sãojudas universidade	FTCE Faculdade de Tecnologia e Ciências Exatas

USJT - 2019 – Programação para Web e Dispositivos Móveis

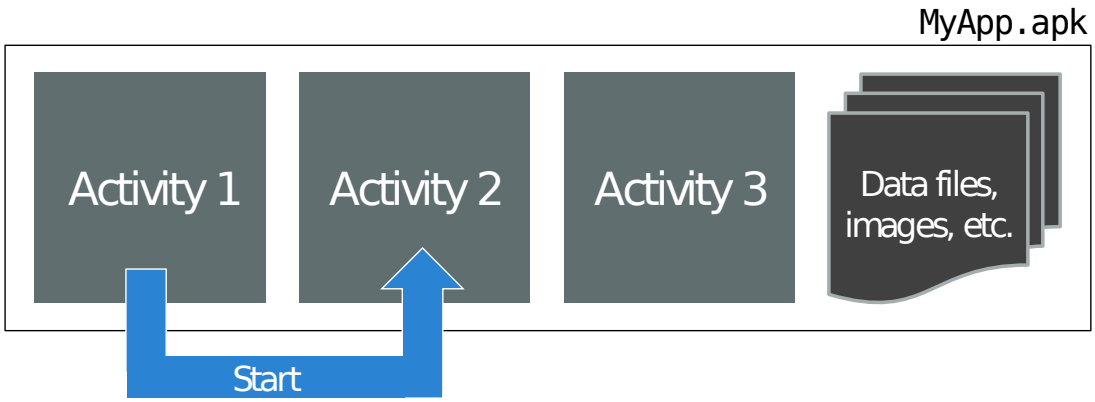
Professor: Anderson e Bossini

Aula: 03
Assunto: Activity - Intent - Stack Navigation

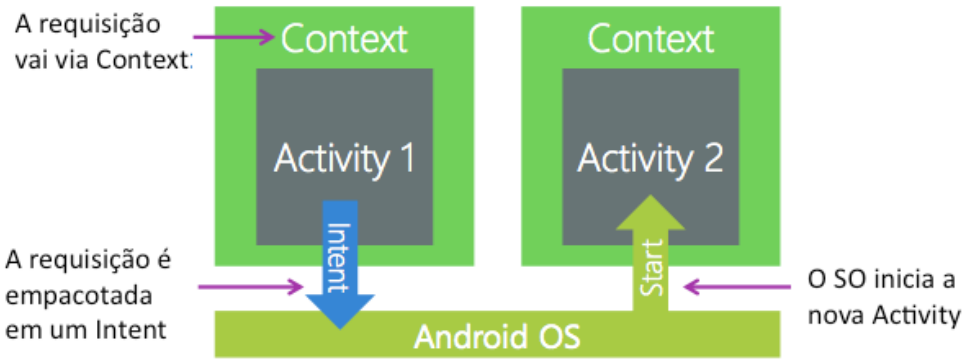
Conceitos básicos

Introdução

Como vimos na aula passada, uma aplicação Android é um conjunto de Activities.



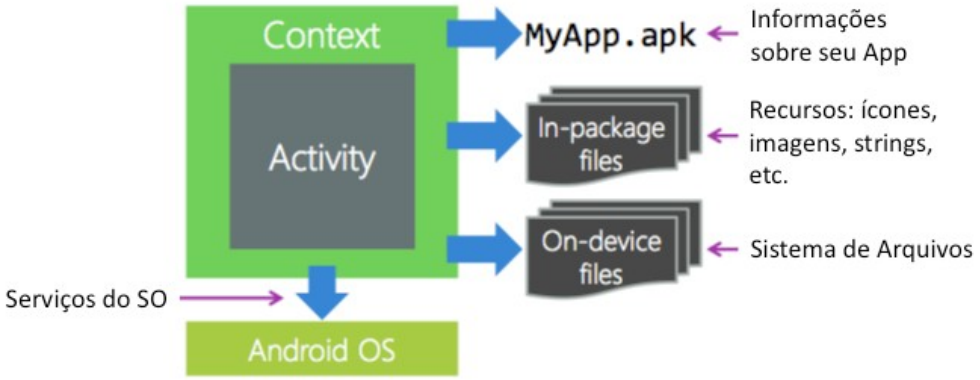
Visão geral do início de uma Activity



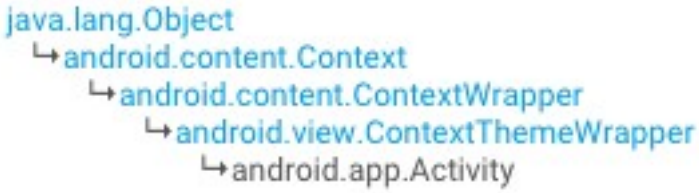
* SO é o Sistema Operacional, ou seja, o Android

O que é um Context?

O Context é um ponto de acesso para o ambiente Android que está rodando seu app.



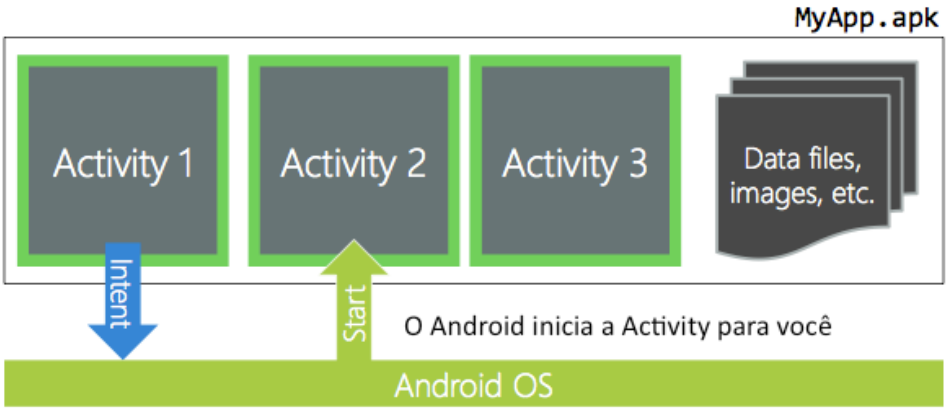
A Activity é subclasse de Context. Isso garante que cada Activity tenha acesso ao ambiente para carregar recursos e interagir com o Android.



O ContextWrapper é uma implementação do pattern Proxy e o ContextThemeWrapper dá suporte para estilos e temas.

O que é um Intent?

Um Intent é uma requisição para o Android começar uma nova Actvity.



O que é um Intent explícito?

É um Intent que identifica exatamente qual a Activity que será iniciada.

```
public Class Intent... {
   public Intent(Context packageContext, Class<?> cls) {
                  ...
   }
}
```

`Context packageContext` deve ser um Context associado com o **.apk** que contem a Activity que você quer iniciar. Use sua Activity atual, uma vez que ela é um Context e que está no mesmo .apk que a Activity que você quer iniciar.

`Class<?> cls` identifica univocamente a Activity a ser iniciada.

Métodos para Iniciar uma Activity

Estes métodos estão na classe Context. Os mais comuns são:

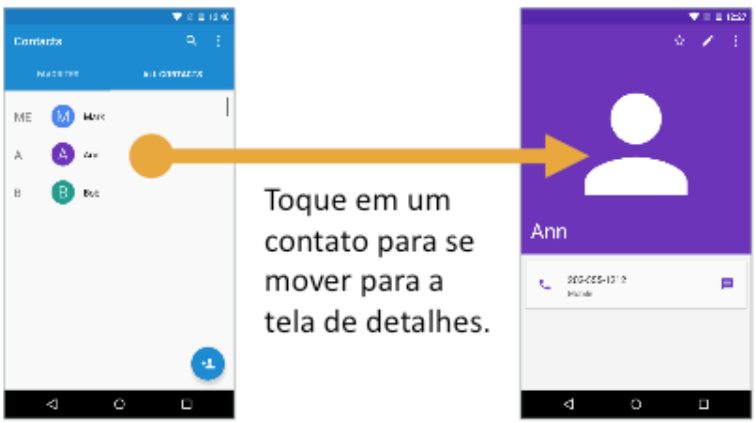
```
public abstract void startActivity(Intent intent);

public abstract void startActivity(Intent intent, Bundle options);
```

Como iniciar uma Activity

```
public class ActivityMain extends Activity{

   public void buscarClientes(View view) {

      Intent intent = new Intent(this, ListaClientesActivity.class);

      startActivity(intent);
    }
}
```

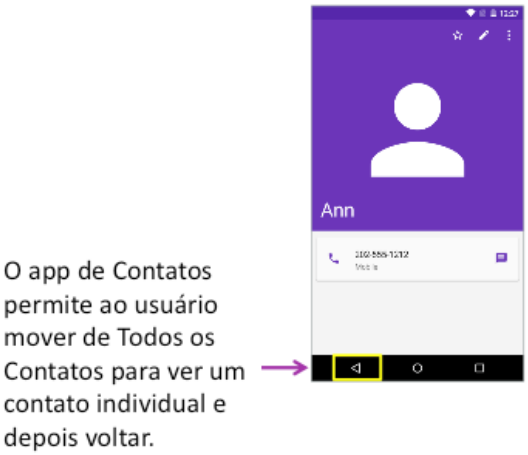
Navegação

A navegação descreve o caminho que você cria no seu App para permitir ao usuário se mover por várias Activities.



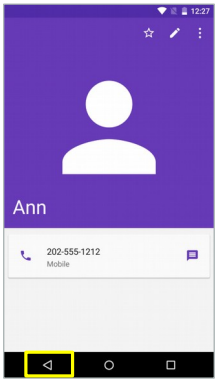
O botão Back

Permite ao usuário voltar para a Activity anterior.

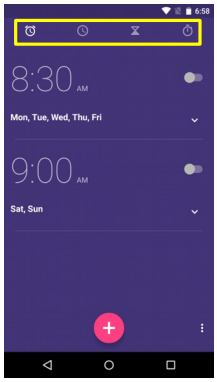


Padrões de Navegação

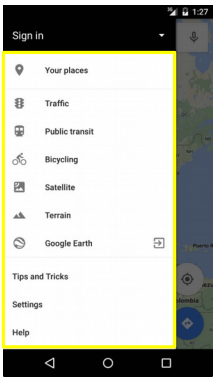
Existem vários no Android. Neste curso iremos aprender somente o Stack.



Stack



Tab



Drawer

Stack Navigation

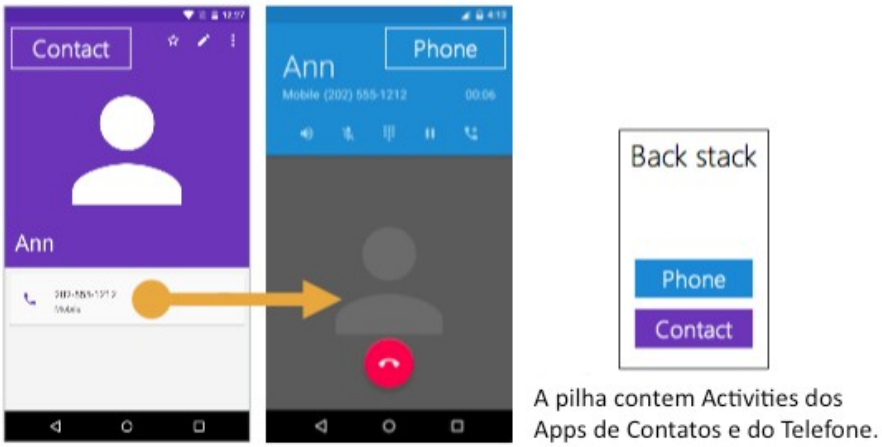
A navegação por pilha (stack navigation) armazena a sequência de Activities pelas quais o usuário passou em uma pilha (stack) e permite ao usuário retornar de qualquer Activity até o início.

O que é o back-stack?

É o histórico armazenado de todas as Activities vivas (live Activities) do usuário.

Escopo do back-stack

As Activities do back-stack podem se expandir por vários apps.



A pilha contem Activities dos Apps de Contatos e do Telefone.

Push e Pop

O Android empilha (push) uma Activity no back-stack assim que você a inicia.

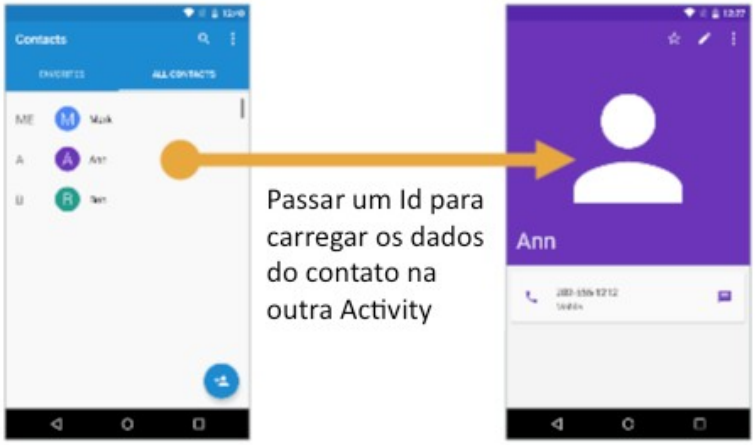
```
startActivity(intent);
```

E desempilha (pop) quando o usuário pressiona o botão Back.

Você pode também "chamar o botão Back" via código. Basta finalizar a Activity chamando o método `public void finish()`

Passagem de Dados entre Activities

As Activities geralmente precisam passar dados umas para as outras.

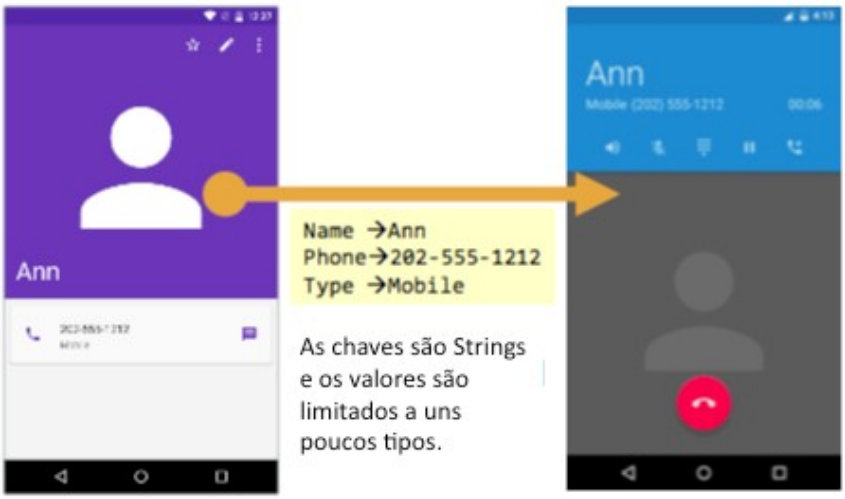


- Cada App roda no seu próprio processo.
- Cada Activity roda no processo do seu próprio App.

- Por este motivo, **apenas tipos primitivos e objetos serializáveis podem ser passados entre Activities;** não é possível passar referências a objetos, uma vez que não há compartilhamento de memória.

O que é um Bundle

Bundle é uma coleção de chave/valor passada entre Activities.



O Bundle tem métodos put/get para os tipos primitivos e strings.

Porém, existe um Bundle interno ao Intent chamado Intent Extra. Geralmente colocamos valores diretamente nele.

Colocar valores no Intent Extra

```
public class MainActivity extends AppCompatActivity {
    private EditText nome;
    public static final String CHAVE = "br.usjt.arqdesis.clientep1.chave";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        nome = (EditText)findViewById(R.id.busca_nome_cliente);

    }

    public void buscarClientes(View view){
        Intent intent = new Intent(this, ListaClientesActivity.class);
        String chave = nome.getText().toString();
        intent.putExtra(CHAVE, chave);
        startActivity(intent);
    }
}
```

Pegar valores do Intent Extra

```
        Intent intent = getIntent();
        String chave = intent.getStringExtra(MainActivity.CHAVE);
```

Recebendo resultados de uma Activity

Para obter um retorno de uma Activity, use o método da classe Activity

```
public void startActivityForResult (Intent intent, int requestCode)
```

Neste caso, a Activity retorna um código de status da sua execução para quem a chamou. Para configurar este status, a Activity chamada usa seu método:

```
public final void setResult (int resultCode)
```

Se a Activity chamada quiser também retornar dados, ela usa o Bundle interno do Intent a ser retornado usando o método

```
protected void onActivityResult (int requestCode, int resultCode,
Intent data)
```

A Activity chamadora pega este resultado chamando o método de callback, que é chamado imediatamente antes do onResume()

```
protected void onActivityResult (int requestCode, int resultCode,
Intent data)
```

Exemplo: Activity, Intent e Navigation

Criando uma busca e visualização de chamados de service desk.

1. Crie um novo projeto no AndroidStudio chamado **ServiceDesk**. Configure o CompanyDomain para **deswebmob.usjt.br**. Use a API 22.

2. Crie uma EmptyActivity chamada MainActivity. Esta activity é quase igual a do tutorial da primeira aula de Android e irá ler o nome da fila a ser buscada na lista. Se o nome for deixado em branco, todos os chamados serão listados. Para isso, altere a classe ActivityMain.java e o layout activity_main.xml conforme o código abaixo. Procure digitar e entender, não apenas copiar e colar. Altere também o strings.xml. Crie o dimens.xml se não houver.

strings.xml

```
<resources>
   <string name="app_name">ServiceDesk</string>
   <string name="busca_fila">fila solucionadora</string>
   <string name="botao_busca">Buscar</string>
</resources>
```

dimens.xml

```xml
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="br.usjt.deswebmob.servicedesk.MainActivity">

    <EditText
        android:id="@+id/busca_fila"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/busca_fila"
        android:layout_weight="1" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/botao_busca"
        android:onClick="buscarChamados"/>
</LinearLayout>
```

MainActivity.java

```java
package br.usjt.deswebmob.servicedesk;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends Activity {
    private EditText txtNome;
    public static final String NOME = "br.usjt.deswebmob.servicedesk.nome";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtNome = (EditText)findViewById(R.id.busca_fila);

    }

    public void buscarChamados(View view){
        Intent intent = new Intent(this, ListarChamadosActivity.class);
        String nome = txtNome.getText().toString();
        intent.putExtra(NOME, nome);
```

```
        startActivity(intent);
    }
}
```

Vamos criar agora um ListView, que é uma tela com uma lista. Faremos um listview bem simples hoje, e na próxima aula faremos um mais sofisticado. Na próxima aula também explicaremos mais sobre layouts.

Crie uma nova activity, chamada ListarChamadosActivity. Quando você clicar no botão Buscar da tela anterior, esta atividade será chamada. O valor que você digitou no campo busca_fila será trazido para a nova activity por meio do intent.

Como ainda não há o acesso a serviços REST configurado, teremos que simular a lista em um ArrayList. Se você tiver digitado alguma chave de busca, o ArrayList conterá apenas os chamados encontrados. Se não tiver digitado nada, todos os chamados serão trazidos.

Note que no código do ListarChamadosActivity.java haverá um listener para clicks no listview. Quando você clicar sobre um nome, uma nova activity irá aparecer mostrando o nome escolhido.

activity_listar_chamados.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="br.usjt.deswebmob.servicedesk.ListarChamadosActivity">

    <ListView
        android:id="@+id/listview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fastScrollEnabled="true">

    </ListView>
</LinearLayout>
```

ListarChamadosActivity.java

```
package br.usjt.deswebmob.servicedesk;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
```

```
import android.widget.ArrayAdapter;
import android.widget.ListView;

import java.util.ArrayList;

public class ListarChamadosActivity extends Activity {

    public static final String DESCRICAO =
"br.usjt.deswebmob.servicedesk.descricao";
    ArrayList<String> lista;
    Activity atividade;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_listar_chamados);
        atividade = this;
        Intent intent = getIntent();
        String chave = intent.getStringExtra(MainActivity.NOME);
        lista = buscaChamados(chave);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, lista);
        ListView listView = (ListView) findViewById(R.id.listview);
        listView.setAdapter(adapter);

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                                    int position, long id) {

                // manda para a tela de detalhe
                Intent intent = new Intent(atividade, DetalheChamadoActivity.class);
                intent.putExtra(DESCRICAO, lista.get(position));

                startActivity(intent);

            }

        });

    }

    public ArrayList<String> buscaChamados(String chave){
        ArrayList<String> lista = geraListaChamados();
        if (chave == null || chave.length() == 0){
            return lista;
        } else {
            ArrayList<String> subLista = new ArrayList<>();
            for(String nome:lista){
                if(nome.toUpperCase().contains(chave.toUpperCase())){
                    subLista.add(nome);
                }
            }
            return subLista;
        }
    }

    public ArrayList<String> geraListaChamados(){
        ArrayList<String> lista = new ArrayList<>();
        lista.add("Desktops: Computador da secretária quebrado.");
        lista.add("Telefonia: Telefone não funciona.");
        lista.add("Redes: Manutenção no proxy.");
        lista.add("Servidores: Lentidão generalizada.");
        lista.add("Novos Projetos: CRM");
        lista.add("Manutenção Sistema ERP: atualizar versão.");
```

```
        lista.add("Novos Projetos: Rede MPLS");
        lista.add("Manutenção Sistema de Vendas: incluir pipeline.");
        lista.add("Manutenção Sistema ERP: erro contábil");
        lista.add("Novos Projetos: Gestão de Orçamento");
        lista.add("Novos Projetos: Big Data");
        lista.add("Manoel de Barros");
        lista.add("Redes: Internet com lentidão");
        lista.add("Novos Projetos: Chatbot");
        lista.add("Desktops: troca de senha");
        lista.add("Desktops: falha no Windows");
        lista.add("Novos Projetos: ITIL V3");
        lista.add("Telefonia: liberar celular");
        lista.add("Telefonia: mover ramal");
        lista.add("Redes: ponto com defeito");
        lista.add("Novos Projetos: ferramenta EMM");
        return lista;
    }
}
```

Crie agora mais uma activity que será a tela de detalhes do chamado que, por enquanto, irá apenas mostrar a descrição do chamado escolhido. Para isso você terá que criar uma nova activity, a DetalheChamadoActivity.

activity_detalhe_chamado.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="br.usjt.deswebmob.servicedesk.DetalheChamadoActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/chamado_selecionado"
        android:textStyle="bold" />

</RelativeLayout>
```

DetalheChamadoActivity.java

```
package br.usjt.deswebmob.servicedesk;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class DetalheChamadoActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detalhe_chamado);
        TextView nome = (TextView)findViewById(R.id.chamado_selecionado);
        Intent intent = getIntent();
```

```
        nome.setText(intent.getStringExtra(ListarChamadosActivity.DESCRICAO));
    }
}
```

Suba o Visual Studio Android Player e rode sua aplicação para testar.

Leia mais sobre listviews em

http://developer.android.com/reference/android/widget/ListView.html

Exercício Prático

1. Ajuste o aplicativo da aula passada (aquele que usa o GPS) para capturar localizações do usuário conforme ele se movimenta e guardar em um ArrayList. O ArrayList deve guardar as 50 localizações mais recentes. Entre cada par de localizações deve haver, ao menos, 200 metros de distância e a aplicação deve receber atualizações no intervalo de 2 em 2 minutos. Note que sua coleção de dados deve mostrar as 50 localizações mais recentes. Isso quer dizer que, quando a quinquagésima primeira for recebida, primeira deve ser removida da coleção.

2. A aplicação deve ter uma tela inicial com um FloatingActionButton. Quando clicado, a tela a seguir deve mostrar uma lista com as localizações. Basta mostrar latitude e longitude em cada item.

Bibliografia

Android API, package android.app; disponível em http://developer.android.com/reference/android/app/package-summary.html ; consultado em 02/09/15.