

Desenvolvimento de Aplicativos para Plataforma Android



Autores: Rafaela Bárbara Custódio

Vítor Carvalho de Melo

Orientador: Marcelo Corrêa Mussel

Apresentação

Hoje em dia, são vários os materiais existentes que permitem a aprendizagem e a prática sobre qualquer linguagem de programação, mas, principalmente para nossa língua, o português, tais materiais não são didáticos o suficiente e ao mesmo tempo não se aprofundam de modo que seja satisfatório para todo o público.

Por essas e outras razões, nos foi proposto o projeto com o objetivo de elaborar uma apostila dividida em etapas, que ao contrário do comum, não são divididas em assuntos (Exemplo: interface, Banco de Dados, etc.), mas sim em níveis de dificuldade, em que envolverá todos estes assuntos, mas que a cada unidade se aprofunda cada vez mais.

Junto ao material escrito, desenvolveremos uma série de vídeo aulas correspondentes. Será desenvolvido junto ao estudante um aplicativo, que seguirá em complexidade o número de páginas da apostila.

Em suma, esta apostila visa tornar mais fácil e intuitiva a aprendizagem de desenvolvimento para a plataforma Android.

Sumário

Apresentação	2
1.0 Introdução	4
1.1 Android Studio	6
2.0 Instalando o Android Studio	6
3.0 Familiarizando-se com a Interface	14
3.1 Criando o primeiro projeto.....	15
3.2 Diretórios do projeto	19
3.3 Testando a aplicação	21
Componentes de Visualização.....	24
4.0 Primeira aplicação.....	26
4.1 Como adicionar uma Imagem	28
4.2 Indo mais além	31
4.3 Intent	36
4.4 Entendendo Arrays	43
4.5 Aplicação usando ArrayAdapter	44
5.0 Media Player	48

Unidade 1

Capítulo 1

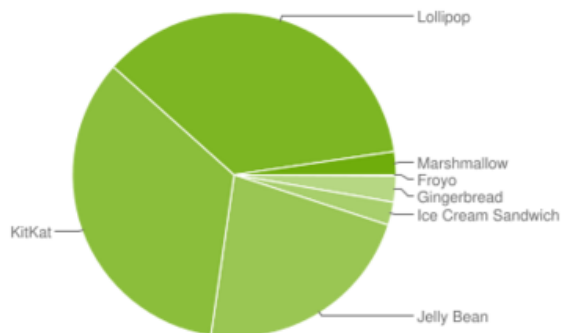
Conhecendo o Android e Android Studio

1.0 Introdução

O **Android** é um sistema operacional baseado no núcleo Linux, é uma plataforma desenvolvida pela Google voltada para dispositivos móveis como celulares, tablets, relógios, etc e totalmente Open Source.

Atualmente o Android é o sistema operacional mais utilizado em todo o mundo, com sua chegada o conceito de smartphone foi modificado. Como já dito o Android é a plataforma mais usada no mundo, porém desde seu surgimento o Android vem recebendo atualizações, atualmente as mais usadas são as versões Jelly Bean, KitKat e Lollipop, mas como essas atualizações acontecem anualmente provavelmente esse cenário mudará em breve.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.3%
4.1.x	Jelly Bean	16	8.1%
4.2.x		17	11.0%
4.3		18	3.2%
4.4	KitKat	19	34.3%
5.0	Lollipop	21	16.9%
5.1		22	19.2%
6.0	Marshmallow	23	2.3%



(Fonte: <https://developer.android.com/about/dashboards/index.html>)

Alguns Dispositivos com Android



Smartphone Motorola Moto Maxx



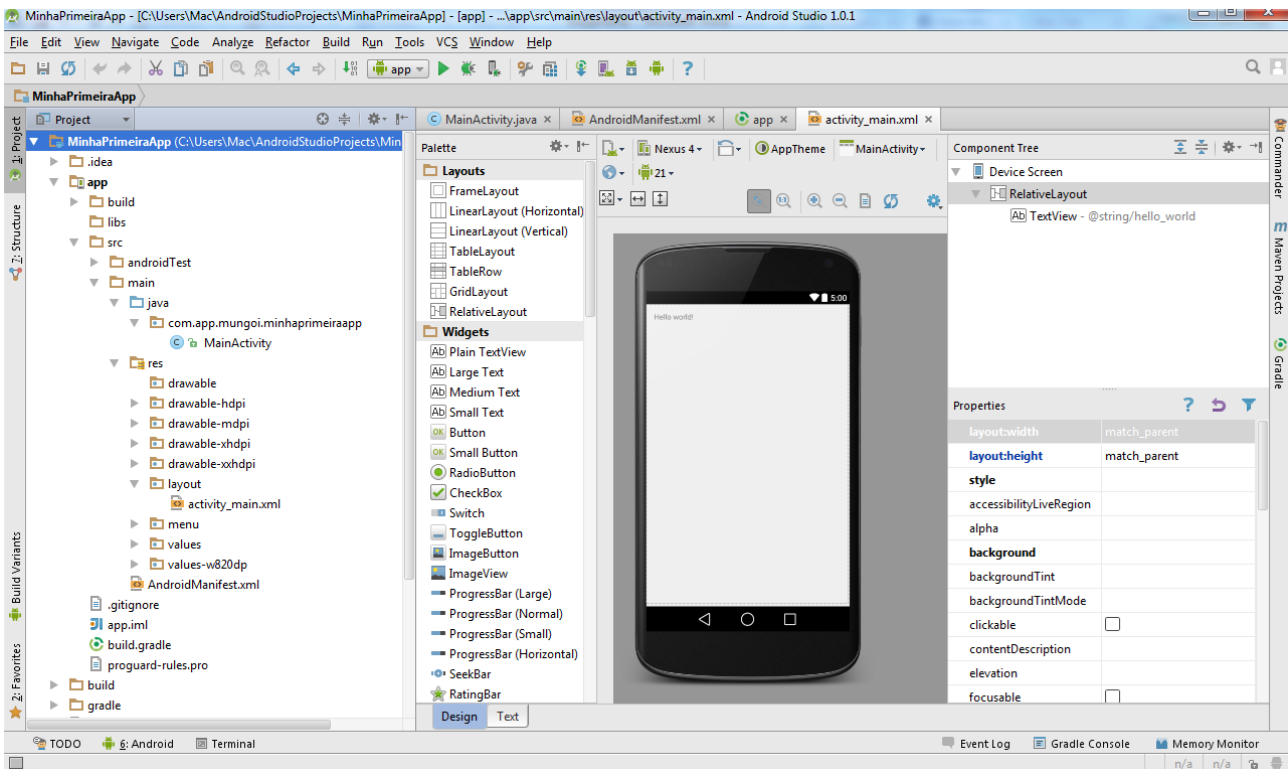
Tablet Samsung Galaxy Tab 3



Relógio Samsung Galaxy Gear S

1.1 Android Studio

O **Android Studio** é o **ambiente de desenvolvimento integrado (IDE)** oferecido pela Google para desenvolvedores que desejam criar aplicações para o Android, e será a ferramenta utilizada durante todo desenvolvimento do material de estudos para que possamos desenvolver nosso aplicativo. O Android Studio pode ser instalado nos sistemas operacionais Windows, OSX e Linux, porém iremos utiliza-lo no SO Linux Mint, em seguida vamos ensinar como baixar, instalar e configurar o Android Studio.



2.0 Instalando o Android Studio

Antes de instalarmos o Android Studio devemos verificar os requisitos mínimos de seu computador para que o android Studio funcione em seu computador, no caso do Linux segue abaixo quais são os pré-requisitos mínimos necessários para o bom funcionamento da ferramenta.

Requerimentos do Sistema

GNOME ou KDE ambiente de trabalho

GNU C Library (glibc) 2,15

2 GB RAM mínimo, 4 GB RAM recomendado

400 MB de espaço no disco rígido

Pelo menos 1 GB para o Android SDK, imagens do sistema emulador, e caches

1280 x 800 resolução mínima de tela

Oracle® Java Development Kit (JDK) 7

Distribuição de 64 bits capaz de executar aplicativos de 32 bits

(fonte: <http://developer.android.com/sdk/index.html#Requirements>)

Caso você esteja utilizando outro sistema operacional, poderá verificar estes em <http://developer.android.com/sdk/index.html#Requirements>.

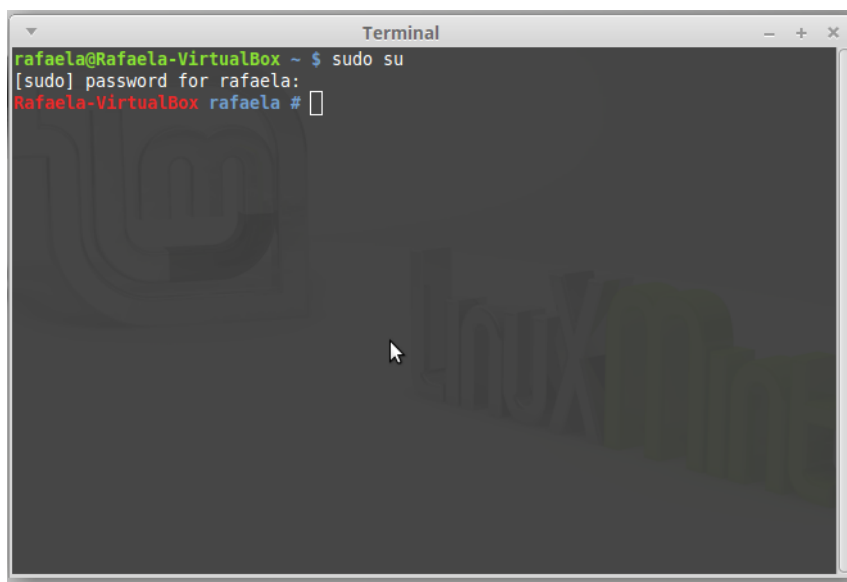
Como visto, um dos requisitos necessários é o JDK (Java Development Kit), este é primordial para que possamos utilizar o Android Studio e criar nossas aplicações.

O **Java Development Kit** está disponível para download em <http://www.oracle.com/technetwork/java/javase/downloads/index.html> e é grátis.

Para baixar e instalar o Java Development Kit basta seguir os seguintes passos:

Passo 1:

Abra o terminal de seu computador e entre como usuário root, digitando o comando **sudo su**, e em seguida sua senha.



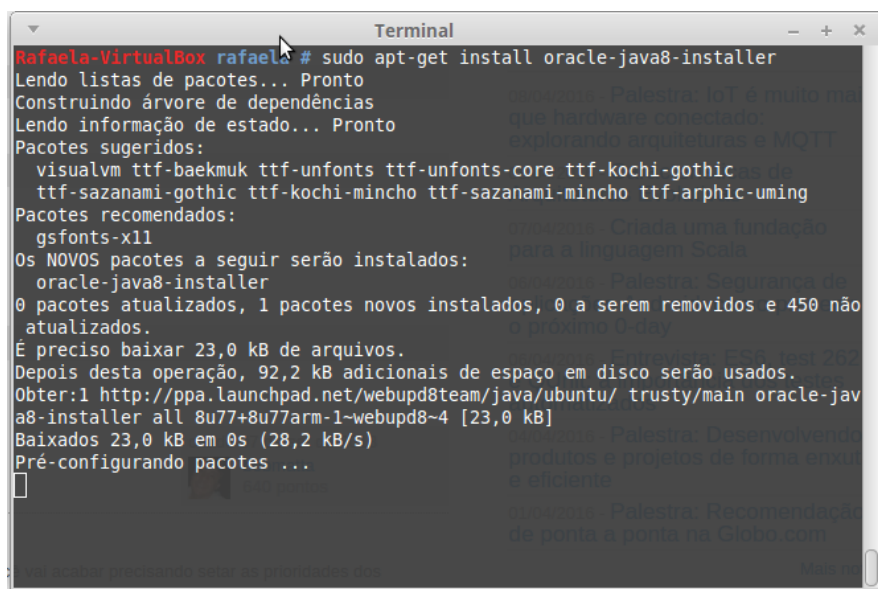
Passo 2:

Digite o seguinte comando e em seguida aperte a tecla ENTER de seu computador:

add-apt-repository ppa:webupd8team/java

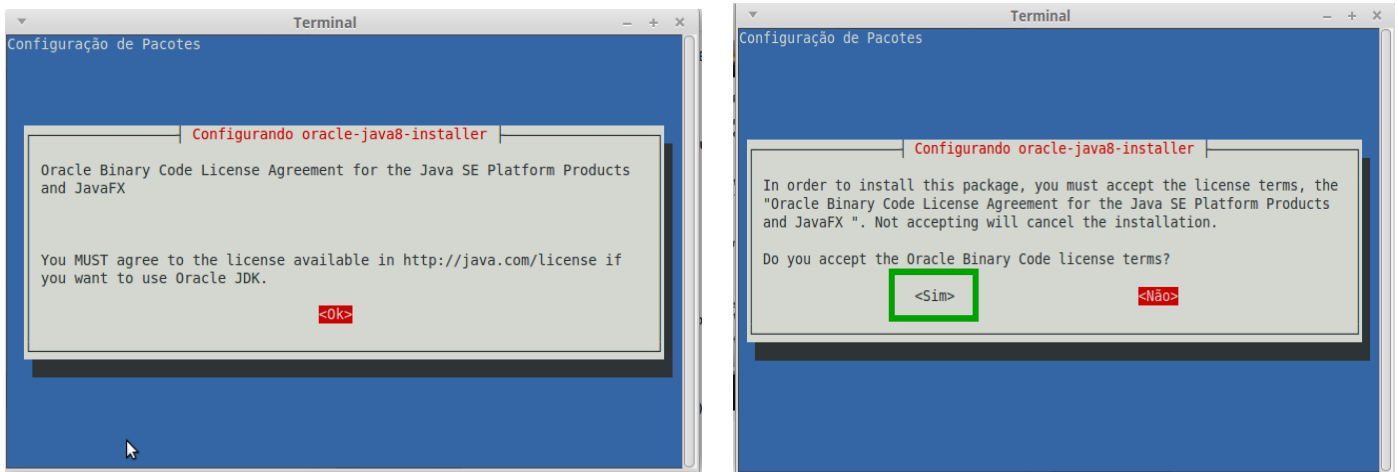
Em seguida digite o comando:

apt-get install oracle-java8-installer como mostra a figura abaixo.



Passo 3:

Por fim as seguintes telas irão aparecer, **aperte [OK] e [SIM]**.



E pronto, após o processo o Java Development Kit(JDK) estará baixado, instalado e configurado em seu computador.

Agora finalmente, podemos fazer o download de nossa ferramenta do trabalho, o [Android Studio](#).

Siga os seguintes passos.

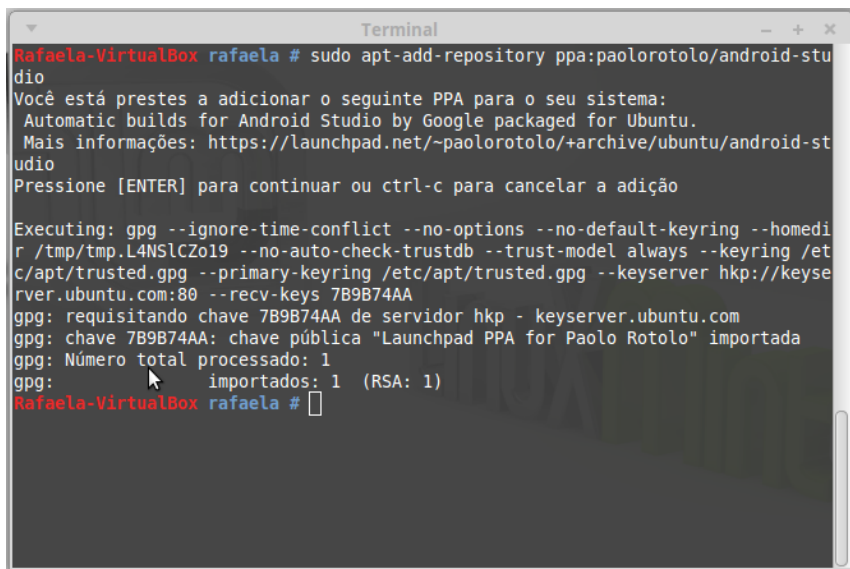
Passo 1:

Novamente abra seu terminal e entre como usuário root, como foi ensinado anteriormente para instalar o JDK.

Passo 2:

Digite o seguinte comando no terminal:

apt-add-repository ppa:paolorotolo/android-studio



```

Rafaela-VirtualBox rafaela # sudo apt-add-repository ppa:paolorotolo/android-studio
Você está prestes a adicionar o seguinte PPA para o seu sistema:
Automatic builds for Android Studio by Google packaged for Ubuntu.
Mais informações: https://launchpad.net/~paolorotolo/+archive/ubuntu/android-studio
Pressione [ENTER] para continuar ou ctrl-c para cancelar a adição

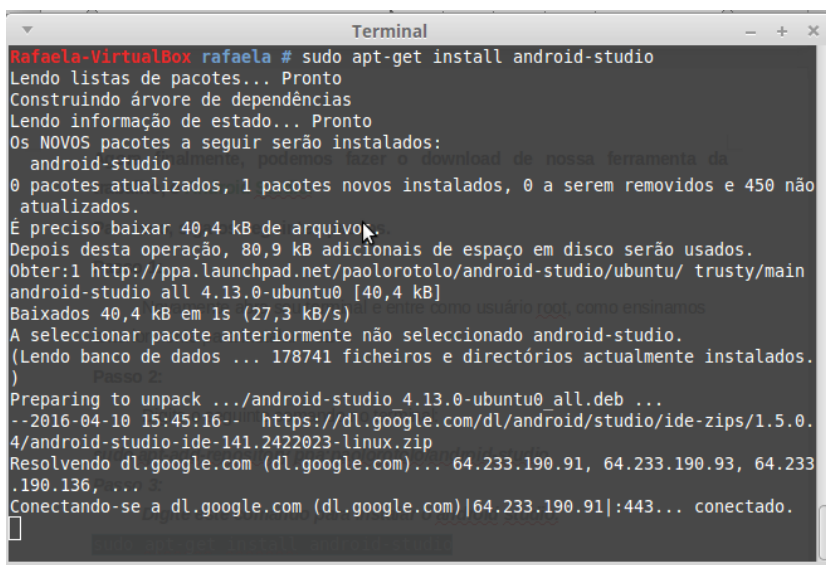
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --homedir /tmp/tmp.L4NSlCZo19 --no-auto-check-trustdb --trust-model always --keyring /etc/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --keyserver hkps://keyserver.ubuntu.com:80 --recv-keys 7B9B74AA
gpg: requisitando chave 7B9B74AA de servidor hkps - keyserver.ubuntu.com
gpg: chave 7B9B74AA: chave pública "Launchpad PPA for Paolo Rotolo" importada
gpg: Número total processado: 1
gpg:      importados: 1 (RSA: 1)
Rafaela-VirtualBox rafaela #

```

Passo 3:

Digite esta linha de comando para instalar o android studio.

apt-get install android-studio



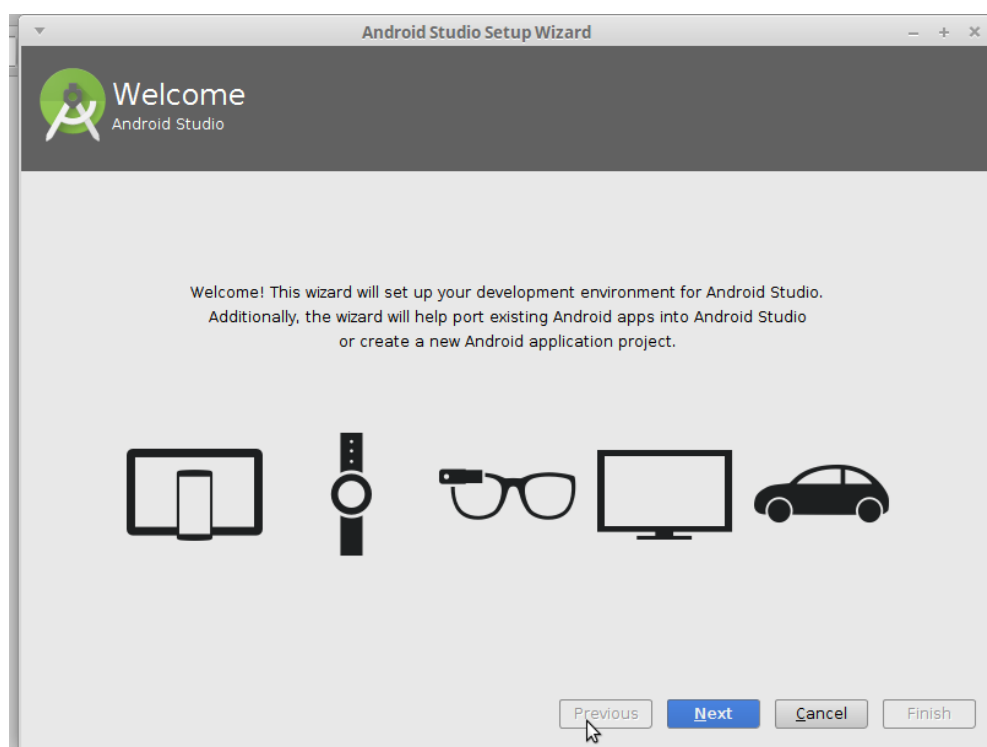
```

Rafaela-VirtualBox rafaela # sudo apt-get install android-studio
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os NOVOS pacotes a seguir serão instalados:
  android-studio
0 pacotes atualizados, 1 pacotes novos instalados, 0 a serem removidos e 450 não atualizados.
É preciso baixar 40,4 kB de arquivos.
Depois desta operação, 80,9 kB adicionais de espaço em disco serão usados.
Obter:1 http://ppa.launchpad.net/paolorotolo/android-studio/ubuntu/ trusty/main android-studio all 4.13.0-ubuntu0 [40,4 kB]
Baixados 40,4 kB em 1s (27,3 kB/s)
A seleccionar pacote anteriormente não seleccionado android-studio.
(Lendo banco de dados ... 178741 ficheiros e directórios actualmente instalados.)
Preparing to unpack .../android-studio_4.13.0-ubuntu0_all.deb ...
--2016-04-10 15:45:16-- https://dl.google.com/dl/android/studio/ide-zips/1.5.0.4/android-studio-ide-141.2422023-linux.zip
Resolvendo dl.google.com (dl.google.com)... 64.233.190.91, 64.233.190.93, 64.233.190.136, ...
Conectando-se a dl.google.com (dl.google.com)[64.233.190.91]:443... conectado.

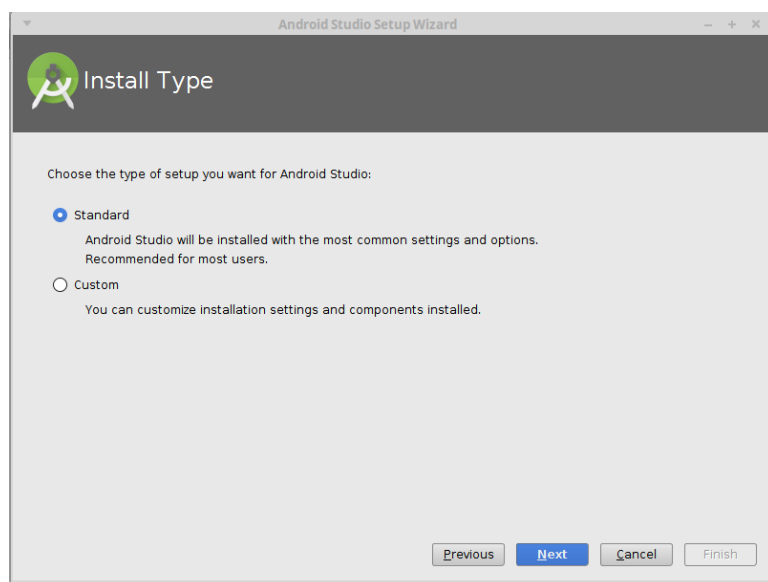
```

Pronto o Android Studio será instalado em seu computador.

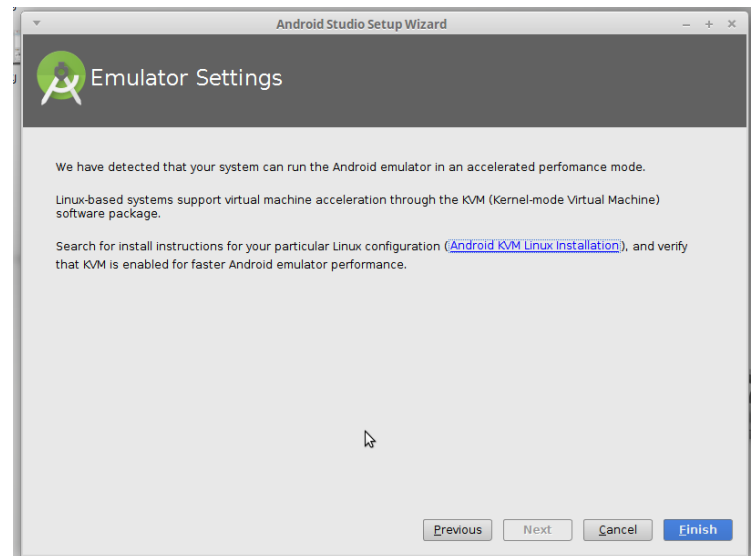
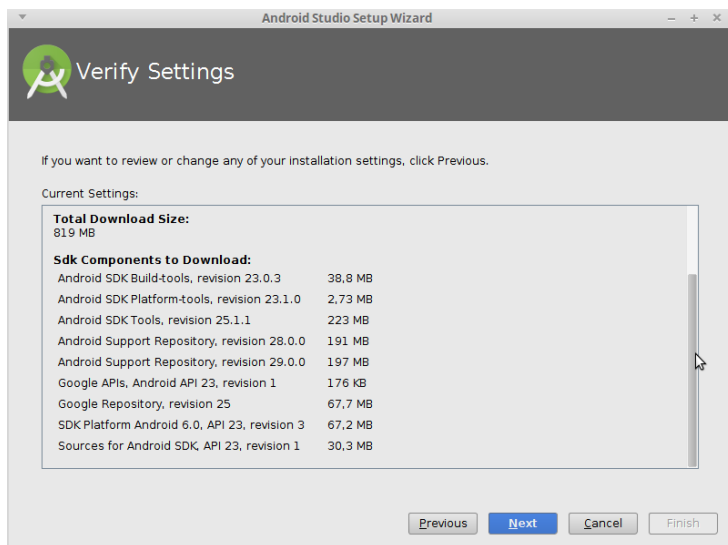
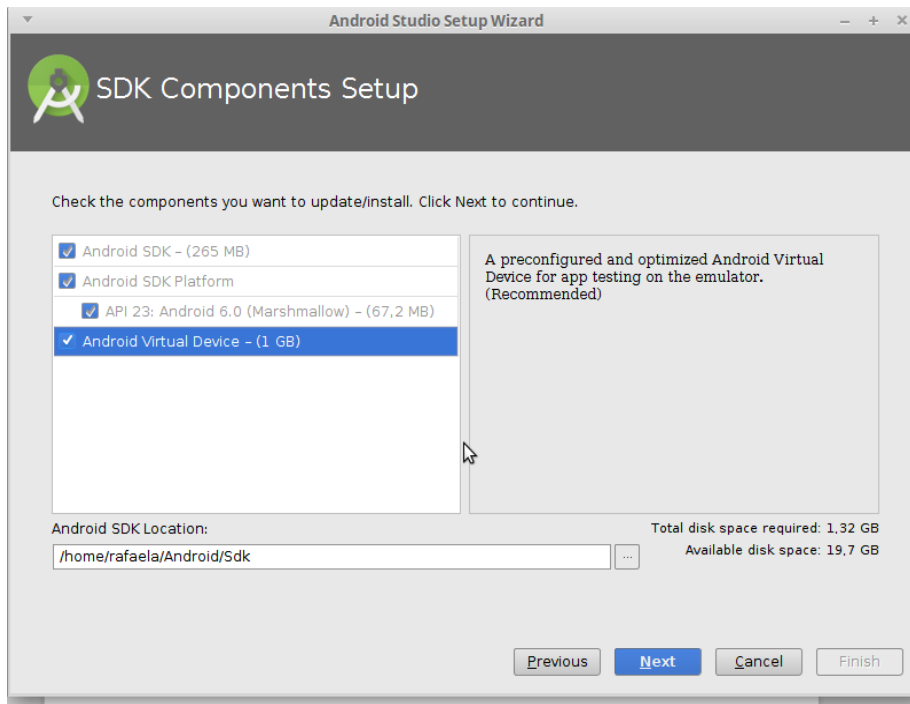
Vamos então iniciar nossa IDE (Android Studio). Para isso vá ao menu iniciar de seu computador e clique no Android Studio, é normal que demore um pouco para abrir. A primeira tela que temos contato é a tela de boas vindas do assistente de configuração, nesta tela podemos apertar o botão [NEXT] no canto inferior direito.



Em seguida marque a opção Standart e depois [next].

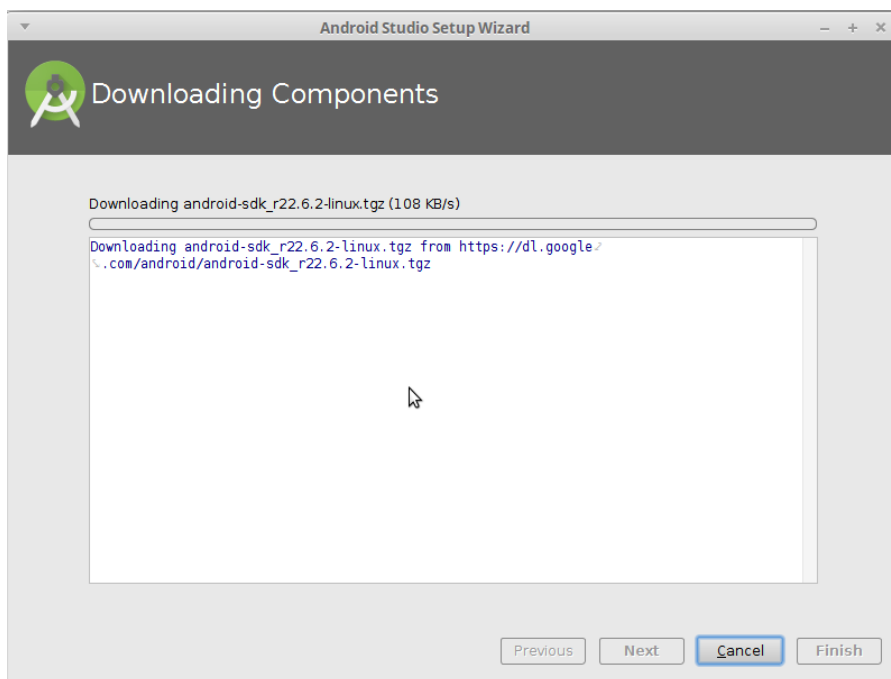


Agora a tela que aparece para nós está mostrando os componentes do Android Studio que serão atualizados ou instalados, selecione todos, inclusive o Android Virtual Device (futuramente será explicado do que se trata) e prossiga.



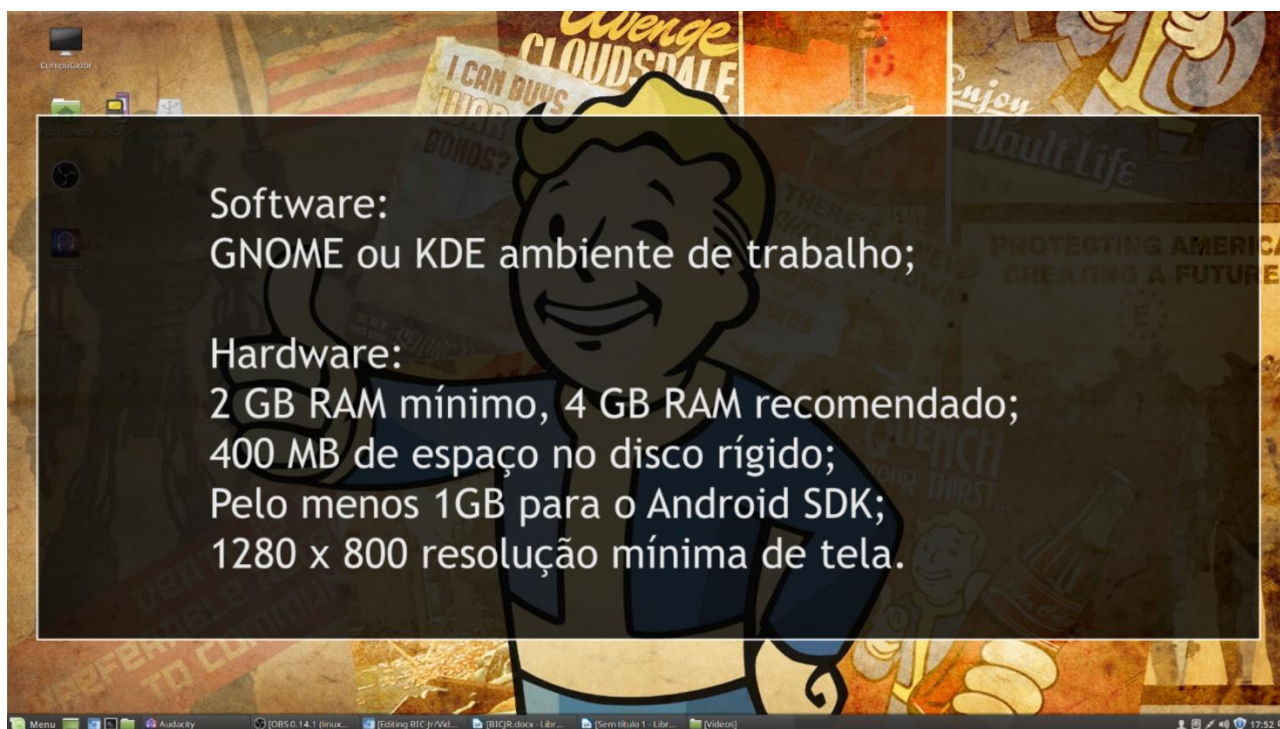
Apenas algumas informações e podemos prosseguir

Ao clicar em **Finish** basta aguardar os pacotes adicionais sejam instalados. Não se preocupe, isso pode demorar um pouco.



1.3.1 Video Tutorial

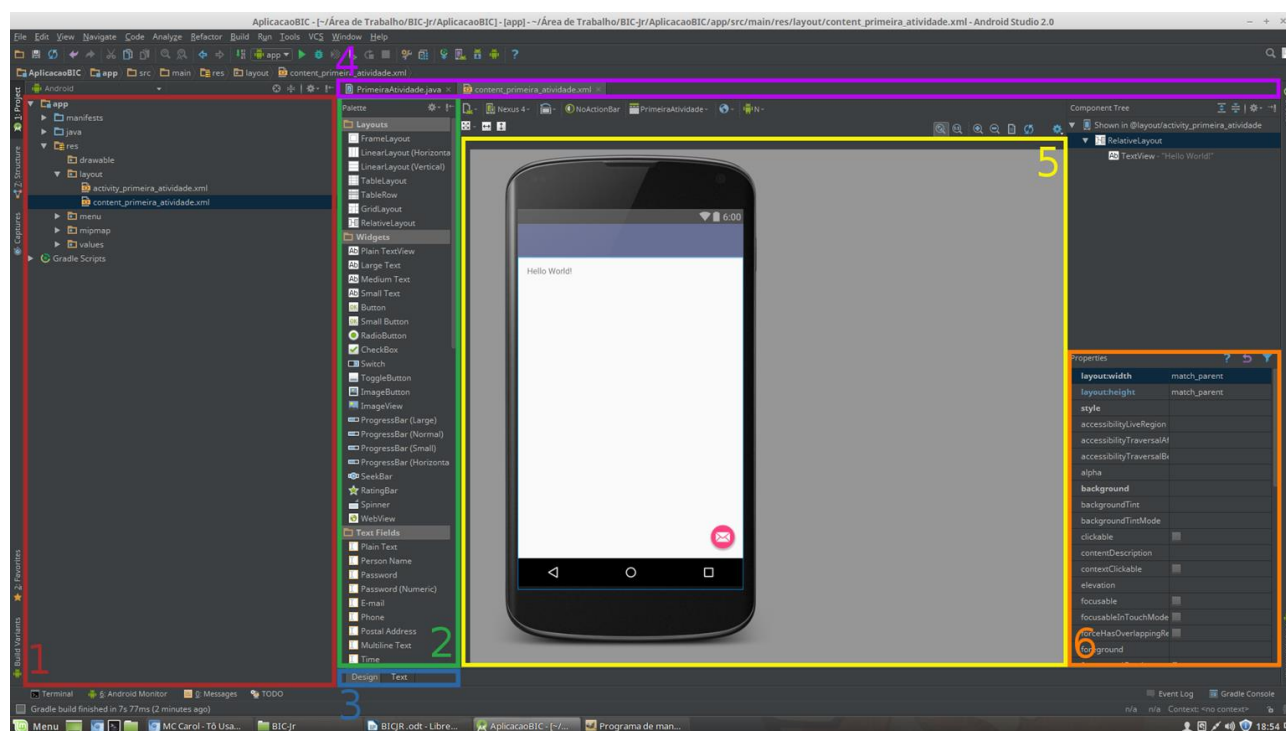
Criamos um vídeo tutorial da instalação do Android Studio seguindo as instruções acima.



3.0. Familiarizando-se com a Interface

Nesta sessão, aprenderemos aquilo que compõe a tela principal e inicial do Android Studio que aparecerá logo após a criação de um novo projeto.

Para nós fazermos mais entendíveis, vamos dividir a tela em 6 diferentes sessões:



(imagem 1)

1 – Nesta sessão encontramos a hierarquia dos diretórios do projeto. Aqui podemos ver binários, códigos fonte, arquivos de layout e outros de configuração. É também possível nesta seção adicionar novos arquivos ao projeto.

2 – Aquilo que é necessário para configurar as interfaces gráficas. Imagens, caixas de texto, botões, barras de progresso e outros widgets. Arrastar e soltar um widget é tudo que se precisa para inseri-lo no layout. É necessário no entanto ter cuidado pois posições são muito sensíveis.

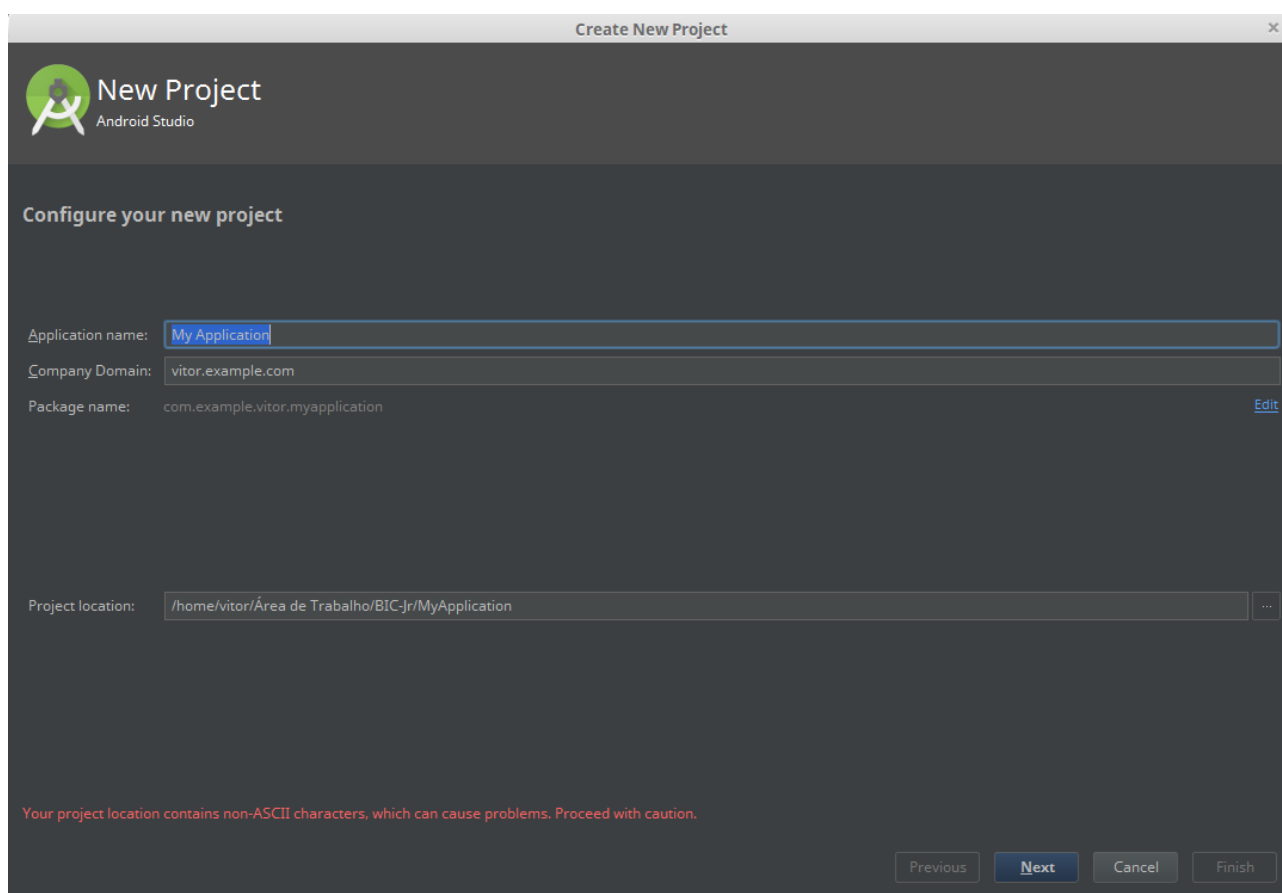
3 – Existem duas formas de desenvolver a interface gráfica: com base no drag and drop (arrastar e soltar) ou no texto puro. As preferências variam, é claro, de pessoa para pessoa. Recomendamos Drag and drop para adicionar o widget e o modo texto para fazer o ajuste fino.

- 4 – É possível abrir vários arquivos ao mesmo tempo. Arquivos .xml para o design da interface gráfica e configurações e arquivos .java para a programação e codificação do app.
- 5 – Aqui fica visível uma prévia da interface do app. É sempre bom ter um feedback para saber se está tudo no lugar certo.
- 6 – É possível selecionar um widget e configurar todas as suas propriedades a partir desses campos. Na prática, quase sempre é preferível usar o modo texto para isso.

3.1. Criando o primeiro projeto

Criemos agora o primeiro projeto no Android Studio. Ao clicar em “Criar novo projeto”, esta janela aparecerá.

Obs: seu Android Studio provavelmente não estará escuro assim, mas não se preocupe, esta é apenas uma opção de layout que veremos mais tarde.



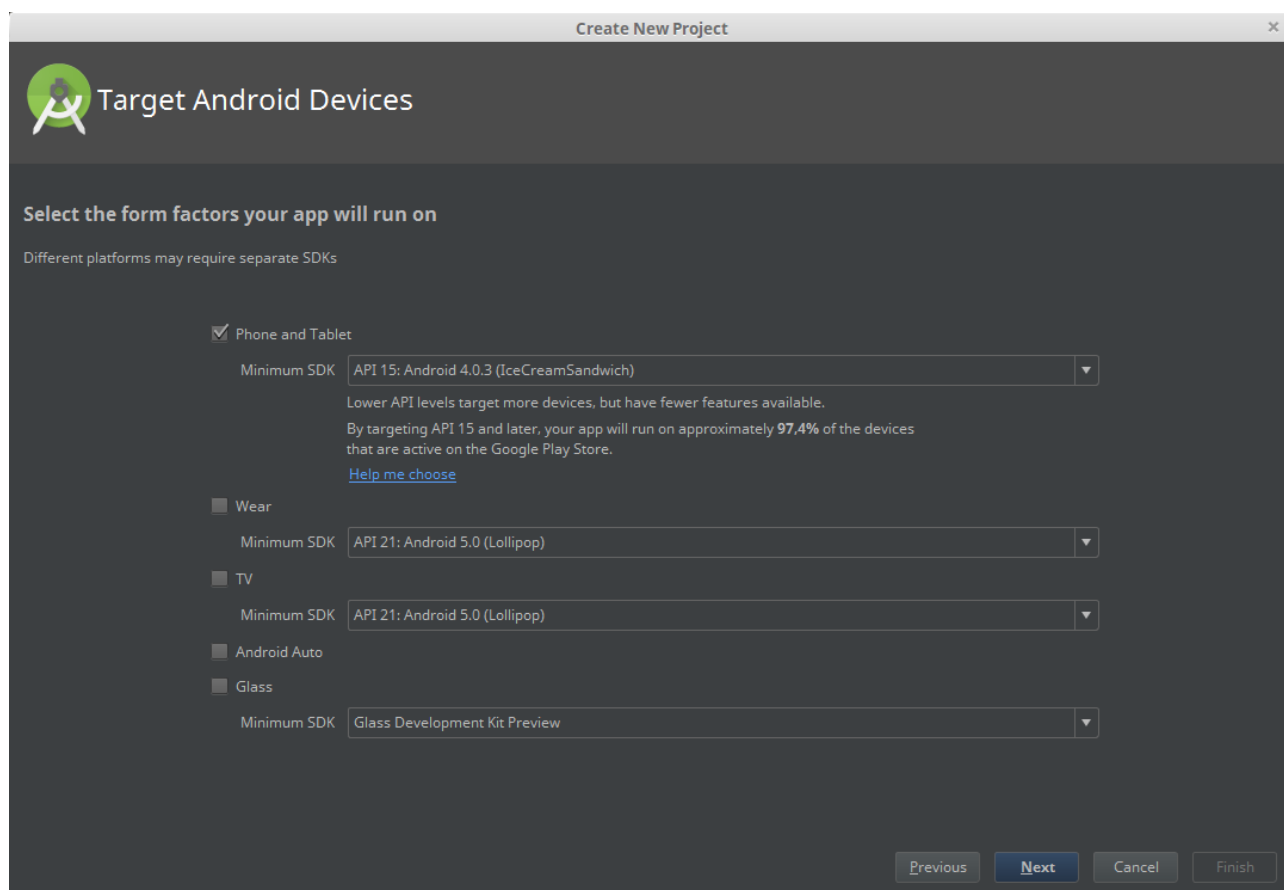
The screenshot shows the 'Create New Project' dialog in Android Studio. The window title is 'Create New Project'. The header area features the Android Studio logo and the text 'New Project' and 'Android Studio'. Below this, the section 'Configure your new project' is visible. It contains four input fields: 'Application name' with the value 'My Application', 'Company Domain' with 'vitor.example.com', 'Package name' with 'com.example.vitor.myapplication' (and an 'Edit' link), and 'Project location' with '/home/vitor/Área de Trabalho/BIC-Jr/MyApplication'. A warning message at the bottom states: 'Your project location contains non-ASCII characters, which can cause problems. Proceed with caution.' At the bottom right, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

“*Application name:*” - neste campo, escolha o nome de seu novo projet;

“*Company Domain*” - domínio da empresa, não se preocupe com esse campo;

“*Project Location*” - localização do projeto em sua máquina; recomendo criar um diretório para manter seu projeto organizado.

Ao clicar em “Next”, a seguinte janela aparecerá:



Nesta janela, poderá selecionar a plataforma à qual deseja desenvolver a aplicação. As opções que o Android Studio oferecem são:

- Smartphone e Tablet;
- Wearables(relógios, headphones);
- Televisões;
- Android Auto(automóveis);
- Glass(Aparelho da Google).

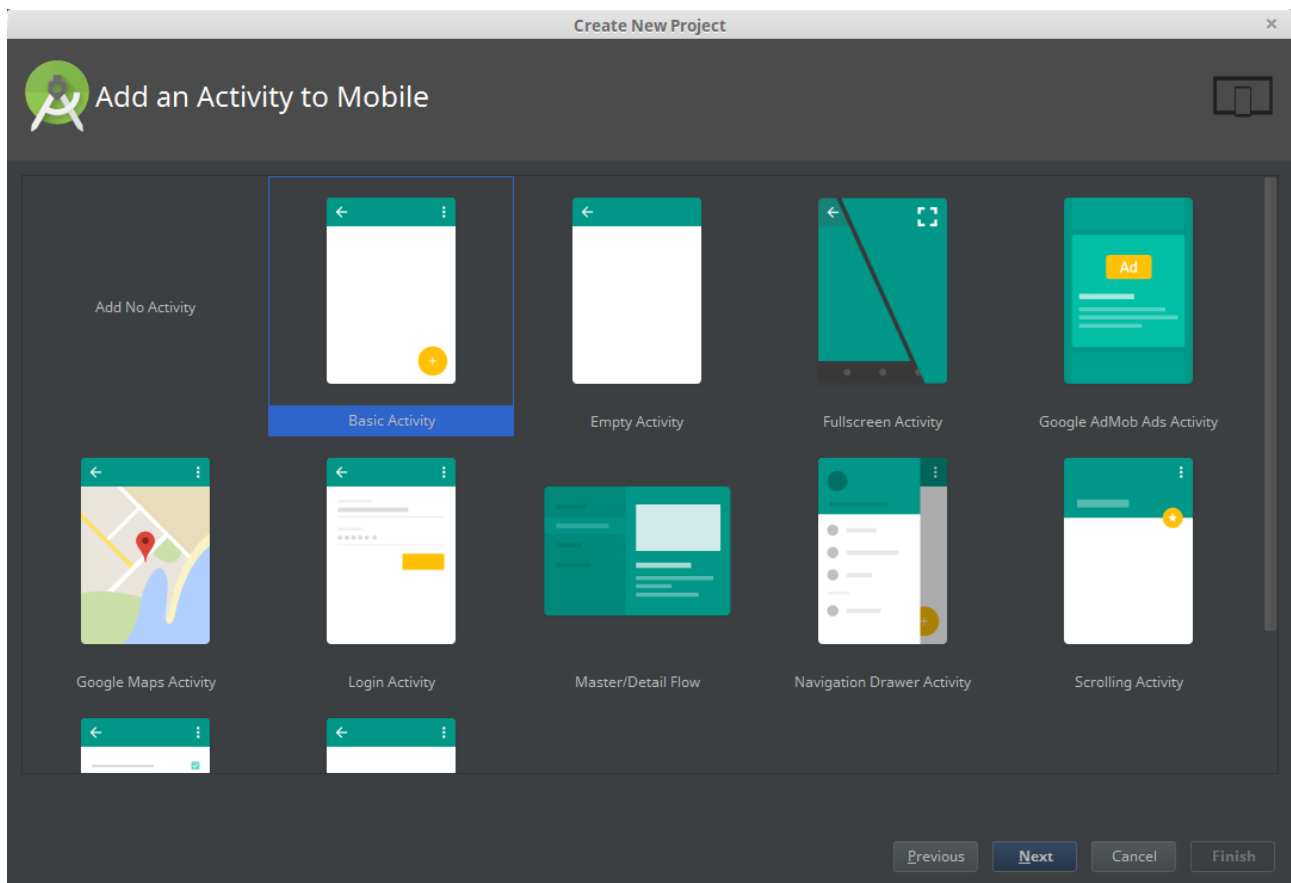
Na opção que utilizaremos(Phone e Tablet), você será apresentado com diversos SDK's disponíveis(Kit de desenvolvimento). A diferença entre eles é que quanto menor e mais

antigo, mais aparelhos poderão utilizá-lo, ao custo de que menos funções estarão disponíveis para uso, por exemplo:

- API 15: Adnroid 4.0.3: esta API para o Android IceCreamSandwitch, de Outubro de 2011, poderá ser utilizada por 97,4% dos aparelhos.

- API 23: Adnroid 6.0: esta API para o Android Marshmallow, de Maio de 2015, poderá ser utilizada por apenas 4.7% dos aparelhos.

Ao clicar em “Next”, teremos a seguinte janela:



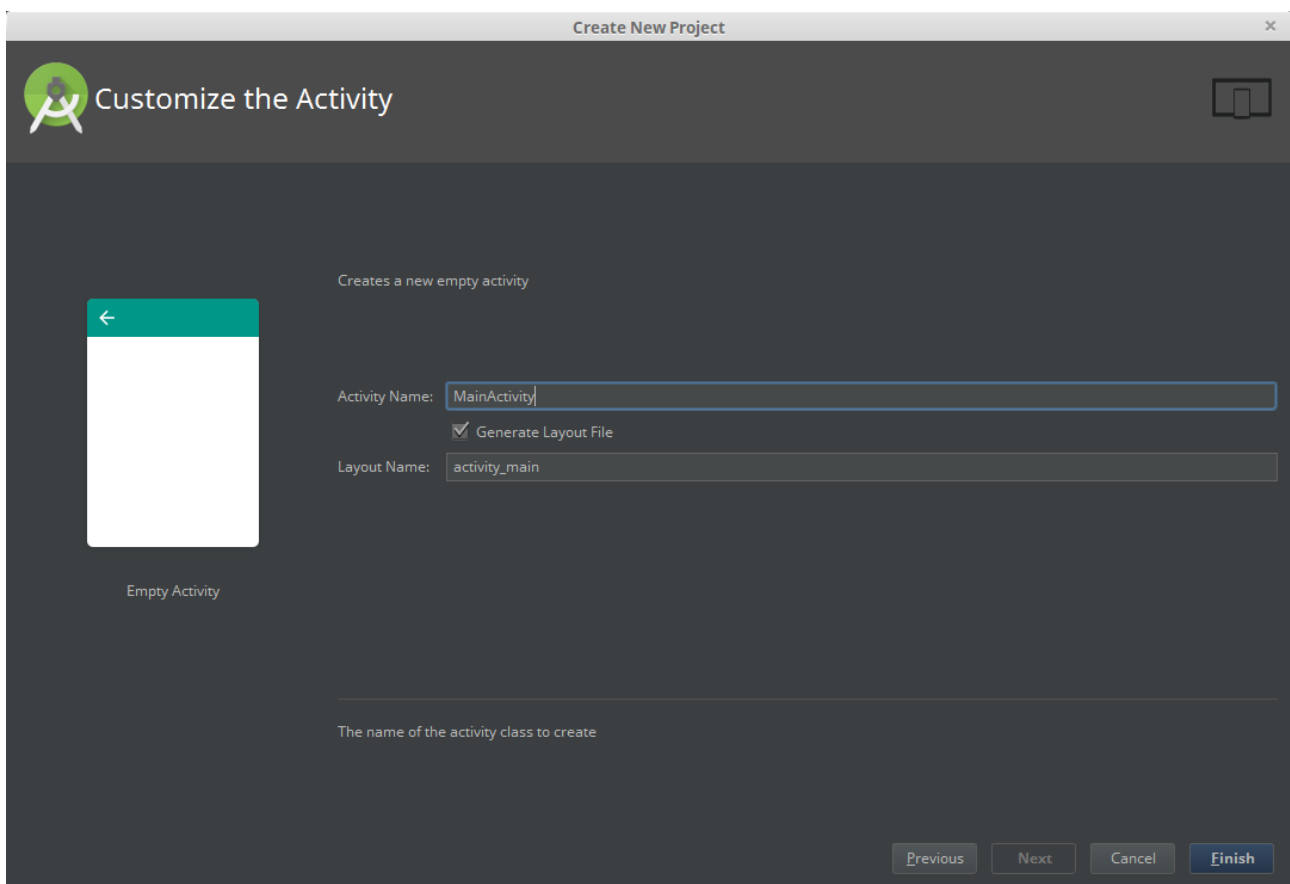
Nesta janela, nos são apresentadas várias opções para a primeira atividade de nosso projeto. Temos as opções de:

- **Add no Activity:** não adicionar atividade neste primeiro momento;
- **Basic Activity:** atividade com apenas um dois botões em preset, um superior para configurações e um inferior com outra função qualquer;
- **Fullscreen Activity:** atividade que ocupa toda a tela, usada geralmente para jogos ou player de vídeos;

- **Google AdMob Ads Activity:** atividade para anúncios in-app do Google;
- **Google Maps Activity:** atividade que usará a API do Google Maps;
- **Login Activity:** layout pré desenvolvido para fazer login;
- **Master/Detail Flow:** atividade para aplicações com seleção e rolamento de informações;
- **Navigation Drawer Activity:** layout de janela arrastável;
- **Scrolling Activity:** atividade que ocupará mais que a tela do aparelho, por isso possuirá rolamento;
- **Settings Activity:** atividade de configuração por parte do usuário da aplicação;
- **Tabbed Activity:** atividade com diversas janelas à esquerda e a direita.

Para nossa aplicação, selecionaremos “Empty Activity”, já que faremos tudo do começo.

Ao clicar em “Next”, a seguinte janela aparecerá:



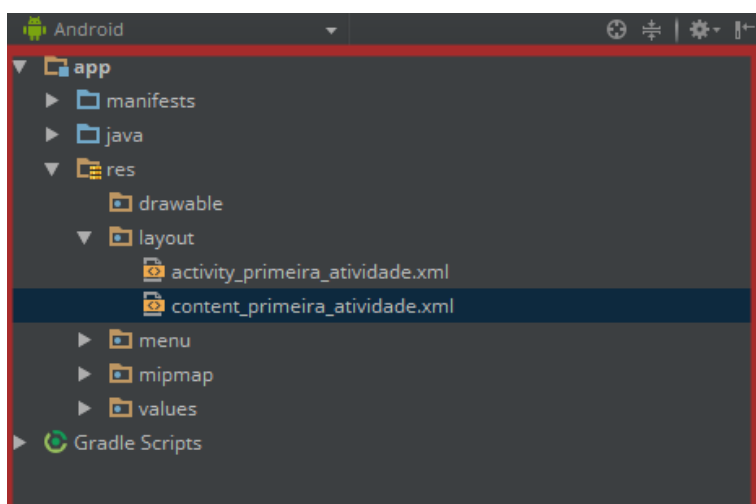
“*Activity Name:*” - aqui poderá escolher o nome desta primeira atividade;

Caixa de seleção “*Generate Layout File*” - selecione caso deseje atrelar ao projeto um arquivo de layout .xml para fazer a customização de visualização;

“*Layout Name*” - nome do arquivo .xml de layout que será criado;

3.2 Diretórios do projeto

Os diretórios do projeto estão na sessão destacada em vermelho na imagem anterior.



Diretório “app”

Neste diretório está presente todos os principais arquivos da aplicação. Dentro deste temos os diretórios manifests e java.

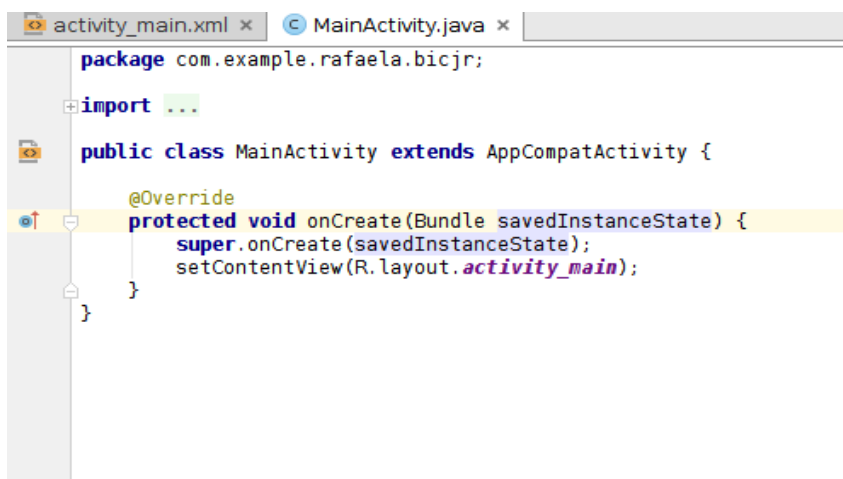
Manifests: Neste diretório encontramos o arquivo xml chamado AndroidManifest.xml, este é responsável pelas definições de permissões e configurações para execução de nosso aplicativo. É obrigatório que cada activity do projeto esteja declarada no arquivo AndroidManifest.xml, caso contrario não é possível utiliza-la. Para declarar a activity é utilizada a tag <activity>, que recebe o nome da classe, e é sempre relativa ao pacote principal.

Obs.:

Xml: Linguagem de marcação recomendada pela W3C para a criação de documentos com dados organizados hierarquicamente, tais como textos, banco de dados ou desenhos vetoriais. A linguagem XML é classificada como extensível porque permite definir os elementos de marcação.

Activity: Uma **Activity** é basicamente uma classe gerenciadora de UI (Interface com o usuário). Todo aplicativo android começa por uma **Activity**. Ou seja, quando uma aplicação android é executada, na verdade é a sua **Activity** principal que é lançada.

Java: No diretório java é onde ficam as classes java desenvolvidas para a aplicação, nela estará nossa classe principal MainActivity.java



```

package com.example.rafaela.bicjr;

import ...

public class MainActivity extends AppCompatActivity {

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

```

Res: A pasta res contém os recursos utilizados na aplicação, como por exemplo, imagens, musicas, sons, layouts e etc. Nela encontramos mais 4 diretórios:

→ Drawable

A função deste diretório é simplesmente armazenar todas as imagens que serão utilizadas na aplicação

→ Layout

Nesse diretório ficam os arquivos responsáveis pelo desing de seu aplicativo, ou seja, todas as telas pertencentes a ele.

→ Mipmap

Imagens referentes aos ícones do aplicativo.

→ Values

Possui configurações xml para definição de contantes ou para metrização do projeto, como cores ou palavras que estarão presentes na maioria ou em todas as telas.

3.3. Testando a aplicação

O Android Studio contém uma espécie de emulador que nos permite executar nossas aplicações. Não é a única forma possível para testar e executar nossos aplicativos, mas iremos usá-la por ser de mais fácil acesso.

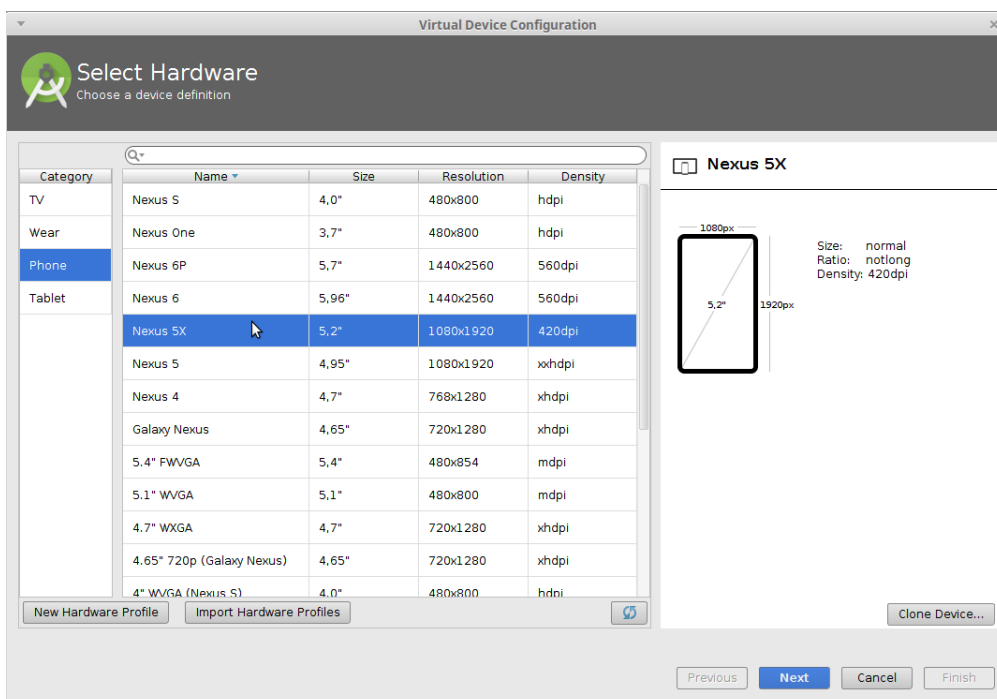
Para iniciar o emulador siga os seguintes passos:

1. No menu superior do android-studio clique na opção “AVD Manager” como mostra a figura abaixo

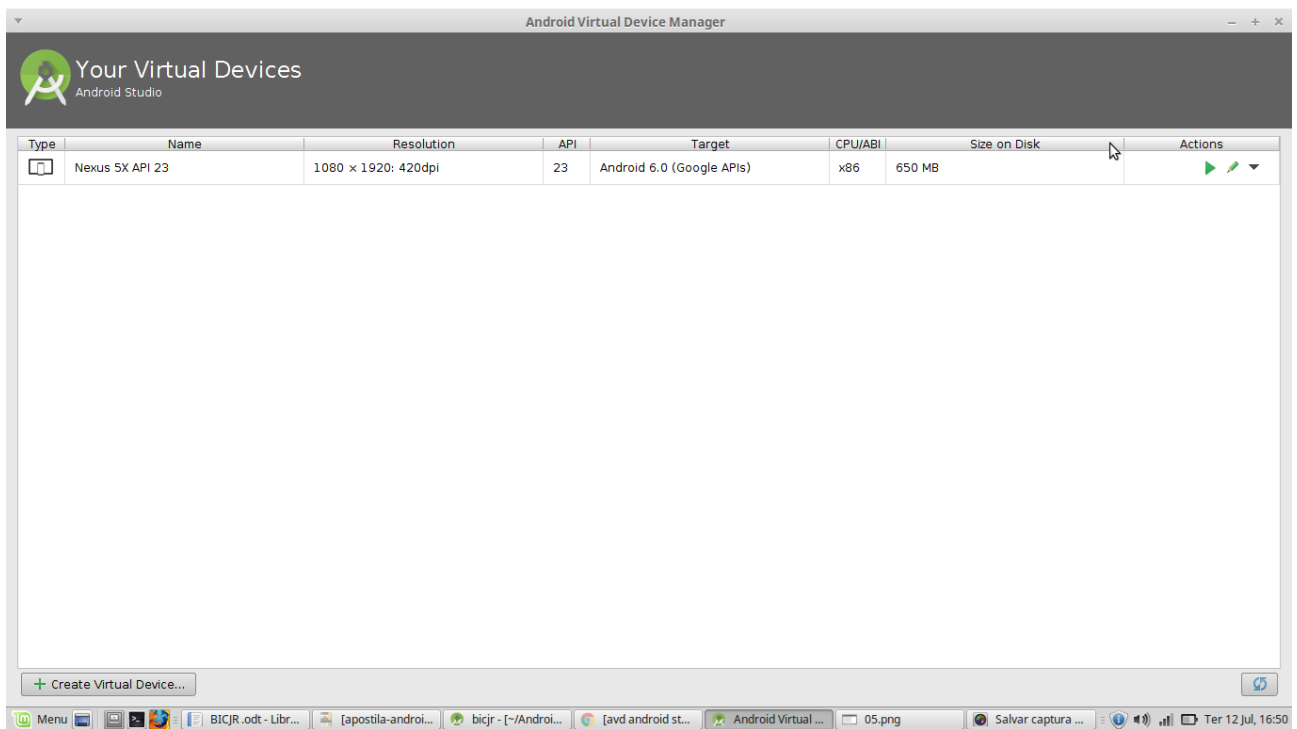
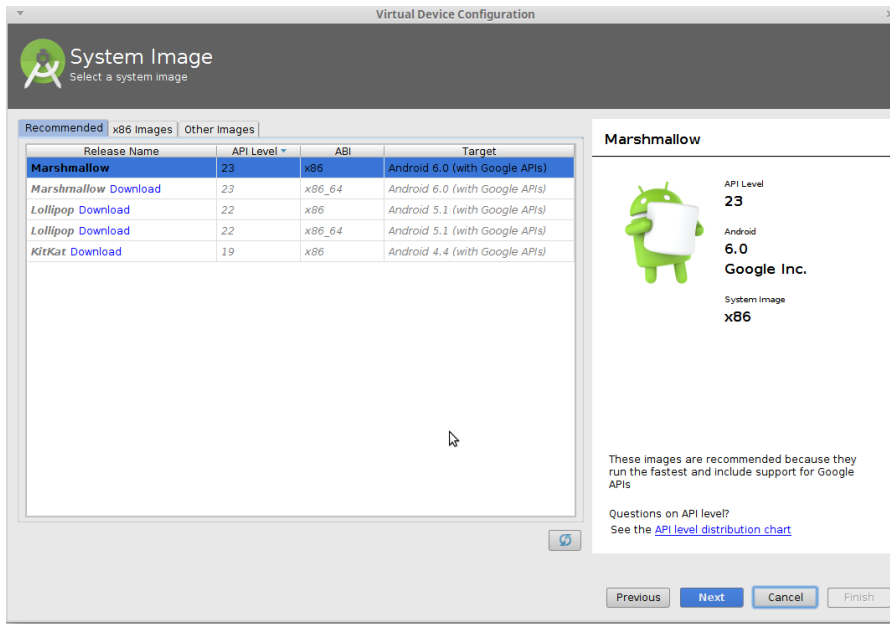


2. Em seguida clique em “Create Virtual Device”

3. Em categoria selecione a opção “phone” e em seguida o dispositivo que será emulado (normalmente este campo não precisa ser modificado).



4. Selecione a versão do android que o dispositivo irá possuir



5. Na tela seguinte basta clicar em finish. A tela seguinte irá aparecer e basta clicar no ícone “play” para inicia-la.

Não se preocupe caso demore um pouco para que o emulador inicie.

Se tudo der certo, é como mostra a figura abaixo que deverá aparecer em sua tela:



Para executar uma aplicação basta clicar no botão “run” do menu superior do Android Studio e escolher o emulador desejado.

Capítulo 2

Programando no Android Studio

Componentes de Visualização

São vários os componentes de visualização do Android Studio, vamos apresentar os principais. São estes:

- TextView
- EditText
- CheckBox
- Button

Utilizando TextView

O TextView permite inserir campos de texto em nossa aplicação. Existem quatro componentes textView. São estes:

- Plain TextView
- Large Text
- Medium Text
- Small Text

A diferença entre eles é simplesmente o tamanho.

Utilizando EditText

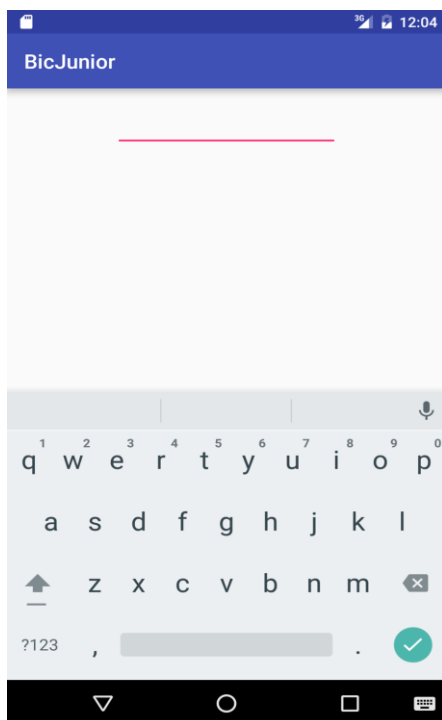
O EditText é um campo que permite que o usuário insira o texto, é bastante utilizado onde o usuário precise informar o nome ou alguma informação importante.

Os principais componentes EditText são:

- Plain Text
- Password
- E-mail

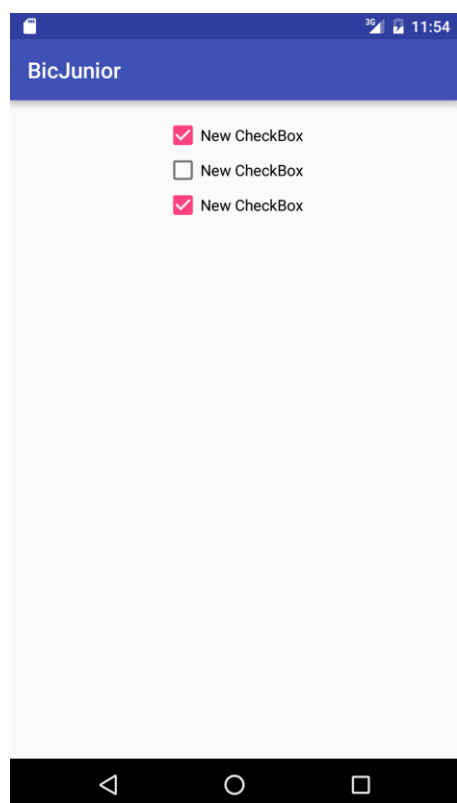
- Phone
- Date
- Number

A diferença entre eles é o tipo de teclado (numérico, presença do ícone “@” e outros) que aparece quando o usuário clica no campo para digitar.



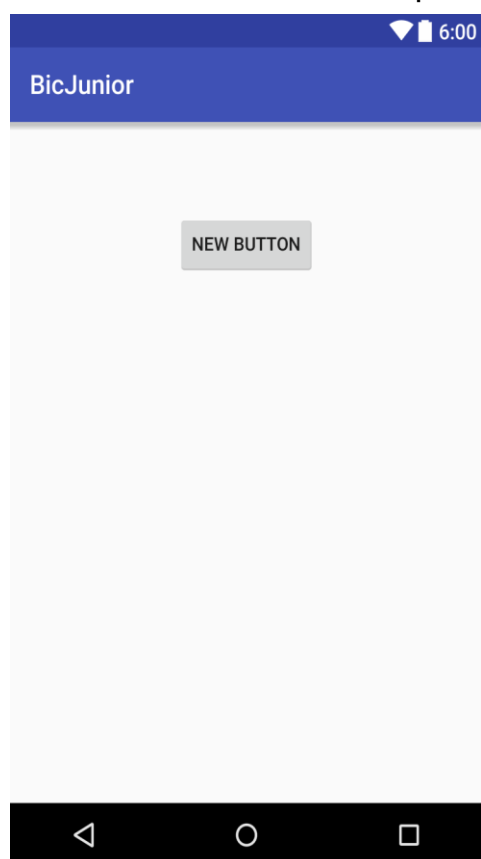
CheckBox

O checkBox nada mais é que caixas de seleção, como mostra a figura abaixo, para que ele seja útil no aplicativo é necessário programação assim como o Button, algo que veremos em breve.



Button

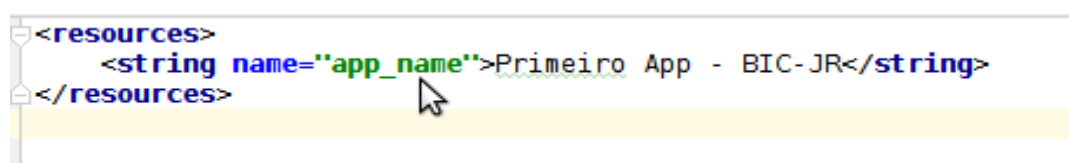
O Button nada mais é do que um botão no aplicativo.



É importante ressaltar que não vamos ficar presos a estes componentes, ao longo desta apostila caso necessário faremos uso de outros para que você possa conhecer e aprender como utiliza-lo.

4.0. Primeira aplicação

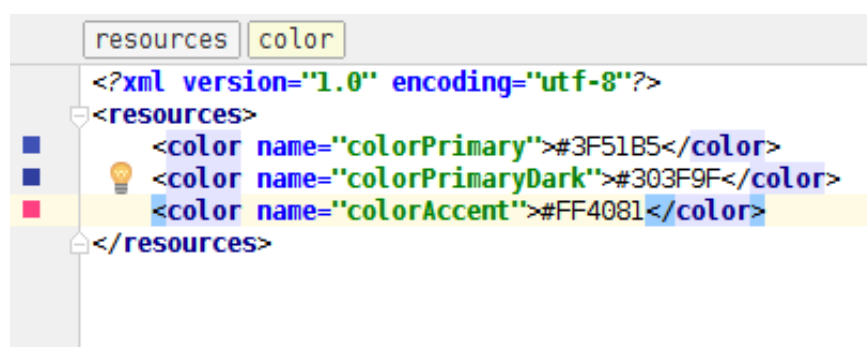
Vamos dar início à nossa primeira aplicação. Primeiramente, vamos ao diretório “res/values”, e em seguida no arquivo strings.xml, como você pode ver na figura abaixo. É nele que determinaremos o nome de nosso aplicativo, ou seja, o nome que ficará ao topo em todas as telas ou na maioria delas. Altere para o nome que desejar.



```
<resources>
    <string name="app_name">Primeiro App - BIC-JR</string>
</resources>
```

Agora vamos alterar a cor: para isso basta ir no arquivo colors.xml. As cores são definidas a partir do código hexadecimal, não se preocupe em decorar ou tentar adivinhar as cores que deseja, basta buscar na internet por uma tabela de cores.

No arquivo colors.xml, você irá se deparar com o seguinte código:



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Para alterar as cores básicas de sua aplicação basta clicar nos “quadrados” coloridos a esquerda e escolher a cor desejada. O nome da tag <color> é o que irá definir onde a cor será modificada, ou seja “colorPrimary” modificará a cor principal do topo de sua aplicação, o “colorPrimaryDark” modificará a cor um pouco acima da barra principal da aplicação.

Obs: colorAccent não será usado por enquanto.

Para ver o resultado basta abrir a aba design do arquivo Activity_main.xml, e pronto já demos uma visão nova para nosso projeto.



Agora vamos alterar o layout do nosso projeto, para isso vá até o arquivo Activity.main.xml. Provavelmente o texto existente é “hello world”, para altera-lo basta clicar duas vezes sobre ele e uma janela irá se abrir para que você possa colocar o texto que deseja, caso seja de sua vontade remove-lo basta clicar uma vez sobre ele e clicar no botão delete de seu teclado.

No menu à direita já mostrado acima (*imagem 1* item 4 em verde) é possível adicionar elementos ao seu layout. Para isso basta clicar e arrastar para a tela, no exemplo abaixo foram utilizados “*Large Text, Button e Person name*”



4.1 Como adicionar uma Imagem

Para adicionar uma imagem primeiramente escolha a imagem desejada e copie para a pasta drawable, em seguida clique e arraste o elemento “imageView” para o layout. Em seguida vá na aba text do arquivo activity_main.xml e você verá o seguinte código:

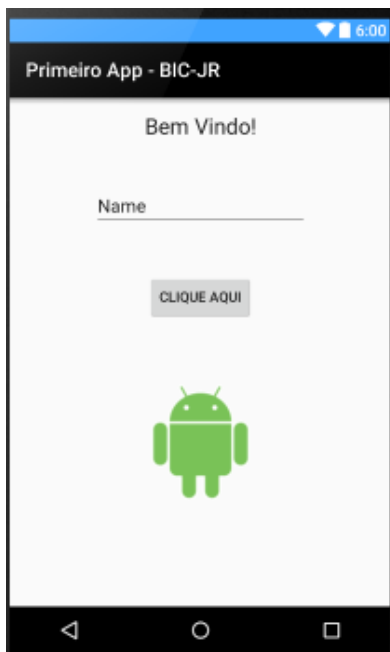
```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:layout_below="@+id/botao"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="63dp" />
```

Adicione a seguinte linha:

```
android:src="@drawable/user"
```

Obs.: No lugar de “user” coloque o nome de sua imagem.

Exemplo:



Até agora, todas as modificações feitas em nosso projeto foram apenas na View, ou seja, nosso botão ainda não executa nenhuma ação.

Para implementar a funcionalidade do botão vamos ao arquivo MainActivity.java.

O primeiro passo é criar dois atributos, um referente ao botão e outro ao campo texto:

```
public class MainActivity extends AppCompatActivity {
    //Atributos tipo Button e EditText
    private Button btn;
    private EditText nome;
}
```

Agora vamos associar os atributos aos elementos da view, para isso utilizamos o metodo **findViewById** que irá associar o elemento pelo seu id como pode ser visto abaixo:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //Associa Elemento com variavel a partir do id
    btn = (Button) findViewById(R.id.botao);
    nome = (EditText) findViewById(R.id.nome);
}
```

Por fim implementamos o método para a funcionalidade do botão;

O método usado é o **setOnClickListener**. No clique pega o campo de texto digitado pelo usuario e mostra uma mensagem, utilizando o AlertDialog, que é um alerta na tela quando o botão é pressionado.

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
1 AlertDialog.Builder dlg=new AlertDialog.Builder(MainActivity.this);
2 dlg.setMessage(nome.getText().toString() + ", Bem vindo(a)!");
3 dlg.setNeutralButton("OK", null);
4 dlg.show();
    }
});
```

No código a cima é importante ressaltar alguns elementos(Em negrito).

Linha 1

Utilizamos a classe **AlertDialog.Builder** (responsavel por criar caixas de dialogo e exibi-las) para instanciar o objeto **dlg**.

Linha 2

O método **setMessage** defini a mensagem que será exibida no alerta.

Linha 3

O método **setNeutralButton** define o botão "OK" da caixa de dialogo, o parametro **null** indica que ao clicar no botão não será executada nenhuma acao, ou seja, a caixa de dialogo será fechada.

Linha 4;

Por fim **dlg.show()** exibe a mensagem na tela assim que o botão é clicado.

O código final deverá estar da seguinte forma

```
public class MainActivity extends AppCompatActivity {
    //Atributos tipo Button e EditText
    private Button btn;
    private EditText nome;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Associa Elemento com variavel a partir do id
        btn = (Button) findViewById(R.id.botao);
        nome = (EditText) findViewById(R.id.nome);
        // Metodo para acao do clique no botão
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AlertDialog.Builder dlg = new AlertDialog.Builder(MainActivity.this);
                dlg.setMessage(nome.getText().toString() + ", Bem vindo(a)!");
                dlg.setNeutralButton("OK", null);
                dlg.show();
            }
        });
    }
}
```

Atividades:

- Adicione mais um campo de texto, e utilize-o na mensagem que é exibida no botão.

→ Adicione mais uma imagem na aplicação.

Desafio:

→ Mostre a mensagem de saudação utilizando um Toast no lugar do Alert.

Dica: *Toast é uma mensagem rápida que aparece para o usuário e desaparece depois de alguns segundos, faça uma pesquisa sobre como utiliza-lo e implemente-o.*

4.2 Indo mais além

O aplicativo anterior era foi simples e apenas uma forma de se familiarizar com alguns recursos do android studio, ao longo deste capítulo iremos nos aprofundar um pouco mais em questões de programação.

Trabalhando com Operadores

Como você já deve saber, em programação existem operadores: lógico, matemático e comparações, a imagem abaixo é uma tabela onde se encontra os principais operadores e suas funcionalidades.

Operadores Matemáticos	
Operação	Operador
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Resto	%

Comparação	
Operação	Operador

Igual	"=="
Diferente	"!="
Menor igual	"<="
Maior igual	">="
Maior	">"
Menor	"<"

Lógico	
Operação	Operador
E	&&
Ou	
Negação	!

Sabendo disso, vamos então fazer uso desses recursos, fazendo uma calculadora bem simples que realiza as 4 operações básicas.

Crie um novo projeto no android studio para iniciarmos.

O primeiro a se fazer é criar a View de seu projeto, ou seja a estrutura XML de nossa aplicação, no exemplo a baixo foram usados os elementos Button e EditText, o elemento EditText utilizado foi o "Number", para que possa aparecer um teclado numérico quando o usuário for inserir os números, impedindo que vá algum caractere indesejado.

Veja o exemplo abaixo (XML / Interface):

<Button

```

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Soma"
    android:id="@+id/soma"
    android:layout_below="@+id/textView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="39dp" />

```

<Button

```

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Subtração"
    android:id="@+id/sub"
    android:layout_below="@+id/soma"
    android:layout_alignLeft="@+id/soma"
    android:layout_alignStart="@+id/soma" />

```

<Button

```

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Divisão"
    android:id="@+id/div"
    android:layout_below="@+id/sub"
    android:layout_centerHorizontal="true" />

```

<Button

```

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Multiplicacao"
    android:id="@+id/mult"
    android:layout_below="@+id/div"
    android:layout_centerHorizontal="true" />

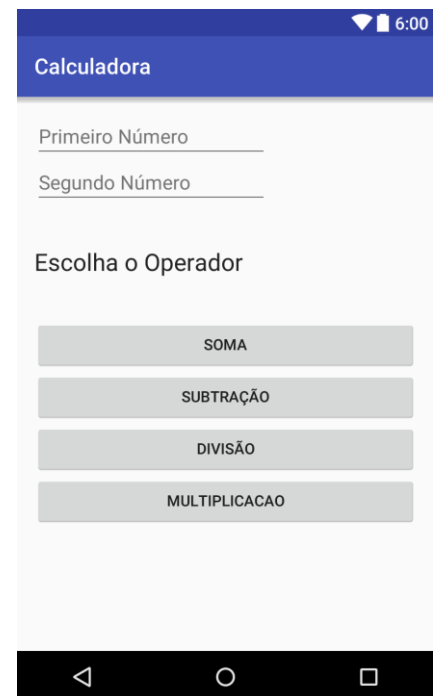
```

<TextView

```

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Escolha o Operador"
    android:id="@+id/textView"

```



```

    android:layout_below="@+id/n2"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="39dp" />
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/n1"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:hint="Primeiro Número" />
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/n2"
    android:layout_below="@+id/n1"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:hint="Segundo Número" />

```

Agora vamos ao arquivo MainActivity.java, para programar as funcionalidades de nossa calculadora.

Primeiramente, devemos declarar uma variavel para cada elemento de nossa interface e associa-las à seus respectivos elementos a partir do id.

```

public class MainActivity extends AppCompatActivity {

    private EditText n1, n2;
    private Button btsoma, btsub, btdiv, btmult;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

```

    n1 = (EditText) findViewById(R.id.n1);
    n2 = (EditText) findViewById(R.id.n2);
    btsoma = (Button) findViewById(R.id.soma);
    btsub = (Button) findViewById(R.id.sub);
    btmult = (Button) findViewById(R.id.mult);
    btdiv = (Button) findViewById(R.id.div);
}
}

```

Agora vamos atribuir para cada botão um método de clique, para que executem suas respectivas operações. Observe que na linha do código que executa a operação matematica usamos o operador aritimético.

```

public class MainActivity extends AppCompatActivity {

    private EditText n1, n2;
    private Button btsoma, btsub, btdiv, btmult;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        n1 = (EditText) findViewById(R.id.n1);
        n2 = (EditText) findViewById(R.id.n2);
        btsoma = (Button) findViewById(R.id.soma);
        btsub = (Button) findViewById(R.id.sub);
        btmult = (Button) findViewById(R.id.mult);
        btdiv = (Button) findViewById(R.id.div);
        //Método para somar
        btsoma.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                double num1 = Double.parseDouble(n1.getText().toString());
                double num2 = Double.parseDouble(n2.getText().toString());
                double soma = num1+num2; //Linha que soma os números
                AlertDialog.Builder resposta = new AlertDialog.Builder(MainActivity.this);
                resposta.setTitle("Resultado da Soma: ");
                resposta.setMessage("A resposta é: " + soma);
                resposta.setNeutralButton("OK", null);
                resposta.show();
            }
        });
    }
}

```

```

    }
});
//Método para subtrair
btsub.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        double num1 = Double.parseDouble(n1.getText().toString());
        double num2 = Double.parseDouble(n2.getText().toString());
        double sub = num1-num2; //Linha que subtrai os numeros
        AlertDialog.Builder resposta = new AlertDialog.Builder(MainActivity.this);
        resposta.setTitle("Resultado da Subtracao: ");
        resposta.setMessage("A resposta é: " + sub);
        resposta.setNeutralButton("OK", null);
        resposta.show();
    }
});

```

//Metodo para multiplicar

```

btmult.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        double num1 = Double.parseDouble(n1.getText().toString());
        double num2 = Double.parseDouble(n2.getText().toString());
        double mult = num1*num2; //Linha que multiplica os numeros
        AlertDialog.Builder resposta = new AlertDialog.Builder(MainActivity.this);
        resposta.setTitle("Resultado da Multiplicacao: ");
        resposta.setMessage("A resposta é: " + mult);
        resposta.setNeutralButton("OK", null);
        resposta.show();
    }
});

```

//Metodo para dividir

```

btdiv.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        double num1 = Double.parseDouble(n1.getText().toString());
        double num2 = Double.parseDouble(n2.getText().toString());
        double div= num1/num2;//linha que divide os numeros
        AlertDialog.Builder resposta = new AlertDialog.Builder(MainActivity.this);
        resposta.setTitle("Resultado da Divisao: ");
    }
});

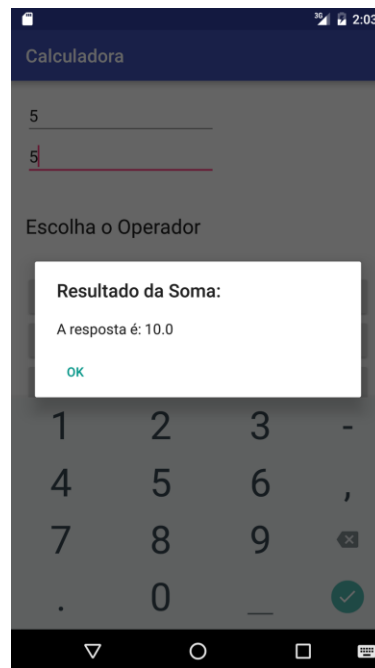
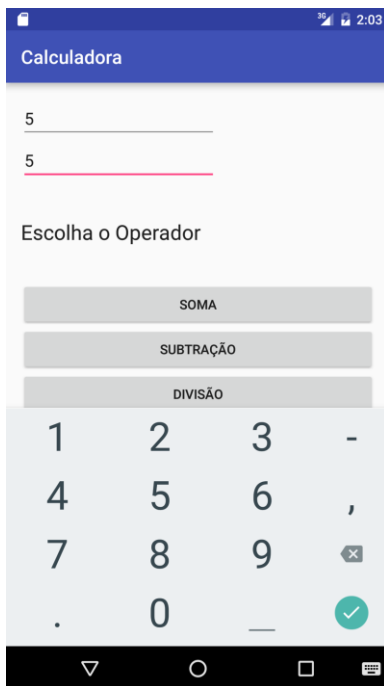
```

```

resposta.setMessage("A resposta é: " + div);
resposta.setNeutralButton("OK", null);
resposta.show();
}});}}

```

Após escrever o código a cima, teste o aplicativo, e veja o resultado.



Atividades:

- Faça um aplicativo que soma e multiplica 3 números diferentes
- Faça um aplicativo que soma dois números e informa se sua soma é maior ou menor que 30.

Dica: Use *if* e *else if* e operadores de comparação.

Desafio:

- Faça uma aplicação que receba como entrada o nome e data de nascimento do usuário, e ao clicar no botão “ok” o aplicativo retorna quantos anos a

pessoa tem e informa se ela é uma criança, um adolescente ou adulto.

X Criança: Menor de 12 anos.

X Adolescente: Maior ou igual a 12 e menor ou igual a 17.

X Adulto: Maior de 17.

Dica: Use *if* e *else if* e operadores de comparação.

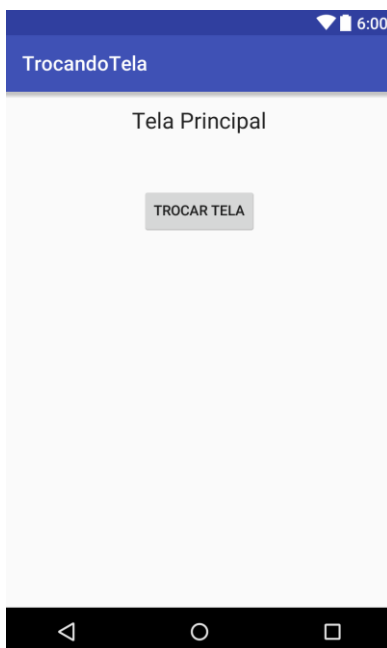
Trabalhando com múltiplas telas

Até agora todos aplicativos que fizemos possuíam apenas uma tela, porém aplicativos fazem uso de varias telas, e é isso que vamos aprender agora, para isso vamos utilizar uma [intent](#).

4.3 Intent

Intents são criadas normalmente a partir de interação com o usuário e objetiva indicar a intenção de se realizar algo. Existem dois tipos de intents, [intents implícita](#) e [intents explícitas](#). Para alternar entre activitys vamos utilizar [intents explícitas](#).

Vamos então criar o layout de nossa activity principal, como o objetivo é mostrar a tocla de tela vamos fazer algo bem simples apenas com texto e um botão, abaixo está a imagem e o código xml.



<TextView

android:layout_width="wrap_content"

android:layout_height="wrap_content"

```

android:textAppearance="?"android:attr/textAppearanceLarge"
  android:text="Tela Principal"
  android:id="@+id/txtPrincipal"
  android:layout_alignParentTop="true"
  android:layout_centerHorizontal="true" />
<Button
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Trocar Tela"
  android:id="@+id/btnTroca"
  android:layout_below="@+id/txtPrincipal"
  android:layout_centerHorizontal="true"
  android:layout_marginTop="52dp" />

```

O próximo passo é criarmos nossa segunda activity. Para isso basta clicar com botão direito no mouse sobre o diretório “app”, em seguida new > Activity > Empty Activity.

Nomeie o nome da activity e clique no botão finish. Como você pode ver foi adicionada uma nova activity e um arquivo xml para ela.

Creates a new empty activity

Activity Name:

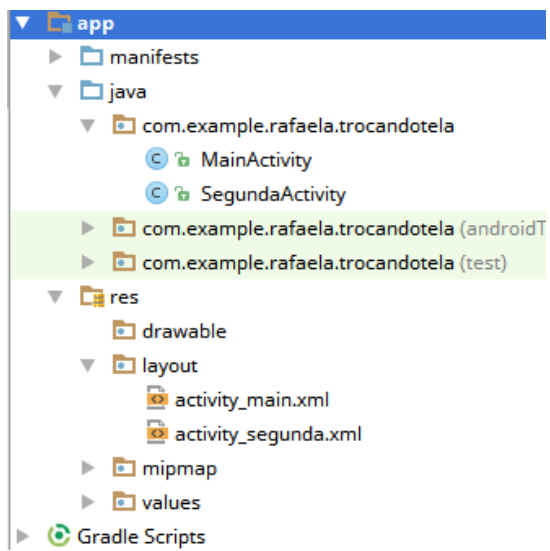
☒ Generate Layout File

Layout Name:

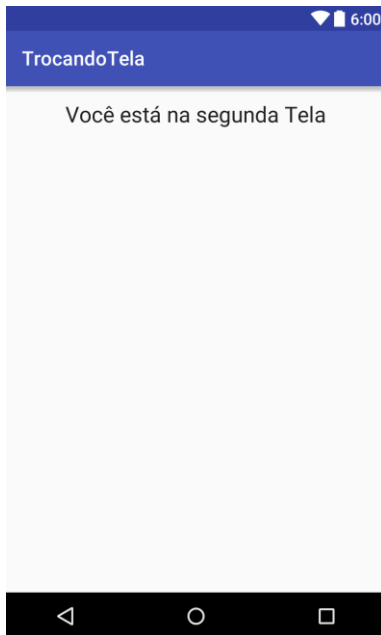
☐ Launcher Activity

Package name:

Target Source Set:



Agora vamos mudar o layout de nossa activity, para esta, vamos utilizar apenas um textView, porém não fique preso a isso, sintá-se livre para utilizar o layout que desejar em seu aplicativo.



Por fim vamos ao arquivo MainActivity.java para programarmos a troca de uma activity para outra. Veja o código abaixo:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button ok = (Button) findViewById(R.id.btnTroca);
        ok.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent trocaActivity = new Intent(MainActivity.this,
                SegundaActivity.class);
                startActivity(trocaActivity);
            }
        });
    }
}
```

Como podem ver temos 2 linhas que não havíamos utilizado antes, então vamos entender onde elas influenciam em nosso aplicativo.

```
Intent trocaActivity = new Intent(MainActivity.this, SegundaActivity.class);
```

Na linha a cima estamos instanciando um objeto do tipo Intent e passamos como parametro a activity de partida e a activity de destino.

```
startActivity(trocaActivity);
```

Na linha a cima, estamos iniciando a activity e passando como parametro a instância que foi criada anteriormente.

Feito os passos descritos anteriormente, teste e veja o resultado!

Passar parametros de uma activity

Passando Parametros

Primeiramente certifique-se que em seu projeto existam duas activitys, caso tenha apenas uma, crie uma outra como ensinamos no tópico anterior. Em seguida vamos para as alterações em nosso layout, abaixo estarão o xml de nossas duas activitys.

MainActivity

<EditText

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:id="@+id/edtNome"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:hint="Nome" />
```

<Button

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Nova Tela"
android:id="@+id/btnNovaTela"
```

```

android:layout_below="@+id/edtNome"
android:layout_centerHorizontal="true"
android:layout_marginTop="41dp" />

```

SegundaActivity

<TextView

```

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textAppearance="?android:attr/textAppearanceLarge"
android:id="@+id/txtTela2"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true" />

```

No arquivo MainActivity.java vamos repetir o procedimento do tópico anterior, porém vamos incrementar a seguinte linha:

```
<instancia>.putExtra("<b>chave</b>", "<valor>");
```

O método putExtra serve para adicionar à um objeto o valor para uma intent para enviar para outra Activity, ou seja um parametro.

Veja o código completo do arquivo MainActivity.java

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText nome = (EditText) findViewById(R.id.edtNome);
        Button btn = (Button) findViewById(R.id.btnNovaTela);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent ok = new Intent(MainActivity.this, SegundaActivity.class);
                ok.putExtra("parametro", nome.getText().toString());
                startActivity(ok);
            }
        });
    }
}

```

```

    });
}
}

```

Obs.: final significa que a variavel não poderá assumir outro valor

Agora vamos editar o arquivo SegundaActivity.java.

```

public class SegundaActivity extends AppCompatActivity {

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

        1 Intent ok = getIntent();
          TextView txt = (TextView) findViewById(R.id.txtTela2);
        2 String nome = (String) ok.getSerializableExtra("parametro");
        3 txt.setText("Olá," + nome);
    }
}

```

No código a cima é importante ressaltar alguns elementos(Em negrito).

Linha 1:

O método **getIntent()** pega a intent.

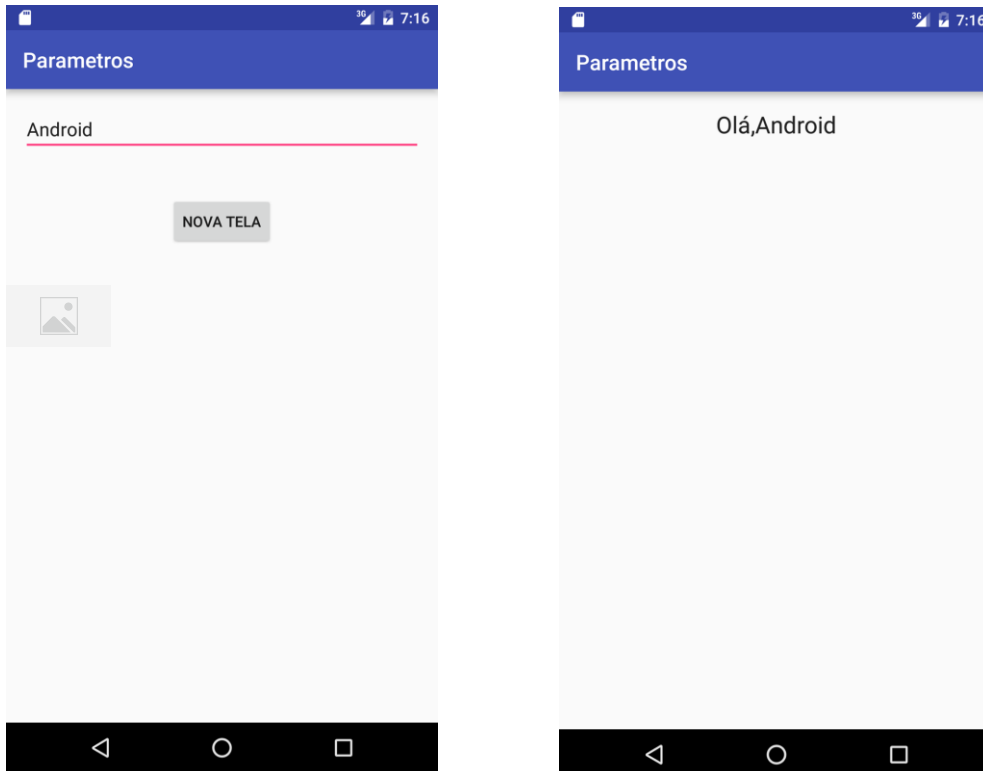
Linha 2:

Usa o nome do nosso parametro e o guarda em uma variavel que no caso é “nome”.

Linha 3:

Exibe o texto.

Caso tudo ocorra bem, é a seguinte tela que deverá aparecer ao testar o aplicativo:



Passando Parâmetros dentro de um Pacote (Bundle)

Para passar parâmetros dentro de um pacote é necessário que todos os dados estejam dentro dele, ou seja, caso seja necessário enviar para outra Activity mais de um parâmetro torna-se mais fácil passar parâmetros dentro do pacote. Para isso vamos utilizar o Bundle. Veja o exemplo a seguir de como fazer isso:

Arquivo MainActivity.java

```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
```

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

```
        super.onCreate(savedInstanceState);
```

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

```
        final EditText nome = (EditText) findViewById(R.id.edtNome);
```

```
        Button btn = (Button) findViewById(R.id.btnNovaTela);
```

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

```
            @Override
```

```
            public void onClick(View v) {
```



```

        Intent ok = new Intent(MainActivity.this, SegundaActivity.class);
        Bundle dados = new Bundle();
        dados.putString("NomeParametro", nome.getText().toString());
        dados.putInt("IdadeParametro", 18);
        ok.putExtras(dados);
        startActivity(ok);
    }
});
}
}

```

Arquivo SegundaActivity.java

public class SegundaActivity extends AppCompatActivity {

@Override

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_segunda);
    Intent ok = getIntent();
    Bundle info = ok.getExtras();
    TextView txt = (TextView) findViewById(R.id.txtTela2);
    String nome = info.getString("NomeParametro");
    int idade = info.getInt("IdadeParametro");
    txt.setText("Olá," + nome + ". Sua idade é: " + idade);
}
}

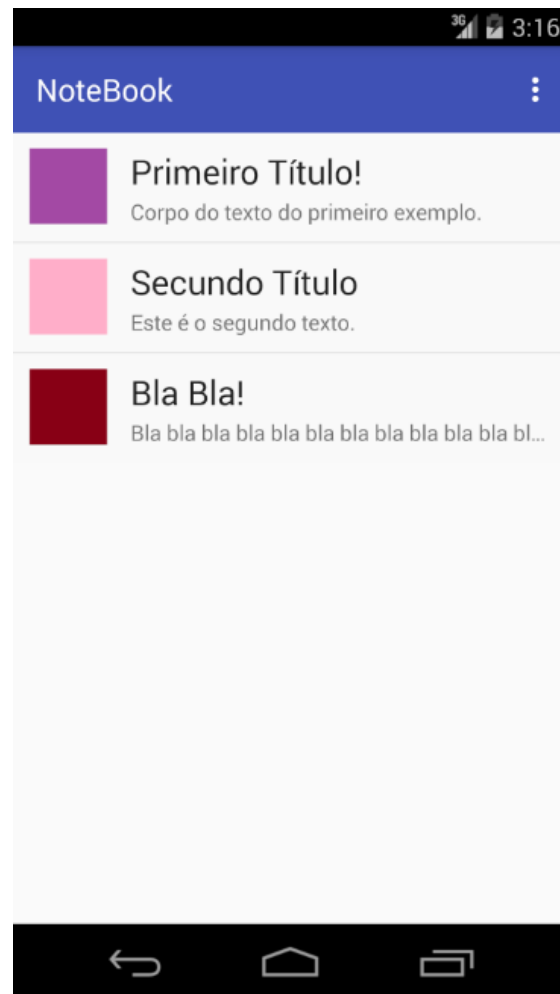
```

Como você deve ter percebido a lógica é a mesma, nomeamos dois parametros e os regatamos.

4.4 Entendendo Arrays

Muitas aplicações Android necessitam que sejam exibidas para usuários listas de informações, sejam estas unitárias, ou em conjunto. Para a construção de tais listas no Android, utilizamos Arrays, que são um conjunto em ordem de informações, um vetor.

[Exemplo de aplicação que utiliza Array:](#)



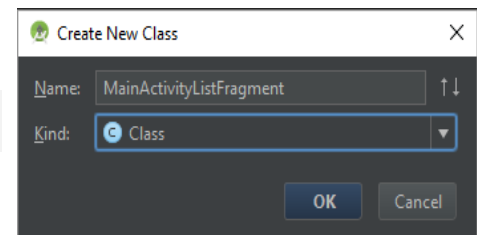
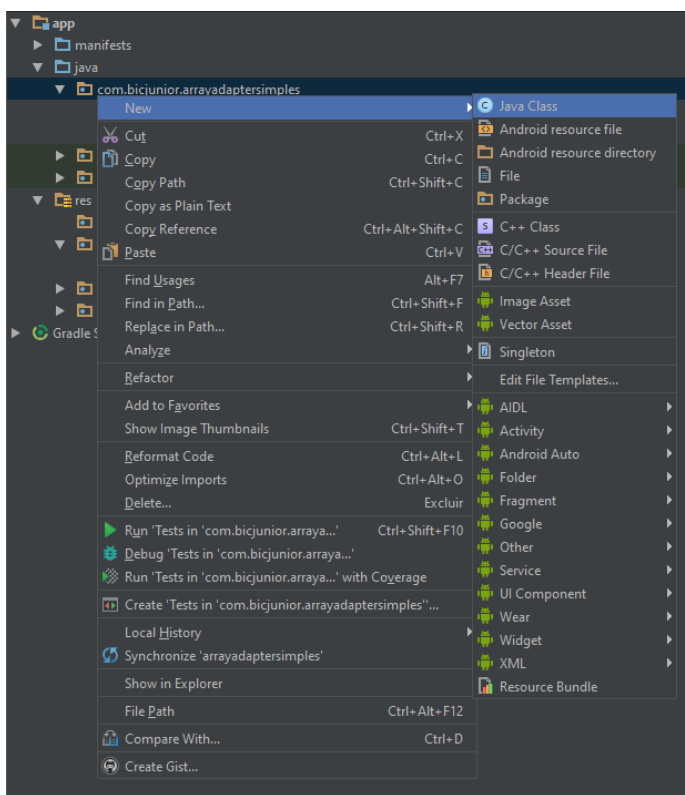
Neste exemplo, cada uma dessas caixas, separadas por uma pequena linha cinza é uma nova instanciação – ou seja, um novo objeto – da classe Notas.java. Em nossa próxima aplicação, aprenderemos a construir o ArrayAdapter acima.

4.5 Aplicação usando ArrayAdapter

Para a exemplificação do uso de arrays, construímos um aplicativo simples que mostra uma lista (“array”) dos nomes de sistemas operacionais.

Criação do fragmento

Devemos primeiramente criar um fragmento, que conterá nossa lista de Sistemas Operacionais. Para criar essa classe, siga as instruções abaixo.



Criada a classe, devemos informar ao AndroidStudio que ela se trata de um fragment, e para fazer isso, basta criarmos uma classe que estende a já implementada pelo Android ListFramgent, como pode ver no código abaixo.

```
public class MainActivityListFragment extends ListFragment{  
}
```

Em seguida, devemos criar, dentro desta classe um método que será executado quando ao ListFragment é executado, que chamamos de método de callback.

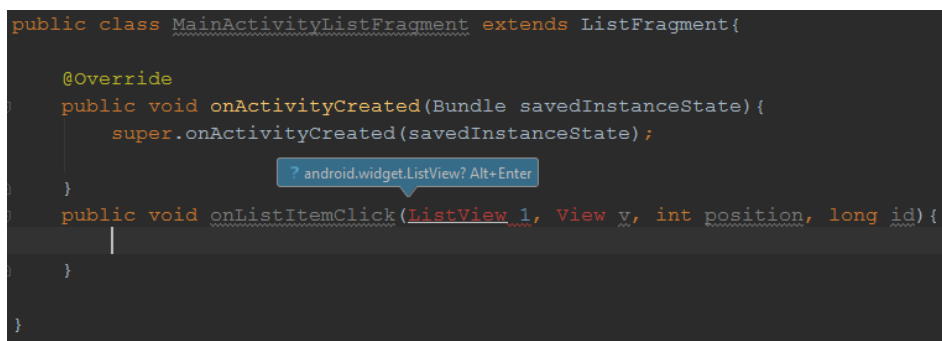
```
@Override  
public void onActivityCreated(Bundle savedInstanceState){  
    super.onActivityCreated(savedInstanceState);  
}
```

Agora, criaremos um método para definir o posicionamento de cada um dos objetos do ListView. “l, v, position, id” são os parâmetros do método.

@Override

```
public void onItemClick(ListView l, View v, int position, long id){  
    super.onItemClick(l, v, position, id);  
}
```

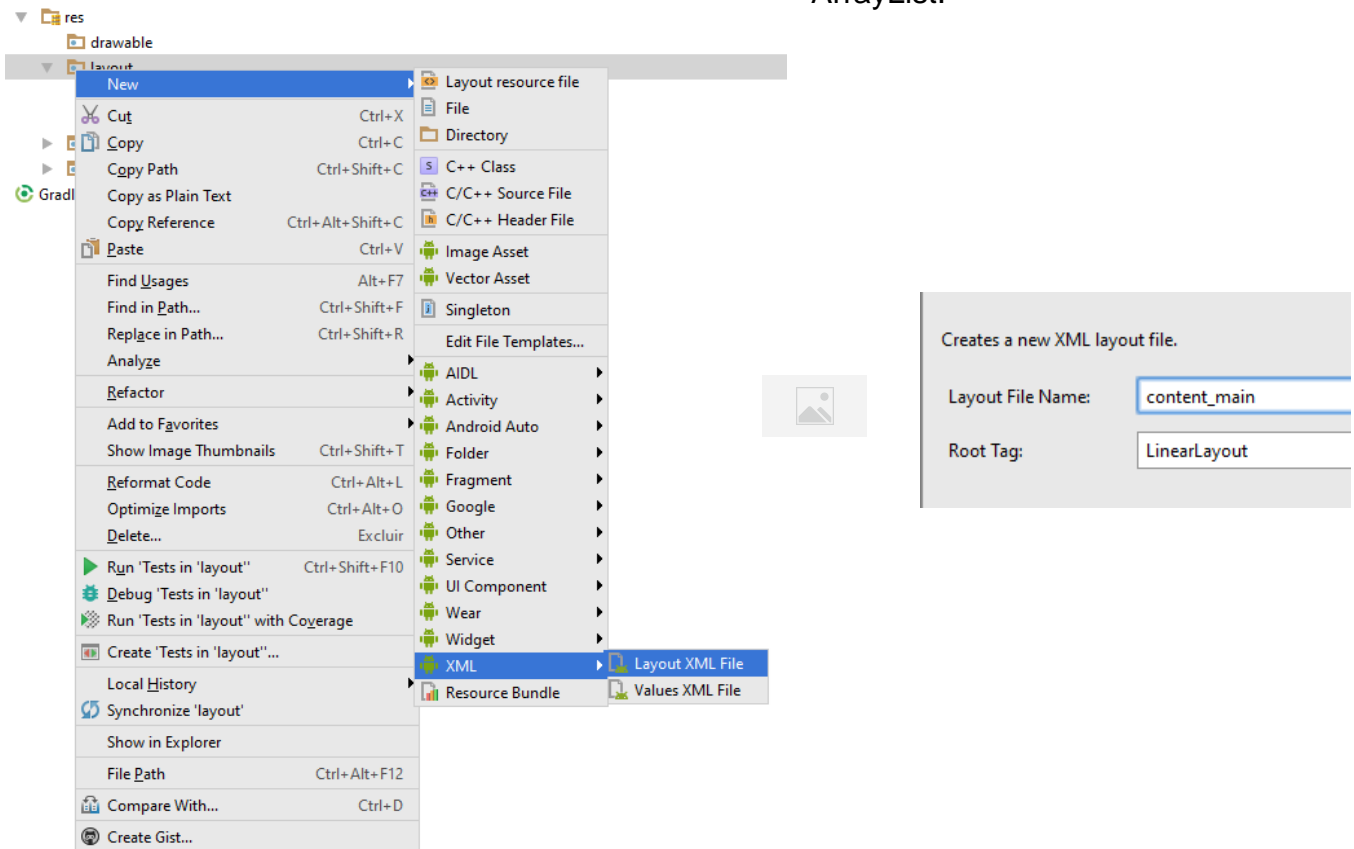
Obs: quando for implementar o programa, tome sempre o cuidado de implementar as bibliotecas necessárias para tudo que está usando. Caso uma linha fique vermelha, o Android vai lhe aconselhar importá-las, e pode fazer isso apertando “alt-enter”.



```
public class MainActivityListFragment extends ListFragment{  
  
    @Override  
    public void onActivityCreated(Bundle savedInstanceState){  
        super.onActivityCreated(savedInstanceState);  
    }  
    public void onItemClick(ListView l, View v, int position, long id){  
        |  
    }  
}
```

Codificação do Layout

Devemos criar, para a utilização deste fragment, um novo arquivo .xml de layout, que chamaremos de `content_main`, por possuir, como já deve ter imaginado, o conteúdo do `ArrayList`.



Arquivo content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.bicjunior.arrayadaptersimples.MainActivity"
    tools:showIn="@layout/activity_main">
    <fragment
        android:id="@+id/mainActivityFrag"
        android:name="com.bicjunior.arrayadaptersimples.MainActivity"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

Este código, especificamente a tag “fragment” adiciona este elemento em nossa tela. Observe a nona linha: “**tools:showIn="@layout/activity_main"**” define em que arquivo este layout será exibido, no caso, o arquivo activity_main.

Arquivo activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.bicjunior.arrayadaptersimples.MainActivity">
    <include layout="@layout/content_main"/>
</RelativeLayout>
```

Em seguida, devemos construir nosso vetor com os valores que desejamos exibir na tela,

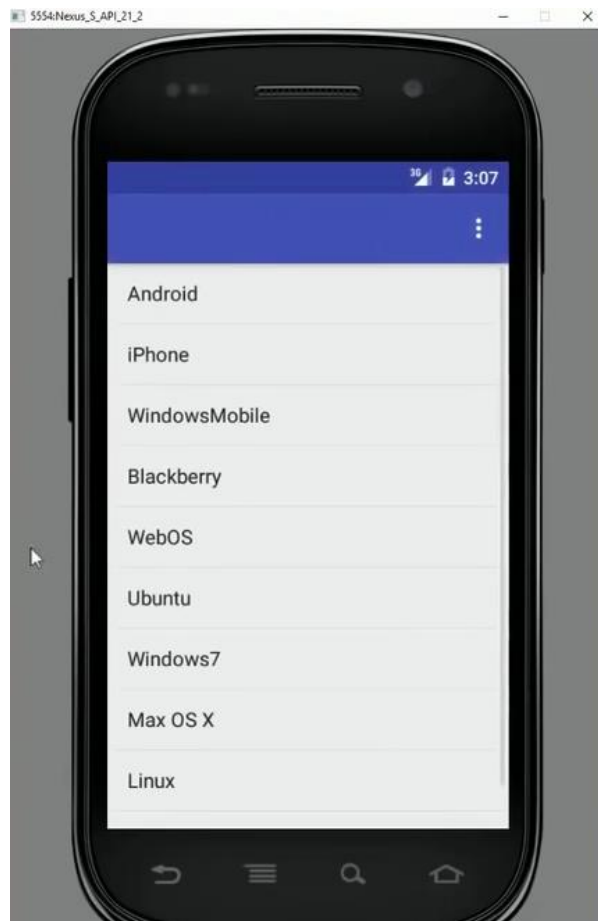
este código deve ser inserido após a linha “`super.onCreate(savedInstanceState);`”:

```
String[] values = new String[]{"Android", "IPhone", "WindowsMobile", "Blackberry",  
"WebOS", "Ubuntu", "Windows7", "Max OS X", "Linux"};
```

Agora, para finalizar, criamos o ArrayAdapter, com os parâmetros sendo, respectivamente, um método para chamar a atividade, um layout já pronto do Android para listar e nosso vetor de itens:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(getActivity(),  
    android.R.layout.simple_list_item_1, values);  
setListAdapter(adapter);
```

Executando:



Unidade 2

Capítulo 3

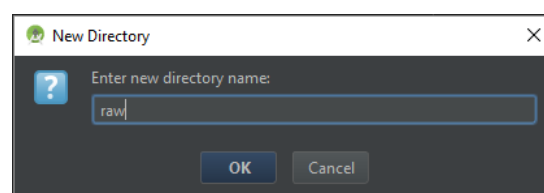
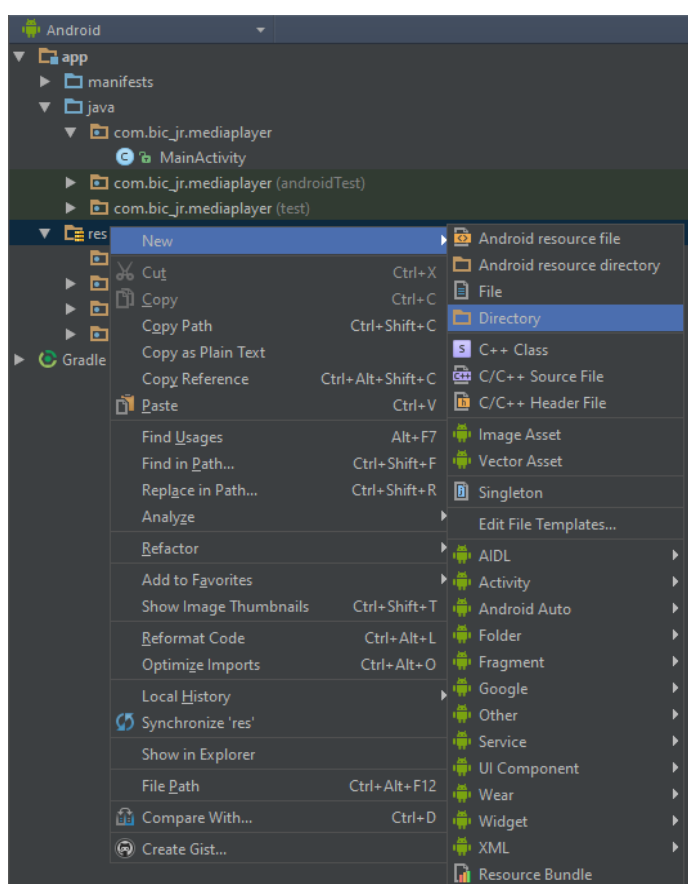
Executando e compartilhando arquivos

5.0 Media Player

É possível, em Java, criar uma função cujo objetivo é tocar um ou vários diferentes áudios ou vídeos em determinado evento. No exemplo a seguir, criaremos dois botões que, ao serem clicados, tocarão seus respectivos áudios.

Criação do diretório

Para mantermos a organização, vamos criar um diretório que armazenará os arquivos de áudio. Por padrão, esta pasta deve se chamar 'raw', e fica dentro da pasta 'res'.



Adicionar arquivos

Agora devemos adicionar os arquivos de áudio na recém-criada pasta 'res/raw' em nosso projeto.

Obs: os arquivos de áudio não devem conter caracteres especiais, como acentos ou barras, apenas letras minúsculas, números e underline.



Arquivo layout_main.xml

Vamos criar agora dois botões simples em nosso layout inicial, cada um responsável por executar seu respectivo arquivo de áudio.

<Button

```

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Áudio 1"
android:id="@+id/btn_audio1"
android:layout_alignParentTop="true"
android:layout_alignParentStart="true" />

```

<Button

```

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Áudio 2"
android:id="@+id/btn_audio2"
android:layout_alignBottom="@+id/btn_audio1"
android:layout_alignParentEnd="true" />

```

Iniciando MainActivity.xml

Em nossa classe principal, vamos primeiramente iniciar os botões do layout e criar os métodos *onClickListener* de ambos.

```
Button btn_audio1 = (Button) findViewById(R.id.btn_audio1);
btn_audio1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

    }
});
Button btn_audio2 = (Button) findViewById(R.id.btn_audio2);
btn_audio2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

    }
});
```

Para tocar um áudio no Android, precisamos criar uma variável nativa do Java chamada de *MediaPlayer*. Para ela, podemos atribuir um arquivo de áudio, e, em seguida, chamar a função *.start()*.

Primeiramente, declare a variável *MediaPlayer* fora da função *onCreate*:

```
MediaPlayer mp = null;
```

Em seguida, crie dentro das respectivas funções *onClickListener*, o seguinte código:

```
mp = MediaPlayer.create(getApplicationContext(), R.raw.audio_1);
mp.start();
```

Observe que como último parâmetro da função *.create()*, passamos o caminho do áudio que inserimos na pasta 'raw'.

Agora, o áudio já será tocado ao clicarmos em seus respectivos botões, mas caso o usuário

clique várias vezes nos botões, os áudios se sobreporão. Portanto, opcionalmente, podemos criar um método que cancele o tocar de um áudio ao clicarmos novamente em um dos botões.

Crie fora da função `onCreate` o seguinte método:

```
private void pararAudio() {  
  
    if (mp!=null) {  
        mp.stop();  
        mp.release();  
        mp = null;  
    }  
}
```

E chame, em cima das linhas `mp.start()`, a função;

```
pararAudio();
```

6.0 Compartilhando áudio

Compartilhar arquivos de áudio, vídeo, etc.(que não sejam de texto) é um pouco mais complicado e requer duas etapas: a primeira é a de copiar o arquivo para o dispositivo Android do usuário, e a segunda é usar um *intent* explícito para compartilhá-lo. Vamos atribuir a codificação a um método *onLongClickListener*, como no exemplo abaixo.

```
btn_audio1.setOnLongClickListener(new View.OnLongClickListener() {
    public boolean onLongClick(View arg0) {

    }
});

...

btn_audio2.setOnLongClickListener(new View.OnLongClickListener() {
    public boolean onLongClick(View arg0) {

    }
});
```

6.1 Copiando para o dispositivo

Para copiar arquivos para o dispositivo, crie o método a seguir fora da função onCreate. Chamaremos o método mais tarde:

```
private void copiarDispositivo(int id, String caminho) throws IOException {
    InputStream in = getResources().openRawResource(id);
    FileOutputStream out = new FileOutputStream(caminho);
    byte[] buff = new byte[1024];
    int read;
    try {
        while ((read = in.read(buff)) > 0) {
            out.write(buff, 0, read);
        }
    }
}
```



```
    } finally {  
        in.close();  
        out.close();  
    }  
}
```

Esta função pede como parâmetro um id (expressões como R.id.... são interpretadas pelo AndroidStudio como um número inteiro), e um caminho, onde o arquivo será baixado.

Em seguida, dentro das respectivas funções *onLongClickListener*, implemente o código abaixo:

```
File pasta = new File(Environment.getExternalStorageDirectory().toString()+"/Audio/");
if(!pasta.exists()) {
    pasta.mkdirs();
}
```

Esta parte cria a pasta 'Audio' dentro do diretório de armazenamento do dispositivo.

```
String caminho = Environment.getExternalStorageDirectory().toString() +
"/Audio/";
File dir = new File(caminho);
try {
    if (dir.mkdirs() || dir.isDirectory()) {
        copiarDispositivo(R.raw.audio_1, path + File.separator + ("audio_1.mp3"));
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Esta parte chama a função *copiarDispositivo* com seus respectivos parâmetros sendo o caminho do arquivo na pasta 'raw' (int), e o caminho onde será gravado o arquivo(String).

6.1 Construindo o Intent

Devemos agora construir uma intenção que o Android interpretará e solicitará ao usuário que decida com qual aplicação ele a quer usar. Esse código deve estar logo abaixo do anterior.

```
final Intent shareIntent = new Intent(android.content.Intent.ACTION_SEND);
shareIntent.setType("audio/mp3");
```

Aqui nós criamos o Intent e informamos qual o tipo de arquivo estamos compartilhando, no caso, um arquivo .mp3.

Obs: também é possível informar o Intent como `.setType("audio/*")`, que funcionará para todos os tipos de arquivo de áudio, porém o aplicativo WhatsApp não aceitará arquivos de sua aplicação.

```
shareIntent.putExtra(android.content.Intent.EXTRA_STREAM,  
Uri.parse(Environment.getExternalStorageDirectory().toString()  
    + "/Audio/" + ("audio_1.mp3")));  
startActivity(Intent.createChooser(shareIntent, "Compartilhar via"));  
longclicked=true;  
return false;
```

Já este código inicializará o Intent, carregando para uma aplicação externa que aceita arquivos .mp3 escolhida pelo usuário, o nosso arquivo audio_1.mp3 ou audio_2.mp3.

Referências

<http://corporate.canaltech.com.br/o-que-e/programacao/O-que-e-Java-JRE-JVM-e-JDK/>

<http://www.cin.ufpe.br/~phmb/ip/MaterialDeEnsino/IntroducaoAoJDK/IntroducaoAoJDK.html>

<https://www.vivaolinux.com.br/artigo/Instalacao-e-Configuracao-do-JDK-7/>

<https://www.edivaldobrito.com.br/instalar-android-studio-no-ubuntu/>

<http://itsfoss.com/install-android-studio-ubuntu-linux/>

<http://www.androidnaveia.com.br/2011/02/desenvolvimento.html>

<http://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm>

<http://www.devmedia.com.br/android-intent-class-trabalhando-com-multiplas-telas-em-android/30496>