



Desenvolvimento de Aplicações Android

@ramonrabello

@androidnarede



SoftSell®



Sobre o treinamento

Duração: 40h (02/09 - 06/09 | 8h30 - 17h30)

Totalmente *hands-on*. Você não perde nenhum passo!

Um curso extensivo sobre como desenvolver apps profissionais em Android

Totalmente prático. Muitos exercícios mão na massa!

Focado em boas práticas

Guia prático + slides das aulas

Construção de uma app por time



Sobre o instrutor

Nascido em
Belém do
Pará

Acompanha
Android desde o
íncio

Idealizador do
“Android na Rede”

Consultor
Android

Desenvolvedor
de Software há 9 anos



Palestrante da
AndroidConf Brasil '11

Membro ativo da
Comunidade Ágil
Tá Safo!

Entusiasta de
Agilidade

Desenvolvedor na
BB Tecnologia e
Serviços

Nerd com
vida social

about.me/ramonrabello



Sobre o Android na Rede

Objetivo de divulgar
a **plataforma Android**
inicialmente em
Belém

Redes sociais
sobre o
ecossistema
Android



Redes sociais
sobre o
ecossistema
Android

**Coaching e
Mentoring**
In Company

Serviço de
Consultoria

Evangelização
de Android no
Brasil



Outros treinamentos



Dominando Android
Criando Applicativos de
Sucesso
(Edukee.com)



Dominando Android
Desenvolvimento Ágil de
Apps Orientado a Testes
(Teresina-PI)



Dominando Android -
Criando Apps Sociais
(Edukee.com)



Dominando Android
Da Ideia ao Sucesso
(Belém-PA)



Dominando Android
Android para Designers



Dominando Android
Android para Web
Developers



Agora fale sobre você!

Nome

Profissão

Suas Experiências

O que espera do curso?



Orientações iniciais

**Perguntem
sempre!**

**Atenção ao passo
a passo!** Você
terá tempo para
praticar, não se
preocupe!

utilizar
hashtag
#treinamento
AndroidNaRede

Aqui é um **bate-papo!**
Sintam-se a vontades
para **compartilhar**
conhecimento.

**Explore o Guia
prático para
desenvolvimento
Android.**



O que você irá aprender?

CONCEITOS ESSENCIAIS

AMBIENTE DE DESENVOLVIMENTO

O que é Android?

Por que devo aprender Android?

Curiosidades, Mitos e Fatos

Doces e mais doces: a evolução da plataforma

O Framework Android

Fundamentos essenciais

Máquina Dalvik



O que você irá aprender?

ITOS ESSENCIAIS

AMBIENTE DE DESENVOLVIMENTO

INTERFACE GRÁFI

Android
Developer
Guide

Java vs. Android

Ferramentas
necessárias

Explorando
o Android SDK

Explorando
a IDE ADT

Criando projetos
Android

Gerenciando versões
da plataforma

Estrutura de um
projeto Android

Criando
dispositivos
virtuais (AVDs)

Executando
uma app
Android

Depurando
com o LogCat



O que você irá aprender?

TE DE DESENVOLVIMENTO

INTERFACE GRÁFICA

TELAS

MAPAS

Criando e acessando
recursos em XML

Views e
ViewGroups

Principais
Widgets

Principais
Layouts

Criando e
utilizando novos
Widgets

Drawables

AdapterViews e
Listas

Enriquecendo sua
UI com Gradiantes
e Seletores

Tratamento
de eventos



O que você irá aprender?

MENTO INTERFACE GRÁFICA

ACTIVITY

PERSISTÊNCIA

MAPAS E GPS

Entendo
melhor uma
Activity

Adicionando
outras Activities
ao projeto

O ciclo de
vida de uma
Activity

Iniciando e
finalizando
uma Activity

Passagem de
parâmetros para
outra Activity

Relação entre
Views e
Activities



O que você irá aprender?

GRÁFICA

ACTIVITY

PERSISTÊNCIA

MAPAS E GPS

COMUNIC

Criando e
conectando
com um banco
de dados
usando SQLite

Explorando as
operações SQL
no SQLite

Criando um
simples
CRUD

Trabalhando
com
Preferências

Criando uma
tela de
preferências
em XML

Acessando a
lista de
contatos com
ContentProvider



O que você irá aprender?

PERSISTÊNCIA

MAPAS E GPS

COMUNICANDO O SERVIDOR

Conhecendo
a API de
Mapas

Trabalhando
com
Mapas

Conhecendo
a API de
Localização

Tipos de
visualização
no mapa

Manipulando
mapas

Obtendo
informações
geográficas
com Geocoder

Obtendo sua
localização
GPS atual

Enriquecendo
mapas com
Overlays

“Escutando”
mudança de
localização



O que você irá aprender?

PERSISTÊNCIA

COMUNICANDO COM O SERVIDOR

RECURSOS MULTIMÉDIA

Enviando dados
via HTTP com a
biblioteca Apache
HttpClient

Obtendo
resposta do
servidor

Consumindo
Sockets

Consumindo
WebServices

Utilizando
tarefas
assíncronas
para tarefas
custosas



O que você irá aprender?

ANDO O SERVIDOR

RECUSOS MULTIMÍDIA E CÂMERA

BROADCAST REC

Formatos
de áudio e
vídeo

Reproduzindo
áudio

Exibindo vídeo
com
VideoView

Adicionando
permissão
para acessar
câmera

Acessando a
câmera para
tirar foto

Aguardando
o resultado
retornado
por outra
Activity

Lendo e
salvando
imagens no
cartão de
memória

Exibindo o
preview da
imagem



O que você irá aprender?

SOS MULTIMÍDIA E CÂMERA

SERVIÇOS

BROADCAST RECEIVERS

Criando
um serviço

Registrando
um serviço no
arquivo de
manifesto

Entendendo
funcionamento
de um Serviço

Ciclo de vida
de um serviço

Iniciando um
serviço



O que você irá aprender?

ENDO O SERVIDOR

BROADCAST RECEIVERS

OUTROS RECURSOS

Entendendo o
funcionamento
de um Broadcast
Receiver

Criando um
Broadcast
Receiver

Registrando um
Broadcast Receiver
no arquivo de
manifesto

Inicializando
um Broadcast
Receiver



O que você irá aprender?

DADCAST RECEIVERS

OUTROS RECURSOS

NOTIFICAÇÕES

SENSORES

Entendo os
recursos
alternativos

Internacionalizando
sua app com outros
idiomas

Fornecendo layouts
alternativos para
resoluções diferentes



O que você irá aprender?

BROADCAST RECEIVERS

NOTIFICAÇÕES

SENSORES

GOOGLE PL

Por que usar
notificações?

Obtendo o serviço
para trabalhar com
notificações

Estrutura de
uma notificação

Criando
notificações

Notificando o
usuário



O que você irá aprender?

NOTIFICAÇÕES

SENSORES

GOOGLE PLAY

Tipos de
Sensores

Trabalhando
com
Acelerômetro

Obtendo serviço
para trabalhar
com sensores

Trabalhando
com
Giroscópio



O que você irá aprender?

SENSORES

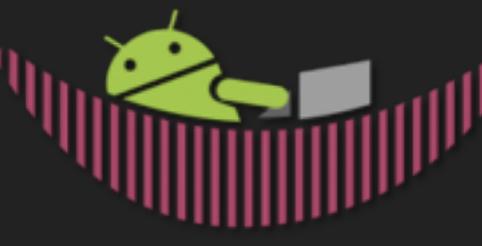
GOOGLE PLAY

Conhecendo
o Guia de
Distribuição

Criando uma
conta de
desenvolvedor

Registrando uma
conta de
comerciante
para apps pagas

Publicando
app no Google
Play Store



O que você irá aprender?

OS INSTRUMENTAÇÃO

CODING DOJO

Duração: 1h

O foco deste **Coding Dojo** será **enfatizar a prática em Android**, onde os **participantes** poderão **resolver o desafio** em questão de forma **totalmente colaborativa** e **praticar testes da app**.

#safaDojo em Fortaleza (2011)



#safaDojo em Castanhal (2012)





Antes de começar...o que você deseja criar?



Pense numa ideia inovadora
para desenvolver uma app
móvel em **Android**. Use a sua
imaginação!

Forme um time de 3 a 4
pessoas no máximo para
formar sua startup!



Introdução à plataforma Android

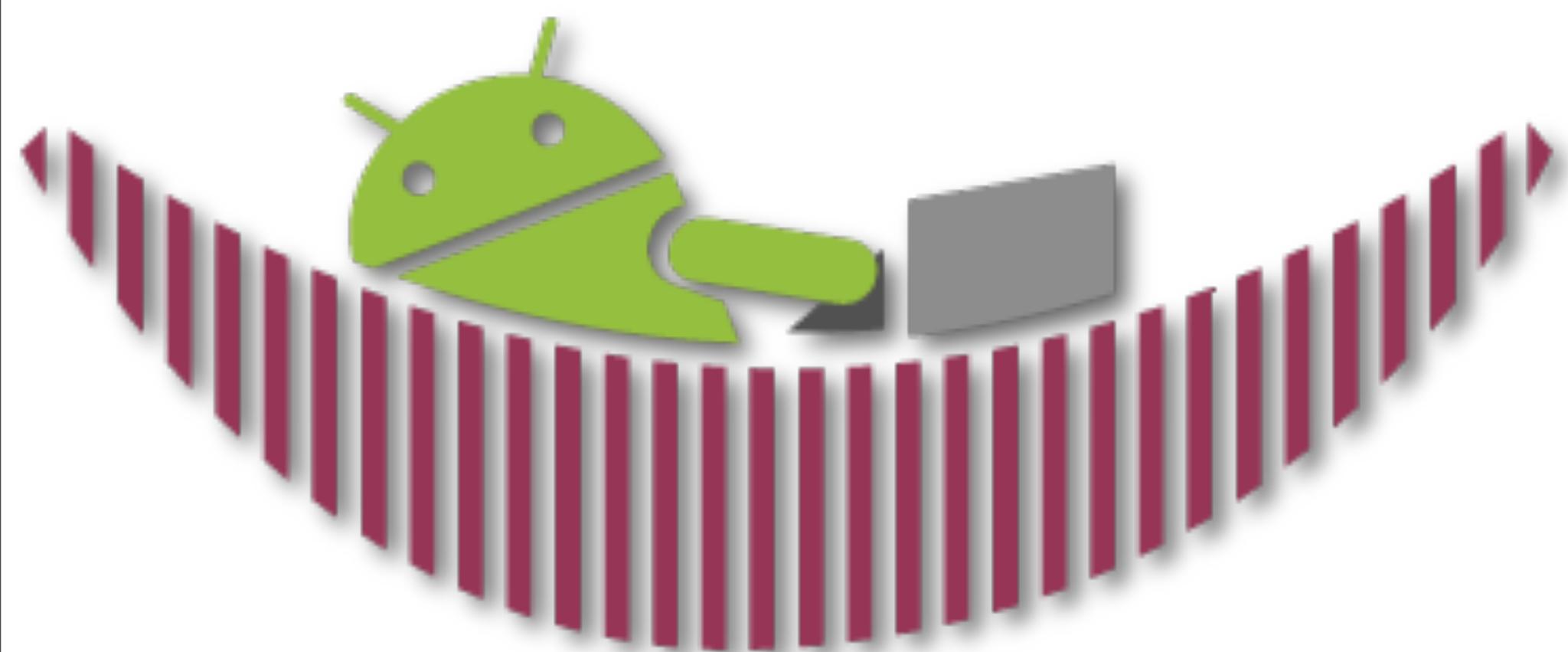


Introdução à plataforma Android





O que é Android?





O que é Android?

**1^a padrão
aberto para
desenvolvimento
móvel.**



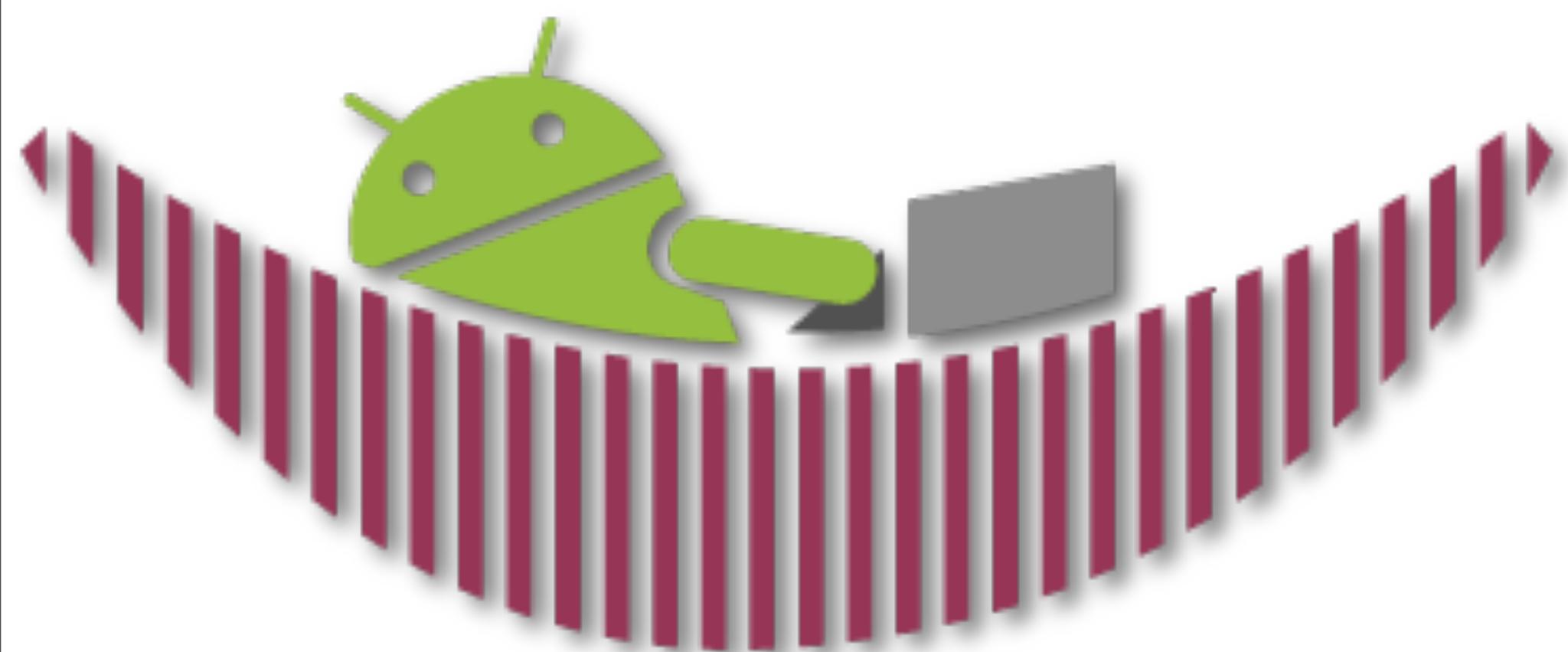


O que é Android?





O que é Android?





O que é Android?

É um
**Sistema
Operacional.**





O que é Android?



Fundadores:

Andy Rich Nick Chris
Rubin Miner Sears White



O que é Android?

Uma
empresa chamada
“Android Inc.”,
fundada em **2003**.

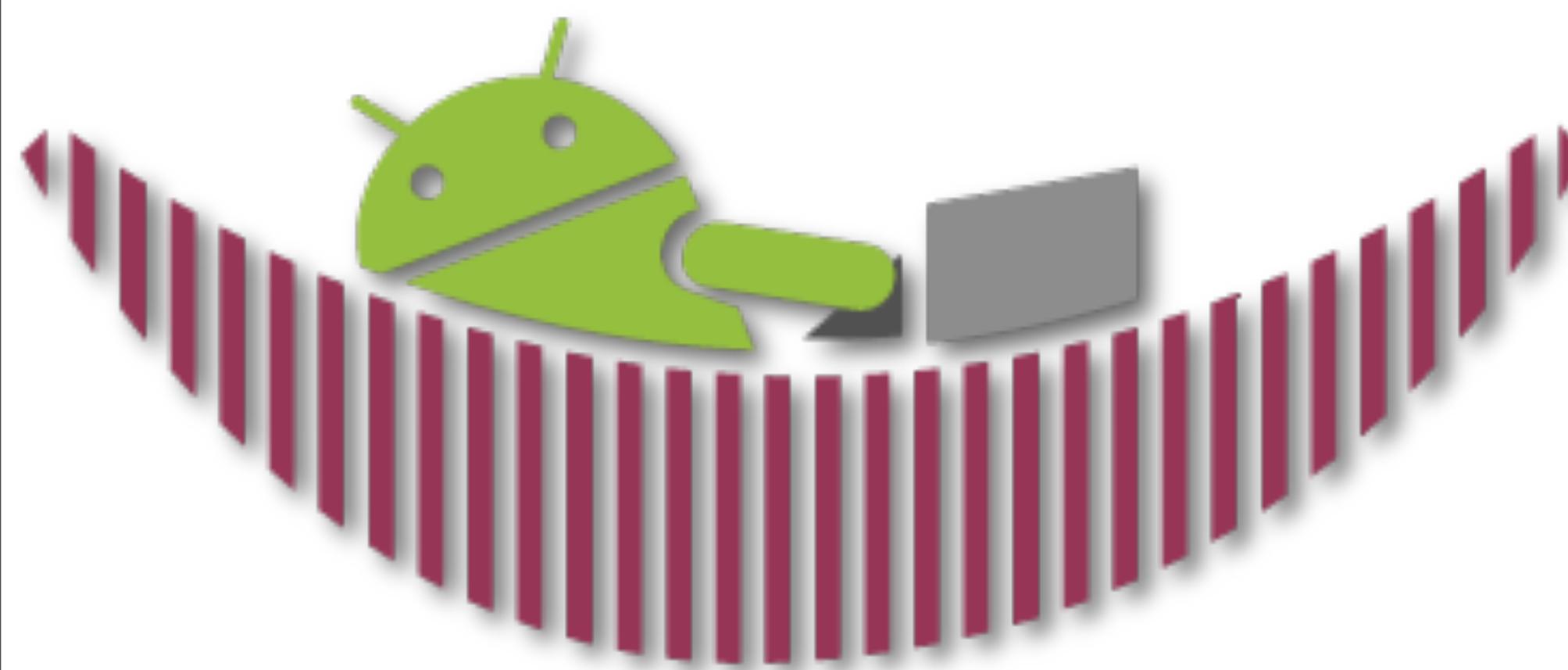


Fundadores:

Andy Rich Nick Chris
Rubin Miner Sears White



O que é Android?





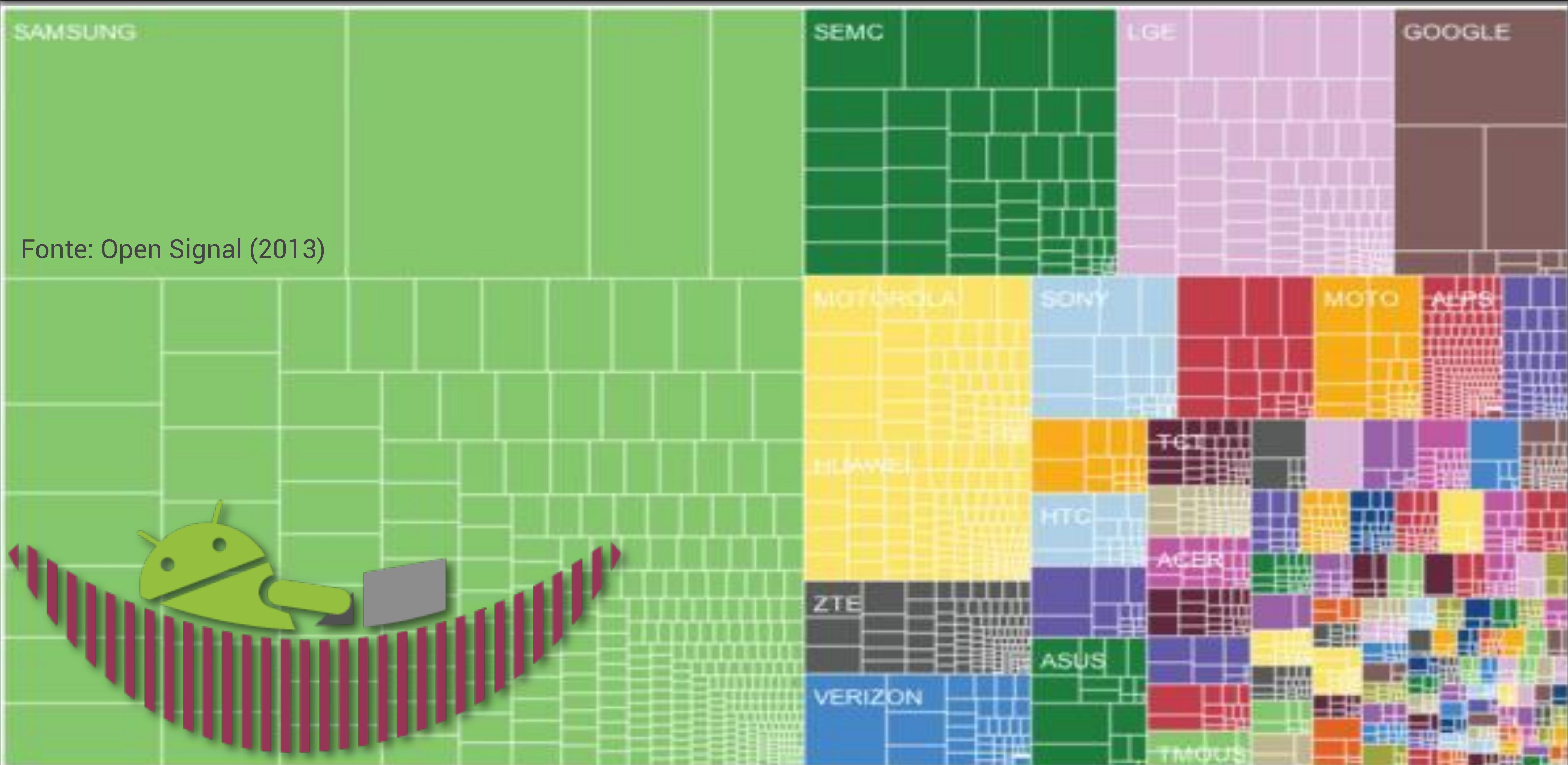
O que é Android?

...que em **2005**
foi comprada pela
Google.





O que é Android?

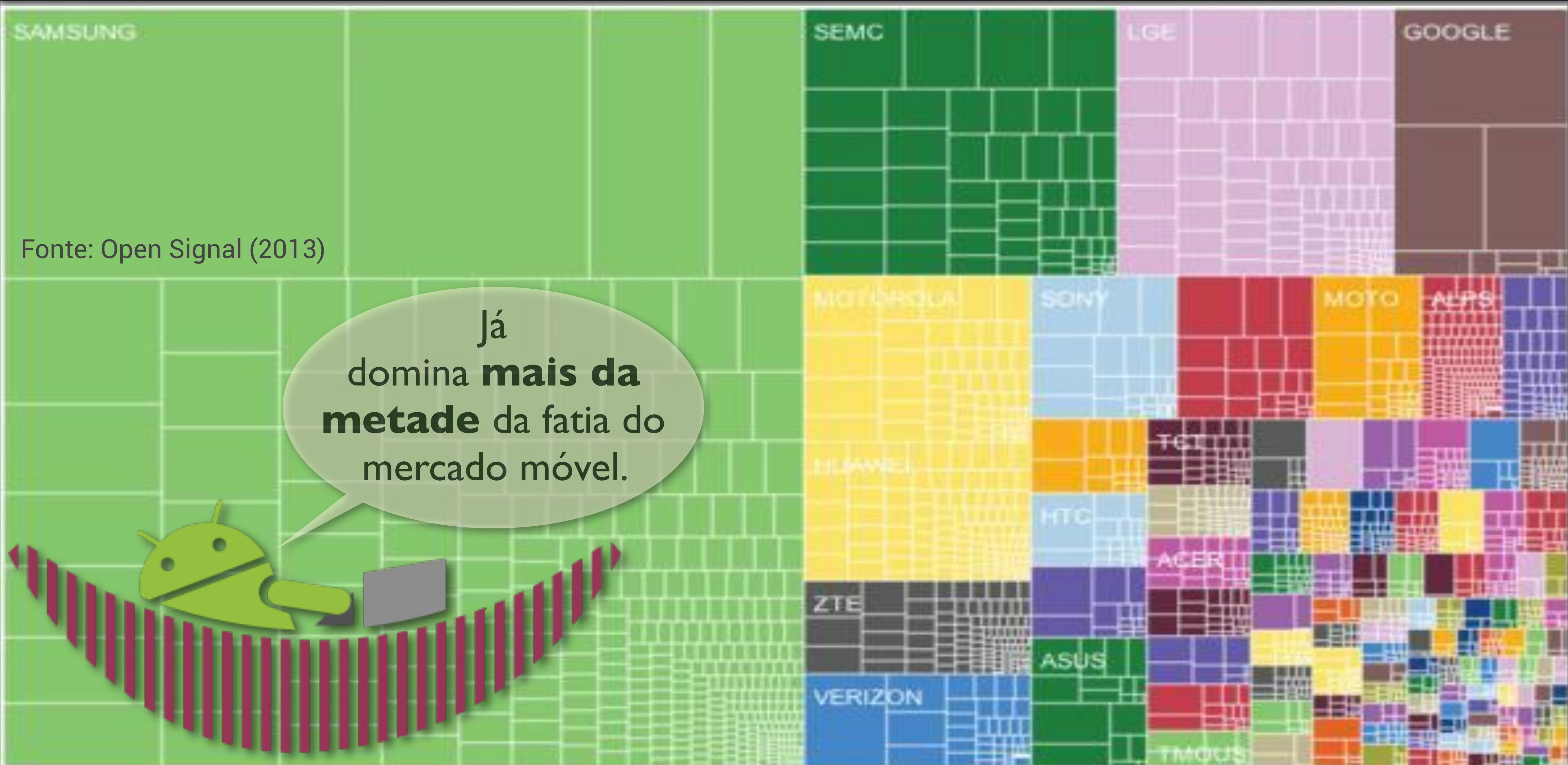




O que é Android?

Fonte: Open Signal (2013)

Já
domina **mais** da
metade da fatia do
mercado móvel.





O que é Android?

Atingiu
900 milhões de
ativações no mundo
todo em **2013**.





O que é Android?

Atingiu
900 milhões de
ativações no mundo
todo em **2013**.





O que é Android?



A Open Handset Alliance

é uma aliança
entre **empresas**
que tem o objetivo
de decidir sobre a
evolução e o futuro
da plataforma
Android.



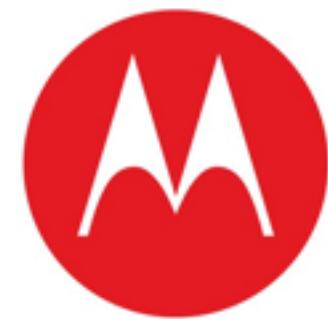
A Open Handset Alliance

Google™

SHARP



NTT docomo



MOTOROLA

TOSHIBA



...T-Mobile



nVIDIA®

ARM

QUALCOMM®



Sprint



é uma aliança
entre empresas
que tem o objetivo
de decidir sobre a
evolução e o futuro
da plataforma
Android.



Por que devo escolher Android?

Para usuários, mais alternativas e
experiência personalizada.



Por que devo escolher Android?

Para usuários, mais alternativas e
experiência personalizada.



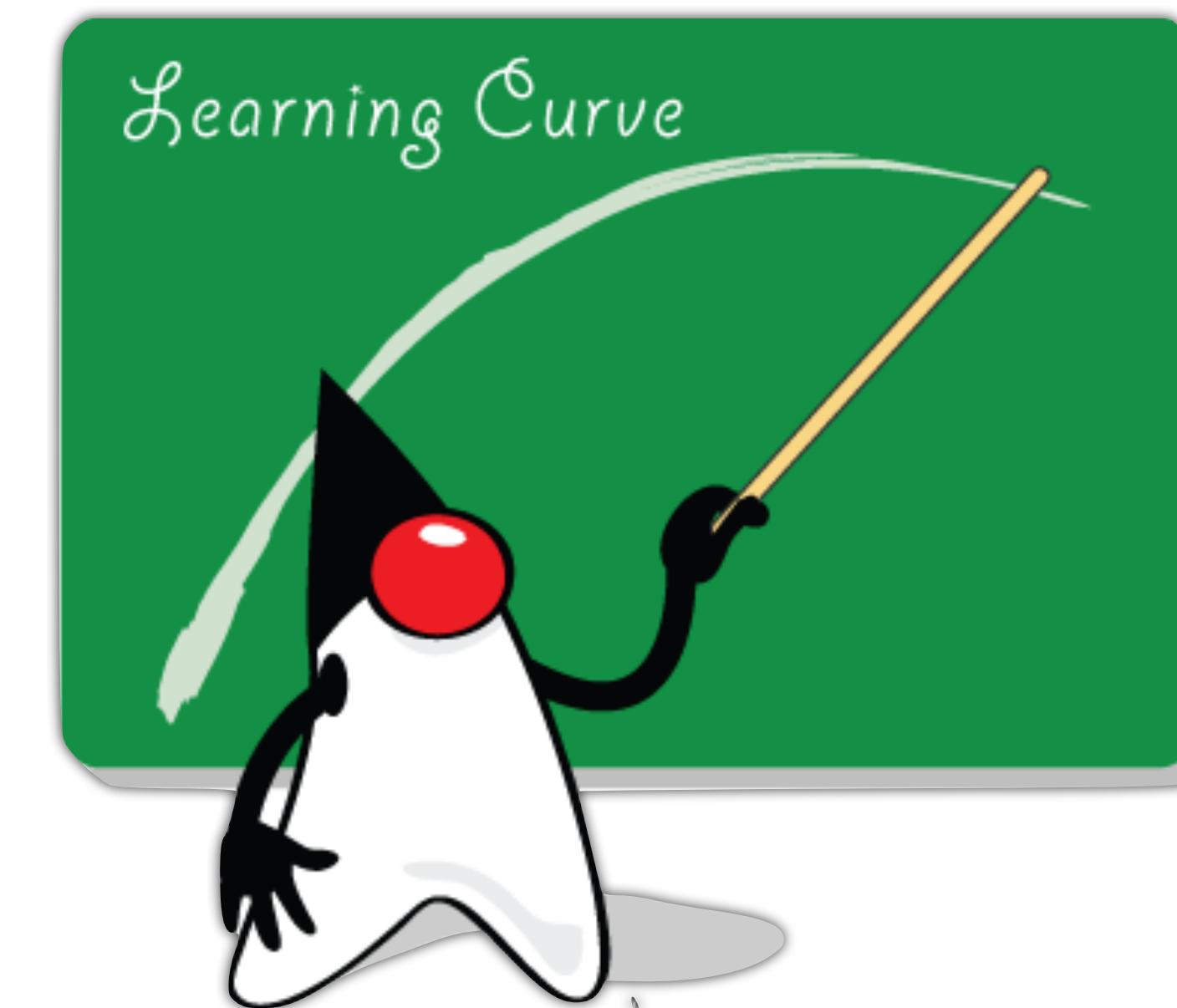


Por que devo escolher Android?

Para nós //desenvolvedores, uma plataforma aberta para poder ser customizada.



Se você conhece Java, a curva de aprendizado para Android será mínima.



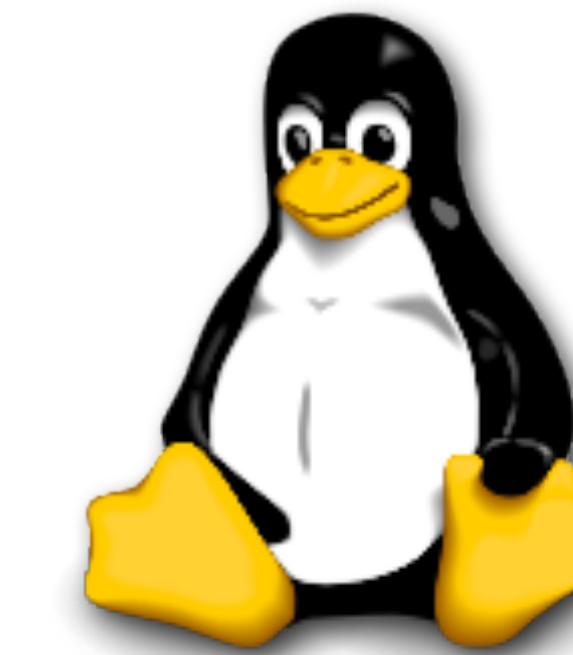
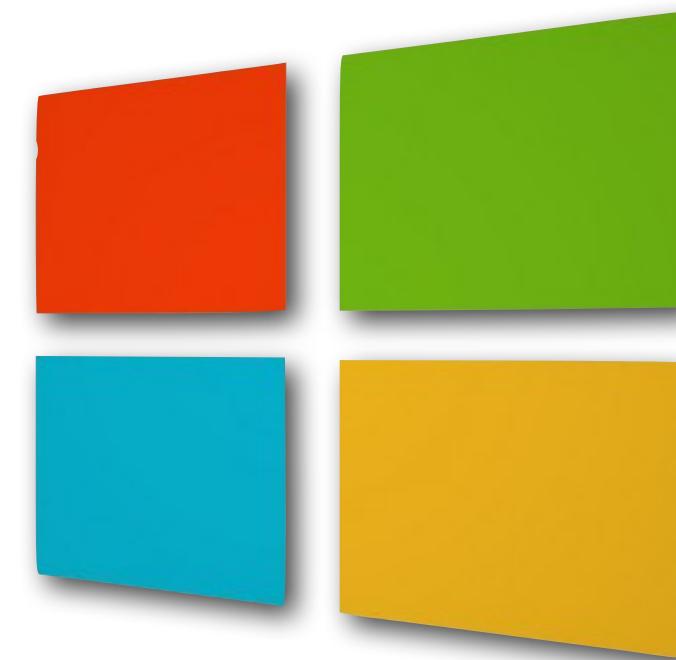


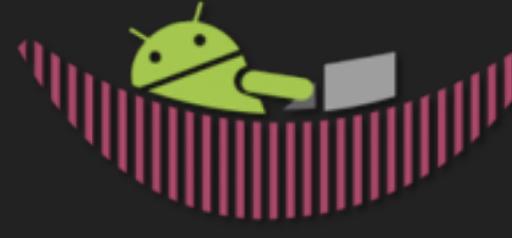
Por que devo escolher Android?

Facilidade na integração e comunicação entre aplicativos.



Você pode desenvolver em qualquer Sistema Operacional.





Doces e mais doces: a evolução da plataforma



Cupcake



Donut



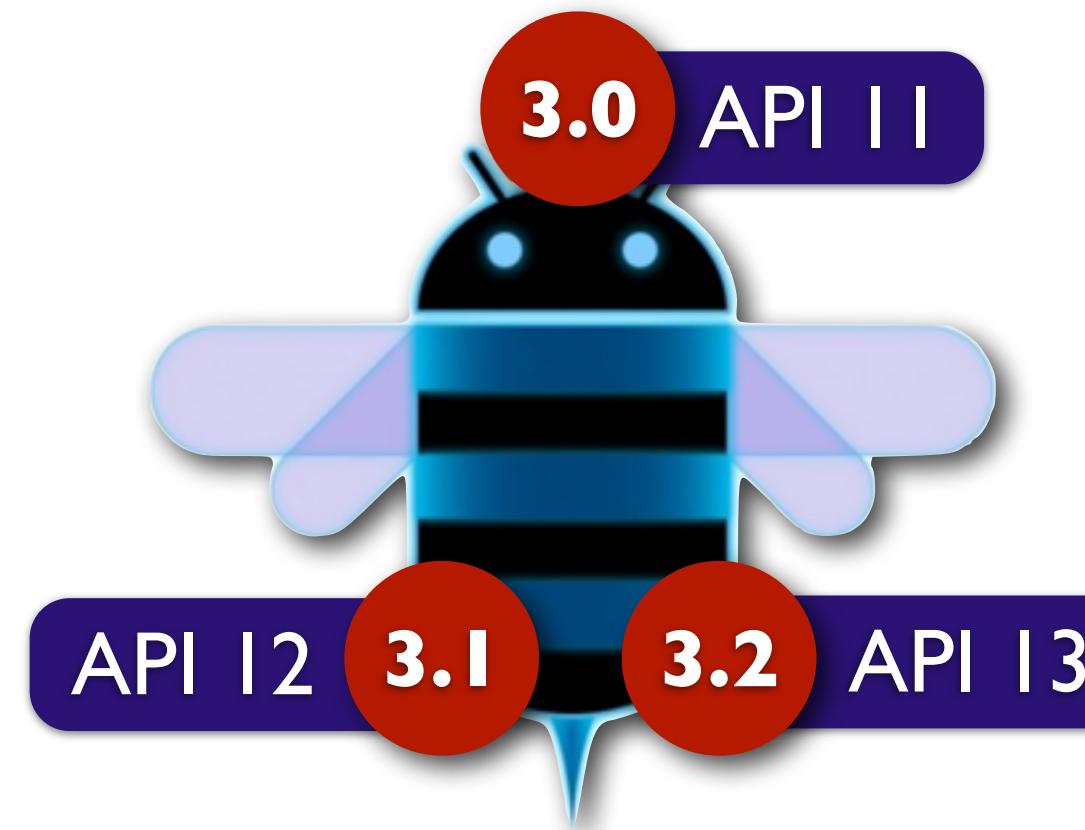
Eclair



FroYo



Gingerbread



Honeycomb



Ice Cream
Sandwich



Jelly Bean



Mitos, Fatos e Curiosidades



Mito

**“Android é uma
plataforma lenta
e difícil de
utilizar!”**

Fato

Na verdade, alguns fabricantes customizavam o sdk com uma nova camada de interface gráfica que acabava influenciando na performance do aparelho e na usabilidade.



Mitos, Fatos e Curiosidades



Mito

“Android é somente para usuários com níveis técnicos avançados.”

Fato

A plataforma vem evoluindo e melhorando a cada atualização, assim como a usabilidade e experiência com o usuário que tem sido o foco principal da plataforma.



Mitos, Fatos e Curiosidades



Mito

“Android não é uma plataforma tão aberta assim pois a Google que disponibiliza as atualizações.”

Fato

Apesar da Google e a Open Handset Alliance ditarem o futuro e evolução da plataforma, a Google disponibiliza o SDK para que o mesmo possa ser baixado e customizado pelos desenvolvedores.



Mitos, Fatos e Curiosidades



Mito

*“Android é uma
cópia fiel das
funcionalidades
do iOS.”*

Fato

Poucos sabem que o projeto Android foi concebido em 2003, pela empresa “Android Inc”, que em 2005 foi adquirida pela Google, muito antes do iOS surgir no mercado (em 2007).



Mitos, Fatos e Curiosidades



Mito

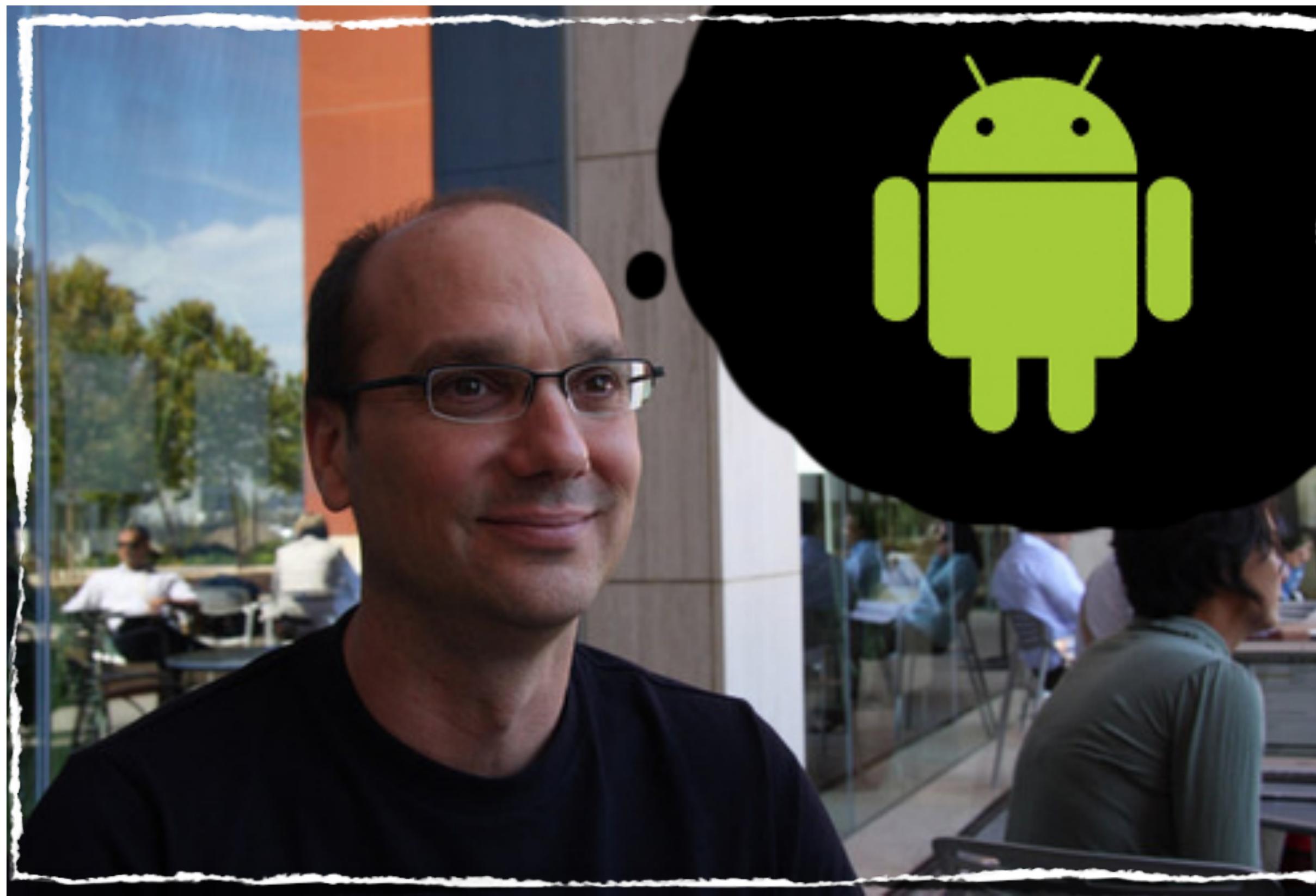
“Android não é uma plataforma tão segura assim.”

Fato

Android tem como base um kernel Linux, que permite que cada app seja executada em processos isolados, evitando assim que uma aplicação burle o acesso aos dados de outra app.



Mitos, Fatos e Curiosidades



Curiosidade

“Rumores informam que o termo “Android” foi escolhido pelo seu fundador, Andrew Rubin, como trocadilho ao seu nome.”



Mitos, Fatos e Curiosidades



Curiosidade

As versões da plataforma tem *nomes de sobremesas*, pois segundo os engenheiros da plataforma, você deve gostar tanto de Android igual como gosta de apreciá-las.



Mitos, Fatos e Curiosidades

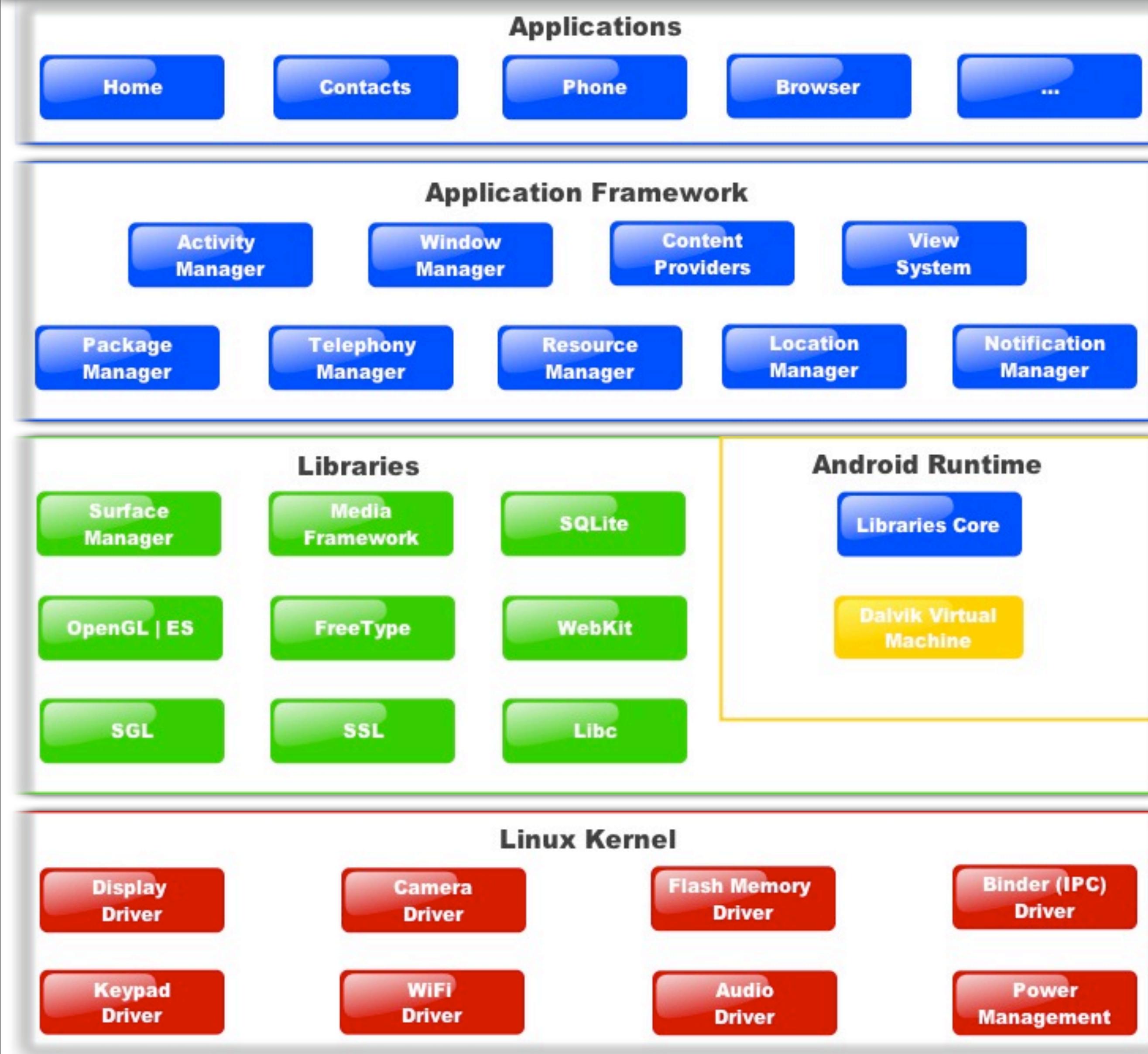
Curiosidade

**O desenvolvimento em
Android foi projetado
para ser tão
transparente quanto se
estivesse criando
aplicativos desktop ou
web
(ex: descriptor de app,
interface gráfica em xml,
binding de xml para
Java, etc).**

```
1<menu  
2  xmlns:android="http://schemas.android.com/apk/res/android">  
3  <item  
4      android:id="@+id/menuFavorites"  
5      android:title="@string/menuFavorites"  
6      android:icon="@android:drawable/ic_menu_view" />  
7  <item  
8      android:id="@+id/menuSettings"  
9      android:title="@string/menuSettings"  
10     android:icon="@android:drawable/ic_menu_preferences" />  
11 <item  
12     android:id="@+id/menuExit"  
13     android:title="@string/menuExit"  
14     android:icon="@android:drawable/ic_menu_close_clear_cancel" />  
15 <item  
16     android:icon="@android:drawable/ic_menu_close_clear_cancel"  
17     android:checkable="true"  
18     android:id="@+id/item01"></item>  
19</menu>
```



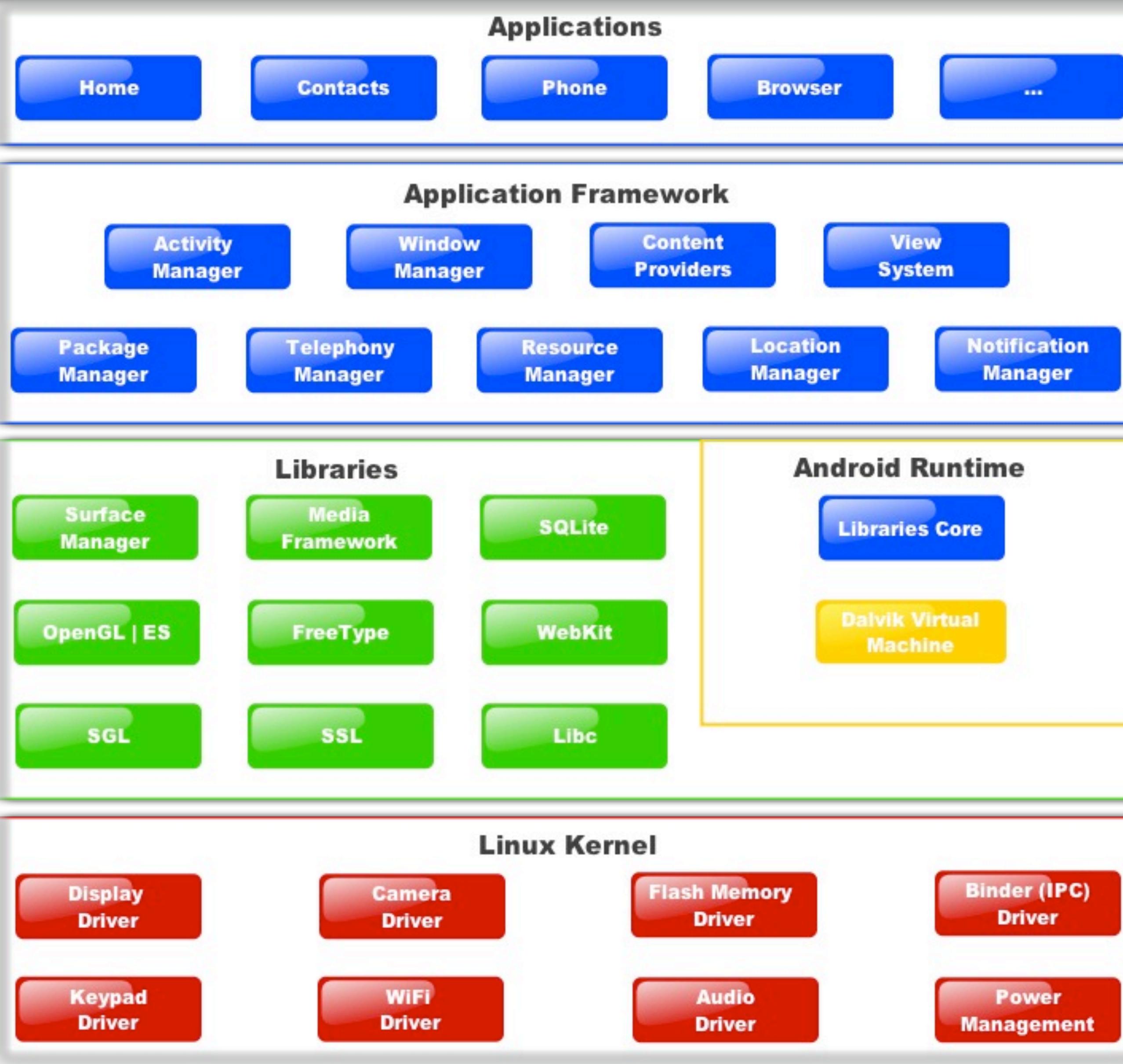
O Framework Android



**Bibliotecas que fazem
parte do Kernel Linux**



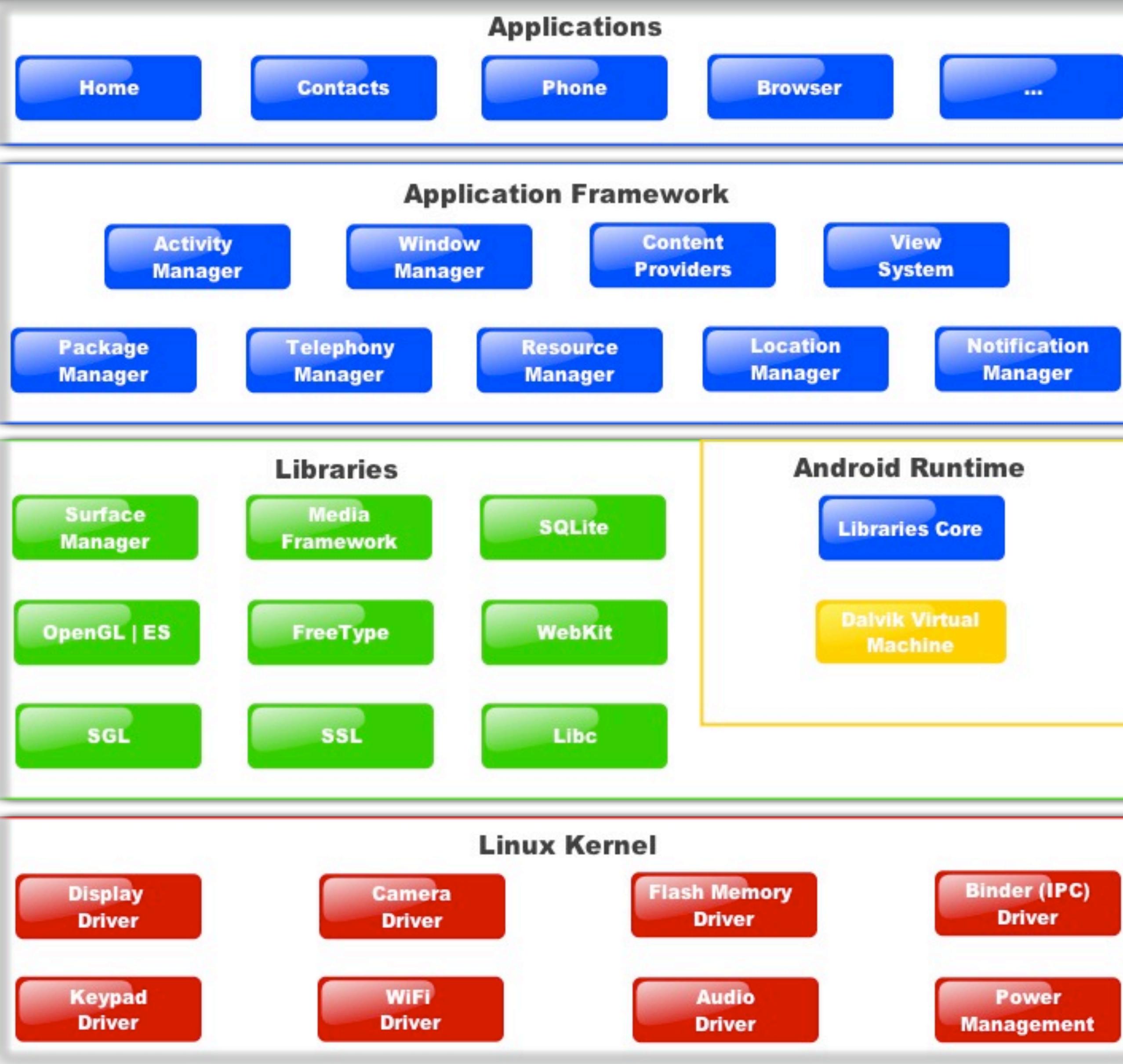
O Framework Android



Implementação em C/C++ das bibliotecas essenciais da plataforma e para a máquina virtual Dalvik



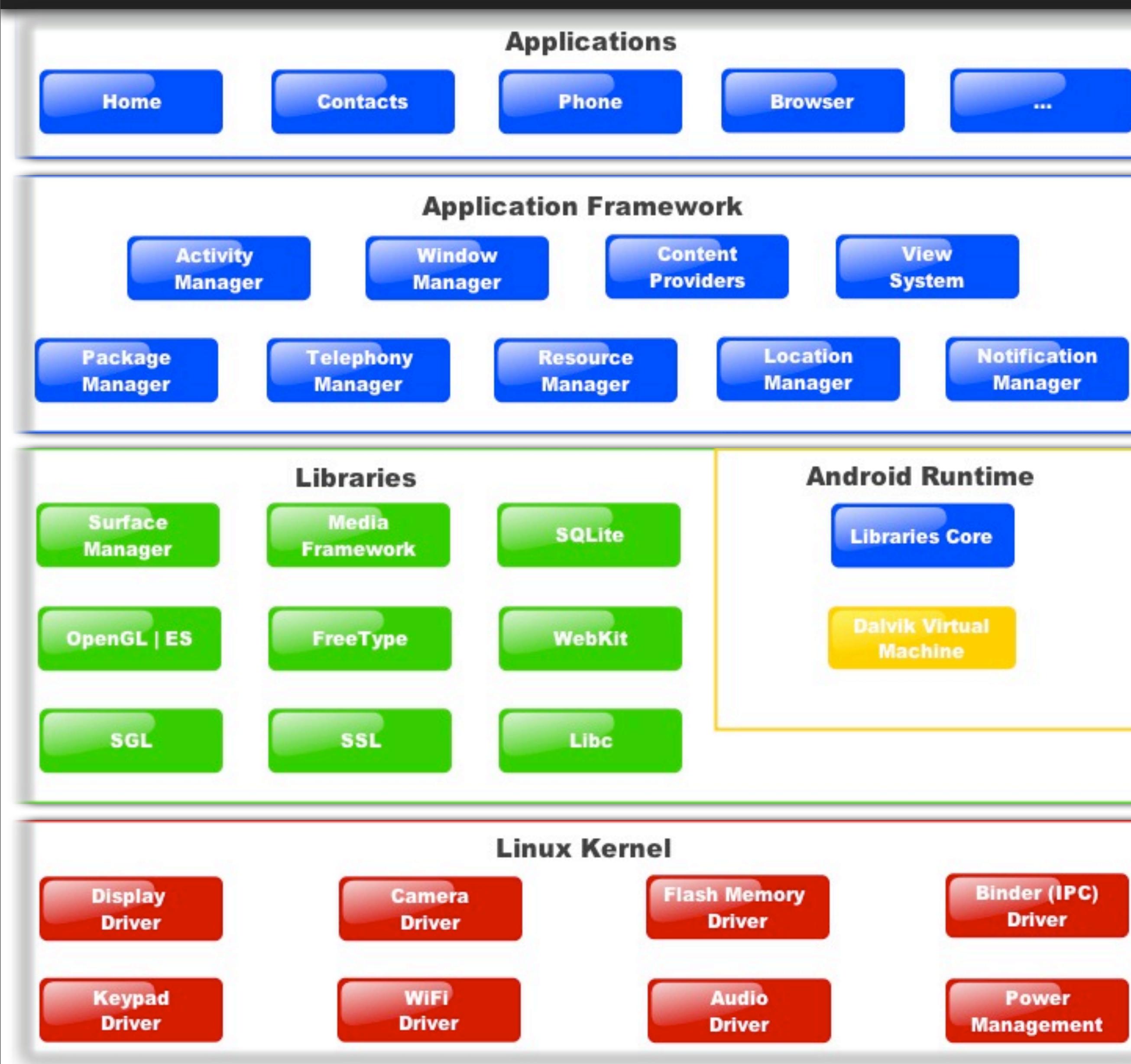
O Framework Android



Serviços que a plataforma disponibiliza,
por meio de gerenciadores para não ser
necessário “reinventar a roda”



O Framework Android



Aplicativos desenvolvidos em Android



A máquina virtual Dalvik

Para que uma aplicação em Android pudesse ser executada, eram necessárias as seguintes exigências:

- Toda aplicação Android deve rodar em um serviço separado.
- Cada aplicação devia possuir sua própria JVM.
- Uma JVM precisaria ser executada otimizadamente em dispositivos móveis.
- Uma aplicação não deveria alterar muito a performance do aparelho, quando estivesse sendo executada



A máquina virtual Dalvik

Sendo assim, uma JVM Hotspot não poderia ser utilizada pois não atende aos requisitos anteriores.

Surgiu a *Dalvik*,
uma máquina
virtual otimizada,
rápida que seria
incorporada no
projeto **Android**.



Dalvik foi batizada por Bornstein em homenagem à vila de pescadores de Dalvík em Eyjafjörður, Islândia, onde alguns de seus antepassados viveram.

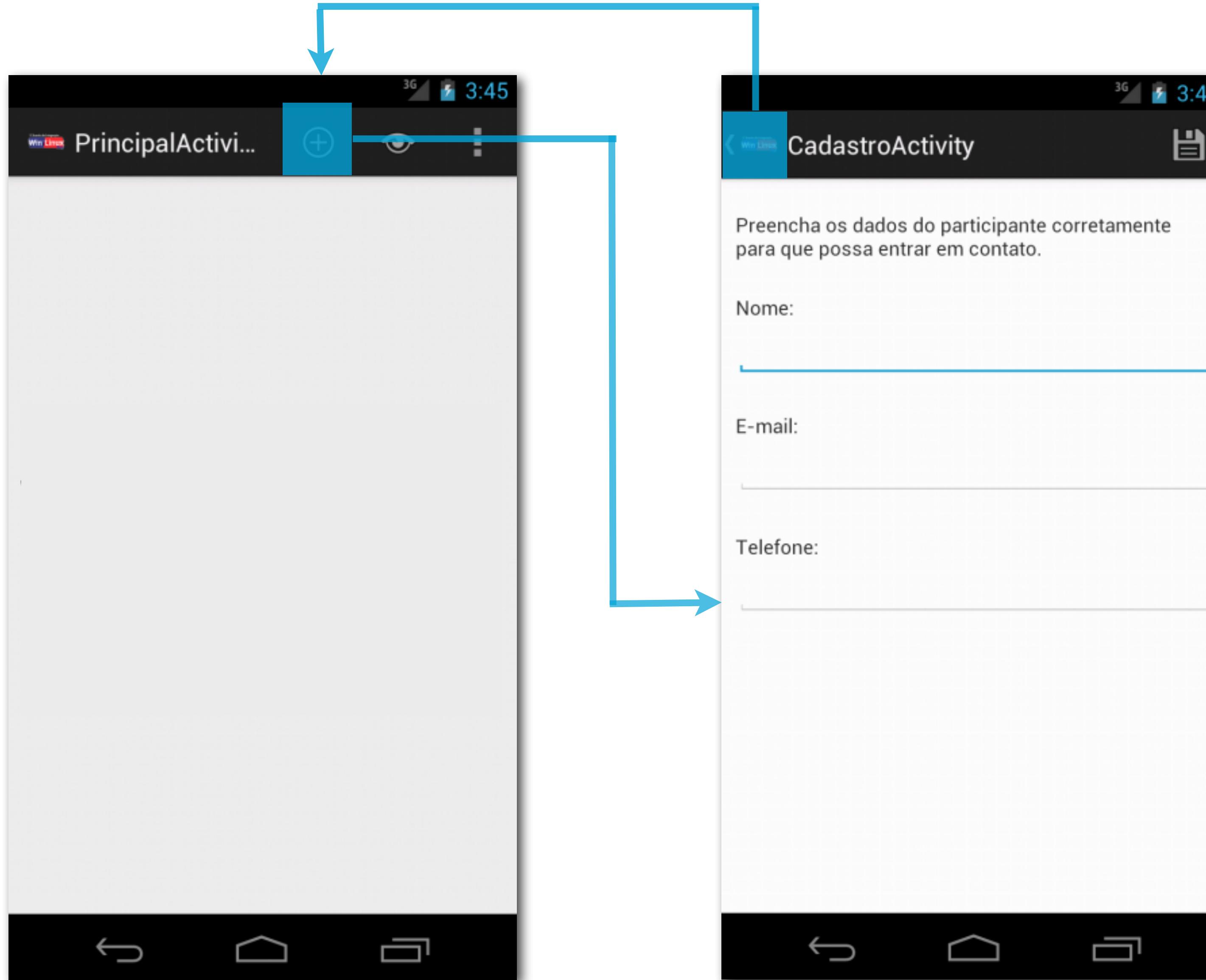


Aspectos essenciais que devemos saber sobre Android





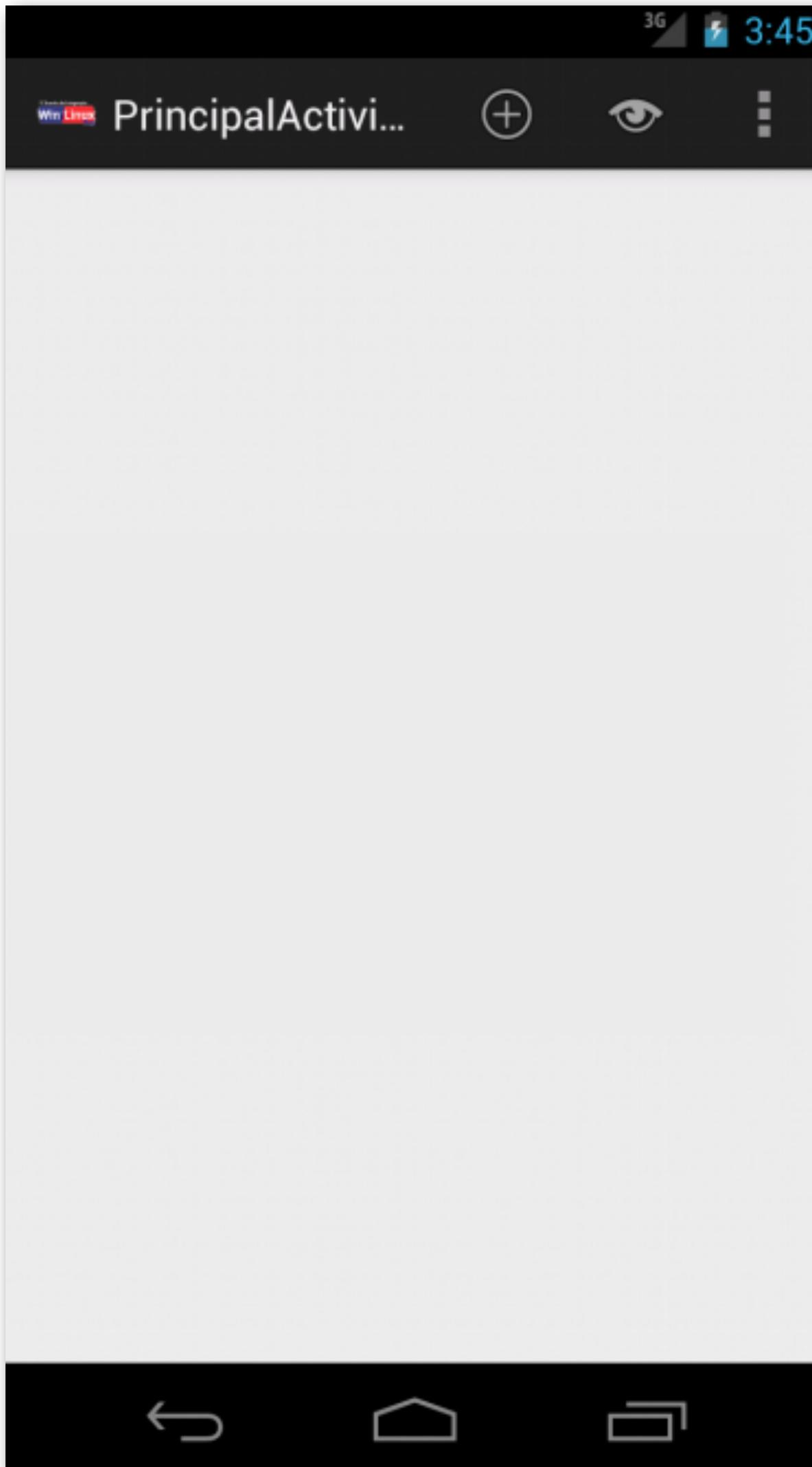
Fundamentos essenciais: Intents



Componente responsável por realizar a comunicação entre os componentes Android (Activity, BroadcastReceiver, Service)



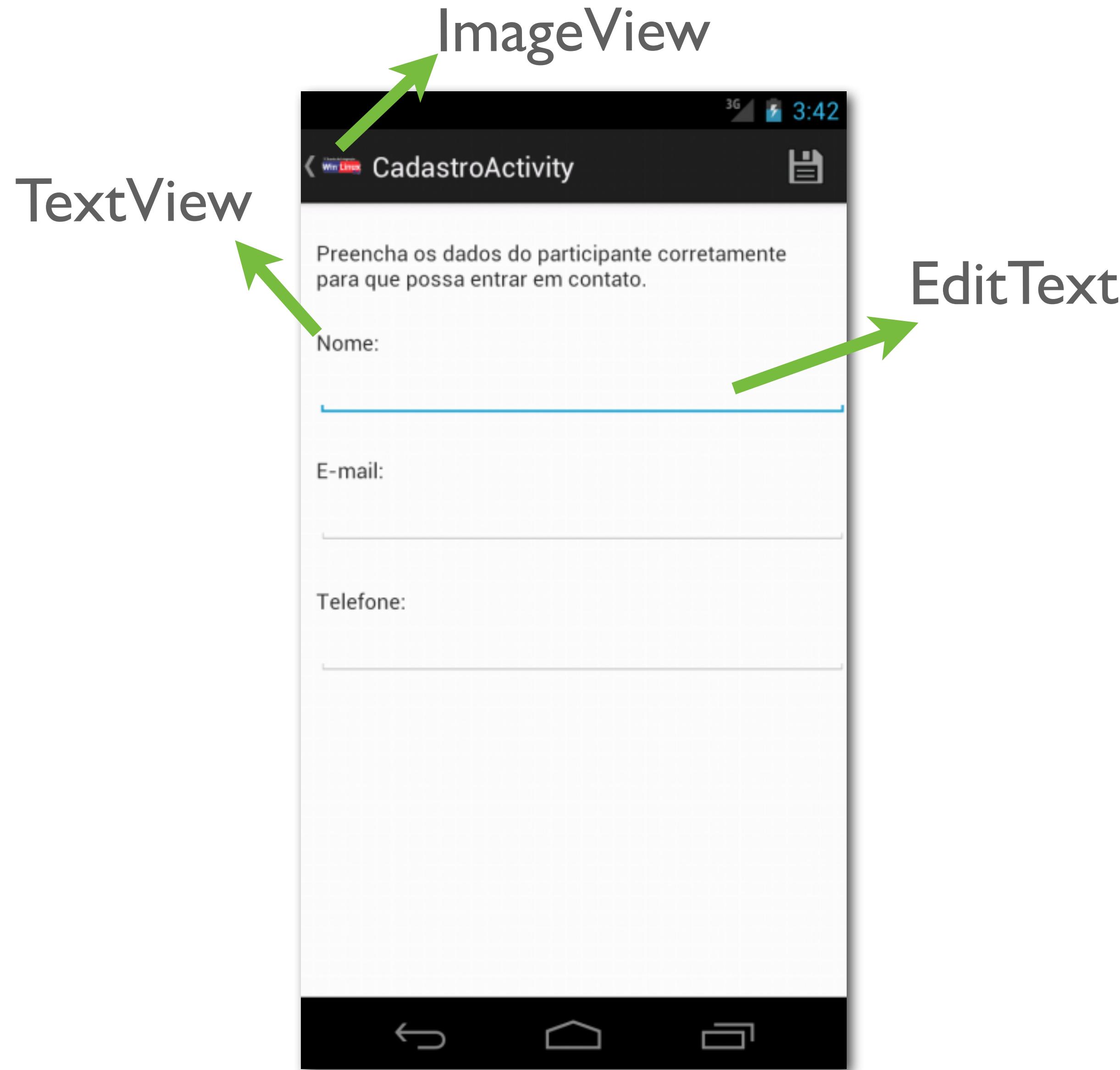
Fundamentos essenciais: Activity



**Componente, que
possui seu próprio
ciclo de vida,
utilizado para
visualizar
interfaces gráficas.**



Fundamentos essenciais: View



Representa
qualquer
componente
gráfico de tela.

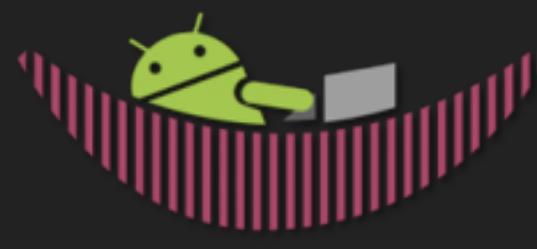


Fundamentos essenciais: AndroidManifest.xml

```
?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.androidnarede.winlinuxdayapp"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="16" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="br.com.androidnarede.winlinuxdayapp.SplashActivity"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

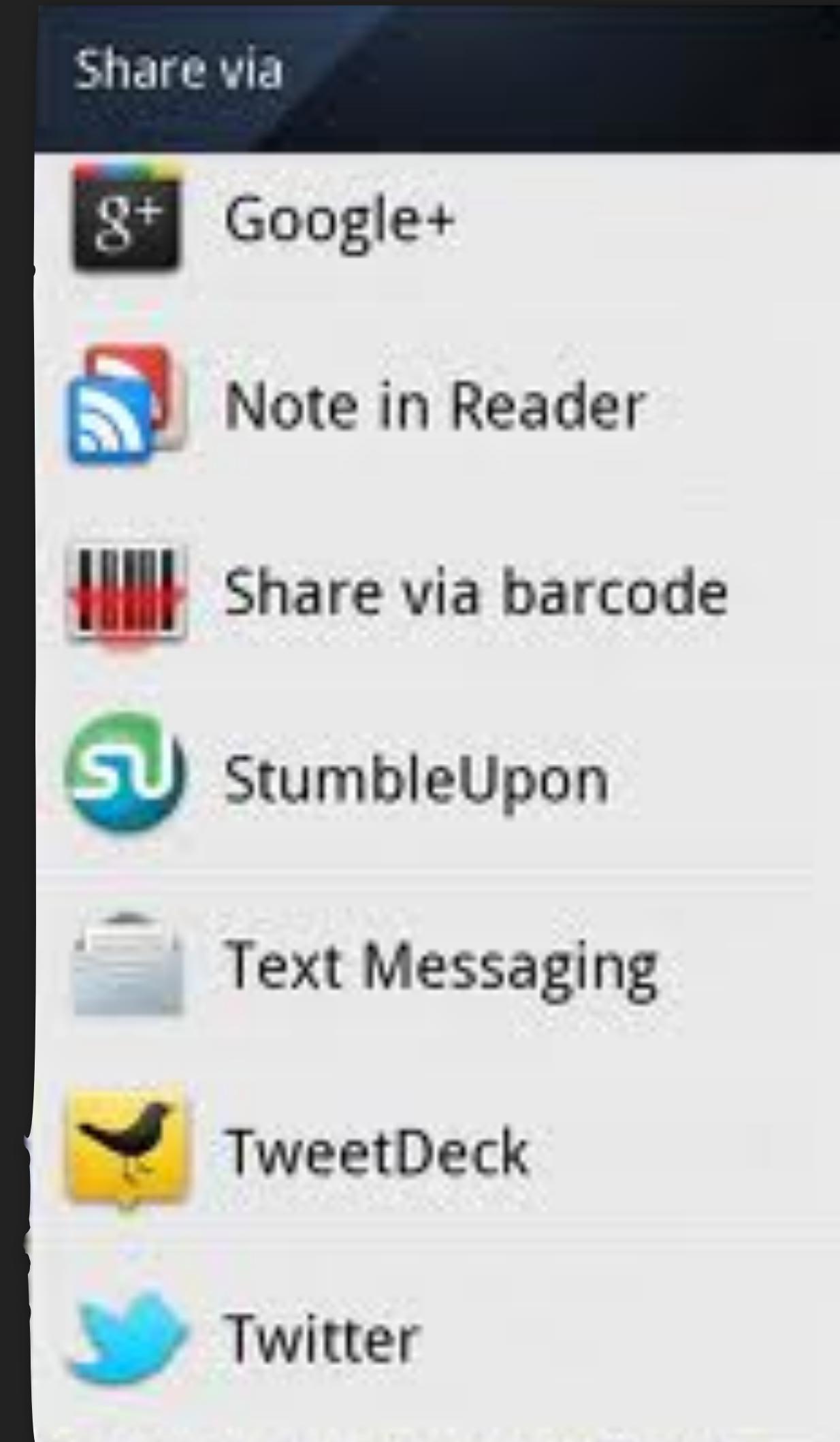
        <!-- mais declarações de Activities -->
    </application>
</manifest>
```

É o descriptor
de uma
aplicação
android.



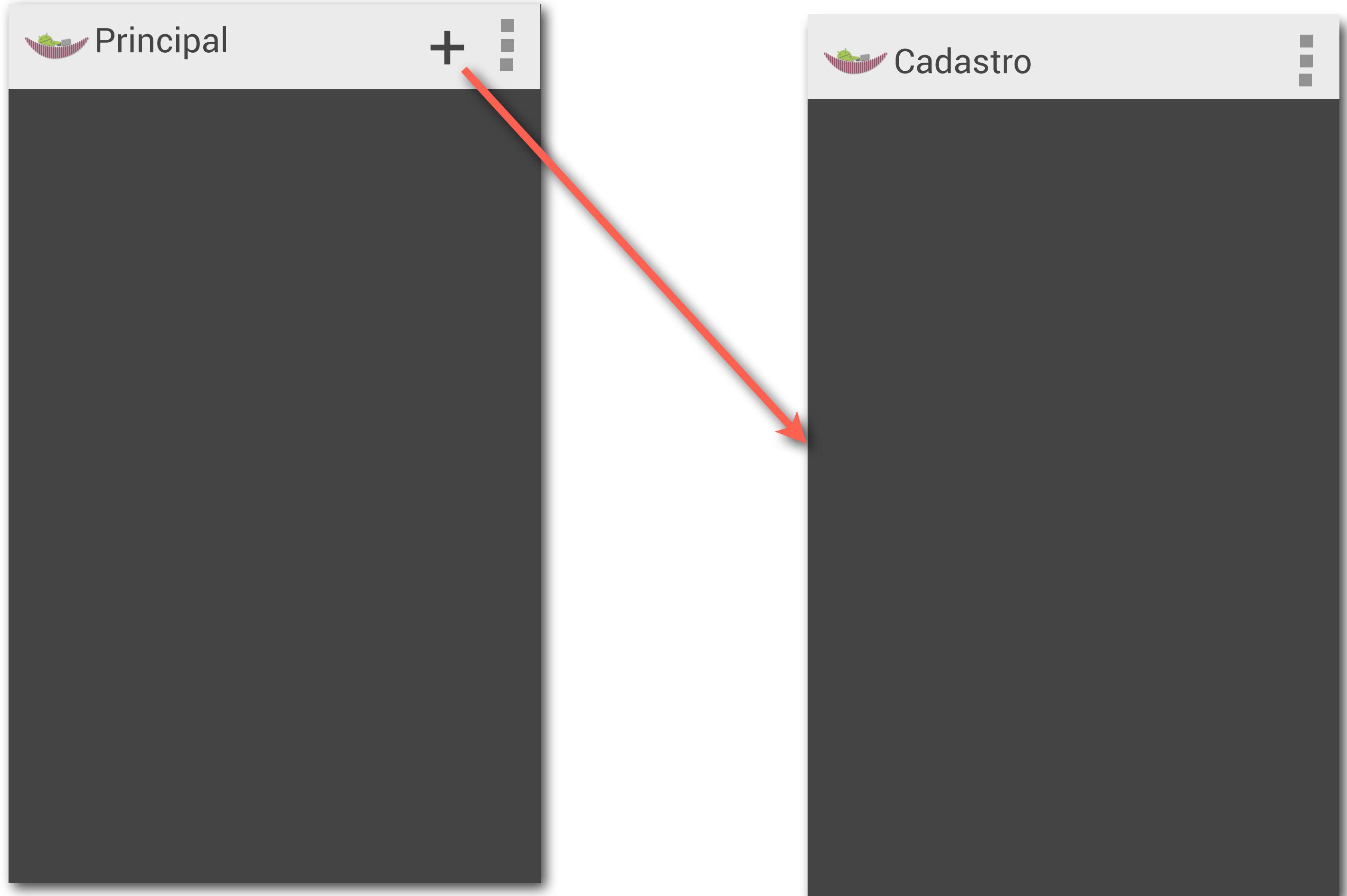
Um pouco mais sobre Intents

Entendendo como ocorre a comunicação entre componentes

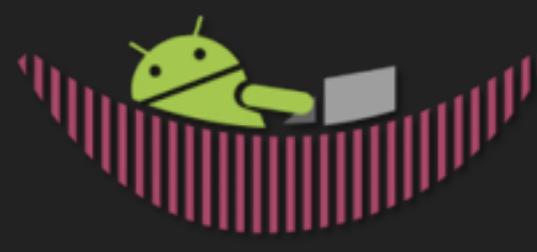




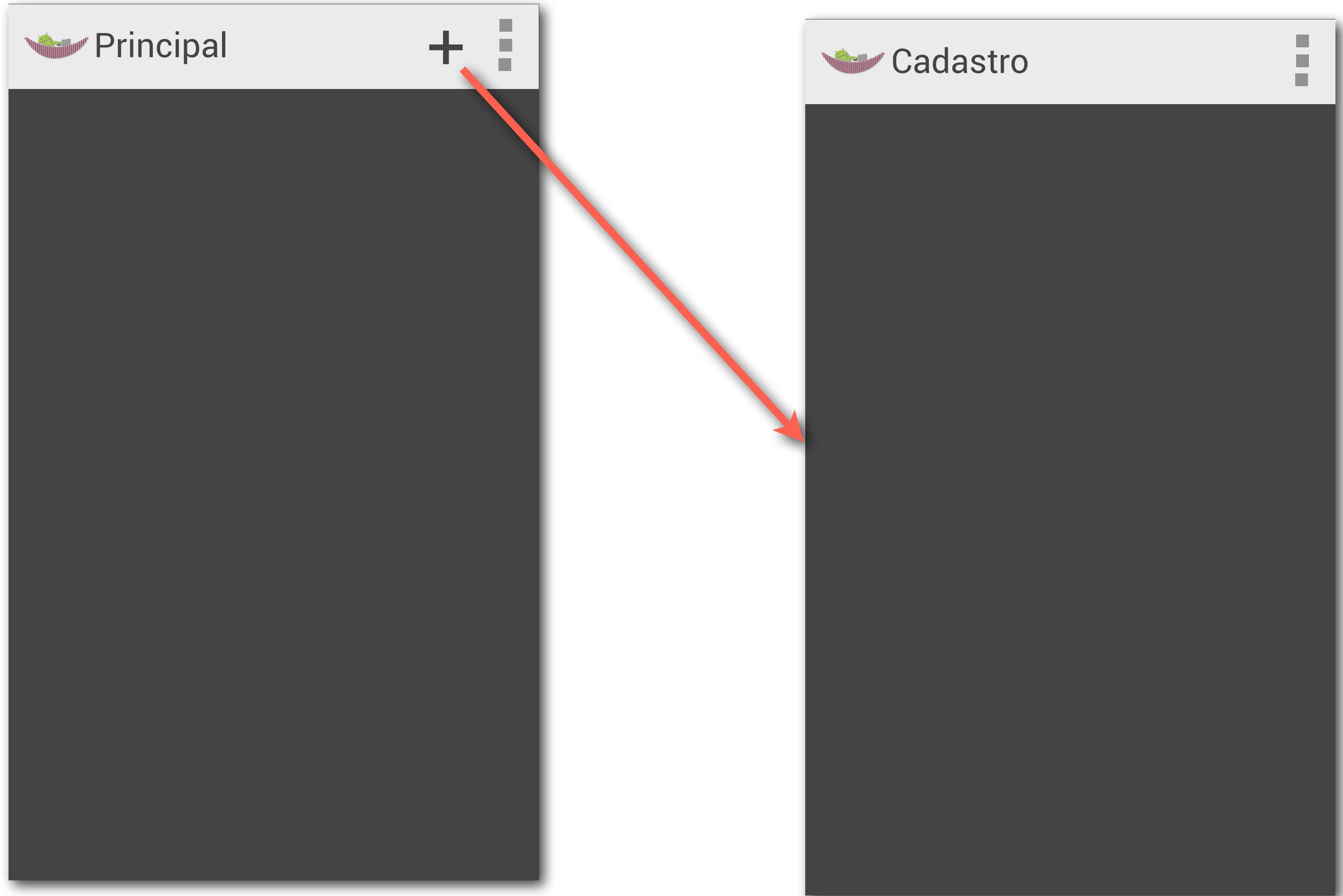
Relembrando...



**Uma Intent é
responsável por
comunicar
componentes
em Android**



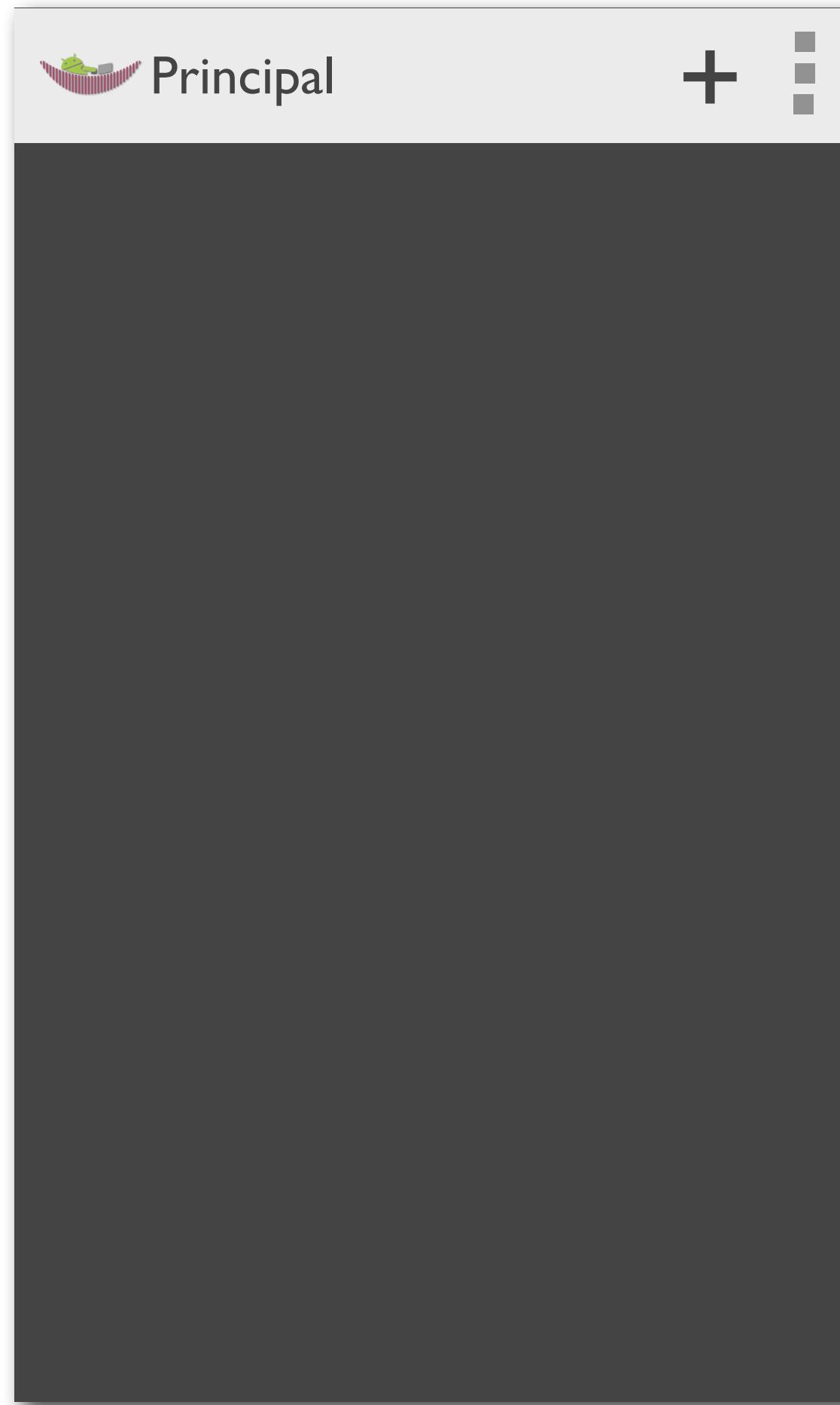
Relembrando...



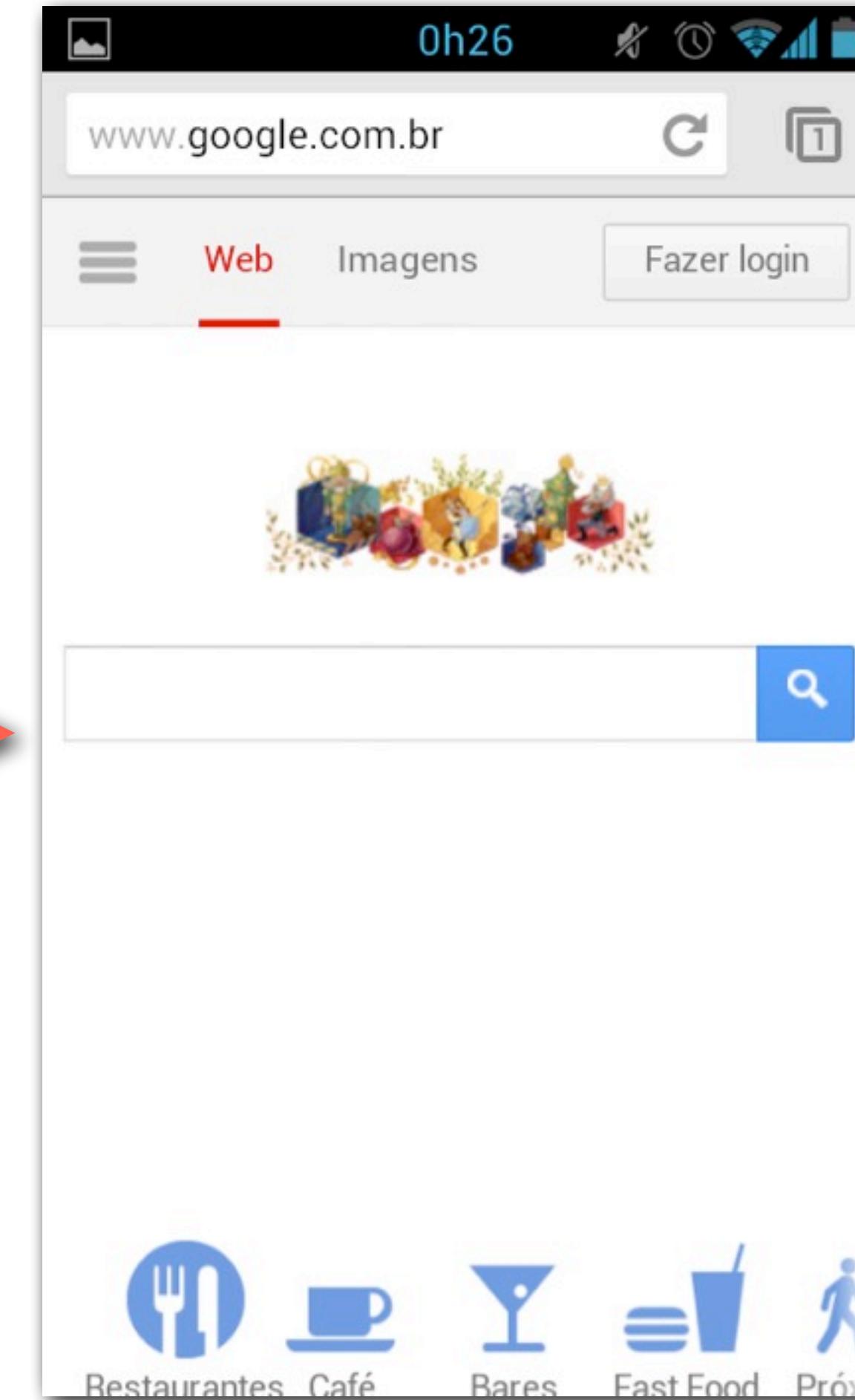
Uma Intent pode ser definida como **abstração** para realizar alguma ação.



Um pouco mais sobre Intents



AÇÃO: VISUALIZAR
“<http://google.com.br>”



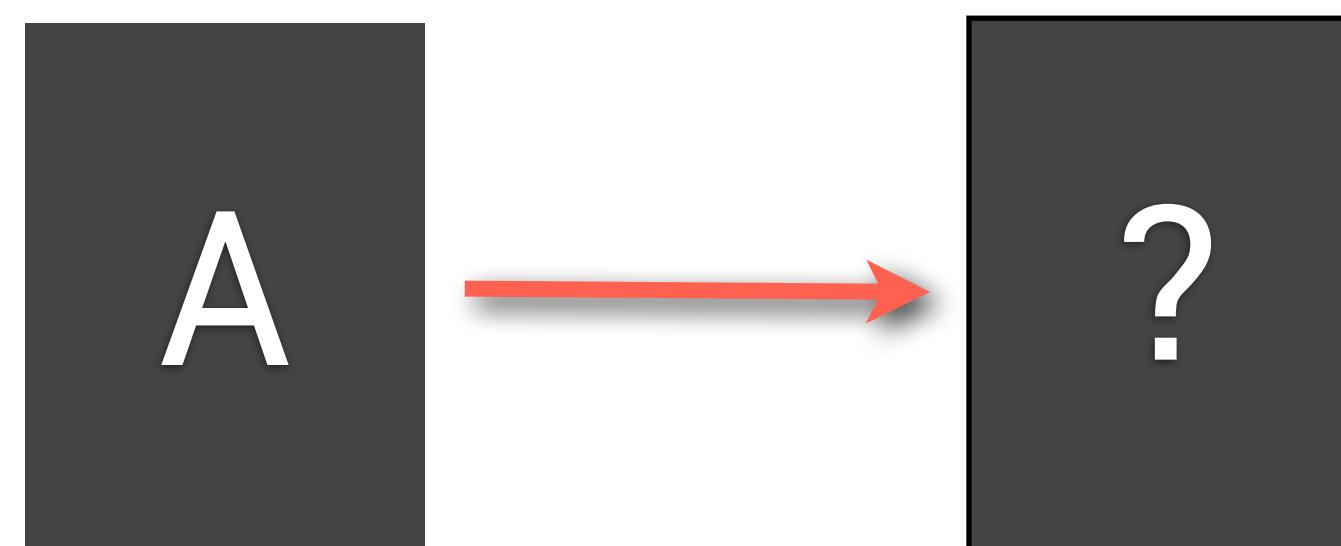
**Como essa ação
será realizada,
dependerá do
tipo do conteúdo
a ser enviado.**



Comunicando componentes diferentes da plataforma

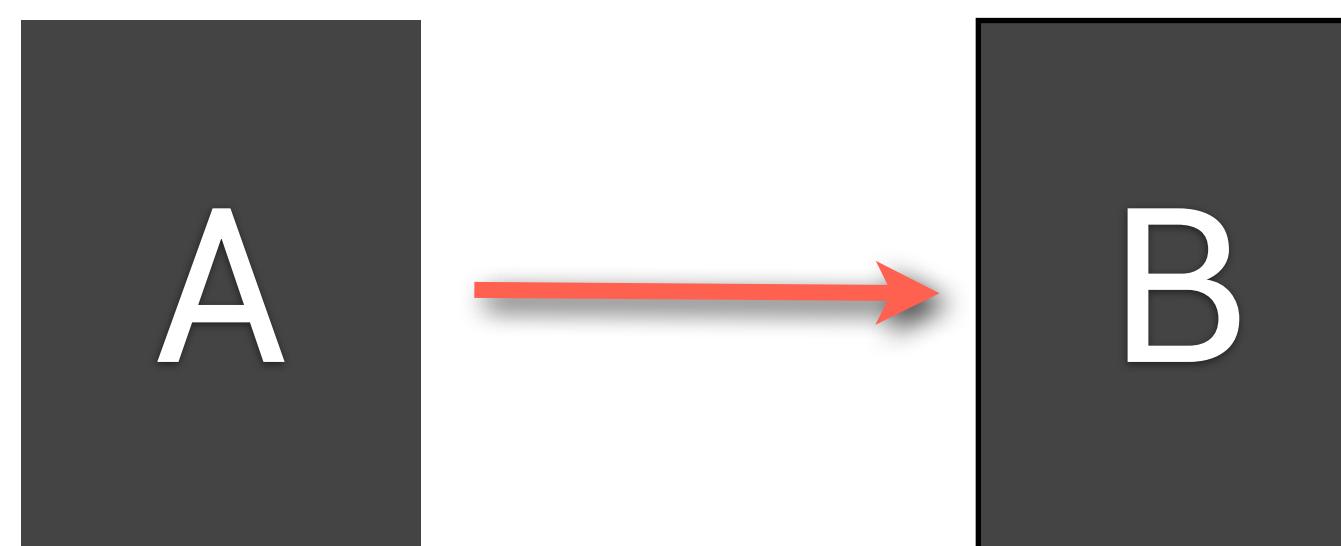
Existem 2 tipos de Intents:

Intent Implícitas



São intents que possuem informação **somente** da **ação** a ser realizada. A escolha do componente que irá realizá-la, fica por conta da **plataforma**.
(ex: Visualizar uma página da Web)

Intent Explícitas



São intents que possuem informação tanto da **origem** quanto do **destino** de determinada **ação**. O componente de destino é explicitamente durante a comunicação (ex: ActivityA inicia Activity B)



Discando para um contato

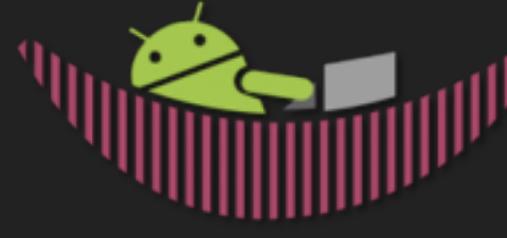
```
Uri uri = Uri.parse("tel:559191322334");
Intent intent = new Intent(Intent.ACTION_DIAL, uri);
startActivity(intent);
```



Ligando para um contato

```
Uri uri = Uri.parse("tel:559191322334");
Intent intent = new Intent(Intent.ACTION_CALL, uri);
startActivity(intent);
```

Deve-se adicionar a tag de permissão <uses-permission android:name="android.permission.CALL_PHONE" /> no AndroidManifest.xml



Enviando mensagens de texto

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_TEXT, "O seu amigo oculto já foi escolhido!");
intent.setType("text/plain");
startActivity(Intent.createChooser(intent, "Enviar usando"));
```



Enviando um e-mail

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, new String[]{ "email_do_amigo@gmail.com" });
intent.putExtra(Intent.EXTRA_SUBJECT, "Já temos o seu amigo oculto!");
intent.putExtra(Intent.EXTRA_TEXT, "O seu amigo oculto já foi escolhido!");
intent.setType("text/html");
startActivity(Intent.createChooser(intent, "Enviar usando"));
```



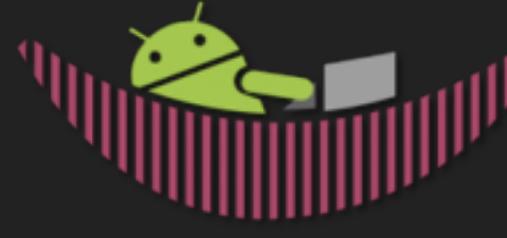
Visualizando páginas no navegador

```
Uri uri = Uri.parse("http://www.google.com");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```



Visualizando os seus contatos

```
Uri uri = ContactsContract.Contacts.CONTENT_URI;  
Intent intent = new Intent(Intent.ACTION_VIEW, uri);  
startActivity(intent);
```



Visualizando uma localidade no mapa

```
String local = "Belém";
Uri uri = Uri.parse("geo:0,0?q="+local);
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```



Pesquisando direto no Google

```
String termoDePesquisa = "android";
Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
intent.putExtra(SearchManager.QUERY, termoDePesquisa);
startActivity(intent);
```



Mudando de tela (Activity)

```
Intent intent = new Intent(this, CadastroActivity.class);  
startActivity(intent);
```



Iniciando um Serviço

```
public class AlgumServiço extends Service {  
    // onStartCommand()  
}
```

```
Intent intent = new Intent(this, AlgumServiço.class);  
startService(intent);
```



Enviando mensagens em Broadcast

```
public class UmBroadcastReceiver extends BroadcastReceiver {  
    // onReceive()  
}
```

```
Intent intent = new Intent(this, UmBroadcastReceiver.class);  
sendBroadcast(intent);
```

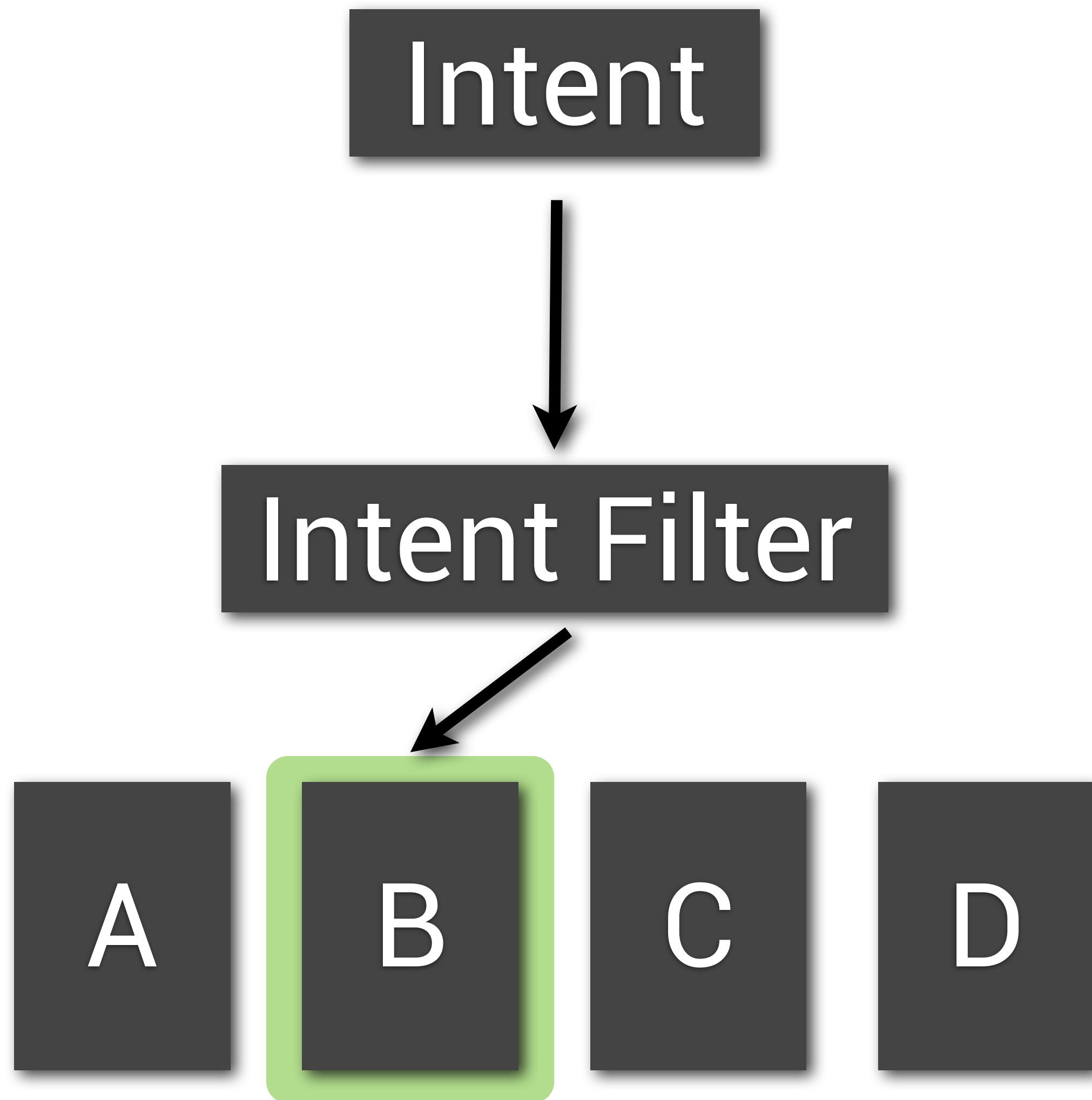


Principais Intents disponíveis na plataforma

Constant	Target component	Action
<code>ACTION_CALL</code>	activity	Initiate a phone call.
<code>ACTION_EDIT</code>	activity	Display data for the user to edit.
<code>ACTION_MAIN</code>	activity	Start up as the initial activity of a task, with no data input and no returned output.
<code>ACTION_SYNC</code>	activity	Synchronize data on a server with data on the mobile device.
<code>ACTION_BATTERY_LOW</code>	broadcast receiver	A warning that the battery is low.
<code>ACTION_HEADSET_PLUG</code>	broadcast receiver	A headset has been plugged into the device, or unplugged from it.
<code>ACTION_SCREEN_ON</code>	broadcast receiver	The screen has been turned on.
<code>ACTION_TIMEZONE_CHANGED</code>	broadcast receiver	The setting for the time zone has changed.



Como a plataforma sabe qual componente chamar?



São os **Intents Filters** que irão filtrar as Intents de acordo com sua ação, categoria ou dados que contém.



Exemplo de Intent Filter

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.androidnarede.meuspresentesapp"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="16" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="br.com.androidnarede.meuspresentesapp.SplashActivity"
            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>
```



**Aprendendo
passo-a-passo a
configurar o
ambiente de
desenvolvimento**





Android Developer Guide

The screenshot shows the top navigation bar with categories: Design, Develop, and Distribute. Under 'Develop', 'Android Training' is highlighted. Below the navigation is a large image of the Android robot character inside a jar filled with colorful jelly beans. To the left of the image is a left arrow, and to the right is a right arrow. The text 'Jelly Bean now available!' is displayed above a paragraph about the features of Android 4.1. A blue button labeled 'More about Jelly Bean' is at the bottom of this section. At the very bottom of the page are links for 'About Android', 'Get the SDK', 'Open Source', 'Support', and 'Legal'.

**Guia essencial
para qualquer
desenvolvedor
Android,
iniciante ou
expert.**

<http://d.android.com/develop/>



Android Developer Guide

The screenshot shows the 'Training' section of the Android Developers website. The navigation bar at the top includes links for 'Developers', 'Design', 'Develop', 'Distribute', and a search bar. Below the navigation, there are tabs for 'Training', 'API Guides', 'Reference', 'Tools', and 'Google Services'. The 'Training' tab is selected. On the left, a sidebar menu under 'Getting Started' lists several categories: 'Building Your First App', 'Managing the Activity Lifecycle', 'Supporting Different Devices', 'Building a Dynamic UI with Fragments', 'Saving Data', 'Interacting with Other Apps', 'Sharing Content', 'Building Apps with Multimedia', 'Building Apps with Graphics & Animation', and 'Building Apps with Connectivity & the Cloud'. The main content area starts with a heading 'Getting Started' and a welcome message about finding sets of lessons for specific tasks. It then focuses on the 'Building Your First App' class, which is marked with a '1 2 3' icon. This class covers creating an Android Project, running the application, building a simple user interface, and starting another activity. Below this, another section titled 'Managing the Activity Lifecycle' is partially visible.

Local para
aprender
passo-a-passo
aspectos gerais
da plataforma.

<http://d.android.com/training/>



Android Developer Guide

The screenshot shows the Android Developers website with the navigation bar "Develop" selected. The main content area is titled "App Components" with a sub-section "APP FUNDAMENTALS". On the left, there's a sidebar with a tree view of "App Components" including "App Fundamentals", "Activities", "Services", "Content Providers", "Intents and Intent Filters", "Processes and Threads", "Permissions", "App Widgets", "Android Manifest", "User Interface", "App Resources", "Animation and Graphics", and "Computation". Below the sidebar, there are sections for "BLOG ARTICLES" (e.g., "Using DialogFragments") and "TRAINING" (e.g., "Managing the Activity Lifecycle"). A large image of a smartphone with floating blue squares represents the app components.

<http://d.android.com/guide/>

Aprenda em detalhes a desenvolver utilizando as APIs que a plataforma disponibiliza.



Android Developer Guide

The screenshot shows the Android Developers website with the 'Reference' tab selected in the top navigation bar. The main content area is titled 'Package Index' and displays a list of Java packages under the 'Android APIs' category at API level 17. Each package has a brief description. The packages listed are: android, android.accessibilityservice, android.accounts, android.animation, android.app, android.app.admin, android.app.backup, android.appwidget, android.bluetooth, android.content, android.content.pm, android.content.res, android.database, and android.database.sqlite. A sidebar on the left lists additional categories like Training, API Guides, Tools, and Google Services.

Package	Description
android	Contains resource classes used by applications included in the platform and defines application permissions for system features.
android.accessibilityservice	The classes in this package are used for development of accessibility service that provide alternative or augmented feedback to the user.
android.accounts	
android.animation	These classes provide functionality for the property animation system, which allows you to animate object properties of any type. <code>int</code> , <code>float</code> , and hexadecimal color values are supported by default. You can animate any other type by telling the system how to calculate the values for that given type with a custom TypeEvaluator . For more information, see the Animation guide.
android.app	Contains high-level classes encapsulating the overall Android application model.
android.app.admin	Provides device administration features at the system level, allowing you to create security-aware

**Referência
para
consultar as
informações
técnicas de
cada API.**

<http://d.android.com/reference/>



Android Developer Guide

The screenshot shows the 'Developer Tools' section of the Android Developers website. The top navigation bar includes links for 'Design', 'Develop' (which is highlighted in orange), and 'Distribute'. Below the navigation are tabs for 'Training', 'API Guides', 'Reference', 'Tools' (also highlighted in orange), and 'Google Services'. On the left, a sidebar menu under 'Developer Tools' lists 'Download', 'Workflow', 'Tools Help', 'Revisions', 'Extras', 'Samples', and 'ADK'. The main content area features a heading 'Developer Tools' and two sections: 'The Android Developer Tools (ADT) plugin for Eclipse provides a professional-grade development environment for building Android apps. It's a full Java IDE with advanced features to help you build, test, debug, and package your Android apps.' and 'Free, open-source, and runs on most major OS platforms. To get started, [download the Android SDK](#)'. Below the text are images of a smartphone and a tablet displaying the Android welcome screen, and a screenshot of the Eclipse IDE interface with the ADT plugin.

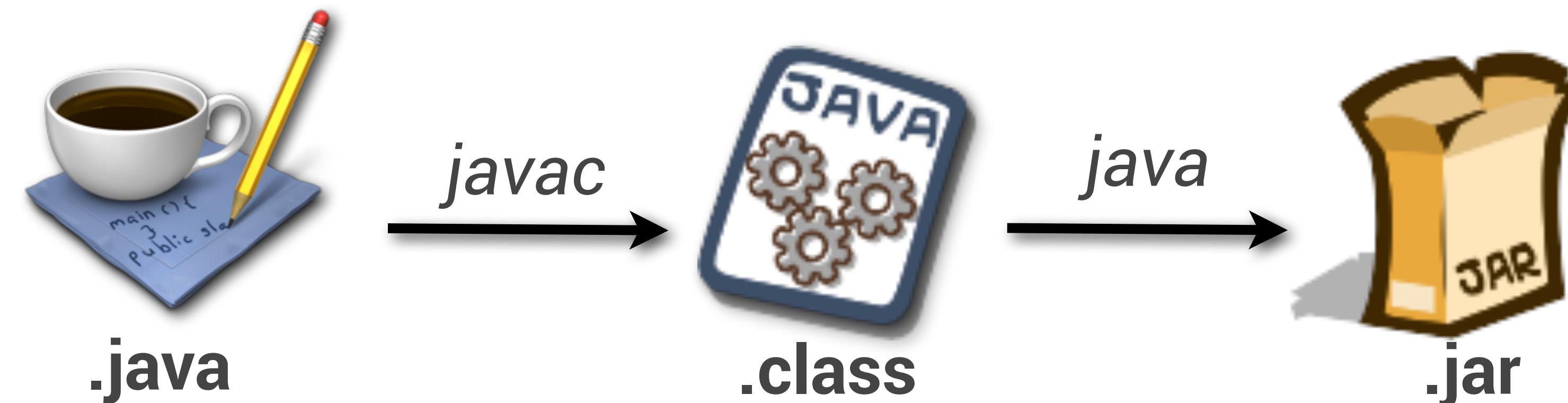
<http://d.android.com/tools/>

Encontre as ferramentas necessárias para configurar o ambiente de desenvolvimento.



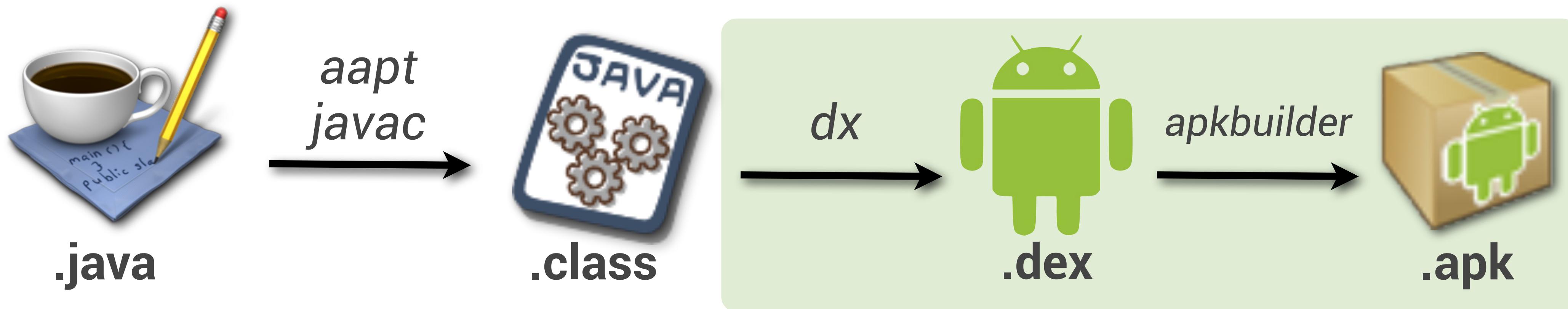
Java vs. Android

No desenvolvimento comum em Java, nós temos este cenário.



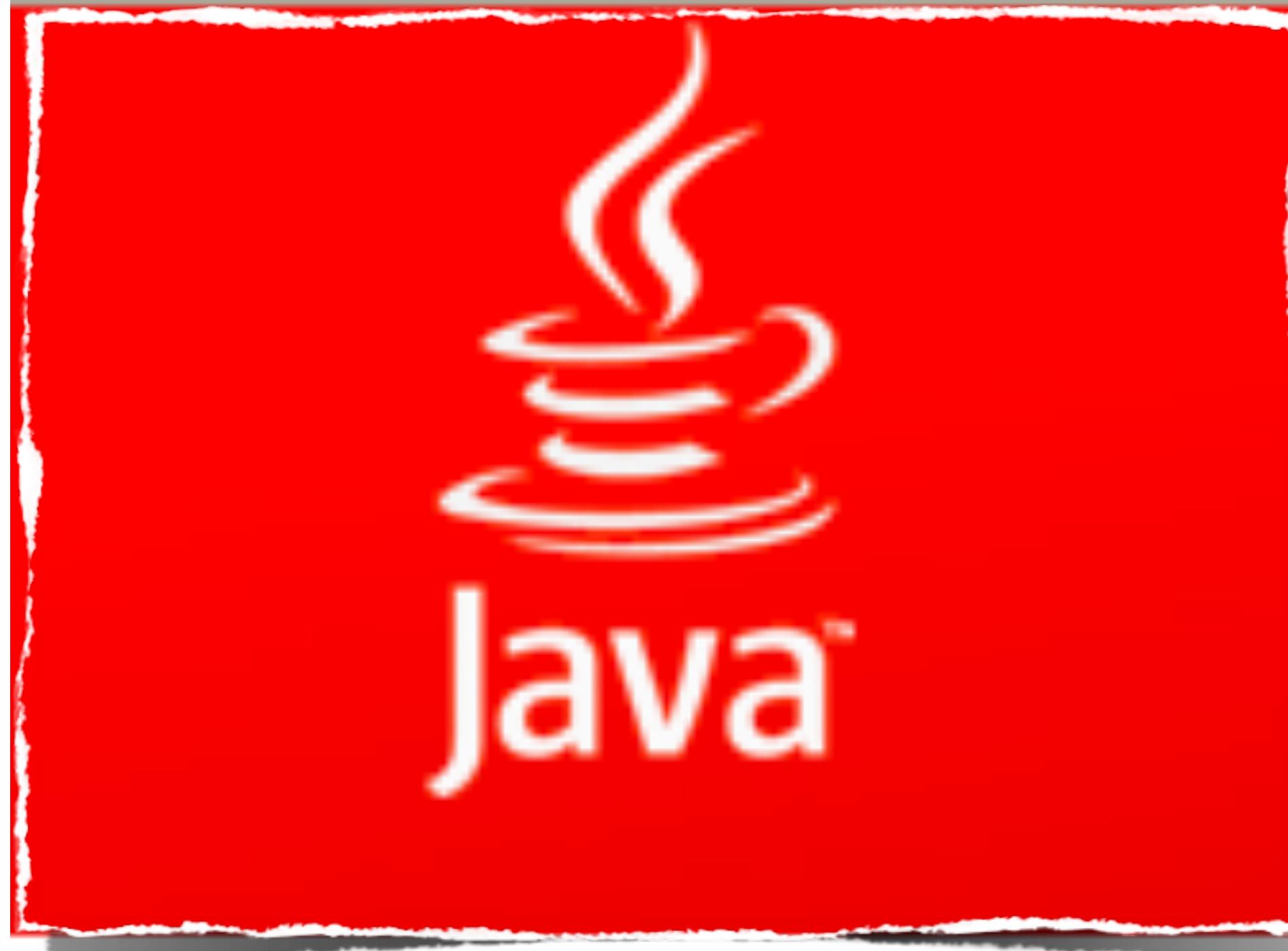


Em Android, temos uma etapa adicional, necessária pela máquina virtual Dalvik.





Baixando e instalando as ferramentas: Oracle Java



<http://goo.gl/Llt7U>

Pelo fato do desenvolvimento em Android ser em Java, é necessário ter instalada uma JDK (5+)

Atenção: JDK 7 pode ser utilizado, porém você deve configurar seu projeto para compilar com o **Java 6**. Para isso, vá na propriedade do seu projeto > *Java Compiler* > Altere *Compiler Compliance Level* para 1.6.



Baixando e instalando as ferramentas: IDEs

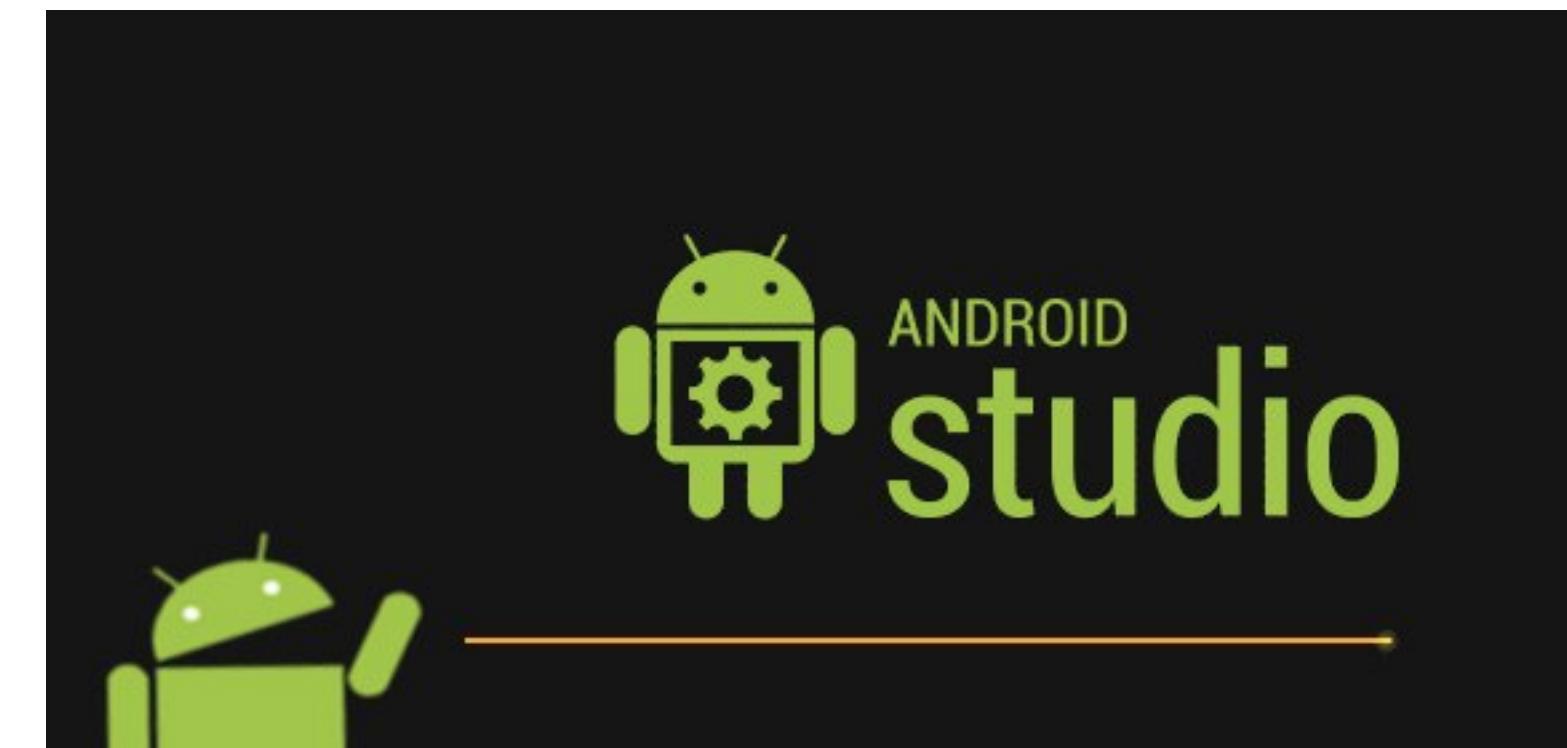
Temos duas excelentes IDEs:

Eclipse
ADT



<http://goo.gl/S1zkTG>

IntelliJ
Android Studio



<http://goo.gl/QcetuV>



Baixando o ADT



Simplesmente, basta baixar e descompactar o **ADT Bundle** que já vem com:

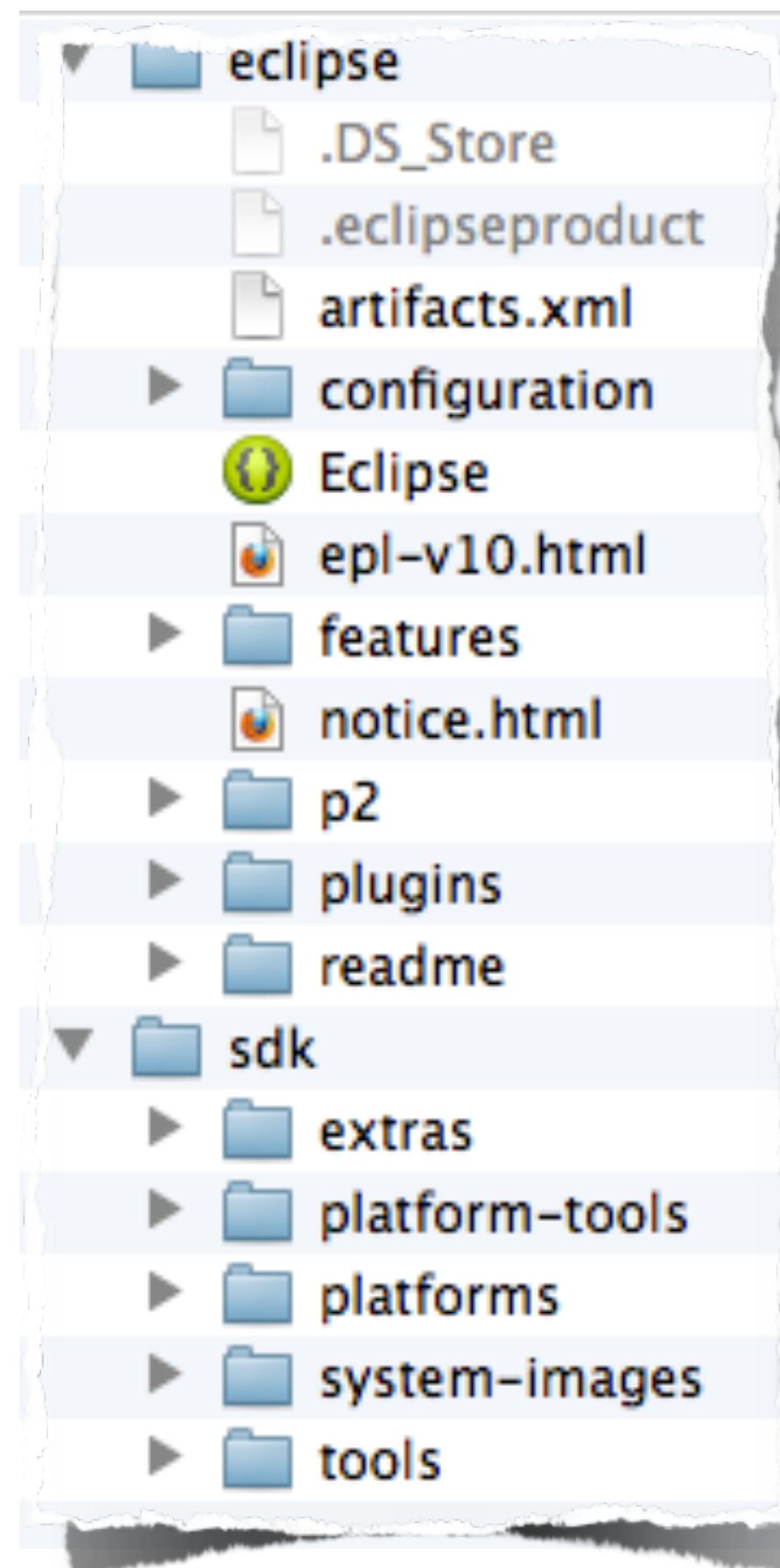
- Eclipse + ADT Plugin
- Android SDK Tools
- Android Platform-tools
- Versão mais recente da plataforma
- Imagem da versão atual da plataforma para utilizar no emulador

<http://d.android.com/sdk/>



Instalando o ADT

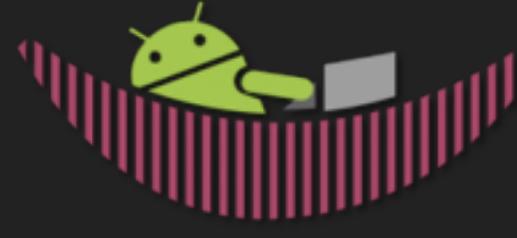
1. Descompacte o arquivo zip *adt-bundle-<os_platform>* em um diretório de preferência, onde *<os_platform>* representa a plataforma que está utilizando: Windows, Linux ou Mac OS X.



2. Abra a pasta onde descompactou o ADT Bundle e abra a pasta *eclipse/* e execute o arquivo *eclipse*.

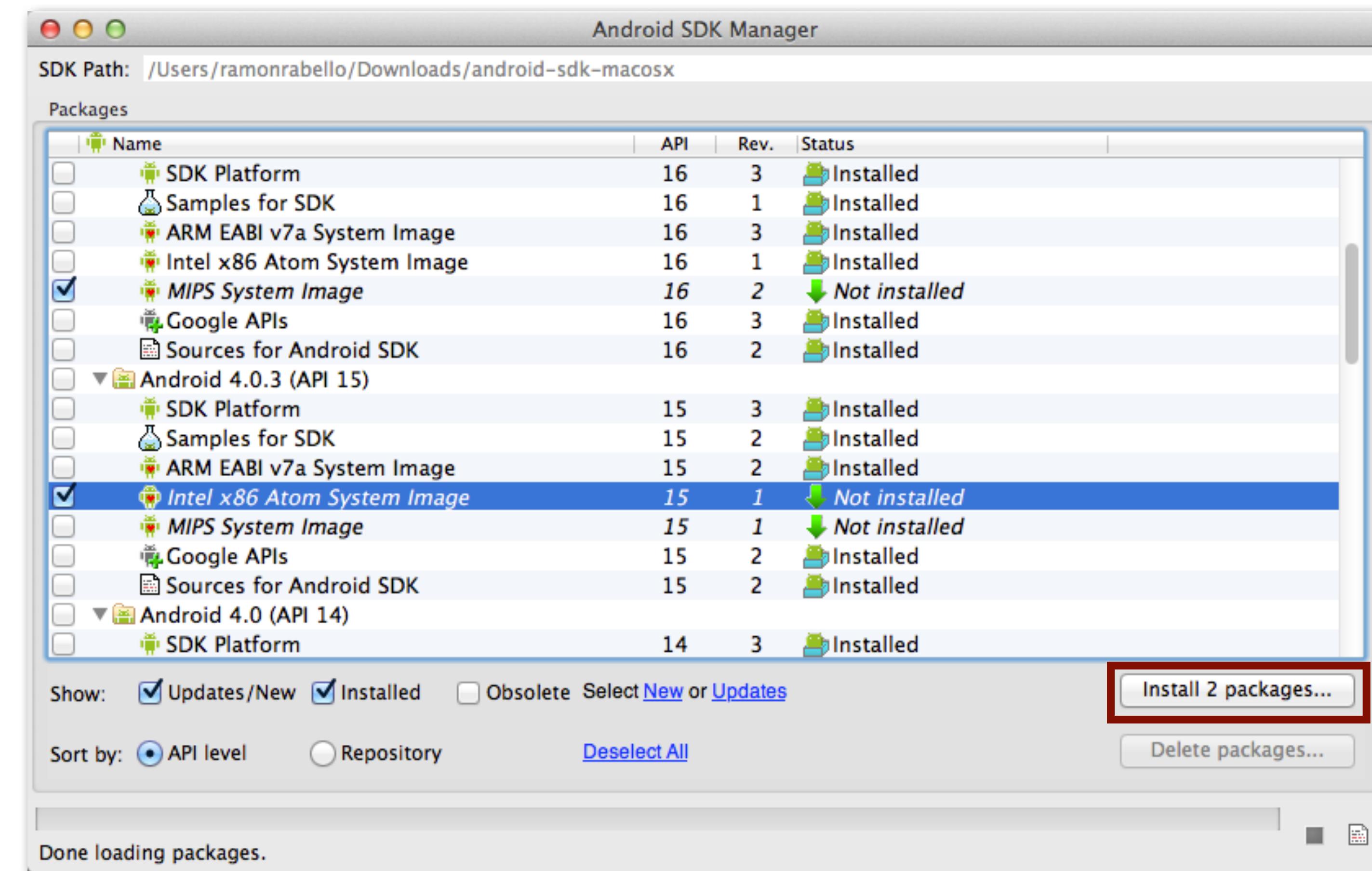
Pronto!

Seguindo apenas 2 etapas,
você já poderá a começar a
construir apps em Android!



Gerenciando versões com o Android SDK Manager

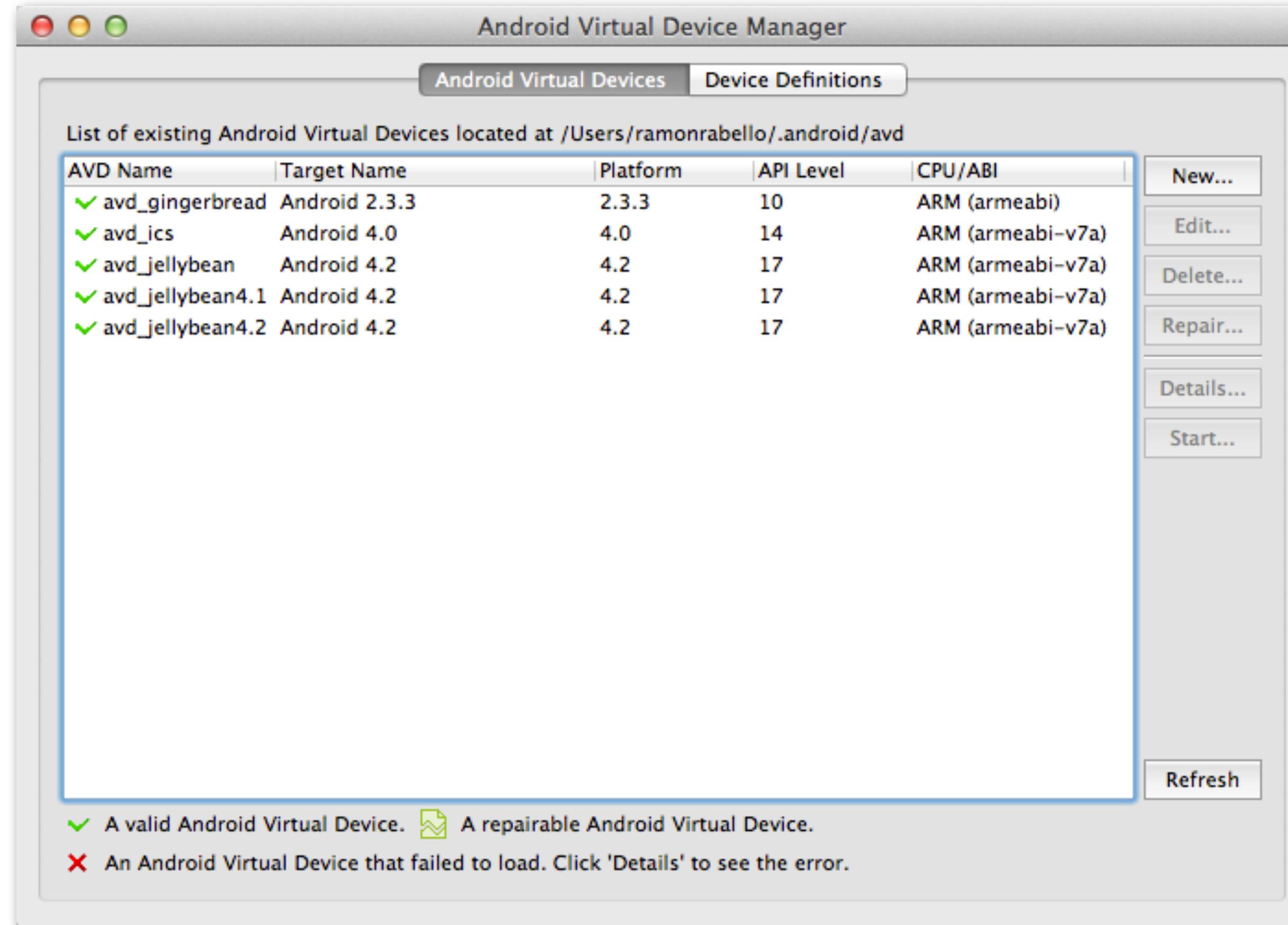
Agora você poderá baixar e instalar as versões da plataforma, extras, exemplos, etc. Para isso, basta selecionar os itens que deseja fazer download e clicar em *Install <n> Package(s)...*





Gerenciando dispositivos virtuais com AVD Manager

Android Virtual Device Manager



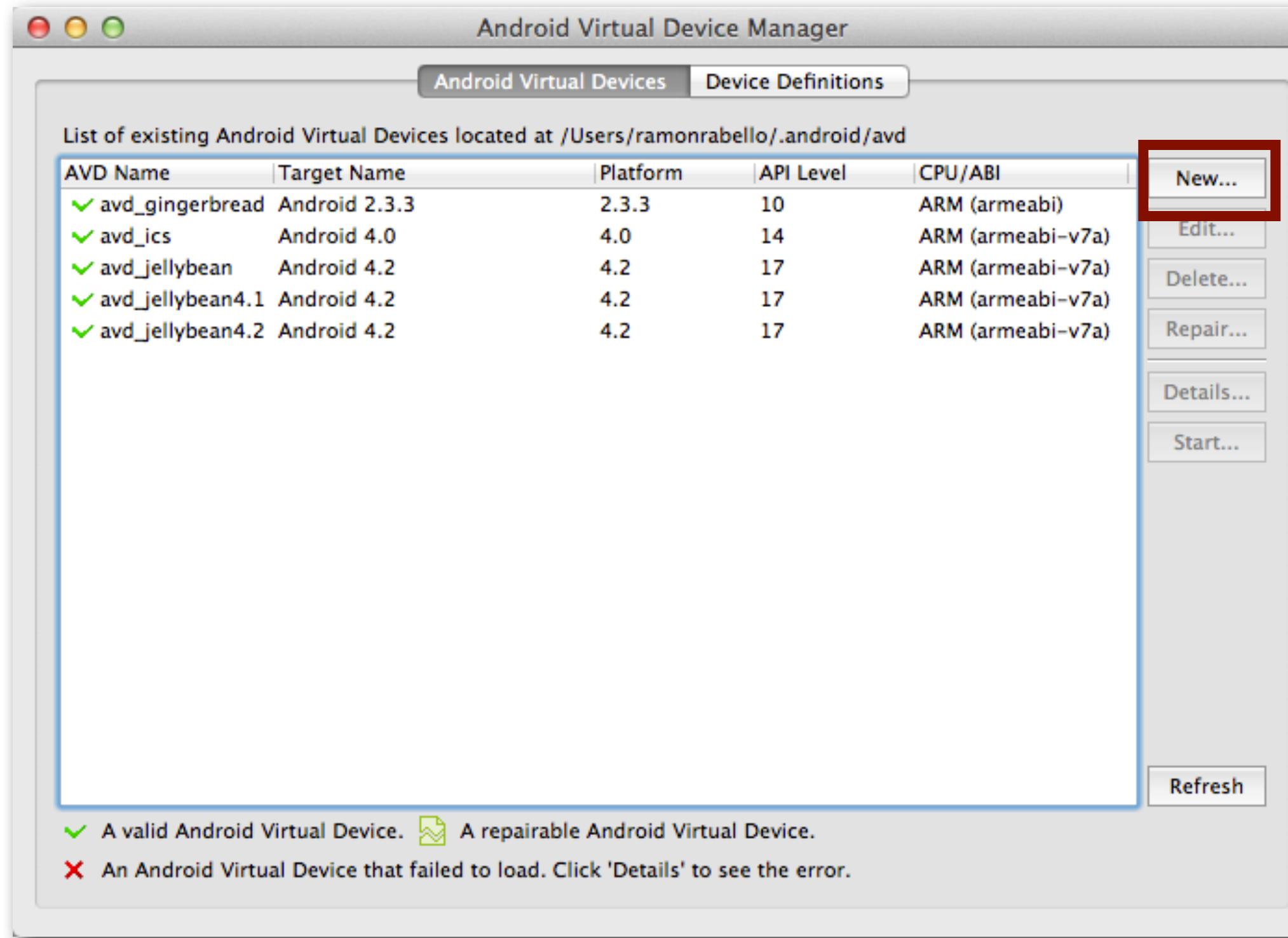
Componente que gerencia os dispositivos virtuais que serão utilizados pelo emulador quando uma app estiver sendo executada.



Criando dispositivos virtuais (AVDs)

1. Primeiramente, para acessar o AVD Manager, aponte para:

Window > *Android Virtual Device Manager* ou clique no ícone  na barra de ferramentas localizada na parte superior.

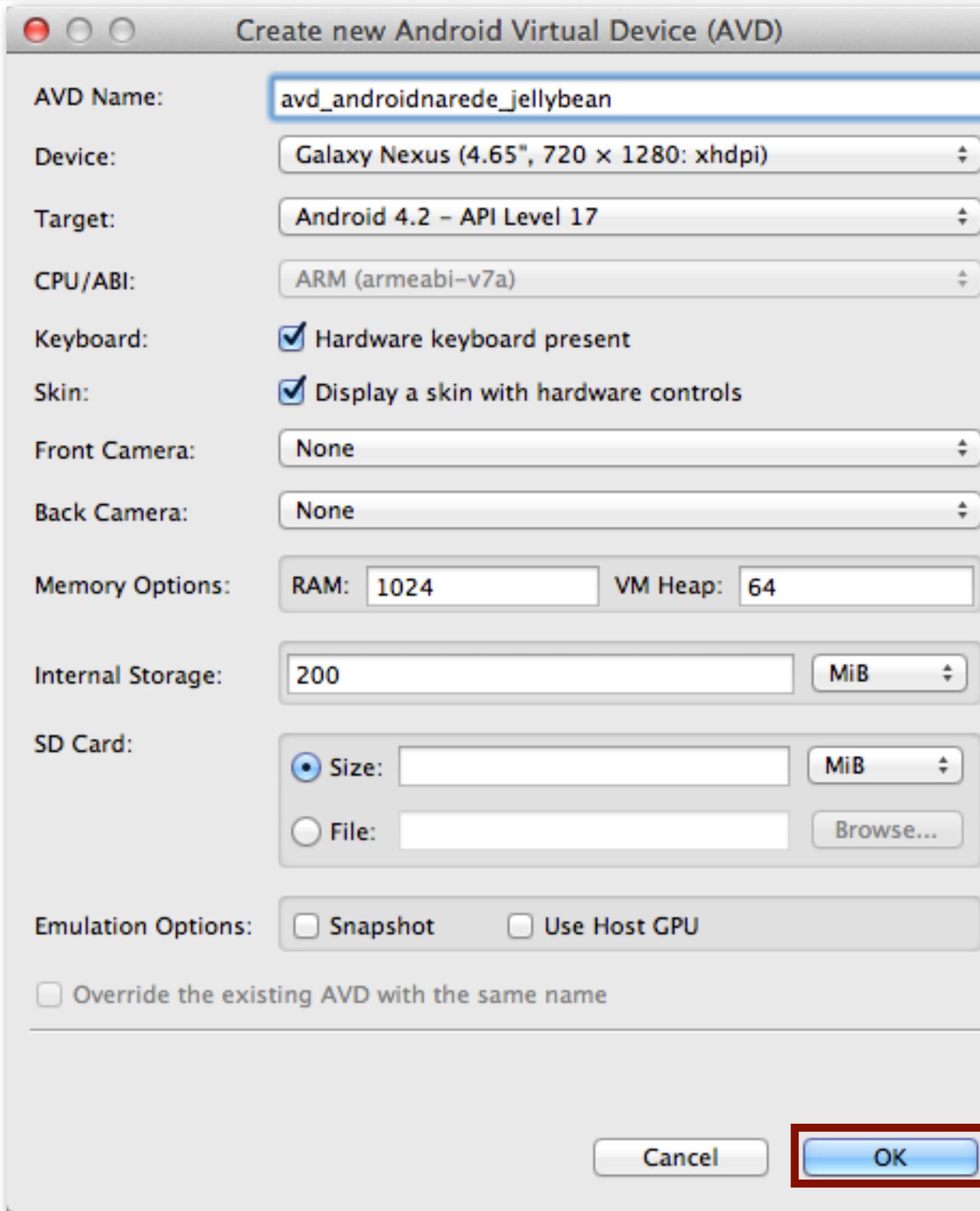


2. Na janela que abrir, clique no botão New...

Dica: Você pode criar um AVD de acordo com as configurações de dispositivo existente. Para isso, clique na aba *Device Definitions*, selecione um dispositivo na lista e clique em *Create AVD*.



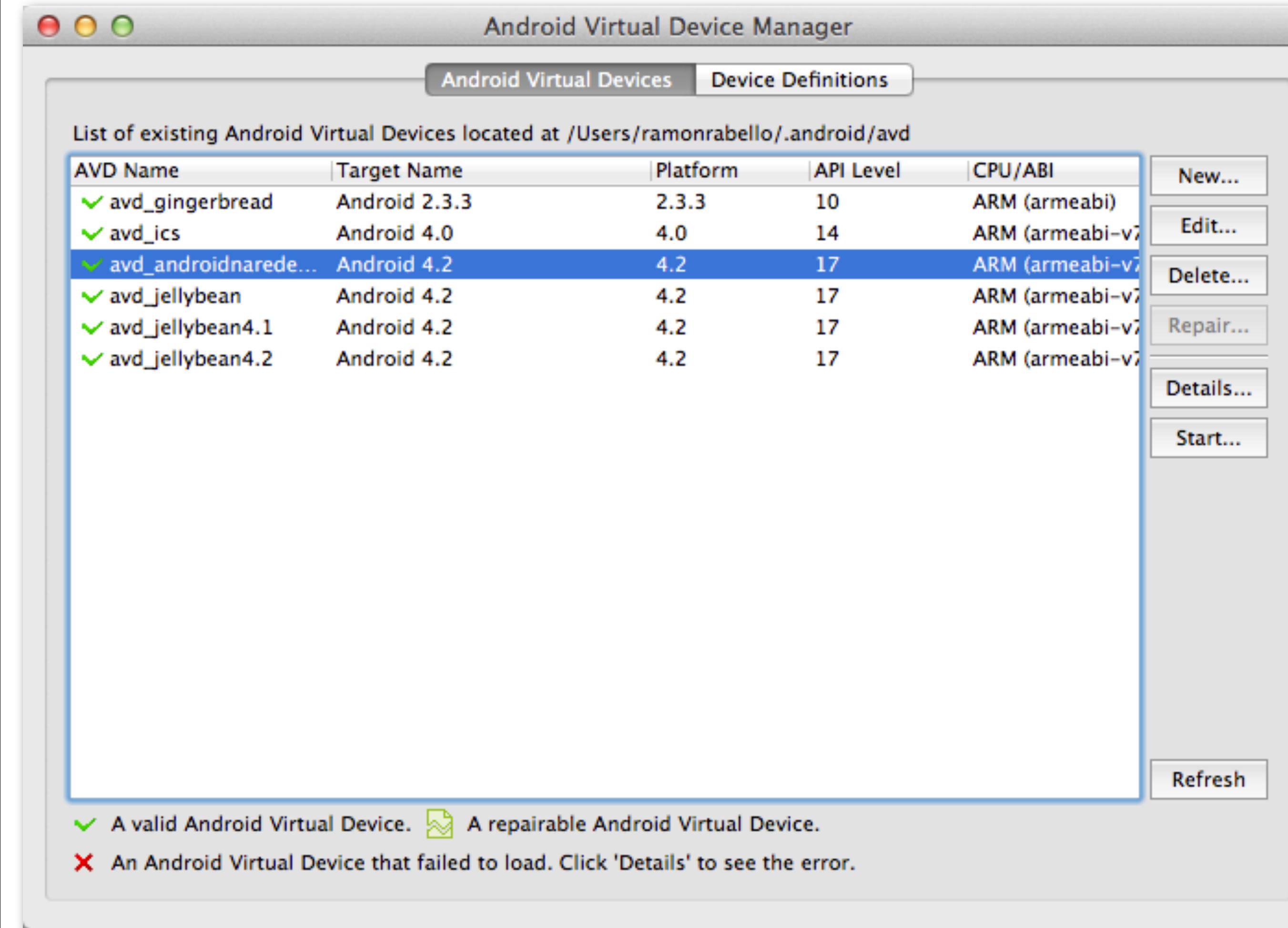
Criando dispositivos virtuais (AVDs)



3. Agora você deve preencher as informações referente ao seu AVD, bem como a descrição (*AVD Name*), dispositivo (*Device*), plataforma alvo (*Target*), suporte a teclado, etc. Para finalizar, clique em *OK*.

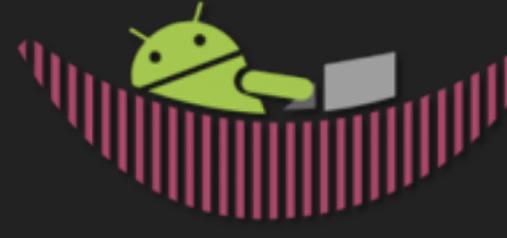


Criando dispositivos virtuais (AVDs)



Se tudo for configurado corretamente, o novo AVD deve ser aparecer na listagem dos dispositivos virtuais. Agora você já pode começar a criar sua app para ser executada no emulador.

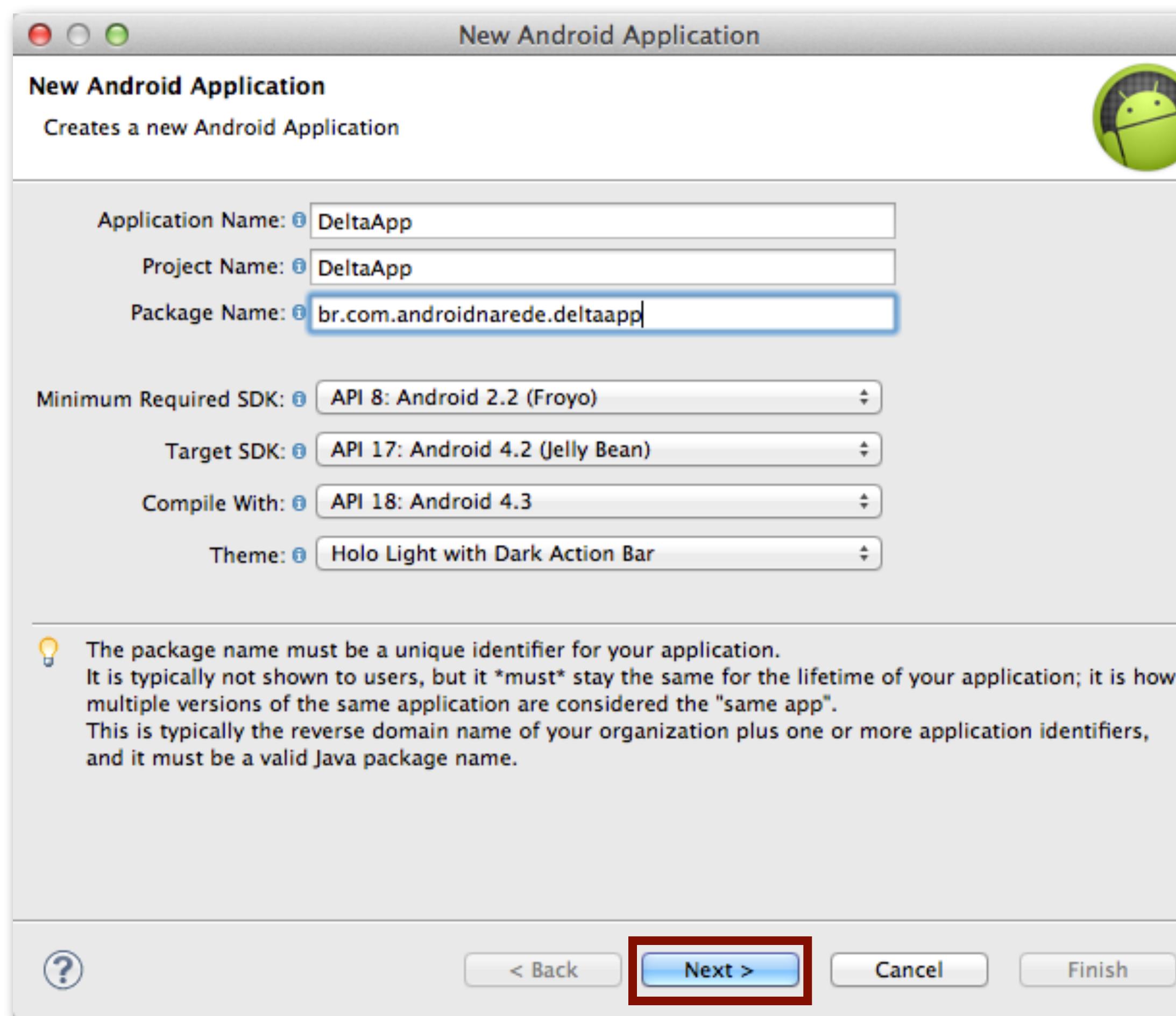
Info: Cada AVD criado é adicionado automaticamente na pasta `~/.android/avd`.



Criando o primeiro projeto em Android

1. No ADT Bundle, aponte para:

File > New > Project > Android Application Project



2. Preencha as informações do projeto e clique em Next:

Application Name: Nome da sua App

Project Name: Nome do seu projeto

Package Name: o pacote da sua app (no mínimo 2 níveis para evitar ambiguidade)

Minimum Required SDK: a versão mínima de Android que um dispositivo tem que ter para que a app seja executada.

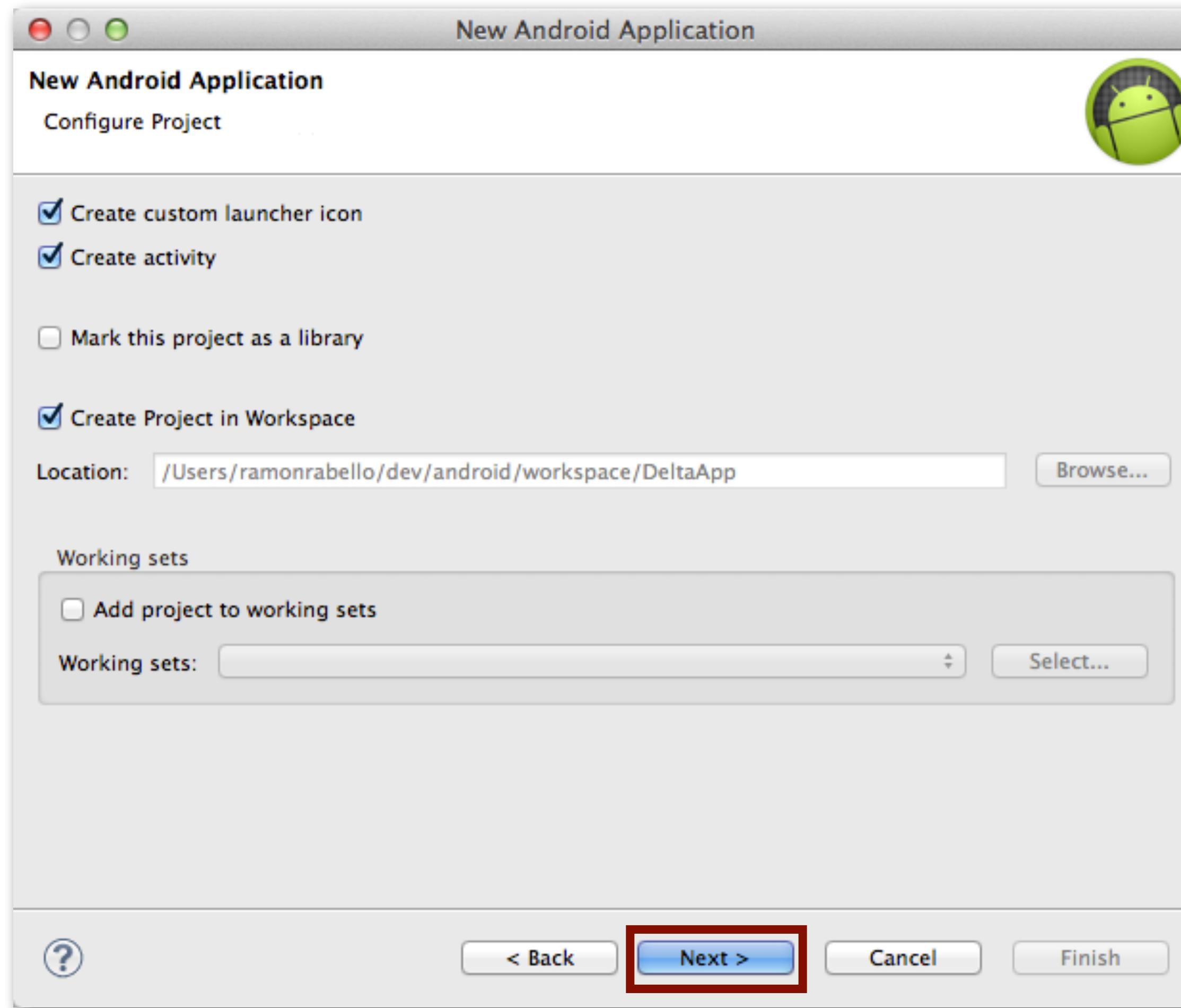
Target SDK: versão da plataforma alvo, que a app será executada

Compile With: Compilar utilizando determinado nível de API

Theme: se você não utilizará um tema (None) ou se irá ser num tom mais claro (*Holo Light/Holo Light with Dark Action Bar*) ou mais escuro (*Holo Dark*)



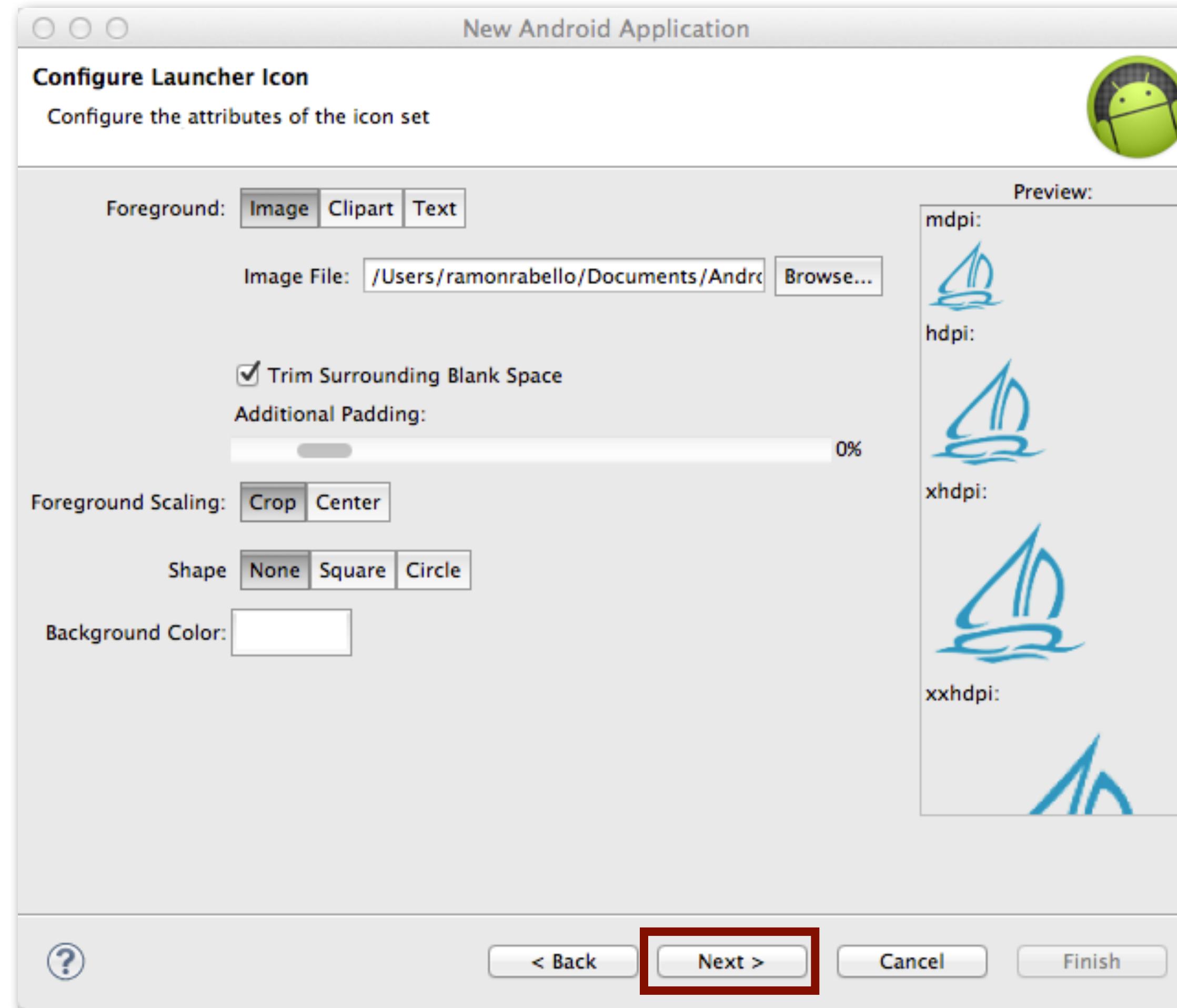
Criando o primeiro projeto em Android



3. Agora você poderá configurar seu projeto, definindo a logo e a Activity inicial para a sua app. Deixe *Create custom launcher icon* e *Create Activity*, *Create Project in Workspace* selecionado e clique em *Next*.



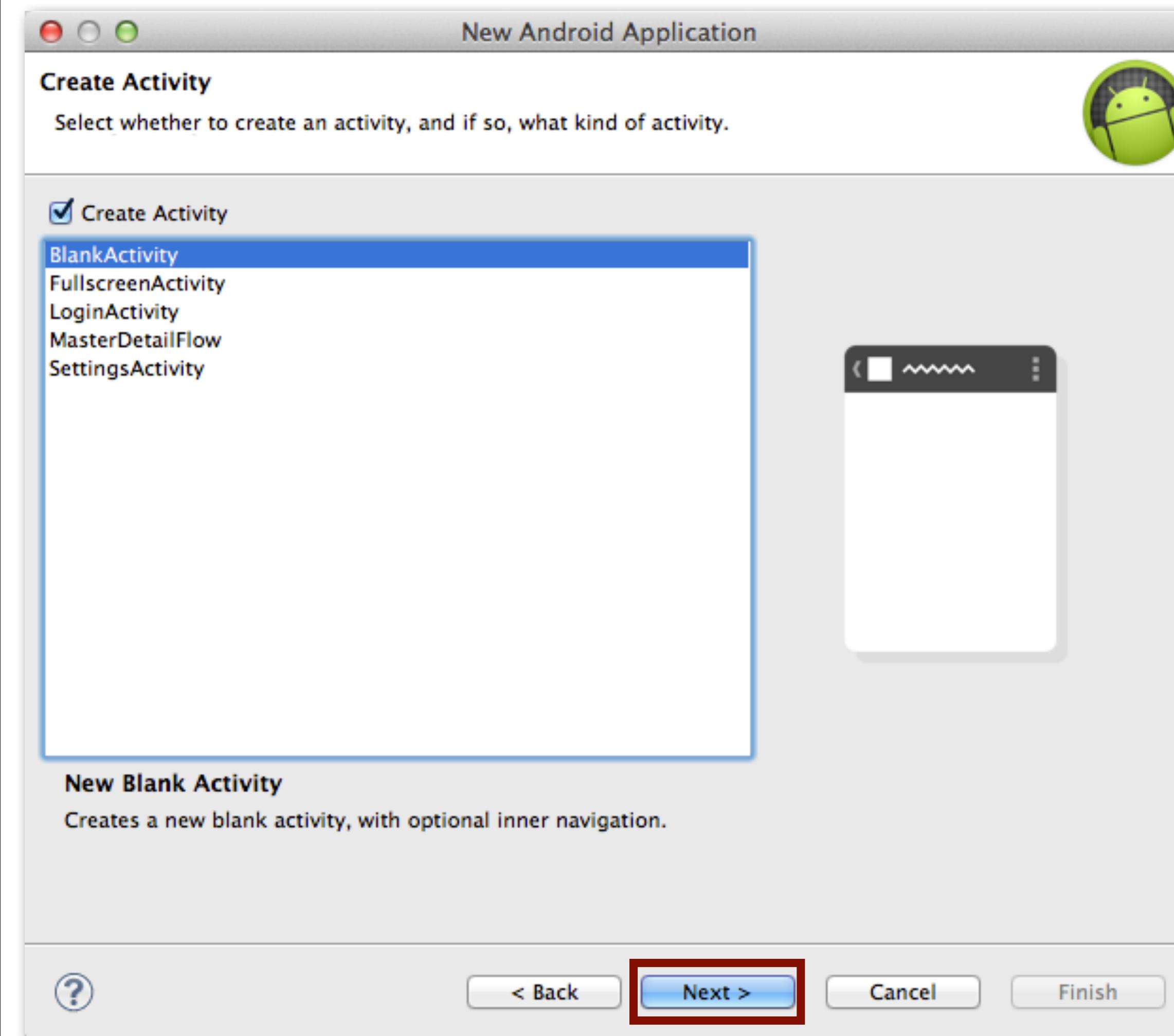
Criando o primeiro projeto em Android



4. Deixe por padrão o ícone ou escolha um de sua preferência, clicando em *Browse...*, na aba *Image*. O ADT automaticamente ajusta os ícones de acordo com as configurações dos dispositivos (ldpi, mdpi, hdpi, xhdpi). Para avançar, clique em *Next*.



Criando o primeiro projeto em Android



5. Na próxima tela, você irá escolher o template para a sua Activity (caso tenha deixado marcado “Create Activity”). Depois clique em **Next**.

BlankActivity: a Activity será criada com o mínimo necessário.

FullScreenActivity: uma Activity com lógica de esconder o Action Bar e rodapé, a deixando em tela cheia.

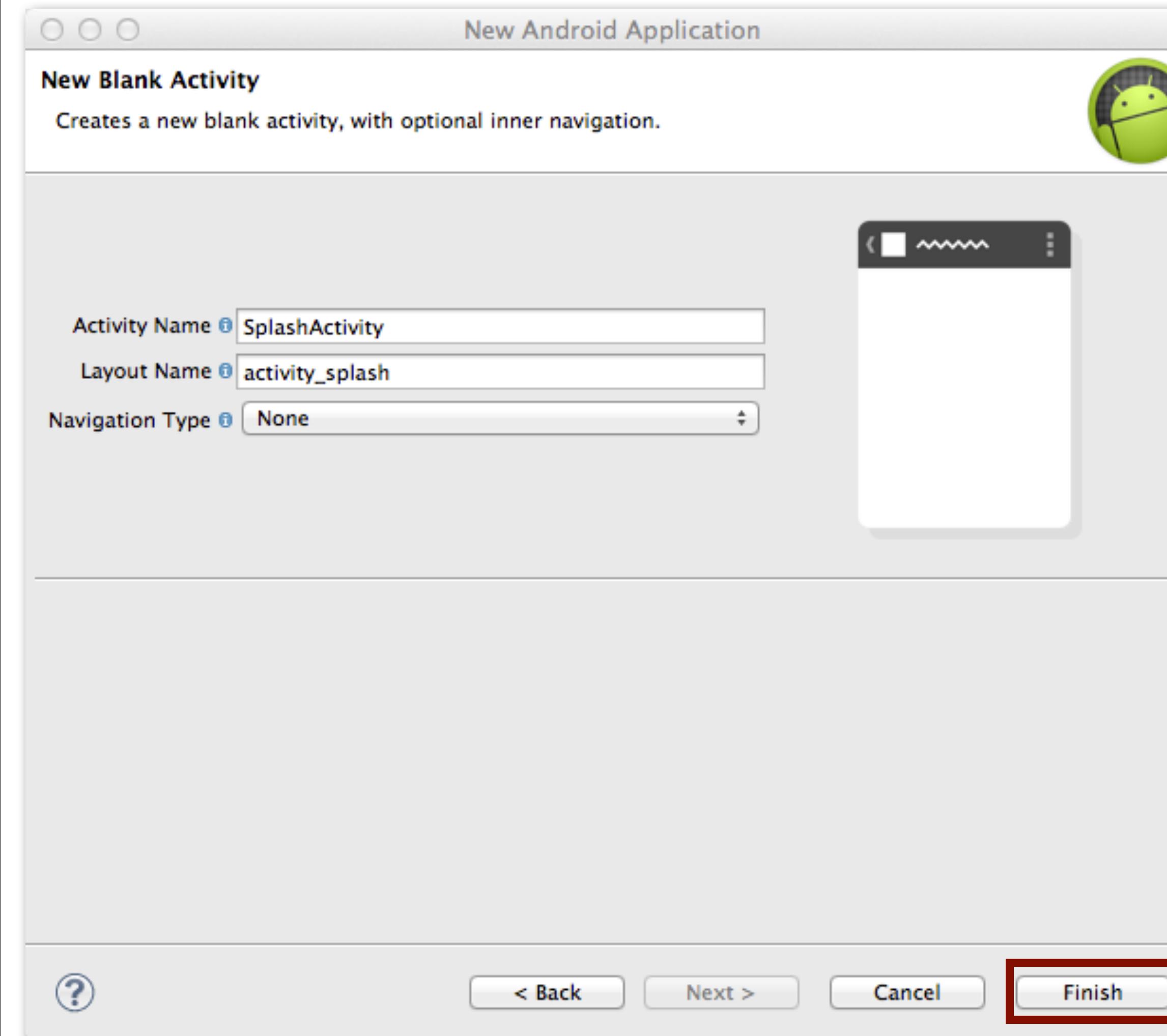
LoginActivity: Activity utilizada para apps que possuem camada de autenticação no back-end.

MasterDetailFlow: Activity no formato mestre-detalhe, onde o conteúdo é atualizado de acordo com o item selecionado. Esse template utiliza Fragments, por isso necessita que a versão mínima do SDK seja API 11 (Honeycomb).

SettingsActivity: Activity para preferências da sua app.



Criando o primeiro projeto em Android



6. Preencha as informações para a sua Activity e para concluir, clique em *Finish*.

Activity Name: o nome da classe Java para a sua Activity.

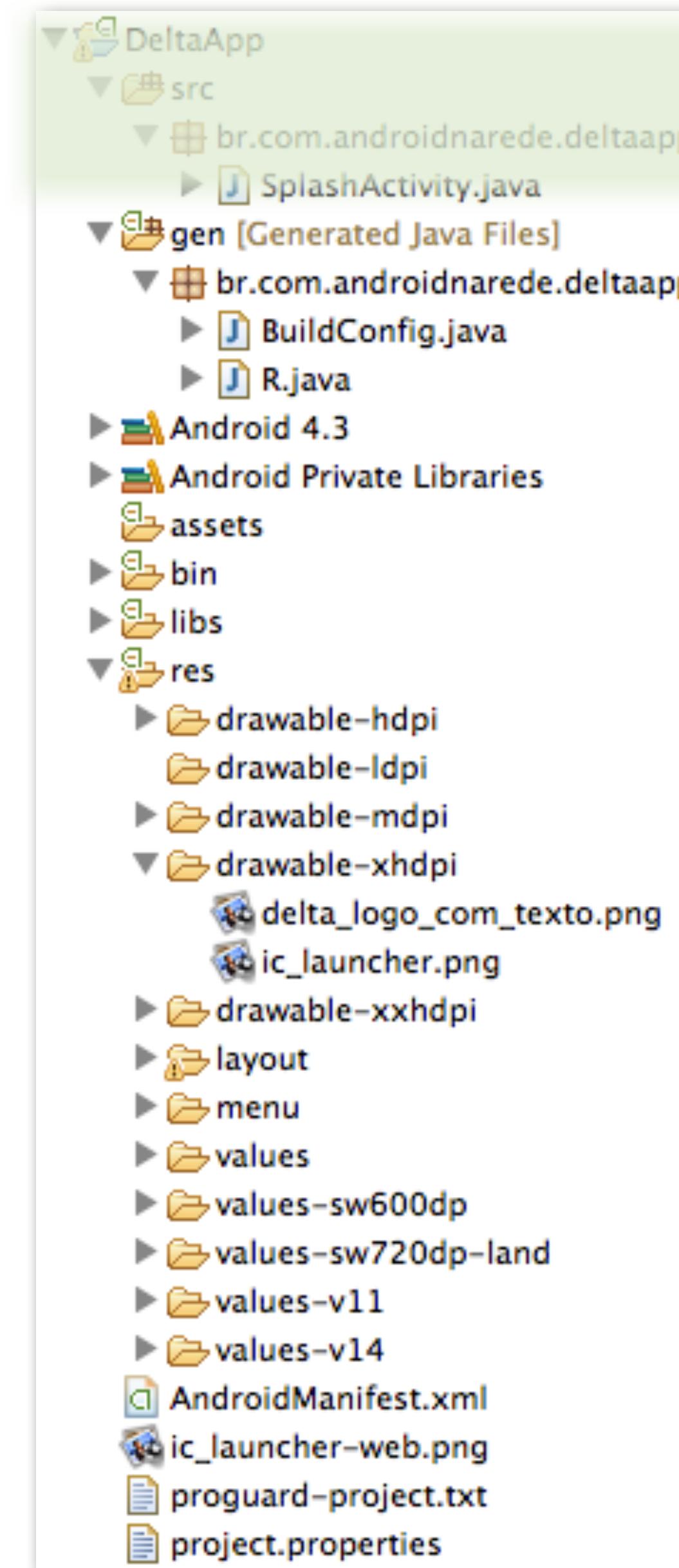
Layout Name: o nome do arquivo de layout a ser utilizado pela Activity.

Navigation Type: modelo navegacional que sua app terá:

- **Tabs:** Navegação será centrada em abas.
- **Tabs + Swipe:** Navegação com abas, permitindo que a troca de abas seja feito com o gesto de deslize para esquerda e direita. Este tipo de navegação exige API 11 (Honeycomb).
- **Swipe Views + Title Stripes:** Navegação com abas, permitindo que a troca de abas seja feito com o gesto de deslize para esquerda e direita, com animação de título da aba. Este tipo de navegação exige API 11 (Honeycomb).
- **Dropdown:** Modelo navegacional para Activity que possua várias visões (ex: GMail que pode visualizar mensagens enviadas, rascunhos, etc). Este tipo de navegação exige API 11 (Honeycomb).



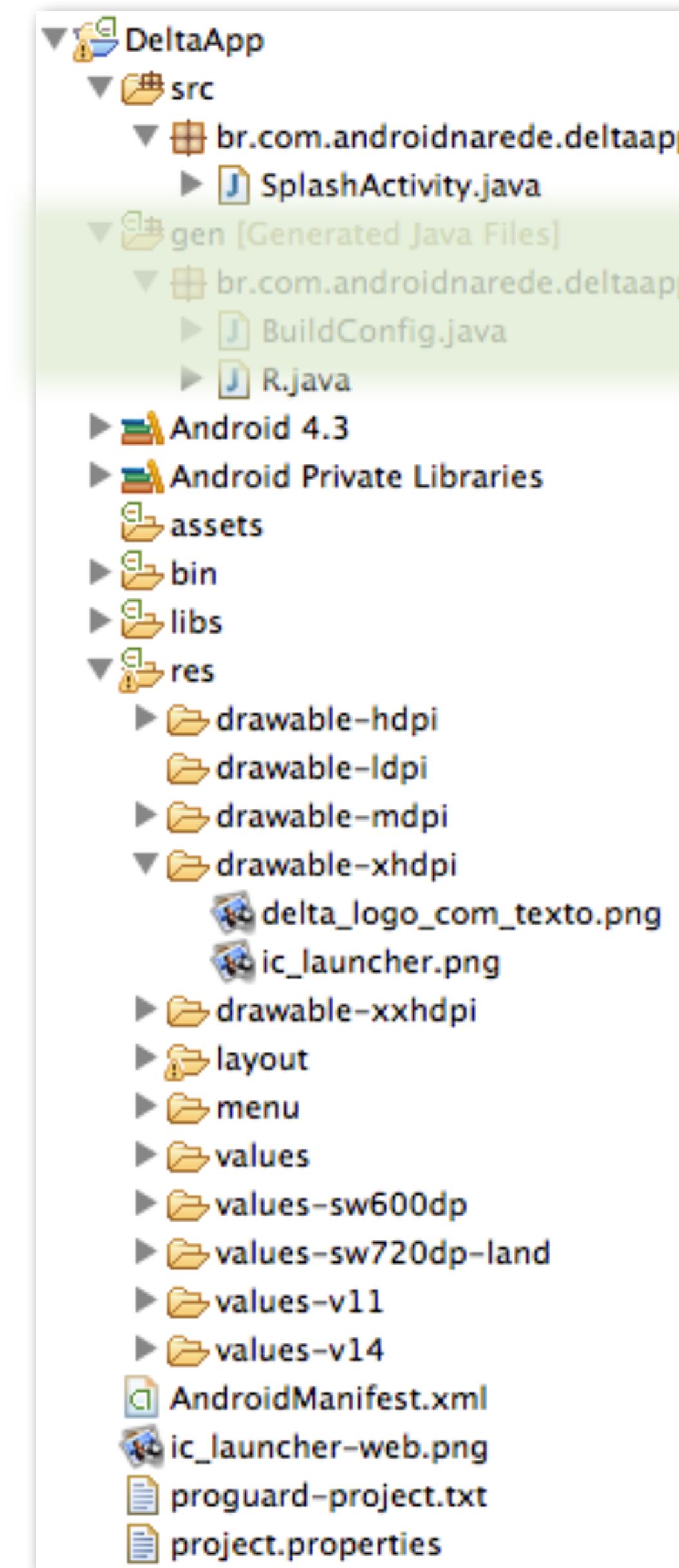
Estrutura de um projeto Android



código-fonte da aplicação,
onde ficam as classes Java



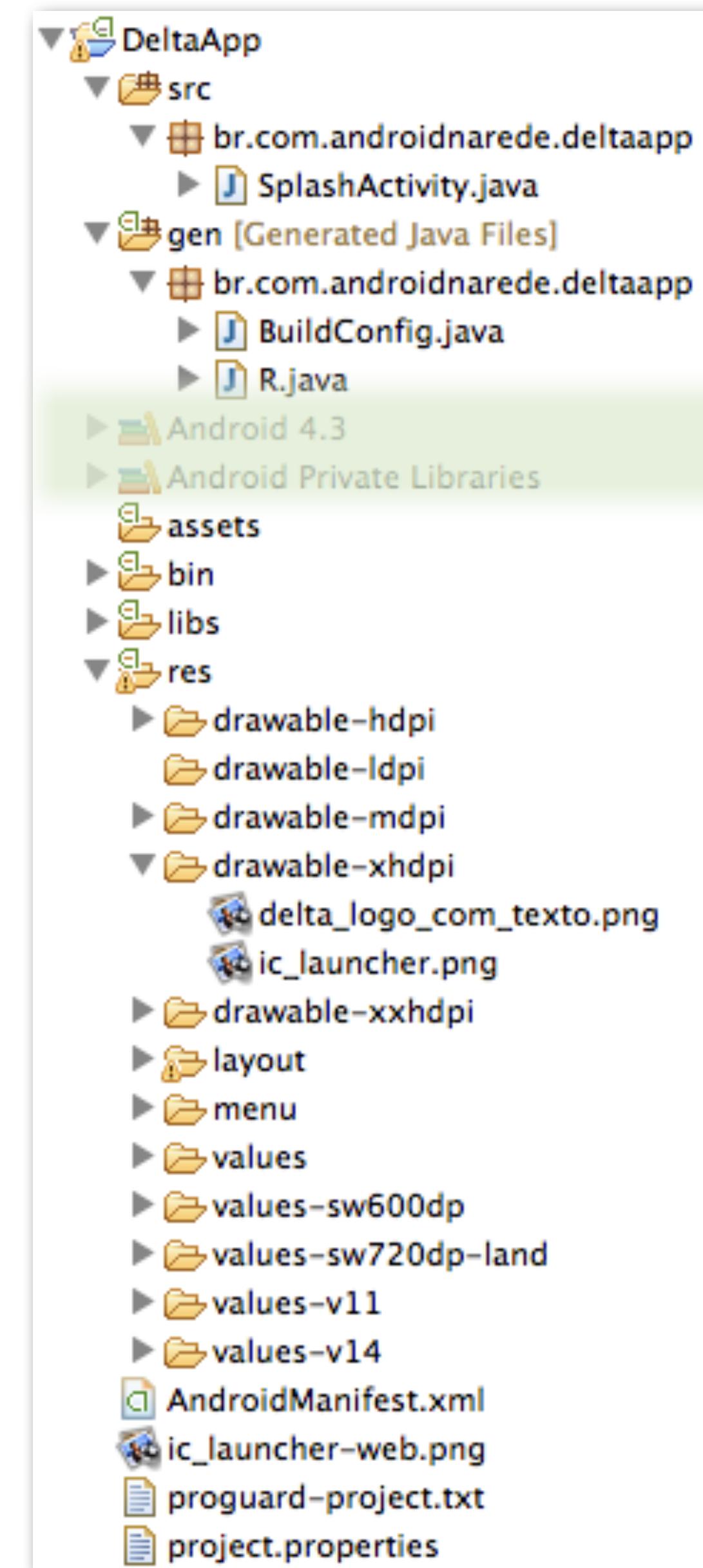
Estrutura de um projeto Android



pacote gerenciado automaticamente pela plataforma, recriado a cada modificação



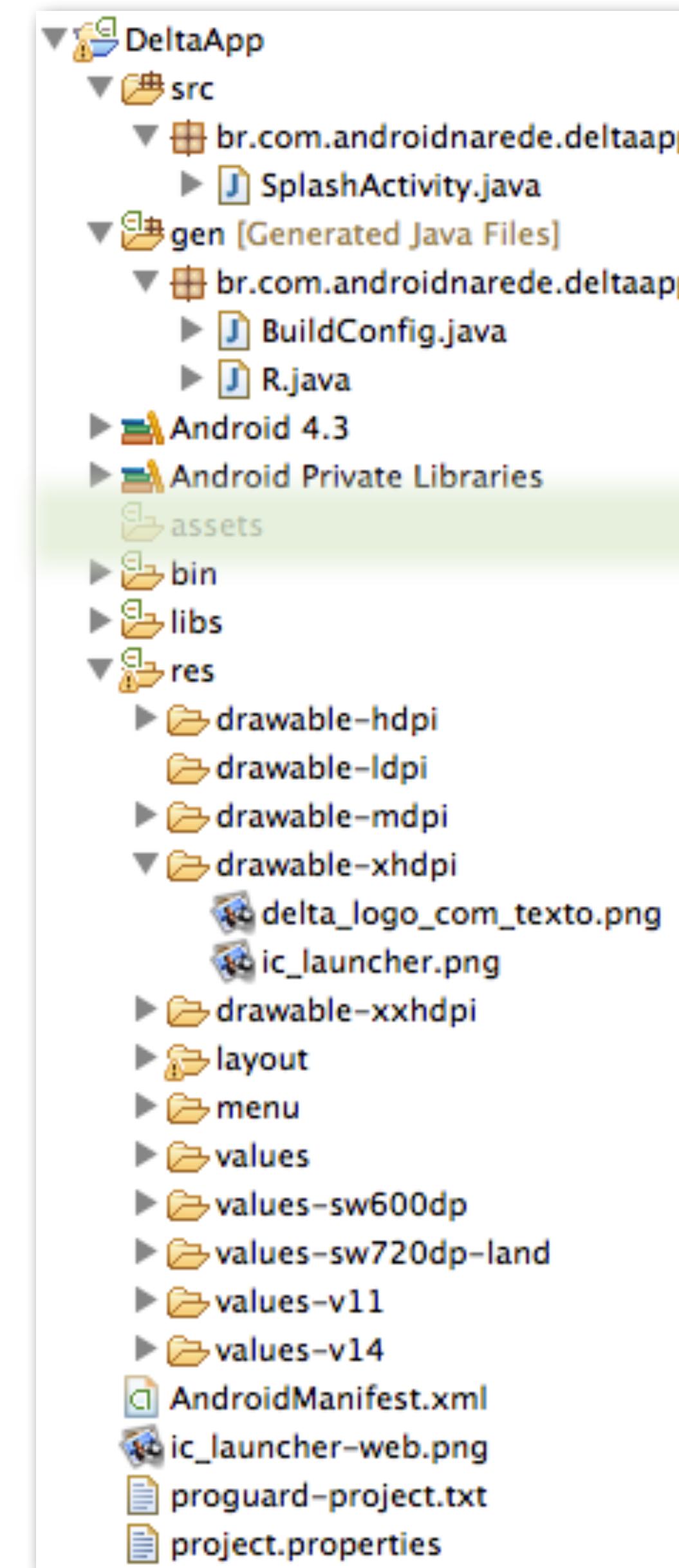
Estrutura de um projeto Android



libs do projeto



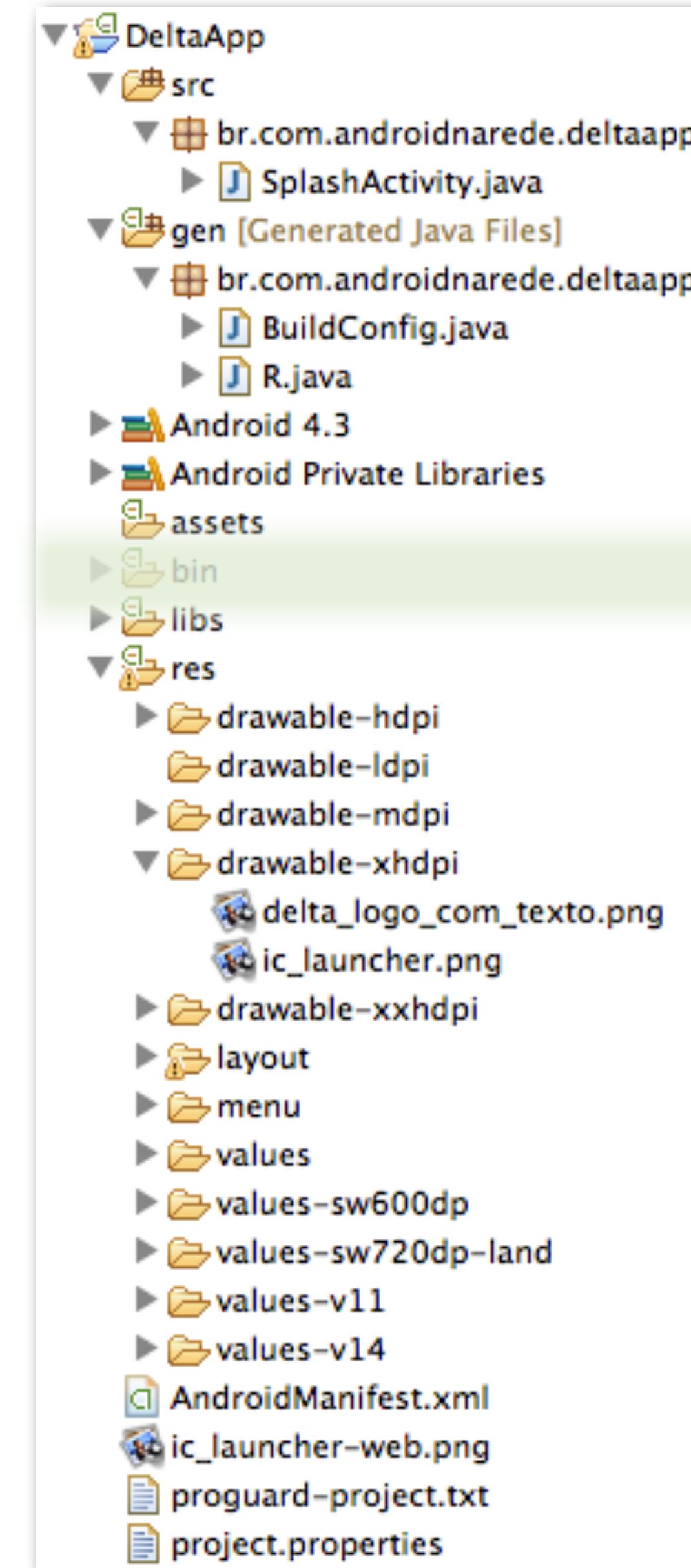
Estrutura de um projeto Android



pasta que contém recursos alternativos



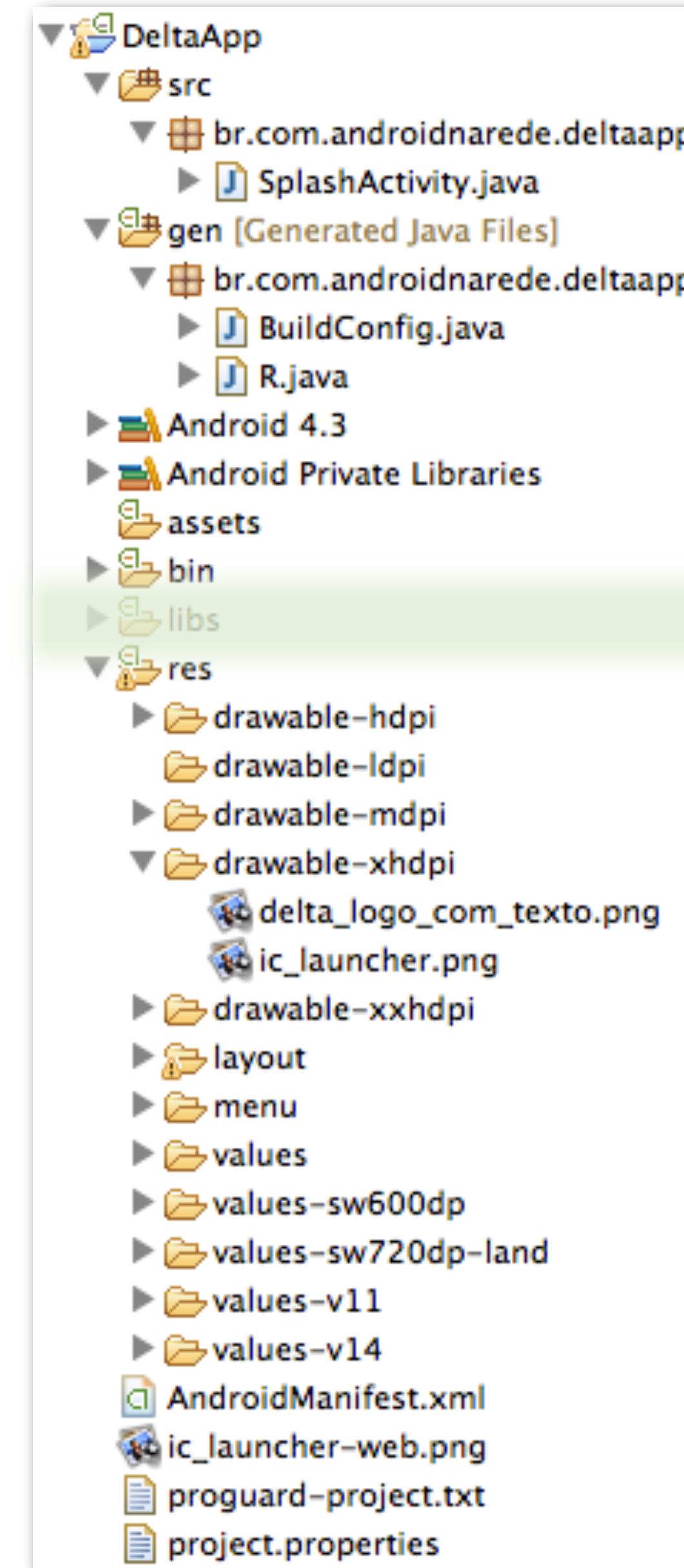
Estrutura de um projeto Android



contém arquivos compilados



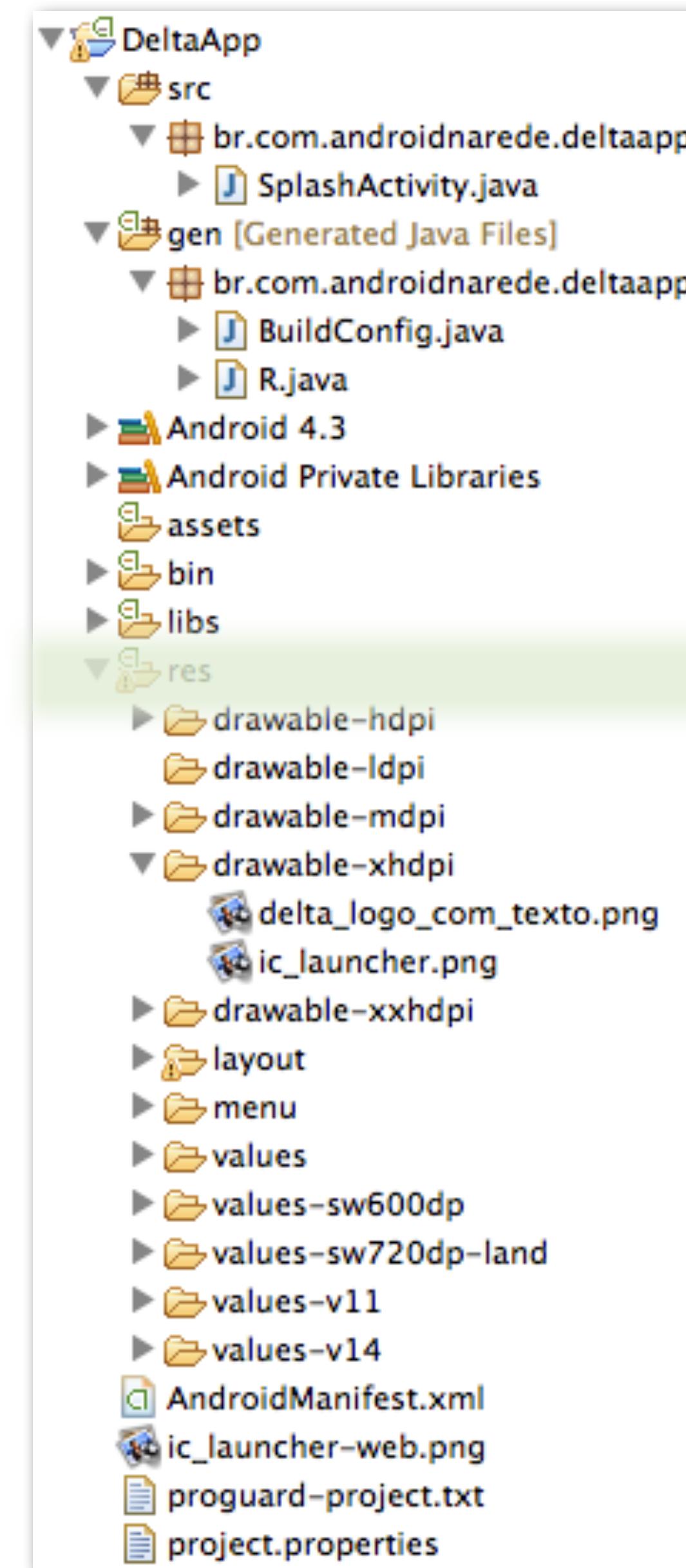
Estrutura de um projeto Android



contém as bibliotecas de terceiros



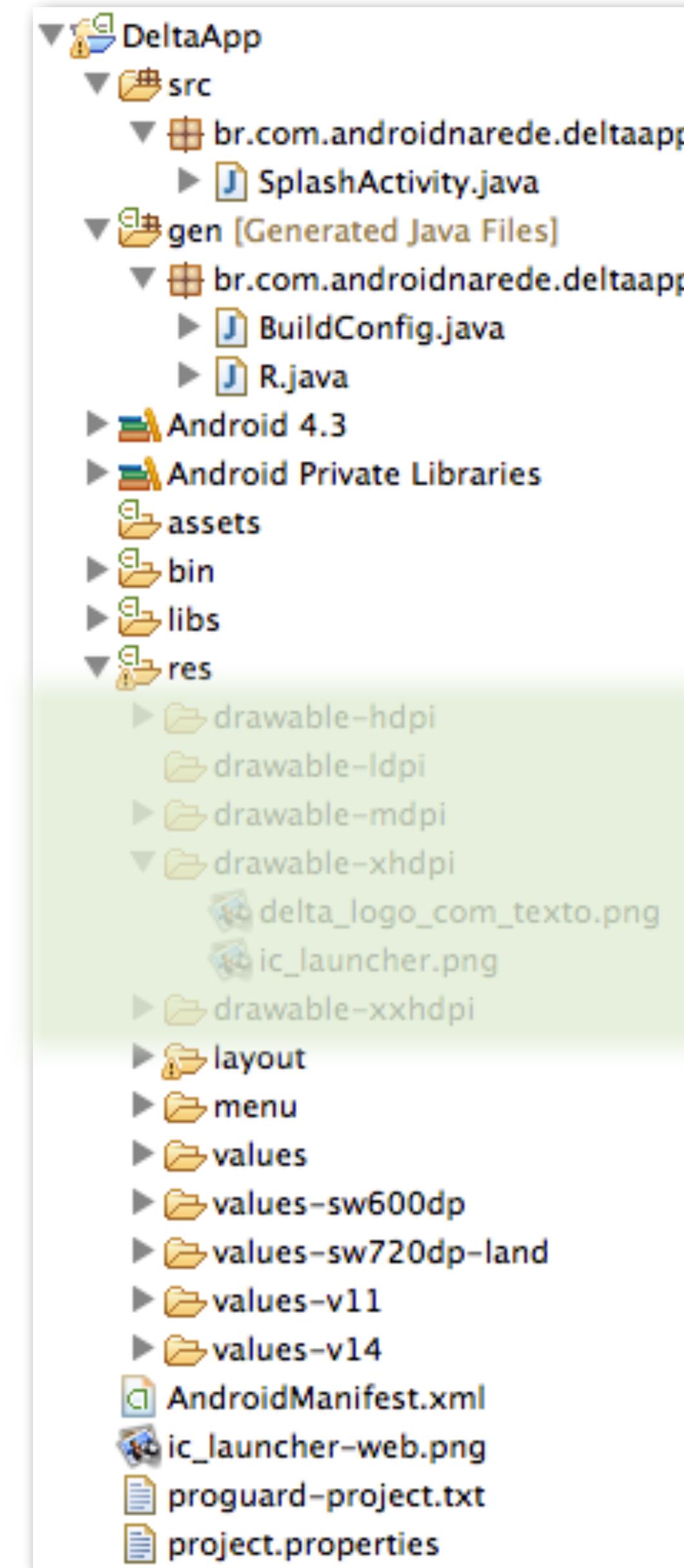
Estrutura de um projeto Android



pasta que contém os recursos



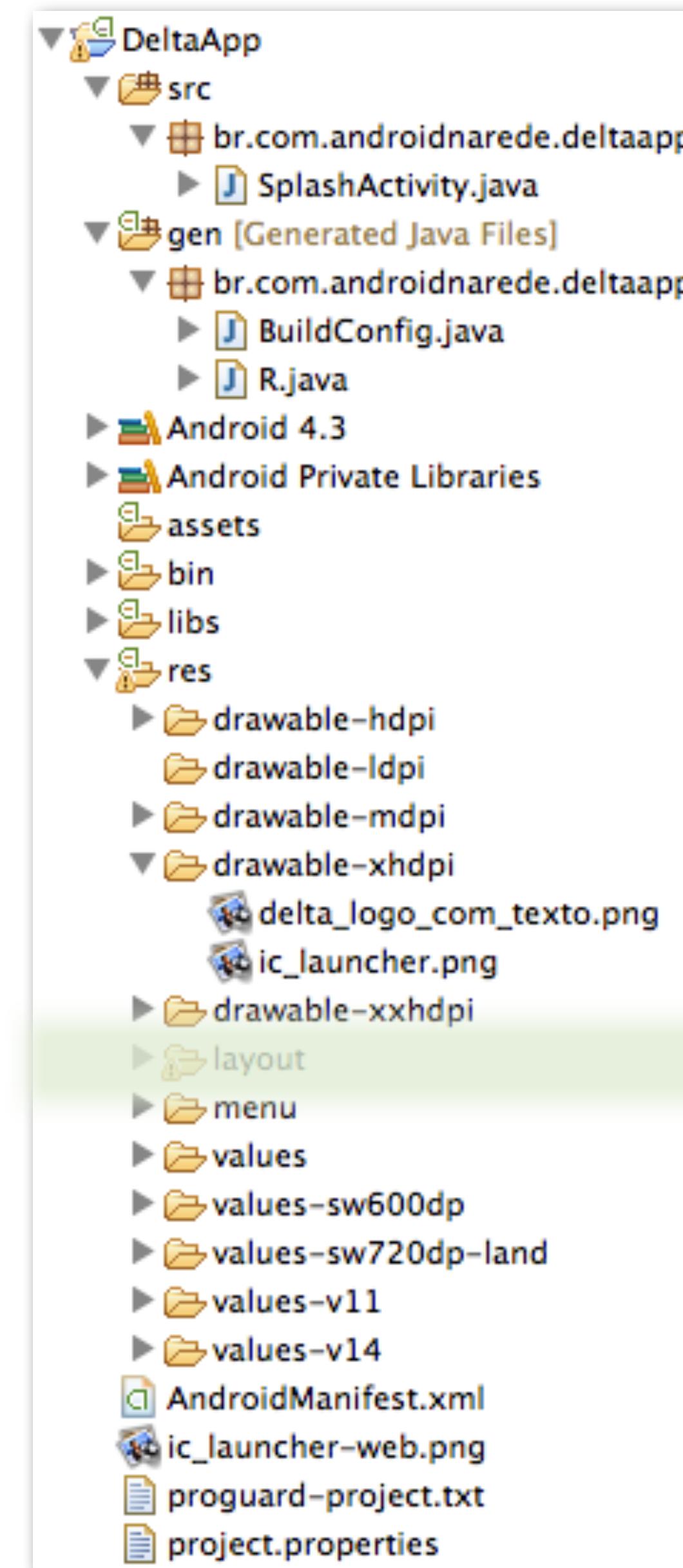
Estrutura de um projeto Android



contém as imagens (drawables) de acordo com a configuração da densidade de tela do dispositivo.



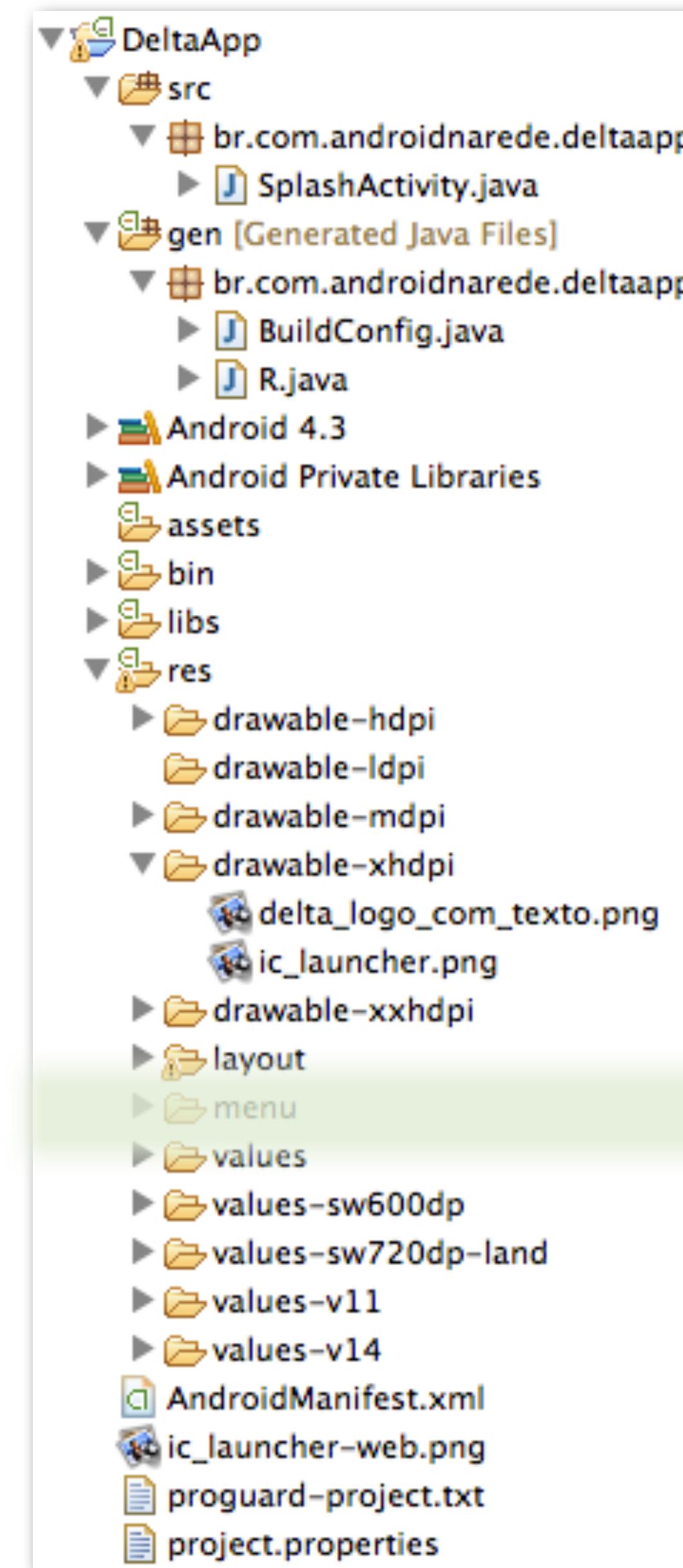
Estrutura de um projeto Android



contém os layouts das telas da app



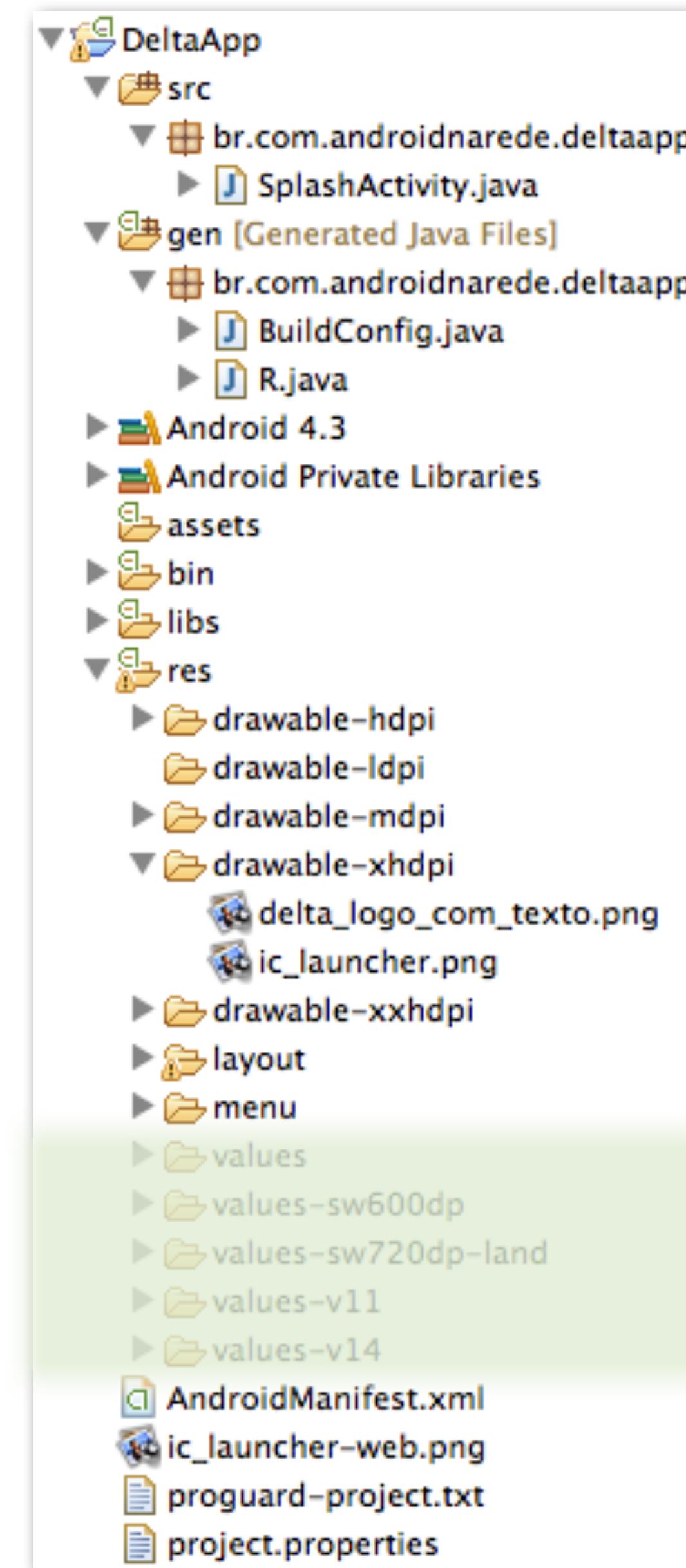
Estrutura de um projeto Android



contém a definição dos menus



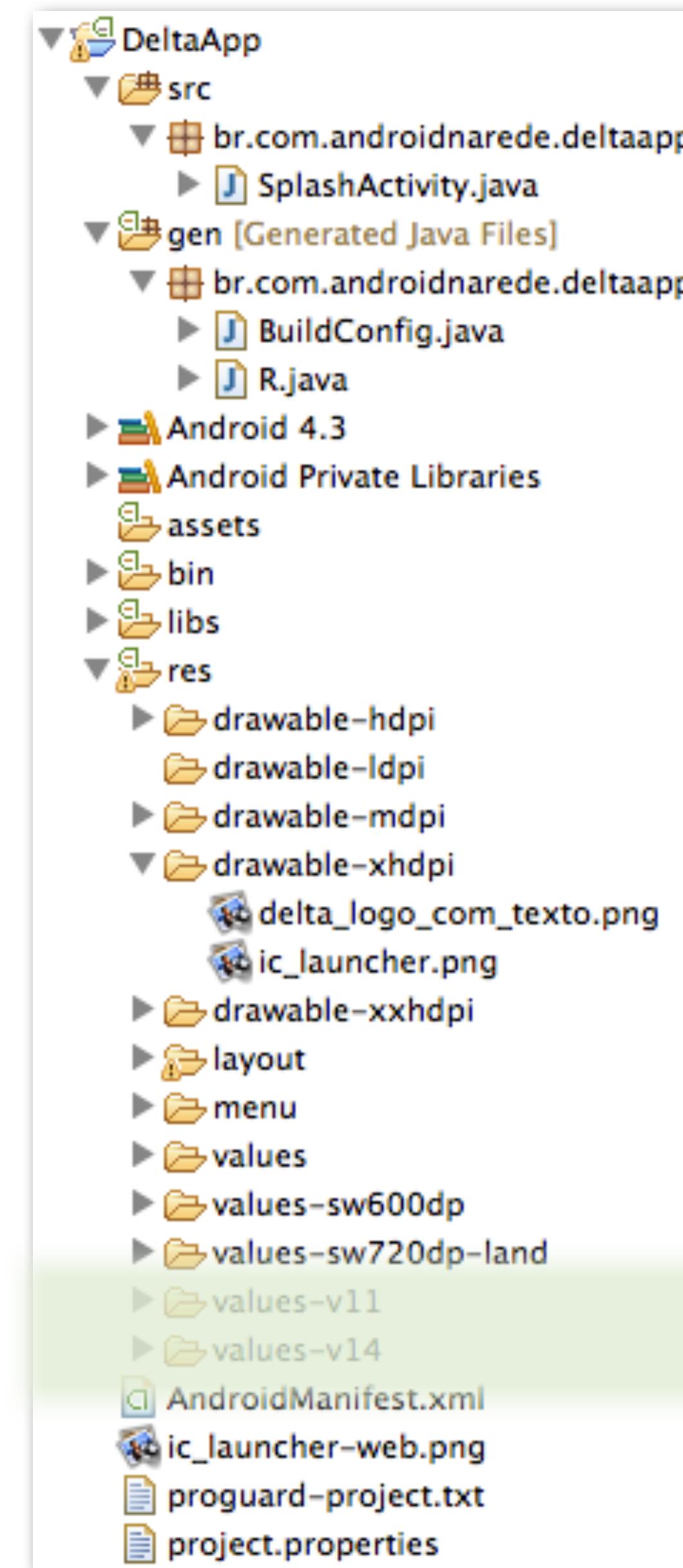
Estrutura de um projeto Android



contém os arquivos para
internacionalização (L10N)



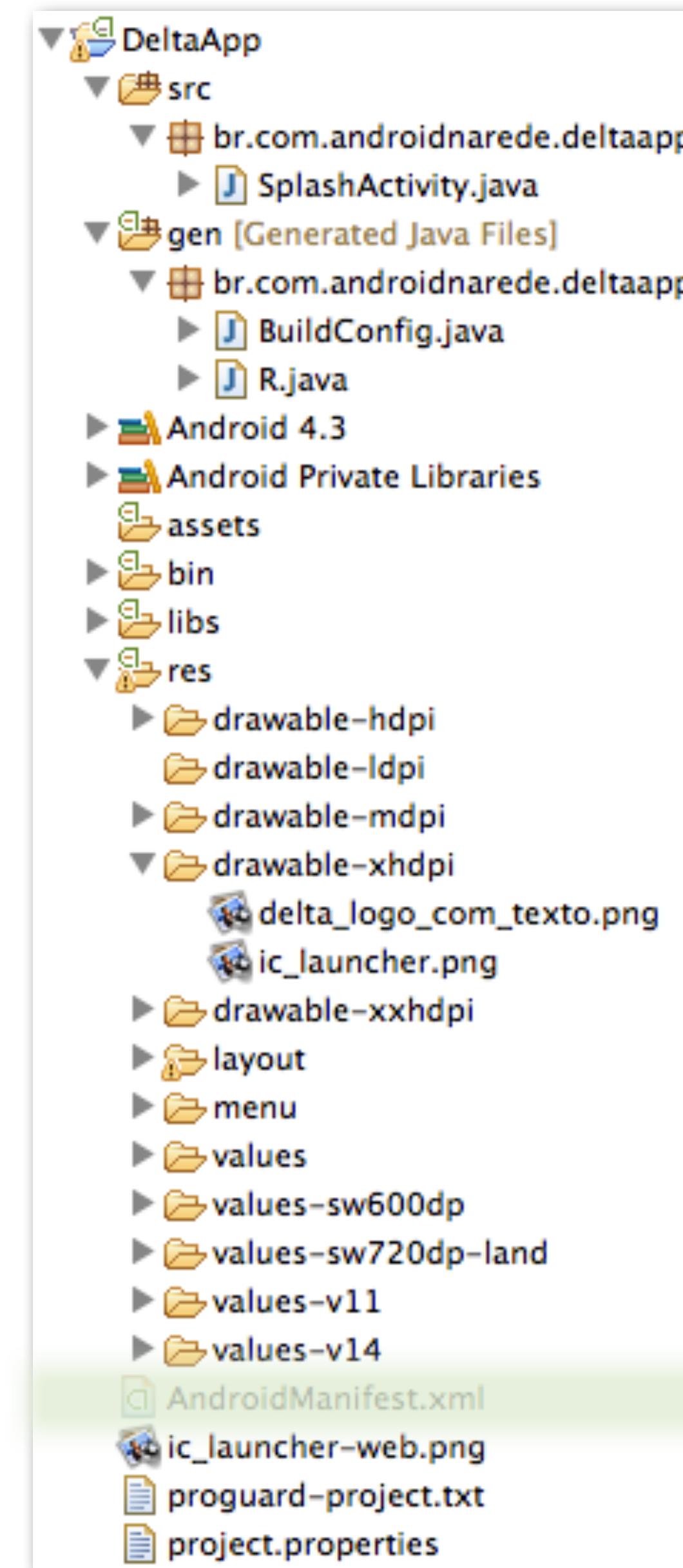
Estrutura de um projeto Android



recursos alternativos quando o dispositivo tiver a API 11 ou 14



Estrutura de um projeto Android



arquivo de manifesto.



Estrutura de um projeto Android

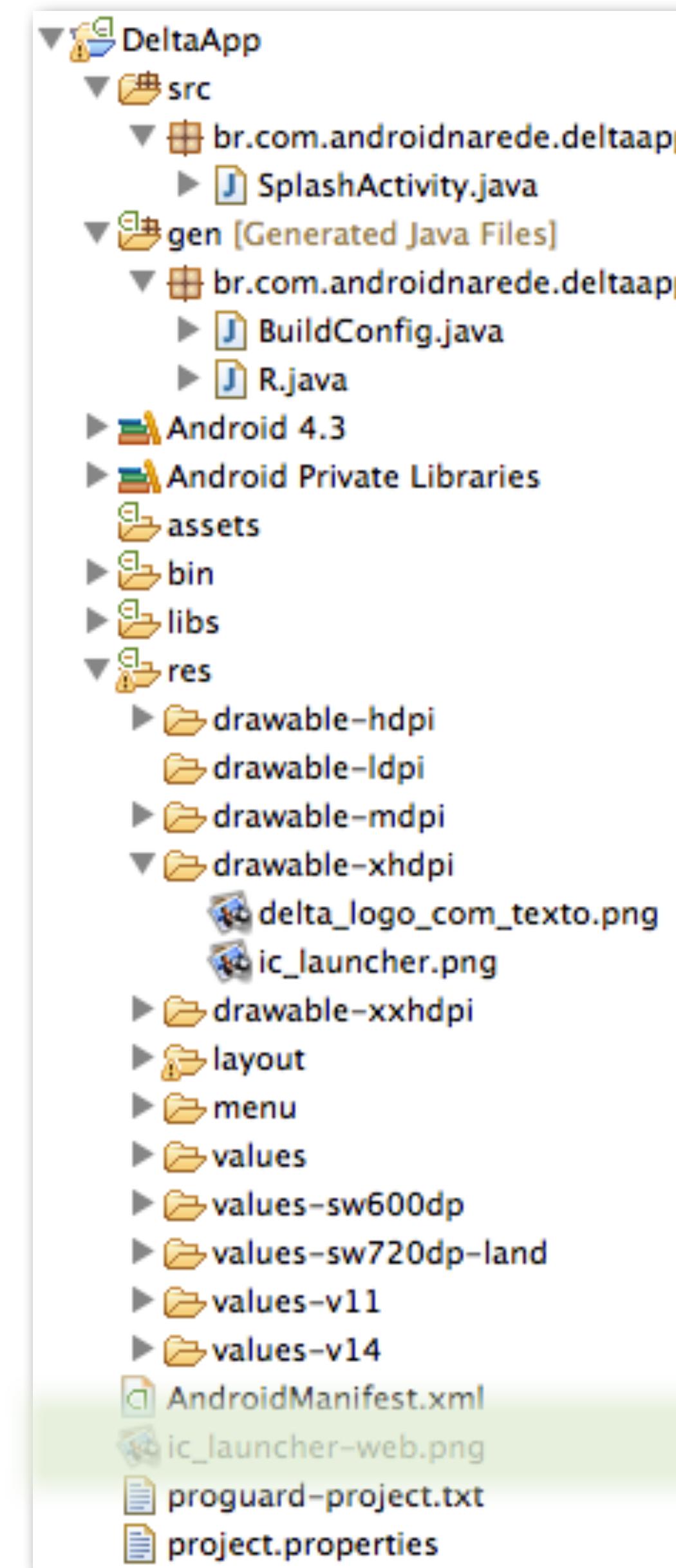
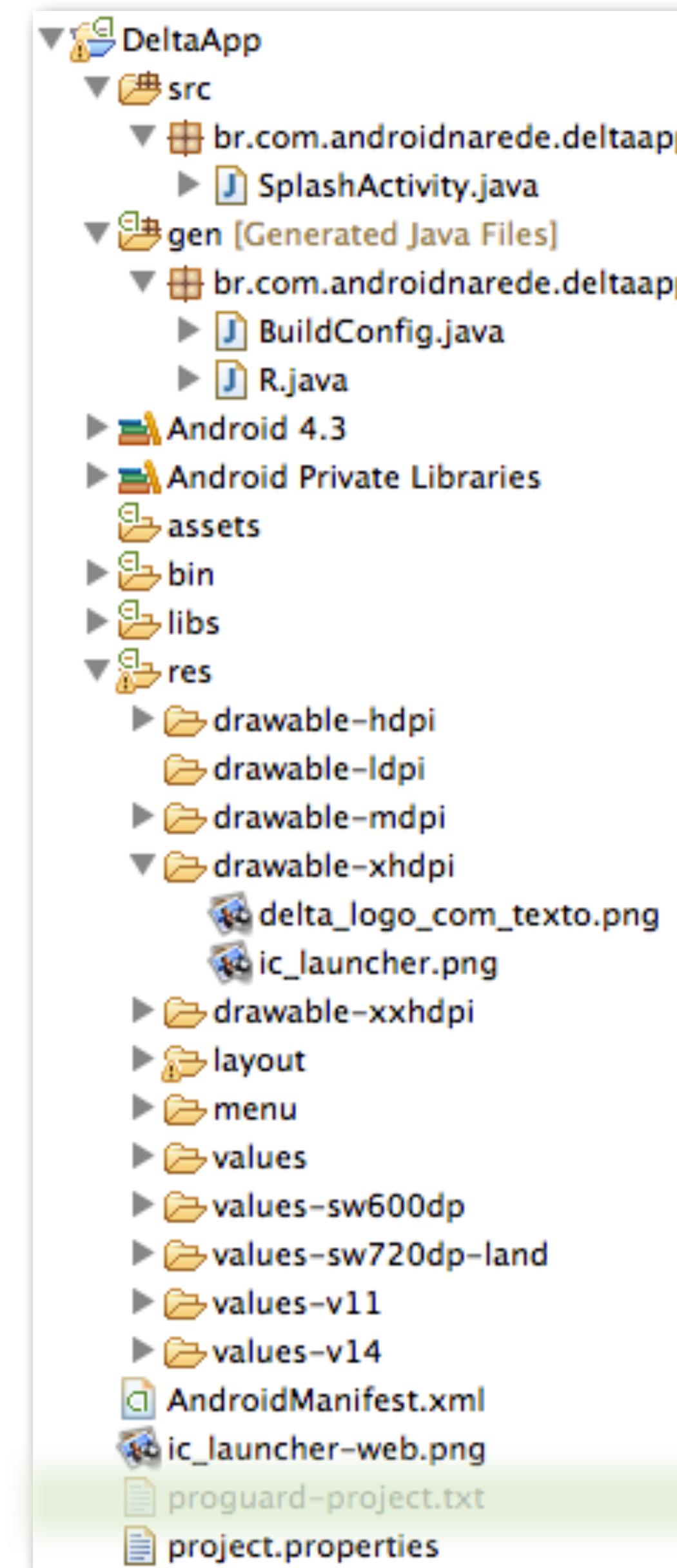


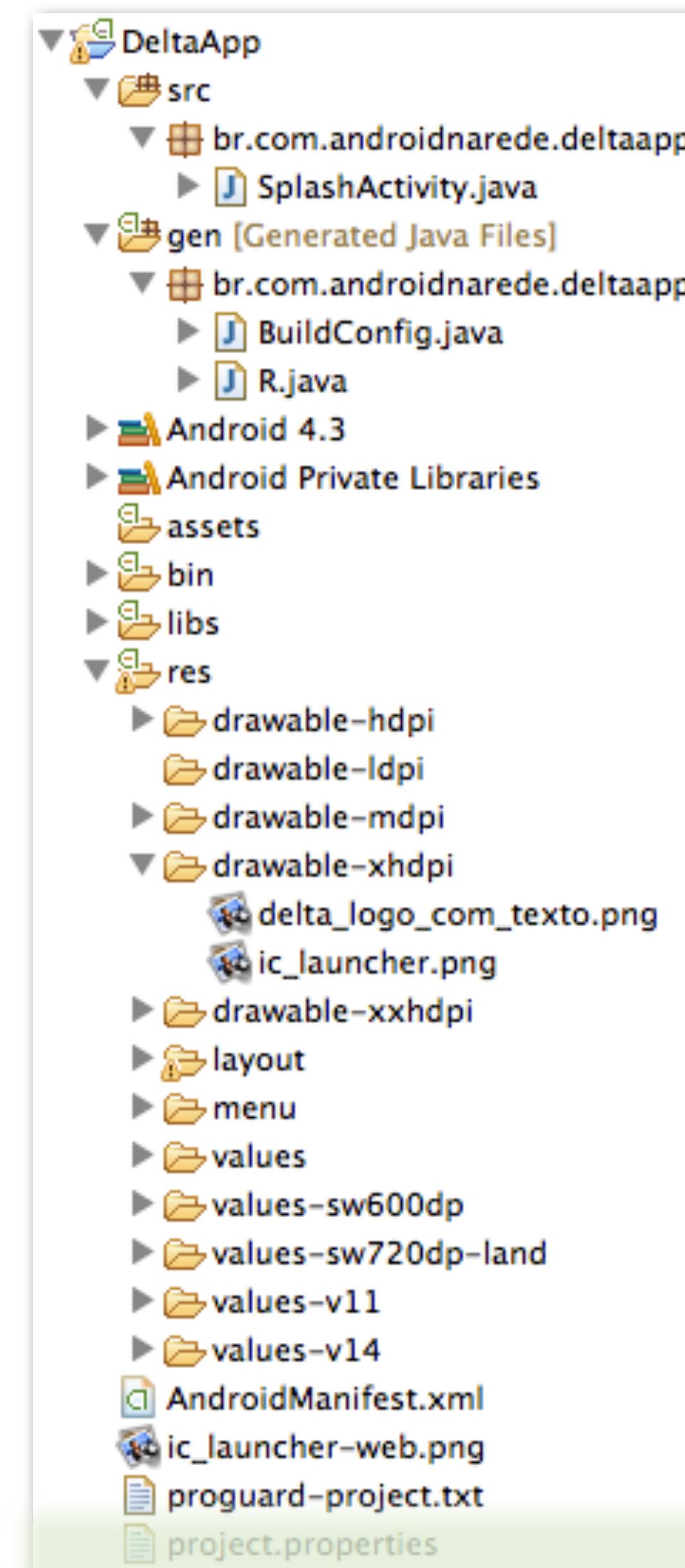
imagem 512x512 para publicação no Google Play Store



Estrutura de um projeto Android



arquivo de configuração do ProGuard



arquivo propriedades do ant



Principais artefatos gerados: SplashActivity.java

```
// package & imports omitidos para brevidade
public class SplashActivity extends Activity {

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

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

Código mínimo da
Activity gerada
durante a criação do
projeto
MeusPresentesApp.



Principais artefatos gerados: SplashActivity.java

```
// package & imports omitidos para brevidade
public class SplashActivity extends Activity {

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

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

Toda Activity deve estender android.app.Activity ou uma de suas subclasses.



Principais artefatos gerados: SplashActivity.java

```
// package & imports omitidos para brevidade
public class SplashActivity extends Activity {

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

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

o método `onCreate()` é chamado no momento da criação da Activity, durante seu ciclo de vida.



Principais artefatos gerados: SplashActivity.java

```
// package & imports omitidos para brevidade
public class SplashActivity extends Activity {

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

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

chamada ao método `onCreate()` da Activity mãe. Caso contrário, será lançada a exceção `SuperNotCalledException`



Principais artefatos gerados: SplashActivity.java

```
// package & imports omitidos para brevidade
public class SplashActivity extends Activity {

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

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

carrega a interface gráfica
contida em *res/layout/
activity_splash.xml*



Principais artefatos gerados: SplashActivity.java

```
// package & imports omitidos para brevidade
public class SplashActivity extends Activity {

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

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

O método *onCreateOptionsMenu()* é chamado durante a criação do menu de opções.



Principais artefatos gerados: SplashActivity.java

```
// package & imports omitidos para brevidade
public class SplashActivity extends Activity {

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

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

Informa que, ao retornarmos `true`, o desenvolvedor que estará responsável por implementar a lógica booleana de retorno do método.



Principais artefatos gerados: R.java

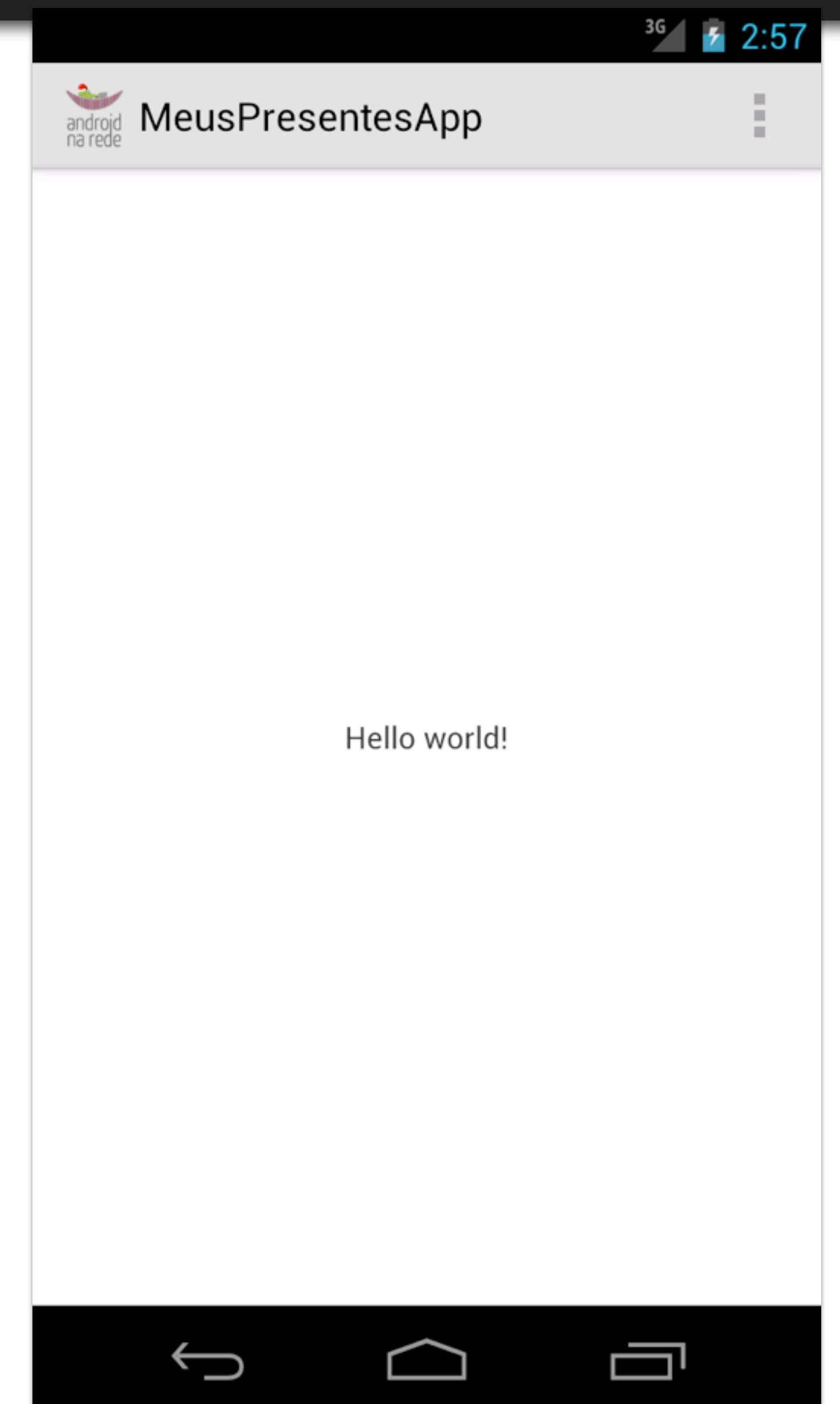
```
public final class R {  
    public static final class attr {  
    }  
    public static final class drawable {  
        public static final int ic_launcher=0x7f020000;  
    }  
    public static final class id {  
        public static final int menu_settings=0x7f070000;  
    }  
    public static final class layout {  
        public static final int activity_splash=0x7f030000;  
    }  
    public static final class menu {  
        public static final int activity_splash=0x7f060000;  
    }  
    public static final class string {  
        public static final int app_name=0x7f040000;  
        public static final int hello_world=0x7f040001;  
        public static final int menu_settings=0x7f040002;  
    }  
    public static final class style {  
        public static final int AppBaseTheme=0x7f050000;  
        public static final int AppTheme=0x7f050001;  
    }  
}
```

Classe gerenciadora de recursos (imagens, layout, menus, etc), abstraindo a localização do artefato, sendo sua identificação por ID.



Principais artefatos gerados: /res/layout/activity_splash.xml

```
<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"  
    tools:context=".SplashActivity" >  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_centerHorizontal="true"  
        android:layout_centerVertical="true"  
        android:text="@string/hello_world" />  
</RelativeLayout>
```

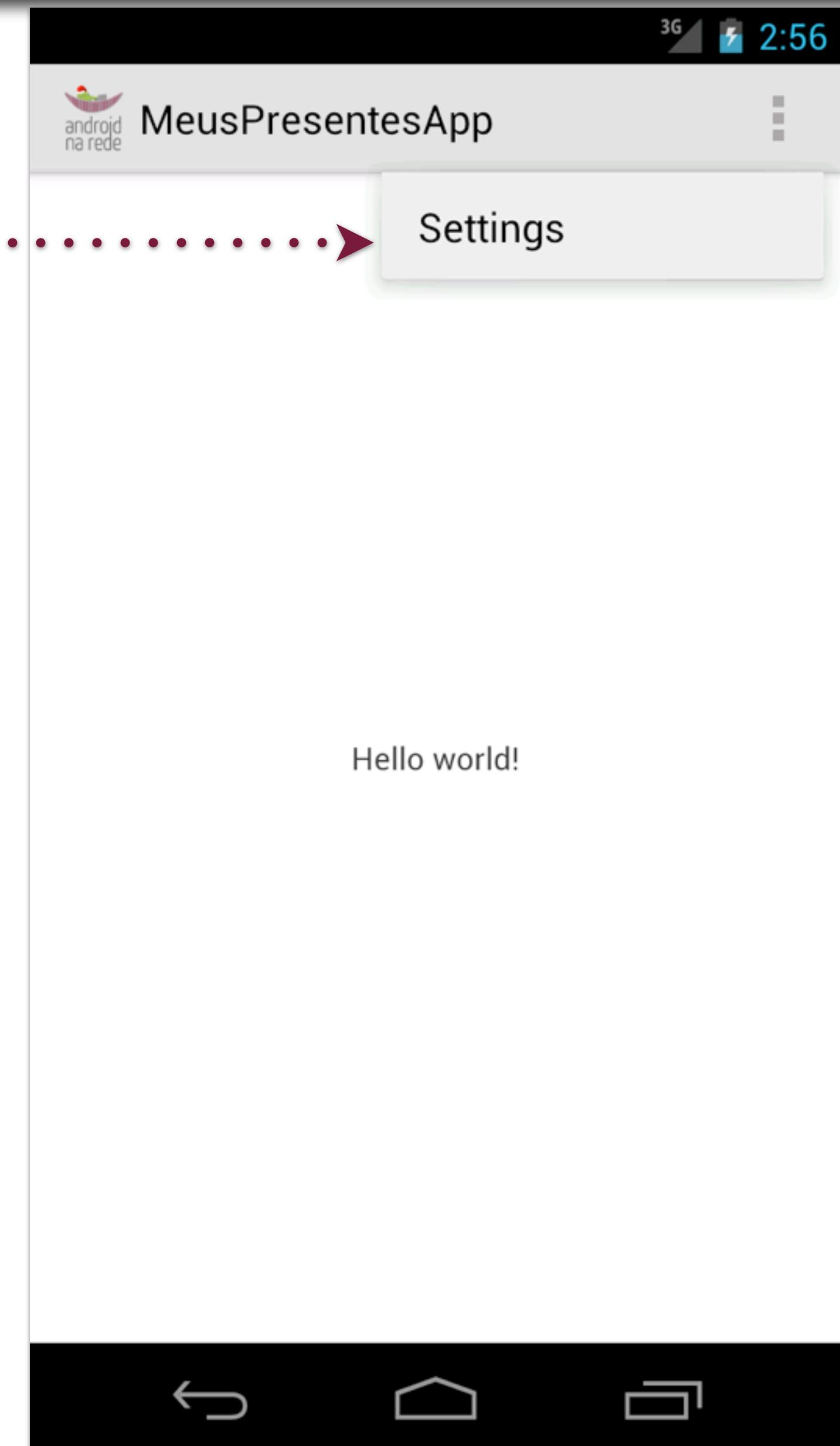


Arquivo de layout
gerado para ser
exibido pela
Activity
SplashActivity.java.



Principais artefatos gerados: /res/menu/activity_splash.xml

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

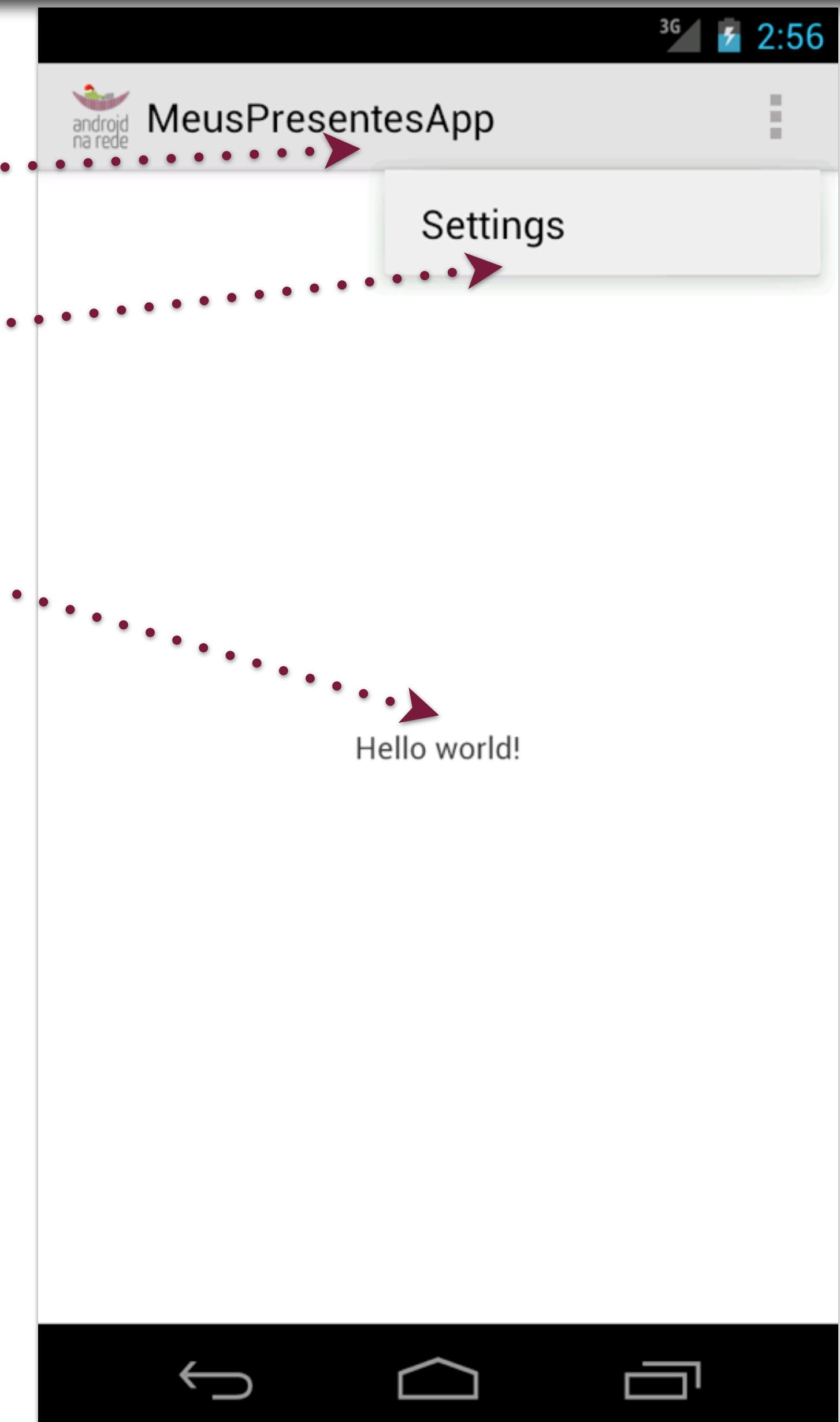


Arquivo de menu
gerado para ser
carregado pela Activity
SplashActivity.java.



Principais artefatos gerados: /res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">MeusPresentesApp</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
</resources>
```

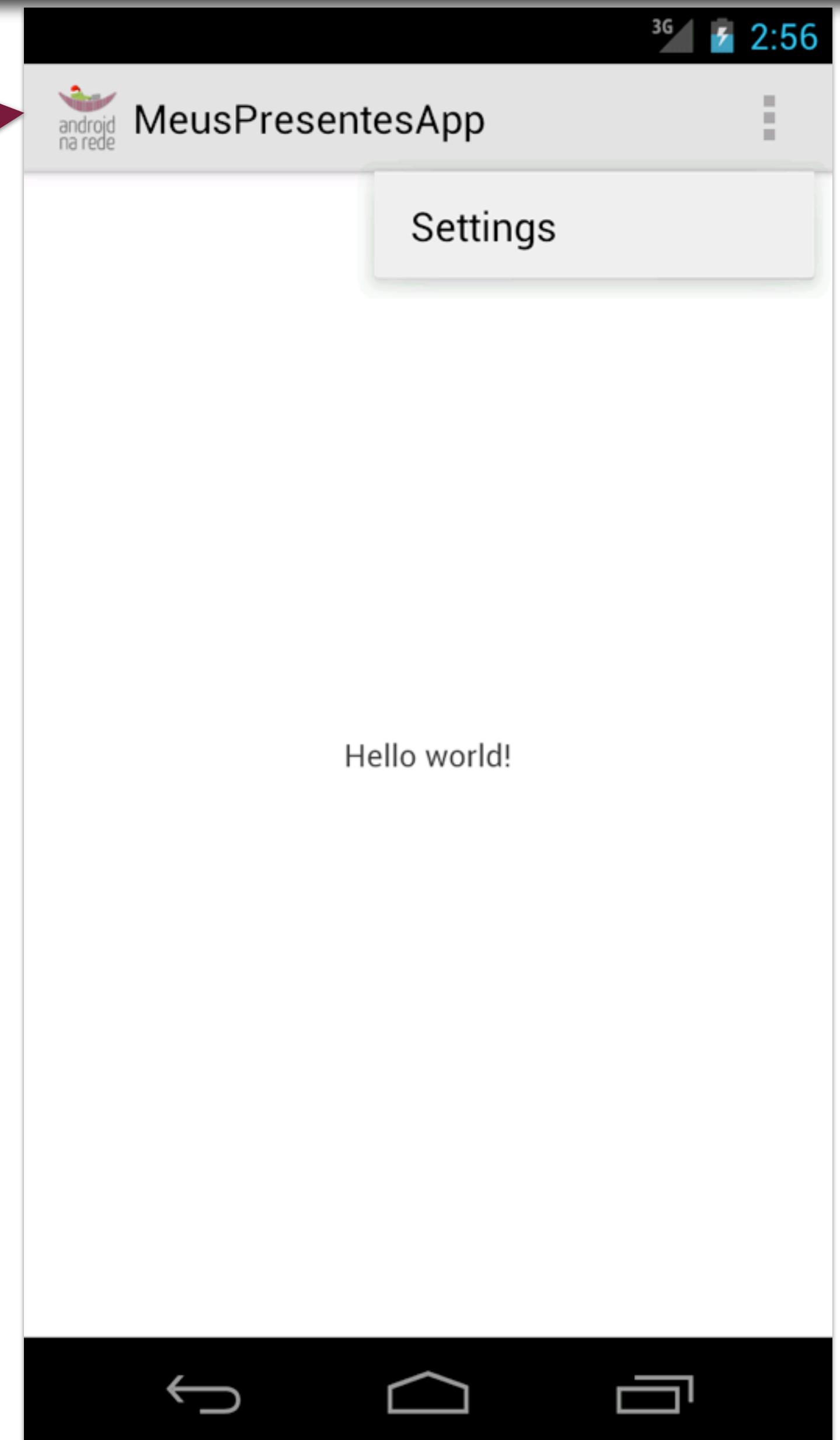


Arquivo contendo **textos estáticos**, específicos quando sua app estiver disponível para mais de um idioma.



Principais artefatos gerados: /res/values/styles.xml

```
<resources>
    <!--
        Base application theme, dependent on API level. This theme is replaced
        by AppBaseTheme from res/values-vXX/styles.xml on newer devices...
    -->
    <style name="AppBaseTheme" parent="android:Theme.Light">      <!--
        Theme customizations available in newer API levels can go in
        res/values-vXX/styles.xml, while customizations related to
        backward-compatibility can go here.
    -->
</style>
<!-- Application theme. -->
<style name="AppTheme" parent="AppBaseTheme">      <!-- All customizations that are NOT
specific to a particular API-level can go here. -->
</style>
</resources>
```



Arquivo de estilos,
referente à aparência e
arte da sua app.



Principais artefatos gerados: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.androidnarede.meuspresentesapp"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="17" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="br.com.androidnarede.meuspresentesapp.SplashActivity"
            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>
```

Arquivo de manifesto,
contendo as configurações da app,
bem como os componentes utilizados,
permissões, etc.

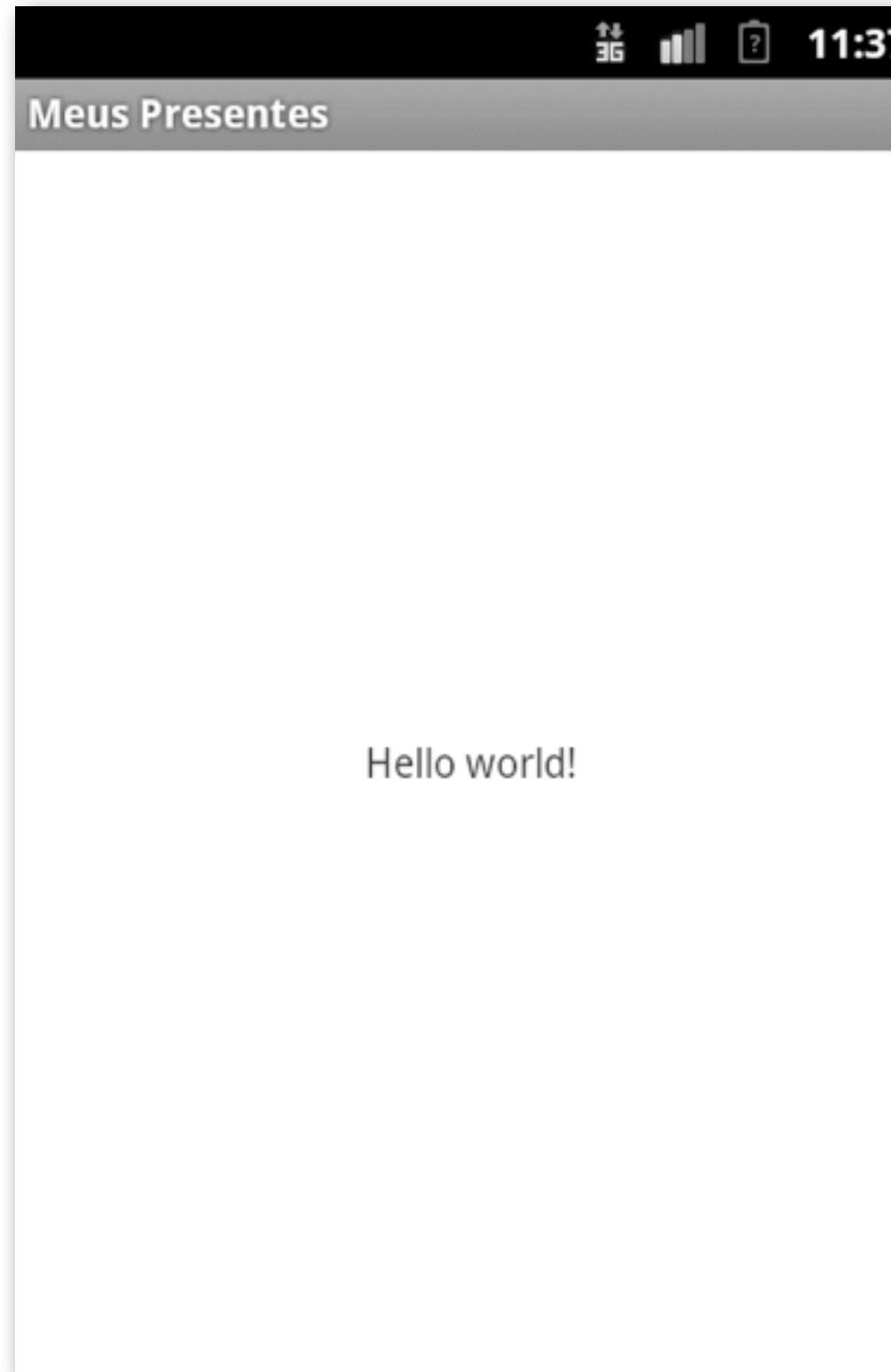


Executando a app

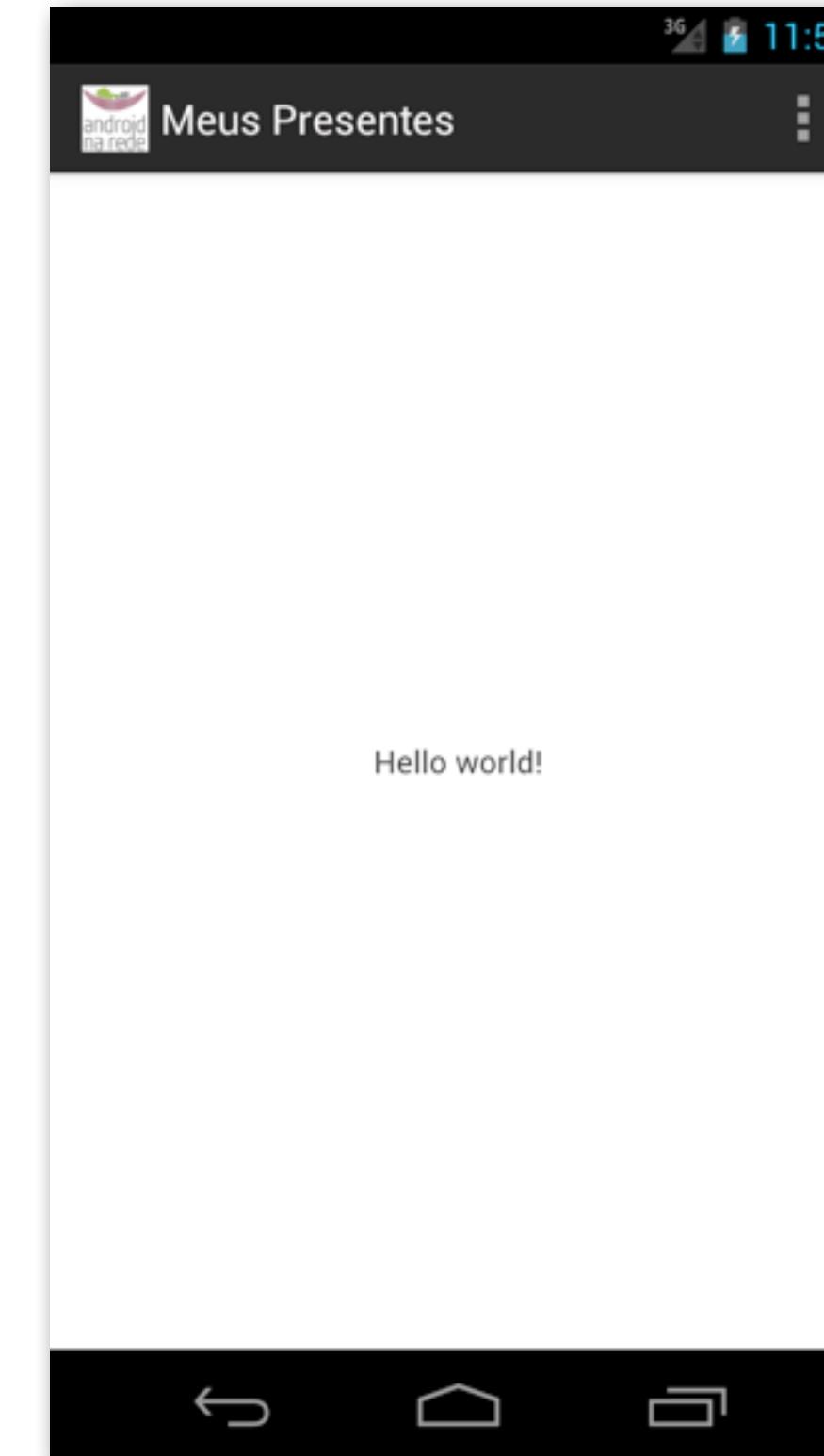
No Emulador

1. Para executar sua app no emulador, você deve selecionar o projeto > *Run* > *Run As* > *Android Application*.

AVD para Gingerbread



AVD para Jelly Bean



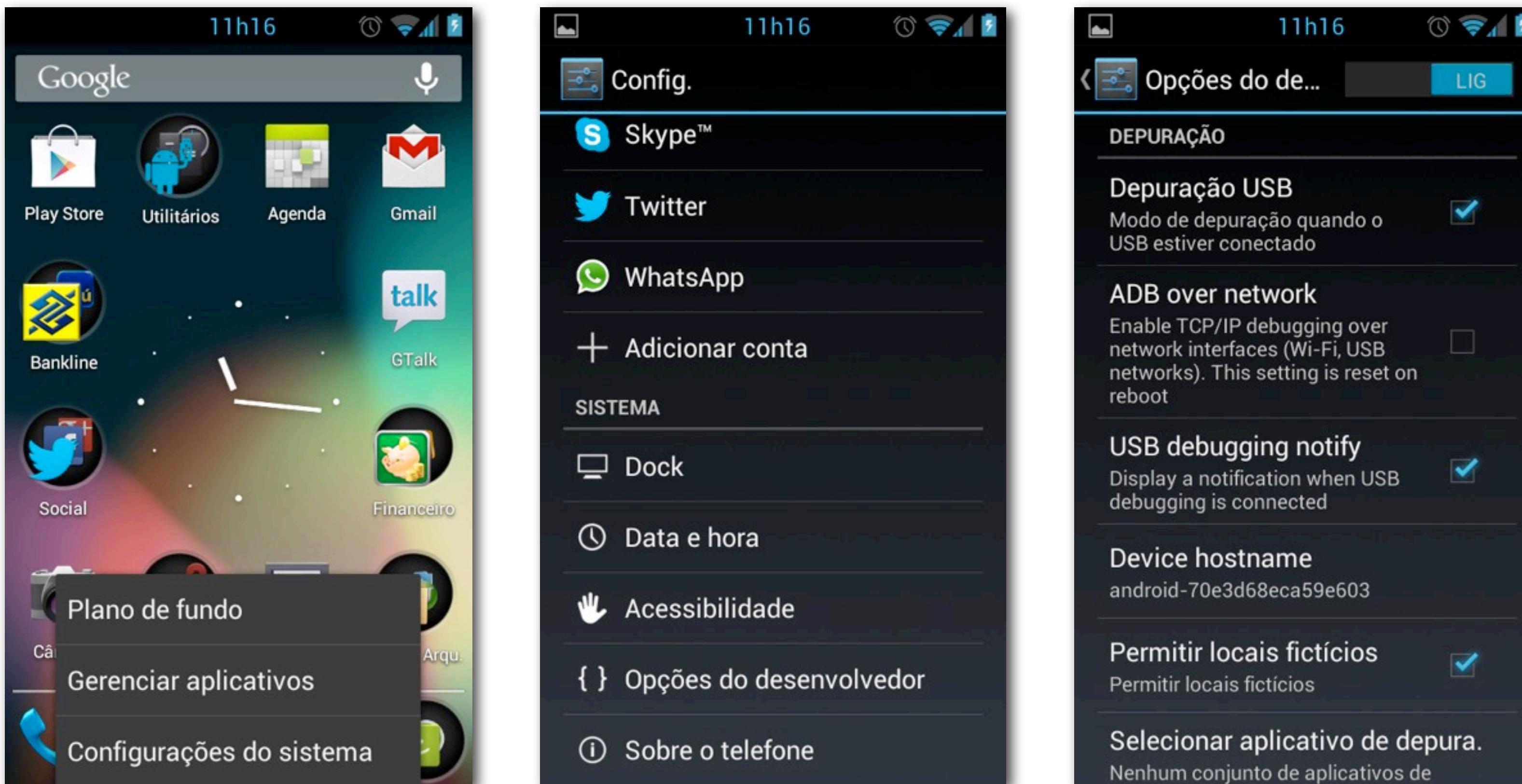
2. Se existir um AVD compatível com as configurações do seu projeto, ele será utilizado e o emulador será iniciado com a sua app sendo executada.



Executando a app

No Smartphone/Tablet

1. Primeiramente, no seu dispositivo real, você deve ir em *Configurações do sistema > Opções do Desenvolvedor* (ou *Aplicativos > Desenvolvimento* caso seja *Gingerbread*) e deixe marcado *Depuração USB (USB Debugging)*.





Executando a app

3. Agora você irá comunicar o seu dispositivo com o adb (Android Debug Bridge), que pertence ao SDK do Android e é utilizado para comunicar com dispositivos remotos (AVDs ou hardwares).

Se estiver utilizando Windows....

Você deve ter instalado os drivers USB de acordo com o seu fabricante. [Neste link](#), está descrito passo-a-passo como fazer isso.

Se estiver utilizando Mac OS X....

Basta conectar seu dispositivo com o computador utilizando o cabo USB.



Executando a app

Se estiver utilizando Ubuntu Linux....

Você terá que adicionar um arquivo de regra udev que contém uma configuração para cada tipo de dispositivo que deseja utilizar. *Neste arquivo, cada fabricante está identificado por um ID único, descrito pela propriedade ATTR{idVendor}. Neste link, você pode conferir o ID para cada fabricante.* Sendo assim, para configurar a detecção do dispositivo, obtenha acesso root e crie este arquivo:

```
$ /etc/udev/rules.d/51-android.rules
```

Use este formato para cada ocorrência de fabricante, no arquivo:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666",
GROUP="plugdev"
```

Agora execute o comando:

```
$ chmod a+r /etc/udev/rules.d/51-android.rules
```



Executando a app

Verificando se seu dispositivo foi identificado para ser executado

Para garantir que seu dispositivo foi identificado, digite o seguinte comando no terminal (para qualquer Sistema Operacional):

```
$ adb devices
```

Executando sua app

Se estiver no Eclipse/ADT Bundle, apenas carregue a app indo em *Run > Run As > Android Application* ou em *Debug > Debug As > Android Application* caso queirar depurar a sua app. Se tudo ocorrer perfeitamente, sua app deverá carregar no seu smartphone/tablet.



Mão na massa!

1. Crie um projeto Android comum, com a versão alvo para rodar no mínimo com o Jelly Bean.
2. Crie dois AVDs: um para o Gingerbread e outro para Jelly Bean. Configure o atributo *android:minSdkVersion* para 10 e execute a app.
3. No projeto criado no exercício anterior, altere o atributo *android:minSdkVersion* para 17, limpe o projeto (*Project > Clean...*) e execute a aplicação novamente. O que você acha que irá acontecer?
4. Crie uma app Android seja executada somente em tablets, isto é, Android 3.0+.
5. Crie uma app Android que tenha uma Activity que será a tela principal de sua app.



Aprendendo a trabalhar com interfaces gráficas em Android





Android Design Guide

<http://d.android.com/design/>

The screenshot shows the 'Iconography' page from the Android Design Guide. The top navigation bar includes links for 'Developers', 'Design' (which is active), 'Develop', and 'Distribute'. A search bar is also present. On the left, a sidebar menu under 'Style' lists various design topics: Devices and Displays, Themes, Touch Feedback, Metrics and Grids, Typography, Color, Iconography (which is selected and highlighted in blue), Writing Style, Patterns, Building Blocks, Downloads, and Videos. The main content area features a large image of a smartphone screen displaying various app icons like Email, Gmail, Play Music, Play Movies, Play Store, Camera, Gallery, and Calendar. Below the image is a definition of an icon: 'An icon is a graphic that takes up a small portion of screen real estate and provides a quick, intuitive representation of an action, a status, or an app.' Further down, the 'Launcher' section is introduced with the text: 'The launcher icon is the visual representation of your app on the Home or All Apps screen. Since the user can change the Home screen's wallpaper, make sure that your launcher icon is clearly visible on any type of background.' At the bottom, there are two smaller images: one showing two versions of the Gmail launcher icon (one red, one white) and another showing a single large version of the Gmail icon.

Iconography

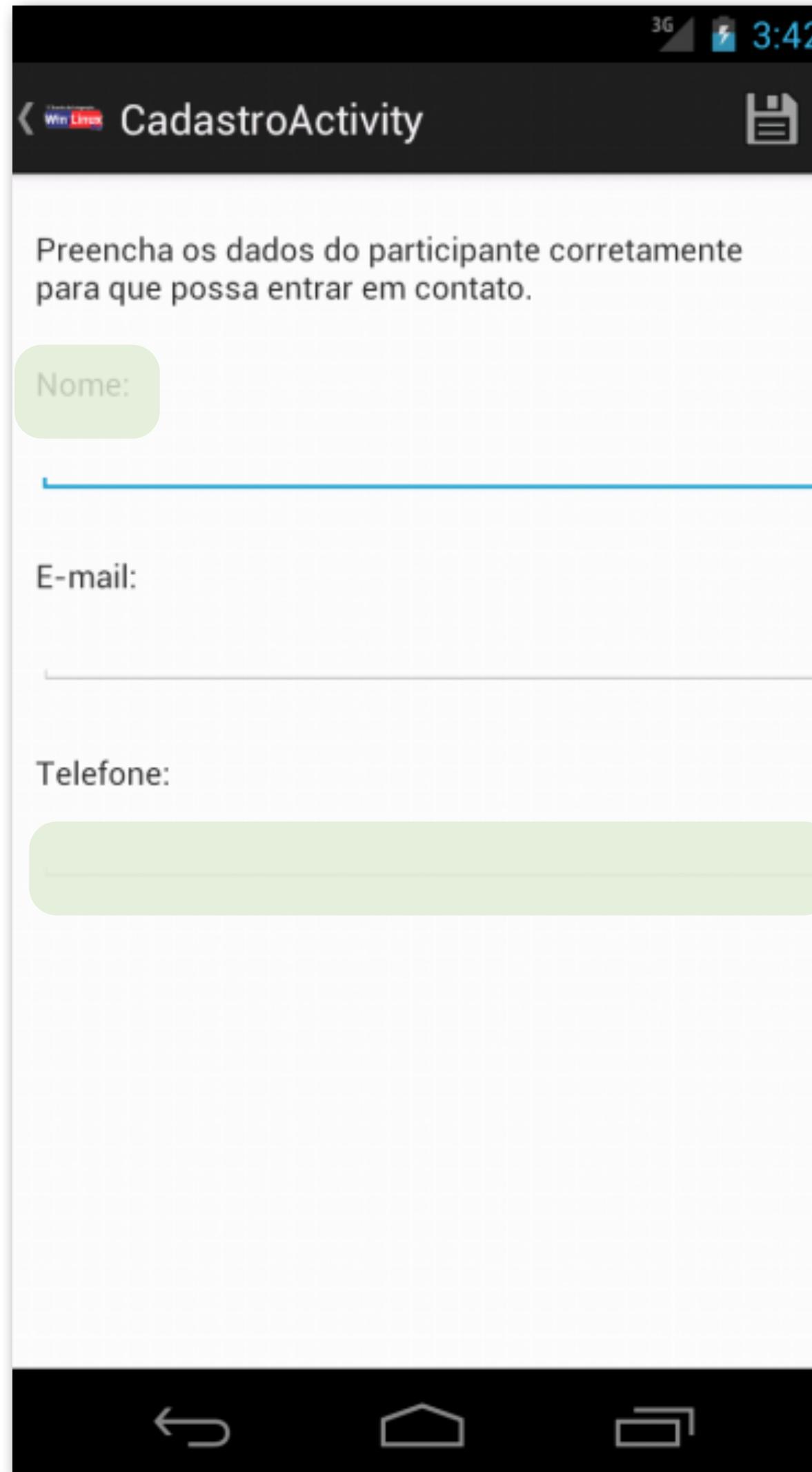
An icon is a graphic that takes up a small portion of screen real estate and provides a quick, intuitive representation of an action, a status, or an app.

Launcher

The launcher icon is the visual representation of your app on the Home or All Apps screen. Since the user can change the Home screen's wallpaper, make sure that your launcher icon is clearly visible on any type of background.



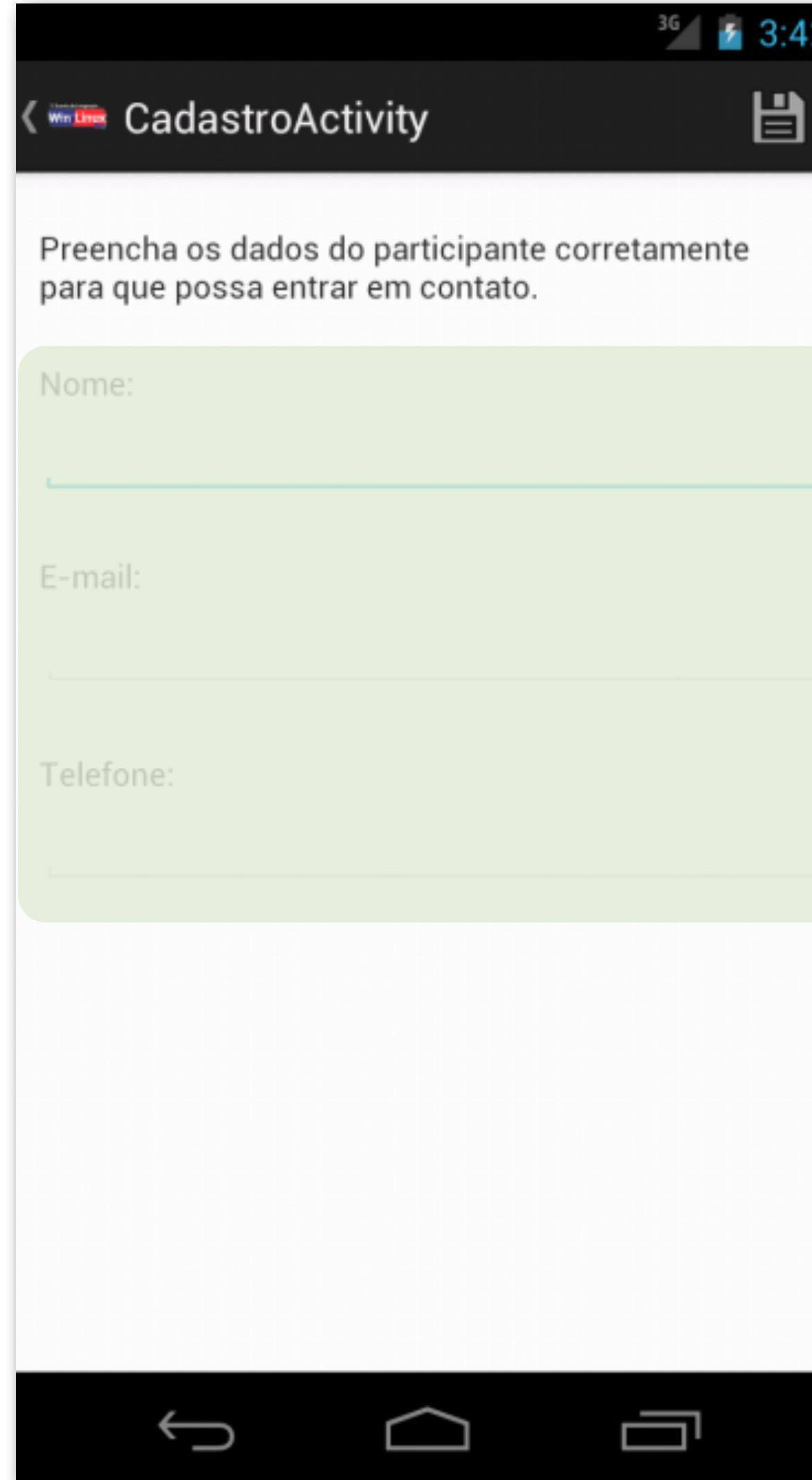
Relembrando: Um View é...



qualquer
componente
gráfico de tela.
Também pode
ser chamado de
Widget.



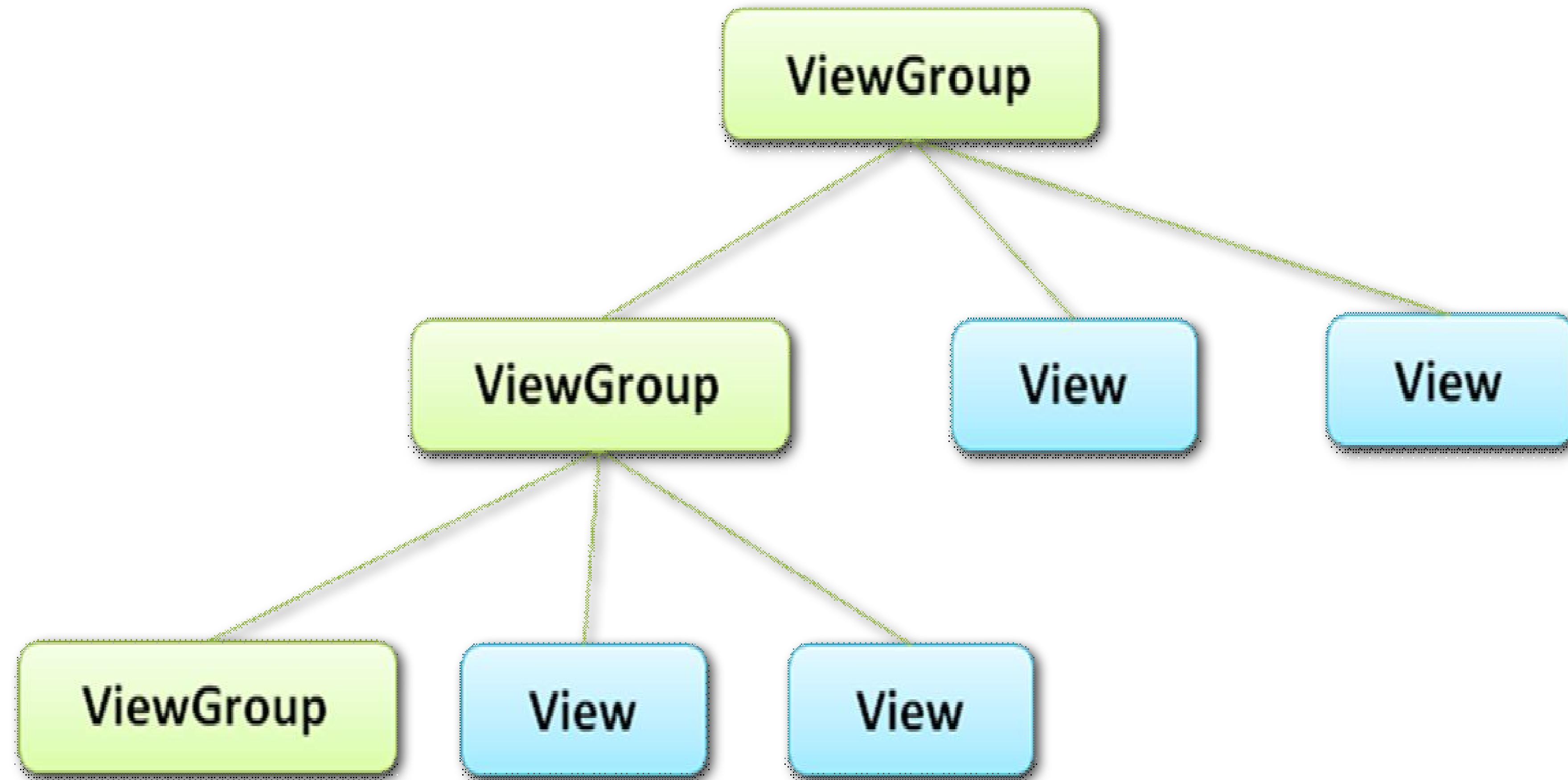
Relembrando: Um ViewGroup é...



**qualquer layout
ou container de
componentes
gráficos.**



A hierarquia de layouts

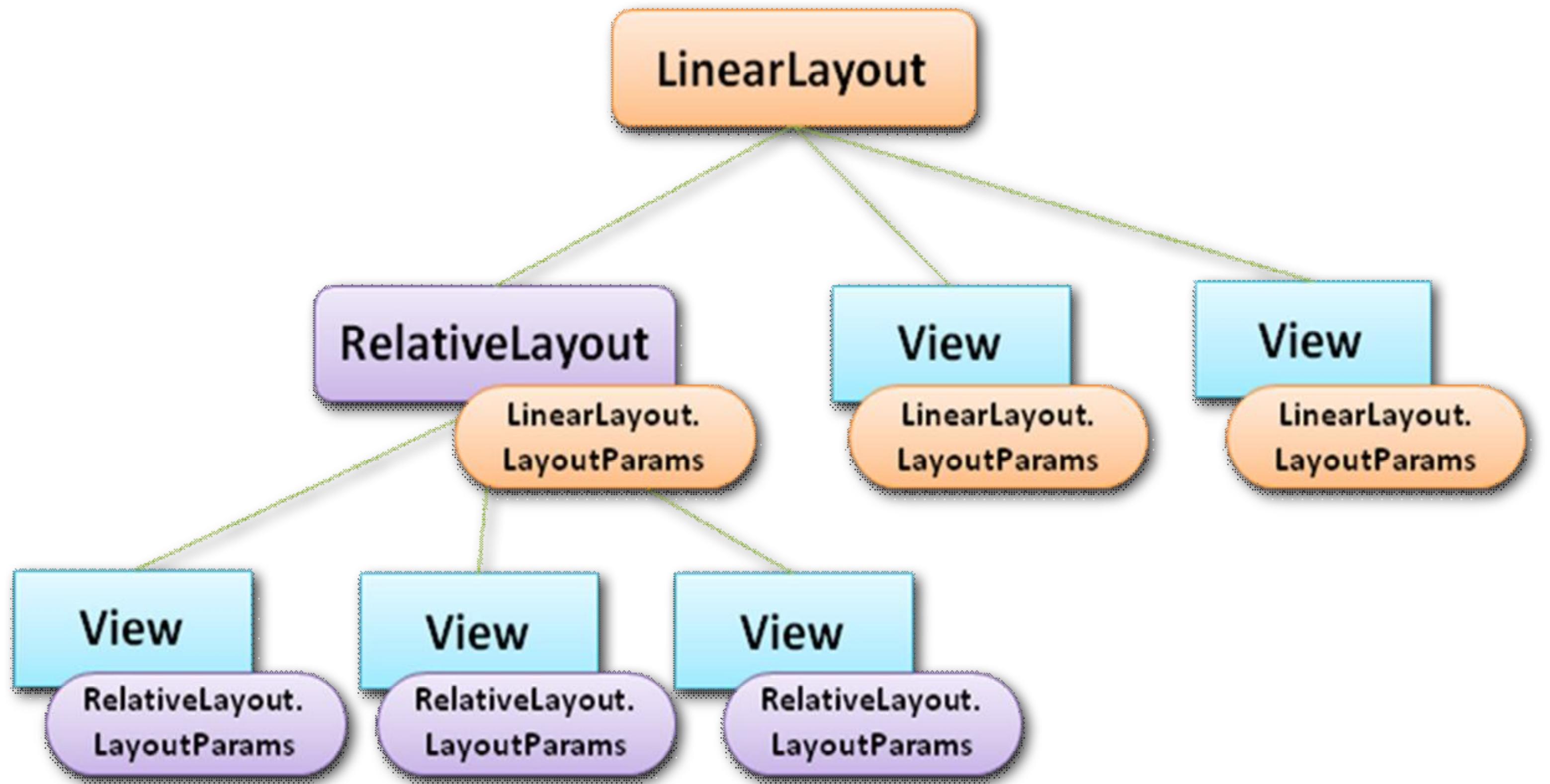




A hierarquia de layouts

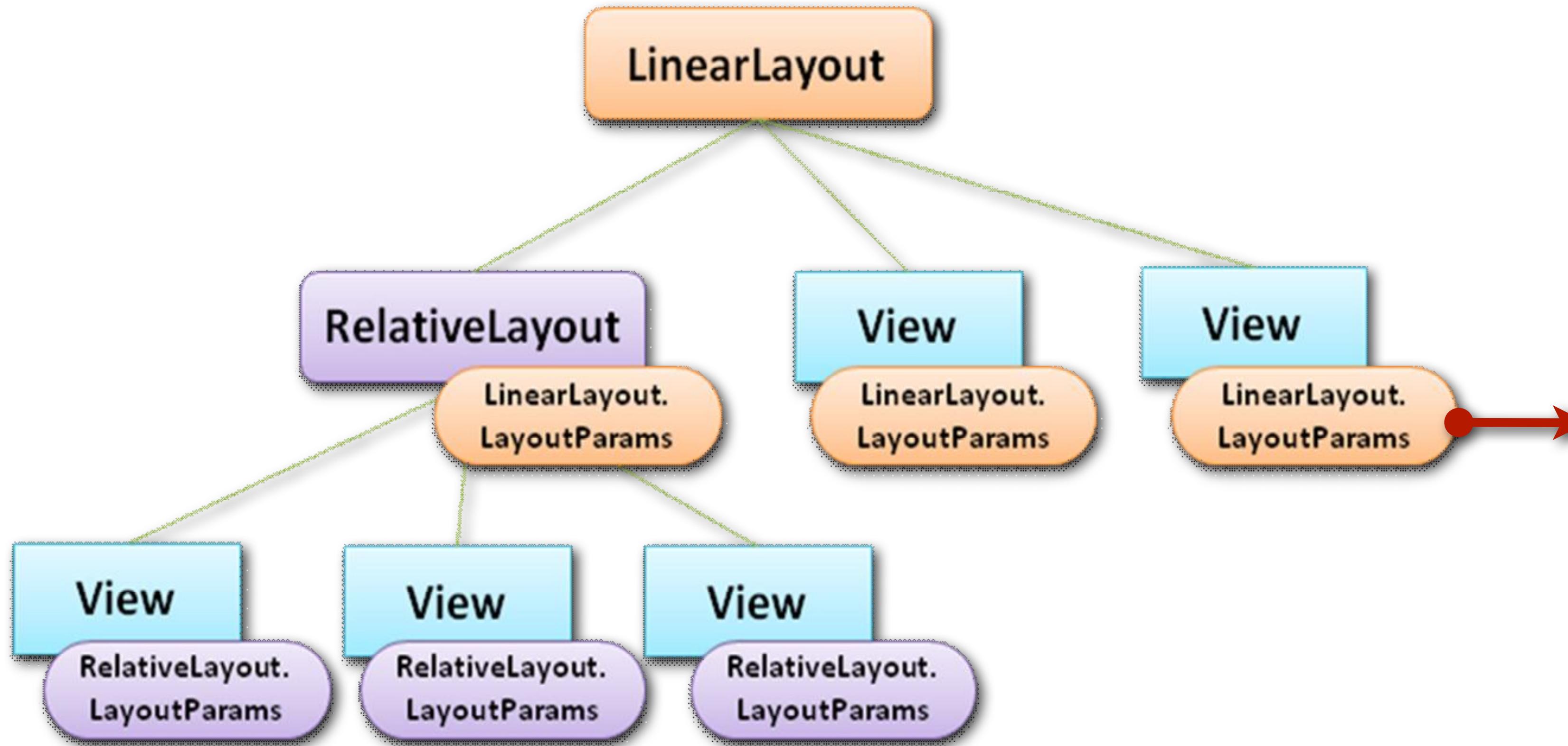


A hierarquia de layouts





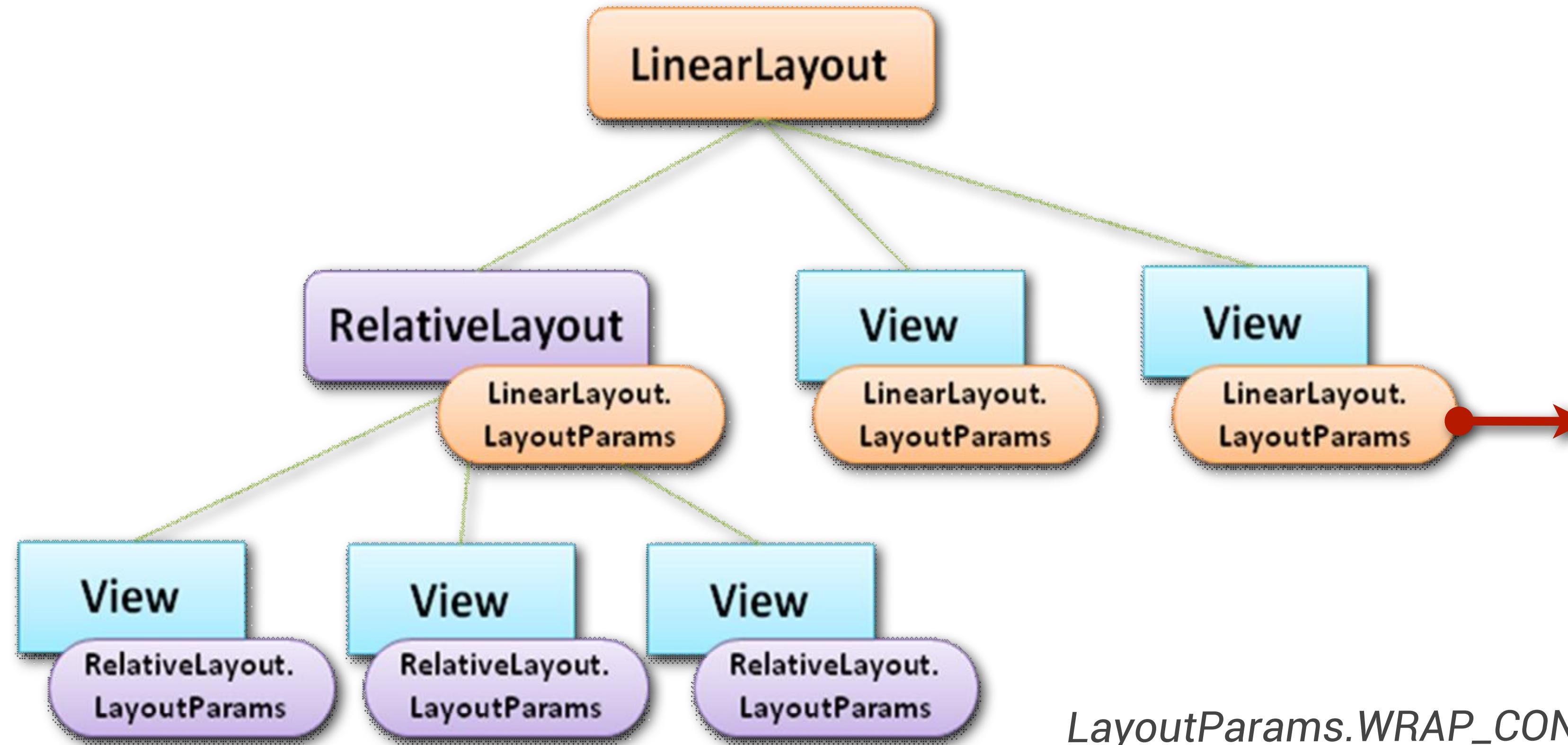
A hierarquia de layouts



parâmetros relacionados
à disposição do
componente
em relação ao seu pai



A hierarquia de layouts

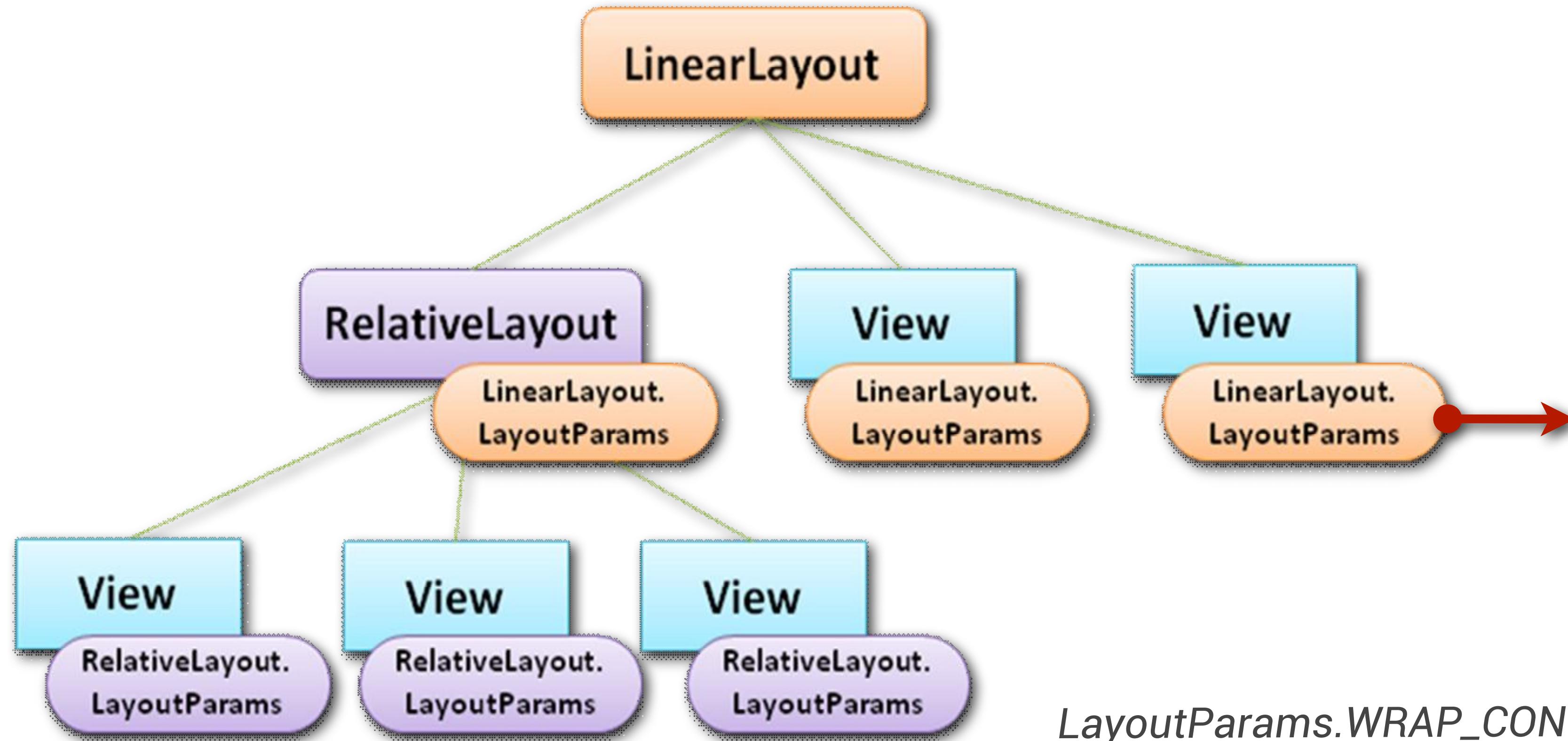


parâmetros relacionados
à disposição do
componente
em relação ao seu pai

LayoutParams.WRAP_CONTENT



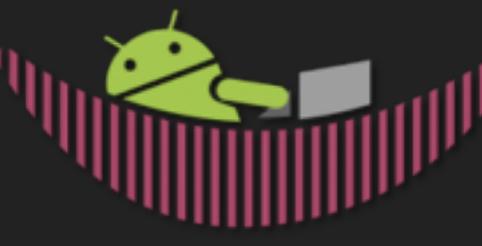
A hierarquia de layouts



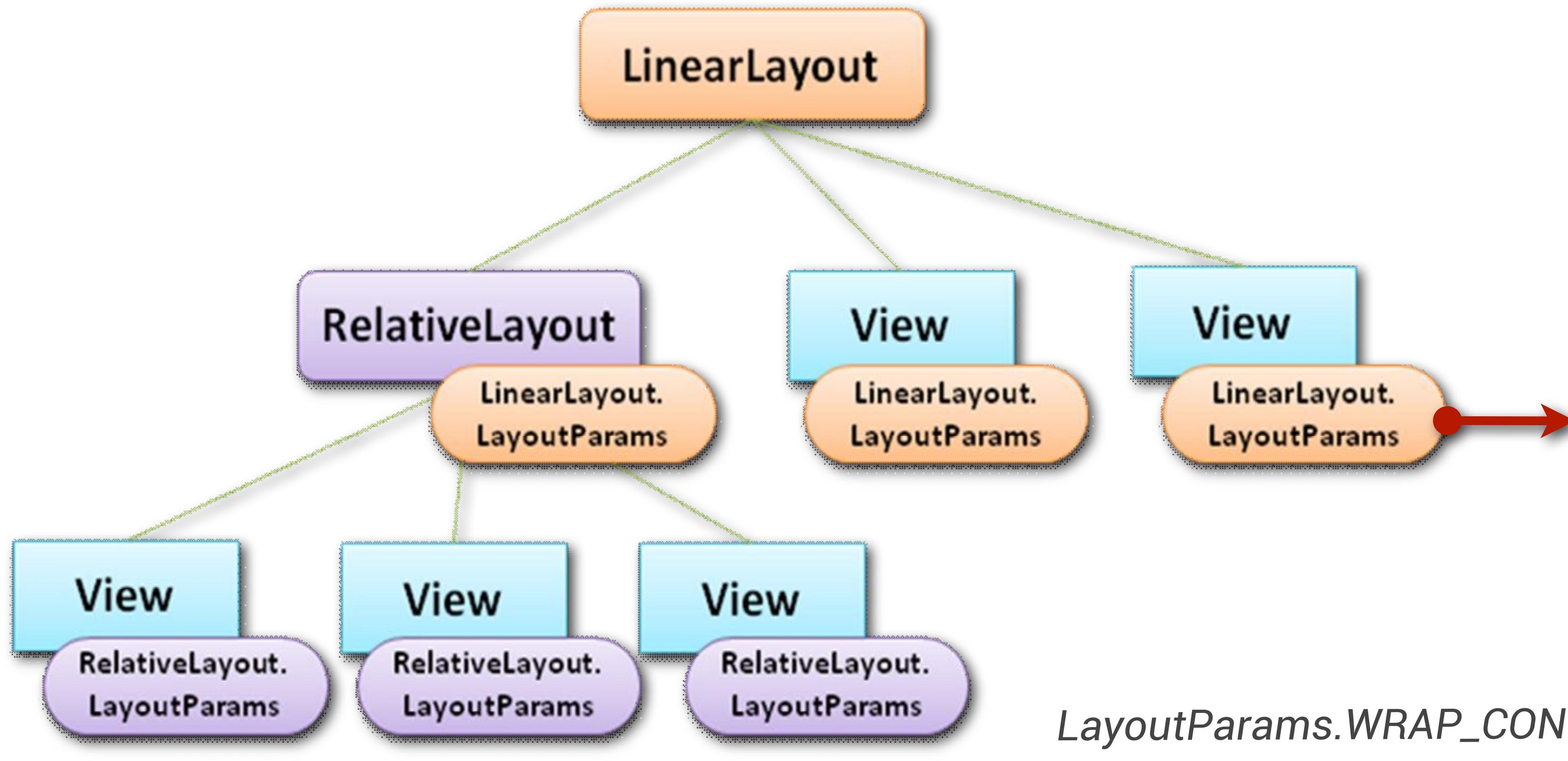
parâmetros relacionados
à disposição do
componente
em relação ao seu pai

`LayoutParams.WRAP_CONTENT`

↓
expande de
acordo
com o conteúdo



A hierarquia de layouts

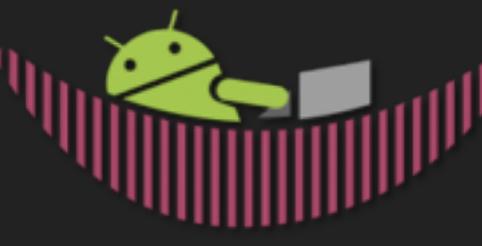


parâmetros relacionados
à disposição do
componente
em relação ao seu pai

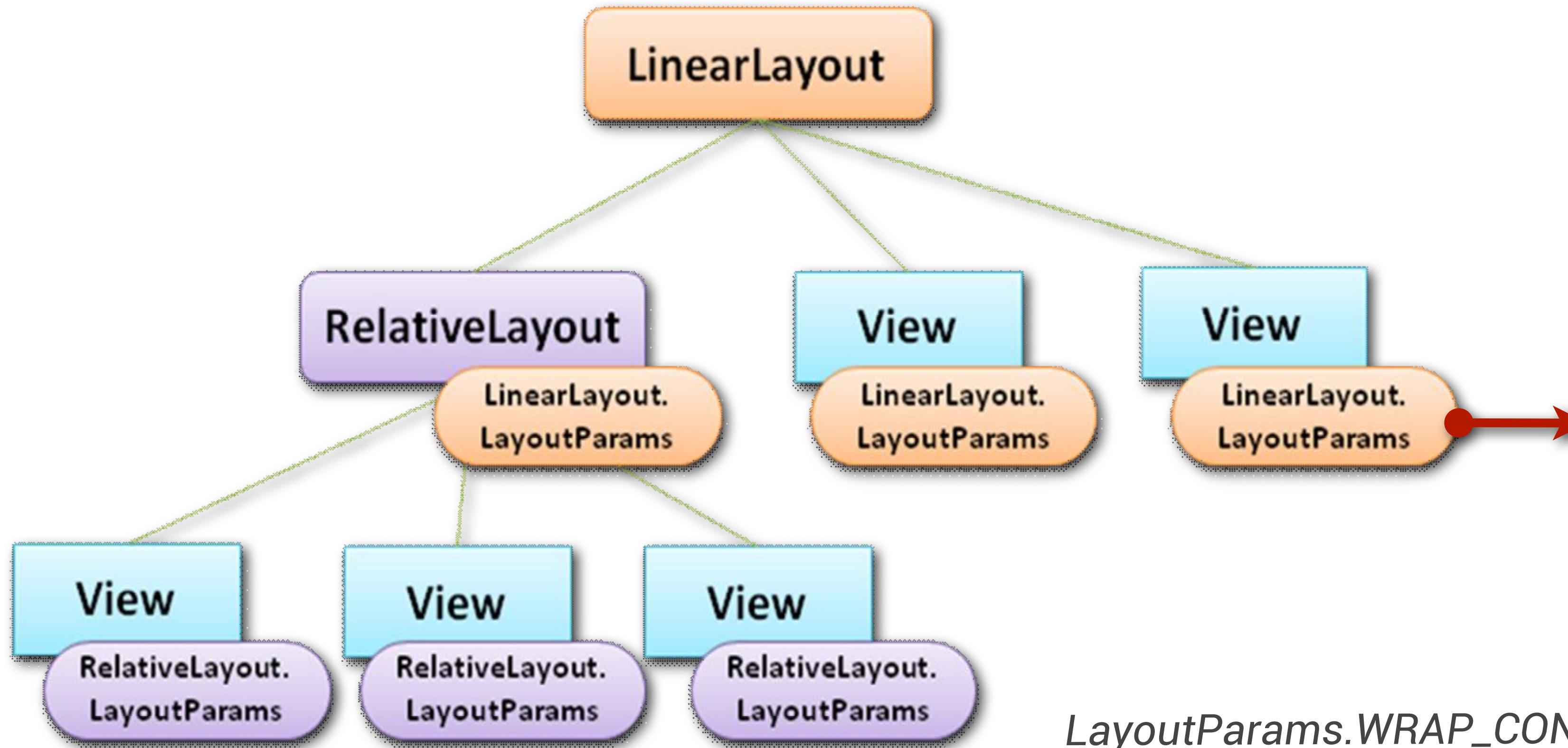
`LayoutParams.WRAP_CONTENT`

`LayoutParams.FILL_PARENT`

↓
expande de
acordo
com o conteúdo



A hierarquia de layouts

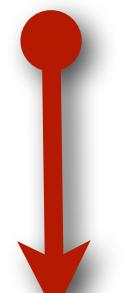


parâmetros relacionados
à disposição do
componente
em relação ao seu pai

`LayoutParams.WRAP_CONTENT`

expande de
acordo
com o conteúdo

`LayoutParams.FILL_PARENT`



expande
preenchendo
o espaço todo do
pai





layout em XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="fill_parent">
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Texto1" />
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Texto2" />
</LinearLayout>
```

Two red arrows point from the bottom towards the opening tags of the first two `<EditText>` elements in the XML code.



layout em XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="fill_parent">
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Texto1" />
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Texto2" />
</LinearLayout>
```

pai



Two red arrows point from the word "pai" to the opening tag of the first `EditText` element and the opening tag of the second `EditText` element.



layout em XML

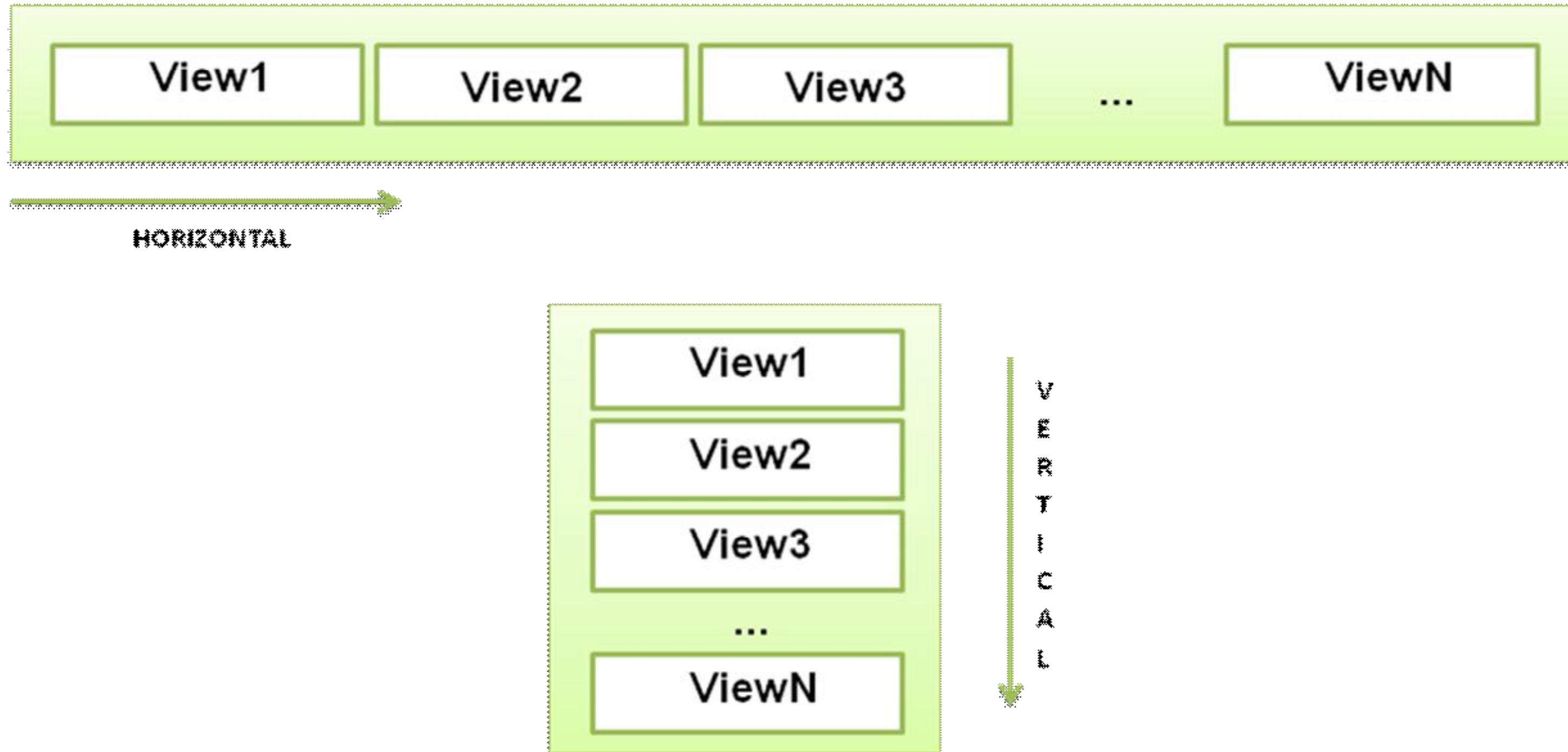
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="fill_parent">
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Texto1" />
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Texto2" />
</LinearLayout>
```

pai

filho



LinearLayout



Dispõe os componentes em uma única direção.

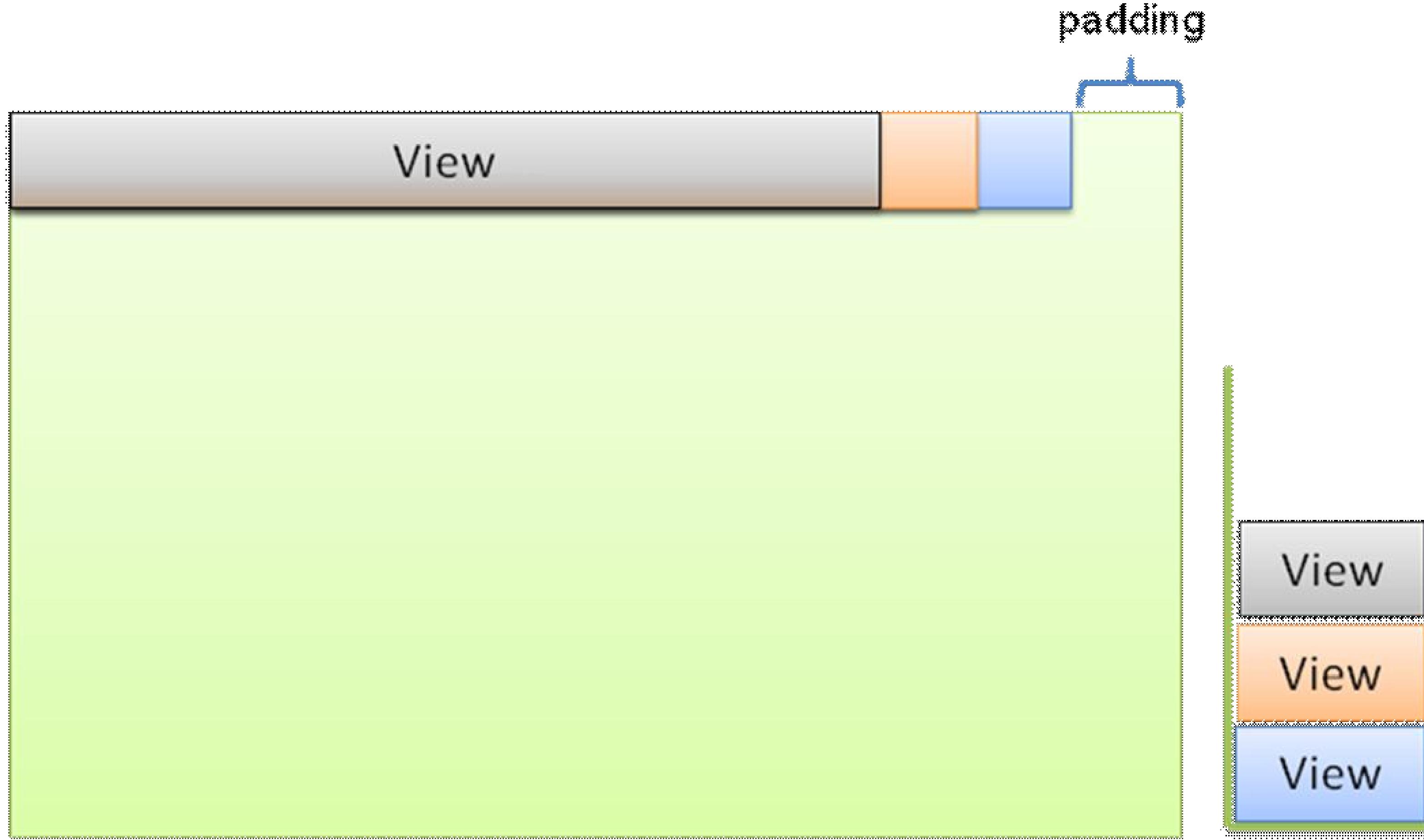


Exemplo de LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="fill_parent">
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Texto1" />
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Texto2" />
</LinearLayout>
```



FrameLayout



Adiciona os componentes em formato de pilha.

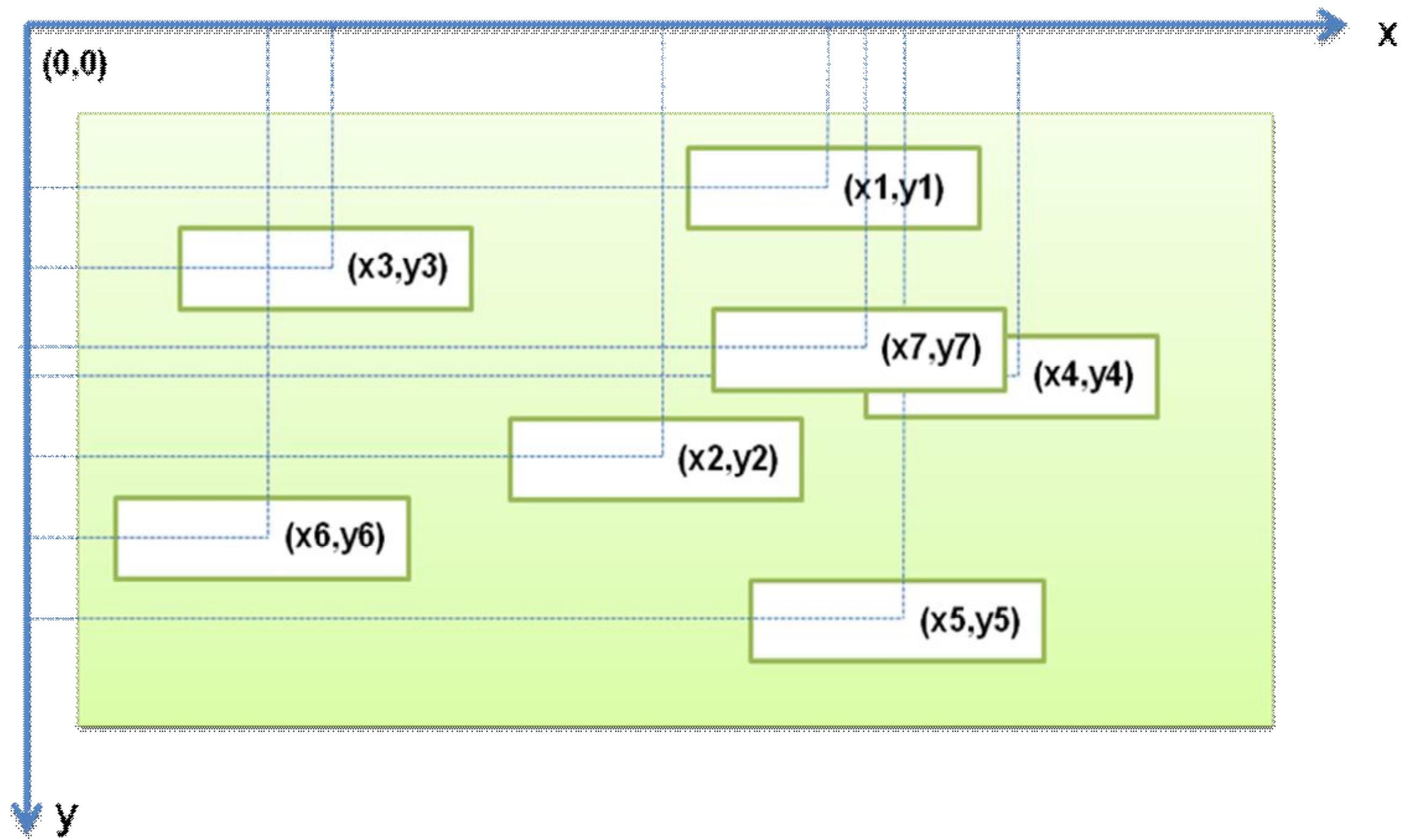


Exemplo de FrameLayout

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <EditText android:layout_width="120px"
        android:layout_height="wrap_content" android:text="Texto1"
        android:layout_weight="1" />
    <EditText android:layout_width="90px"
        android:layout_height="wrap_content" android:text="Texto2"
        android:layout_weight="1" />
    <EditText android:layout_width="60px"
        android:layout_height="wrap_content" android:text="Texto3" />
</FrameLayout>
```



AbsoluteLayout (Depreciado)



Os componentes
são arranjados
num plano
cartesiano, por
meio de posições
fixas (x,y)

Esse layout foi depreciado pela plataforma,
apesar de ainda poder ser utilizado.



Exemplo de AbsoluteLayout

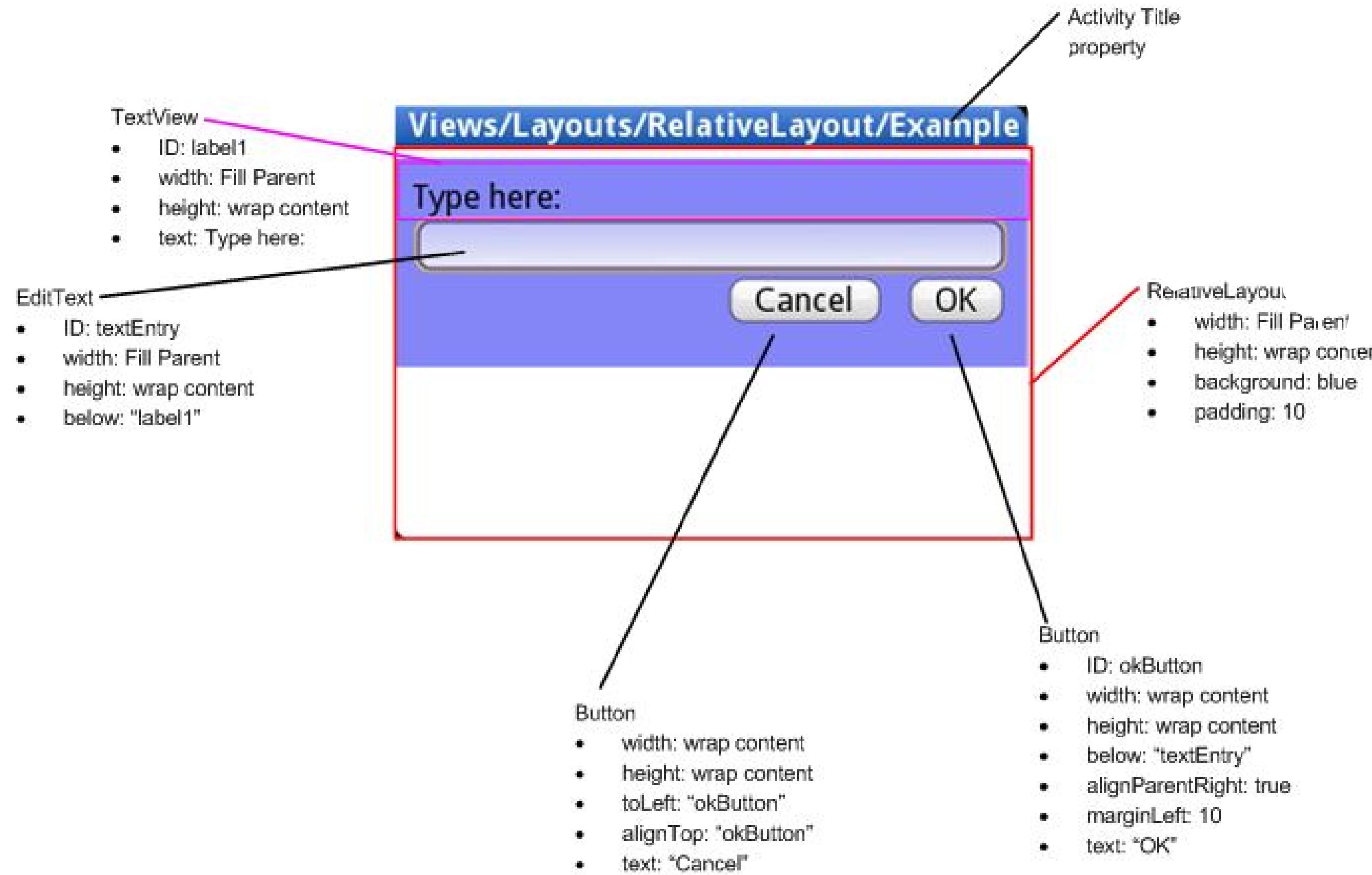
```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="fill_parent">
    <EditText android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Texto1"
        android:layout_x="45px"
        android:layout_y="87px" />

    <EditText android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Texto2"
        android:layout_x="90px"
        android:layout_y="12px" />

    <EditText android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Texto 3"
        android:layout_x="90px"
        android:layout_y="250px" />
</AbsoluteLayout>
```



RelativeLayout



Dispõe os componentes em relação ao seu pai ou à seus irmãos.



Exemplo de RelativeLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="fill_parent">
    <TextView android:id="@+id/tvLogin"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_centerHorizontal="true" android:text="Usuário:" />

    <EditText android:id="@+id/etLogin"
        android:layout_width="250px" android:layout_height="wrap_content"
        android:layout_centerHorizontal="true" android:text="usuario"
        android:layout_below="@id/tvLogin" />

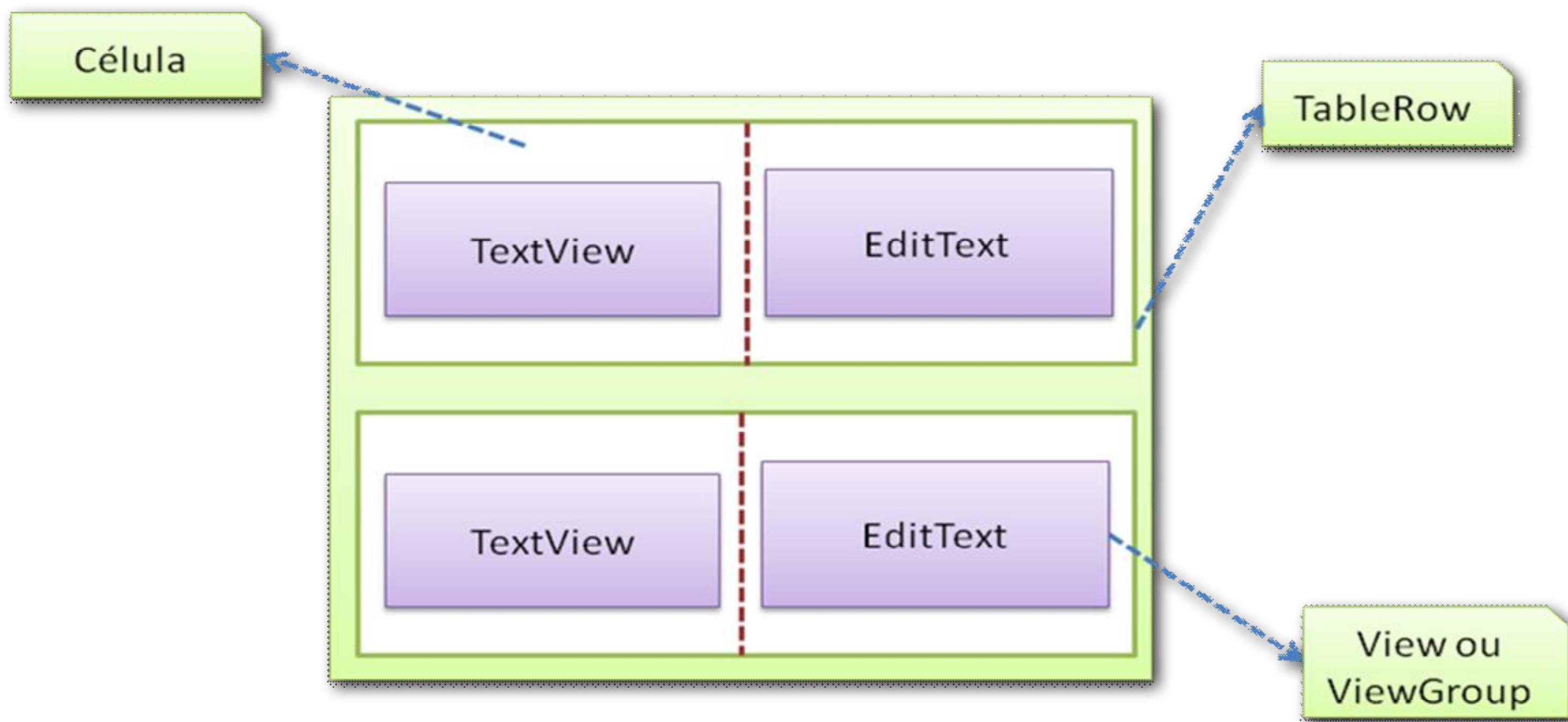
    <TextView android:id="@+id/tvSenha"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_centerHorizontal="true" android:text="Senha:"
        android:layout_below="@id/etLogin" />

    <EditText android:id="@+id/etSenha"
        android:layout_width="250px" android:layout_height="wrap_content"
        android:layout_centerHorizontal="true" android:text="Texto2"
        android:password="true" android:layout_below="@id/tvSenha" />

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Logar"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/etSenha" />
</RelativeLayout>
```



TableLayout



Organiza os componentes em linha e coluna.

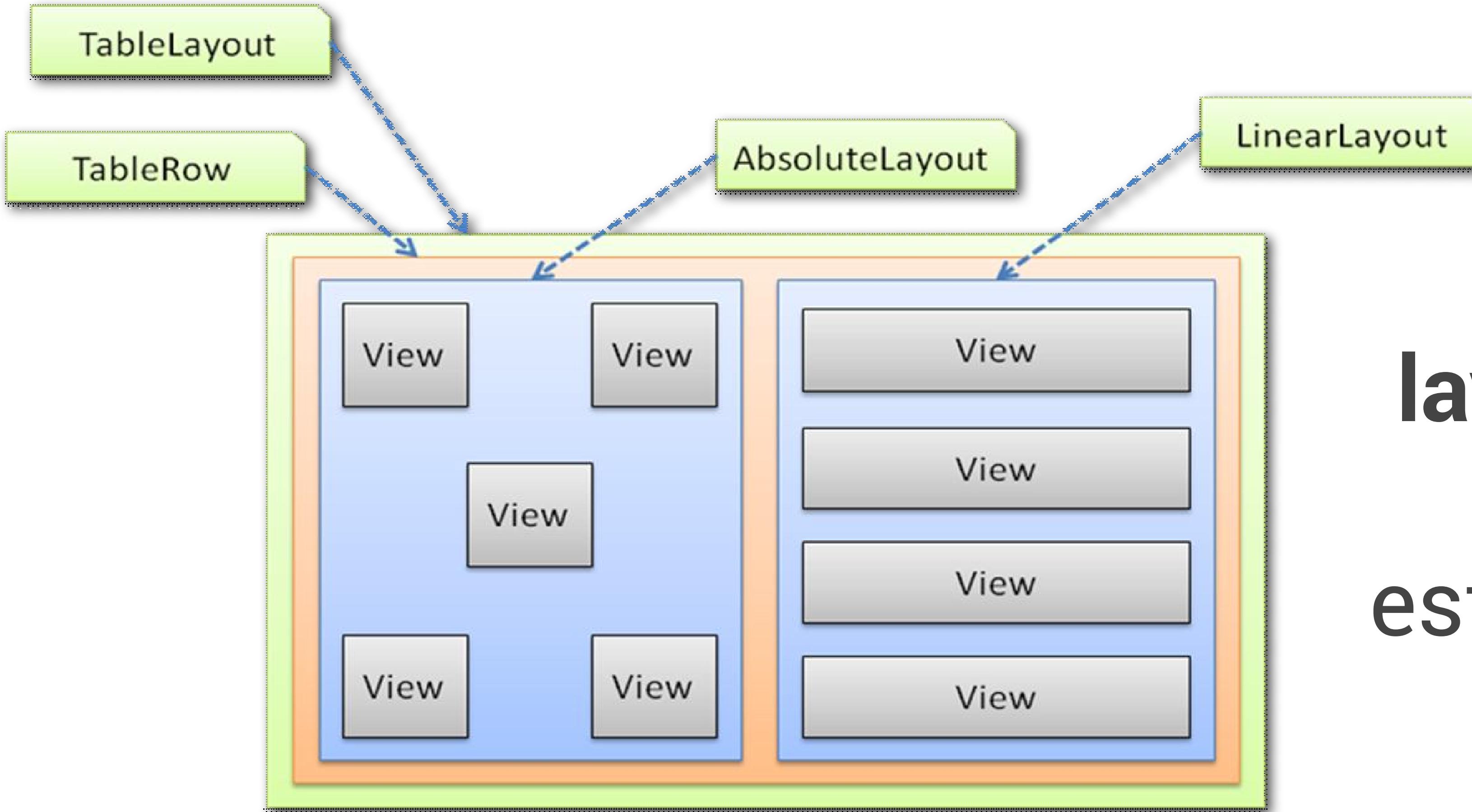


Exemplo de TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="fill_parent">
    <TableRow>
        <TextView android:text="Nome:" />
        <EditText android:text="Ramon Rabello" />
    </TableRow>
    <TableRow>
        <TextView android:text="Data Nasc.:" />
        <EditText android:text="21/03/1986" />
    </TableRow>
    <TableRow>
        <Button android:text="Cadastrar" />
    </TableRow>
</TableLayout>
```



Criando layouts complexos



A hierarquia de layouts em Android permite criar estruturas bem ricas e complexas.



Exemplo de um layout complexo

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TableRow>
        <AbsoluteLayout android:layout_width="fill_parent"
            android:layout_height="fill_parent">
            <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:text="View"
                android:layout_x="10px" android:layout_y="123px" />
            <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:text="View"
                android:layout_x="10px" android:layout_y="12px" />
            <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:text="View"
                android:layout_x="66px" android:layout_y="67px" />
            <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:text="View"
                android:layout_x="120px" android:layout_y="12px" />
            <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:text="View"
                android:layout_x="120px" android:layout_y="123px">
                </Button>
        </AbsoluteLayout>

        <LinearLayout android:orientation="vertical">
            <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:text="View" />
            <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:text="View" />
            <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:text="View" />
            <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:text="View" />
        </LinearLayout>
    </TableRow>
</TableLayout>
```



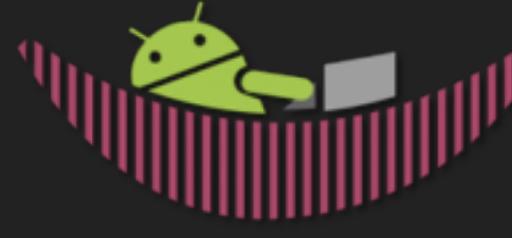
Podemos criar layouts...



Podemos criar layouts...

totalmente em XML...

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```



Podemos criar layouts...

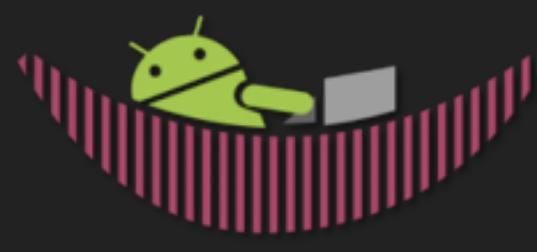
totalmente em XML...

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

ou em Java! Vc escolhe!

```
LinearLayout l = new LinearLayout(this);
l.setOrientation(LinearLayout.VERTICAL);
TextView tv = new TextView(this);
tv.setText("Olá, eu sou um TextView");
l.addView(tv);
```

```
Button b = new Button(this);
b.setText("Olá, eu sou um Button");
l.addView(b);
setContentView(l);
```



Carregando uma UI em XML



Carregando uma UI em XML

quando a aplicação é compilada, todo arquivo XML de layout é convertido para um objeto **View**.



Carregando uma UI em XML

quando a aplicação é compilada, todo arquivo XML de layout é convertido para um objeto **View**.

basta chamar o método **setContentView()** passando como parâmetro, uma referência (ID) para o layout.

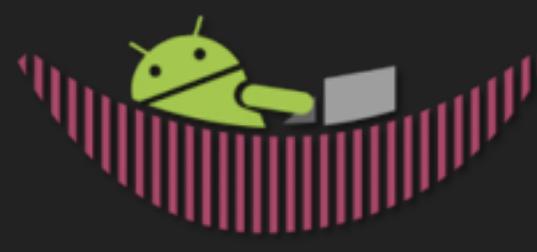


Carregando uma UI em XML

quando a aplicação é compilada, todo arquivo XML de layout é convertido para um objeto **View**.

basta chamar o método **setContentView()** passando como parâmetro, uma referência (ID) para o layout.

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main_layout);  
}
```



Atributos de um arquivo XML



Atributos de um arquivo XML

um componente gráfico pode possuir um atributo específico, mas todos eles herdam atributos da classe **View**
(ex: *android:id*)



Atributos de um arquivo XML

um componente gráfico pode possuir um atributo específico, mas todos eles herdam atributos da classe **View**
(ex: *android:id*)

existem atributos específicos que definem os parâmetros de layout referente a cada componente
(ex: *android:layout_width* e *android:layout_height*)



Nomeando componentes gráficos



Nomeando componentes gráficos

todo componente gráfico deve possuir um ID específico para que o mesmo possa ser identificado unicamente na árvore de componentes.



Nomeando componentes gráficos

todo componente gráfico deve possuir um ID específico para que o mesmo possa ser identificado unicamente na árvore de componentes.

A sintaxe para a definição de um ID é a seguinte:



Nomeando componentes gráficos

todo componente gráfico deve possuir um ID específico para que o mesmo possa ser identificado unicamente na árvore de componentes.

A sintaxe para a definição de um ID é a seguinte:

android:id="@+id/nomeDoComponente"



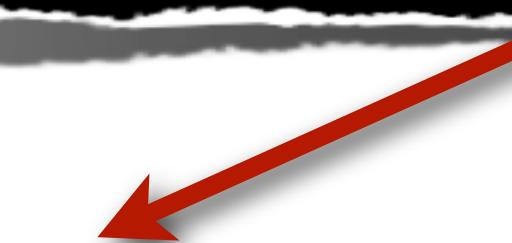
Nomeando componentes gráficos

todo componente gráfico deve possuir um ID específico para que o mesmo possa ser identificado unicamente na árvore de componentes.

A sintaxe para a definição de um ID é a seguinte:

android:id="@+id/nomeDoComponente"

símbolo para
informar
acesso a um
recurso





Nomeando componentes gráficos

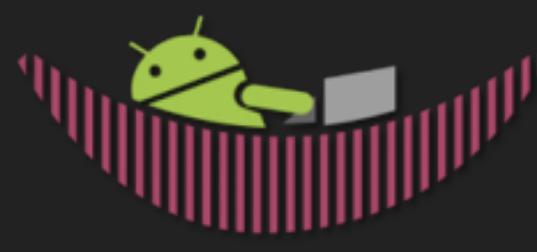
todo componente gráfico deve possuir um ID específico para que o mesmo possa ser identificado unicamente na árvore de componentes.

A sintaxe para a definição de um ID é a seguinte:

android:id="@+id/nomeDoComponente"

símbolo para
informar
acesso a um
recurso

indica que o recurso (id)
deve ser adicionado como
um novo recurso (na classe
R.java)



Recuperando componentes em código



Recuperando componentes em código

feito isso, podemos recuperar os componentes simplesmente assim:



Recuperando componentes em código

feito isso, podemos recuperar os componentes simplesmente assim:

```
ViewQualquer v = (ViewQualquer) findViewById(R.id.nomeDoComponente);
```



Recuperando componentes em código

feito isso, podemos recuperar os componentes simplesmente assim:

```
ViewQualquer v = (ViewQualquer) findViewById(R.id.nomeDoComponente);
```



tipo do View



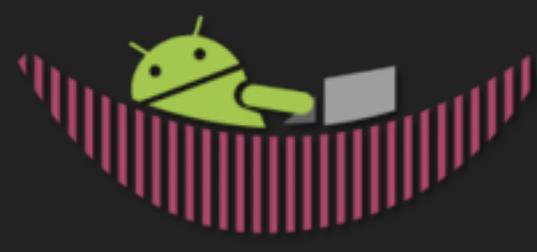
Recuperando componentes em código

feito isso, podemos recuperar os componentes simplesmente assim:

```
ViewQualquer v = (ViewQualquer) findViewById(R.id.nomeDoComponente);
```

 tipo do View

 ID do componente,
correspondente ao
atributo *android:id*



Exemplificando...



Exemplificando...

definição do código XML...

```
<Button android:id="@+id/botao"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/texto_botao"/>
```



Exemplificando...

definição do código XML...

```
<Button android:id="@+id/botao"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/texto_botao"/>
```

recuperando em Java...

```
Button botao = (Button) findViewById(R.id.botao);  
botao.setText("Alterando o texto do botão");
```



Exemplificando...

definição do código XML...

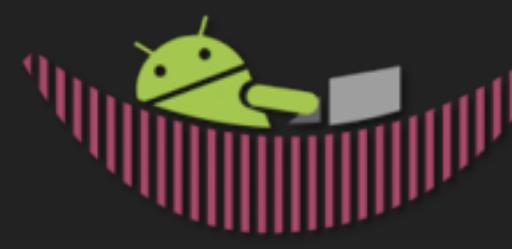
```
<Button android:id="@+id/botao"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/texto_botao"/>
```

recuperando em Java...

```
Button botao = (Button) findViewById(R.id.botao);  
botao.setText("Alterando o texto do botão");
```



após converter para objeto Java, os
métodos e atributos do componente
gráficos podem ser manipulados



Acessando outros recursos em XML



Acessando outros recursos em XML

referenciando textos estáticos...

```
<Button android:id="@+id/botao"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/texto_botao"/>
```



Acessando outros recursos em XML

referenciando textos estáticos...

```
<Button android:id="@+id/botao"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/texto_botao"/>
```



referencia o arquivo de
recurso res/values/
strings.xml



Acessando outros recursos em XML

referenciando textos estáticos...

```
<Button android:id="@+id/botao"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/texto_botao"/>
```



referencia o arquivo de
recurso res/values/
strings.xml

referenciando imagens...

```
<Button android:id="@+id/botao"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:background="@drawable/fundo_botao"/>
```



Acessando outros recursos em XML

referenciando textos estáticos...

```
<Button android:id="@+id/botao"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/texto_botao"/>
```



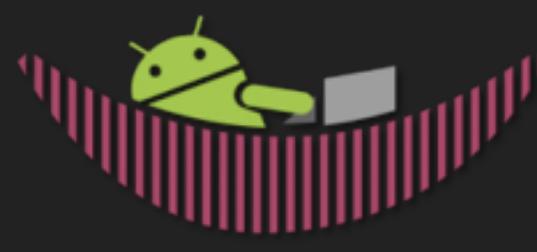
referencia o arquivo de
recurso res/values/
strings.xml

referenciando imagens...

```
<Button android:id="@+id/botao"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:background="@drawable/fundo_botao"/>
```



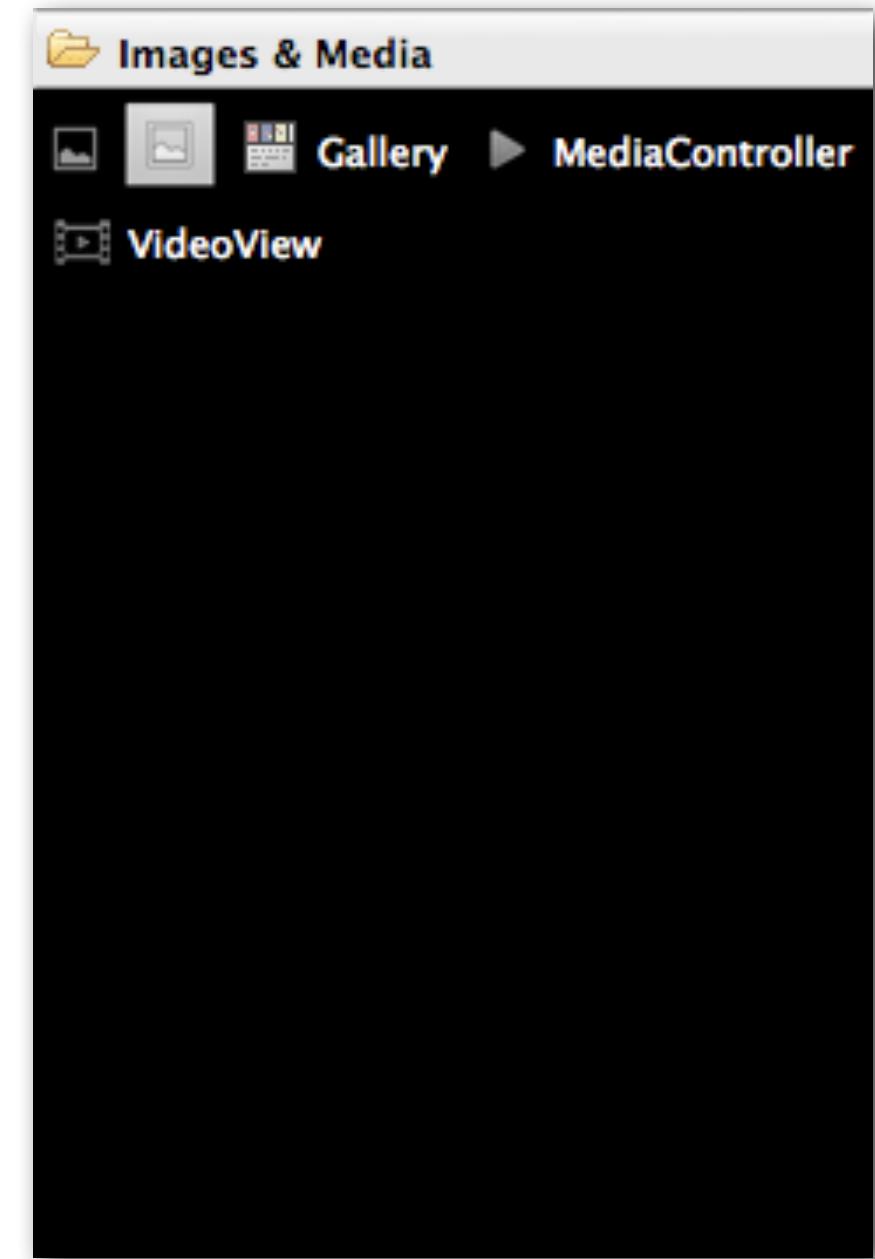
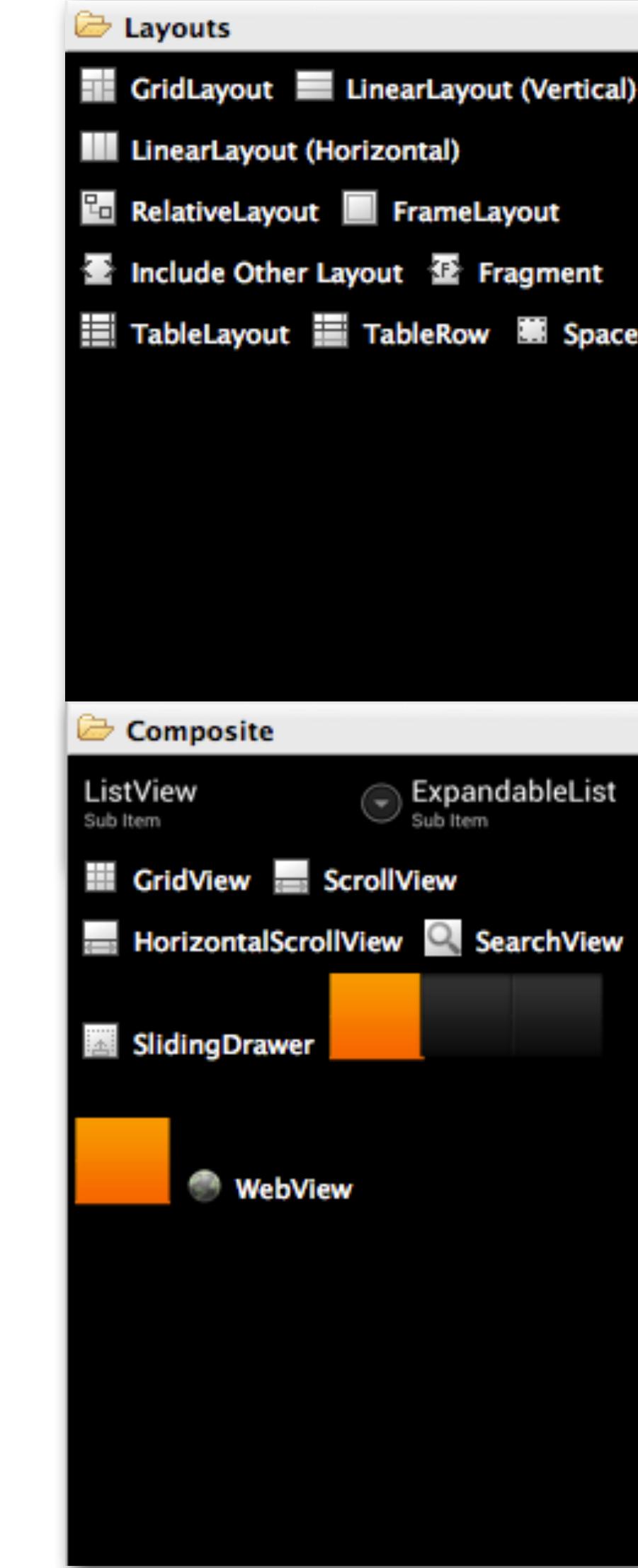
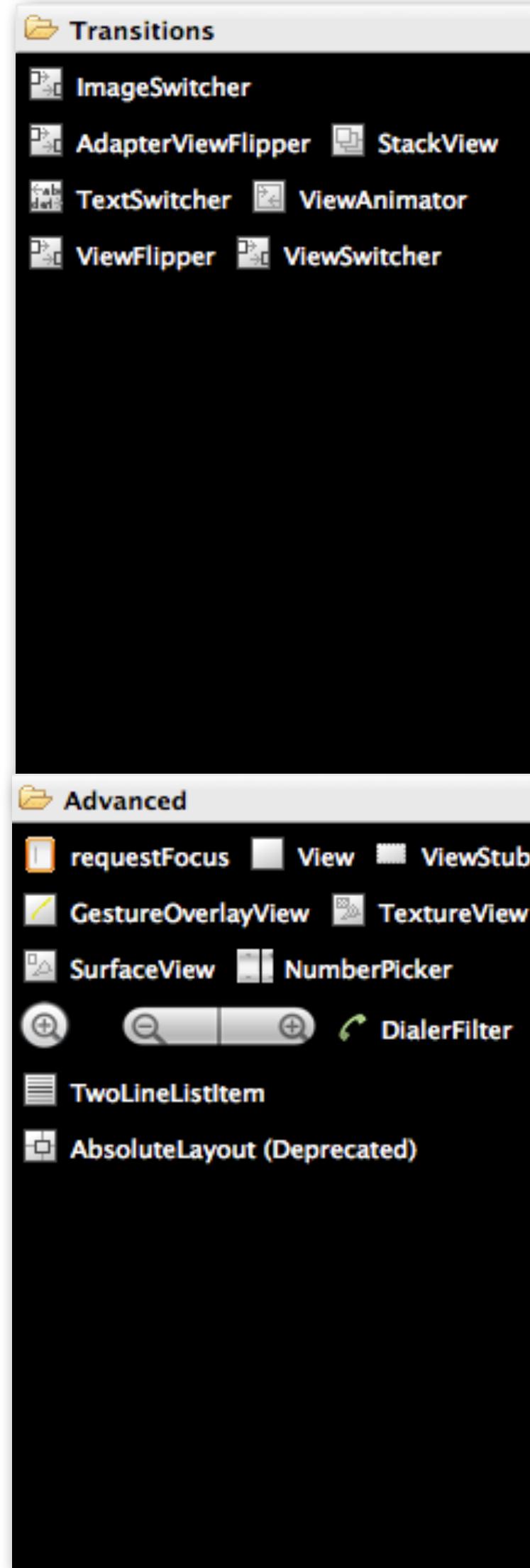
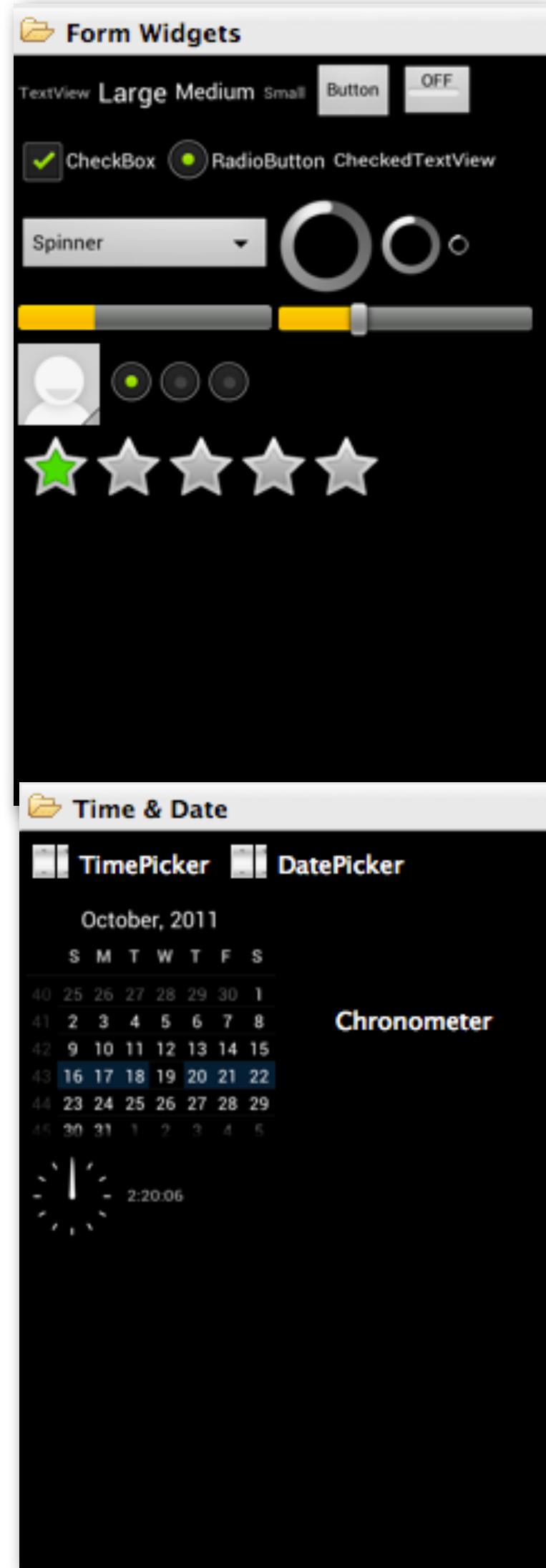
referencia o arquivo
fundo_botao.png localizado em
res/drawable-{nívelResolução}



Widgets e Layouts disponíveis

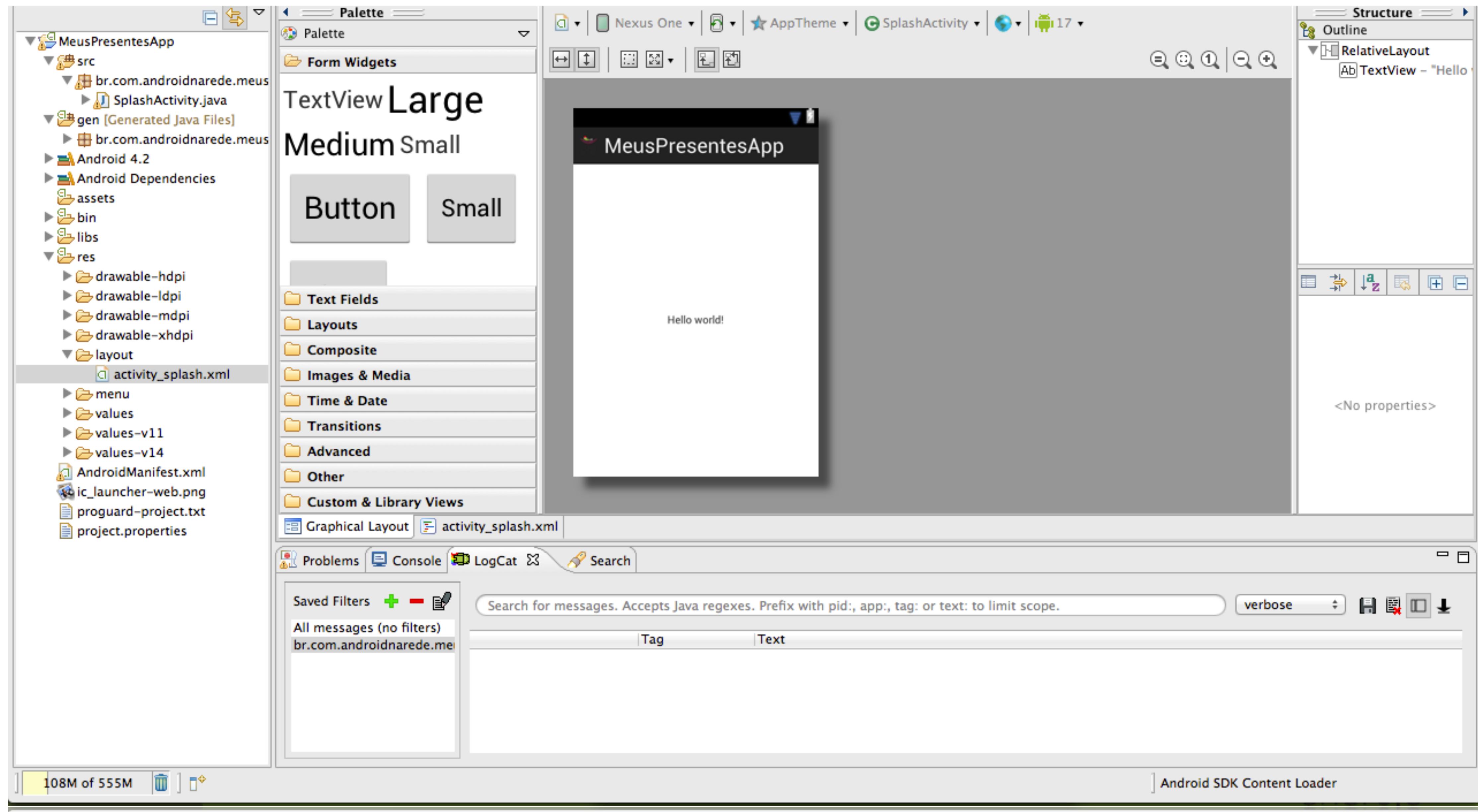


Widgets e Layouts disponíveis





O ADT facilita a criação de UIs com um Poderoso Editor!





```
⊕ <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:id="@+id/container"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="#000000"  
    android:orientation="vertical"  
    tools:context=".PrincipalActivity"  
    tools:ignore="MergeRootFrame" >  
    <TextView  
        android:id="@+id/textView1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_margin="10dp"  
        android:text="@string/descricao_app"  
        android:textColor="#ff9835" />  
/</LinearLayout>
```

Graphical Layout

activity_principal.xml



O ADT facilita a criação de UI em XML...

The screenshot shows the XML code for an Android layout file named `activity_principal.xml`. The code defines a `LinearLayout` with a vertical orientation, containing a single `TextView` with descriptive text. The `LinearLayout` has a black background and a white text view with orange text color and 10dp margin. The XML code is as follows:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000"
    android:orientation="vertical"
    tools:context=".PrincipalActivity"
    tools:ignore="MergeRootFrame" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="@string/descriacao_app"
        android:textColor="#ff9835" />
</LinearLayout>
```

At the bottom of the screen, there are two tabs: "Graphical Layout" and "activity_principal.xml".



Mão na massa!



Mão na massa!

- 1) Crie a interface gráfica de sua app utilizando uma mesclagem de layouts.
- 2) Crie uma interface gráfica utilizando AbsoluteLayout e altere a resolução para ver o que acontece com a aplicação.
- 3) Construa uma aplicação que simule o layout de uma calculadora simples, com todos os números e as principais operações: adição, subtração, multiplicação e divisão.
- 4) Construa uma aplicação que seja um visualizador de várias imagens.



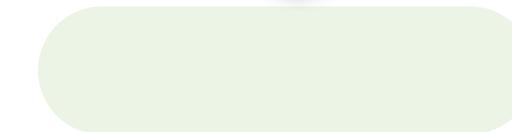
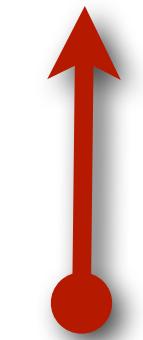
Activity

**Explorando a Activity,
o componente de tela
em Android**



Activity

classe mãe para
qualquer Activity





Activity

```
package hello.world;  
  
import android.app.Activity;  
  
public class HelloworldActivity extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

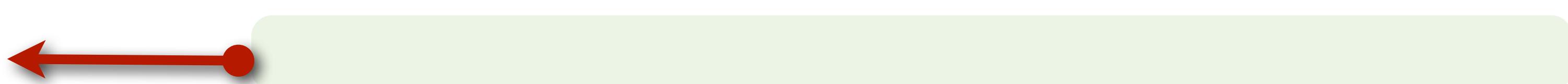
classe mãe para
qualquer Activity





Activity

método chamado
no momento da
criação de uma
Activity
(obrigatório)





Activity

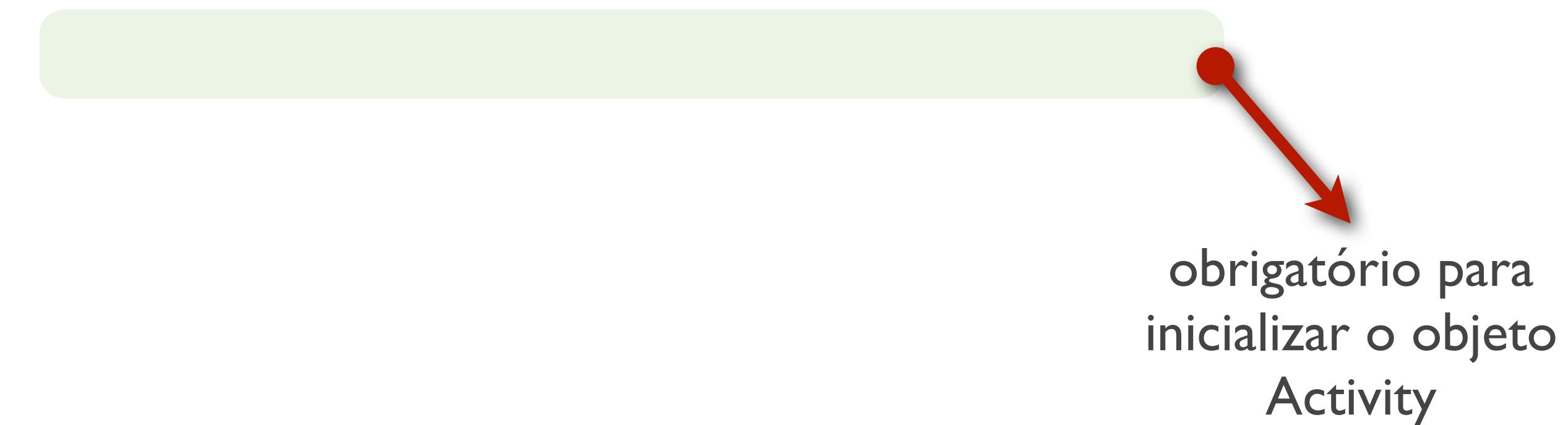
método chamado
no momento da
criação de uma
Activity
(obrigatório)

```
package hello.world;  
  
import android.app.Activity;  
  
public class HelloworldActivity extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```





Estrutura de uma Activity





Estrutura de uma Activity

```
package hello.world;  
  
import android.app.Activity;  
  
public class HelloworldActivity extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

obrigatório para
inicializar o objeto
Activity



Estrutura de uma Activity



carrega todo
conteúdo gráfico a
ser exibido pela
Activity



Estrutura de uma Activity

```
package hello.world;  
  
import android.app.Activity;  
  
public class HelloworldActivity extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```


carrega todo
conteúdo gráfico a
ser exibido pela
Activity



Relação entre Views e Activities



Relação entre Views e Activities

Existem alguns **Views**
que possuem um tipo de
Activity específico para
exibir o seu conteúdo



Relação entre Views e Activities

ListActivity:
específica para objetos
ListView

Existem alguns **Views**
que possuem um tipo de
Activity específico para
exibir o seu conteúdo



Relação entre Views e Activities

Existem alguns **Views** que possuem um tipo de Activity específico para exibir o seu conteúdo

ListActivity:
específica para objetos
`ListView`

ExpandableListActivity:
específica para objetos
`ExpandableListView`



Relação entre Views e Activities

Existem alguns **Views** que possuem um tipo de **Activity** específico para **exibir o seu conteúdo**

ListActivity:
específica para objetos
ListView

ExpandableListActivity:
específica para objetos
ExpandableListView

MapActivity:
específica para o componente
MapView



Relação entre Views e Activities

Existem alguns **Views** que possuem um tipo de **Activity** específico para exibir o seu conteúdo

ListActivity:
específica para objetos
`ListView`

ExpandableListActivity:
específica para objetos
`ExpandableListView`

MapActivity:
específica para o componente
`MapView`

TabActivity:
específica para o componente de abas
(`TabWidget`).



Exemplo: Exibindo lista simples

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

    <ListView android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#00FF00"
        android:layout_weight="1"
        android:drawSelectorOnTop="false"/>

    <TextView android:id="@+id/empty"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FF0000"
        android:text="No data"/>
</LinearLayout>
```



Exemplo: Exibindo ListView com ListActivity

```
package exemplo.listview;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;

public class ExemploListViewActivity extends ListActivity {

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

        String[] versoesAndroid = {"Cupcake", "Donut", "Eclair", "FroYo",
            "Gingerbread", "Honeycomb", "Ice Cream Sandwich",
            "Jelly Bean"};

        ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1,
            versoesAndroid);

        setListAdapter(arrayAdapter);
    }
}
```

Explaination of annotations:

- /** Called when the activity is first created. */: This is a multi-line comment indicating the method is called when the activity is first created.
- @Override: This annotation is placed above the onCreate method to indicate it overrides the parent's onCreate method.
- String[] versoesAndroid = {"Cupcake", "Donut", "Eclair", "FroYo", "Gingerbread", "Honeycomb", "Ice Cream Sandwich", "Jelly Bean"};: This line creates an array of strings containing the names of various Android versions.
- ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, versoesAndroid);: This line creates a new ArrayAdapter object, passing the current context (this), the layout for each list item (simple_list_item_1), and the array of strings (versoesAndroid).
- setListAdapter(arrayAdapter);: This line sets the list adapter for the ListActivity.

o layout do
ListView

o conteúdo
do ListView



Exemplo: Exibindo uma lista mais complexa

```
<?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="wrap_content"
    android:orientation="vertical">

    <TextView android:id="@+id/texto_nome"
        android:textSize="16sp"
        android:textStyle="bold"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <TextView android:id="@+id/texto_empresa"
        android:textSize="16sp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```



Exemplo: Exibindo uma lista mais complexa

lista_customizada_layout.xml

```
<?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="wrap_content"
    android:orientation="vertical">

    <TextView android:id="@+id/texto_nome"
        android:textSize="16sp"
        android:textStyle="bold"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <TextView android:id="@+id/texto_empresa"
        android:textSize="16sp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item,  
            cursor,  
            new String[] { People.NAME, People.COMPANY },  
            new int[] { android.R.id.text1, android.R.id.text2 });  
  
        setListAdapter(adapter);  
    }  
}
```

Subclasse de Activity que possui um ListView embutido utilizado para exibir listas simples (apenas textos).



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
        Carrega o layout para a lista, definido em  
        res/layout/lista_customizada_layout.xml  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item,  
            cursor,  
            new String[] { People.NAME, People.COMPANY },  
            new int[] { android.R.id.text1, android.R.id.text2 });  
  
        setListAdapter(adapter);  
    }  
}
```



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item,  
            cursor,  
            new String[] { People.NAME, People.COMPANY },  
            new int[] { android.R.id.text1, android.R.id.text2 });  
  
        setListAdapter(adapter);  
    }  
}
```

Carrega um Cursor (abstração para resultado de uma consulta) para todos os contatos registrados no dispositivo do usuário, por meio de content provider.



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item,  
            cursor,  
            new String[] { People.NAME, People.COMPANY },  
            new int[] { android.R.id.text1, android.R.id.text2 });  
  
        setListAdapter(adapter);  
    }  
}
```

Gerencia o cursor de acordo com o ciclo de vida da Activity (ex: finaliza conexão quando a Activity estiver pausada).



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item,  
            cursor,  
            new String[] { People.NAME, People.COMPANY },  
            new int[] { android.R.id.text1, android.R.id.text2 });  
  
        setListAdapter(adapter);  
    }  
}
```

Instancia um SimpleCursorAdapter, adaptador utilizado para exibir resultados armazenados em um Cursor.



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this, contexto do adaptador (a Activity atual)  
            android.R.layout.two_line_list_item,  
            cursor,  
            new String[] { People.NAME, People.COMPANY },  
            new int[] { android.R.id.text1, android.R.id.text2 });  
  
        setListAdapter(adapter);  
    }  
}
```



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item, layout que será utilizado na lista de 2 linhas  
            cursor,  
            new String[ ] { People.NAME, People.COMPANY },  
            new int[ ] { android.R.id.text1, android.R.id.text2 } );  
  
        setListAdapter(adapter);  
    }  
}
```



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item,  
            cursor, Cursor contendo todos os contatos  
            new String[ ] { People.NAME, People.COMPANY },  
            new int[ ] { android.R.id.text1, android.R.id.text2 } );  
  
        setListAdapter(adapter);  
    }  
}
```



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item,  
            cursor,  
            new String[] { People.NAME, People.COMPANY },  
            new int[] { android.R.id.text1, android.R.id.text2 });  
  
        setListAdapter(adapter);  
    }  
}
```

Array de Strings representando
os dados que serão exibidos
(nome e empresa do contato)



Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item,  
            cursor,  
            new String[] { People.NAME, People.COMPANY },  
            new int[] { android.R.id.text1, android.R.id.text2 } );  
        setListAdapter(adapter);  
    }  
}
```

Array de inteiros representando os IDs para qual serão vinculados os dados do contato



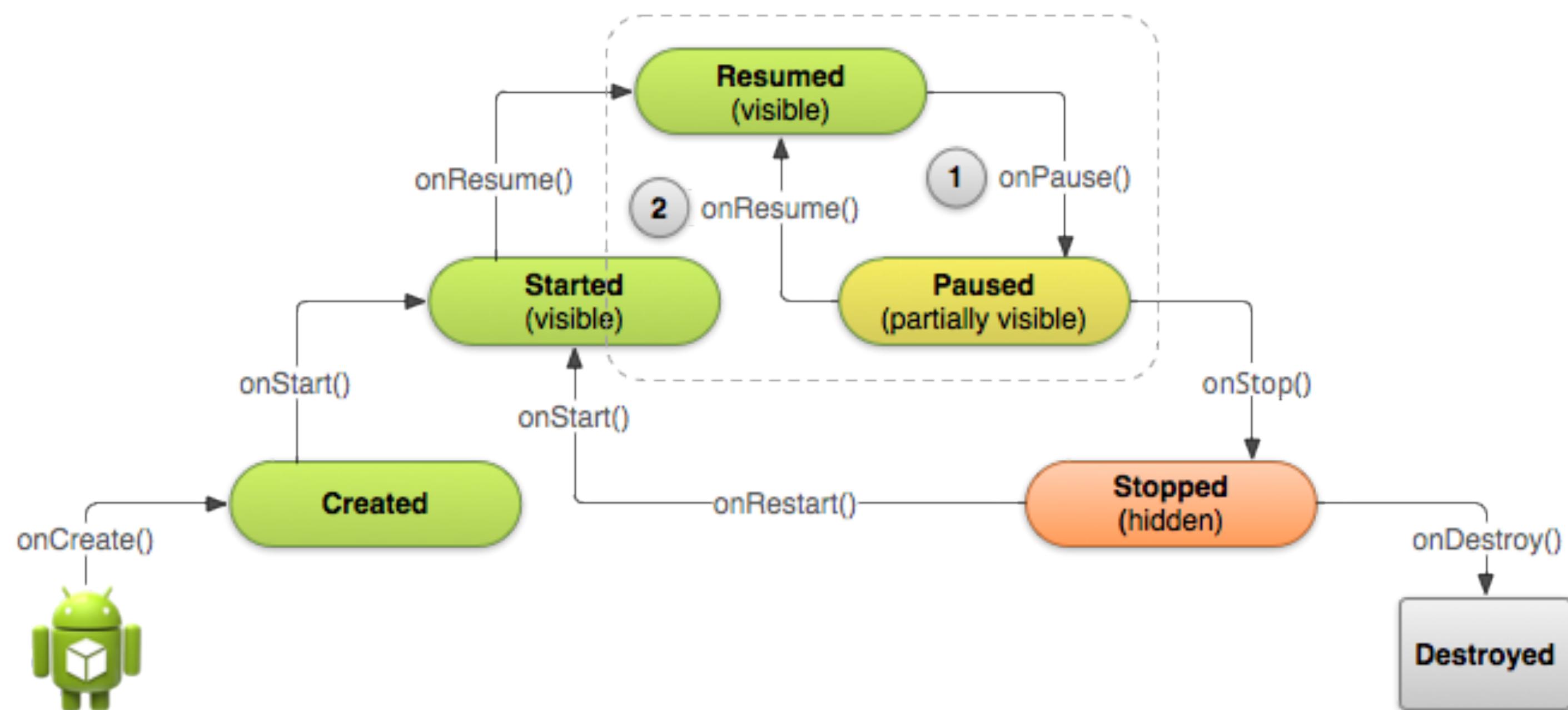
Exemplo: Exibindo lista complexa com ListActivity

```
public class MyListAdapter extends ListActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState){  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.lista_customizada_layout);  
  
        // recupera um Cursor para os contatos salvos no dispositivo do usuário  
        Cursor cursor = this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,  
null, null, null, null);  
  
        startManagingCursor(cursor);  
  
        ListAdapter adapter = new SimpleCursorAdapter(  
            this,  
            android.R.layout.two_line_list_item,  
            cursor,  
            new String[] { People.NAME, People.COMPANY },  
            new int[] { android.R.id.text1, android.R.id.text2 });  
  
        setListAdapter(adapter);  
    }  
}
```

Vincula o adaptador ao ListView
embutido pela ListActivity

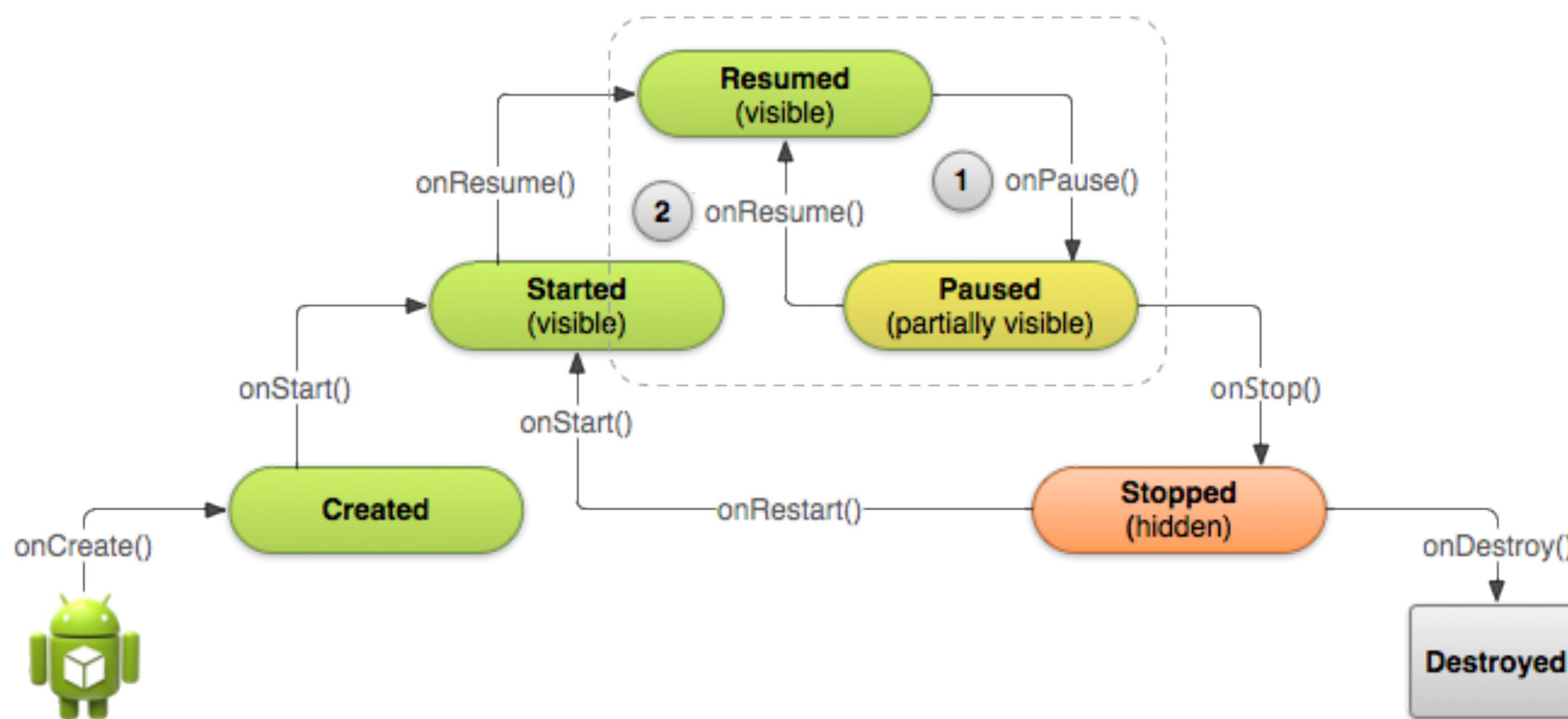


Ciclo de vida de uma Activity





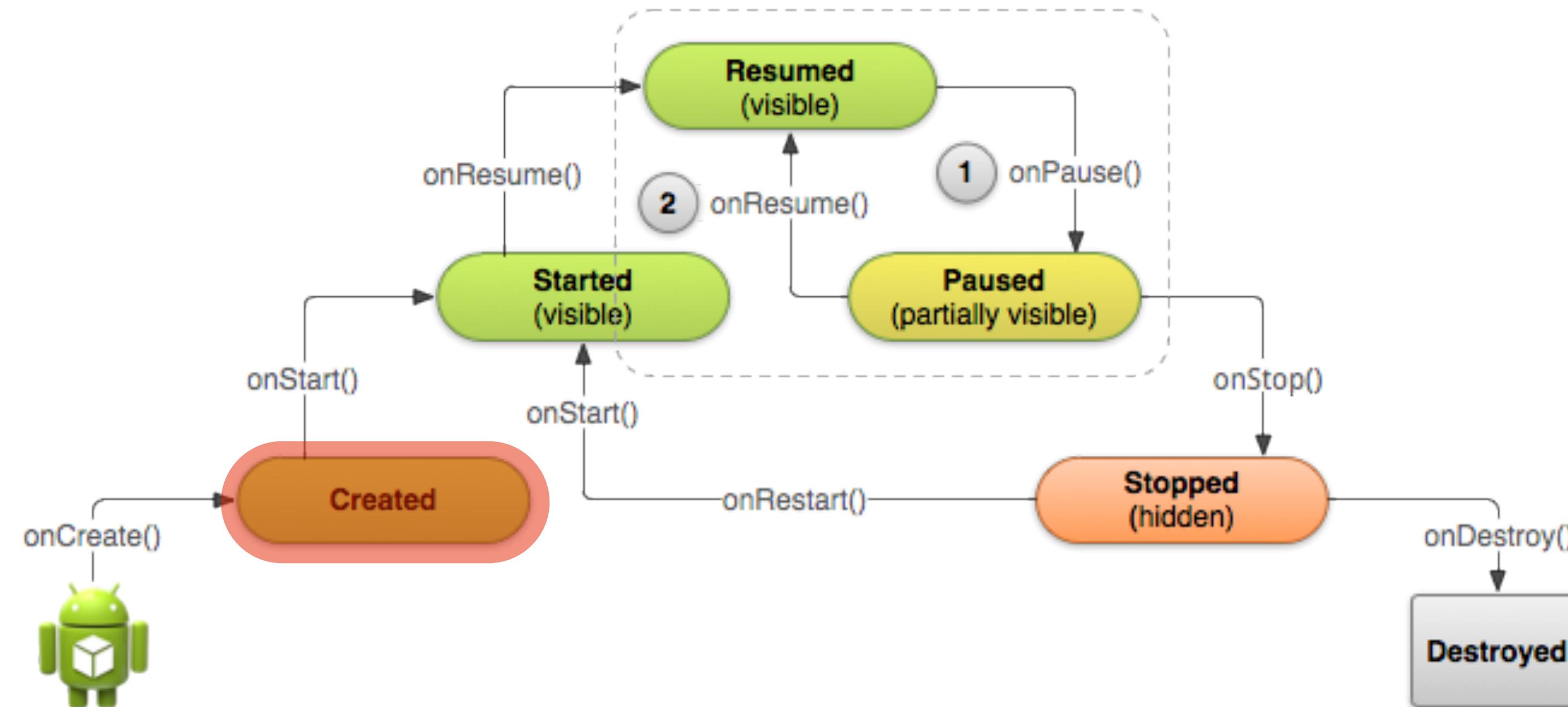
Ciclo de vida de uma Activity



Assim como outros componentes, uma **Activity** possui um ciclo de vida, num formato piramidal, representado por várias chamadas para métodos de *callback*.

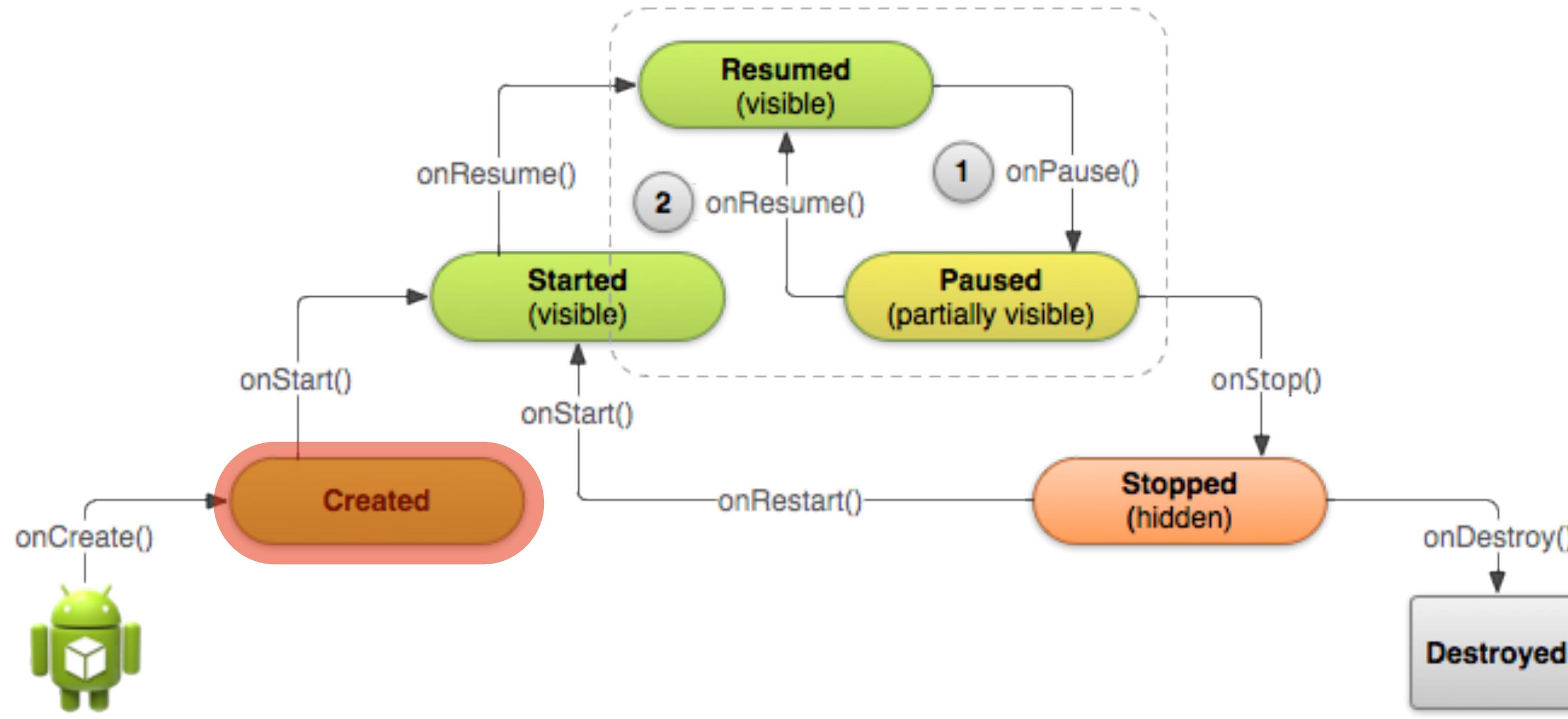


Principais estados de uma Activity





Principais estados de uma Activity



Criada:
A Activity acabou de ser instanciada, logo depois da chamada ao método **onCreate()**.



Principais estados de uma Activity: Criada

```
public class SplashActivity extends Activity {  
    // declaração de variáveis  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_splash);  
    }  
}
```



Principais estados de uma Activity: Criada

```
public class SplashActivity extends Activity {  
    // declaração de variáveis  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_splash);  
    }  
}
```

Boas práticas: Inicialize os componentes gráficos no método `onCreate()` e evite colocar trechos de código “pesados”, pois pode aparentar que a tela travou, haja vista que a Activity não está visível para o usuário.



Principais estados de uma Activity: Criada

```
public class SplashActivity extends Activity {  
    // declaração de variáveis  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_splash);  
    }  
}
```

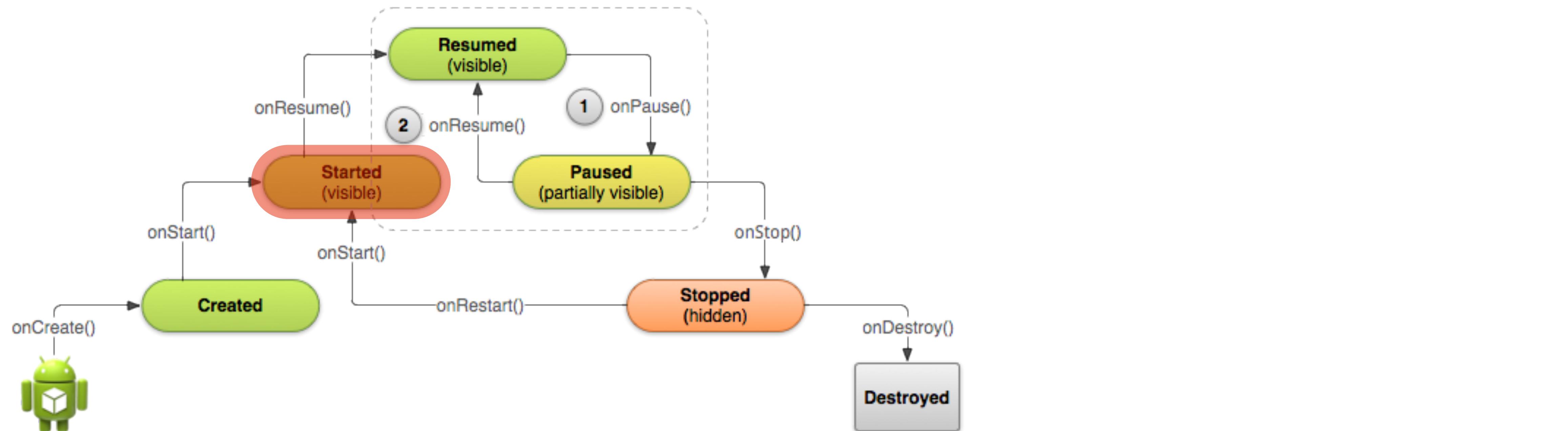
Em Java:

Deve-se sobrescrever o método **onCreate()**.

Boas práticas: Inicialize os componentes gráficos no método `onCreate()` e evite colocar trechos de código “pesados”, pois pode aparentar que a tela travou, haja vista que a Activity não está visível para o usuário.

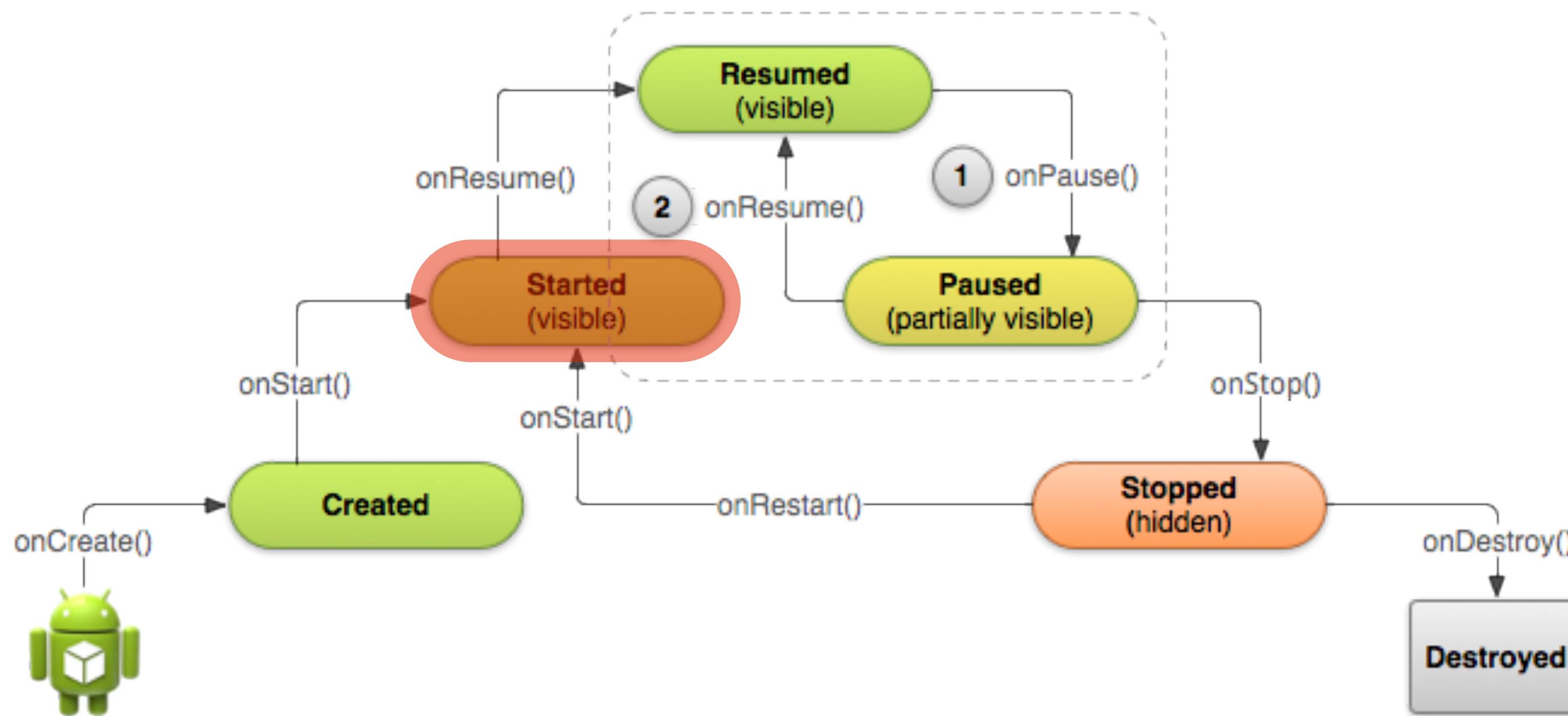


Principais estados de uma Activity





Principais estados de uma Activity



Iniciada:
A Activity foi iniciada
e está prestes a ser
tornar visivel para o
usuário.



Principais estados de uma Activity: Iniciada

```
public class SplashActivity extends Activity {  
  
    // declaração de variáveis  
  
    @Override  
    protected void onStart() {  
        super.onStart();  
    }  
}
```



Principais estados de uma Activity: Iniciada

```
public class SplashActivity extends Activity {  
  
    // declaração de variáveis  
  
    @Override  
    protected void onStart() {  
        super.onStart();  
    }  
}
```

Boas práticas: Utilize trechos de código que possam ser processados de maneira não custosa (ex: recuperar preferências).



Principais estados de uma Activity: Iniciada

```
public class SplashActivity extends Activity {  
    // declaração de variáveis  
  
    @Override  
    protected void onStart() {  
        super.onStart();  
    }  
}
```

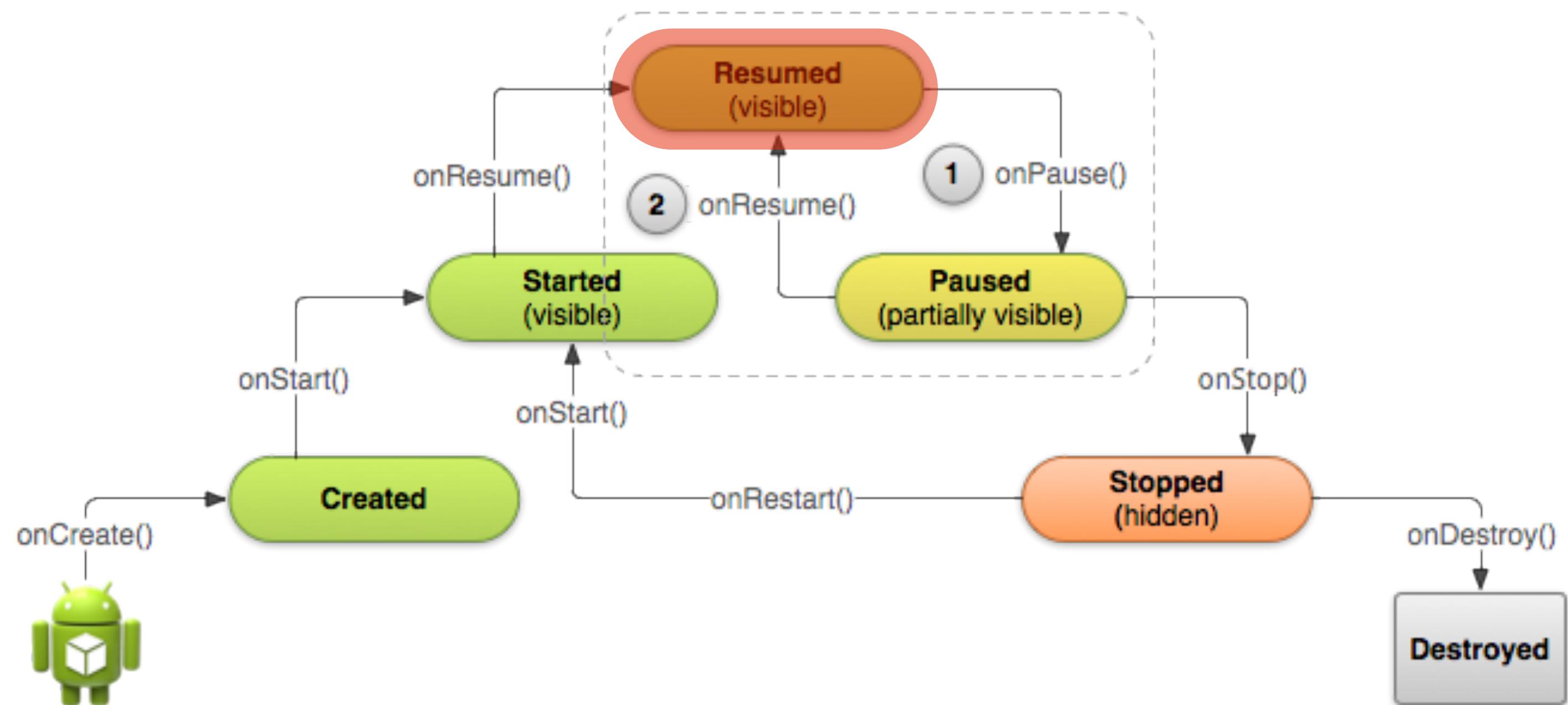
Em Java:

Deve-se sobrescrever o método **onStart()**.

Boas práticas: Utilize trechos de código que possam ser processados de maneira não custosa (ex: recuperar preferências).

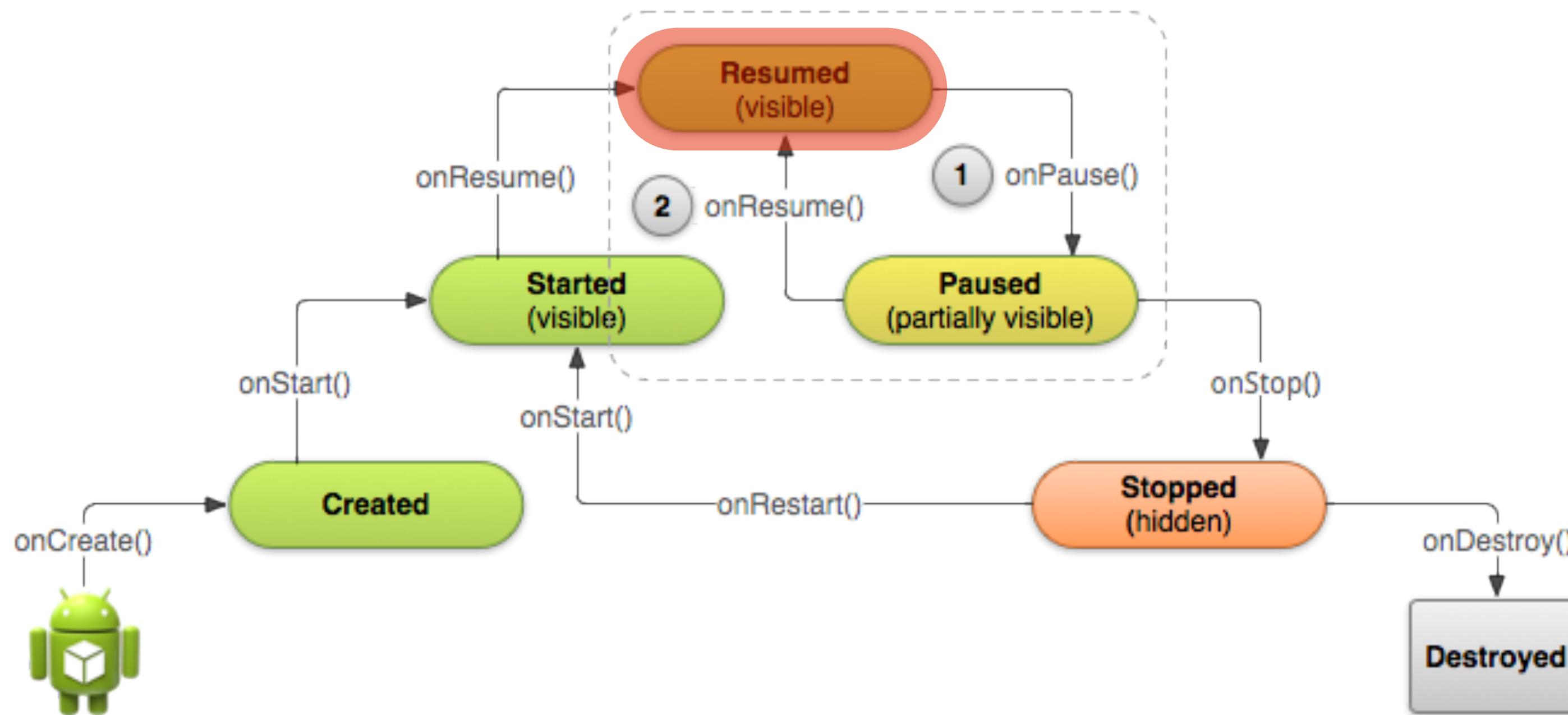


Principais estados de uma Activity





Principais estados de uma Activity



Recuperada:
A Activity foi iniciada
e está visivel e é
possível sua
interação com o
usuário.



Principais estados de uma Activity: Restaurada

```
public class SplashActivity extends Activity {  
  
    // declaração de variáveis  
  
    @Override  
    protected void onResume() {  
        super.onResume();  
    }  
}
```



Principais estados de uma Activity: Restaurada

```
public class SplashActivity extends Activity {  
    // declaração de variáveis  
    @Override  
    protected void onResume() {  
        super.onResume();  
    }  
}
```

Em Java:

Deve-se sobrescrever o método **onResume()**.



Principais estados de uma Activity: Restaurada

```
public class SplashActivity extends Activity {  
    // declaração de variáveis  
  
    @Override  
    protected void onResume() {  
        super.onResume();  
    }  
}
```

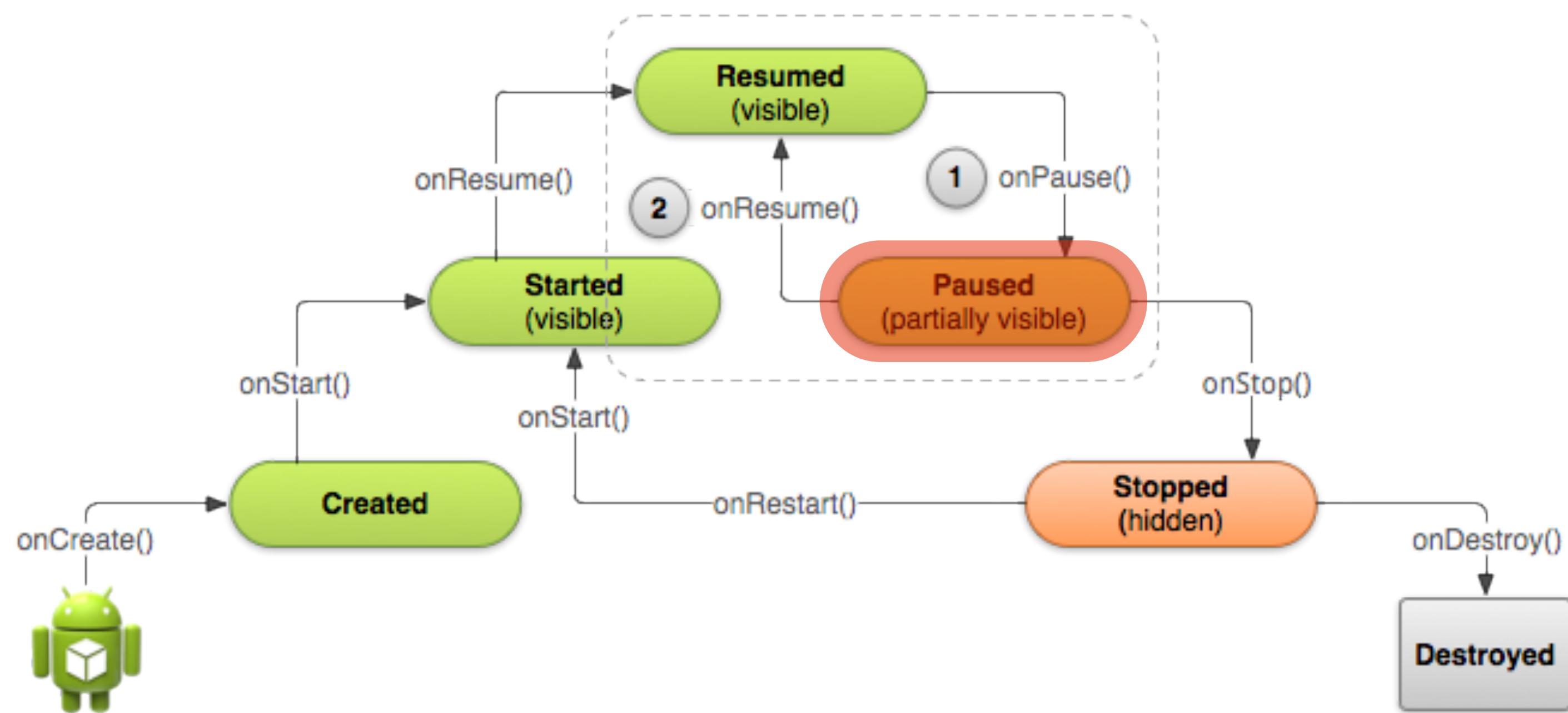
Em Java:

Deve-se sobrescrever o método **onResume()**.

Boas práticas: Caso exista trechos de códigos que executam tarefas dispendiosas (ex: acesso remoto ao servidor), exiba um componente ProgressBar ou ProgressDialog para dar maior feedback visual para o usuário.

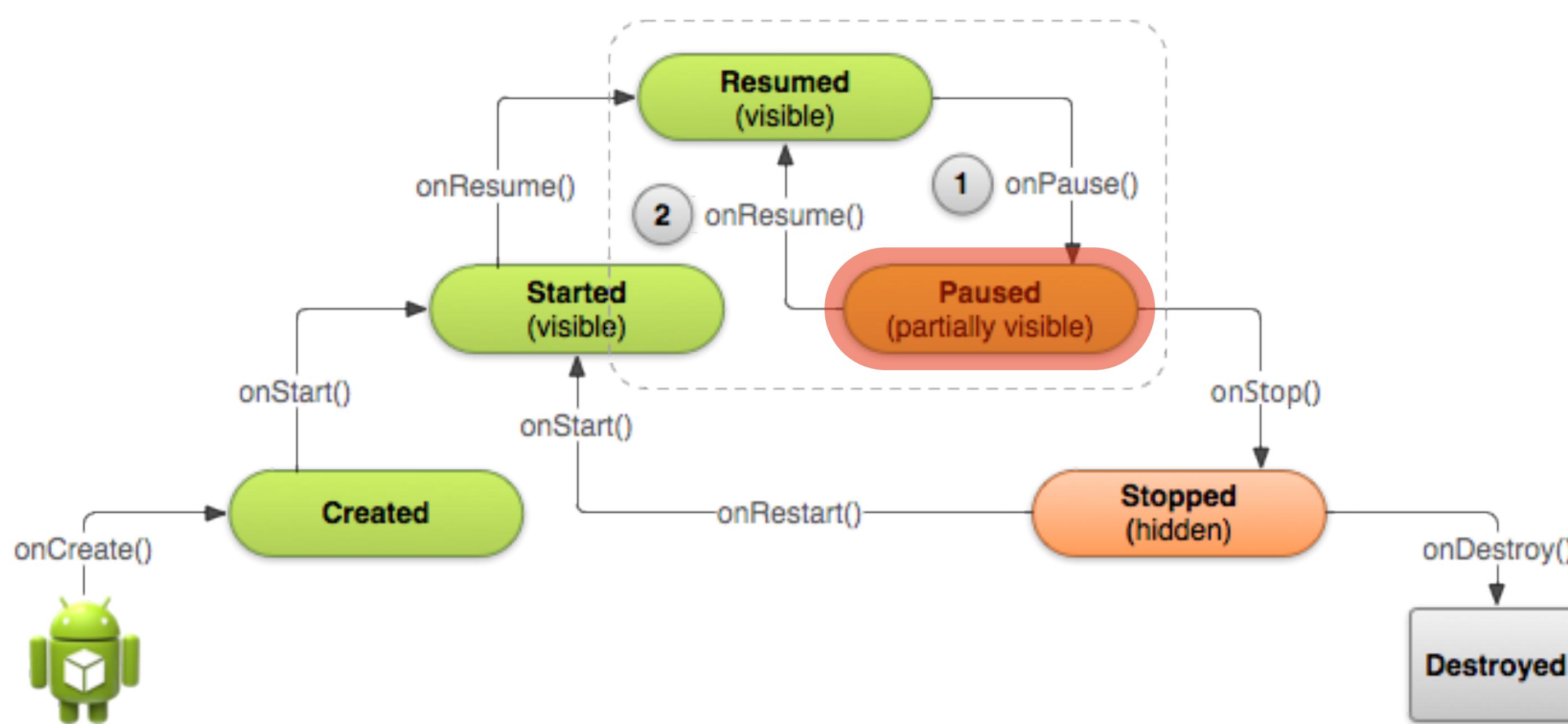


Principais estados de uma Activity





Principais estados de uma Activity



Pausada:
A Activity foi
obstruída por outra e
está parcialmente
visível. Nesse estado,
não há interação com
usuário.



Principais estados de uma Activity: Pausada

```
public class SplashActivity extends Activity {  
  
    // declaração de variáveis  
  
    @Override  
    protected void onPause() {  
        super.onPause();  
    }  
}
```



Principais estados de uma Activity: Pausada

```
public class SplashActivity extends Activity {  
  
    // declaração de variáveis  
  
    @Override  
    protected void onPause() {  
        super.onPause();  
    }  
}
```

Boas práticas: Utilize trechos de código que possam ser processados de maneira não custosa (ex: recuperar preferências).



Principais estados de uma Activity: Pausada

```
public class SplashActivity extends Activity {  
    // declaração de variáveis  
  
    @Override  
    protected void onPause() {  
        super.onPause();  
    }  
}
```

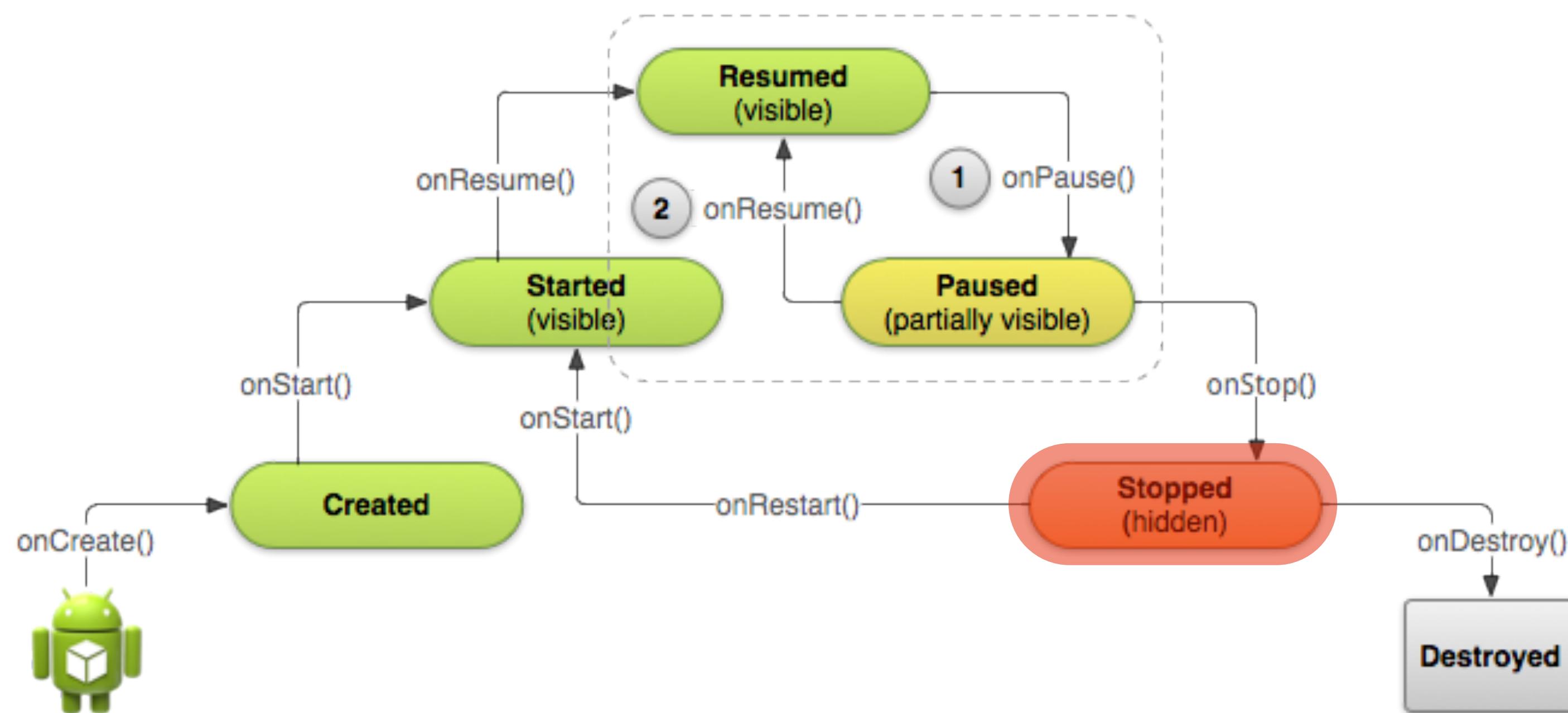
Em Java:

Deve-se sobrescrever o método **onPause()**.

Boas práticas: Utilize trechos de código que possam ser processados de maneira não custosa (ex: recuperar preferências).

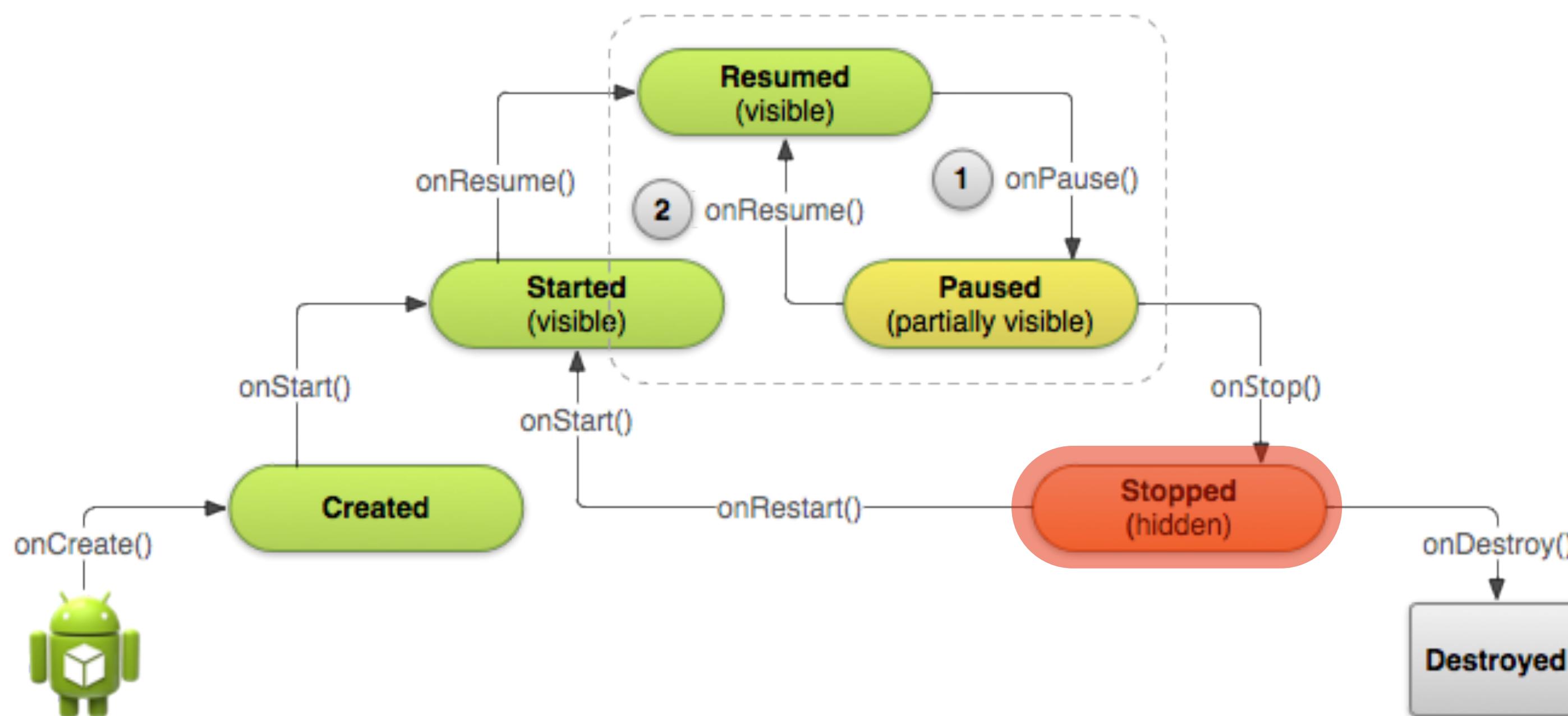


Principais estados de uma Activity





Principais estados de uma Activity



Parada:
A Activity não está
mais visível e sua
execução foi
interrompida.



Principais estados de uma Activity: Parada

```
public class SplashActivity extends Activity {  
  
    // declaração de variáveis  
  
    @Override  
    protected void onStop() {  
        super.onStop();  
    }  
}
```



Principais estados de uma Activity: Parada

```
public class SplashActivity extends Activity {  
  
    // declaração de variáveis  
  
    @Override  
    protected void onStop() {  
        super.onStop();  
    }  
}
```

Boas práticas: Desaloque recursos que estavam sendo utilizados pela aplicação (acesso à banco de dados, registro de atualização de posição geográfica, etc).



Principais estados de uma Activity: Parada

```
public class SplashActivity extends Activity {  
    // declaração de variáveis  
  
    @Override  
    protected void onStop() {  
        super.onStop();  
    }  
}
```

Em Java:

Deve-se sobrescrever o método **onStop()**.

Boas práticas: Desaloque recursos que estavam sendo utilizados pela aplicação (acesso à banco de dados, registro de atualização de posição geográfica, etc).



Iniciando uma Activity



Iniciando uma Activity

**toda Activity é inicializada por meio
da chamada:**

Activity.startActivity(Intent);



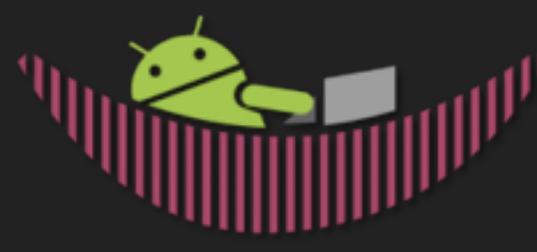
Finalizando uma Activity



Finalizando uma Activity

**toda Activity é finalizada por meio
da chamada:**

`Activity.finish();`



Uma aplicação real contém várias telas



Uma aplicação real contém várias telas

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="exemplo.listview"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >

        <activity
            android:label="@string/app_name"
            android:name=".TelaPrincipal" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:label="@string/app_name"
            android:name=".TelaListagem" />

        <activity
            android:label="@string/app_name"
            android:name=".TelaCadastro" />

    </application>

</manifest>
```



Uma aplicação real contém várias telas

Activity principal
(inicia a aplicação)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="exemplo.listview"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".TelaPrincipal" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:label="@string/app_name"
            android:name=".TelaListagem" />
        <activity
            android:label="@string/app_name"
            android:name=".TelaCadastro" />
    </application>
</manifest>
```



Uma aplicação real contém várias telas

Activity principal
(inicia a aplicação)

Activity para
tela de listagem
de produtos

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="exemplo.listview"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".TelaPrincipal" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:label="@string/app_name"
            android:name=".TelaListagem" />
        <activity
            android:label="@string/app_name"
            android:name=".TelaCadastro" />
    </application>
</manifest>
```



Uma aplicação real contém várias telas

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="exemplo.listview"
    android:versionCode="1"
    android:versionName="1.0" >

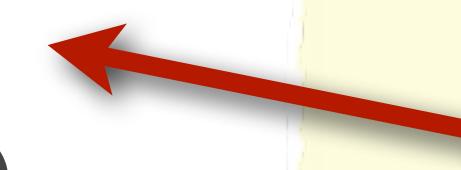
    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".TelaPrincipal" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:label="@string/app_name"
            android:name=".TelaListagem" />
        <activity
            android:label="@string/app_name"
            android:name=".TelaCadastro" />
    </application>
</manifest>
```

Activity principal
(inicia a aplicação)

Activity para tela de listagem de produtos

Activity para tela de cadastro de produtos





Tela Principal

TelaPrincipal.java

```
package androidnarede.variasactivities;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;

public class TelaPrincipal extends Activity {

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

tela_principal.xml

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_margin="10sp"
        android:layout_marginLeft="5sp"
        android:text="Seja bem vindo à aplicação!
Utilize o menu principal para realizar as operações."
        android:textColor="@android:color/white"
        android:textSize="20sp" />

</LinearLayout>
```



Tela de Cadastro

TelaCadastro.java

```
package androidnarede.variasactivities;

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

public class TelaCadastro extends Activity {

    @Override
    protected void onCreate(Bundle
    savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tela_cadastro);
    }
}
```

tela_cadastro.xml

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nome:" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sobrenome:" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="E-mail:" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:gravity="center_vertical"
        android:text="Cadastrar" />

</LinearLayout>
```



Tela de Listagem

TelaListagem.java

```
package androidnarede.variasactivities;

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

public class TelaListagem extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tela_listagem);
    }
}
```

tela_listagem.xml

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

    <ListView android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#00AA00"
        android:layout_weight="1"
        android:drawSelectorOnTop="false"/>

    <TextView android:id="@+id/empty"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="No data"/>
</LinearLayout>
```



Passando parâmetros para outra Activity

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.tela_principal);
    Intent intent = new Intent(this, OutraActivity.class);
    intent.putExtra("nome", "Fulano");
    intent.putExtra("idade", "26");
    intent.putExtra("sexo", "masculino");
    startActivity(intent);
}
```

Activity
de destino

Activity
de origem



Passando parâmetros para outra Activity

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.tela_principal);
    Intent intent = new Intent(this, OutraActivity.class);
    intent.putExtra("nome", "Fulano");
    intent.putExtra("idade", "26");
    intent.putExtra("sexo", "masculino");
    startActivity(intent);
}
```

Activity
de destino

Activity
de origem

Para passar parâmetros entre telas, podemos utilizar a própria classe **Intent** por meio dos métodos sobrecarregados **putExtra()**



Recuperando dados entre telas

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Intent intent = getIntent();
    String nome = intent.getStringExtra("nome");
    Toast.makeText(this, "Nome do Contato: " + nome).show();
}
```

Intent contendo os parâmetros fornecidos

nome do parâmetro fornecido

```
graph TD; A[Intent intent = getIntent();] --> B[Intent]; C[String nome = intent.getStringExtra("nome");] --> D[nome]
```



Recuperando dados entre telas

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Intent intent = getIntent();
    String nome = intent.getStringExtra("nome");
    Toast.makeText(this, "Nome do Contato: " + nome).show();
}
```

Intent contendo os parâmetros fornecidos

nome do parâmetro fornecido

Para recuperar o dado informado, basta utilizarmos o método **Activity.getIntent()**



Recuperando dados entre telas

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Intent intent = getIntent();
    String nome = intent.getStringExtra("nome");
    Toast.makeText(this, "Nome do Contato: " + nome).show();
}
```

Intent contendo os parâmetros fornecidos

nome do parâmetro fornecido

```
graph TD; A[Intent intent = getIntent();] --> B[Intent intent]; C[String nome = intent.getStringExtra("nome");] --> D[nome]
```



Recuperando dados entre telas

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Intent intent = getIntent();
    String nome = intent.getStringExtra("nome");
    Toast.makeText(this, "Nome do Contato: " + nome).show();
}
```

Intent contendo os parâmetros fornecidos

nome do parâmetro fornecido

```
graph TD; A[Intent intent = getIntent();] --> B[getStringExtra("nome")]; B <--> C[nome]
```

Depois, é suficiente chamarmos o método **Intent.get<Tipo>Extra()** informando o nome do parâmetro.



Adicionando Menus para as Activities

res/menu/menu_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/
    android">
    <item android:id="@+id/itemCadastro"
        android:title="Cadastro"></item>

    <item android:id="@+id/itemListagem"
        android:title="Listagem"></item>

    <item android:id="@+id/itemSair"
        android:title="Sair"></item>
</menu>
```



Adicionando Menus para as Activities

res/menu/menu_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/
    android">
    <item android:id="@+id/itemCadastro"
        android:title="Cadastro"></item>

    <item android:id="@+id/itemListagem"
        android:title="Listagem"></item>

    <item android:id="@+id/itemSair"
        android:title="Sair"></item>
</menu>
```

Todo e qualquer menu fica armazenado no diretório **res/menu/**. Ele inicia com a tag **<menu>** que contém outras tags **<item>**, que representam os itens pertencentes ao menu de opções.



Adicionando Menus para as Activities

res/menu/menu_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/
    android">
    <item android:id="@+id/itemCadastro"
        android:title="Cadastro"></item>

    <item android:id="@+id/itemListagem"
        android:title="Listagem"></item>

    <item android:id="@+id/itemSair"
        android:title="Sair"></item>
</menu>
```



Adicionando Menus para as Activities

res/menu/menu_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/
    android">
    <item android:id="@+id/itemCadastro"
        android:title="Cadastro"></item>

    <item android:id="@+id/itemListagem"
        android:title="Listagem"></item>

    <item android:id="@+id/itemSair"
        android:title="Sair"></item>
</menu>
```

android:id: identificador
para o item de menu

android:title: título para o
item de menu

android:icon: título para o
item de menu



Adicionando Menus para versões Android 3.0+ (API 11)

res/menu/menu_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/
    android">
    <item android:id="@+id/itemCadastro"
        android:showAsAction= "ifRoom"
        android:title="Cadastro"></item>

    <item android:id="@+id/itemListagem"
        android:showAsAction= "ifRoom"
        android:title="Listagem"></item>

    <item android:id="@+id/itemSair"
        android:title="Sair"></item>
</menu>
```



Adicionando Menus para versões Android 3.0+ (API 11)

res/menu/menu_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/
    android">
    <item android:id="@+id/itemCadastro"
        android:showAsAction= "ifRoom"
        android:title="Cadastro"></item>

    <item android:id="@+id/itemListagem"
        android:showAsAction= "ifRoom"
        android:title="Listagem"></item>

    <item android:id="@+id/itemSair"
        android:title="Sair"></item>
</menu>
```

OBS: Os itens que não tiverem com o atributo **android:showAsAction**, serão mapeados para o overflow button.

Caso você esteja desenvolvendo para versões da plataforma acima da Android 3.0+ (Nível 11), é possível utilizarmos o atributo **android:showAsAction**, que adiciona os itens de menu à barra de ações (ActionBar).



Carregando Menus na Activity

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.splash, menu);  
    return true;  
}
```



Carregando Menus na Activity

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.splash, menu);  
    return true;  
}
```

Para carregar os itens de menu, devemos primeiramente sobrepor o método **onCreateOptionsMenu()**.



Carregando Menus na Activity

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.splash, menu);  
    return true;  
}
```



Carregando Menus na Activity

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.splash, menu);  
    return true;  
}
```

Depois, chamamos o método **getMenuInflater()** que retorna um **MenuItem**, classe específica para “inflar” arquivos de menu.



Carregando Menus na Activity

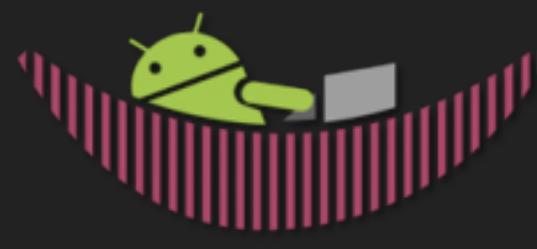
```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.splash, menu);  
    return true;  
}
```



Carregando Menus na Activity

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.splash, menu);  
    return true;  
}
```

Logo em seguida,
chamamos o método
inflate() informando o id
do arquivo e o próprio
objeto menu.



Carregando Menus na Activity

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.splash, menu);  
    return true;  
}
```



Carregando Menus na Activity

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.splash, menu);  
    return true;  
}
```

Retornamos **true** para
que o menu seja **exibido**.



Tratando eventos de clique em itens de menu

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action_cadastrar:{  
            startActivity(new Intent(this, CadastroActivity.class));  
            break;  
        }  
    }  
    return true;  
}
```



Tratando eventos de clique em itens de menu

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action_cadastrar:{  
            startActivity(new Intent(this, CadastroActivity.class));  
            break;  
        }  
    }  
    return true;  
}
```

Depois, adicionamos um switch-case em relação ao item selecionado, por meio do método **MenuItem.getItemId()**.



Tratando eventos de clique em itens de menu

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action_cadastrar:{  
            startActivity(new Intent(this, CadastroActivity.class));  
            break;  
        }  
    }  
    return true;  
}
```



Tratando eventos de clique em itens de menu

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action_cadastrar:{  
            startActivity(new Intent(this, CadastroActivity.class));  
            break;  
        }  
    }  
    return true;  
}
```

Agora basta utilizar a referência ao item de menu pela classe R.id.<item_menu> e adicionar o trecho necessário.



Tratando eventos de clique em itens de menu

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action_cadastrar:{  
            startActivity(new Intent(this, CadastroActivity.class));  
            break;  
        }  
    }  
    return true;  
}
```



Tratando eventos de clique em itens de menu

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action_cadastrar:{  
            startActivity(new Intent(this, CadastroActivity.class));  
            break;  
        }  
    }  
    return true;  
}
```

Por fim, retornamos **true** para informar à plataforma que o processamento do menu foi realizado.



Mão na massa!



Mão na massa!

1) Crie uma aplicação que simule um organizador pessoal, no qual vc poderá cadastrar e listar suas tarefas. Cada tarefa deverá conter:

- descrição
- local do evento
- data de execução da tarefa.

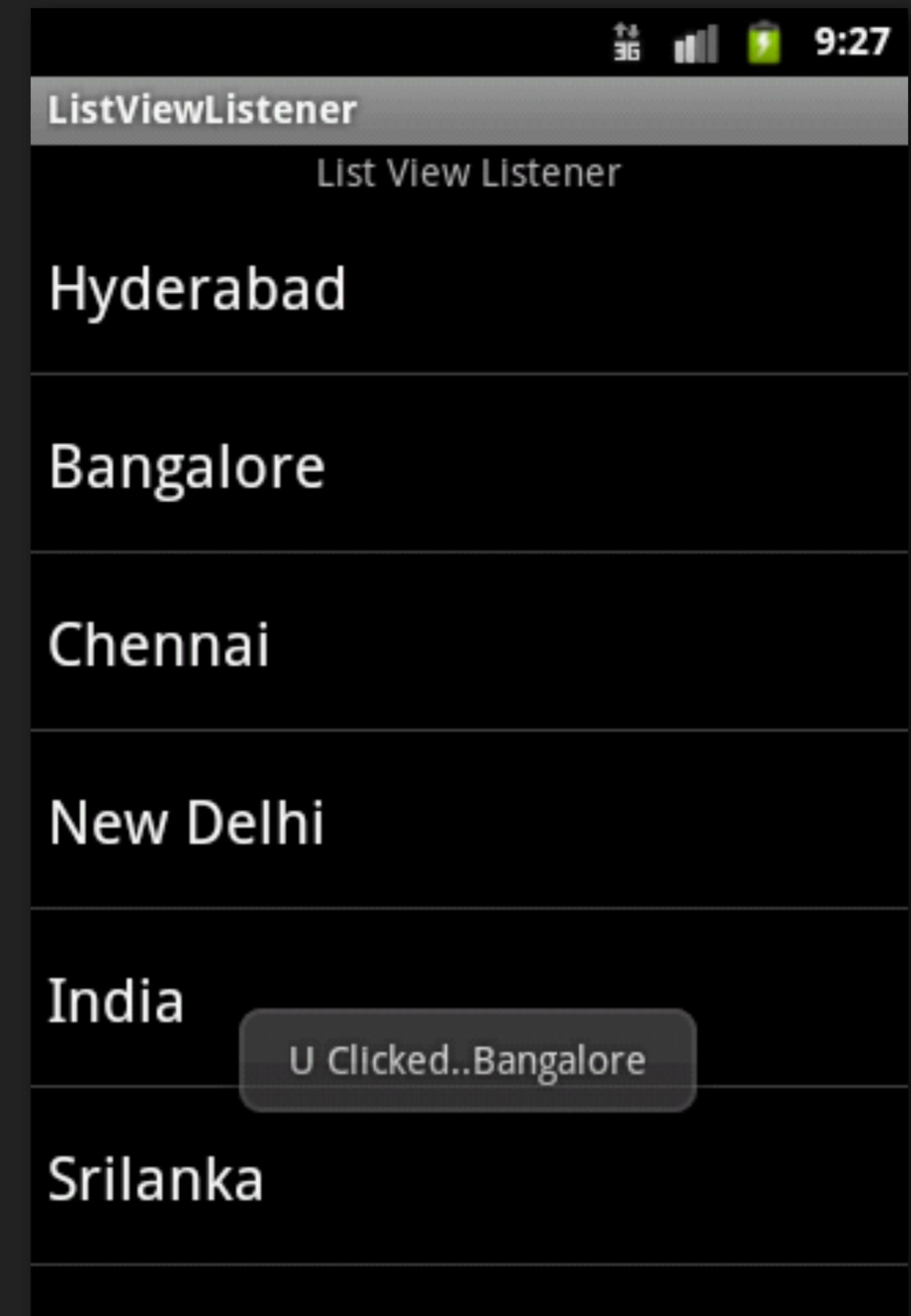
A aplicação deverá ser composta de **várias telas** e um **menu principal** para realizar as funcionalidades do sistema.

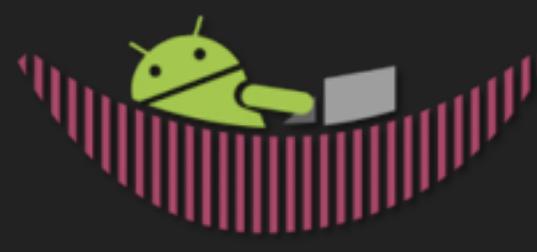
2) Utilize em sua aplicação, alguns métodos relacionados ao ciclo de vida de uma Activity e veja como eles se comportam no momento da execução.



Drawables

Tratamento de Eventos em Android





Views e Ouvintes de Eventos



Views e Ouvintes de Eventos

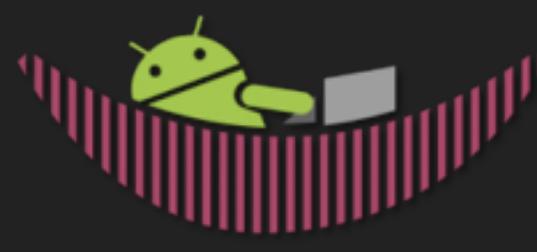
Várias **Views** possuem métodos que *callback* que serão chamados no momento que algum evento ocorrer (ex: clique num botão, seleção de um elemento na lista, etc).

Os **ouvintes de evento** (Event Listeners) serão responsáveis por notificá-lo que algum evento ocorreu.



Métodos de callback

método	classe ouvinte	descrição
onClick()	View.OnClickListener	chamado quando o usuário toca no componente ou pressiona "enter" quando estiver selecionado.
onLongClick()	View.OnLongClickListener	chamado quando o usuário realiza um pressionamento longo neste componente.
onFocusChange()	View.OnFocusChangeListener	chamado quando o usuário navega para frente ou para trás utilizando as teclas de navegação ou uma trackball.
onKey()	View.OnKeyListener	evento realizado quando o usuário pressiona ou libera uma tecla do dispositivo.
onTouch()	View.OnTouchListener	evento genérico que é disparado quando o usuário realiza qualquer ação classificada como toque (ex: pressionamento ,liberação ou qualquer gesto na tela).
onCreateContextMenu()	View.OnCreateContextMenuListener	chamado quando um menu de contexto está para ser exibido (ex: durante o pressionamento longo na tela).



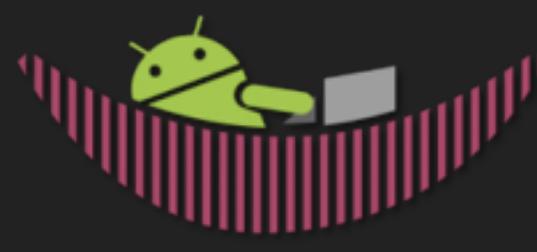
Registrando evento de clique no botão



Registrando evento de clique no botão

1. Criando uma implementação anônima da interface do evento

```
public class UmaActivity extends Activity {  
  
    private OnClickListener botaoListener = new OnClickListener() {  
        public void onClick(View v) {  
            // realiza algo quando o botão for clicado  
        }  
    };  
  
    protected void onCreate(Bundle savedValues) {  
        ...  
        // recupera o componente como um objeto Java  
        Button button = (Button) findViewById(R.id.botao);  
        // registra o botão para escutar o evento  
        button.setOnClickListener(botaoListener);  
        ...  
    }  
}
```



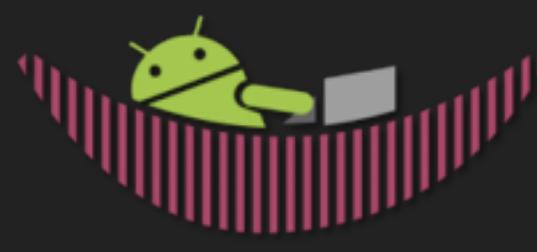
Registrando evento de clique no botão



Registrando evento de clique no botão

2. Implementando a interface de evento direto pela Activity

```
public class UmaActivity extends Activity implements OnClickListener {  
  
    protected void onCreate(Bundle savedInstanceState) {  
        ...  
        Button button = (Button) findViewById(R.id.botao);  
        button.setOnClickListener(this);  
    }  
  
    // implementa o callback do OnClickListener  
    public void onClick(View v) {  
        // realiza alguma ação quando o botão for clicado  
    }  
}
```



Registrando evento de clique no botão



Registrando evento de clique no botão

3. Ainda tem uma forma mais simples...

Defina o atributo android:onClick do componente em XML

```
<Button android:id="@+id/botao" android:layout_gravity="center_horizontal"  
        android:onClick="tratarEvento" android:layout_marginTop="50px"  
        android:background="@drawable/transicao_normal_play"  
        android:layout_width="112dp" android:layout_height="123dp" />
```

Depois, crie o método interno à Activity, com o mesmo nome

```
public class UmaActivity extends Activity {  
  
    // onCreate()  
  
    // implementa o callback do OnClickListener  
    public void tratarEvento(View v) {  
        // realiza alguma ação quando o botão for clicado  
    }  
}
```



Registrando evento de clique no botão

3. Ainda tem uma forma mais simples...

Defina o atributo android:onClick do componente em XML

```
<Button android:id="@+id/botao" android:layout_gravity="center_horizontal"  
        android:onClick="tratarEvento" android:layout_marginTop="50px"  
        android:background="@drawable/transicao_normal_play"  
        android:layout_width="112dp" android:layout_height="123dp" />
```

Depois, crie o método interno à Activity, com o mesmo nome

```
public class UmaActivity extends Activity {  
  
    // onCreate()  
  
    // implementa o callback do OnClickListener  
    public void tratarEvento(View v) {  
        // realiza alguma ação quando o botão for clicado  
    }  
}
```



Mão na Massa!



Mão na Massa!

- 1) Na calculadora que você criou no exercício anterior, implemente a parte comportamental utilizando o tratamento de eventos. Sendo assim, a calculadora deverá permitir que ao clicar nos botões, o display deve exibir o progresso da operação.
- 2) Explore outros métodos e tratadores de eventos e utilize em sua aplicação.

Dica: Os métodos para registrar eventos, tem o seguinte formato:
componente.set...Listener()



**Enriquecendo a
aparência da sua app
com gradientes e
seletores**





Drawables

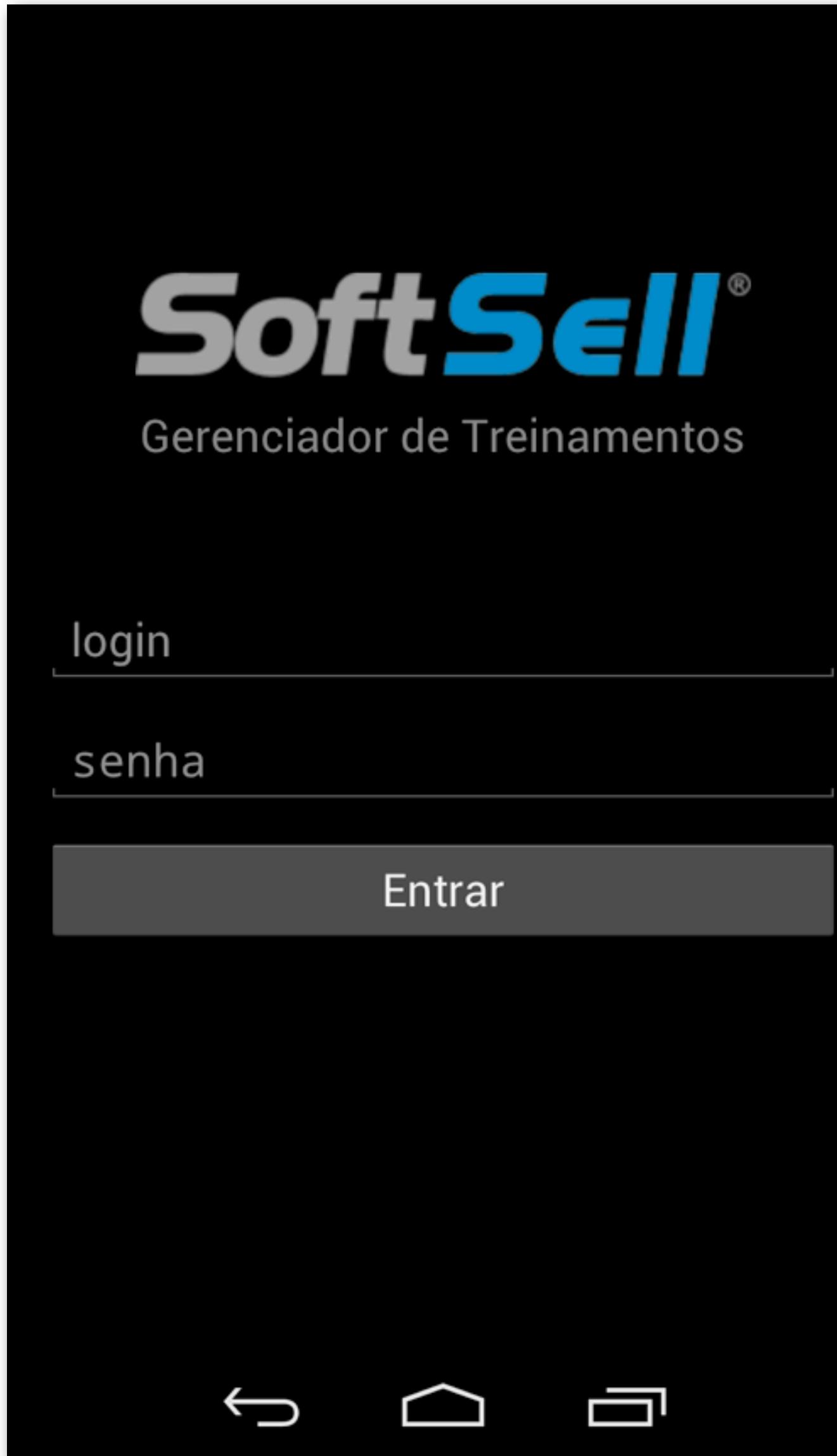
Drawable é
qualquer recurso
que possua
característica de
imagem
(ex: imagens
png, gradientes,
transições,
seletores, etc)

Estão
localizados na
pasta **res/**
drawable-
{qualificador} e
podem ser
definidos
também em
XML

Um drawable é
facilmente
referenciado por
meio do comando
'@drawable/
{nome_do_recurso}'



Drawables



Existem vários tipos de Drawables:
ImageDrawable
ColorDrawable
ColorListDrawable
ShapeDrawable
StateListDrawable

Drawables podem ser
muito poderosos
quando bem utilizados,
como por exemplo, para
dar mais identidade
visual para a sua app.



O que são gradientes?

Entrar

Representado pelo componente GradientDrawable, permite que aplicar um gradiente (intervalo de cores) ao layout do componente

res/drawable/fundo_botao.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <gradient
        android:angle="270"
        android:startColor="#00a5e0"
        android:endColor="#0092c6" />

    <corners android:radius="2dp" />

    <padding
        android:bottom="10dp"
        android:left="10dp"
        android:right="10dp"
        android:top="10dp" />

</shape>
```



Utilizando um gradiente

Basta referenciar o nome do xml, por meio do comando '@drawable/nome_do_xml_do_gradiente'

<Button

```
    android:id="@+id/button1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editText2"
    android:layout_centerHorizontal="true"
    android:layout_marginLeft="2dp"
    android:layout_marginRight="2dp"
    android:layout_marginTop="10dp"
    android:background="@drawable/fundo_botao"
    android:onClick="entrar"
    android:text="@string/texto_botao_entrar"
    android:textColor="@android:color/white"
    android:textSize="20sp" />
```

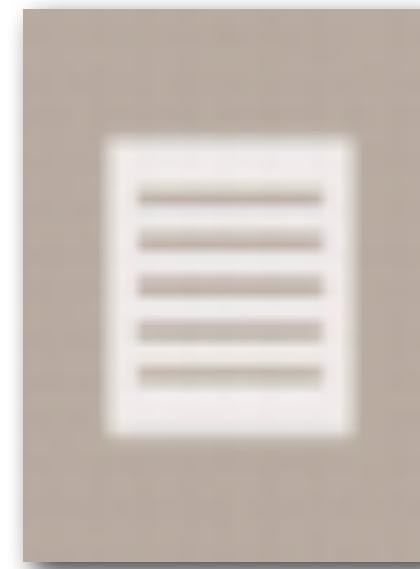


Seletores

Representado pelo componente `StateListDrawable`, permite assumir comportamentos de acordo com seu estado (ex: pressionado, em foco, etc)



normal



pressionado



em foco



Definindo um seletor para estados do botão

Para criarmos um seletor, basta criarmos um XML com a tag <selector> contendo os drawables que serão carregados de acordo com o estado do seletor.

res/drawable/botao.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/
res/android" >
    <item android:state_pressed="false"
        android:drawable="@drawable/fundo_botao"></item>
    <item android:state_pressed="true"
        android:drawable="@drawable/fundo_botao_pressionado"></
item>
</selector>
```



Utilizando o seletor

Devemos
referenciar o
arquivo xml que
representa o
seletor por meio
de algum atributo
(ex: background)

```
<Button  
    android:id="@+id/botao_entrar"  
    android:background="@drawable/botao"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentTop="true"  
/>
```