

Desenvolvimento com Google

ANDROID

Level I



Release 001
2010

ENG


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

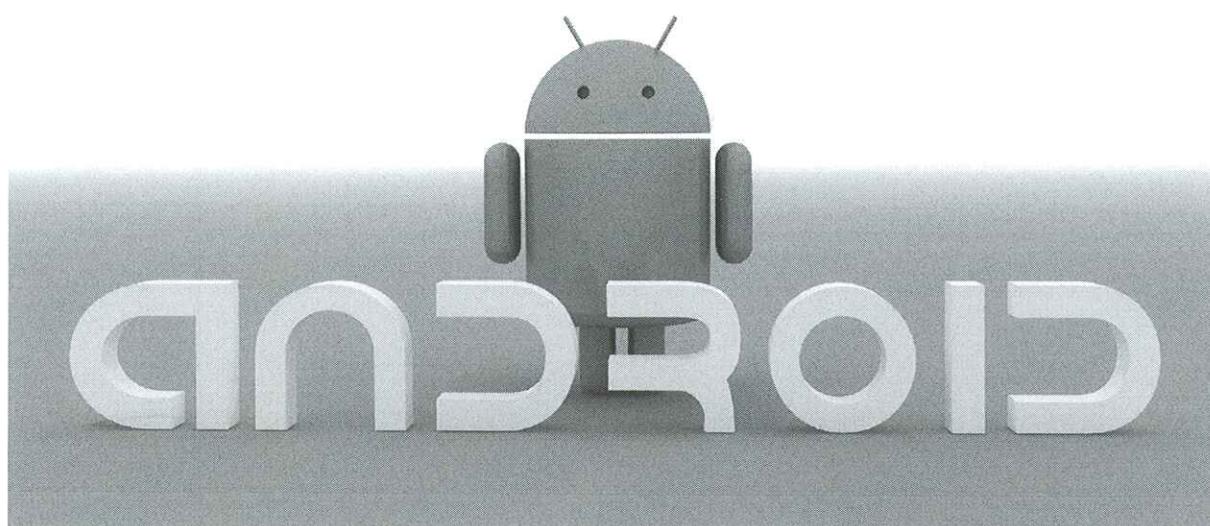
Introdução

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).

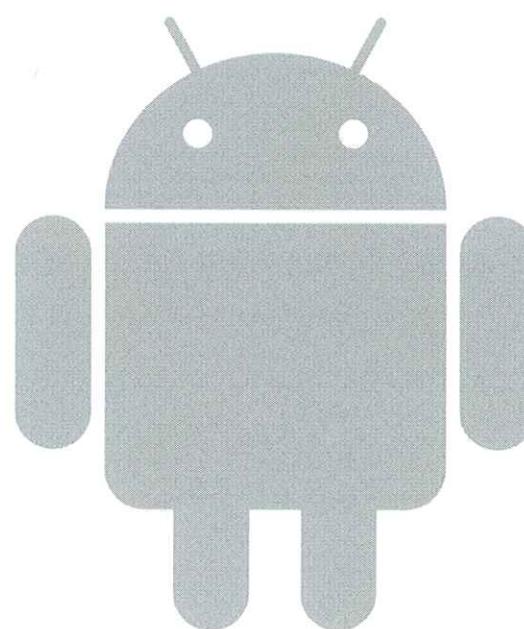
alfrad.com.br



Introdução

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.



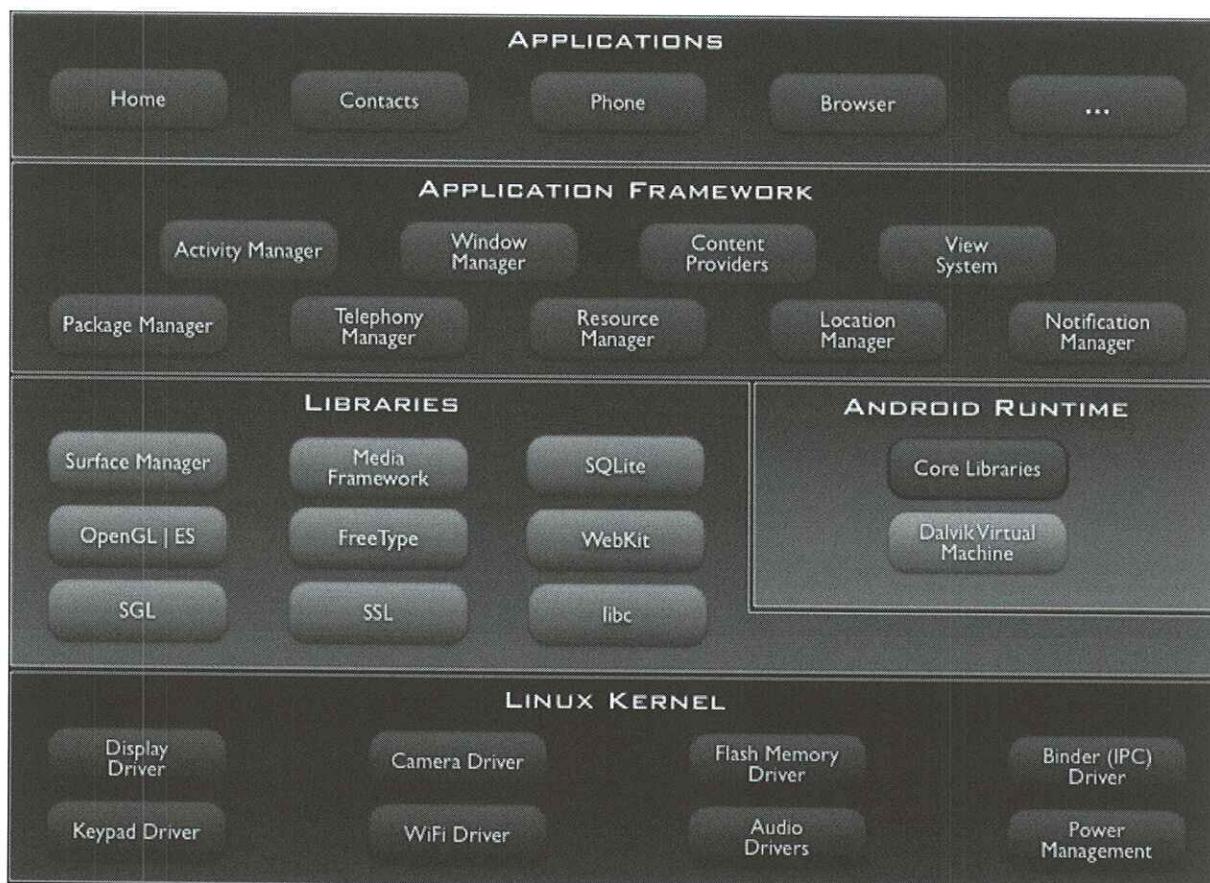
O que faz o Android diferente?

- 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, mapas, navegador, contatos entre outros. Todos os aplicativos são escritos utilizando a linguagem de programação Java.



- 
- A screenshot of the Android inbox. The title bar says "AndroidDev (769)" and the email address "leonardo.sobral@gmail.com". Below is a list of messages with checkboxes, subject lines, senders, and times:
- [androi...] Display some p... mansur Android 10h50
 - [androi...] dependency be... dnak, Dianne (3) 10h29
 - [androi...] accessing and c... zehunter .. remy (3) 10h16
 - [androi...] Opaque views - H... ankita.nhst, ~ (2) 10h10
 - [androi...] compatibility w... kec6227 .. Dianne, ~ (12) 9h56
 - [androi...] Drawable Setb... Ashwini 9h49
 - [androi...] Customize Spin... Ajay, Tim, Mark (4) 9h44
 - [androi...] Re: how to add ... ulqui, Mark (5) 9h21
 - [androi...] how to clear de... pli 9h11

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.

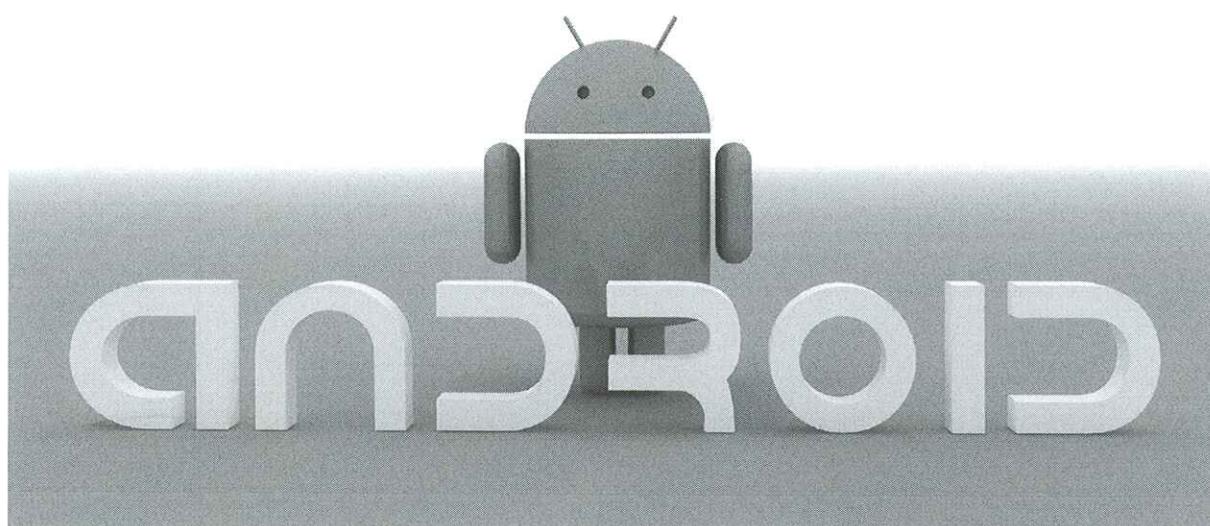
Introdução

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).

alfrad.com.br



Introdução

Introdução ao Android

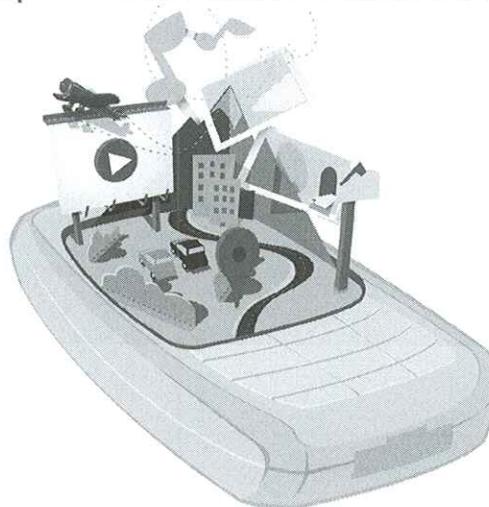
- Visão Geral e Histórico
- O que é o Android?
- O que faz o Android diferente
- Arquitetura
- Aplicações
- Application Framework
- Bibliotecas nativas
- Máquina virtual Dalvik
- Kernel GNU/Linux
- Android Market



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.



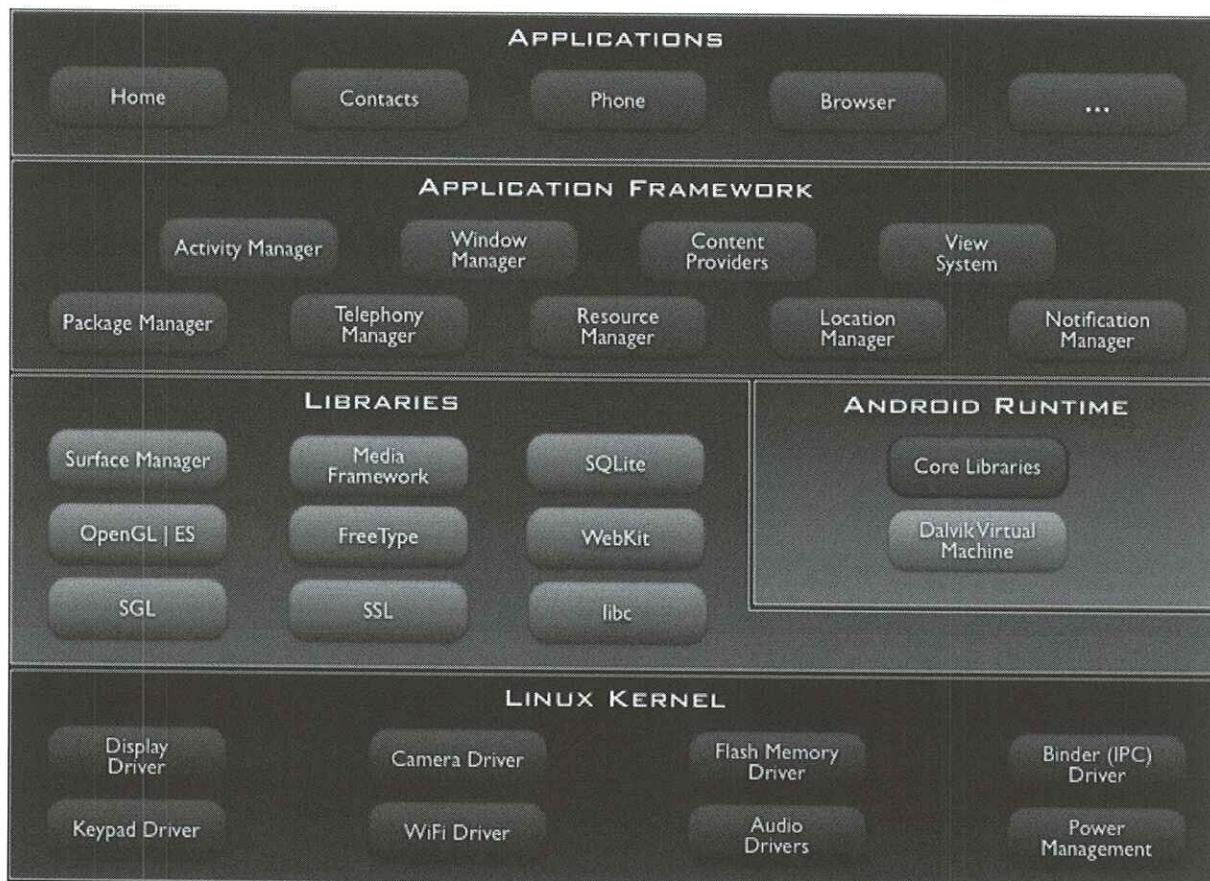
O que faz o Android diferente?

- 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

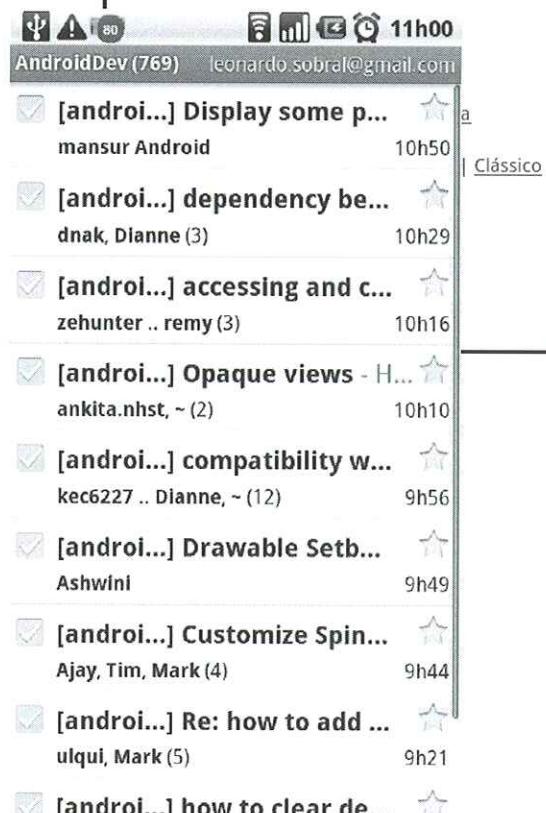
O diagrama abaixo mostra os principais componentes do sistema operacional Android. Cada seção é descrita em mais detalhes nas próximas páginas.



Aplicações

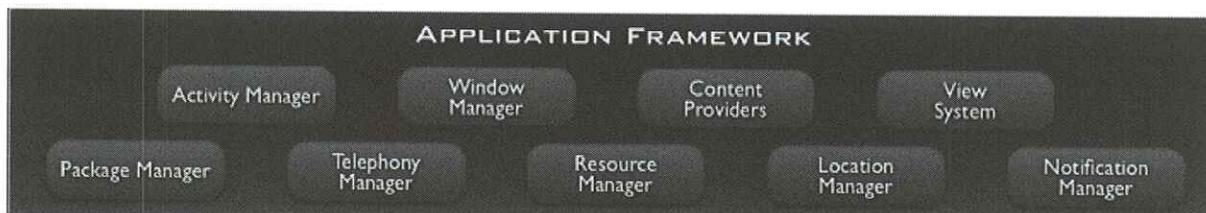
O Android vem com um conjunto de aplicações, incluindo um cliente de e-mail, programa de SMS, calendário, mapas, navegador, contatos entre outros. Todos os aplicativos são escritos utilizando a linguagem de programação Java.



- 
- A screenshot of the Android inbox. The title bar says "AndroidDev (769)" and the email address "leonardo.sobral@gmail.com". Below is a list of messages with their subjects and senders:
- [androi...] Display some p... mansur Android 10h50
 - [androi...] dependency be... dnak, Dianne (3) 10h29
 - [androi...] accessing and c... zehunter .. remy (3) 10h16
 - [androi...] Opaque views - H... ankita.nhst, ~ (2) 10h10
 - [androi...] compatibility w... kec6227 .. Dianne, ~ (12) 9h56
 - [androi...] Drawable Setb... Ashwini 9h49
 - [androi...] Customize Spin... Ajay, Tim, Mark (4) 9h44
 - [androi...] Re: how to add ... ulqui, Mark (5) 9h21
 - [androi...] how to clear de... pli 9h11

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.



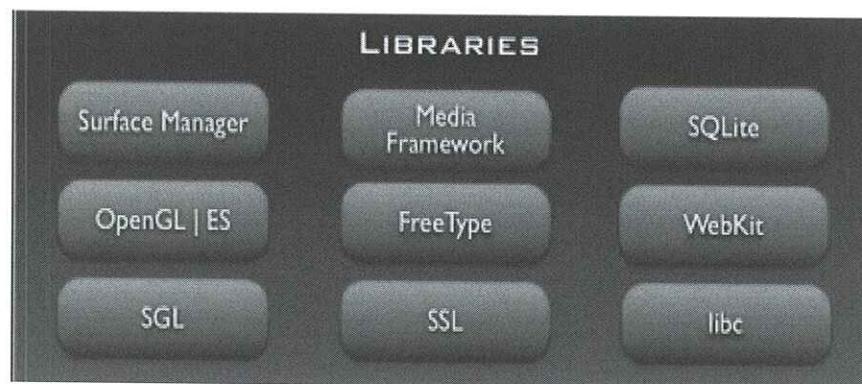
Desenvolvedores tem pleno acesso às mesmas APIs usadas pelos aplicativos nativos. A arquitetura do aplicativo é projetada para simplificar a reutilização dos componentes, qualquer aplicação pode publicar suas capacidades e qualquer outra aplicação pode então fazer uso destas capacidades (sujeito a restrições 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 acessem 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.

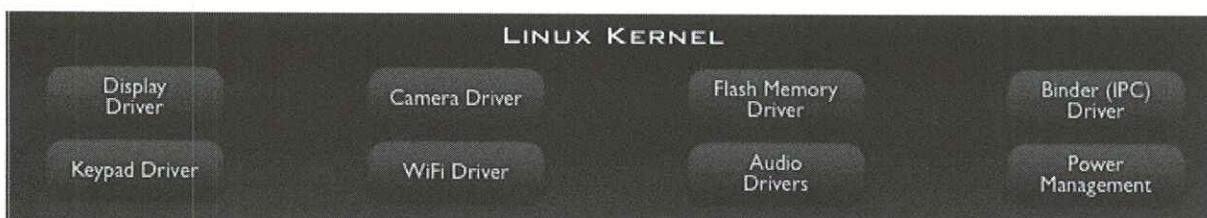


Cada aplicação Android roda em seu próprio processo, com a sua própria instância da máquina virtual Dalvik. A Dalvik foi escrita de modo que um dispositivo pode executar várias VMs eficientemente. A VM Dalvik executa os arquivos em formato executável Dalvik (.DEX) formato que é otimizado para uso mínimo 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.

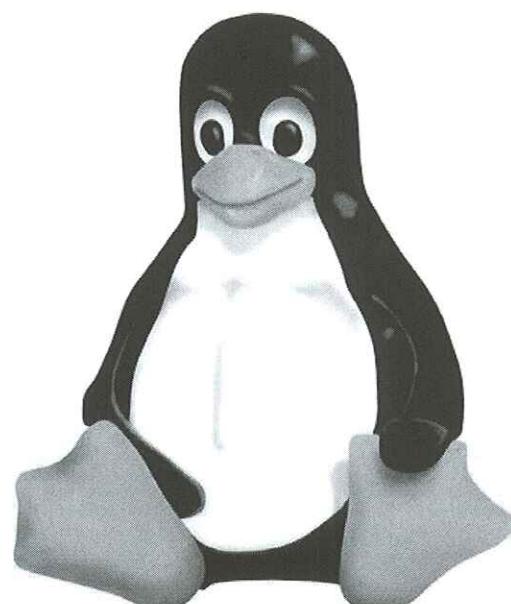
Linux Kernel

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.



	Nome	Desenvolvedor	Status	Avaliação
	Trapster	Trapster.com, Inc.	GRÁTIS	
	3banana Notes	Snaptic	GRÁTIS	
	Google Goggles	Google Inc.	Instalado	
	Handcent SMS	handcent_admin	GRÁTIS	

	Nome	Desenvolvedor	Status	Avaliação
	Taskiller free	Thibaut Nicolas	Instalado	
	Quick System Info	Shawn Q.	Instalado	
	twidroid for twitter	Zimmermann & Marban	Instalado	
	Wallpapers	Mabilo Android Team	Instalado	
	SNesoid Lite (SNES Emulator)	yongzh	Instalado	
	Nesoid Lite (NES Emulator)	yongzh	Instalado	
	Engadget	AOL Inc.	Instalado	



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



Instalar

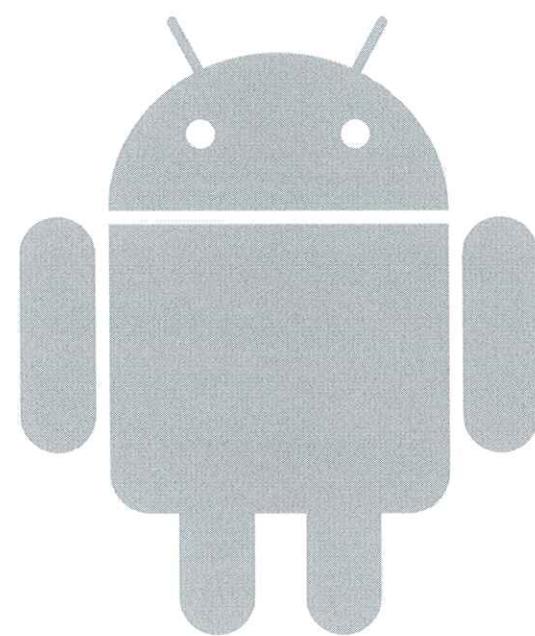
Questões de memorização

- 1 - O que é necessário para instalar um arquivo .apk em diversos aparelhos?
- 2 - Em qual ano formou-se a Open Handset Alliance?
- 3 - É sabido que no Android é possível substituir aplicações do sistema operacional, essa afirmação também aplica-se à aplicação da home?
- 4 - É necessário instalar algum plugin para depuração quando desenvolvendo aplicativos com o ADT e Eclipse?
- 5 - Qual API poderia ser utilizada para desenvolver aplicativos 3D no Android?
- 6 - Com quais linguagens é possível desenvolver aplicativos em Android?

Módulo 1

Gerenciamento Avançado de Interface

- Componentes complexos de interface
- Componentes de seleção
- Adapters
- Menus e diálogos



Outros Componentes complexos de interface

A seguir veremos um complemento das ferramentas de interface gráfica do Android. Algumas delas envolvem apenas alguns passos a mais do que os outros componentes, como é o caso dos menus de seleção, gridview, galeria, tab host, etc.

Embora esses componentes (que na verdade são um conjunto de GroupView com outros componentes View e - em alguns casos - adaptadores) pareçam complicados de serem criados, eles não o são.. apenas são necessários alguns passos a mais para configurar o componente final.

Criando componentes de seleção, gridview, e gallery, utilizando “Adaptadores”.

GridView

Um GridView é usualmente utilizado para exibir uma galeria de fotos, embora também possa ser utilizado para exibir outras Views dentro dele.

Para exibir um GridView faz-se necessário um ListAdapter, que pode facilmente ser criado extendendo BaseAdapter.

Abaixo vemos um exemplo de ListAdapter que demonstra a implementação de BaseAdapter, e do método `getView(int position, View convertView, ViewGroup parent)`, responsável por retornar a View (ou subclasses dela) de cada ítem do GridView.

```
public class CustomAdapter extends BaseAdapter{
    Context ctx;
    Integer[] imgs;

    public CustomAdapter(Context context, Integer[] images) {
        this.ctx = context;
        this.imgs = images;
    }

    @Override
    public int getCount() {
        return imgs.length;
    }
    @Override
    public Object getItem(int position) {
        return position;
    }
    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView,
ViewGroup parent) {
        ImageView iv = new ImageView(ctx);
        iv.setImageResource(imgs[position]);
        iv.setLayoutParams(new Gallery.
LayoutParams.LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_
CONTENT));
        return iv;
    }
}
```

Agora que já criamos nosso ListAdapter, podemos atribuí-lo à um GridView.

```
public class MainActivity extends Activity{  
  
    Integer[] imagens = {  
        R.drawable.img1,  
        R.drawable.img2,  
        R.drawable.img3,  
        R.drawable.img4  
    };  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.grid_view_teste);  
  
        final GridView gv = (GridView) findViewById(R.id.GridView01);  
  
        //Aqui vai o adaptador criado anteriormente  
        gv.setAdapter(new CustomAdapter(this, imagens));  
  
        gv.setOnItemClickListener(new AdapterView.  
OnItemClickListener() {  
  
            @Override  
            public void onItemClick(AdapterView parent,  
View v, int position,  
                long id) {  
                Toast.makeText(getApplicationContext(), "Imagen: " + imagens[position] + " foi clicada.", Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

E o arquivo XML que declara o GridView

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
        android:layout_height="fill_parent" android:layout_
        width="fill_parent"
        android:orientation="vertical">
    <TextView android:id="@+id/TextView01" android:layout_
        width="wrap_content"
            android:layout_height="wrap_content"
        android:text="Teste GridView"></TextView>
    <GridView android:id="@+id/GridView01" android:layout_
        width="wrap_content"
            android:layout_height="wrap_content"
        android:numColumns="auto_fit"
            android:columnWidth="50dip"></GridView>
</LinearLayout>
```

Gallery

Gallery é utilizada para formar a famosa galeria de imagens, que é encontrada em diversos programas do android, nativos e não nativos.

A Gallery, assim como o GridView, necessita de um ListAdapter para ser contruído, e - boa notícia - podemos utilizar o CustomAdapter criado no exemplo anterior.

```
public class GalleryActivity extends Activity{

    Integer[] imagens = {
        R.drawable.img1,
        R.drawable.img2,
        R.drawable.img3,
        R.drawable.img4
    };

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

        Gallery gallery = (Gallery)findViewById(R.id.Gallery01);
        gallery.setAdapter(new CustomAdapter(this, imagens));

        gallery.setOnItemClickListener(new AdapterView.
OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView parent,
View v, int position,
long id) {
                Toast.makeText(getApplicationContext(),
                imagens[position], Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

A seguir o XML que gera o Gallery:

```
<?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/TextView01" android:layout_
        width="wrap_content"
            android:layout_height="wrap_content"
        android:text="Utilizando Gallery"></TextView>
    <Gallery android:id="@+id/Gallery01" android:layout_
        width="fill_parent"
            android:layout_height="fill_parent"
            android:gravity="center"
        ></Gallery>
</LinearLayout>
```

Componentes de seleção

Os combos de seleção no Android, são chamados de Spinner.

O Spinner utiliza um ArrayAdapter como adaptador para exibir o conteúdo.

Abaixo conferimos uma Activity que referencia o Spinner criado no XML, e atribui valores à ele.

```
public class SpinnerTest extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        final String[] nomes = {"Nome 1" , "Nome 2" , "Nome 3" ,  
"Nome 4" , "Nome 5"};  
  
        //Cria o adaptador para ser consumido pelo spinner  
        ArrayAdapter<String> adaptador = new ArrayAdapter<String>(getApplicationContext(), android.R.layout.simple_spinner_item,nomes);  
        adaptador.setDropDownViewResource(android.R.layout.simple_spinner_item);  
  
        Spinner spinner = (Spinner)findViewById(R.id.Spinner01);  
        spinner.setAdapter(adaptador);  
  
        spinner.setOnItemSelectedListener(new AdapterView.  
OnItemSelectedListener() {  
  
            @Override  
            public void onItemSelected(AdapterView  
parent, View v,  
                int posicao, long id) {  
  
                    Toast.makeText(getApplicationContext(), "Nome selecionado: " + nomes[posicao], Toast.LENGTH_SHORT).  
show();  
  
                }  
  
            @Override  
            public void onNothingSelected(AdapterView  
arg0) {  
  
                }  
            }) ;  
    }  
}
```

XML que configura o layout da 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/txt_tela_spinner"
    />
<Spinner android:id="@+id/Spinner01" android:layout_width="wrap_
    content" android:layout_height="wrap_content"></Spinner>
</LinearLayout>
```

Diálogos - Criando um diálogo de confirmação

A API do Android oferece recursos muito interessantes para criação de diálogos.

Existem diversas maneiras de criar um diálogo, mas talvez o mais prático e rápido deles é através da classe `AlertDialog.Builder`, conforme demonstrado a seguir:

```
public class DialogoActvt extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        Button btn = (Button)findViewById(R.id.Button01);  
        btn.setOnClickListener(new View.OnClickListener() {  
  
            @Override  
            public void onClick(View arg0) {  
                AlertDialog.Builder builder = new  
                AlertDialog.Builder(DialogoActvt.this);  
                builder.setMessage("Deseja  
continuar?")  
                    .setCancelable(false)  
                    .setIcon(android.R.drawable.arrow_  
down_float)  
  
                    //Utilizando um recurso de imagem da  
API do Android, mas poderia ser utilizado qualquer imagem presente  
no projeto  
                    .setTitle("E então...")  
                    .setPositiveButton("Continuar", new  
DialogInterface.OnClickListener() {  
                        public void  
onClick(DialogInterface dialog, int id) {  
                            Toast.makeText(getApplicationContext(), "Clicou em continuar!", Toast.LENGTH_SHORT).  
show();  
                        }  
                    })  
                    .setNegativeButton("Cancelar", new  
DialogInterface.OnClickListener() {  
                        public void  
onClick(DialogInterface dialog, int id) {  
                            // put your code here  
                            dialog.cancel();  
                        }  
                    });  
  
                    AlertDialog alertDialog = builder.  
create();  
                    alertDialog.show();  
            }  
        });  
    }  
}
```

Menus

A adição de menu à sua aplicação resume-se a sobreescriver o método `onCreateOptionsMenu(menu)`. Esse método é contido em Activity com uma implementação vazia, ou seja, se não for sobreescrito, a aplicação não executará nenhuma ação quando o botão menu for pressionado.

Abaixo vemos a adição de um menu na aplicação e a chamada do método `onOptionsItemSelected(MenuItem item)`, responsável por tratar os eventos do menu.

```
public class MenuActivity extends Activity {

    //Identificadores dos itens do menu
    private final int MENU_ITEM_ABRIR_PÁGINA_DA_INTERNET = 0;
    private final int MENU_ITEM_EXIBIR_ALERTA = 1;

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, MENU_ITEM_ABRIR_PÁGINA_DA_INTERNET, 0,
        "Abrir página da internet");
        menu.add(0, MENU_ITEM_EXIBIR_ALERTA, 0, "Exibir um
        alerta");
        return true;
    }

    //Chamado quando um ítem do menu é selecionado
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case MENU_ITEM_ABRIR_PÁGINA_DA_INTERNET:
            abrirPagina();
            break;
        case MENU_ITEM_EXIBIR_ALERTA:
            Toast.makeText(getApplicationContext(),
            "Ítem do menu clicado!", Toast.LENGTH_SHORT).show();
            break;
        default:
            // put your code here
        }
        return false;
    }
}
```

Também é possível definir um menu em um arquivo XML (que deve ser inserido dentro de /res/menu), e “inflá-lo” pelo código:

menu_aplicacao.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="Item 1" android:id="@+id/item01"></item>
    <item android:id="@+id/item02" android:title="Item 2"></item>
</menu>
```

Na activity, basta utilizar um MenuInflater, dentro do método onCreateOptionsMenu(Menu menu):

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    MenuInflater inflater = getMenuInflater();
    //Arquivo XML de referência
    inflater.inflate(R.menu.menu_aplicacao, menu);
    return true;
}
```

Questões de memorização

1 - O que é necessário para que um componente Gallery funcione?

Precisa do Adapter

2 - Qual é a diferença entre as propriedades layout_gravity e gravity?

gravity → do componente com seu interior

layout_gravity → do componente com o exterior

3 - Quais as maneiras de construir uma aplicação que só exibirá uma lista?

listActivity

listView

4 - Como são chamados os componentes de seleção combobox no Android?

Spinner

5 - Em qual método podemos salvar dados quando a aplicação entra em pausa? onSaveInstanceState usando bundle.

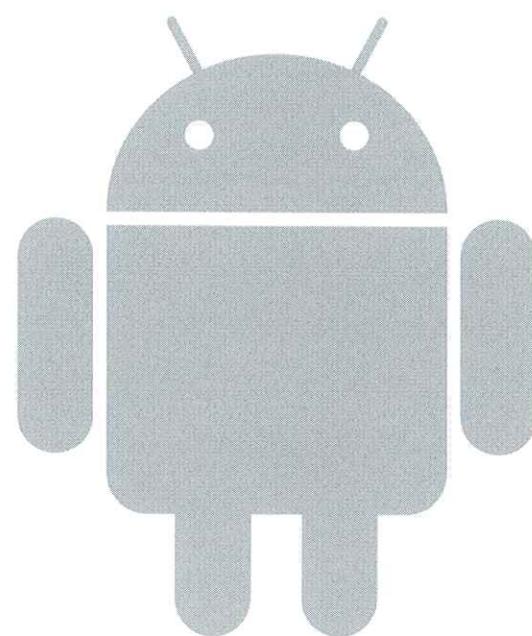
(tela)

6 - Qual método devemos sobrescrever para configurar menus em nossa aplicação? onCreateOptionsMenu (menu menu)

Módulo 2

Acesso a dados na internet

- Integrando com um Servlet
- Enviando informações para um Servlet
- Consumindo Webservices



Acesso a dados na internet

O Android disponibiliza diversas maneiras de buscar dados na internet. Por aceitar qualquer biblioteca java, é possível inclusive acessar web services utilizando bibliotecas de SOAP, bastando apenas importar os JAR's para dentro do projeto.

Agora seu instrutor irá montar um exemplo para vermos na prática o acesso a dados pelo Android.

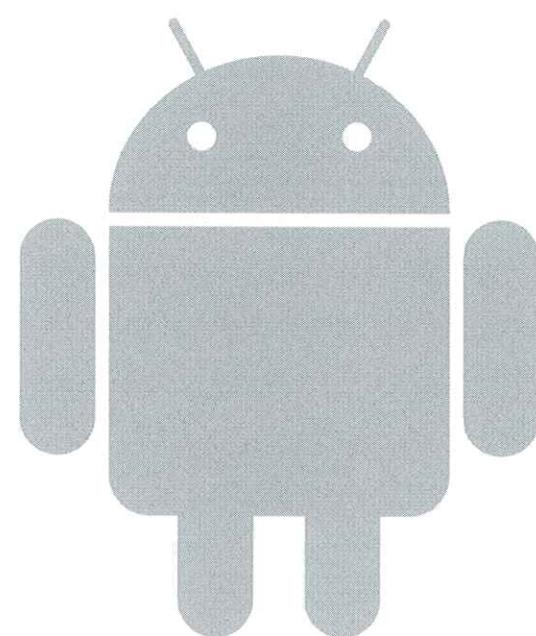
Questões de memorização

- 1 - Qual classe provê conexão por protocolo HTTP?
- 2 - É possível fazer requisições GET e POST?
- 3 - De classe é o objeto que o método openConnection() da classe URL retorna?
- 4 - O que faz o método getInputStream() de HttpURLConnection?
- 5 - O que retorna o método estático BitmapFactory.decodeByteArray()?
- 6 - Como funciona o método estático BitmapFactory.decodeStream([InputStream])?

Módulo 3

Multimídia

- Formatos de áudio e vídeo suportados
- Reprodução de áudio
- Reprodução de vídeo



Multimídia

Existem diversas maneiras de exibir áudio e vídeo no Android, mas a maneira mais fácil e rápida é utilizando as classes VideoView e MediaPlayer.

Os formatos de vídeo suportados pelo Android, até a data em que essa apostila estava sendo escrita, são mp4 e 3gp.

A seguir vemos um exemplo simples do uso de um VideoView, que exibe um vídeo dentro da aplicação:

```
public class VideoViewTest extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        VideoView videoView = new VideoView(this);  
        setContentView(videoView);  
  
        videoView.setVideoPath("/sdcard/my_video.3gp");  
        videoView.requestFocus();  
  
    }  
}
```

Para reproduzir um arquivo de áudio o processo é tão fácil quanto àquele de reproduzir um vídeo.

Os formatos de áudio suportados até a data em que essa apostila estava sendo escrita, são m4a, mid, ogg, wav, mp3 e 3gp.

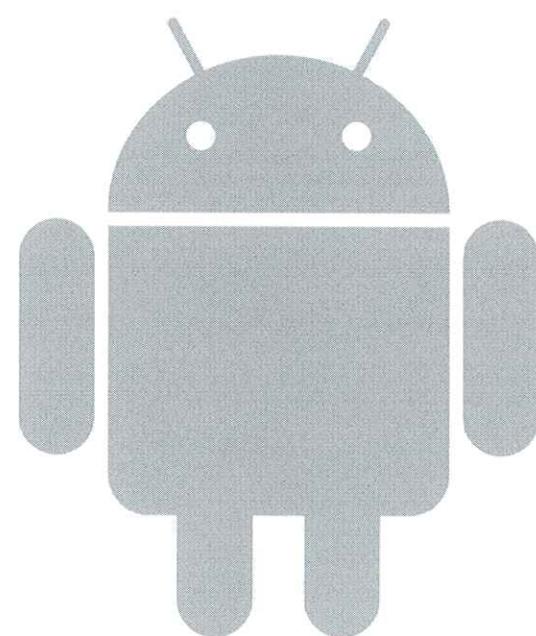
A seguir vemos um código que faz uso da classe MediaPlayer para reproduzir um arquivo de áudio que está dentro do nosso projeto:

```
public class SimpleAudioReproducer extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        MediaPlayer player = MediaPlayer.create(this, R.raw.pat_  
metheny);  
        player.start();  
  
    }  
}
```

Módulo 4

Camera

- Tirando fotos
- Montando um preview da imagem
- Trabalhando com o cartão de memória



Para utilizar a câmera não precisamos implementar a utilização em baixo nível. Utilizar a câmera é muito fácil, devido aos recursos de integração de aplicações, com o uso de Intentos.

Para utilizar a câmera, basta chamar o aplicativo responsável por exibi-la, e receber o resultado, coisa que pode facilmente ser configurada com o método `startActivityForResult(Intent)`.

A seguir demonstraremos um exemplo para ficar mais claro.

O programa abaixo chama o aplicativo da câmera, e após a foto ter sido tirada, retorna à aplicação que originou a chamada à câmera, cria um Bitmap com a foto tirada, e coloca esse bitmap na aplicação.

```
public class CameraActivityTest extends Activity {

    public static final int CODIGO_INTENT_CHAMAR_CAMERA = 0;
    Bitmap bitmap = null;
    ImageView imageView = null;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn = (Button) findViewById(R.id.Button01);
        imageView = (ImageView) findViewById(R.
id.ImageView01);

        if(null != savedInstanceState) {
            if(savedInstanceState.containsKey("imagem"))
{
                bitmap = (Bitmap)savedInstanceState.
getParcelable("imagem");
                imageView.setImageBitmap(bitmap);
            }
        }

        btn.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View arg0) {
                Intent chamarCamera = new
Intent("android.media.action.IMAGE_CAPTURE");
                startActivityForResult(chamarCamera,
CODIGO_INTENT_CHAMAR_CAMERA);
            }
        });
    }

    @Override
    public void onActivityResult(int requestCode , int
resultCode , Intent data){
        super.onActivityResult(requestCode, resultCode,
data);

        if(null != data){
            Bundle bundle = data.getExtras();
            if(null != bundle){
                bitmap = (Bitmap)bundle.get("data");
                imageView.setImageBitmap(bitmap);
            }
        } else {
            Toast.makeText(this, "Nenhuma foto foi
tirada", Toast.LENGTH_SHORT).show();
        }
    }
}
```

```
}

@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    if(bitmap != null)    outState.
putParcelable("imagem", bitmap);

}

}
```

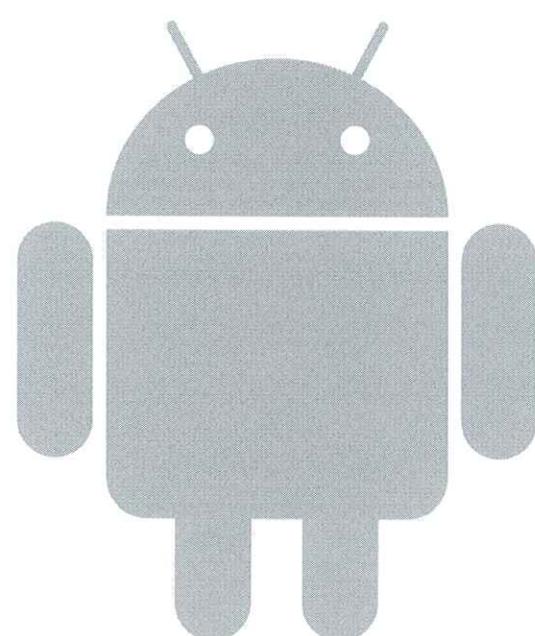
Questões de memorização

- 1 - Qual é o caminho para carregar a aplicação da câmera?
- 2 - Para iniciar a atividade da câmera, qual dos dois métodos de chamada de atividade deve ser utilizado?
- 3 - Ao capturar uma imagem e exibi-la na aplicação (em um ImageView), como é possível manter a exibição após mudar a orientação do celular (horizontal / vertical)?
- 4 - Qual é o tipo de objeto retornado pela aplicação da câmera?
- 5 - É possível apenas invocar a aplicação da câmera, capturar a imagem e gravá-la no sdcard ao invés de exibi-la na aplicação ou isso somente é feito implementando código de baixo nível?
- 6 - Através de qual objeto (propriedade de Intent) a imagem capturada é devolvida à aplicação?

Módulo 5

Mapas e GPS

- Exibindo um mapa
- Modos de visualização
- Controle de zoom
- Trabalhando com coordenadas
- GPS
- Chaves Google Maps



Exibindo um mapa

O MapView é um complemento extra da API do Android, e portanto, não vem junto com as bibliotecas padrões, sendo necessário baixar o pacote do Google API através do AVD Manager.

Antes de começar a utilizar o MapView, é necessário declará-lo no arquivo `AndroidManifest.xml` com a tag `<uses-library>`, dentro da tag `<application>`:

```
<uses-library android:name="com.google.android.maps" />
```

Uma vez que o MapView necessita da conexão com a internet para exibir os mapas, também temos que declarar a permissão de internet.

```
<user-permission android:name="android.permission.INTERNET" />
```

Assim como nos sites web, no Android é necessário o fornecimento de uma chave do google maps para que se possa utilizar o MapView.
Informações de como obter a chave podem ser obtidas no site

<http://code.google.com/intl/pt-BR/android/maps-api-signup.html>

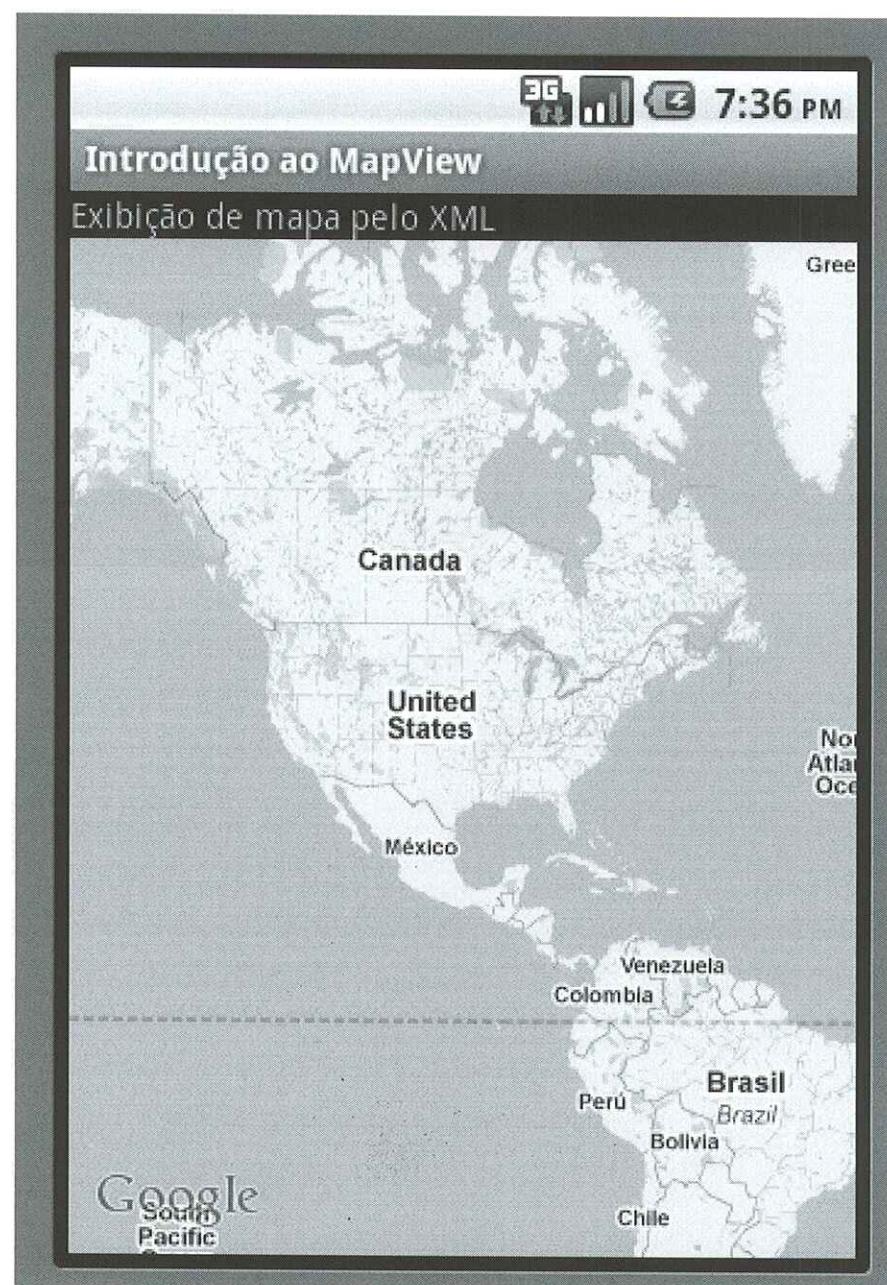
Criando um mapa simples por código

```
public class Atividade_Mapa extends MapActivity {  
  
    //Deve ser inserida a chave fornecida pelo google  
    String strChave = "";  
  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        MapView mapView = new MapView(this, strChave);  
  
        setContentView(mapView);  
    }  
  
    @Override  
    protected boolean isRouteDisplayed() {  
        return false;  
    }  
}
```

Notem que tivemos que sobrescrever o método `isRouteDisplayed()`. Esse método na verdade é um método abstrato, portanto precisa ser implementado.

O MapView também pode ser criado através de um arquivo de layout.

Conforme demonstrado a seguir:



```
<?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="Exibição de mapa pelo XML"
        />
<com.google.android.maps.MapView
        android:id="@+id/mapa"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true"
        android:apiKey="[sua_chave_aqui]"
        />
</LinearLayout>
```



```
public class Atividade_Mapa extends MapActivity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.main);  
  
        //Ainda sim é possível referenciar o mapView e tratá-lo da  
        maneira desejada  
        mapView = (MapView)findViewById(R.id.mapa);  
        MapController controlador = mapView.getController();  
        controlador.setZoom(12);  
  
        mapView.setBuiltInZoomControls(true);  
    }  
  
    @Override  
    protected boolean isRouteDisplayed() {  
        return false;  
    }  
}
```

Fizemos uso do método `setBuiltInZoomControls`, passando `true` como argumento, para que seja exibido o controle de zoom no mapa quando a tela é tocada.

Pegamos a referência ao controlador do mapa também, e nele setamos o tamanho do zoom para 12.

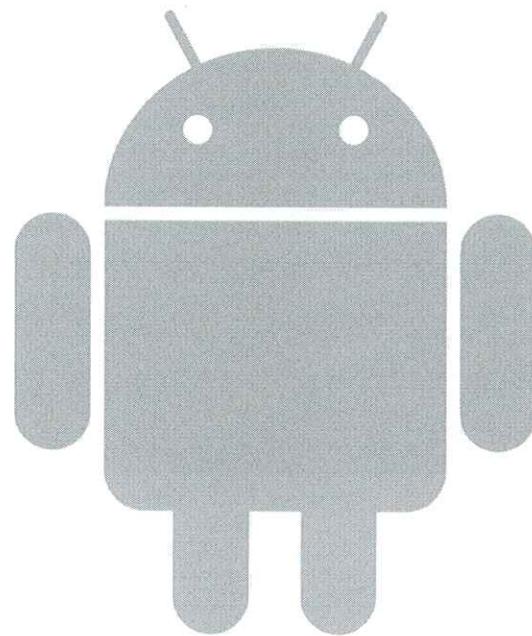
Questões de memorização

- 1 - O que é necessário para trabalhar com mapas no Android?
- 2 - É possível exibir um MapView em um emulador que estiver utilizando a API padrão do Google Android?
- 3 - É necessário declarar alguma permissão no AndroidManifest.xml para utilizar mapas?
- 4 - O que é necessário para instanciar um objeto do tipo MapView?
- 5 - Qual é o método que a documentação do Google exige que sobrescrevemos ao utilizar MapView?
- 6 - Qual é a maneira mais fácil de inserir controles de zoom em um MapView?

Módulo 6

Notificações do Sistema

- Criando notificações
- Adicionando ícones às notificações
- Controle de interações do usuário nas notificações



Ao arrastar a barra para baixo é possível visualizar a notificação completa, sendo possível clicar nesta notificação para executar alguma ação:



Colocamos um ícone utilizando um recurso de imagem padrão do Android, mas poderia ser utilizada qualquer imagem de dentro do projeto. O método `setLatestEventInfo()` recebe um `PendingIntent`, que é a Intent chamada quando o usuário clica na notificação da barra.

O XML que gera a tela é bem simples e não tem nada de mais. Apenas um botão para gerar a notificação:

```
<?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_height="wrap_content"
        android:text="@string/hello"
            android:gravity="center" android:layout_
    width="wrap_content"
                android:layout_gravity="center" />
    <LinearLayout android:id="@+id/LinearLayout01"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
            android:gravity="center">
        <Button android:id="@+id/Button01" android:layout_
    width="wrap_content"
                    android:layout_height="wrap_content"
        android:text="Gerar Notification"></Button>
        <Button android:id="@+id/Button02" android:layout_
    width="wrap_content"
                    android:layout_height="wrap_content"
        android:text="Cancelar Notificaiton"></Button>
    </LinearLayout>
</LinearLayout>
```

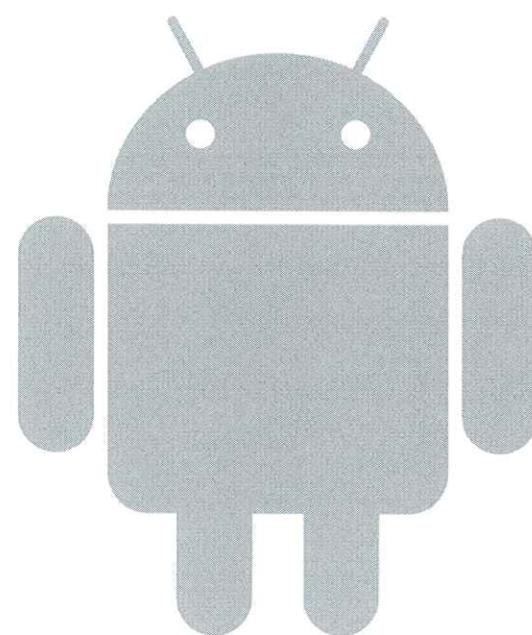
Questões de memorização

- 1 - Qual é o principal objetivo de uma notificação?
- 2 - Qual é a classe que gerencia as notificações do sistema?
- 3 - Qual é a classe responsável por exibir notificações?
- 4 - Qual método envia a notificação para o sistema?
- 5 - Qual é o método utilizado para cancelar uma notificação (removê-la da barra de notificações)?
- 6 - Que tipo de intent é utilizado para exibir uma notificação?

Módulo 7

Banco de dados SQLite

- Visão geral do SQLite
- Criação de um banco de dados
- Abertura e fechamento da conexão com um banco de dados
- Operações de SQL: listagens, inserções, edições e exclusões



- Visão geral do SQLite

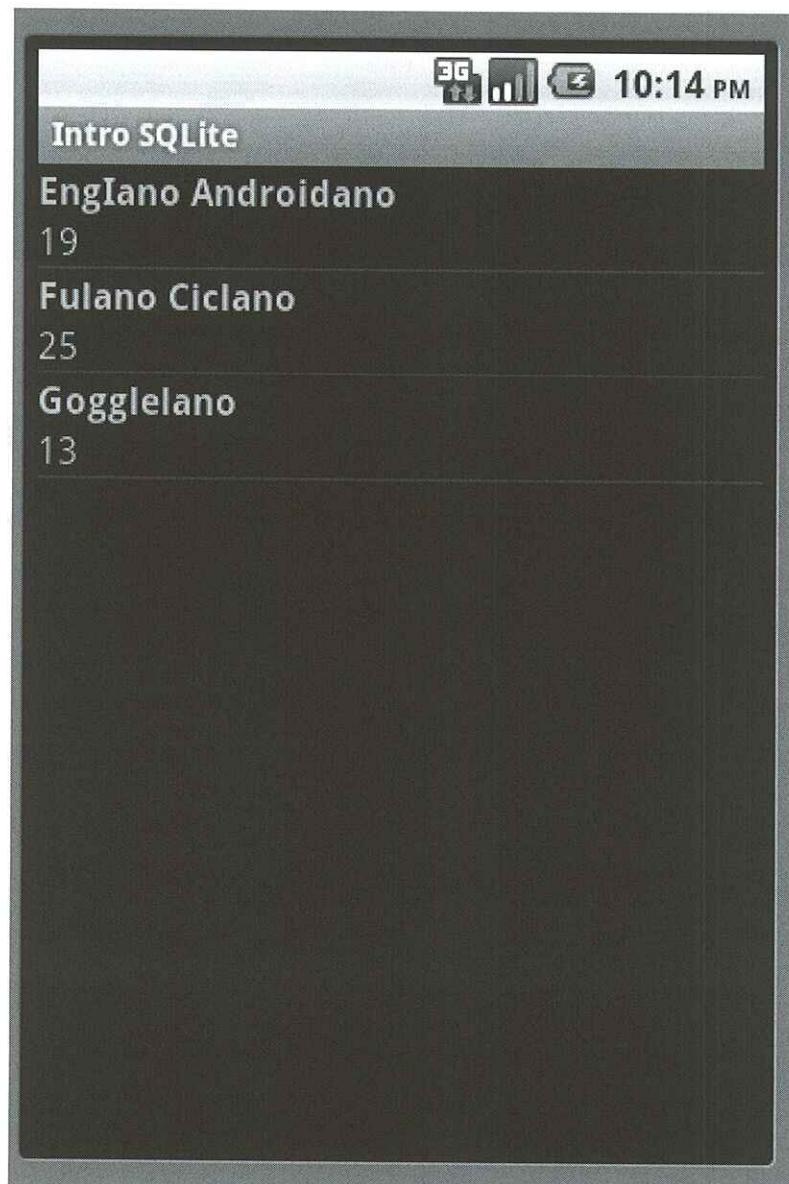
O banco de dados SQLite é muito poderoso e surgiu muito antes do Android existir. Hoje em dia é utilizado em diversas aplicações nas mais variadas arquiteturas.

No Android ele é nativo e a programação assemelha-se muito àquela do JDBC que estamos acostumados.

- Criação de um banco de dados

Existem duas maneiras de criar um banco SQLite. Uma é utilizando softwares gráficos, e a outra é através do próprio código da aplicação (utilizando um SQLiteOpenHelper), o que é extremamente útil em casos de atualização do banco em versões posteriores do seu software.

Abaixo vemos uma imagem do software Sqlditeman (<http://sqlditeman.com>), um utilitário gráfico para criação / manipulação de bancos de dados SQLite.



- Operações de SQL: listagens, inserções, edições e exclusões

Inserindo registros

Abaixo vemos uma demonstração de inserção de registros, que utiliza a classe ContentValues para especificar as colunas e valores, respectivamente, e a função insert([nome_tabela], [nullCollumHackk], [valores]) que retorna um long com o id do novo registro:

```
public class SQLiteTeste extends Activity {  
    SQLiteDatabase db = null;  
  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        //Nada de mais no layout do arquivo  
        setContentView(R.layout.main);  
  
        SQLiteHelper db_helper = new SQLiteHelper(this, "DB_  
EXEMPLO", null, 1);  
        db = db_helper.getWritableDatabase();  
  
        Toast.makeText(getApplicationContext(), "Id do novo  
registro: " + inserirRegistro("Primeiro nome", 25), Toast.LENGTH_  
SHORT).show();  
        Toast.makeText(getApplicationContext(), "Id do novo  
registro: " + inserirRegistro("Mais um nome", 19), Toast.LENGTH_  
SHORT).show();  
        Toast.makeText(getApplicationContext(), "Id do novo  
registro: " + inserirRegistro("Eu mesmo", 13), Toast.LENGTH_SHORT).  
show();  
  
    }  
  
    public long inserirRegistro(String nome, int idade){  
  
        ContentValues valores = new ContentValues();  
        valores.put("nome", nome);  
        valores.put("idade", idade);  
  
        return db.insert("tbl1", null, valores);  
    }  
}
```

Um exemplo mais completo

Abaixo um exemplo de uma ListView que exibe todos os cadastros do banco de dados e remove o ítem selecionado (do banco e da lista), e atualiza o adaptador.

Ao clicar em um ítem um alerta de confirmação é exibido pedindo para o usuário confirmar a ação de excluir, caso seja confirmada, o método apagarRegistro é chamado.

Embora não utilizado, o método editarRegistro(id) foi implementado para fins didáticos. Para utilizá-lo bastaria chamá-lo passando como argumento o id do ítem selecionado na lista.

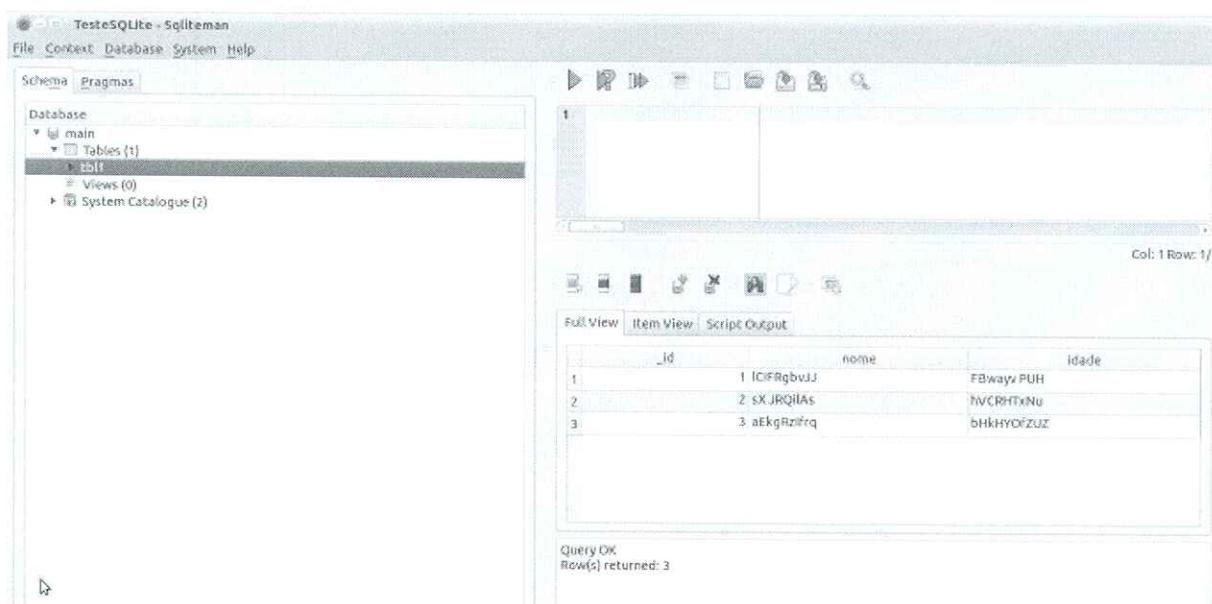
```
public class SQLiteTeste extends ListActivity {  
    SQLiteDatabase db = null;  
    String [] from = {"nome", "idade"};  
    int [] to = {android.R.id.text1, android.R.id.text2};  
    List<HashMap<String, Object>> lista = new ArrayList<HashMap  
<String, Object>>();  
    SimpleAdapter adapter = null;  
  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        SQLiteHelper db_helper = new SQLiteHelper(this, "DB_  
EXEMPLO", null, 1);  
        db = db_helper.getWritableDatabase();  
  
        inserirRegistro("Fulano Ciclano", 25);  
        inserirRegistro("EngIano Androidano", 19);  
        inserirRegistro("Gogglelano", 13);  
  
        Cursor registrosBanco = getAmigos();  
  
        if(registrosBanco.moveToFirst()) {  
            do{  
                HashMap<String, Object> amigo = new  
HashMap<String, Object>();  
                amigo.put("nome", registrosBanco.  
getString(registrosBanco.getColumnIndex("nome")));  
                amigo.put("idade", registrosBanco.  
getInt(registrosBanco.getColumnIndex("idade")));  
                amigo.put("_id", registrosBanco.  
getInt(registrosBanco.getColumnIndex("_id")));  
                lista.add(amigo);  
            } while(registrosBanco.moveToNext());  
        }  
        adapter = new SimpleAdapter(this, lista,  
        android.R.layout.two_line_list_item, from, to);  
  
        setListAdapter(adapter);  
    }  
    public long inserirRegistro(String nome, int idade){  
  
        ContentValues valores = new ContentValues();  
        valores.put("nome", nome);  
        valores.put("idade", idade);  
  
        return db.insert("tbl1", null, valores);  
    }  
  
    public int editarRegistro(long id, String nome, int idade){
```

```
ContentValues valores = new ContentValues();
valores.put("nome", nome);
valores.put("idade", idade);
return db.update("tbl1", valores, "_id=" + id, null);
}

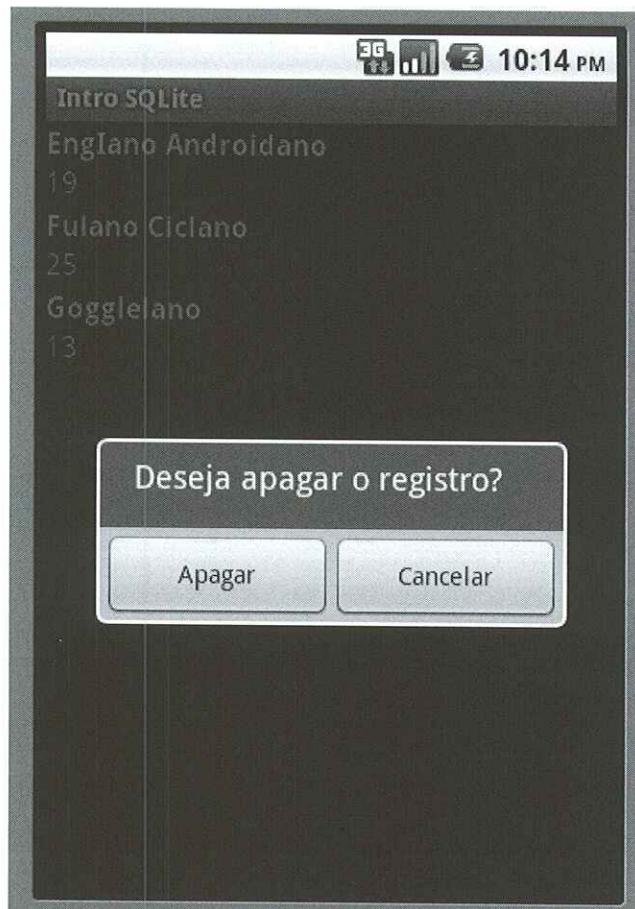
public int apagarRegistro(long id){
    Toast.makeText(getApplicationContext(), "Removido
item com id: " + id, Toast.LENGTH_SHORT).show();
    return db.delete("tbl1", "_id=" + id, null);
}

public Cursor getAmigos(){
    return db.query("tbl1", null, null, null, null, null,
"nome");
}

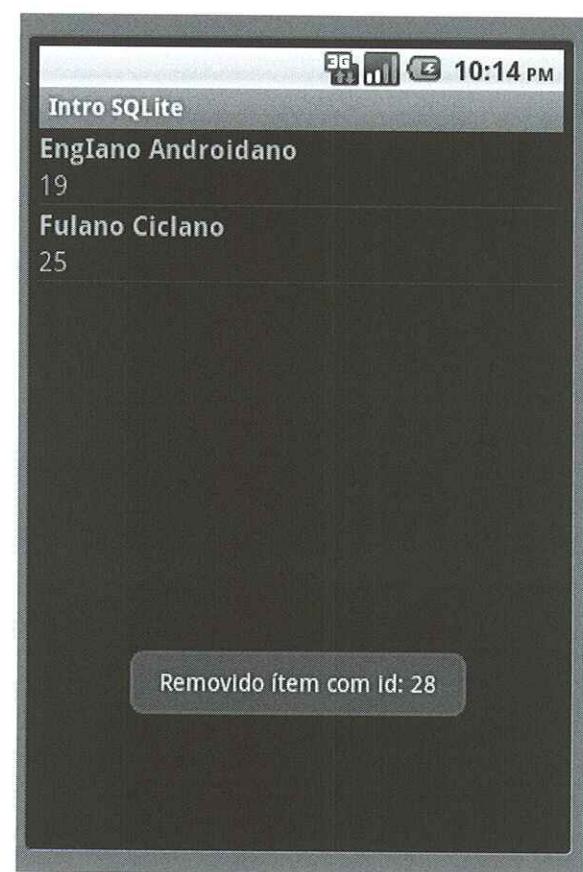
@Override
public void onListItemClick(ListView l, View v, int position,
long id){
    super.onListItemClick(l, v, position, id);
    final HashMap<String, Object> item = (HashMap<String,
Object>)this.getListAdapter().getItem(position);
    final int p = position;
    AlertDialog.Builder builder = new AlertDialog.
Builder(this);
    builder.setMessage("Deseja apagar o registro?")
.setCancelable(false)
.setPositiveButton("Apagar", new DialogInterface.
OnClickListener() {
        public void onClick(DialogInterface dialog,
int id) {
            apagarRegistro((Integer)item.get("_
id"));
            lista.remove(p);
            ((SimpleAdapter) getListAdapter()).
notifyDataSetChanged();
        }
    })
.setNegativeButton("Cancelar", new DialogInterface.
OnClickListener() {
        public void onClick(DialogInterface dialog,
int id) {
            dialog.cancel();
        }
});
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
}
```



Banco de dados sendo editado visualmente



O arquivo de criação do banco



Registro excluído ao clicar no ítem da lista

Para a criação do banco foi implementada uma subclasse de SQLiteOpenHelper, demonstrada a seguir:

```
public class SQLiteHelper extends SQLiteOpenHelper {  
  
    public SQLiteHelper(Context context, String name,  
CursorFactory factory,  
            int version) {  
        super(context, name, factory, version);  
        // TODO Auto-generated constructor stub  
    }  
  
    //Chamado apenas uma vez, quando o banco é criado.  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        String sqlCreate = "CREATE TABLE tbl1 (" +  
                            "_id INTEGER PRIMARY KEY  
AUTOINCREMENT NOT NULL," +  
                            "nome TEXT," +  
                            "idade TEXT" +  
                            ")";  
        db.execSQL(sqlCreate);  
    }  
  
    //Chamado quando o banco é atualizado (mudança na versão,  
    passada pelo usuário)  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int  
newVersion) {  
        //Ações mais complexas a serem executadas quando a  
versão do banco for atualizada  
    }  
}
```

Questões de memorização

1 - Qual é a classe utilitária que pode ser estendida para auxiliar na abertura de um banco de dados SQLite?

2 - Qual classe é utilizada para armazenar os valores a serem inseridos em uma tabela?

3 - Quais métodos CRUD disponíveis na API do SQLite?

4 - Qual é a classe de acesso a um banco SQLite?

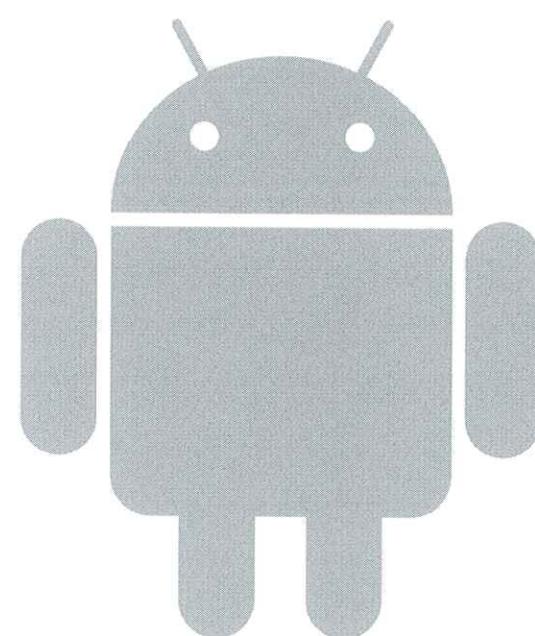
5 - Qual é o método para verificar se existem registros como resultado de uma seleção?

6 - Qual é o método para posicionar um Cursor para o primeiro registro?

Módulo 8

Provedores de Conteúdo

- Trabalhando com a agenda do telefone



Como faríamos se quiséssemos por exemplo selecionar um contato da agenda do telefone, a partir de nossa aplicação?

Poderíamos simplesmente chamar a Atividade nativa do Android que faz isso, através de uma intent e uma ação, conforme o código abaixo:

```
Uri uri = Uri.parse("content://com.android.contacts/contacts/");
Intent itVisualizarContatos = new Intent(Intent.ACTION_PICK,uri);

startActivityForResult(itVisualizarContatos,1);
```

Mas e se quiséssemos exibir os contatos EM NOSSA aplicação? Ou ainda, se quiséssemos manipulá-los, fazendo alterações, inserções e exclusões?

Para esse fim existem os Content Providers!

O código a seguir demonstra o uso de um provedor de conteúdo que busca todos os contatos da agenda e os exibe em uma `ListActivity`.

```
public class CProviderTeste extends ListActivity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        Uri uriContatos = android.provider.ContactsContract.  
Contacts.CONTENT_URI;  
  
        Cursor cursor = managedQuery(uriContatos, null, null,  
null, null);  
        startManagingCursor(cursor);  
  
        ArrayList<String> nomes = new ArrayList<String>();  
  
        if(cursor.moveToFirst()) {  
            do{  
                nomes.add(cursor.getString(cursor.  
getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME)));  
            }while(cursor.moveToNext());  
        }  
  
        CustomAdapter adapter = new CustomAdapter(this, nomes);  
        setListAdapter(adapter);  
    }  
  
    @Override  
    public void onListItemClick(ListView l, View v, int position,  
long id){  
    Toast.makeText(getApplicationContext(), "Escolhido: " +  
l.getAdapter().getItem(position), Toast.LENGTH_SHORT).show();  
    }  
}
```

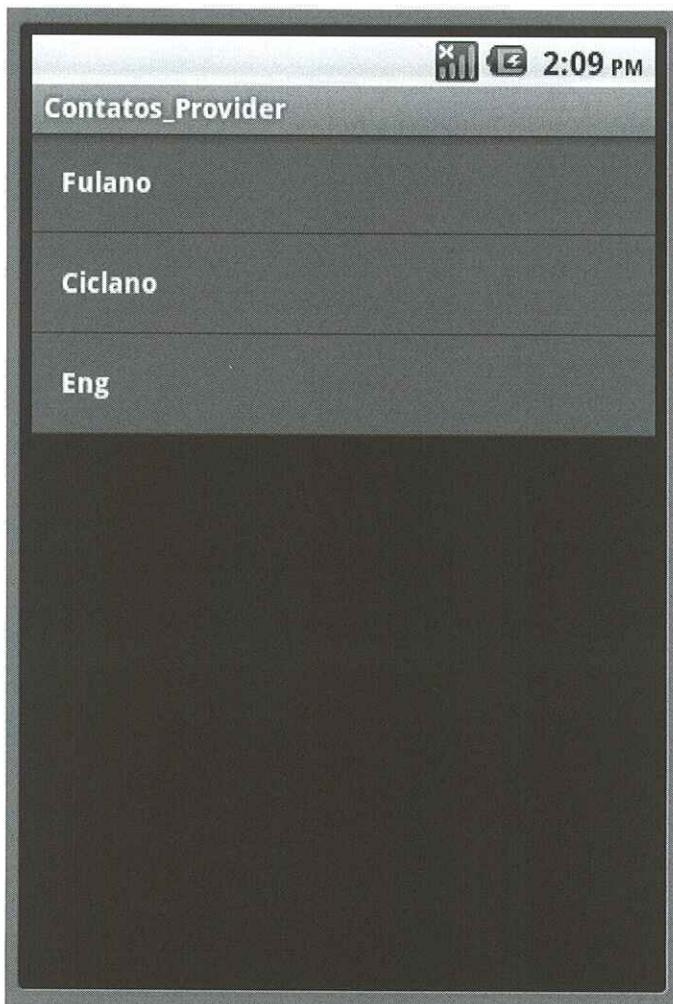
O código do CustomAdapter:

```
public class CustomAdapter extends BaseAdapter {  
    ArrayList<String> nomes = null;  
    Context context = null;  
  
    public CustomAdapter(Context context, ArrayList<String>  
nomes) {  
        this.nomes = nomes;  
        this.context = context;  
    }  
  
    @Override  
    public int getCount() {  
        return nomes.size();  
    }  
  
    @Override  
    public Object getItem(int position) {  
        return nomes.get(position);  
    }  
  
    @Override  
    public long getItemId(int position) {  
        return position;  
    }  
  
    @Override  
    public View getView(int posicao, View view, ViewGroup  
parent) {  
        LayoutInflator layoutInflater = (LayoutInflator)  
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
        View v = layoutInflater.inflate(R.layout.nome_agenda_  
layout, null);  
        TextView txtNome = (TextView)v.findViewById(R.  
id.txtNome);  
        txtNome.setText(nomes.get(posicao));  
  
        return v;  
    }  
}
```

E finalmente o XML que dita as regras de exibição do nosso ítem da lista.

```
<?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"
        android:background="#ffffffff">
    <TextView
        android:layout_width="fill_parent"
        android:text="@string/hello"
        android:id="@+id/txtNome"
        android:layout_height="50dip" android:gravity="center_
    vertical" android:background="#994444" android:textStyle="bold"
        android:textColor="#ffffffff" android:clickable="true"
        android:paddingLeft="15dp"/>
</LinearLayout>
```

Êis que temos a agenda de contatos DENTRO de nossa própria aplicação e portanto podemos customizar sua exibição da maneira que quisermos.



Questões de memorização

- 1 - Aonde é declarado um provedor de conteúdos?
- 2 - Qual é a ação utilizada em uma Intent para selecionar contatos da agenda telefônica?
- 3 - Qual é o método utilizado para retornar dados de um provedor de conteúdos, que utiliza uma URI como um de seus parâmetros?
- 4 - Em que formato é retornado o conjunto de dados de um provedor de conteúdos?
- 5 - Quais as operações podem ser disponibilizadas em um provedor de conteúdos?
- 6 - Qual é a sintaxe da uri de acesso a provedores de conteúdo?