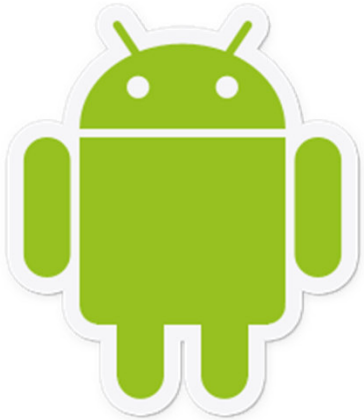
Android Avançado

Marcelo Alves
malves.info@gmail.com
@malvesinfo
www.portalandroid.org



Ementa

* **Inteface avançada**
  - Fragments
  - Async Tasks
  - Notification, barra de notificação do aparelho
  - Widgets

Ementa

* **Mapas e GPS**
  - Acessando o Google Maps
  - Adicionando pontos a um local do mapa
  - Exibindo a localização com GPS
* **Sensores**
  - Sensor e SensorManager
  - Orientação, acelerômetro, iluminação, campo magnético, proximidade, temperatura

Ementa

* **BroadcastReceiver**
  - Ciclo de Vida
  - Broadcasts importantes do sistema operacional

* **Service**
  - Utilizando serviços para execuções em segundo plano
  - A linguagem AIDL

* **AlarmManager**
  - Agendando a execução

Ementa

* **Rede**
  - Manipulando Sockets
  - Acessando informações na internet via HTTP
  - Consumindo Web Services
  - Redes Bluetooth
  - Mensagens SMS
* **Multimidia**
  - Manipulando audio e vídeo
  - Acessando a câmera
  - Além do Android: um overview do FFMpeg

Interface Avançada

Fragments

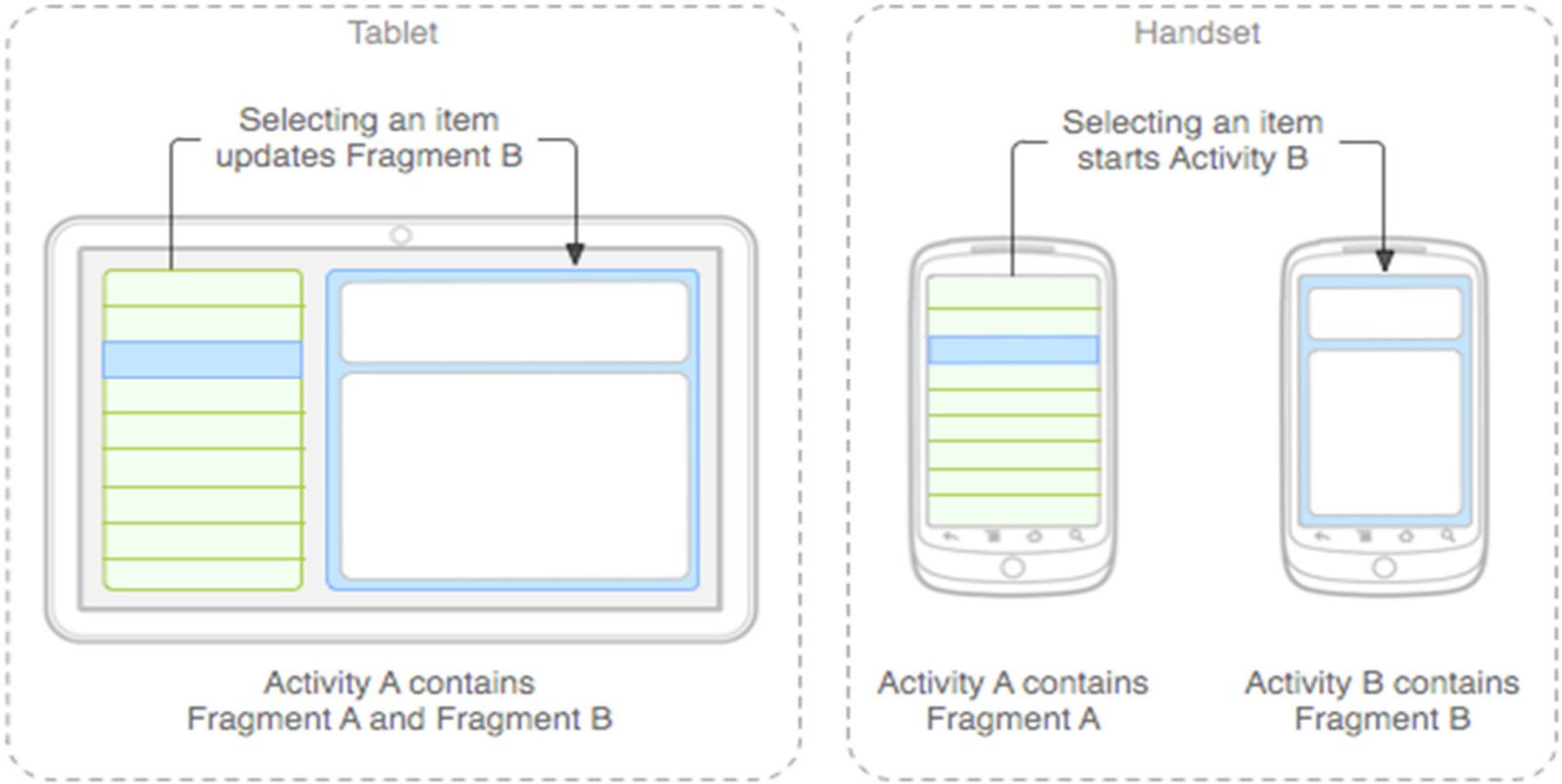
* Características
  * Fragments representam uma seção modular em uma Activity.
    * Pode ser anexado dentro de uma ViewGroup do layout
  * Possui “ciclo de vida” próprio.
    * Porém atrelado à da Activity qua a contiver
      * Se a Acitivity entra em pausa, o seus fragments também entram, e assim por diante
    * Exceto em onCreateView(), sempre chame o super quando sobrescrever métodos
  * Podem ser manipulados de forma independente.

Fragments

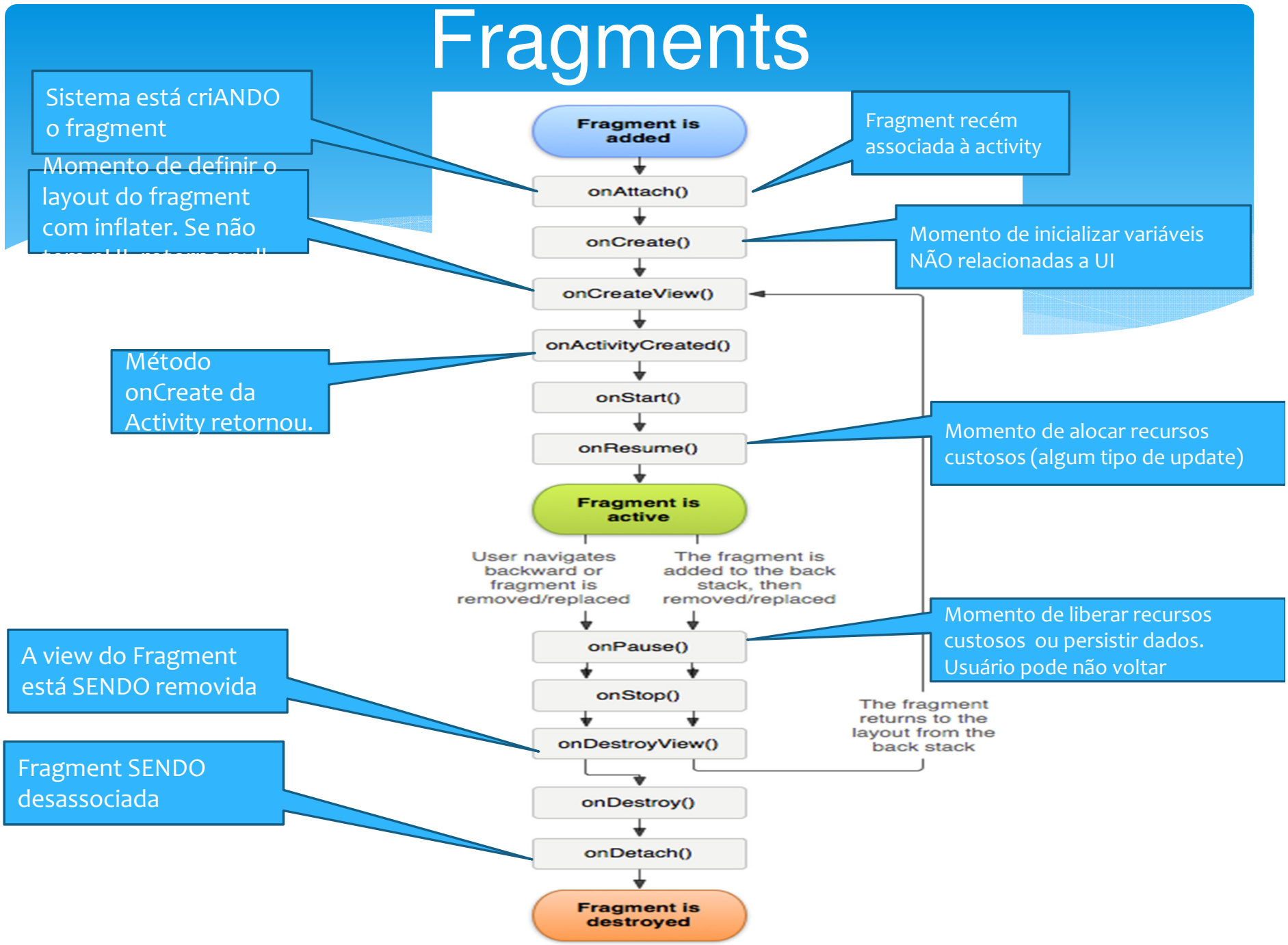
* Aplicações
  * Criados a partir da versão 3.0, permitem usar toda a amplitude das telas do tablets sem que haja um gerenciamento complexo das views.
  * Fragments possuem seu próprio layout e ciclo de vida, por isso são ferramentas poderosas em termos de reusabilidade.

Fragments

* Aplicações



Fragments


Sistema está criANDO o fragment
Momento de definir o layout do fragment com inflater. Se não
Método onCreate da Activity retornou.
A view do Fragment está SENDO removida
Fragment SENDO desassociada
Fragment is added
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
Fragment is active
User navigates backward or fragment is removed/replaced
The fragment is added to the back stack, then removed/replaced
onPause()
onStop()
onDestroyView()
onDestroy()
onDetach()
Fragment is destroyed
The fragment returns to the layout from the back stack
Fragment recém associada à activity
Momento de inicializar variáveis NÃO relacionadas a UI
Momento de alocar recursos custosos (algum tipo de update)
Momento de liberar recursos custosos ou persistir dados. Usuário pode não voltar

Fragments

```
public class FirstFragment extends Fragment implements
OnClickListener {
@Override
   public View onCreateView(LayoutInflater inflater, ViewGroup
   container,
   Bundle savedInstanceState) {
      View view = inflater.inflate(R.layout.first_fragment,
      container, false);
      Button nextButton = (Button)
      view.findViewById(R.id.button_first);
      nextButton.setOnClickListener(this);
      return view;
}
// ...
}
```

Fragments

- Associando à uma Activity com <fragment>
  - Para fragments sempre presentes
    - Não é possível remover em tempo de execução
  - Em android:name deve-se especificar o nome completo da classe correpondente

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="match_parent"
android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
    android:id="@+id/list"
    android:layout_weight="1"
    android:layout_width="0dp"
    android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
    android:id="@+id/viewer"
    android:layout_weight="2"
    android:layout_width="0dp"
    android:layout_height="match_parent" />
</LinearLayout>
```

Fragments

* Associando à uma Activity dinamicamente
  * Para inserir/retirar em tempo de execução
  * Requer um FragmentManager
    * Chamado via Activity.getFragmentManager()
  * Qualquer operação requer criar uma transação
    * FragmentManager.beginTransaction()/commit()
  * Operações de add/attach/dettach/remove

```
FragmentManager fragmentManager = getFragmentManager()
// Or: FragmentManager fragmentManager = getSupportFragmentManager()
FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

Fragments

* Associando à uma Activity dinamicamente
  * Os comandos podem ser encadeados
    * O commit() é sempre o último
    * commit() invocado após onSavedInstanceState() da Activity gera exceção

```
FragmentManager fragmentManager = getFragmentManager();
fragmentManager.beginTransaction()
.remove(fragment1)
.add(R.id.fragment_container, fragment2)
.show(fragment3)
.hide(fragment4)
.commit();
```

Fragments

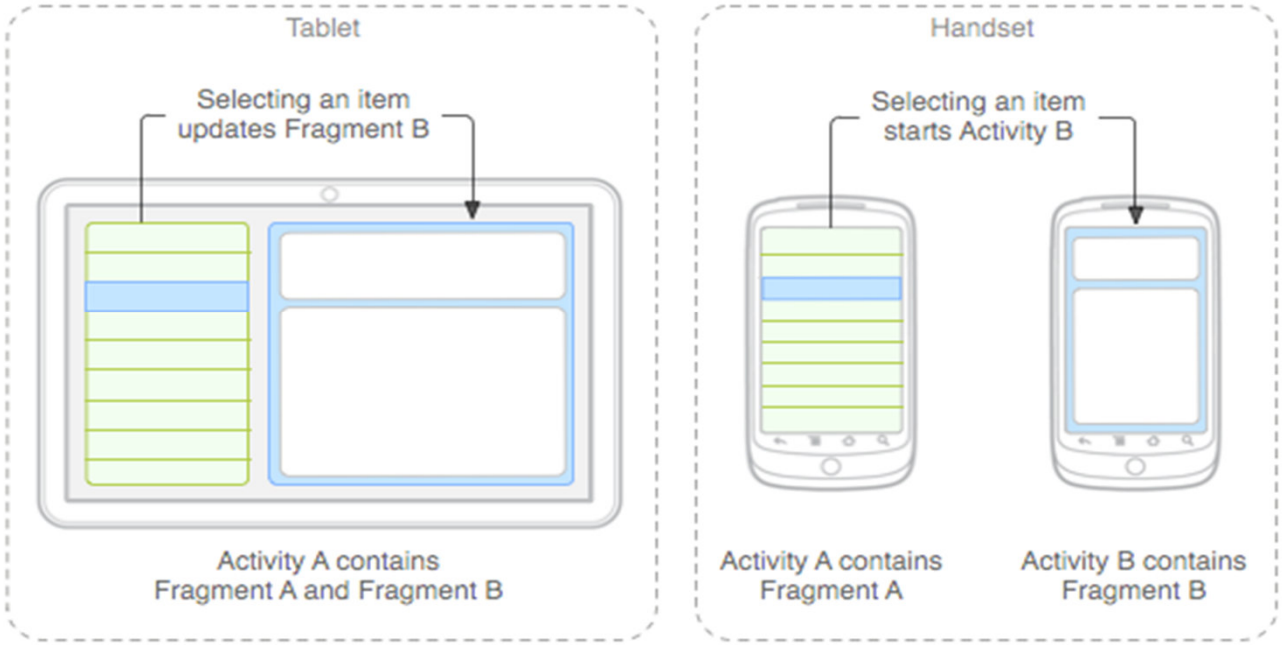
* Backstack de Fragments
  * Para adicionar: FragmentTransaction.addToBackStack(String)
    * As transações feitas até o último commit ficam na mesma posição da pilha
  * “Back” desfaz a última transação “commitada”
  * Chegando na activity, destrói a activity
  * Se chamar addToBackStack() após remover ou substituir um fragment:
    * O sistema chama onPause(), onStop(), e onDestroyView() no fragment que estiver na back stack
    * Se o usuário pressionar back, o sistema chama onCreateView(), onActivityCreated(), onStart(), e onResume() do fragment

Fragments

* FragmetnTransaction
  * add()
    * Adiciona o fragment na activity
  * remove()
    * Remove da activity e o destrói , a não ser que a transação esteja na backstak
  * replace()
    * Retira um fragment da UI e o substitui por outro
  * hide()
    * Deixa o fragment invisível, sem destruir o seu layout
  * show()
    * Deixa o fragment visível
  * detach() (API 13)
    * Desassocia fragment da UI, destrói o layout, mas mantém a instância do fragment.
  * attach() (API 13)
    * Reassocia o fragment anteriormente desassociado, recriando o layout

Fragments

* Exercício
  * Procure criar uma aplicação sensível ao tamanho de tela do aparelho reutilizando os fragments de menu e conteúdo
  * Em caso de dúvida, veja API demos



Fragments

* Mudanças de configuração em tempo de execução
  * Recriação da Activity
    * Se foi recriada, o Bundle no onCreate() vem diferente de null
  * Reinstanciação dos fragments

```
public void onCreate(Bundle savedInstanceState) {
    // ...
    if (savedInstanceState == null) {
        FragmentManager fragmentManager = getFragmentManager()
        // Or: FragmentManager fragmentManager =
        getSupportFragmentManager()
        FragmentTransaction fragmentTransaction =
        fragmentManager.beginTransaction();
        ExampleFragment fragment = new ExampleFragment();
        fragmentTransaction.add(R.id.fragment_container, fragment);
        fragmentTransaction.commit();
    }
}
```

Fragments

* Mudanças de configuração em tempo de execução
  * Caso queira manter a instância, setar `Fragment.setRetainInstance(boolean)` passando true
    * `Activity.onRetainNonConfigurationInstance()` foi depreciado
    * Útil quando o Fragment é host de alguma Thread
  * Assim, o `onDestroy()` e `onCreate()` não seriam chamados
    * Os demais métodos, sim, com Bundle null

Fragments

* Fragments sem layouts
  * Manter instância de threads ou estado
  * Não precisa sobrescrever onCreateView()
  * Ao adicionar, prover um id
  * Para encontrar o fragment em questão:
    * findFragmentById(int id) OU
    * findFragmentByTag(String tag)

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction=fragmentManager.beginTransaction()
BackgroundFragment fragment = new BackgroundFragment();
fragmentTransaction.add(fragment, "thread_manager");
fragmentTransaction.commit();
```

Fragments

* Exercício:
  * Crie um aplicativo que escreva 1, 2, 3 a cada 1 segundo indefinidamente em arquivo no sdcard
  * A activity deve ter um botão para iniciar e interromper o processo
  * Aperte o botão de iniciar e veja que horas são
  * Mude o idioma do aparelho
  * De volta a activity, espere mais alguns minutos e interrompa o processo
    * O arquivo deve ser aberto normalmente contendo na últma linha a quantidade de segundos correpondente ao tempo entre iniciar e interromper o processo

Fragments

* v4 Library APIs
  * Permite uso de Fragments em versões abaixo da 3.0
  * Copiar .jar em <sdk>/extras/android/support/v4 para a sua pasta de projeto “libs”
  * Diferenças
    * Para que uma activity use fragments, é necessário que ela herde de FragmentActivity invés de somente Activity.
    * O gerenciamento é feito pelos métodos FragmentActivity.getSupportFragmentManager() e FragmentActivity.getSupportLoaderManager().
    * Não dá suporte a classe ActionBar,
      * Porém é possível selecionar quais itens serão adicionados a ActionBar quando ela estiver disponível para versões a partir da 3.0 em diante.

Fragments

* Support-v4-googlemaps
  * Torna possível o uso de MapView em Fragments
  * Todas as classes FragmentActivity herdam de MapActivity.

Fragments – boas práticas

* Não inserir um Fragment no layout de outro Fragment
  * Comportamento inesperado
  * Exceto para Android 4.2
* Usar apenas quando se deseja reaproveitar o layout em diversos formatos (tablets, smartphones)
  * Reuso de comportamento
  * Gerenciamento de pilha
  * Transições específicas de tela
  * Dialogs (AlertDialogs não morrem sozinhos quando são de progress)
  * Se Desempenho não for um problema
* Usar sempre o construtor padrão do Fragment
  * Em caso de recriação de tela o construtor *default* que é chamado
  * Caso necessário, usar setArguments()

Fragments – boas práticas

* Evitar que seu fragment seja muito dependente do que há na activity
  * Se isso ocorre, repense se você precisa mesmo de um Fragment
  * Ao invés de acessar métodos da activity, crie callbacks
* Usar a Activity como intermediária entre os fragments
  * Evitar comunicação direta entre eles

DialogFragment

* Possui um objecto Dialog, mas não dê show/hide diretamente nele
* Sobrescever métodos onCreateDialog() retornando uma instância de Dialog
  * É possível sobrescrever apenas onCreateView(), se não quiser usar como Dialog
* **`int show(FragmentTransaction transaction, String tag)`**
  * Exibe dialog, incluindo-o na transaction existente e dá commit.
  * Essa transação pode estar na backstack =)
* **`void show(FragmentManager manager, String tag)`**
  * Exibe dialog criando uma trnasação a partir do manager, adiciona e dá commit.

Exercício

* Crie uma activity que tenha um botão, cuja ação exibe um fragment
* Neste fragment deve haver outro botão, cuja ação remove este fragment e abre um dialogFragment
* Ao dar ‘back’ nesse dialogFragment, o fragment antigo deve reaparecer

Async Tasks

* Ideal para executar tarefas longas enquanto mostra um sinal de progresso para o usuário
* Classes estendendo AsyncTask

```
public abstract class Task extends AsyncTask<Void, Void, Void> {

        protected AnimeTogetherController controller;

        private CustomProgressDialog dialog;

        @Override
        abstract protected Void doInBackground(Void... params);

        public Task() {
                this.controller = AnimeTogetherController.getInstance();
        }

        @Override
        protected void onPreExecute() {
                // Mostrar progress dialog
        }
```

Async Tasks

```
@Override
        protected void onPostExecute(Void result) {
                //fechar dialog
        }

        @Override
        protected void onCancelled(Void result) {
                //fechar dialog
        }

        }
}
```

Async Tasks

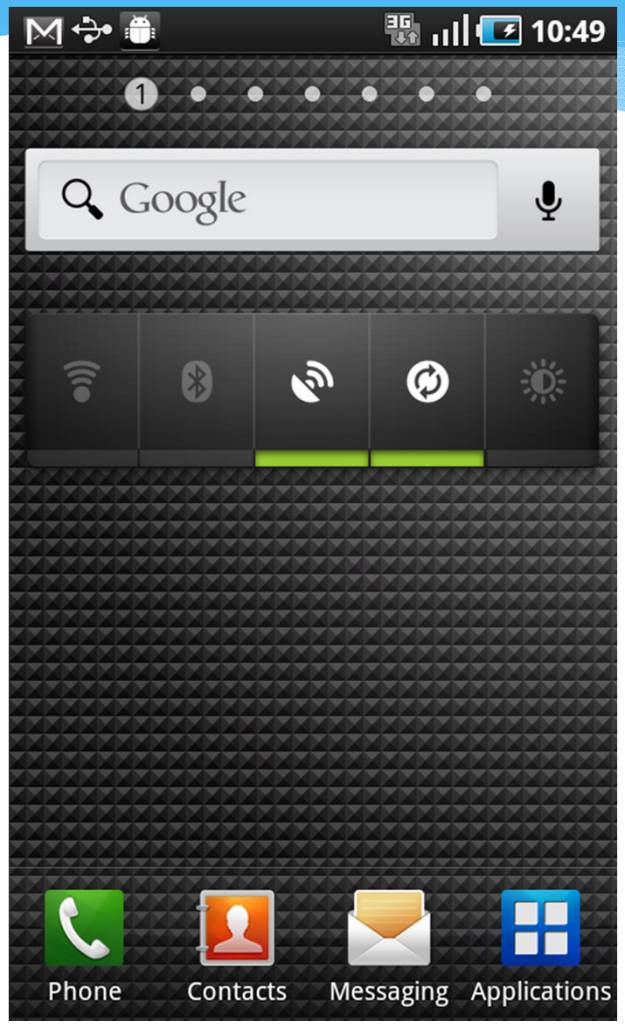
* Exercícios
  * Refaça o exemplo de contador, agora incluindo uma barra de progresso.
  * Ou seja, troque thread por uma async task
    * Atenção, pois agora o fragment possui uma UI

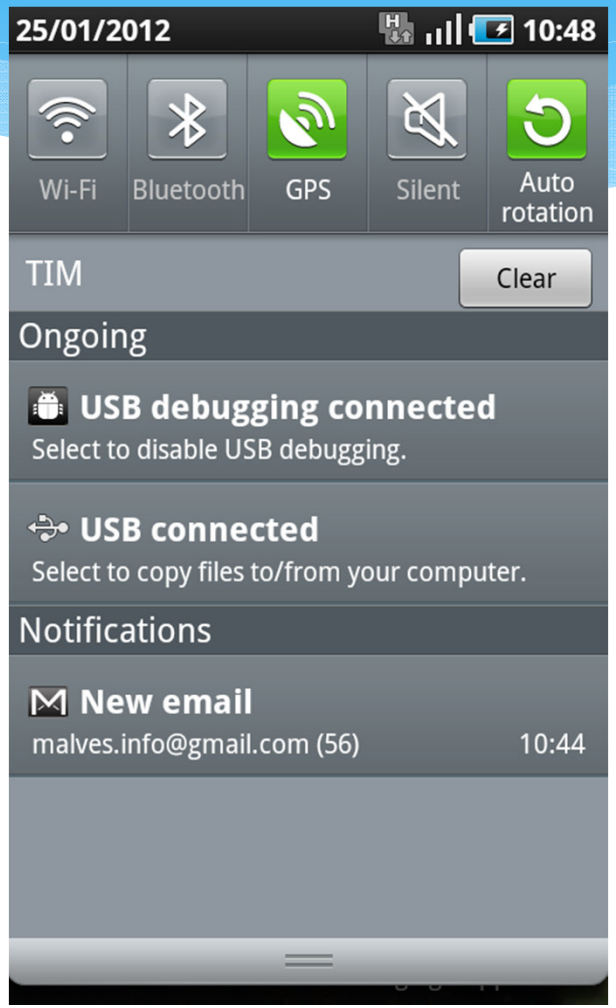
Notification

* Notificantion são notificações enviadas para o usuário para que o mesmo seja notificado sobre uma determinada ação da aplicação.

* Através do notification é possível abrir uma Activity da aplicação, iniciar um serviço ou BroadcastReceiver

Exemplo de Notifications





Criando uma Notificação

* Primeiro faz a solititação do serviço de notification recebendo o objeto NotificationManager

```
NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
```

* Criamos um Notification que irá representar a notificação com: Mensagens, Sons, Ação ao selecionar e etc.

```
Notification notification = new Notification( R.drawable.ic_launcher,
"Chegou Notificação", System.currentTimeMillis());
```

Criando uma Notificação

* Na criação da notificação é exigido um PendingIntent, API responsável por definir qual ação será executada quando do usuário selecionar a notificação.

```
Context context = getApplicationContext();
CharSequence contentTitle = “Minha Notificação";
CharSequence contentText = “Olá Notification!";
Intent notificationIntent = new Intent(this, MinhaActivity.class);
PendingIntent contentIntent = PendingIntent.getActivity(this, 0, notificationIntent, 0);

notification.setLatestEventInfo(context, contentTitle, contentText, contentIntent);
```

Criando uma Notificação

* Após criar nossa notificação basta chamar o método notify do NotificationManager e passar a Notification criada

```
notificationManager.notify(NOT_ID, notification);
```

* Adicionando Sons

```
//Som Default
notification.defaults |= Notification.DEFAULT_SOUND;

//Som do SDCARD
notification.sound = Uri.parse("file:///sdcard/notification/ringer.mp3");
```

Criando uma Notificação

* Adicionado Vibrate

```
//Vibrate Default
notification.defaults |= Notification.DEFAULT_VIBRATE;

//Vibrate Personalisado
long[] vibrate = {0,100,200,300};
notification.vibrate = vibrate;
```

Widget

* Widgets são pequenos aplicativos que podem fazer parte de outras aplicações, que irão exibir atualizações periodicas. Como: Notícias, Temperatura, Atualizações de uma Rede Social e etc.

Exemplo de Widgets

Widget - RemoteViews

* Os layouts do widgets são baseados na classe RemoteViews, que permite acesso aos componentes do layout que esta “inflado” no RemoteViews.
* RemoteViews suporta os seguintes Layouts:
  * FrameLayout
  * LinearLayout
  * RelativeLayout

Widget - RemoveViews

* RemoveViews suporta os seguintes componentes de View:
  * AnalogClock
  * Button
  * Chronometer
  * ImageButton
  * ImageView
  * ProgressBar
  * TextView
  * ViewFlipper
  * ListView

Widget - RemoveViews

* GridView
* StackView
* AdapterViewFlipper

Criando Widgets

* **Para criar um widget é necessário ter:**
  * **AppWidgetProviderInfo**: Descreve a metadata de uma app widget. Tais como: Layout do Widget, Tempo de Update e a AppWidgetProvider.

  * **AppWidgetProvider:** Contém os métodos básico para a implementação do widget via código. Todos os métodos são chamados baseando em chamadas via eventos de Broadcast. Onde serão enviados Broadcast quando o widget for: Atualizado, habilitado, desabilitado e deletado.

Criando Widgets

* O primeiro passo para a criação de um Widget é declarar o AppWidgetProvider no AndroidManifes.xml

```
<receiver android:name=“MeuWidgetProvider" >
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>
    <meta-data android:name="android.appwidget.provider"
               android:resource="@xml/appwidget_info" />
</receiver>
```

Criando Widgets

* Próximo passo é configurar o **AppWidgetProviderInfo:**

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="300dp"
    android:minHeight="72dp"
    android:updatePeriodMillis="5000"
    android:initialLayout="@layout/appwidget_layout"
    android:configure="com.example.android.MeuWidgetConfigure"
    android:resizeMode="horizontal|vertical"
    >
</appwidget-provider>
```

Criando Widgets – AppWidgetProvider

* Próximo passo é criar a Classe que “extends” de AppWidgetProvider. Onde deverá implementar os seguintes métodos:
  * **onUpdate() -** Método é chamado no intervalo definido no atributo updatePeriodMillis em AppWidgetProviderinfo (XML). Esse método também é chamado quando o widget for adicionado.
  * **onDeleted(Context, int[]) -** Chamado quando o widget é deletado da home

Criando Widgets – AppWidgetProvider

* **onEnabled(Context) -** Chamado quando o widget é criado pela primeira vez.

* **onDisabled(Context) -** É chamado quando a ultima instancia do Widget for deletada.

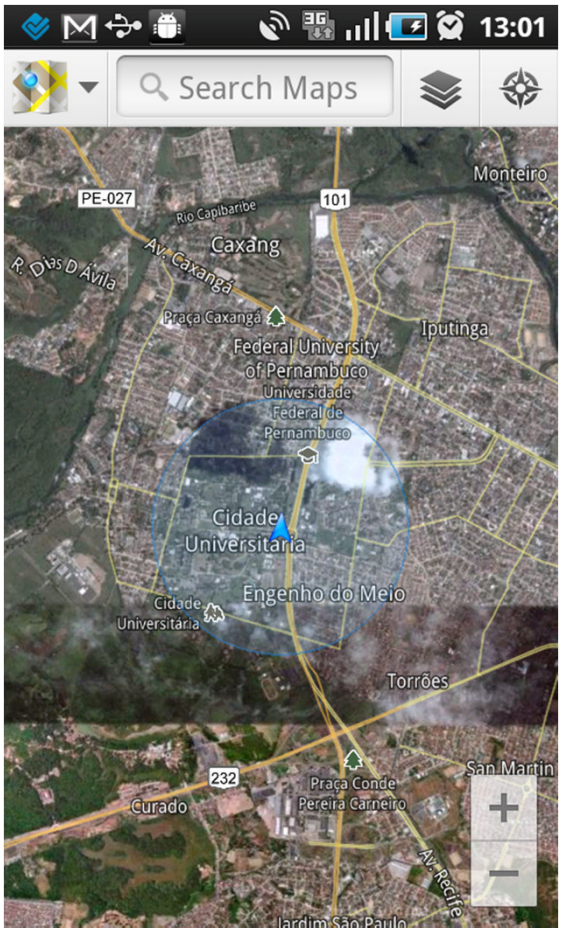
* **onReceive(Context, Intent) -** É chamado sempre que receber algum Broadcast antes da execução dos métodos anteriores.

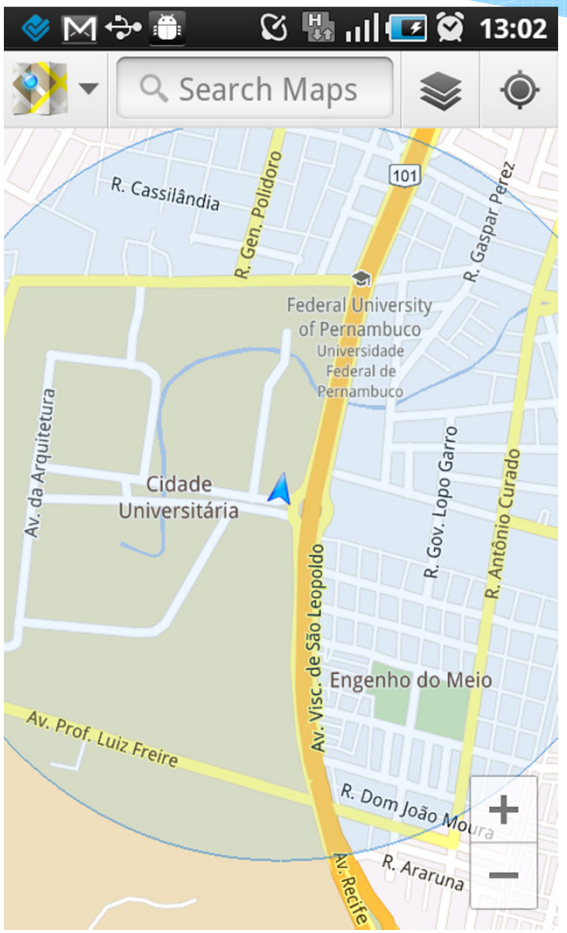
Mapas, GPS e Sensores

Mapas e GPS

* Android permite o acesso e utilização de Mapas e GPS em sua API
* Isso permite criar aplicações que exibam mapas e pontos no mapa através de Latitude e Longitude obtidas através do GPS.

Mapas e GPS


13:01
Search Maps
PE-027
101
Rio Capibaribe
Caxangá
Av. Caxangá
R. Dias D Ávila
Praça Caxangá
Federal University of Pernambuco
Universidade Federal de Pernambuco
Monteiro
Iputinga
Cidade Universitária
Engenho do Meio
Torrões
232
Curado
Praça Conde Pereira Carneiro
San Martin
Av. Recife
Jardim São Paulo


13:02
Search Maps
R. Cassilândia
R. Gen. Polidoro
101
R. Gaspar Perez
Federal University of Pernambuco
Universidade Federal de Pernambuco
Av. da Arquitetura
Cidade Universitária
R. Gov. Lopo Garro
R. Antônio Curado
Av. Visc. de São Leopoldo
Engenho do Meio
Av. Prof. Luiz Freire
R. Dom João Moura
R. Araruna
Av. Recife


13:03
Search Maps
R. Galvão Raposo
Praça Eça de Queiroz
R. Visc. do Uruguai
R. Mns. Silva
R. Ernani Braga
Madalena
R. José Osório
Av. Caxangá, 157-179 - Madalena
Up To Date Eventos & more
R. Guapé
R. Carlos Gomes
R. Benfica
Prado
Av. Eng. Abdias de Carvalho
R. Pinto Júnior
R. Teófilo Twortz

Configurando Google Maps

* Baixar a API do Google no gerenciador de Downloads do AVD
* Obter a chave para o acesso ao Google Maps!. Intruções para esse procedimento em:
  * http://code.google.com/android/add-ons/google-apis/mapkey.html
  * Após gerar a Chave, basta utilizá-la na API do Google Maps
* Incuir entre as tags <application></application>:

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

Utilizando Mapas

* Existem duas formas de exibir um mapa em uma aplicações Android. Primeira é vida código Java utilizando a Classe **MapActivity** e a outra forma é declarando o **MapView** dentro de um XML de layout.
* Só se pode usar apenas uma MapView por activity
  * Até a chegada da nova API!

Utilizando Mapas

* Via MapActivity

```
public class ExemploMapa extends MapActivity {
        private MapView mapView;
        @Override
        protected void onCreate(Bundle icicle) {
                super.onCreate(icicle);
                mapView = new MapView(this,
"0D02uiEmQKCrWEL2Ujxv-4eZFUp9MoYZ5pXsQOA");
                setContentView(mapView);
        }
}
```

Utilizando Mapas

* Via XML

```
<com.google.android.maps.MapView
        android:id="@+id/mapa"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:enabled="true"
        android:clickable="true"
         android:apiKey="0D02uiEmQKCrWEL2Ujxv-
4eZFUp9MoYZ5pXsQOA"
/>
```

Utilizando Mapas

* Via XML

```
public class ExemploMapa extends MapActivity {
        private MapView mapView;
        @Override
        protected void onCreate(Bundle icicle) {
                super.onCreate(icicle);
                setContentView(R.layout.meu_mapa_layout);
                MapView mapView = (MapView) findViewById(R.id.mapa);
        }
}
```

Utilizando Mapas

* Alguns métodos de manipulação do mapa
  * **setSatellite(boolean)** Altera a exibição do mapa para o modo satelite
  * **setScreenView(boolean)** Altera o mapa para exibir em modo de Rua
  * **setTraffic(boolean)** Altera o mapa para exibir ruas e condições de tráfego
  * setBuiltInZoomControls(boolean) Configura o Mapa para que exiba ou não o controle de Zoom pelo usuário.

Utilizando Mapas

* **animateTo(GeoPoint)** Faz animação e centralização do mapa através do Geo Point informado
* **setZoom(int) , zoomIn() e zomOut()** Modifica o zoom do mapa

Utilizando Mapas - GeoPoint

* A classe GeoPoint é utilizada para representar uma localização usando as coordenadas de latitude e longitude. Que devem ser informadas em **microdegress**

```
MapView mapView = new MapView(this,
            "0D02uiEmQKCrWEL2Ujxv-4eZFUp9MoYZ5pXsQOA");

int latitudeE6 = (int) (-25.442580 * 1E6);
int longitudeE6 = (int) (-49.279840 * 1E6);

GeoPoint geoPoint = new GeoPoint(latitudeE6, longitudeE6);

mapView.getController().setCenter(point)
```

GPS Location

* Android disponibiliza de uma API que possibilita uma aplicação receber a localização atual do usuário (Latitude e Longitude).
* A API de Location esta disponível no pacote ***android.location***
* Para utilizar a API de Location é necessário recuperar o serviço ***LOCATION_SERVICE*** onde irá retornar ***com.location.LocationManager***

```
LocationManager locManager = (LocationManager)
context.getSystemService(Context.LOCATION_SERVICE);
```

GPS Location

* Após recuperar o ***LocationManager***, basta chamar o método ***requestLocationUpdates*** :
  * ***provider –*** Quem irá prove a localização do usuário:
    * ***GPS_PROVIDER –*** *Irá utilizar o GPS para prover dados de localização através de satélite. Tem uma melhor precisão mas o custo de bateria é alto.*
    * ***NETWORK_PROVIDER –*** *Utiliza a torre de celular e Wifi para prover a localização do usuário. Tem Baixa precisão mas o custo de bateria é baixo.*
  * ***minTime –*** *Tempo minímo de intervalo entre cada busca por localização. Tempo em milisegundos.*

GPS Location

* ***minDistance*** **–** Distancia mínima entre as atualizações em metros.
* ***Listener*** **–** Listener onde a API irá notificar cada update. Método de notificação é o ***onLocationChanged(Location)***.

GPS Location

- ***minDistance*** **–** Distancia mínima entre as atualizações em metros.
- ***Listener*** **–** Listener onde a API irá notificar cada update. Método de notificação é o ***onLocationChanged(Location)***.

* Exemplo: MapSchedule

Sensores

* Para medição de movimento, orientação, condições ambientais
* Categorias
  * Movimento
    * Aceleração ao longo dos 3 eixos
  * Ambiente
  * Posição
    * Temperatura, pressão, iluminação, humidade
    * Magnômetros, sensores de orientação

Sensores

* Exemplo: Shake
  * Ver projeto ShakeEvent

Broadcast Receivers, Services, AIDL e AlarmManager

BroadcastReceiver

* Utilziado para responder eventos enviados por uma intent, vinda de uma determinada aplicação ou pelo SO

* BroadcastReceiver sempre será executado em segundo plano durante pouco tempo sem a percepção do usuário

Configurando BroadcastReceiver

* É necessário criar uma classe que extends de BroadcastReceiver. Onde será implementado o método:

  onReceive(Context context, Intent intent)

* Próximo passo é registrar o Broadcast desejado na aplicação para que a mesma seja notificada quando for enviado um evento do Broadcast registrado. Broadcast pode ser registrado de duas formas: Através da API dentro do código ou através do arquivo AndroidManifest.xml

Configurando BroadcastReceiver

* Registrando Broadcast através da API :

```
IntentFilter filter = new IntentFilter("ABRIR_RECEIVER");
registerReceiver(new ExemploReceiver(), filter);
```

* *Registrando Broadcast através do AndroidManifest:*

```
<receiver android:name=".receiver.ExemploReceiver">
    <intent-filter>
            <action android:name="ABRIR_RECEIVE" />
            <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</receiver>
```

Configurando BroadcastReceiver

* Exemplo de como ficaria a Classe ExemploReceiver:

```
public class ExemploReceiver extends BroadcastReceiver {

      @Override
      public void onReceive(Context context, Intent intent) {

       }
}
```

Services

* Services são utilizados para executar serviços em segundo plano. Muitas das vezes são processos executados por tempo inderteminado que tenha um consumo alto de CPU e Memória.

* Serviços podem ser utilzados para baixar algum contéudo da interner, tocar um mp3 entre outros processamentos demorados sem que o usuário perceba.

Services

* Geralmente um service é inciado por um BroadcastReceiver, no qual precisa ser executado rapidamente.

* Service faz parte do ciclo de vida controlado pelo Android e o seu processo não é finalizado enquanto estiver sendo executado.

* Caso o Service seja encerrado pelo Android por falta de Memoria o Android irá tentar levantar o Service novamente.

Service

* **Por que utilizar Service?**
  * Faz parte do ciclo de vida do Sistema Operacional, além de ser gerenciado por ele.
  * Caso exista alguma Thread executando um processo na sua Activity e a mesma é encerrada (Back), o Android poderá matar a Activity e automaticamente a Thread com o processo será encerrado.
  * BroadcastReceiver que precisem executar processos demorados.

Service

* **Dica:** É aconselhável que cada Service iniciado cria uma Thread internamente. Isso porque os Services são executados na Thread principal. Caso alguma activity tenha iniciado o Service, ele ficará preso na Thread principal.

Service

* Configurando um Service:
  * Primeiro é necessário criar uma subclasse de android.app.Service
  * Após é necessário fazer a configuração no arquivo AndroidManifest.xml

```
<service android:name=".pacote.servicos.MeuServico" >
        <intent-filter >
            <action android:name=“MEU_SERVICO" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
 </service>
```

Service

* **Duas formas de iniciar um serviço**:
  * **startService(intent)** Esse método inicia um serviço que ficará executando por um tempo inderteminado até que o método stopService(intent) seja chamado.
  * **bindService(intent,com,flags)** Esse método pode iniciar um serviço caso ele ainda não tenha sido iniciado ou simplismente cirar uma conexão com o serviço. Esse método permite recuperar uma referência de uma classe ou interface para manipulação do serviço. **Mas o serviço é finalizado quando a Activity for finalizada.**

Service – Linguagem AIDL

* AIDL é a abreviação de **A**ndroid **I**nterface Definition Language. É utilzado pelo Android para possíbilitar a comunicação remota de dois processos. Ou seja, possibilita criar uma comunicação com um serviço que esteja em outro processo do sistema operacional.
* Primeiro passo é criar um arquivo com extensão “.aidl” que automaticamente o Eclipse irá gerar uma interface Java que implementa a interface criada no arquivo AIDL.

```
package com.exemplo.service.aidl;
interface AcessoRemoto {
        int count();
}
```

Service – Linguagem AIDL

* Próximo passo é usar a nova interface criada automaticamente pelo Eclipse na Classe de Serviço e retornar essa instancia no método onBind

Service – Linguagem AIDL

```
private final AcessoRemoto.Stub conexao = new AcessoRemoto.Stub() {
        private int count;

         // Implementa o método da interface AcessoRemoto
         public int count() throws DeadObjectException {
                  count++;
                  return count;
         }
};

@Override
public IBinder onBind(Intent intent) {
        return conexao;
}
```

AlarmManger

* Permite agendar uma tarefa para ser executada no futuro. Para isso o AlarmManager utiliza uma Intent para que seja enviada uma mensagem ao SO na data e hora desejada.

* AlarmManager ficará ativo para executar a sua tarefa em qualquer estado do telefone, mesmo ele travado. Somente se o telefone for reiniciado o AlarmManager será removido

AlarmManager

* Métodos de manipulação do AlarmManager:
  * **set(int tipo, long dataHora, Intent intent)**: Método para agendar uma tarefa, onde a mesma será executada na dataHora definida na chamada do método
  * **setRepeating(int tipo, long dataHora, long intervalo, Intent Intent):** Método parecido com o anterior só que irá repetir de tempos em tempos a tarefa agendada. Será repetido até que o método cancel(intent) seja chamado
  * **cancel(Intent intent):** Cancela o Alarm baseado na intent fornecida, que deve ser a mesma utilizada na criação do alarm.

Redes e Multimídia

Socket

* Para a comunicação via Socket, o Android disponibiliza a API ***java.net.Socket***. Para este exemplo será utilizado o construtor que contém os seguintes parâmetros:
  * **String dstName –** Endereço de IP do servidor Socket
  * **int dstPort –** Porta disponível no servidor Socket.

Socket

* Para iniciar o processo de leitura e escrita basta utilizar os métodos
  * **socket.getOutputStream() -** Recupera um DataOutputStream para escrita.
  * **socket.getInputStream() –** Recupera um DataInputStream para leitura.

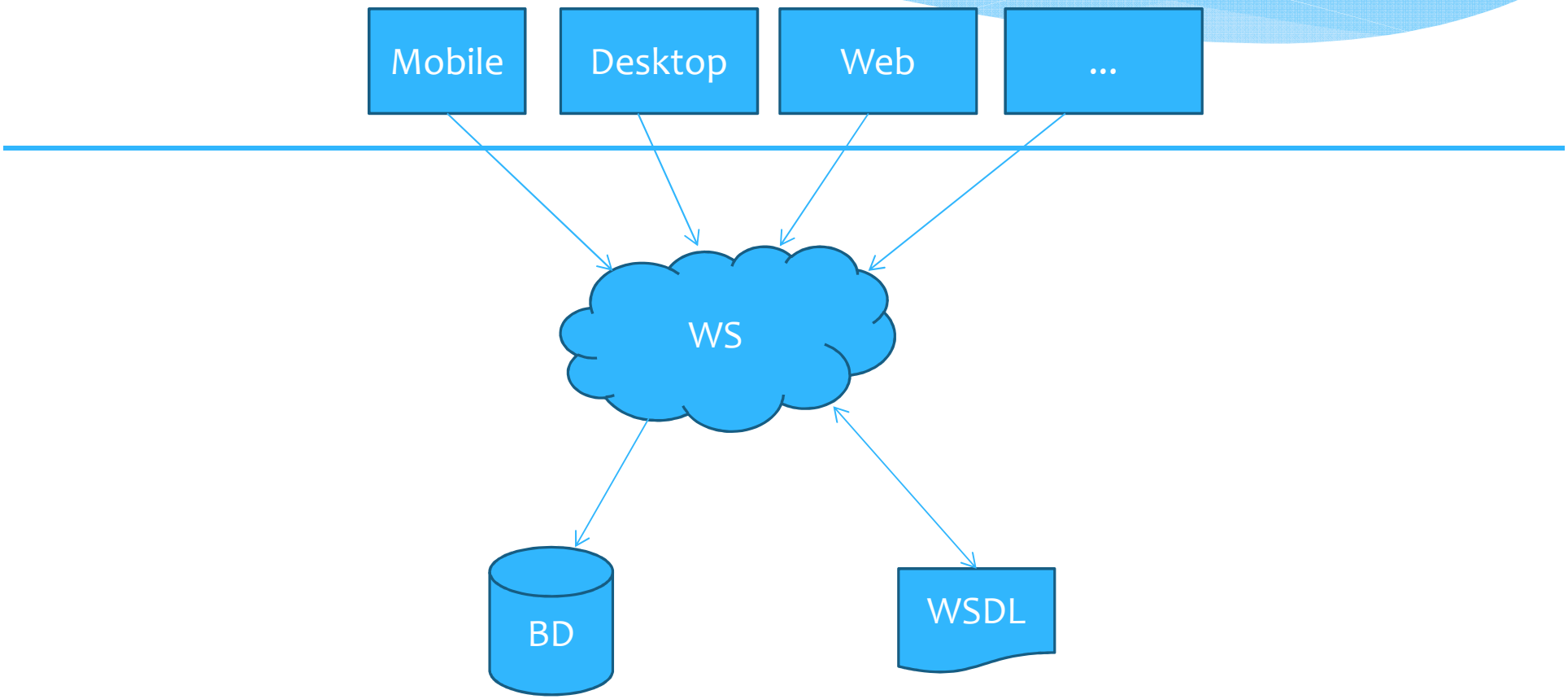
Consumindo WebServices

* **WebService** - Disponibiliza serviços na Web oferecendo acesso a métodos e dados para aplicações cliente. Padrão de comunicação utilizado pelo WebService é o SOAP.
* **SOAP** - Sigla para "Simple Object Access Protocol". Padrão XML para comunicação e envio de dados entre Web Services e Cliente.

Consumindo WebServices

* **WSDL** - Sigla para "Web Services Description Language". É uma interface que define os métodos pertencentes ao serviço. Não é necessária na programação, é apenas para especificar o "contrato" que o cliente deverá cumprir para utilizar o serviço.

Consumindo WebServices



Referências

* Fragments
  * http://developer.android.com/guide/components/fragments.html
  * http://assets.en.oreilly.com/1/event/68/Fragments%20for%20All%20Presentation.pdf