

Desenvolvimento Mobile II

Pablo Leon Rodrigues



1. Desenvolvimento Mobile II

1. Arquitetura
2. Principais Componentes de um App
3. Ambiente de desenvolvimento
4. Primeira Activity

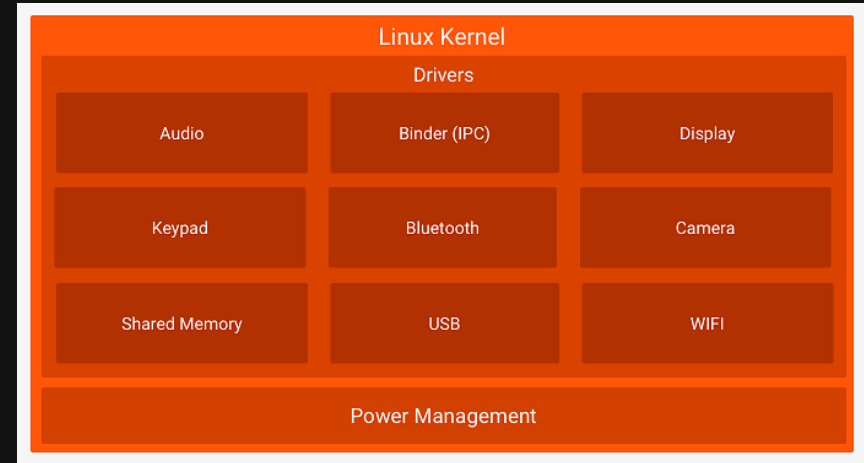
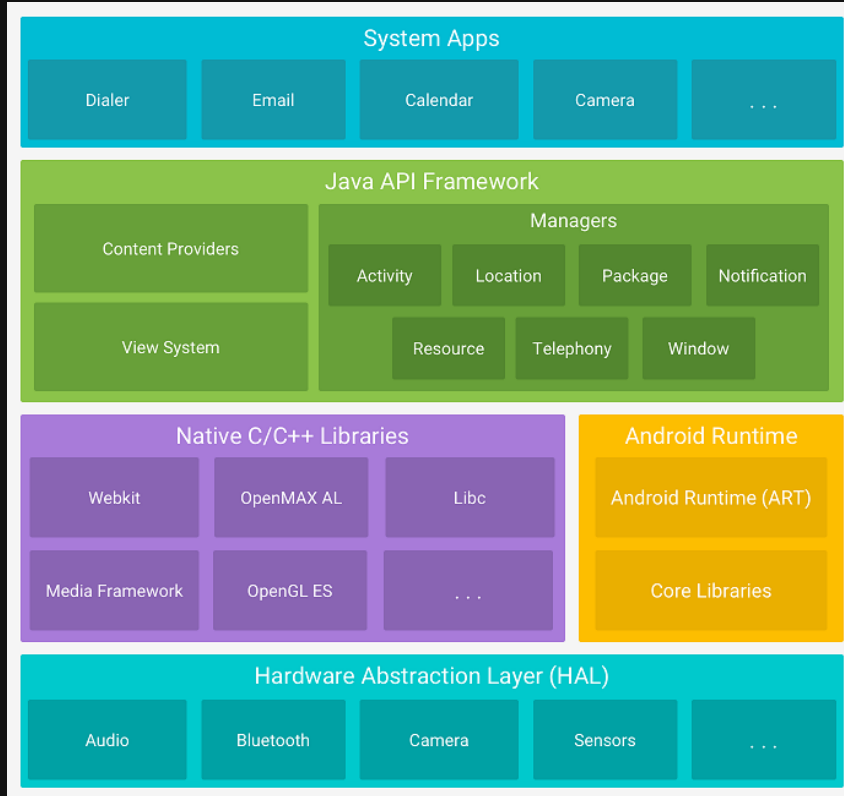
O Android é o sistema operacional para dispositivos móveis mais conhecido do mundo. Ele está presente em bilhões de dispositivos, desde smartphones até relógios, tablets TVs e muito mais.



- Diversas lojas: Google Play, Amazon, Samsung, Positivo
- Diversas opções de fabricantes de aparelhos
- Baseado em software livre
- Pouca burocracia para cadastrar-se como desenvolvedor e publicação de Apps
- Preço baixo dos aparelhos
- Paga apenas uma taxa entrada de \$25 para desenvolvedor

Arquitetura

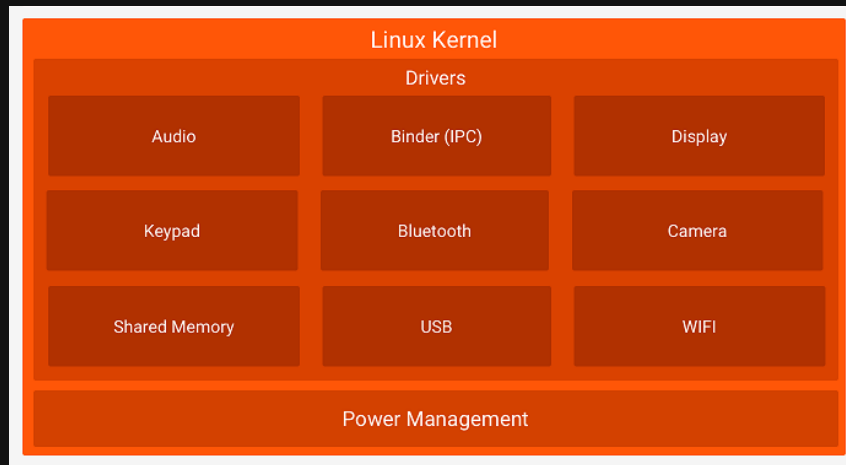
O Android é uma pilha de software com base em Linux de código aberto criada para diversos dispositivos. O diagrama ao lado mostra a maioria dos componentes da plataforma Android.



Linux Kernel

A fundação da plataforma Android é o kernel do linux. O Android Runtime (ART) confia no kernel do Linux funcionalidades como encadeamento e gerenciamento de memória de baixo nível.

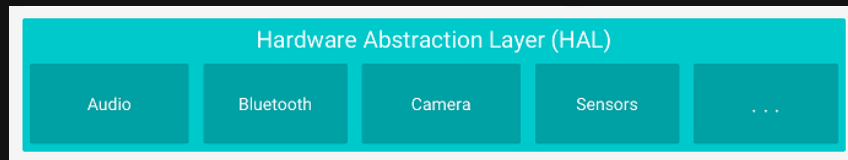
Permite que o Android aproveite os recursos de segurança principais e que os fabricantes dos dispositivos desenvolvem.



Camada de abstração de hardware (HAL)

A camada de abstração de hardware (HAL) fornece interfaces padrão que expõem as capacidades de hardware do dispositivo para a estrutura da Java API de maior nível.

A HAL consiste em módulos de biblioteca, que implementam uma interface para um tipo específico de componente de hardware, como o módulo de câmera ou bluetooth.

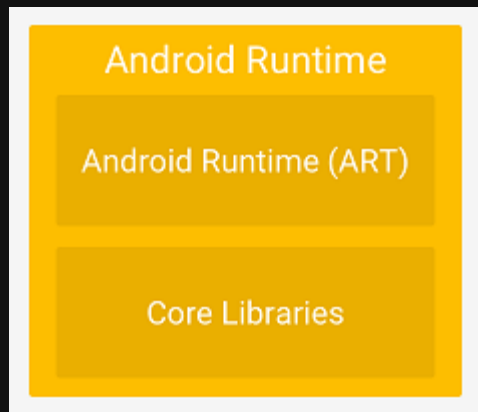


Quando uma Framework API faz uma chamada para acessar o hardware do dispositivo, o sistema Android carrega o módulo da biblioteca para este componente de hardware.

Android Runtime

Cada aplicativo executa o próprio processo com uma instância própria do Android Runtime (ART).

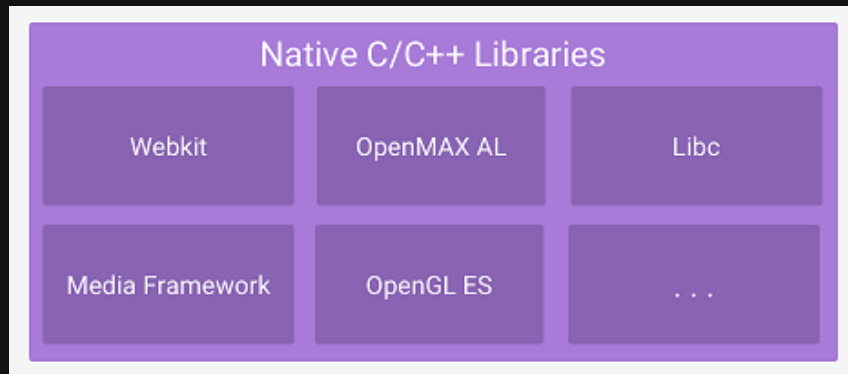
O ART é projetado para executar várias máquinas virtuais em dispositivos de baixa memória executando arquivos DEX (um formato de bytecode projetado especialmente para Android). Também contém um conjunto das principais bibliotecas de tempo de execução



Bibliotecas

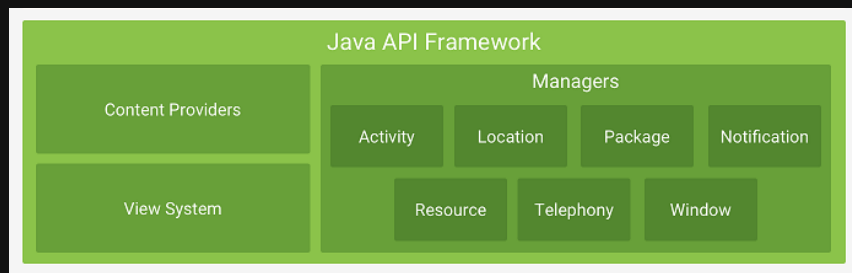
Vários componentes e serviços principais do sistema Android. São implementados por código nativo que exige bibliotecas nativas programadas em C e C++.

Por exemplo, é possível acessar OpenGL ES pela Java OpenGL API da estrutura do Android para adicionar a capacidade de desenhar e manipular gráficos 2D e 3D no aplicativo.



Framework

O conjunto completo de recursos do SO Android, disponível pelas APIs programadas na linguagem Java, incluindo:

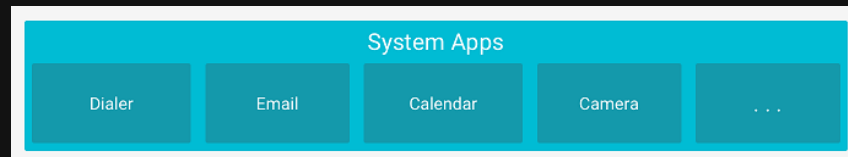


- Sistema de visualização para programar a UI de um aplicativo, com listas, grades, caixas de texto, botões e até mesmo um navegador da web incorporado
- Gerenciador de recursos, fornecendo acesso a recursos sem código como strings localizadas, gráficos e arquivos de layout
- Gerenciador de notificação que permite que todos os aplicativos exibam alertas personalizados na barra de status
- Gerenciador de atividade que gerencia o ciclo de vida dos aplicativos e fornece uma pilha de navegação inversa
- Provedores de conteúdo que permite que aplicativos acessem dados de outros aplicativos

Aplicações de Sistema

O SO Android vem com um conjunto de aplicativos principais para e-mail, envio de SMS, calendários, navegador de internet, contatos etc.

Os aplicativos inclusos na plataforma não têm status especial entre os aplicativos que o usuário opta por instalar. Portanto, um aplicativo terceirizado pode se tornar o navegador da Web, o aplicativo de envio de SMS ou até mesmo o teclado padrão do usuário (existem algumas exceções, como o aplicativo Configurações do sistema).



Principais Componentes de um App

Atividades(*Activity*)

Atividades representam uma tela única com uma interface do usuário. Por exemplo, um aplicativo de e-mail pode ter uma *Activity* que mostra uma lista de novos e-mails, outra *Activity* que compõe um e-mail e outra ainda que lê e-mails.

View

Uma view é uma interface de usuário(UI, *user interface*), é a tela propriamente dita.

Lógica

A lógica da *Activity* é o código fonte que faz a tela funcionar, componentes, bibliotecas, etc...

Serviços

Serviços são componentes executados em segundo plano para realizar operações ou para utilizar processos remotos.

Por exemplo, um serviço pode tocar música em segundo plano enquanto o usuário está em um aplicativo diferente ou buscar dados na rede sem bloquear a interação do usuário com uma atividade.

Outro componente, como uma atividade, pode iniciar o serviço e deixá-lo executar ou vincular-se a ele para interagir.

Um serviço é implementado como uma subclasse de `Service`.

Provedores de conteúdo

Provedores de conteúdo gerenciam um conjunto compartilhado de dados do aplicativo.

É possível armazenar os dados no sistema de arquivos, em um banco de dados SQLite ou em qualquer local de armazenamento persistente que o aplicativo possa acessar.

Por meio do provedor de conteúdo, outros aplicativos podem consultar ou até modificar os dados (se o provedor de conteúdo permitir).

Um provedor de conteúdo é implementado como uma subclasse de `ContentProvider` e precisa implementar um conjunto padrão de APIs que permitem a outros aplicativos realizar transações.

Receptores de transmissão

Receptores de transmissão são componentes que respondem a anúncios de transmissão por todo o sistema.

Muitas transmissões se originam do sistema, por exemplo, uma transmissão que anuncia que uma tela foi desligada, a bateria está baixa ou uma tela foi capturada.

Os aplicativos também podem iniciar transmissões, por exemplo, para comunicar a outros dispositivos que alguns dados foram baixados no dispositivo e estão disponíveis para uso.

Embora os receptores de transmissão não exibam nenhuma interface do usuário, eles podem criar uma notificação na barra de status para alertar ao usuário quando ocorre uma transmissão.

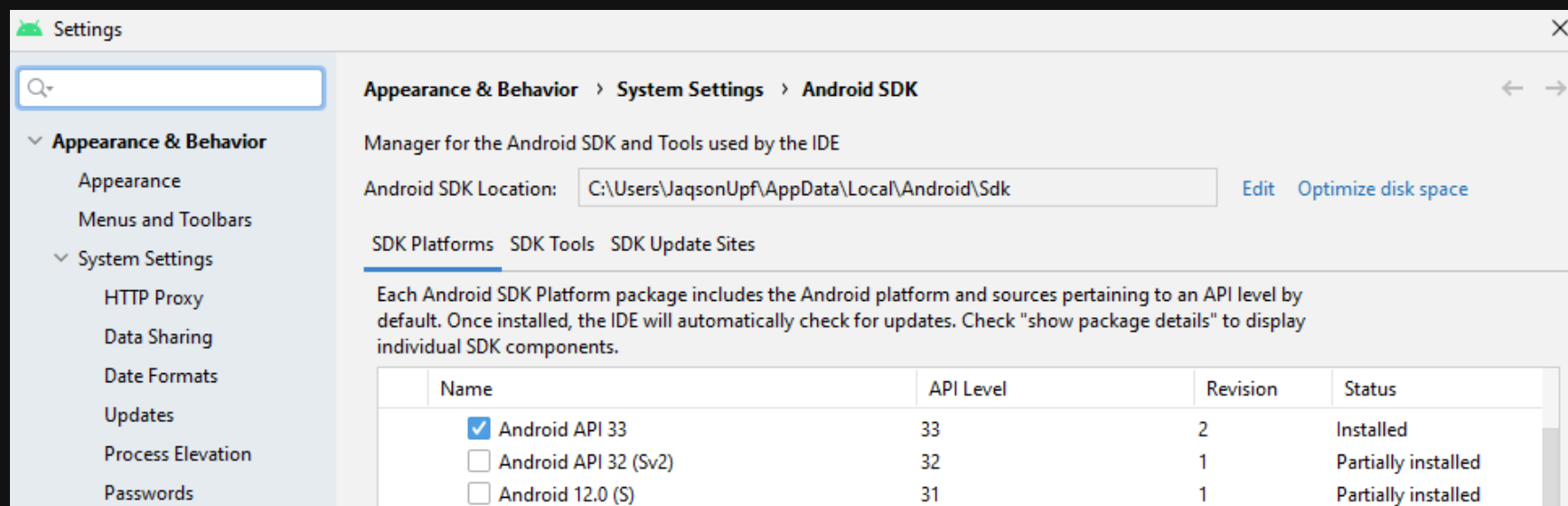
Os receptores de transmissão são implementados como subclasses de `BroadcastReceiver`.

Ambiente de desenvolvimento

SDK

O Android SDK (*Software Development Kit*) contém várias bibliotecas coletivamente chamadas de Android Support Library. Os desenvolvedores Android podem incluir qualquer uma dessas bibliotecas se quiserem incorporar a funcionalidade da biblioteca em seus aplicativos.

O Android Studio vem com o SDK Manager, que pode ser acessado clicando em Tools > SDK Manager



The screenshot shows the 'Settings' window in Android Studio, specifically the 'Android SDK' tab under 'System Settings'. The 'Android SDK Location' is set to 'C:\Users\JaquesonUpf\AppData\Local\Android\Sdk'. Below this, the 'SDK Platforms' tab is selected, showing a list of installed and partially installed SDK platforms.

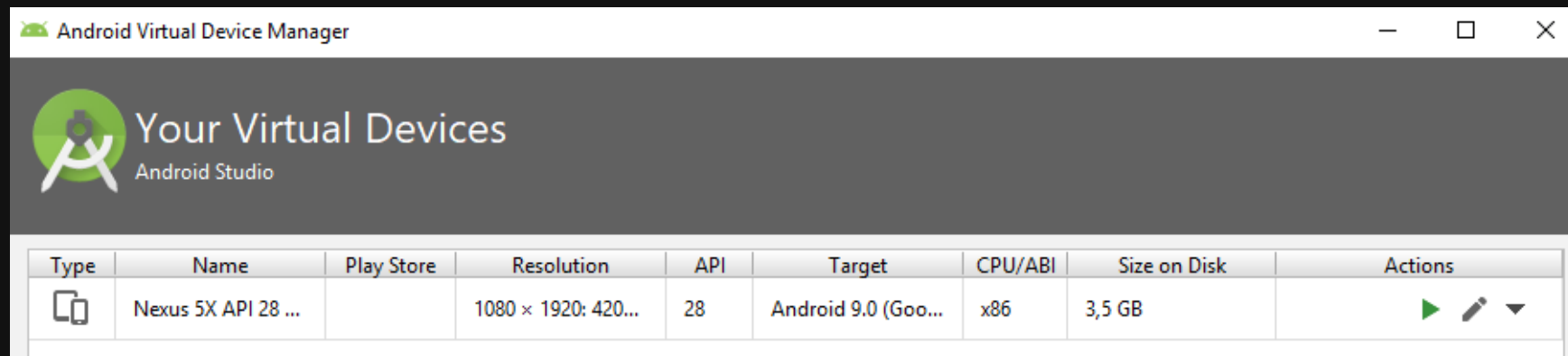
Name	API Level	Revision	Status
<input checked="" type="checkbox"/> Android API 33	33	2	Installed
<input type="checkbox"/> Android API 32 (Sv2)	32	1	Partially installed
<input type="checkbox"/> Android 12.0 (S)	31	1	Partially installed

AVD

O AVD (*Android Virtual Device*) oferece diversos emuladores para executar e testar os aplicativos.

O Android Studio vem com o AVD Manager, que pode ser acessado clicando em Tools > AVD Manager

Para criar um dispositivo virtual clicar em “*Create Virtual Device*”



Empty Activity

Creates a new empty activity

Name

Package name

Save location




Language




Minimum SDK

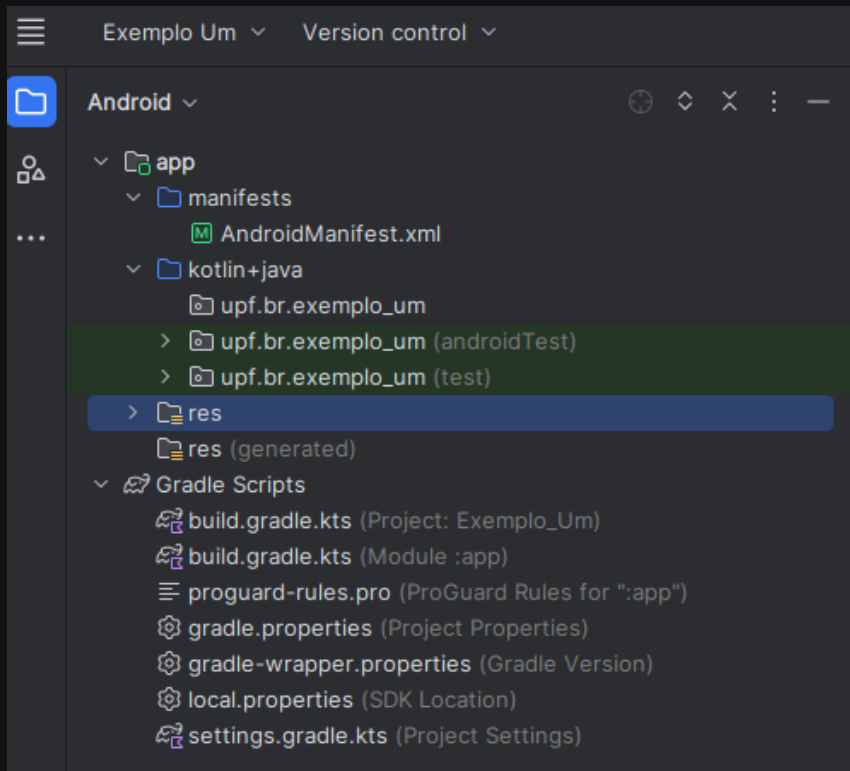


 Your app will run on approximately **98,6%** of devices.

[Help me choose](#)

☐ Use legacy android.support libraries 

Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries



`app > manifests > AndroidManifest.xml`

- O arquivo de manifesto descreve as características fundamentais do aplicativo e define cada um dos seus componentes.

`kotlin + java = source`

- Diretório onde fica a lógica do sistema, activities, packages, components...

`res`

- Diretório que contém todos os recursos em seus subdiretórios, imagens, layouts, mipmap(para ícones) e string(para textos como internacionalização)

Primeira Activity

No package `upf.br.exemplo_um` botão direito > New > JavaClass

`MainActivity.java`

```
package upf.br.exemplo_um;
```

```
public class MainActivity {  
}
```

```
package upf.br.exemplo_um;
```

```
import android.app.Activity;
```

```
public class MainActivity extends Activity{  
}
```

```
manifests > AndroidManifest.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ExemploUm"
        tools:targetApi="31" />
</manifest>
```

manifests > AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ExemploUm"
        tools:targetApi="31">
        <activity android:name=".MainActivity" android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Adicionar conteúdo

As entidades do Android (*a Activity é uma entidade*) possuem ciclos de vida(*life cycles*), isso quer dizer que ela possui estados, como `create` ou `destroy`. O Bundle é uma classe onde temos a opção de enviar informações entre as *Activities*

MainActivity.java

```
package upf.br.exemplo_um;

import android.app.Activity;
import android.os.Bundle;
import androidx.annotation.Nullable;

public class MainActivity extends Activity{
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Toast.makeText(this, "Olá", Toast.LENGTH_LONG).show();
    }
}
```

View

MainActivity.java

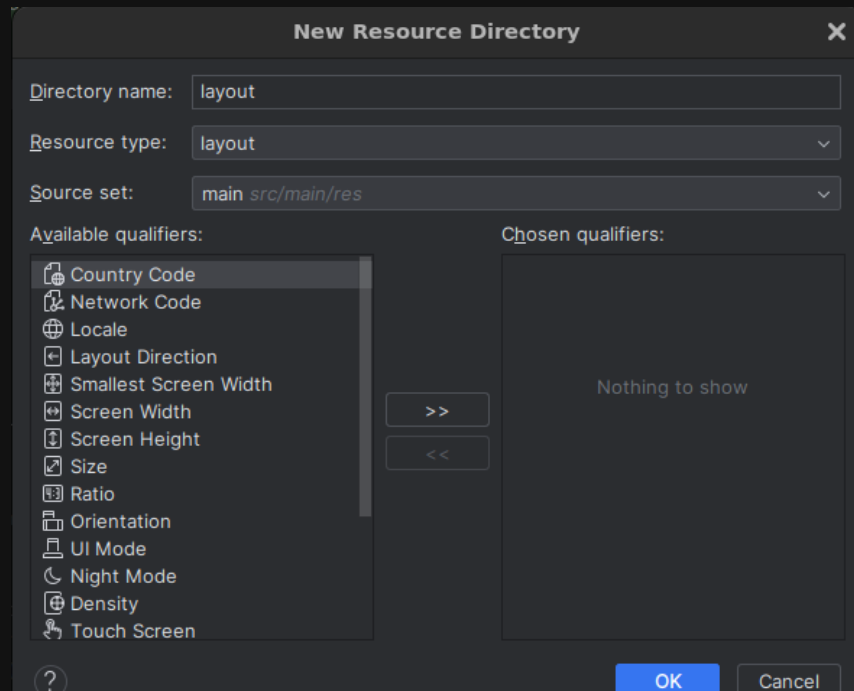
```
package upf.br.exemplo_um;

import android.app.Activity;
import android.os.Bundle;
import androidx.annotation.Nullable;
import android.widget.TextView;

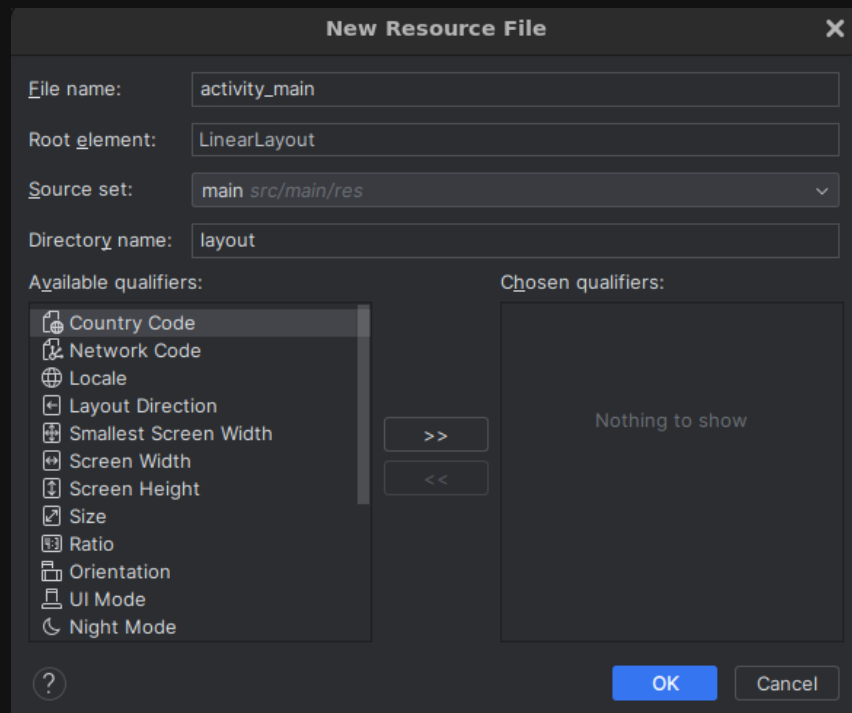
public class MainActivity extends Activity{
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView textView = new TextView(this);
        textView.setText("Olá");
        setContentView(textView);
    }
}
```

Seguindo as boas práticas, não devemos delegar views para o Main activity, com o tempo o código fica difícil de manter.

Com o botão direito em `res` > `New` > `Android Resource Directory`



Com o botão direito em `res` > `layout` > `New` > `Layout Resource File`



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk,
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Olá" />
</LinearLayout>
```

MainActivity.java

```
package upf.br.exemplo_um;

import android.app.Activity;
import android.os.Bundle;
import androidx.annotation.Nullable;

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


Strings

Uma boa prática é não armazenar texto *hard coded*, o android por default armazena as strings do sistema na pasta `res > values > string.xml` para facilitar manutenção e internacionalização.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Exemplo Um</string>
    <string name="texto">Olá</string>
</resources>
```

Para utilizar esse texto

`activity_main.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk,
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/texto" />

</LinearLayout>
```

IDE Android Studio: <https://developer.android.com/about>

IDE Android Studio: <https://developer.android.com/studio>

First app: <https://developer.android.com/training/basics/firstapp>