

Desenvolvimento com Google

ANDROID

Level II



Release 001
2010

ENG
[Redacted]

NÃO COPIAR OU DISTRIBUIR
ESTE DOCUMENTO. USO
INTERNO/TREINAMENTO.

Introdução

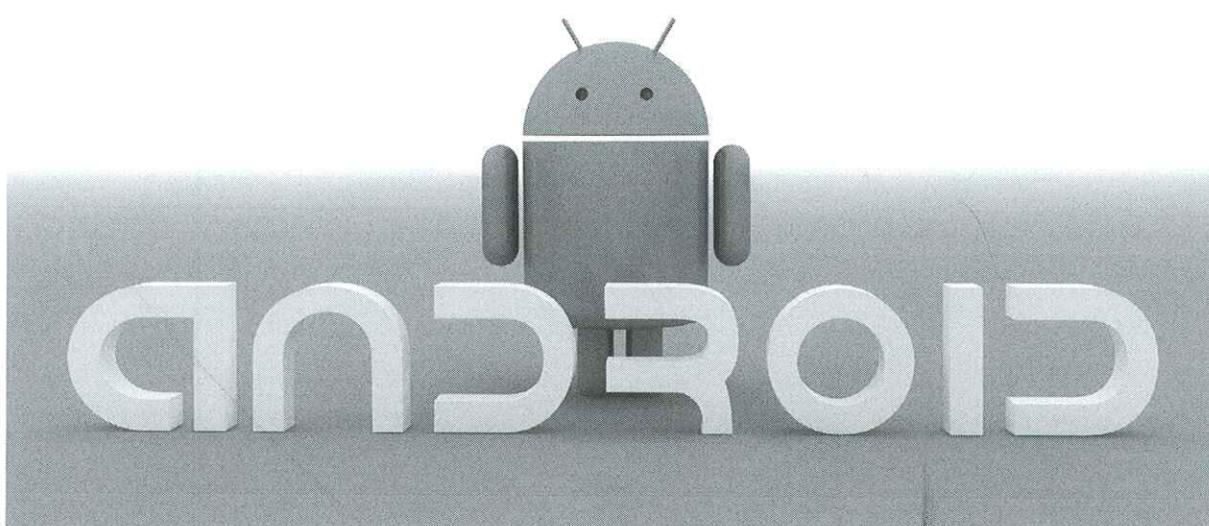
Android é a primeira plataforma móvel aberta e totalmente personalizável. O Android oferece uma solução completa: um sistema operacional baseado no GNU/Linux customizado para dispositivos móveis, middleware e os aplicativos essenciais como um browser para navegar na internet, o Google Maps, player de vídeo e música entre outros. Contém também um rico conjunto de APIs que permite que desenvolvedores criem grandes aplicações.

O Android é o resultado da união entre o Google e gigantes do mercado de telefonia, que juntos criaram a OHA (Open Handset Alliance).

No Android quase tudo é possível, tendo como ponto forte a customização das aplicações nativas do celular, além da troca dessas por versões mais recentes ou modificadas, o que atrai os grandes fabricantes de dispositivos móveis (atualmente marcas potenciais como HTC, Samsung e Motorola estão vendendo celulares Android com interfaces e programas 100% customizados).

Pro
Android
2
Nashville
stable

almeida.com.br



Objetivos do Curso

Desenvolver aplicativos com recursos visuais para dispositivos baseados no sistema operacional Android do Google;

Funcionamento dos aplicativos Android, ciclo de vida, Intents etc;

Montagem do ambiente de programação;

Criar interfaces gráficas das suas aplicações usando XML e/ou Java;

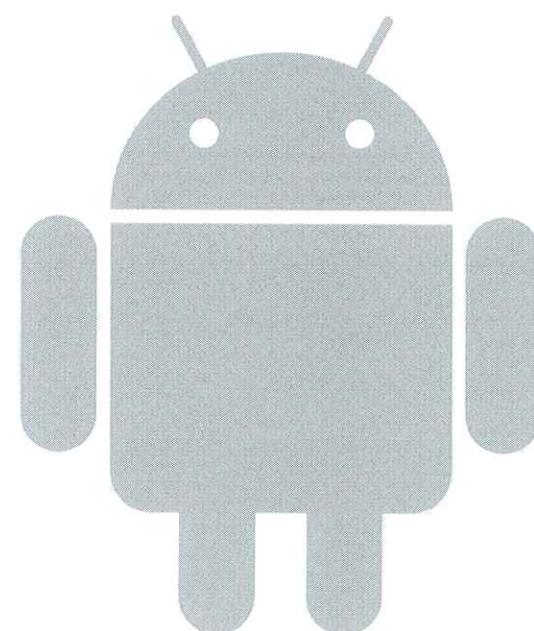
Publicação na loja de aplicativos do Google, Android Market;



Módulo 1

Introdução ao Android

- Visão Geral e Histórico
- O que é o Android?
- O que faz o Android diferente
- Arquitetura
- Aplicações
- Application Framework
- Bibliotecas nativas
- Máquina virtual Dalvik
- Kernel GNU/Linux
- Android Market



Visão Geral e Histórico

Em julho de 2005 o Google adquiriu a Android Inc., uma pequena startup sediada em Palo Alto, Califórnia, EUA. O que se sabia era que a empresa desenvolvia um software baseado em linux tendo como objetivo ser uma plataforma móvel aberta, flexível e de fácil migração por parte dos fabricantes. No Google, a equipe era liderada por Andy Rubin que era co-fundador e CEO da Android Inc. Assim começaram os boatos de que o Google estava planejando entrar no ramo de telefonia móvel.

open handset alliance



Mais especulações de que o Google estaria entrando no mercado de telefonia móvel apareceram em dezembro de 2006, quando a BBC e o Wall Street Journal noticiaram que o Google queria sua busca e aplicações em telefones celulares. A mídia impressa e lojas virtuais começaram com rumores de que o Google estava desenvolvendo um aparelho com sua marca em uma aliança com um fabricante de aparelhos móveis. Logo surgiram protótipos para os fabricantes de telefone celular e operadoras de telefonia.

Em setembro de 2007, o Google já tinha apresentado várias patentes na área de telefonia móvel. Em novembro de 2007, anunciou que estava à frente da OHA, Open Handset Alliance, um consórcio de 65 empresas de hardware, software e telecom dedicadas ao avanço de padrões abertos para dispositivos móveis, disponibilizando o SDK (Software Development Kit) dias depois.

Em Outubro de 2008 foi colocado a venda o primeiro aparelho com o Android 1.0 chamado G1 da HTC. Em fevereiro de 2010 o Google anunciou que 60.000 telefones celulares com Android estão entrando no mercado todos os dias.

O que é o Android?

Android é a primeira plataforma móvel aberta e totalmente personalizável. O Android oferece uma solução completa: um sistema operacional baseado no GNU/Linux customizado para dispositivos móveis, middleware e aplicativos essenciais como um browser para navegar na internet, o Google Maps, player de vídeo e áudio, gerenciador de contatos e ligações entre outros. Contém também um rico conjunto de APIs que permite que desenvolvedores criem suas próprias aplicações.

O Android permite aos desenvolvedores escrever código utilizando a linguagem Java, controlando o dispositivo através bibliotecas Java desenvolvidas pelo próprio Google. Boa parte do código do Android foi liberado publicamente no âmbito da Apache License, sendo assim um software livre com licença de código aberto.

No Android quase tudo é possível, tendo como ponto forte a customização das aplicações nativas do celular, além da troca dessas por versões mais recentes ou modificadas, o que atrai os grandes fabricantes de dispositivos móveis. Atualmente marcas potenciais como HTC, Samsung, Dell, Sony e Motorola estão vendendo celulares Android com interfaces e programas 100% customizados.



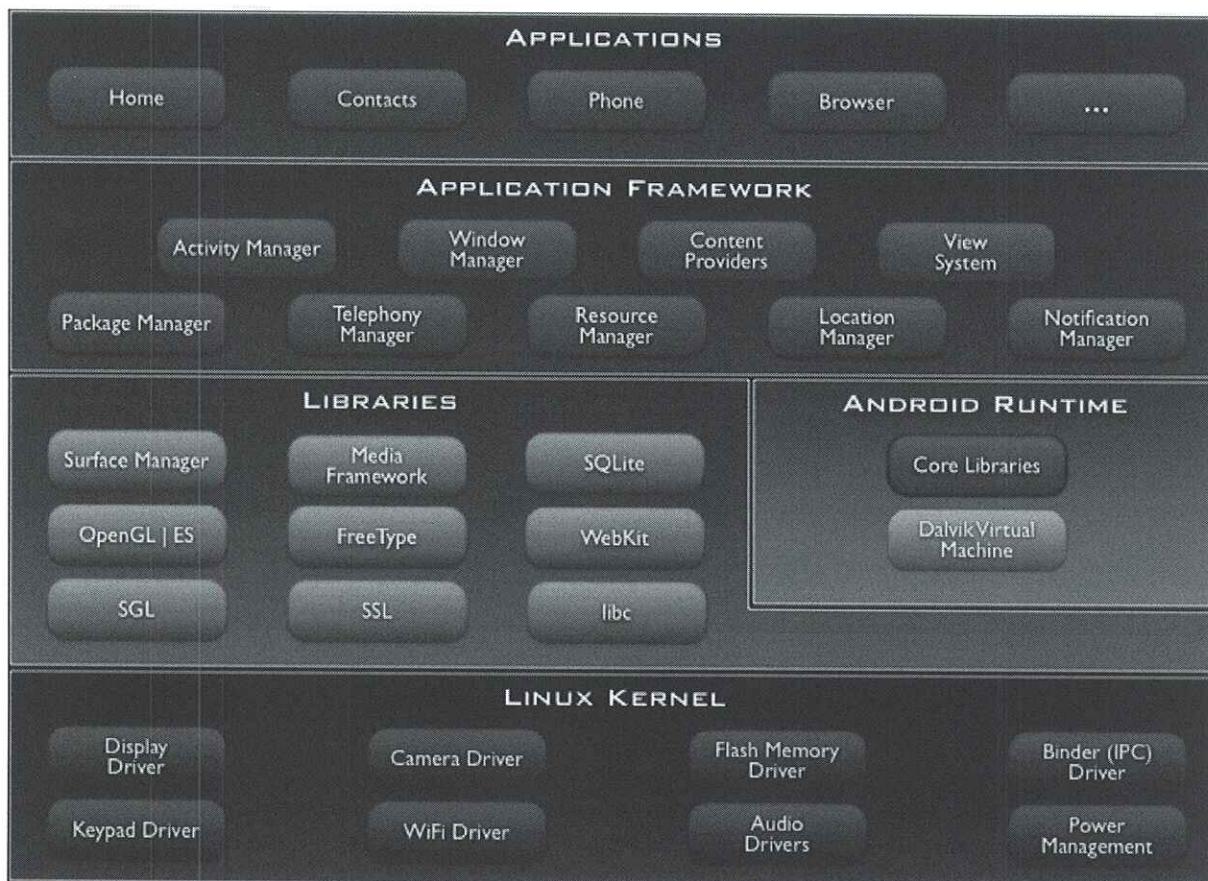
O que faz o Android diferente?

- Framework que permite a reutilização e substituição de componentes;
- Máquina virtual Dalvik otimizada para dispositivos móveis;
- Browser integrado baseado no interpretador open source WebKit;
- Gráficos otimizados através de uma biblioteca de gráficos 2D personalizada;
- Gráficos 3D baseado na especificação OpenGL ES 1.0 (aceleração de hardware opcional);
- SQLite para armazenamento de dados estruturados;
- Suportes aos formatos de áudio e vídeo mais comuns, e ainda os formatos de imagem (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF);
- Telefonia GSM (dependente de hardware);
- Bluetooth, EDGE, 3G e Wi-Fi (dependente de hardware)
- Câmera, GPS, bússola e acelerômetro (dependente de hardware)
- Rico ambiente de desenvolvimento, incluindo um emulador de dispositivos, ferramentas de depuração, perfis de memória e desempenho, e um plugin para o Eclipse;



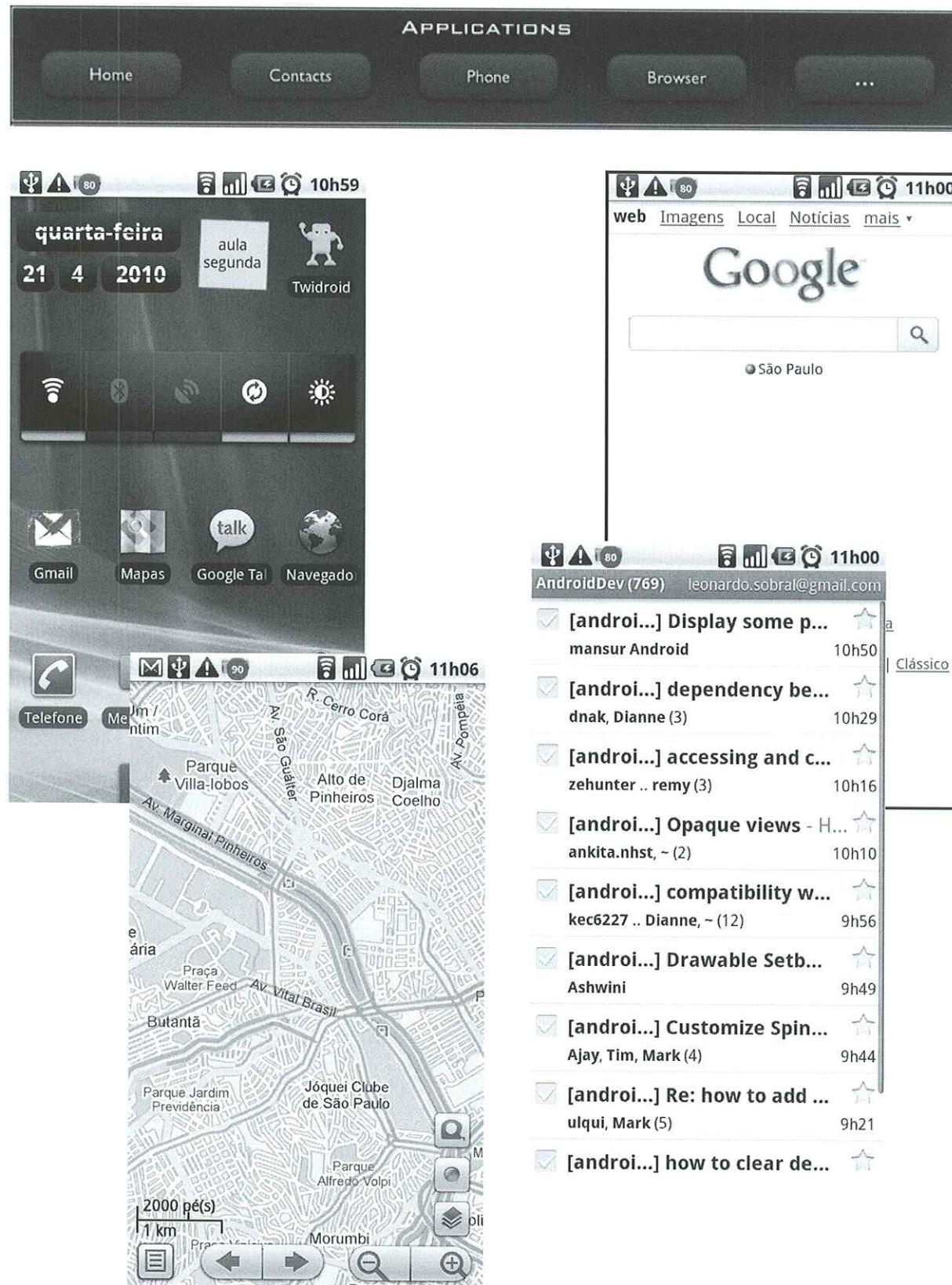
Arquitetura do Android

O diagrama abaixo mostra os principais componentes do sistema operacional Android. Cada seção é descrita em mais detalhes nas próximas páginas.



Aplicações

O Android vem com um conjunto de aplicações, incluindo um cliente de e-mail, programa de SMS, calendário, mapa, navegador, contatos entre outros. Todos os aplicativos são escritos utilizando a linguagem de programação Java.



Application Framework

Ao fornecer uma plataforma de desenvolvimento aberta, o Android oferece aos desenvolvedores a capacidade de construir aplicações extremamente ricas e inovadoras. Os desenvolvedores estão livres para aproveitar ao máximo o hardware do dispositivo, as informações de localização de acesso, execução de serviços em background, definir alarmes, notificações na barra de status, e muito mais.



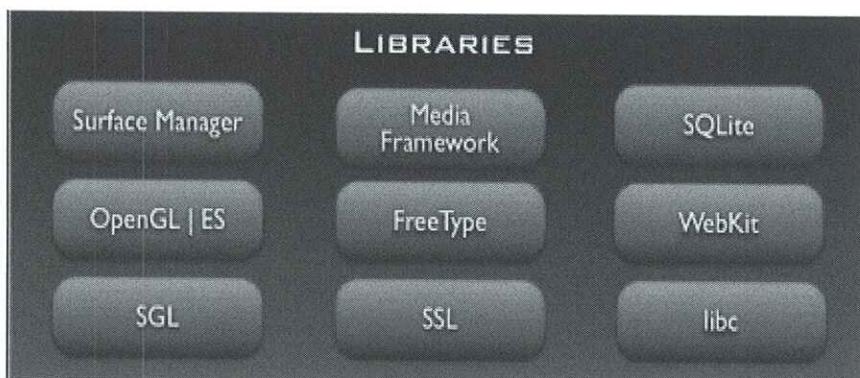
Desenvolvedores tem pleno acesso às mesmas APIs usadas pelos aplicativos nativos. A arquitetura do aplicativo é projetada para simplificar a reutilização dos componentes, qualquer aplicação pode publicar suas capacidades e qualquer outra aplicação pode então fazer uso destas capacidades (sujeito a restrições de segurança impostas pelo framework). Este mesmo mecanismo permite que componentes nativos sejam substituídos pelo usuário.

Abaixo de todas as aplicações existe um conjunto de serviços e sistemas, incluindo:

- Um rico e extensível conjunto de Views que pode ser usado para construir um aplicativo, incluindo listas, tabelas, caixas de texto, botões, e até mesmo um navegador web embutido;
- Os content providers que permitem que aplicativos accedam dados de outros aplicativos (contatos por exemplo), ou compartilhem seus próprios dados;
- Resource Manager, que dá acesso aos recursos não-codificados como seqüências de texto multi-idioma, gráficos e arquivos de layout em XML;
- Notification Manager que permite a todos os aplicativos a exibir alertas personalizados na barra de status;
- Activity Manager que gerencia o ciclo de vida das aplicações e fornece um recurso de navegação;

Bibliotecas Nativas

O Android inclui um conjunto de bibliotecas C e C++ usadas por diversos componentes do sistema. Estas bibliotecas são abertas aos desenvolvedores através de um framework.



Algumas das principais bibliotecas estão listadas abaixo:

- **Biblioteca de Sistema C** - uma implementação da biblioteca padrão C (libc) derivada do BSD, otimizada para dispositivos móveis baseados em Linux;
- **Media Libraries** - baseada na PacketVideo's OpenCORE; as bibliotecas suportam reprodução e gravação dos mais populares formatos de áudio e vídeo, bem como arquivos de imagem, incluindo MPEG4, H.264, MP3, AAC, AMR, JPG e PNG;
- **Surface Manager** - gerencia o acesso ao subsistema display e composição de camadas de gráficos 2D e 3D para múltiplas aplicações;
- **LibWebCore** - um navegador web moderno que alimenta tanto o navegador padrão do Android quanto uma WebView que pode ser embutida em uma aplicação;
- **SGL** - uma engine de gráficos 2D;
- **3D libraries** - uma aplicação baseada nas APIs da OpenGL ES 1.0, as bibliotecas usam tanto aceleração 3D por hardware (quando disponível) ou por software com rasterizador 3D altamente otimizado;
- **FreeType** - renderização de fontes bitmap e vetoriais;
- **SQLite** - um banco de dados relacional leve e potente, disponível para todas as aplicações;

Máquina virtual Dalvik

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

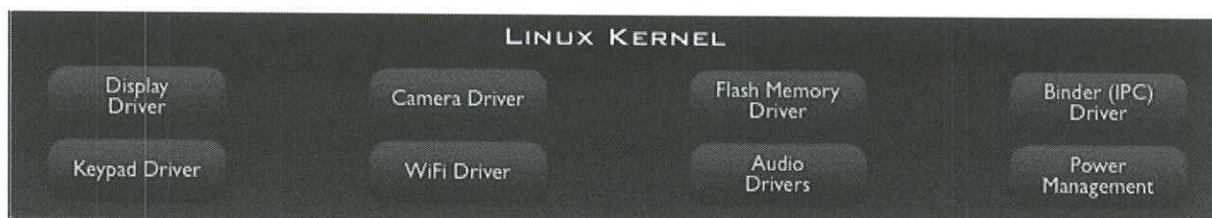


Cada aplicação Android roda em seu próprio processo, com a sua própria instância da máquina virtual Dalvik. A Dalvik foi escrita de modo que um dispositivo pode executar várias VMs eficientemente. A VM Dalvik executa os arquivos em formato executável Dalvik (.DEX) formato que é otimizado para uso mínimo de recursos de hardware. A VM é register-based, ao contrário da maioria de máquinas virtuais e executa classes compiladas por um compilador em linguagem Java que foram transformadas para o formato .dex pela ferramenta "dx".

A VM Dalvik invoca o kernel do Linux para a funcionalidades como threading e gerenciamento de memória de baixo nível.

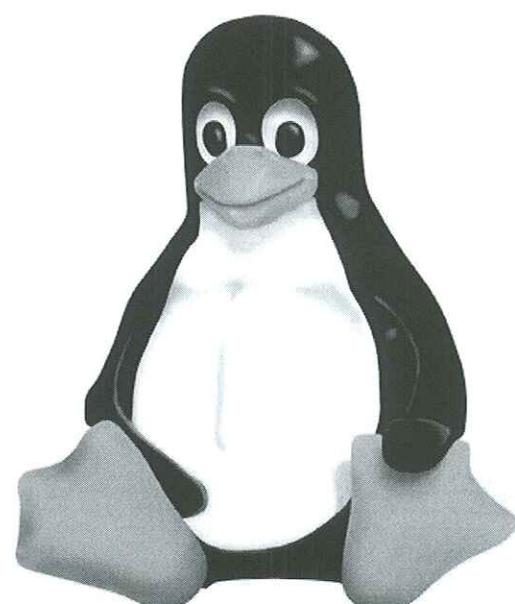
Linux Kernel

O Android é baseado no kernel do Linux versão 2.6 para serviços de sistema tais como segurança, gerenciamento de memória, gerenciamento de processos, rede e drivers. O kernel também atua como uma camada de abstração entre o hardware e o resto do software.



O Linux usado no Android não é uma versão convencional e sim apenas o kernel, modificado pelo Google para adequar às necessidades do Android e separado da árvore principal do kernel Linux. Ele não tem um X Window System nativo nem todas as bibliotecas GNU de sistema. Isso torna um pouco complicado para reutilizar aplicativos e bibliotecas Linux existentes no Android.

O Linux foi utilizado principalmente pelos drivers, gerenciamento de memória e segurança já existentes.



Android Market

Android Market é um serviço hospedado pelo Google que torna mais fácil aos usuários encontrar e baixar aplicativos para seus dispositivos Android. Ele torna fácil para os desenvolvedores publicar as suas aplicações para os usuários do Android.



Introducing the exciting Papaya Farm and Ranch! Farm is a virtual farming game which allows you to sow, grow and harvest in your land. Ranch is an animal raising game to raise all kinds of animals and collect their eggs and milk. You can visit friends and steal their fruits/animals too. Come and try these games today!

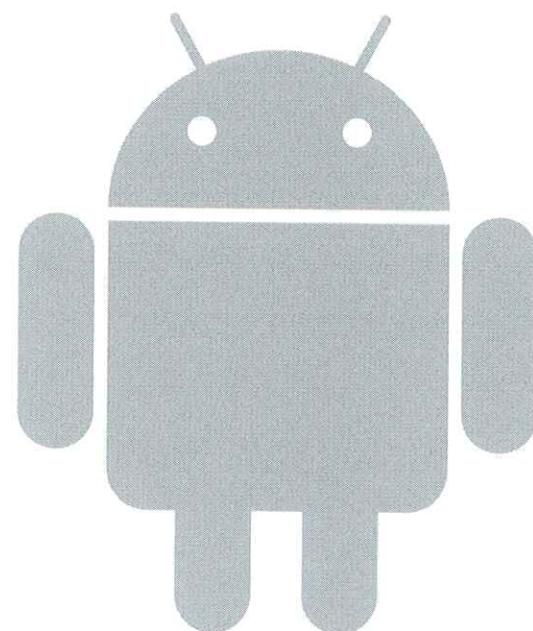
Versão 1.3 2,24MB



Módulo 2

Visão geral do sistema Android

- Home, menu de aplicativos, aplicativos
- Preferências
- Notificações
- Modelo de Segurança

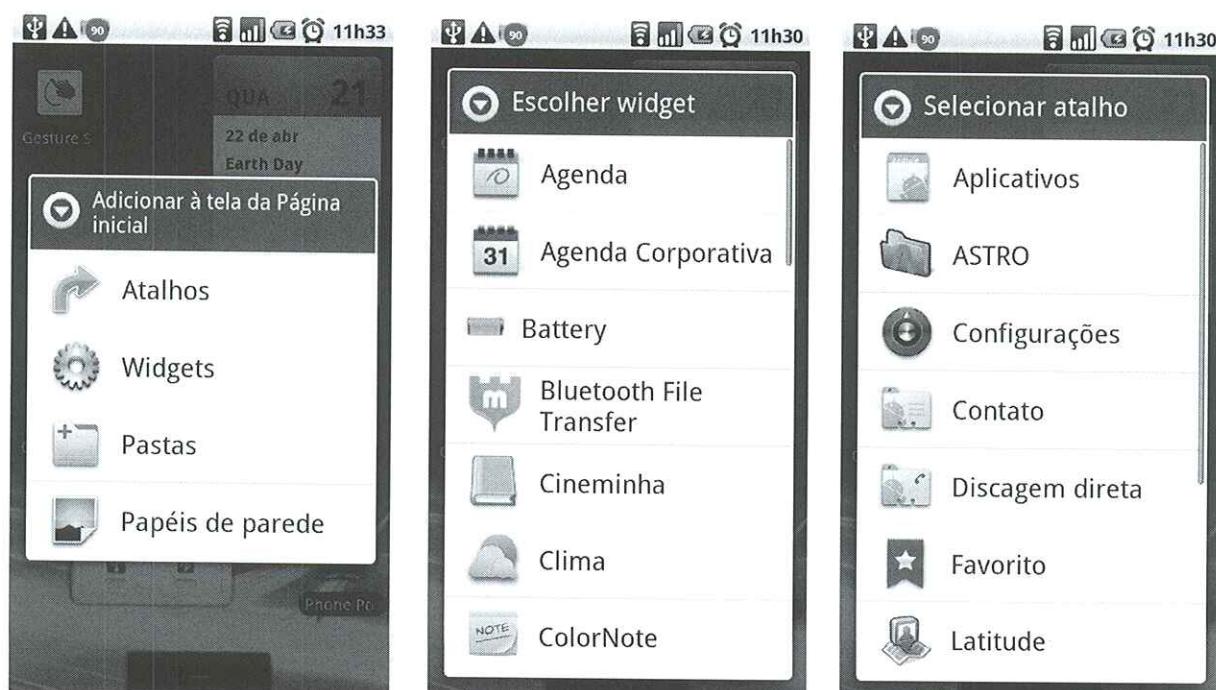


Home, menu de aplicativos, aplicativos

Atribuimos o nome Home à tela inicial do sistema Android, dependendo da versão do sistema podemos ou do aplicativo Home instalado, podemos ter 3 ou mais telas que podem ser visualizadas quando arrastamos para a esquerda ou direita. Na Home, podemos colocar ícones e widgets instalados no Android.



Quando pressionamos a tela e seguramos (long press) um menu contextual é exibido com diversas opções de configuração da Home.

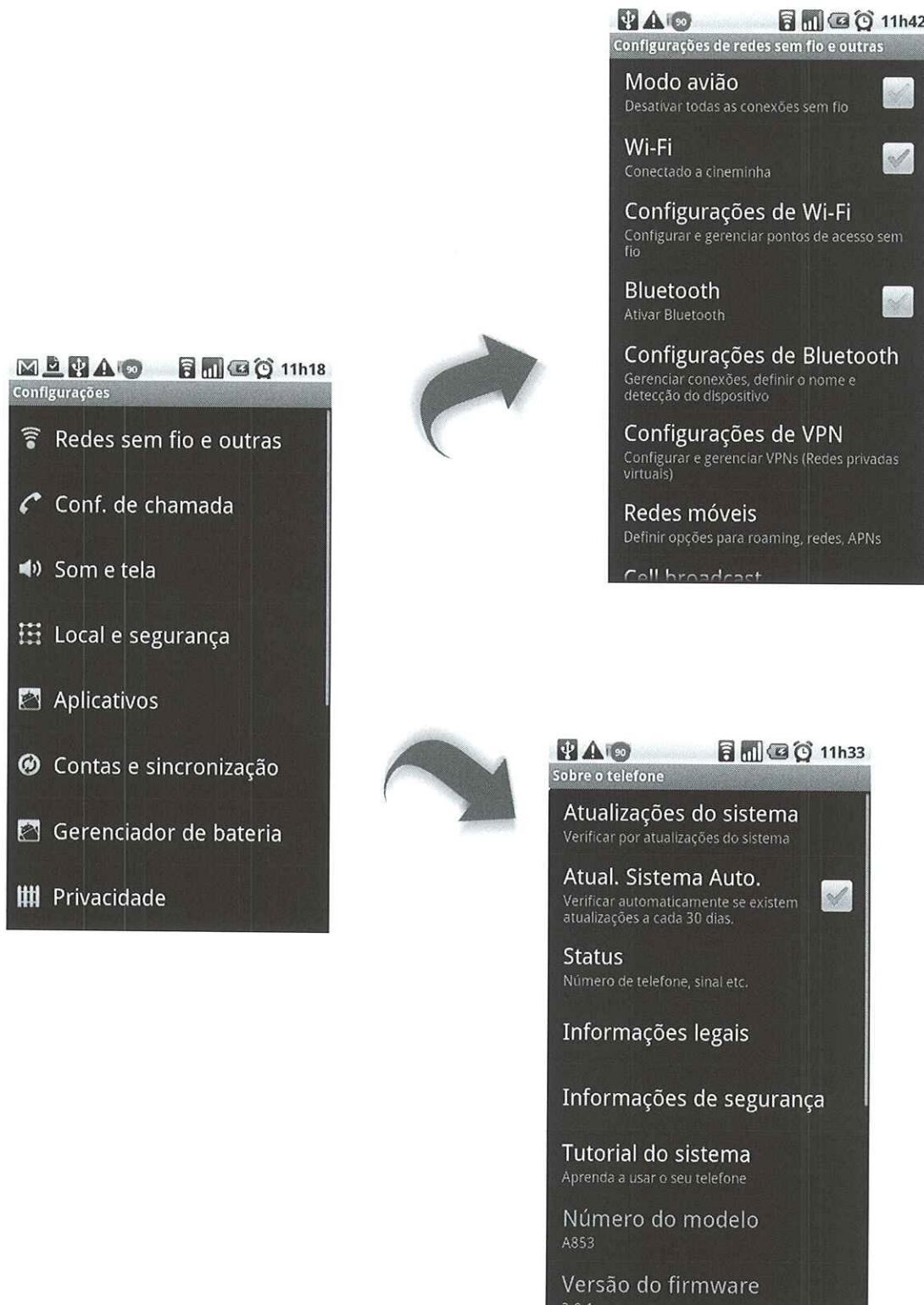


O menu de aplicativos do Android fica originalmente na parte de baixo do dispositivo e quando clicado ou arrastado para cima exibe a lista de aplicativos instalados.



Preferências

Todas as configurações do sistema Android podem ser alteradas através do aplicativo configurações ou pelo menu da Home.



Notificações

A notificação adiciona um ícone na barra de status do sistema (com uma mensagem de texto opcional) e uma mensagem expandida na janela “Notificações”. Quando o usuário seleciona a mensagem expandida, o Android dispara um Intent que é definido pela notificação (geralmente para lançar uma Atividade). Você também pode configurar a notificação para alertar o usuário com um som, uma vibração, e fazer piscar as luzes do dispositivo.

A notificação de barra de status deve ser usada caso um serviço em background precise alertar o usuário sobre um evento que exige uma resposta. Um serviço em background nunca deve iniciar uma atividade por conta própria, a fim de receber a interação do usuário. O serviço deve criar uma notificação de barra de status que vai lançar a atividade quando selecionada pelo usuário.



Modelo de Segurança

O Android trabalha com controle de permissões para controlar o quais recursos do celular as aplicações podem controlar. Ao iniciar a instalação de um novo aplicativo o Android dá ao usuário a opção de dar as permissões solicitadas pelo aplicativo ou interromper a instalação. Quando se está utilizando o emulador para testar suas aplicações as solicitações de permissões não são exibidas pois assume-se que todas as permissões são concedidas.

Ao construir uma aplicação não inclua permissões que não serão utilizadas, isso dará mais credibilidade à sua aplicação... por exemplo, pense como seria estranho uma aplicação que rodará localmente apenas, solicitar permissão de acesso à internet.

Abaixo segue uma lista das permissões possíveis atualmente (retirado de developer.android.com, em inglês):

- ACCESS_CHECKIN_PROPERTIES - Allows read/write access to the "properties" table in the checkin database, to change values that get uploaded.
- ACCESS_COARSE_LOCATION - Allows an application to access coarse (e.g., Cell-ID, WiFi) location
- ACCESS_FINE_LOCATION - Allows an application to access fine (e.g., GPS) location
- ACCESS_LOCATION_EXTRA_COMMANDS - Allows an application to access extra location provider commands
- ACCESS_MOCK_LOCATION - Allows an application to create mock location providers for testing
- ACCESS_NETWORK_STATE - Allows applications to access information about networks
- ACCESS_SURFACE_FLINGER - Allows an application to use SurfaceFlinger's low level features
- ACCESS_WIFI_STATE - Allows applications to access information about Wi-Fi networks
- ACCOUNT_MANAGER - Allows applications to call into AccountAuthenticators.
- AUTHENTICATE_ACCOUNTS - Allows an application to act as an AccountAuthenticator for the AccountManager
- BATTERY_STATS - Allows an application to collect battery statistics
- BIND_APPWIDGET - Allows an application to tell the AppWidget service which application can access AppWidget's data.
- BIND_INPUT_METHOD - Must be required by input method services, to ensure that only the system can bind to them.
- BLUETOOTH - Allows applications to connect to paired bluetooth devices
- BLUETOOTH_ADMIN - Allows applications to discover and pair bluetooth devices

- BRICK - Required to be able to disable the device (very dangerous!).
- BROADCAST_PACKAGE_REMOVED - Allows an application to broadcast a notification that an application package has been removed.
- BROADCAST_SMS - Allows an application to broadcast an SMS receipt notification
- BROADCAST_STICKY - Allows an application to broadcast sticky intents.
- BROADCAST_WAP_PUSH - Allows an application to broadcast a WAP PUSH receipt notification
- CALL_PHONE - Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed.
- CALL_PRIVILEGED - Allows an application to call any phone number, including emergency numbers, without going through the Dialer user interface for the user to confirm the call being placed.
- CAMERA - Required to be able to access the camera device.
- CHANGE_COMPONENT_ENABLED_STATE - Allows an application to change whether an application component (other than its own) is enabled or not.
- CHANGE_CONFIGURATION - Allows an application to modify the current configuration, such as locale.
- CHANGE_NETWORK_STATE - Allows applications to change network connectivity state
- CHANGE_WIFI_MULTICAST_STATE - Allows applications to enter Wi-Fi Multicast mode
- CHANGE_WIFI_STATE - Allows applications to change Wi-Fi connectivity state
- CLEAR_APP_CACHE - Allows an application to clear the caches of all installed applications on the device.
- CLEAR_APP_USER_DATA - Allows an application to clear user data
- CONTROL_LOCATION_UPDATES - Allows enabling/disabling location update notifications from the radio.
- DELETE_CACHE_FILES - Allows an application to delete cache files.
- DELETE_PACKAGES - Allows an application to delete packages.
- DEVICE_POWER - Allows low-level access to power management
- DIAGNOSTIC - Allows applications to RW to diagnostic resources.
- DISABLE_KEYGUARD - Allows applications to disable the keyguard
- DUMP - Allows an application to retrieve state dump information from system services.
- EXPAND_STATUS_BAR - Allows an application to expand or collapse the status bar.

- FACTORY_TEST - Run as a manufacturer test application, running as the root user.
- FLASHLIGHT - Allows access to the flashlight
- FORCE_BACK - Allows an application to force a BACK operation on whatever is the top activity.
- GET_ACCOUNTS - Allows access to the list of accounts in the Accounts Service
- GET_PACKAGE_SIZE - Allows an application to find out the space used by any package.
- GET_TASKS - Allows an application to get information about the currently or recently running tasks: a thumbnail representation of the tasks, what activities are running in it, etc.
- GLOBAL_SEARCH - This permission can be used on content providers to allow the global search system to access their data.
- HARDWARE_TEST - Allows access to hardware peripherals.
- INJECT_EVENTS - Allows an application to inject user events (keys, touch, trackball) into the event stream and deliver them to ANY window.
- INSTALL_LOCATION_PROVIDER - Allows an application to install a location provider into the Location Manager
- INSTALL_PACKAGES - Allows an application to install packages.
- INTERNAL_SYSTEM_WINDOW - Allows an application to open windows that are for use by parts of the system user interface.
- INTERNET - Allows applications to open network sockets.
- MANAGE_ACCOUNTS - Allows an application to manage the list of accounts in the AccountManager
- MANAGE_APP_TOKENS - Allows an application to manage (create, destroy, Z-order) application tokens in the window manager.
- MASTER_CLEAR - MODIFY_AUDIO_SETTINGS - Allows an application to modify global audio settings
- MODIFY_PHONE_STATE - Allows modification of the telephony state - power on, mmi, etc.
- MOUNT_FORMAT_FILESYSTEMS - Allows formatting file systems for removable storage.
- MOUNT_UNMOUNT_FILESYSTEMS - Allows mounting and unmounting file systems for removable storage.
- PERSISTENT_ACTIVITY - Allow an application to make its activities persistent.
- PROCESS_OUTGOING_CALLS - Allows an application to monitor, modify, or abort outgoing calls.
- READ_CALENDAR - Allows an application to read the user's calendar data.

- READ_CONTACTS - Allows an application to read the user's contacts data.
- READ_FRAME_BUFFER - Allows an application to take screen shots and more generally get access to the frame buffer data
- READ_HISTORY_BOOKMARKS - Allows an application to read (but not write) the user's browsing history and bookmarks.
- READ_INPUT_STATE - Allows an application to retrieve the current state of keys and switches.
- READ_LOGS - Allows an application to read the low-level system log files.
- READ_OWNER_DATA - Allows an application to read the owner's data.
- READ_PHONE_STATE - Allows read only access to phone state.
- READ_SMS - Allows an application to read SMS messages.
- READ_SYNC_SETTINGS - Allows applications to read the sync settings
- READ_SYNC_STATS - Allows applications to read the sync stats
- REBOOT - Required to be able to reboot the device.
- RECEIVE_BOOT_COMPLETED - Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting.
- RECEIVE_MMS - Allows an application to monitor incoming MMS messages, to record or perform processing on them.
- RECEIVE_SMS - Allows an application to monitor incoming SMS messages, to record or perform processing on them.
- RECEIVE_WAP_PUSH - Allows an application to monitor incoming WAP push messages.
- RECORD_AUDIO - Allows an application to record audio
- REORDER_TASKS - Allows an application to change the Z-order of tasks
- RESTART_PACKAGES - Allows an application to restart other applications.
- SEND_SMS - Allows an application to send SMS messages.
- SET_ACTIVITY_WATCHER - Allows an application to watch and control how activities are started globally in the system.
- SET_ALWAYS_FINISH - Allows an application to control whether activities are immediately finished when put in the background.
- SET_ANIMATION_SCALE - Modify the global animation scaling factor.
- SET_DEBUG_APP - Configure an application for debugging.

- SET_ORIENTATION - Allows low-level access to setting the orientation (actually rotation) of the screen.
- SET_PREFERRED_APPLICATIONS - This constant is deprecated. No longer useful, see addPackageToPreferred() for details.
- SET_PROCESS_LIMIT - Allows an application to set the maximum number of (not needed) application processes that can be running.
- SET_TIME_ZONE - Allows applications to set the system time zone
- SET_WALLPAPER - Allows applications to set the wallpaper
- SET_WALLPAPER_HINTS - Allows applications to set the wallpaper hints
- SIGNAL_PERSISTENT_PROCESSES - Allow an application to request that a signal be sent to all persistent processes
- STATUS_BAR - Allows an application to open, close, or disable the status bar and its icons.
- SUBSCRIBED_FEEDS_READ - Allows an application to allow access the subscribed feeds ContentProvider.
- SUBSCRIBED_FEEDS_WRITE
- SYSTEM_ALERT_WINDOW - Allows an application to open windows using the type TYPE_SYSTEM_ALERT, shown on top of all other applications.
- UPDATE_DEVICE_STATS - Allows an application to update device statistics.
- USE_CREDENTIALS - Allows an application to request authtokens from the AccountManager
- VIBRATE - Allows access to the vibrator
- WAKE_LOCK - Allows using PowerManager WakeLocks to keep processor from sleeping or screen from dimming
- WRITE_APN_SETTINGS - Allows applications to write the apn settings
- WRITE_CALENDAR - Allows an application to write (but not read) the user's calendar data.
- WRITE_CONTACTS - Allows an application to write (but not read) the user's contacts data.
- WRITE_EXTERNAL_STORAGE - Allows an application to write to external storage
- WRITE_GSERVICES - Allows an application to modify the Google service map.
- WRITE_HISTORY_BOOKMARKS - Allows an application to write (but not read) the user's browsing history and bookmarks.
- WRITE_OWNER_DATA - Allows an application to write (but not read) the owner's data.
- WRITE_SECURE_SETTINGS - Allows an application to read or write the secure system settings.

- WRITE_SETTINGS - Allows an application to read or write the system settings.
- WRITE_SMS - Allows an application to write SMS messages.
- WRITE_SYNC_SETTINGS - Allows applications to write the sync settings

Módulo 3

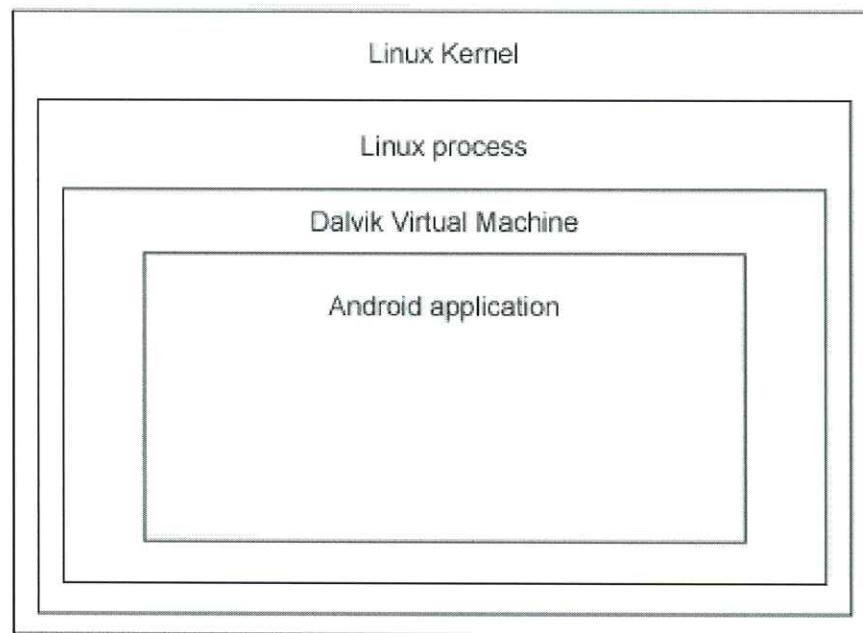
Visão geral do SDK

- Plataformas
- Versões do SDK
- Ferramentas de desenvolvimento (Eclipse com ADT, MotoDev Studio, Elips)



Plataformas

A plataforma do Android utiliza a linguagem de programação Java, embora os bytecodes convertidos não sejam bytecodes JVM e sim bytecodes Dalvik. Utiliza como base o kernel do GNU/Linux 2.6. Obviamente diversas mudanças foram feitas no kernel de modo que o sistema ficasse mais leve e totalmente adaptado a dispositivos móveis.



Exemplo de Java no Android:

```
package com.leonardosobral;

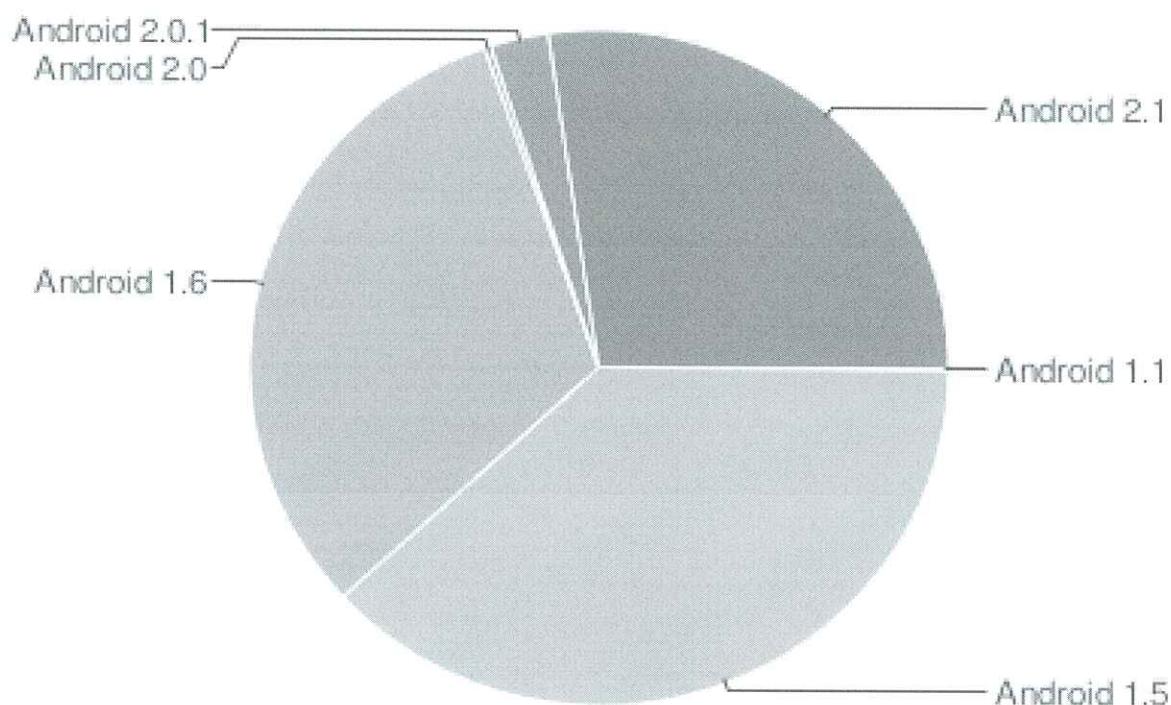
import android.app.Activity;
import android.os.Bundle;

public class HelloWorld extends Activity {

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

Versões do SDK

Atualmente o SDK do Android encontra-se na versão 2.1 (codinome froyo), entretanto nem todos os celulares Android utilizam essa versão, mesmo porque até o momento ela é homologada para um número muito pequeno de aparelhos. As versões mais comuns atualmente são a 1.5 (codinome cupcake) e 1.6 (codinome donut).



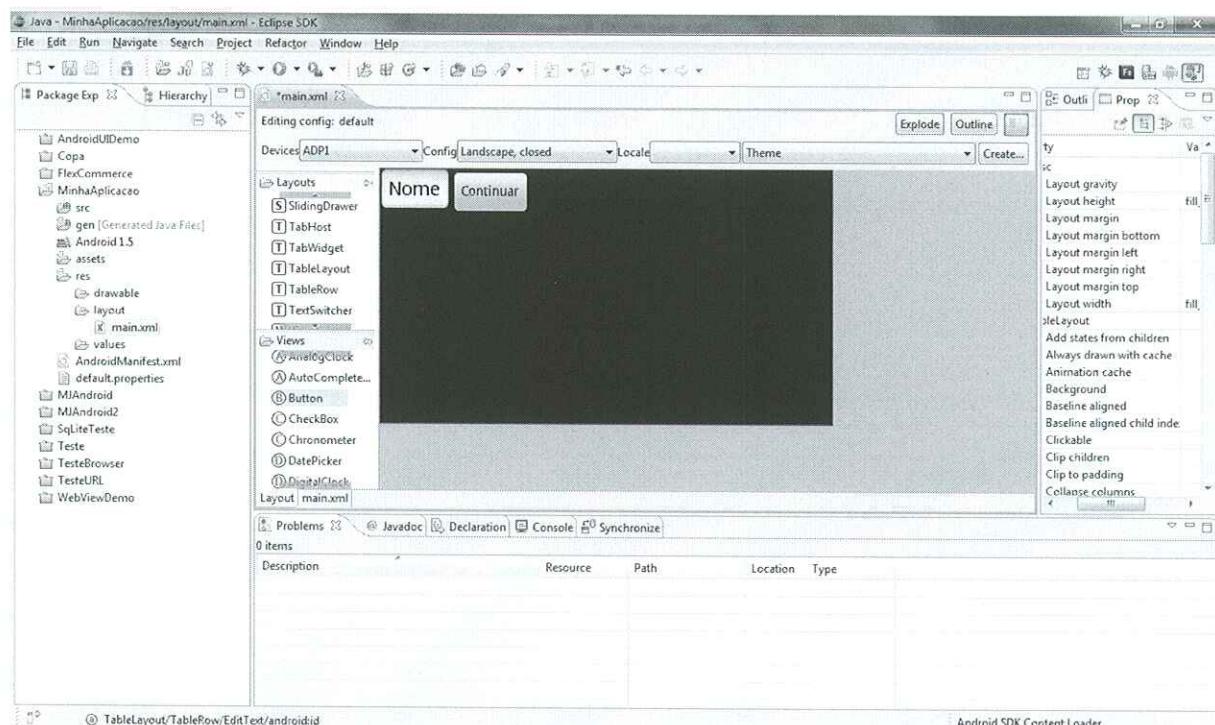
O SDK do Android pode ser baixado pelo endereço:

[http://developer.android.com/sdk/.](http://developer.android.com/sdk/)

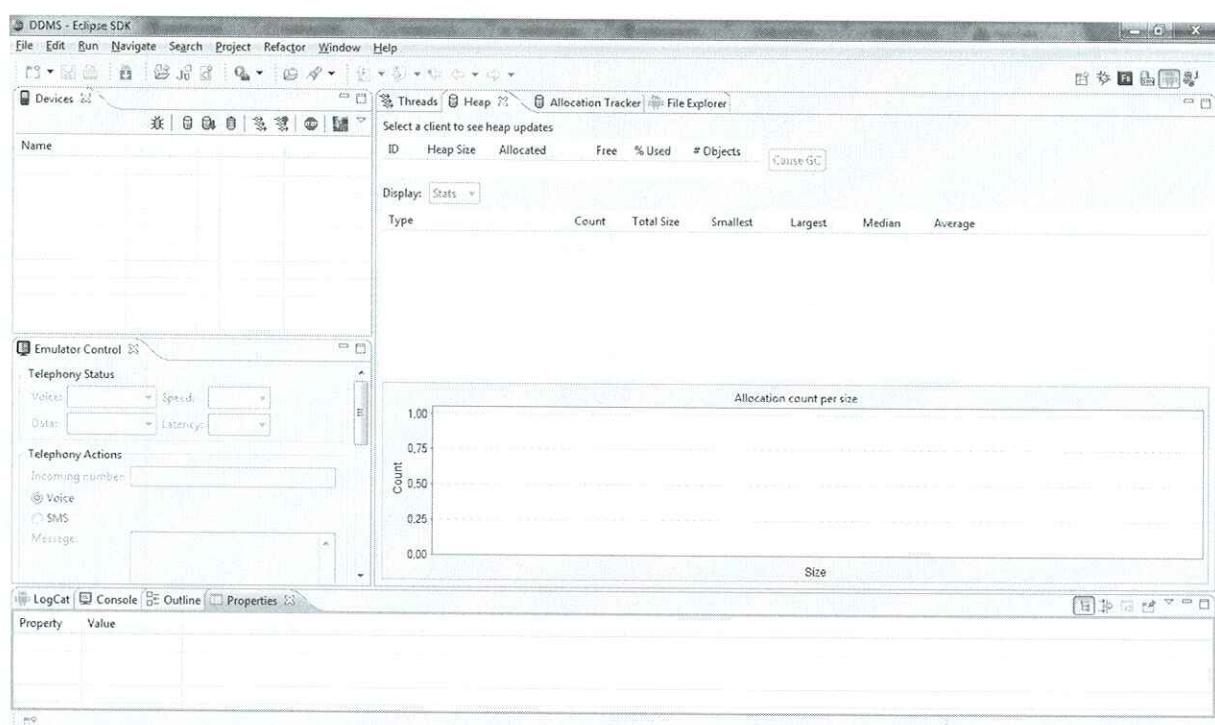
A instalação do SDK é coberta com detalhes no próximo módulo.

Ferramentas de desenvolvimento

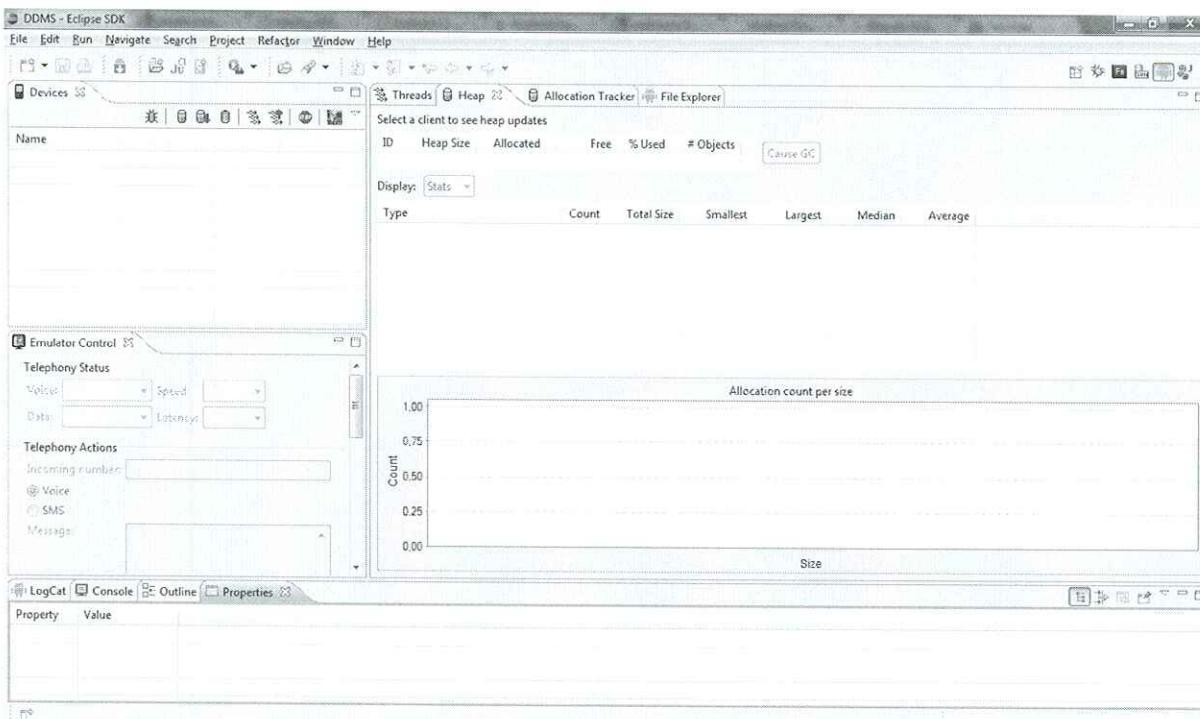
Diversas IDE's (Ambientes Integrados de Desenvolvimento) suportam plugins de desenvolvimento para o Android. Dentre eles estão os famosos Eclipse e NetBeans.



A Empresa Motorola lançou recentemente um plugin para o Eclipse, visando facilitar algumas ações padrões ao criar um programa para Android... A esse plugin ela deu o nome de MotoDev Studio.



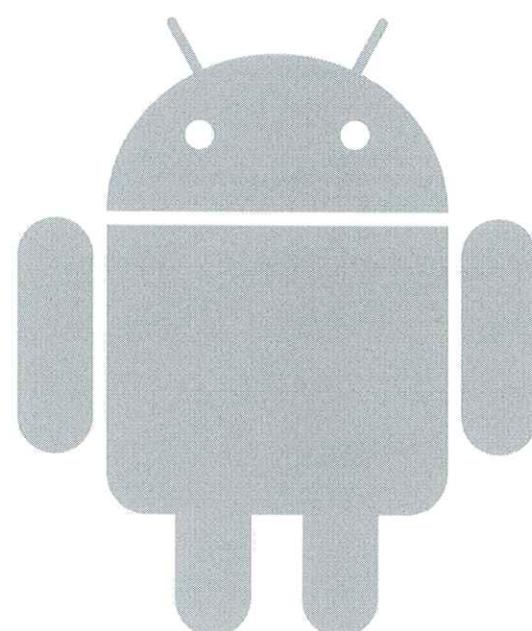
Cada um possui sua particularidade, mas por questões de melhor compatibilidade e maior suporte na comunidade, utilizaremos durante todo o curso o plugin ADT do Google para Eclipse.



Módulo 4

Preparação do ambiente de desenvolvimento

- Visão geral do Eclipse
- Requisitos do Sistema
- Instalação do Eclipse
- Instalando o SDK Starter Package
- Instalando o plugin ADT no Eclipse
- Configuração do plugin ADT no Eclipse
- Instalando diferentes versões do SDK



Visão geral do Eclipse

O plugin Android Development Tools (ADT) para o Eclipse adiciona poderosas extensões para o ambiente de desenvolvimento integrado Eclipse. Ele permite que você crie e depure aplicações Android de uma forma fácil e rápida. Se você já é usuário do Eclipse, o plugin ADT dá-lhe um impulso incrível no desenvolvimento de aplicativos Android.

Entre eles podemos citar:

- Dá acesso a outras ferramentas de desenvolvimento do Android de dentro do Eclipse. Por exemplo, o ADT permite acesso a muitos recursos da ferramenta DDMS: tirar screenshots, gerenciar o encaminhamento da porta, definir break points, e ver threads e informações de processos direto do Eclipse.
- Fornece wizard para a criação de projetos, que lhe ajuda rapidamente a criar e configurar todos os arquivos básicos que você precisa para uma nova aplicação Android.
- Automatiza e simplifica o processo de construção de sua aplicação Android.
- Fornece um editor de código que ajuda a você a escrever XML válido para o manifest e arquivos de layout.
- Exporta seu projeto em um APK assinado, que pode ser distribuído para os usuários através do market.

Para começar a desenvolver aplicações Android no Eclipse com ADT, você primeiro precisa baixar o Eclipse, fazer o download do plugin ADT e instalar. Siga os passos do seu instrutor para a correta instalação do ambiente.

Sempre procure utilizar a última versão disponível do plugin ADT, ele está disponível no site do desenvolvedor Android.

Requisitos de Sistema

Abaixo seguem os requisitos de sistema e software para desenvolver aplicações Android utilizando o Android SDK.

Sistema Operacional

Windows XP (32 bits) ou Vista (32 ou 64-bit)

Mac OS X 10.5.8 ou posterior (somente Intel x86)

Linux (testado no Linux Ubuntu Hardy Heron)

-Distribuições de 64 bits devem ser capazes de executar aplicativos de 32 bits.

Eclipse IDE

Eclipse 3.4 Ganymede ou 3.5 Galileo

Eclipse JDT Plugin (incluído na maioria dos pacotes do Eclipse). As distribuições recomendadas são:

- Eclipse IDE para desenvolvedores Java EE
- Eclipse IDE for Java Developers
- Eclipse para RCP / Plug-in Developers
- Eclipse Classic (versões 3.5.1 e superiores)

JDK 5 ou JDK 6 (JRE não é suficiente)

Android Development Tools plugin

Instalação do Eclipse

Se o Eclipse já está instalado no seu computador, certifique-se que é uma versão compatível com o ADT e o SDK do Android.

Para instalar ou atualizar o Eclipse, você pode visitar este endereço:

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

The screenshot shows the Eclipse Downloads page at <http://www.eclipse.org/downloads/>. The page features a navigation bar with links to Home, Downloads, Users, Members, Committers, Resources, Projects, and About Us. A search bar is also present. The main content area is titled "Eclipse Downloads" and contains several software packages listed in a grid:

Category	Name	Description	Downloads	Platform
Galileo Packages (based on Eclipse 3.5 SR2) - Compare Packages	Eclipse IDE for Java EE Developers (190 MB)	Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn and others. More...	1,160,551	Windows 32bit Mac Carbon 32bit Mac Cocoa 32bit 64bit Linux 32bit 64bit
	Eclipse IDE for Java Developers (92 MB)	The essential tools for any Java developer, including a Java IDE, a CVS client, XML Editor and Mylyn. More...	468,023	Windows 32bit Mac Carbon 32bit Mac Cocoa 32bit 64bit Linux 32bit 64bit
	Eclipse IDE for C/C++ Developers (79 MB)	An IDE for C/C++ developers with Mylyn integration. More...	209,499	Windows 32bit Mac Carbon 32bit Mac Cocoa 32bit 64bit Linux 32bit 64bit
	Eclipse for PHP Developers (139 MB)	Tools for PHP developers creating Web applications, including PHP Development Tools (PDT), Web Tools Platform, Mylyn and others. More...	180,051	Windows 32bit Mac Carbon 32bit Mac Cocoa 32bit 64bit Linux 32bit 64bit
	Eclipse IDE for Java and Report Developers (221 MB)	JEE tools and BIRT reporting tool for Java developers to create JEE and Web applications that also have reporting needs. More...	40,375	Windows 32bit Mac Carbon 32bit Mac Cocoa 32bit 64bit Linux 32bit 64bit
	Eclipse Modeling Tools (includes Incubating components) (372 MB)	This modeling package contains a collection of Eclipse Modeling Project components, including EMF, GMF, MDT XSD/OCL/UML2, M2M, M2T, and EMFT elements. It includes a complete SDK, developer tools and source code. Note that the Modeling package includes some incubating components, as indicated by feature numbers less than 1.0.0 on the feature list. More...	33,579	Windows 32bit Mac Carbon 32bit Mac Cocoa 32bit 64bit Linux 32bit 64bit

On the right side of the page, there are promotional banners for "Jazz" (Innovation through Collaboration) and "IBM" (Learn more about Jazz and Rational Team Concert). Below these is a banner for the "Eclipse Training Series" (April 12 - May 28 RCP • OSGi • Modeling). At the bottom right, there is a link to "Popular projects (Apr 6/10)".

Instalando o SDK Starter Package

Antes que você possa configurar ou usar o ADT, você deve instalar o pacote Android SDK starter.

O Starter SDK não é um ambiente de desenvolvimento completo, inclui apenas o SDK Tools, que você pode usar para baixar o resto dos componentes do SDK.

Você pode obter a versão mais recente do pacote Starter SDK na página de download do SDK:

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

Após o download, descompacte o arquivo do Android SDK para uma pasta na sua máquina. Por padrão, os arquivos SDK são descompactados em um diretório chamado *android-sdk-<machine-platform>*.

Seria interessante adicionar a pasta de localização do SDK na variável de sistema PATH. A pasta /tools está localizada na raiz da pasta do SDK. Adicionando ao Path você não precisa entrar com todo o caminho no caso de usar o Android Debug Bridge (adb) por exemplo.

Neste curso usaremos como padrão o sistema Windows, para Mac ou Linux, acesse: <http://developer.android.com/sdk/installing.html#Installing>, para instruções sobre como configurar o Path.

No Windows, clique com o botão direito em "Meu Computador" e selecione Propriedades. Na aba Avançado, pressione o botão Variáveis de Ambiente, e na janela que aparecer, clique duas vezes no caminho (em Variáveis do Sistema). Adicione o caminho completo da pasta /tools.

Instalando o plugin ADT no Eclipse

Android Development Tools (ADT) é um plugin para o Eclipse que é projetado para lhe proporcionar um integrado e poderoso ambiente para a criação de aplicativos Android.

O ADT amplia os recursos do Eclipse incluindo a criação rápida de projetos Android, montagem da interface do usuário, uso de componentes baseados na API do Android, depuração do código usando as ferramentas do SDK, e até mesmo exportação de aplicativos assinados ou não, a fim de distribuir sua aplicação.

Vamos ver um passo a passo de como fazer o download do plugin ADT e instalá-lo em seu ambiente de desenvolvimento Eclipse. Note-se que antes de instalar ou usar ADT, você deve ter versões compatíveis tanto da IDE Eclipse e SDK Android instalado.

Baixar o plugin ADT

Use o Update Manager do Eclipse para instalar a última revisão do ADT em seu computador.

No Eclipse 3.4 (Ganymede)

1. No Eclipse, selecione Help > Software Updates.... Na janela que aparecer, clique na aba Available Software.
2. Clique Add Site...
3. Na caixa de diálogo, digite esta URL no campo "Location":
<https://dl-ssl.google.com/android/eclipse/>
Nota: Se você tiver problemas para baixar o plugin, tente usar "http" na URL, ao invés de "https" (https é preferido por razões de segurança).
Clique em OK.
4. Voltar na exibição Available Software, você verá o plugin listado pela URL, com o "Developer Tools" aninhado dentro dele. Selecione a caixa de seleção ao lado de Developer Tools e clique em Install ...
5. Na janela de instalação posterior, "Android DDMS" e "Android Development Tools" ambos devem ser checados. Clique em Next.

6. Leia e aceite o contrato de licença e clique em Finish.

7. Reinicie o Eclipse.

No Eclipse 3.5 (Galileo)

1. No Eclipse, selecione Help > Install New Software...

2. Clique Add...

3. Na caixa de diálogo, digite esta URL no campo "Location":

<https://dl-ssl.google.com/android/eclipse/>

Nota: Se você tiver problemas para baixar o plugin, tente usar "http" na URL, ao invés de "https" (https é preferido por razões de segurança).

Clique em OK.

4. Voltar na exibição de Available Software, você verá o plugin listado pela URL, com o "Developer Tools" aninhado dentro dele. Selecione a caixa de seleção ao lado de Developer Tools e clique em Install ...

5. Na janela de instalação posterior, "Android DDMS" e "Android Development Tools" ambos devem ser checados. Clique em Next.

6. Leia e aceite o contrato de licença e clique em Finish.

7. Reinicie o Eclipse.

Configuração do plugin ADT no Eclipse

O próximo passo é modificar as preferências do ADT no Eclipse para apontar para o diretório do SDK do Android:

1. Selecione Window > Preferences ... para abrir o painel de Preferências
2. Selecione Android no painel esquerdo.
3. Para SDK Location no painel principal, clique em Browse ... e localize a pasta do SDK.
4. Clique em Apply e depois em OK.

Instalando diferentes versões do SDK

O último passo na criação de seu SDK é usando uma ferramenta incluída no SDK starter package (o Android SDK e AVD Manager) para baixar componentes essenciais para seu ambiente de desenvolvimento.

O SDK usa uma estrutura modular que separa os componentes principais do SDK (versões da plataforma Android, add-ons, ferramentas, exemplos e documentação da API) em um conjunto de componentes instaláveis separadamente.

SDK starter package inclui apenas um único componente: a versão mais recente das ferramentas do SDK. Para desenvolver qualquer aplicação Android, você também precisará fazer o download, pelo menos, uma plataforma Android em seu ambiente, embora a baixar componentes adicionais seja altamente recomendado.

O repositório SDK oferece esses tipos de componentes:

SDK Tools (pré-instalado no SDK starter package) - Contém o conjunto completo de ferramentas do SDK para o desenvolvimento, depuração e teste.

Plataformas Android - Cada componente inclui uma biblioteca Android e imagem do sistema, código de exemplo, skins do emulador, e qualquer ferramenta específica da versão.

SDK Add-ons - Proporciona um ambiente de desenvolvimento para uma biblioteca externa do Android específica ou uma imagem do sistema Android personalizada (mas totalmente compatível).

Driver USB para Windows - Contém drivers que você pode instalar no seu Windows, para que você possa executar e depurar seus aplicativos em um dispositivo real.

Samples - Contém código de exemplo e aplicativos disponíveis para cada plataforma de desenvolvimento do Android.

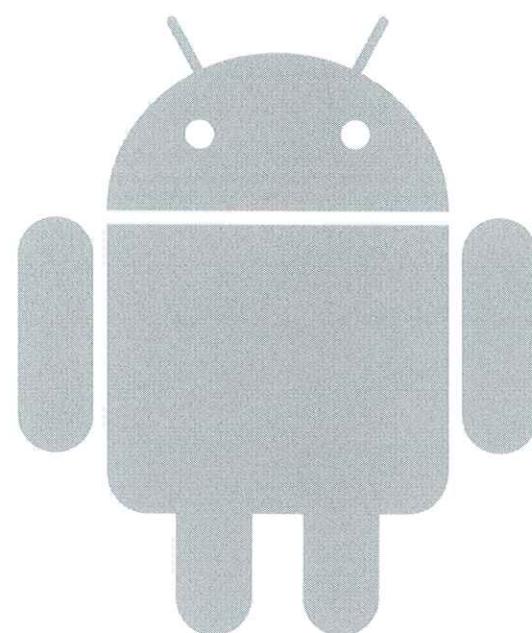
Documentação - Contém uma cópia da documentação mais recente da API.

Para baixar os componentes, use a interface gráfica do Android SDK e AVD Manager para procurar o repositório SDK, e depois instale os componentes selecionados em seu ambiente.

Módulo 5

Criando uma sua primeira aplicação (hello world)

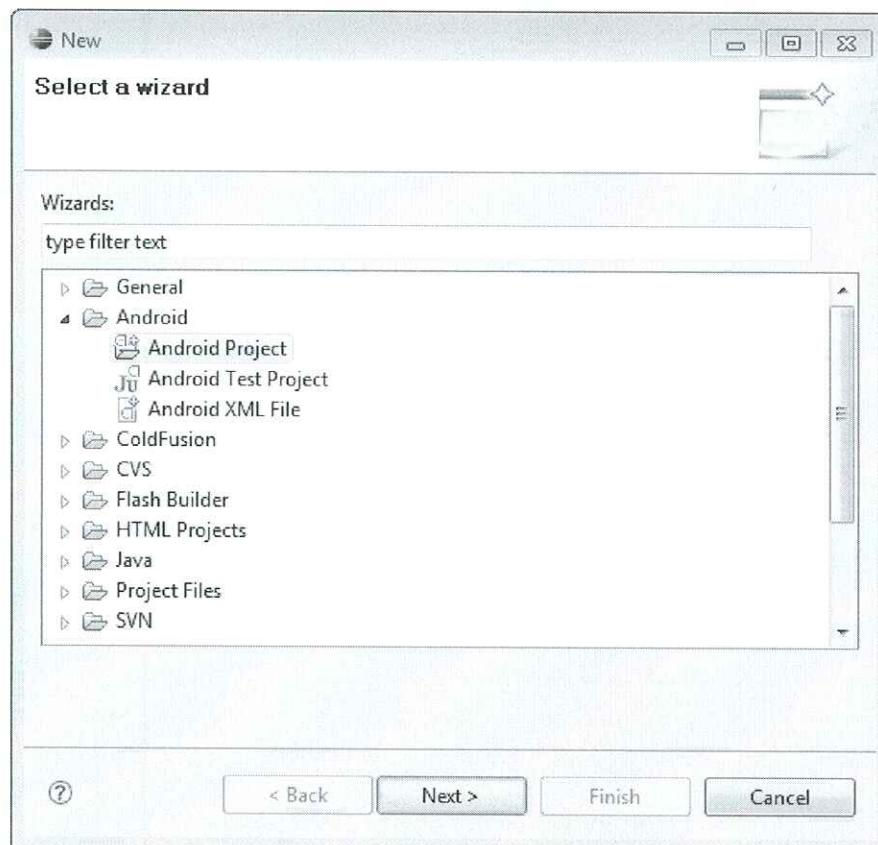
- Criação do projeto
- O arquivo de manifest
- Layout com XML
- Executando seu aplicativo no emulador



Criação do projeto

O plugin ADT fornece um novo assistente de projeto que você pode usar para criar um projeto Android rapidamente. Para criar um novo projeto:

1. Selecione File > New > Project.
2. Selecione Android > Android Project e clique em Next.
3. Dê um nome do projeto. Este será o nome da pasta onde o projeto será criado.

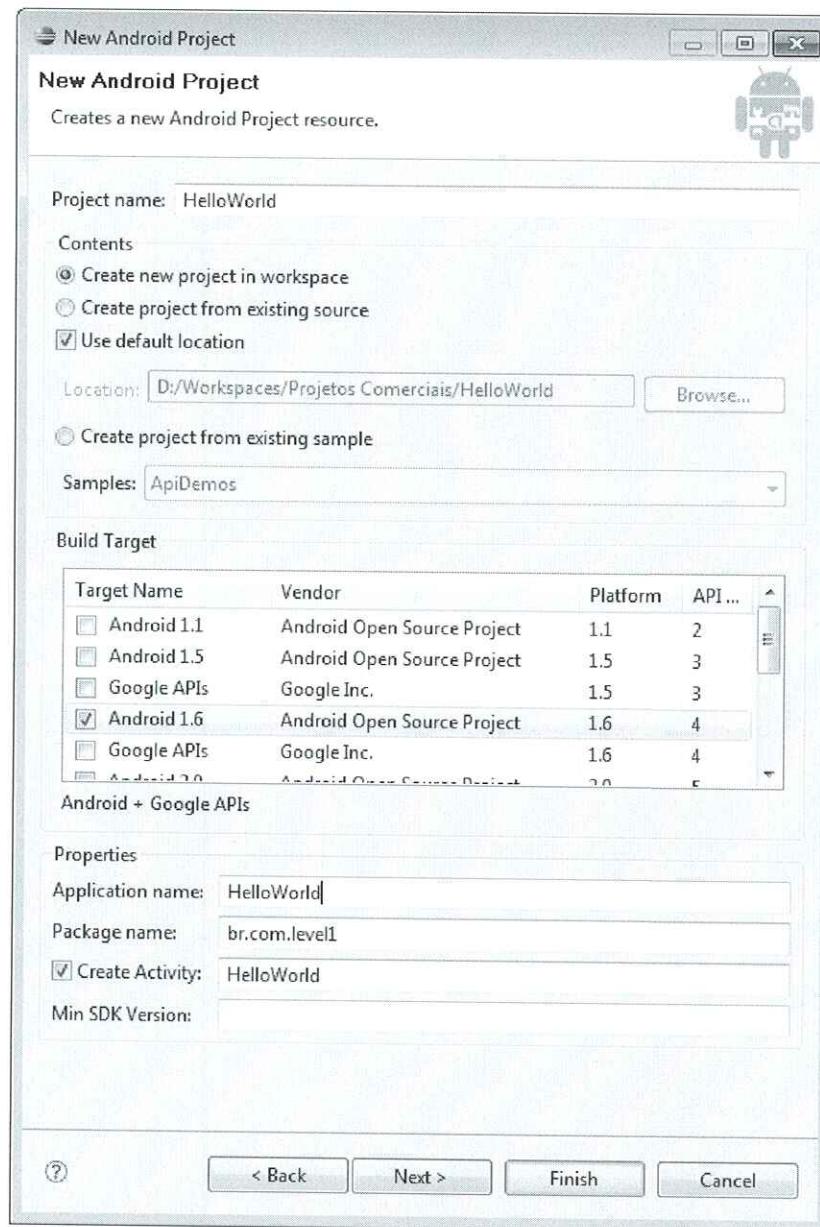


4. Selecione "Create a new project in workspace". Selecione "use default location", ele irá usar o workspace do projeto como local padrão.
5. Em target, selecionar uma versão do SDK do Android para ser usado como "Build Target". Target especifica qual plataforma Android você deseja portar seu aplicativo.

A menos que você queira usar componentes específicos da última versão do Android, você deve selecionar como target a versão de plataforma mais baixa possível, como o Android 1.1 ou 1.5.

Nota: Você pode mudar o target do seu projeto a qualquer momento: clique com o botão direito no projeto no Package Explorer, selecione Properties, selecione Android e, em seguida, mude o target.

6. Em Propriedades, preencha todos os campos necessários.



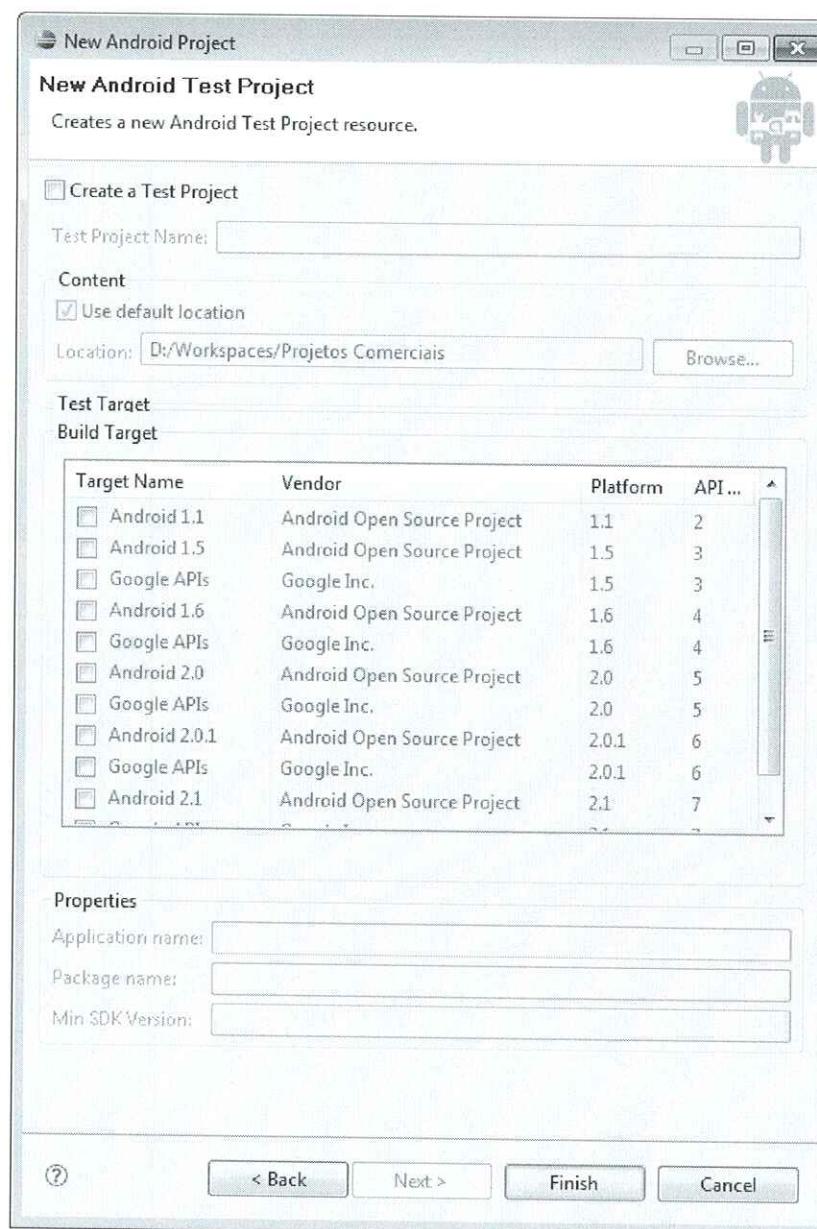
Entre com um nome para o aplicativo . Este é o título da sua aplicação, é o nome que aparecerá no dispositivo Android.

Package Name, digite um nome de pacote (seguindo as mesmas regras para os pacotes na linguagem de programação Java), onde todas as fontes de seu código irão residir.

Selecione Create Activity (opcional, claro, mas comum) e digite um nome para sua classe de atividade principal.

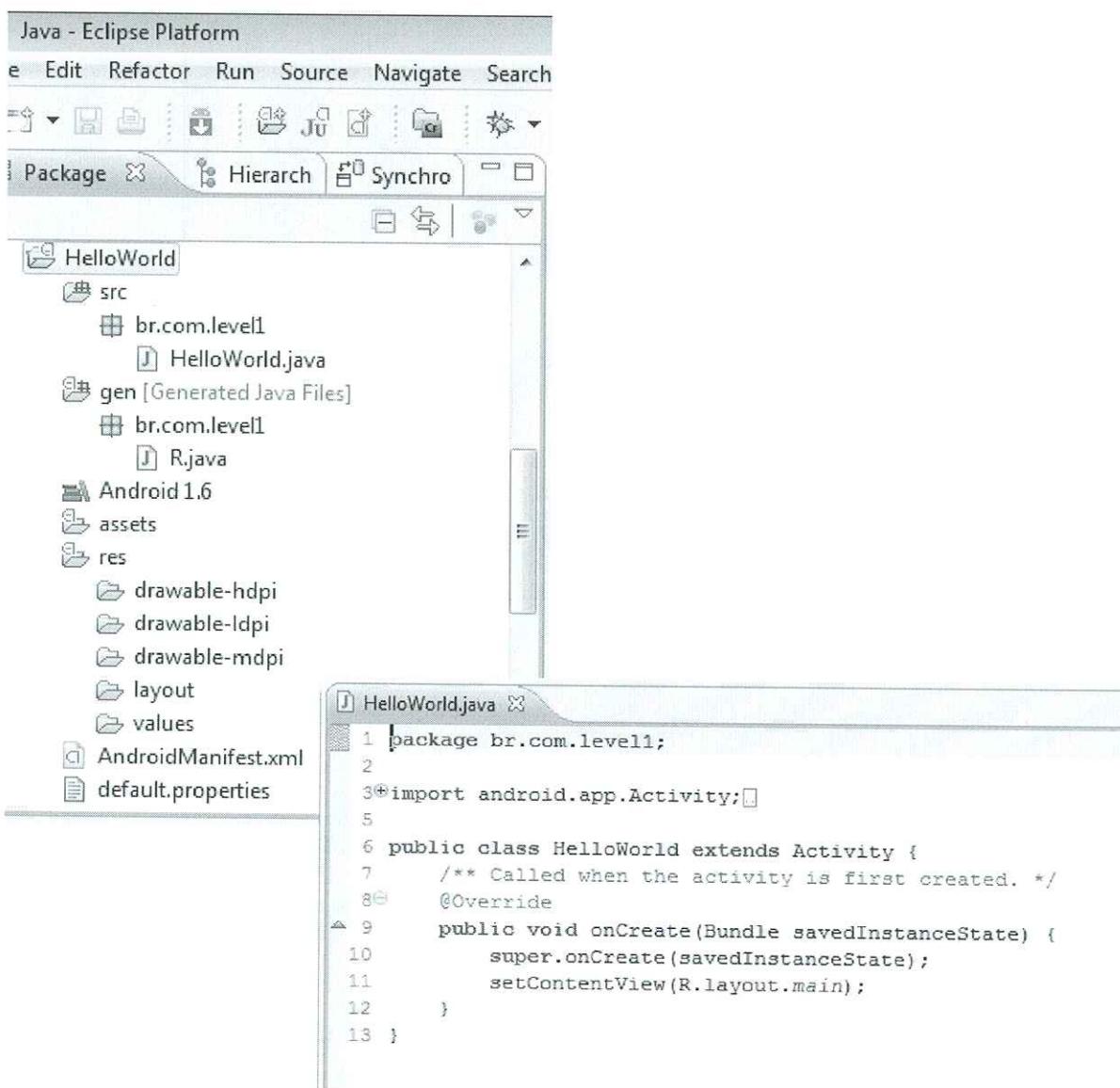
Enter a Min SDK Version. Está é um número que indica a API mínima para rodar sua aplicação. Caso coloque um número o ADT automaticamente seta o atributo minSdkVersion no nó <uses-sdk> do arquivo Manifest. Se você não tem certeza desse número, use o mesmo escolhido acima como target.

7. Ignore por enquanto o Test Project e clique em Concluir.



Estrutura do Projeto

Depois de concluir o New Project Wizard, o ADT cria as seguintes pastas e arquivos em seu novo projeto:



src /

Inclui um de Atividade em arquivo Java. Todos os outros arquivos Java para o aplicativo devem ser colocados nessa pasta

<Android Version>/ (Por exemplo, Android 1.5 /)

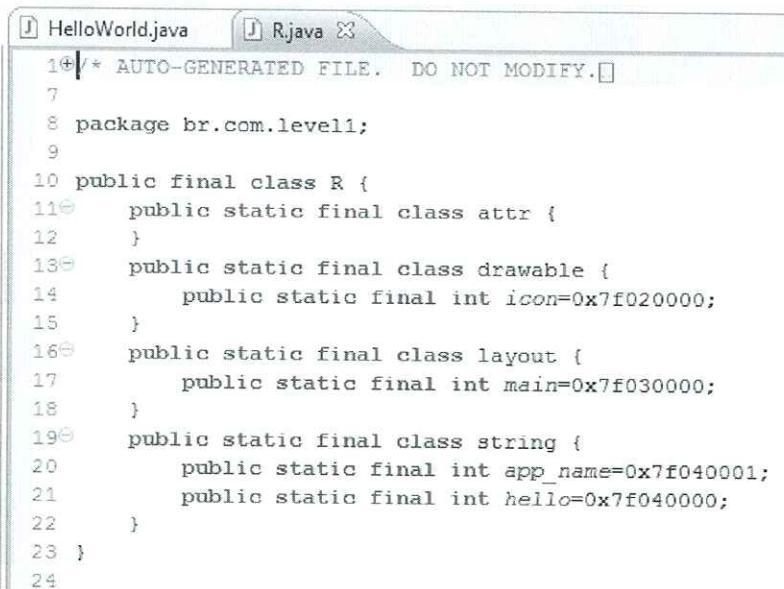
Inclui o arquivo android.jar. A versão da biblioteca é determinada pela escolha do target que você escolheu no Assistente para um novo projeto.

assets /

Por enquanto estará vazia. Você pode usá-la para armazenar arquivos fonte dos recursos que estarão na pasta res/.

gen /

Ela contém os arquivos gerados pelo ADT, como o seu arquivo R.java e interfaces criadas a partir de arquivos AIDL (Android Interface Definition Language).



```
1/* AUTO-GENERATED FILE. DO NOT MODIFY. */
2
3package br.com.level1;
4
5
6public final class R {
7    public static final class attr {
8    }
9
10   public static final class drawable {
11       public static final int icon=0x7f020000;
12   }
13   public static final class layout {
14       public static final int main=0x7f030000;
15   }
16   public static final class string {
17       public static final int app_name=0x7f040001;
18       public static final int hello=0x7f040000;
19   }
20 }
21
22 }
```

res /

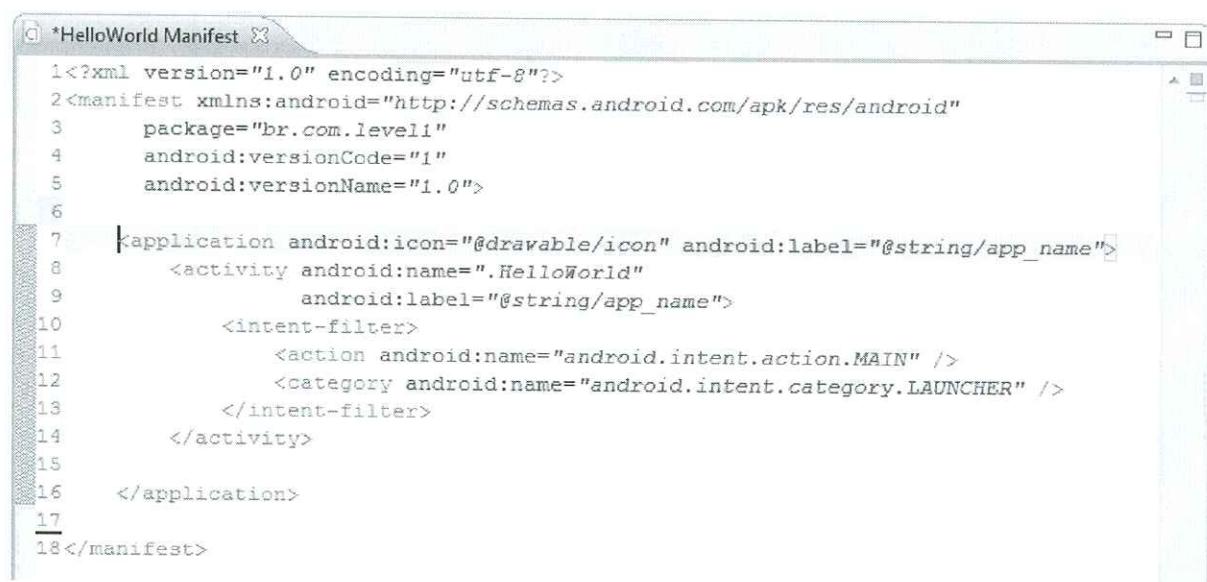
Todos os recursos que serão usados no projeto, como imagens, arquivos de layout, seqüências de caracteres, etc.

default.properties

Este arquivo contém as configurações do projeto, tais como a construção do target. Este arquivo é parte integrante do projeto. Nunca deve ser editado manualmente, para editar as propriedades do projeto, clique com o botão direito do mouse na pasta do projeto e selecione "Properties".

O arquivo de manifest

O Manifesto Android para o seu projeto. Veja o arquivo `AndroidManifest.xml`.

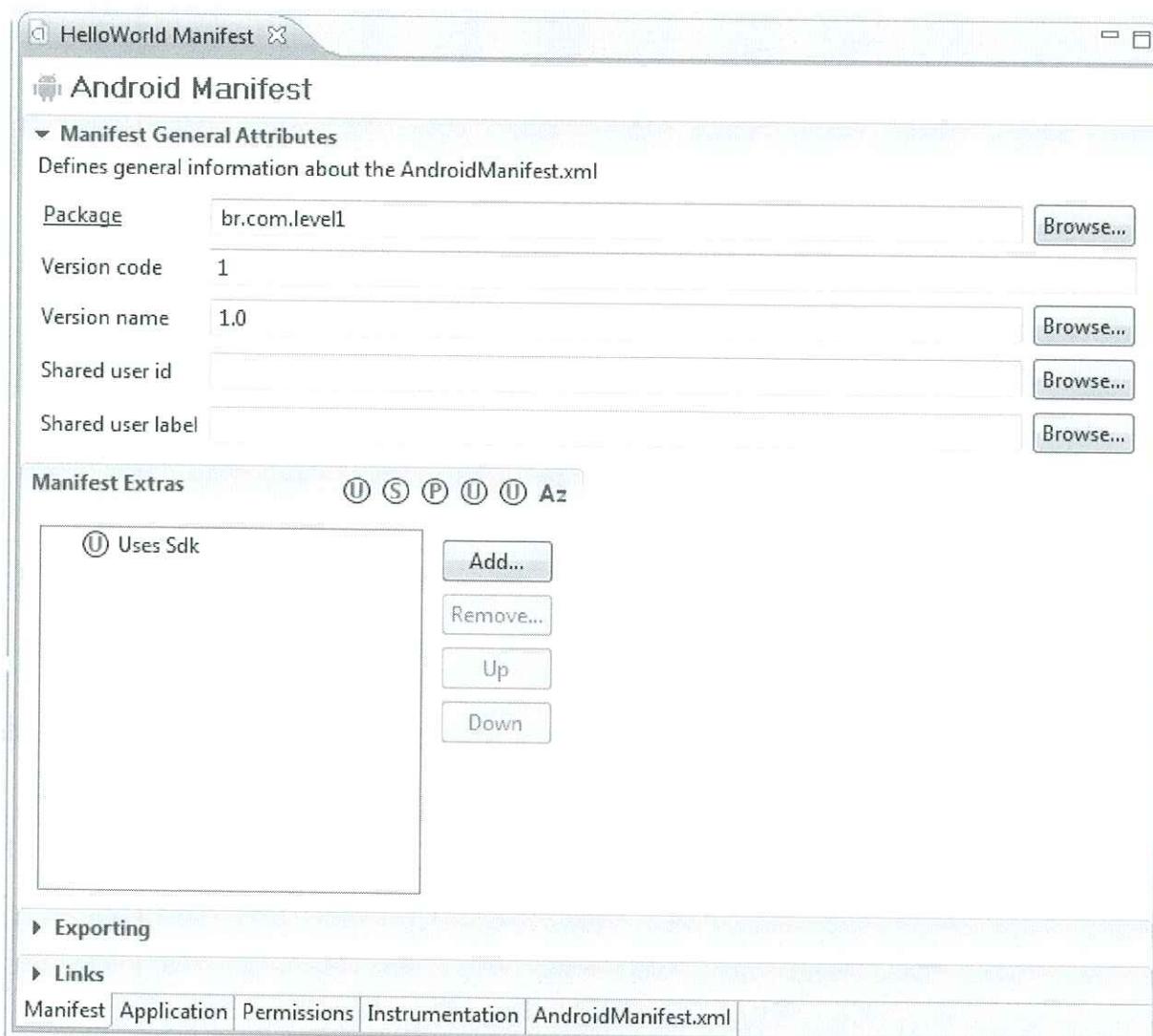


```
1<?xml version="1.0" encoding="utf-8"?>
2<manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="br.com.level1"
4    android:versionCode="1"
5    android:versionName="1.0">
6
7    <application android:icon="@drawable/icon" android:label="@string/app_name">
8        <activity android:name=".HelloWorld"
9            android:label="@string/app_name">
10            <intent-filter>
11                <action android:name="android.intent.action.MAIN" />
12                <category android:name="android.intent.category.LAUNCHER" />
13            </intent-filter>
14        </activity>
15    </application>
16
17</manifest>
```

Cada projeto deve ter um arquivo `AndroidManifest.xml` (necessário que seja esse nome) em seu diretório raiz. O manifesto apresenta informações essenciais sobre o aplicativo para o sistema Android, o sistema precisa dessas informações antes de executar qualquer aplicativo de código. O manifesto contém informações como:

- Especifica o pacote de Java para o aplicativo. O nome do pacote serve como um identificador exclusivo para o aplicativo.
- Descreve os componentes da aplicação, atividades, serviços, broadcast receivers e content providers. Ele especifica as classes que implementam cada um dos componentes e publica as suas capacidades (por exemplo, as intents). Estas declarações fazem o sistema Android saber o que os componentes são e em que condições eles podem ser lançados.
- Determina que os processos serão os componentes do aplicativo host.
- Declara que permissões o aplicativo deve ter para acesso protegido a partes da API e interação com outras aplicações.
- Também declara as permissões que os outros são obrigados a ter, a fim de interagir com os componentes do aplicativo.

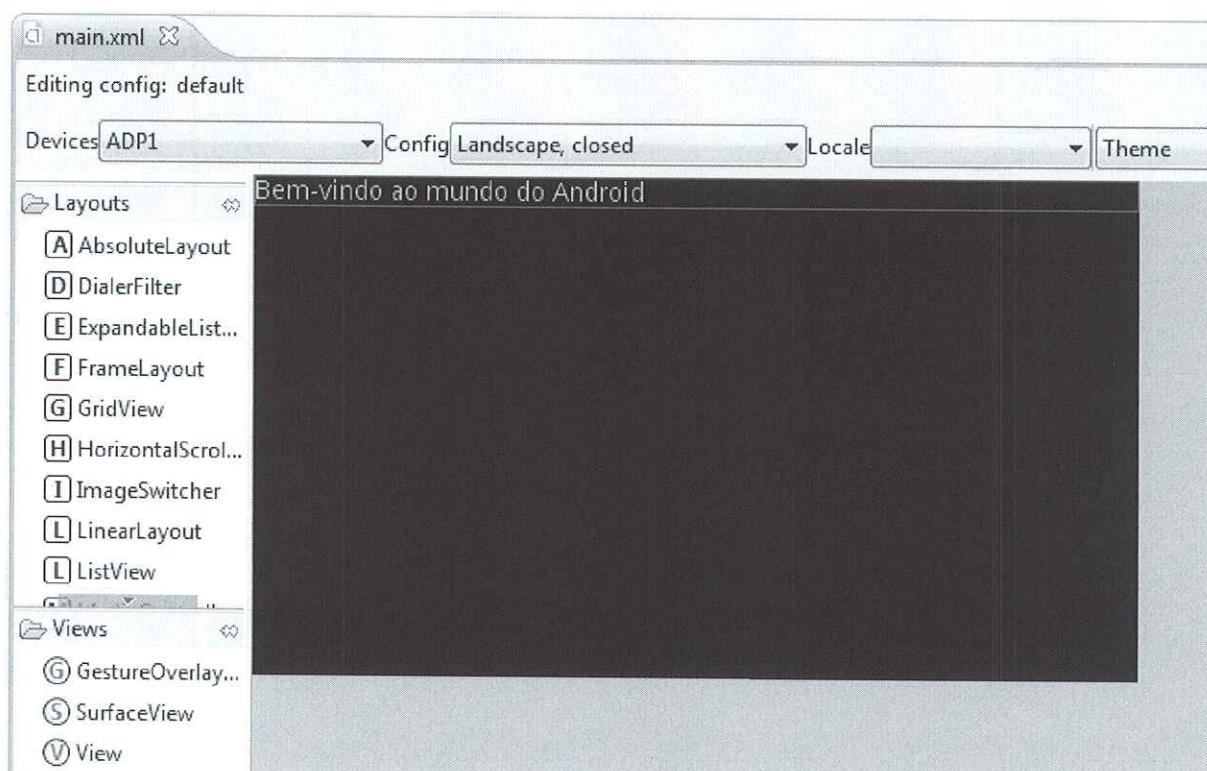
- Lista as classes *Instrumentation* que disponibilizam informações de profiling. Estas declarações estão presentes no manifesto somente enquanto o aplicativo está sendo desenvolvido e testado, eles são removidos antes da aplicação ser publicada.
- Declara API level mínimo que o aplicativo requer.
- Lista as bibliotecas que o aplicativo está usando.



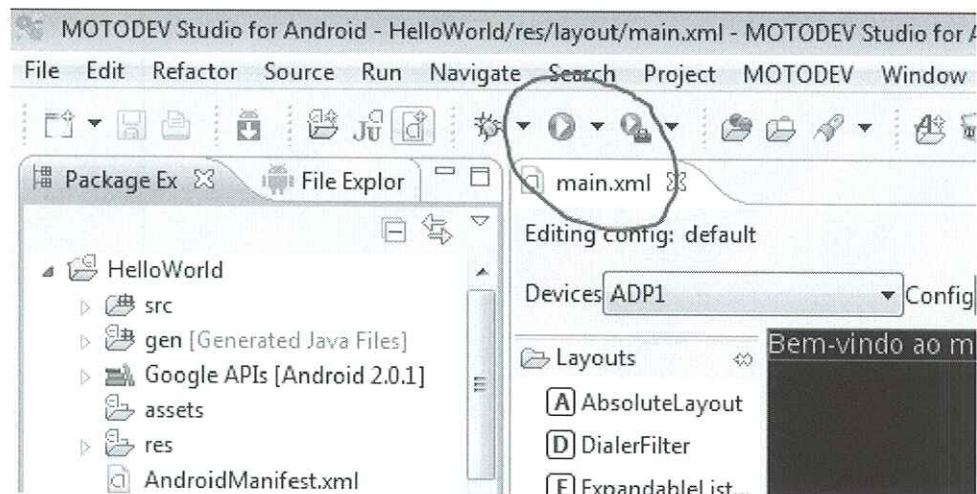
Layout com XML

A parte sobre layout será discutida no próximo módulo, mas quando criamos um novo projeto, o ADT já cria um arquivo main.xml com uma estrutura básica de layout. A estrutura consiste em um layout vertical com um componente de texto dentro.

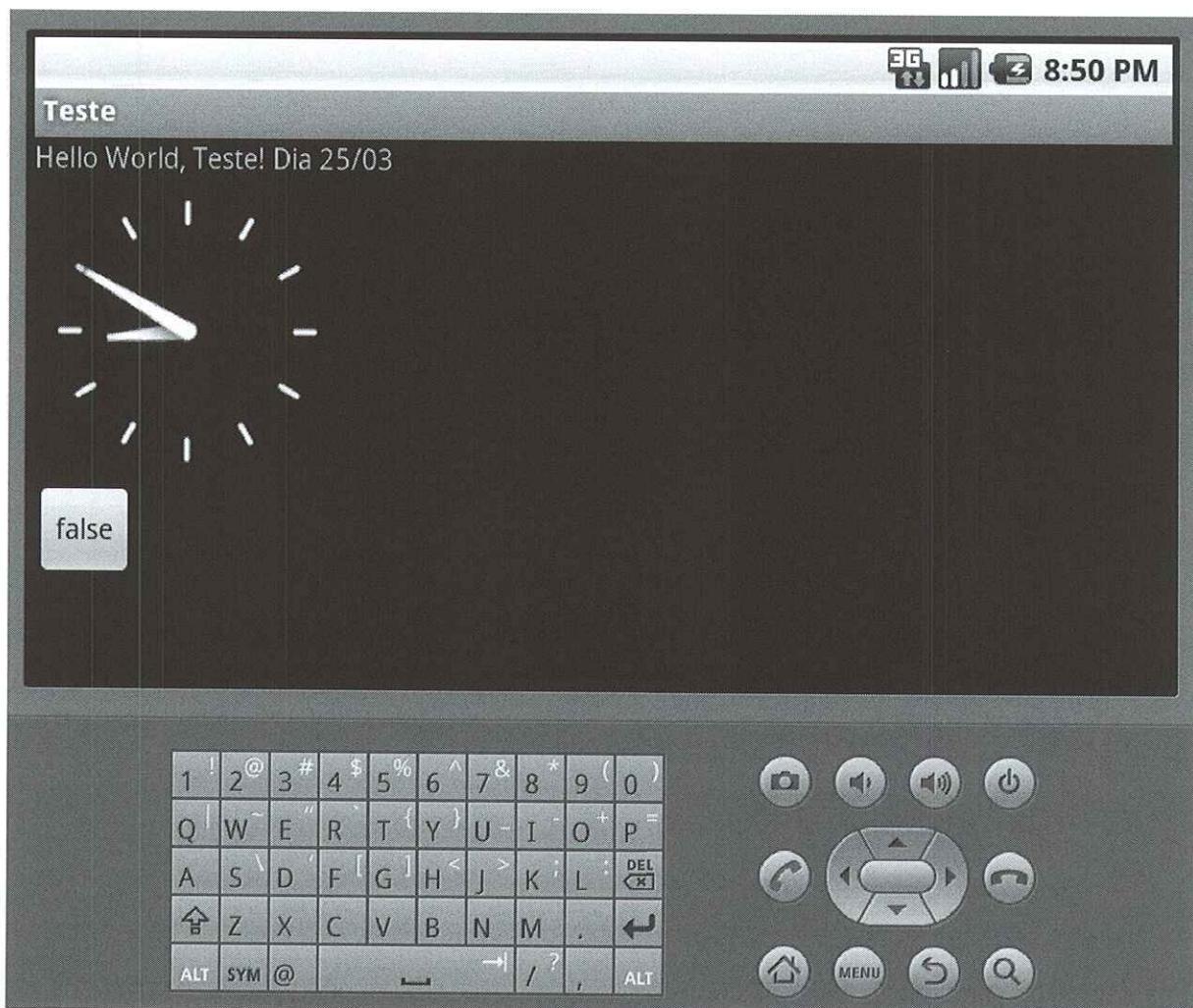
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"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



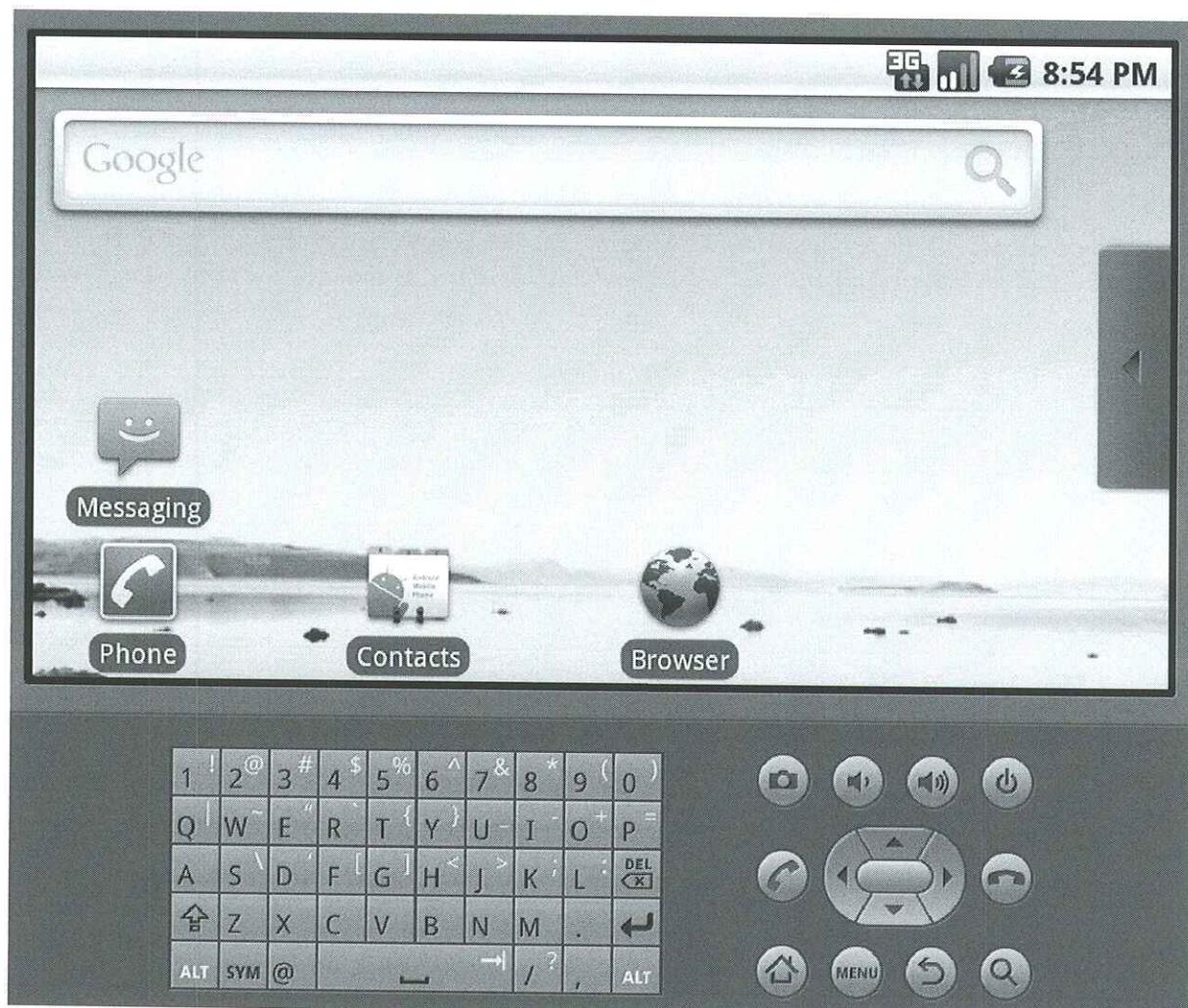
Executando seu aplicativo no emulador



Quando pedimos para rodar a aplicação, o ADT executa o emulador do Android, instala o aplicativo no sistema e assim podemos ver o resultado, como o a seguir:



O emulador do ADT é um sistema real Android, você poderá acessar todos os aplicativos nativos e também instalar novos aplicativos para serem emulados no sistema.



Anotações

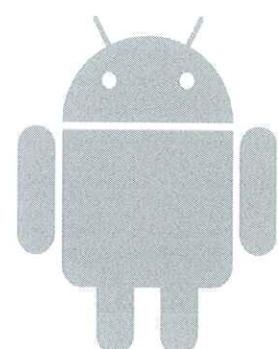
as pastas drawable está dividida pelo tamanho de imagem
baixa, media, alta resolução. Assets pra colocar qq arquivo bruto (extra)

(@ qualquer coisa manda família variáveis @ id vai pro-
(classe)
anar na família "id". O sinal '+' informa que não há este
id e o mesmo deve ser criado. Com (+) crie, sem use existente.

Pesos iguais componentes ocupam mesmo espaço tela.

gravity sempre os container; layout gravity ao componente
temendo como referencia de alinhamento seu "pai".

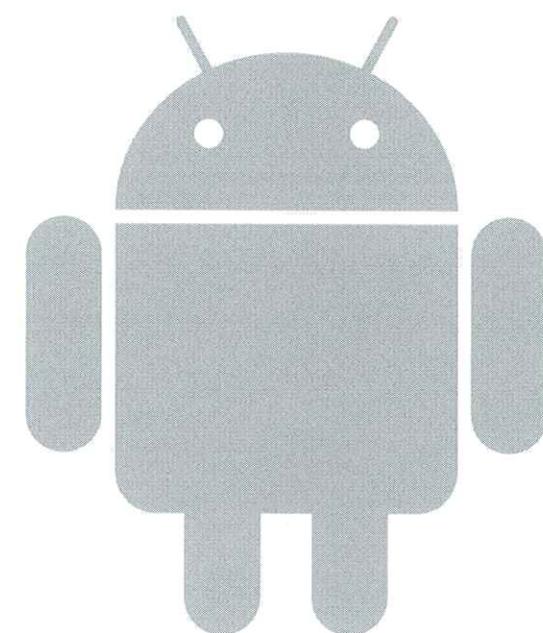
Qualquer arquivo xml não pode começar com letra minúscula.



Módulo 6

Estrutura da Aplicação

- Activities
- Ciclo de vida
- Intents
- Serviços



Activities

Uma Atividade é a unidade básica de interação com o usuário de uma aplicação. A classe principal de uma aplicação estende uma Activity, que é responsável por cuidar da criação de uma janela para que você posicione seus componentes de interface com o usuário (Views) ou mesmo outras atividades.

A atividade pode ser mais facilmente entendida como uma aplicação desktop ou uma aplicação visual que você inicia em seu Android, ou seja, toda aplicação é uma Atividade.

Embora a maior parte das Atividades seja exibida em modo Fullscreen, seus recursos não limitam-se a esse tipo de exibição. Atividades podem ser configuradas como janelas flutuantes ou atividades embutidas dentro de outras (através de um ActivityGroup).

Sua aplicação pode não conter uma Activity, caso se trate de um Serviço (visto com mais detalhes posteriormente).

Subclasses de Activity geralmente sobrescrevem e implementam dois métodos: onCreate(Bundle bundle) e onPause().

onCreate(Bundle bundle)

É o método chamado ao inicializar a sua atividade. É nesse método que você geralmente chamará setContentView(int) para exibir um UI na tela.

onPause()

Chamado quando a aplicação é interrompida pelo sistema operacional. Todos os dados em vigor nesse ponto devem ser salvos para serem recuperados quando a aplicação voltar à sua atividade.

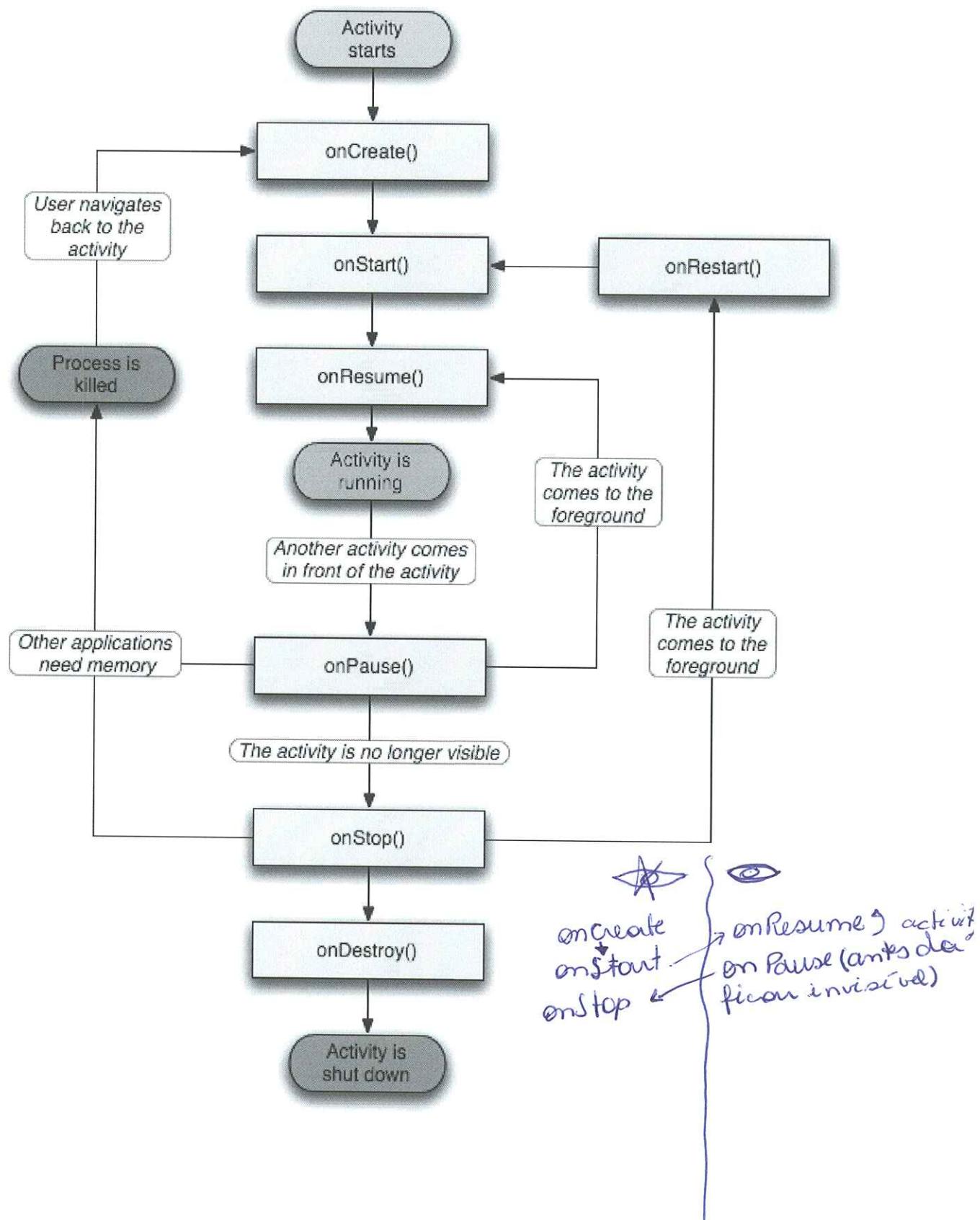
Ciclo de vida

As atividades são gerenciadas pelo sistema em uma espécie de pilha. Quando uma nova atividade é iniciada, ela é colocada no topo da pilha e se torna a atividade em funcionamento. A atividade anterior permanece sempre abaixo da nova e retornará ao primeiro plano após a atividade em execução encerrar.

Uma atividade tem essencialmente quatro estados:

- Se uma atividade está em primeiro plano na tela (no topo da pilha), ela está em estado ativo ou em estado de execução.
- Se uma atividade perde o foco, mas ainda está visível (ou seja, não está em modo tela-cheia ou uma atividade transparente está por cima da sua atividade), ela entra em estado de pausa. Quando a atividade é interrompida eles ainda sim permanece ativa (em estado de pause()), o que mantém todas as informações ativas e permanece no gerenciador de janelas. Entretanto a atividade pode a qualquer momento ser fechada pelo sistema em casos extremos como falta de memória.
- Se uma atividade é totalmente obscurecida (não está mais visível para o usuário) por outra atividade, ela entra em estado de parada (onStop()). Ainda nesse estado, todas as informações da atividade são mantidas.
- Se uma atividade está em estado de pausa ou parada, o sistema pode tirar a atividade da memória, ou enviar uma solicitação de término da aplicação, ou simplesmente matar o processo. Quando ela é exibida novamente para o usuário, deve ser reiniciada e completamente restaurada ao seu estado anterior.

O diagrama a seguir apresenta os métodos callback que você pode implementar para executar operações na troca de estados de atividades.



Existem três métodos-chave a se focar em sua atividade:

- O tempo de vida de uma atividade acontece entre a primeira chamada de `onCreate(Bundle)` até a única chamada final de `onDestroy()`. Uma atividade fará toda a configuração “global” em `onCreate()`, e liberar todos os recursos pendentes em `onDestroy()`. Por exemplo, se uma thread está rodando em background para fazer download de dados da rede, a thread deve ser criada em `onCreate()` e então ser parada em `onDestroy()`.
- O tempo de visibilidade de uma atividade acontece entre uma chamada de `onStart()` até a correspondente chamada para `onStop()`. Durante todo esse tempo o usuário pode ver a atividade na tela, embora ela possa não estar em primeiro plano. Entre esses dois métodos você pode manter os recursos necessários para exibir a atividade para o usuário. Por exemplo, você pode registrar um `BroadcastReceiver` em `onStart()` para monitorar as mudanças que impactam sua interface, e cancelar o registro em `OnStop()` quando o usuário a deixar de ver o que você está exibindo. Os métodos `onStart()` e `OnStop()` podem ser chamados várias vezes, conforme a atividade se torna visível e oculta para o usuário.
- A vida em primeiro plano de uma atividade acontece da chamada para `onResume()` até a correspondente chamada para `onPause()`. Durante esse tempo a atividade está à frente de todas as outras, e interagindo com o usuário. Uma atividade pode frequentemente alternar entre os estados de pausa (`onPause`) e resumo (`onResume`) – por exemplo, quando o dispositivo entra em modo sleep; quando o resultado de uma atividade é entregue; quando uma nova intenção é entregue – sendo assim o código nesses métodos deve ser bastante leve.

Abaixo é exibido um exemplo de uma Atividade que sobrescreve o método `onCreate(Bundle)`:

```
package com.estudos.classes;

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

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

Intents

Intents são elementos de essencial importância no Android, pois permitem a interação entre aplicações e serviços, que fornecem informações adicionais necessárias para a sua aplicação.

Uma Intent é uma descrição abstrata de uma operação a ser executada. A intent fornece facilidade para a realizar a vinculação de runtime entre o código em diferentes aplicações. Sua utilização mais significativa está no lançamento de atividades, onde pode ser pensado como a “cola” entre as atividades. As peças principais de informação em um Intent são:

- Ação - A ação geral a ser realizada, como ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc
- Dados - Os dados para operar, como por exemplo um registro de pessoa no banco de dados de contatos, expressado como uma URI.

Criando uma intenção

```
Intent novaIntencao = new Intent(Intent.ACTION_WEB_SEARCH);  
startActivity(novaIntencao);
```

O trecho de código acima abre o buscador de web padrão do Android. O construtor recebeu uma das “Ações de Atividade Padrões”, mostradas na tabela abaixo. O método startActivity(nomeDaIntentao) é utilizado para lançar a intent criada, desta forma o sistema delegará a função de carregar a atividade desejada ao receiver responsável.

Ações mais comuns:

- ACTION_MAIN
- ACTION_VIEW
- ACTION_ATTACH_DATA
- ACTION_EDIT
- ACTION_PICK
- ACTION_CHOOSER
- ACTION_GET_CONTENT
- ACTION_DIAL
- ACTION_CALL
- ACTION_SEND
- ACTION_SENDTO
- ACTION_ANSWER
- ACTION_INSERT
- ACTION_DELETE
- ACTION_RUN
- ACTION_SYNC
- ACTION_PICK_ACTIVITY
- ACTION_SEARCH
- ACTION_WEB_SEARCH
- ACTION_FACTORY_TEST

Outro exemplo de uma intent que chama uma atividade qualquer:

```
Intent chamarAtividade = new Intent(this, AtividadeTal.class);
startActivity(chamarAtividade);
```

O código acima recebe o contexto atual da aplicação e o nome da classe (seguido do .class, NÃO do .java) a ser chamada.

Serviços

Trata-se de códigos que são executados em segundo plano. Um exemplo de um serviço é um player de músicas que continua executando uma lista de músicas, mesmo que o usuário não esteja na aplicação.

Os serviços, assim como nos sistemas Desktop, não possuem interface de usuário e rodam por um período de tempo indefinido.

Serviços serão amplamente abordados no curso Desenvolvimento com Google Android Level III.

Módulo 7

Interface com o usuário

- Classe View e ViewGroup
- Unidades de medida
- Widgets
- Tipos de layout
- Tratamento de eventos
- Navegação entre telas

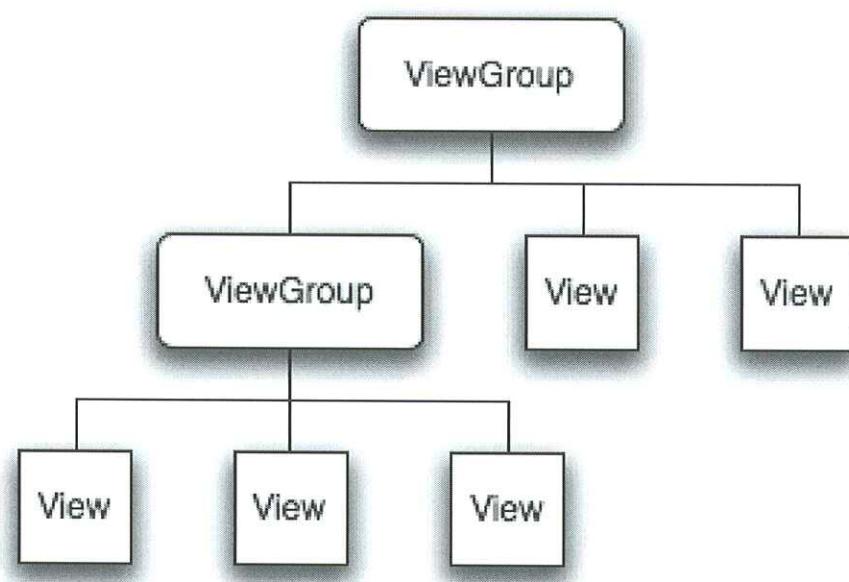


Classe View e ViewGroup

Em um aplicativo Android, a interface do usuário é construída usando objetos View e ViewGroup. Existem muitos tipos de view e view groups, cada um dos quais é um descendente da classe View.

Objetos view são as unidades básicas de interface do usuário na plataforma Android. A classe View serve como base para as subclasses chamadas de “widgets”, que são objetos de interface do usuário, como campos de texto e botões. A classe ViewGroup serve como base para as subclasses de layout, e oferecem vários tipos de layout de interface, como linear, tabular e relativo.

Um objeto de exibição é uma estrutura de dados, cujas propriedades guardam parâmetros de layout e conteúdo de uma determinada área retangular da tela. Um objeto View lida com sua própria medida, layout, desenho, foco, rolagem e interações na a área retangular da tela na qual reside.



Unidades de medida

px - Pixels - corresponde a pixels reais na tela.

in - Polegadas - com base no tamanho físico da tela.

mm - Milímetros - com base no tamanho físico da tela.

pt - Pontos - 1 / 72 de polegada com base no tamanho físico da tela.

dp - Density-independent Pixels - uma unidade abstrata que se baseia na densidade física da tela. Estas unidades são relativas a uma tela de 160 dpi, um dp é um pixel em uma tela de 160 dpi. A relação entre a DP-pixel mudará com a densidade da tela, mas não necessariamente na proporção direta.

sp - Scale-independent Pixels - funciona como a unidade dp, mas também é escalado pelo tamanho da fonte de preferência do usuário. É recomendável usar essa unidade ao especificar o tamanho da fonte, assim elas serão ajustadas tanto pela densidade da tela quanto preferência do usuário.

Widgets

Um widget é um objeto View que serve como uma interface de interação com o usuário. Android fornece um conjunto de widgets nativos, como botões, checkboxes e caixas de entrada de texto, assim você pode construir rapidamente sua interface de usuário.

Alguns widgets fornecidos pelo Android são mais complexos, como um selecionador de data, um relógio, e controles de zoom. Mas você não está limitado aos elementos fornecidos pela plataforma Android. Se você deseja fazer algo mais personalizado e criar seus próprios elementos de interface, poderá fazer através da definição de seu próprio objeto ou extendendo os componentes atuais e combinando elementos existentes.

Layouts

Os layouts definem os diversos tipos de tela disponíveis em dispositivos Android. Eles podem assumir diferentes proporções e densidades de pixels.

Através dos gerenciadores de layout é possível alterar a orientação da tela entre os modos retrato e paisagem em tempo de execução graças à eficácia da estrutura implantada nesses gerenciadores. Layouts no Android são armazenados no caminho res/layouts no formato XML.

Os Layouts são o nó principal de um arquivo de layouts, que por sua vez segue uma estrutura hierárquica de árvore. Todas as outras Views do arquivo são filhas da View de layout.

Tipos de layout

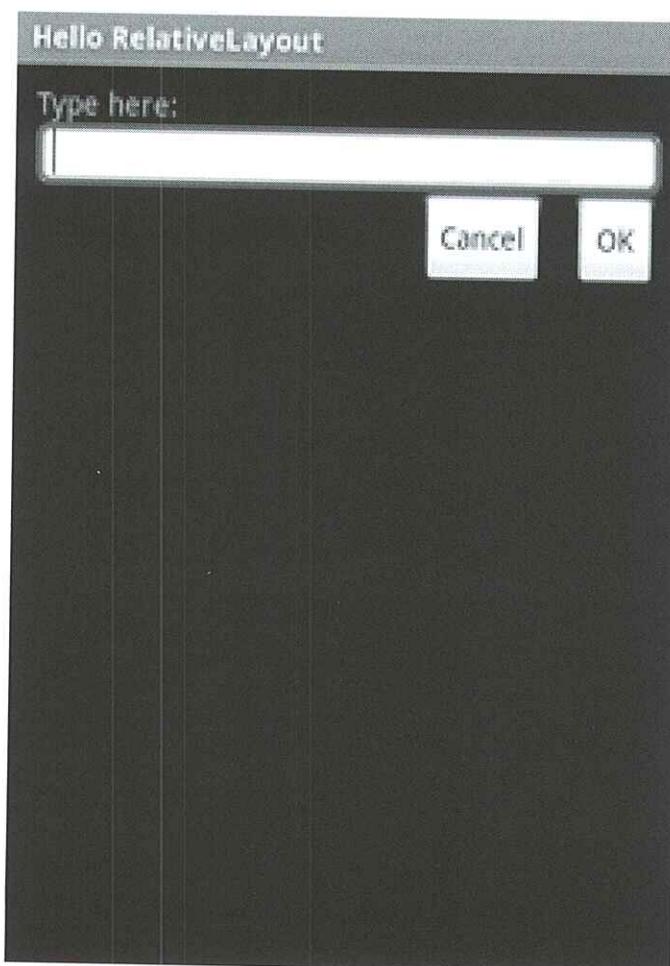
FrameLayout

É talvez o mais simples de todos os layouts. Tem a capacidade de alocar na tela espaço para apenas uma View, sempre no canto superior esquerdo, ou seja, se houverem duas views elas serão sobrepostas.

AbsoluteLayout

Organiza os componentes na tela de maneira livre, ou seja, da maneira que você especificar. É um layout que provê maior liberdade de composição na tela porém não alinha nem redimensiona os componentes automaticamente, seguindo apenas o que foi especificado para cada view-filha.

O AbsoluteLayout vai em desencontro ao objetivo inicial do Android no que se diz respeito à portabilidade, pois como citado anteriormente ele não redimensiona os componentes quando o programa é executado em diferentes tipos de tela, gerando um risco muito grande de que a sua aplicação seja exibida de maneira errônea em outros dispositivos.

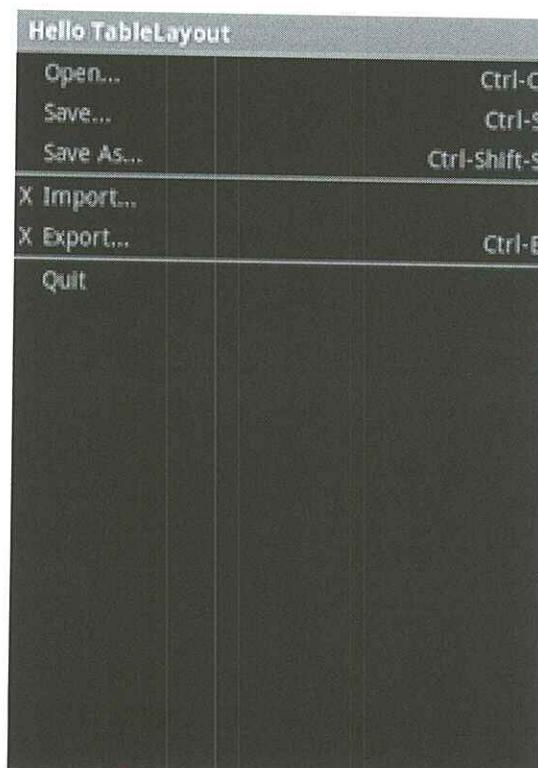


TableLayout

Organiza as views-filhas da mesma maneira que uma tabela HTML organiza componentes em uma página.

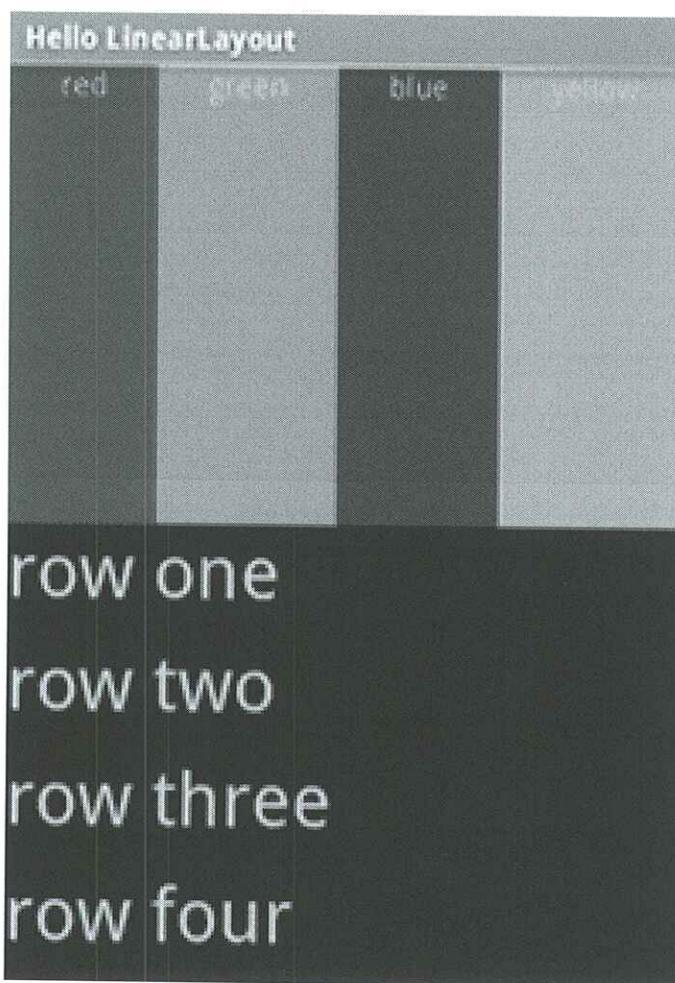
```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="@string/table_layout_4_open"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_open_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:text="@string/table_layout_4_save"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_save_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>
</TableLayout>
```



LinearLayout

Tipo bastante usado nas aplicações, ele organiza as Views de forma que todas sejam exibidas horizontalmente ou verticalmente. Em modo vertical, o LinearLayout configura a largura de todas as linhas por igual, assumindo a largura da view-filha que possuir a medida mais alta de largura. Em modo horizontal, por sua vez, todas as Views contidas na linha, possuem uma mesma altura, sendo assumida a altura da view-filha com maior altura.



```
<?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>
```

RelativeLayout

Posiciona as Views na tela tomando como referências umas às outras. Você configura a posição de um componente especificando qual é a localização dele com relação a outro e o gerenciador de layout se encarregará de posicioná-los o mais próximo possível do que foi especificado.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/blue"
    android:padding="10px" >

    <TextView android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:" />

    <EditText android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label" />

    <Button android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10px"
        android:text="OK" />

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:text="Cancel" />
</RelativeLayout>
```

Tratamento de Eventos

Tratar eventos no Android é relativamente simples e assemelha-se com a maneira como é feito em aplicações desktop.

Abaixo é mostrado um trecho de código que escuta eventos do teclado e exibe o código inteiro do caractere pressionado em uma mensagem de torrada (Toast):

```
public class MinhaAplicacao extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        EditText txtTexto = (EditText)findViewById(R.id.EditText01);  
  
        txtTexto.setOnKeyListener(new View.OnKeyListener() {  
  
            @Override  
            public boolean onKey(View view, int keyCode,  
KeyEvent event) {  
                if(KeyEvent.ACTION_DOWN == event.  
getAction()){  
                    Toast.makeText(getApplicationContext(), “Tecla : “ + keyCode, Toast.LENGTH_SHORT).show();  
                }  
                return false;  
            }  
        });  
    }  
}
```

Abaixo é mostrado um trecho de código que trata o evento que ocorre ao pressionar um determinado botão:

```
public class MinhaAplicacao extends Activity implements OnClickListener {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        Button btnEnvio = (Button)findViewById(R.id.Button01);  
  
        btnEnvio.setOnClickListener(this);  
    }  
  
    @Override  
    public void onClick(View v) {  
        Toast.makeText(getApplicationContext(), R.string.btn_  
clicado, Toast.LENGTH_SHORT).show();  
    }  
}
```

O método `makeText` nesse caso utilize um id de recurso de Strings contigo no arquivo `strings.xml`, exibido abaixo:

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="hello">Hello World, MinhaAplicacao!</string>  
    <string name="app_name">Minha Aplicacao</string>  
    <string name="btn_clicado">O botão foi clicado!</string>  
</resources>
```

Navegação entre telas

A navegação entre telas no Android é parecida com a maneira feita em seu primo JME, porém de maneira mais simples e flexível.

Toda navegação entre telas é feita através da chamada do método sobrecarregado `setContentView()`. Sendo chamado o método com a assinatura propícia para cada situação. Abaixo estão alguns exemplos:

Chamando `setContentView(int layoutResId)` passando como parâmetro o layout (arquivo XML de layout) desejado:

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

O arquivo `main.xml` contém o layout formatado da nossa aplicação:

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

<EditText
    android:text=""
    android:id="@+id/EditText01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

<Button
    android:text="@+id/Button01"
    android:id="@+id/Button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>
```

Outra maneira de mudar o layout:

O código abaixo demonstra um TextView criado de forma programática para ser exibido na tela com o método sobrecarregado setContentView que aceita como argumento uma View:

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    TextView textview = new TextView(getApplicationContext());  
    textview.setText("Olá! Esse é um TextView criado de forma  
programática!");  
    setContentView(textview);  
}
```

Módulo 8

Publicação no Android Market

- Publicação no Android Market
- Assinando sua aplicação com o ADT
- Enviando sua aplicação ao Android Market



Publicação no Android Market

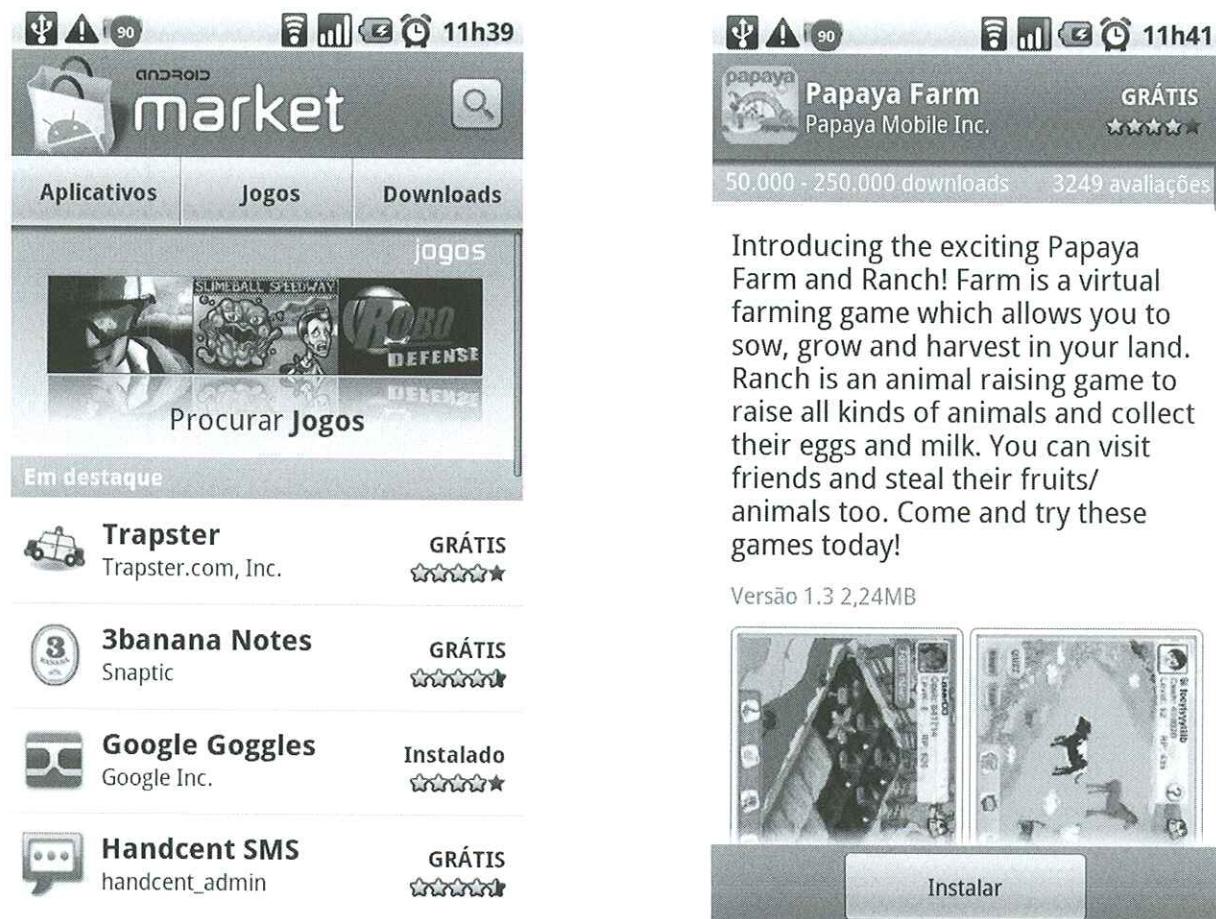
Desenvolver suas aplicações para utilizar para seus próprios objetivos é uma tarefa bastante simples. Entretanto eventualmente você pode querer compartilhar sua aplicação com outros usuários, seja de maneira gratuita ou cobrando por isso.

Para esse fim, o Google criou o Android Market, visando disponibilizar aplicativos para usuários de telefones Android.

Desenvolvedores podem escrever e publicar aplicações no Android Market com apenas alguns passos.

Talvez o mais importante passo antes de publicar sua aplicação seja testá-la exaustivamente em diversos aparelhos, afinal nada como testes em dispositivos reais para ter 100% de certeza de que sua aplicação funciona de fato.

Após ter limpado o código e ter feito todos os testes é hora de assinar a sua aplicação (obrigatório para publicação no Android Market) a fim de assegurar que a aplicação é sua mesmo (possui a sua assinatura digital).

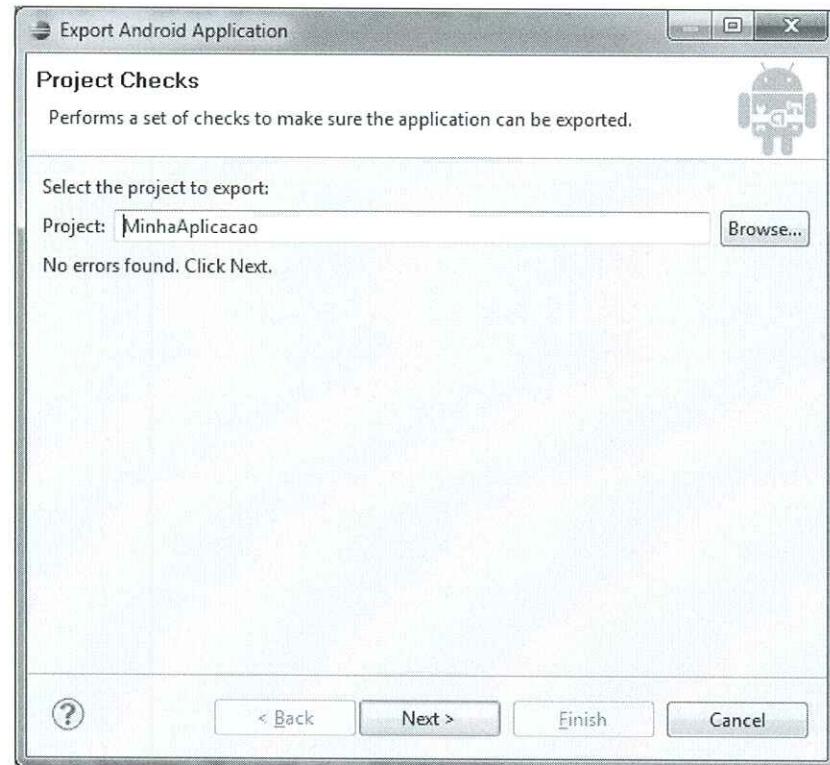


Assinando sua aplicação com o ADT

Para assinar a sua aplicação basta clicar com o botão direito do mouse em cima do nome do projeto, posicionar o mouse sobre a opção “Android Tools” e selecionar o item “Export Signed Application Package”

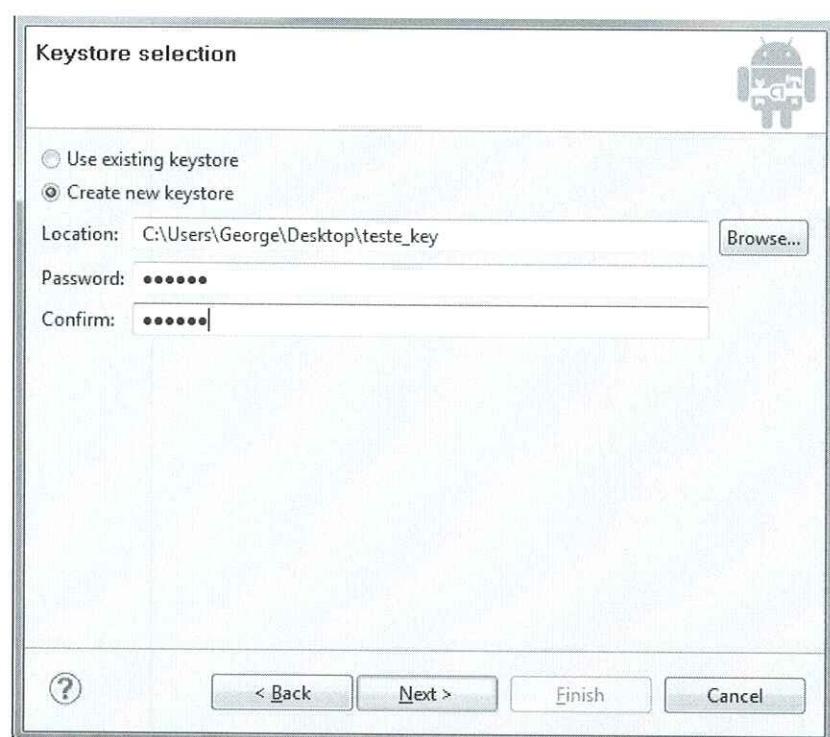


Uma tela aparecerá pedindo para que você confirme qual é o projeto deseja exportar:



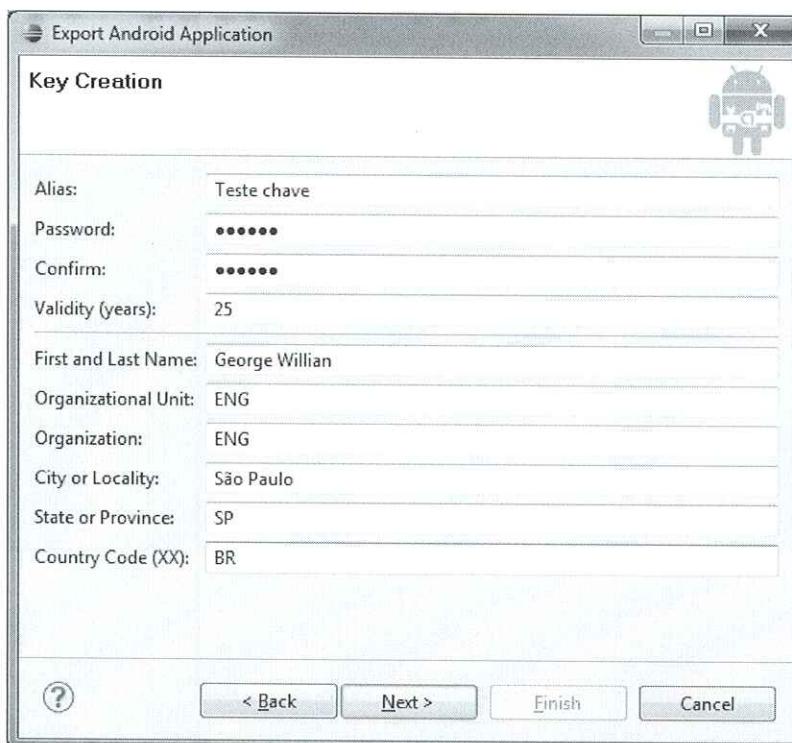
Você deve criar uma chave de assinatura digital caso ainda não possua uma. O ADT então perguntará aonde você quer salvar o arquivo que conterá a sua chave, dê um nome para o arquivo e selecione um local para salvá-lo.

Configure uma senha (que será utilizada sempre que quiser reutilizar essa chave de assinatura para suas futuras aplicações) e clique em "Next".

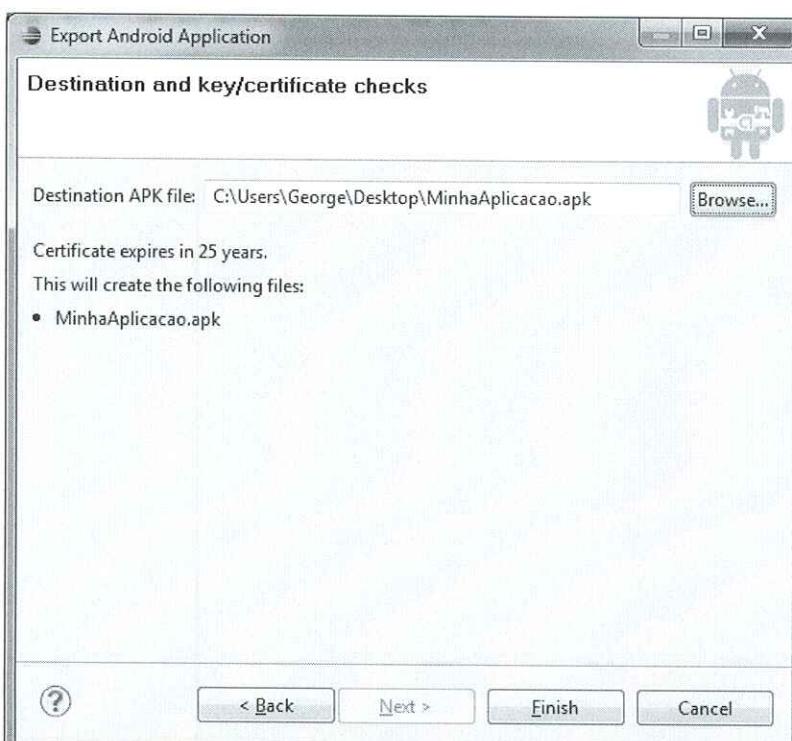


A tela seguinte pedirá informações sobre você, para que as informações sejam armazenadas em seu arquivo de assinatura.

Ainda nessa tela, você precisa especificar por quanto tempo vale a chave que está criando. Vale lembrar que o Android Market apenas aceita assinaturas com validade ativa pelo menos até 22 de outubro de 2033.



Após preencher os dados dessa tela, você deve especificar o local onde quer salvar o arquivo APK assinado do seu projeto.



Ao clicar no botão “Finish” o arquivo .apk de sua aplicação (nesse exemplo, MinhaAplicacao.apk) será criado.



Pronto! Isso é tudo o que você precisa fazer antes de enviar a sua aplicação ao Android Market.

Enviando sua aplicação ao Android Market

Para a publicação, acesse o endereço <http://market.android.com/publish>. Você será solicitado a efetuar o login com uma conta sua do Google. Se ainda não possuir uma basta cadastrar-se rapidamente.

Crie seu perfil de desenvolvedor e – mais importante – pague a taxa de registro, no valor de U\$25,00 (Atualmente pagamento disponível apenas por cartão de crédito, utilizando o Google Checkout). Leia e aceite os termos de Licença do Android Market.

The screenshot shows the Android Market developer registration interface. At the top, there's a navigation bar with links for Home, Help, Android.com, and Sign out. Below that, it says "leonardo.sobral@gmail.com". The main content area has a large green banner with the text "Your Registration to the Android Market is approved! You can now upload and publish software to the Android Market." Below the banner, it shows the user's profile: "Leonardo Sobral" and "leonardo.sobral@gmail.com", with a link to "Edit profile". At the bottom, there's a link to "All Android Market listings".

Após criar o seu perfil faça o upload de sua aplicação especificando o nome e o local do arquivo apk (o mesmo gerado anteriormente), o título e a descrição, o tipo de aplicação, a categoria, o preço (se é gratuito ou não) e a geografia (é possível limitar a disponibilidade de seu aplicativo por região).

Após todos os passos é só aguardar alguns dias para que sua aplicação esteja localizável no Android Market, sendo disponibilizada para qualquer aparelho Android.