

Desenvolvimento com Google

ANDROID

Level III



Release 001
2010

ENG

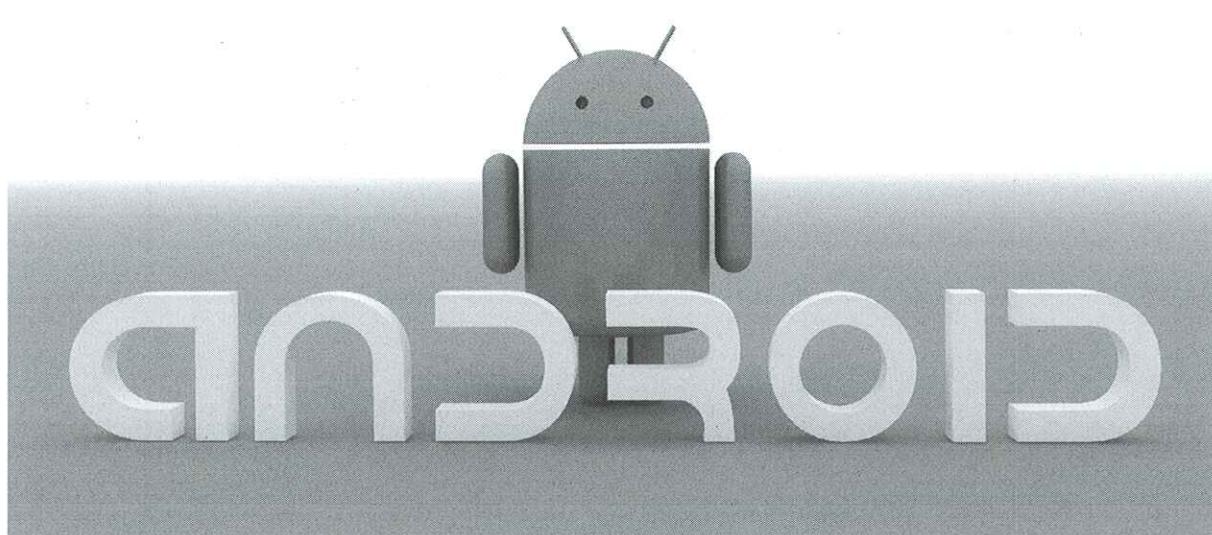
NÃO COPIAR OU DISTRIBUIR
ESTE DOCUMENTO. USO
INTERNO/TREINAMENTO.

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).

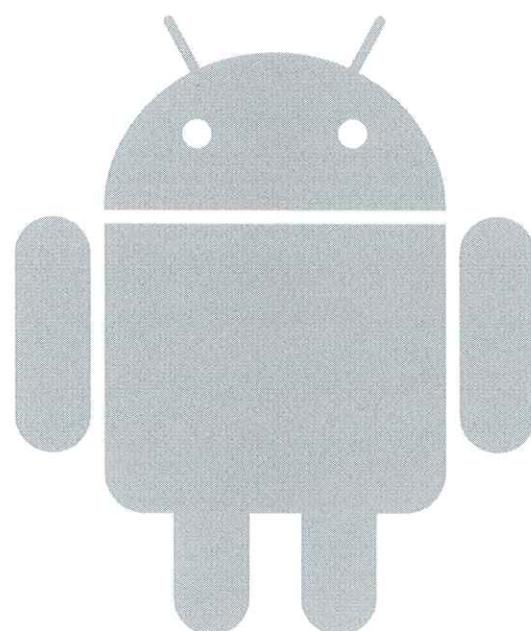
No Android quase tudo é possível, tendo como ponto forte a customização das aplicações nativas do celular, além da troca dessas por versões mais recentes ou modificadas, o que atrai os grandes fabricantes de dispositivos móveis (atualmente marcas potenciais como HTC, Samsung e Motorola estão vendendo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



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.



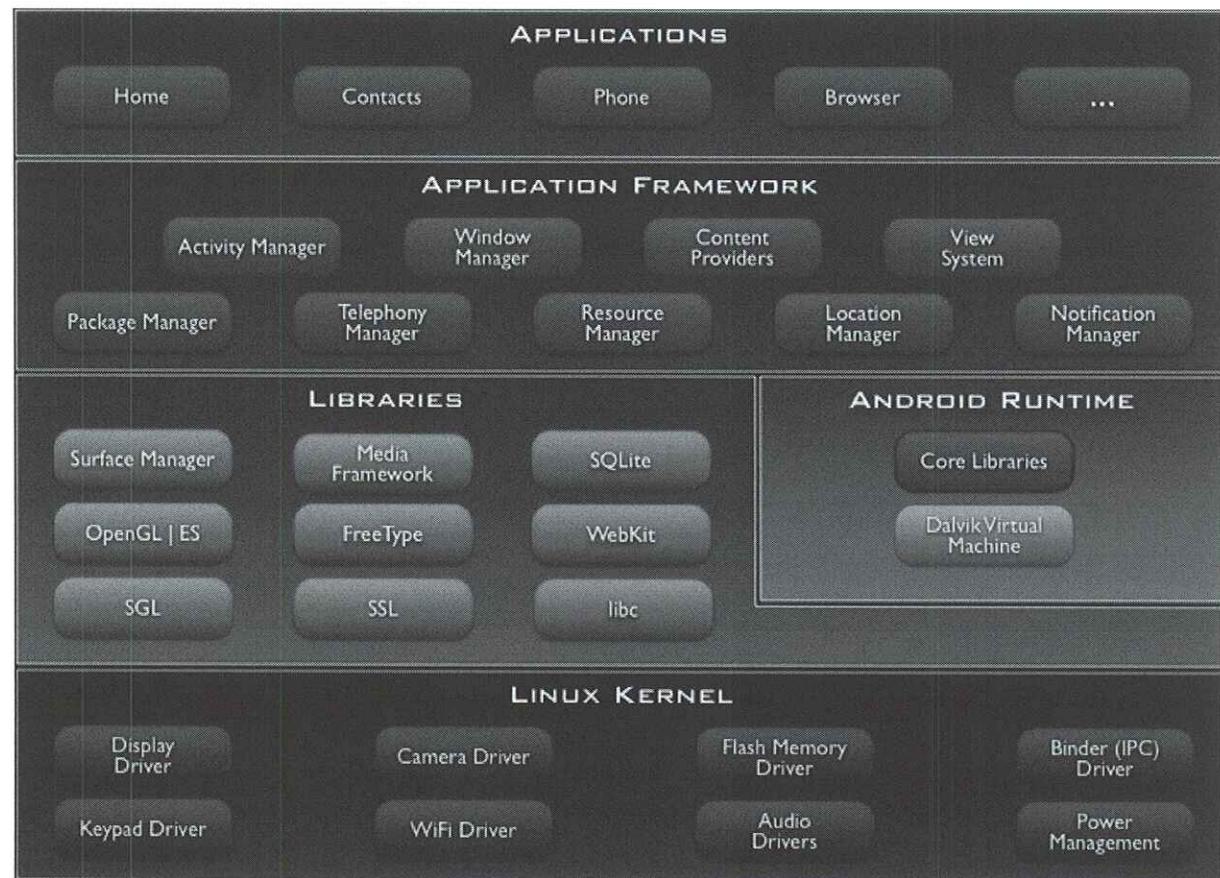
O que faz o Android diferente?

- 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. It lists several messages with checkboxes and star icons. The messages are:
- [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 Set... Ashwini 9h49
 - [androi...] Customize Spin... Ajay, Tim, Mark (4) 9h44
 - [androi...] Re: how to add ... ulqui, Mark (5) 9h21
 - [androi...] how to clear de... oli 9h16

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.



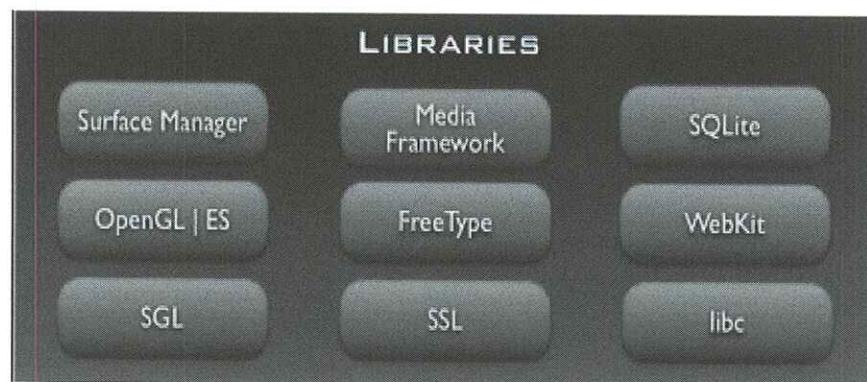
Desenvolvedores tem pleno acesso às mesmas APIs usadas pelos aplicativos nativos. A arquitetura do aplicativo é projetada para simplificar a reutilização dos componentes, qualquer aplicação pode publicar suas capacidades e qualquer outra aplicação pode então fazer uso destas capacidades (sujeito a restrições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

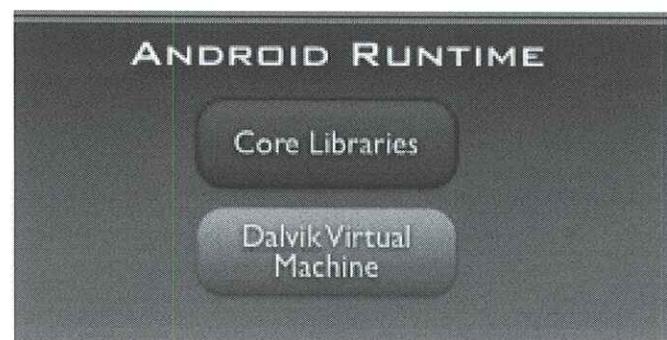


Algumas das principais bibliotecas estão listadas abaixo:

- **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.

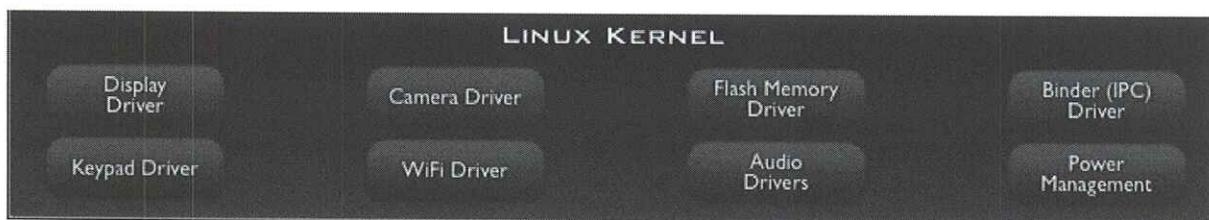


Cada aplicação Android roda em seu próprio processo, com a sua própria instância da máquina virtual Dalvik. A Dalvik foi escrita de modo que um dispositivo pode executar várias VMs eficientemente. A VM Dalvik executa os arquivos em formato executável Dalvik (.DEX) formato que é otimizado para uso mínimo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

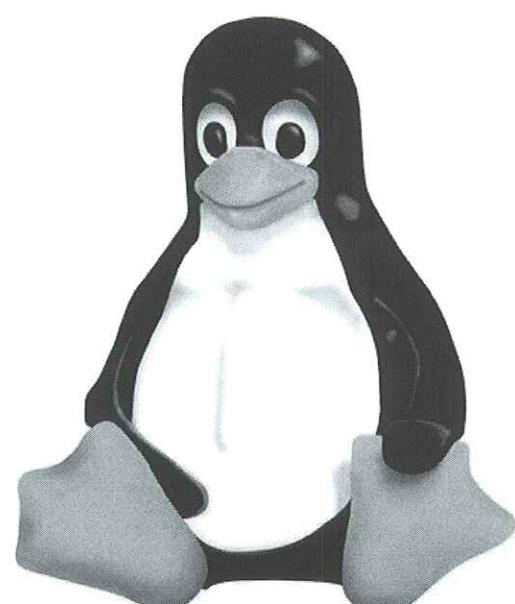
Linux Kernel

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.



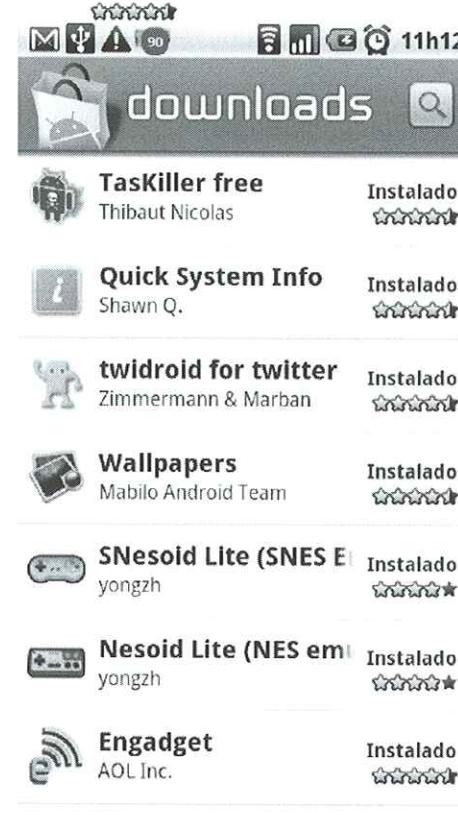
Android Market

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.



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



Questões de memorização

- 1 - Em qual plataforma é baseado o Android?
- 2 - Como é chamada a aliança responsável por manter o projeto do Android?
- 3 - Como é chamada a máquina virtual do Android?
- 4 - Qual é a extensão dos aplicativos Android?
- 5 - Em que ano o Google adquiriu a Android Inc?
- 6 - O browser integrado do Android é baseado em qual interpretador open source?

Anotações

exercício conceitual da página 40 - apostila nível I (mod II)

- 01) VideoView 02) MediaController 03) MediaPlayer
- 04) setPath() 05) reset(); 06) release();

(página 47)

- 01) android.media.action.Image-Capture é a action do Intent e estantá-la com startActivityForResult
- 02) startActivityForResult();
- 03) saveInstanceState ou autoOrientation pelo manifest configurando a atividade com configChanges
- 04) Parcelable que é convertido em Bitmap
- 05) é possível com File, BufferedWriter(file) e salvando byte a byte no arquivo

06) Bundle

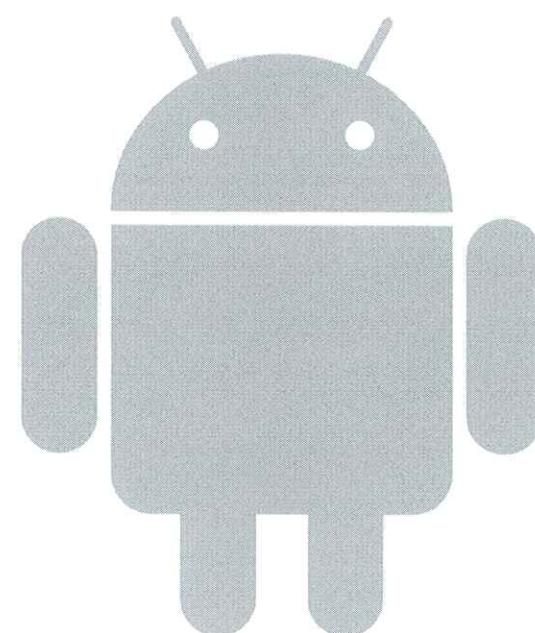
(64)

- 01) gerar um aviso sem intervir o usuário em sua ação executada no momento.
- 02) Notification manager 03) Notification
- 04) notify(); 05) cancel(); 06) PendingIntent.

Pasta databases da pasta data tem de ser criada manualmente via linha de comando usando mkdatabases
para causar tem de estar dentro platform-tools, depois comando adb shell

Módulo 1

Intent Filters



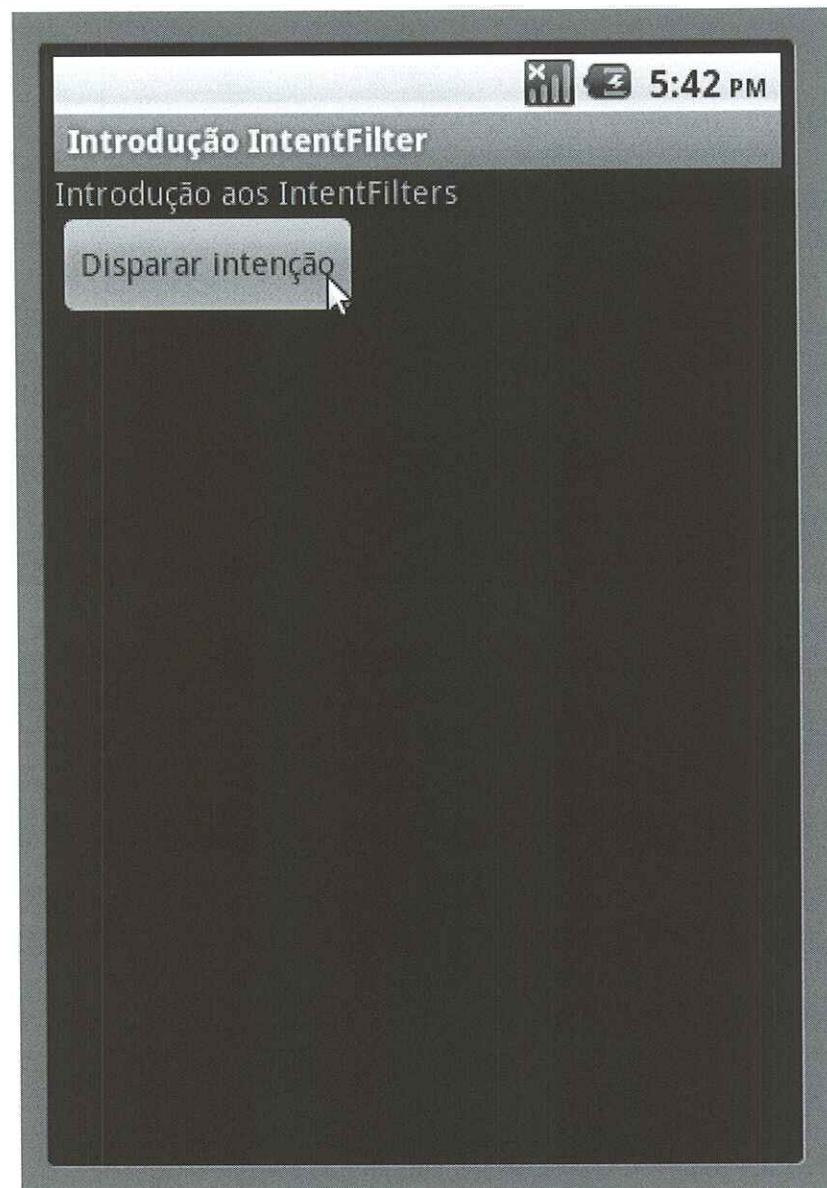
Intent Filters

Os Intent Filters são responsáveis por filtrar as mensagens enviadas ao sistema operacional.

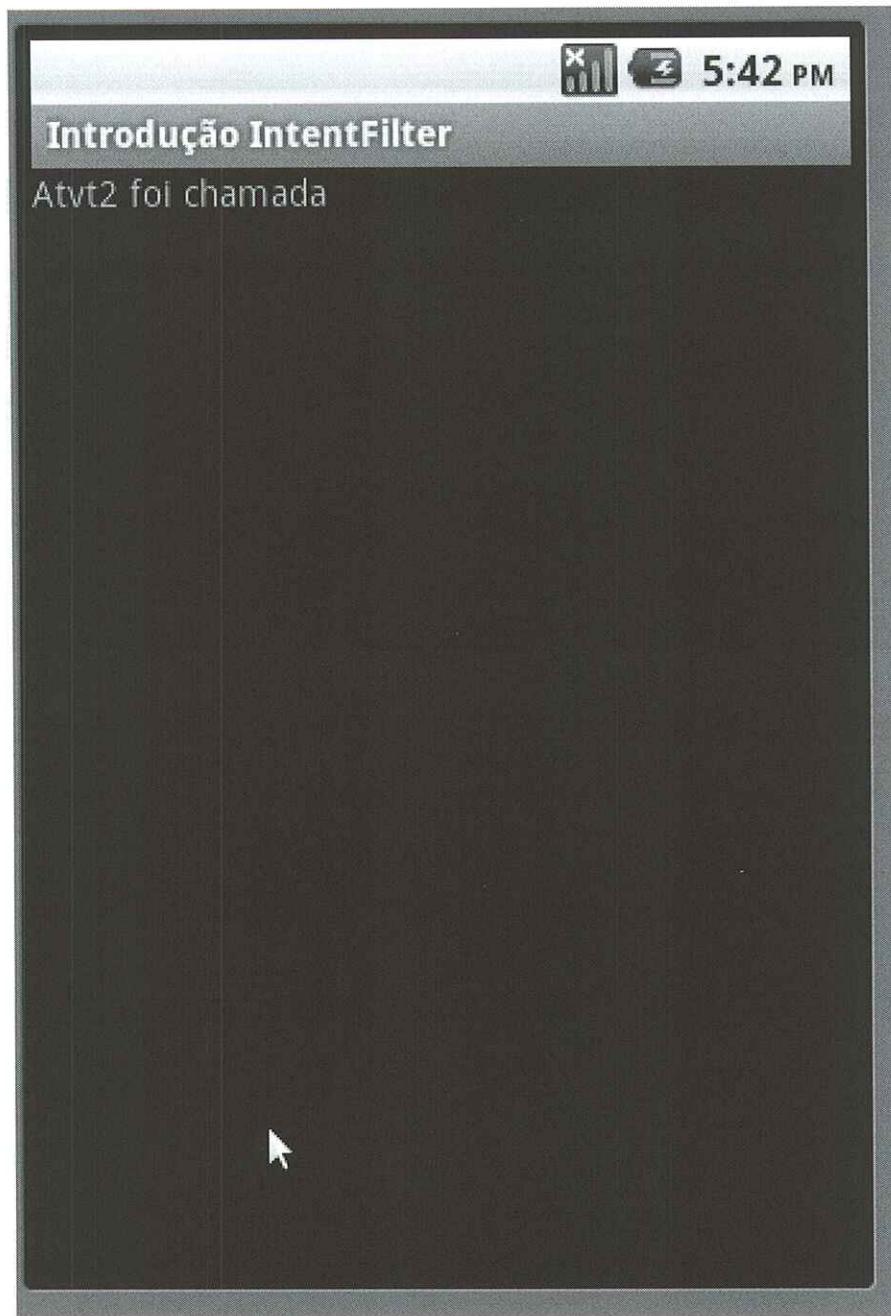
Quando disparamos uma nova intenção para o Android, na verdade estamos enviando à ele uma mensagem, para que ele decida o que fazer com essa mensagem. O sistema então verifica qual(is) aplicação(ões) que está(ão) filtrando essa mensagem e então inicia a aplicação.

```
Intent intencao = new Intent(this, SuaClasse.class);
startActivity(intencao);
```

Isso dispararia uma mensagem para o sistema operacional e abriria a classe SuaClasse.class



Tela de disparar intenção para abrir uma nova atividade



Nova atividade, após ter sido chamada utilizando literal de classe

Os benefícios do Intent Filter não param por aí... existe ainda a possibilidade de trabalhar com Ações e Categorias.

Quando o Android encontra dois intent filters com ações iguais, um prompt é exibido para o usuário escolher qual aplicação ele deseja abrir (qual intent executar). Esse é um recurso muito poderoso do Android para integrar aplicações.

Com isso é possível por exemplo, substituir a home screen do Android (nos projetos de exemplos que vêm junto com a API do Android é possível encontrar um exemplo de Custom Home Screen).

Abaixo vemos um exemplo de uma atividade sendo chamada por um intent que passa como parâmetro a ação, que por sua vez é capturada pelo intent filter da atividade chamada:

```
public class TesteIntentFilter 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) {  
                Intent intencao = new  
Intent("ATIVIDADE_2");  
                try{  
                    startActivity(intencao);  
                }catch(ActivityNotFoundException e){  
                    Toast.  
makeText(TesteIntentFilter.this, "Erro ao carregar a próxima  
tela", Toast.LENGTH_SHORT).show();  
                    Log.e("[CURSO ANDROID ENG]",  
"Erro ao abrir Intent: " + e.getLocalizedMessage());  
                }  
            }  
        });  
    }  
}
```

Temos então a segunda atividade, a qual será chamada pela primeira:

```
public class Atvt2 extends Activity {  
    /**  
     * @see android.app.Activity#onCreate(Bundle)  
     */  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        LinearLayout layout = new LinearLayout(this);  
        layout.setLayoutParams(new  
LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));  
  
        TextView txt = new TextView(this);  
        txt.setText("Atvt2 foi chamada");  
  
        layout.addView(txt);  
  
        setContentView(layout);  
    }  
}
```

O AndroidManifest configura o intentfilter da segunda atividade (Atvt2.java):

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest android:versionCode="1" android:versionName="1.0"
    package="com.intentfilter_intro" xmlns:android="http://schemas.
    android.com/apk/res/android">
    <application android:icon="@drawable/icon" android:label="@
    string/app_name">
        <activity android:label="@string/app_name" android:name=".TesteIntentFilter">
            <intent-filter>
                <action android:name="android.intent.action.
    MAIN"/>
                <category android:name="android.intent.category.
    LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:name=".Atvt2">
            <intent-filter>
                <action android:name="ATIVIDADE_2"/>
                <category android:name="CATEGORIA_ATVT_2"/>
                <category android:name="android.intent.
    category.DEFAULT"/>
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="8"/>
</manifest>
```

Questões de memorização

1 - Em um contexto geral, para que serve um intent-filter?

criar um padrão de inicialização

2 - É possível substituir a aplicação da home com um intent-filter?

Sim, com action_main ; android.intent.category.HOME.

3 - Se uma atividade possui um intent-filter com uma ação configurada, como é possível invocar essa atividade?

Setando uma categoria a ela [addCategory()] e configurar o manifest com tal categoria + Default no intent-filter.

4 - O intent-filter deve ser declarado e configurado em qual arquivo?

AndroidManifest.xml

5 - Caso uma categoria não seja especificada em um intent-filter, qual é a categoria que deve ser inserida?

Categoria Default

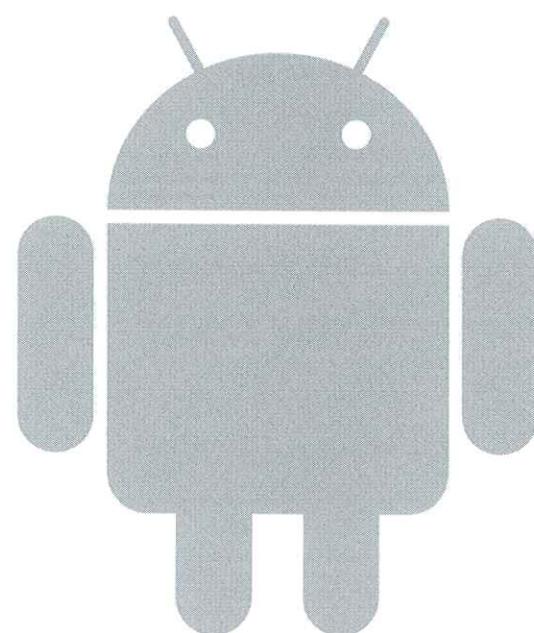
6 - É possível ter mais de uma categoria em um intent-filter?

Sim

Sempre que for atividade precisa de Action e Category
quando se faz uso do intent-filter; já os serviços e broadcastReceiver não é necessário ter categoria apenas a action.

Módulo 2

Broadcast Receivers



Broadcast Receivers

No capítulo anterior vimos o uso de intent-filter. Os broadcast receivers trabalham em conjunto com os intent-filter, pois ficam escutando por uma mensagem e ao receberem-na executam uma ação em segundo plano (que por definição, não deve atrapalhar o usuário ou interferir no que ele está fazendo no momento de execução do receiver).

Para disparar um broadcast basta chamar o método sendBroadcast(intent), e todas as aplicações serão notificadas desse broadcast. A aplicação que estiver configurada para receber aquela mensagem, iniciará o broadcastReceiver correspondente.

Para declarar um receiver basta inserir a tag <receiver> no AndroidManifest.xml, conforme demonstrado abaixo:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.condomitti.notifications"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".AppPrincipal"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>
            <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".ExemploReceiver">
            <intent-filter>
                <action android:name="GERAR_NOTIFICACAO"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </receiver>
    (...)
```

Como dito anteriormente, a tag <receiver> também faz uso do intent-filter, isso porque quando um broadcast é disparado, ele é capturado pelo intent-filter atrelado à cada broadcast receiver.

No capítulo 6 da apostila anterior vimos um exemplo de notificação na barra de status, que quando clicada cancelava a notificação. Esse exemplo foi feito utilizando um broadcast receiver, que recebia a ação “CANCELAR_NOTIFICACAO”.

Abaixo vemos esse exemplo novamente, mas dessa vez com todas as partes do código:

Primeiramente nossa Activity que contém o botão para exibir e cancelar a notificação (o botão cancelar é apenas para demonstrar o comportamento de cancelar a notificação, coisa que é feita pelo broadcast receiver ao clicar na mensagem na barra de notificações):

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

        Button btn1 = (Button)findViewById(R.id.Button01);
        btn1.setOnClickListener(new View.OnClickListener()
    {

        @Override
        public void onClick(View arg0) {
            int notificationID = 10;
            NotificationManager notificationManager
= (NotificationManager) getSystemService(Context.NOTIFICATION_
SERVICE);
            Notification notification = new
Notification(android.R.drawable.btn_radio, "Nova interação...", 
System.currentTimeMillis());
            Intent intent = new
Intent("CANCELAR_NOTIFICACAO");
            PendingIntent pendingIntent =
PendingIntent.getBroadcast(Notificacao.this, 0, intent, 0);
            notification.
setLatestEventInfo(Notificacao.this, "Atenção", "Só clicar para
sumir...", pendingIntent);
            //notification.vibrate = new long[]
{2000 , 250 , 100 , 500};
            notificationManager.
notify(notificationID, notification);

        }
    });

    Button btn2 = (Button)findViewById(R.id.Button02);
    btn2.setOnClickListener(new View.OnClickListener()
    {

        @Override
        public void onClick(View arg0) {
            int notificationID = 10;
            NotificationManager notificationManager
= (NotificationManager) getSystemService(Context.NOTIFICATION_
SERVICE);
            notificationManager.
cancel(notificationID);

        }
    });
}
```

Abaixo vemos o BroadcastReceiver:

```
public class ExemploReceiver extends BroadcastReceiver {  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        int notificationID = 10;  
        NotificationManager notificationManager =  
(NotificationManager) context.getSystemService(Context.  
NOTIFICATION_SERVICE);  
        notificationManager.cancel(notificationID);  
    }  
}
```



Notificação recebida; Permanece na barra que seja clicada, ou o botão “Limpar” (em inglês “Clear”) seja pressionado

Questões de memorização

1 - Em qual lugar da aplicação são declarados os broadcast receivers?

AndroidManifest.xml

2 - Qual é o método utilizado para enviar um broadcast para o sistema?

sendBroadcast();

3 - Qual é a relação entre BroadcastReceiver e intent-filter?

BR so responde pela configuração do intent-filter.

4 - Como é possível configurar um broadcast para receber chamadas de mensagens?

5 - Caso nenhuma categoria seja configurada para o broadcastreceiver, é necessário declarar alguma?

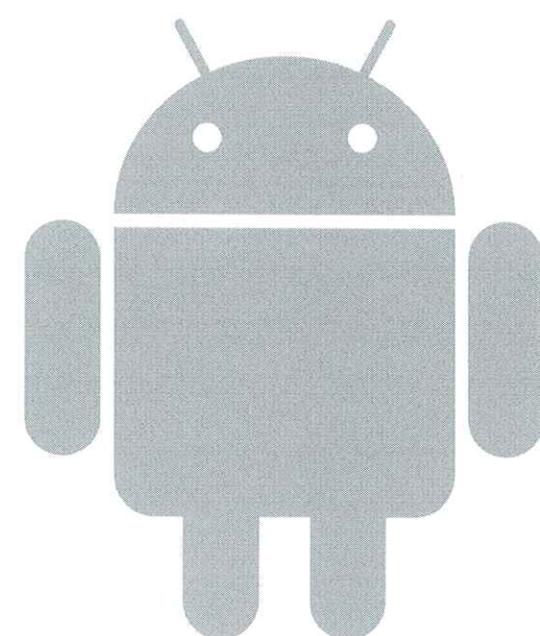
Sim

6 - Cite pelo menos dois exemplos de utilização de um broadcast receiver.

é possível usar o método registerReceiver(new ClassBR(), new IntentFilter('qualquer')) e no método onReceive é necessário desregister o BR com unregisterReceiver(new BR), no manifest basta declarar a tag receiver name... />

Módulo 3

Serviços



Conforme vimos no capítulo anterior, os Broadcast receivers servem para executar uma ação em segundo plano.

O problema é que eles são limitados a 10 segundos de execução, caso esse tempo seja ultrapassado, uma mensagem de erro de timeout é exibida pelo android (chamada de ANR - Application not responding).

Isso significa que o broadcastReceiver deve ser utilizado para executar ações rápidas.

Para execuções mais demoradas é recomendado o uso da classe Service ao invés de broadcastReceiver.

A classe Service segue o mesmo conceito de broadcast receiver, de executar apenas em segundo plano, sem causar efeitos visuais à ao usuário.

Um serviço pode ser iniciado de duas maneiras:

- através do método startService()
- ou
- bindService()

A diferença entre os dois é que com o segundo é possível ter acesso à interface de recursos do serviço. Por exemplo, em um serviço que toca uma música em segundo plano, pode oferecer maneiras de controlar o andamento da música, e isso só seria possível através do método bindService.

O primeiro método apenas inicia o serviço, que fica ativo até que alguém chame o método stopService(intent) seja chamado ou o próprio serviço chame o método stopSelf(). Ainda que a atividade que iniciou o serviço foi fechada, ele CONTINUA ATIVO (a menos que tenha sido iniciado com o método bindService).

Para os serviços também é necessário declá-los no arquivo **AndroidManifest.xml**, conforme demonstrado abaixo:

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest android:versionCode="1" android:versionName="1.0"
    package="com.service_intro" xmlns:android="http://schemas.
    android.com/apk/res/android">
    <application android:icon="@drawable/icon" android:label="@
    string/app_name">
        <activity android:label="@string/app_name"
        android:name=".ServiceActvt">
            <intent-filter>
                <action android:name="android.
    intent.action.MAIN" />
                <category android:name="android.
    intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".Exemplo_Service">
            <intent-filter>
                <action android:name="INICIAR_
    SERVICO"/>
                <category android:name="android.
    intent.category.DEFAULT"/>
            </intent-filter>
        </service>
    </application>
    <uses-sdk android:minSdkVersion="8" />
</manifest>
```

E novamente podemos ver o famoso <intent-filter> presente também na declaração do Serviço. Isso faz sentido, uma vez que os serviços são iniciados com uma intent, que pode conter um nome de ação específico.

A seguir vemos o código da nossa classe Service e o código que a inicia, em seguida a classe encerra e ainda sim a classe service continua ativa, o que conseguimos facilmente demonstrar logando no LogCat.

```
public class Exemplo_Service extends Service implements Runnable {  
  
    private int contador = 0;  
  
    /**  
     * @see android.app.Service#onBind(Intent)  
     */  
    @Override  
    public IBinder onBind(Intent intent) {  
        // TODO Insira seu código aqui  
        return null;  
    }  
  
    /**  
     * @see android.app.Service#onCreate()  
     */  
    @Override  
    public void onCreate() {  
        Log.i("[ENG CURSO ANDROID]", "Serviço Exemplo_ Service chamou método onCreate()");  
        new Thread(this).start();  
    }  
  
    /**  
     * @see android.app.Service#onStart(Intent, int)  
     */  
    @Override  
    public void onStart(Intent intent, int startId) {  
        Log.i("[ENG CURSO ANDROID]", "Serviço Exemplo_ Service chamou método onStart()");  
    }  
  
    @Override  
    public void run() {  
        while(contador < 10) {  
            try{  
                Thread.sleep(5000);  
            }catch(InterruptedException e){  
                e.printStackTrace();  
            }  
            Log.i("[ENG CURSO ANDROID]", "Serviço Exemplo_Service executando...");  
            contador++;  
        }  
    }  
}
```

```
public class ServiceActvt extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        Log.i("[ENG CURSO ANDROID]", "Activity ServiceActvt  
criada. Metodo onCreate() chamado.");  
        Intent intentChamarServico = new Intent("INICIAR_  
SERVICO");  
  
        //Inicia o serviço, passando a intent com a ação apropriada  
        startService(intentChamarServico);  
        Log.i("[ENG CURSO ANDROID]", "Activity ServiceActvt  
chamando metodo finish()");  
        finish();  
    }  
  
    @Override  
    public void onDestroy(){  
        super.onDestroy();  
        Log.i("[ENG CURSO ANDROID]", "Activity ServiceActvt  
destruida. Metodo onDestroy() chamado.");  
    }  
}
```

Podemos conferir o log do LogCat, que mostra o ciclo de vida da Activity e Service, tendo esse segundo continuado a executar mesmo depois que a atividade foi encerrada

- Definir interface
 - Serviço serviço
 - Atividade instância Service connection
starts bind service.

Binder é a implementação do IBinder.
A classe Service tem de ser um objeto de Binder.

Questões de memorização

- 1 - Qual é a relação entre Service e Intent-filter?

- 2 - Como seria possível iniciar um serviço apenas pela ação MEU_SERVICO?

Iniciando uma Intent ~~que sera~~ que sera passada como parâmetro

- 3 - Quais são as duas maneiras de iniciar um serviço?

startService e bindService

- 4 - Qual é a diferença entre um Service e um BroadcastReceiver?

- 5 - Quais são as maneiras de interromper a execução de um serviço?

Se iniciado com a atividade ao fechar a encerra o serviço ou stopSelf

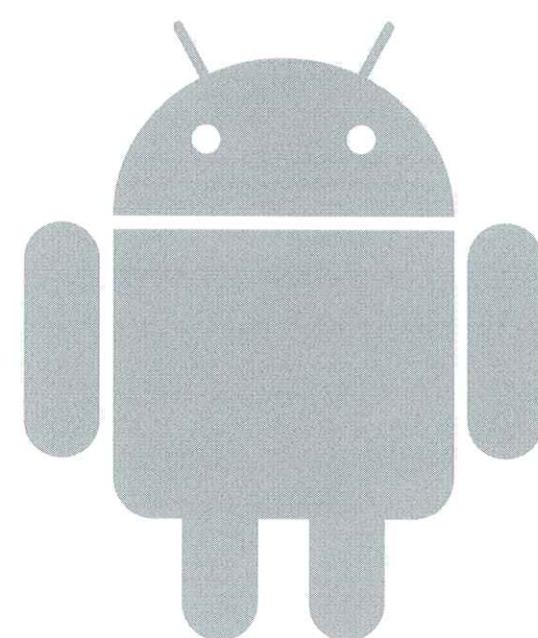
- 6 - Caso nenhuma categoria tenha sido definida para o serviço, é necessário declarar alguma categoria padrão?

Default

startService - serviço atrelado a Atividade, qdo esta é destruída ele tbm
bindService - ciclo de vida do serviço é independente do ciclo da atividade, p/ destrui-lo preciso do stopSelf() sempre 1 instance do serviço na memória

Módulo 4

SMS



Com o android é possível interceptar uma mensagem que é recebida, através de um BroadcastReceiver, de maneira muito simples.

Também é possível enviar mensagens de SMS através da classe SmsManager. O exemplo abaixo demonstra uma aplicação que é capaz de interceptar mensagens recebidas (Broadcast receiver) e enviar à um número qualquer.

Primeiramente criamos o código XML que contém a interface de envio de mensagens:

```
<?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="Enviar SMS pela aplicação"
        android:textStyle="bold" />
    <LinearLayout android:id="@+id/LinearLayout01"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent">
        <TextView android:id="@+id/TextView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Número:"/></TextView>
        <EditText android:layout_height="wrap_content"
            android:id="@+id/TxtNumero"
            android:layout_width="300dip"/></EditText>
    </LinearLayout>
    <LinearLayout android:id="@+id/LinearLayout02"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent">
        <TextView android:id="@+id/TextView02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Mensagem:"/></TextView>
        <EditText android:layout_height="wrap_content"
            android:id="@+id/Mensagem"
            android:layout_width="300dip"/></EditText>
    </LinearLayout>
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:id="@+id/BtnEnviar" android:text="Enviar"/></Button>
</LinearLayout>
```

Abaixo segue a definição da Atividade principal, chamada de SMSHandler:

```
public class SMSHandler extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        final EditText numeroDestino = (EditText)findViewById(R.  
id.TxtNumero);  
        final EditText msg = (EditText)findViewById(R.id.Mensagem);  
  
        Button btnEnviar = (Button)findViewById(R.id.BtnEnviar);  
        btnEnviar.setOnClickListener(new View.OnClickListener() {  
  
            @Override  
            public void onClick(View arg0) {  
                SMS sms = new SMS();  
                sms.enviarSms(SMSHandler.this, numeroDes-  
tino.getText().toString(), msg.getText().toString());  
                Toast.makeText(SMSHandler.this, "Mensagem envia-  
da.", Toast.LENGTH_LONG).show();  
            }  
        });  
    }  
}
```

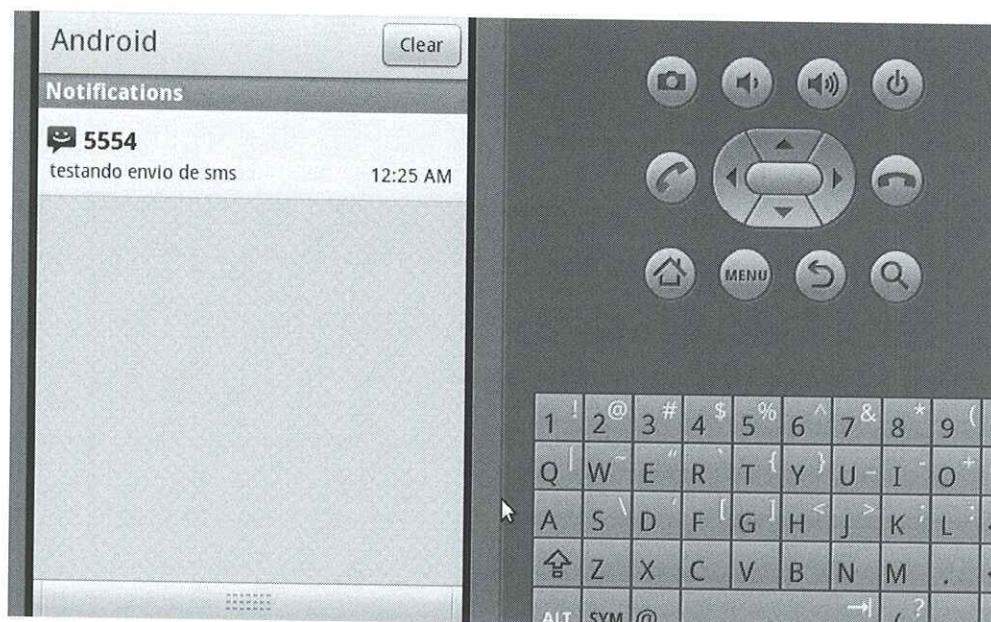
Essa atividade faz uso de um objeto criado no projeto, chamado SMS, que é descrito a seguir:

```
public class SMS {  
    public void enviarSms(Context context, String destino,  
String mensagem) {  
        try{  
            SmsManager smsManager = SmsManager.getDefault();  
            PendingIntent pendingIntent = PendingIntent.  
getBroadcast(context, 0, new Intent(), 0);  
            smsManager.sendTextMessage(destino, null, mensagem,  
pendingIntent, null);  
        }catch(Exception e){  
            Toast.makeText(context, "Erro ao enviar SMS:  
" + e.getMessage(), Toast.LENGTH_LONG).show();  
        }  
    }  
  
    public SmsMessage receberMensagem(Intent i){  
        SmsMessage[] mensagens = getMessagesFromIntent(i);  
        if(mensagens != null){  
            return mensagens[0];  
        }  
        return null;  
    }  
  
    public SmsMessage[] getMessagesFromIntent(Intent it){  
        Object[] messages = (Object[]) (Object[]) it.getSerializableExtra("pdus");  
        byte[][] pduObjs = new byte[messages.length][];  
        for(int i = 0 ; i < messages.length ; i++){  
            pduObjs[i] = (byte[]) (byte[]) messages[i];  
        }  
        byte[][] pdus = new byte[pduObjs.length][];  
        if(pdus == null){  
            return null;  
        }  
        int pduCount = pdus.length;  
        if(pduCount == 0)  
            return null;  
        SmsMessage[] msgs = new SmsMessage[pduCount];  
        for(int i = 0 ; i < pduCount ; i ++){  
            pdus[i] = pduObjs[i];  
            msgs[i] = SmsMessage.createFromPdu(pdus[i]);  
        }  
        return msgs;  
    }  
}
```

Finalmente temos a definição do broadcast receiver, responsável por interceptar as mensagens recebidas:

```
public class SmsReceiver extends BroadcastReceiver {  
    /**  
     * @see android.content.BroadcastReceiver#onReceive(Context  
, Intent)  
     */  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        SMS sms = new SMS();  
        SmsMessage[] msg = sms.  
getMessagesFromIntent(intent);  
        for(SmsMessage mensagem : msg){  
            String celular = mensagem.getDisplayOriginatingAddress();  
            String corpoMensagem = mensagem.getDisplayMessage-  
Body();  
  
            Toast.makeText(context, "Mensagem recebida de : " + celular  
+ ".\n" + corpoMensagem, Toast.LENGTH_LONG).show();  
        }  
    }  
}
```

É importante ressaltar a necessidade de declarar o uso de permissões como `android.permission.SEND_SMS` e `android.permission.RECEIVE_SMS` no `AndroidManifest.xml`, caso contrário não será possível enviar e receber mensagens de SMS.



Mensagem recebida

Também é necessário configurar o intent-filter de nosso receiver.

O código do arquivo Manifest é demonstrado na íntegra a seguir:

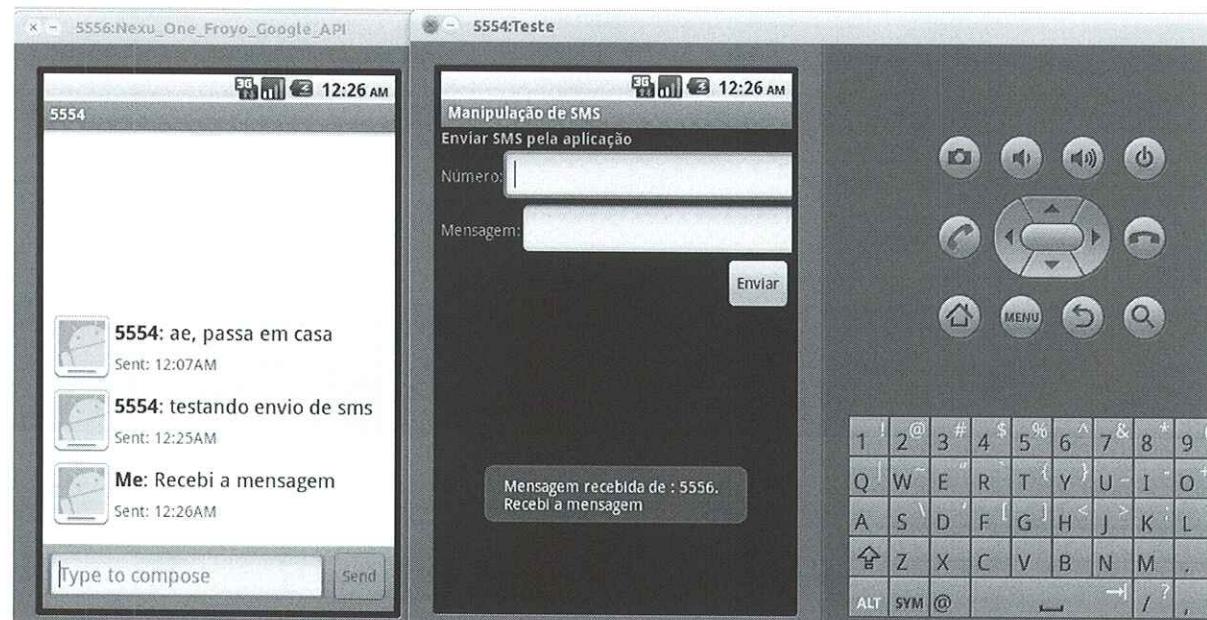
```
<?xml version="1.0" encoding="UTF-8"?>
<manifest android:versionCode="1" android:versionName="1.0"
    package="com.eng.sms.manipulacao" xmlns:android="http://
schemas.android.com/apk/res/android">
    <application android:icon="@drawable/icon" android:label="@
string/app_name">
        <activity android:label="@string/app_name"
        android:name=".SMSHandler">
            <intent-filter>
                <action android:name="android.in-
tent.action.MAIN" />
                <category android:name="android.in-
tent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    <receiver android:name="com.eng.sms.manipulacao.receivers.SmsRe-
ceiver">
        <intent-filter>
            <action android:name="android.pro-
vider.Telephony.SMS_RECEIVED"></action>
            <category android:name="android.in-
tent.category.LAUNCHER"></category>
        </intent-filter>
    </receiver>
    </application>
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.SEND_
SMS"></uses-permission>
    <uses-permission android:name="android.permission.RECEIVE_
SMS"></uses-permission>
</manifest>
```



Mensagem sendo enviada de uma instância de emulador para outra



Mensagem sendo recebida e interceptada pelo broadcast receiver



Mensagem foi interceptada pela aplicação e exibida, utilizando um Toast (poderia ser feito qualquer coisa com a mensagem, como por exemplo guardar em um log de mensagens recebidas, guardar em um arquivo de textos, banco de dados etc)

Questões de memorização

- 1 - É possível instanciar dois emuladores do Android?
- 2 - Como é possível interceptar o recebimento de mensagens SMS em sua própria aplicação?
- 3 - Ao configurar a aplicação para interceptar o recebimento de SMS, qual permissão é necessário declarar no AndroidManifest.xml?
- 4 - Como é possível enviar SMS pela sua própria aplicação?
- 5 - Ao enviar mensagens por sua aplicação, qual permissão é necessário declarar no AndroidManifest.xml?
- 6 - Como é possível enviar (apenas para fins de testes) mensagens sms de um celular para outro?