux=pri;
```

Retorna para a variável *"aux"* o endereço do primeiro registro, que está armazenado em *"pri"*. Já as linhas:

```
txtnome.setText(aux.nome);
txtidade.setText(aux.idade);
txtprofissao.setText(aux.profissao);
```

Joga as informações obtidas (nome, idade e profissão) para os campos (TextViews), para que eles possam ser exibidas.

Vamos agora para o evento *click* do botão *btvoltar*. Esse botão mostra os registros anteriores. Antes de voltar um registro, verifico se eu me encontro no primeiro registro pela condição:

```
if(pos==1)
```

Se a condição for verdadeira, saio do evento, senão, continuo executando as instruções. A linha:

```
aux=aux.Ant;
```

Retorna para *aux* o endereço do registro anterior. Depois disso são executados instruções para que os dados possam ser exibidos.

Já no evento *click* do botão *btavancar*, antes de passar para o próximo registro, verifico se já está no último registro pela instrução:

```
if(pos==numreg)
```

Se a condição for verdadeira, saio do evento, senão, continuo executando as instruções. A linha:

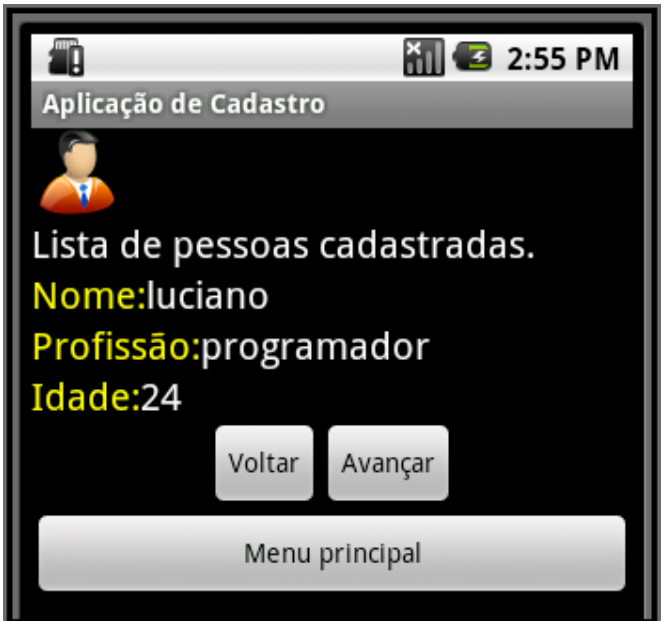
```
aux=aux.Prox;
```

Retorna para *aux* o endereço do próximo registro. Depois disso são executados instruções para que os dados possam ser exibidos.

Bom, vamos executar a nossa aplicação ? O resultado você confere nas figuras abaixo:

(Tela principal da aplicação)

(Tela de cadastro)



(Tela de visualização dos cadastrados)

7) Trabalhando com menus em uma aplicação

É possível em uma aplicação Android adicionar menus em uma aplicação. Os menus são visualizados quando pressionamos o botão "Menu" do emulador. Vamos a um exemplo, crie um novo projeto com os seguintes dados abaixo:

Project Name: ExemploMenus

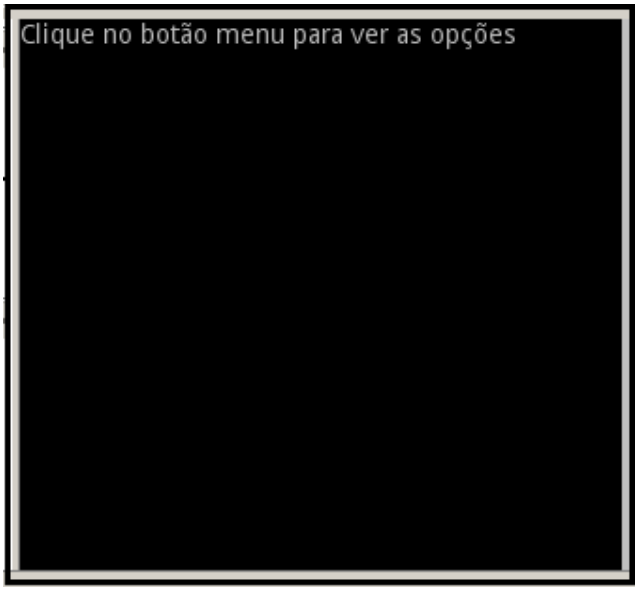
Package Name : br.com.android

Activity Name: AppMenu

Application Name: Exemplo com menus

Carregue o arquivo "main.xml" e modifique o conteúdo da TextView com a seguinte frase: "Clique no botão menu para ver as opções".

Seguindo o passo acima a aplicação deve estar de acordo com a figura abaixo:



No arquivo “AppMenu.java”, coloque o seguinte código abaixo:

```
package br.com.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.*;


public class AppMenu extends Activity {

       private static final int GRAVAR = Menu.FIRST;

       private static final int EDITAR = Menu.FIRST+1;

       private static final int SAIR = Menu.FIRST+2;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
     public boolean onCreateOptionsMenu(Menu menu) {


        menu.add(0, GRAVAR, 0, "Gravar");
        menu.add(0, EDITAR, 0, "Editar");
        menu.add(0, SAIR, 0, "Sair");


        return super.onCreateOptionsMenu(menu);
    }
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case GRAVAR:
           //Executa algo
            return true;
        case EDITAR:
           //Executa algo
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
 }
```

.

Vamos à explicação do código. Nas linhas :

```
private static final int GRAVAR = Menu.FIRST;

private static final int EDITAR = Menu.FIRST+1;

private static final int SAIR = Menu.FIRST+2;
```

São criadas três constantes. A constante *"GRAVAR"*, recebe o valor contido no campo *"FIRST"*, do objeto Menu , que normalmente é o valor "1". Logo, as constantes *"EDITAR"* e *"SAIR"* recebem, respectivamente os valores 2 e 3.

Agora vamos analisar o método onCreateOptionsMenu. Esse método é responsável por criar os menus e adiciona-los à aplicação. Vamos analisar as instruções abaixo:

```
menu.add(0, GRAVAR, 0, "Gravar");
menu.add(0, EDITAR, 0, "Editar");
menu.add(0, SAIR, 0, "Sair");
```

Elas são responsáveis por adicionar os menus "Gravar","Editar" e "Sair".

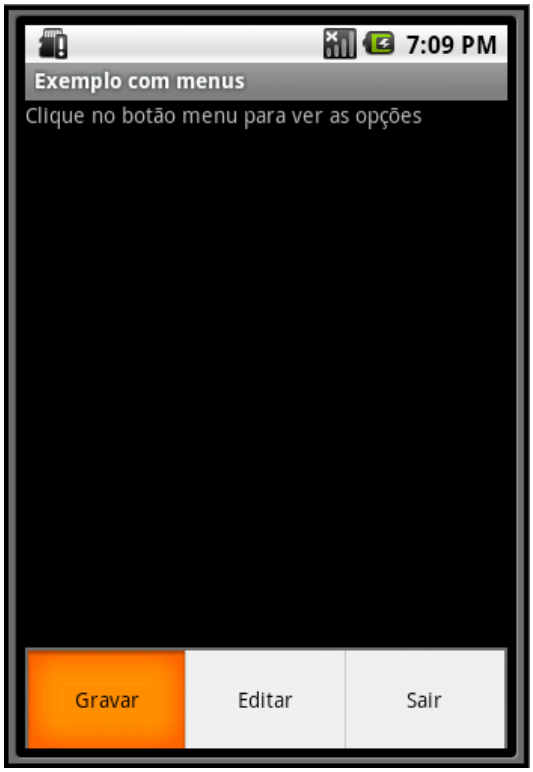
Toda vez que um item de menu é selecionado, o método onOptionsItemSelected é disparado. Vamos ver o código dele abaixo:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case GRAVAR:
           //Executa algo
            return true;
        case EDITAR:
           //Executa algo
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
```

```
}
```

Vamos analisar o código desse método. Observe que na estrutura switch é avaliado o valor retornado pelo método getItemId(), do objeto "item". Esse método retorna o "Id" do menu selecionado, que é representado pelas constantes, como pode ser observado na estrutura switch.

Vamos executar a nossa aplicação. O resultado você vê na figura abaixo:



(Demonstração do uso de menus)

Também é possível adicionar sub-menus a um determinado menu. Vamos a um outro exemplo, agora utilizando os sub-menus. Vamos modificar todo o código do arquivo "AppMenu.java" , pelo novo código abaixo:

```
package br.com.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SubMenu;

public class AppMenu extends Activity {

	private static final int ABRIR = Menu.FIRST;

	private static final int SALVAR = Menu.FIRST+1;

	private static final int FERRAMENTAS = Menu.FIRST+2;
```

```
   private static final int PERSONALIZAR = Menu.FIRST+3;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}

@Override public boolean onCreateOptionsMenu(Menu menu) {


    menu.add(0, ABRIR, 0, "Abrir");
    menu.add(0, SALVAR, 0, "Salvar");


    SubMenu outros = menu.addSubMenu("Outros");

    outros.add(0, FERRAMENTAS, 0, "Ferramentas");
    outros.add(0, PERSONALIZAR, 0, "Personalizar");


    return super.onCreateOptionsMenu(menu);
}


@Override public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case ABRIR:
           //Executa algo
            return true;
        case SALVAR:
           //Executa algo
            return true;
        case FERRAMENTAS:
            //Executa algo
             return true;
         case PERSONALIZAR:
           //Executa algo
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
 }
```

Vamos analisar as mudanças. Observe que nesse novo código eu criei quatro constantes, conforme mostra as instruções abaixo:

```
private static final int ABRIR = Menu.FIRST;

private static final int SALVAR = Menu.FIRST+1;

private static final int FERRAMENTAS = Menu.FIRST+2;

private static final int PERSONALIZAR = Menu.FIRST+3;
```

Agora vamos ver as mudanças feitas no método onCreateOptionsMenu. Veja as instruções abaixo:

```
menu.add(0, ABRIR, 0, "Abrir");
menu.add(0, SALVAR, 0, "Salvar");


SubMenu outros = menu.addSubMenu("Outros");

outros.add(0, FERRAMENTAS, 0, "Ferramentas");
outros.add(0, PERSONALIZAR, 0, "Personalizar");
```

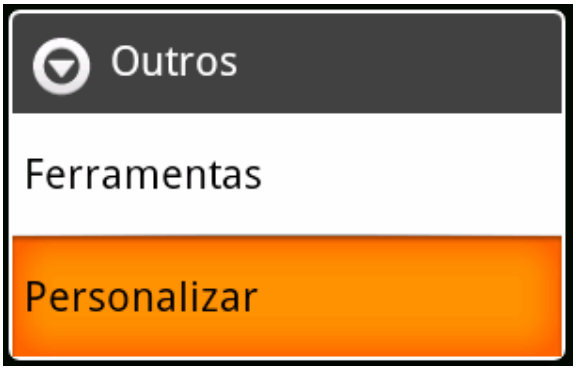
Observando o código acima, são criados dois menus, um menu chamado “Abrir” e um outro chamado “Salvar”.

Depois de criar os dois menus, eu crio logo em seguida um sub-menu , cujo rótulo é “Outros” e dentro desse sub-menu, adiciono dois itens, um chamado “Ferramentas” e “Personalizar”.

Vamos executar a nossa aplicação. O resultado você vê na figura abaixo:

(Aplicação usando menus)

Selecione a opção “Outros”, como mostra a figura acima, e será mostrado mais dois itens : “Ferramentas” e ”Personalizar”. Veja a figura abaixo:



(Usando submenus)

8) Entendendo melhor a classe AlertDialog

Nos já usamos essa classe nos alguns dos nossos programas anteriores. Mas agora, vamos entender melhor essa classe enfatizando aqui alguns métodos interessantes que não foram usados nos programas anteriores.

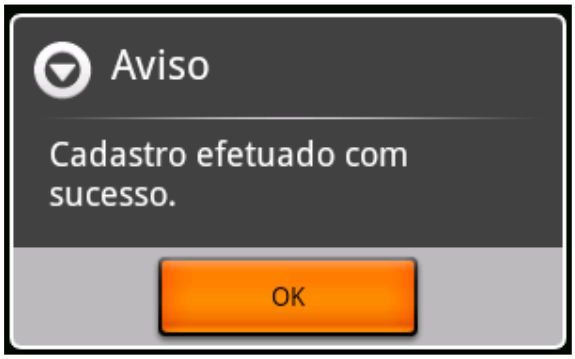
Em Java	Descrição
setIcon(int IconId)	Neste método, você define um ícone para a sua caixa de diálogo.
setMessage(CharSequence mensagem)	Neste método, você define a mensagem que será exibida na caixa de diálogo.
setTitle(CharSequence titulo)	Neste método, você define o título que será exibido na caixa de diálogo.
setNeutralButton(CharSequence <rotulo do botão>, OnClickListener evento)	Neste método você define um botão neutro (normalmente rotulado com o título “OK”) e um evento, caso ele seja clicado.
setPositiveButton(CharSequence <rotulo do botão>, DialogInterface.OnClickListener evento)	Neste método você define um botão positivo (normalmente rotulado com o título “Sim” ou “Yes”) e um evento, caso ele seja clicado.
setNegativeButton(CharSequence <rotulo do botão>, DialogInterface.OnClickListener evento)	Neste método você define um botão negativo(normalmente rotulado com o título “Não” ou “No”) e um evento, caso ele seja clicado.

Veja alguns exemplos abaixo:

```
AlertDialog.Builder dialog = new
AlertDialog.Builder(this);
dialog.setMessage("Cadastro efetuado com sucesso.");
dialog.setNeutralButton("OK", null);
dialog.setTitle("Aviso");
```

```
dialog.show();
```

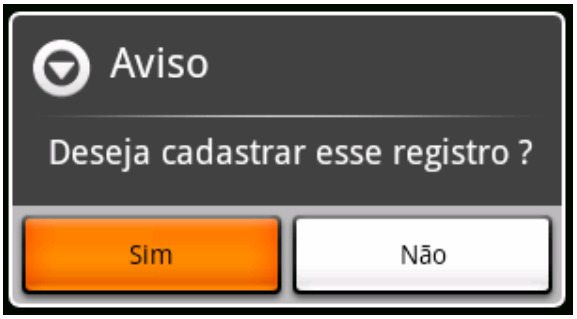
O resultado será :



Outro Exemplo:

```
AlertDialog.Builder dialog = new AlertDialog.Builder(this);
dialog.setMessage("Deseja cadastrar esse registro ?");

dialog.setPositiveButton("Sim", new
DialogInterface.OnClickListener(){

            public void onClick(DialogInterface di, int arg) {

                  //Executa algo se o botão "Sim" por pressionado

            }

      });


dialog.setNegativeButton("Não", new
DialogInterface.OnClickListener(){

            public void onClick(DialogInterface di, int arg) {

                  //Executa algo se o botão "Não" por pressionado

            }

      });


  dialog.setTitle("Aviso");
  dialog.show();
```

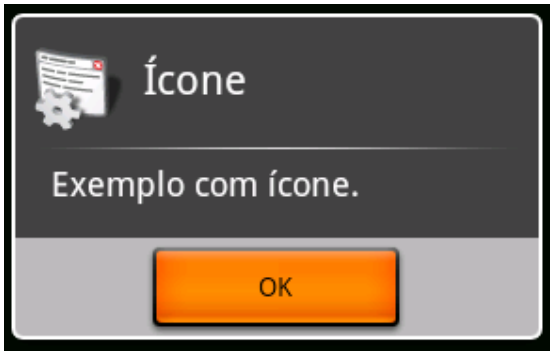
O resultado será :



Mais um exemplo :

```
AlertDialog.Builder dialog = new AlertDialog.Builder(this);
dialog.setMessage("Exemplo com ícone.");
dialog.setNeutralButton("OK", null);
dialog.setTitle("Ícone");
dialog.setIcon(R.drawable.icon);
dialog.show();
```

O resultado sera :



Com os conhecimentos até aqui obtidos , já da para fazer boas aplicações em Android com mais detalhes que enriquecem a aplicação.

9) Propriedades e eventos dos componentes trabalhados

Nesta seção eu irei mostrar e descrever as propriedades e eventos de todos os componentes que trabalhamos neste material.

Widget TextView

- Propriedades

Propriedade	Em XML	Em Java	Descrição
Text	android:text	setText(CharSequence c)	Nessa propriedade, você define o texto a ser exibido na tela.
Text color	android:textColor	setTextColor(Color c)	Nessa propriedade, você define cor do texto a ser exibido na tela. Valor: #000000 até #FFFFFF.
Background	android:background	setBackGroundColor(Color c)	Nessa propriedade , você define o cor de fundo do componente exibido. Valor: #000000 até #FFFFFF.
Text size	android:textSize	setTextSize(float tamanho) ou setTextSize(int unidade, int tamanho)	Define o tamanho do texto. O tamanho da fonte pode ser especificado em várias notações : px (pixels),sp(scaled-pixels) , mm(milímetros), in (polegadas) e etc.
Typeface	android:typeface	setTypeface(Typeface fonte)	Essa propriedade serve para definir uma fonte ao texto (normal,sans,serif,monospace).

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
setOnClickListener	OnClickListener	onClick(View v)	Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.

Widget EditText

- Propriedades

Propriedade	Em XML	Em Java	Descrição
Text	android:text	setText(CharSequence c)	Nessa propriedade, você define o texto a ser exibido na tela.
Text color	android:textColor	setTextColor(Color c)	Nessa propriedade, você define cor do texto a ser exibido na tela.

Background	android:background	setBackGroundColor(Color c)	Nessa propriedade , você define o cor de fundo do componente exibido.
Capitalize	android:capitalize		Essa propriedade serve para definir o tipo capitalização das palavras. Por padrão, o valor e “none”(nenhum). Os possíveis valores para essa propriedade são : “words”,”sentences” e ”characters”
Numeric	android:numeric		Com essa propriedade habilitada, o EditText só irá aceitar números (inteiros e decimais). O valor padrão desse atributo é “false”. Os possíveis valores para essa propriedade são : “integer” (número inteiro), “decimal” (número decimal) e “signed” (número com sinal). Esses valores podem ser combinados, por exemplo : android:numeric=”integer|signed”
Password	android:password		Com essa propriedade você habilita a digitação de senhas. O valor padrão desse atributo é “false”.
Text size	android:textSize	setTextSize(float tamanho) ou setTextSize(int unidade, int tamanho)	Define o tamanho do texto. O tamanho da fonte pode ser especificado em várias notações : px (pixels),sp(scaled-pixels) , mm(milímetros), in (inches) e etc.
Typeface	android:typeface	setTypeface(Typeface fonte)	Essa propriedade serve para definir uma fonte ao texto. Os possíveis valores são : “normal”,”monospace”,”sans” e “serif”.
Hint	android:hint	setHint(CharSequence c)	Nessa propriedade você define uma mensagem que aparecerá quando a EditText estiver vazia.
Text color hint	android:textColorHint	setHintTextColor(Color c)	Define a cor do texto do hint.

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
setOnClickListener	OnClickListener	onClick(View v)	Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.

setOnKeyListener	OnKeyListener	onKey(View v,int KeyCode, KeyEvent event)	Esse evento é disparado toda vez que a tecla é acionada, disparando o método onKey.
setOnFocusChangeListener	OnFocusChangeListener	onFocusChange(View v, boolean hasFocus)	Esse método é disparado toda vez quando um componente EditText ganha ou perde foco.

Widget Button

- Propriedades

Propriedade	Em XML	Em Java	Descrição
Text	android:text	setText(CharSequence c)	Nessa propriedade, você define o texto a ser exibido na tela.
Text color	android:textColor	setTextColor(Color c)	Nessa propriedade, você define cor do texto a ser exibido na tela. Valor: #000000 até #FFFFFF.
Text size	android:textSize	setTextSize(float tamanho) ou setTextSize(int unidade, int tamanho)	Define o tamanho do texto. O tamanho da fonte pode ser especificado em várias notações : px (pixels),sp(scaled-pixels) , mm(milímetros), in (polegadas) e etc.
Typeface	android:typeface	setTypeface(Typeface fonte)	Essa propriedade serve para definir uma fonte ao texto (normal,sans,serif,monospace).

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
setOnClickListener	OnClickListener	onClick(View v)	Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.
setOnKeyListener	OnKeyListener	onKey(View v,int KeyCode, KeyEvent event)	Esse evento é disparado toda vez que a tecla é acionada, disparando o método onKey.

Widget CheckBox

- Propriedades

Propriedade	Em XML	Em Java	Descrição
Text	android:text	setText(CharSequence c)	Nessa propriedade, você define o texto a ser exibido na tela.

Text color	android:textColor	setTextColor(Color c)	Nessa propriedade, você define cor do texto a ser exibido na tela. Valor: #000000 até #FFFFFF.
Checked	android:checked	setChecked(boolean estado)	Nessa propriedade você define o estado do CheckBox, se estará marcado (true) ou não (false).

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
setOnClickListener	OnClickListener	onClick(View v)	Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.
setOnCheckedChangeListener	OnCheckedChangeListener	onCheckedChanged (CompoundButton cb,boolean b)	Esse evento será disparado toda vez que o estado do CheckBox for modificado, ou seja, marcado ou desmarcado, disparando o método onCheckedChanged

Widget RadioButton

- Propriedades

Propriedade	Em XML	Em Java	Descrição
Text	android:text	setText(CharSequence c)	Nessa propriedade, você define o texto a ser exibido na tela.
Text color	android:textColor	setTextColor(Color c)	Nessa propriedade, você define cor do texto a ser exibido na tela. Valor: #000000 até #FFFFFF.
Checked	android:checked	setChecked(boolean estado)	Nessa propriedade você define o estado do RadioButton, se estará marcado (true) ou não (false).

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
setOnClickListener	OnClickListener	onClick(View v)	Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.
setOnCheckedChangeListener	OnCheckedChangeListener	onCheckedChanged (CompoundButton cb,boolean b)	Esse evento será disparado toda vez que o estado do RadioButton for modificado, ou seja, marcado ou desmarcado, disparando o método onCheckedChanged

Widget Spinner

- Propriedades

Métodos	Descrição
setAdapter(SpinnerAdapter a)	Nesse método você define os elementos que irão compor esse componente através de um vetor (array).
int getSelectedPosition()	Essa função retorna a posição do elemento selecionado. Por exemplo, se for o primeiro elemento, retorna 0, se for o segundo, retorna 1 e assim sucessivamente.
Object getSelectedItem()	Essa função retorna em um tipo Object, o item selecionado.
Object getItemAtPosition(int posicao)	Retorna em um tipo Object o elemento de uma determinada posição, passada como parâmetro.

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
setOnClickListener	OnClickListener	onClick(View v)	Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.
setOnItemClickListener	OnItemClickListener	onItemClick (AdapterView<?> a, View v, int I, long l)	Esse evento será disparado toda vez que um derminado item for clicado,

			disparando o método onItemClick.
setOnItemSelectedListener	OnItemSelectedListener	onItemSelected(AdapterView av, View v,int posição,long id) onNothingSelected(AdapterView av)	Esse evento será disparado toda vez que um derminado item for selecionado, disparando o método onItemSelected. Caso nenhum item seja selecionado, será disparado o método onNothingSelected.

Widget ListView

- Propriedades

Métodos	Descrição
setAdapter(SpinnerAdapter a)	Nesse método você define os elementos que irão compor esse componente através de um vetor (array).
int getSelectedPosition()	Essa função retorna a posição do elemento selecionado. Por exemplo, se for o primeiro elemento, retorna 0, se for o segundo, retorna 1 e assim sucessivamente.
Object getSelectedItem()	Essa função retorna em um tipo Object, o item selecionado.
Object getItemAtPosition(int posicao)	Retorna em um tipo Object o elemento de uma determinada posição, passada como parâmetro.

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
setOnClickListener	OnClickListener	onClick(View v)	Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.
setOnItemClickListener	OnItemClickListener	onItemClick (AdapterView<?> a, View v, int I, long l)	Esse evento será disparado toda vez que um derminado item for clicado, disparando o método onItemClick.

setOnItemSelectedListener	OnItemSelectedListener	onItemSelected(AdapterView av, View v,int posição,long id) onNothingSelected(AdapterView av)	Esse evento será disparado toda vez que um derminado item for selecionado, disparando o método onItemSelected. Caso nenhum item seja selecionado, será disparado o método onNothingSelected.

Widget ImageView

- Propriedades

Propriedade	Em XML	Em Java	Descrição
Src	android:src	setImageResource(int Id)	Nessa propriedade, você define a imagem que será exibida na tela.
		setImageURI(Uri link)	Esse método é similar ao método acima, sendo que aqui você especifica o Uri (como se fosse um link de internet) como caminho de localização da imagem.

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
setOnClickListener	OnClickListener	onClick(View v)	Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.

Widget Gallery

- Propriedades

Método	Descrição
setAdapter(SpinnerAdapter a)	Nesse método você define os elementos que irão compor esse componente através de um vetor (array).

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
setOnClickListener	OnClickListener	onClick(View v)	Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.
setOnItemClickListener	OnItemClickListener	onItemClick (AdapterView<?> a, View v, int l, long l)	Esse evento será disparado toda vez que um derminado item for selecionado, disparando o método onItemClick.

ProgressBar

- Propriedades

Propriedade	Em XML	Em Java	Descrição
Style	style		Nessa propriedade, você define o estilo da progressbar. Essa propriedade assume os seguintes valores: "?android:attr/progressBarStyleHorizontal", "?android:attr/progressBarStyle" "?android:attr/progressBarStyleLarge" "?android:attr/progressBarStyleSmall"
		setMax(int valor_maximo)	Neste método você define o valor máximo da faixa. Ou seja, se definir o valor máximo como 100, a faixa de progresso será entre 0 e 100.
		setMax(int valor_maximo)	Neste método você define o valor máximo da faixa. Ou seja, se definir o valor máximo como 100, a faixa de progresso será entre 0 e 100.
		setProgress(int progresso)	Neste método, você define o valor corrente do progresso.
		incrementProgressBy(int incr)	Neste método você define o quando o progresso será incrementado.

DatePicker

- Propriedades

Métodos	Descrição
init(int ano, int mês, int dia, onDateChangedListener evento)	Neste método você define o valor inicial do ano , mês e dia inclusive também, você define um evento toda vez que uma data for

	modificada (onDateChangedListener).
updateDate(int ano, int mês, int dia)	Neste método você atualiza a data passando como parâmetros o ano, o mês e o dia.
int getYear()	Esse método retorna o ano da data.
int getMonth()	Esse método retorna o mês do ano, sabendo se que, para o primeiro mês (janeiro) ele retorna 0, para o segundo mês (fevereiro) retorna 1 e assim por diante.
int getDayOfMonth()	Esse método retorna o dia do mês.

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
Init	onDateChangedListener	onDateChanged(View v, int ano, int mês , int dia)	Esse evento é disparado toda vez que a data for alterada, disparando o método onDateChanged.

TimePicker

- Propriedades

Métodos	Descrição
setCurrentHour(int hora)	Neste método você define o valor da hora sabendo se que , uma hora.
setCurrentMinute(int minuto)	Neste método você define o valor do minuto entre 0-59.
int getCurrentMinute()	Esse método retorna o minuto corrente.
int getCurrentHour()	Esse método retorna a hora corrente.

- Eventos

Método que define o evento	Evento	Métodos relacionados	Descrição
			Esse evento é disparado toda

setOnTimeChangedListener	OnTimeChangedListener	onTimeChanged(TimePicker tp, int hora, int minuto)	vez que a hora for alterada, disparando o método onTimeChanged.

Propriedades comuns a todos os objetos

- Propriedades

Propriedade	Em XML	Em Java	Descrição
Id	android:id		Nessa propriedade , definimos o nome do nosso componente.
Layout width	android:layout_width		Nessa propriedade, você define a largura do componente a ser exibido. Normalmente essa propriedade assume dois valores : “fill_parent” (preenche toda a largura restante do dispositivo) e “wrap_content” (a largura do componente será definida de acordo com o seu conteúdo) . Também podem especificar valores números com suas respectivas escalas, ex: “160px”,”50sp” e etc.
Layout height	android:layout_heigth		Nessa propriedade, você define a altura do componente a ser exibido. Normalmente essa propriedade assume dois valores : “fill_parent” (preenche toda a altura restante do dispositivo) e “wrap_content” (a altura do componente será definida de acordo com o seu conteúdo) . Também podem especificar valores números com suas respectivas escalas, ex: “160px”,”50sp” e etc.
Visibility	android:visibility	setVisibility(int modo_visibilidade)	Essa propriedade serve para definir se o componente estará visível ou não. Ela assume os seguintes valores : “visible”,”invisible” e “gone”.

Conclusão

Nesta apostila aprendemos a desenvolver aplicações em Android para diversas finalidades. Vimos um pouco sobre a plataforma Android, como ela surgiu e tudo mais. Aprendemos a instalar o eclipse e os plugins necessários para o funcionamento do Android, incluindo o SDK. Aprendemos a construir uma aplicação Android básica e depois conhecemos os componentes (widgets) que constituem uma aplicação Android para a construção de aplicações mais interessantes. Vimos também como trocar de layouts em uma aplicação Android tendo como exemplo prático, uma aplicação de cadastro. Aprendemos a usar menus e submenus em uma aplicação e por último, compreendemos melhor o funcionamento da classe AlertDialog.Builder.

Espero que esse material lhe tenha sido útil.

Abraços