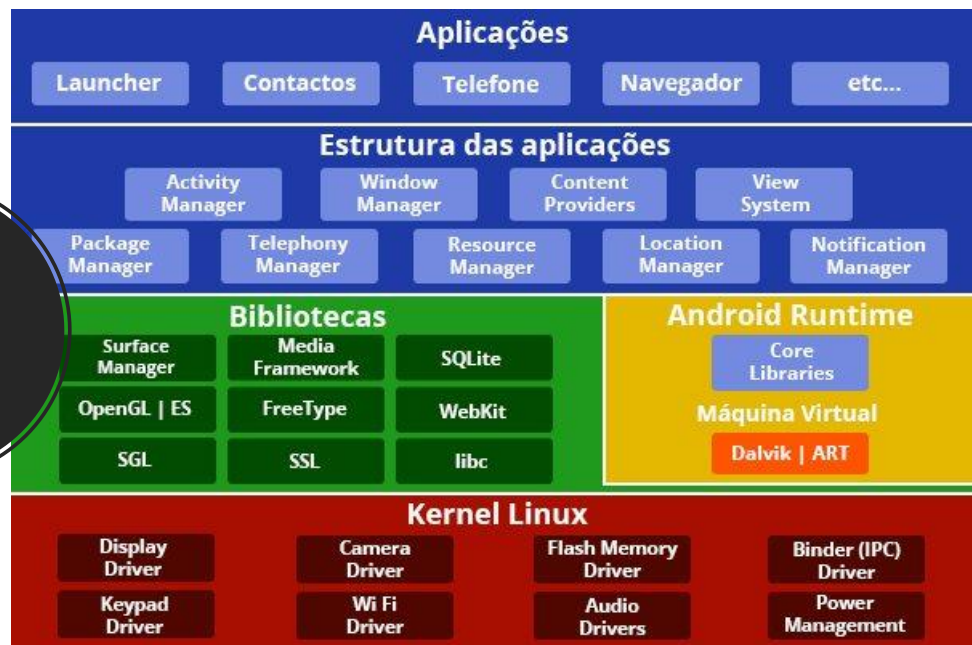


Arquitetura do Sistema Android

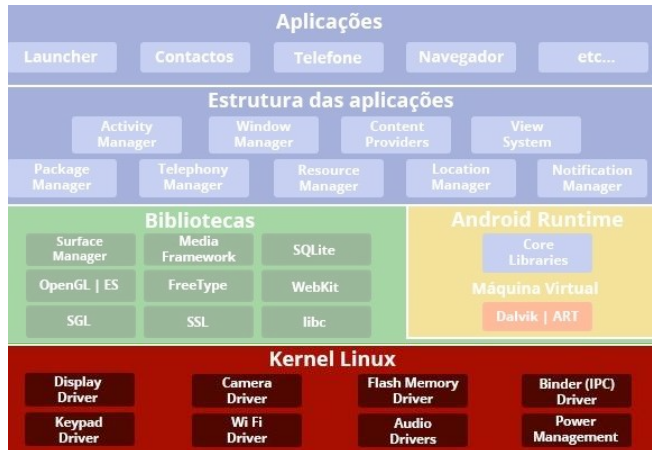
Camadas & Componentes



Arquitetura do sistema



Arquitetura do sistema



Kernel Linux

- Nível mais baixo da arquitetura, é a base para todas as demais camadas.
- Oferece um nível de abstração entre os dispositivos periféricos e suas especificidades, evitando assim o problema de ter que fazer a manipulação direta.
- O núcleo também é responsável pelo gerenciamento de rede e pelos drivers que realizam a interface com o hardware periférico.

Arquitetura do sistema

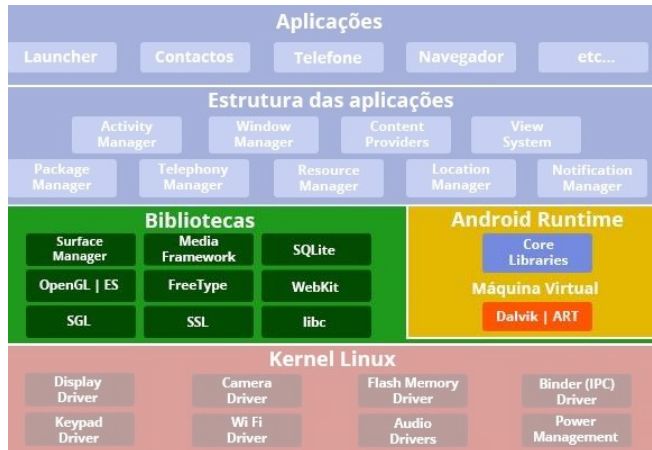


Bibliotecas

Acima do kernel, consiste num conjunto de bibliotecas para:

- Armazenamento e compartilhamento de dados (SQLite)
- Navegação web (Webkit)
- Bibliotecas da ling. C (libc)
- Reprodução e gravação de conteúdo multimídia (áudio e vídeo)
- Segurança de rede (SSL)

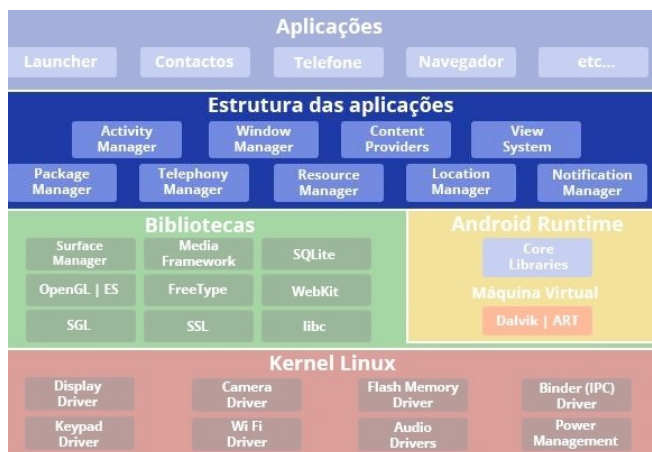
Arquitetura do sistema



Android Runtime

- Consiste na terceira camada da arquitetura e atua em conjunto com a segunda.
- Nela encontramos a Dalvik Virtual Machine, uma espécie de JVM especialmente projetada e otimizada para o Android.
Cada app roda seu próprio processo, que é uma instância da Dalvik VM.
- Contém também as bibliotecas básicas que permitem aos programadores utilizarem a linguagem Java no processo de desenvolvimento.

Arquitetura do sistema



Framework

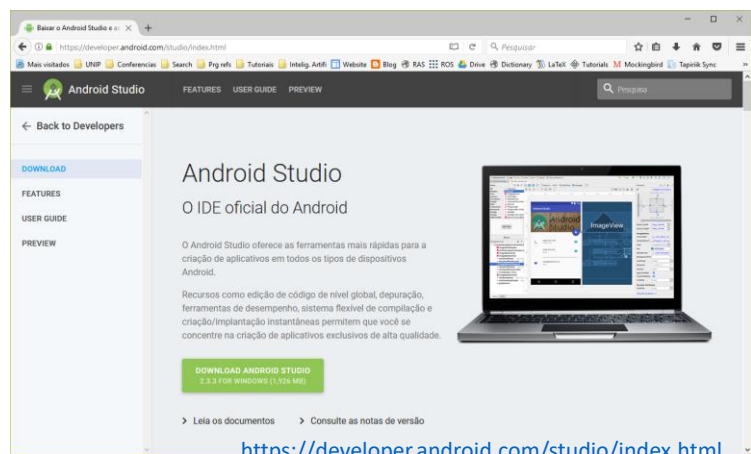
- Bibliotecas baseadas em Java que são específicas para o desenvolvimento em Android.
- Oferecem recursos que facilitam a criação de interfaces gráficas, manipulação de componentes e acesso a dados.
 - android.app
 - android.content
 - android.text
 - android.view
 - android.widget
 - android.database
 - android.webkit
 - android.os
 - android.opengl
 - ... dentre outros.
- Servem de classes base para construção do app

Ambiente de Desenvolvimento: *Android Studio*

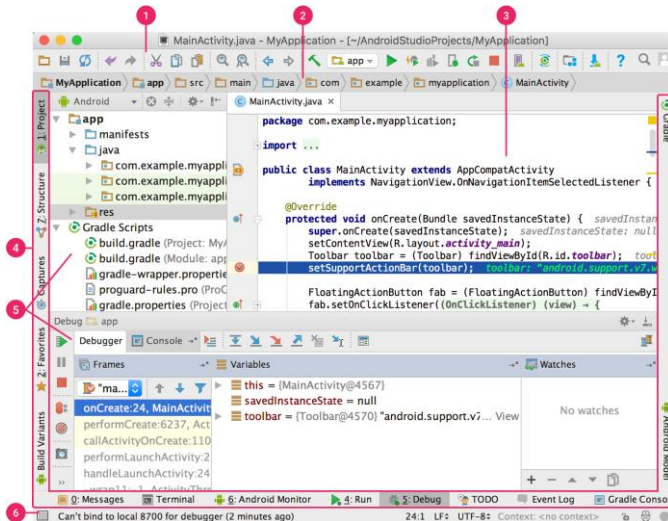


Android Studio

- Recursos da IDE:
 - *Instant Run*
 - Emulador
 - Autocompletar inteligente
 - Ferramenta de testes
- Gradle: compilação configurável e flexível
 - Inclusão de bibliotecas
 - Geração simultânea p/ diferentes dispositivos
 - Compatibilidade com C++ e o NDK
- Integração com Git
- Gerenciador de APIs



Conhecendo a IDE: interface com o usuário



Composto por:

- 1. Barra de ferramentas**
permite executar diversas ações, incluindo executar aplicativos e inicializar ferramentas do Android.
- 2. Barra de navegação**
navegação pelo projeto e na abertura de arquivos para edição.
- 3. Janela do editor**
o editor muda dependendo do tipo de arquivo.
- 4. Barra de janela de ferramentas**
fica fora da janela do IDE e permitem expandir ou recolher a janela de cada ferramenta.
- 5. Janela das ferramentas**
dá acesso a tarefas específicas, como gerenciamento de projetos, busca, controle de versão e muitos outros. Você pode expandi-las e recolhê-las.
- 6. Barra de status**
mostra o status do projeto e do próprio IDE, além de advertências e mensagens.

Estrutura de um projeto

AndroidManifest.xml: arquivo de configuração, descreve as características da aplicação e cada um de seus componentes.

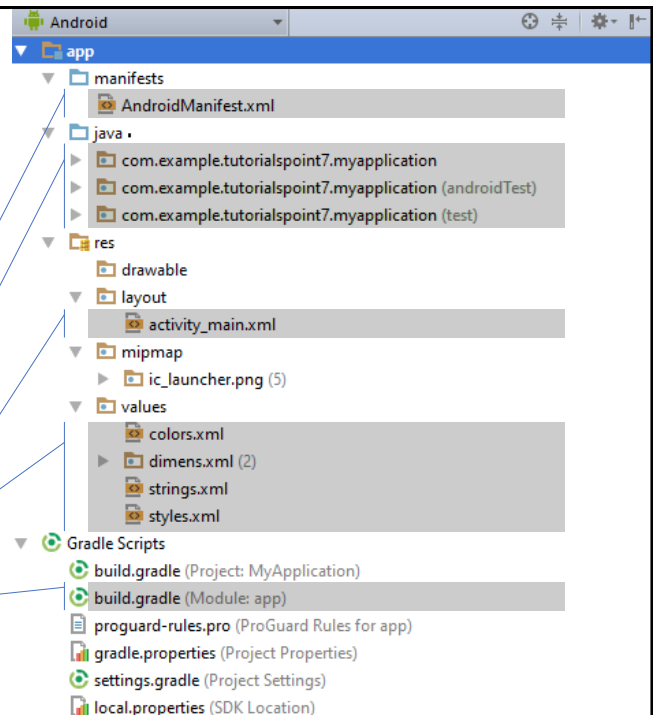
java: Contém todos os fontes da aplicação e, por padrão, incluem a *MainActivity.java*

res/drawable-hdpi: objetos desenháveis projetados para telas de alta resolução.

res/layout: arquivos que definem a interface com o usuário (telas).

res/values: diretório que armazena arquivos XML para vários recursos, como strings e cores.

Build.gradle: arquivo autogerado que contém informações sobre as versões do SDK (compilador e alvo), além do ID, versão nome da aplicação.



Componentes da Aplicação

- São os blocos básicos para a construção de uma aplicação Android
- Estes componentes são incluídos na aplicação através de seu arquivo manifesto (*AndroidManifest.xml*)

Principais tipos:

- **Activities:** responsáveis pela interface e a interação com o usuário.
- **Fragments:** representam uma parte ou porção da UI em uma Activity.
- **Views:** componentes gráficos que compõem a UI.
- **Intents:** mensagens trocadas entre componentes (ações).
- **Resources:** elementos externos, tais como: imagens, strings e constantes.
- Outros tipos:
 - **Services:** Manipulam os processos de background associados a aplicação.
 - **Broadcast Receivers:** Gerenciam a comunicação entre o Android (SO) e a aplicação.
 - **Content Providers:** Manipulam dados e gerenciam questões ligadas ao banco de dados.

O arquivo *AndroidManifest.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.unip.tap.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

A tag *<application>* encapsula todos os componentes relacionados à aplicação. Seus atributos definem o nome das variáveis que representam o ícone, o nome do app e o tema utilizado.

A tag *<activity>* identifica uma atividade. Enquanto seus atributos determinam a subclasse de Activity (*android:name*) e seus parâmetros (*android:label*)

Os elementos *<action>* e *<category>* aqui indicam respectivamente que a *activity* serve como ponto de início do app e que pode ser disparada através do ícone no launcher do sistema

O arquivo *MainActivity.java*

```
package br.unip.tap.helloworld;
```

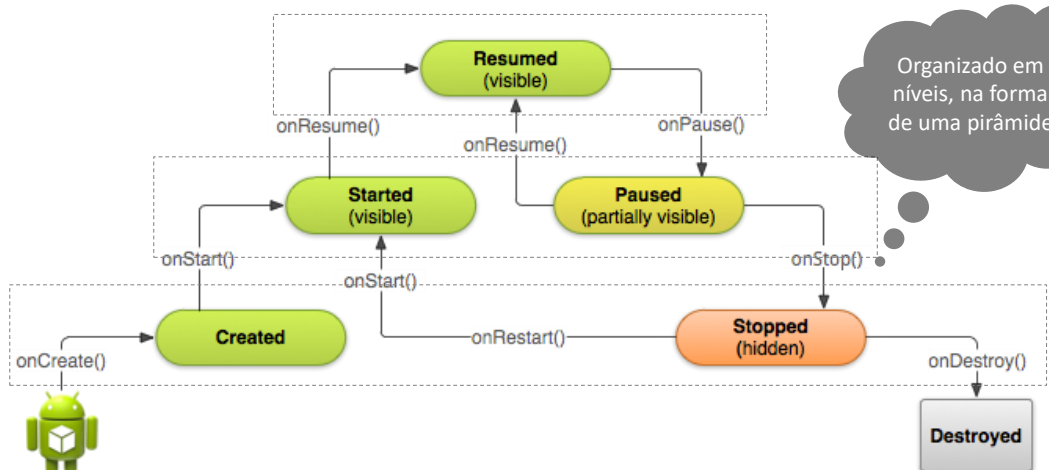
```
import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Faz referência a pasta *res/layout* e ao arquivo *activity_main.xml*

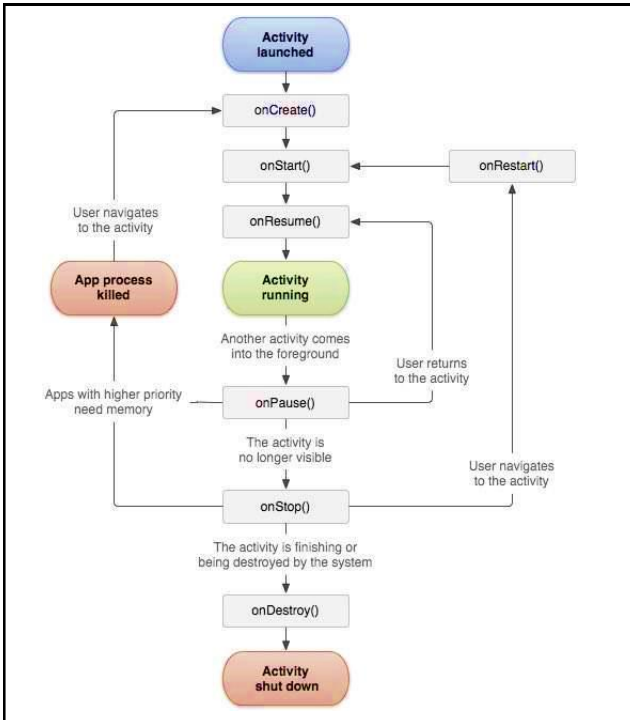
Activities: Ciclo de Vida



Activities: Ciclo de Vida

• A classe *Activity* define vários *callbacks* para manipular seus eventos, que são:

- **onCreate()**: quando a activity é criada
- **onStart()**: sempre que se torna visível na tela
- **onResume()**: quando o usuário inicia a interação
- **onPause()**: quando a activity não recebe mais entrada do usuário e não pode executar qualquer trecho de código.
- **onStop()**: quando a activity não está mais visível
- **onRestart()**: chamada no retorno da activity após ser parada
- **onDestroy()**: invocada antes de ser destruída pelo sistema



O layout: *activity_main.xml*

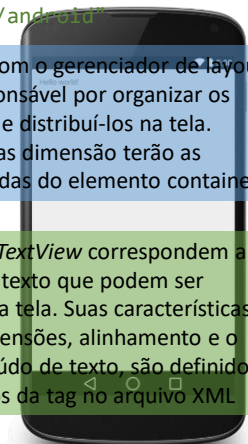
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:padding="@dimen/padding_medium"
    android:text="@string/hello_world"
    tools:context=".MainActivity" />
```

```
</RelativeLayout>
```

Começamos com o gerenciador de layout que será responsável por organizar os componentes e distribuí-los na tela. Neste caso suas dimensão terão as mesmas medidas do elemento container.

Componente *TextView* correspondem a elementos de texto que podem ser desenhados na tela. Suas características, tais como dimensões, alinhamento e o próprio conteúdo de texto, são definidos como atributos da tag no arquivo XML





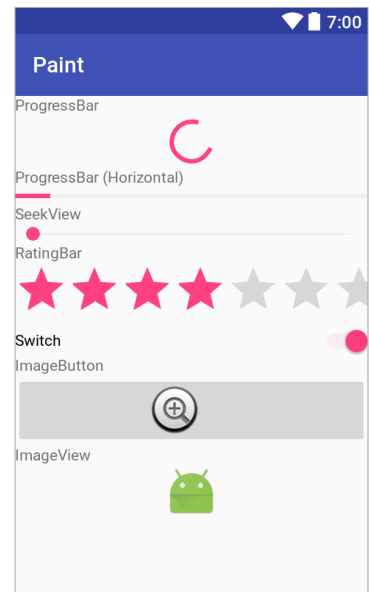
Layouts e Design de telas

Componentes (*View*)

- Conjunto rico de elementos para a construção de UI, cujo papel é coletar e exibir informações ao usuário.
- Os componentes mais comuns são do tipo *TextView*, porém oferecendo formatos para atender a propósitos específicos:
 - Textos planos de única linha ou múltiplas linhas
 - Senha (caracteres ou numéricas)
 - Máscaras para: e-mail, data, hora, etc.
- ... além de componentes de seleção e botões.
 - *CheckBox* e *RadioButton*
 - *Button* e *ButtonSwitch*

Componentes (*View*)

- Indicadores de progresso:
 - *ProgressBar* e *ProgressBar (Horizontal)*
- Controles deslizantes:
 - *SeekBar* (contínuo e discreto)
- Classificadorio:
 - *RatingBar* (estrelas)
- Elementos gráficos:
 - Exibição de imagens: *ImageView*
 - Botões icônicos: *ImageButton*



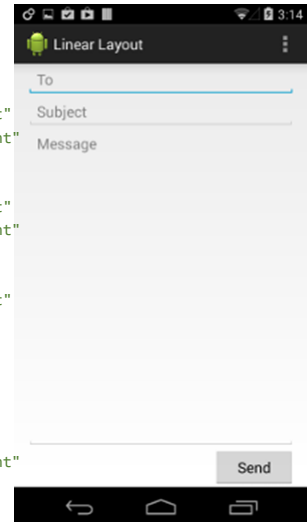
Layouts comuns:

LinearLayout



- Grupo de exibições que alinha todos os filhos em uma única direção vertical ou horizontal.
 - o atributo *android:orientation* especifica a direção.
- Os elementos são empilhados um após o outro.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



Layouts comuns:

RelativeLayout



- Especifica a localização de objetos filhos relativos entre si (filho A à esquerda do filho B) ou relativos aos pais (alinhados no topo em relação ao pai)
- Pode ser uma opção muito boa por evitar o uso de layouts aninhados

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:name="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:name="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:name="@+id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



Layouts comum:

WebView



- Exibição de conteúdo HTML
- As páginas são carregadas através do método `loadUrl(String)` do objeto `WebView`

```
WebView myWebView =
    (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```
- Conteúdos com JavaScript podem ser utilizados, mas precisam ser habilitados

Layouts com adaptador:

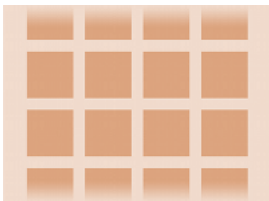
ListView



- Exibição na forma de lista de itens
- Elementos roláveis, inseridos automaticamente por um *Adapter*
 - Guiados por um cursor, como os resultado de uma consulta ao BD por exemplo.
- Bastante interessante para a exibição de conteúdos sob demanda.
 - Exemplo: postagens

Layouts com adaptador:

GridView



- `ViewGroup` que apresenta os itens na forma de uma matriz rolável e bidimensional.
- Assim como o *ListView*, precisa ser alimentado por um *Adapter*.

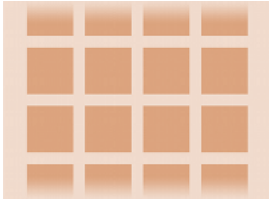
```
** res/layout/main.xml **
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="90dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
/>

** HelloGridView.java **
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    GridView gridview = (GridView) findViewById(R.id.gridview);
    gridview.setAdapter(new ImageAdapter(this));

    gridview.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View v,
            int position, long id) {
            Toast.makeText(HelloGridView.this, "" + position,
                Toast.LENGTH_SHORT).show();
        }
    });
}
```

Layouts com adaptador: GridView



- ViewGroup que apresenta os itens na forma de uma matriz rolável e bidimensional.
- Assim como o *ListView*, precisa ser alimentado por um *Adapter*.

```
** HelloGridView.java **
public class ImageAdapter extends BaseAdapter {
    private Context mContext;

    public ImageAdapter(Context c) { mContext = c;}
    public int getCount() { return mThumbIds.length;}
    public Object getItem(int position) { return null; }
    public long getItemId(int position) { return 0;}

    // cria um ImageView para cada item referenciado pelo Adapter
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
        if (convertView == null) {
            // se não reciclado, inicializa os atributos
            imageView = new ImageView(mContext);
            imageView.setLayoutParams(new GridView.LayoutParams(85, 85));
            imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
            imageView.setPadding(8, 8, 8, 8);
        } else {
            imageView = (ImageView) convertView;
        }

        imageView.setImageResource(mThumbIds[position]);
        return imageView;
    }

    // referências para as imagens
    private Integer[] mThumbIds = {
        R.drawable.sample_0, R.drawable.sample_1,
        R.drawable.sample_2, R.drawable.sample_3,
        R.drawable.sample_4, R.drawable.sample_5,
        R.drawable.sample_6, R.drawable.sample_7
    };
}
```