

Desenvolvimento Mobile

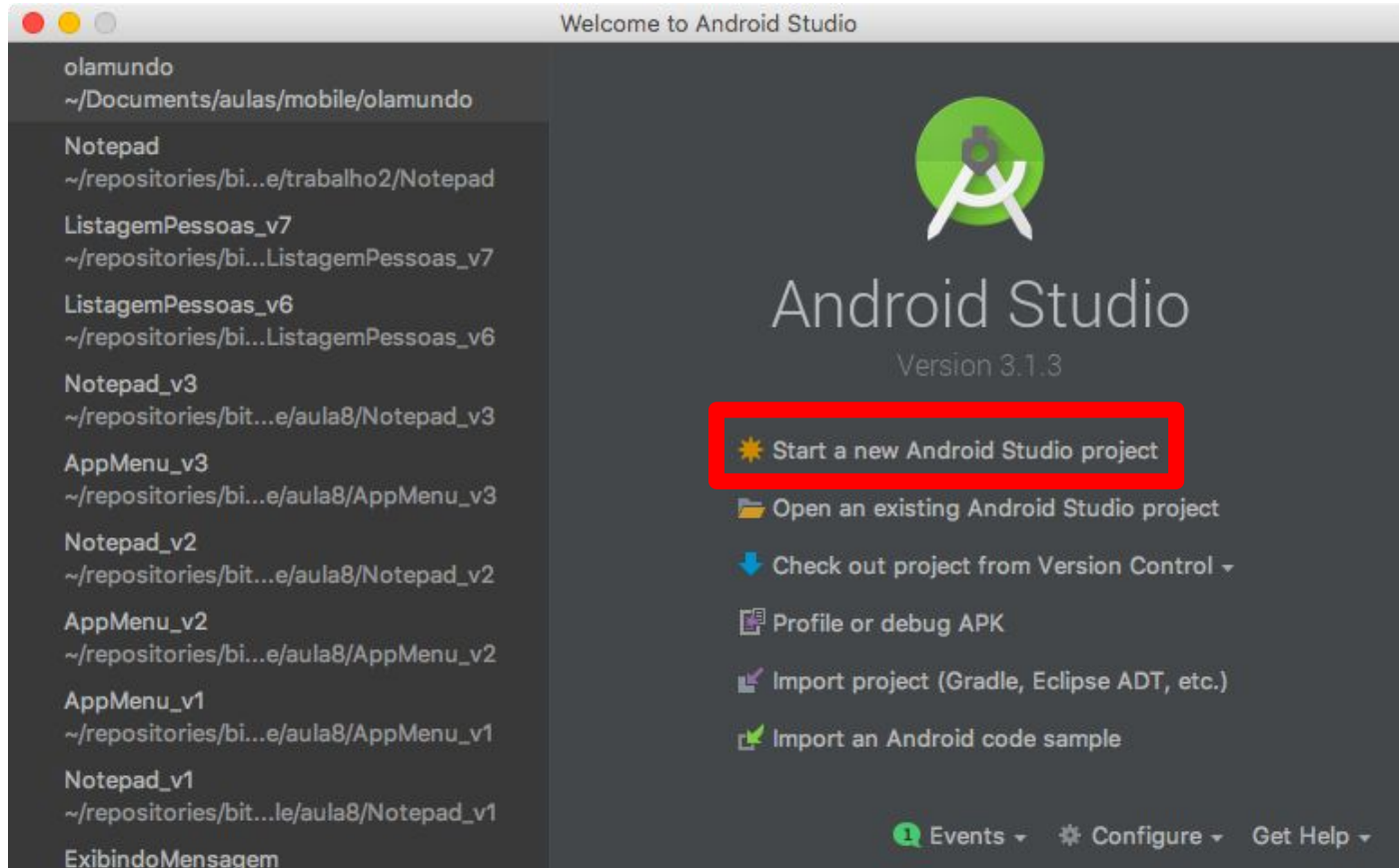
Aula 9

Abrir PDF

App para abrir arquivos PDF



Primeiros passos (Pdf4Me_v1)



Primeiros passos (Pdf4Me_v1)

Application name

Pdf4Me

Company domain

lgapontes.com

Project location

/Users/lgapontes/repositories/bitbucket/aulas/desenvolvimento-mobile/aula9/Pdf4Me_v1

...

Package name

com.lgapontes.pdf4me_v1

Done

☐ Include C++ support

☐ Include Kotlin support

Primeiros passos (Pdf4Me_v1)

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ Phone and Tablet

API 21: Android 5.0 (Lollipop)

By targeting **API 21 and later**, your app will run on approximately **71,3%** of devices. [Help me choose](#)

☐ Include Android Instant App support

☐ Wear

API 21: Android 5.0 (Lollipop)

☐ TV

API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Android Things

API 24: Android 7.0 (Nougat)

Primeiros passos (Pdf4Me_v1)

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels

☒ Phone and Tablet

API 21: Android 5.0 (Lollipop)

By targeting **API 21 and later**, your app will run

☐ Include Android Instant

☐ Wear

API 21: Android 5.0

☐ TV

API 21: Android 5.0

☐ Android Auto

☐ Android Things

API 24: Android

*Há algo que justifica a
escolha da API 21?*



Primeiros passos (Pdf4Me_v1)

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ Phone and Tablet

API 21: Android 5.0 (Lollipop)

By targeting **API 21 and later**, your app will run on approximately **71,3%** of devices. [Help me choose](#)

☐ Include Android Instant App support

☐ Wear

API 21: Android 5.0 (Lollipop)

☐ TV

API 21: Android 5.0 (Lollipop)

☐ Android Auto

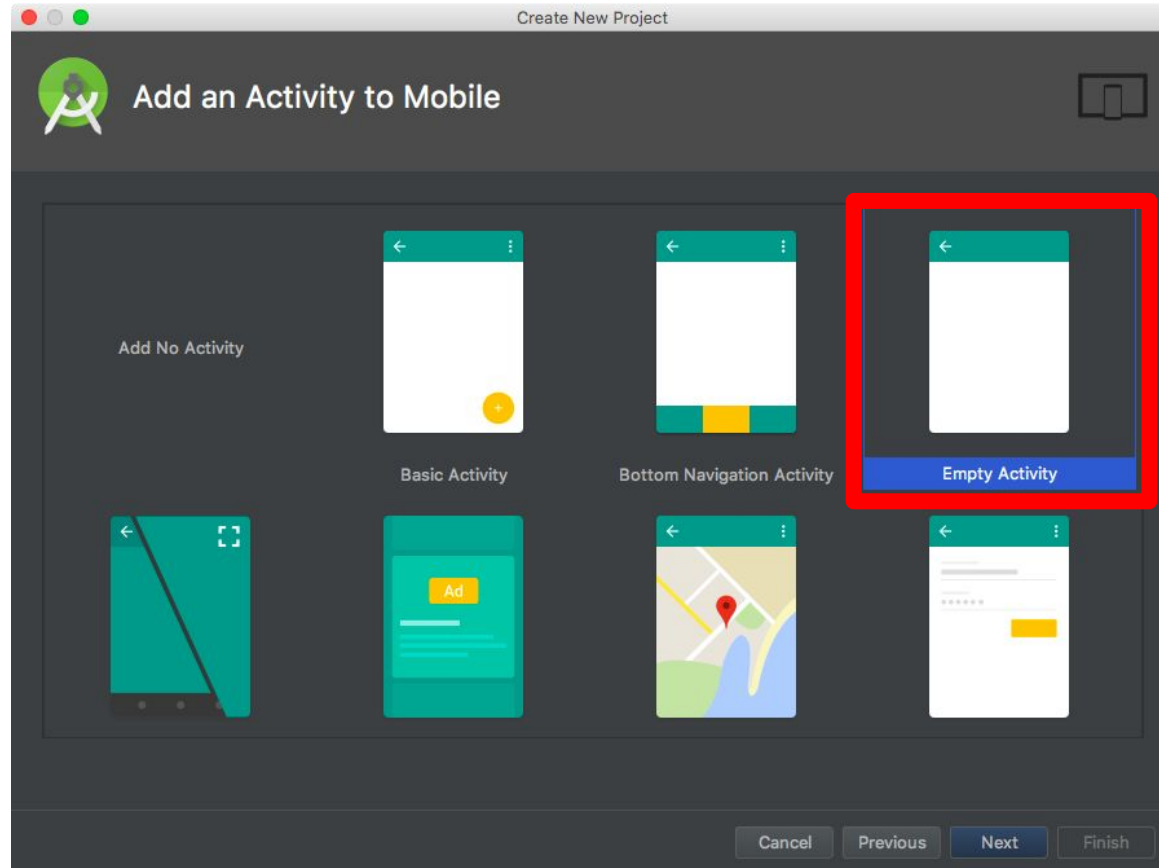
☐ Android Things

API 24: Android 7.0 (Nougat)

O componente **PdfRenderer** oficial do Android funciona a partir da API Level 21.

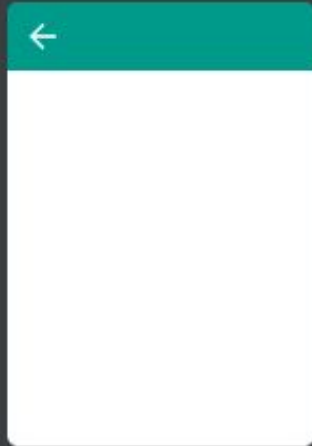
O componente **Storage Access Framework (SAF)** está disponível a partir da API Level 19.

Primeiros passos (Pdf4Me_v1)



Primeiros passos (Pdf4Me_v1)

Creates a new empty activity



Activity Name:

MainActivity

☒ Generate Layout File

Layout Name:

activity_main

☒ Backwards Compatibility (AppCompat)

Exercício em Sala

Altere os seguintes detalhes na aplicação Pdf4Me:

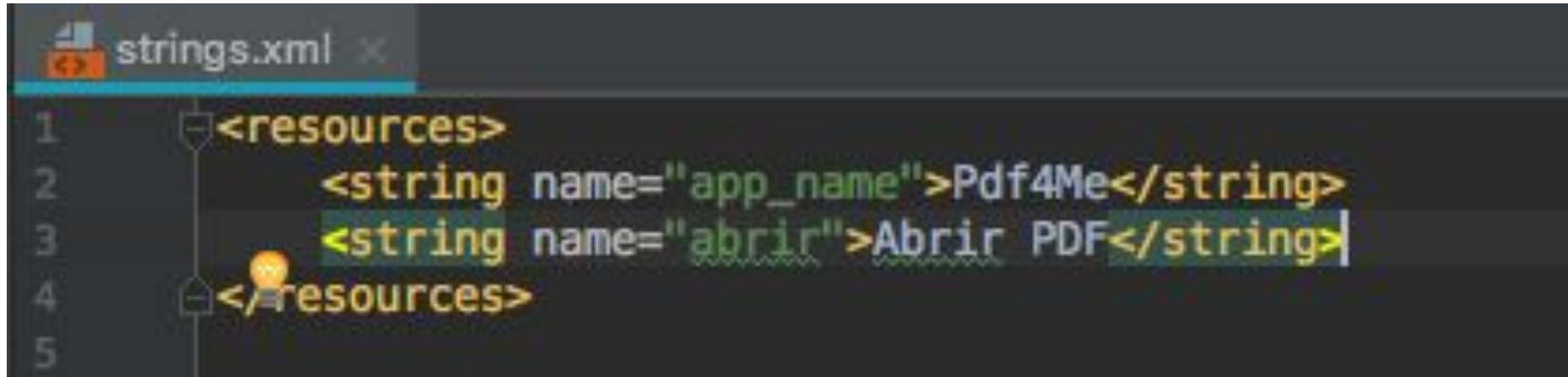
- **Cores do projeto:**
 - `colorPrimary:` #754646
 - `colorPrimaryDark:` #533535
 - `colorAccent:` #ee5541
 - `branco:` #ffffff
- **Configure o Launcher Icon com as imagens disponíveis no EAD**
- **Acrescente o item "Abrir PDF" na ActionBar com a imagem *abrir.svg***
- **Crie 2 FABs localizados na parte inferior da tela. Eles devem conter as imagens *pagina_anterior.svg* e *pagina_posterior.svg***

Resolvendo o Exercício



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <color name="colorPrimary">#754646</color>
4     <color name="colorPrimaryDark">#533535</color>
5     <color name="colorAccent">#ee5541</color>
6     <color name="branco">#ffffff</color>
7 </resources>
```

Resolvendo o Exercício



```
1 <resources>
2     <string name="app_name">Pdf4Me</string>
3     <string name="abrir">Abrir PDF</string>
4 </resources>
5
```

Resolvendo o Exercício

Inclua os arquivos:

- abrir.svg
- pagina_anterior.svg
- pagina_posterior.svg



`abrir.svg`

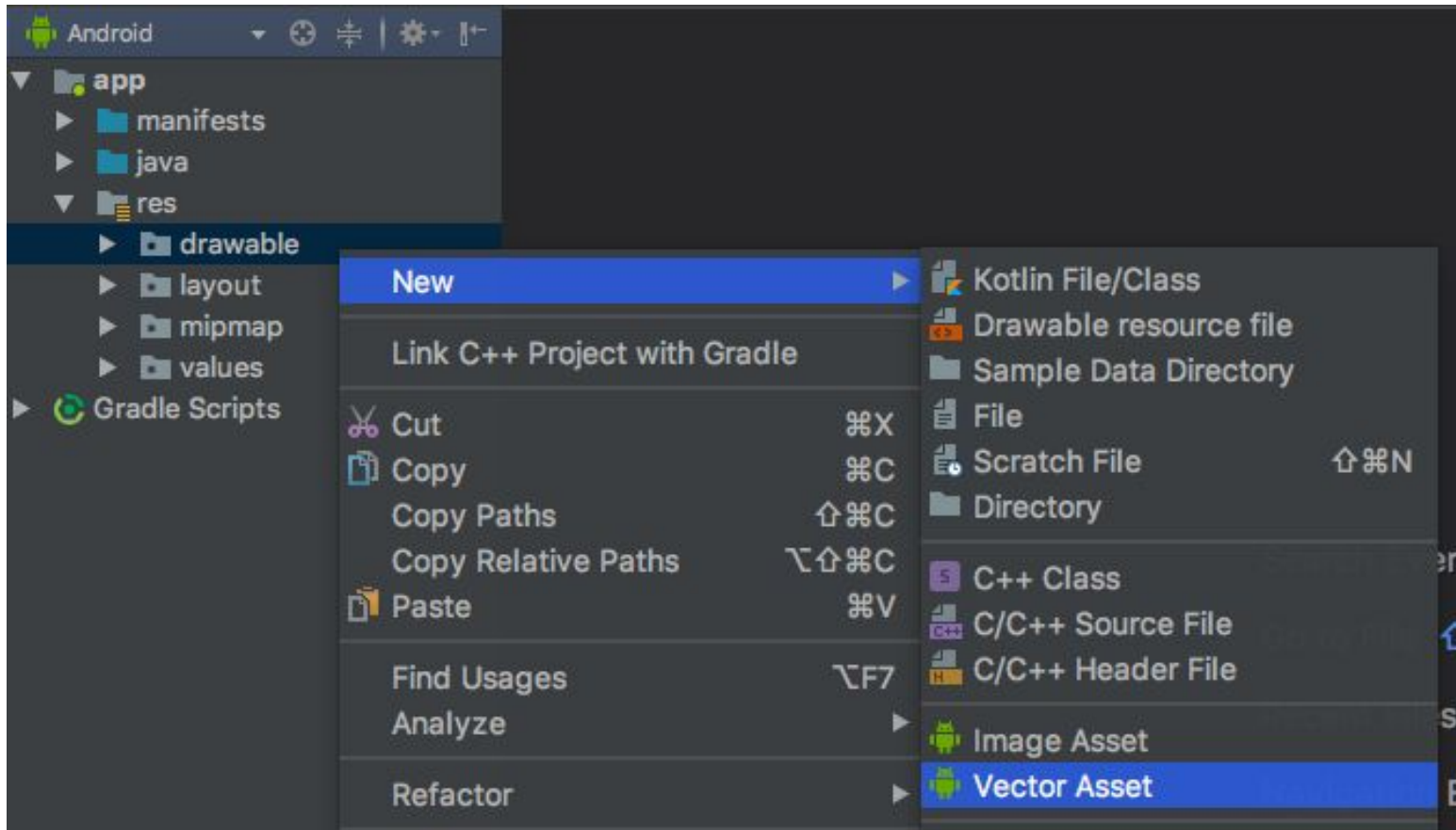


`pagina_anterior.svg`

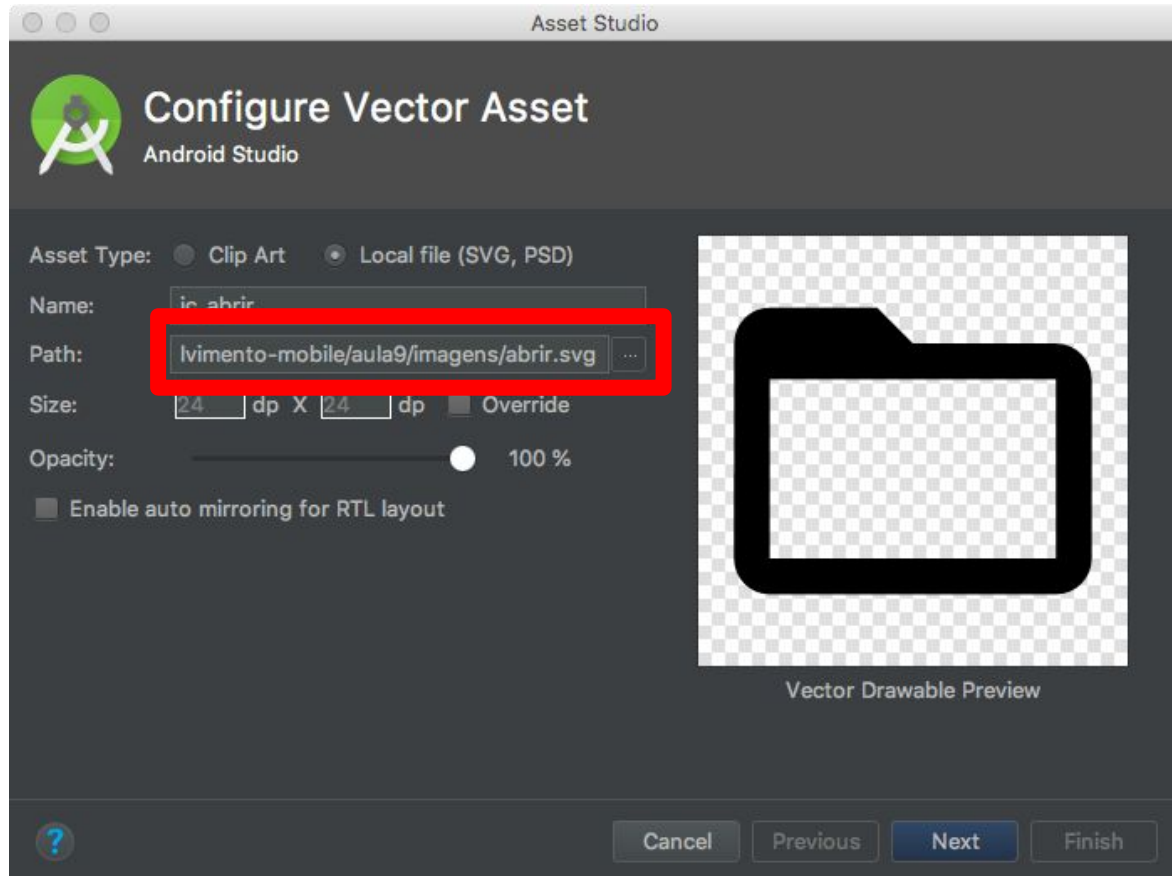


`pagina_posterior.svg`

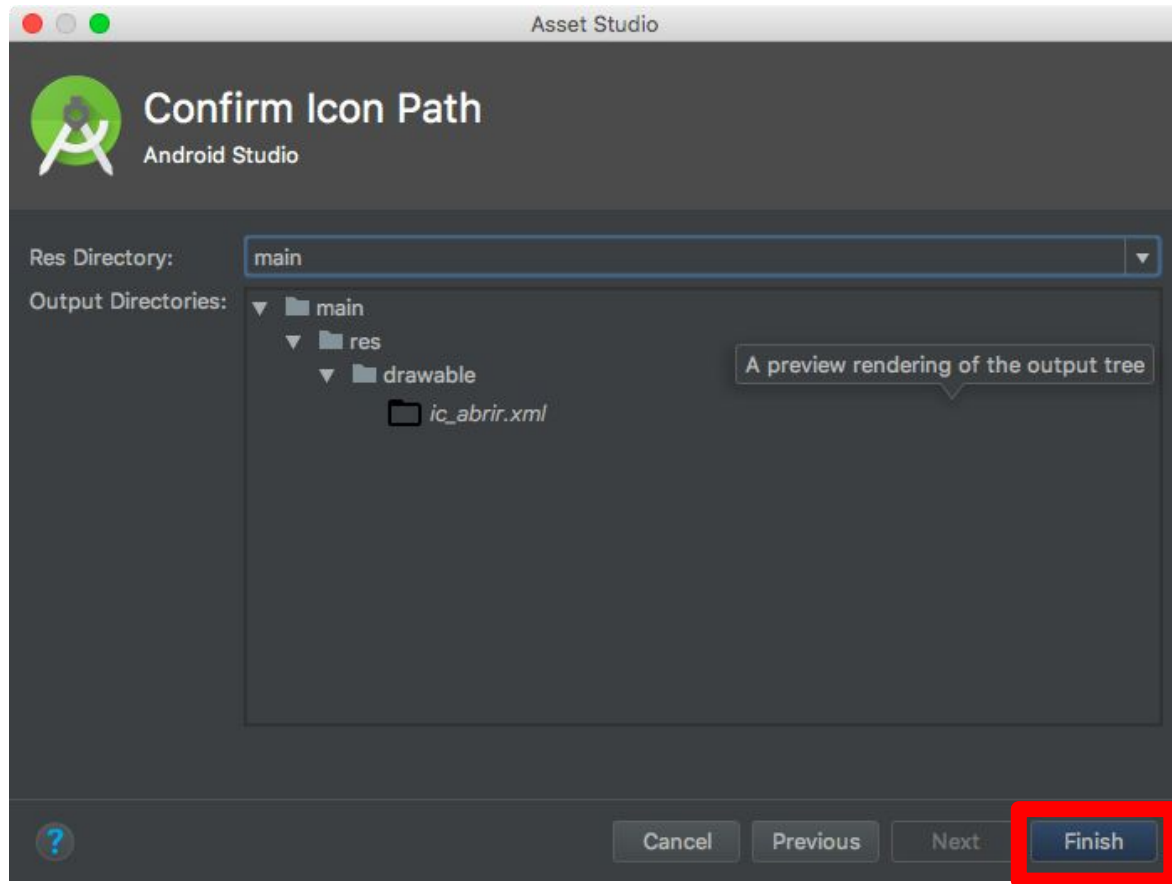
Resolvendo o Exercício



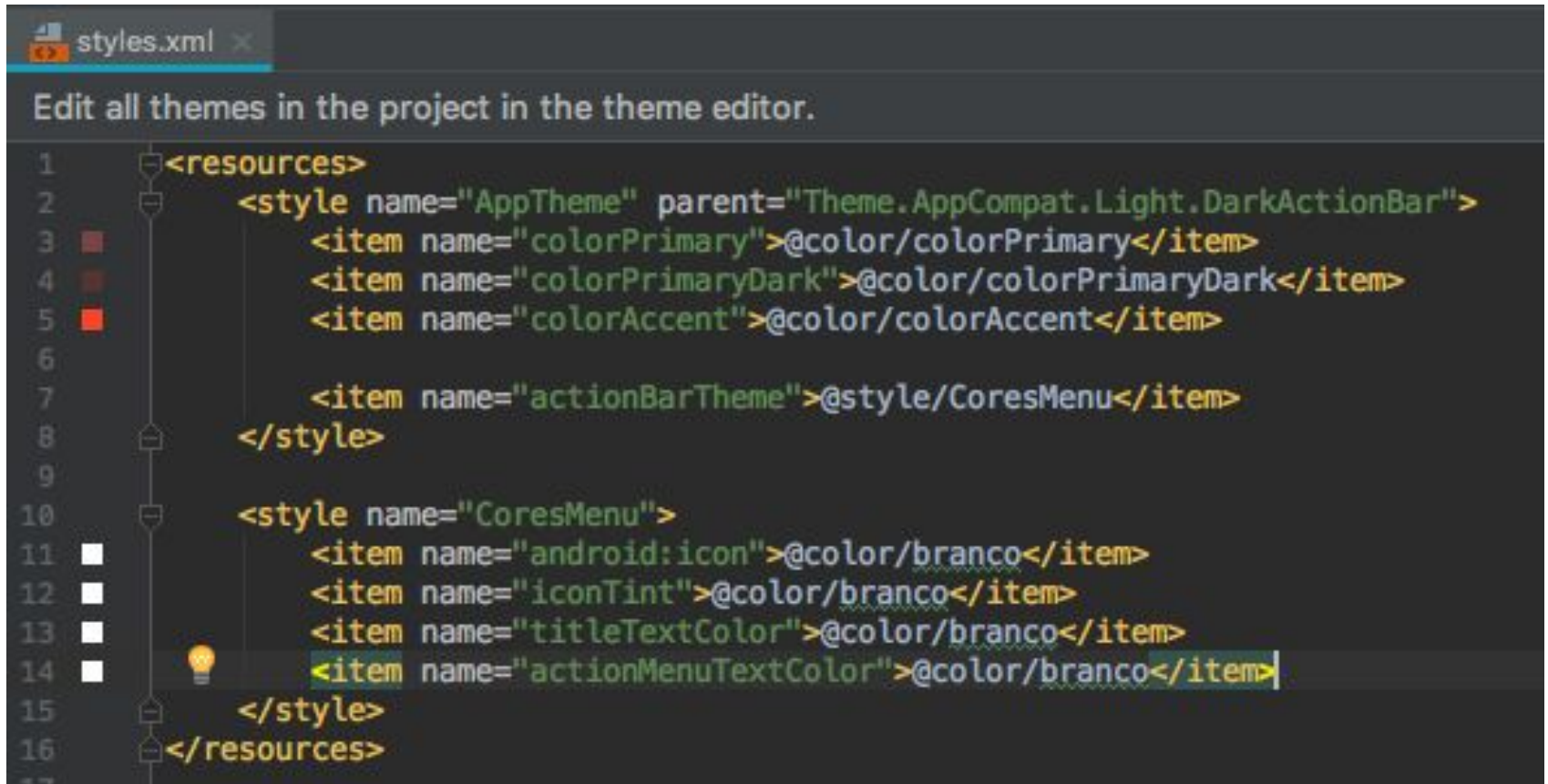
Resolvendo o Exercício



Resolvendo o Exercício

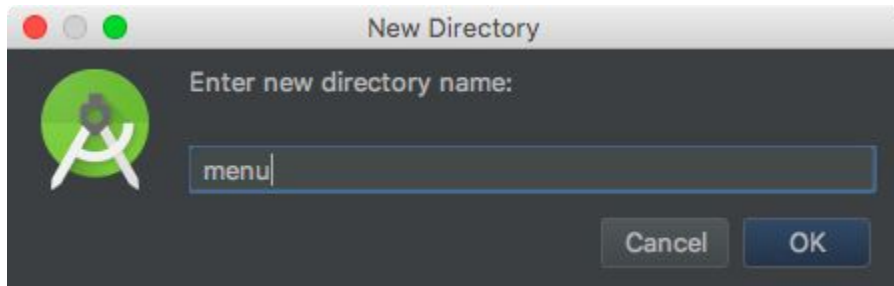
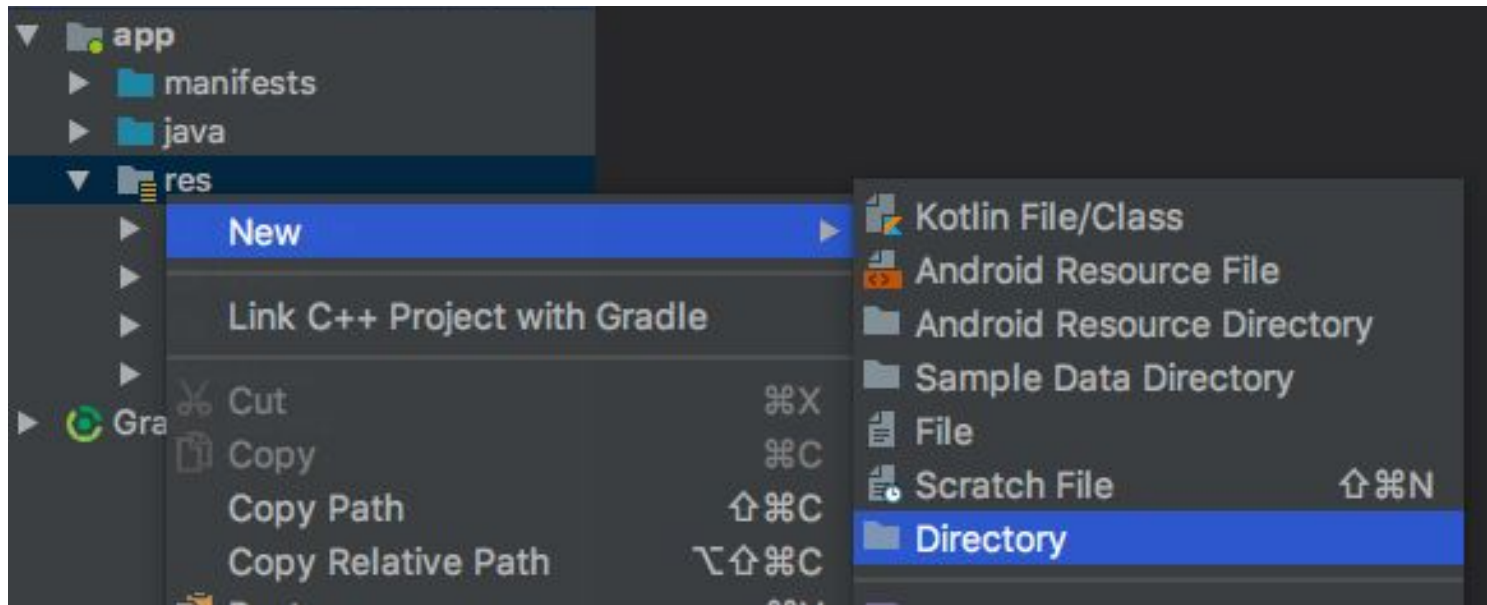


Resolvendo o Exercício

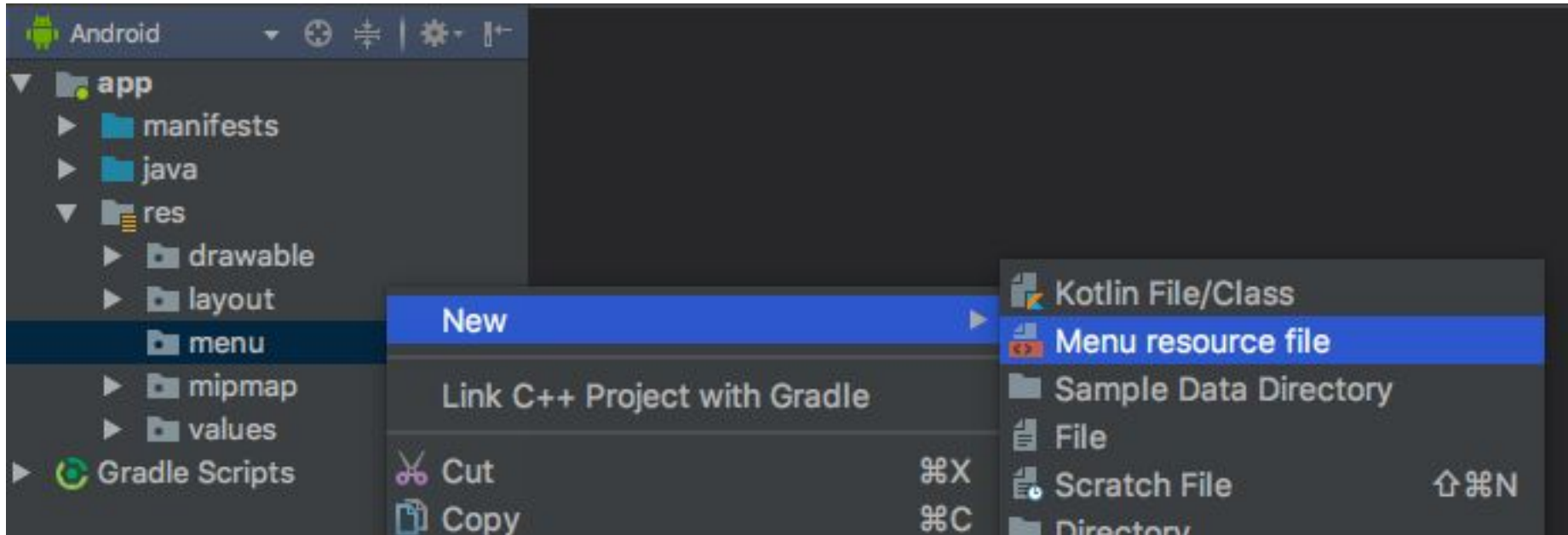


```
1 <resources>
2   <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
3     <item name="colorPrimary">@color/colorPrimary</item>
4     <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
5     <item name="colorAccent">@color/colorAccent</item>
6
7     <item name="actionBarTheme">@style/CoresMenu</item>
8   </style>
9
10  <style name="CoresMenu">
11    <item name="android:icon">@color/branco</item>
12    <item name="iconTint">@color/branco</item>
13    <item name="titleTextColor">@color/branco</item>
14    <item name="actionMenuTextColor">@color/branco</item>
15  </style>
16 </resources>
```

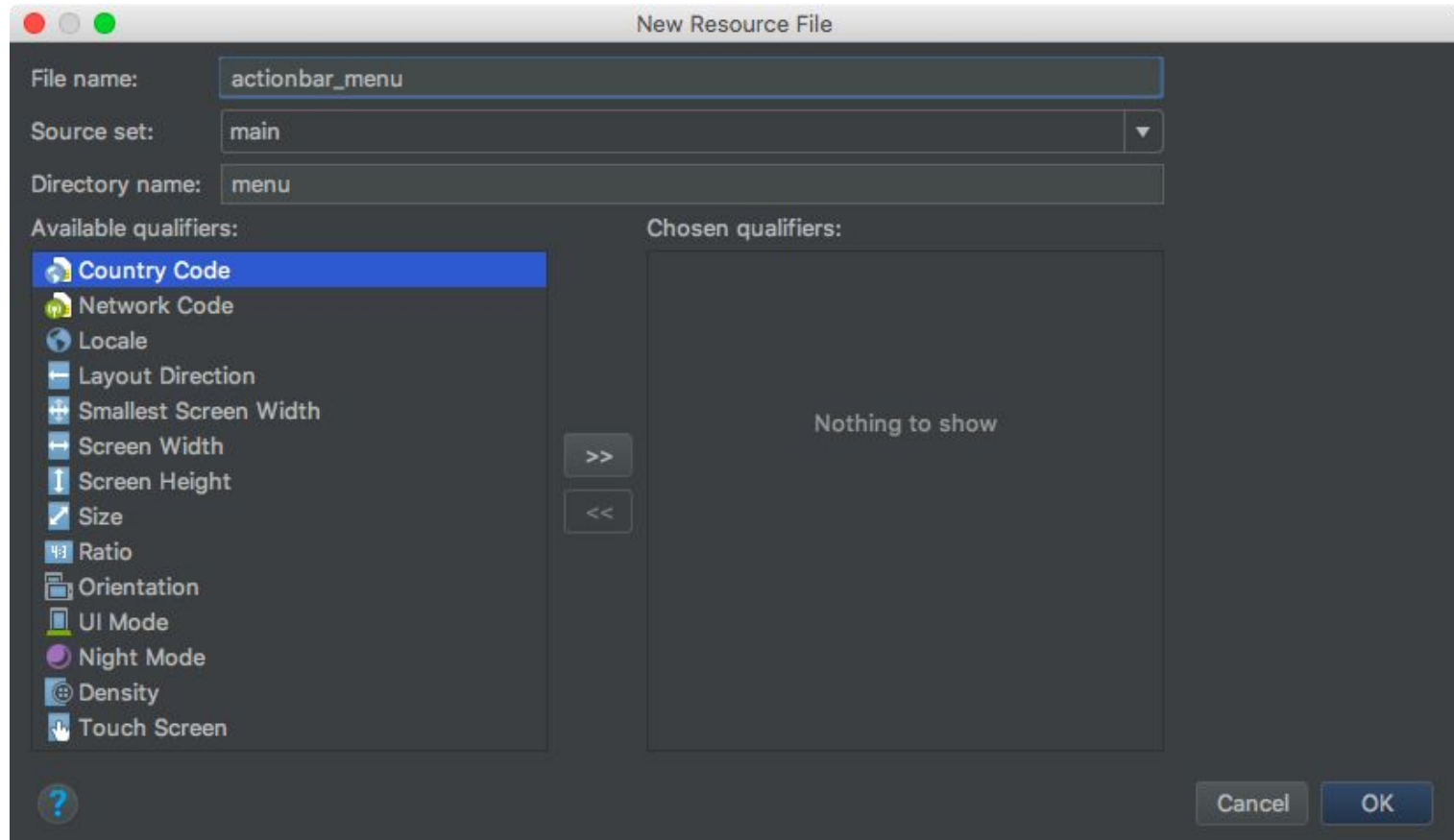
Resolvendo o Exercício



Resolvendo o Exercício



Resolvendo o Exercício

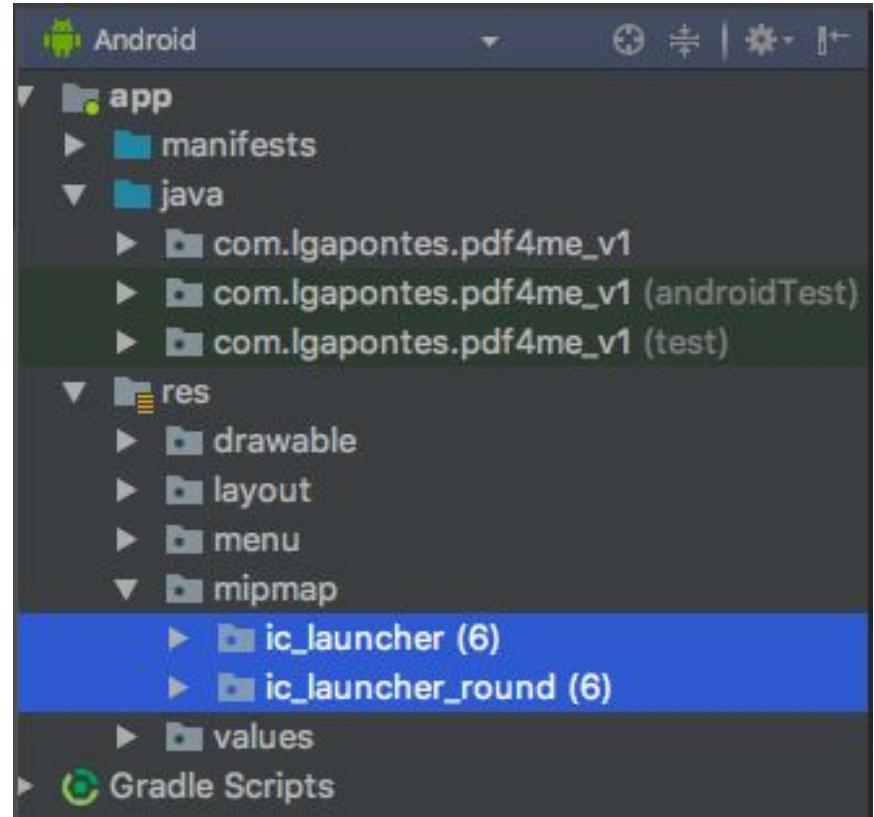
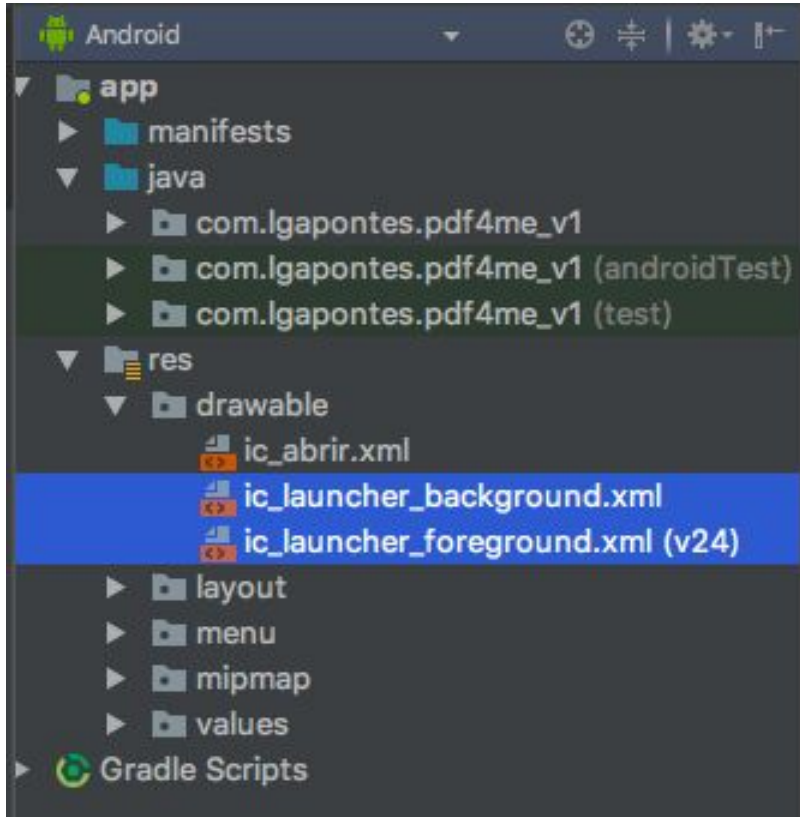


Resolvendo o Exercício

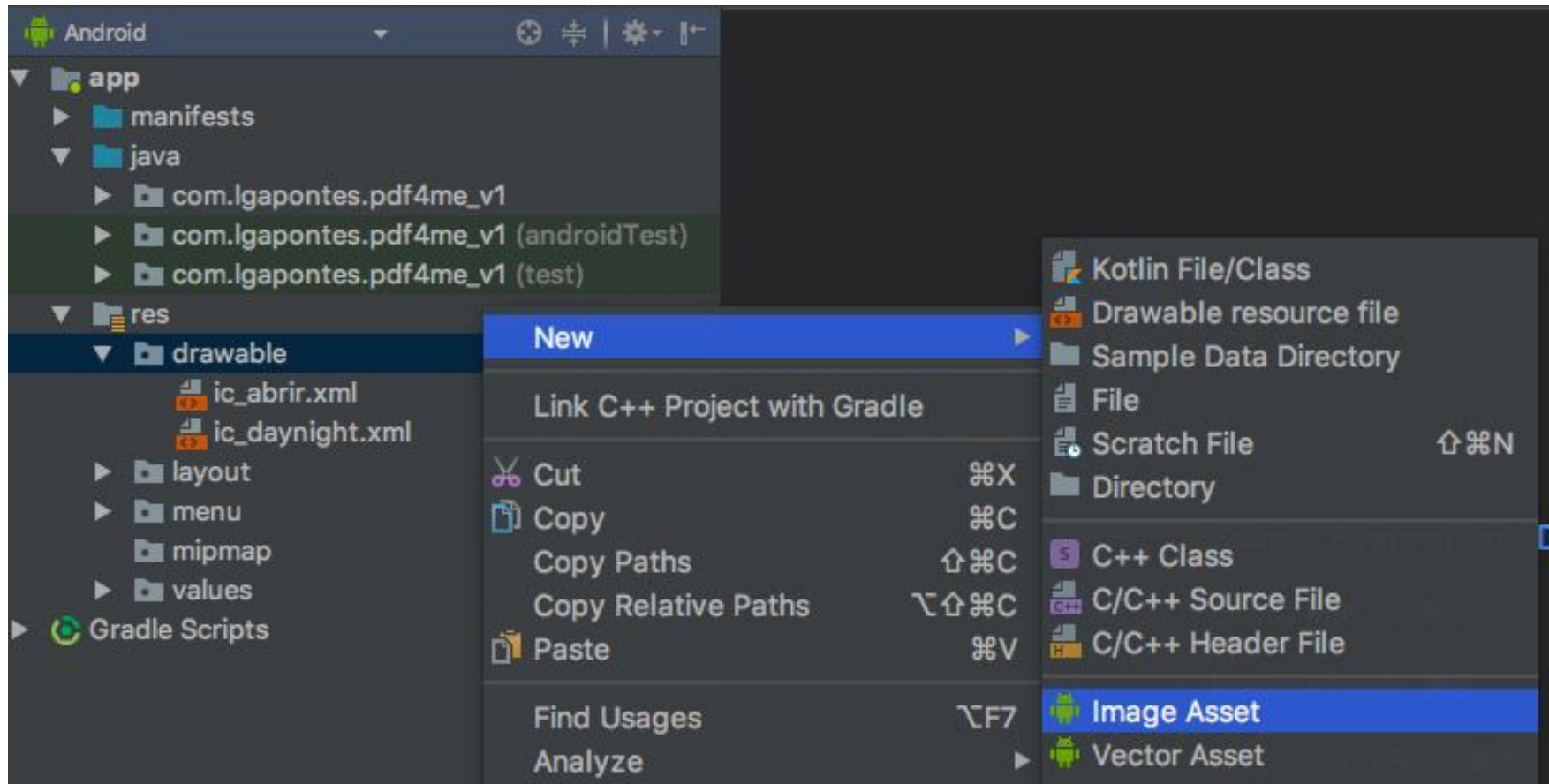
```
actionbar_menu.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto">
4
5      <item
6          android:id="@+id/menu_abrir"
7          android:icon="@drawable/ic_abrir"
8          android:title="@string/abrir"
9          app:showAsAction="always" />
10
11  </menu>
```

Resolvendo o Exercício

Apague os arquivos indicados a seguir.



Resolvendo o Exercício





Configure Image Asset

Android Studio

Icon Type: Launcher Icons (Adaptive and Legacy) ▼

Preview

xxxhdpi ▼

☒ Show Safe Zone☐ Show Grid

Name: ic_launcher

Foreground Layer

Background Layer

Legacy

Layer Name: ic_launcher_foreground

Source Asset

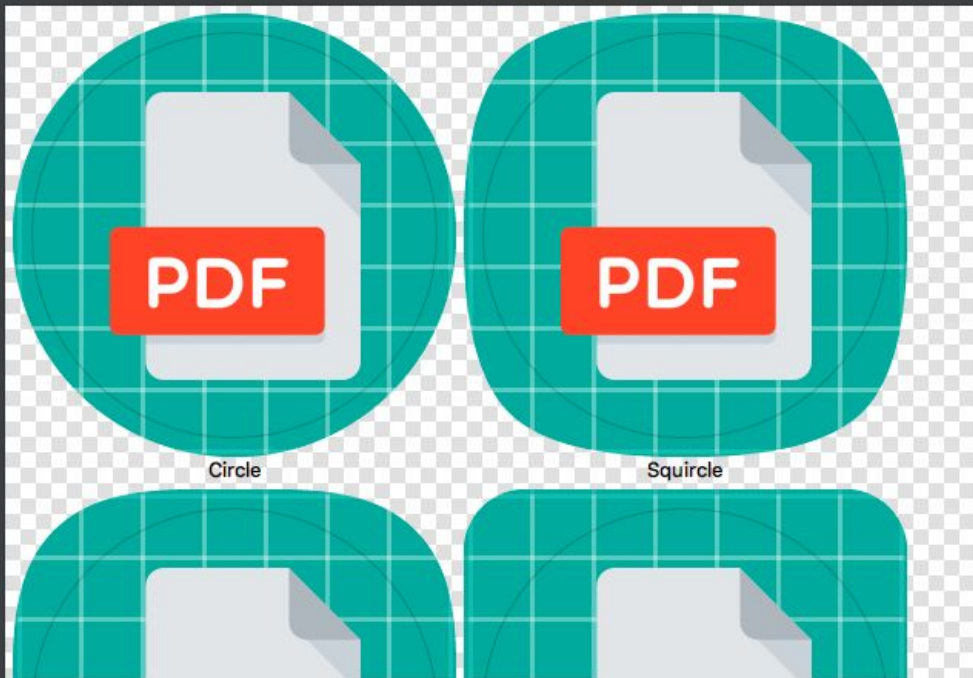
Asset Type: ☒ Image ☐ Clip Art ☐ Text

Path: nagens/launcher_foreground.png ...

Scaling

Trim: ☒ Yes ☐ No

Resize: 71 %



Circle

Squirele



Cancel

Previous

Next

Finish



Configure Image Asset

Android Studio

Icon Type: Launcher Icons (Adaptive and Legacy) ▼

Preview

xxxhdpi ▼

☒ Show Safe Zone☐ Show Grid

Name: ic_launcher

Foreground Layer Background Layer Legacy

Layer Name: ic_launcher_background

Source Asset

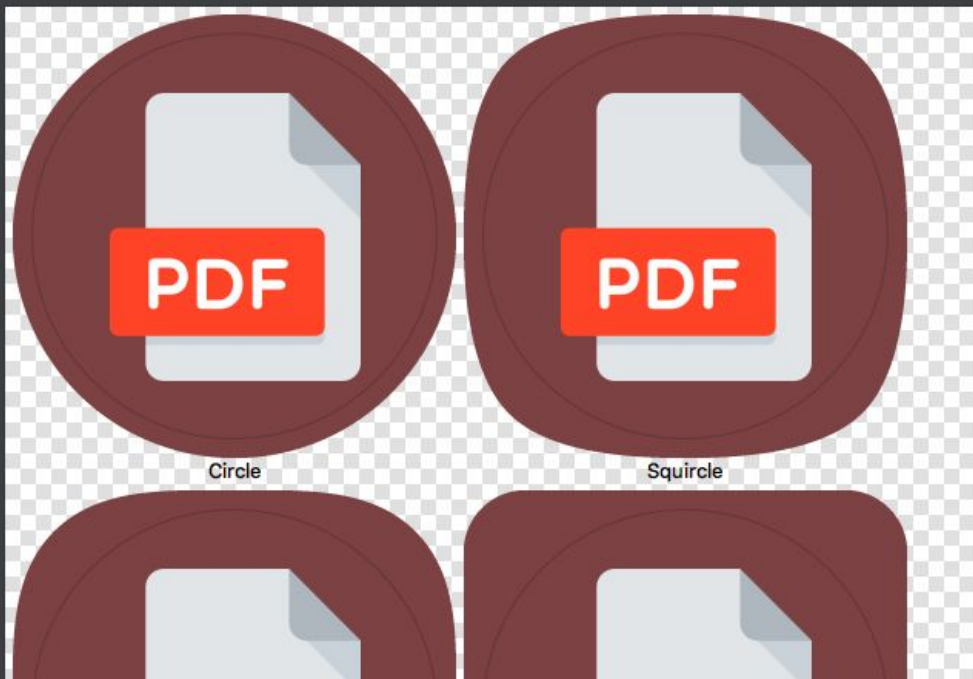
Asset Type: ☐ Color ☒ Image

Path: assets/launcher_background.png ...

Scaling

Trim: ☐ Yes ☒ No

Resize: 100 %



Cancel

Previous

Next

Finish



Configure Image Asset

Android Studio

Icon Type: Launcher Icons (Adaptive and Legacy) ▼

Preview

xxxhdpi ▼

☒ Show Safe Zone☐ Show Grid

Name: ic_launcher

Foreground Layer

Background Layer

Legacy

Legacy Icon (API ≤ 25)

Generate: ☒ Yes ☐ No

Shape: Square ▼

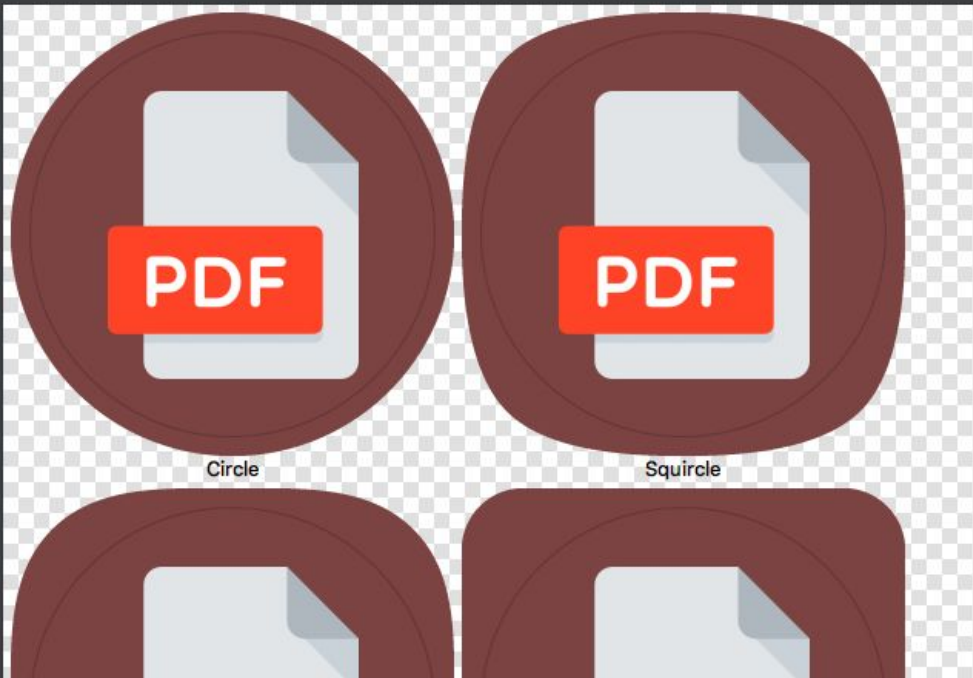
Round Icon (API = 25)

Generate: ☒ Yes ☐ No

Google Play Store Icon

Generate: ☒ Yes ☐ No

Shape: Square ▼



Circle

Squirele



Cancel

Previous

Next

Finish



Confirm Icon Path

Android Studio

Res Directory:

main

Output Directories:

- ▼ main
 - ▼ res
 - ▼ mipmap-anydpi-v26
 - ic_launcher.xml
 - ic_launcher_round.xml
 - ▼ mipmap-mdpi
 - ic_launcher.png
 - ic_launcher_background.png
 - ic_launcher_foreground.png
 - ic_launcher_round.png
 - ▼ mipmap-hdpi
 - ic_launcher.png
 - ic_launcher_background.png
 - ic_launcher_foreground.png
 - ic_launcher_round.png
 - ▼ mipmap-xhdpi
 - ic_launcher.png
 - ic_launcher_background.png
 - ic_launcher_foreground.png
 - ic_launcher_round.png
 - ▼ mipmap-xxhdpi
 - ic_launcher.png

Output File

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
  <background android:drawable="@mipmap/ic_launcher_background"/>
  <foreground android:drawable="@mipmap/ic_launcher_foreground"/>
</adaptive-icon>
```



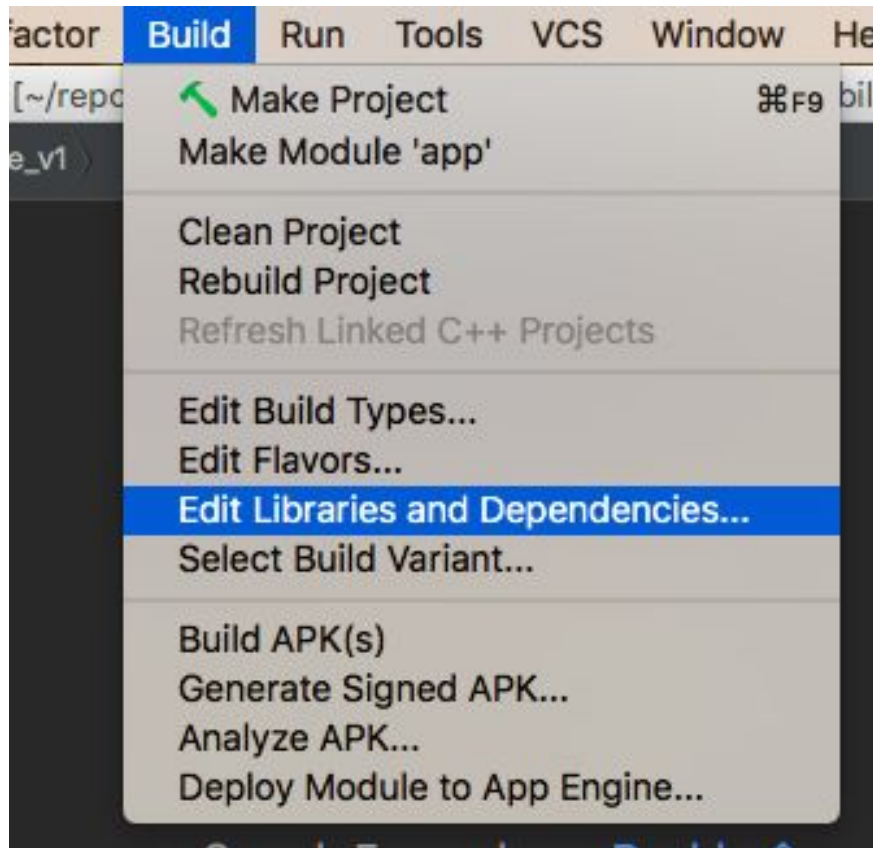
Cancel

Previous

Next

Finish

Resolvendo o Exercício



Vamos adicionar a **Design Support Library** para criar os FABs necessários.

Project Structure



Properties Signing Flavors Build Types Dependencies

SDK Location

Project

Developer Ser...






Ads




Authentication

Notifications

Modules

app

	Scope
{include=[*.jar], dir=libs}	Implementation ▾
 com.android.support:appcompat-v7:28.0.0-rc01	Implementation ▾
 com.android.support.constraint:constraint-layout:1.1.2	Implementation ▾
 junit:junit:4.12	Unit Test imple... ▾
 com.android.support.test:runner:1.0.2	Test implement... ▾
 com.android.support.test.espresso:espresso-core:3.0.2	Test implement... ▾

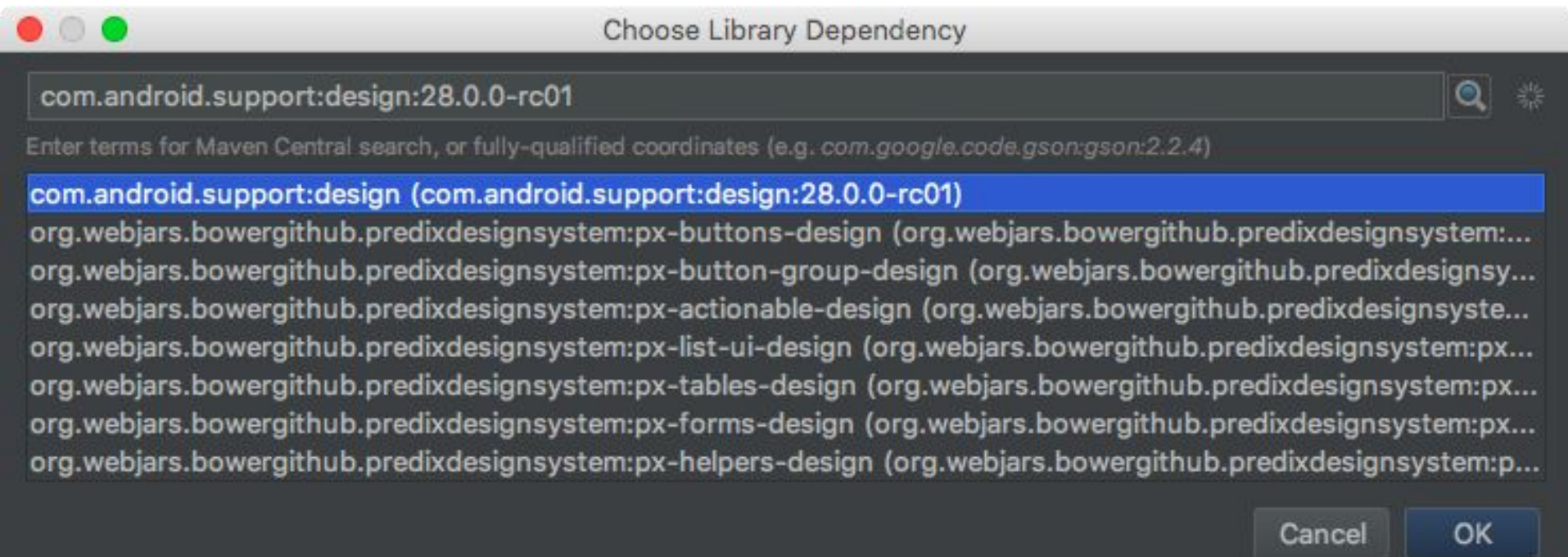
 1 Library dependency 2 Jar dependency 3 Module dependency

Cancel

OK

Resolvendo o Exercício

Procure por *design* e selecione *com.android.support:design*

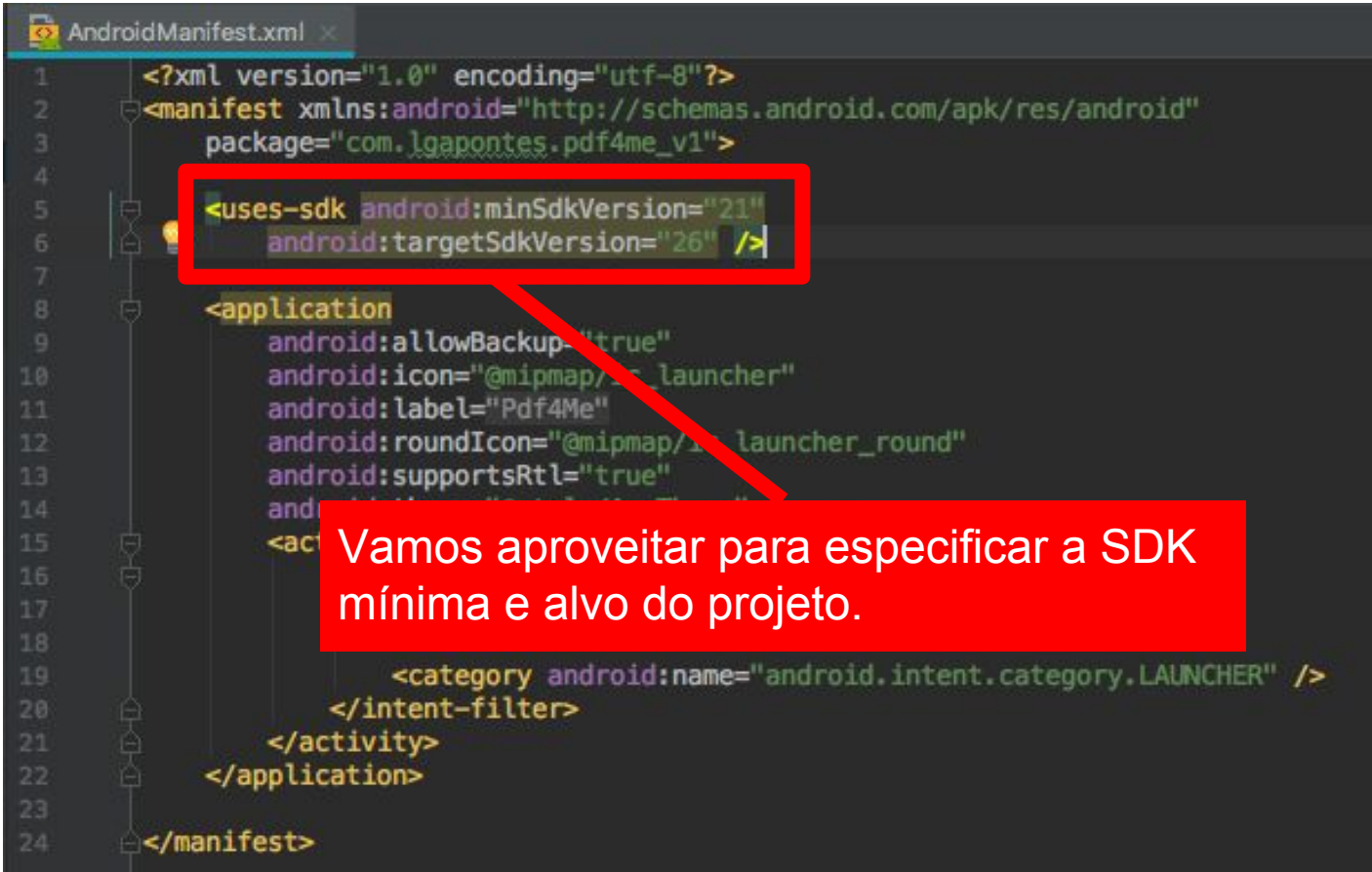


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent">
6
7     <android.support.design.widget.FloatingActionButton
8         android:id="@+id/fab_pagina_anterior"
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:layout_alignParentLeft="true"
12        android:layout_alignParentBottom="true"
13        android:layout_margin="16px"
14        android:src="@drawable/ic_pagina_anterior"
15        android:tint="@color/branco" />
16
17    <android.support.design.widget.FloatingActionButton
18        android:id="@+id/fab_pagina_posterior"
19        android:layout_width="wrap_content"
20        android:layout_height="wrap_content"
21        android:layout_alignParentRight="true"
22        android:layout_alignParentBottom="true"
23        android:layout_margin="16px"
24        android:src="@drawable/ic_pagina_posterior"
25        android:tint="@color/branco" />
26
27 </RelativeLayout>
```



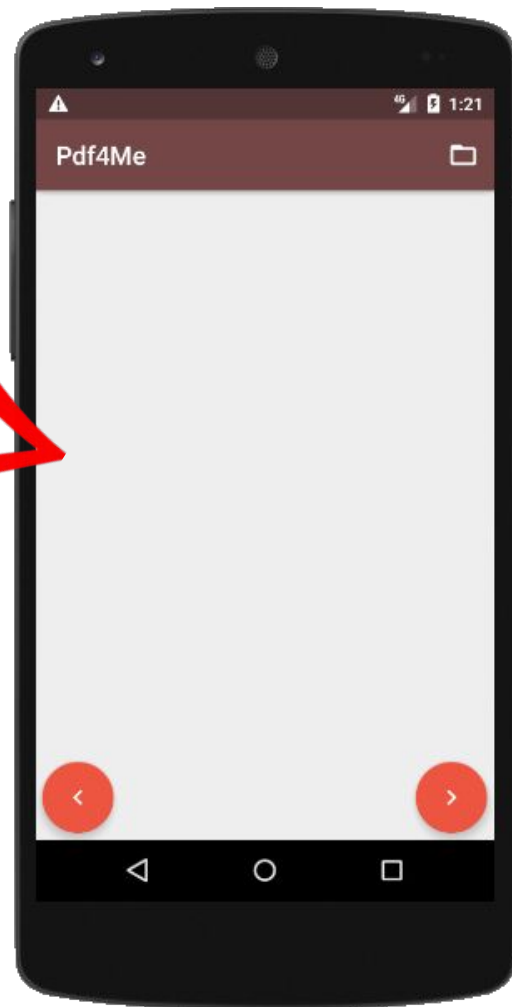
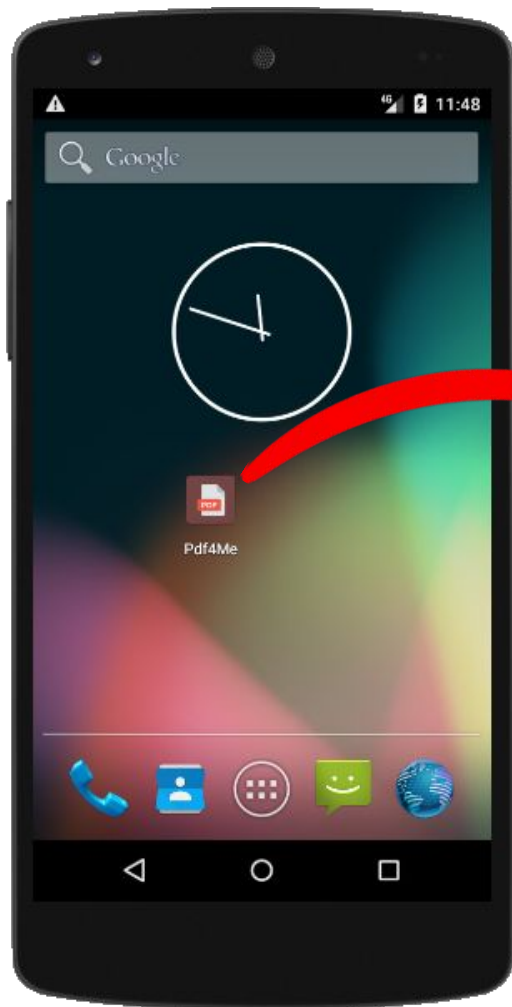
```
1  package com.lgapontes.pdf4me_v1;
2
3  import ...
21
22  public class MainActivity extends AppCompatActivity {
23
24      private FloatingActionButton fabPaginaAnterior;
25      private FloatingActionButton fabPaginaPosterior;
26
27      @Override
28      protected void onCreate(Bundle savedInstanceState) {
29          super.onCreate(savedInstanceState);
30          setContentView(R.layout.activity_main);
31
32          fabPaginaAnterior = (FloatingActionButton) findViewById(R.id.fab_pagina_anterior);
33          fabPaginaPosterior = (FloatingActionButton) findViewById(R.id.fab_pagina_posterior);
34      }
35
36      @Override
37      public boolean onCreateOptionsMenu(Menu menu) {
38          MenuInflater inflater = getMenuInflater();
39          inflater.inflate(R.menu.actionbar_menu, menu);
40          return super.onCreateOptionsMenu(menu);
41      }
42
43  }
```

Resolvendo o Exercício



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.lgapontes.pdf4me_v1">
4
5     <uses-sdk android:minSdkVersion="21"
6         android:targetSdkVersion="26" />
7
8     <application
9         android:allowBackup="true"
10        android:icon="@mipmap/ic_launcher"
11        android:label="Pdf4Me"
12        android:roundIcon="@mipmap/ic_launcher_round"
13        android:supportsRtl="true"
14        android:theme="@style/AppTheme">
15
16        <activity
17            android:name=".MainActivity"
18            android:label="@string/app_name"
19            android:theme="@style/AppTheme.NoActionBar">
20            <intent-filter>
21                <action android:name="android.intent.action.MAIN" />
22                <category android:name="android.intent.category.LAUNCHER" />
23            </intent-filter>
24        </activity>
25    </application>
26</manifest>
```

Vamos aproveitar para especificar a SDK mínima e alvo do projeto.



Trabalhando com Intents explícitas

Já trabalhamos com *Intents explícitas*. Através dela o Android já sabe exatamente qual *Activity* será aberta.



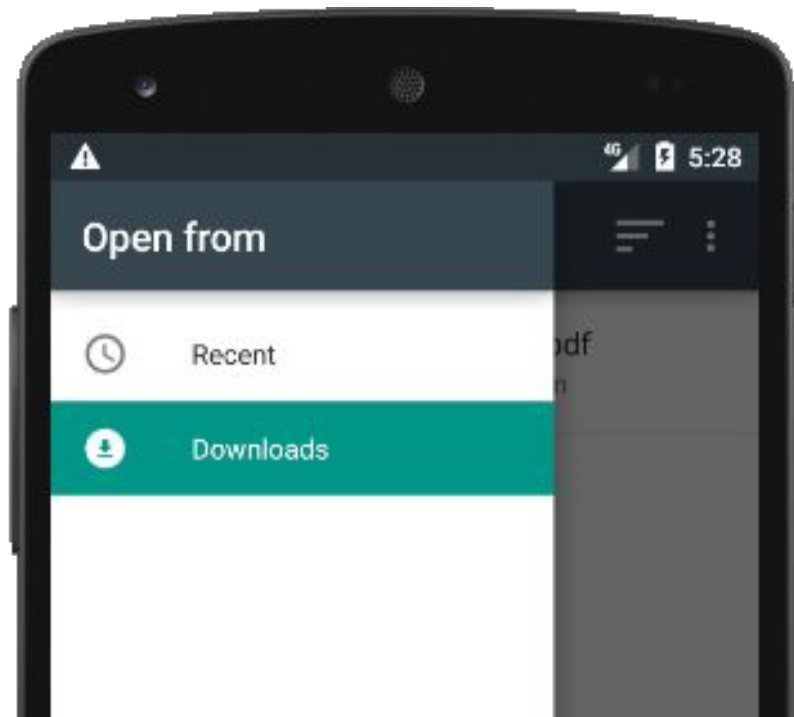
Trabalhando com **Intents implícitas**

A ideia das Intents implícitas é a mesma das explícitas. A diferença é que o Android recebe uma "dica" sobre qual *Activity* deve ser aberta. Quando há várias opções disponíveis, ele pergunta ao próprio usuário.



Trabalhando com **Framework Access Storage**

Neste exemplo, vamos informar ao Android que queremos abrir uma ferramenta para localizar arquivos no dispositivo. A partir do Android 4.4 (API Level 19) temos disponível o **Framework Access Storage**, que é a melhor opção para abrir arquivos a partir de todos os provedores de armazenamento.

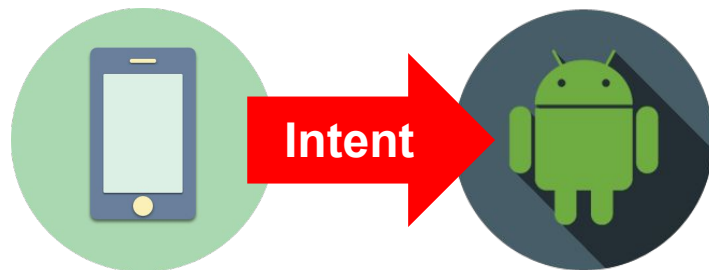


Intents implícitas & Framework Access Storage

<https://developer.android.com/guide/components/intents-filters?hl=pt-br>

Ao criar uma Intent implícita, devemos indicar detalhes sobre a Activity que precisamos.

Ao criar um intent implícito, o sistema Android encontra o componente adequado para iniciar, comparando o conteúdo do intent aos *filtros de intents* declarados no [arquivo de manifesto](#) de outros aplicativos no dispositivo. Se o intent corresponder a um filtro de intents, o sistema iniciará esse componente e entregará o objeto **Intent**. Se diversos filtros de intents corresponderem, o sistema exibirá uma caixa de diálogo para que o usuário selecione o aplicativo que deseja usar.



Nome do
Componente

Ação

Dados

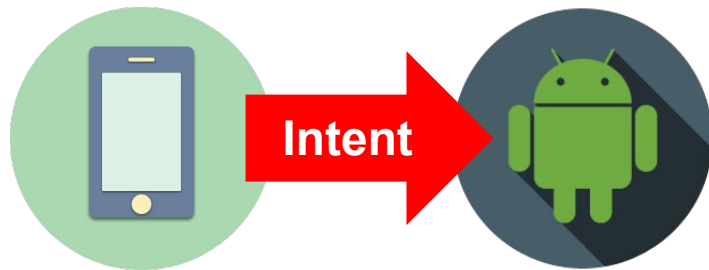
Categoria

Extras

Sinalizadores

Intents implícitas & Framework Access Storage

<https://developer.android.com/guide/components/intents-filters?hl=pt-br>



Para casos de Intents explícitas

Nome do Componente

Ação

Ação que o Android deve realizar

Dados ou o tipo dos dados

Dados

Categoria

Informações adicionais sobre o tipo de App

Pares de chave-valor que serão enviadas

Extras

Sinalizadores

Informar ao Android detalhes da Intent

Intents implícitas & Framework Access Storage

Veja mais detalhes de criação de Intents em:

<https://developer.android.com/guide/components/intents-filters?hl=pt-br>

Veja a lista de ações e categorias em:

<https://developer.android.com/reference/android/content/Intent?hl=pt-br>

Categoria: indica que nossa App quer apenas a URI do arquivo.

Ação: indica que queremos abrir um arquivo. Na prática vai abrir o *Framework Access Storage*.

```
Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);  
intent.addCategory(Intent.CATEGORY_OPENABLE);  
intent.setType("application/pdf");  
startActivityForResult(intent, requestCode: 1);
```

Dados: neste caso, está informando que o *Framework Access Storage* vai abrir apenas arquivos PDF.

Intents implícitas & Framework Access Storage

```
Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);  
intent.addCategory(Intent.CATEGORY_OPENABLE);  
intent.setType("application/pdf");  
startActivityForResult(intent, requestCode: 1);
```

Note que para abrir essa Intent convocamos o método *startActivityForResult()*. Isso se faz necessário porque esperamos um resultado (a URI do arquivo).

Intents implícitas & Framework Access Storage

Ok, mas para que serve
esse *requestCode*?

```
Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);  
intent.addCategory(Intent.CATEGORY_OPENABLE);  
intent.setData(Uri.parse("file://"));  
startActivityForResult(intent, requestCode);
```

Note que usamos o método `startActivityForResult()`. Isso se faz necessário para obter o resultado (o conteúdo do arquivo).



Intents implícitas & Framework Access Storage

```
Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);  
intent.addCategory(Intent.CATEGORY_OPENABLE);  
intent.setType("application/pdf");  
startActivityForResult(intent, requestCode 1);
```

```
public void onActivityResult(int requestCode, int resultCode, Intent resultData) {  
}
```

Devemos implementar o método `onActivityResult()` para receber o retorno das Intents. Porém, como podemos invocar várias Intents dentro de nossa aplicação, o Android permite definirmos um código para identificar qual delas retornou. Além disso, temos também os parâmetros:

`resultCode`: resultado. A constante `Activity.RESULT_OK` indica sucesso.

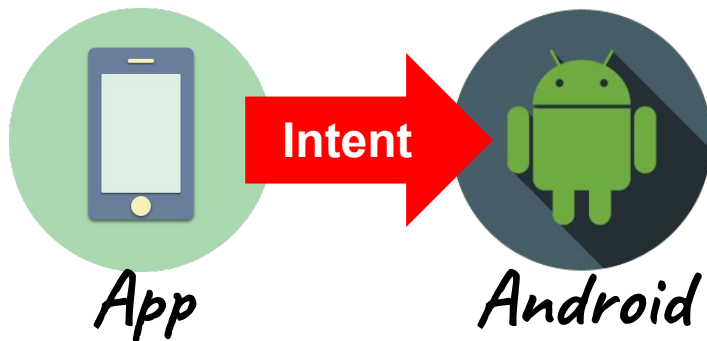
`resultData`: através deste parâmetro podemos obter os dados de retorno da Intent. Neste exemplo, vamos pegar a URI do arquivo.

Intents implícitas & Framework Access Storage

<https://developer.android.com/guide/components/intents-filters?hl=pt-br>

PASSO 1

Primeiramente criamos uma **Intent** indicando detalhes sobre a Activity que precisamos.

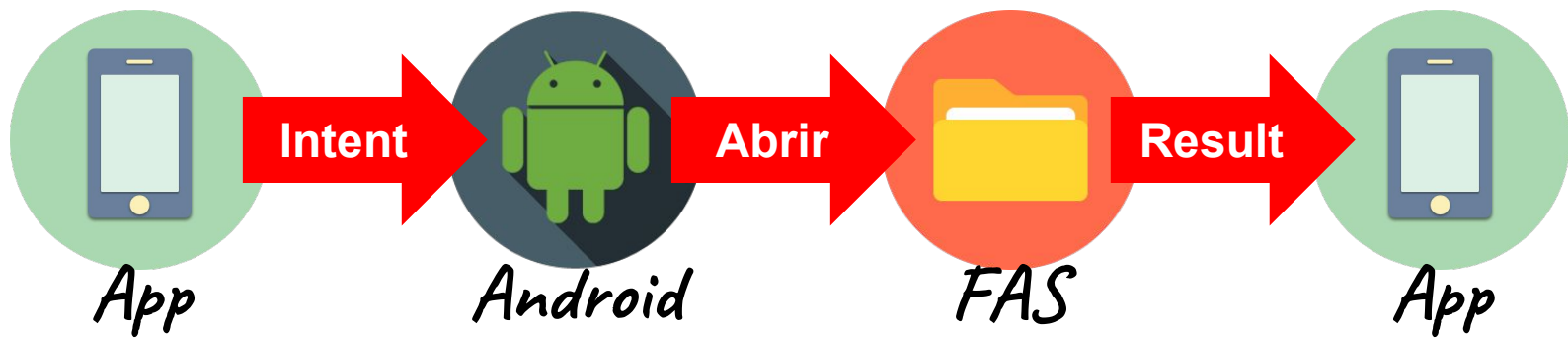


```
private static final int CODIGO_ABRIR_ARQUIVO = 1;
```

Crie uma constante para guardar o *requestCode*

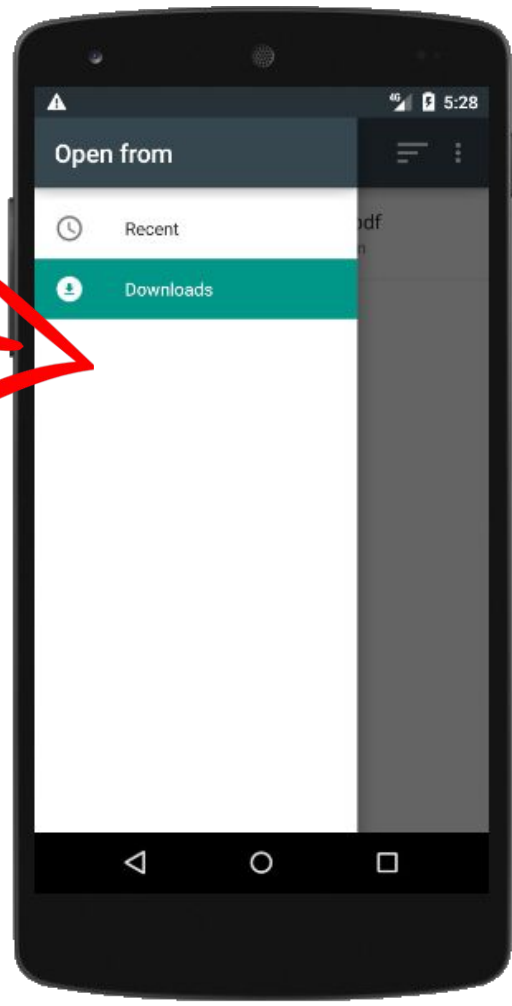
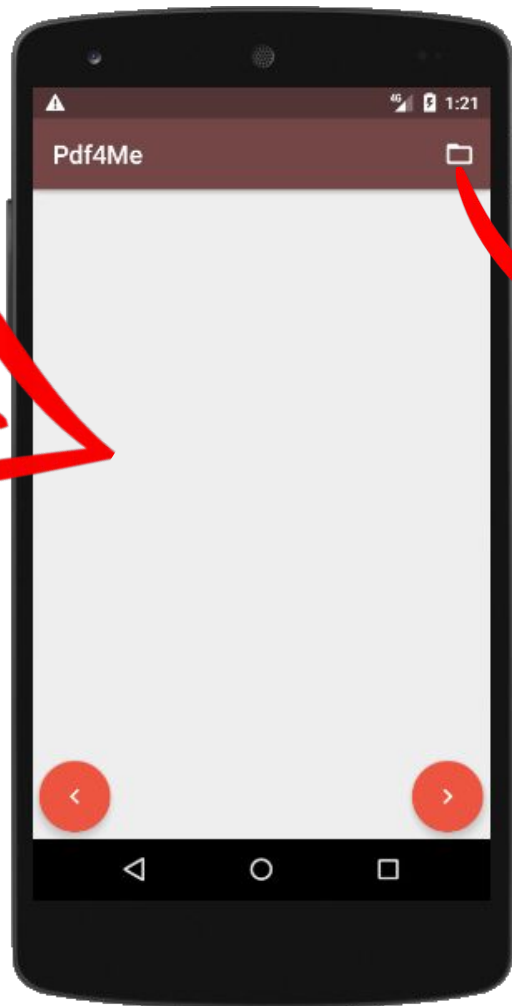
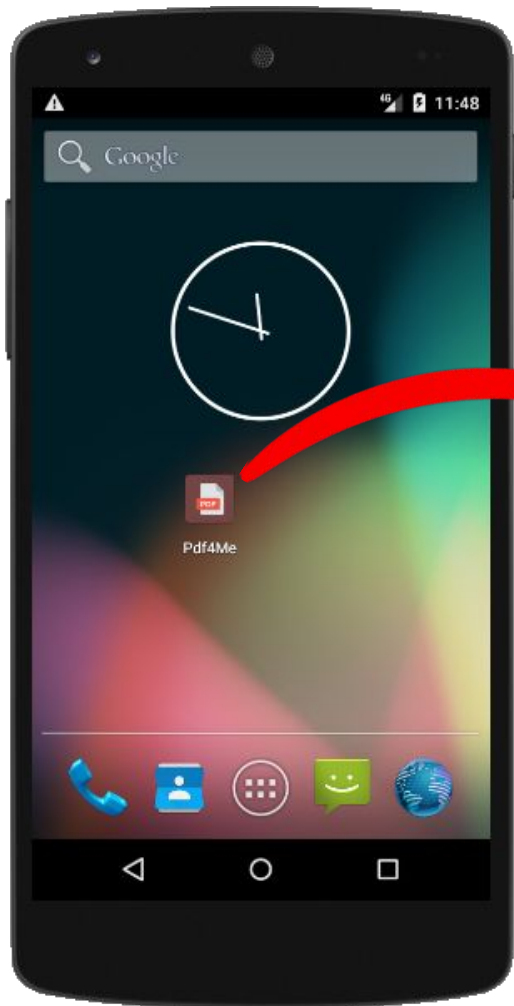
```
private void abrirFAS() {  
    Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);  
    intent.addCategory(Intent.CATEGORY_OPENABLE);  
    intent.setType("application/pdf");  
    startActivityForResult(intent, CODIGO_ABRIR_ARQUIVO);  
}
```

Crie um método para abrir a Intent do *Framework Access Storage*



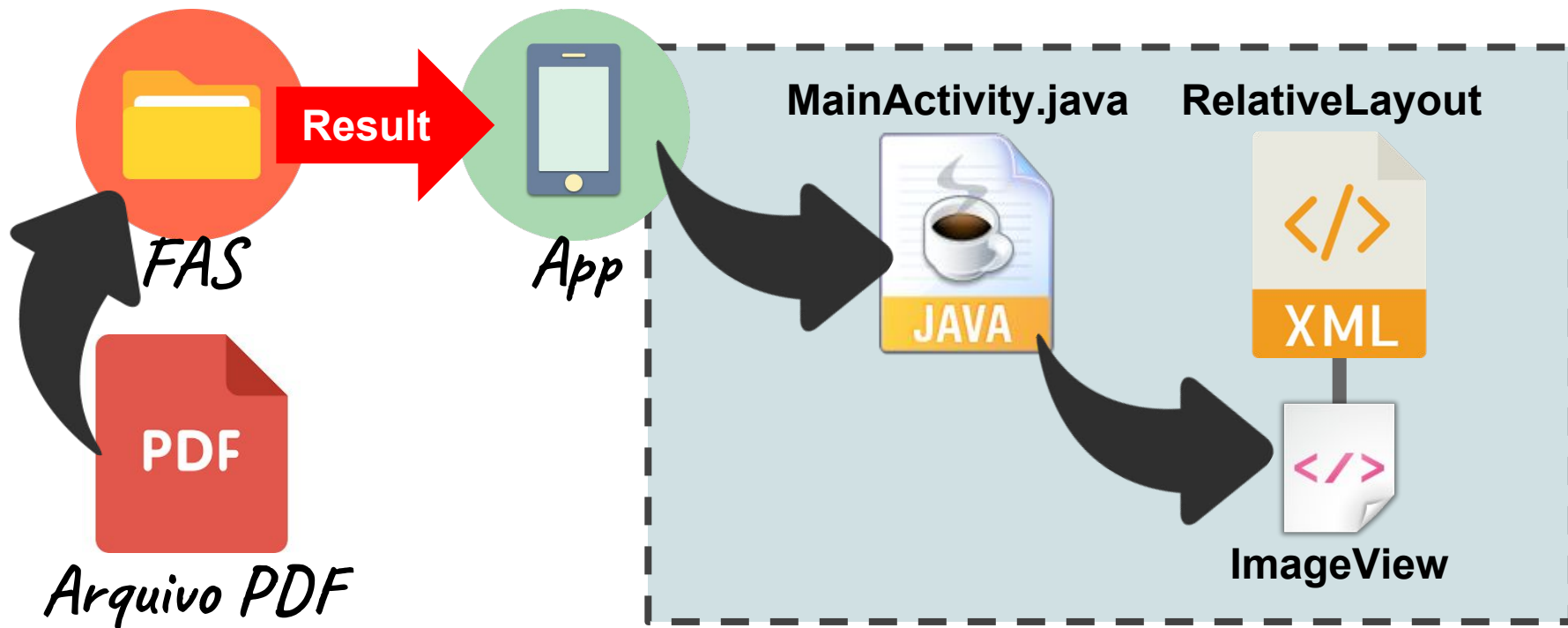
PASSO 2 - Em `onActivityResult()`, se o resultado for sucesso e o `requestCode` for o mesmo enviado na abertura da Intent, podemos obter a URI do arquivo pelo método `getData()` do parâmetro `resultData`. Para garantir, verificamos se ele está instanciado.

```
public void onActivityResult(int requestCode, int resultCode, Intent resultData) {  
    if (resultCode == Activity.RESULT_OK) {  
        if (requestCode == CODIGO_ABRIR_ARQUIVO) {  
            if (resultData != null) {  
                Uri uriArquivo = resultData.getData();  
                // Aqui devemos abrir o PDF  
            }  
        }  
    }  
}
```

Trabalhando com PdfRenderer

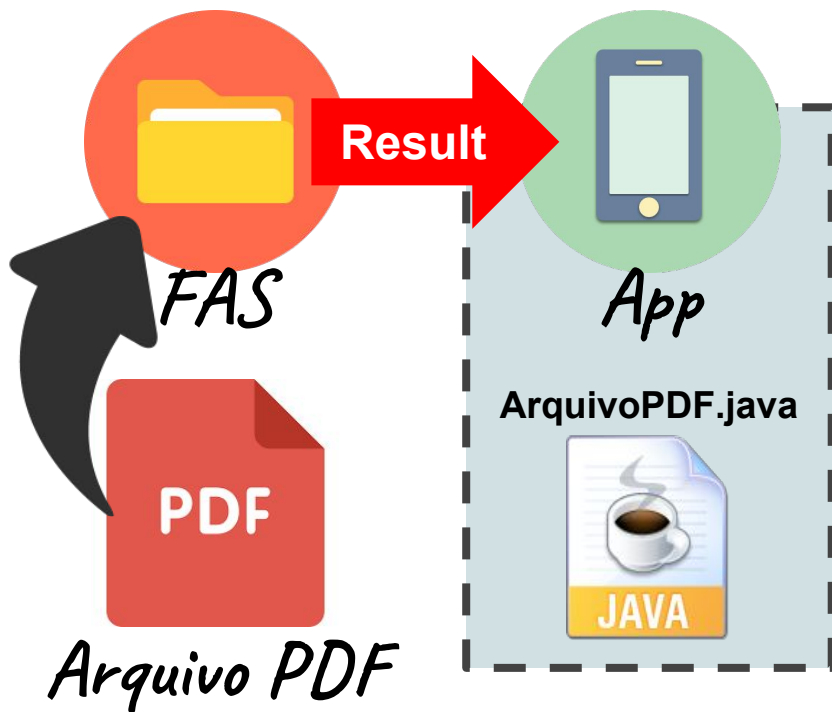
O PdfRenderer é um componente capaz de renderizar um PDF dentro de um ImageView. Ele renderiza uma página por vez.



Trabalhando com PdfRenderer

PASSO 1

Vamos criar uma classe *ArquivoPDF* para gerenciar a lógica de acesso ao PDF. De início, além da URI, precisamos gerenciar o total de páginas e a página atual.

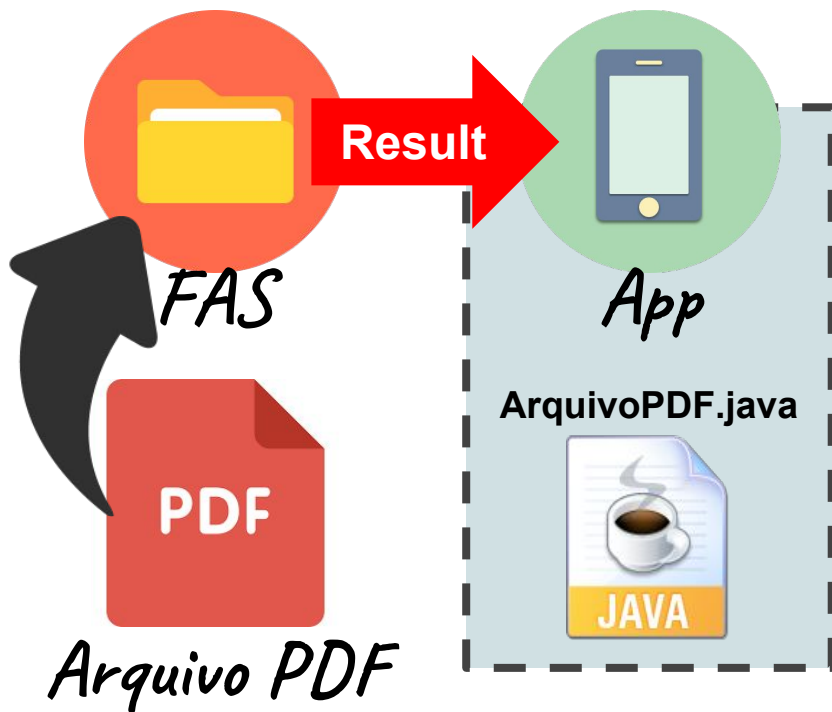


```
ArquivoPDF.java x
1  package com.lgapontes.pdf4me_v1;
2
3  import android.net.Uri;
4
5  public class ArquivoPDF {
6
7      private Uri uri;
8      private int totalPaginas;
9      private int paginaAtual;
10
11      public ArquivoPDF(Uri uri) {
12          this.uri = uri;
13          this.totalPaginas = 0;
14          this.paginaAtual = 0;
15      }
16
17  }
```

Trabalhando com PdfRenderer

PASSO 2

De posse dos atributos *paginaAtual* e *totalPaginas* nós podemos criar métodos para decrementar e incrementar a página atual. O método *exibir()* ainda não existe!



```
public void paginaAnterior() {  
    int pagina = paginaAtual - 1;  
    if (pagina >= 0) {  
        paginaAtual = pagina;  
        exibir();  
    }  
}  
  
public void paginaPosterior() {  
    int pagina = paginaAtual + 1;  
    if (pagina < totalPaginas) {  
        paginaAtual = pagina;  
        exibir();  
    }  
}
```

Trabalhando com PdfRenderer

PASSO 3

O objeto *PdfRenderer* deve receber como parâmetro um *ParcelFileDescriptor*, que uma classe utilizada no Android para abrir e transmitir arquivos entre processos. O *ParcelFileDescriptor*, por sua vez, será criado a partir da URI e do *context*.

```
private void exibir() {  
    /* Obter PDF Renderer */  
    ParcelFileDescriptor fileDescriptor = context.getContentResolver()  
        .openFileDescriptor(this.uri, "r");  
    PdfRenderer pdfRenderer = new PdfRenderer(fileDescriptor);  
}
```

Como não temos o *context*, vamos precisar ajustar o constructor para recebê-lo no momento da criação do objeto. Aproveitando, vamos também já passar uma referência ao *ImageView* no qual o PDF será aberto (o *PdfRenderer* exibe a página do PDF em um elemento *ImageView*).

Trabalhando com PdfRenderer

PASSO 4

Então, para receber o contexto e o *ImageView*, crie duas propriedades e receba-os por parâmetro no constructor.

```
private Context context;  
private ImageView pdfView;  
private Uri uri;  
private int totalPaginas;  
private int paginaAtual;  
  
public ArquivoPDF(Context context, ImageView pdfView, Uri uri) {  
    this.context = context;  
    this.pdfView = pdfView;  
    this.uri = uri;  
    this.totalPaginas = 0;  
    this.paginaAtual = 0;  
}
```

Trabalhando com PdfRenderer

PASSO 5

De volta ao método *exibir()*, observem que o constructor do *PdfRenderer()* pode disparar uma *IOException*. Para assegurar que isso não prejudique o funcionamento do App, vamos encapsular tudo em try/catch com *Exception* e, caso ocorra o erro, exibir um *Toast* e logar o erro (que pode ser monitorado na aba Logcat do Android Studio).

```
private void exibir() {  
    try {  
  
        /* Obter PDF Renderer */  
        ParcelFileDescriptor fileDescriptor = context.getContentResolver()  
            .openFileDescriptor(this.uri, mode: "r");  
        PdfRenderer pdfRenderer = new PdfRenderer(fileDescriptor);  
  
    } catch (Exception e) {  
        Toast.makeText(context, text: "Erro ao abrir o arquivo!", Toast.LENGTH_LONG).show();  
        Log.e( tag: "PDF4ME", e.getMessage());  
    }  
}
```


PASSO 6

Obter o total de páginas e guardar no atributo *totalPaginas*.

```
/* Obter o total de páginas */  
totalPaginas = pdfRenderer.getPageCount();  
  
/* Abrir na página específica */  
PdfRenderer.Page rendererPage = pdfRenderer.openPage(paginaAtual);  
int rendererPageWidth = rendererPage.getWidth();  
int rendererPageHeight = rendererPage.getHeight();  
Bitmap bitmap = Bitmap.createBitmap(  
    rendererPageWidth,  
    rendererPageHeight,  
    Bitmap.Config.ARGB_8888);  
rendererPage.render(bitmap, destClip: null, transform: null,  
    PdfRenderer.Page.RENDER_MODE_FOR_DISPLAY);  
pdfView.setImageBitmap(bitmap);
```

PASSO 7

Criar um objeto *Page* apontando para a página guardada em *paginaAtual*, que inicial é 0 (primeira página).

```
/* Obter o total de páginas */
totalPaginas = pdfRenderer.getPageCount();

/* Abrir na página específica */
PdfRenderer.Page rendererPage = pdfRenderer.openPage(paginaAtual);
int rendererPageWidth = rendererPage.getWidth();
int rendererPageHeight = rendererPage.getHeight();
Bitmap bitmap = Bitmap.createBitmap(
    rendererPageWidth,
    rendererPageHeight,
    Bitmap.Config.ARGB_8888);
rendererPage.render(bitmap, destClip: null, transform: null,
    PdfRenderer.Page.RENDER_MODE_FOR_DISPLAY);
pdfView.setImageBitmap(bitmap);
```

PASSO 8

Para exibir a página do PDF no *ImageView*, precisamos criar uma imagem através da factory *Bitmap.createBitmap()*. Neste caso, devemos informar o tamanho (através dos *width* e *height* obtidos da própria página PDF) e o formato da imagem. Devemos obrigatoriamente utilizar o formato *ARGB_8888*.

https://developer.android.com/reference/android/graphics/Bitmap.Config#ARGB_8888

```
/* Abrir na página específica */
PdfRenderer.Page rendererPage = pdfRenderer.openPage(paginaAtual);
int rendererPageWidth = rendererPage.getWidth();
int rendererPageHeight = rendererPage.getHeight();
Bitmap bitmap = Bitmap.createBitmap(
    rendererPageWidth,
    rendererPageHeight,
    Bitmap.Config.ARGB_8888);
rendererPage.render(bitmap, destClip: null, transform: null,
    PdfRenderer.Page.RENDER_MODE_FOR_DISPLAY);
pdfView.setImageBitmap(bitmap);
```


PASSO 9

Com a imagem criada, devemos invocar o método *render* do *rendererPage*. Neste passo, nós poderíamos especificar o local onde a página seria desenhada na imagem (*destClip*) ou recortar um pedaço da imagem para associá-la à imagem (*transform*).

```
Bitmap bitmap = Bitmap.createBitmap(  
    rendererPageWidth,  
    rendererPageHeight,  
    Bitmap.Config.ARGB_8888);  
rendererPage.render(bitmap, destClip: null, transform: null,  
    PdfRenderer.Page.RENDER_MODE_FOR_DISPLAY);  
pdfView.setImageBitmap(bitmap);
```

Devemos informar também o modo de renderização, que pode variar entre um formato para exibição na tela (*RENDER_MODE_FOR_DISPLAY*) ou impressora (*RENDER_MODE_FOR_PRINT*). Para mais detalhes sobre essa conversão, vide: <https://developer.android.com/reference/android/graphics/pdf/PdfRenderer.Page>

Por fim, definimos na *ImageView* (que chamamos de *pdfView*) a imagem criada.

```

private void exibir() {
    try {
        /* Obter PDF Renderer */
        ParcelFileDescriptor fileDescriptor = context.getContentResolver()
            .openFileDescriptor(this.uri, mode: "r");
        PdfRenderer pdfRenderer = new PdfRenderer(fileDescriptor);

        /* Obter o total de páginas */
        totalPaginas = pdfRenderer.getPageCount();

        /* Abrir na página específica */
        PdfRenderer.Page rendererPage = pdfRenderer.openPage(paginaAtual);
        int rendererPageWidth = rendererPage.getWidth();
        int rendererPageHeight = rendererPage.getHeight();
        Bitmap bitmap = Bitmap.createBitmap(
            rendererPageWidth,
            rendererPageHeight,
            Bitmap.Config.ARGB_8888);
        rendererPage.render(bitmap, destClip: null, transform: null,
            PdfRenderer.Page.RENDER_MODE_FOR_DISPLAY);
        pdfView.setImageBitmap(bitmap);

        /* Fechar referência ao PDF */
        rendererPage.close();
        pdfRenderer.close();
        fileDescriptor.close();
    } catch (Exception e) {
        Toast.makeText(context, text: "Erro ao abrir o arquivo!", Toast.LENGTH_LONG).show();
        Log.e( tag: "PDF4ME", e.getMessage());
    }
}

```

PASSO 10

O último passo é fechar todos os objetos abertos associados à leitura do arquivo PDF e conversão da imagem.

PASSO 11

Da forma como codificamos, precisaríamos chamar *exibir()* logo após a criação do novo objeto. Podemos encapsular isso em um método estático e tornar o constructor privado.

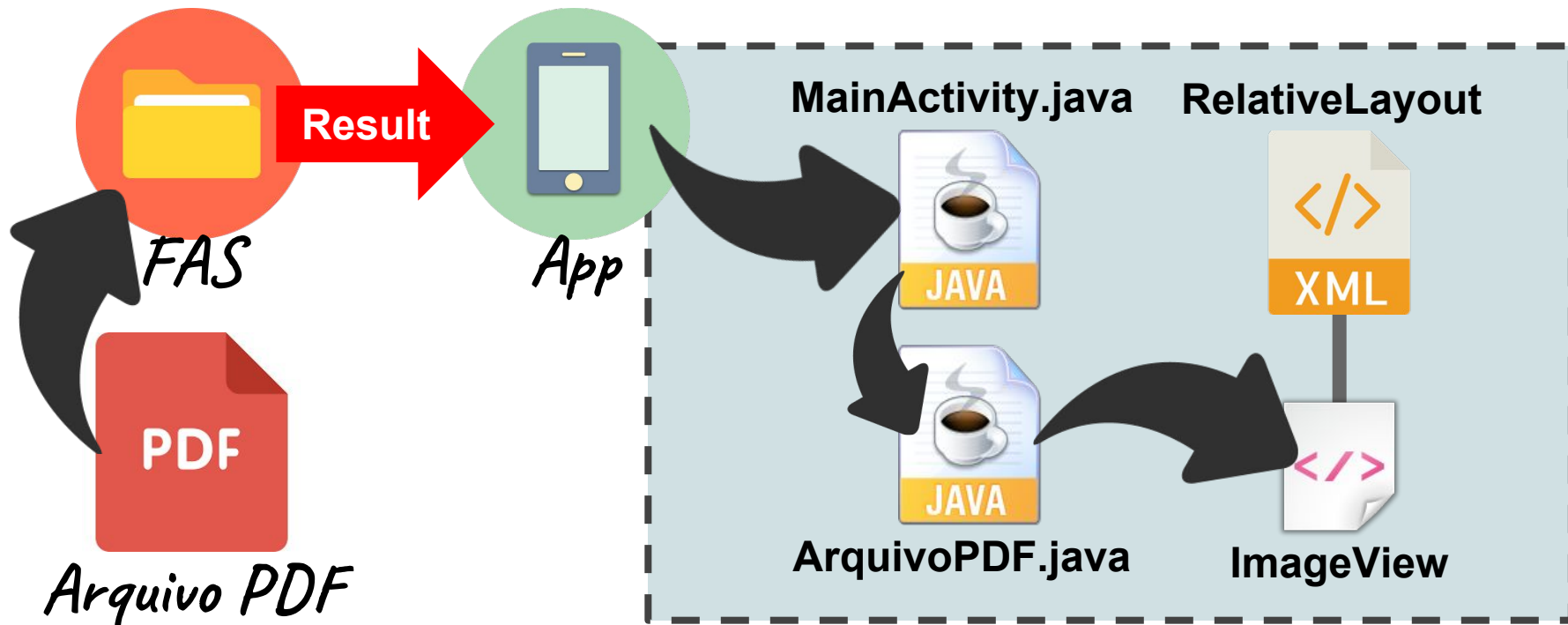
```
private ArquivoPDF(Context context, ImageView pdfView, Uri uri) {
    this.context = context;
    this.pdfView = pdfView;
    this.uri = uri;
    this.totalPaginas = 0;
    this.paginaAtual = 0;
}

public static ArquivoPDF abrir(Context context, ImageView pdfView, Uri uri) {
    ArquivoPDF arquivo = new ArquivoPDF(context, pdfView, uri);
    arquivo.exibir();
    return arquivo;
}
```

Esta é uma simples forma de implementar o *Factory Method* ou *Simple Factory*.
<https://refactoring.com/catalog/replaceConstructorWithFactoryMethod.html>

Trabalhando com PdfRenderer

Com isso nossa classe *ArquivoPDF* está pronta para o uso. Você poderá baixar o código completo em: <https://bit.ly/2MecyF0>



Trabalhando com PdfRenderer

Agora vamos ajustar o Layout XML e a *MainActivity.java* para abrir e manipular as páginas do arquivo PDF.

Adicione uma *ImageView* no *activity_main.xml* ocupando todo o espaço do pai e com o ID *pdfview*. Como estamos trabalhando com *RelativeLayout*, fixe o posicionamento do *ImageView* no topo, à esquerda.

```
<?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">
    <ImageView
        android:id="@+id/pdfview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true" />
    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab_pagina_anterior"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="16px"
        android:src="@drawable/ic_pagina_anterior"
        android:tint="@color/branco" />
    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab_pagina_posterior"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="16px"
        android:src="@drawable/ic_pagina_posterior"
        android:tint="@color/branco" />
</RelativeLayout>
```


Trabalhando com PdfRenderer

MainActivity.java

Crie dois atributos para guardar o *ImageView* e o *ArquivoPDF*.

No método *onCreate()*, defina o evento de clique dos FABs para chamar os métodos *paginaAnterior()* e *paginaPosterior()* caso o objeto *arquivo* esteja instanciado.

Obtenha o *ImageView* pelo ID e guarde-o no atributo *pdfView*.

```
private ImageView pdfView;  
private ArquivoPDF arquivo;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    fabPaginaAnterior = (FloatingActionButton)  
        findViewById(R.id.fab_pagina_anterior);  
    fabPaginaAnterior.setOnClickListener((view) -> {  
        if (arquivo != null) {  
            arquivo.paginaAnterior();  
        }  
    });  
  
    fabPaginaPosterior = (FloatingActionButton)  
        findViewById(R.id.fab_pagina_posterior);  
    fabPaginaPosterior.setOnClickListener((view) -> {  
        if (arquivo != null) {  
            arquivo.paginaPosterior();  
        }  
    });  
  
    pdfView = (ImageView) findViewById(R.id.pdfview);  
}
```

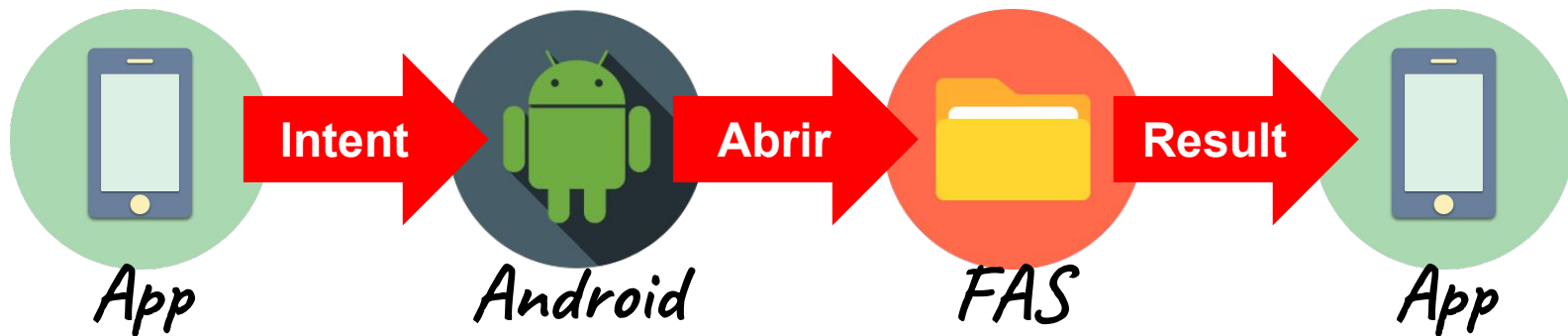
Trabalhando com PdfRenderer

MainActivity.java

Por fim, altere o método `onActivityResult()` para criar um novo *ArquivoPDF* através do factory `abrir()`. Guarde a instância no atributo `arquivo`.

```
public void onActivityResult(int requestCode, int resultCode, Intent resultData) {  
    if (resultCode == Activity.RESULT_OK) {  
        if (requestCode == CODIGO_ABRIR_ARQUIVO) {  
            if (resultData != null) {  
                Uri uriArquivo = resultData.getData();  
                this.arquivo = ArquivoPDF.abrir(context: this, pdfView, uriArquivo);  
            }  
        }  
    }  
}
```

Testando o PdfRenderer



1- No seu celular (ou emulador), baixe o arquivo abaixo:

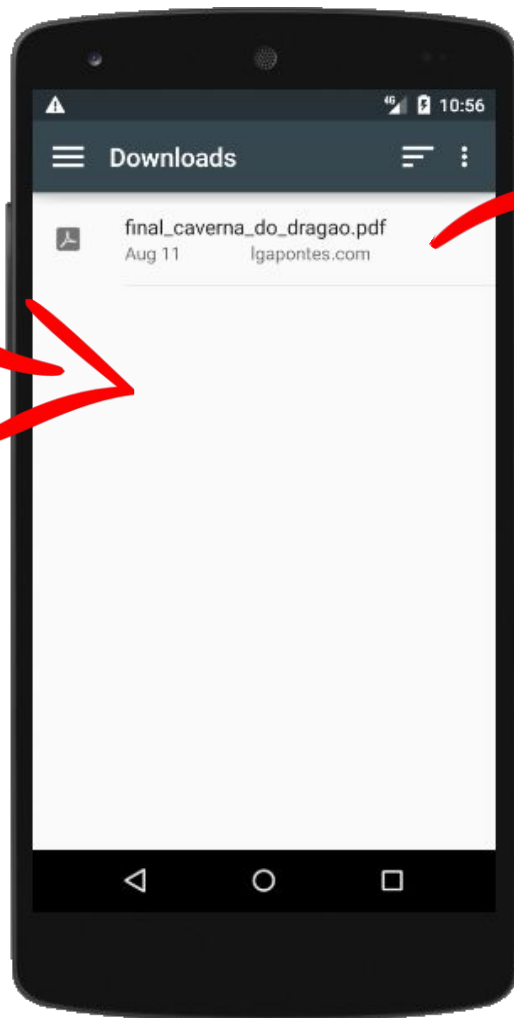
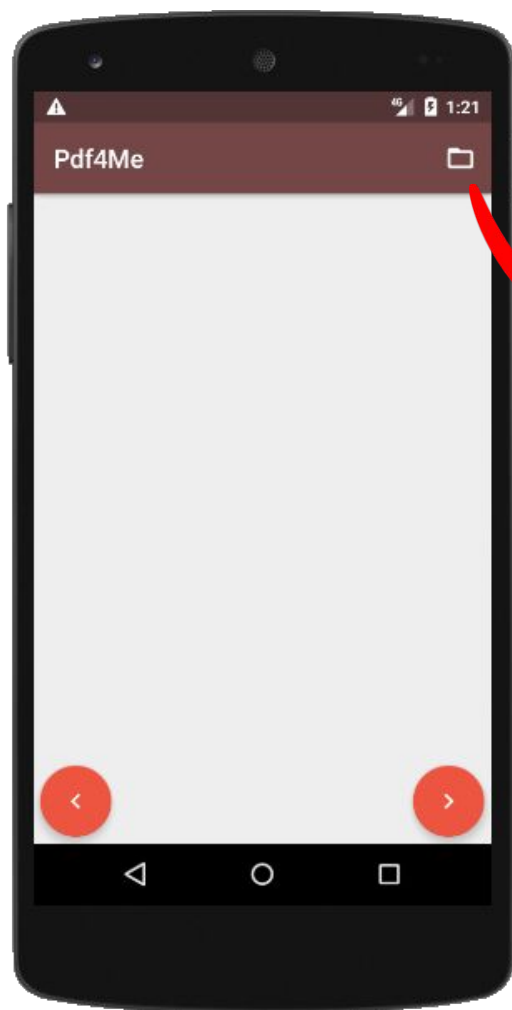
http://lgapontes.com/aulas/pdf/final_caverna_do_dragao.pdf

2- Publique o projeto Pdf4Me no celular (ou emulador)

3- Execute-o, clique no ícone da pasta no ActionBar

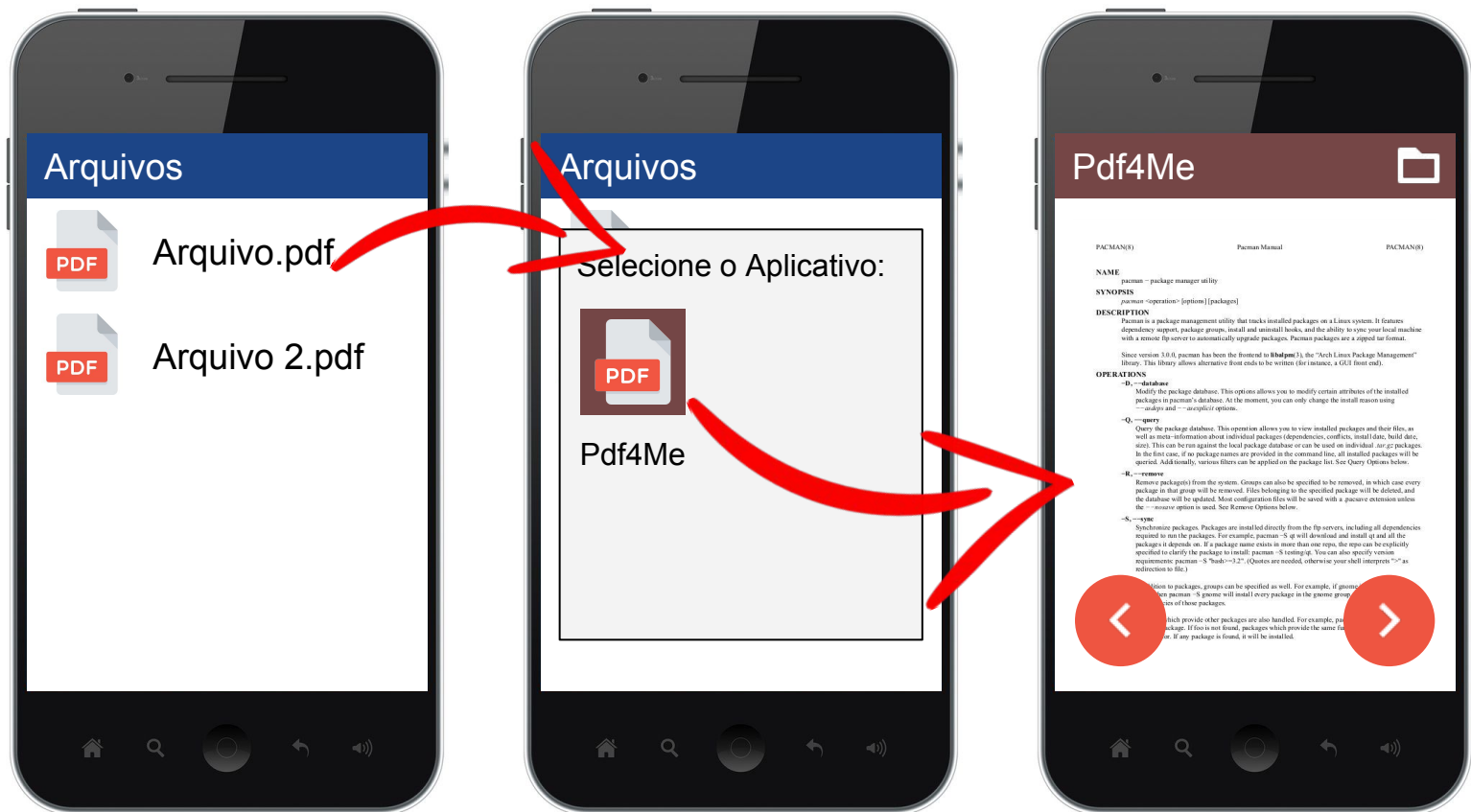
4- Através do *Framework Access Storage*, acesse o arquivo PDF baixado e veja que ele será exibido na tela. Navegue pelas páginas através dos FABs.





Intent Filter

App com *Intent Filter* para arquivos PDF



Como funciona o *Intent Filter* ?

Quando o usuário clica em um arquivo no celular, o Android faz uma busca em todos os aplicativos instalados verificando qual deles é capaz de abri-lo.



Como funciona o *Intent Filter* ?

No arquivo *Manifest.xml* de cada App há informações que permitem o Android "escolher" quem pode abrir arquivos PDF. Caso exista mais de um aplicativo, uma tela de seleção será exibida ao usuário.



PASSO 1

Configurar no arquivo *Manifest.xml* os dados do Intent Filter.

```
AndroidManifest.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    xmlns:tools="http://schemas.android.com/tools"
4    package="com.lgapontes.pdf4me_v1">
5    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="26" />
6    <application
7        android:allowBackup="true"
8        android:icon="@mipmap/ic_launcher"
9        android:label="@string/app_name"
10       android:roundIcon="@mipmap/ic_launcher_round"
11       android:supportsRtl="true"
12       android:theme="@style/AppTheme">
13       <activity android:name=".MainActivity">
14           <intent-filter>
15               <action android:name="android.intent.action.MAIN" />
16               <category android:name="android.intent.category.LAUNCHER" />
17           </intent-filter>
18           <intent-filter tools:ignore="AppLinkUrlError">
19               <action android:name="android.intent.action.SEND" />
20               <action android:name="android.intent.action.VIEW" />
21               <category android:name="android.intent.category.DEFAULT" />
22               <data android:mimeType="application/pdf" />
23           </intent-filter>
24       </activity>
25   </application>
26 </manifest>
```


PASSO 1

Configurar no arquivo *Manifest.xml* os dados do Intent Filter.

Android App Link

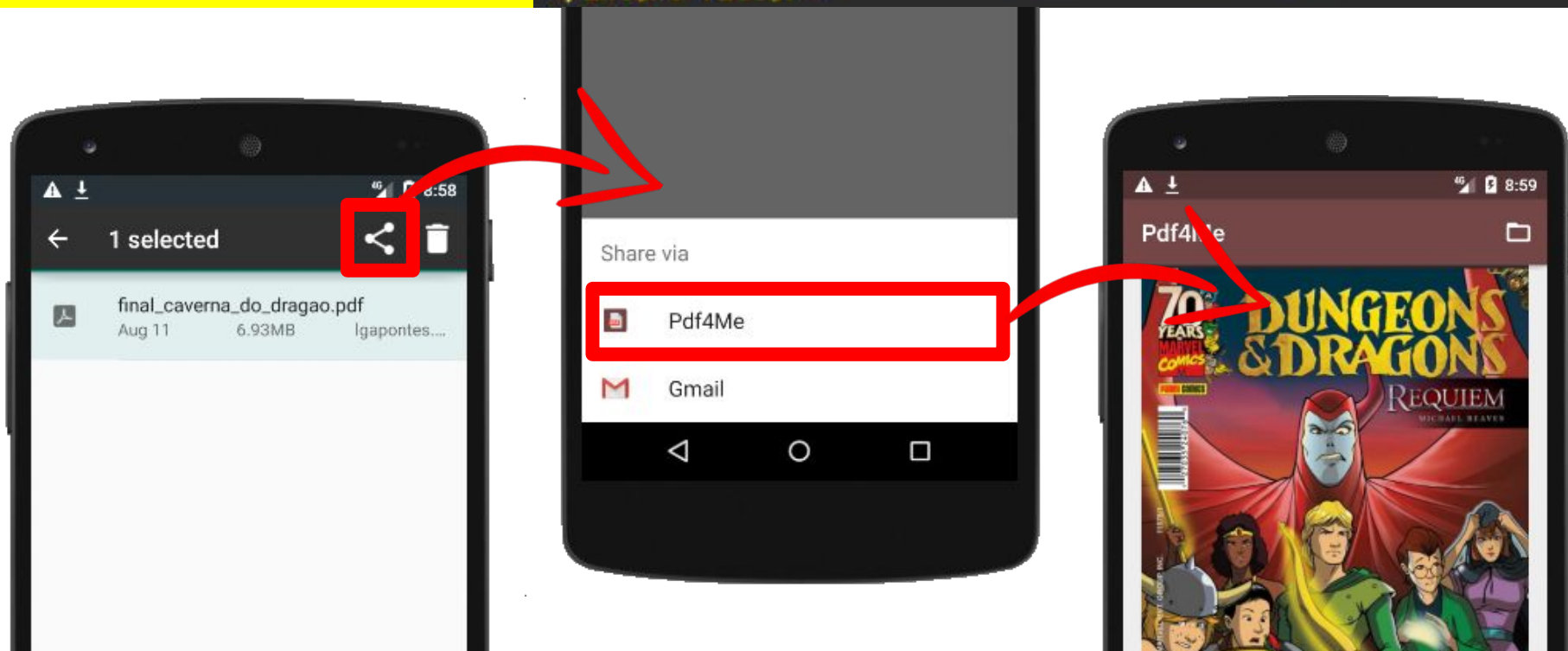
A partir da API Level 23 o Android fornece um meio de oferecer a abertura de seu aplicativo a partir de uma determinada URL. Como não vamos utilizar este recurso, devemos especificar esse atributo.

```
AndroidManifest.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    xmlns:tools="http://schemas.android.com/tools"
    package="com.lgapontes.pdf4me_v1">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="26" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        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>
            <intent-filter tools:ignore="AppLinkUrlError">
                <action android:name="android.intent.action.SEND" />
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="application/pdf" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

PASSO 1 - Configurar no arquivo *Manifest.xml* os dados do Intent Filter.

SEND: esta opção indica ao Android que podemos receber arquivos através da opção de compartilhamento.

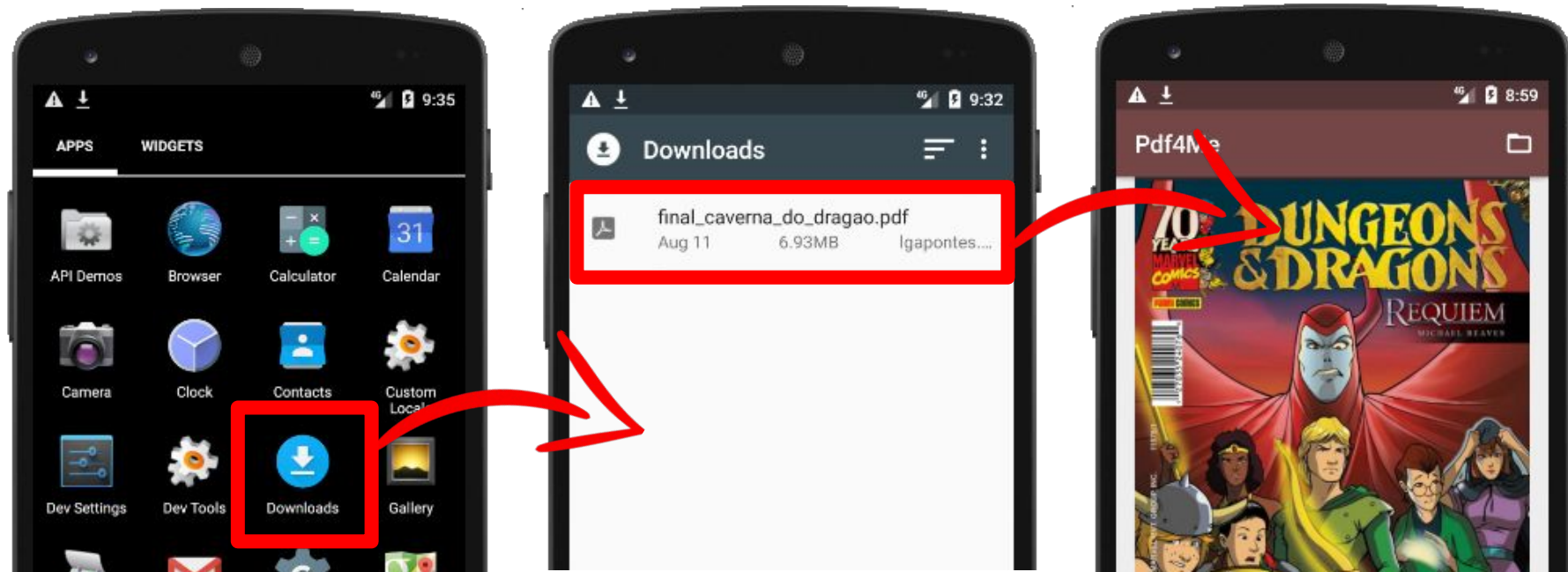
```
<intent-filter tools:ignore="AppLinkUrlError">  
  <action android:name="android.intent.action.SEND" />  
  <action android:name="android.intent.action.VIEW" />  
  <category android:name="android.intent.category.DEFAULT" />  
  <data android:mimeType="application/pdf" />  
</intent-filter>
```



PASSO 1 - Configurar no arquivo *Manifest.xml* os dados do Intent Filter.

VIEW: permite abrir um arquivo diretamente através de um App de visualização de arquivos no celular.

```
<intent-filter tools:ignore="AppLinkUrlError">  
  <action android:name="android.intent.action.SEND" />  
  <action android:name="android.intent.action.VIEW" />  
  <category android:name="android.intent.category.DEFAULT" />  
  <data android:mimeType="application/pdf" />  
</intent-filter>
```



PASSO 1

Configurar no arquivo *Manifest.xml* os dados do Intent Filter.

category.DEFAULT

Como vimos, a criação de uma Intent implícita pode informar uma categoria. Ao definir este valor no *Intent Filter*, estamos avisando ao Android que nossa App **não** exige uma categoria explícita.

```
AndroidManifest.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    xmlns:tools="http://schemas.android.com/tools"
    package="com.lgapontes.pdf4me_v1">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="26" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        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>
            <intent-filter tools:ignore="AppLinkUrlError">
                <action android:name="android.intent.action.SEND" />
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="application/pdf" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

PASSO 1

Configurar no arquivo *Manifest.xml* os dados do Intent Filter.

MIME Type

Por fim, podemos informar ao Android que nossa App poderá abrir arquivos do tipo *application/pdf*. Isso é necessário para o Android associar nossa App a esse tipo específico de arquivo.

```
AndroidManifest.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    xmlns:tools="http://schemas.android.com/tools"
    package="com.lgapontes.pdf4me_v1">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="26" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        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>
            <intent-filter tools:ignore="AppLinkUrlError">
                <action android:name="android.intent.action.SEND" />
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="application/pdf" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```


PASSO 2

Configurar a *MainActivity.java* para receber os dados da *Intent* que a abriu. Isso pode ser realizado no próprio método *onCreate()*

```
Intent intent = getIntent();
if (intent != null) {
    String action = intent.getAction();
    String type = intent.getType();
    Uri uriArquivo = null;

    if ((type != null) && type.equalsIgnoreCase("application/pdf")) {
        if (Intent.ACTION_VIEW.equals(action)) {
            uriArquivo = (Uri) intent.getData();
        } else if (Intent.ACTION_SEND.equals(action)) {
            uriArquivo = (Uri) intent.getParcelableExtra(Intent.EXTRA_STREAM);
        }
    }

    if (uriArquivo != null) {
        this.arquivo = ArquivoPDF.abrir(context: this, pdfView, uriArquivo);
    }
}
```

PASSO 2

Configurar a *MainActivity.java* para receber os dados da *Intent* que a abriu. Isso pode ser realizado no próprio método *onCreate()*

```
Intent intent = getIntent();  
if (intent != null) {  
    String action = intent.getAction();  
    String type = intent.getType();  
    Uri uriArquivo = null;
```

Este trecho inicial obtém a *Intent* responsável pela abertura de nossa App através do método *getIntent()*, caso exista.

```
    if ((type != null) && type.equalsIgnoreCase(Intent.ACTION_VIEW)) {  
        if (Intent.ACTION_VIEW.equals(action)) {  
            uriArquivo = (Uri) intent.getData();  
        } else if (Intent.ACTION_SEND.equals(action)) {  
            uriArquivo = (Uri) intent.getData();  
        }  
    }  
}
```

A partir dela, obtemos o tipo de ação pela qual a App foi aberta e o MIME Type do arquivo aberto.

Por fim, criamos uma variável para guardar a URI do arquivo.

```
if (uriArquivo != null) {  
    this.arquivo = ArquivoPDF.abrir(context: this, pdfView, uriArquivo);  
}  
}
```

PASSO 2

Configurar o `MainActivity` para receber os dados do `Intent` que o abriu. Isso pode ser realizado da seguinte maneira:

A forma de obter a URI do arquivo via `Intent` varia de acordo com a ação pela qual a App foi aberta. Se ela foi aberta a partir do clique direto no arquivo PDF (VIEW), vamos utilizar o método `getData()`. Se o PDF foi compartilhado (SEND), vamos obter a URI pelo método `getParcelableExtra()`

Como podemos ter outras Intents abrindo nossa App, é importante comparar se o MIME Type de fato é um *application/pdf*.

```
if ((type != null) && type.equalsIgnoreCase( anotherString: "application/pdf")) {  
    if(Intent.ACTION_VIEW.equals(action)) {  
        uriArquivo = (Uri) intent.getData();  
    } else if (Intent.ACTION_SEND.equals(action)) {  
        uriArquivo = (Uri) intent.getParcelableExtra(Intent.EXTRA_STREAM);  
    }  
}
```

Parcelable é um `Serializable` "turbinado" que permite criar mecanismos de serialização próprios. Geralmente tem maior performance.

PASSO 2

Configurar a *MainActivity.java* para receber os dados da *Intent* que a abriu. Isso pode ser realizado no próprio método *onCreate()*

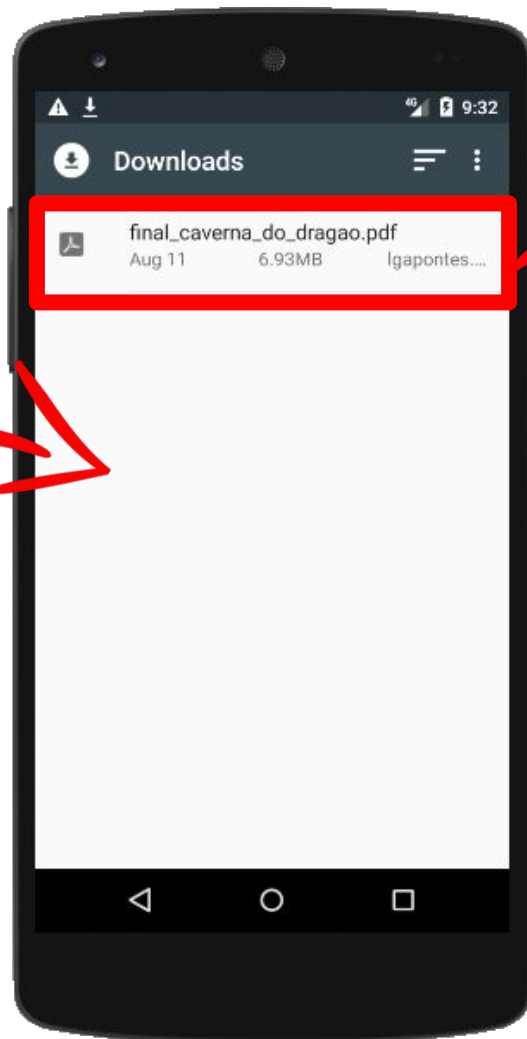
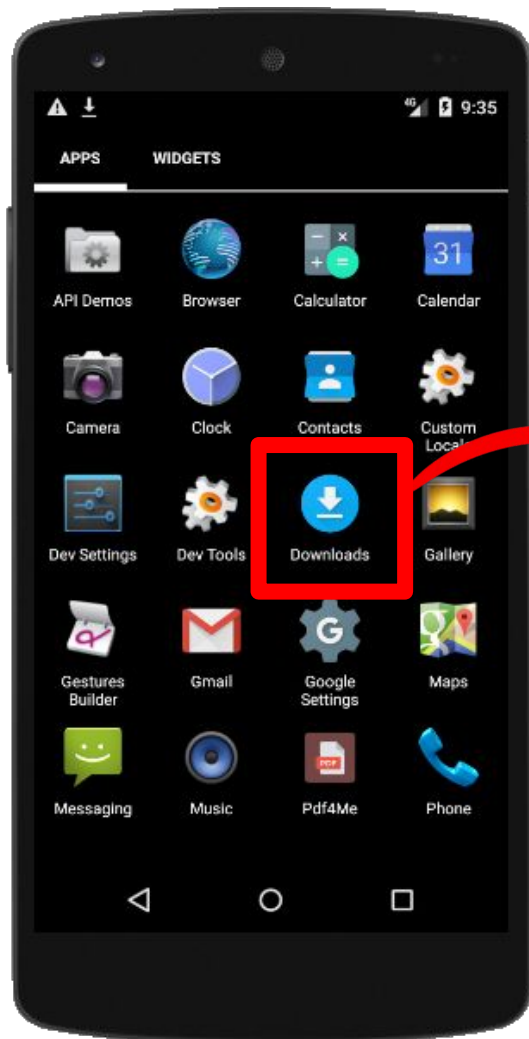
```
Intent intent = getIntent();
if (intent != null) {
    String action = intent.getAction();
    String type = intent.getType();
    Uri uriArquivo = null;

    if ((type != null) && type.equalsIgnoreCase("application/pdf")) {
        if(Intent.ACTION_VIEW.equals(action)) {
            uriArquivo = (Uri) intent.getData();
        } else if (Intent.ACTION_SEND.equals(action)) {
            uriArquivo = (Uri) intent.getData();
        }
    }
}
```

Por fim, se nós conseguirmos obter uma URI (via SEND ou VIEW), podemos abrir o arquivo com nossa classe *ArquivoPDF.java* normalmente.

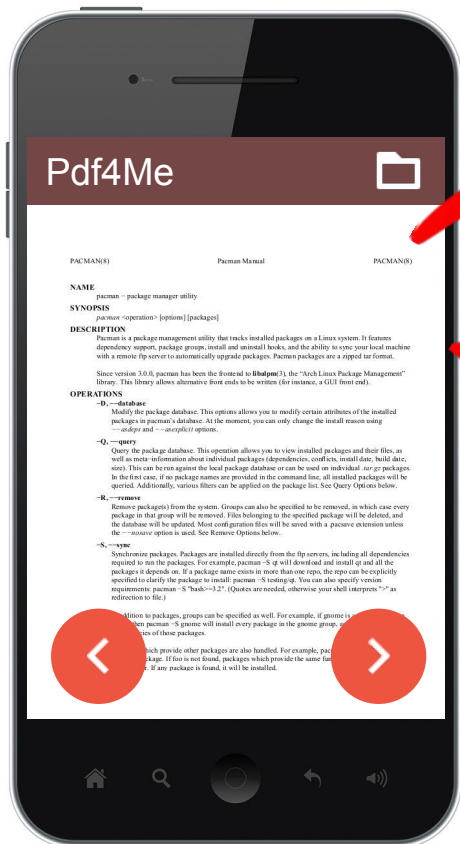
```
if(uriArquivo!=null) {
    this.arquivo = ArquivoPDF.abrir(context: this, pdfView, uriArquivo);
}
```

```
}
```

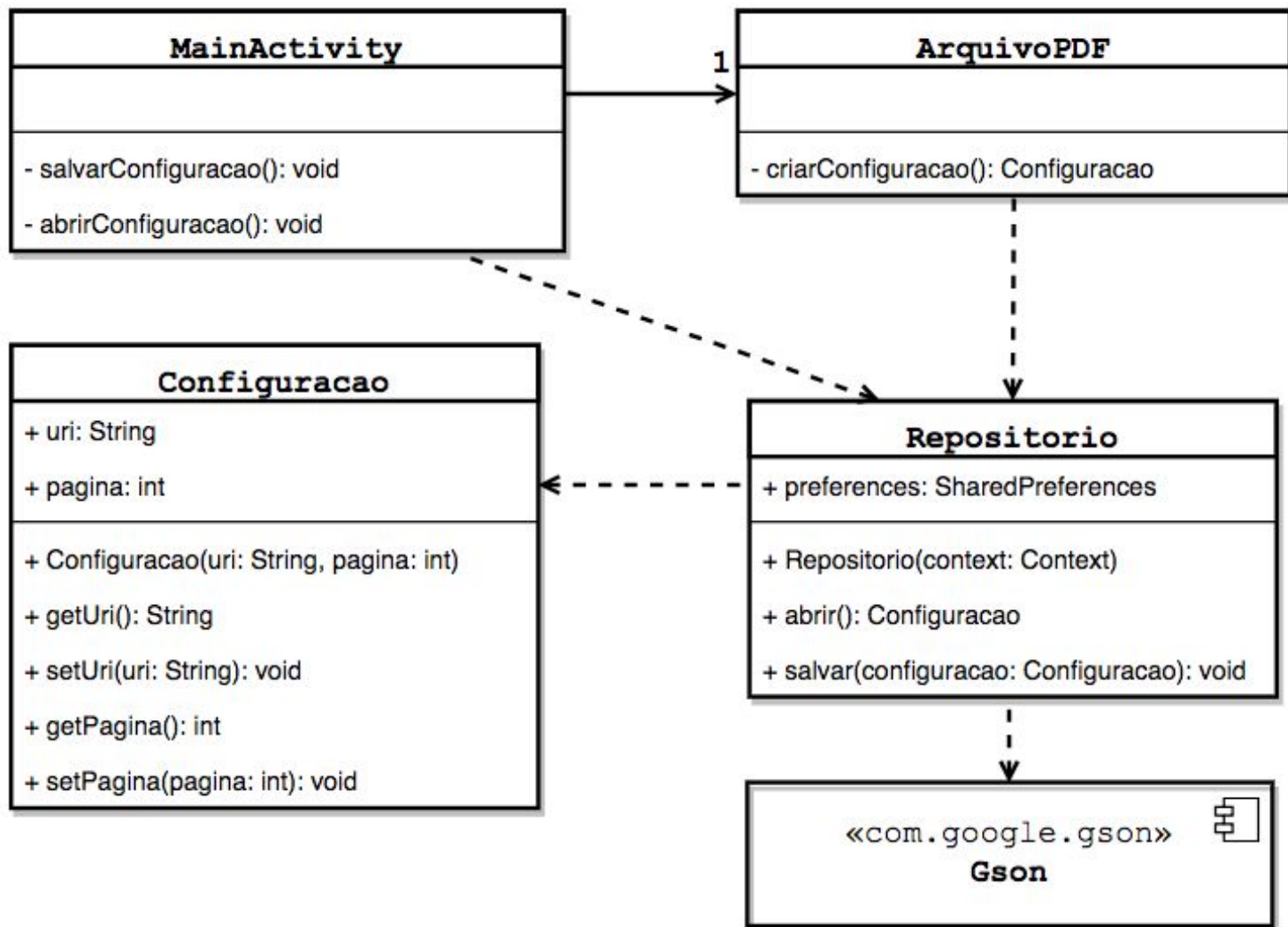
Shared Preferences

App salvando dados no *SharedPreferences*



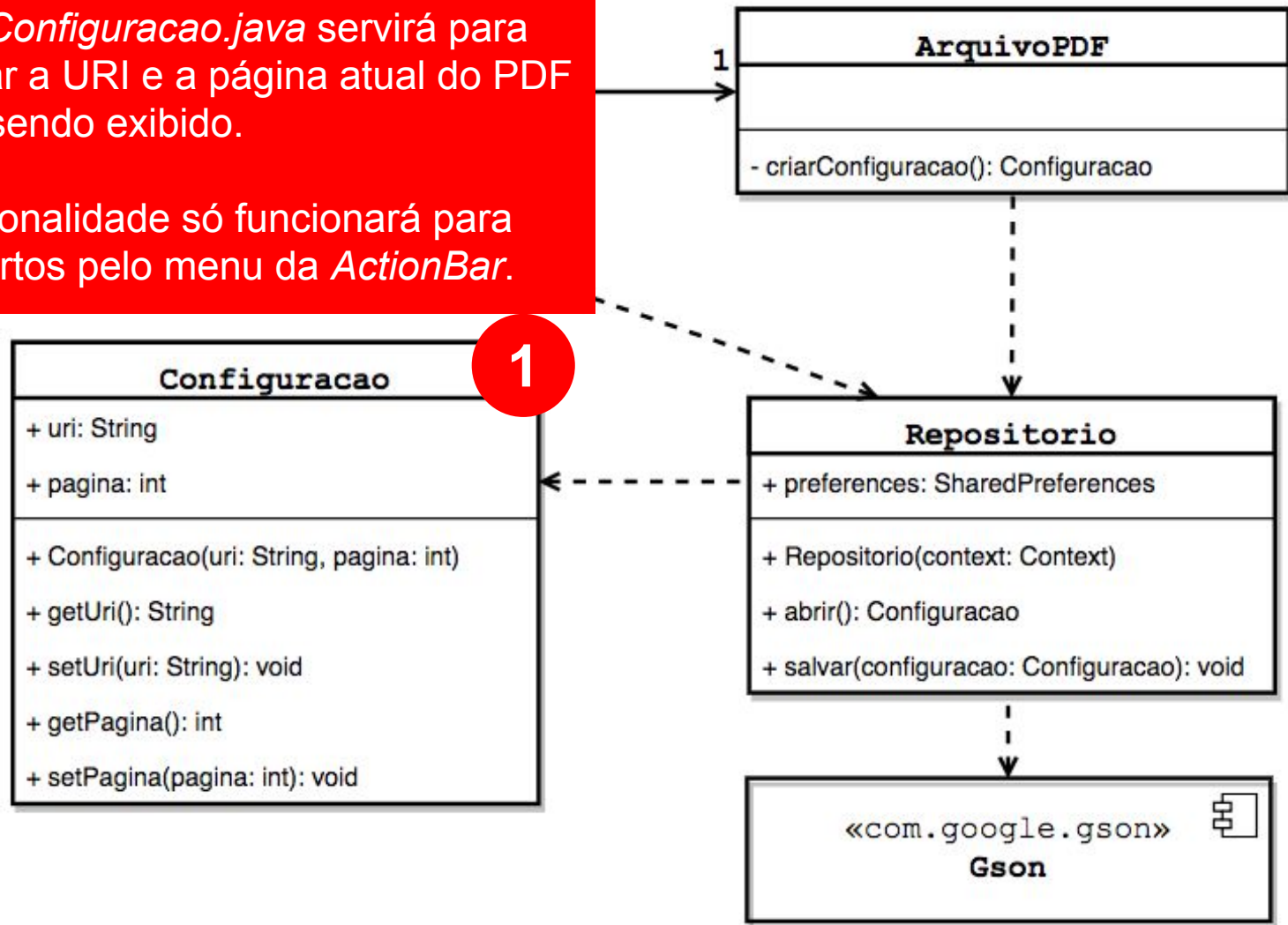
SharedPreferences

```
"pdf4me": {  
  
  "uri": "content://downloads/all_downloads/5",  
  "pagina": 10  
}
```



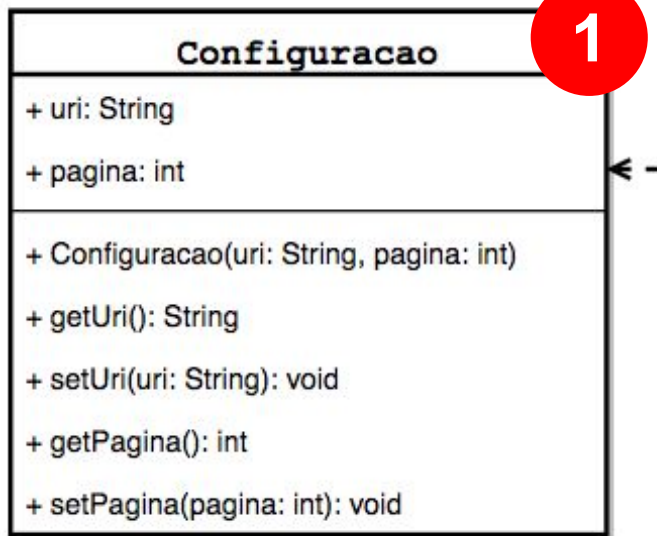
A classe *Configuracao.java* servirá para encapsular a URI e a página atual do PDF que está sendo exibido.

Esta funcionalidade só funcionará para PDFs abertos pelo menu da *ActionBar*.



A classe *Configuracao.java* servirá para encapsular a URI e a página atual do PDF que está sendo exibido.

Esta funcionalidade só funcionará para PDFs abertos pelo menu da *ActionBar*.



```
Configuracao.java x
1 package com.lgapontes.pdf4me_v1;
2
3 public class Configuracao {
4
5     private String uri;
6     private int pagina;
7
8     public Configuracao(String uri, int pagina) {
9         this.uri = uri;
10        this.pagina = pagina;
11    }
12
13    public String getUri() {
14        return uri;
15    }
16
17    public void setUri(String uri) {
18        this.uri = uri;
19    }
20
21    public int getPagina() {
22        return pagina;
23    }
24
25    public void setPagina(int pagina) {
26        this.pagina = pagina;
27    }
28 }
29
```

MainActivity

ArquivoPDF

1

A classe *Repositorio.java* será utilizada para salvar e recuperar as configurações. Como o *SharedPreferences* trabalha apenas com chaves e valores, vamos precisar converter o objeto *Configuracao.java* em um JSON para salvá-lo.

Configuracao

+ uri: String
+ pagina: int

+ Configuracao(uri: String, pagina: int)
+ getUri(): String
+ setUri(uri: String): void
+ getPagina(): int
+ setPagina(pagina: int): void

2

Repositorio

+ preferences: SharedPreferences
+ Repositorio(context: Context)
+ abrir(): Configuracao
+ salvar(configuracao: Configuracao): void

«com.google.gson»
Gson

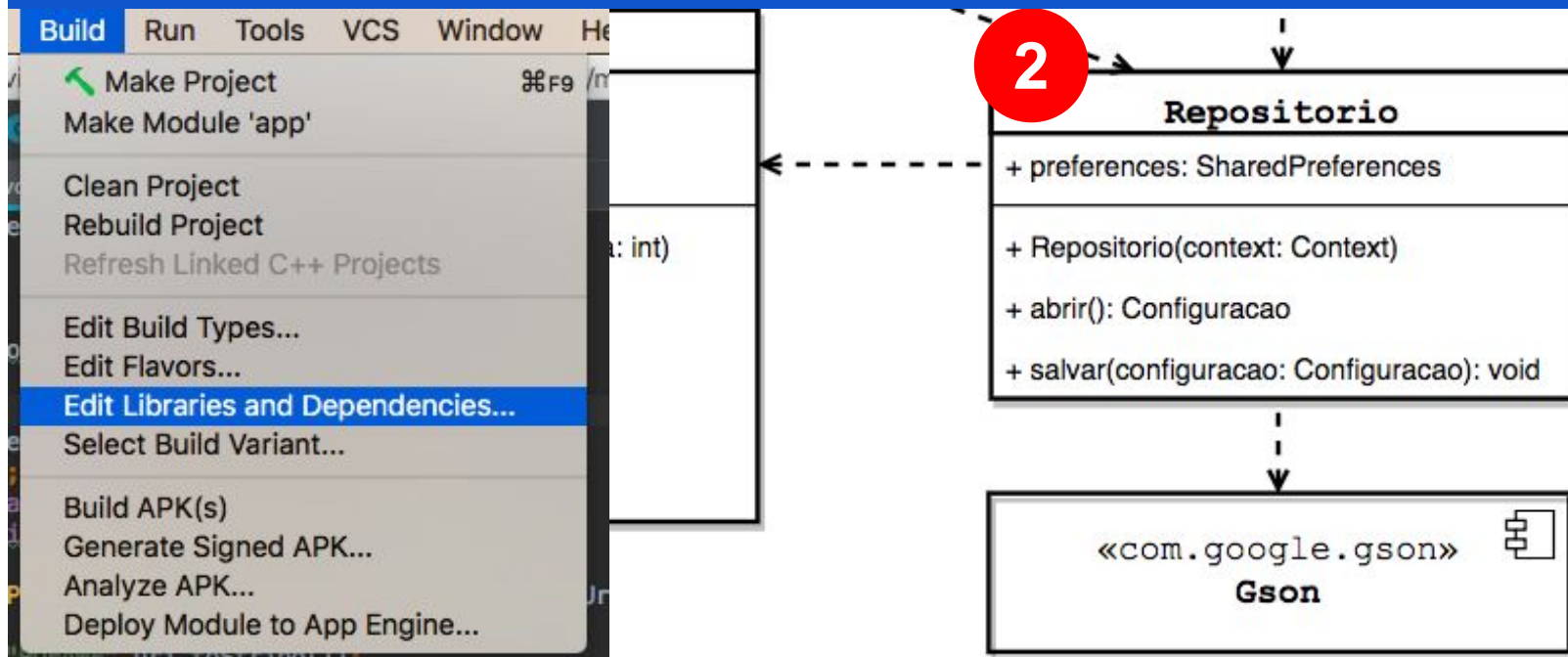
MainActivity

ArquivoPDF

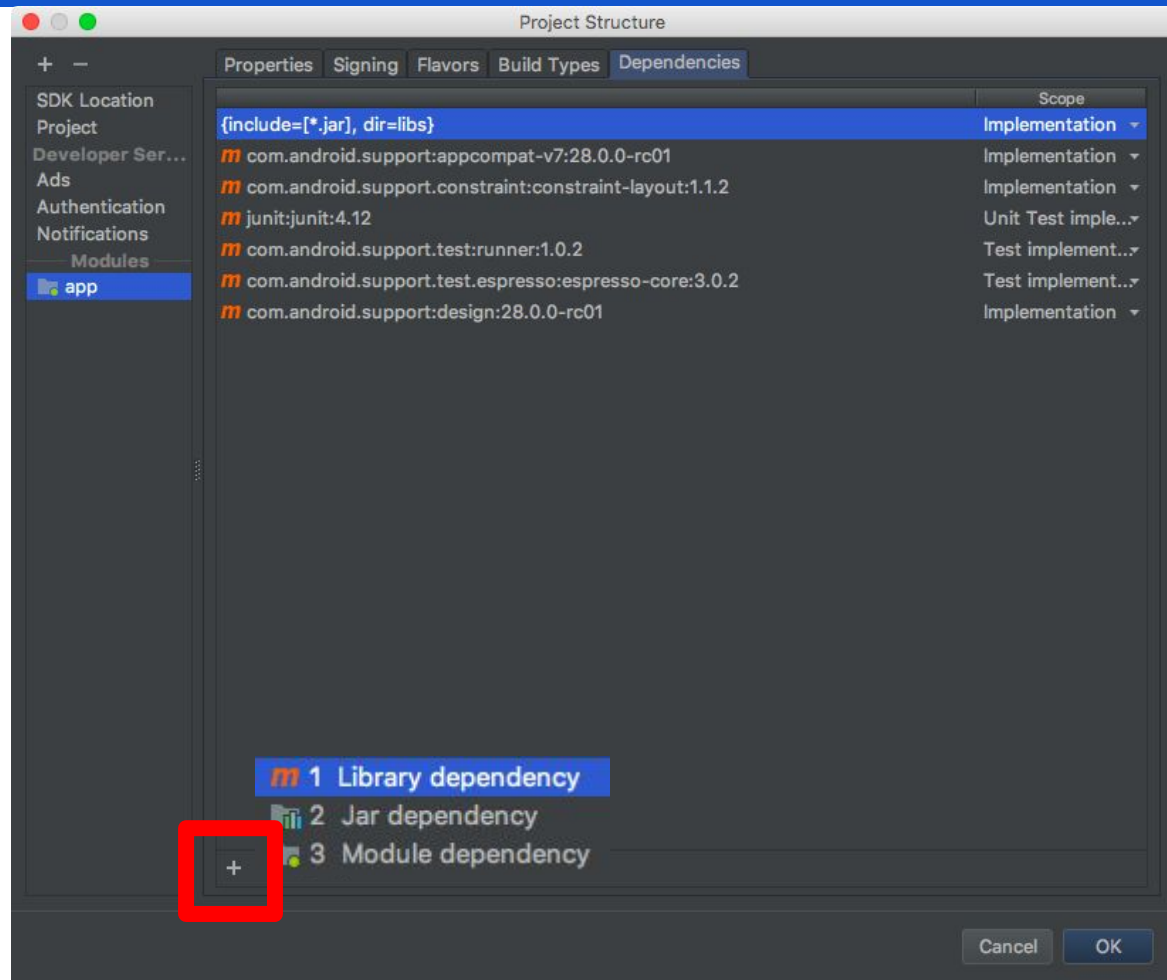
1

A classe *Repositorio.java* será utilizada para salvar e recuperar as configurações. Como o *SharedPreferences* trabalha apenas com chaves e valores, vamos precisar converter o objeto *Configuracao.java* em um JSON para salvá-lo.

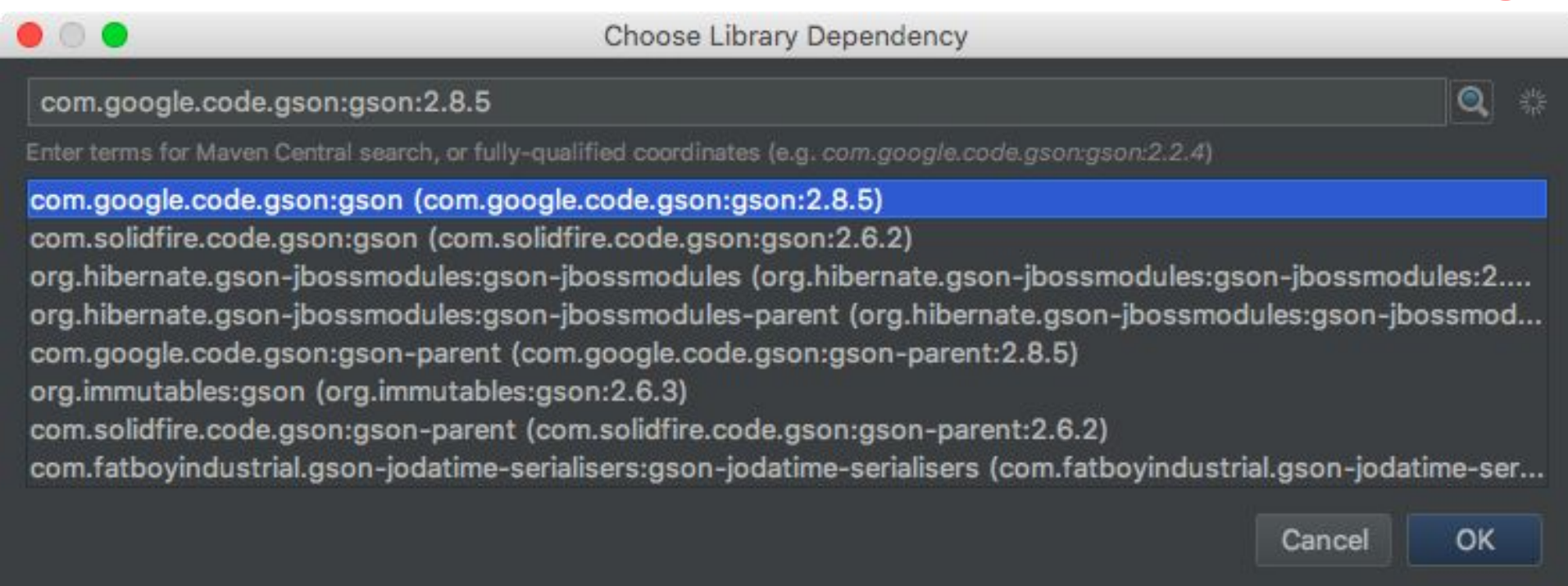
A conversão *objeto-json* será realizada pela biblioteca Gson. Vamos importá-la!



A conversão *objeto-json* será realizada pela biblioteca Gson. Vamos importá-la!



A conversão *objeto-json* será realizada pela biblioteca Gson. Vamos importá-la!



```
public class Repositorio {  
  
    private SharedPreferences preferences;  
  
    public Repositorio(Context context) {  
        this.preferences = context.getSharedPreferences( s: "pdf4me", Context.MODE_PRIVATE);  
    }  
  
    public Configuracao abrir() {  
        String json = this.preferences.getString( s: "configuracao", s1: null);  
        if (json != null) {  
            Gson gson = new Gson();  
            Configuracao configuracao = gson.fromJson(json, Configuracao.class);  
            return configuracao;  
        }  
        return null;  
    }  
  
    public void salvar(Configuracao configuracao) {  
        SharedPreferences.Editor editor = preferences.edit();  
        Gson gson = new Gson();  
        String json = gson.toJson(configuracao);  
        editor.putString( s: "configuracao", json);  
        editor.commit();  
    }  
}
```

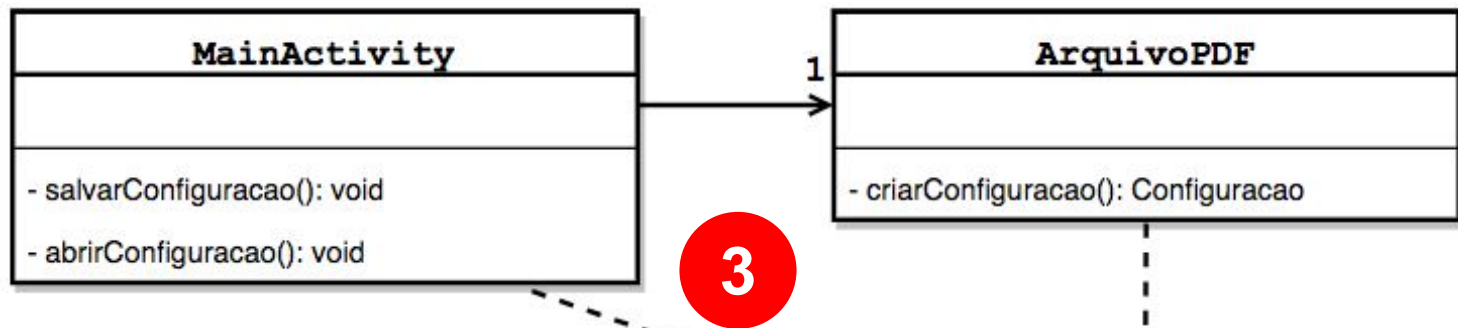
Repositorio

- + preferences: SharedPreferences
- + Repositorio(context: Context)
- + abrir(): Configuracao
- + salvar(configuracao: Configuracao): void



«com.google.gson»
Gson



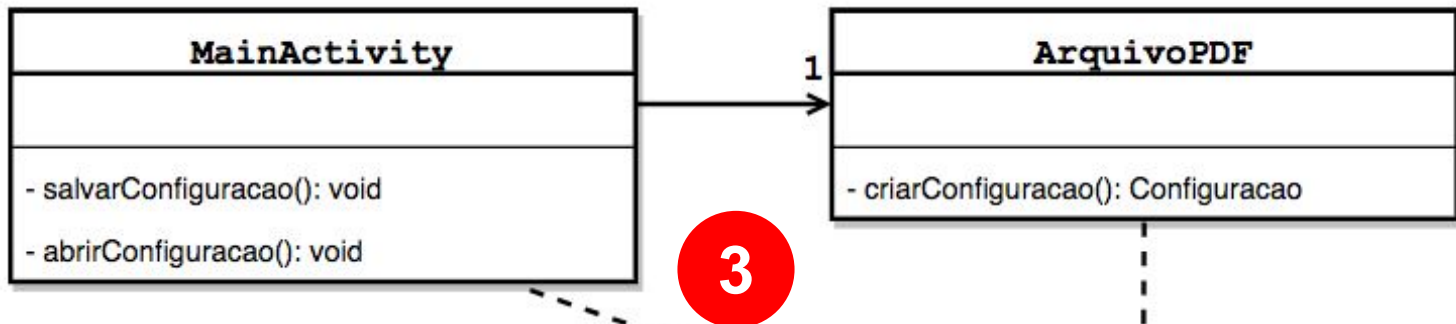


As classes *MainActivity.java* e *ArquivoPDF.java* vão utilizar o repositório para salvar e recuperar as configurações.

Salvar: a ação de salvar a configuração se dará na abertura do arquivo e na passagem de páginas.

Recuperar: a ação de obter a configuração se dará na abertura do App através do *Icon Launcher* disponível na área de trabalho do celular. Por questões de usabilidade, a abertura do App via compartilhamento (SEND) ou clique no arquivo (VIEW) sempre abrirá o PDF na página inicial.





A classe *ArquivoPDF.java* deve sobrescrever o constructor passando uma página (para os casos em que o App será aberto em uma página específica), e deve implementar o método *criarConfiguracao()*.

```
public static ArquivoPDF abrir(Context context, ImageView pdfView, Uri uri, int paginaAtual) {
    ArquivoPDF arquivo = new ArquivoPDF(context, pdfView, uri);
    arquivo.paginaAtual = paginaAtual;
    arquivo.exibir();
    return arquivo;
}
```

```
public Configuracao criarConfiguracao() {
    return new Configuracao(this.uri.toString(), this.paginaAtual);
}
```


MainActivity
- salvarConfiguracao(): void - abrirConfiguracao(): void

3

A classe *MainActivity.java* deve implementar os métodos *salvarConfiguracao()* e *abrirConfiguracao()*.

```
private void salvarConfiguracao() {  
    if (this.arquivo != null) {  
        Configuracao configuracao = this.arquivo.criarConfiguracao();  
        Repositorio repositorio = new Repositorio( context: this);  
        repositorio.salvar(configuracao);  
    }  
}  
  
private void abrirConfiguracao() {  
    if (this.arquivo == null) {  
        Repositorio repositorio = new Repositorio( context: this);  
        Configuracao configuracao = repositorio.abrir();  
        if (configuracao != null) {  
            Uri uri = Uri.parse(configuracao.getUri());  
            this.arquivo = ArquivoPDF.abrir( context: this, pdfView, uri, configuracao.getPagina());  
        }  
    }  
}
```


3

Além disso, a classe *MainActivity* deve invocar o método *salvarConfiguracao()* nos listeners de página anterior e posterior.

Deve também invocar o método *abrirConfiguracao()* no final do método *onCreate()*

```
abrirConfiguracao();
```

```
fabPaginaAnterior = (FloatingActionButton)
    findViewById(R.id.fab_pagina_anterior);
fabPaginaAnterior.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (arquivo != null) {
            arquivo.paginaAnterior();
            salvarConfiguracao();
        }
    }
});
```

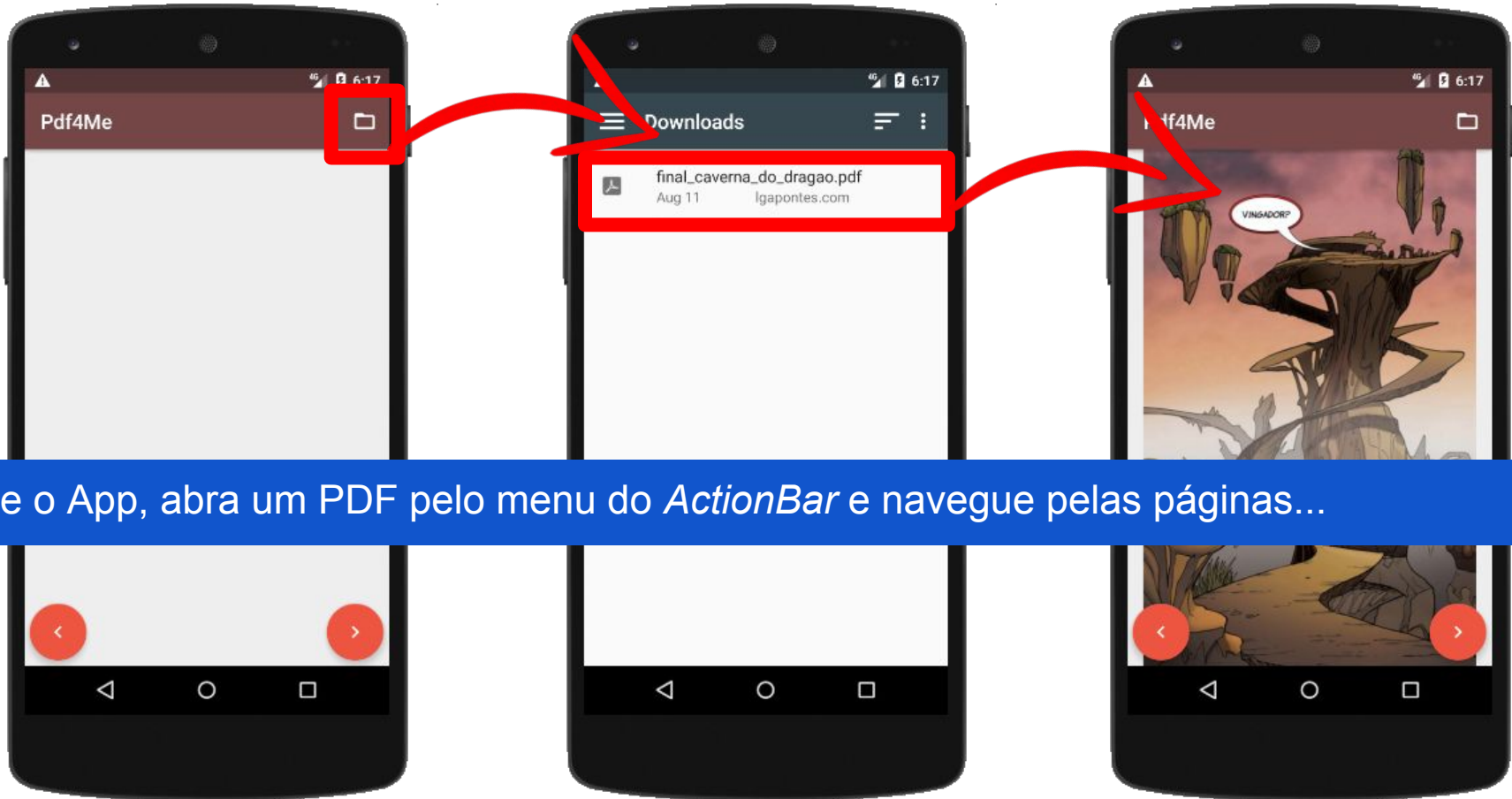
```
fabPaginaPosterior = (FloatingActionButton)
    findViewById(R.id.fab_pagina_posterior);
fabPaginaPosterior.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (arquivo != null) {
            arquivo.paginaPosterior();
            salvarConfiguracao();
        }
    }
});
```

3

Por fim, deve invocar o método `salvarConfiguracao()` dentro do `onActivityResult()` - que será executado quando o *Framework Access Storage* retornar uma URI de um arquivo.

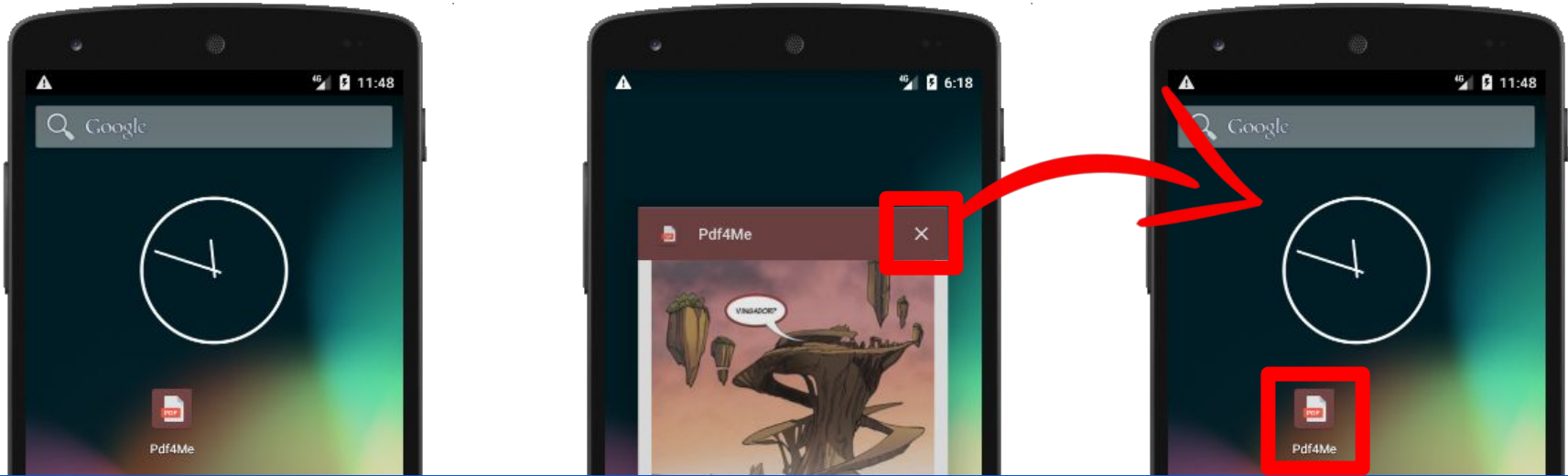
```
public void onActivityResult(int requestCode, int resultCode, Intent resultData) {  
    if (resultCode == Activity.RESULT_OK) {  
        if (requestCode == CODIGO_ABRIR_ARQUIVO) {  
            if (resultData != null) {  
                Uri uriArquivo = resultData.getData();  
                this.arquivo = ArquivoPDF.abrir(context: this, pdfView, uriArquivo);  
                salvarConfiguracao();  
            }  
        }  
    }  
}
```

Testando o App Pdf4Me

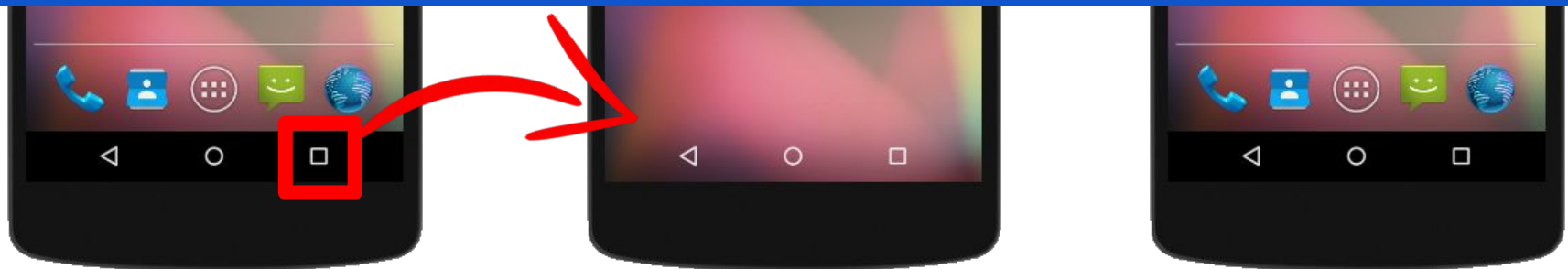


Rode o App, abra um PDF pelo menu do *ActionBar* e navegue pelas páginas...

Testando o App Pdf4Me



Feche o App (inclusive no menu de contexto) e depois execute novamente pelo Launcher...



Testando o App Pdf4Me

Veja que o App iniciará com o PDF aberto na mesma página em que ele foi finalizado.



SharedPreferences

```
"pdf4me": {  
  "uri": "content://downloads/all_downloads/5",  
  "pagina": 10  
}
```



Obrigado!