www.apostilaandroid.net

Apostila de Android - Programando Passo a Passo

Programação Básica (Versão Android Studio)

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

/ApostilaAndroid

[EDIÇÃO FREE]

Abril/2015

Aviso sobre esta apostila

Antes de iniciar a leitura deste material, veja esse aviso:

Este material usa a licensa Creative Commons isto significa

que **ELE PODE SER DISTRIBUÍDO LIVREMENTE**, porém, **SOBRE AS SEGUINTES REGRAS** :

Esse material **NÃO PODERÁ SER COMERCIALIZADO**
Essa material **NÃO PODERÁ SER DEVIRADO**
E todos os créditos do autor **DEVERÃO SER MANTIDOS**

O Sucesso da Apostila de Android

A Apostila de Android – Programando Passo a Passo é hoje referência didática de material sobre desenvolvimento de aplicações e sistemas para a plataforma Google Android, conhecido tanto aqui no Brasil quanto em outros países (somando-se mais de 65 países). Hoje a Apostila de Android já chegou a aproximadamente 200.000 acessos (feito pelos mais diversos usuários como estudantes e profissionais da área de programação) Hoje ela é usada por universidades e professores que ministram cursos sobre desenvolvimento Android (Tanto na área de criação de aplicações e jogos).

Sobre o Autor da Apostila

Luciano Alves da Silva é Bacharelado em Ciência da Computação pela UNISUAM (Rio de Janeiro – RJ) e Pós-Graduado em Docência do Ensino Superior pelo Instituto A Vez do Mestre (Universidade Cândido Mendes – UCAM, no Rio de Janeiro). Possui conhecimento e domínio das linguagens de programação Pascal, Java, C/C++, C#, Visual Basic, Delphi, PHP e HTML. Já criou Ambientes de Desenvolvimento Integrado (conhecidos como IDE) como o MakeWare (que trabalha com as linguagens Pascal, C++ e Java) e o AlgoWare (interpretador de algoritmos).

É autor também dos seguintes livros, pela editora AGBOOK

- Aprenda Passo a Passo a Programar em Android – Guia Essencial para Desenvolvedores

- Desenvolvendo Jogos com a Plataforma XNA 2ª Edição – Guia para Desenvolvedores.

- Desenvolvendo Jogos com o Framework MONOGAME – Guia para Desenvolvedores.

- Desenvolvendo Jogos 2D com Plataforma Java – Guia para Desenvolvedores.

Apresentação

O 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 foram criadas com a linguagem Java, o que facilita muitos programadores com conhecimentos em Java (ou de outras linguagens próximas de Java como C++ e C#) a desenvolver aplicações para o Android.

Esta apostila tem por objetivo mostrar de modo fácil e claro como desenvolver aplicações para dispositivos móveis que utilizam o sistema operacional Google Android através a ferramenta Android Studio (baseada na IDE IntelliJ IDEA).

Para quem dedico este material?

Este material é dedicado aos usuários experientes ou iniciantes em programação (tanto para Desktop, Móbile e etc.), que já tenha algum contato com a linguagem Java ou com uma de suas derivadas (como C/C++ ou C#).

Índice analítico

Capitulo 1 Visão geral sobre o Google Android 7

1.1) Introdução 7

1.2) Estrutura Geral da plataforma Google Android 9

1.2.1) A arquitetura do Android 10

1.2.2) Aplicações 10

1.2.3) Android Runtime 11

1.2.4) Linux Kernel 11

1.3) Para qual versão do Android devemos desenvolver as aplicações ? ... 12

Capitulo 2 Instalando e Configurando a Ferramenta de Desenvolvimento para Android 14

2.1) A ferramenta de desenvolvimento Android Studio 14

2.2) Observações gerais sobre o Android Studio 15

2.3) Baixando e configurando os componentes da ferramenta de desenvolvimento 17

2.3.1) A Máquina Virtual Java 17

2.3.2) O Android SDK 21

2.3.3) O Android Studio 35

Capitulo 3 Começando a programar no Android 50

Conhecendo a estrutura geral de um projeto no Android Studio 58

O diretório “app” (application) 59

O diretório “res” (resources) 62

O diretório “drawable” 62

O diretório “layout” 63

O diretório “values” 63

O diretório “mipmap” 64

O diretório “menu” 64

Visão geral da ferramenta de desenvolvimento 64

Executando a nossa aplicação 66

Capitulo 4 Conhecendo as widgets do Android 86

4.1) A paleta de componentes e suas widgets 86

4.1.1) A seção “Widgets” 87

4.1.2) A seção “Text Fields” 90

4.1.3) A seção “Layouts” 91

4.1.4) A seção “Containers” 92

4.1.5) A seção “Date & Time” 92

4.1.6) A seção “Expert” 94

Capitulo 5 Construindo nossas aplicações no Android 95

5.1) Desenvolvendo uma Calculadora Básica 95

Aplicação da calculadora em execução 115

5.2) Desenvolvendo uma aplicação simples de compras 115

5.3) Desenvolvendo uma aplicação de cálculo de salário 121

5.4) Desenvolvendo uma aplicação de lista de contatos 130

5.5) Desenvolvendo uma aplicação que visualiza imagens 134

Capitulo 6 Trabalhando com mais de uma tela em uma aplicação 145

6.1) Desenvolvendo um Sistema de Cadastro (Primeira versão) 153

Capitulo 7 Trabalhando com menus em uma aplicação 177

Capitulo 8 Propriedades e eventos dos componentes trabalhados 187

Widget TextView 187

Widget EditText 188

Widget Button 190

Widget CheckBox/RadioButton 191

Widget ListView 192

Widget ImageView 193

Conclusão a respeito do material 195

Capitulo 1 Visão geral sobre o Google Android

1.1) Introdução

Conforme mencionado na apresentação deste material, o Android é uma plataforma desenvolvida pela Google voltada para dispositivos móveis, totalmente aberta é livre (Open Source), que foi divulgada em 5 de novembro de 2007. Inicialmente o sistema Android foi desenvolvido pelo Google e atualmente essa plataforma é mantida pela OHA (Open Handset Alliance. Visite o link : http://www.openhandsetalliance.com), um grupo constituído por aproximadamente 84 empresas as quais se uniram para inovar e acelerar o desenvolvimento de aplicações e serviços, com o objetivo e trazer aos consumidores uma experiência mais rica em termos de recursos, menos dispendiosa em ternos financeiros para o mercado móvel.

Um dos primeiros SmartPhones que ofereceu suporte a esse sistema operacional foi o G1 da empresa T-Mobile. Confira na imagem seguinte:

G1 - T-Mobile

Atualmente o sistema Android se encontra hoje disponível tanto em SmartPhones quanto nos Tablets. Confira abaixo alguns dos dispositivos encontramos hoje no mercado com o sistema operacional Android:

SmartPhone Samsung Galaxy S3

Tablet Samgung Galaxy Note

1.2) Estrutura Geral da plataforma Google Android

O Android SDK é uma ferramenta de desenvolvimento que disponibiliza um conjunto de APIs necessárias para desenvolver aplicações para a plataforma Android, utilizando a linguagem Java.

Vamos conhecer os recursos encontrados nessa plataforma:

- **Application framework:** Permite a reutilização e substituição de componentes ;

- **Dalvik virtual machine:** É uma Máquina Virtual Java (JVM) voltada para dispositivos móveis ;

- **Browser Integrado** baseado no webkit engine ;

- **Gráficos Otimizados** O Android é constituído por bibliotecas 2D e 3D baseada na especificação OpenGL ES 1.0 ;

- **SQLite:** Sistema Gerenciador de Banco de Dados (SGBD) já embutido no Android para guardar dados ;

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

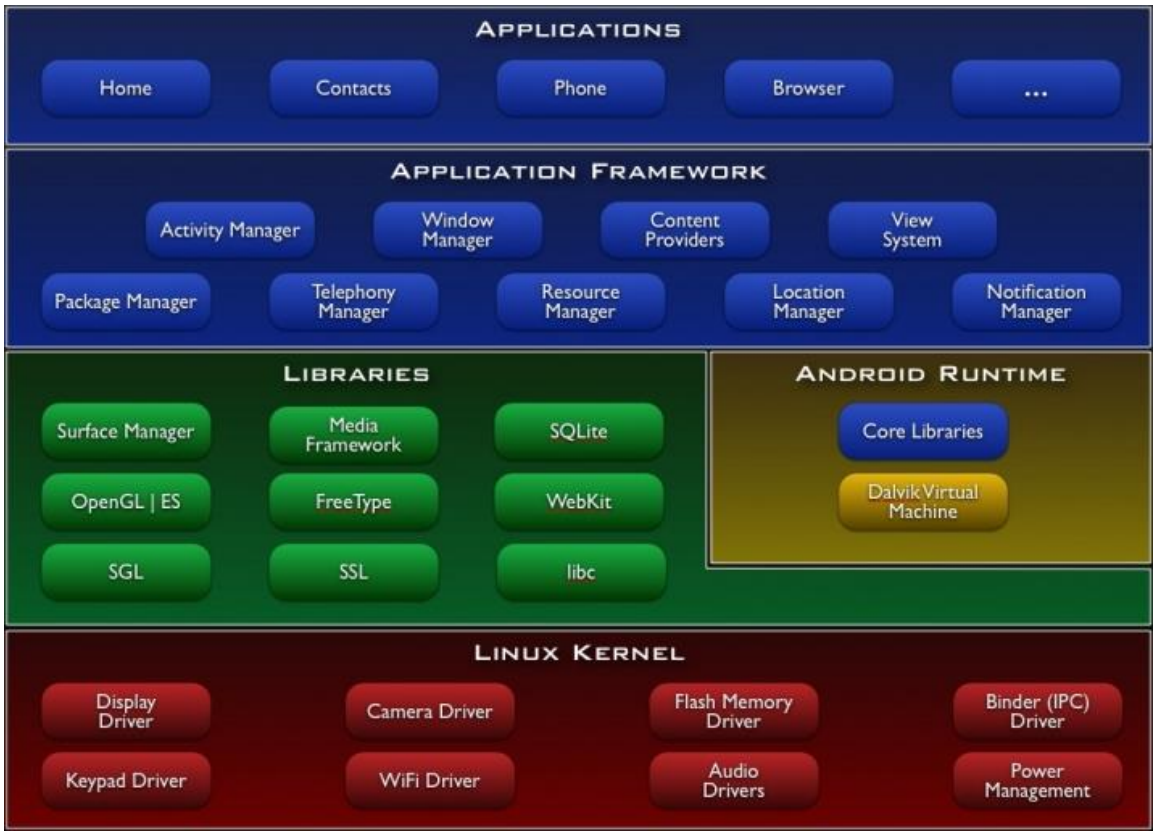
- **Telefonia GSM** (dependente de hardware) ;

- **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 e performance.

1.2.1) A arquitetura do Android



Arquitetura geral da plataforma

1.2.2) Aplicações

O Android nos fornece um conjunto de aplicações fundamentais, são elas:

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

Todos os aplicativos acima presentes no Android foram desenvolvidos na linguagem de programação Java.

O Android nos fornece um conjunto de bibliotecas C/C++ utilizadas por vários componentes do sistema. Veja algumas das bibliotecas abaixo:

- **System C library:** Consiste em uma implementação derivada da biblioteca C padrão baseado no sistema (libc) do BSD sintonizada para dispositivos rodando Linux.

- **Media Libraries:** Baseado no PacketVideo's OpenCORE; são as bibliotecas que suportam os mais diversos formatos de áudio e vídeo, incluindo também imagens.

- **Surface Manager:** Responsável pelo acesso ao subsistema de exibição bem como as múltiplas camadas de aplicações 2D e 3D;

- **LibWebCore**: Consiste em 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** – Biblioteca responsável pela renderização de fontes bitmap e vector;

- **SQLite** – Conforme já mencionado, consiste no sistema gerenciador de banco de dados (SGBD) relacional disponível para todas as aplicações.

1.2.3) Android Runtime

O Android é constituído por um conjunto 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 foi baseado no kernel do Linux para funcionalidades subjacentes como o encadeamento e a gestão de baixo nível de memória.

1.2.4) Linux Kernel

O Android foi projetado em cima da 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.

1.3) Para qual versão do Android devemos desenvolver as aplicações ?

Quando desenvolvemos uma aplicação para um determinado sistema operacional, normalmente, precisamos fazer a seguinte pergunta : Para qual versão do S.O devemos desenvolver ?

Considero esse um dos pontos mais importantes que devemos refletir antes de desenvolvermos uma aplicação. Como neste material iremos desenvolver aplicações voltados para a plataforma Android, devemos pensar para qual versão da plataforma precisamos desenvolver.

1.3.1) Qual versão da plataforma Android é a mais utilizada no momento ?

Para falar a respeito dessa situação, irei mostrar aqui um gráfico que mostra quais versões do Android são as mais usadas no mundo todo :

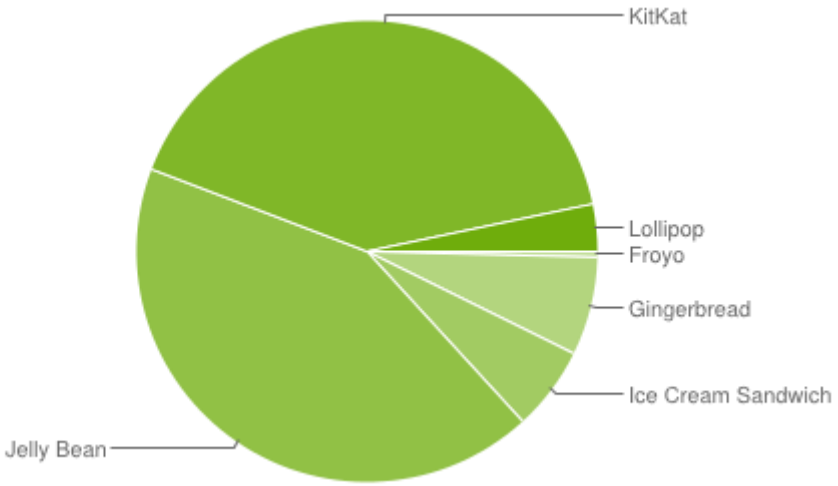


Gráfico de Estatística a respeito do S.O Android mais usado

O gráfico da estática acima foi feito em dezembro de 2014, onde nele podemos observar que as versões do Android mais utilizadas são o Jelly Bean (versão 4.1 – 4.3) e KitKat (4.4), praticamente as duas estão no empate. As versões mais antigas do Android como Eclair (versão 2.1) e Donut (versão 1.6) já nem são mais citadas e faladas hoje em dia. A versão 5.x do Android recentemente lançada, o Lollipop, ainda está começando a se popularizar.

Hoje em dia, se fomos em alguma loja para comprar um aparelho (Smartphone ou Tablet Android) iremos adquiri-lo com o S.O Android versão 4.0 para cima. Se olharmos por esse ponto de vista, devemos pensar em desenvolvermos nossa aplicação utilizando, como base, a versão 4.0.

1.3.2) O Publico

Um outro fator muito importante , e que destaco aqui , é a questão O PUBLICO. Nesse fator , a questão S.O deveria ser deixada “teoricamente” de lado, visto que muitos usuários ainda possuem aparelhos Android com uma versão mais antiga (como a versão 2.3 e 2.2), logo, devemos pensar também nesses usuários para “usufruir” das nossas aplicações desenvolvidas.

1.3.3) Qual prioridade devemos dar : Publico ou Versão do S.O ?

Agora a questão é : Como combinar a questão PUBLICO e VERSÃO do S.O para desenvolvermos a aplicação ? Se você pretende desenvolver uma aplicação Android simples (como um pequeno sistema de cadastro), podemos, se quisermos, dar prioridade a questão PUBLICO, procurando desenvolver sua aplicação Android para uma versão mais antiga, porém, ainda NÃO OBSOLETA . Agora se você desenvolver uma aplicação Android cheia de muitos recursos, cujos componentes só existem em versões mais atuais do sistema, devemos dar prioridade a questão VERSÃO do S.O.

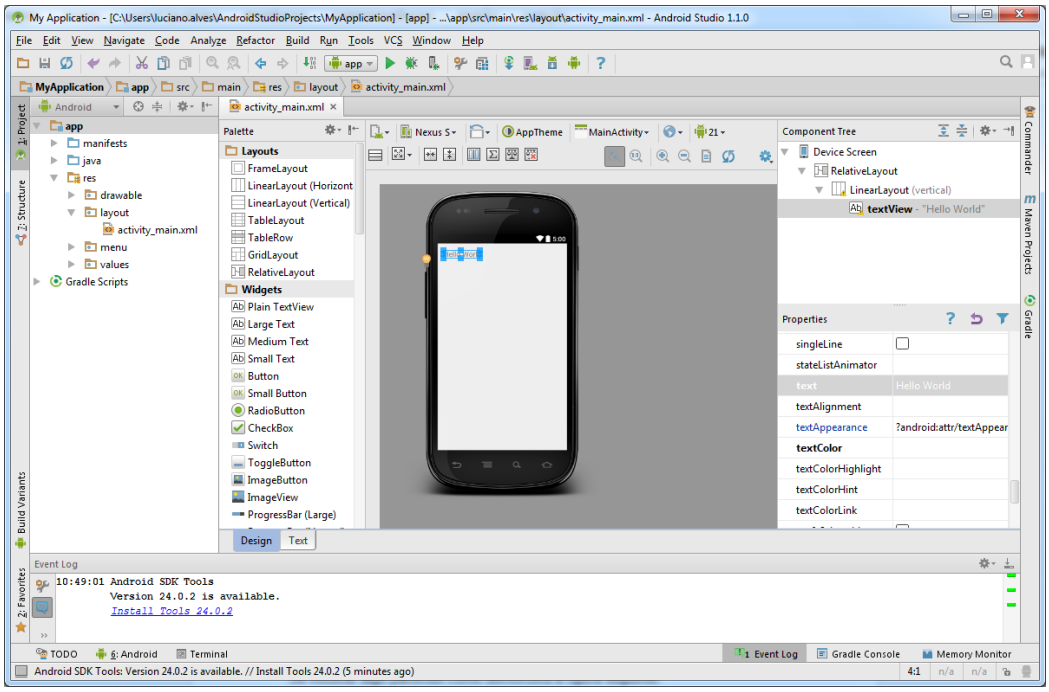
Capitulo 2 Instalando e Configurando a Ferramenta de Desenvolvimento para Android

Para a elaboração dessa nova versão do material, fiz o uso da nova ferramenta de desenvolvimento para Android, o Android Studio (baseado na ferramenta IntelliJ IDEA). Para que a ferramenta de desenvolvimento funcione é necessário que você tenha instalado, antes de tudo, a Máquina Virtual Java (de preferência a versão 7 ou posterior). Bom, mãos a obra.

2.1) A ferramenta de desenvolvimento Android Studio

Há algum tempo atrás a ferramenta de desenvolvimento Eclipse com o plugin ADT focado para o desenvolvimento Android (junto com o Android SDK), deixou de ser a ferramenta de desenvolvimento oficial para Android, sendo deixada “teoricamente” de lado (pois na prática ainda muitos programadores usam ela).

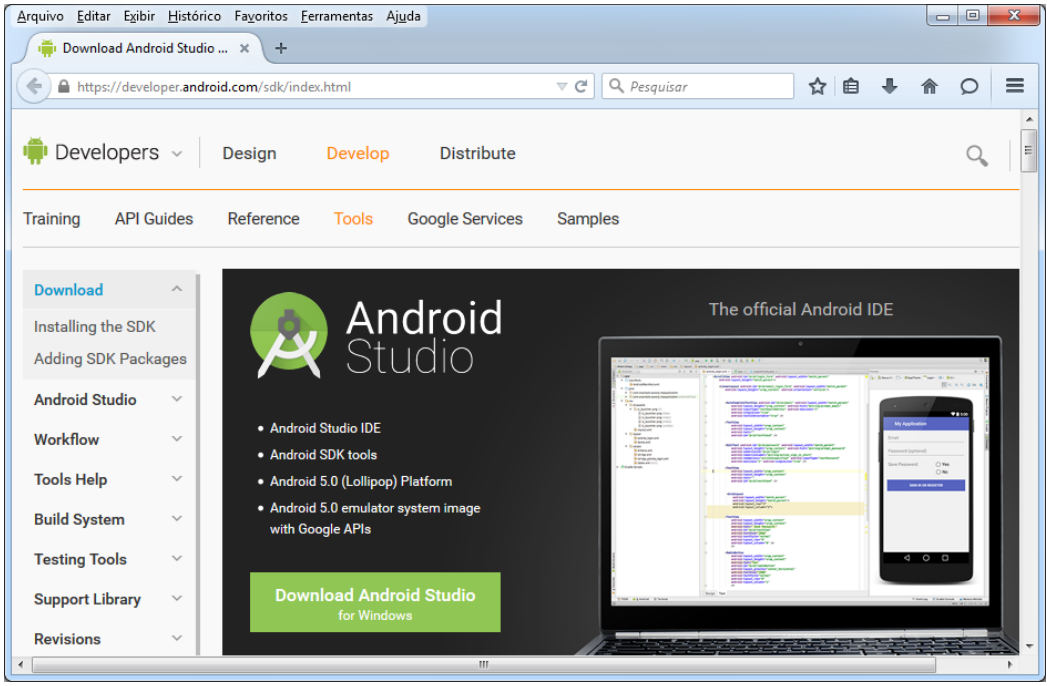
Agora o Google adotou como ferramenta oficial de desenvolvimento o Android Studio (baseada e inspirada na ferramenta IntelliJ IDEA) , onde podemos desenvolver nossas aplicações para Android utilizando a linguagem de programação Java. Vejamos a figura da IDE na imagem seguinte :



A ferramenta de desenvolvimento Android Studio

2.2) Observações gerais sobre o Android Studio

A ferramenta Android Studio encontra-se disponível para download no seu atual site oficial : **https://developer.android.com/sdk/index.html**. Eis a página do site carregada abaixo :



Site oficial da ferramenta Android Studio

NÃO BAIXE A VERSÃO SUGERIDA PELO SITE

Isso mesmo leitor, NÃO IREMOS BAIXAR a versão sugerida pelo site (conhecida como "bundle" (pacote) que já acompanha a IDE configurada com o Android SDK).

A versão "bundle" oferecida pelo site , além de ser bem pesada (quase 1GB), requer do computador onde será instalado configurações mais "pesadas" como processador, suporte a aceleração de HARDWARE e etc.

Possivelmente você (que está lendo esse material) e outras pessoas , encontraram muitos problemas durante a configuração e execução da versão "bundle" recomendada pelo site (assim como eu já tive).

Em resposta a alguns leitores das minhas apostilas, recomendei que para NÃO USEM a ferramenta Android Studio, devido a toda essa problemática encontrada na versão "bundle" (sugerindo o bom e velho "Android Developer Tools").

SOLUÇÃO MAIS VIÁVEL (E ALTAMENTE) RECOMENDADA

Depois de algumas análises e pesquisas sobre o Android Studio, descobri uma solução mais viável e prática para usarmos essa nova ferramenta (oficial) sem muitos problemas (independente "teoricamente" da configuração de sua máquina).

O ideal para quem for programar no Android Studio é baixarmos todos os seus componentes de desenvolvimento "separados", ou seja, baixar somente a IDE Android Studio (sem o SDK) e o Android SDK (avulso) realizando suas configurações de forma independente.

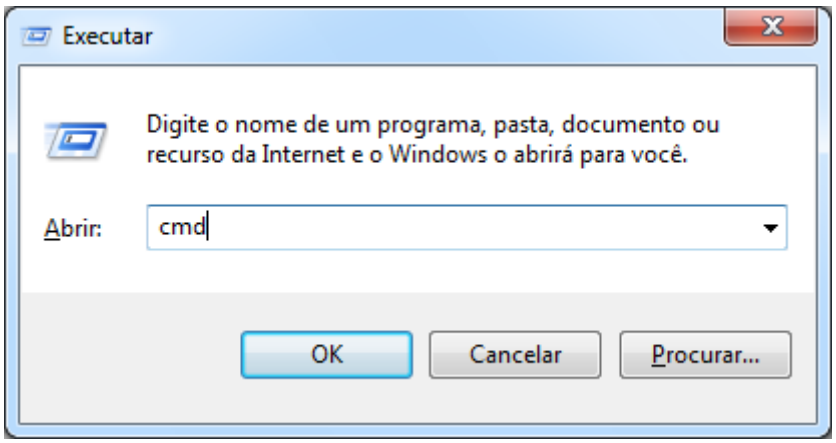
2.3) Baixando e configurando os componentes da ferramenta de desenvolvimento

2.3.1) A Máquina Virtual Java

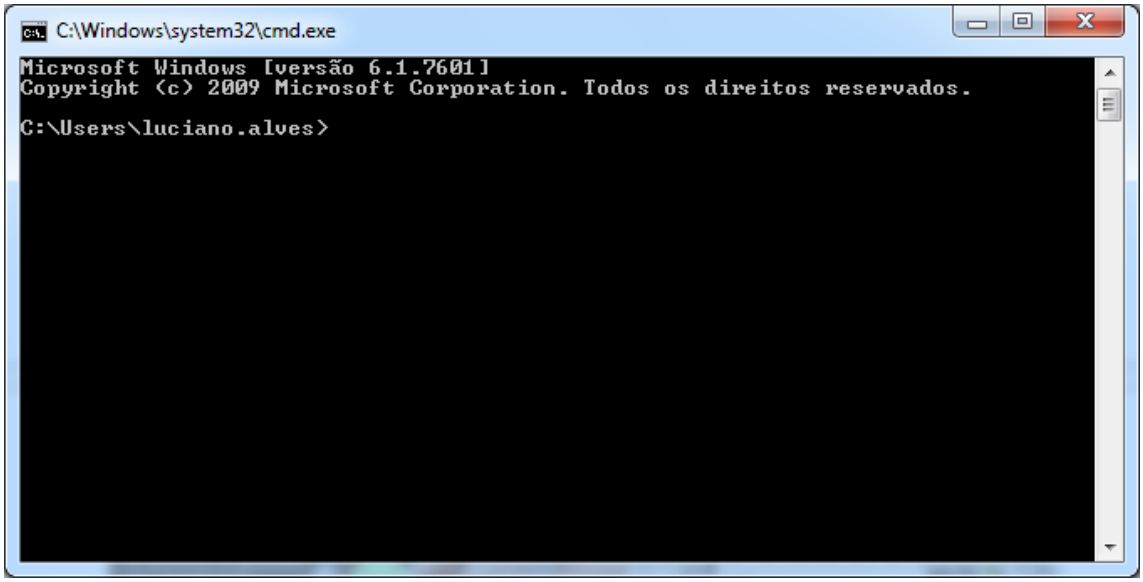
Bom, como todo bom programador deve saber, para iniciarmos o nosso desenvolvimento com Android devemos ter instalado (como pré-requisito do Android Studio e do Android SDK) a Máquina Virtual Java.

Como saber se eu tenho a Máquina Virtual Java instalada ?

Basta chamarmos o “Prompt de Comando” do Windows, digitando na caixa de diálogo “Executar” (pressioando as teclas “Windows” + “R”) o seguinte comando :



Feito isso clique em “OK” e será aberta a seguinte caixa de diálogo , conforme podemos ver a seguir :

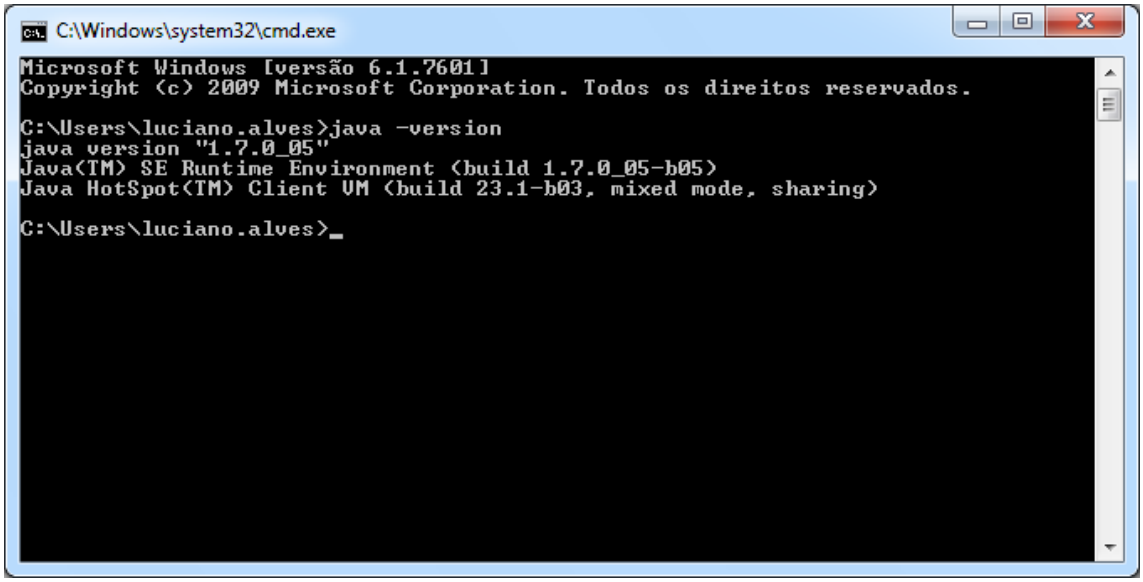


Prompt de comando do Windows

Experimente agora digitar o seguinte comando abaixo:

```
java -version
```

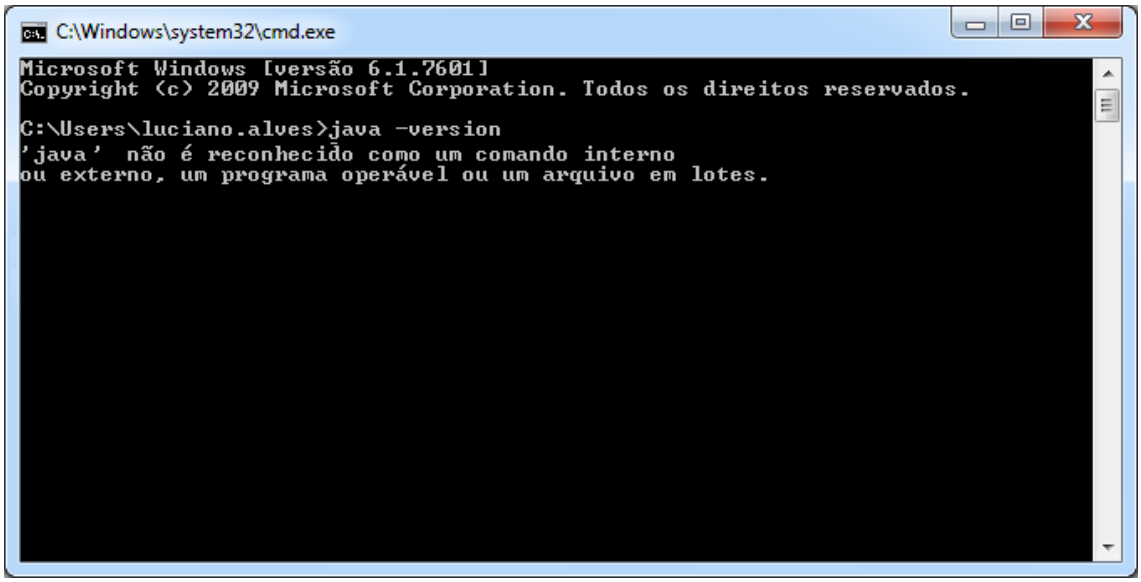
Se mostrar a seguinte mensagem a seguir :



Máquina Virtual Java instalada no computador

Significa que você tem a Máquina Virtual Java em seu computador, PORÉM, na versão do Android Studio utilizada nesta apostila (versão 1.1.0) será preciso que sua Máquina Virtual Java seja mais atualizada (como o do "Prompt de Comando" acima, com "Update 75"). Caso contrário, atualize a sua Máquina Virtual Java.

Agora, se ao digitar o comando solicitado aparecer :



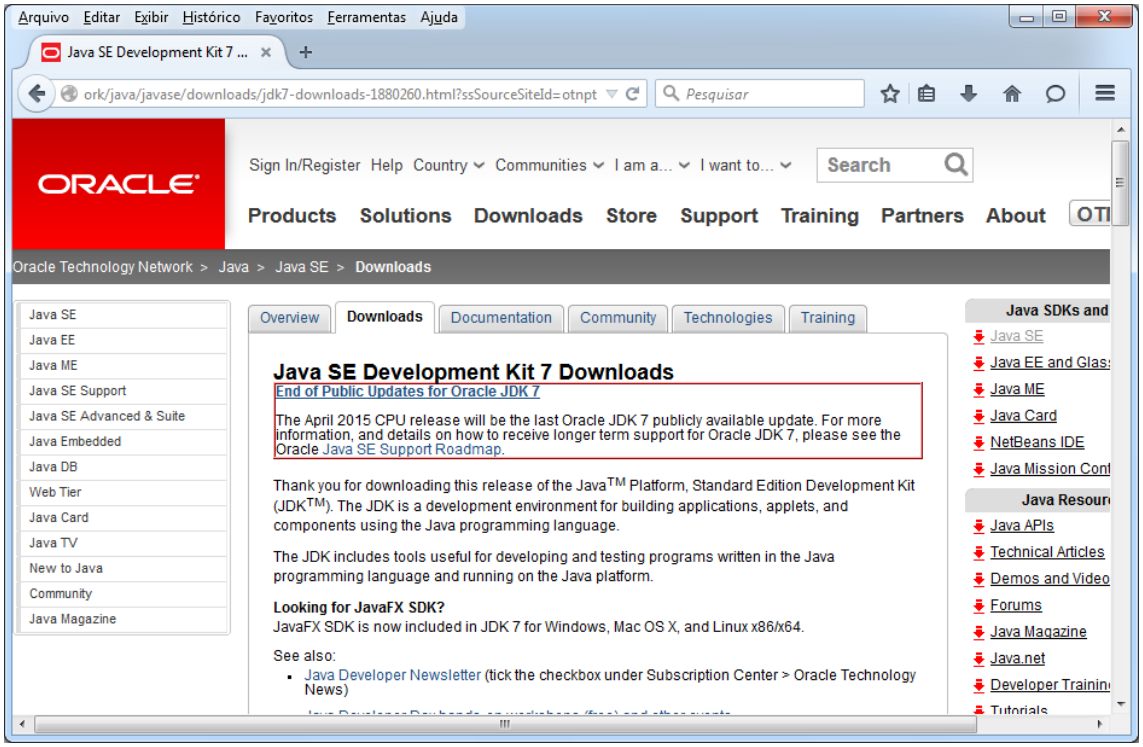
Sua Máquina Virtual Java não se encontra instalada

Significa que você precisa realizar o download da Máquina Virtual Java, pois a mesma não se encontra instalada no computador.

Para fazer o download da Máquina Virtual Java basta abrir o seguinte link em seguida :

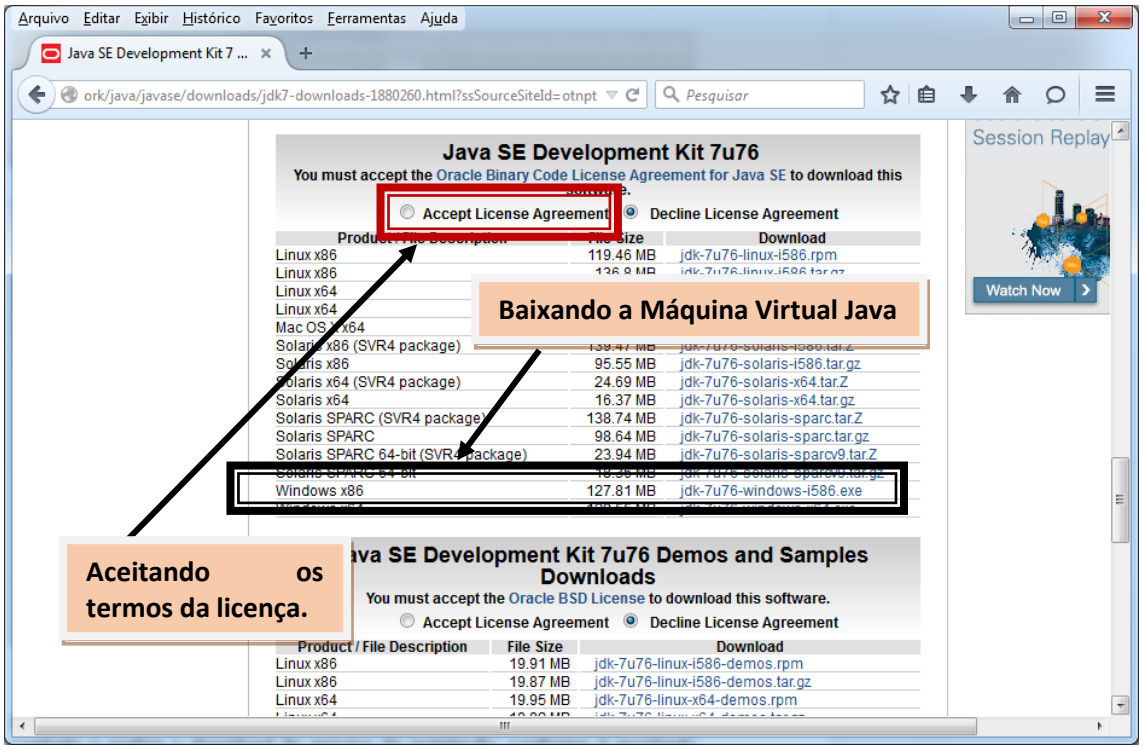
http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html?ssSourceSiteId=otnpt

Feito isso deverá se abrir a seguinte página em seguida:



Página de download do Java 7

Procure pela versão mais recente do Java 7 (bastando rolar a página até embaixo). Feito isso aceite os termos de contrato e realize o download do arquivo de instalação, conforme é mostrado em seguida:2.2.1) A Máquina Virtual Java



Baixando a Máquina Virtual Java

A instalação da Máquina Virtual Java é bem simples, basta ir seguindo toda a orientação exibida durante a instalação (praticamente só clicando em “Next”).

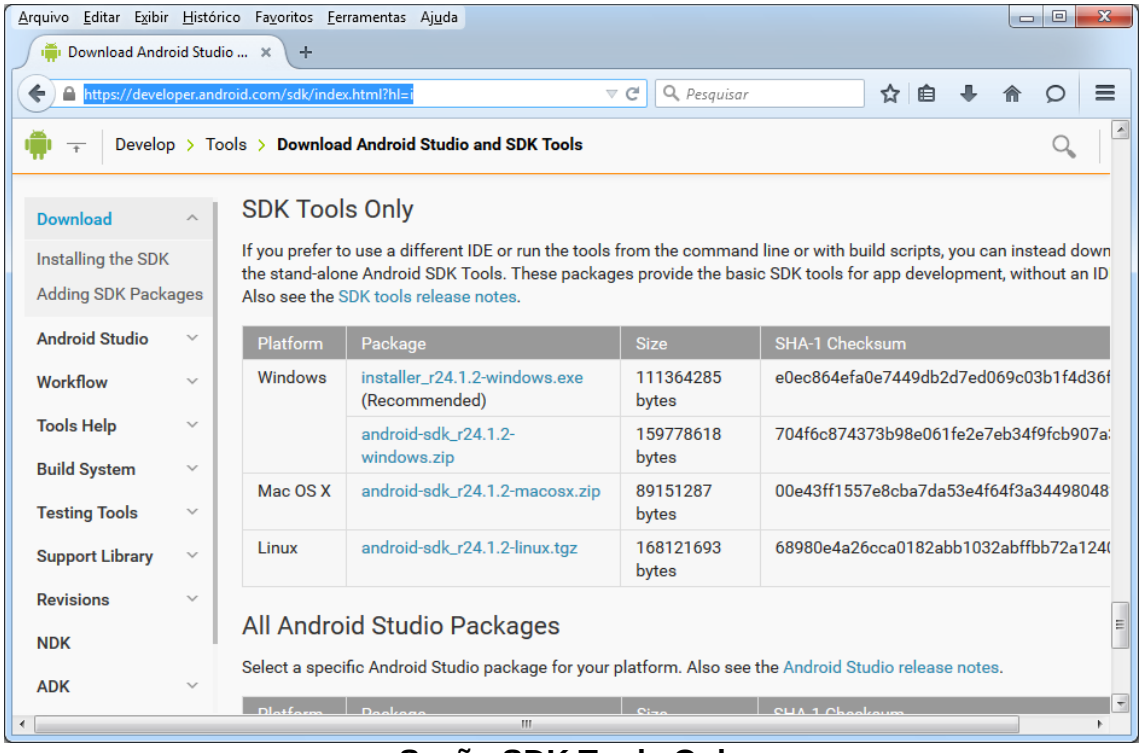
2.3.2) O Android SDK

Depois de instalado a Máquina Virtual em seu computador, vamos baixar e configurar agora o Android SDK que iremos utilizar junto com o Android Studio.

Como falei, o Android SDK que já vem configurado com a versão “bundle” do Android Studio vai requerer muito de sua máquina (e possivelmente irá apresentar muitos erros). Vamos baixar a versão “avulsa” dele, para isso basta visitar o seguinte link :

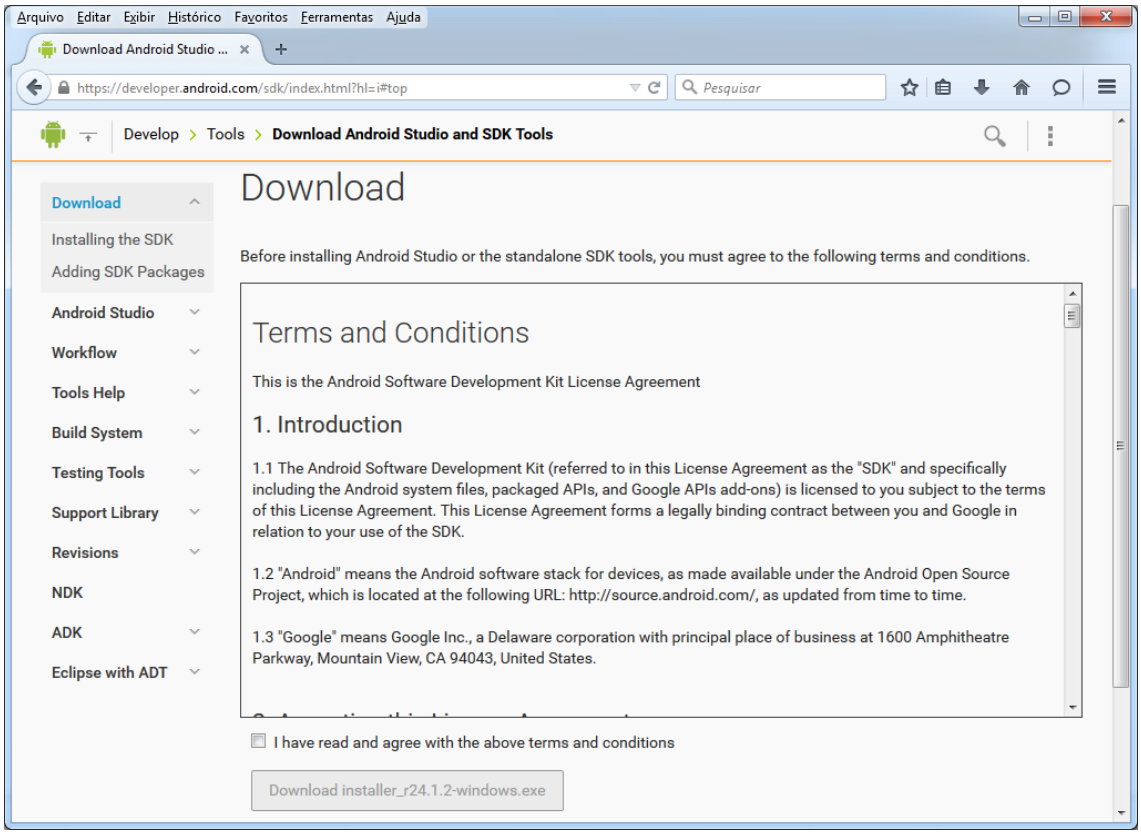
https://developer.android.com/sdk/index.html?hl=i

No link apresentado role a página abaixo e procure a seção "SDK Tools Only". Veja na figura em seguida :



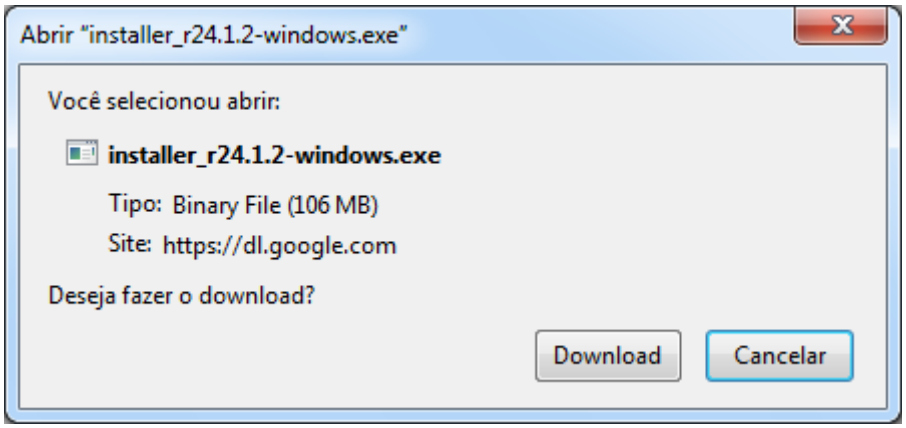
Seção SDK Tools Only

Nesta obra estarei fazendo uso da versão "r24.1.2" do Android SDK (qualquer versão superior serve). Basta clicar no link : "installer_r24.1.2-windows.exe" para abrir a seguinte caixa de diálogo em seguida :



Janela de Termos e Condições

Marque a opção “I have read and agree with the above terms and conditions” e em seguida clique no botão “Download installer_r24.1.2-windows.exe”, que será exibida a seguinte caixa de diálogo em seguida :

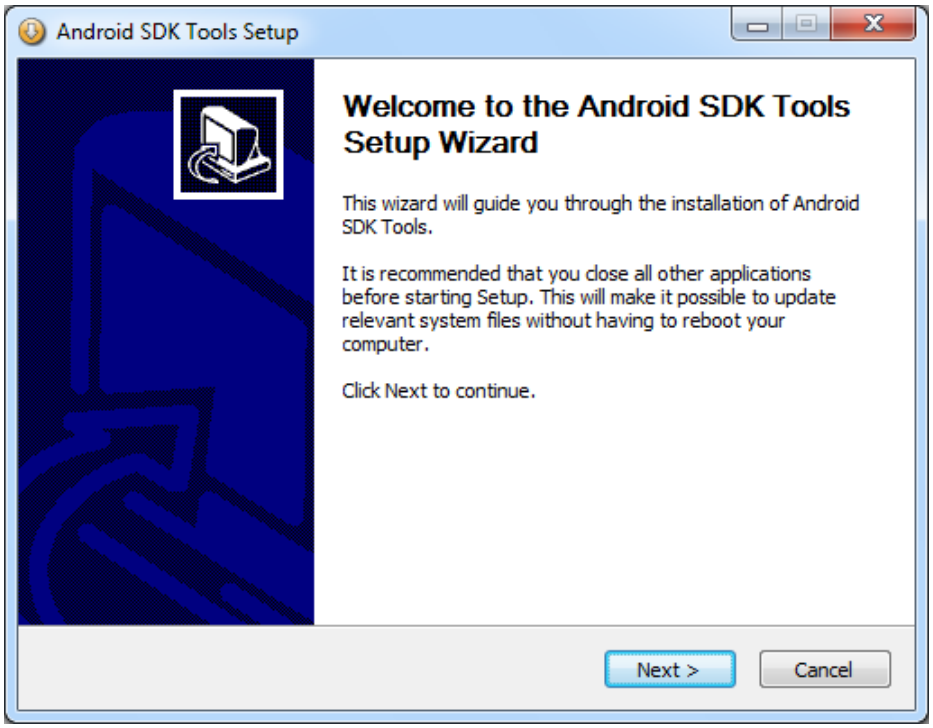


Caixa de diálogo de download do Android SDK

OBS : Caso você queira utilizar a mesma versão do Android SDK que está sendo utilizada aqui na obra, segue abaixo o link para download :

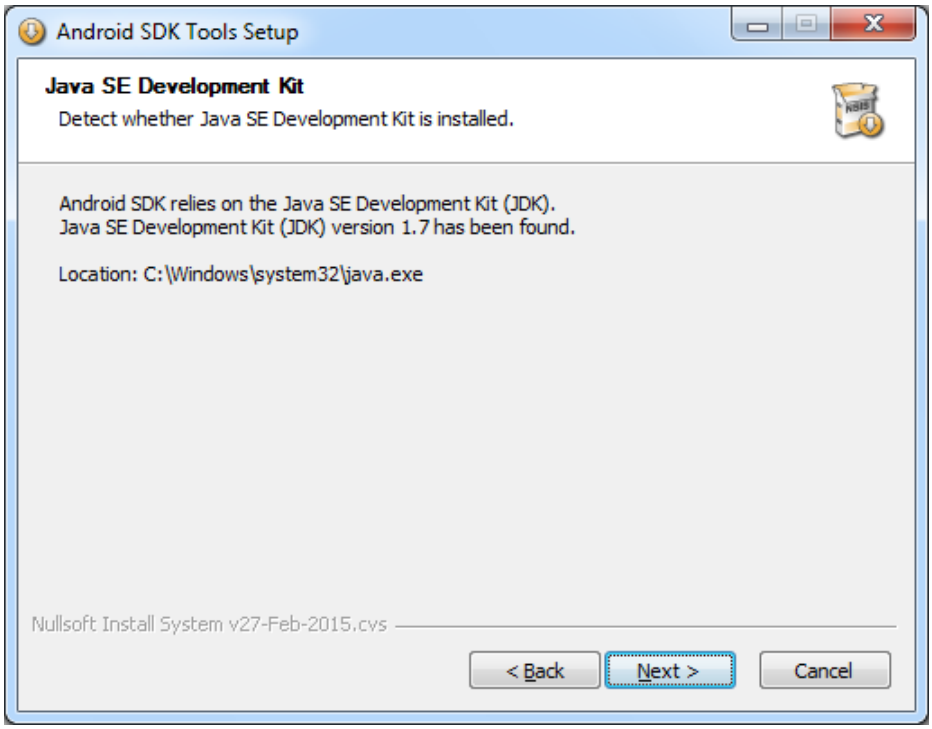
http://dl.google.com/android/installer_r24.1.2-windows.exe

Após efetuar o download do Android SDK, vamos executar o seu instalador. Feito veremos a seguinte tela em seguida :



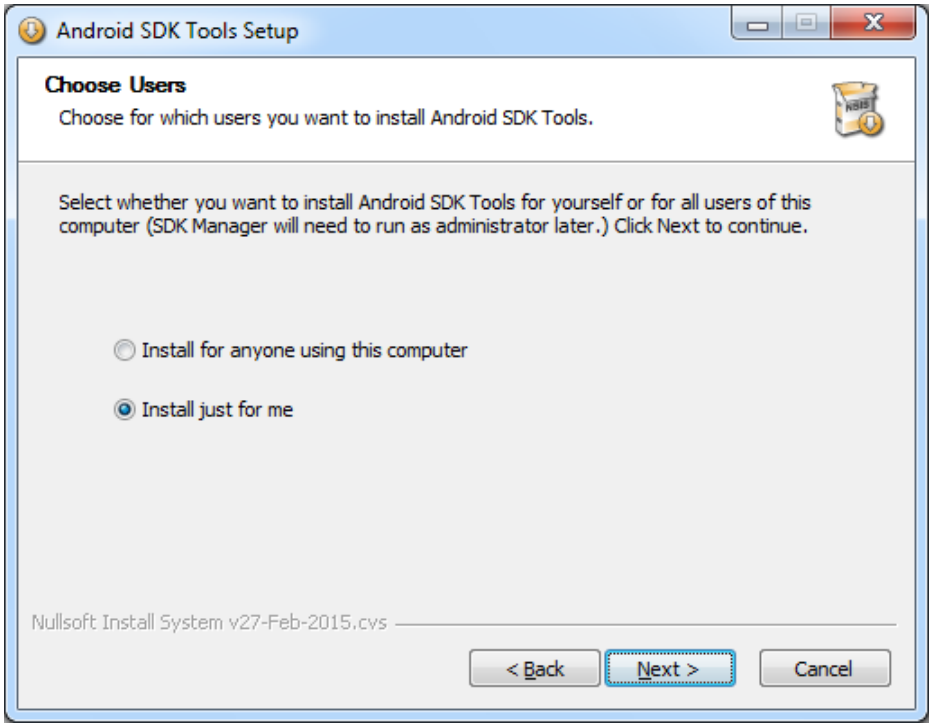
Instalação do Android SDK

Ao clicarmos no botão “Next” o mesmo irá verificar se existe uma Máquina Virtual Java instalada em seu computador (como pré-requisito), conforme podemos ver na figura seguinte :



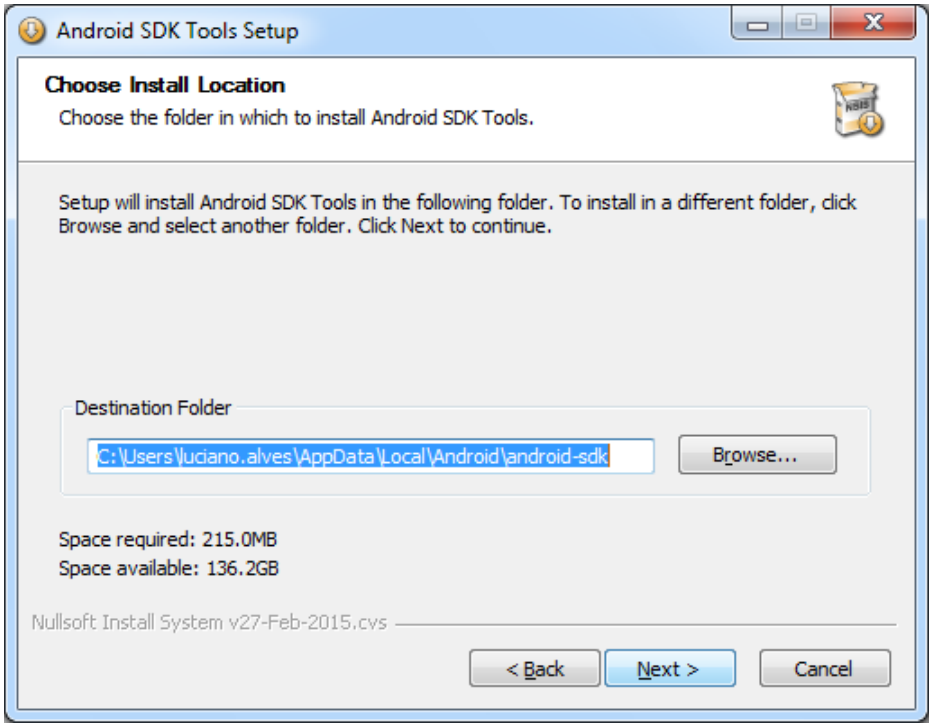
Instalação do Android SDK

Vamos clicar agora no botão “Next” para avançarmos para a próxima etapa. Na etapa seguinte é questionado para quais usuários será destinado o acesso ao Android SDK. Nessa opção a escolha fica por sua conta.



Instalação do Android SDK

Clicando em “Next” podemos escolher o local onde a ferramenta será instalada, conforme mostra a figura seguinte :

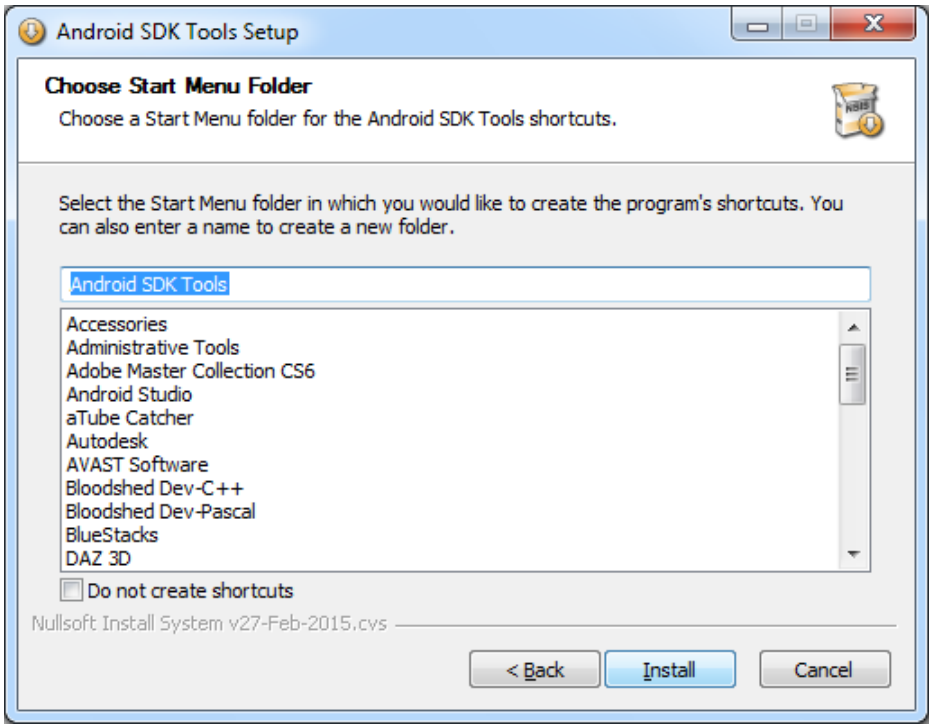


Instalação do Android SDK

Bom, nesta etapa iremos definir <u>COMO PADRÃO</u> o diretório “c:\android_sdk\”, conforme podemos ver em seguida :

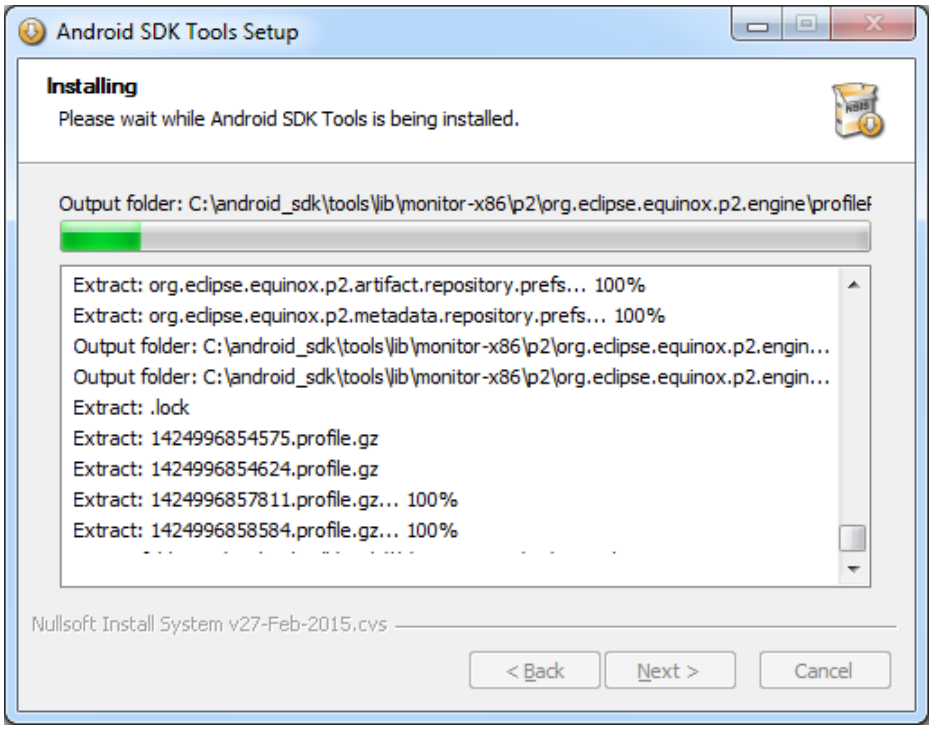
Mudando o diretório de instalação

Ná próxima etapa definimos o nome da pasta que será exibida no menu iniciar que servirá como atalho para iniciarmos as ferramentas oferecidas pelo Android SDK. Aqui não precisaremos modificar nada, deixe como está.



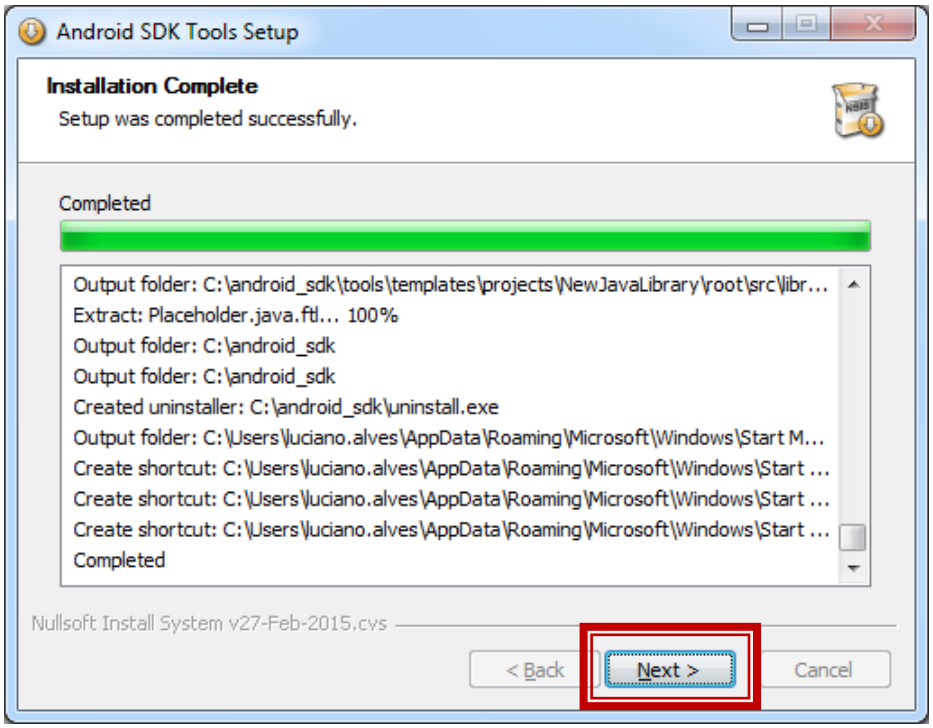
Instalação do Android SDK

Clique no botão “Install” para iniciarmos a instalação, conforme mostra a próxima figura :



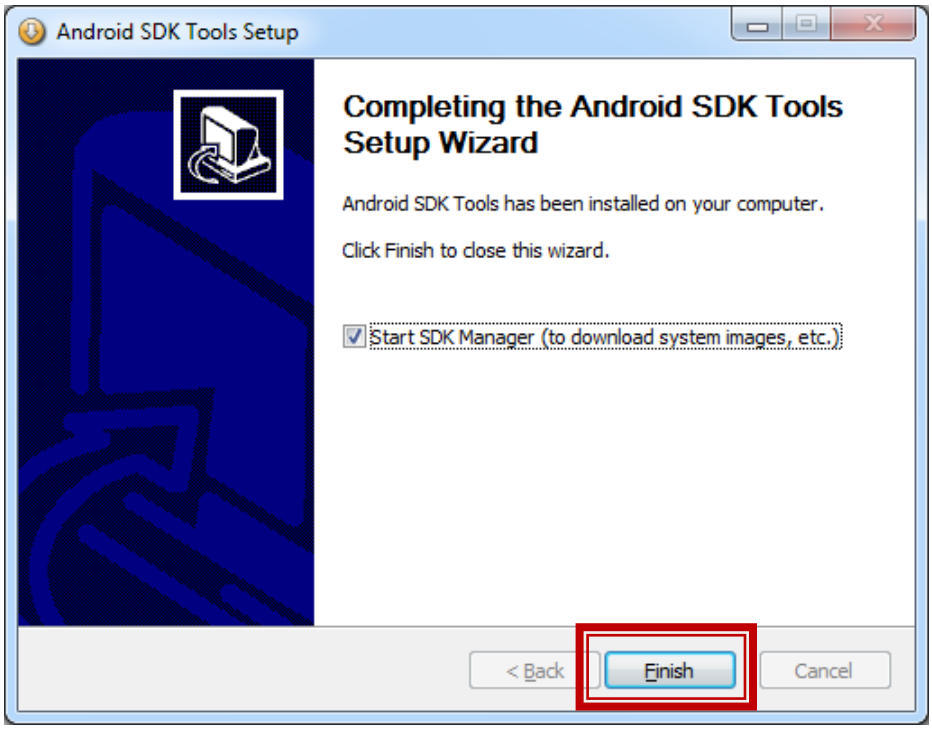
Instalação do Android SDK

Após concluir a instalação do Android SDK clique no botão “Next”, conforme mostra a próxima figura :



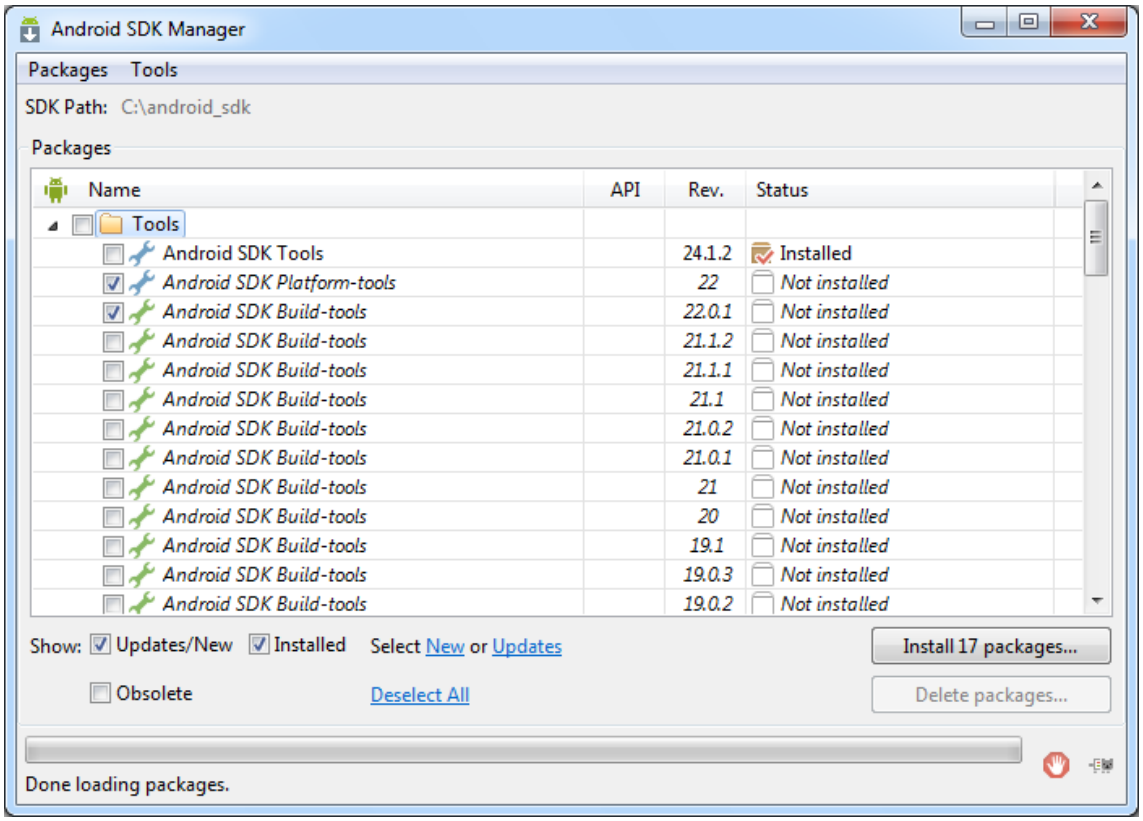
Instalação do Android SDK

Na próxima tela deixe marcado a opção “Start SDK Manager” e clique no botão “Finish” para finalizarmos e iniciarmos a execução do “SDK Manager”.



Finalizando a instalação do Android SDK

Após clicarmos no botão “Finish” será aberta a seguinte caixa de diálogo do “Android SDK Manager”, conforme mostra a figura seguinte :



Caixa de diálogo “Android SDK Manager”

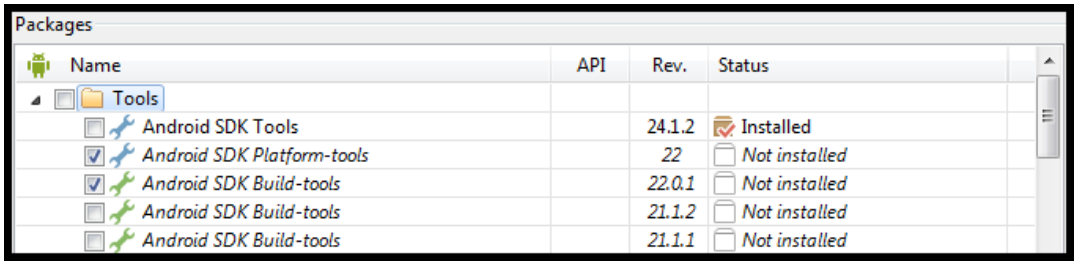
Através do Android SDK iremos baixar e configurar todos os recursos necessários que serão utilizados durante o desenvolvimento das nossas aplicações para Android no Android Studio.

Aqui para testarmos e executarmos as nossa aplicações em Android faremos uso de um emulador que irá simular um dispositivo com o sistema Android. No momento do desenvolvimento deste material, a versão oferecida do sistema Android erá a versão 5.1 (Lollipop). Essa versão ainda não ganhou popularidade, logo, iremos utilizar a versão 4.0.3 (Ice cream Sandwish), sendo esta a versão mais utilizada pelo mundo no momento em dispositivos Android.

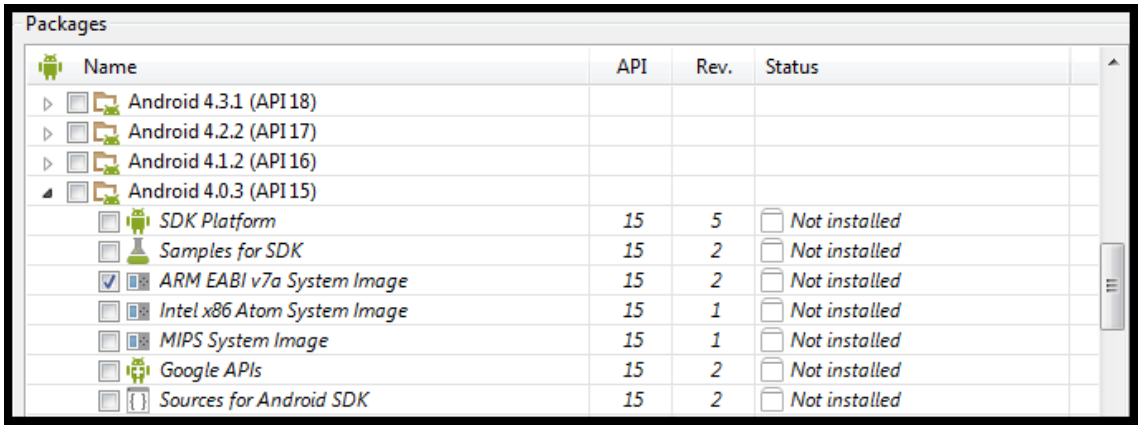
Vamos no Android SDK **"desmarcar"** <u>todas as opções</u> marcadas por padrão pela ferramenta e marcar as opções a seguir :

1) Marque "Android SDK Platform-tools" (marcar a versão mais atual)

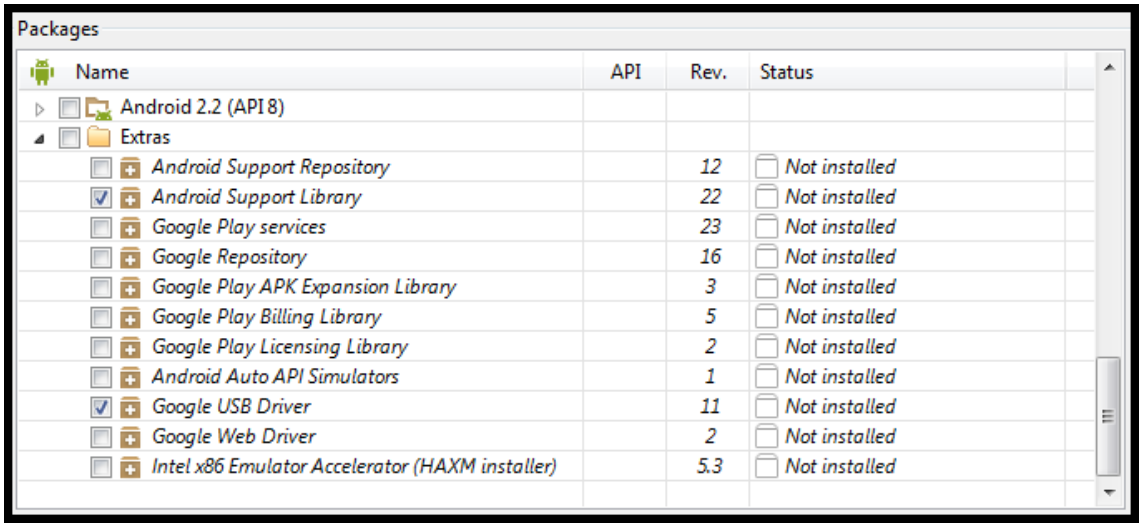
2) Marque "Android SDK Build-tools" (marcar a versão mais atualizada)



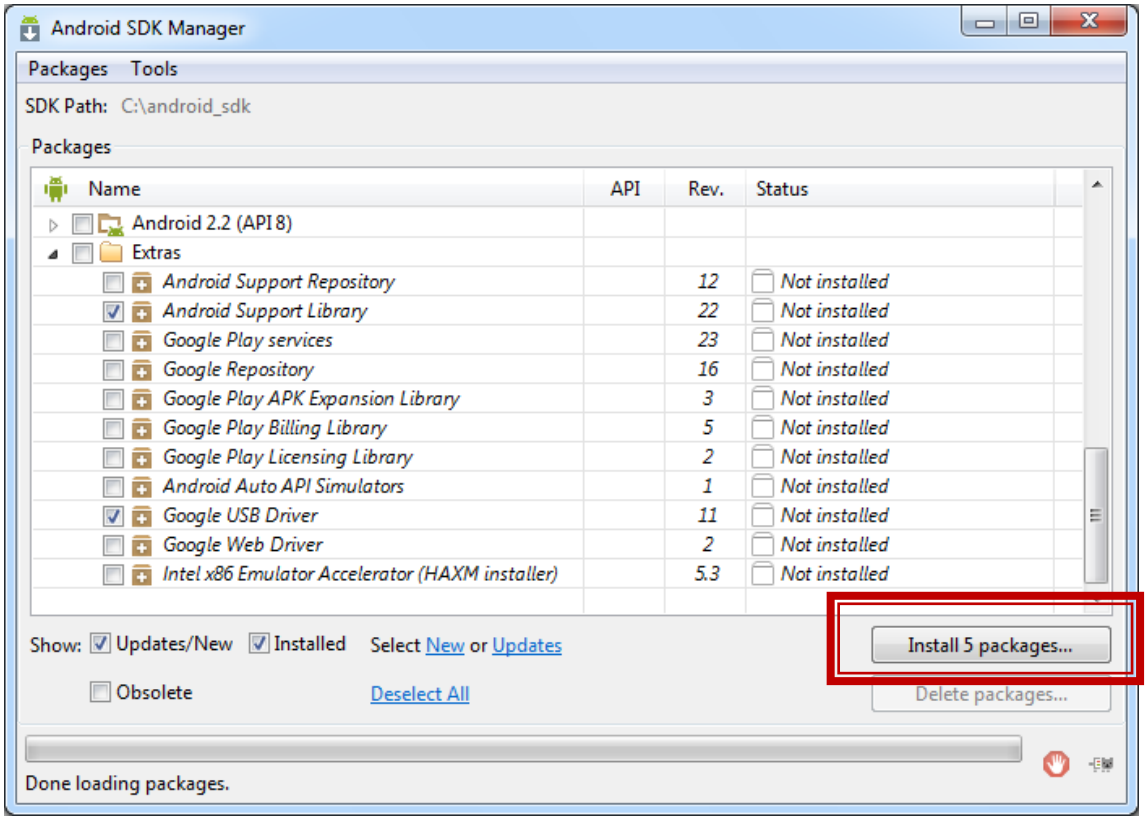
3) Expanda o item "Android 4.0.3 (API 15)" e marque a opção "ARM EABI v7 System Image".



4) Expanda a pasta “Extras” e marque as opções “Android Suport Library” e “Google USB Driver”

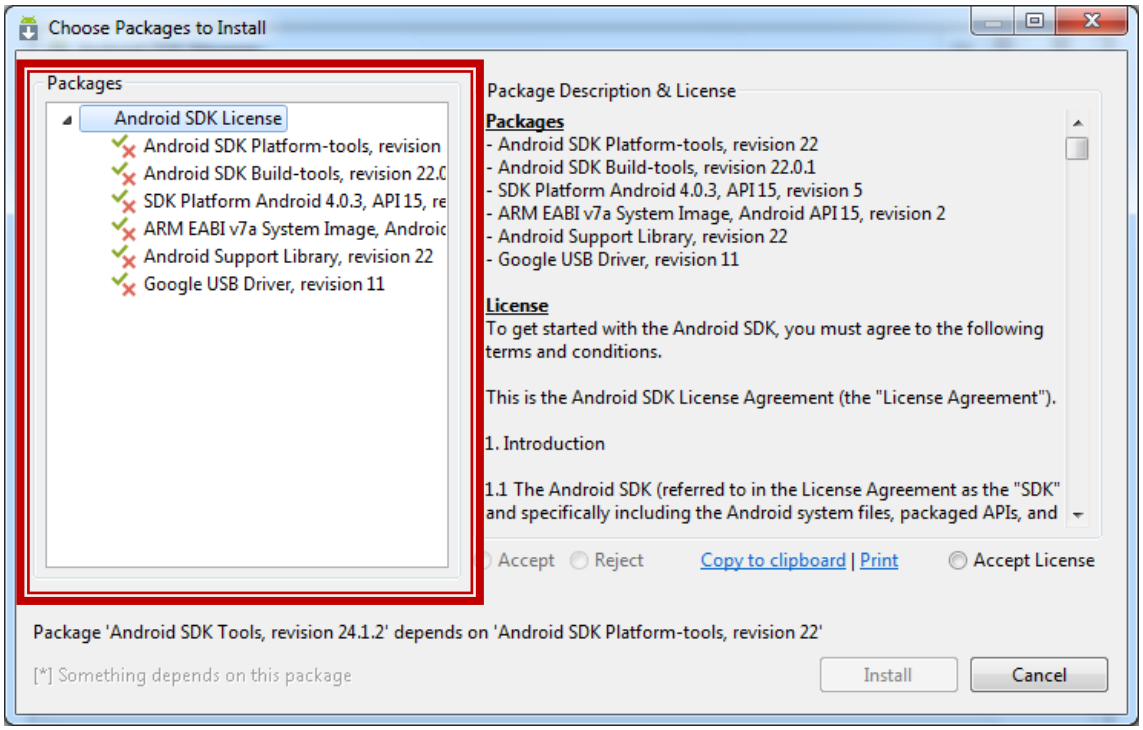


Depois disso vamos clicar no botão “Install 5 Packages”, conforme indica a próxima figura :



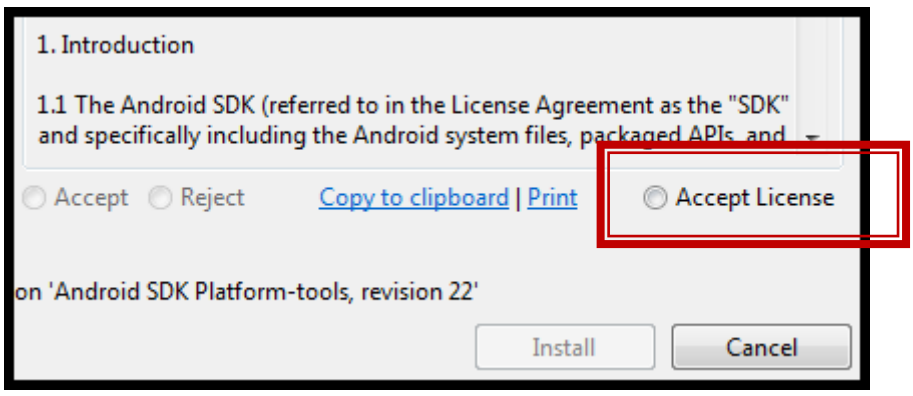
Clicando no botão “Install 5 packages”

Feito isso será aberta a seguinte caixa de diálogo a seguir:



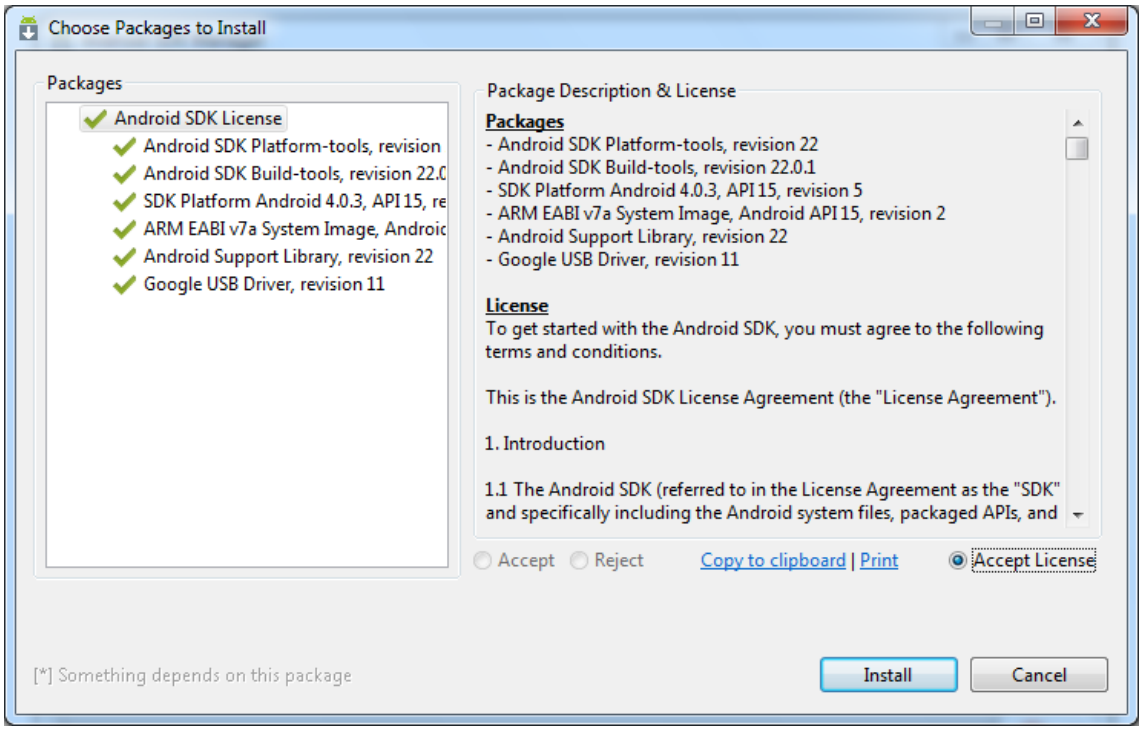
Caixa de diálogo – Choose Packages to Install

Observe que todos os itens de instação (da seção “Packages”) estão marcados com um “X” vermelho (conforme destacado acima). Para solucionarmos isso basta marcarmos a opção “Accept License”, conforme mostra a figura a seguir :



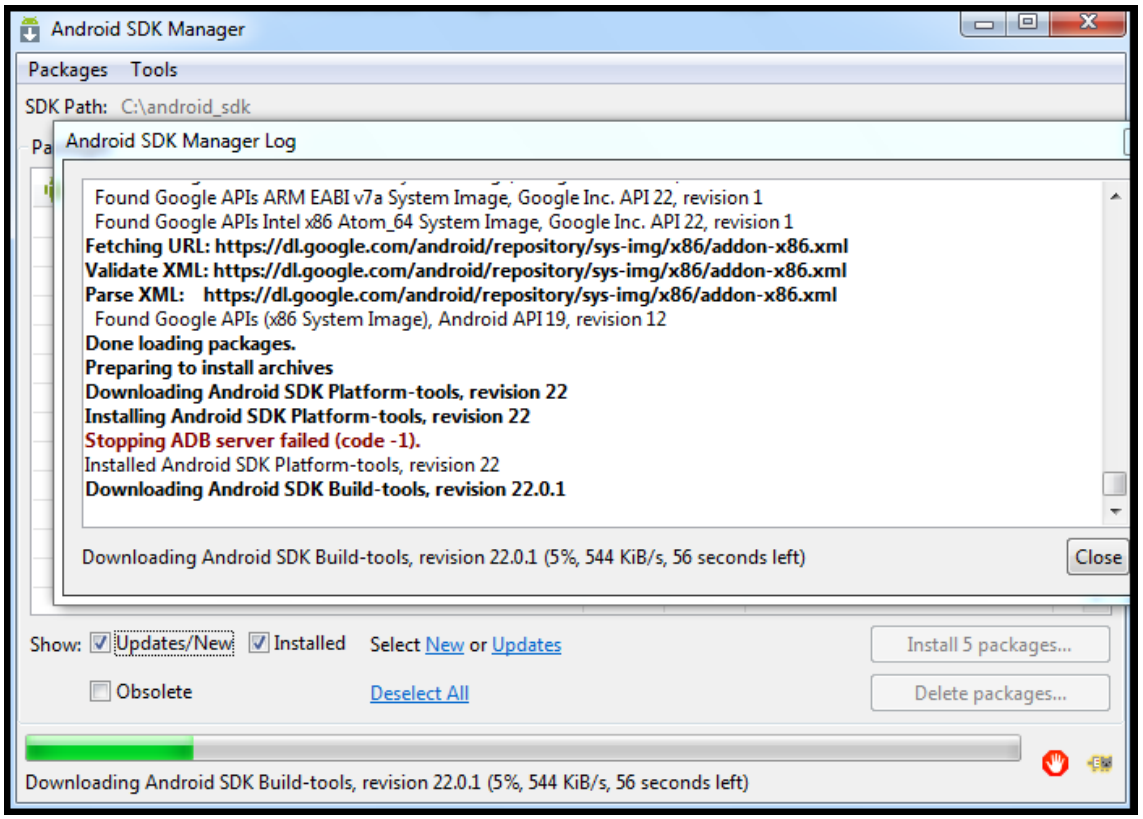
Marcando a opção “Accept License”

Feito isso teremos o seguinte resultado :



Caixa de diálogo – Choose Packages to Install

Clique no botão “Install” para instalarmos os itens selecionados. Feito isso será exibida a seguinte caixa de diálogo a seguir :



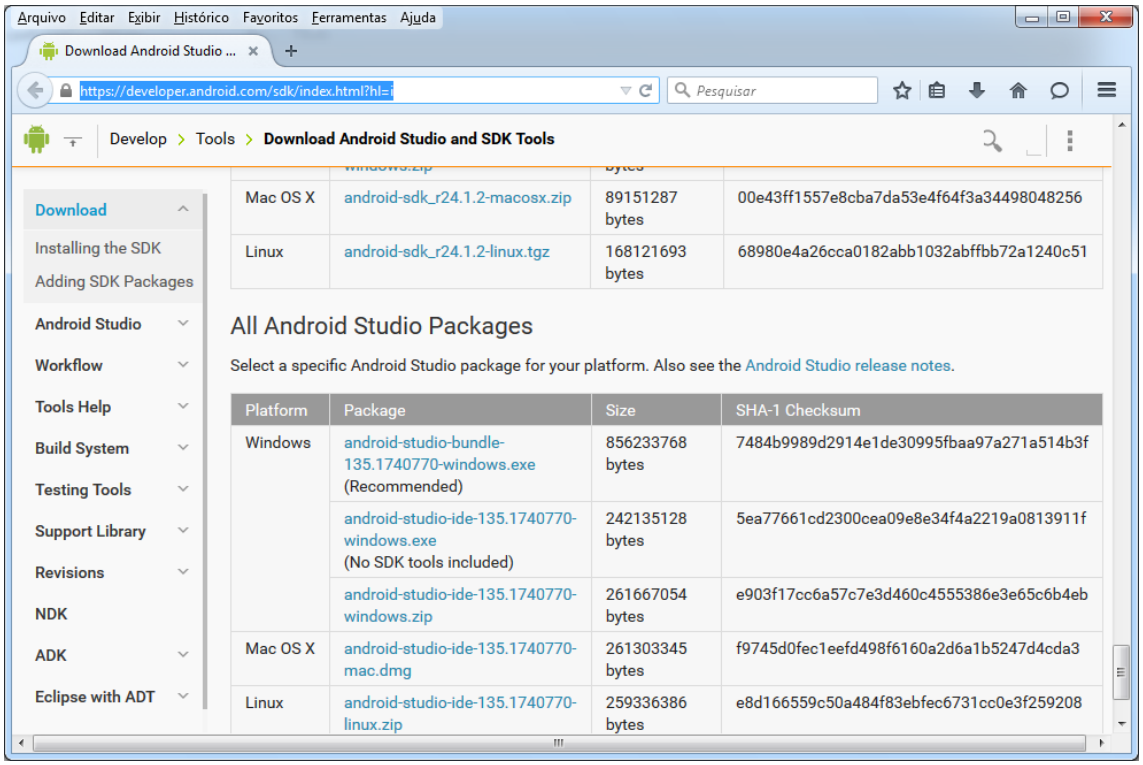
Instalação dos pacotes selecionados

Aguarde a instalação e configuração de todos os pacotes através do Android SDK Manager. Depois disso, podemos fechar o Android SDK Manager.

2.3.3) O Android Studio

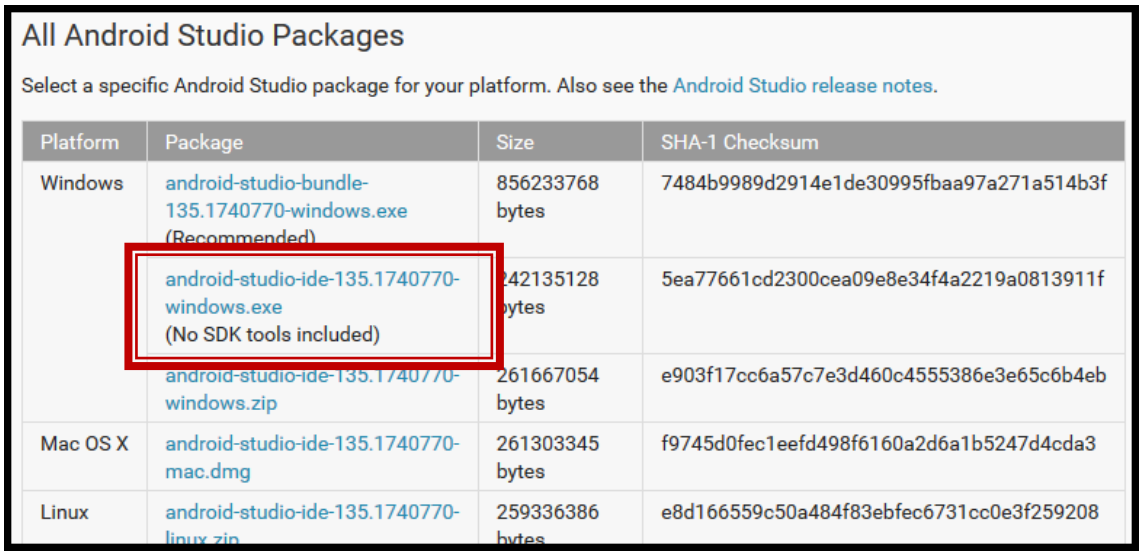
Agora para finalizar, vamos baixar e configurar o Android Studio. O download encontra-se na mesma página onde baixamos o Android SDK (**https://developer.android.com/sdk/index.html?hl=i**)

Após carregar a página, role até abaixo até encontrar a seção “All Android Studio Packages”, conforme mostra a figura seguinte :



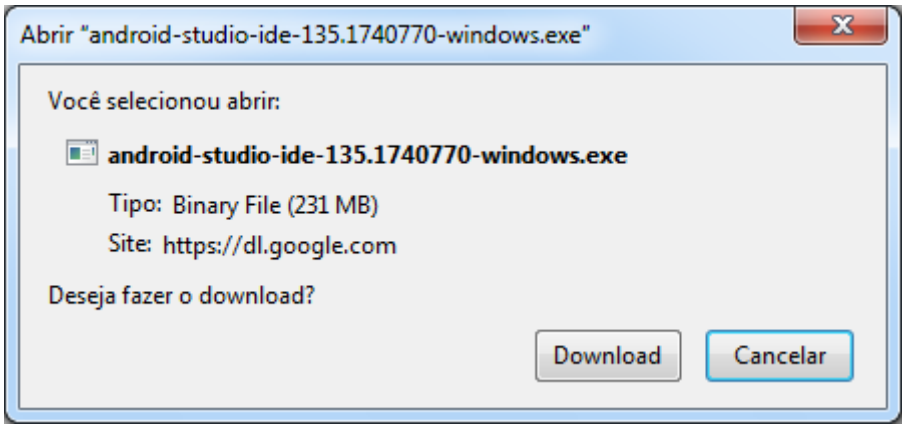
Seção All Android Studio Packages"

Agora vamos instalar a versão do Android Studio sem o SDK incluso (pois já baixamos ele por fora e configuramos), clicando no link indicado a seguir :



Selecionando a versão do Android Studio sem o SDK incluso

Feito isso irá se abrir a caixa de diálogo de termos e condições (como já vimos no “Android SDK”). Confirme os termos de condições e clique no botão de “Download”. Feito isso irá se abrir a caixa de diálogo abaixo :



Caixa de diálogo – Download do Android Studio

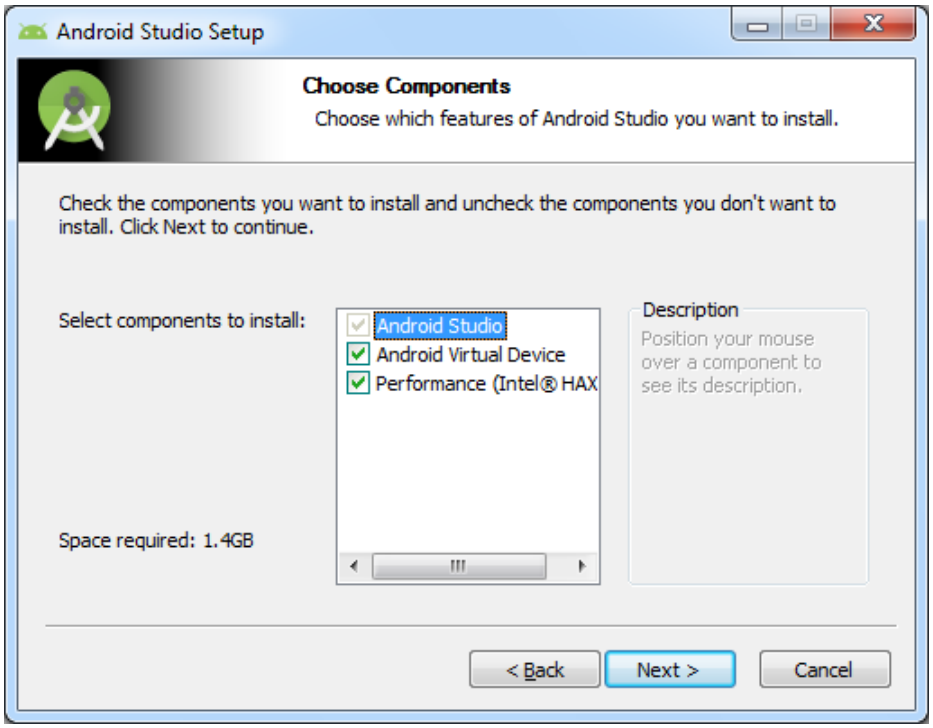
OBS : Caso você queira utilizar a mesma versão do Android Studio que está sendo utilizado neste material, segue abaixo o link para download :

https://dl.google.com/dl/android/studio/install/1.1.0/android-studio-ide-135.1740770-windows.exe

Após efetuar o download do Android Studio vamos executar o seu arquivo de instalação. Feito isso será aberta a seguinte caixa de diálogo a seguir :

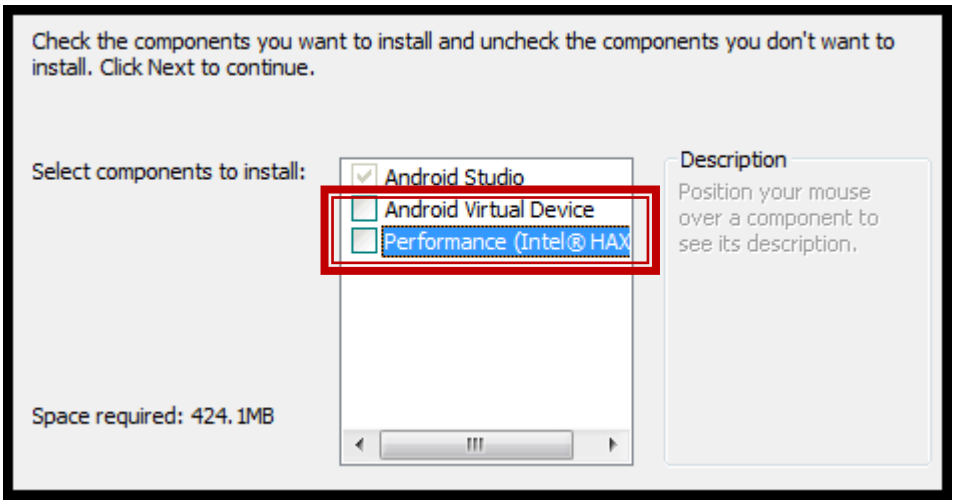
Instalação do Android Studio

Vamos clicar no botão “Next” para avançarmos com a instalação. Feito isso será aberta a seguinte caixa de diálogo a seguir :



Instalação do Android Studio

Por padrão as opções “Android Virtual Device” e “Performance (Intel ® HAXM)” estão selecionadas. Vamos **desmarcar** todas as duas opções, conforme mostra a figura seguinte :

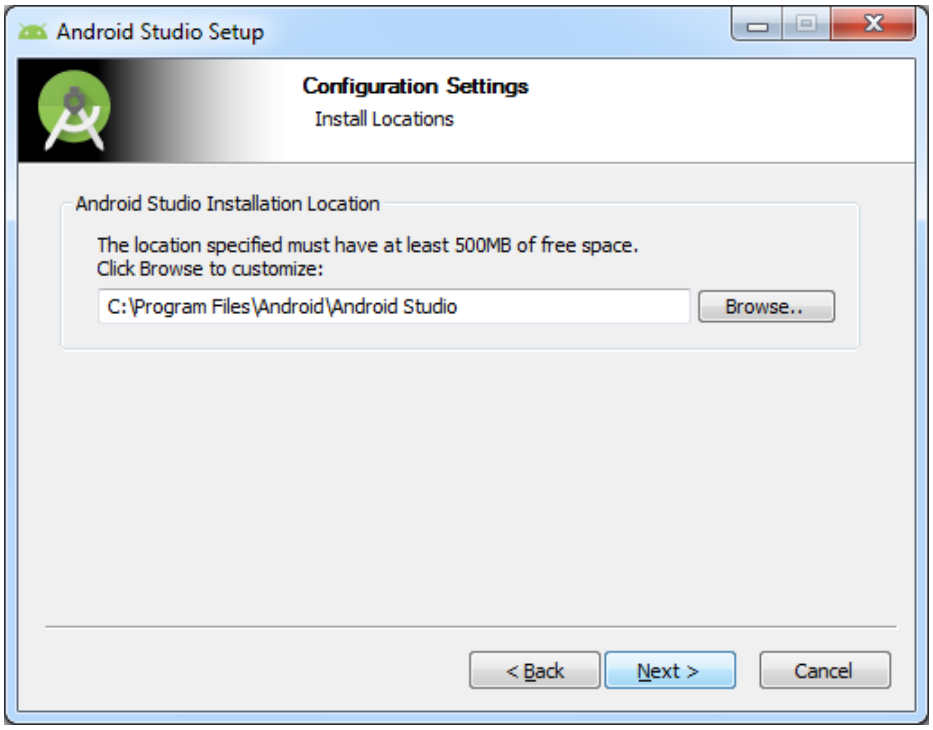


Desmarcando as duas opções acima

Sobre a opção “Performance (Intel ® HAXM)”

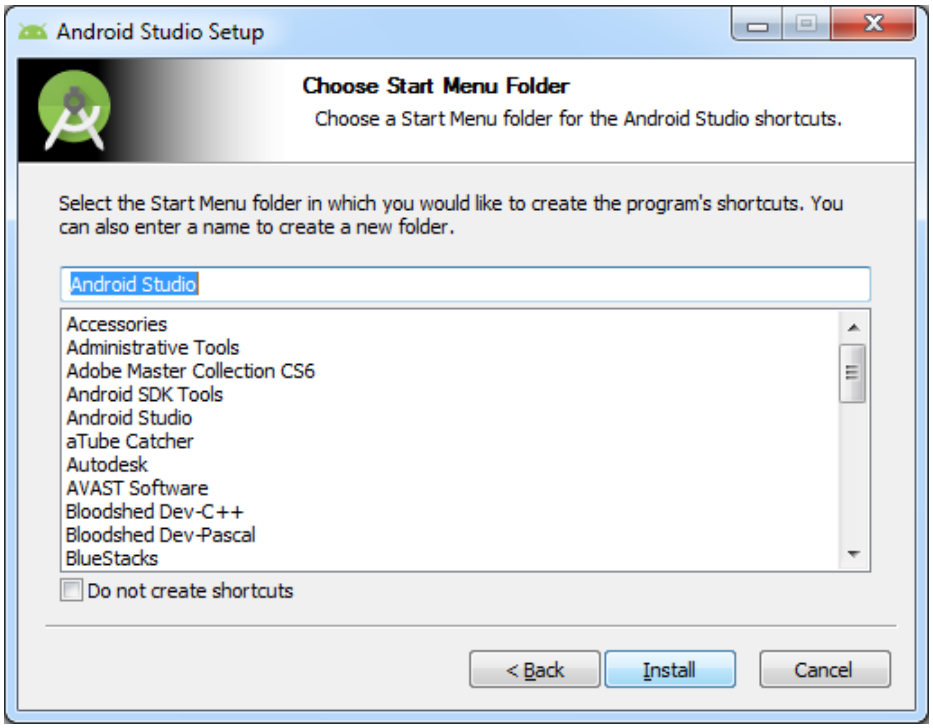
Possivelmente essa opção **NÃO DEVE TER APARECIDO** em seu computador. Calma, está tudo bem. Essa opção só é exibida dependendo da configuração de seu PC. Essa opção permite o recurso de simulação de “aceleração de hardware” (suportado por computadores com uma configuração mais “pesada”), porém, ela não será utilizada neste material.

Depois de seguir as orientações acima, vamos clicar em “Next”. Feito isso será aberta a seguinte caixa de diálogo a seguir :



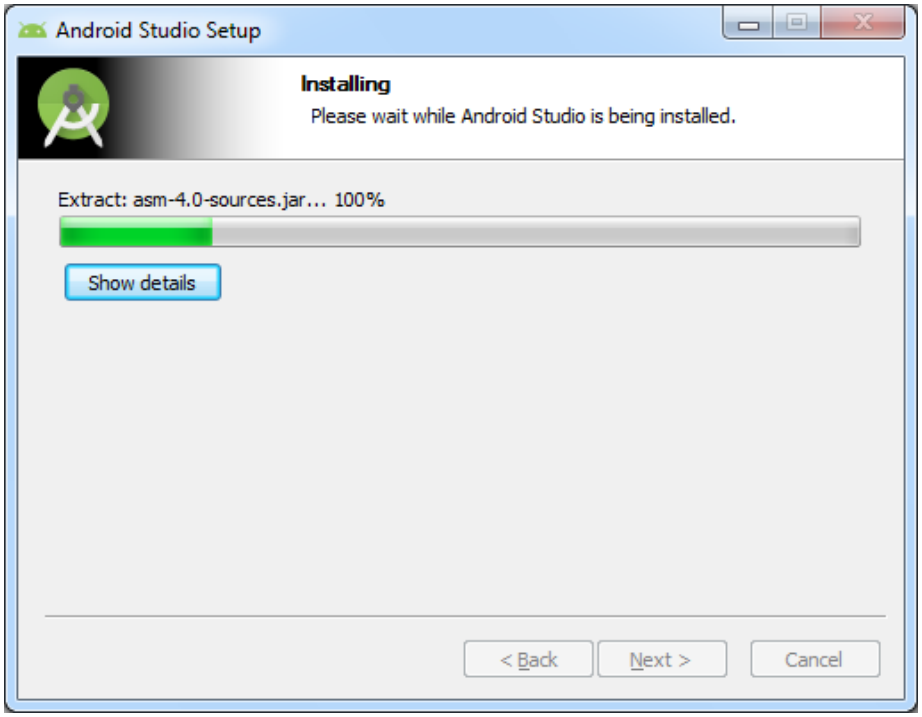
Instalação do Android Studio

Vamos deixar o diretório “default’ de instalação do Android Studio. Clique em “Next” para irmos para a próxima etapa, conforme mostra a figura :



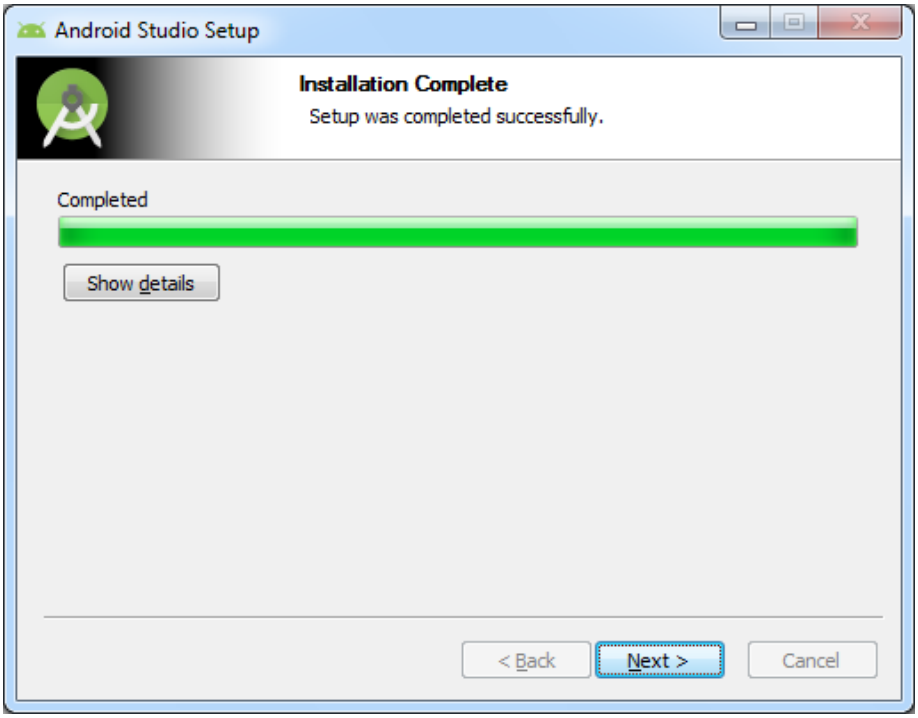
Instalação do Android Studio

Na tela da figura anterior, basta clicarmos no botão “Install” para instalarmos o Android Studio em nosso computador. Vejamos o resultado na figura seguinte :



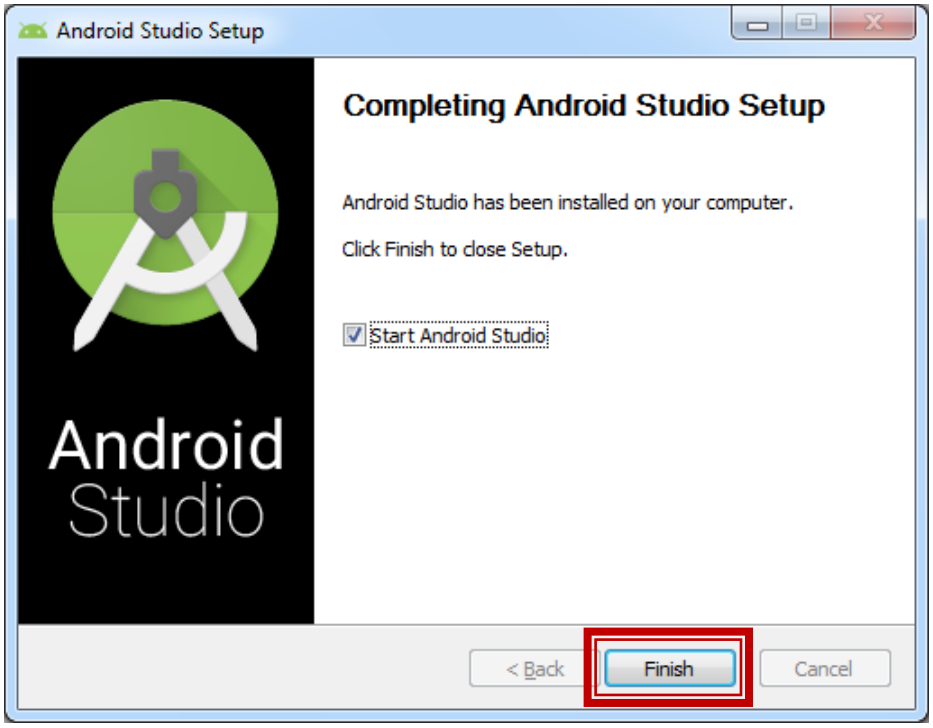
Instalando o Android Studio

Aguarde a instalação se concluída, conforme podemos ver abaixo:



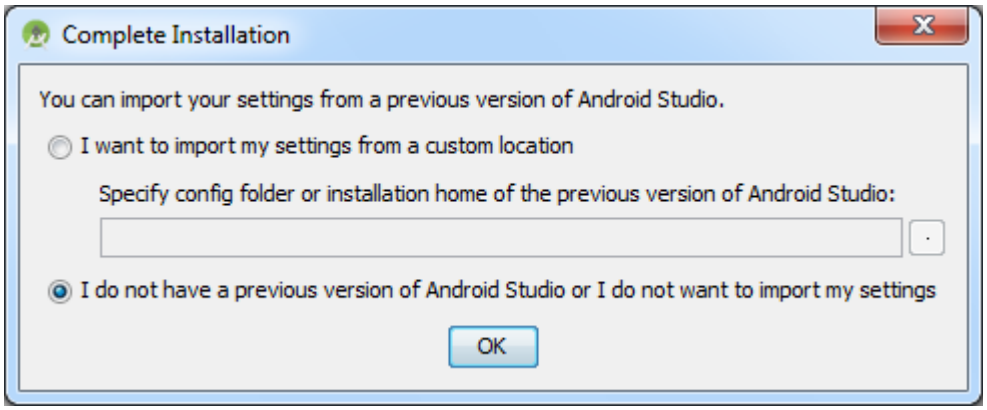
Instalação concluída

Clique em “Next” e na próxima tela vamos clicar em “Finish” para iniciarmos a execução do “Android Studio”, conforme podemos ver na figura seguinte :



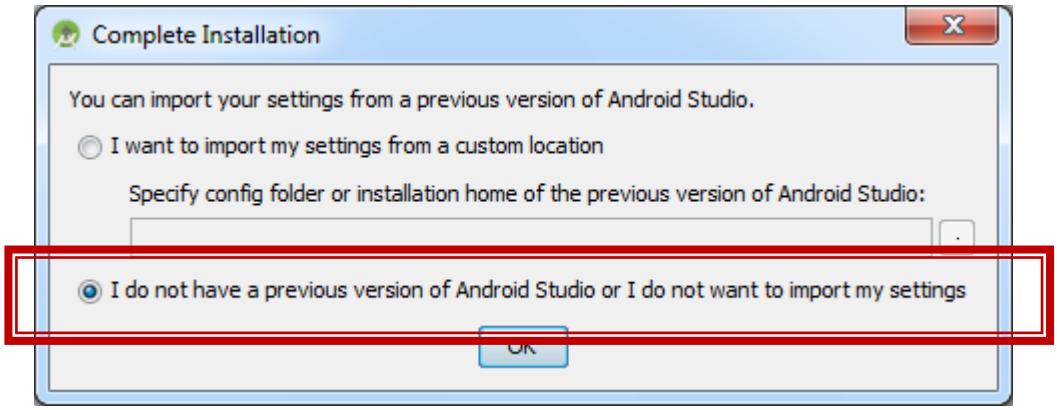
Instalação do Android Studio concluída

Após clicarmos no botão “Finish”, será aberta a seguinte caixa de diálogo abaixo, do Android Studio :



Caixa de diálogo – Complete Installation

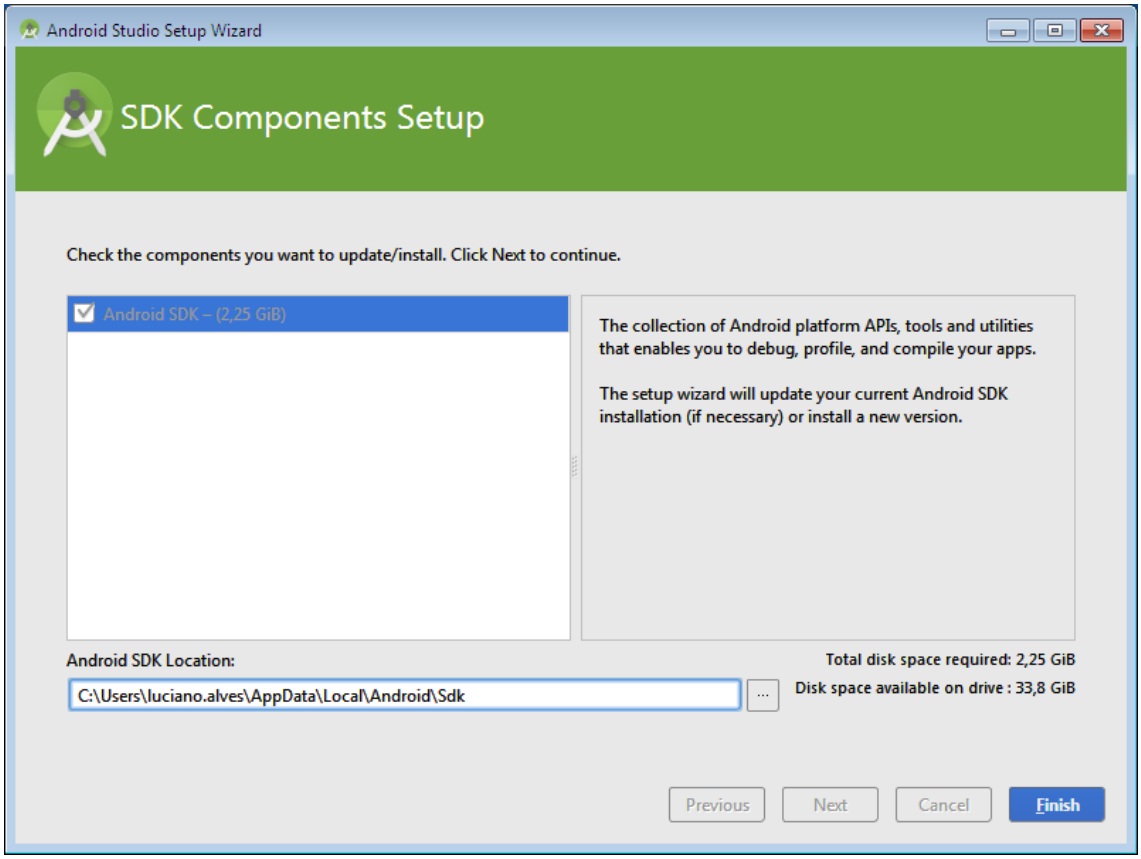
Nessa caixa de diálgo é perguntado se você gostaria de importar os projetos desenvolvidos em uma versão anterior do Android Studio. Supondo que estou instalando o Android Studio pela primeira vez, vamos deixar marcada a segunda opção , conforme mostra a figura abaixo :



Deixando marcada a segunda opção

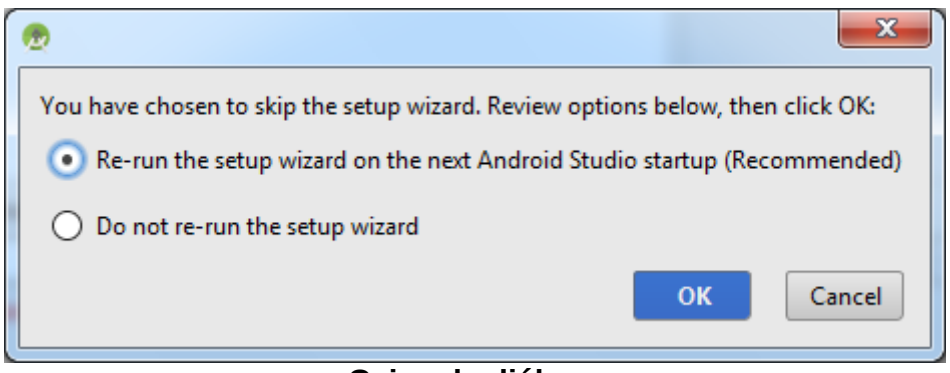
Vamos clicar no botão “OK” para que possamos inicializar o Android Studio, conforme podemos ver na figura seguinte :

Logo em seguida deverá ser mostrada a seguinte caixa de diálogo a seguir, solicitando o download do Android SDK através do Android Studio :



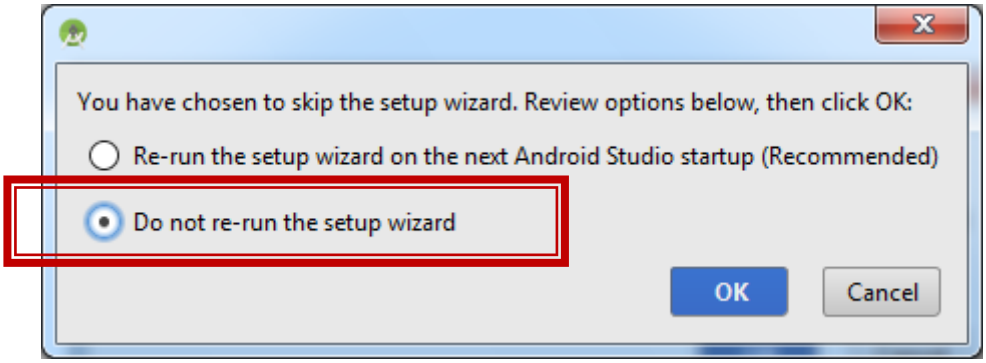
Caixa de diálogo – Android Studio Wizard

Bom, como já instalamos o Android SDK, não será preciso realizar nenhum download de SDK. Logo, vamos clicar no botão “Fechar” da caixa de diálogo e em seguida será mostrada a seguinte mensagem :



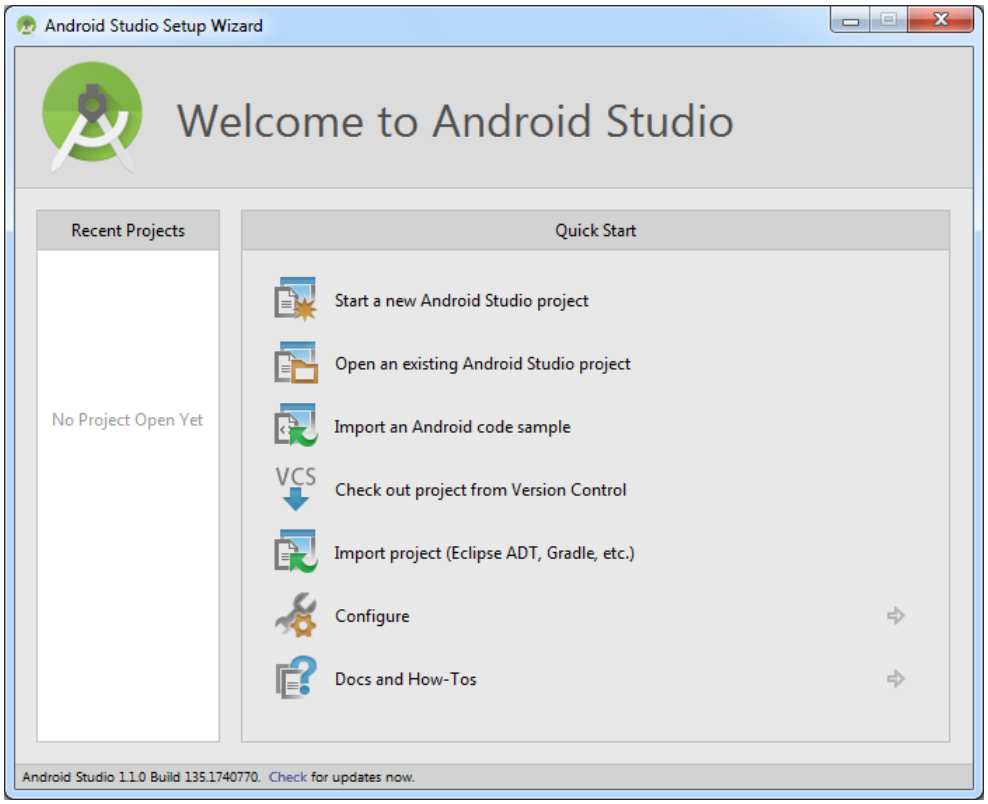
Caixa de diálogo

Na caixa de diálogo que é exibida vamos desmarcar a primeira opção para marcamos a opção de baixo “Do not re-run the setup wizard”.



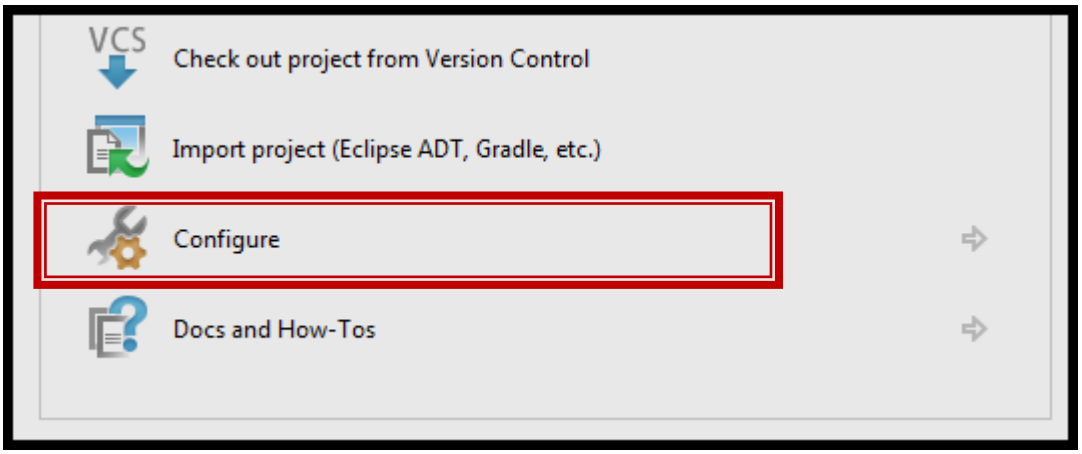
Caixa de diálogo

Feito isso vamos clicar em “OK”. Em seguida será exibida a seguinte caixa de diálogo a seguir :



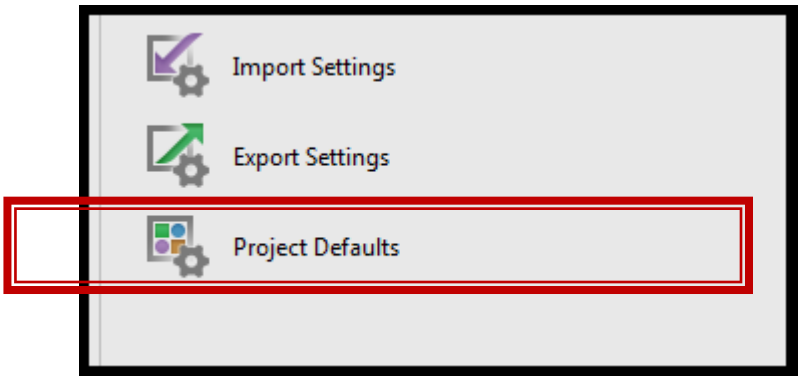
Caixa de diálogo – Android Setup Wizard

A primeira coisa que iremos fazer agora, antes de criarmos qualquer projeto, é configurar o Android SDK para atuar em conjunto com o Android Studio. Para isso, vamos clicar na opção “Configure”, conforme podemos ver abaixo :



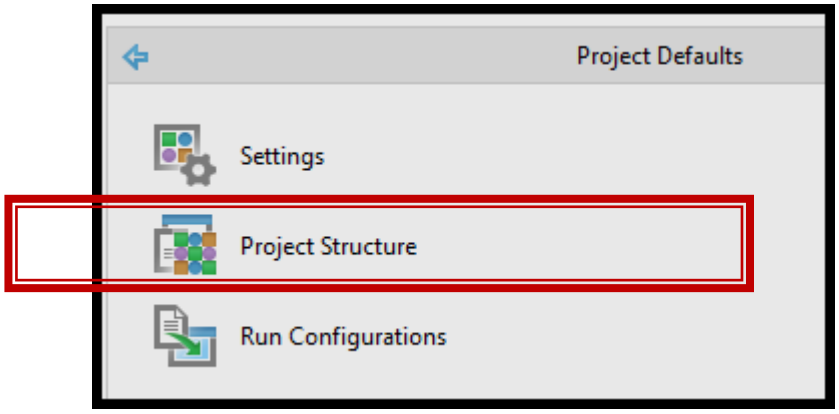
Selecionando a opção “Configure”

Na próxima caixa de diálogo que surge selecione a opção “Project Defaults” :



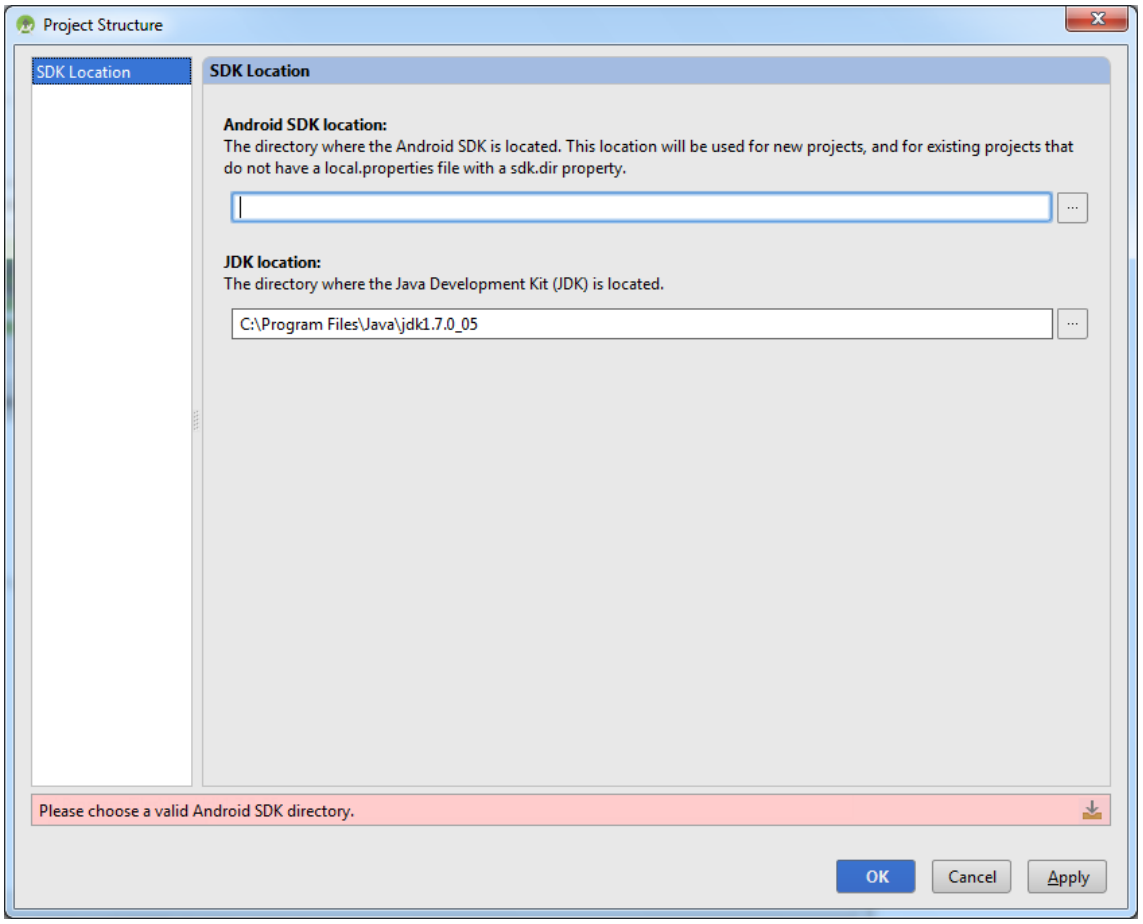
Selecionando a opção “Project Defaults”

Feito isso, vamos selecionar agora a opção “Project Structure” :



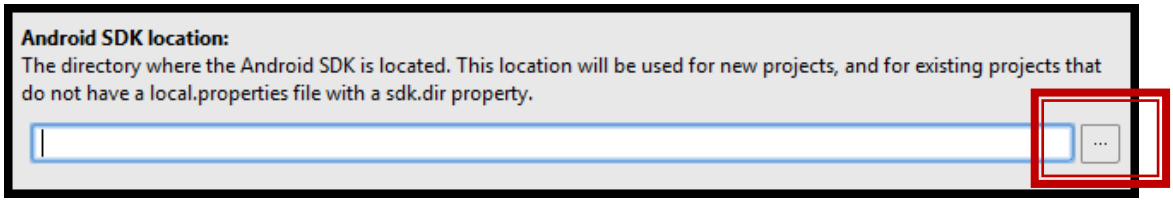
Selecionando a opção “Project Structure”

Feito isso irá se abrir a seguinte caixa de diálgo a seguir :



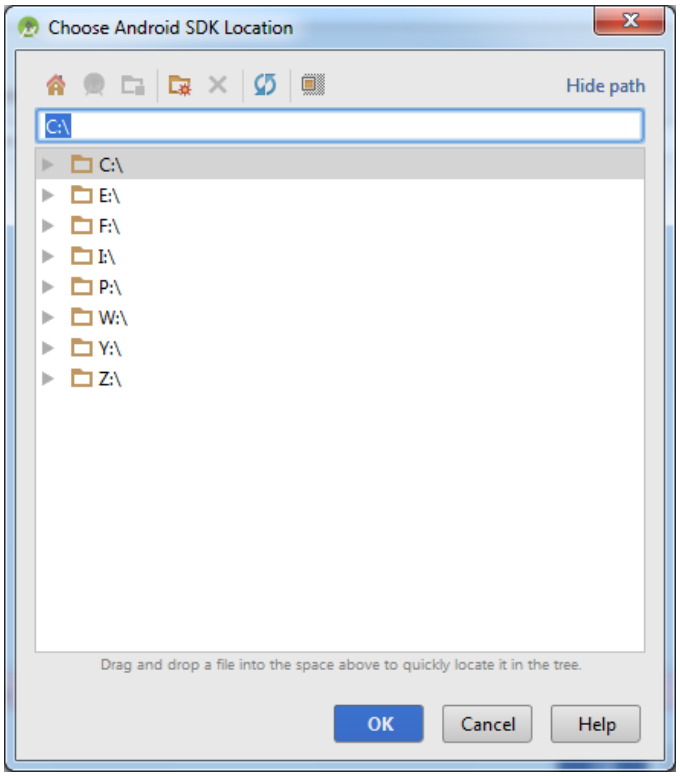
Caixa de diálogo – Project Structure

Nesta tela iremos definir o diretório onde nós instalamos o nosso Android SDK. Para isso vamos clicar no botão “Browser” (...), conforme é indicado na figura a seguir :



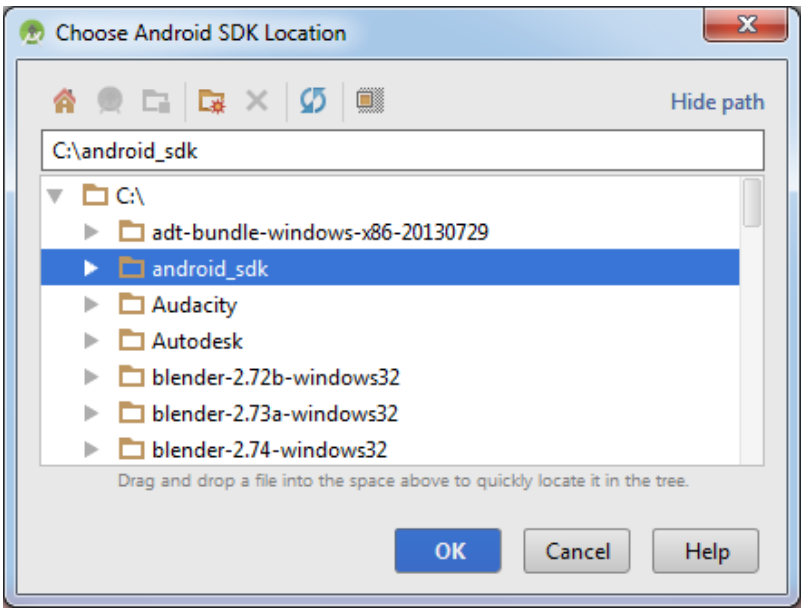
Selecionando o botão “Browser”

Feito isso será aberta a seguinte caixa de diálogo a seguir :



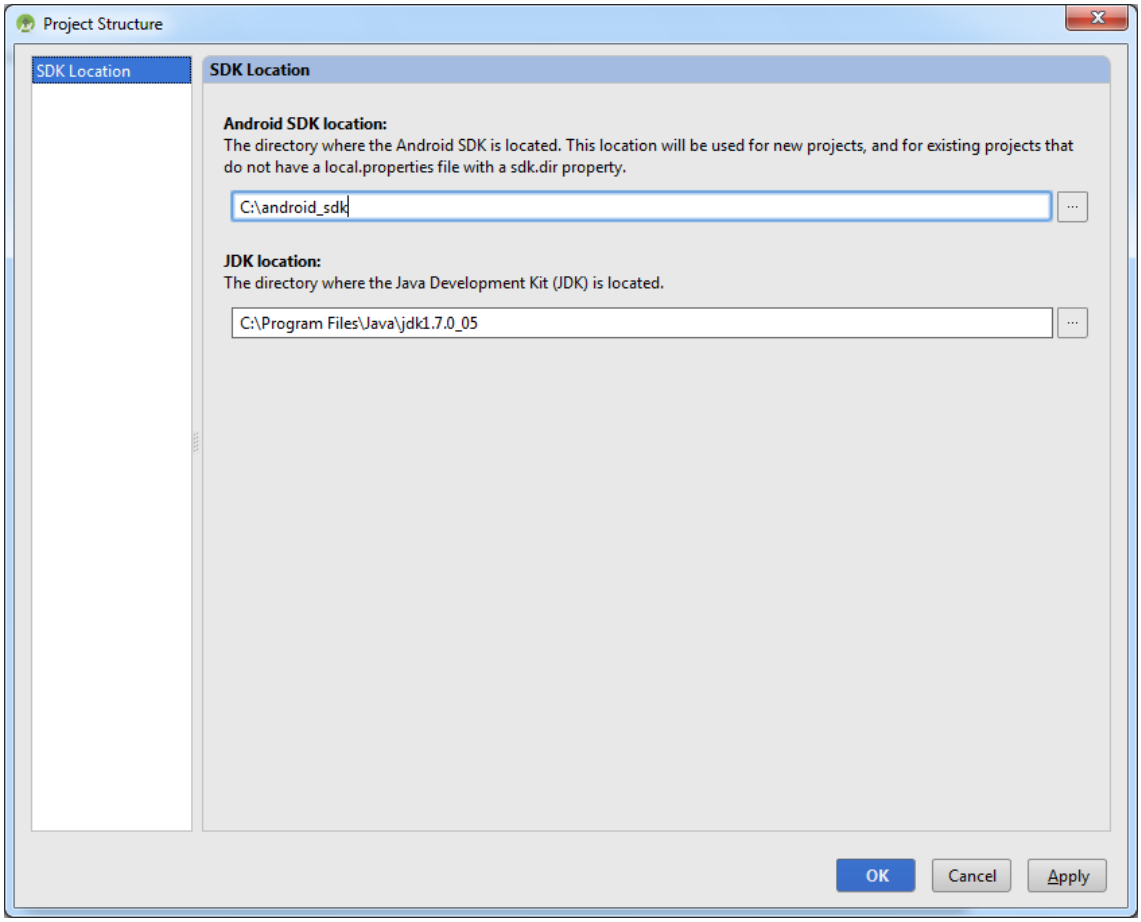
Caixa de diálogo – Choose Android SDK Location

Se lembra que instalamos o nosso “Android SDK” separadamente no diretório “c:\android-sdk” ? Pois bem, vamos apontar para o diretório de instalação do “Android SDK”, conforme podemos ver na figura seguinte :



Apontando para o diretório do Android SDK

Depois de apontar para o diretório do Android SDK, podemos clicar no botão “OK” para finalizarmos o processo. Veja o resultado :



Diretório do SDK definido

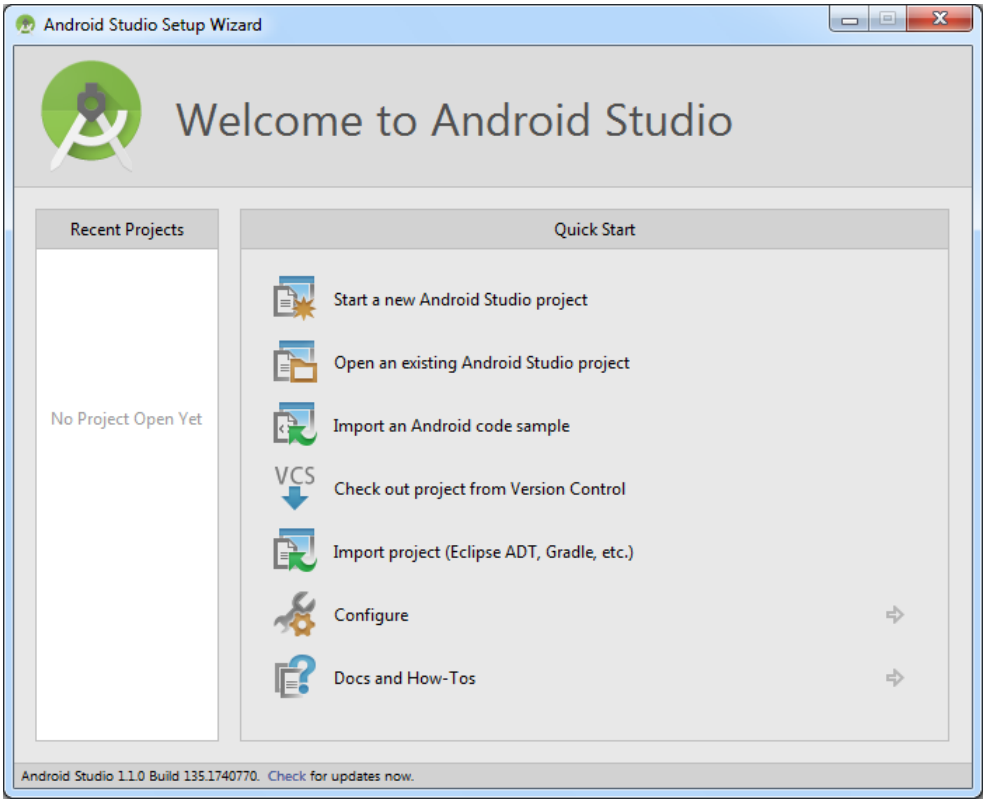
Na caixa de diálogo da figura acima podemos clicar no botão “OK” para confirmarmos o diretório do Android SDK.

Depois disso, feche o Android Studio.

Capitulo 3 Começando a programar no Android

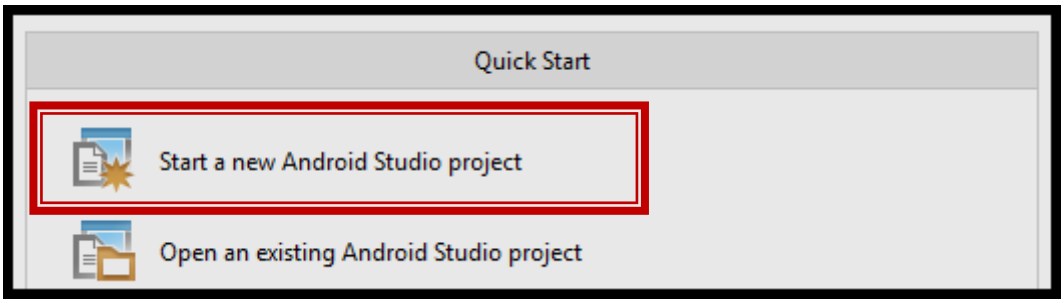
Depois de tudo configurado e instalado, conforme foi mostrado no capítulo anterior, vamos a partir de agora começar a desenvolver nossas aplicações para a plataforma Android usando o Android Studio.

Vamos primeiramente abrir e executar o nosso Android Studio . Quando a ferramenta é executada pela primeira vez e nenhum projeto anterior é criado nele, é exibida a seguinte tela abaixo:



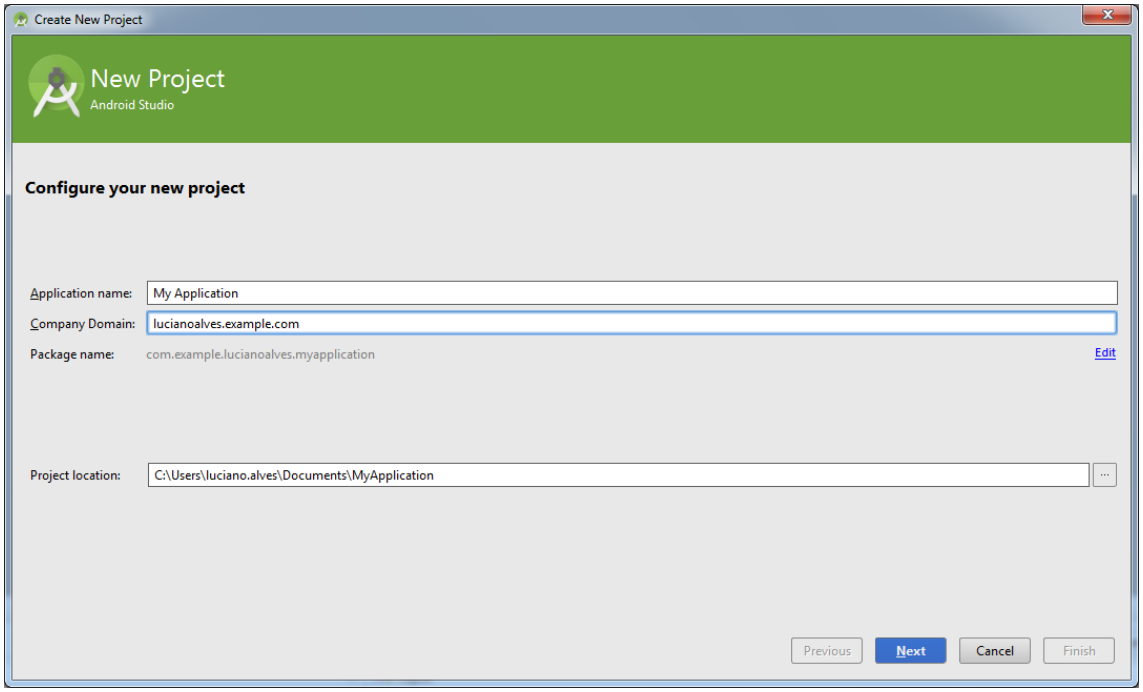
Caixa de diálogo – Quick Start

Para criarmos um novo projeto no Android Studio basta clicar no botão “Start a new Android Studio project”, conforme indicado na figura em seguida:



Criando um novo projeto

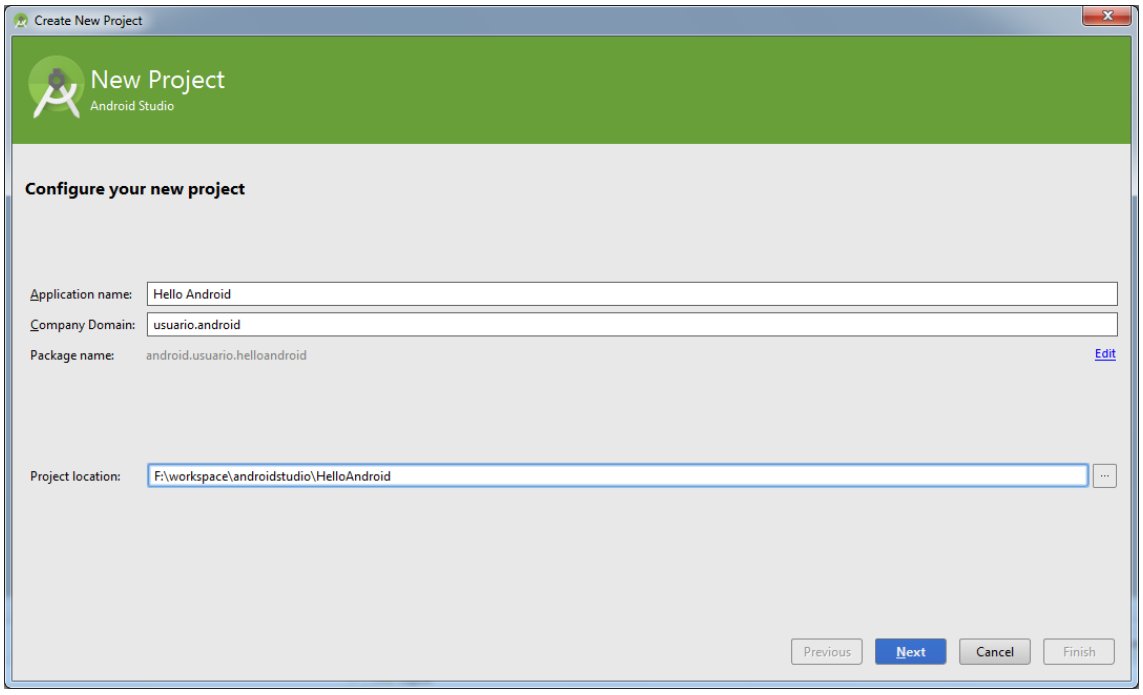
Feito isso será aberta a seguinte caixa de diálogo , conforme mostra a próxima figura:



Criando um novo projeto no Android Studio

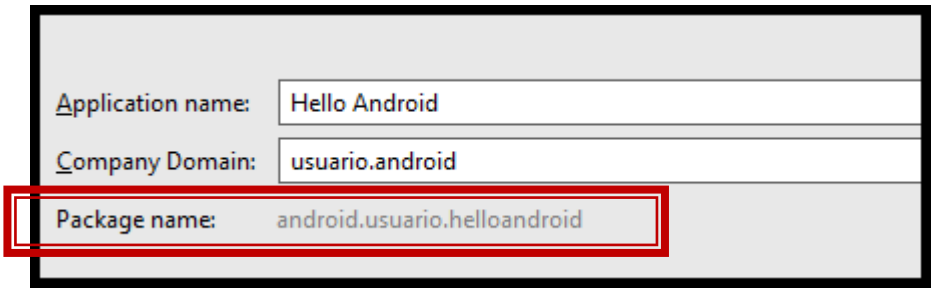
Na caixa de diálogo acima informarmos o nome da nossa aplicação (Application Name), o nome da empresa (Company Domain) e o local onde o mesmo será salvo (Project location).

No campo “Application Name” vamos definir o nome da nossa aplicação, que irá se chamar “HelloWorld” (sem aspas, é claro). No campo “Company Domain” vamos digitar “usuario.android” (expresso como nome de pacote Java). Agora em “Project location” , que indica onde o projeto será criado, fica por sua escolha, PORÉM, o diretório (caminho) que você escolher não deverá conter espaços em branco, fica a dica . Vejamos como ficou na figura abaixo :



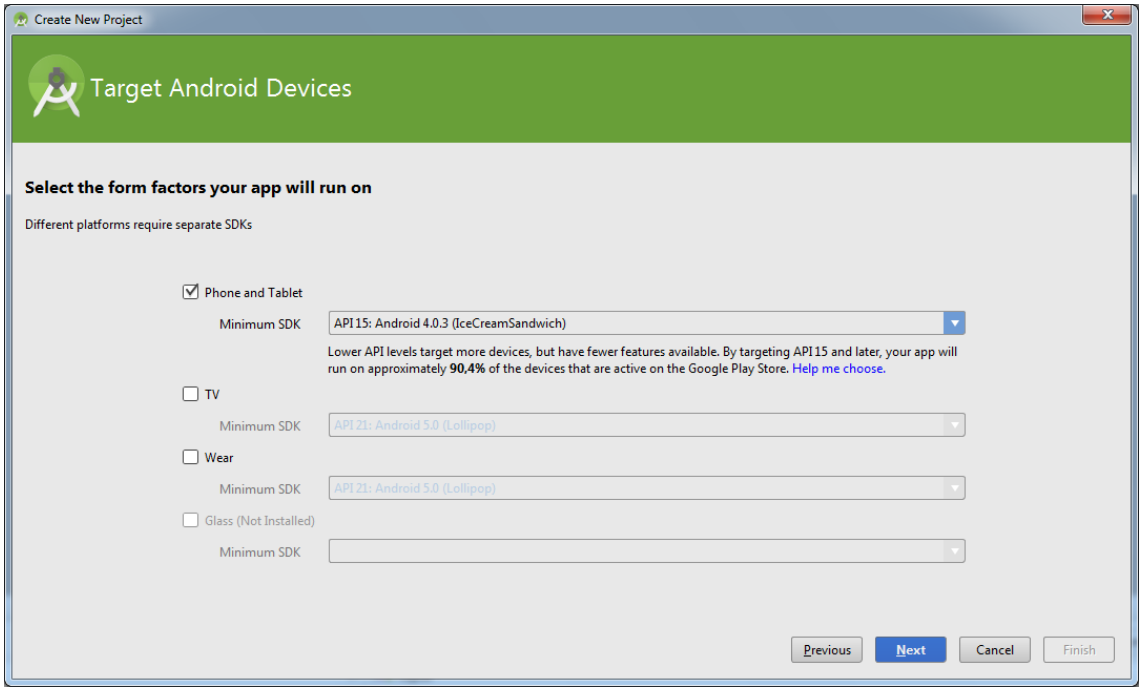
Criando um novo projeto no Android Studio

Observe que um dos campos, que ainda não citei acima , é preenchido automaticamente (o campo “Package name”). Ele é formado pelo nome da aplicação “concatenado” com o nome do campo “Company Domain”. Vejamos abaixo :



O campo “Package name”

Depois de preencher as informações acima necessárias, clique no botão “Next” para continuarmos. Feito isso será exibida a seguinte tela abaixo:

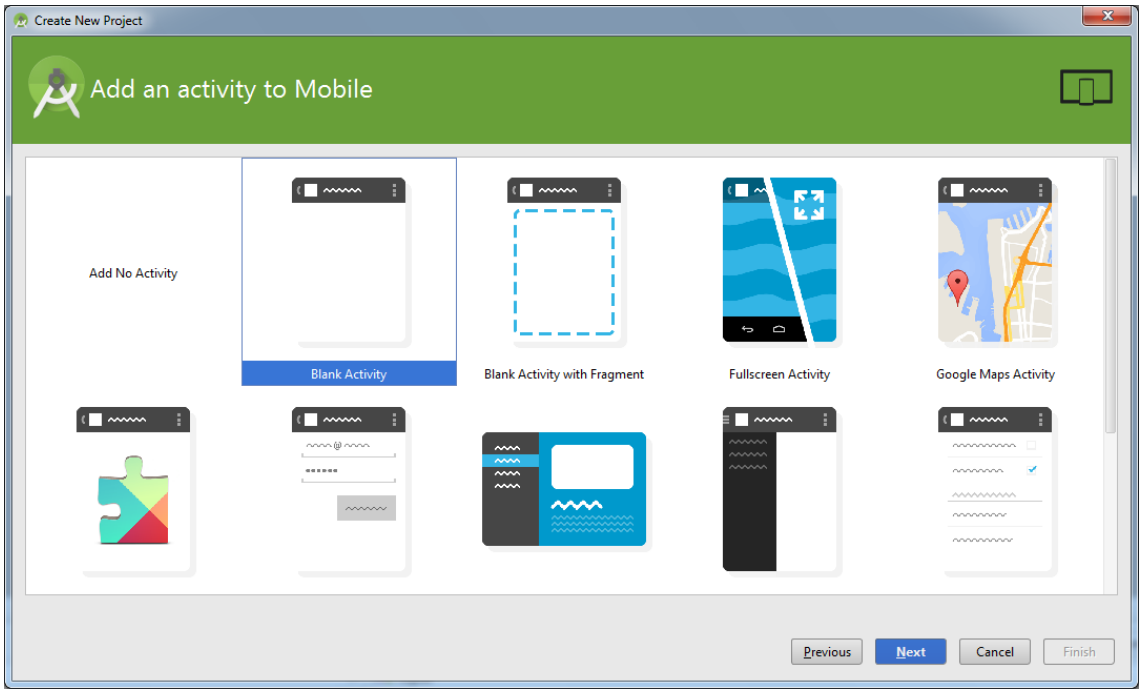


Criando um novo projeto no Android Studio

No formulário acima especificamos para quais dispositivos iremos desenvolver as nossas aplicações. Nesta apostila estaremos desenvolvendo aplicações para Smartphones e Tablets Android em geral, logo, deixaremos a opção “Phone and Tablets” marcada (as opções de TV e Android Wear só é destinada para sistemas Android cuja versão seja 5.x, o Lollipop).

Observe que em “Mininum SDK” é especificada a versão mínima do Android instalada para desenvolvimento das aplicações, que neste caso é a versão 4.0.3 que instalamos através do Android SDK. Deixaremos esta opção selecionada (até porque também só existe essa opção).

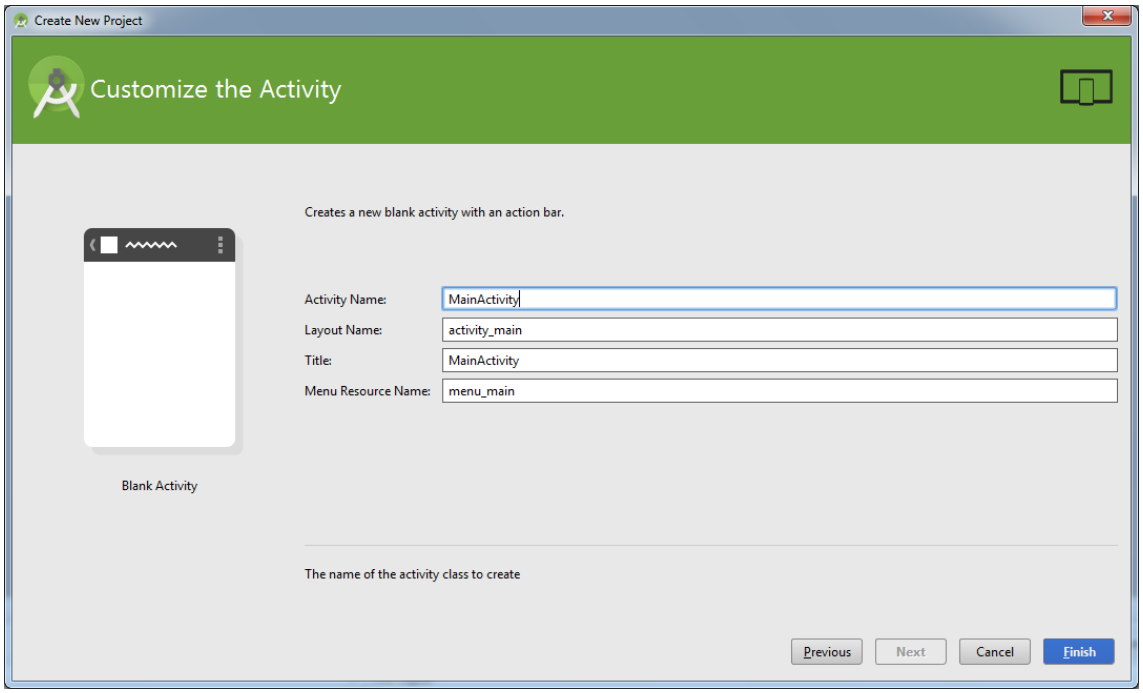
Na próxima tela (clicando em “Next”) é aberto o seguinte formulário abaixo :



Criando uma nova Activity

Toda classe (arquivos ".java") na plataforma Android, que representa uma aplicação, é considerada uma "atividade" (Activity). Para isso, precisamos definir uma atividade que vai representar a nossa aplicação Android. Para isso vamos criar uma nova atividade "em branco" (Blank Activity), conforme a opção selecionada na figura acima.

Com a opção "Blank Activity" selecionada, vamos clicar no botão "Next" e surgirá a seguinte tela abaixo :



Informações da Activity

Na seção acima definimos o nome da nossa atividade (Activity Name) assim como o nome do arquivo XML que vai representar a tela da nossa aplicação (Layout Name), o título que será exibido na aplicação (Title) e o nome do arquivo que irá gerenciar os itens de menus a serem exibidos na aplicação (Menu Resource Name). Por padrão o nome da “Activity” é “MainActivity”, e o nome do arquivo de layout é “activity_main”. Vamos alterar as informações conforme os dados abaixo :

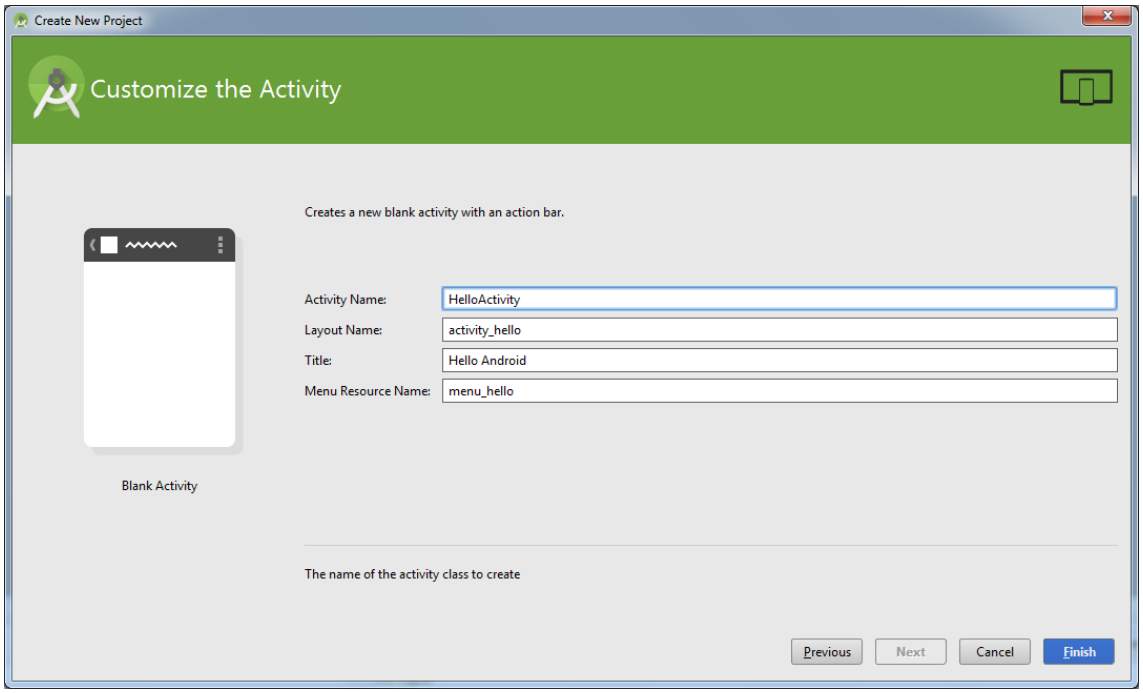
Activity Name : HelloActivity

Layout Name : activity_hello

Title : Hello Android

Menu Resource Name : menu_hello

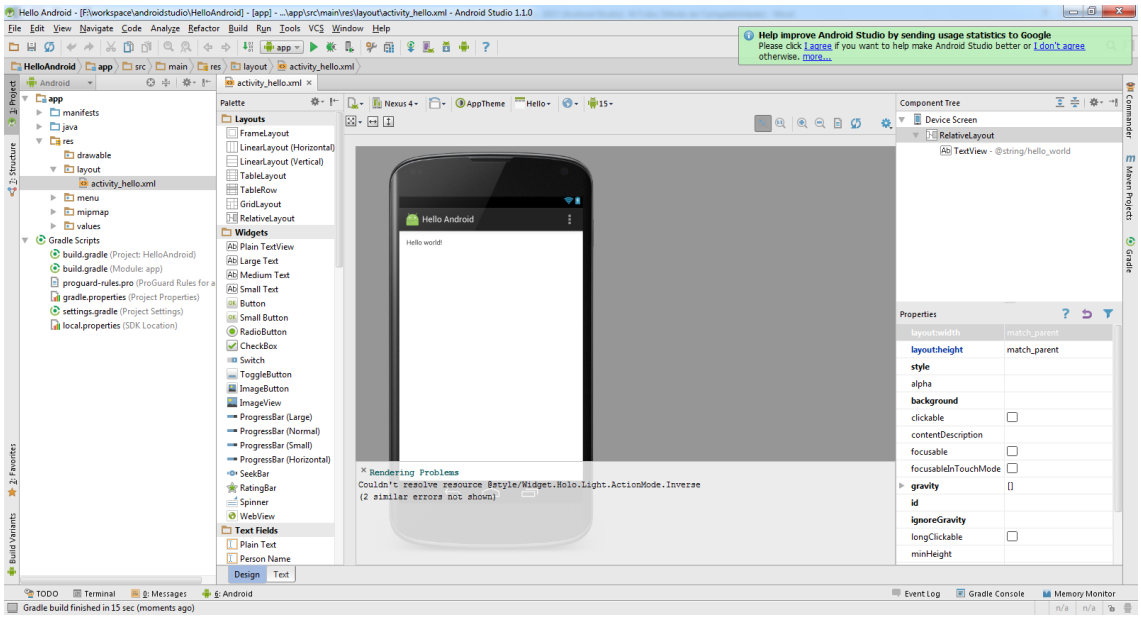
Vamos conferir como ficou na figura seguinte :



Informações da Activity

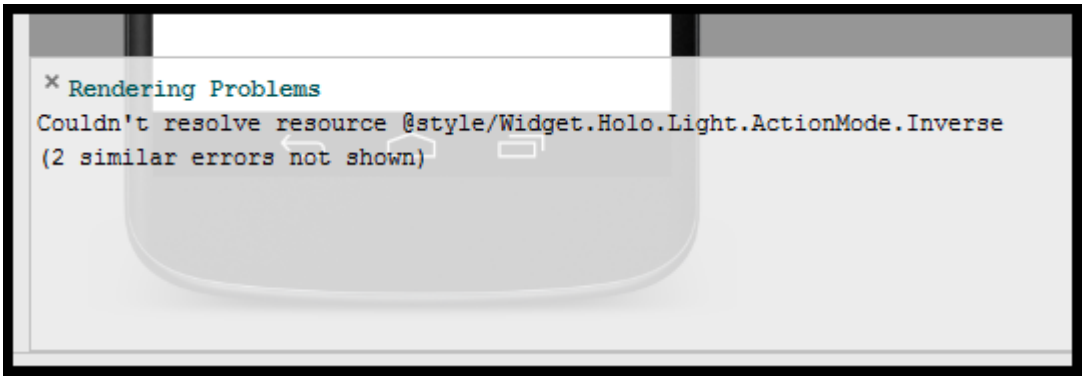
Por padrão, como na primeira seção, quando mudamos o conteúdo do campo “Activity Name”, o campo “Layout Name” é alterado automaticamente. Caso deseje você poderá escolher outros nomes para o campo alterado “automaticamente” sem afetar o nome da atividade.

Depois disso, clique em “Finish” para nosso projeto possa ser gerado. O resultado você confere na figura seguinte:



Projeto criado

Se por caso for exibido a seguinte mensagem abaixo :

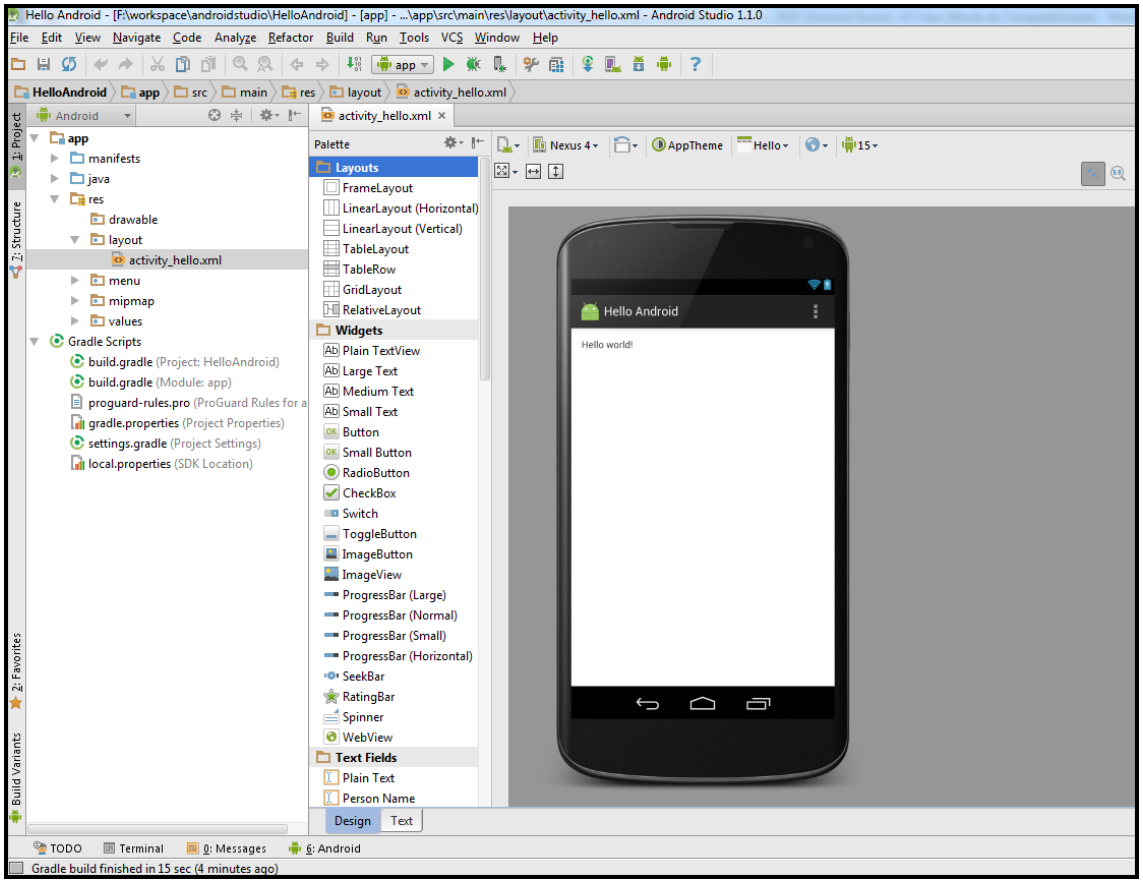


Mensaegem de erro

Feche esta caixa de diálogo e prossiga para a próxima etapa.

Conhecendo a estrutura geral de um projeto no Android Studio

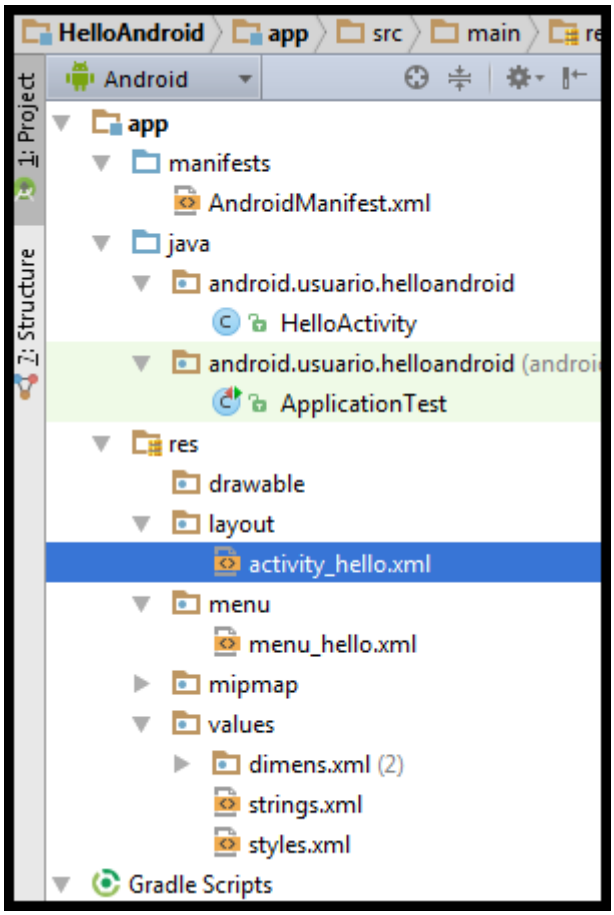
A primeira coisa que visualizamos que nos chama atenção é a tela do dispositivo com a frase “Hello world”, conforme podemos conferir na figura seguinte:



Visualização da tela do dispositivo

Se observamos a figura anterior, ao lado da tela do dispositivo temos uma paleta de componentes disponíveis que podemos utilizar para construir as nossas aplicações. Explicarei mais em detalhes no próximo capítulo.

Se visualizarmos o lado esquerdo, existe uma seção chamada “Project”, onde nela existe uma pasta chamada “app” (que é o nosso projeto), constituído por vários subdiretórios , que por sua vez, possui seus respectivos arquivos, conforme demonstra a figura seguinte:



Visualização dos diretórios do projeto

Irei comentar agora toda a estrutura de um projeto Android. Ele é composto por várias pastas e arquivos, cada um com uma finalidade em especifico.

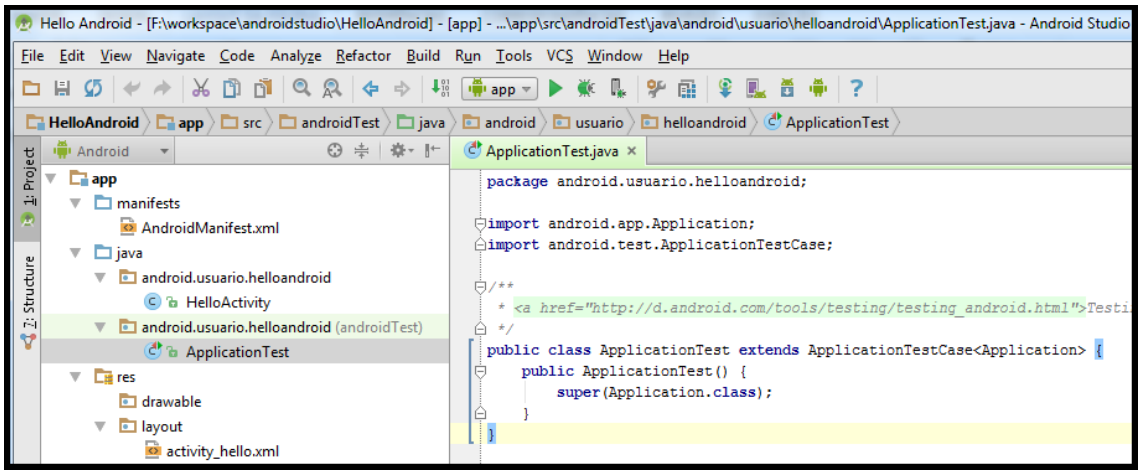
O diretório “app” (application)

Vamos começar conhecendo primeiramente o diretório “app” (application). Dentro desse diretório temos todos os arquivos principais de nossa aplicação, distribuído pelos seus diretórios. Ao expandirmos o diretório “app” temos as pastas “manifest” e “java”. Dentro da pasta “manifest” temos o arquivo “AndroidManifest.xml”, com o seguinte código abaixo :


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="android.usuario.helloandroid" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".HelloActivity"
            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>
```

Esse arquivo é considerado o “sistema nervoso” do Android, é através dele que definimos todas as permissões e configurações primordiais de nossa aplicação para que a mesma seja executada (não iremos fazer nenhuma modificação nesse arquivo).

Dentro da pasta “java” temos as nossas classes (arquivos “.java”) que fazem parte da nossa aplicação. Observe que dentro da nossa pasta existem dois pacotes de mesmo nome (“android.usuario.helloandroid”), sendo o último voltado para testes, onde dentro dele existe uma classe chamada “AndroidTest.java” , com o seguinte conteúdo abaixo :



Visão geral do arquivo “ApplicationTest.java”

No primeiro pacote existe um arquivo chamado “HelloActivity.java” , que é a classe principal do programa, que é através dela que definimos todos os ações da nossa aplicação :

Visão geral do arquivo “HelloActivity.java”

O arquivo aberto acima (HelloActivity.java) possui o seguinte conteúdo:

```
package android.usuario.helloandroid;

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


public class HelloActivity extends Activity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
        //present.
        getMenuInflater().inflate(R.menu.menu_hello, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
```


```
        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Diferentemente das aplicações comuns em Java (J2SE), toda classe que representa uma aplicação Android deve ser derivada da classe **Activity** (atividade), conforme você pode ver através do seguinte trecho de código:

```
public class HelloActivity extends Activity {
    :
```

A classe “HelloActivity” possui como método principal o **onCreate**. A primeira linha de código presente neste método faz chamada ao método **onCreate** da classe base (a classe **Activity**), conforme podemos conferir abaixo:

```
super.onCreate(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 “activity_hello.xml”.

O diretório “res” (resources)

Dentro da estrutura de um projeto Android existe um diretório chamado “res” (resources), onde dentro dele colocamos todos os recursos que podemos utilizar em um programa Android (como imagens, sons, músicas e etc). Vamos conhecer os subdiretórios existentes dentro dessa pasta e a finalidade de cada um deles.

O diretório “drawable”

O diretório “drawable” presente dentro da pasta “res” do nosso projeto possui uma única finalidade : armazenar imagens que serão visualizadas dentro de uma aplicação Android. Uma coisa que precisa ser reforçada aqui é que os arquivos de imagens presentes nessa pasta não podem estar escritos de qualquer forma. Os arquivos a serem copiados dentro desse diretório devem possuir somente letras minúsculas (de “a” até “z”), números (de “0” a “9”) e underline (“_”).

O diretório “layout”

O diretório “layout” armazena todos os arquivos referentes as telas de uma aplicação Android, que normalmente são arquivos “.xml”. Para aqueles que conhecem a programação de HTML com JavaScript para a construção de páginas Web, no Android é similar, é a combinação de Java com XML para a construção de aplicativos Mobile.

O diretório “values”

Um dos recursos que o Android permite que usemos na construção das telas de nossa aplicação são “constantes”. Como bom programador que todos nós somos, sabemos que uma “constante” nada mais é do que um endereço de memória que vai armazenar um determinador valor, que será único até o fim da execução do programa. Mas, o que isso tem a ver com as telas da nossa aplicação ? Vou explicar.

Normalmente usamos constantes para armazenar valores fixos que, naturalmente, estarão presentes na maior parte do seu programa. É muito comum criarmos um título como nome de uma aplicação (como por exemplo : “Alpha Software”), que vai estar presente na maioria das telas (ou em todas) da sua aplicação. Imagine você digitar “manualmente” o nome de seu software em todas as telas da sua aplicação, seria um processo trabalhoso você não acha ? Agora imagine “futuramente” que você precise “mudar” o nome de sua aplicação ? Você terá novamente o mesmo trabalho para digitar em cada tela o nome da nova aplicação, sem dúvida. Usando constantes, você não terá esse trabalho.

Dentro dessa pasta temos os seguintes arquivos:

- “strings.xml” : Esse arquivo guarda constantes relacionadas à nossa aplicação em geral (como o nome da aplicação, o título que vai aparecer na aplicação e etc.).

- “dimen.xml” : Esse arquivo guarda constantes relacionadas as dimensões que podemos utilizar em nossa aplicação.

- “styles.xml” : Esse arquivo guarda constantes relacionadas à estilos que podemos utilizar em nossa aplicação.

O diretório “mipmap”

O diretório “mipmap” possui a mesma características do diretório “drawable” (armazenar imagens) , porém, o mesmo foi destinado a armazenar nesta pasta somente imagens referentes ao ícone da nossa aplicação Android, que possa se comportar em várias resoluções de tela.

O diretório “menu”

Nesse diretório é armazenado o arquivo responsável por gerenciar e exibir os itens de menu em noss aplicação. Por padrão o arquivo que contem os itens de menu possui a seguinte estrutura abaixo :

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".HelloActivity">
    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100" android:showAsAction="never" />
</menu>
```

Os itens de menu podem ser adicionados através do uso das tags específicas do XML para esse propósito (veremos isso mais adiante).

Visão geral da ferramenta de desenvolvimento

Acabamos de entender e conhecer acima toda a estrutura de diretórios de um projeto Android no Android Studio. Agora vamos dar uma visão geral na nossa ferramenta de desenvolvimento (IDE) para a plataforma Android.

Visão geral da ferramenta de desenvolvimento

Vamos entender as numerações indicadas na figura anterior :

Na indicação (1) temos a paleta de componentes onde os mesmos estão separados por seções. As seções são : Widgets , Text Fields, Layouts , Containers, Date e Time , Expert e Custom (Componentes personalizados, normalmente criados pelo usuário e entre outros recursos).

Na indicação (2) temos um preview de como ficará a interface da nossa aplicação quando a mesma for executada, nessa interface podemos arrastar e soltar os componentes (indicados por (1)) , construindo assim a nossa aplicação.

Na indicação (3) temos as propriedades do componente (Properties). Quando selecionamos um componente, as suas propriedades são visualizadas e assim, conforme nossa necessidade, podemos alterá-las.

Na indicação (4) temos uma seção chamada “Component Tree”. Nessa seção podemos ver , de forma hierárquica, todos os componentes que estão presentes na tela da nossa aplicação

Na indicação (5) podemos alternar entre o modo gráfico (“Design”, onde temos a visualização da tela da aplicação) e o modo XML (“Text”, onde visualizamos o

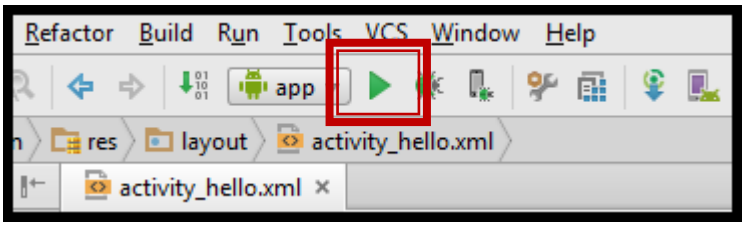
código XML do arquivo, que corresponde a estrutura que forma a tela de nossa aplicação).

Na indicação (6) podemos alterar os temas que podemos utilizar em nossa aplicação Android, orientação da tela do dispositivo (retrato ou paisagem), resolução da tela do dispositivo e etc. A maioria das opções disponíveis nessa seção só terá efeito em tempo de projeto. É necessário configurar as mesmas opções para que elas aconteçam em tempo de execução (quando você executa o emulador ou dispositivo real), de acordo com a necessidade.

Executando a nossa aplicação

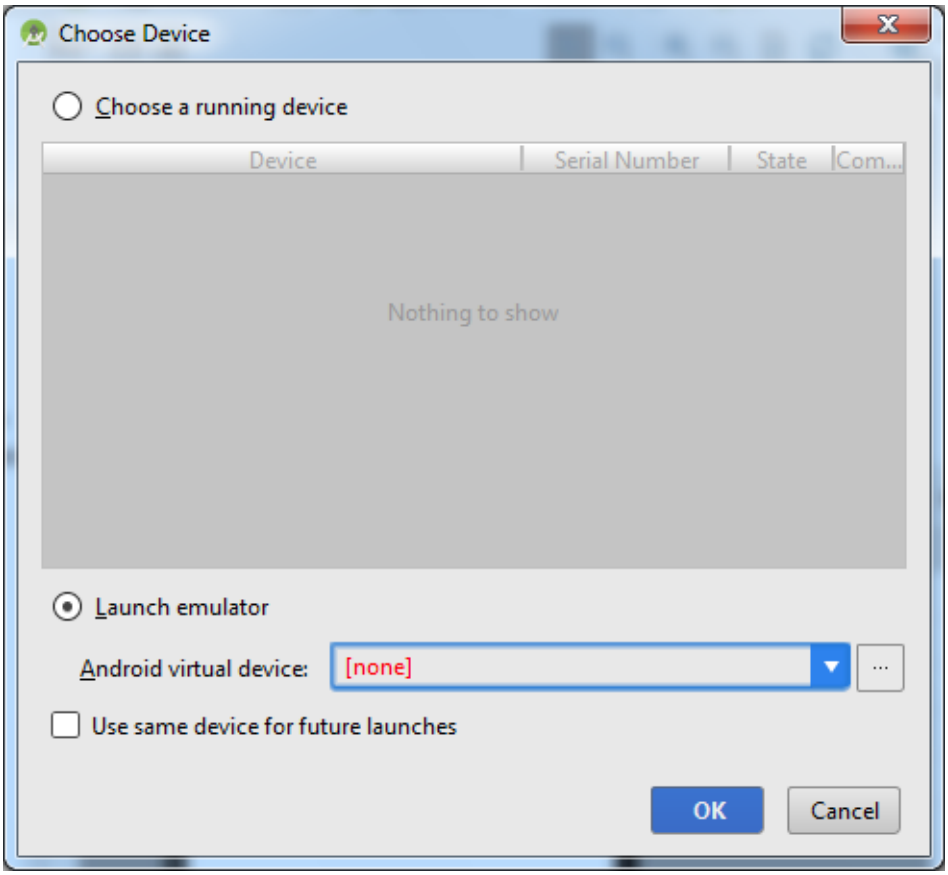
Agora que já tivemos uma visão geral da ferramenta de desenvolvimento e do projeto em Android (com toda a sua estrutura de diretórios e arquivos), podemos a partir de agora executarmos a nossa aplicação. Mas antes de executarmos a nossa aplicação, irei fazer alguns comentários. Observe que quando criamos um projeto em Android, na tela do dispositivo já tem um componente **TextView** (situado na paleta de componentes , dentro da seção "Widgets") onde nele é exibida a frase "Hello world". Quando executarmos a nossa aplicação através do emulador, ela sairá idêntica como está sendo exibida no projeto (uma tela "em branco" com a frase "Hello world").

Como fazer para executarmos a nossa aplicação ? Para executarmos a nossa aplicação basta irmos no menu "Run" / "Run 'app' " (ou pressionar as teclas SHIFT+F10) . Uma outra forma também é clicando no botão pressente na barra de ferramentas :



Botão "Run" do Android Studio

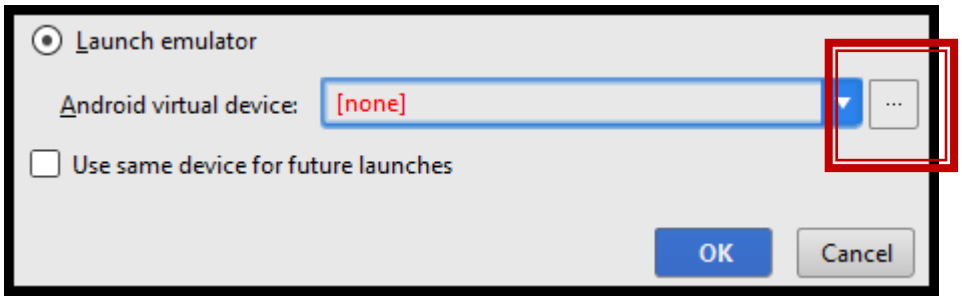
Feito isso será aberta a seguinte caixa de diálogo em seguida :



Caixa de diálogo – Choose device

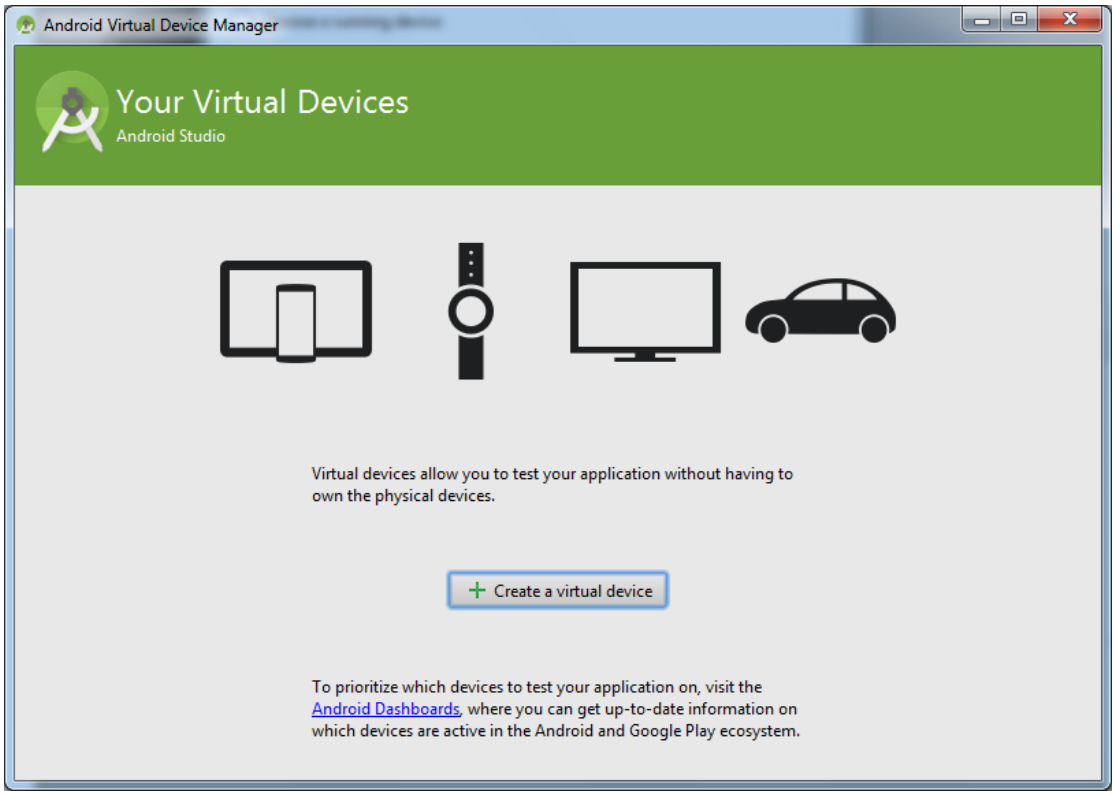
Para executarmos a nossa aplicação iremos fazer uso de um dispositivo virtual (conhecido como “Android Virtual Device”), porém, esse dispositivo virtual não está criado (lembre-se : simplesmente baixamos através do Android SDK a imagem do sistema Android 4.0.3 ,mas, ainda não efetuamos nenhuma configuração e criação de dispositivo).

Vamos nesse exato momento criar um dispositivo virtual que irá executar as nossas aplicações Android. Para criarmos um dispositivo basta clicarmos no botão (...) presente ao lado do campo “Android virtual device”, conforme indica a figura a seguir :



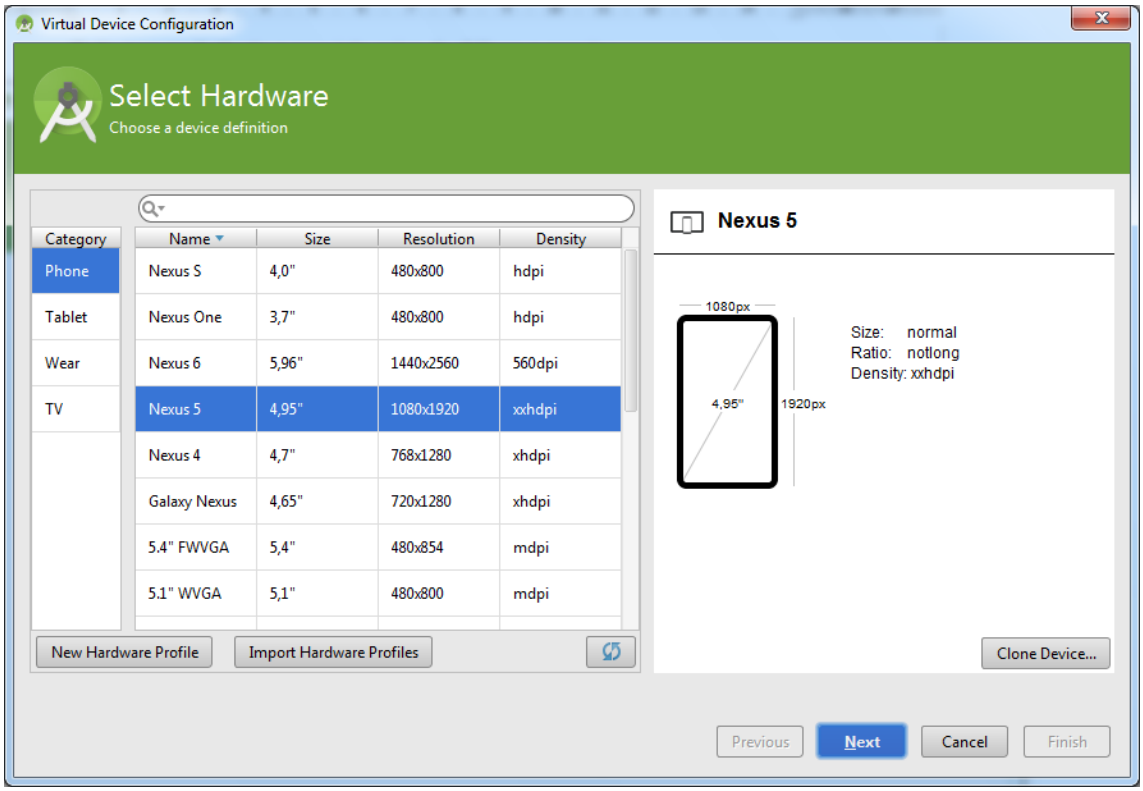
Criando um novo dispositivo

Feito isso será aberta a seguinte caixa de diálogo abaixo :



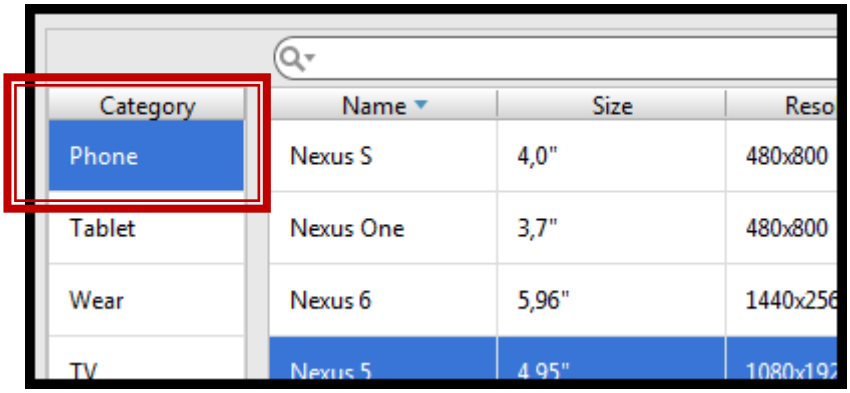
Android Virtual Device Manager

Vamos clicar agora no botão “Create a virtual device”. Feito isso será aberta a seguinte tela :



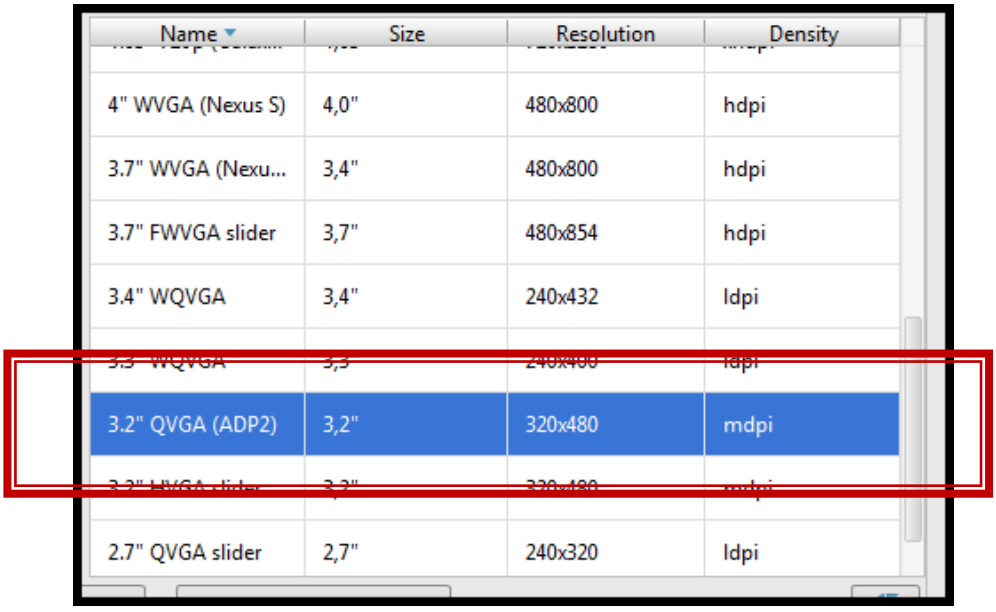
Virtual Device Configuration

Na guia “Category” vamos deixar marcado a opção “Phone”, conforme é mostrado na figura a seguir :



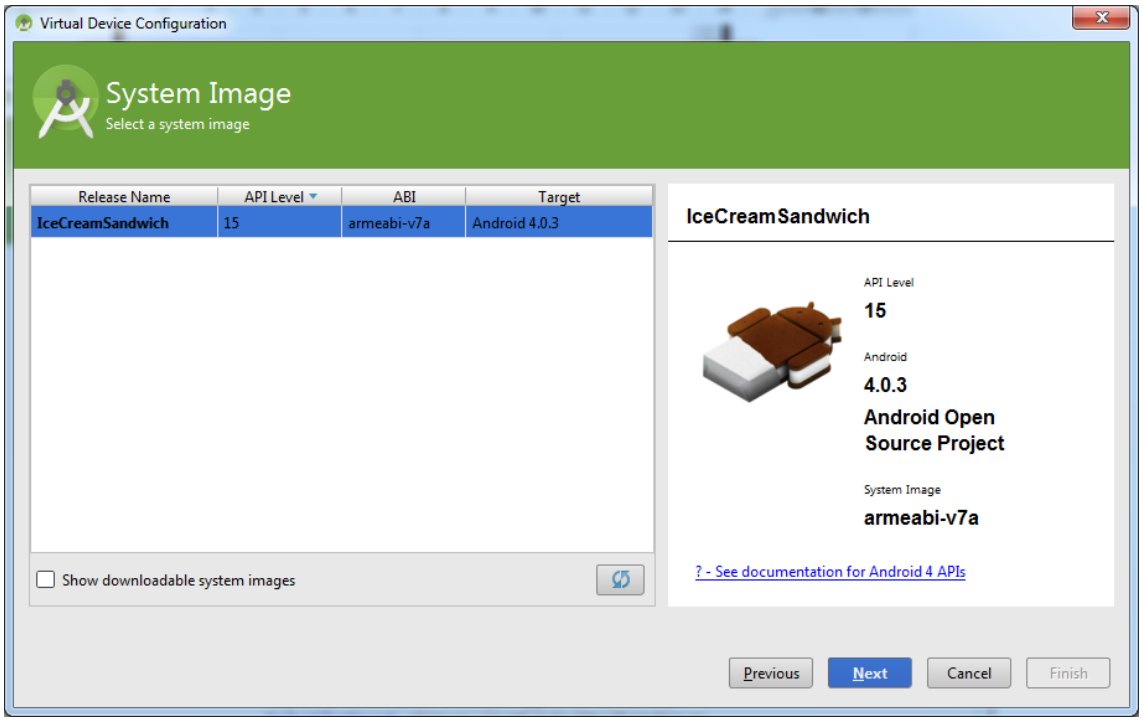
Escolhendo a categoria

Como modelo de smatphone utilizado para executar as nossas aplicações selecione a opção “3.2 QVGA (ADP2)”,conforme indicado na figura abaixo :



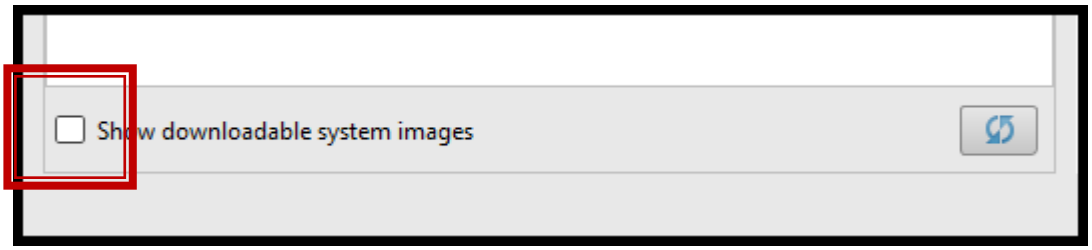
Selecionado o modelo

Agora vamos clicar no botão “Next” para avançarmos para a próxima etapa, conforme indica a próxima figura:



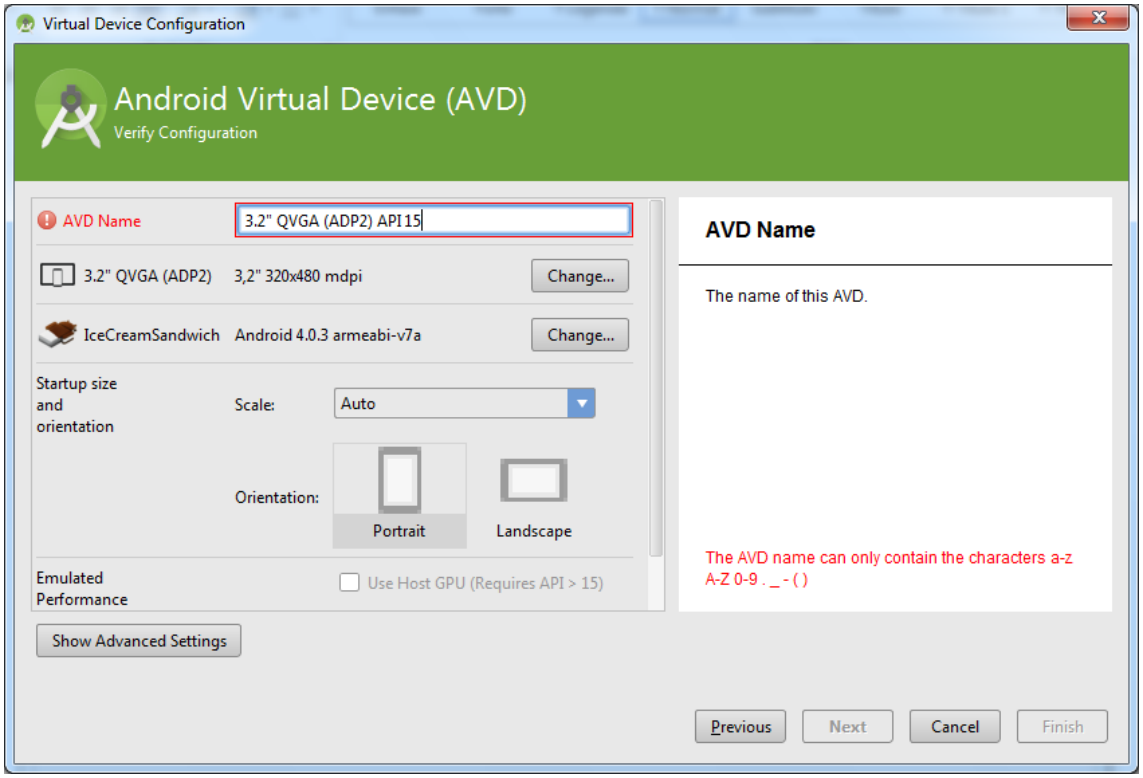
Virtual Device Configuration

Muito bem, nesta seção escolhemos o sistema Android que iremos utilizar , que no caso será o que baixamos através do Android SDK (a versão 4.0.3 Ice Cream Sandwish). Primeiramente, certifique-se de que a opção “Show downloadable system images” ESTA DESMARCADA, se estiver marcada, DESMARQUE (já que não iremos fazer nenhum donwload de imagem nenhuma) :



Deixar esta opção DESMARCADA

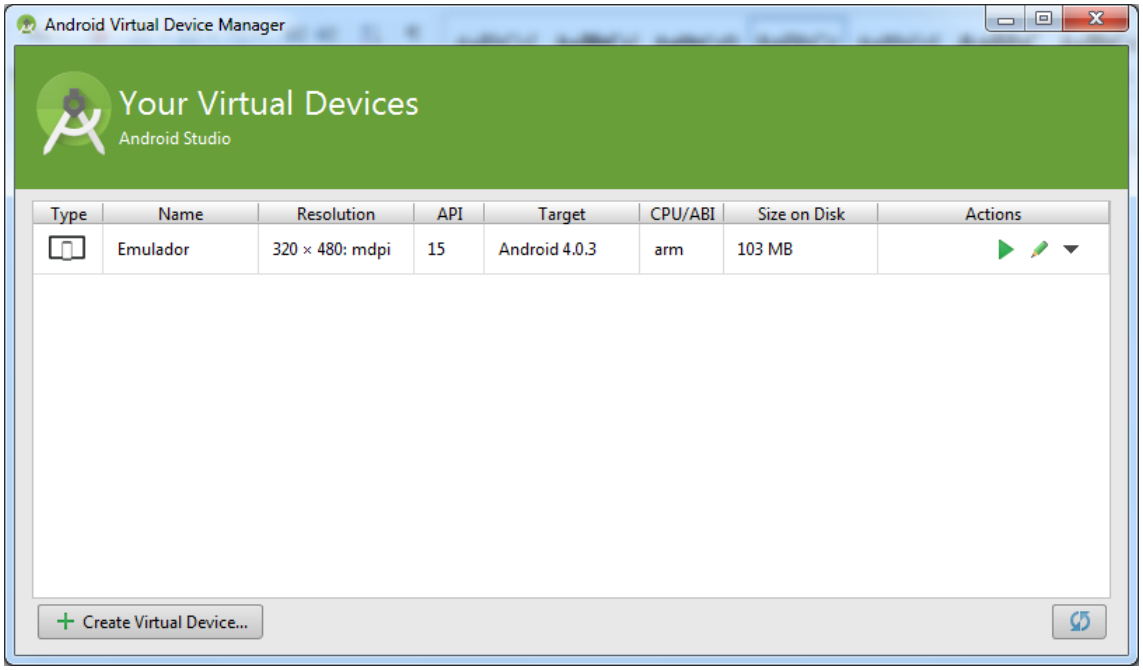
Já que iremos fazer do sistema de imagem do Android 4.0.3 que baixamos, basta simplesmente clicarmos em “Next” para avançarmos para a próxima tela, conforme mostra a figura a seguir :



Virtual Device Configuration

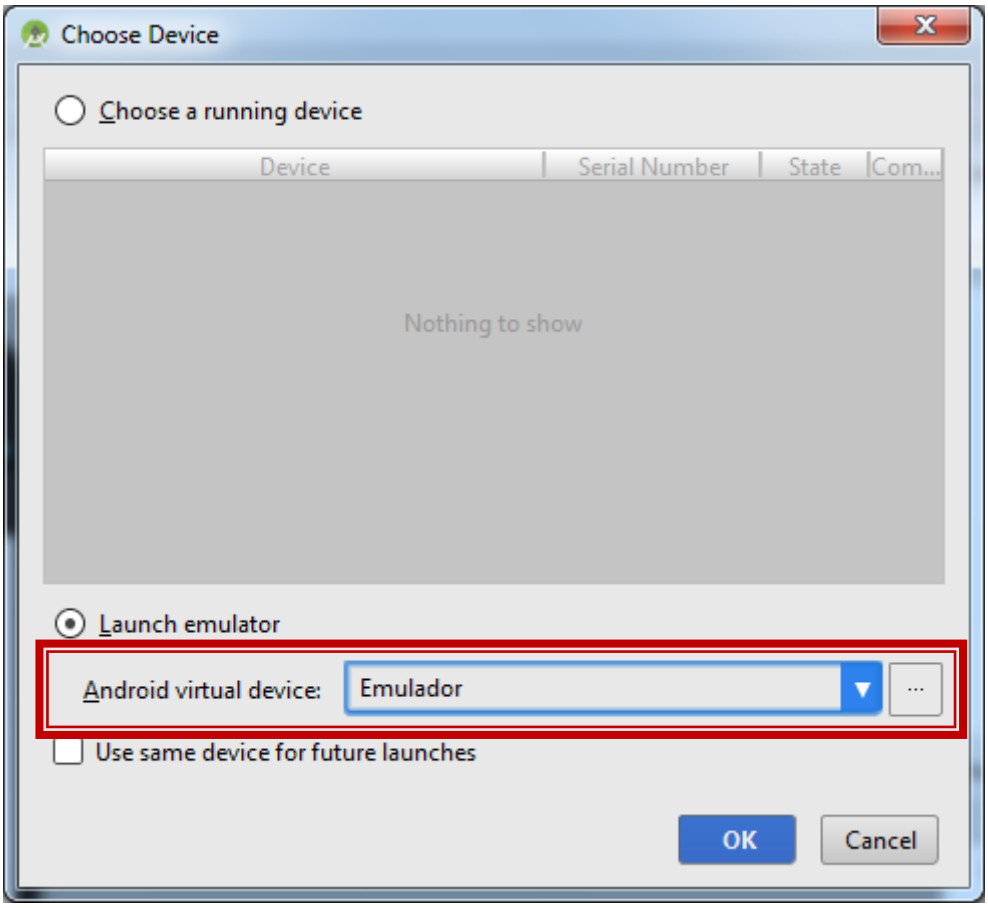
No campo “ADV Name” vamos especificar o nome do nosso dispositivo virtual, que se chamará “Emulador” . Logo, vamos digitar “Emulador” (sem aspas, é claro).

As demais configurações vamos deixá-las como está (default). Feito isso, vamos clicar no botão “Finish” para que o nosso dispositivo virtual possa ser gerado. Vejamos a figura abaixo :



Dispositivo virtual criado

Agora vamos fechar a janela da figura acima e em seguida na caixa de diálogo “Choose Device” vamos selecionar o nosso emulador criado, conforme podemos ver na figura seguinte :



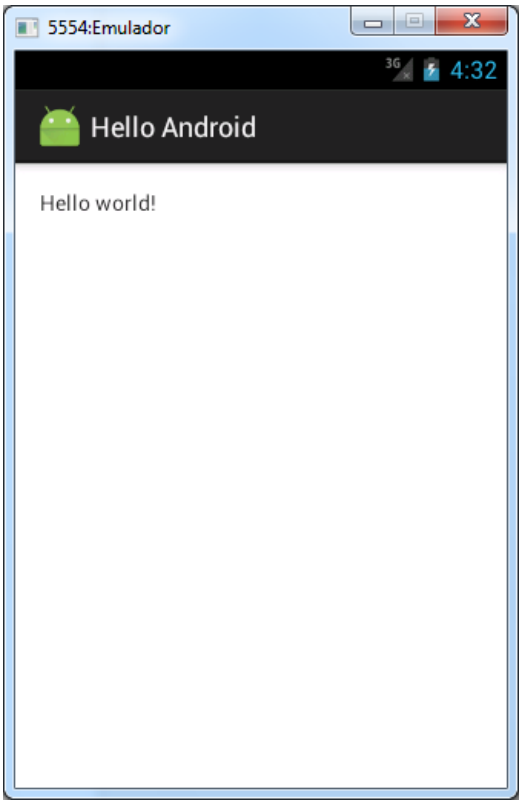
Selecionando o emulador

Depois de selecionarmos o emulador a ser executado basta clicarmos no botão “OK”. Feito isso nosso emulador irá executar, conforme mostra a figura seguinte :

Emulador do Android em Execução

No início da execução do emulador mostra o título Android, conforme você vê na figura acima. Depois vem um outro título escrito “Android”, um pouco maior e cinza , com uma animação. Esse processo normalmente demora em torno de 2 a 10 minutos (dependendo da sua máquina. É recomendável que você tenha no mínimo 1GB 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 em execução, conforme podemos ver na figura seguinte:

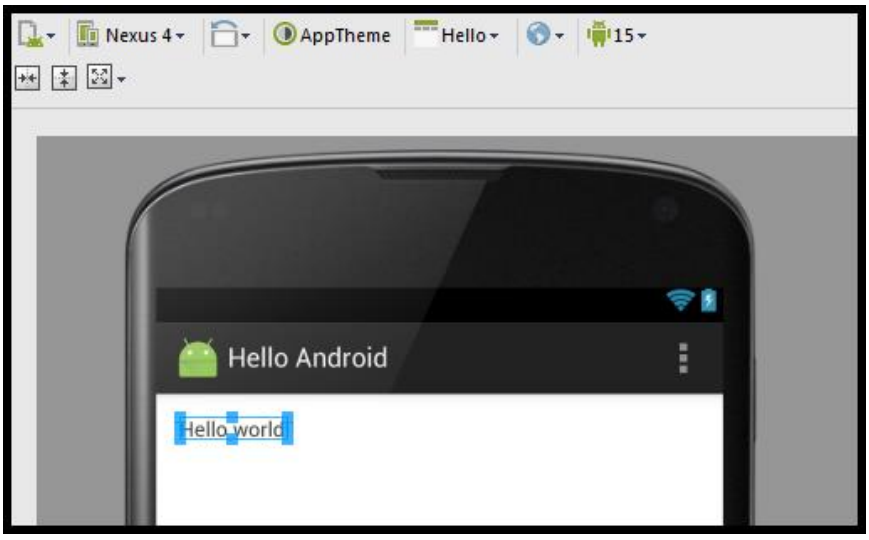


Emulador do Android em Execução

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

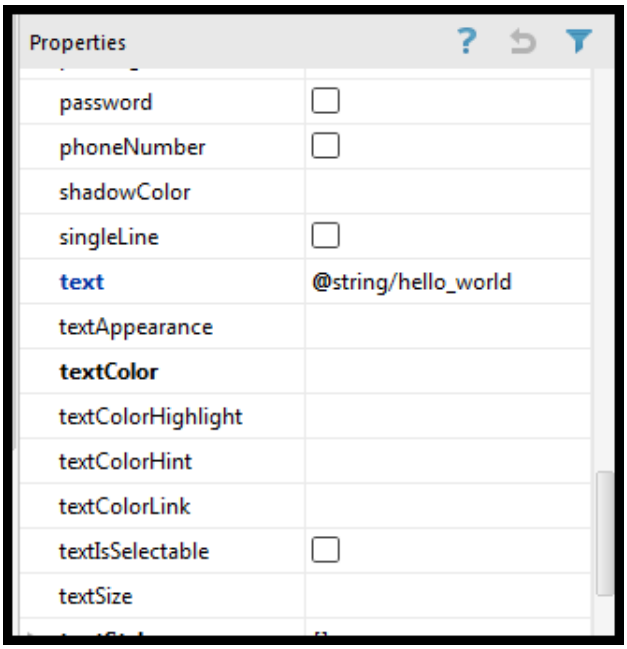
Se você neste exato momento fechou o emulador após a execução da aplicação, vou te dizer uma coisa: “Não era para você ter feito isso”. Se você esperou muito tempo para ver essa aplicação em execução, ao executar novamente a aplicação, possivelmente você vai esperar o mesmo tempo. Ao executar pela primeira 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 outro programa, o Android Studio 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!

Bom, vamos fazer algumas modificações em nossa aplicação. Minimize o emulador e vamos no projeto. Para começar vamos mudar o texto que está sendo exibido na tela. Selecione o texto (componente **TextView**), conforme demonstra a figura seguinte:



Componente selecionado

Se observarmos no lado direito, temos as propriedades do componente selecionado, conforme demonstra a figura seguinte:

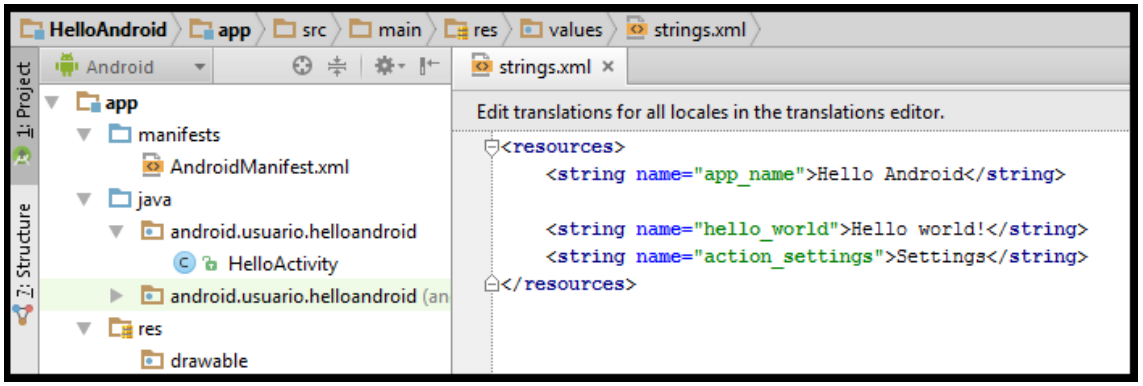


Propriedades do componente selecionado (TextView)

Para alterarmos o conteúdo do texto, devemos alterar a propriedade “text” (conforme você pode ver na figura acima). Se você observar, o conteúdo que está dentro dessa propriedade é “@string/hello_world”. Mas espere ai, o que significa esse expressão se o conteúdo que está sendo exibido é uma frase ? Essa expressão significa que o conteúdo exibido na tela está dentro de uma “constante”.

Quando vimos a estrutura de um projeto Android, nós conferimos que dentro dele existe um diretório chamado “values”, onde dentro do mesmo havia um arquivo que guardava constantes em geral certo ? Esse arquivo é o “strings.xml”. Dentro desse arquivo existe uma constante chamado “hello_world” que armazena a seguinte frase “Hello World”, logo, o conteúdo exibido por esse componente na tela na verdade é o conteúdo da constante “hello_world”, situado dentro do arquivo “strings.xml”, por isso o uso da notação “@string/hello_world”.

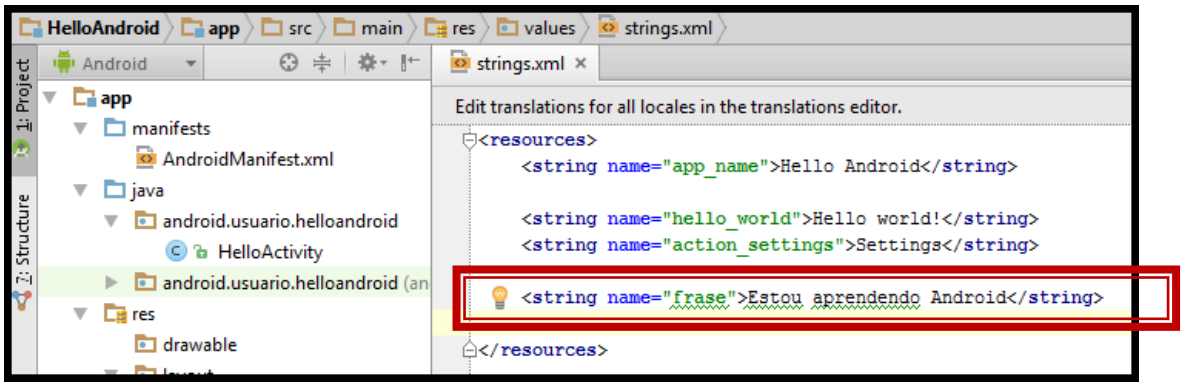
Vamos abrir o arquivo “strings.xml” para visualizarmos seu conteúdo, conforme demonstra a figura seguinte :



O arquivo “strings.xml”

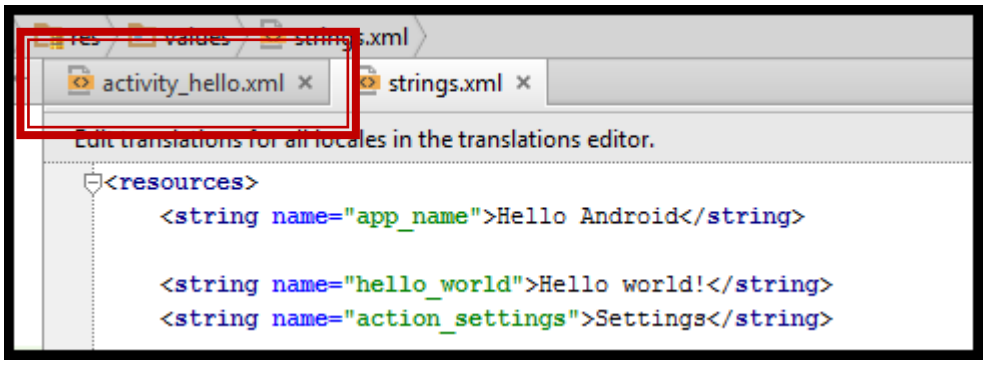
Se observamos o conteúdo dele, temos disponíveis 3 constantes “pré-definidas” e uma delas e a constante “hello_world”.

Na estrutura XML do arquivo “strings.xml” vamos adicionar uma nova constante chamada “frase”, com o seguinte conteúdo “Estou aprendendo Android”. Veja como ficará conforme indica a figura seguinte :



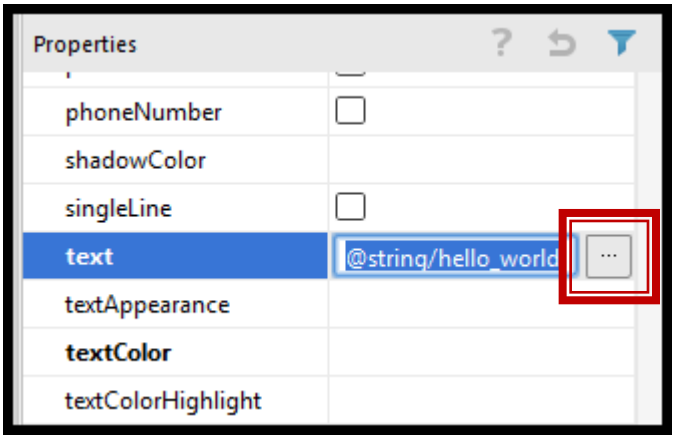
Constante criada

Agora vamos voltar para o arquivo “activity_hello.xml” simplesmente clicando na guia de arquivos, conforme demonstra a figura seguinte:



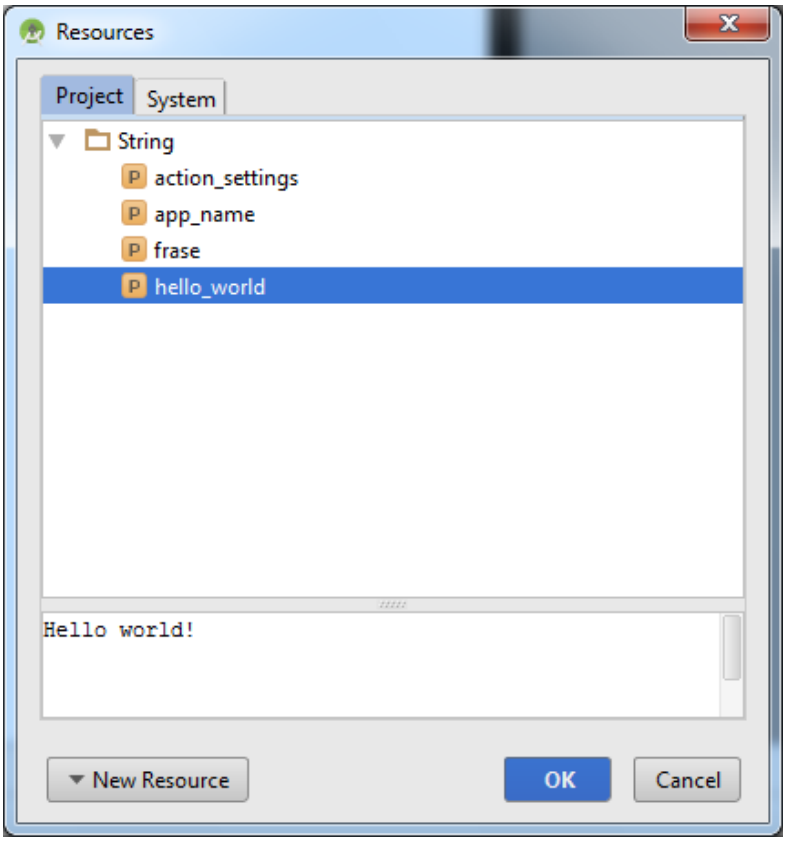
A guia “actitivy_hello.xml”

Agora vamos na propriedade “text” e em seguida clique no botão (...) que está aparecendo ao lado do conteúdo da propriedade. Veja abaixo :



Alterando o conteúdo da propriedade

Feito isso será aberta a seguinte caixa de diálogo abaixo :

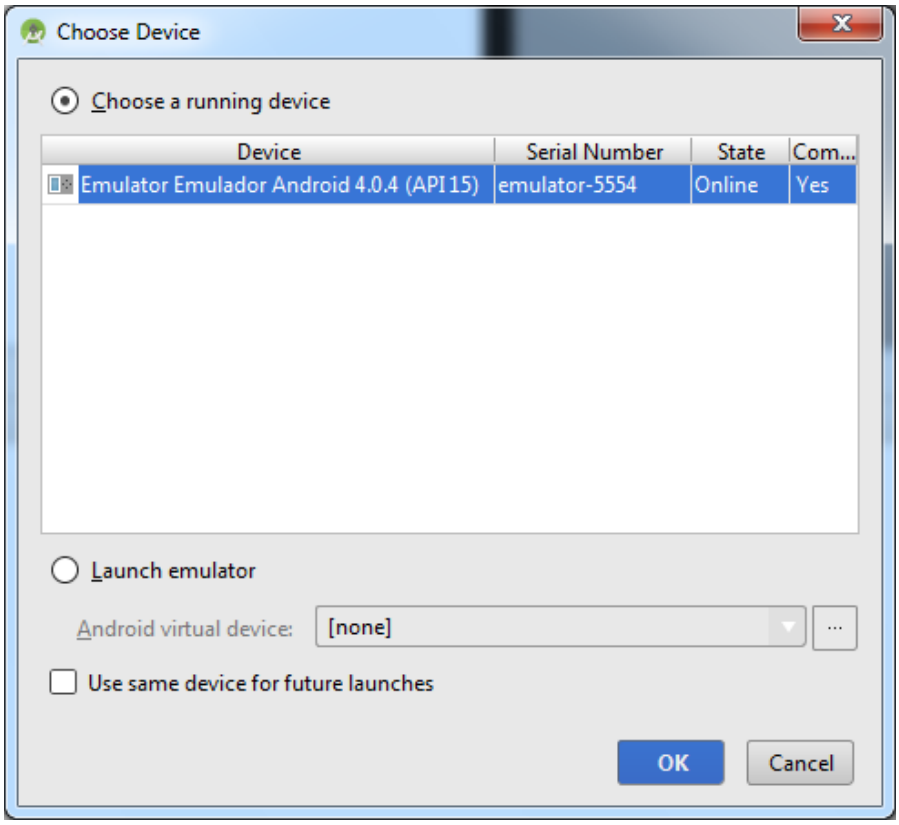


Caixa de diálogo – Resources

Observe que nessa caixa de diálogo temos todas as constantes que existem dentro do arquivo “strings.xml”, incluindo a nossa constante “frase” que acabamos de criar. Vamos selecionar a nossa constante e em seguida clique no botão “OK”. O resultado você confere em seguida :

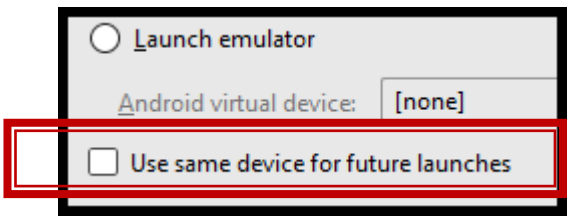
Conteúdo modificado

Depois de alterar o conteúdo da **TextView** vamos executar novamente a aplicação, podemos fazer isso realizando o mesmo procedimento já explicado anteriormente. Possivelmente ao executar a aplicação novamente, deverá ser exibida essa caixa de diálogo :



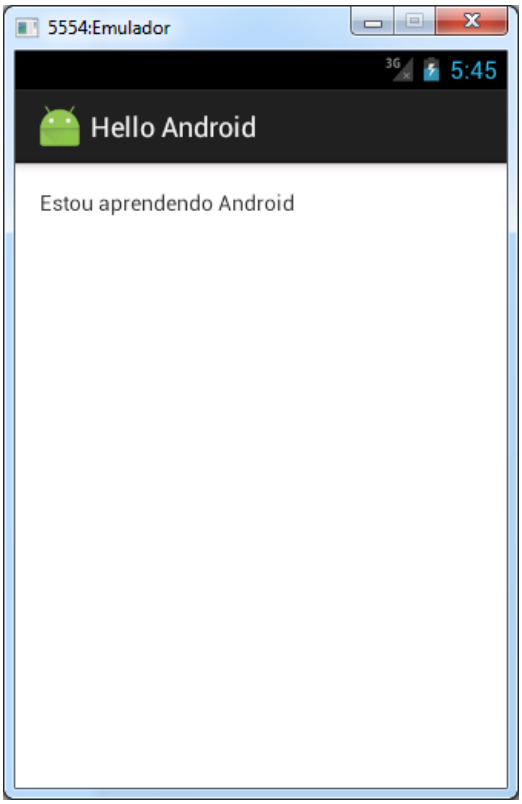
Caixa de diálogo – Choose Device

Bom, vamos continuar a usar o emulador que já está aberto (identificado por "emulator-5554", no campo "Serial Number"). Vamos marcar a opção "Use same device for future launches" :



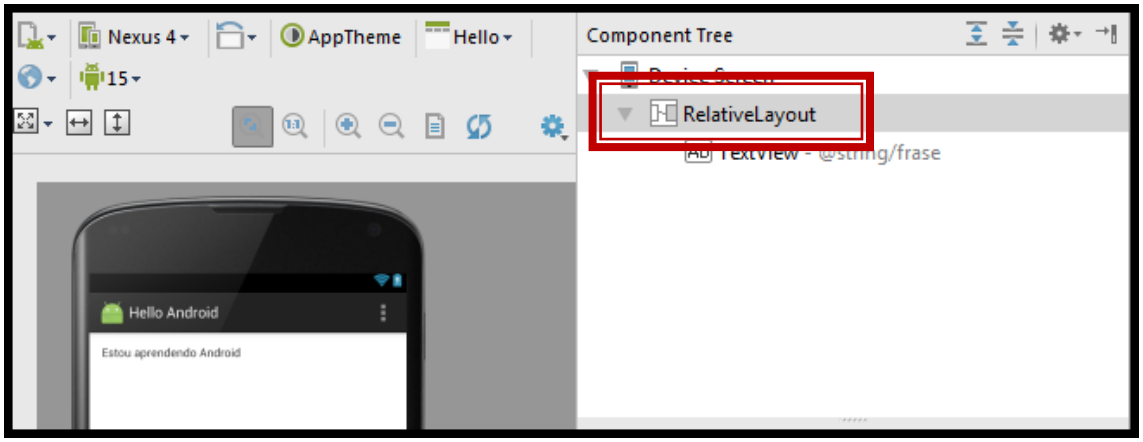
Marcar esta opção

Depois de marcar, clique em "OK" para executarmos a nossa aplicação. Vejamos o resultado :



Aplicação em execução

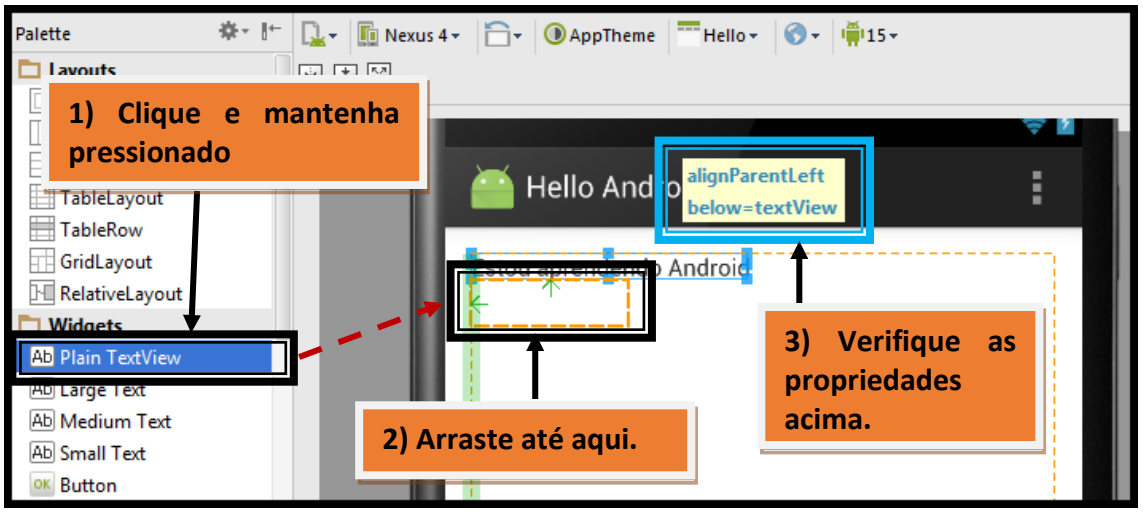
Toda tela de uma aplicação é uma estrutura de layout que permite organizar e distrbuir os componentes nela inseridos conforme a nossa necessidade. A estrutura de layout dessa versão do Android SDK, que e inserida por padrão toda vez que criamos este um projeto em Android, é a “RelativeLayout”. Essa estrutura permite que os componentes nela inseridos possam ser organizados e distribuídos de forma “relativa” a outros componentes presentes também na tela.



A estrutura RelativeLayout

Nas versões mais antigas do Android SDK (usado muito em conjunto com a IDE Eclipse) a estrutura de layout padrão da tela do dispositivo era a "LinearLayout". Essa estrutura "organiza" os componentes de forma que eles sejam distribuídos tanto na horizontal (um ao lado do outro) quanto na vertical (um abaixo do outro).

Vamos agora usando a paleta de componentes arrastar e soltar na tela do dispositivo o componente **TextView** (que se encontra na seção "Widgets", com o título "Plain TextView"). Acompanhe o processo abaixo :



Adicionando o componente

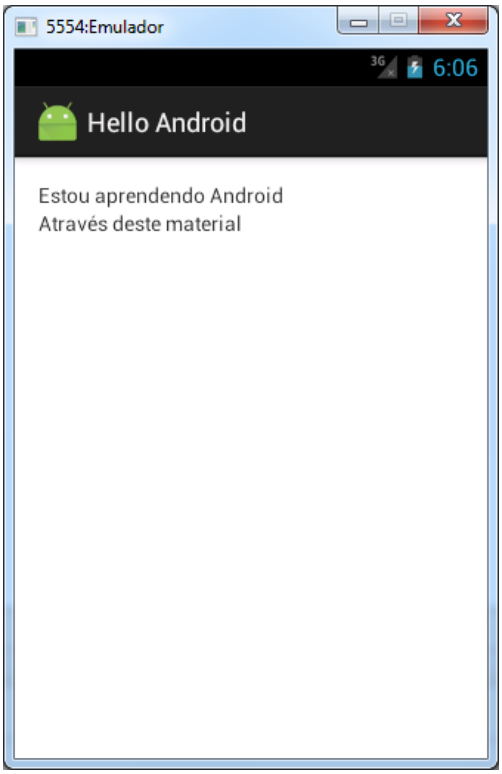
A estrutura **RelativeLayout** distribui os componentes de forma relativa, portando ao arrastar o componente solicitado verifique se as propriedades do mesmo (indicado pela caixa "3") possui os atributos *alignParentLeft* e *bellow=textView*, indicando que o componente estará alinhado a esquerda e abaixo do texto acima. Veja o resultado:

Componente inserido na tela

Quando alteramos o conteúdo (texto) do primeiro componente **TextView** tivemos que fazer uso das constantes (situado dentro do arquivo “strings.xml”), mas, não somos obrigamos a usá-las. Podemos digitar “manualmente” o texto a ser exibido na propriedade “text” do componente. Vamos digitar na propriedade “text” do componente a seguinte frase “Através deste material” (sem aspas, lógico). Veja o resultado na figura seguinte:

Conteúdo (texto) do componente alterado

Vamos salvar as alterações feitas no arquivo e em seguida executar a aplicação para conferir o resultado, como demonstra a figura seguinte:



Aplicação em execução

Conforme já havia mencionado, toda estrutura que constitui a tela de uma aplicação em Android nada mais é do que um código XML. Para visualizarmos o código XML basta clicar na guia “Text”, situado ao lado da guia “Design” . Veja o seu código abaixo:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".HelloActivity">
    <TextView android:text="@string/frase"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Através deste material"
        android:id="@+id/textView2"
        android:layout_below="@+id/textView"
        android:layout_alignParentLeft="true" />
</RelativeLayout>
```

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

Como podemos ver no Android Studio 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.

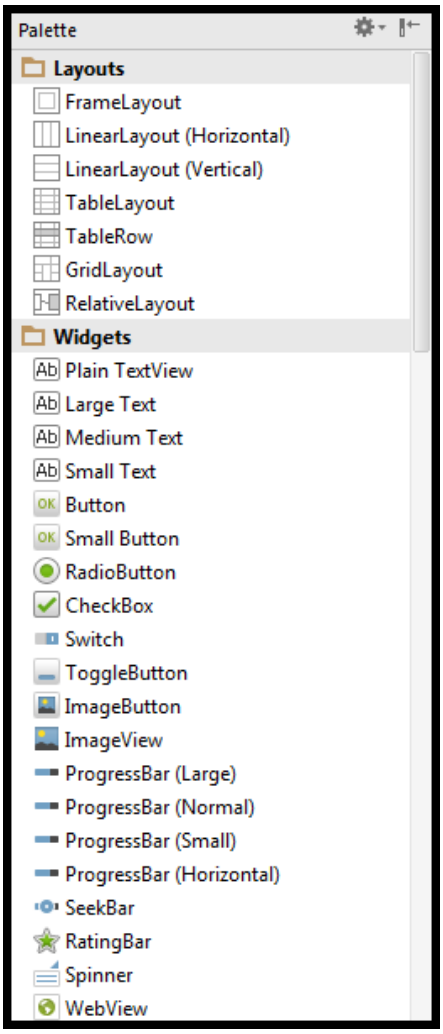
Capitulo 4 Conhecendo as widgets do Android

Toda aplicação Android normalmente é formado por um ou mais widgets, que são componentes gráficos que constituem uma aplicação. A partir de agora iremos conhecer os widgets básicos disponíveis e oferecidos pela plataforma Android, para o desenvolvimento das aplicações. De acordo com alguns widgets que fomos conhecendo, vamos desenvolver aplicações que demonstrem o uso deles.

No capítulo anterior aprendemos a desenvolver nossa primeira aplicação em Android, algo absolutamente simples. Agora vamos conhecer melhor a paleta de componentes onde nela estão disponíveis todos os widgets que podemos utilizar em uma aplicação (todos eles separados por categorias).

4.1) A paleta de componentes e suas widgets

A ferramenta de desenvolvimento do Android SDK nos oferece uma gama de componentes (ou widgets, como preferirem) que podemos utilizar em uma aplicação a ser construída. Podemos conferir esses widgets na paleta mostrada pela figura seguinte:



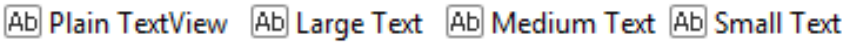
A paleta de componentes

Conforme havia falado, os componentes estão distribuídos nas mais diversas seções presentes na paleta de componentes. Vamos conhecer as seções desta paleta e os componentes nela presentes.

4.1.1) A seção “Widgets”

Nesta seção estão disponíveis os componentes mais básicos que podem ser utilizados em uma aplicação Android, são eles:

TextView : Componente que 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. Ele está disponível em quatro estilos (Plain , Large, Medium e Small). Veja os ícones desse componente na imagem seguinte:

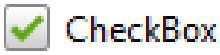


Ícones do componente TextView (Versão normal, large, medium e small)

Button : Componente que representa um botão onde podemos clicar nele e também atribuir ações que podem ser executadas caso isso aconteça. Ele se encontra nas versões normal e small (pequena): Veja seu ícone na paleta de componentes :

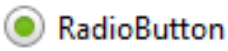
Ícones do componente Button (Versão normal e small)

CheckBox : Esse componente funciona como uma opção, onde nele podemos marcá-lo e desmarcá-lo. Veja o ícone do componente abaixo:



Ícone do componente CheckBox

RadioButton : Esse componente funciona como uma opção, normalmente utilizado quando temos uma situação onde devemos escolher uma entre várias opções (como numa prova de múltipla escolha). Veja o ícone desse componente na paleta de componentes :



Ícone do componente RadioButton

Spinner : Esse componente nada mais é do que uma caixa de combinação (também conhecido como Combo Box). Nesse componente podemos adicionar vários itens que poderão ser selecionados pelo usuário através do mesmo. Veja o ícone desse componente na figura seguinte:

Ícone do componente Spinner

ProgressBar : Esse componente exibe uma barra de progresso na tela e está disponível em 3 versões (normal e large (giratórias) e horizontal (barra)). Veja seus ícones abaixo:

Ícone do componente ProgressBar

RatingBar : Esse componente é bastante utilizado para fazer sistemas de votações e classificações (aqueles sistemas em que você define se uma coisa é ruim, regular, boa, ótima e etc). Veja o ícone do componente na figura seguinte:

Ícone do componente RatingBar

ImageView : Esse componente simplesmente serve para exibir imagens que se colocam nele. Os formatos de imagens suportados por esse componente são : PNG, JPEG, BMP, GIF. Veja o ícone desse componente em seguida:

Ícone do componente ImageView

ImageButton : Esse componente é derivado do componente Button só que ao invés de exibir um texto dentro dele, exibe uma imagem. Os formatos de imagens suportados por esse componente são : PNG, JPEG, BMP, GIF. Veja o ícone desse componente abaixo:

Ícone do componente ImageButton

Gallery : Esse componente funciona como uma galeria de imagens, onde nele podemos adicionar várias imagens e ao mesmo, visualiza-las. Veja o ícone desse componente abaixo:

Ícone do componente Gallery

4.1.2) A seção “Text Fields”

Nesta seção estão disponíveis todos os componentes baseados em caixas de texto, e todos eles são baseados no componente **EditText**. Vamos ver alguns desses componentes abaixo:

Plain Text: Esse é o modelo de caixa de texto “padrão”, que permite a digitação de qualquer tipo de caractere. Veja seu ícone abaixo:

Ícone do componente Plain Text

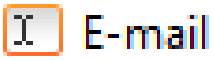
Person Name: Esse modelo de caixa de texto permite a digitação de nomes pessoais (colocando a inicial de cada palavra em maiúsculo). Veja seu ícone abaixo:

Ícone do componente Plain Text

Password: Esse modelo de caixa de texto permite a digitação de senhas e está disponível tanto na versão alfanumérica quanto na numérica (Numeric). Veja os ícones de componente abaixo:

Ícones do componente Password (na versão normal e numérica)

E-mail: Esse modelo de caixa de texto permite a digitação de e-mail. Veja o ícone de componente abaixo:



Ícone do componente E-mail

Phone: Esse modelo de caixa de texto permite a digitação de telefones. Veja o ícone de componente abaixo:

Ícone do componente Phone

Multiline Text : Esse modelo de caixa de texto permite várias linhas de texto, de acordo com a nossa necessidade. Veja o ícone desse componente abaixo:

Ícone do componente Multiline Text

4.1.3) A seção “Layouts”

Nesta seção estão disponíveis estruturas de layouts que podemos utilizar em nossas aplicações para organizar a disponibilidade dos componentes dentro da tela, no dispositivo. Vejamos esses componentes:

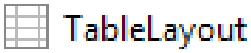
LinearLayout : Essa estrutura (conforme já havia mencionado) organiza os componentes dentro dela de forma que os mesmos sejam distribuídos de forma horizontal (um ao lado do outro) ou vertical (um abaixo do outro), de acordo com a necessidade. Veja os ícones desse componente:

Ícones do componente LinearLayout (horizontal e vertical)

RelativeLayout : Essa estrutura (conforme também já havia mencionado) organiza os componentes dentro dela de forma que os mesmos sejam distribuídos livremente na tela (em qualquer ponto em que você desejar, relativo a outros componentes que , possivelmente, estejam na tela). Veja os ícones desse componente:

Ícone do componente RelativeLayout

TableLayout: Essa estrutura organiza os componentes dentro dela de forma como se estivessem em uma tabela (com o auxílio de um componente útil, o **TableRow**, também presente nesta seção). Veja o ícone desse componente abaixo:



Ícone do componente TableLayout

4.1.4) A seção “Containers”

Nesta seção estão disponíveis componentes que naturalmente funciona como complementos (compostos) para outros componentes. Vejamos os componentes dessa seção :

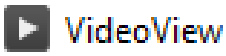
ListView : Esse componente funciona como uma lista onde nele podemos adicionar itens e visualizar os mesmos (conhecido em algumas ferramentas de desenvolvimento como “ListBox”). Vejamos o ícone desse componente:

Ícone do componente ListView

RadioGroup : Esse componente nada mais é do que um estrutura constituída (por padrão) por três RadioButtons , que podem ser distribuídos de forma horizontal e vertical. Veja o ícone desse componente na figura seguinte:

Ícone do componente RadioGroup

VideoView : Esse componente serve para reproduzir vídeos que queiramos visualizarmos através dele. Veja o ícone desse componente abaixo:

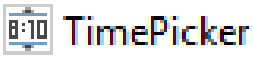


Ícone do componente VideoView

4.1.5) A seção “Date & Time”

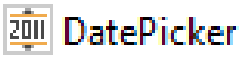
Nesta seção estão disponíveis os componentes que trabalham com data e hora. Vejamos os componentes abaixo:

TimePicker : Esse componente é muito útil , nele podemos estipular ou definir uma determinada hora, de acordo com a nossa necessidade. Veja o ícone desse componente em seguida:



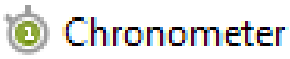
Ícone do componente TimePicker

DatePicker : Esse componente é muito útil , nele podemos estipular ou definir uma determinada data, de acordo com a nossa necessidade. Veja o ícone desse componente abaixo:



Ícone do componente DatePicker

Chronometer : Esse componente nada mais é do que um cronometro digital, que podemos utilizá-lo de acordo com a nossa necessidade . Ele é derivado do componente TextView. Vejamos seu ícone abaixo:

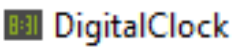


Ícone do componente Chronometer

AnalogClock : Esse componente nada mais é do que um relógio analógico que mostra a hora do sistema. Vejamos seu ícone abaixo:

Ícone do componente AnalogClock

DigitalClock : Esse componente nada mais é do que um relógio digital (também derivado do TextView) que mostra a hora do sistema. Vejamos seu ícone abaixo:



Ícone do componente DigitalClock

4.1.6) A seção “Expert”

Nesta guia estão componentes de categorias miscelâneas (mistas) , que envolvem manipulação de animações e efeitos de transição e etc. Vamos ver alguns componentes

ViewFlipper : Esse componente funciona como se fosse uma estrutura de layout para elementos, onde cada elemento é exibido (e também ocultado) usando um efeito de transição. Vejamos seu ícone abaixo:

Ícone do componente ViewFlipper

AutoCompleteTextView : Esse componente é muito útil quando queremos que durante a digitação de um conteúdo , o mesmo apresente sugestões de palavras (muito comum nas aplicações de hoje em dia). Veja seu ícone abaixo :

Ícone do componente AutoCompleteTextView

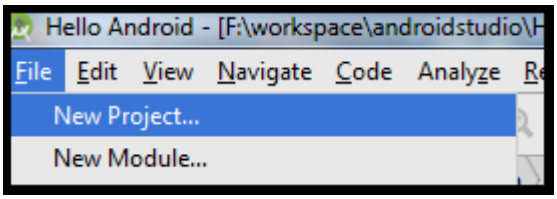
Neste capítulo tivemos a oportunidade de conhecer os componentes básicos e essenciais da plataforma Android. No próximo capitulo iremos construir diversas aplicações que façam uso da maioria dos componentes descritos nessa seção.

Capitulo 5 Construindo nossas aplicações no Android

Vamos colocar a mão na massa ? A partir de agora iremos começar a desenvolver as nossas aplicações no Android utilizando os componentes descritos no capítulo anterior. Começaremos com aplicações simples e aos poucos iremos evoluir, criando aplicações mais ricas.

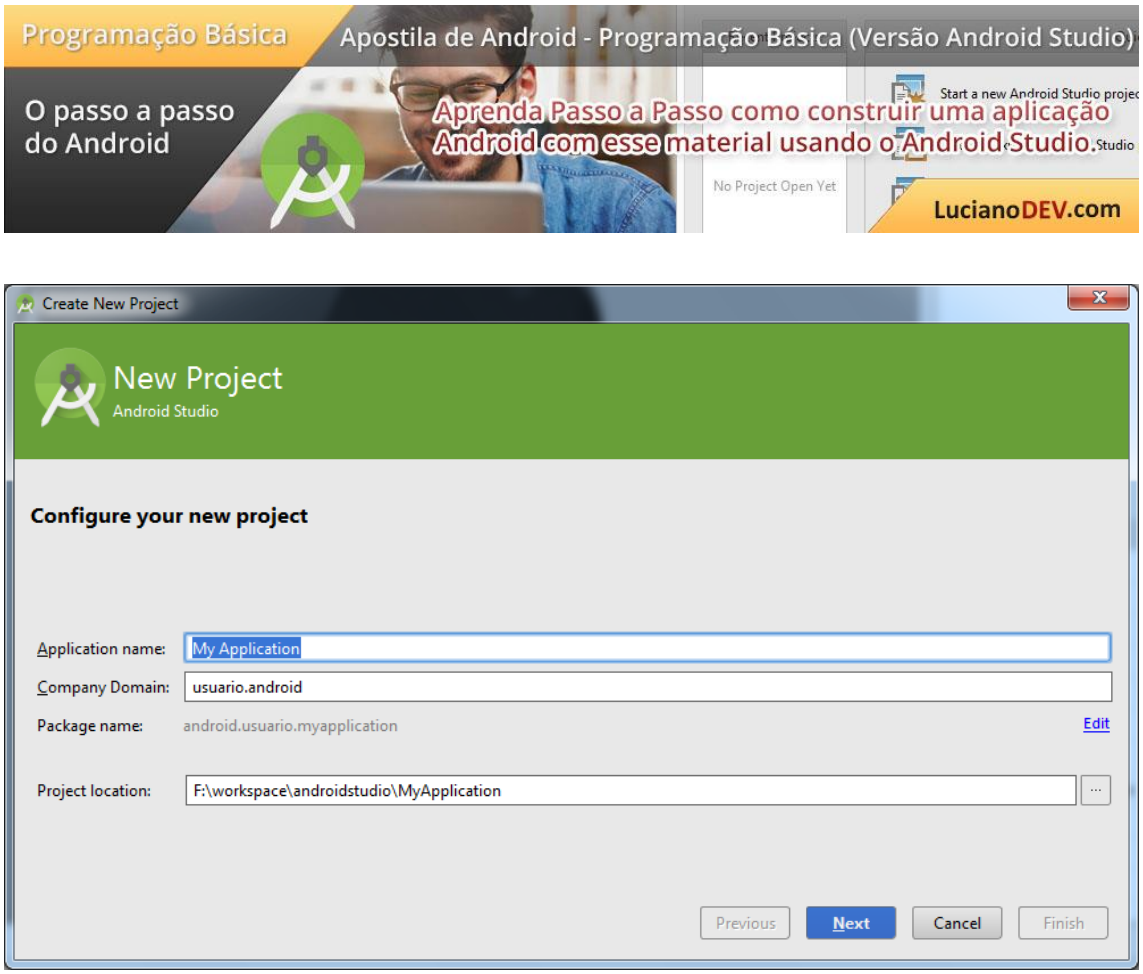
5.1) Desenvolvendo uma Calculadora Básica

Vamos construir a nossa primeira aplicação que vai consistir em uma calculadora básica com as quatro operações aritméticas. Para criar um projeto no Android Studio vamos no menu “File”/“New Project”. Confira na figura seguinte:



New Project (pelo menu)

Seguido os passos descritos acima, irá se abrir a caixa de diálogo abaixo:



Criando o projeto Calculadora

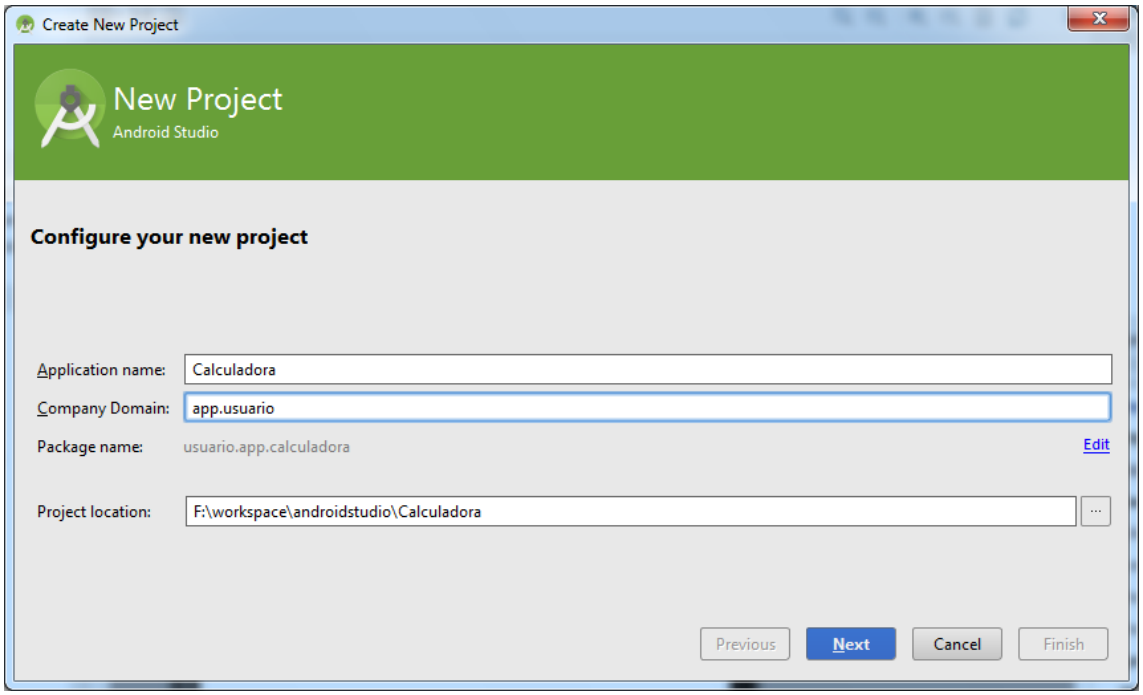
Agora vamos preencher os campos, conforme abaixo:

Application Name : Calculadora

Company Domain : app.usuario

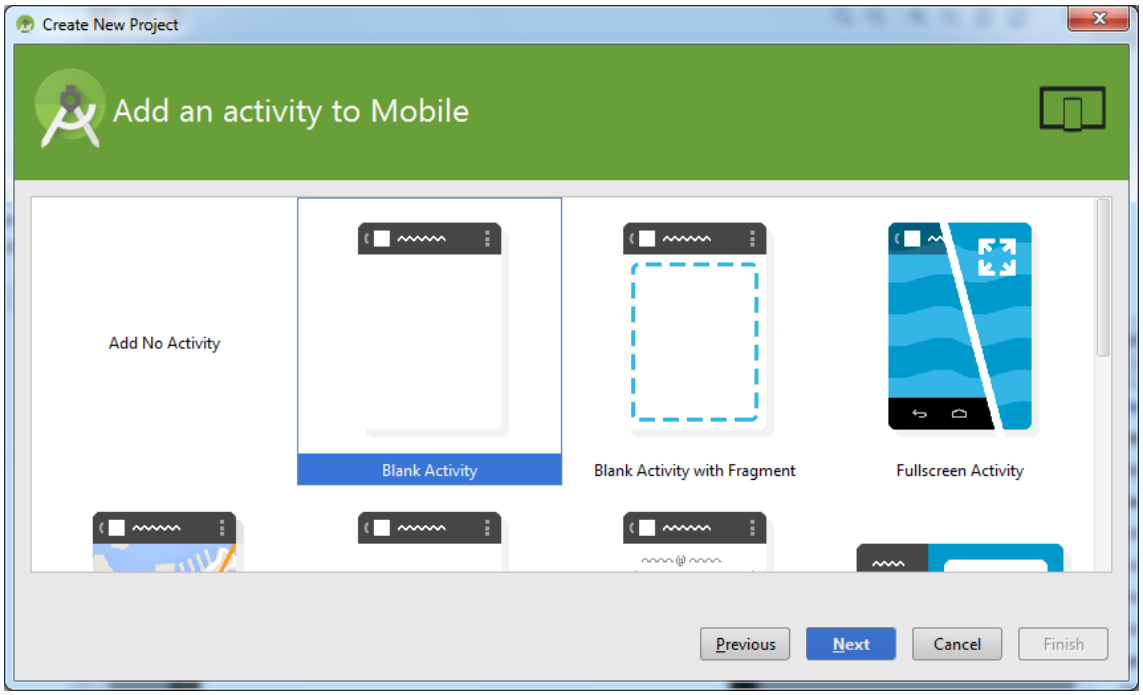
Project location : (Fica a sua escolha)

Confira como ficou na figura seguinte:



Criando o projeto Calculadora – Campos preenchidos

Agora na próxima seção (clicando em “Next”) será aberta a tela de configurações do projeto (que já estão previamente definidas.Nela não iremos alterar nada, simplesmente vamos clicar em “Next”). Na próxima etapa vamos escolher qual tipo de Activity iremos criar (por padrão, o BlackActivity), conforme demonstra a próxima imagem:



Criando o projeto Calculadora – Definindo a Activity

Agora na próxima seção (clicando em “Next”) vamos preencher as informações da Activity, conforme é mostrado abaixo:

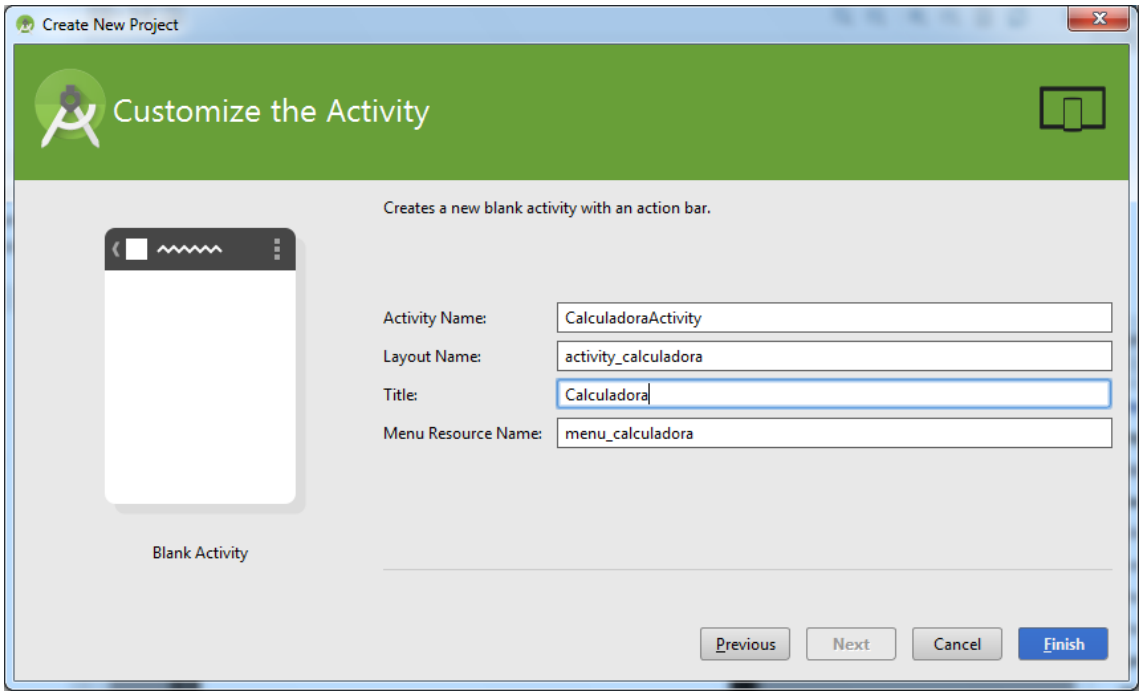
Activity Name : CalculadoraActivity

Layout Name : activity_calculadora

Title : Calculadora

Resource Menu Name : menu_calculadora

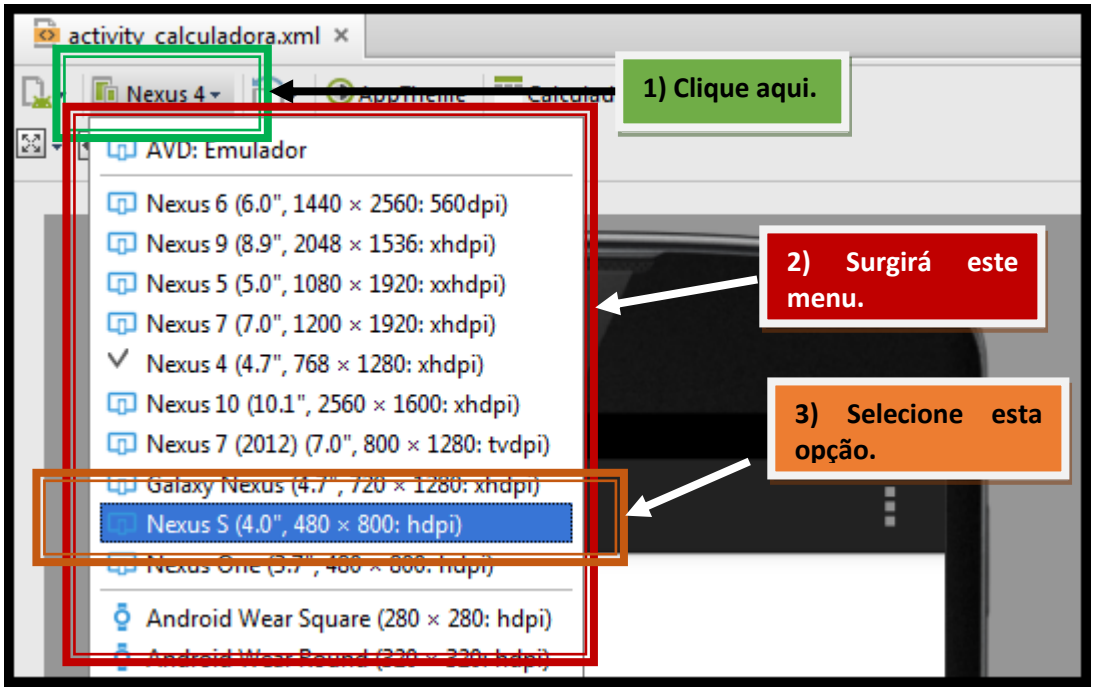
Confira como ficou na figura seguinte:



Criando o projeto Calculadora – Informações preenchidas

Depois de preenchidas as informações, vamos criar o nosso projeto clicando no botão “Finish”. Feito isso o nosso projeto será criado.

Antes de começarmos a desenvolver nosso projeto, vamos alterar a “Skin” do nosso dispositivo (preview). O padrão é o “Nexus 4”. Vamos trocar para o “Nexus S”, conforme indica a figura a seguir :



A escolha da Skin mencionada é pela fato de a mesma “se aproximar” das dimensões do emulador, porém , FIQUE A VONTADE para escolher a Skin que você desejar.

Na tela da aplicação selecione o componente **TextView** na tela (cuja frase está escrito “Hello world”) e vamos alterar as seguintes propriedades, como segue:

TextView

Propriedade	Valor
text	Digite o primeiro número

Veja o resultado:

Tela da aplicação em desenvolvimento

Agora arraste e solte um componente “Plain Text” (**EditText**) presente dentro da seção “Text Fields” abaixo do título. Veja como ficou em seguida :

Componente "Plain Text" (EditText) inserido

Com o componente selecionado, vamos em suas propriedades para alterar os seguintes valores, como segue :

EditText (Plan Text)

Propriedade	Valor
layout:width	match_parent
id	ednumero1

Na propriedade "layout:width" especificamos a largura do nosso componente, que no caso de nossa aplicação possuirá (ocupará) a mesma largura da tela do dispositivo ("match_parent").

Em "id" especificamos o nome do nosso componente, que será tratado (e identificado) na programação ao utilizarmos a linguagem Java. Porque alterar a sua ID ? Isso é necessário pois vamos "manipular" esse componente através do código Java, então nada mais justo do que trabalhar com componentes cujos nomes estejam de forma clara e organizada.

Agora arraste e solte um componente **TextView** abaixo da caixa de texto que inserimos, e em seguida altere as seguintes propriedades:

TextView

Propriedade	Valor
text	Digite o segundo número

Logo após , arraste e solte um componente Plain Text (**EditText**), situado na guia “Text Fields”, abaixo do componente acima inserido, e altere seu nome (ID) para “ednumero2” e sua largura (layout:width) para “match_parent”. (conforme já foi mostrado).

Veja o resultado:

Tela da aplicação em desenvolvimento

Agora vamos adicionar um componente **Button** abaixo da caixa de texto, que vai ser o nosso botão de “somar” os números. Depois de adicionar, vamos alterar as suas propriedades, conforme é mostrado abaixo:

Button

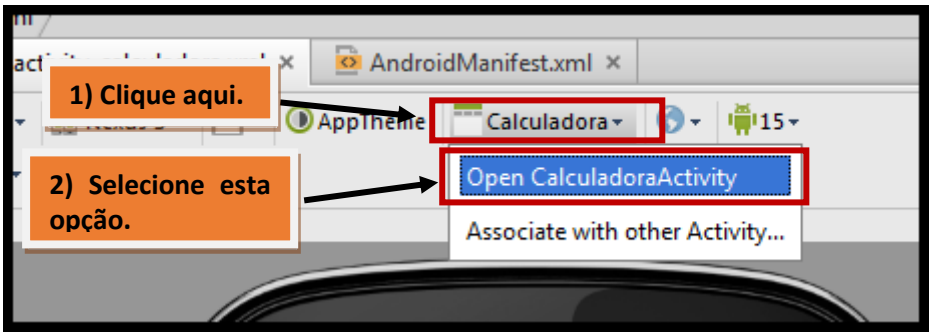
Propriedade	Valor
text	Somar
layout:width	match_parent
id	btsomar

Vejamos o resultado abaixo:

Tela da aplicação em desenvolvimento

Para começarmos, vamos fazer o teste da nossa aplicação realizando somente soma dos números (implementaremos as outras operações restantes daqui a pouco).

Para inserirmos o código em nossa classe (o arquivo “CalculadoraActivity.java”) , basta seguir os procedimentos da figura a seguir :



Chamando o arquivo “CalculadoraActivity.java”

Feito isso será aberto o seu conteúdo conforme é demonstrado na imagem seguinte:

CalculadoraActivity.java

```
package usuario.app.calculadora;

import ...


public class CalculadoraActivity extends Activity {

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


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_calculadora, menu);
        return true;
    }
```

Conteúdo do arquivo "CalculadoraActivity.java"

Se você observar no código acima, na seção onde se declaram os pacotes, existe a seguinte instrução :

```
import ...;
```

Nessa linha se você observar (conforme demonstra a figura acima), existe um sinal de "+", que na verdade indica que há mais de uma importação (processo esse que o eclipse faz para "simplificar" e "organizar" a compreensão do código). Para você visualizar todos os pacotes utilizados basta clicar nesse sinal. Confira o resultado na próxima figura:

CalculadoraActivity.java

```
package usuario.app.calculadora;

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


public class CalculadoraActivity extends Activity {
```

Visualizando todos os pacotes

Para começar, vamos importar alguns pacotes da plataforma Android que serão necessários para o desenvolvimento da nossa aplicação. Na seção onde se encontram os pacotes importados , vamos importar mais alguns pacotes digitando as seguintes linhas de comando abaixo:

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

Agora no código do nosso programa, antes (EU DISSE ANTES) da linha:

```
@Override
```

Digite:

```
EditText ednumero1,ednumero2;
Button btsomar;
```

Vejamos como ficou na figura seguinte :

Atributos declarados

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 dentro do método **onCreate** após a linha:

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

Digite as seguintes linhas de código:

```
ednumero1 = (EditText) findViewById(R.id.ednumero1);
ednumero2 = (EditText) findViewById(R.id.ednumero2);
btsomar  = (Button) findViewById(R.id.btsomar);
```

Veja como ficou na figura abaixo :

```
public class CalculadoraActivity extends Activity {

    EditText ednumero1,ednumero2;
    Button btsomar;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_calculadora);
        ednumero1 = (EditText) findViewById(R.id.ednumero1);
        ednumero2 = (EditText) findViewById(R.id.ednumero2);
        btsomar  = (Button) findViewById(R.id.btsomar);

    }
```

Código digitado

Agora vou explicar as linhas de comando acima que adicionamos. A linha:

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

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

Se lembra do nome da primeira **EditText** que está no código XML? Ela se chama *"ednumero1"*.

Vamos entender. Observe que para fazer referência ao **EditText** pelo método **findViewById** eu passei o parâmetro "*R.id.numero1"*.

Na segunda instrução que digitamos, para fazer referência à segunda **EditText**, cujo nome é "*ednumero2"*, pelo método **findViewById**, passei o parâmetro "*R.id.numero2"*.

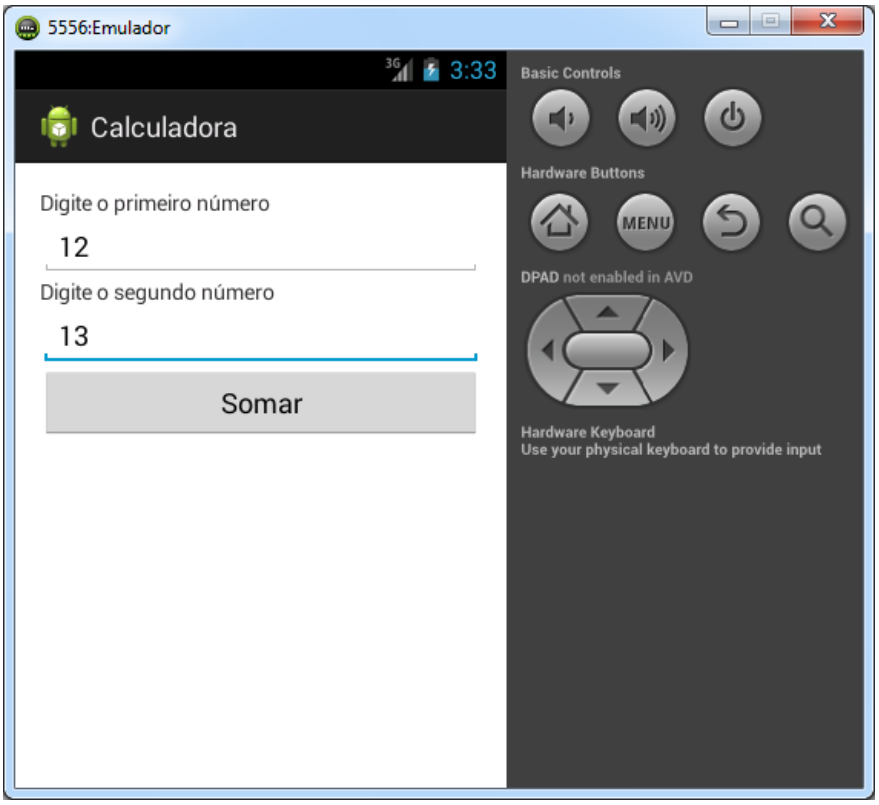
Como você pode ver, estou fazendo uso da classe **R** (uma classe "interna" do Android, onde todos os elementos e diretórios são estruturados em atributos da classe) que funciona como interface entre o código Java e o arquivo XML. O procedimento é o mesmo para o componente **Button**.

Agora iremos adicionar um evento em nosso componente **Button** que será responsável por "detectar" toda vez que ele for "clicado" (tocado na tela), executando um conjunto de instruções após o evento (que vai consistir na soma dos números e na exibição do resultado). Para adicionarmos esse evento em nosso componente, basta escrevermos, após a última instrução que adicionamos, a seguinte linha de código **destacado em azul**:

```
protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_calculadora);

        ednumero1 = (EditText) findViewById(R.id.ednumero1);
        ednumero2 = (EditText) findViewById(R.id.ednumero2);
        btsomar  = (Button) findViewById(R.id.btsomar);

        btsomar.setOnClickListener(new View.OnClickListener() {

              @Override
              public void onClick(View v) {

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

                  double num2 = Double.parseDouble(
                  ednumero2.getText().toString());
                  double soma = num1 + num2;

                  AlertDialog.Builder dialogo = new
                  AlertDialog.Builder(CalculadoraActivity.this);

                   dialogo.setTitle("Resultado soma");

                   dialogo.setMessage("A soma é " + soma);

                   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 seguinte:



Aplicação em execução

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

Irei descrever o código do evento de "clique". O método **setOnClickLisneter** serve para definir um evento de "clique" em um componente. Como parâmetro, criamos uma instância da interface **OnClickListener**, e dentro da mesma existe um 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 dentro do componente identificado como *ednumero1*. 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 similar ao que já foi explicado.

Logo após a soma dos números que será armazenada na variável *soma* , vem o código em seguida:

```
AlertDialog.Builder dialogo = new
AlertDialog.Builder(CalculadoraActivity.this);

dialogo.setTitle("Resultado soma");

dialogo.setMessage("A soma é " + soma);

dialogo.setNeutralButton("OK", null);

dialogo.show();
```

Que mostra a soma dos números digitados na tela. Para conseguirmos exibir uma mensagem na tela, tivemos que fazer uso da classe **AlertDialog.Builder**, responsável por criar caixas de diálogo e exibi-las. Vamos aos comentários.
A linha de comando:

```
AlertDialog.Builder dialogo = new
AlertDialog.Builder(CalculadoraActivity.this);
```

Cria a instância da classe **AlertDialog.Builder** que será representada e guardada dentro da variável *dialogo*. Na linha seguinte:

```
dialogo.setTitle("Resultado soma");
```

Define o título da caixa de diálogo através do método **setTitle**. Na linha seguinte:

```
dialogo.setMessage("A soma é " + soma);
```

Define a mensagem a ser exibida através do método **setMessage**. Na linha seguinte:

```
dialogo.setNeutralButton("OK", null);
```

Define o botão “OK” da caixa de texto através do método **setNeutralButton**. O parâmetro **null** indica que nenhuma ação será executada quando o botão for clicado (simplesmente a caixa será fechada e nada mais). E para finalizar:

```
dialogo.show();
```

Que é responsável por “exibir” a mensagem na tela por imediato.

Agora vamos continuar as outras operações certo ? Retornaremos então para a tela da nossa aplicação e vamos adicionar mais 3 botões referentes as operações restantes. Vamos adicionar na tela mais três botões como segue (um em baixo do outro, conforme a sequência abaixo):

Button

Propriedade	Valor
Id	btsubtrair
text	Subtrair
layout:width	match_parent

Button

Propriedade	Valor
id	btmultiplicar
text	Multiplicar
layout:width	match_parent

Button

Propriedade	Valor
id	btdividir
text	Dividir
layout:width	match_parent

Depois de “finalizado” o que foi se pedido acima, veja como ficou a tela da nossa aplicação:

Tela da aplicação da calculadora

Conforme já havia falado, a tela da aplicação nada mais é do que uma estrutura XML. Vamos ver agora a estrutura XML que existe por trás dessa tela que acompanhamos na figura acima:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".CalculadoraActivity">

    <TextView android:text="Digite o primeiro número"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:id="@+id/textView" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ednumero1"
        android:layout_below="@+id/textView"
```

```
        android:layout_alignParentLeft="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Digite o segundo número"
        android:id="@+id/textView2"
        android:layout_below="@+id/ednumero1"
        android:layout_alignParentLeft="true" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ednumero2"
        android:layout_below="@+id/textView2"
        android:layout_alignParentLeft="true" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Somar"
        android:id="@+id/btsomar"
        android:layout_below="@+id/ednumero2"
        android:layout_alignParentLeft="true" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Subtrair"
        android:id="@+id/btsubstrair"
        android:layout_below="@+id/btsomar"
        android:layout_alignParentLeft="true" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Multiplicar"
        android:id="@+id/btmultiplicar"
        android:layout_below="@+id/btsubstrair"
        android:layout_alignParentLeft="true" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Dividir"
        android:id="@+id/btdividir"
        android:layout_below="@+id/btmultiplicar"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

Agora retornando para o código do arquivo “CalculadoraActivity.java”, vamos declarar mais três atributos (variáveis) que vão corresponder aos botões que representam as operações restantes, conforme destaca a **linha em azul**:

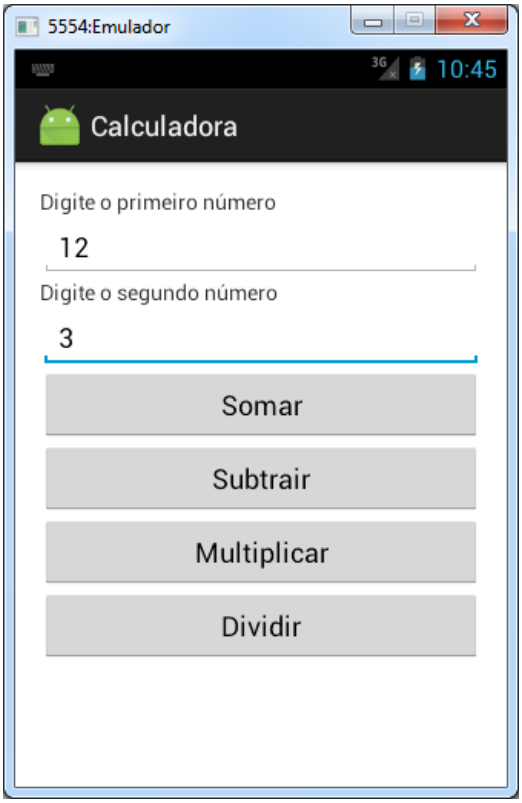
```
:
Button btsomar, btsubtrair,btmultiplicar, btdividir;
:
```

Agora vamos atribuir para cada botão um evento de clique, fazendo com que eles efetuem a sua respectiva operação aritmética. Vamos continuar a codificação do método **onCreate** , digitando o seguinte código **destacado em azul** em seguida:

```
protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_calculadora);

        ednumero1 = (EditText) findViewById(R.id.ednumero1);
        ednumero2 = (EditText) findViewById(R.id.ednumero2);
        btsomar  = (Button) findViewById(R.id.btsomar);

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

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

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

        btsomar.setOnClickListener(new View.OnClickListener() {

            :

        });

        btsubtrair.setOnClickListener(new View.OnClickListener() {

           @Override
           public void onClick(View arg0) {

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

             double num2 = Double.parseDouble
             (ednumero2.getText().toString());

             double soma = num1 - num2;

             AlertDialog.Builder dialogo = new
             AlertDialog.Builder(CalculadoraActivity.this);

             dialogo.setTitle("Resultado subtração");

             dialogo.setMessage("A subtração é " + soma);

             dialogo.setNeutralButton("OK", null);
```

```
            dialogo.show();

        }
    });
    btmultiplicar.setOnClickListener(new View.

      OnClickListener() {

        @Override
        public void onClick(View arg0) {

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

            double num2 = Double.parseDouble
            (ednumero2.getText().toString());

            double soma = num1 * num2;

            AlertDialog.Builder dialogo = new
            AlertDialog.Builder(CalculadoraActivity.this);

            dialogo.setTitle("Resultado multiplicação");
            dialogo.setMessage("A multiplicação é " + soma);
            dialogo.setNeutralButton("OK", null);

            dialogo.show();

        }
    });

    btdividir.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View arg0) {

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

            double num2 = Double.parseDouble
            (ednumero2.getText().toString());

            double soma = num1 / num2;

            AlertDialog.Builder dialogo = new
            AlertDialog.Builder(CalculadoraActivity.this);

            dialogo.setTitle("Resultado divisão");
            dialogo.setMessage("A divisão é " + soma);
            dialogo.setNeutralButton("OK", null);

            dialogo.show();
    } });
}
```

Depois de escrever o código acima, salve o arquivo e em seguida teste a aplicação. Veja o resultado na figura seguinte:



Aplicação da calculadora em execução

5.2) Desenvolvendo uma aplicação simples de compras

Agora para aprimorarmos o nosso conhecimento no desenvolvimento de aplicações para Android, vamos criar um outro aplicativo que consiste em um sistema de compras, bem simples. Em nossa aplicação terei disponível 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.

Na aplicação que iremos desenvolver vamos utilizar os seguintes widgets : **TextView**, **CheckBox** e **Button**.

Bom, vamos criar um novo projeto no Eclipse para Android chamado “SistemaDeCompras”. Siga os dados do projeto abaixo:

Application Name : Sistema de Compras

Company Domain : app.usuario

Project location : (Fica ao seu critério onde salvar)

Activity Name: ComprasActivity

Layout Name : activity_compras

Title : Sistema de Compras

Resource Menu Name : menu_compras

Depois de carregado e criado o projeto modifique o componente **TextView** situado na tela, de acordo com a tabela abaixo:

TextView

Propriedade	Valor
text	Escolha seu produto

Feito o que se foi pedido, adicione os seguintes componentes na sequência:

CheckBox

Propriedade	Valor
text	Arroz (R$ 2,69)
id	Chkarroz

CheckBox

Propriedade	Valor
text	Leite (R$ 5,00)
id	Chkleite

CheckBox

Propriedade	Valor
text	Carne (R$ 9,70)
id	Chkcarne

CheckBox

Propriedade	Valor
text	Feijão (R$ 2,30)
id	Chkfeijao

Button

Propriedade	Valor
text	Total das compras
id	bttotal
layout:width	match_parent

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

Layout da tela da aplicação

Vamos ver agora a estrutura XML da tela desta aplicação:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".ComprasActivity">
    <TextView android:text="Escolha o seu produto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView" />
    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Arroz (R$ 2,69)"
        android:id="@+id/chkarroz"
        android:layout_below="@+id/textView"
        android:layout_alignParentLeft="true" />
    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Leite (R$ 5,00)"
        android:id="@+id/chkleite"
        android:layout_below="@+id/chkarroz"
        android:layout_alignParentLeft="true" />

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Carne (R$ 9,70)"
        android:id="@+id/chkcarne"
        android:layout_below="@+id/chkleite"
        android:layout_alignParentLeft="true" />

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Feijão (R$ 2,70)"
        android:id="@+id/chkfeijao"
        android:layout_below="@+id/chkcarne"
        android:layout_alignParentLeft="true" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Total das compras"
        android:id="@+id/bttotal"
        android:layout_below="@+id/chkfeijao"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

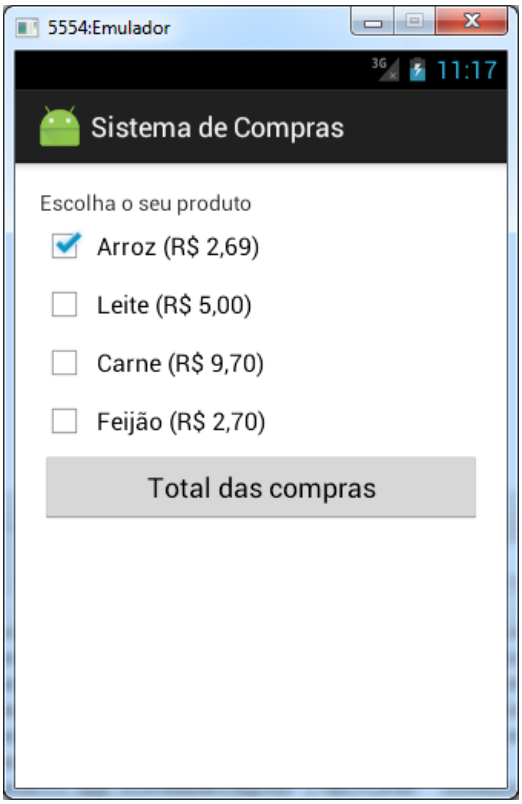
Agora vamos modificar o arquivo “ComprasActivity.java“. O código “completo” desse arquivo será como o código que é exibido abaixo:

```
package usuario.app.sistemadecompras;

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

public class ComprasActivity extends Activity {

    CheckBox chkarroz, chkleite, chkcarne, chkfeijao;

    Button bttotal;

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

        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() {

            @Override
            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(
                ComprasActivity.this);

                dialogo.setTitle("Aviso");
                dialogo.setMessage("Valor total da compra :"
                + String.valueOf(total));

                dialogo.setNeutralButton("OK", null);
                dialogo.show();

            }
        });

    }
}
```

Agora vou descrever o código situado no método **onClick**. Dentro do método 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 na variável *total*. No final é exibido o valor total das compras na tela.

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



Aplicação simples de compras em execução

5.3) Desenvolvendo uma aplicação de cálculo de salário

Agora vamos desenvolver uma nova aplicação que vai consistir em um sistema onde nós vamos digitar o salário de um funcionário permitindo escolher o seu percentual de aumento, que pode ser de 40% , 45% e 50%. Ao final de tudo, o sistema irá mostrar o salário reajustado com o novo aumento.

Para essa aplicação vamos utilizar os seguintes widgets : **TextView, EditText, RadioButton** e **Button**.

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

Application Name: Calculo de Salario

Company Domain : app.usuario

Project location : (Fica a sua escolha)

Activity Name: SalarioActivity

Layout Name : activity_salario

Title : Cálculo de Salário

Resource Menu Name : menu_salario

Depois de carregado e criado o projeto modifique o componente **TextView** situado na tela, de acordo com a tabela abaixo:

TextView

Propriedade	Valor
text	Digite seu salário

Em seguida, adicione os seguintes componentes na sequência:

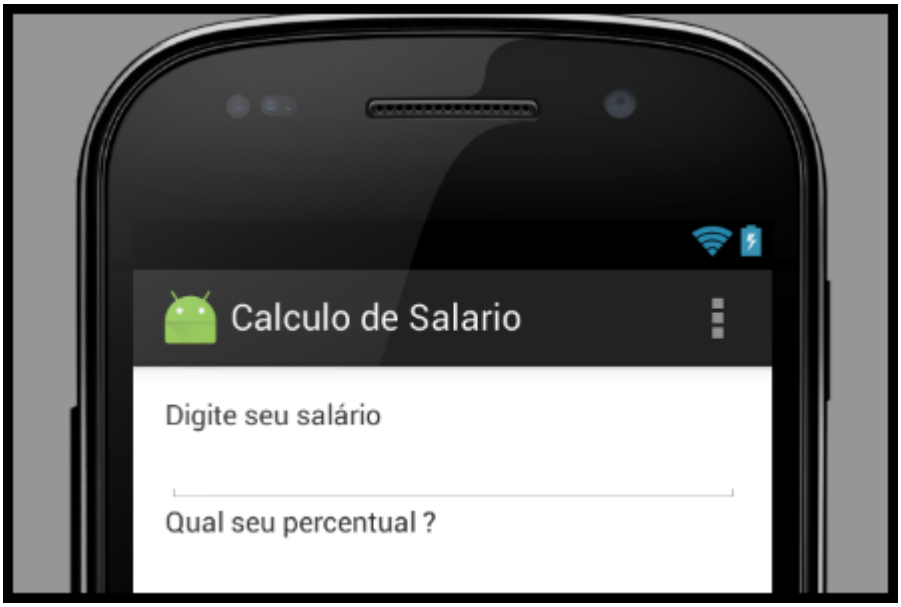
EditText (Number Decimal)

Propriedade	Valor
text	
id	edsalario
layout:width	match_parent

TextView

Propriedade	Valor
text	Qual é o seu percentual ?

Seguindo os passos acima ate aqui, a aplicação deve estar de acordo com o dá figura abaixo:



Tela de layout da aplicação Cálculo de salário

Bom, agora vamos adicionar um componente, ou melhor, uma estrutura que será responsável por agrupar as RadioButtons dentro dela, que se chama **RadioGroup** (Para mais informações veja o Capítulo 4). Clique e arraste o componente abaixo do ultimo widget adicionado. O resultado você confere na figura abaixo:

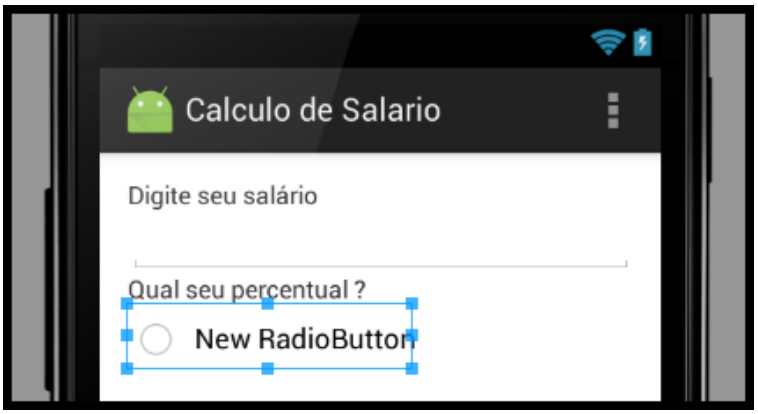
Estrutura RadioGroup inserida

Com o **RadioGroup** selecionado, modifique as propriedades abaixo:

RadioGroup

Propriedade	Valor
layout:width	match_parent
id	rgopcoes

Observe que o **RadioGroup** ESTÁ VAZIO. Vamos arrastar para dentro dele três elementos, cada um deles do tipo **RadioButton** e identificados por um nome. Vamos arrastar o primeiro **RadioButton** para dentro do componente **RadioGroup**. Vejamos o resultado :



Agora vamos alterar as seguintes propriedades do **RadioButton**, conforme mostra a tabela :

RadioButton

Propriedade	Valor
text	40%
id	rb40

Agora vamos arrastar mais dois componentes do tipo **RadioButton** para dentro do **RadioGroup,** e em seguida modifique as suas propriedades conforme mostra a tabela :

RadioButton

Propriedade	Valor
text	45%
id	rb45

RadioButton

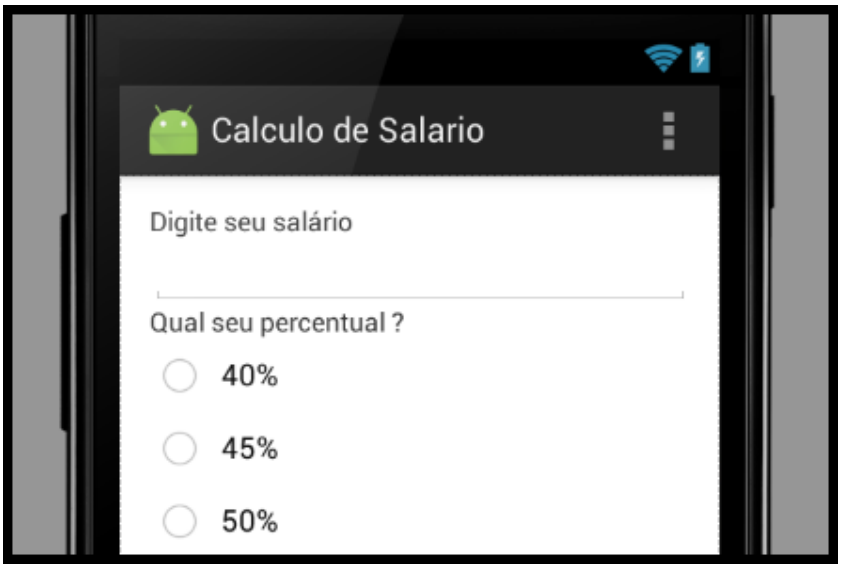
Propriedade	Valor
text	50%
id	rb50

Agora vamos selecionar novamente o componente **RadioGroup** para alterarmos a seguinte propriedade em seguida :

RadioGroup

Propriedade	Valor
layout:height	wrap_content

Vejamos o resultado:



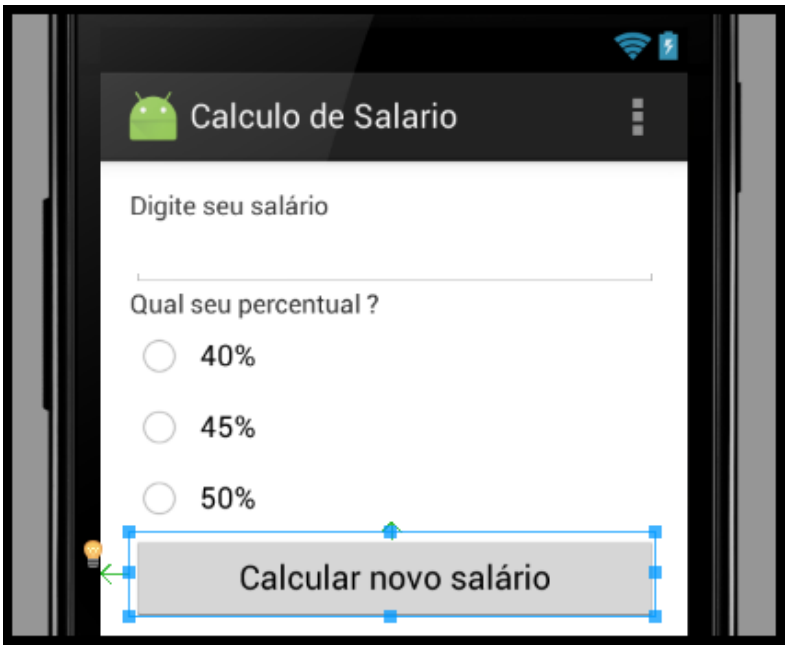
Tela da aplicação em construção

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** (e abaixo do mesmo).

Depois de colocar o **Button**, modifique as propriedades abaixo:

Propriedade	Valor
text	Calcular novo salário
id	btcalcular
layout:width	match_parent

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



Layout da tela da aplicação

Vamos analisar agora parte de um trecho de código produzido. Como havia falado acima, as **RadioButtons** precisam ficar dentro de uma estrutura chamada **RadioGroup** certo ? Vamos ver como isso é estruturado dentro de um código XML, como você confere abaixo:

```
<RadioGroup
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:id="@+id/rgopcoes">

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="40%"
            android:id="@+id/rb40" />
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="45%"
            android:id="@+id/rb45" />
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="50%"
            android:id="@+id/rb50" />
 </RadioGroup>
```

Observe acima que logo após a definição da estrutura **RadioGroup**, existe dentro dela as RadioButtons, que serão utilizadas na aplicação.

Agora confira a estrutura XML da tela da aplicação em desenvolvimento:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".SalarioActivity">

    <TextView android:text="Digite seu salário"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:ems="10"
        android:id="@+id/edsalario"
        android:layout_below="@+id/textView"
        android:layout_alignParentLeft="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Qual seu percentual ?"
        android:id="@+id/textView2"
        android:layout_below="@+id/edsalario"
        android:layout_alignParentLeft="true" />

    <RadioGroup
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:id="@+id/rgopcoes">

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="40%"
            android:id="@+id/rb40" />
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="45%"
```

```
            android:id="@+id/rb45" />
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="50%"
            android:id="@+id/rb50" />
    </RadioGroup>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Calcular novo salário"
        android:id="@+id/btcalcular"
        android:layout_below="@+id/rgopcoes"
        android:layout_alignParentLeft="true"
        />

</RelativeLayout>
```

No arquivo “SalarioActivity.java” vamos colocar o seguinte código abaixo:

```
package usuario.app.calculodesalario;

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

public class SalarioActivity extends Activity {

    RadioGroup rgopcoes;
    Button btcalcular;

    EditText edsalario;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_salario);


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

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

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

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

```
            @Override
            public void onClick(View arg0) {

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

                int op = rgopcoes.getCheckedRadioButtonId();

                double novo_salario = 0;

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

                AlertDialog.Builder dialogo = new
                AlertDialog.Builder(SalarioActivity.this);

                dialogo.setTitle("Novo salário");

                dialogo.setMessage("Seu novo salário é : R$"
                        + String.valueOf(novo_salario));

                dialogo.setNeutralButton("OK", null);

                dialogo.show();

            }
        });

    }


}
```

Vamos à explicação de alguns códigos interessantes. Dentro do método **onClick**, 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:

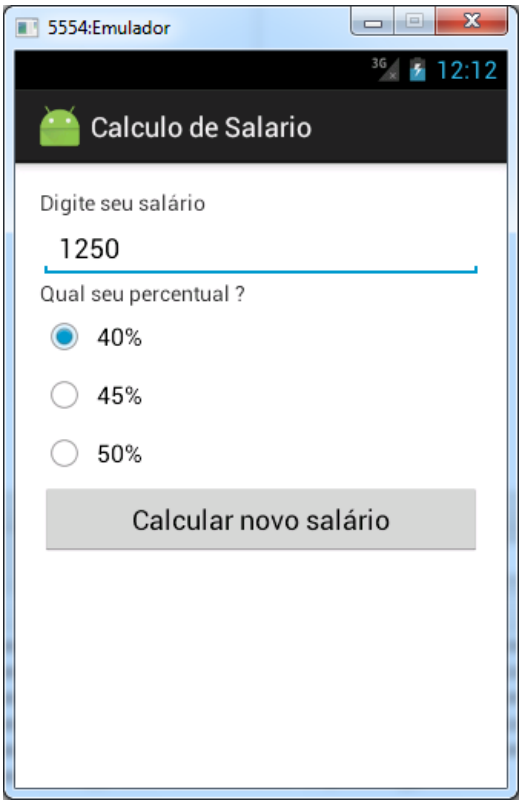
```
int op = rgopcoes.getCheckedRadioButtonId();
```

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 foi selecionada, realiza 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 seguinte:



Aplicação de cálculo de salário em execução

5.4) Desenvolvendo uma aplicação de lista de contatos

Agora vamos fazer uma nova aplicação em Android que consiste em uma aplicação de lista de contatos. Para essa aplicação iremos utilizar um componente chamado **ListView** , que seja bastante útil para esse tipo de situação (quando queremos exibir itens). Toda vez que clicarmos (ou melhor "tocarmos") em um contato na lista, será exibida uma mensagem com o nome do contato selecionado.

Vamos criar agora um novo projeto no Android Studio, conforme os dados em seguida:

Application Name: Lista de Contatos

Company Domain : app.usuario

Project location : (Fica a sua escolha)

Activity Name: ListaContatosActivity

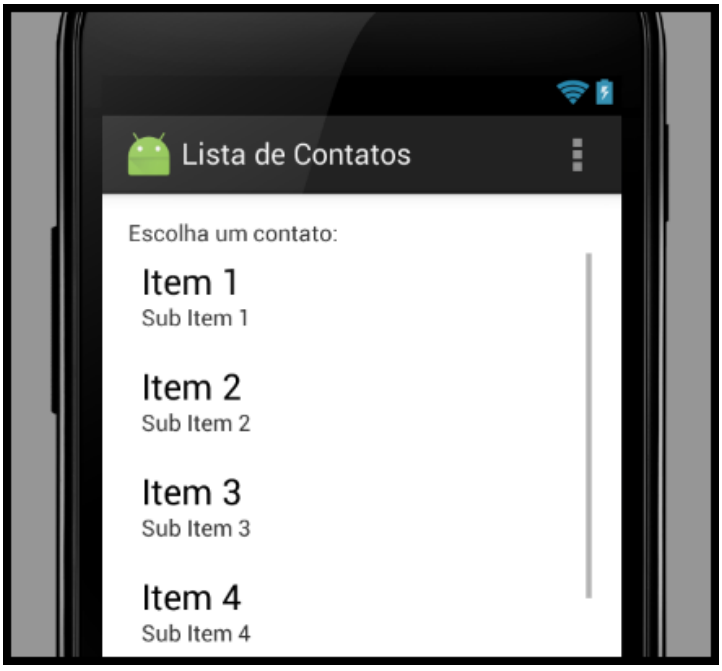
Layout Name : activity_lista_contatos

Title : Lista de Contatos

Resource Menu Name : menu_lista_contatos

Após o projeto ser carregado selecione o componente **TextView**, e em seguida dentro da sua propriedade “text” digite a seguinte frase : “Escolha um contato:”.

Em seguida vamos adicionar o componente **ListView** (que se encontra na seção “Containers”). Seguindo o que foi se pedido, a tela da aplicação ficará de acordo com a seguinte figura:



Layout da tela da aplicação em desenvolvimento

Agora vamos criar um objeto (String Array) que vai armazenar os contatos que serão exibidos no componente, que iremos chamado de “contatos” (criar no

arquivo “strings.xml”). Os contatos que estarão nessa lista são : "Aline","Lucas","Rafael","Gabriela" e "Silvana".

Depois de criar os contatos, selecione o objeto **ListView** que você adicionou e altere as seguintes propriedades.

ListView

Propriedade	Valor
id	lista_contatos
entries	@array/contatos

Agora vamos visualizar a estrutura XML do arquivo que forma a tela da aplicação acima:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".ListaContatosActivity">
    <TextView android:text="Escolha um contato:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_below="@+id/textView"
        android:layout_alignParentLeft="true"
        android:entries="@array/contatos" />

</RelativeLayout>
```

Agora vamos no arquivo “ListaContatosActivity.java” para colocar o seguinte código abaixo:

```
package usuario.app.listadecontatos;

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

public class ListaContatosActivity extends Activity {
```

```
    ListView lista_contatos;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lista_contatos);
        lista_contatos = (ListView) findViewById(R.id.lista_contatos);
        lista_contatos.setOnItemClickListener(new
        AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> parent, View view,
                int position, long id) {

                    String nome = ((TextView) view).getText().toString();
                    AlertDialog.Builder dialogo = new AlertDialog.Builder(
                    ListaContatosActivity.this);
                    dialogo.setTitle("Contato");
                    dialogo.setMessage("Contato selecionado: " + nome);
                    dialogo.setNeutralButton("OK", null);
                    dialogo.show();
            }
      });
    }
}
```

Como havia falado (e também como vocês podem conferir no código acima), quando se clica em um item, o sistema mostra uma mensagem do item selecionado (no caso, o nome contato selecionado). Isso é conseguido fazendo uso da interface **OnItemClickListener**, como mostra a instrução abaixo:

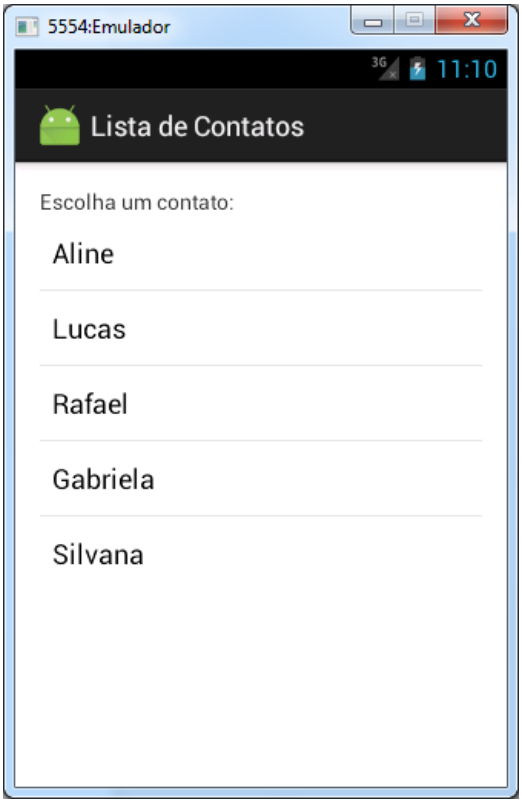
```
    lista_contatos.setOnItemClickListener(new
        AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> parent, View view,
                int position, long id) {

                    String nome = ((TextView) view).getText().toString();
                    AlertDialog.Builder dialogo = new AlertDialog.Builder(
                    ListaContatosActivity.this);
                    dialogo.setTitle("Contato");
                    dialogo.setMessage("Contato selecionado: " + nome);
                    dialogo.setNeutralButton("OK", null);
                    dialogo.show();
            }
    });
```

Vamos analisar alguns trechos do código. A linha de comando:

```
      String nome = ((TextView) view).getText().toString();
```

Guarda na variável *"nome"* o conteúdo retornado pelo objeto *"view"* (que contém o contato selecionado). Como o conteúdo precisa ser retornado para a variável que é do tipo **String**, foi preciso converte-lo em **TextView** para que o conteúdo fosse retornado em uma **String** (através do método **toString** situado em **getText**).

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



Aplicação de lista de contatos em execução

5.5) Desenvolvendo uma aplicação que visualiza imagens

Agora vamos desenvolver uma aplicação básica que visualiza imagens através do uso o componente **ImageView**. Vamos criar um projeto com os seguintes dados abaixo:

Application Name: Visualizador de Imagens

Company Domain : app.usuario

Project location : (Fica a sua escolha)

Activity Name: VisualizadorImagensActivity

Layout Name : activity_visualizador_imagens

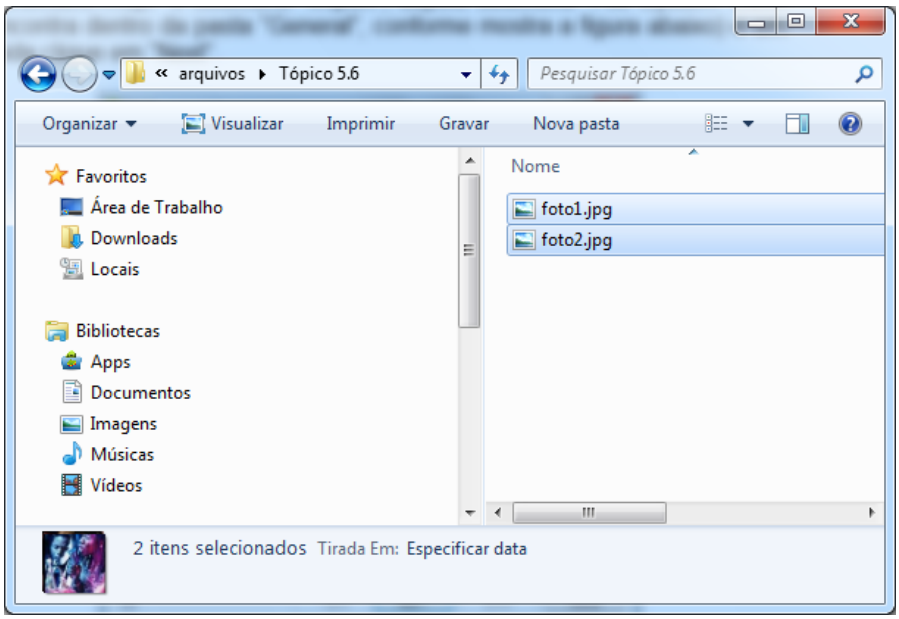
Title : Visualizador de Imagens

Resource Menu Name : menu_ visualizador_imagens

Antes de iniciarmos a codificação do programa, quero que você coloque duas imagens JPEG (com a extensão .jpg), dentro da pasta “drawable”, presente dentro do diretório “res” (para esse projeto usei duas imagens chamadas “foto1.jpg” e “foto2.jpg”, que já acompanham este projeto, presente dentro do diretório “Tópico 5.5”).

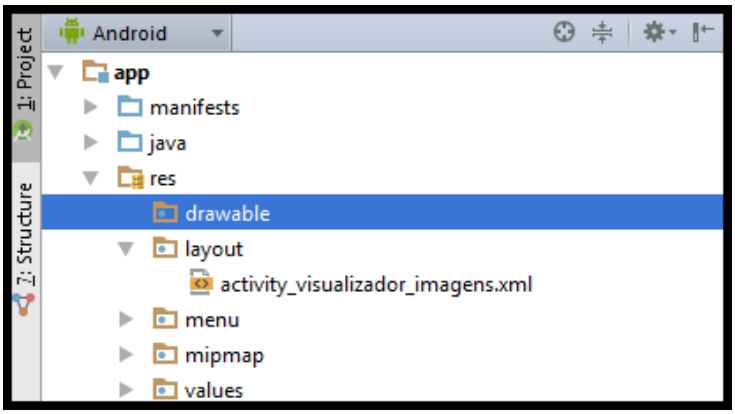
Como colocar as imagens dentro do diretório “drawable” ?

Vamos através do Windows Explorer abrir o diretório onde se encontra esses arquivos, e em seguida selecione-os , pressionando as teclas CTRL+C.



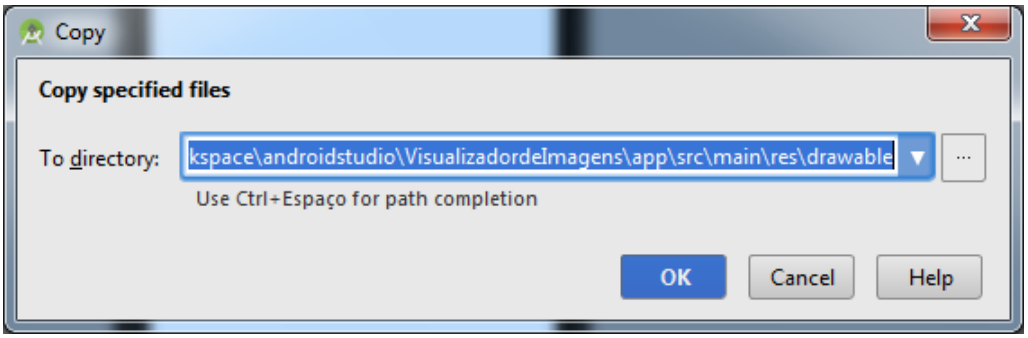
Selecionando e copiando os arquivos

Depois de copiarmos os arquivos de imagem vamos voltar para o Android Studio para o nosso projeto aberto, e em seguida, selecione a pasta “drawable”:

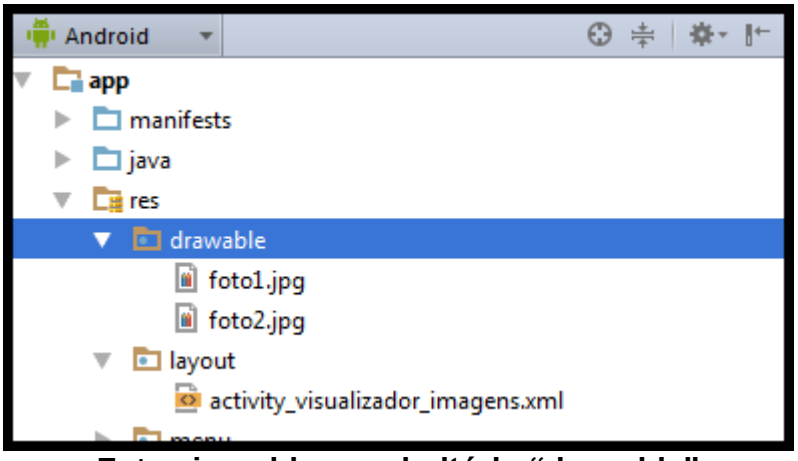


Selecionando a pasta “drawable”

Com a pasta selecionada, pressione CTRL+V, e em seguida surgirá a seguinte caixa de diálogo :



Para finalizar clique em “OK” para que as imagens possam ser adicionadas no diretório “drawable”.



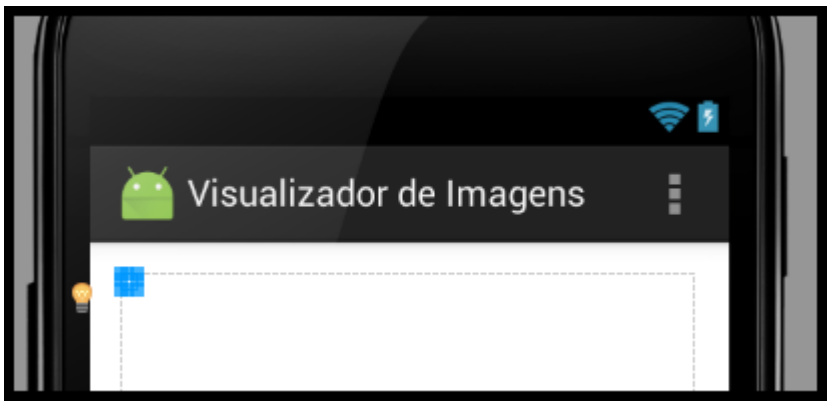
Fotos inseridas no dreitório “drawable”

Agora vamos remover o componente **TextView** da tela do nosso dispositivo e em seguida vamos adicionar dentro da tela da nossa aplicação uma estrutura

LinearLayout (Horizontal), que se encontra na guia “Layouts”, simplesmente arrastando o componente para a tela da aplicação. O resultado você confere na figura abaixo:

Estrutura LinearLayout inserida

Agora dentro da estrutura **LinearLayout** que adicionamos acima, vamos inserir o componente **ImageView** (que se encontra na guia “Widgets”). Vejamos o resultado:



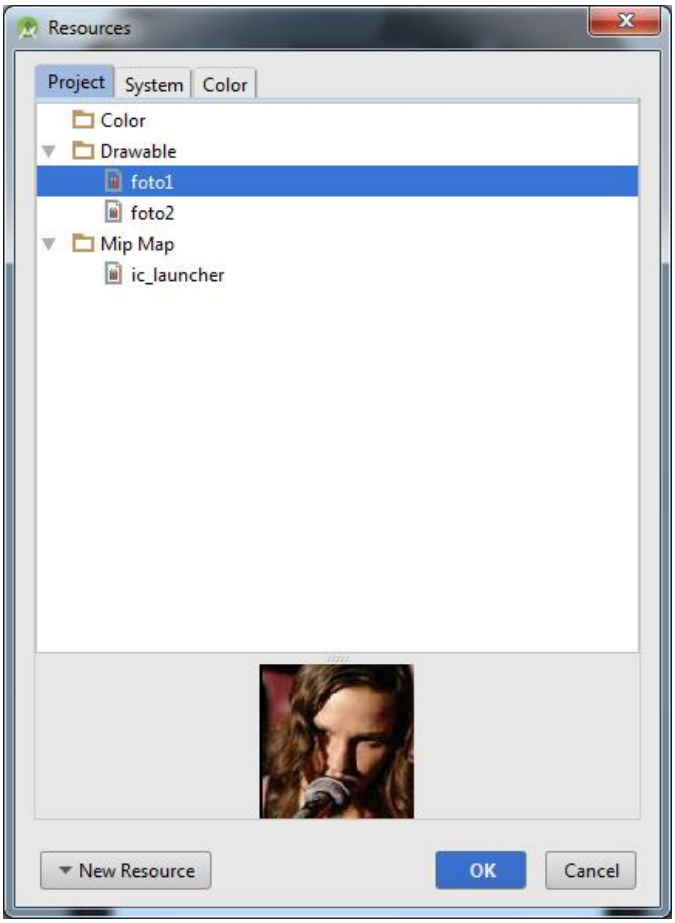
Componente ImageView inserido

Observe que nós podemos ver um pequeno quadrinho minúsculo representando o componente **ImageView**. Isso ocorre pelo fato de o mesmo não possuir nenhuma imagem.

Na guia “Properties” selecione a propriedade “src”, e em seguida clique no botão (...), conforme indica a figura seguinte :

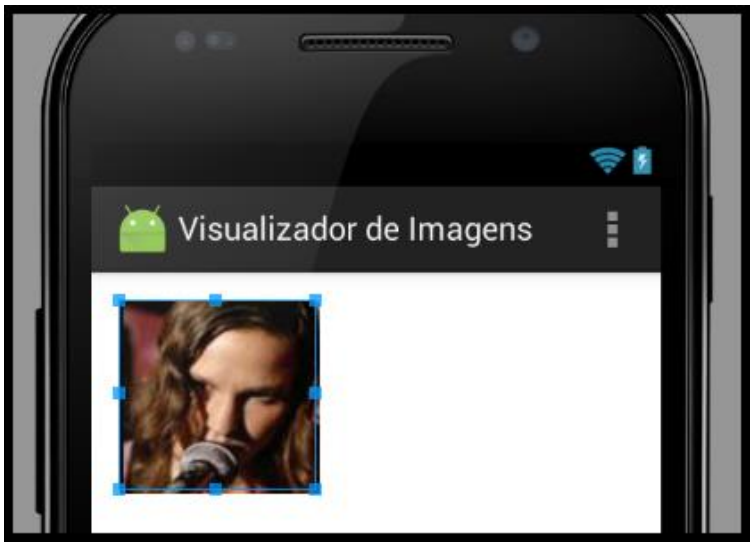
Inserindo uma imagem

Na caixa de dialogo que se abre, expanda o item drawable (se não estiver expandido) e selecione a imagem da “foto1.jpg” :



Caixa de diálogo – Resource

Nesta caixa de diálogo escolhemos a imagem que o nosso componente vai assumir. Depois de escolher a imagem clique em “OK”. Veja o resultado em seguida:



Resultado da operação

Agora vamos alterar a propriedade do componente **ImageView** conforme abaixo:

ImageView

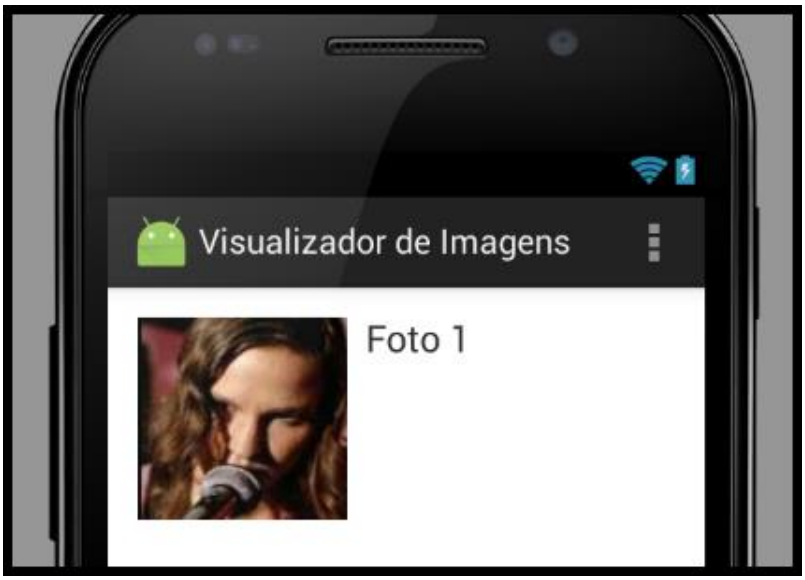
Propriedade	Valor
id	Imgfoto

Agora vamos adicionar um componente **TextView** que ficará ao lado da imagem. Altere suas propriedades conforme a tabela abaixo:

TextView

Propriedade	Valor
id	Txtinformacao
text	Foto 1
textSize	20dp
padding / left	10dp

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



Layout da aplicação

Agora vamos selecionar a estrutura **LinearLayout** que adicionarmos (onde inserimos a imagem e o título acima) para modificarmos a seguinte propriedade :

LinearLayout

Propriedade	Valor
layout:height	wrap_content

Agora vamos adicionar na sequência dois componentes do tipo **Button**, só que esses dois componentes vão estar dentro da tela da aplicação e fora (e também abaixo) da estrutura de layout que adicionamos. Segue abaixo as propriedades que precisam ser modificadas:

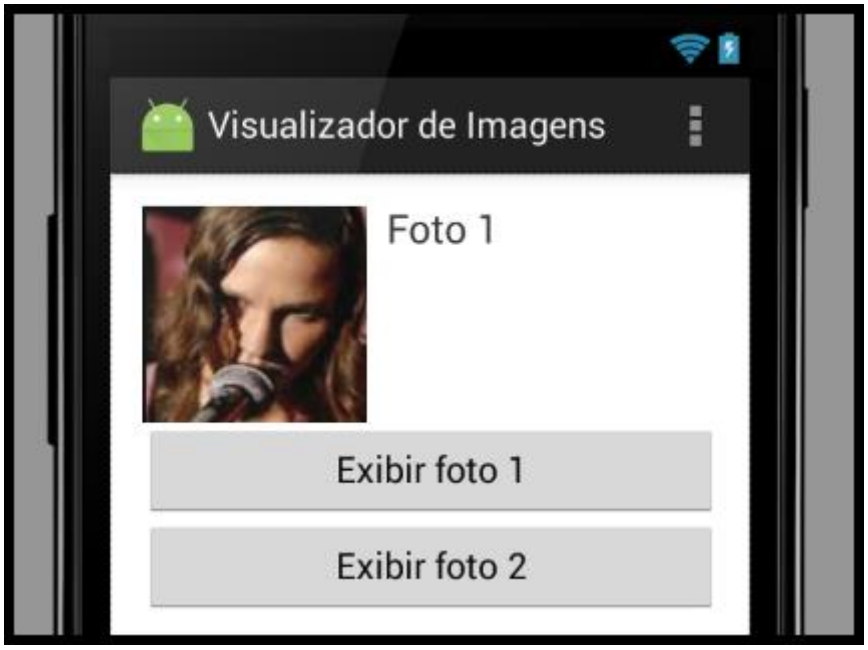
Button

Propriedade	Valor
id	btfoto1
text	Exibir foto 1
layout:width	match_parent

Button

Propriedade	Valor
id	btfoto2
text	Exibir foto 2
layout:width	match_parent

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



Layout da aplicação

Vamos conferir agora como ficou o código XML da tela da nossa aplicação :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".VisualizadorImagensActivity">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:id="@+id/linearLayout">

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/imgfoto"
            android:src="@drawable/foto1" />
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Foto 1"
            android:id="@+id/txtinformacao"
            android:textSize="20sp"
            android:paddingLeft="10dp" />
    </LinearLayout>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Exibir foto 1"
        android:id="@+id/btfoto1"
        android:layout_below="@+id/linearLayout"
        android:layout_alignParentLeft="true" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Exibir foto 2"
        android:id="@+id/btfoto2"
        android:layout_below="@+id/btfoto1"
        android:layout_alignParentLeft="true" />
</RelativeLayout>
```

Agora vamos no arquivo “VisualizadorImagensActivity.java” para colocarmos o código em seguida :

```
package usuario.app.visualizadordeimagens;



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

public class VisualizadorImagensActivity extends Activity {

    ImageView imgfoto;

    Button btfoto1, btfoto2;

    TextView txtinformacao;

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

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

```
        imgfoto = (ImageView) findViewById(R.id.imgfoto);

        btfoto1 = (Button) findViewById(R.id.btfoto1);
        btfoto2 = (Button) findViewById(R.id.btfoto2);

        txtinformacao = (TextView) findViewById(R.id.txtinformacao);

        btfoto1.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View arg0) {

                imgfoto.setImageResource(R.drawable.foto1);

                txtinformacao.setText("Foto 1");

            }
        });

        btfoto2.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View arg0) {

                imgfoto.setImageResource(R.drawable.foto2);
                txtinformacao.setText("Foto 2");

            }
        });

    }

}
```

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

```
imgfoto.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:

```
txtinformacao.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 à abertura da segunda imagem.

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

Aplicação de visualização de imagens em execução

Capitulo 6 Trabalhando com mais de uma tela em uma aplicação

Até agora as aplicações que desenvolvemos tinham somente uma única tela, mas, sabemos que algumas aplicações possuem normalmente mais de uma tela. A partir de agora iremos aprender como inserir e gerenciar várias telas em uma aplicação Android através dos exemplos que serão demonstrados nesse capítulo.

Para começarmos, vamos criar um novo projeto Android com os seguintes dados abaixo:

Application Name: Troca de Telas

Company Domain : app.usuario

Project location : (Fica a sua escolha)

Activity Name: TrocaTelasActivity

Layout Name : tela_principal

Title : Troca de Telas

Resource Menu Name : menu_ tela_principal

Altere a estrutura de layout da sua aplicação para o **LinearLayout** e em seguida altere o componente **TextView** de acordo com a tabela abaixo.

TextView

Propriedade	Valor
text	Você está na tela principal

Agora adicione um componente **Button** e modifique as seguintes propriedades:
Button

Propriedade	Valor
id	bttela2
layout:width	match_parent
text	Ir pata tela 2

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

Layout da tela 1

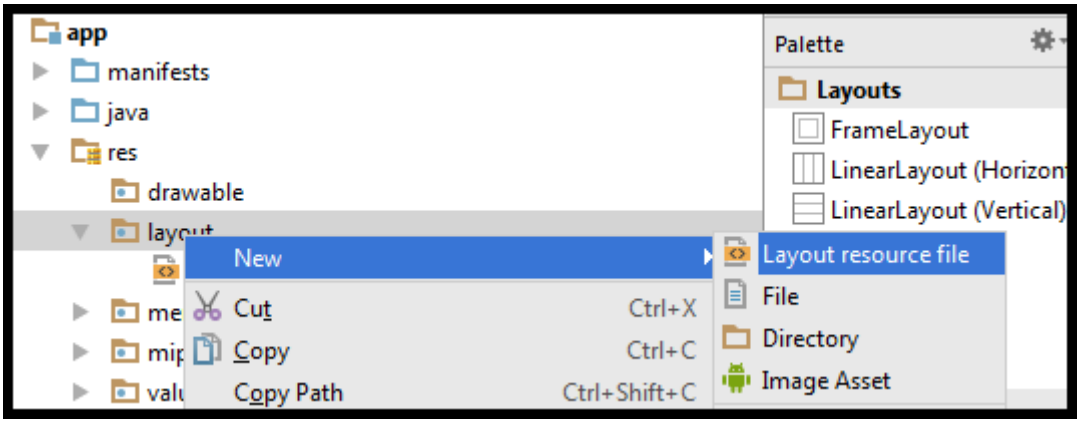
Vejamos agora o código XML da tela da nossa aplicação:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".TrocaTelasActivity">

    <TextView android:text="Você está na tela principal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView" />
```

```
<Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ir para tela 2"
        android:id="@+id/bttela2"
        android:layout_below="@+id/textView"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

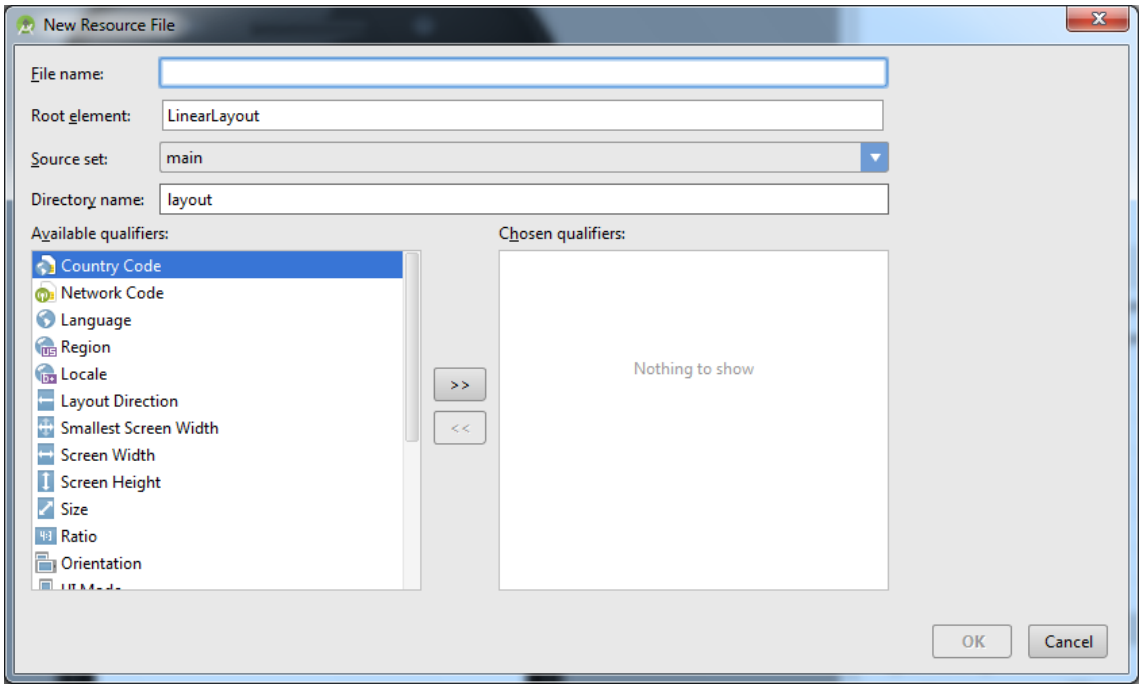
Nossa primeira tela está pronta, muito bem! Agora vamos criar uma nova tela para a nossa aplicação. O nome do arquivo que vai representar a segunda tela da nossa aplicação vai se chamar "tela2.xml" (um arquivo XML). Conforme já foi explicado (e explico novamente aqui), todos os arquivos que representam a tela da aplicação devem estar dentro do diretório "layout" (situado dentro da pasta "res" do projeto), logo, vamos criar o nosso arquivo dentro desse diretório.

Para criarmos um novo arquivo XML dentro do diretório "layout" basta clicar com o botão direito sobre a pasta "layout" e em seguida selecionar "New" e em seguida clicar em "Layout Resource File". Veja na figura abaixo :



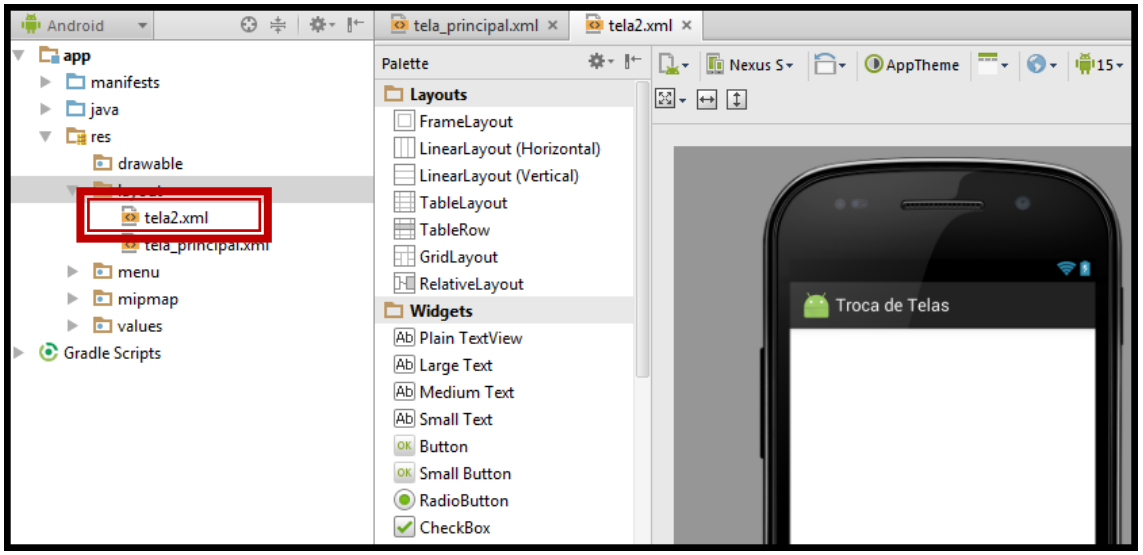
Criando um novo arquivo de layout

Feito isso surgirá uma caixa de diálogo, conforme mostra a figura a seguir :



New Resource File

Por padrão a estrutura de layout padrão selecionada é **LinearLayout** (definida em "Root element"). Vamos no campo "Root element" para substituir **LinearLayout** por **RelativeLayout**. Feito isso , vamos no campo "File name" para digitar "tela2" (que será o nome do nosso arquivo XML) e em seguida clique em "OK" para que o arquivo seja gerado. Veja o resultado na figura seguinte:



Tela de layout em branco

No componente **RelativeLayout** vamos modificar as seguintes propriedades a seguir:

RelativeLayout

Propriedade	Valor
padding / right	@dimen/activity_horizontal_margin
padding / left	@dimen/activity_horizontal_margin
padding / top	@dimen/activity_vertical_margin
padding / bottom	@dimen/activity_vertical_margin

Agora vamos adicionar os seguintes componentes, na sequência:

TextView

Propriedade	Valor
text	Você está na tela 2

Button

Propriedade	Valor
id	bttelaprincipal
padding:width	match_parent
text	Ir pata tela principal

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

Layout da tela 2

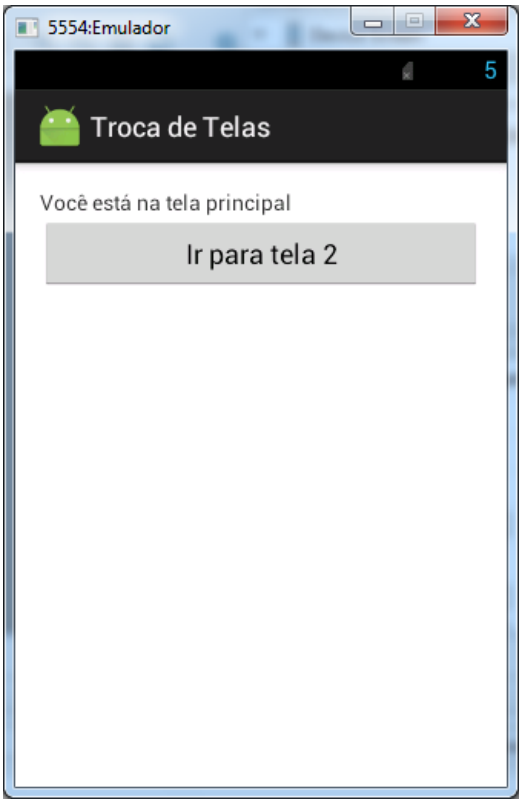
Vejamos agora o código XML da tela da nossa aplicação:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Você está na tela 2"
        android:id="@+id/textView2"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ir para tela principal"
        android:id="@+id/bttelaprincipal"
        android:layout_below="@+id/textView2"
        android:layout_alignLeft="@+id/textView2" />
</RelativeLayout>
```

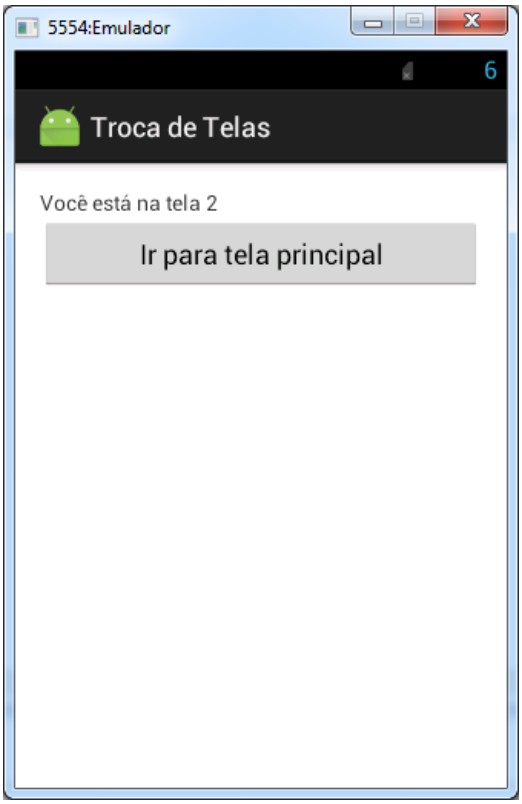
Agora vamos no arquivo “TrocaTelasActivity.java” para digitarmos o código em seguida:

```
package usuario.app.trocadetelas;

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

public class TrocaTelasActivity extends Activity {

    Button bttelaprincipal, bttela2;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        CarregarTelaPrincipal();
    }
    public void CarregarTelaPrincipal()
    {
       setContentView(R.layout.tela_principal);
       bttela2 = (Button) findViewById(R.id.bttela2);
       bttela2.setOnClickListener(new
        View.OnClickListener() {

          @Override
          public void onClick(View v) {

                CarregarTela2();

            }
          });
    }

    public void CarregarTela2()
    {
       setContentView(R.layout.tela2);
       bttelaprincipal = (Button) findViewById
      (R.id.bttelaprincipal);
       bttelaprincipal.setOnClickListener(new
        View.OnClickListener() {

              @Override
              public void onClick(View v) {

                 CarregarTelaPrincipal();

              }
          });
    }
}
```

Observem que nesta classe eu criei dois métodos : **CarregaTelaPrincipal** e **CarregaTela2**. Para toda aplicação que utilize mais de um layout (tela), 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 em execução (na tela principal)



Aplicação em execução (na segunda tela)

6.1) Desenvolvendo um Sistema de Cadastro (Primeira versão)

Com o que já aprendemos neste capítulo (e também nós capítulos e tópicos anteriores) já podemos desenvolver agora uma aplicação mais interessante. Para isso, vamos criar um sistema de cadastro de usuários, onde através dele iremos cadastrar as seguintes informações: nome, telefone e endereço.

Essa aplicação vai possuir três telas, cada uma com as seguintes funcionalidades:

Tela principal (tela com opções): Nessa tela da aplicação teremos um menu que dará acesso ao cadastro do usuário e a visualização dos usuários cadastrados.

Tela de cadastro: Nesse tela o usuário irá preencher os campos solicitados pela aplicação e em seguida o mesmo poderá cadastrar para que as informações sejam registradas.

Tela de visualização de dados: Nessa tela poderão ser visualizados os dados (usuários) cadastrados. Se nada foi cadastrado na aplicação, será exibida uma mensagem informando essa situação.

Nessa primeira versão da nossa aplicação que iremos desenvolver, utilizaremos somente uma única Activity com vários arquivos XML (onde cada arquivo XML representa uma tela da nossa aplicação) que será gerenciados pela mesma.

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

Application Name: Sistema de Cadastro

Company Domain : app.usuario

Project location : (Fica a sua escolha)

Activity Name: MainActivity

Layout Name : tela_principal

Title : Sistema de Cadastro

Resource Menu Name : menu_ aplicacao

Dentro da pasta “drawable” vamos colocar uma imagem que acompanha este material (que se encontra dentro do diretório “Tópico 6.x”) chamada “icone_programa.png”.

Depois de colocar a imagem solicitada, abra o código XML do arquivo “tela_principal.xml” para adicionarmos o seguinte código XML abaixo :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >
    <LinearLayout
        android:id="@+id/layouttopo_tp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#fdb46d"
        android:orientation="horizontal" >
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/icone_programa" />
```

```
        <LinearLayout
            android:id="@+id/layout_titulo_programa_tp"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_gravity="center"
            android:gravity="center_vertical"
            android:orientation="vertical" >

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:gravity="center"
                android:text="Sistema de"
                android:textAppearance="?android:attr/textAppearanceLarge" />

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="Cadastro"
                android:textAppearance="?android:attr/textAppearanceLarge" />

        </LinearLayout>

    </LinearLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingBottom="15dp"
        android:paddingTop="15dp"
        android:text="Bem vindo ao Sistema de Cadastro. Esse é um pequeno
        software de cadastro de usuários. Escolha uma das opções abaixo :"
        android:textSize="18sp" />

    <Button
        android:id="@+id/btcadastrar_usuario"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Cadastrar Usuário" />

    <Button
        android:id="@+id/bt_listar_usuarios_cadastrados"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Listar usuários cadastrados" />
</LinearLayout>
```

Depois de colocarmos o código XML solicitado, teremos o seguinte resultado abaixo :

Layout da tela principal da aplicação

Agora vamos criar mais uma tela (arquivo de layout XML) para nossa aplicação, referente à tela de cadastro de usuário. Para isso vamos clicar com o botão direito sobre a pasta “layout” e em seguida vamos chamar o recurso de criação de arquivos de layout XML para Android (o “Layout Resource File”, conforme já foi mostrado anteriormente no último exemplo). O nome do nosso arquivo de layout XML vai ser “cadastro_de_usuario”.

Depois de criado o arquivo vamos abrir o editor de XML para digitarmos o seguinte código refente a tela de cadastro de usuários :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:id="@+id/layouttopo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#fdb46d"
        android:orientation="horizontal" >
```

```
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/icone_programa" />

    <LinearLayout
        android:id="@+id/layout_titulo_programa"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center"
        android:gravity="center_vertical"
        android:orientation="vertical" >

        <TextView

            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center"
            android:text="Cadastro de"
            android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="Usuário"
            android:textAppearance="?android:attr/textAppearanceLarge" />

    </LinearLayout>
</LinearLayout>

<TextView
    android:id="@+id/txtnome"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="10dp"
    android:text="Preencha os dados abaixo :"
    android:textAppearance="?android:attr/textAppearanceMedium" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="15dp"
    android:text="Nome :"
    android:textAppearance="?android:attr/textAppearanceMedium" />
<EditText
    android:id="@+id/ednome"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <requestFocus />
</EditText>
```

```
<TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="10dp"
        android:text="Endereço :"
        android:textAppearance="?android:attr/textAppearanceMedium" />
    <EditText
        android:id="@+id/edendereco"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="10dp"
        android:text="Telefone :"
        android:textAppearance="?android:attr/textAppearanceMedium" />
    <EditText
        android:id="@+id/edtelefone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <LinearLayout
        android:id="@+id/layout_botoes_cadastro"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center" >

       <Button
            android:id="@+id/btcadastrar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Cadastrar" />

        <Button
            android:id="@+id/btcancelar_cadastro"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Cancelar" />

    </LinearLayout>

</LinearLayout>
```

Depois de digitarmos o código solicitado, teremos a seguinte aparência abaixo :

Layout da tela de cadastro

Agora vamos criar mais uma tela (arquivo de layout XML) para nossa aplicação, que será referente à tela de listagem de usuários cadastrados. O nome da nossa tela vai se chamar “listagem_usuarios_cadastros”.

Após criar a tela da nossa aplicação, abra o editor de XML para digitarmos o seguinte código a seguir :

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

    <LinearLayout
        android:id="@+id/layouttopo_luc"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#fdb46d"
        android:orientation="horizontal" >

        <ImageView

            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:src="@drawable/icone_programa" />

        <LinearLayout
            android:id="@+id/layout_titulo_programa_luc"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_gravity="center"
            android:gravity="center_vertical"
            android:orientation="vertical" >

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:gravity="center"
                android:text="Listagem de "
                android:textAppearance="?android:attr/textAppearanceLarge" />

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="Usuários"
                android:textAppearance="?android:attr/textAppearanceLarge" />

        </LinearLayout>

    </LinearLayout>

    <LinearLayout
        android:id="@+id/layout_status_luc"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="right"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/txtstatus"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingRight="10dp"
            android:text="Registros : 1/10"
            android:textAppearance="?android:attr/textAppearanceMedium" />

    </LinearLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nome :"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/txtnome"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="5dp"
        android:text="[NOME]"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#fb7600" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="15dp"
        android:text="Endereço :"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/txtendereco"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="5dp"
        android:text="[ENDERECO]"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#fb7600" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="15dp"
        android:text="Telefone :"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/txttelefone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="5dp"
        android:text="[TELEFONE]"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#fb7600" />

    <LinearLayout
        android:id="@+id/layout_botoes_luc"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal"
        android:paddingTop="10dp" >

        <Button
            android:id="@+id/btanterior"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Anterior" />
```

```
        <Button
            android:id="@+id/btproximo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Próximo" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/layout_botao_fechar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="vertical" >

        <Button
            android:id="@+id/btfechar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Fechar" />

    </LinearLayout>

</LinearLayout>
```

Depois de digitarmos o código XML de nossa tela, teremos o seguinte resultado, conforme mostra a figura a seguir :

Layout da tela de listagem de usuários

Todos os registros dos usuários cadastrados estarão dentro de um objeto do tipo **ArrayList**. Para isso, iremos criar agora uma classe que irá armazenar (e retornar) os dados referentes ao usuário (nome, telefone e endereço) chamada **Registro**.

Essa classe vai estar dentro do mesmo pacote onde se encontra o arquivo "MainActivity.java" (neste caso, o primeiro pacote "usuario.app.sistemadecadastro"). Vamos criar agora uma classe (conforme já foi mostrado) chamada **Registro**, e em seguida vamos escrever o seu código conforme é mostrado em seguida:

```
package usuario.app.sistemadecadastro;

public class Registro {

	private String nome;
	private String endereco;
	private String telefone;
```

```
	public Registro(String nome, String endereco, String telefone)
	{
		this.nome = nome;
		this.endereco = endereco;
		this.telefone = telefone;
	}

	public String getNome() { return nome; }
	public String getTelefone() { return telefone; }
	public String getEndereco() { return endereco; }

}
```

Agora vamos abrir o arquivo “MainActivity.java” para digitarmos o seguinte código abaixo:

```
package usuario.app.sistemadecadastro;

import java.util.ArrayList;

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


public class MainActivity extends Activity {

	private ArrayList<Registro> aRegistro;
	TelaPrincipal tela_principal;
	TelaCadastroUsuario tela_cadastro;
	TelaListagemUsuarios tela_listagem;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		aRegistro = new ArrayList<Registro>();

		tela_principal = new TelaPrincipal(this);
		tela_cadastro = new TelaCadastroUsuario(this, tela_principal);
		tela_listagem = new TelaListagemUsuarios(this, tela_principal);
		tela_principal.setTelaCadastro(tela_cadastro);
		tela_principal.setTelaListagem(tela_listagem);

		tela_principal.CarregarTela();

	}

	public ArrayList<Registro> getRegistros() {

		return aRegistro;

	}
```

```
	public void ExibirMensagem(String msg) {
		AlertDialog.Builder d = new
		AlertDialog.Builder(MainActivity.this);
		d.setTitle("Aviso");
		d.setMessage(msg);
		d.setNeutralButton("OK", null);
		d.show();
	}

}
```

Muito provavelmente em seu editor de código do Android Developer Tools deverá aparecer inúmeros erros (sublinhados em vermelho e marcados com um “X”). Isso é pelo fato de o código presente nessa classe ainda não existir (ou seja, será criado posteriormente).

A classe **MainActivity** será responsável por inicializar o vetor onde serão armazenados os registros e carregar as telas da nossa aplicação (onde cada uma delas será gerenciada por uma classe). Vamos analisar o seguinte bloco de código em seguida :

```
	aRegistro = new ArrayList<Registro>();

	tela_principal = new TelaPrincipal(this);
	tela_cadastro = new TelaCadastroUsuario(this, tela_principal);
	tela_listagem = new TelaListagemUsuarios(this, tela_principal);

	tela_principal.setTelaCadastro(tela_cadastro);
	tela_principal.setTelaListagem(tela_listagem);


	tela_principal.CarregarTela();
```

Na primeira linha do trecho, iniciamos e criamos em memória o nosso **ArrayList** que armazenará todos os registros que serão gerenciados pelo nosso programa.

No trecho restante criamos as instâncias das classes responsáveis pelo gerenciamento de cada dela do programa (que AINDA VÃO SER CRIADAS, com seus respectivos métodos) , são elas : **TelaPrincipal**, **TelaCadastroUsuario** , **TelaListagemUsuarios**.

Na última instrução do trecho acima, carregamos a tela principal do programa, através do método **CarregarTela** da classe **TelaPrincipal** (método e classes que ainda serão criados).

Vamos criar agora uma classe (dentro do pacote "usuario.app.sistemadecadastro") chamada **TelaPrincipal** com o seguinte código a seguir :

```
package usuario.app.sistemadecadastro;

import android.view.View;
import android.widget.Button;

public class TelaPrincipal {

	MainActivity act;

	Button btcadastrar_usuario;
	Button bt_listar_usuarios_cadastrados;
	TelaCadastroUsuario tela_cadastro;
	TelaListagemUsuarios tela_listagem;

	public TelaPrincipal(MainActivity act) {

		this.act = act;

	}

	public void CarregarTela()
	{
		act.setContentView(R.layout.tela_principal);
		btcadastrar_usuario = (Button)
		act.findViewById(R.id.btcadastrar_usuario);
		bt_listar_usuarios_cadastrados = (Button)
		act.findViewById(R.id.bt_listar_usuarios_cadastrados);

		btcadastrar_usuario.setOnClickListener(new
		View.OnClickListener() {

			@Override
			public void onClick(View view) {

				tela_cadastro.CarregarTela();
			}
		});

		bt_listar_usuarios_cadastrados.setOnClickListener(new
		View.OnClickListener() {

			@Override
			public void onClick(View view) {

				tela_listagem.CarregarTela();
			}
		});
	}
```

```
	public void setTelaCadastro(TelaCadastroUsuario tela_cadastro)
	{
		this.tela_cadastro = tela_cadastro;
	}

	public void setTelaListagem(TelaListagemUsuarios tela_listagem)
	{
		this.tela_listagem = tela_listagem;
	}


}
```

Essa classe simplesmente tem a finalidade de mostrar a tela principal do programa, permitindo que o usuário escolha uma das opções da aplicação que permite cadastrar o usuário e listar os usuários cadastrados.

Agora vamos criar uma nova classe chamada **TelaCadastroUsuario** (dentro do pacote “usuario.app.sistemadecadastro”) com o seguinte código em seguida :

```
package usuario.app.sistemadecadastro;

import android.app.*;
import android.content.DialogInterface;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class TelaCadastroUsuario {

	MainActivity act;
	EditText ednome,edendereco, edtelefone;
	Button btcadastrar, btcancelar_cadastro;
	TelaPrincipal tela_principal;


   public TelaCadastroUsuario(MainActivity act, TelaPrincipal
   tela_principal)
   {
		this.act = act;
		this.tela_principal = tela_principal;
   }

    public void CarregarTela()
    {
      act.setContentView(R.layout.cadastro_de_usuarios);
      ednome = (EditText) act.findViewById(R.id.ednome);
      edtelefone = (EditText) act.findViewById(R.id.edtelefone);
```

```
        edendereco = (EditText) act.findViewById(R.id.edendereco);
        btcadastrar = (Button) act.findViewById(R.id.btcadastrar);
        btcancelar_cadastro = (Button)
         act.findViewById(R.id.btcancelar_cadastro);


        btcadastrar.setOnClickListener(new View.OnClickListener() {

              @Override
              public void onClick(View view) {

                    AlertDialog.Builder dialogo = new
                    AlertDialog.Builder(act);
                    dialogo.setTitle("Aviso");
                    dialogo.setMessage("Cadastrar usuário ?");
                    dialogo.setNegativeButton("Não", null);
                    dialogo.setPositiveButton("Sim", new
                    DialogInterface.OnClickListener() {

                          @Override
                          public void onClick(DialogInterface dialog,
                           int which) {

                             String nome =
                             ednome.getText().toString();
                             String telefone =
                             edtelefone.getText().toString();
                             String endereco =
                             edendereco.getText().toString();
                             act.getRegistros().add(new
                             Registro(nome,telefone,endereco));
                             act.ExibirMensagem("Cadastro efetuado com
                             sucesso.");
                             tela_principal.CarregarTela();

                    }

              });
             dialogo.show();
            }
});

        btcancelar_cadastro.setOnClickListener(new
         View.OnClickListener() {

              @Override
              public void onClick(View view) {


                    AlertDialog.Builder dialogo = new
                    AlertDialog.Builder(act);
                    dialogo.setTitle("Aviso");
                    dialogo.setMessage("Sair do cadastro ?");
                    dialogo.setNegativeButton("Não", null);
```

```
                    dialogo.setPositiveButton("Sim", new
                    DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog,
                    int which) {

                          tela_principal.CarregarTela();

                    }
              });
              dialogo.show();
        }
    });
 }

}
```

O código da classe **TelaCadastroUsuario** será responsável por efetuar os registros dos cadastros dos usuários na aplicação.

Agora vamos criar uma classe chamada **TelaListagemUsuarios** (dentro do pacote “usuario.app.sistemadecadastro”), com o seguinte código em seguida:

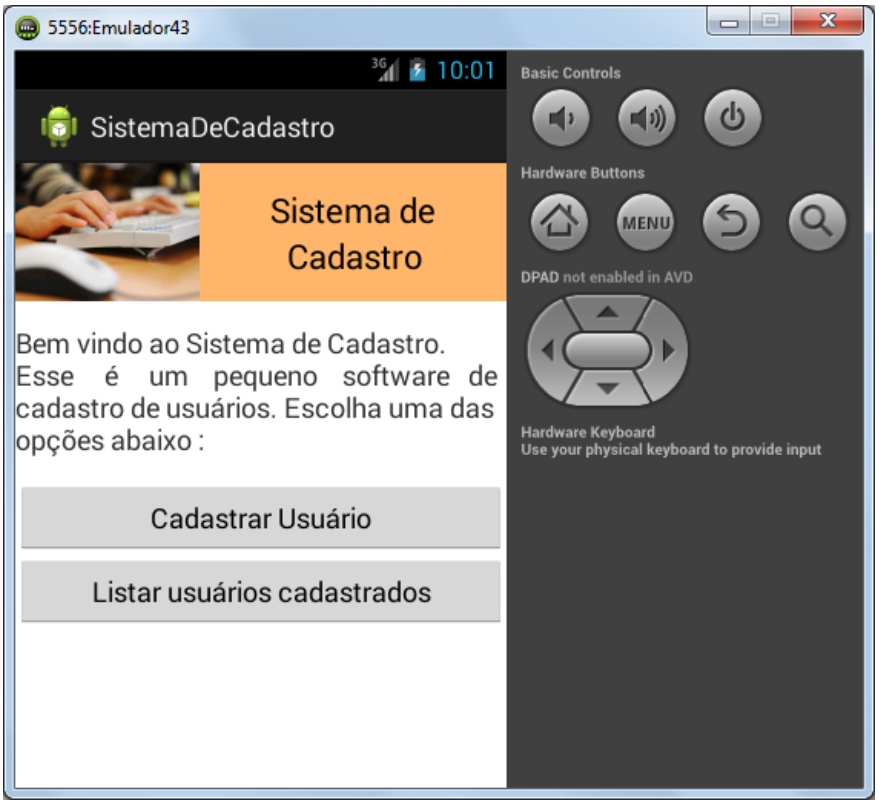
```
package usuario.app.sistemadecadastro;

import android.app.AlertDialog;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class TelaListagemUsuarios {


      MainActivity act;
      TelaPrincipal tela_principal;

      Button btanterior, btproximo, btfechar;

      TextView txtnome, txttelefone, txtendereco, txtstatus;

      int index;

      public TelaListagemUsuarios(MainActivity act,TelaPrincipal
       tela_principal) {

            this.act = act;
            this.tela_principal = tela_principal;
            index = 0;
      }
```

```
public void CarregarTela()
{
	//Antes de carregar a tela, verifica se existe registros
	//inseridos
	if(act.getRegistros().size() == 0)
	{
		(new AlertDialog.Builder(act))
		.setTitle("Aviso")
		.setMessage("Não existe nenhum registro cadastrado.")
		.setNeutralButton("OK", null)
		.show();

		return;
	}

	act.setContentView(R.layout.listagem_usuarios_cadastrados);
	btanterior = (Button) act.findViewById(R.id.btanterior);
	btproximo = (Button) act.findViewById(R.id.btproximo);
	btfechar = (Button) act.findViewById(R.id.btfechar);

	txtnome = (TextView) act.findViewById(R.id.txtnome);
	txtendereco = (TextView) act.findViewById(R.id.txtendereco);
	txttelefone = (TextView) act.findViewById(R.id.txttelefone);
	txtstatus = (TextView) act.findViewById(R.id.txtstatus);

	PreencheCampos(index);
	AtualizaStatus(index);

	btanterior.setOnClickListener(new View.OnClickListener() {

		@Override
		public void onClick(View view) {

			if(index > 0)
			{
				index--;
				PreencheCampos(index);
				AtualizaStatus(index);
			}
		}
	});
	btproximo.setOnClickListener(new View.OnClickListener() {

		@Override
		public void onClick(View view) {

			if(index < act.getRegistros().size() - 1)
			{
				index++;
				PreencheCampos(index);
				AtualizaStatus(index);
			}
```

```
                }
        });

        btfechar.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View view) {

                        tela_principal.CarregarTela();

                }
        });
    }

    private void PreencheCampos(int idx)
    {
        txtnome.setText(act.getRegistros().get(idx).getNome());
        txttelefone.setText(act.getRegistros().get(idx).getTelefone());
        txtendereco.setText(act.getRegistros().get(idx).getEndereco());
    }

    private void AtualizaStatus(int idx)
    {
        int total = act.getRegistros().size();
        txtstatus.setText("Registros : " + (idx+1) + "/" + total);
    }


}
```

Essa classe será responsável por mostrar a listagem dos usuários cadastrados no sistema.

Depois de digitarmos todos os códigos das classes mencionadas anteriormente, vamos executar a nossa aplicação. O resultado você confere na figura seguinte :

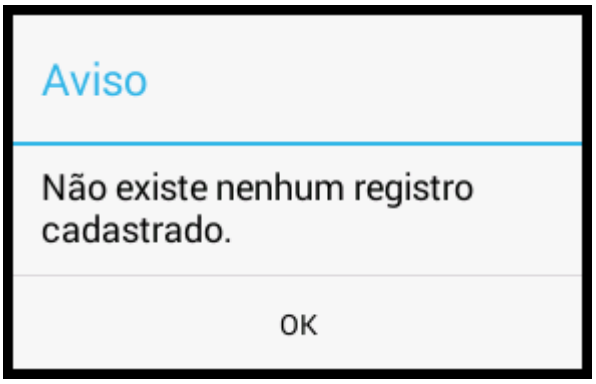


Aplicação em execução – Tela principal

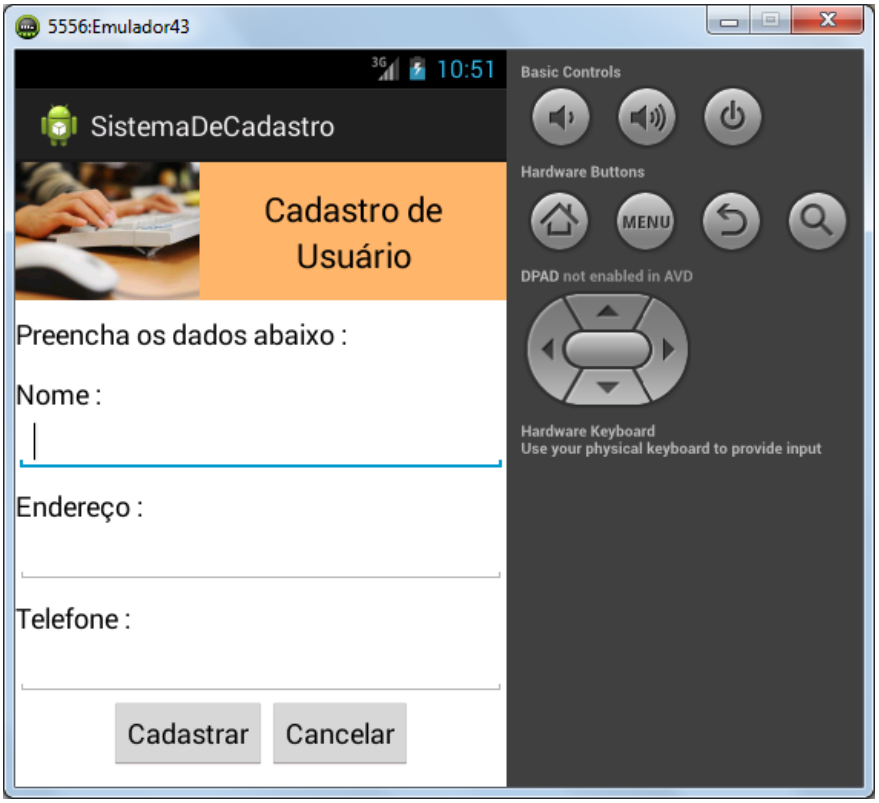
Quando executamos a nossa aplicação, o que visualizamos e a tela principal da aplicação. Quando clicamos no botão “Listar usuários cadastrados”, é executado o seguinte trecho de código dentro da classe **TelaListagemUsuarios** :

```
if(act.getRegistros().size() == 0)
{
	(new AlertDialog.Builder(act))
	.setTitle("Aviso")
	.setMessage("Não existe nenhum registro cadastrado.")
	.setNeutralButton("OK", null)
	.show();

	return;
}
```

Que verifica se o total de registros armazenados na aplicação é igual a “0” (ou seja, se não existe nenhum registro cadastrado). Se a condição for verdadeira (e será verdadeira), será visualizada a seguinte mensagem :



Agora vamos clicar no botão “Cadastrar Usuário”, para visualizarmos a seguinte tela abaixo :

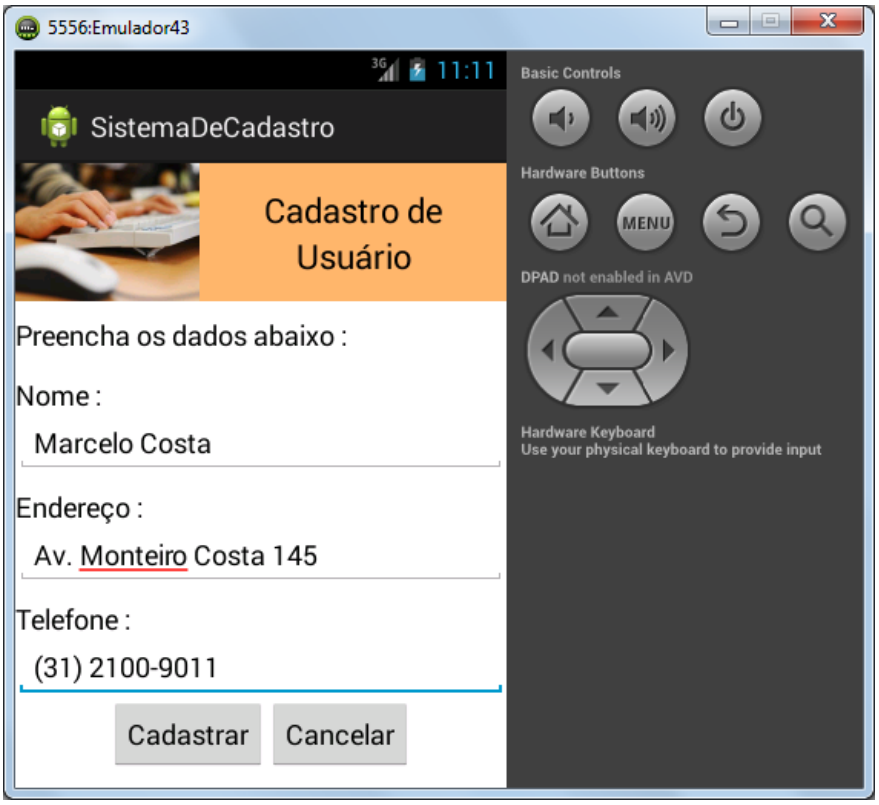


Aplicação em execução – Tela de cadastro de usuário

Vamos preencher abaixo o seguintes dados (fictícios) a seguir :

Nome	Marcelo Costa
Endereço	Av. Monteiro Costa 145
Telefone	(31) 2100-9011

Vejamos o resultado :

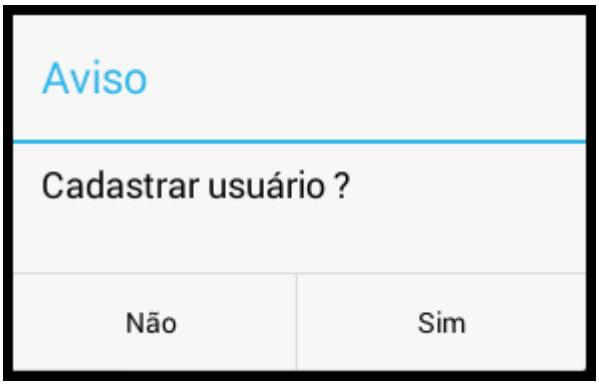


Preenchendo os dados na aplicação

Quando clicamos no botão “Cadastrar” é executado o seguinte trecho de código presente dentro da classe **TelaCadastradoUsuario** :

```
 AlertDialog.Builder dialogo = new AlertDialog.Builder(act);
dialogo.setTitle("Aviso");
dialogo.setMessage("Cadastrar usuário ?");
dialogo.setNegativeButton("Não", null);
dialogo.setPositiveButton("Sim", new DialogInterface.OnClickListener()
{ ... });
```

Que será responsável por exibir a seguinte mensagem abaixo :



Perguntando se você deseja confirmar o cadastrar o usuário. Se clicarmos no botão “Sim”, será executado o seguinte trecho de código abaixo (presente dentro do evento **OnClickListener** presente dentro do método **setPositiveButton** (responsável por mostrar o título “Sim” do botão), da classe **AlertDialog.Builder** :

```
String nome = ednome.getText().toString();
String telefone = edtelefone.getText().toString();
String endereco = edendereco.getText().toString();

act.getRegistros().add(new Registro(nome,telefone,endereco));

act.ExibirMensagem("Cadastro efetuado com sucesso.");
tela_principal.CarregarTela();
```

As três primeiras linhas obtem dos campos os valores digitados (nome , telefone e endereço) , guardando cada valor em suas respectivas variáveis (*nome*, *telefone* e *endereço*) :

Na linha seguinte :

```
act.getRegistros().add(new Registro(nome,telefone,endereco));
```

É efetuado o registro do usuário, onde as suas informações são armazenadas dentro de uma instância da classe **Registro**. A instância da classe **Registro** após registrar as informações passadas como parâmetro no método construtor, a mesma é armazenada dentro do array de registros (retornado pelo método **getRegistros**), que foi declarado dentro da classe **MainActivity**.

Após o registro e mostrado uma mensagem informando que o registro foi feito com sucesso e em seguida carregada a tela principal do programa, pela instrução :

```
tela_principal.CarregarTela();
```

Na tela principal do programa, vamos clicar no botão “Listar usuários cadastrados”. Feito isso será mostrada a seguinte tela abaixo :

Aplicação em execução – Listagem dos usuários cadastrados

Experimente agora efetuar mais alguns registros na aplicação e confira a listagem de aplicações.

Capitulo 7 Trabalhando com menus em uma aplicação

Em uma aplicação Android é possível implementarmos e trabalharmos com menus , onde neles podemos agregar funções que poderão ser executadas de acordo com a nossa necessidade. Os menus da aplicação são visualizados quando pressionamos o botão “Menu” do dispositivo (existentes nos Smartphones, Tablets). Neste capítulo vamos demonstrar o uso dos menus em uma aplicação Android.

Vamos começar criando primeiramente um projeto de acordo com os seguintes dados abaixo:

Application Name: Exemplo com Menus

Company Domain : app.usuario

Project location : (Fica a sua escolha)

Activity Name: MenusActivity

Layout Name : activity_menus

Title : Exemplo com Menus

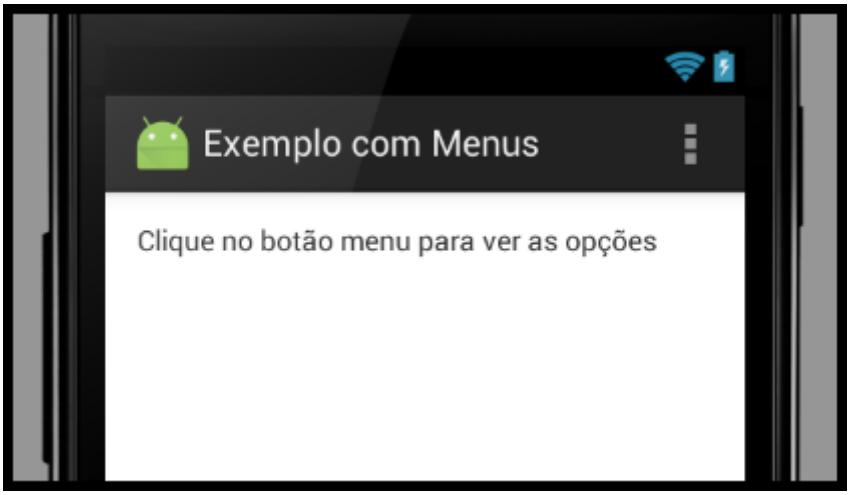
Resource Menu Name : menu_ aplicacao

Depois de criado o projeto mude as propriedades do componente **TextView** conforme demonstra a tabela seguinte:

TextView

Propriedade	Valor
text	Clique no botão menu para ver as opções

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



Layout da tela de aplicação

Vejamos agora a estrutura XML da tela desta aplicação:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MenusActivity">

    <TextView android:text="Clique no botão menu para ver as opções"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

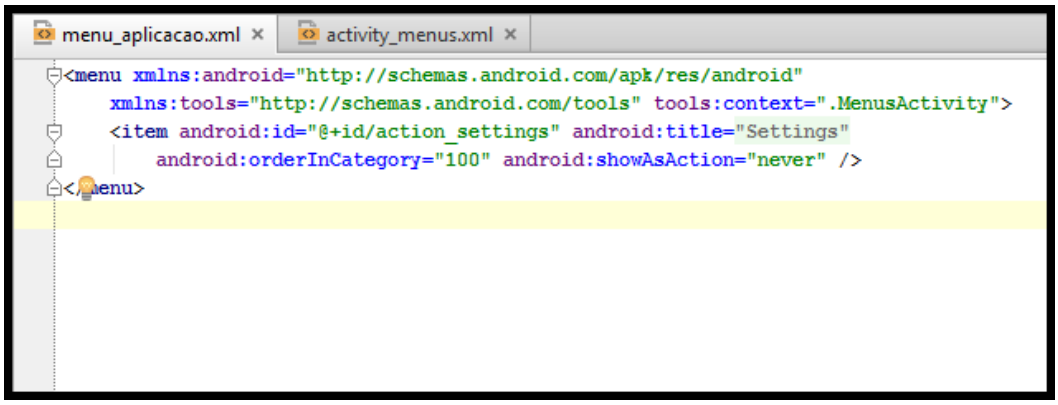
Vejamos agora o código do arquivo “MenusActivity.java”:

```
package usuario.app.exemplocommenus;

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


public class MenusActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menus);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
        // present.
        getMenuInflater().inflate(R.menu.menu_aplicacao, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Se observarmos o código acima, por padrão, existe um método responsável por carregar um menu do arquivo XML (o arquivo “menu_aplicacao.xml”). O nome desse método é **onCreateOptionsMenu.** Ele é executado durante o carregamento do programa na memória. O arquivo que contém todos os itens de menu se encontra dentro do diretório “menu”, dento da pasta de recursos do projeto (a pasta “res”).

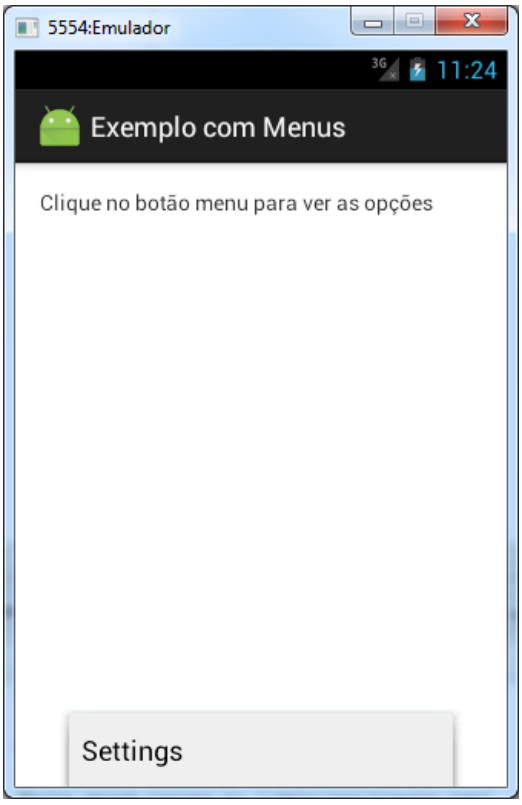
Vamos carregar o arquivo “menu_aplicacao.xml” situado dentro do diretório “menu” para adicionarmos os menus da nossa aplicação. Vejamos o seu código:



Código XML do arquivo "menu_aplicacao.xml"

Se observarmos o seu código acima, existe um item de menu que exibe o item "Settings".

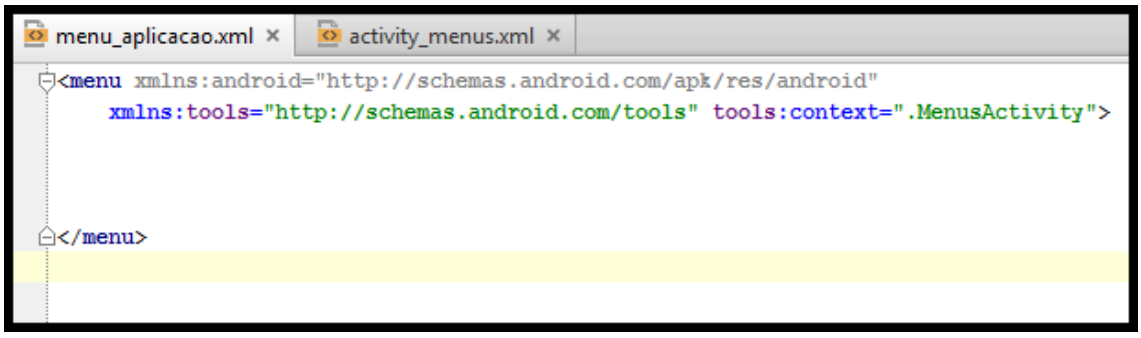
Vamos executar a nossa aplicação para observarmos como os menus funcionam na prática. Com o emulador aberto, pressione a tecla "F2" para que possamos exibir a opção de menu, conforme podemos ver na figura em seguida :



Menu em exibição

Para demonstrar o uso deste recurso de menus no Android irei criar um menu com três itens de opções : “Novo” , “Abrir” e “Salvar”. Nas versões mais atuais do Android (como o 4.x), diferentemente das versões anteriores (como o 2.x e etc), não é e não será possível atribuir ícones para os itens de menu adicionados na aplicação.

Para começar vamos “remover” o item “Settings” do arquivo “menu_aplicacao.xml”, conforme mostra a figura abaixo :

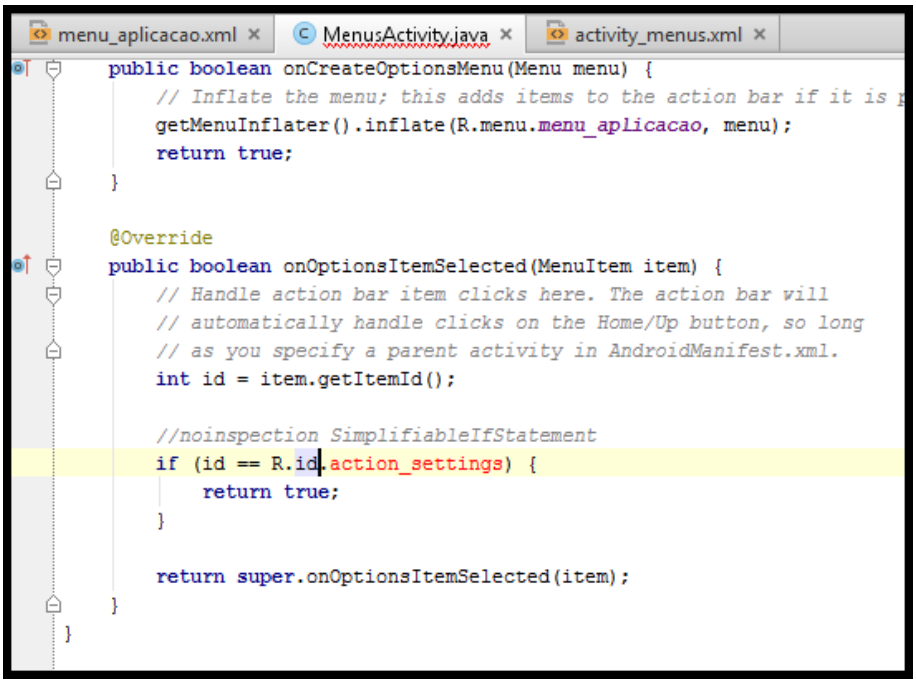


Item removido

Para adicionarmos os itens de menu adicione o seguinte código XML destacado em azul, conforme podemos ver em seguida:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MenusActivity">

    <item android:id="@+id/item_novo"
        android:title="Novo" ></item>
    <item android:id="@+id/item_abrir"
        android:title="Abrir" ></item>
    <item android:id="@+id/item_salvar"
        android:title="Salvar" ></item>


</menu>
```

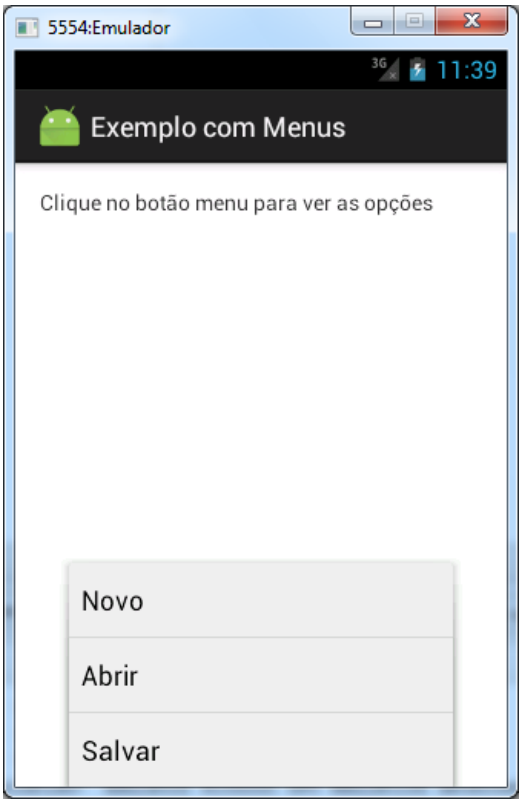
Agora vamos voltar para o arquivo “MenusActivity.java”. Possivelmente você deve ter se deparado com esse erro, conforme mostra a próxima figura :



Erro - O item de menu não existe mais.

O erro acima ocorreu pelo fato de removermos aquele item padrão que já fica presente no arquivo de menu. Para solucionarmos esse erro basta remover o seguinte trecho de código destacado abaixo:

Depois de remover o trecho de código citado, basta executarmos novamente a aplicação e em seguida pressionar a tecla “F2” para chamarmos o menu. Confira o resultado:



Menus em exibição

Irei comentar agora as propriedades que nós alteramos para a criação de nosso menu.

Uma das propriedades que nós alteramos (e que já estamos acostumados a mexer nela) foi a propriedade “Id”. Uma coisa interessante é que para informar o nome do componente usamos a seguinte notação:

```
@+id/<nome do componente>
```

Dentro do Android quando informando um nome (ou “Id”) para um componente, ele usa sempre essa notação. Por exemplo, quando atribuímos um nome para um componente cujo nome é “btcalcular” (como já fizemos em um dos exemplos anteriores), “internamente”, o nome desse componente está nessa forma:

```
@+id/btcalcular
```

Aqui na criação de “menus” estamos atribuindo os nomes utilizando a notação “real” de como o Android associa esses nomes.

A propriedade “Title” serve para definir o título (ou rótulo) do menu.

Por enquanto fizemos a parte visual, porém, não definimos nenhuma ação para os menus. Vamos aprender agora como atribuir ações para cada menu no Android. Vamos no arquivo “MenusActivity.java” e dentro do método **onOptionsItemSelected** digite o seguinte código abaixo:

```
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
   switch (item.getItemId()) {

   case R.id.item_novo:
        //Executa algo
         break;
   case R.id.item_abrir:
        //Executa algo
         break;
   case R.id.item_salvar:
         //Executa algo
          break;
   }
   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.

Também é possível adicionar sub-menus a um determinado menu. Vamos voltar agora para o arquivo de menu “menu_aplicacao.xml” para adicionarmos um sub-menu em um dos itens. Irei escolher aqui o item “salvar” (cujo nome é “item_salvar”) que vai ter um sub-menu associado a ele. Nesse sub-menu haverá duas opções : “Salvar cópia” , “Salvar como”. Para adicionarmos um sub-menu abra o arquivo “menu_aplicacao.xml” e digite o seguinte trecho de código **destacado em azul** em seguida :

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MenusActivity">

    <item android:id="@+id/item_novo"
        android:title="Novo" ></item>
    <item android:id="@+id/item_abrir"
        android:title="Abrir" ></item>
    <item android:id="@+id/item_salvar"
        android:title="Salvar" >

        <menu>

            <item android:id="@+id/item_salvar_copia"
                android:title="Salvar Cópia" ></item>

            <item android:id="@+id/item_salvar_como"
                android:title="Salvar Como" ></item>

        </menu>
    </item>
</menu>
```

Toda vez que pressionarmos o botão salvar, será aberto um sub-menu com as duas opções que colocamos acima. Confira na figura seguinte:

Sub-Menu em exibição

Vejamos agora o código dentro do método **onOptionsItemSelected** para detectar a ação dos dois itens de sub-menu:

```
@Override
public boolean onOptionsItemSelected(MenuItem item)  {

   switch (item.getItemId()) {

   case R.id.item_novo:
         //Executa algo
         break;
   case R.id.item_abrir:
         //Executa algo
         break;
   case R.id.item_salvar_copia:
         //Executa algo
          break;
    case R.id.item_salvar_como:
         //Executa algo
          break;
   }
   return super.onOptionsItemSelected(item);
}
```

Praticamente no código acima não houve grandes mudanças, o que foi necessário foi associar a “id” de cada item de sub-menu para executar uma tarefa dentro da estrutura **switch**.

Capitulo 8 Propriedades e eventos dos componentes trabalhados

Neste capitulo 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
`text`	`android:text`	`setText(CharSequence c)`
Nessa propriedade, você define o texto a ser exibido na tela.		

Propriedade	Em XML	Em Java
`textColor`	`android:textColor`	`setTextColor(Color c)`
Nessa propriedade, você define a cor de texto.		

Propriedade	Em XML	Em Java
`background`	`android:background`	`setBackGroundColor(Color c)`
Nessa propriedade , você define o cor de fundo do componente exibido. Valor: #000000 até #FFFFFF.		

Propriedade	Em XML	Em Java
`textSize`	`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.		

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

Propriedade	Em XML	Em Java
textColor	android:textColor	setTextColor(Color c)
Nessa propriedade, você define a cor do texto.		

Propriedade	Em XML	Em Java
background	android:background	setBackGroundColor(Color c)
Nessa propriedade , você define o cor de fundo do componente exibido. Valor: #000000 até #FFFFFF.		

Propriedade	Em XML	Em Java
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”		

Propriedade	Em XML	Em Java
password	android:password	
Com essa propriedade você habilita a digitação de senhas. O valor padrão desse atributo é “false”.		

Propriedade	Em XML	Em Java
textSize	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.		

Propriedade	Em XML	Em Java
typeface	android:typeface	setTypeface(Typeface fonte)
Essa propriedade serve para definir uma fonte ao texto. Os possíveis valores são : “normal”,”monospace”,”sans” e “serif”.		

Propriedade	Em XML	Em Java
hint	android:hint	setHint(CharSequence c)
define uma mensagem que aparecerá quando a EditText estiver vazia.		

- Eventos

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

Método que define o evento	Evento	Métodos relacionados ao evento
setOnKeyListener	OnKeyListener	onKey(View v,int KeyCode, KeyEvent event)
Esse evento é disparado toda vez que a tecla é acionada, disparando o método onKey.		

Método que define o evento	Evento	Métodos relacionados ao evento
`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
`text`	`android:text`	`setText(CharSequence c)`
Nessa propriedade, você define o texto a ser exibido na tela.		

Propriedade	Em XML	Em Java
`textColor`	`android:textColor`	`setTextColor(Color c)`
Nessa propriedade, você define a cor do texto.		

Propriedade	Em XML	Em Java
`textSize`	`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.		

Propriedade	Em XML	Em Java
`typeface`	`android:typeface`	`setTypeface(Typeface fonte)`
Essa propriedade serve para definir uma fonte ao texto. Os possíveis valores são : “normal”,”monospace”,”sans” e “serif”.		

- Eventos

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

Método que define o evento	Evento	Métodos relacionados ao evento
`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/RadioButton

- Propriedades

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

Propriedade	Em XML	Em Java
`textColor`	`android:textColor`	`setTextColor(Color c)`
Nessa propriedade, você define a cor do texto.		

Propriedade	Em XML	Em Java
`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 ao evento
`setOnClickListener`	`OnClickListener`	`onClick(View v)`
Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.		

Método que define o evento	Evento	Métodos relacionados ao evento
`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 ListView

- Propriedades

Método	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 ao evento
`setOnClickListener`	`OnClickListener`	`onClick(View v)`
Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.		

Método que define o evento	Evento	Métodos relacionados ao evento
`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.		

Método que define o evento	Evento	Métodos relacionados ao evento

`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
`src`	`android:src`	`setImageResource(int Id)`
Nessa propriedade, você define a imagem que será exibida na tela.		

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

Propriedades comuns a todos os componentes

Propriedade	Em XML	Em Java
`id`	`android:id`	
Nessa propriedade , definimos o nome do nosso componente.		

Propriedade	Em XML	Em Java
`layout:width`	`android:layout_width`	
Nessa propriedade, você define a largura do componente a ser exibido. Normalmente essa propriedade assume dois valores : “match_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.		

Propriedade	Em XML	Em Java
`layout:height`	`android:layout_heigth`	
Nessa propriedade, você define a altura do componente a ser exibido. Normalmente essa propriedade assume dois valores : “match_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.		

Propriedade	Em XML	Em Java
`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 a respeito do material

Nesta apostila vimos de forma bem básica e introdutória como desenvolver aplicações para Android para algumas situações . Começamos vendo um pouco sobre a plataforma Android, como ela surgiu e tudo mais. Aprendemos a instalar e configurar Android Studio , que é a ferramenta de desenvolvimento para a criação de Aplicações Android, e em seguida aprendemos a construir algumas pequenas aplicações para Android,como uma calculadora básica, um aplicativo de compras, um aplicativo de cálculo de salário e etc.

Se você quiser uma abordagem “mais completa” de como desenvolver aplicações para Android , adquira a “Apostila de Android – Programando Passo a Passo ” Completa, efetuando o pagamento do seu valor através do **PagSeguro**. Visite o site **www.apostilaandroid.net** para mais informações à respeito da Apostila de Android “COMPLETA”

Espero que esse material lhe tenha sido útil.

Abraços