

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Trabalho Prático (Avaliação Contínua) Curso (DWDM) Desenvolvimento para a Web e para Dispositivos Móveis Unidade Curricular (PDM1) Programação para Dispositivos Móveis I	Ano letivo 2024/2025
--	---	--------------------------------

Observações V1.0 – Versão inicial

1. Destinatários

Este trabalho destina-se a todos os estudantes inscritos na unidade curricular de Programação para Dispositivos Móveis I (PDM I) e que tenham optado pelo regime de Avaliação Contínua, que decorrerá durante o semestre letivo. A opção pela Avaliação Contínua obriga a uma assiduidade em pelo menos 2/3 das aulas ao longo do semestre. Esta escolha é efetivada a partir do momento em que o aluno se associe a um dos grupos de trabalho (através do Moodle).

Os estudantes deverão juntar-se em grupos de 2 a 3 elementos de modo a dividir, da melhor forma, as tarefas definidas neste trabalho. O trabalho tem um peso de 100% na classificação final de PDMI e tem como requisito mínimo uma classificação de 9,5 valores. O trabalho é composto por 2 entregas (*milestones*: M1 e M2, com peso de 40% e 60%, respetivamente), onde em cada entrega o aluno deverá obter um mínimo de 7,5 valores. Os elementos de um mesmo grupo podem ter classificações diferentes.

As datas e procedimentos de cada entrega estarão devidamente assinaladas no Moodle.

2. Objetivos

Este projeto funcionará como um elemento integrador dos conhecimentos adquiridos nas Unidade Curricular de Programação para Dispositivos Móveis I.

Cada grupo/equipe deverá trabalhar sobre o tema definido neste enunciado, especificar um conjunto de funcionalidades, e implementá-las na plataforma Android. Apesar do tema não ter um caráter livre, os alunos devem trabalhar no sentido de completar o tema descrito.

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Trabalho Prático (Avaliação Contínua) Curso (DWDM) Desenvolvimento para a Web e para Dispositivos Móveis Unidade Curricular (PDM1) Programação para Dispositivos Móveis I	Ano letivo 2024/2025
--	---	--------------------------------

Os objetivos específicos serão os seguintes:

- Especificar e coordenar um projeto em grupo de pequena dimensão;
- Compreender e dominar os conhecimentos teóricos e práticos sobre desenvolvimento de aplicações móveis na plataforma Android;
- Adquirir competências com vista à resolução de problemas, nomeadamente através da pesquisa e utilização autónoma de conteúdos e ferramentas externas;
- Estimular o trabalho em equipa como elemento essencial do processo de aprendizagem individual.

3. Enunciado

O tema do trabalho prático prende-se com o desenvolvimento de uma aplicação relacionada com jogos de vídeo, consultando a [RAWG Video Games Database API](#)¹. Pode utilizar outra API desde que tenha o mesmo contexto.

A aplicação deverá listar os jogos de vídeo disponibilizados pela API, incluindo imagem, *ratings*, plataformas suportadas e número de *reviews*, com suporte para paginação. Deve também possibilitar a filtragem por género de jogo, plataformas e lojas. O tamanho de cada página (nº de jogos em cada pedido) deve ser configurável nas *settings* da aplicação.

A vista de detalhe de cada jogo deve conter informação útil aos utilizadores, como descrição do jogo, galeria de imagens, comentários, vídeos do Twitch ou Youtube, requisitos de hardware, entre outros dados que considere relevantes.

A aplicação deverá fazer cache de todos os dados que obtiver através da API, de forma a minimizar o consumo de dados utilizados, quer seja na obtenção da listagem dos jogos, quer esteja relacionado com o detalhe dos mesmos. Para isso, deverá estipular um período de validade desta cache, a partir do

¹ <https://rawg.io/apidocs>

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Trabalho Prático (Avaliação Contínua) Curso (DWDM) Desenvolvimento para a Web e para Dispositivos Móveis Unidade Curricular (PDM1) Programação para Dispositivos Móveis I	Ano letivo 2024/2025
--	---	--------------------------------

qual o conteúdo expirado deve ser eliminado, que deve ser configurável das definições da aplicação.

O utilizador deve ser notificado diariamente com um jogo de vídeo selecionado aleatoriamente, que deverá permitir abrir a vista de detalhe do mesmo. A hora preferencial para lançar esta notificação deve ser definida das definições da aplicação. Contudo, deve existir a possibilidade de solicitação de um jogo aleatório que, além de poder ser aplicável a outras funcionalidades, deve lançar a notificação descrita anteriormente.

A aplicação deve disponibilizar aos utilizadores uma *wishlist* (lista de jogos pretendidos), que manterá uma lista local ou na *cloud* dos jogos assinalados como tal. Deve também prever a funcionalidade de retirar um jogo dessa lista. Deve também permitir a partilha de um jogo de vídeo nas suas redes sociais (por exemplo, Facebook, Instagram, Twitter, etc.).

Deve considerar a utilização de pelo menos um sensor do dispositivo para uma funcionalidade que defina para a aplicação (proximidade, luz ambiente, acelerómetro, giroscópio, GPS, etc.)

Como um requisito de bonificação, pode acrescentar uma secção com “A descoberta de hoje”, que mostrará diariamente um jogo de vídeo ainda não visualizado pelo utilizador. Outro requisito de bonificação assenta na possibilidade de atribuir progresso/*achievements* ao utilizador, seja por tamanho da *wishlist* (por exemplo, 1º jogo na *wishlist*, 5 jogos, 10 jogos ou 20 jogos), por visualizações dos jogos sugeridos, ou outros.

Recomenda-se que os alunos explorem o tema aqui especificado e enriquecendo-o. Às funcionalidades aqui enunciadas, as equipas de trabalho devem acrescentar outras que façam sentido para o problema enunciado e valorizado na avaliação do projeto.

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Trabalho Prático (Avaliação Contínua) Curso (DWDM) Desenvolvimento para a Web e para Dispositivos Móveis Unidade Curricular (PDM1) Programação para Dispositivos Móveis I	Ano letivo 2024/2025
--	---	--------------------------------

Na secção 4 são apresentados alguns serviços que podem servir para estender as funcionalidades aqui descritas, mas os grupos de trabalho têm a liberdade de identificarem mais funcionalidades a implementar na aplicação.

4. Realização do Trabalho Prático

O trabalho prático consiste na exploração do tema proposto pelos docentes da Unidade Curricular. O desenvolvimento do trabalho proposto será obrigatoriamente na plataforma Android com recurso à ferramenta Android Studio lecionada nas aulas.

A ordem de trabalhos proposta para os alunos da Unidade Curricular é a seguinte:

1. Escolha dos grupos através do Moodle;
2. Cada grupo especifica e propõe um grupo de funcionalidades a implementar de acordo com o tema descrito no enunciado;
3. Após uma clara especificação do problema o grupo passa à implementação das funcionalidades propostas.

Durante a análise e especificação do problema, cada grupo de trabalho deverá ter atenção ao tema proposto e aos requisitos obrigatórios e de bonificação. Entenda-se que os requisitos obrigatórios devem estar presentes em todos os projetos desenvolvidos enquanto os requisitos de bonificação serão apenas sugestões que os alunos podem seguir.

Podem também propor novos elementos de bonificação desde que documentados no relatório final.

4.1. Requisitos Obrigatórios

Os projetos deverão obrigatoriamente cumprir os seguintes requisitos:

- Suporte para ecrãs de diferentes dimensões (Telemóvel e Tablet);
- Uso de fragmentos (Fragments);
- Uso de listas (RecyclerView e Adapters);
- Uso de base de dados (Room/Firebase Firestore);
- Uso de operações assíncronas (AsyncTask/Thread/IntentService);

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Trabalho Prático (Avaliação Contínua) Curso (DWDM) Desenvolvimento para a Web e para Dispositivos Móveis Unidade Curricular (PDM1) Programação para Dispositivos Móveis I	Ano letivo 2024/2025
--	---	--------------------------------

- Uso de notificações (ex.: veículo aleatório para visualizar/analisar...);
- Uso das *guidelines* do material design;
- Integração com a [RAWG Video Games Database API](#)² via pedidos REST;
- Utilização de sensores.

4.2. Requisitos de Bonificação

Para bonificação dos trabalhos devem ser tidos em consideração os seguintes elementos:

- Uso *web services* adicionais via pedidos REST;
- Gestão de um sistema de progresso/*achievements* dos utilizadores da aplicação;
- Gestão de um sistema de descoberta de jogos para os utilizadores da aplicação;
- Interacção com elementos do Android (Contacts, Messages, etc.).

4.3. Implementação

Para a implementação das soluções ao problema proposto devem ser tomadas em consideração as seguintes orientações:

- Deverá ser usada a linguagem de desenvolvimento para Android (Java) e o ambiente de programação Android Studio para o desenvolvimento da solução;
- Podem ser utilizadas bibliotecas externas desde que documentadas no relatório do projeto e que não prejudiquem o cumprimento dos objetivos de aprendizagem e requisitos obrigatórios definidos;
- Podem ser utilizados *web services* disponíveis na Internet para completar o tema proposto com novas funcionalidades.

4.4. Elaboração do Relatório

Esta tarefa consiste na escrita de um relatório que descreva todo o trabalho realizado, que mostre e comente os resultados obtidos e que apresente as respetivas conclusões. Deve ser justificada a utilização dos vários componentes Android, bibliotecas externas utilizadas, funcionamento da

² <https://rawg.io/apidocs>

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Trabalho Prático (Avaliação Contínua) Curso (DWDM) Desenvolvimento para a Web e para Dispositivos Móveis Unidade Curricular (PDM1) Programação para Dispositivos Móveis I	Ano letivo 2024/2025
--	---	--------------------------------

aplicação e qualquer informação adicional e relevante para o projeto.

Para a elaboração do relatório do projeto, deverá ser utilizado o *template* disponibilizado na página do moodle da UC, sendo que a definição da estrutura do documento fica à responsabilidade de cada grupo. Considere que, pelo menos, deverá constar uma especificação das funcionalidades, as decisões de implementação e uso de recursos auxiliares como bibliotecas externas no desenvolvimento do projeto.

É esperado que o relatório contenha, entre outros:

- Visão do produto e Análise do Problema;
- Requisitos do projeto;
- Detalhes de implementação;
- Descrição de funcionalidades obrigatórias e de bonificação.

4.5. Calendarização O trabalho terá **2 entregas (milestones): M1 e M2**. Em cada um destes momentos, os grupos devem efetuar a submissão do trabalho desenvolvido, como *source code* e relatório, nas datas designadas no Moodle. As apresentações e as defesas dos trabalhos decorrerão nas datas indicadas no Moodle, durante o período das aulas TP/PL.

Em seguida, descrevem-se os elementos/funcionalidades a entregar/implementar para cada entrega (milestone):

4.5.1. Milestone M1

Esta entrega terá um **peso 40%**, com uma nota mínima de **7,5 valores**, estando previsto o desenvolvimento de:

- Versão inicial do relatório.
 - Visão do produto e Análise do problema;
 - Requisitos do projeto;
 - Funcionalidades de bonificação previstas;
 - *Mockups* das interfaces gráficas.

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Trabalho Prático (Avaliação Contínua) Curso (DWDM) Desenvolvimento para a Web e para Dispositivos Móveis Unidade Curricular (PDM1) Programação para Dispositivos Móveis I	Ano letivo 2024/2025
--	---	--------------------------------

- Versão inicial da aplicação
 - Definição do *App icon* e do *Splash screen*;
 - Navegação e organização geral da aplicação (AppBar, Toolbars, Navigation drawer, Bottom navigation...);
 - Definições/settings da aplicação (por exemplo, tema preferido, nº de jogos por pedido, hora preferencial conhecer novo jogo, etc.);
 - Configuração da Room database, definição dos DAO (Data Access Objects) e queries necessárias já identificadas;
 - Configuração e lançamento de notificações.

4.5.2. Milestone M2 Esta entrega terá um **peso 60%**, com uma nota mínima de **7,5 valores**, estando previsto o desenvolvimento de:

- Versão final do relatório
 - Detalhes de implementação;
 - Descrição de funcionalidades obrigatórias e de bonificação.
- Versão final da aplicação
 - Implementação/melhoria das funcionalidades obrigatórias propostas;
 - Implementação das funcionalidades de bonificação previstas.

5. Defesa

No processo de defesa o grupo poderá utilizar um computador próprio ou um dos computadores do laboratório da sala onde decorre a unidade curricular. O uso de um dispositivo Android para demonstração da aplicação é opcional e será sempre complementar ao uso de um computador. A execução do projeto é da exclusiva responsabilidade do grupo.

A cada elemento do grupo poderá ser pedido a resposta a questões sobre os conteúdos da unidade curricular e em específico de implementação no trabalho prático. Adicionalmente, poderão ser pedidas alterações ou a implementação de novas funcionalidades, de forma a demonstrar os

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova	Trabalho Prático (Avaliação Contínua)	Ano letivo 2024/2025
	Curso	(DWDM) Desenvolvimento para a Web e para Dispositivos Móveis	
	Unidade Curricular	(PDM1) Programação para Dispositivos Móveis I	

conhecimentos.

Elementos de um mesmo grupo poderão ter notas diferentes, consoante o seu desempenho na apresentação e na defesa.

6. Critérios de Avaliação

A nota de cada aluno será calculada com base em três componentes principais:

1. Desempenho individual durante a defesa;
2. Qualidade do projeto;
3. Qualidade do relatório.

Para avaliar a componente de desempenho individual durante a defesa serão colocadas perguntas teóricas ou solicitada a implementação ou alteração de algumas funcionalidades relativas ao seu trabalho, a reimplementação de funcionalidades já existentes ou questões genéricas sobre o projeto.

Serão avaliados o desempenho do aluno e o seu conhecimento dos conteúdos abordados na Unidade Curricular.

Para avaliar a componente qualidade do projeto, serão tidos em conta os seguintes critérios:

- Estruturação da aplicação e uso de componentes da plataforma Android;
- Qualidade e complexidade da aplicação desenvolvida;
- Qualidade da implementação dos requisitos obrigatórios;
- Qualidade da implementação dos requisitos de bonificação;
- Utilidade das funcionalidades implementadas;
- Utilização dos conteúdos lecionados em aula.

A componente do relatório será avaliada de acordo com a clareza, objetividade e detalhe do relatório.

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Trabalho Prático (Avaliação Contínua) Curso (DWDM) Desenvolvimento para a Web e para Dispositivos Móveis Unidade Curricular (PDM1) Programação para Dispositivos Móveis I	Ano letivo 2024/2025
--	---	--------------------------------

A nota final do aluno será a média ponderada dos resultados obtidos em cada entrega/*milestone*, salvaguardando uma nota mínima de 9,5 valores.

7. Avaliação pelos Pares

Cada grupo deverá realizar uma análise coletiva sobre o contributo e esforço dos recursos humanos envolvidos (que deverá ser anexada ao documento que representa a proposta de solução). Desta análise deverá ser possível identificar as contribuições realizadas por cada elemento do grupo e, se as mesmas, estão acima, na média, ou abaixo face às contribuições realizadas pelos restantes elementos.

Está previsto a atribuição de 5% da avaliação (um valor) para cada aluno de acordo com a sua contribuição individual no desenvolvimento da proposta de solução.

8. Código de Conduta

Os autores do trabalho deverão declarar terem atuado com integridade e não terem recorrido às práticas de plágio nem auto-plágio, falsificação de resultados ou qualquer outra prática que desrespeite o código de conduta do Politécnico do Porto.

9. Outras Informações

A deteção de trabalhos fraudulentos, em parte ou na totalidade inviabiliza a avaliação dos mesmos. Neste caso, será anulada a avaliação a todos os elementos do grupo de trabalho.

Programação Para Dispositivos Móveis I

INTENTS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, [cfpa@estg.ipp.pt](mailto:cfpap@estg.ipp.pt)

Índice

- Intents;
- Composição do Intent;
- Casos de uso dos Intents;
- Broadcast;
- Pending Intents;
- Leitura Adicional.

Intent

Intent representa uma descrição abstrata de uma operação a ser executada na plataforma Android;

- É apenas uma **estrutura de dados que descreve uma operação a ser efetuada**;
- Também é visto como **uma mensagem** que é enviada ao sistema com instruções sobre operações a realizar;

Três dos componentes mais importantes do Android (Activity, Service e BroadcastReceivers) são iniciados através de mensagens (Intents).

Intent (2)

- Iniciar uma Activity

```
startActivity(Intent)
```

- Iniciar um Service

```
startService(Intent)
```

- Enviar broadcast

```
sendBroadcast(Intent)
```

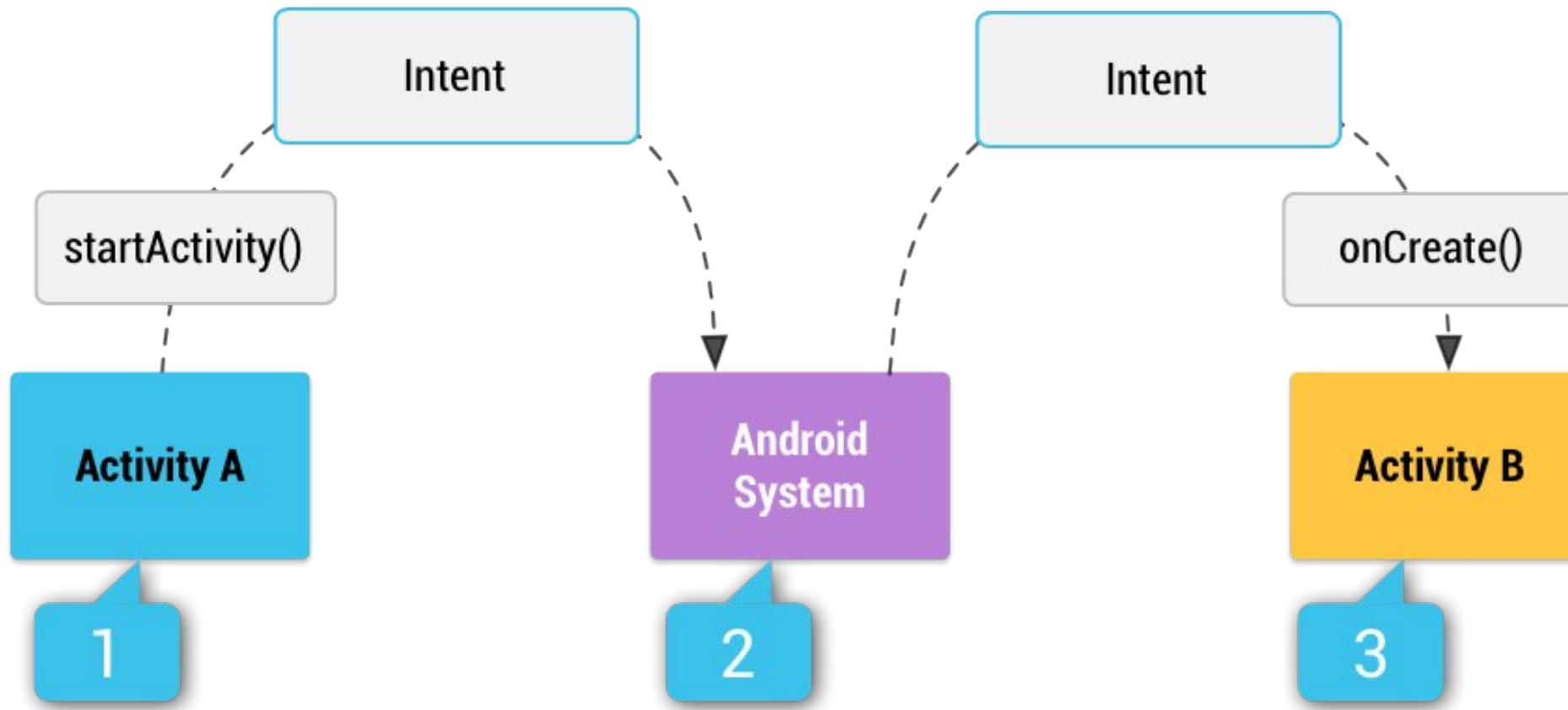
Para cada caso o Android encontra a Activity, Service ou BroadcastReceiver apropriado para receber o Intent e se necessário inicializa esse objeto.

Intent (2)

Futuramente iremos abordar os diferentes usos de Intents, mas neste momento iremos utilizar Intents apenas para:

- Iniciar atividades;
- Interagir com elementos da plataforma Android;
- Interagir com outras aplicações na plataforma Android.

Intent



Fonte: <https://developer.android.com/guide/components/intents-filters>

Composição de um Intent

Um Intent pode ser constituído pelos seguintes parâmetros:

- **Ação:** Uma String que representa a ação a ser desencadeada;
- **URI:** Identificador que representa as especificações do componente a inicializar;
- **Contexto:** Contexto da aplicação que contém a classe;
- **Classe:** Classe que o sistema terá de encontrar e instanciar.

Composição de um Intent

Passagem de Argumentos

Opcionalmente podemos com os nossos Intents passar argumentos via:

- Bundles;
- Extras.

No fundo são objetos semelhantes onde os argumentos são armazenados um a um ou todos de uma só vez.

Uso de Intents

Iniciar nova Atividade

Iniciar uma nova atividade e passar argumentos de uma atividade para outra.

Intent(Context, Classe)

- Contexto em que se insere a classe a inicializar;
- Classe a inicializar pelo Android;

Iniciar uma nova Activity com INTENT

```
Intent mIntent = new Intent(this, NewActivity.class);
mIntent.putExtra("KEY", "VALUE");
startActivity(mIntent);
```

Uso de Intents

Iniciar Aplicações -> Realizar uma chamada

Intent(Action, URI)

- A string refere-se à ação a ser realizada;
- O URI refere-se a um objeto de dados de um determinado tipo.

Realizar uma chamada através de um INTENT

```
Intent mIntent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:+351255314002"));
startActivity(mIntent);
```

Uso de Intents

Iniciar Aplicações -> Visualizar um contacto existente

Intent(Action, URI)

- A string refere-se à ação a ser realizada;
- O URI refere-se a um objeto de dados de um determinado tipo.

Visualizar contacto existente através de um INTENT

```
Intent mIntent = new Intent(Intent.ACTION_VIEW, Uri.parse("content://contacts/people/1"));
startActivity(mIntent);
```

Uso de Intents

Iniciar Aplicações -> Visualizar localização no mapa

Intent(Action, URI)

- A string refere-se à ação a ser realizada;
- O URI refere-se a um objeto de dados de um determinado tipo.

Visualizar localização no mapa através de um INTENT

```
Intent mIntent = new Intent(Intent.ACTION_VIEW, Uri.parse("geo:41.366788, -8.194751"));  
startActivity(mIntent);
```

Uso de Intents

Iniciar Aplicações -> Iniciar direções no mapa

Intent(Action, URI)

- A string refere-se à ação a ser realizada;
- O URI refere-se a um objeto de dados de um determinado tipo.

Iniciar direções no mapa através de um INTENT

```
Intent mIntent = new Intent(Intent.ACTION_VIEW,Uri.parse("google.navigation:q=Felgueiras"));  
startActivity(mIntent);
```

Uso de Intents

Iniciar Aplicações -> Extra

Mais casos de uso de Intents com aplicações do sistema Android podem ser encontradas em:

<https://developer.android.com/guide/components/intents-common>

Uso de Intents

Broadcast Intent

Intent(String)

- A String refere-se à ação a ser realizada.

Intent utilizado para enviar por Broadcast

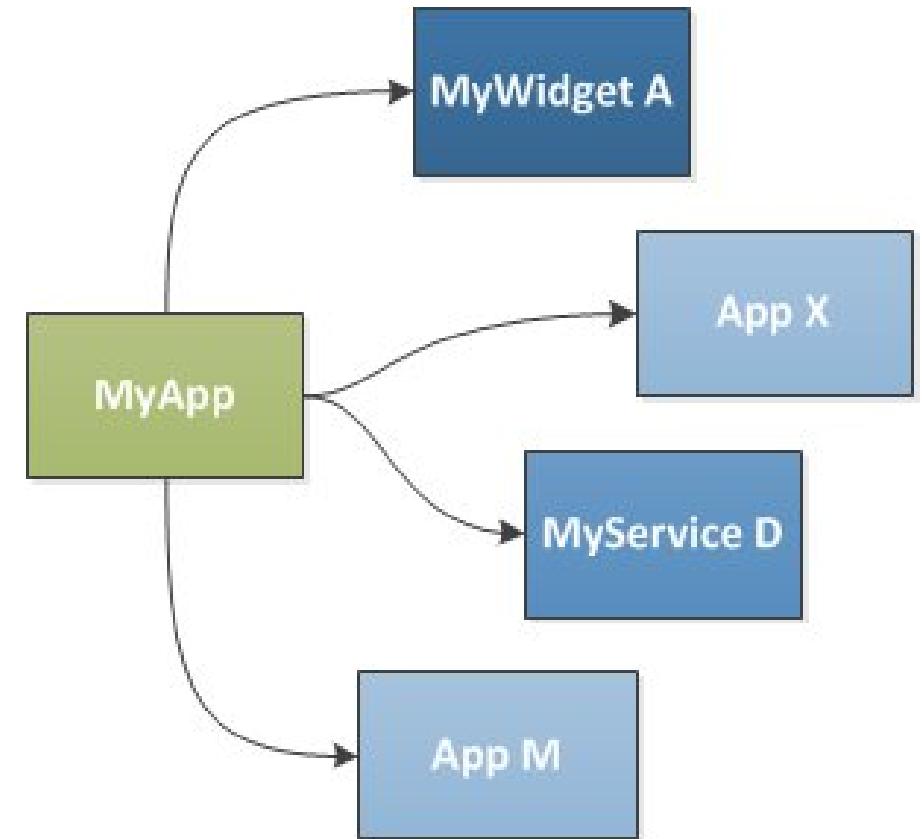
```
Intent mIntent = new Intent("pt.ipp.estg.pdm.DOWNLOAD_FINISHED");  
sendBroadcast(mIntent);
```

Broadcast

Utilizado para comunicar informações entre aplicações.

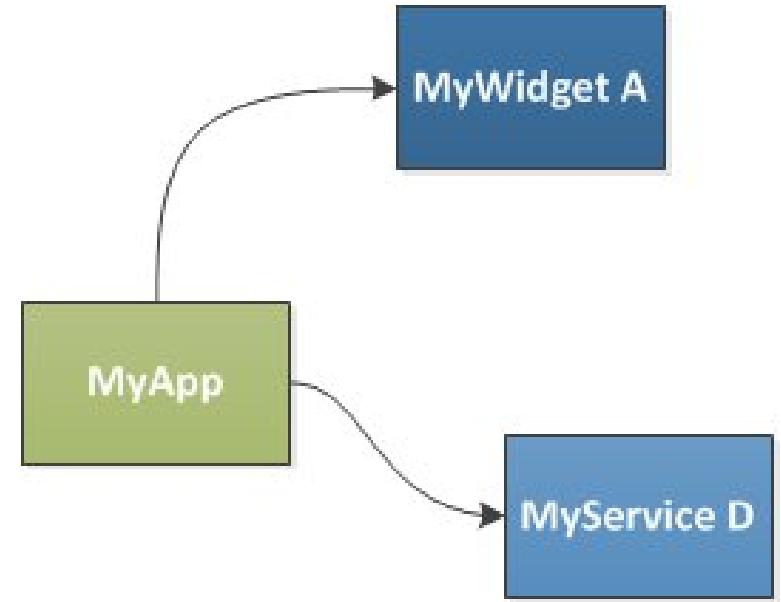
O método `Context.sendBroadcast()` envia um Intent para os BroadcastReceivers registrados no sistema (que recebem esse mesmo Intent)

- `sendBroadcast(Intent intent)`



Broadcast (2)

Caso não seja necessário enviar Intents para o sistema, mas sim apenas entre os vários componentes da aplicação deve-se utilizar o método



```
LocalBroadcastManager.getInstance(Context).sendBroadcast(Intent)
```

Broadcast

Broadcast Receiver

Componente que é utilizado para receber Intents enviados pelo método `sendBroadcast()`.

Estes têm de ser registados no `AndroidManifest.xml` ou dinamicamente no código Java utilizando o método
`Context.registerReceiver(BroadcastReceiver, IntentFilter)`

ou

`LocalBroadcastManager.getInstance(Context).registerReceiver(BroadcastReceiver, IntentFilter)`

Estes receivers devem ser registados no método `onResume()` da Activity ou Fragment.

No método `onPause()`, da Activity ou Fragment, deve-se remover esse registo
`unregisterReceiver(BroadcastReceiver receiver)`

Broadcast

Broadcast Receiver -> Intent Filter

Componente que é utilizado para informar o sistema operativo quais os Intents que o BroadcastReceiver vai tratar.

Definição em XML

```
<action android:name="pt.ipp.estg.pdm1.exemplo.mybroadcast" />
```

Definição em Java

```
IntentFilter mIF = new IntentFilter();
mIF.addAction("pt.ipp.estg.pdm1.exemplo.mybroadcast");
mIF.addAction("pt.ipp.estg.pdm1.exemplo.ourbroadcast");
```

Broadcast

Broadcast Receiver

É necessário criar uma nova classe do tipo Broadcast Receiver onde será também criado o método `onReceive()`

O método `onReceive()` **não foi desenhado** para realizar operações **assíncronas e pesadas**.

Considera-se como um processo em **foreground** e é mantido com uma **prioridade elevada**.

```
public class MyReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        // Código a implementar  
    }  
}
```

PendingIntent

Objeto para ser **utilizado por outras aplicações** para executarem uma ação descrita por nós;

- Ao passar o PendingIntent à outra aplicação, estamos-lhe a **garantir** o direito de executar operações conforme fosse a nossa aplicação (com as mesmas permissões e identidade);

O PendingIntent é simplesmente uma **referência para um token** que é mantido pelo sistema para descrever os dados que devem ser retornados;

- Mesmo que o processo da aplicação termine, o PendingIntent **é mantido no sistema para ser utilizado por outras aplicações.**

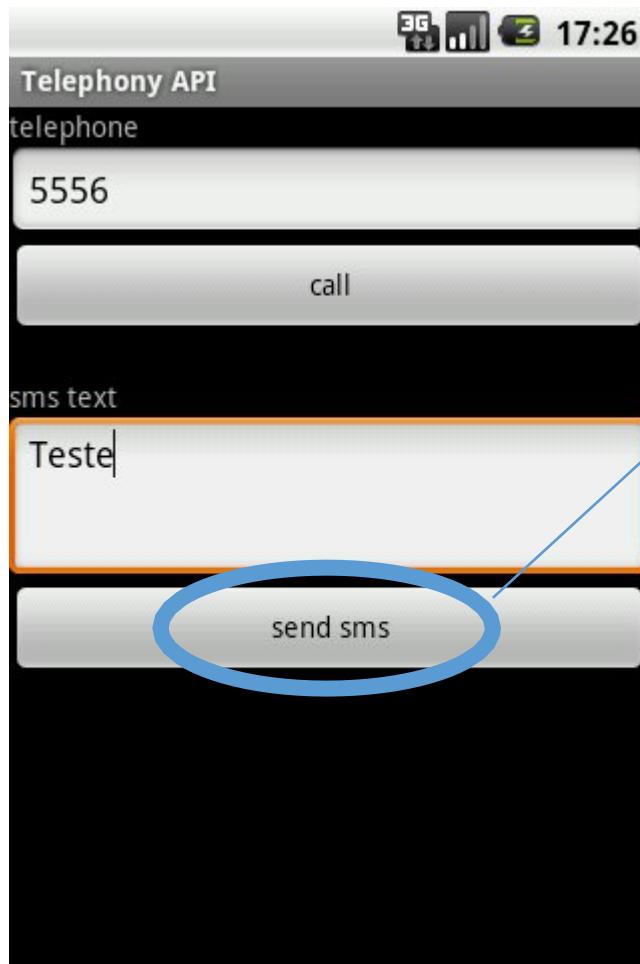
PendingIntent

O PendingIntent pode ser inicializado como

- `getActivity(Context, int [não utilizado], Intent, int);`
- `getService(Context, int [não utilizado], Intent, int);`
- `getBroadcast(Context, int [não utilizado], Intent, int);`

```
Intent mIntent = new Intent("STRING");
mIntent.putExtra("KEY", "VALUE");
PendingIntent mPI = getBroadcast(getApplicationContext(), 0, mIntent, 0);
```

PendingIntent



```
SmsManager.sendTextMessage(  
    NUMERO,  
    null,  
    TEXTO,  
    PendingIntent sent,  
    PendingIntent delivery)
```

Quando a SMS for
enviada envia-me
o Intent mIntent

sendBroadcast(mIntent)

Leitura Adicional

- **Intents:**

<https://developer.android.com/guide/components/intents-filters#Receiving>

- **Common Intents:**

<https://developer.android.com/guide/components/intents-common>

- **Intents Google Maps:**

<https://developers.google.com/maps/documentation/urls/android-intents>

Programação Para Dispositivos Móveis I

INTENTS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, [cfpa@estg.ipp.pt](mailto:cfpap@estg.ipp.pt)

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

ListView Adapters

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

ListView

É um grupo de **views** que apresenta uma **lista** através de uma **fonte de dados** (por ex. um array ou um objeto).

No seu funcionamento, apresenta um conjunto de items idênticos, que partilham o mesmo tipo de layout (e respetivas configurações).

<https://developer.android.com/reference/android/widget/ListView>

CountryList

Name: Angola
Continentes: Africa

Name: Argentina
Continentes: America

Name: Brazil
Continentes: America

Name: Chile
Continentes: America

Name: Cuba
Continentes: America

Name: Egypt
Continentes: Africa

Name: Germany
Continentes: Europe

Name: Greece
Continentes: Europe

ListView

main_activity.xml

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

    <ListView
        android:id="@+id/list_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:entries="@array/array_countries" />

</LinearLayout>
```

No ficheiro de layout onde desejarmos incluir a nossa list, deveremos inserir o elemento ListView.

Na propriedade android:entries é possível definir a fonte de dados a ser preenchida na lista. Esta fonte de dados está explica no ficheiro strings.xml

ListView

strings.xml

```
<string-array name="array_countries">
    <item>Portugal</item>
    <item>United Kingdom</item>
    <item>Mexico</item>
    <item>Argentina</item>
    <item>Brazil</item>
    <item>Egypt</item>
    <item>Sidney</item>
    <item>Greece</item>
    <item>Japan</item>
    <item>Istanbul</item>
    <item>South Africa</item>
    <item>Thailand</item>
</string-array>
```

Exemplo de uma entrada no ficheiro strings.xml que contém um array de strings composto pelos nomes de diferentes países.

ListView

onItemClick()

```
private ListView listView;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    listView = (ListView) findViewById(R.id.list_view);  
  
    listView.setOnItemClickListener(this);  
}  
  
@Override  
public void onItemClick(AdapterView<?> parent,  
                        View view,  
                        int position,  
                        long id) {  
  
    Log.i("ITEM_CLICKED", listView.getItemAtPosition(position).toString());  
}
```

Cada elemento da nossa lista (item) pode sofrer interações por parte do utilizador.

À semelhança do método `onClickListener()`, as `ListViews` tem uma propriedade para verificar que item sofreu um click, o `onItemClickListener`.

A activity deverá implementar a interface `AdapterView.OnItemClickListener` que lhe irá indicar para a implementação do método `onItemClick`

ListView

Conteúdo dinâmico

A inclusão da propriedade `android:entries` e respetiva caracterização no ficheiro `strings.xml` através de um string-array, é limitativa e apenas contempla **conteúdo estático**.

Se pretendemos popular as nossas listas com **conteúdo dinâmico**, é necessário a inclusão de um **adapter**.

Adapters

Os **adapters** são responsáveis pelo **tratamento e apresentação dos dados**. Estabelecem uma ponte entre um AdapterView (ex. ListView, Spinner) e os dados que serão preenchidos nessa View.

Permitem a gestão de grandes conjuntos de dados de uma maneira **eficiente** e **escalável**.

Constroem o layout para cada um dos elementos da lista (requer a criação de um layout adicional que representa como cada elemento da lista será apresentado).

ListView (com adapter)

main_activity.xml

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

    <ListView
        android:id="@+id/list_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Com a utilização de adapters (e conteúdo dinâmico) removemos a propriedade android:entries. A gestão do conteúdo da lista será feita de forma dinâmica.

ListView (com adapter)

list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/txt_item"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:textSize="45dp"
    android:layout_marginLeft="12dp"
    android:layout_marginTop="5dp">

</TextView>
```

Com a utilização de adapters (e conteúdo dinâmico) é necessário definir como cada item da lista será apresentado.

Isto é feito através da criação de um novo layout que contém as definições de um elemento da lista. Este layout será replicado por todos os elementos da lista.

ListView (com adapter)

Adapter

```
private ListView listView;
private String[] itemList;
private ArrayAdapter<String> adapter;
```



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    listView = (ListView) findViewById(R.id.list_view);
    listView.setOnItemClickListener(this);

    itemList = getResources().getStringArray(R.array.array_countries);

    adapter = new ArrayAdapter<String>
        (this, R.layout.list_item, R.id.txt_item, itemList);

    listView.setAdapter(adapter);
}
```

Lista que irá ser preenchida com o conteúdo do array de strings em strings.xml

Definição do adapter

Preencher o array com o conteúdo do array de strings em strings.xml

Inicialização do adapter, com o layout que deverá utilizar para cada item, e respetivos dados

Associação do adapter à ListView

ListView (com adapter)

onItemClick()

```
@Override  
public void onItemClick(AdapterView<?> parent,  
    View view,  
    int position,  
    long id) {  
  
    adapter.getItem(position); ←  
    Log.i("ITEM_CLICKED", listView.getItemAtPosition(position).toString());  
}
```

Com a utilização do adapter podemos aceder ao item que foi clicado através do método getItem()

Spinner

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);

// Criar um ArrayAdapter utilizando um array de strings e o layout predefinido da
spinner

ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this, R.array.data,
android.R.layout.simple_spinner_item);

// Especificar o layout a utilizar quando a lista aparece
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

Custom ArrayAdapter

CustomAdapter.java

```
public class CustomAdapter extends ArrayAdapter<E>

public CountryListAdapter(Activity context, E[] itemList){
    super(context, R.layout.list_item, itemList);
    this.context = context;
    this.itemList = itemList;
}
```

- Quando pretendemos atualizar e apresentar uma **lista de objetos complexos com um layout personalizado** devemos utilizar um **ArrayAdapter**.
- O tipo <E> deverá ser substituído pelo tipo de dados que queremos utilizar (ex. uma classe);
- No construtor devemos especificar (para a classe superior) qual o layout a utilizar e passar os objetos (dados) a serem apresentados na interface.

Custom ArrayAdapter

CustomAdapter.java

```
public View getView(int position, View view, ViewGroup parent){  
    LayoutInflator layoutInflater = context.getLayoutInflater();  
    View itemRowView = layoutInflater.inflate(R.layout.list_item, null, true);  
  
    txtView = (TextView) itemRowView.findViewById(R.id.txt_view);  
  
    txtView.setText(itemList[position].getName());  
  
    return itemRowView;  
}
```

- Sempre que a interface necessita de ser atualizada o método `getView()`, da classe do ArrayAdapter, é invocado para **cada item** da lista.

Gestão dos Adapter e ListView

- Se uma ListView consegue apresentar em simultâneo apenas 5 elementos, nunca deverá instanciar mais do que 5 views, mesmo que na estrutura de dados existam 100 elementos;
- As views utilizadas para cada item, serão sempre reaproveitadas quando é efetuado o scroll/drag da lista;

`getView (int position, View view, ViewGroup parent)`

- `position`: o índice do elemento da lista a apresentar
- `view`: se != null, contém um layout previamente instanciado, e que deve ser reutilizado
- `parent`: contentor das views correspondentes a elementos da lista

Programação Para Dispositivos Móveis I

ListView Adapters

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

RecyclerView

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

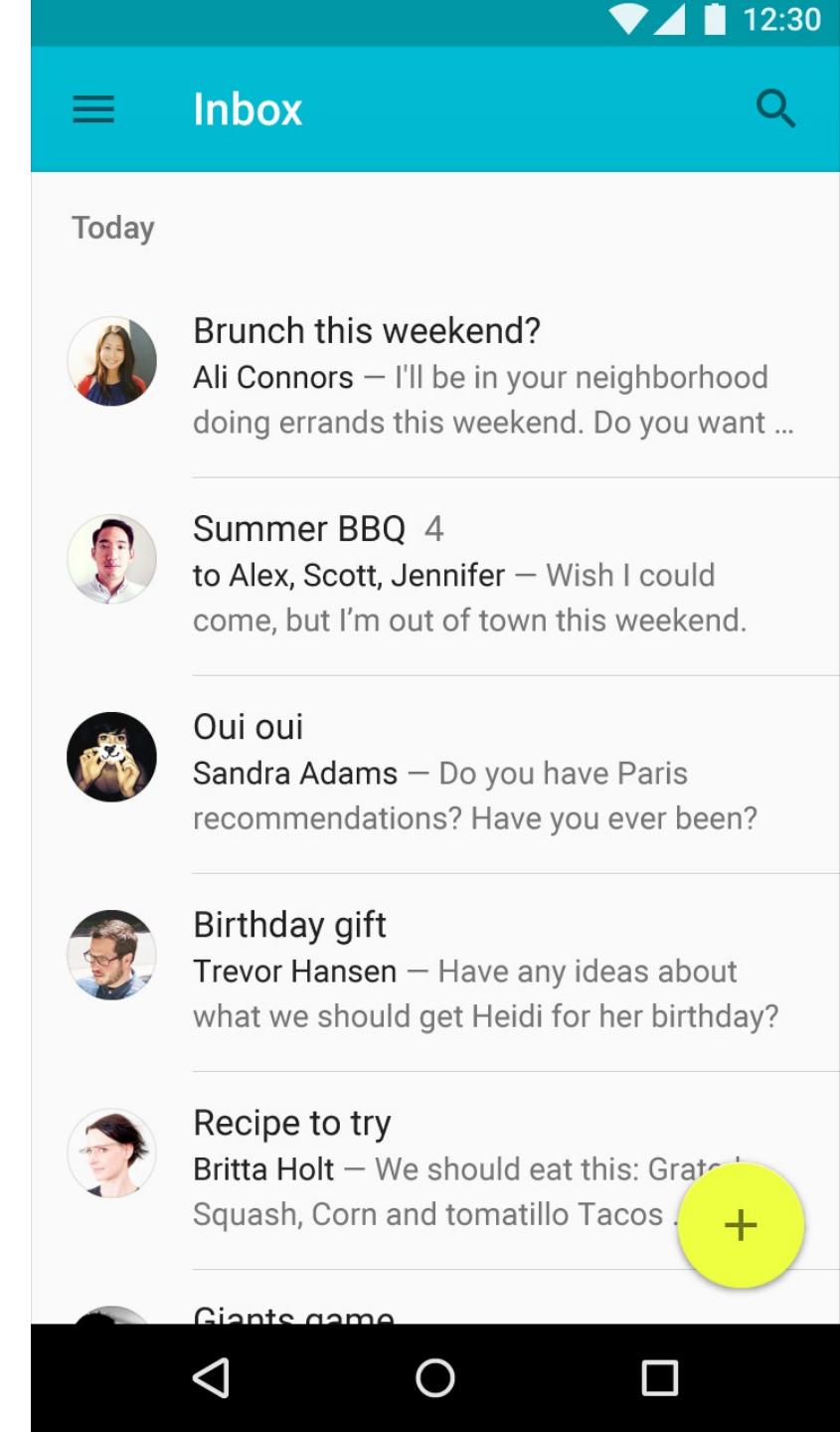
Índice

- Introdução;
- RecyclerView;
- Arquitetura;
- Implementação;
- Atualizar dados na RecyclerView;
- Leitura Recomendada.

Introdução

Em certas aplicações é necessário mostrar uma grande quantidade de dados ou apresentar dados que mudam regularmente;

Por exemplo, numa aplicação de SMS a quantidade de mensagens que um utilizador possui pode ascender a milhares. Ao mostrar esta informação ao utilizador **apenas um subset das mensagens serão visíveis no ecrã.**

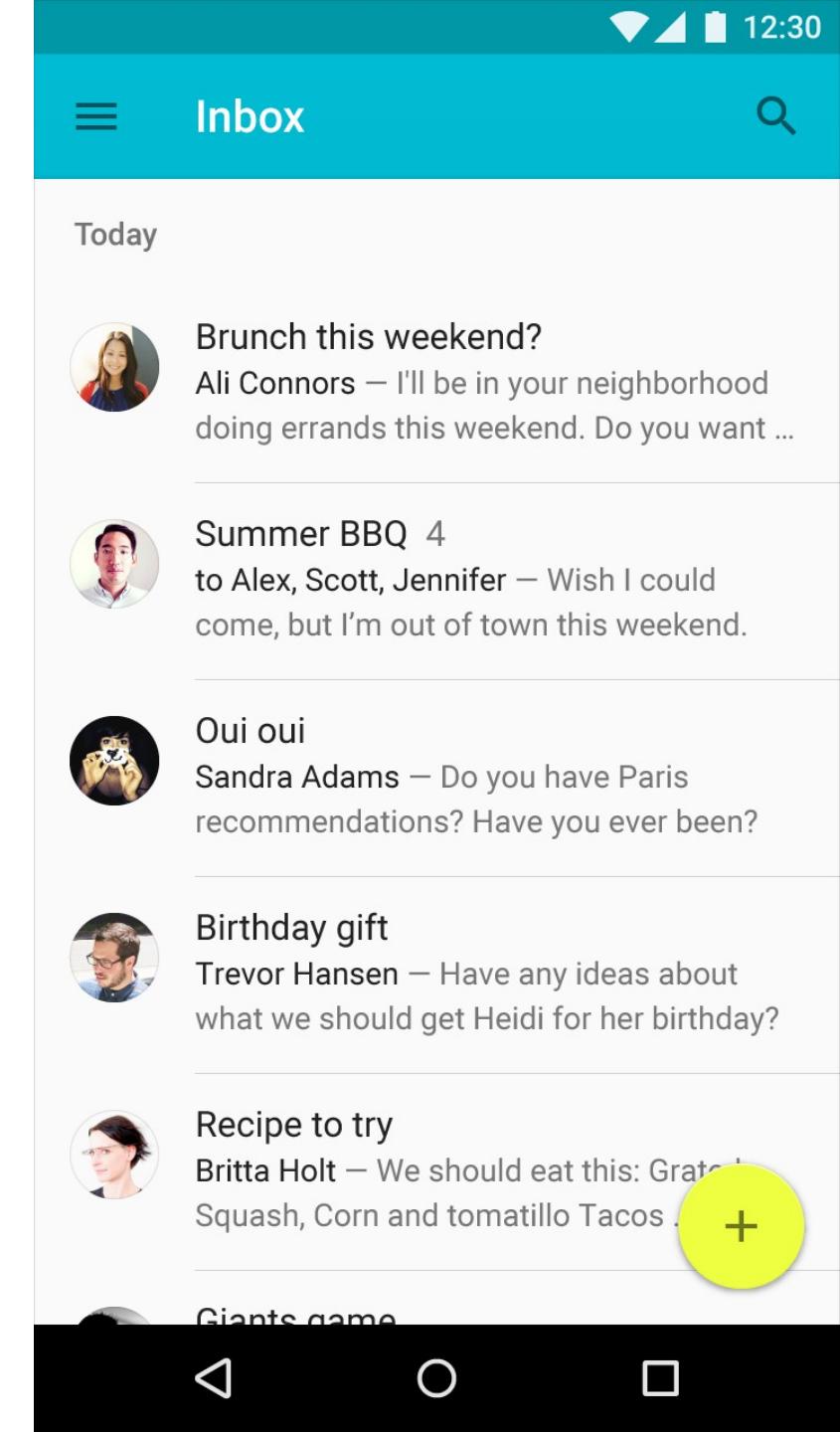


Introdução

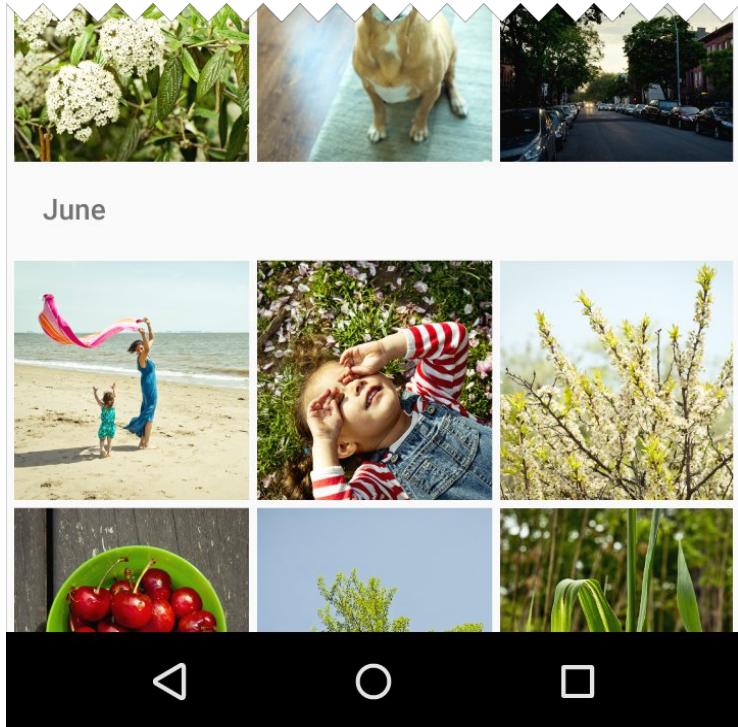
O problema

Não faz sentido a nível de performance criar uma view por cada mensagem, visto que podemos ficar sem memória;

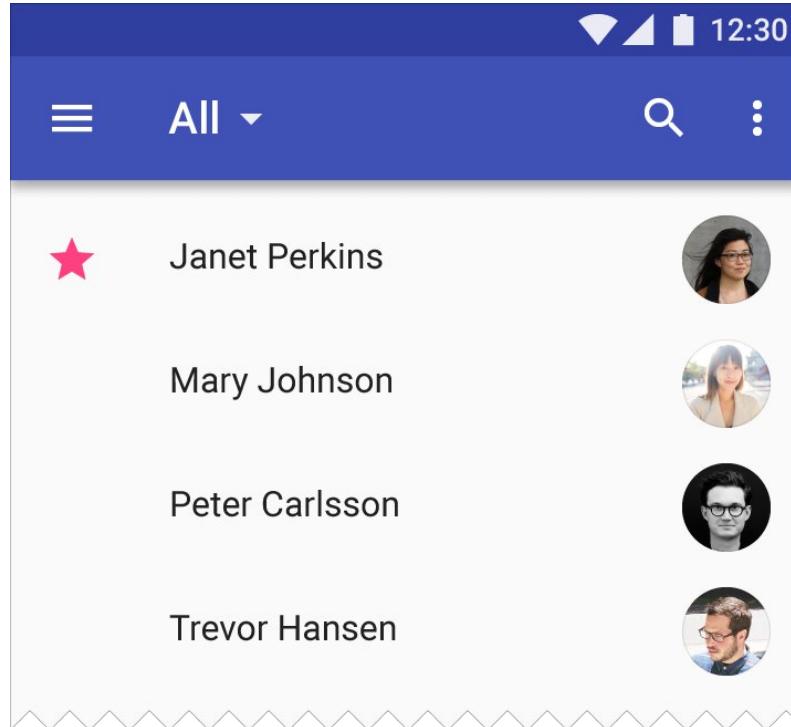
Também não é viável criar apenas as views quando o utilizador faz scroll na lista pois a instanciação de uma view é bastante dispendiosa e vai afetar a performance da aplicação.



RecyclerView



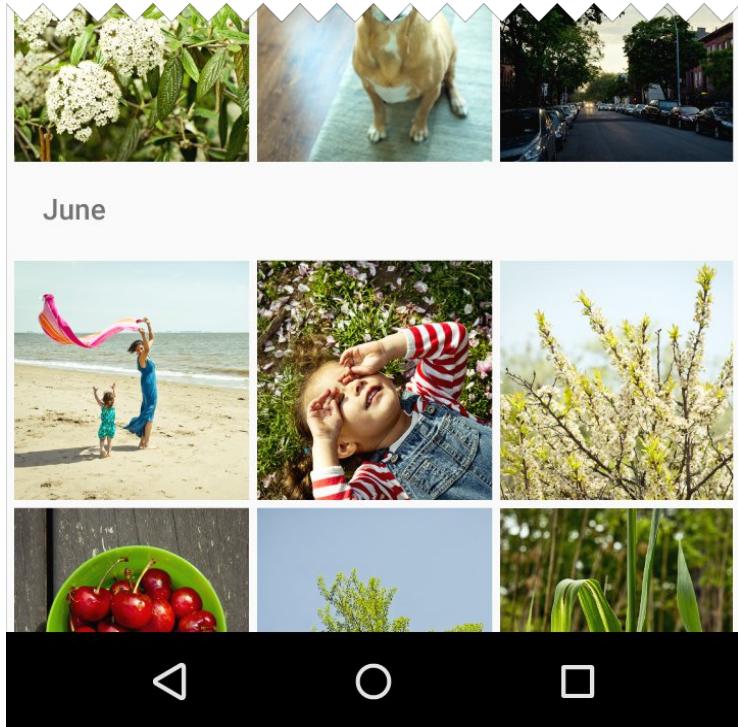
Grid



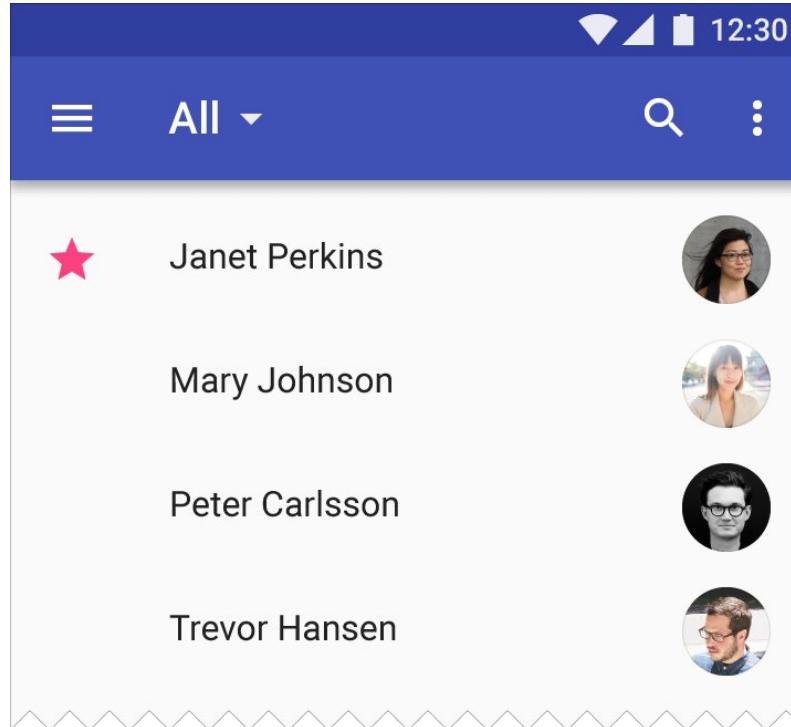
Lista

RecyclerView é um componente da biblioteca de suporte (v7) que facilita a apresentação de grandes quantidades de dados.

RecyclerView



Grid



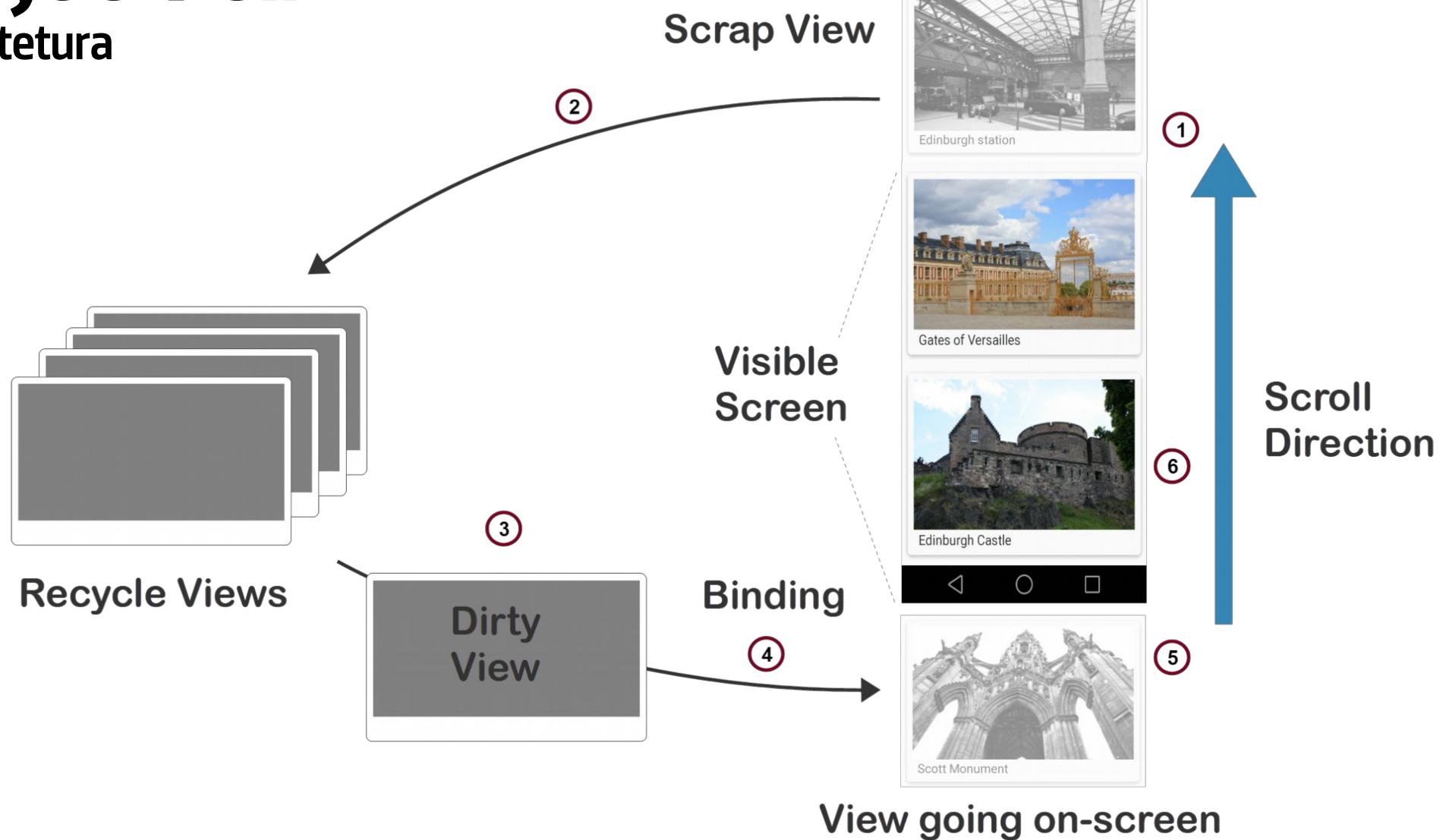
Lista

É uma view flexível que permite incluir uma fonte de dados extensa numa “janela” com espaço limitado.

Faz o scroll automático de dados se ficar sem “espaço” para serem mostrados no ecrã.

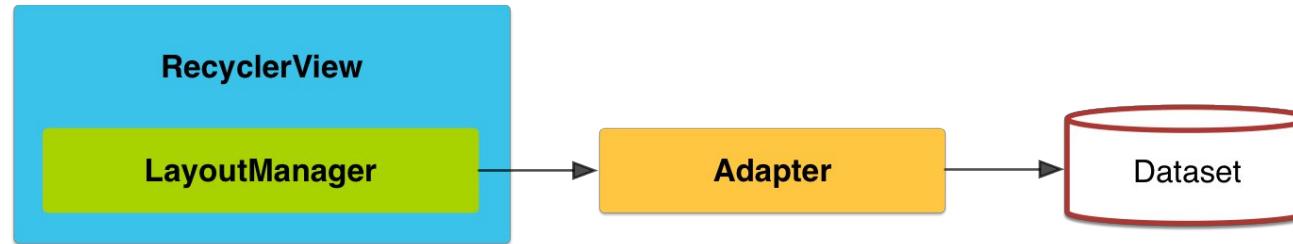
RecyclerView

Arquitetura



RecyclerView

Arquitetura



LayoutManager

- Responsável pelo posicionamento das views, e pela gestão das mesmas;
- São utilizados diferentes LayoutManagers para obter diferentes tipos de layouts: (1) LinearLayoutManager, (2) GridLayoutManager, (3) StaggeredGridLayoutManager.

Adapter

- Possui o dataset que vai ser mostrado;
- Serve de ponte entre o modelo de dados e a RecyclerView;

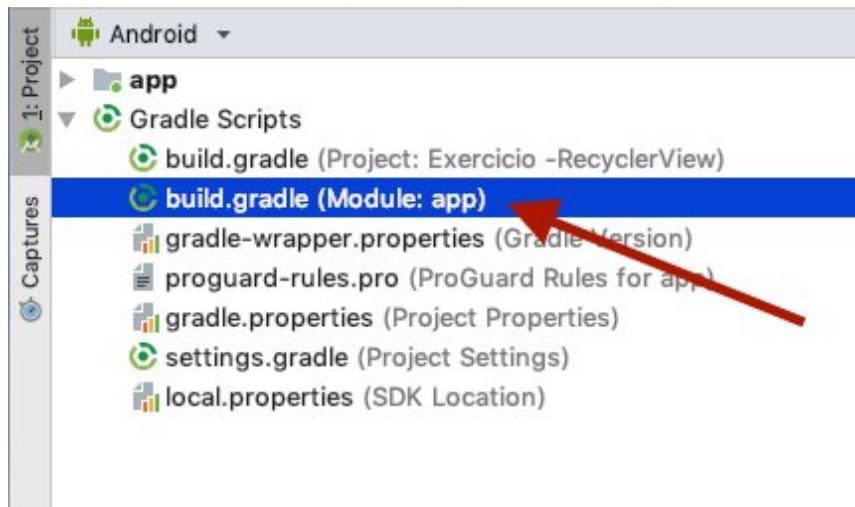
ViewHolder

- Possui uma referencia para uma view a ser mostrada no ecrã. É neste componente que fazemos “binding” dos dados à view.

RecyclerView

Implementação

1. Adicionar a dependência da RecyclerView ao ficheiro gradle do módulo



```
dependencies {  
    ...  
    implementation "androidx.recyclerview:recyclerview:1.4.0"  
  
    // For control over item selection of both touch and mouse driven selection  
    implementation "androidx.recyclerview-selection:1.1.0"  
    ...  
}
```

Nota: devem usar a versão mais recente disponível!

<https://developer.android.com/jetpack/androidx/releases/recyclerview>

RecyclerView

Implementação (2)

```
public class Contact {  
  
    private String name;  
    private Boolean isOnline;  
  
    public Contact(String name, Boolean isOnline){  
        this.name = name;  
        this.isOnline = isOnline;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public Boolean isOnline() {  
        return isOnline;  
    }  
  
    public void setOnline(Boolean onlineStatus) {  
        isOnline = onlineStatus;  
    }  
}
```

2. Criar a class que define o modelo de dados (ex. **Contact.java**)

RecyclerView

Implementação (3)

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

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical" />

</LinearLayout>
```

3. Adicionar o widget da RecyclerView ao layout da activity activity_main.xml

RecyclerView

Implementação (4)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal">

    <TextView
        android:id="@+id/txt_contact_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="10"
        android:padding="12dp" />

    <Button
        android:id="@+id/btn_contact_isOnline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1" />

</LinearLayout>
```

4. Criar o layout do item da RecyclerView. Este layout será replicado para todos os elementos a serem apresentados.

item_contact.xml

RecyclerView

Implementação (5)

```
public class ContactAdapter extends RecyclerView.Adapter<ContactAdapter.ContactViewHolder> {  
  
    private ArrayList<Contact> contactList;  
  
    public ContactAdapter(ArrayList<Contact> contactList) {  
        this.contactList = contactList;  
    }  
  
    @Override  
    public int getItemCount() {  
        return contactList.size();  
    }  
  
    ...  
}
```

5. Criar o adapter que vai conter os dados.(ex. ContactAdapter.java)

RecyclerView

Implementação (6)

```
// Provide a reference to the views for each data item
// Complex data items may need more than one view per item, and
// you provide access to all the views for a data item in a view holder
public static class ContactViewHolder extends RecyclerView.ViewHolder{

    public TextView txtView_contact_name;
    public Button btnMessage;

    public ContactViewHolder(@NotNull View itemView) {
        super(itemView);
        txtView_contact_name = itemView.findViewById(R.id.txt_contact_name);
        btnMessage = itemView.findViewById(R.id.btn_contact_isOnline);
    }
}
```

6. Ao extender `RecyclerView.Adapter<ContactAdapter.ContactViewHolder>` somos indicados para a implementação da respetiva class. Nesta classe realizamos a referencia às views presentes no layout para cada elemento (`item_contact.xml`).

RecyclerView

Implementação (7)

```
// Create new views (invoked by the layout manager)
@NonNull
@Override
public ContactViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    //Get Layout inflater from Context
    Context context = parent.getContext();
    LayoutInflator layoutInflater = LayoutInflator.from(context);

    //Inflate Layout
    View contactView = layoutInflater.inflate(R.layout.item_contact, parent, false);

    return new ContactViewHolder(contactView);
}
```

7. Implementar o método `onCreateViewHolder` que é invocado pelo Layout Manager escolhido e é responsável pela criação de novas vistas.

RecyclerView

Implementação (8)

```
// Replace the contents of a view (invoked by the layout manager)
@Override
public void onBindViewHolder(@NonNull ContactViewHolder viewHolder, int position) {
    //Get the data model based on position
    Contact contact = contactList.get(position);

    //Set name
    TextView textView = (TextView) viewHolder.txtView_contact_name;
    textView.setText(contact.getName());

    //Set button status
    Button button = (Button) viewHolder.btnMessage;
    button.setText(contact.isOnline() ? "Message" : "Offline");
    button.setEnabled(contact.isOnline());
}
```

8. Implementar o método `onBindViewHolder` que é invocado pelo Layout Manager escolhido e é responsável pela associação de dados às views presentes no layout.

RecyclerView

Implementação (9)

```
private RecyclerView recyclerView;
private ContactAdapter contactAdapter;
private ArrayList<Contact> contactList;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    //Define the LayoutManager
    recyclerView = (RecyclerView) findViewById(R.id.recyclerView);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));

    //Create contact List
    contactList = new ArrayList();
    addContacts(contactList);

    //Set the RecyclerView adapter
    contactAdapter = new ContactAdapter(contactList);
    recyclerView.setAdapter(contactAdapter);
}
```

9. Na `MainActivity.class` realizamos a instanciação do `adapter` e do respetivo `LayoutManager`.

Nota: se o conteúdo não alterar o tamanho do layout da `RecyclerView` devem utilizar a seguinte definição adicional (melhora a performance).

```
recyclerView.hasFixedSize(true);
```

Atualizar dados na RecyclerView

Sempre que queremos adicionar, atualizar ou remover itens do nosso dataset é necessário notificar os componentes da RecyclerView das alterações.

- Desta forma o Adapter possui os seguintes métodos:

```
notifyItemChanged(int position)           // Item na posição foi atualizado  
notifyItemInserted(int position)          // Adicionado item na posição  
notifyItemRemoved(int position)           // Removido item na posição  
notifyDataSetChanged()                   // Toda a lista foi atualizada
```

Exemplo: *// Adiciona um novo contacto na posição 0*
contacts.add(0, new Contact("Barney", true));

// Notificamos o adapter que foi adicionado um elemento
// novo na posição 0
adapter.notifyItemInserted(0);

Decorations

- As decorations permitem adicionar certos elementos visuais a cada view de uma RecyclerView.
 - O uso mais comum é o de adicionar os dividers (linhas divisórias) entre as views.
 - Também se usam para modificar o espaçamento entre os elementos da lista.



Exemplo:

```
RecyclerView.ItemDecoration itemDecoration = new DividerItemDecoration(this, DividerItemDecoration.VERTICAL);  
recyclerView.addItemDecoration(itemDecoration);
```

Leitura Recomendada

Recyclerview

<https://developer.android.com/jetpack/androidx/releases/recyclerview>

Guia recyclerview

<https://developer.android.com/guide/topics/ui/layout/recyclerview>

Programação Para Dispositivos Móveis I

RecyclerView

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

FRAGMENTS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

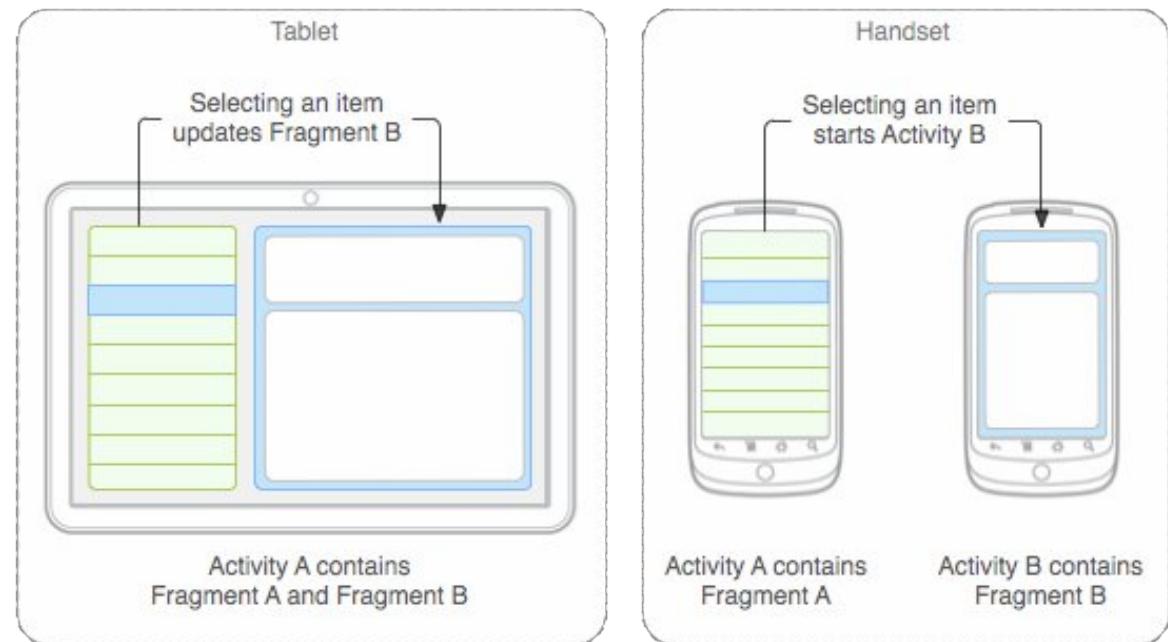
Índice

- Fragments;
- Ciclo de Vida dos Fragments;
- Criar um Fragment;
- Substituir Fragments num FrameLayout;
- Comunicação Entre Fragments;
- Tablet e Smartphone;
- Leitura Adicional.

Fragments

Um **comportamento** ou uma **parte da interface gráfica** de uma **Activity**;

Podem ser **utilizados mais do que um** por Activity e podem ser utilizados em **mais do que uma** Activity.



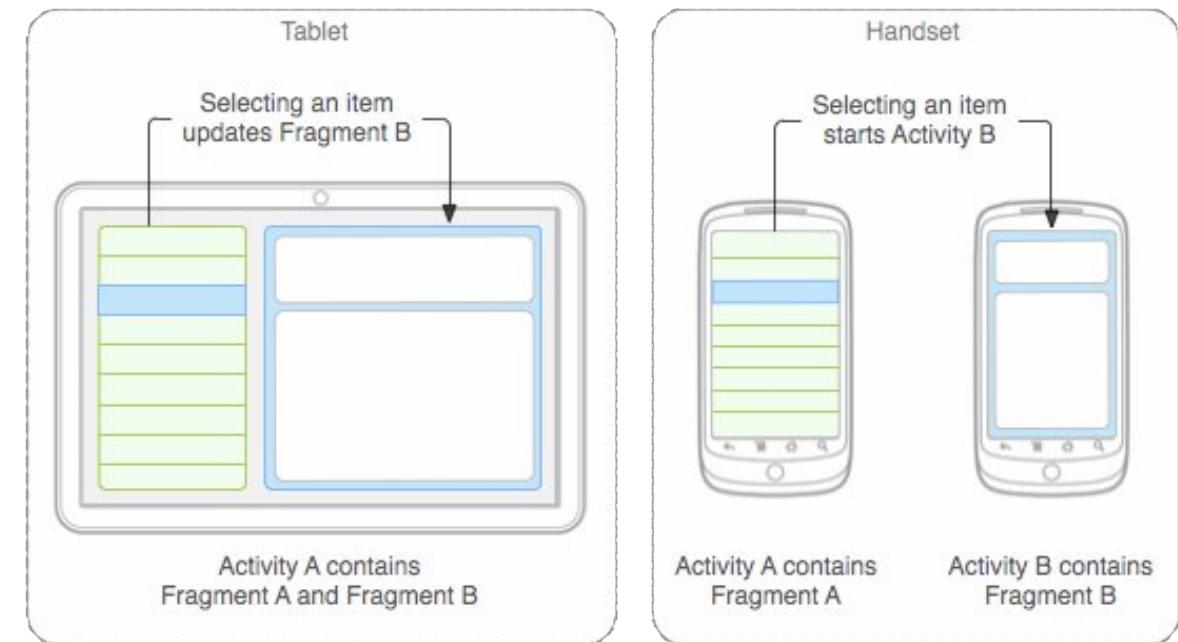
Fragments

Definição

É uma combinação de um layout xml, e uma classe (de forma semelhante a uma Activity);

Fazem o encapsulamento de views e lógica, para que sejam de fácil reutilização em Activities;

São componentes que contêm views, eventos e lógica;



Fragments

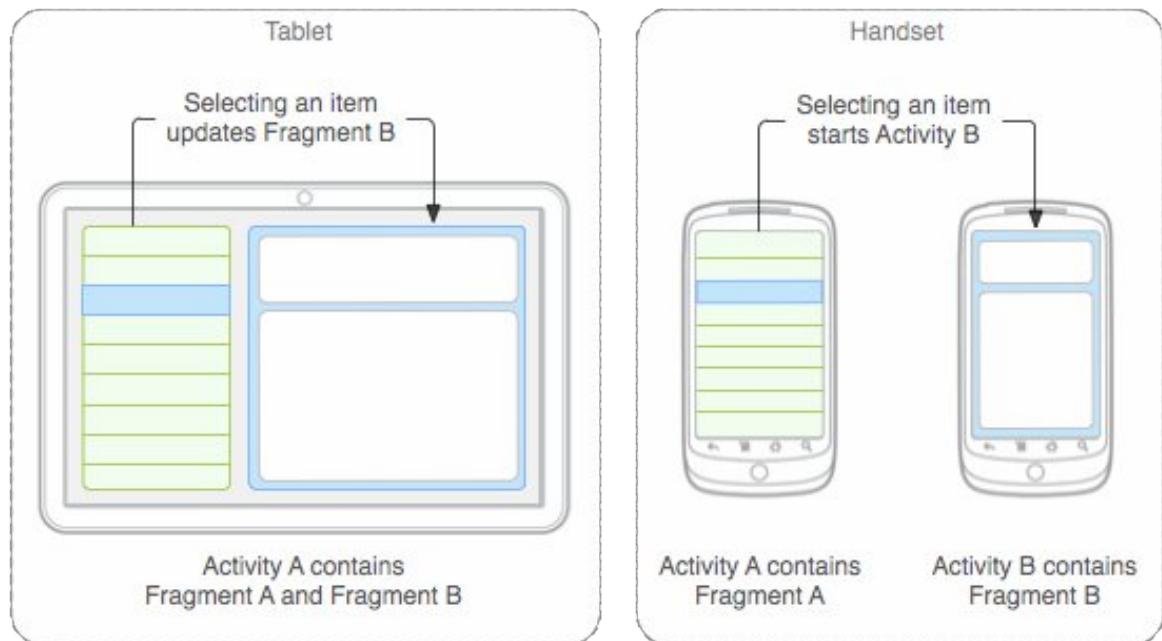
Características

Foram introduzidos com o Android 3.0, de modo a fornecer uma interface mais dinâmica e flexível;

Existe uma biblioteca de suporte (`android-support-v4.jar`) que permite utilizar Fragments em versões anteriores à 3.0;

Por exemplo, numa aplicação de notícias em Tablet:

- À esquerda, um Fragment para mostrar uma lista com várias notícias
- À direita, um Fragment que apresenta o conteúdo da notícia selecionada



Fragments

Características (2)

Recebem os seus **próprios** inputs (Intents);

Possuem o seu **próprio** ciclo de vida e o seu **próprio** layout;

Podem ser **adicionados** ou **removidos** de uma Activity em tempo real;

Criados para serem **reutilizados**;

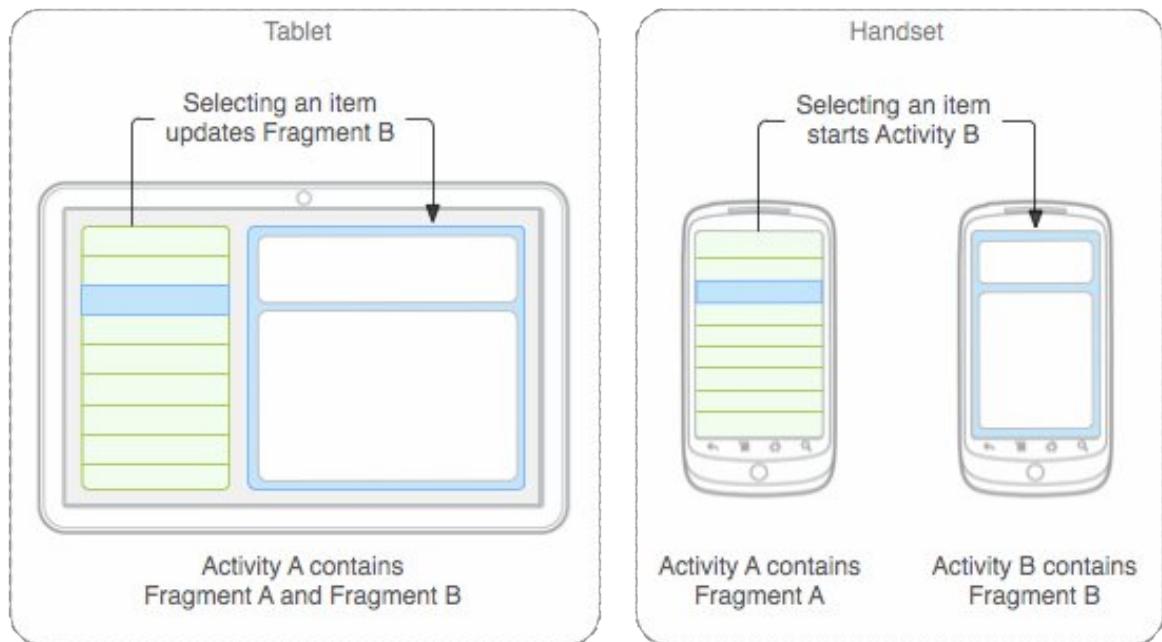
Têm de ser embebidos numa Activity e o seu ciclo de vida é **diretamente afetado** pelo ciclo de vida da Activity:

- Quando a Activity passa pelo `onPause()` **todos** os Fragments no interior da Activity também o fazem.

Fragments

Importância de utilização

- Reutilização de views e componentes lógicos;
- Suporte a tablets;
- Orientação de ecrã;



Fragments

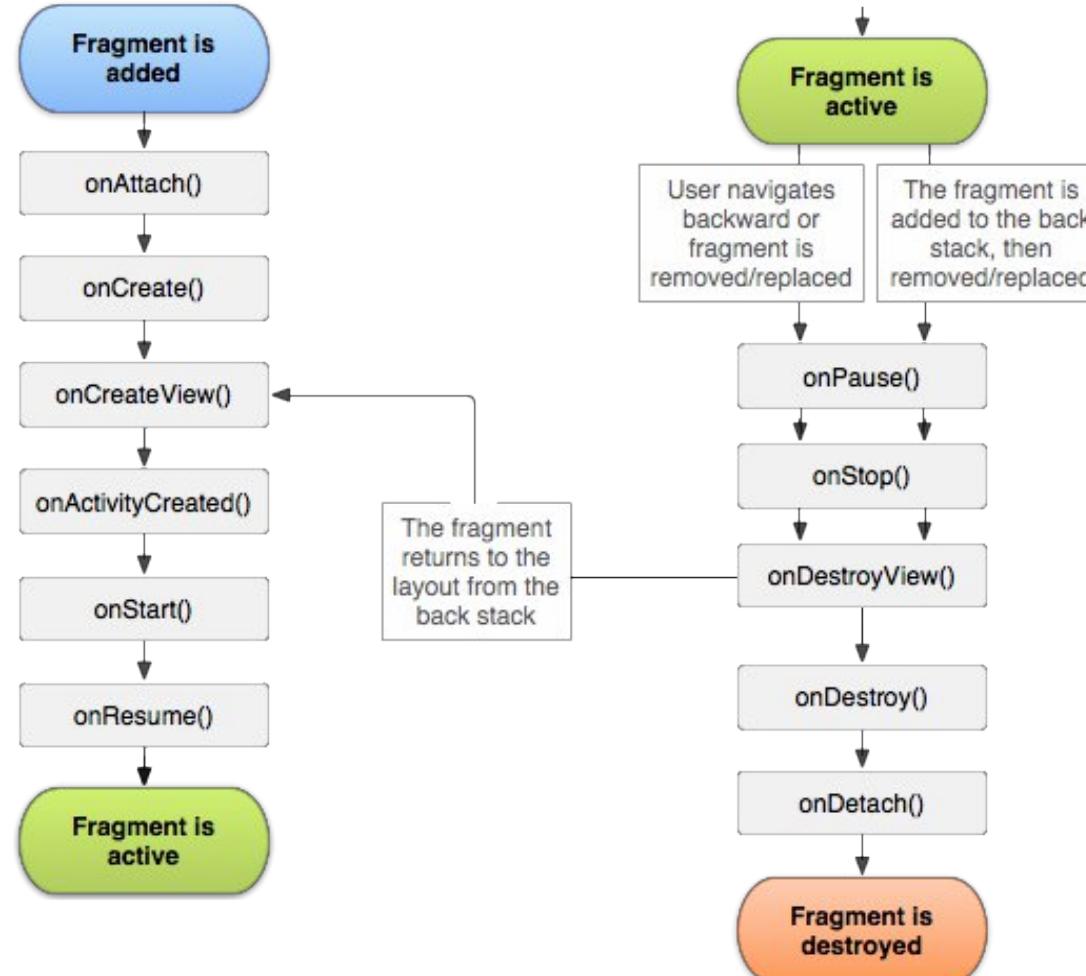
Diferenças de uma Activity

Não representam um contexto, pelo que convém obter sempre o contexto da Activity e guardá-lo numa variável;

Atribuição do layout não é efetuada através do método setContentView, mas sim utilizando o LayoutInflater do método onCreateView;

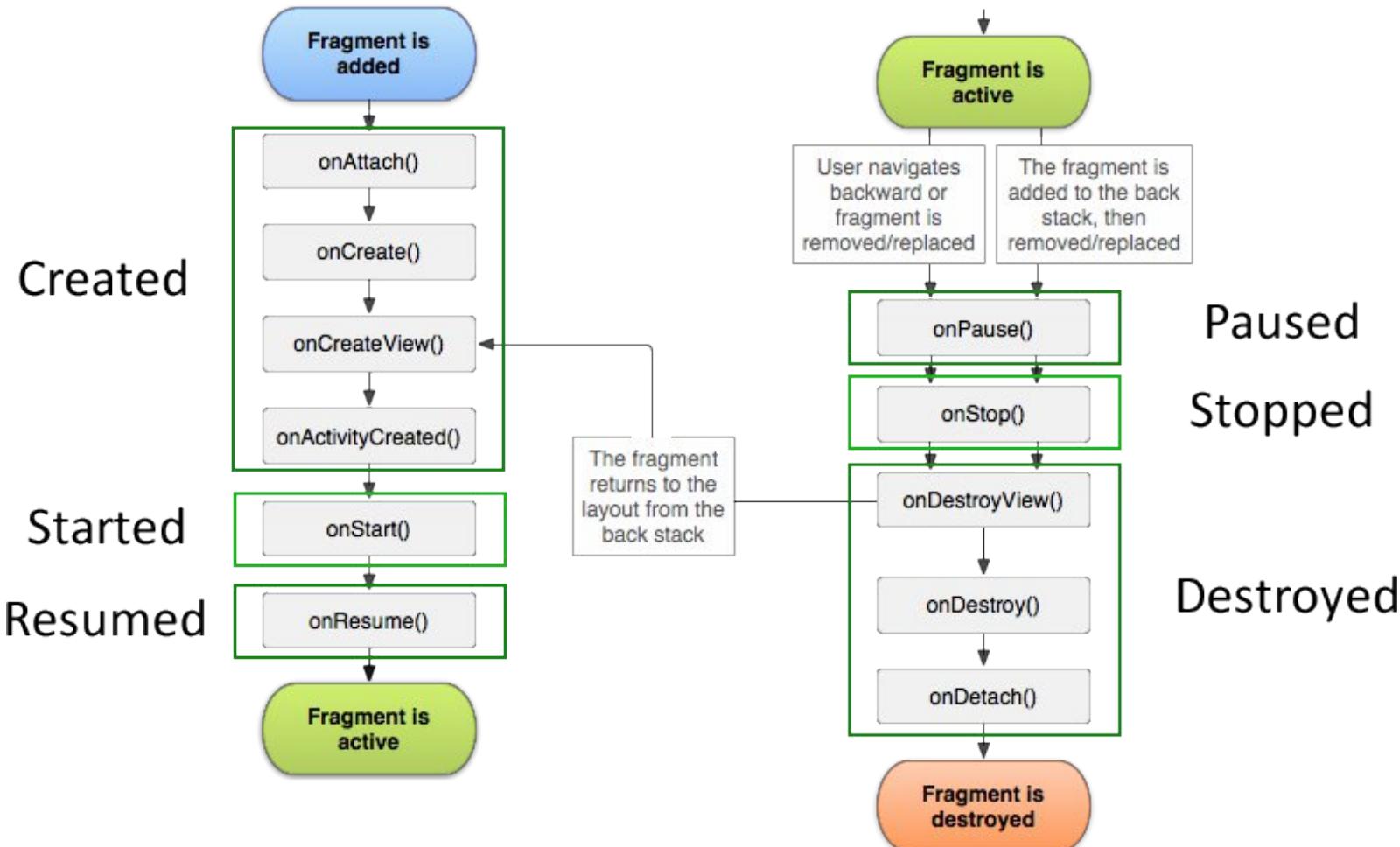
Ciclo de vida ligeiramente diferente embora **dependente** do ciclo de vida da Activity em que se insere.

Ciclo de Vida



Cada Fragment tem um ciclo de vida próprio...

Ciclo de Vida



Cada Fragment tem um **ciclo de vida próprio**... que **interage** com o da Activity em que se insere.

Fragments

Ciclo de Vida -> Novos Métodos

- `onAttach` – Invocado quando o Fragment é associado a uma Activity;
- `onCreateView` – É invocado quando a interface gráfica do Fragment vai ser desenhada. Este método deve devolver uma View, ou caso seja um Fragment sem interface gráfica, deve devolver null;
- `onActivityCreated` – Invocado imediatamente após a execução do método `onCreate` da Activity;
- `onDestroyView` – O inverso do `onCreateView`, ou seja, é invocado quando os componentes gráficos do Fragment vão ser removidos da interface gráfica;
- `onDetach` – O inverso do `onAttach`, é invocado quando o Fragment vai ser desassociado da Activity.

Fragments

Criação

A superclasse deve ser Fragment ou uma das suas subclasses:

- **DialogFragment** – cria uma dialog flutuante
- **ListFragment** – similar à ListView
- **PreferenceFragment** – similar à PreferenceActivity (só existe a partir do Android 3.0)
- **WebViewFragment** – mostra uma WebView, para visualizar websites

A estrutura de código é idêntica à de uma Activity;

Pode-se inclusive facilmente **converter** uma Activity **existente** num Fragment.

!! Não precisa de ser declarado no AndroidManifest.xml !!

Fragments

fragment_example.xml

```
<?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:orientation="vertical">

    <TextView
        android:id="@+id/txt_example"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/example" />

    <Button
        android:id="@+id/btn_example"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn_example" />
</LinearLayout>
```

Fragments

FragmentExample.java

```
public class FragmentExample extends Fragment {

    private Context mContext;
    private TextView txtViewExample;
    private Button btnExample;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mContext = getActivity();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View mContentView = inflater.inflate(R.layout.fragment_example, container, false);
        txtViewExample = (TextView) mContentView.findViewById(R.id.txt_example);
        btnExample = (Button) mContentView.findViewById(R.id.btn_example);

        return mContentView;
    }
}
```

Fragments

activity_main.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:orientation="vertical"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/fragment_test"
        android:name="pt.ipp.estg.t_fragment.FragmentExample"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Inserção de forma estática

Fragments

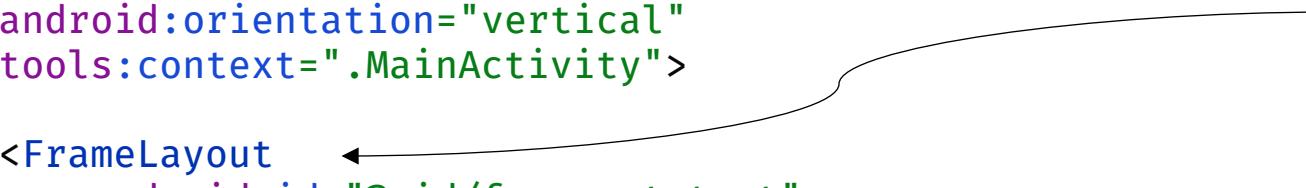
activity_main.xml

```
<?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:orientation="vertical"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/fragment_test"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Inserção de forma dinâmica



Fragments

Atributos Importantes

android:name – é aqui que é especificado qual o Fragment que irá ser colocado no layout da Activity. Corresponde ao identificador da classe Java do Fragment (package + nome)

[Apenas se aplica na inserção estática de Fragments]

android:id – necessário para termos uma referência no código da Activity para o Fragment no Layout em XML;

android:tag – identificador único (string) para representar o Fragment;

Fragments

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        FragmentExample fragmentExample = new FragmentExample();

        FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();
        fragmentTransaction.replace(R.id.fragment_container, fragmentExample);
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
    }
}
```

Fragments

Gestão

Os Fragment são **adicionados, removidos e substituídos** numa Activity utilizando a classe `FragmentTransaction`.

Esta é inicializada da seguinte forma

```
FragmentManager fragmentManager = getSupportFragmentManager();  
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

Não esquecer de fazer commit no final da transação

```
fragmentTransaction.commit();
```

Fragments

Gestão (2)

Adicionar Fragment

```
fragmentTransaction.add(ID_VIEW_FRAGMENT, FRAGMENT);
```

Substituir Fragment

```
fragmentTransaction.replace(ID_VIEW_FRAGMENT, FRAGMENT);
```

Remover Fragment

```
fragmentTransaction.remove(FRAGMENT);
```

Utilizando o método `fragmentTransaction.addToBackStack(String)`, antes de fazer `fragmentTransaction.commit()`, podemos adicionar o Fragment à stack de Activities.

Para retirar um Fragment da stack podemos invocar o método `popBackStack()`.

Fragments

Encontrar Fragment por ID

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        FragmentExample fragmentExample = new FragmentExample();

        FragmentManager fragmentManager = getSupportFragmentManager();
        fragmentManager.findFragmentById(R.id.fragment_container);
    }
}
```

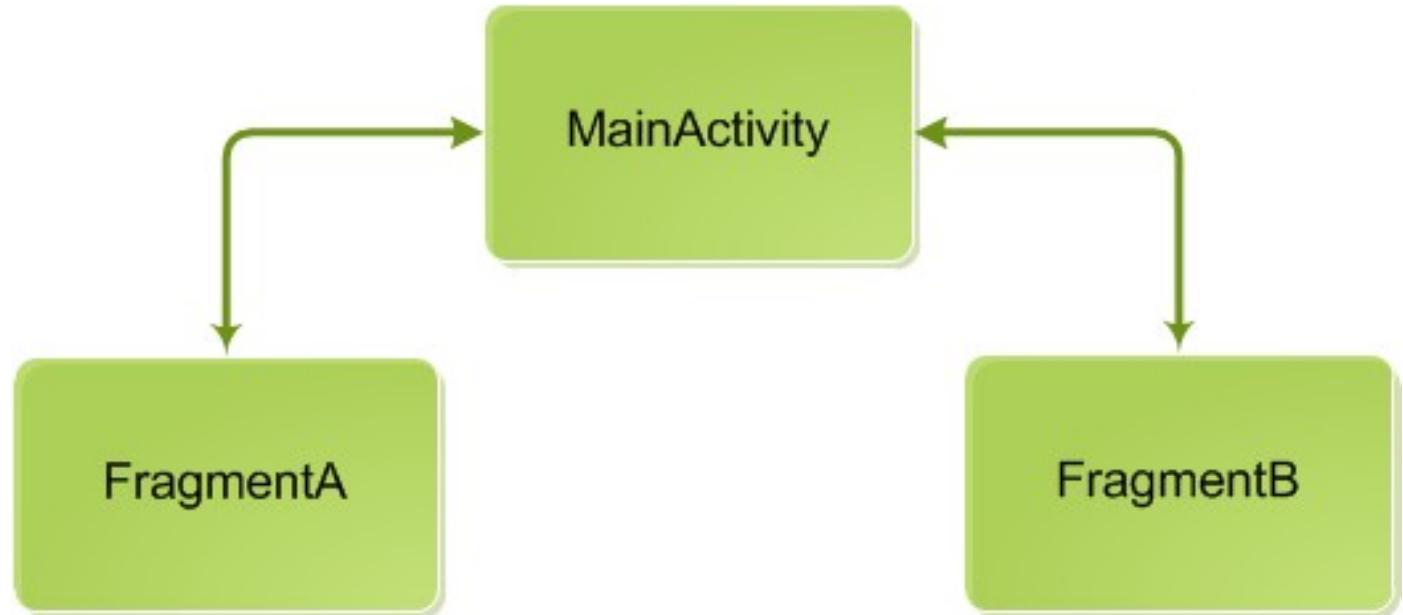
Fragments

Encontrar Fragment por TAG

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        FragmentExample fragmentExample = new FragmentExample();  
  
        FragmentManager fragmentManager = getSupportFragmentManager();  
        fragmentManager.findFragmentByTag("FRAGMENT_TAG");  
    }  
}
```

Comunicação entre Fragments

Fragments não conseguem comunicar entre si, para o fazerem têm de comunicar através da Activity, que depois comunica com o outro Fragment.



Comunicação entre Fragments

App Lista de Países do Mundo

Vamos imaginar uma aplicação que lista os detalhes de países de todo o mundo. Podemos ter os seguintes componentes:

- MainActivity: que incorpora dois Fragments;
- CountriesListFragment: FragmentList com a lista de países;
- DetailsFragment: Fragment que mostra os detalhes do país escolhido;
- OnCountrySelectedListener interface: para haver comunicação entre Fragment e Activity.

Comunicação entre Fragments

interface OnCountrySelectedListener

```
static interface OnCountrySelectedListener {  
    //Método utilizado para notificar a instância que está a "escuta"  
    public void onCountrySelected(int position);  
}
```

Comunicação entre Fragments

CountryListFragment.java

```
public class CountriesListFragment extends ListFragment {  
  
    static interface OnCountrySelectedListener{  
        public void OnCountrySelected(long id);  
    }  
  
    private OnCountrySelectedListener listener;
```

A MainActivity.java irá registar-se no CountryListFragment de forma a executar eventos de seleção de um país

Comunicação entre Fragments

CountryListFragment.java

```
@Nullable  
@Override  
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,  
                           @Nullable Bundle savedInstanceState) {  
    String[] countryNames = new String[Country.countries.length];  
    for(int i = 0; i < countryNames.length; i++){  
        countryNames[i] = Country.countries[i].getName();  
    }  
  
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(  
        inflater.getContext(),  
        android.R.layout.simple_list_item_1,  
        countryNames  
    );  
    setListAdapter(adapter);  
  
    return super.onCreateView(inflater, container, savedInstanceState);  
}
```

Comunicação entre Fragments

CountryListFragment.java

```
@Override  
public void onAttach(@NotNull Context context) {  
    super.onAttach(context);  
    try{  
        this.listener = (OnCountrySelectedListener) context;  
    }catch (ClassCastException e){  
        throw new ClassCastException(getActivity().toString() + "must implement OnCountrySelectedListener");  
    }  
}
```

No método `onAttach()` verificamos se a `MainActivity` está a implementar a interface

Comunicação entre Fragments

CountryListFragment.java

```
@Override  
public void onListItemClick(@NotNull ListView l, @NotNull View v, int position, long id) {  
    if(listener != null){  
        listener.OnCountrySelected(id);  
    }  
}
```

O FragmentList já possui um listener de eventos (click). Sempre que houver uma interação com um item, invocamos o método OnCountrySelected() presente na MainActivity.

Comunicação entre Fragments

DetailsFragment.java

```
private long countryId;  
  
public void setCountryId(long id){  
    this.countryId = id;  
}
```

Registamos uma variável que representa o id do objeto que queremos apresentar, e um método para alterar esse valor.

Comunicação entre Fragments

MainActivity.java

Implementamos a interface

```
public class MainActivity extends AppCompatActivity implements CountriesListFragment.OnCountrySelectedListener {
```

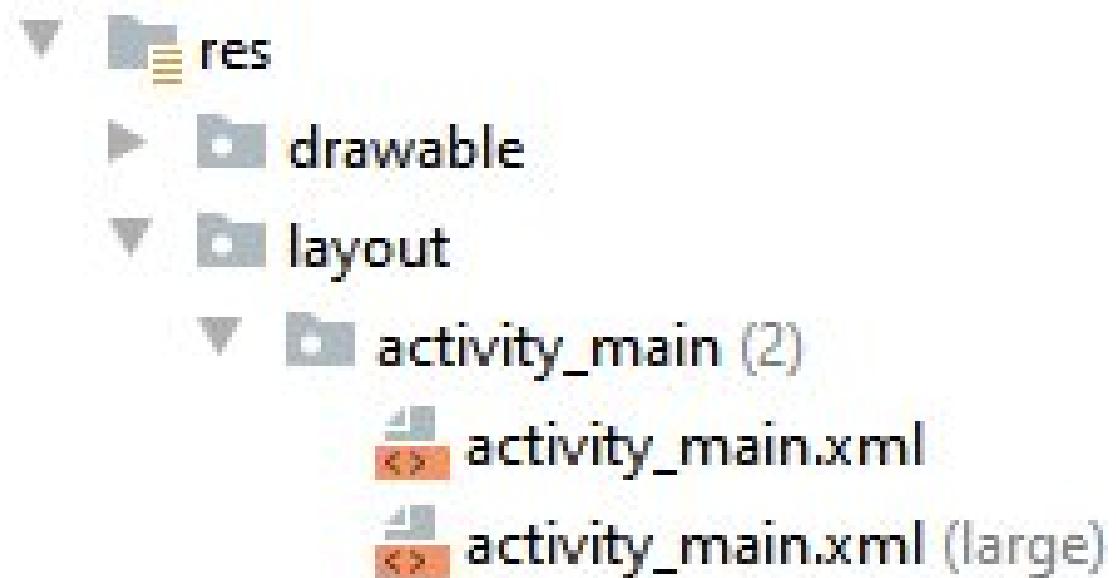
Implementamos o método da interface, que
será responsável por toda a comunicação

```
@Override  
public void OnCountrySelected(long id) {  
    DetailsFragment detailsFragment = new DetailsFragment();  
    detailsFragment.setCountryId(id);  
  
    fragmentTransaction = fragmentManager.beginTransaction();  
    fragmentTransaction.replace(R.id.fragment_container, detailsFragment);  
    fragmentTransaction.addToBackStack(null);  
    fragmentTransaction.commit();  
}
```

Tablet e Smartphone

Utilização automática de diferentes layouts conforme o dispositivo seja uma Tablet ou um Smartphone

- Diferentes ficheiros de layout
- Com o mesmo nome
- Mas com configurações distintas

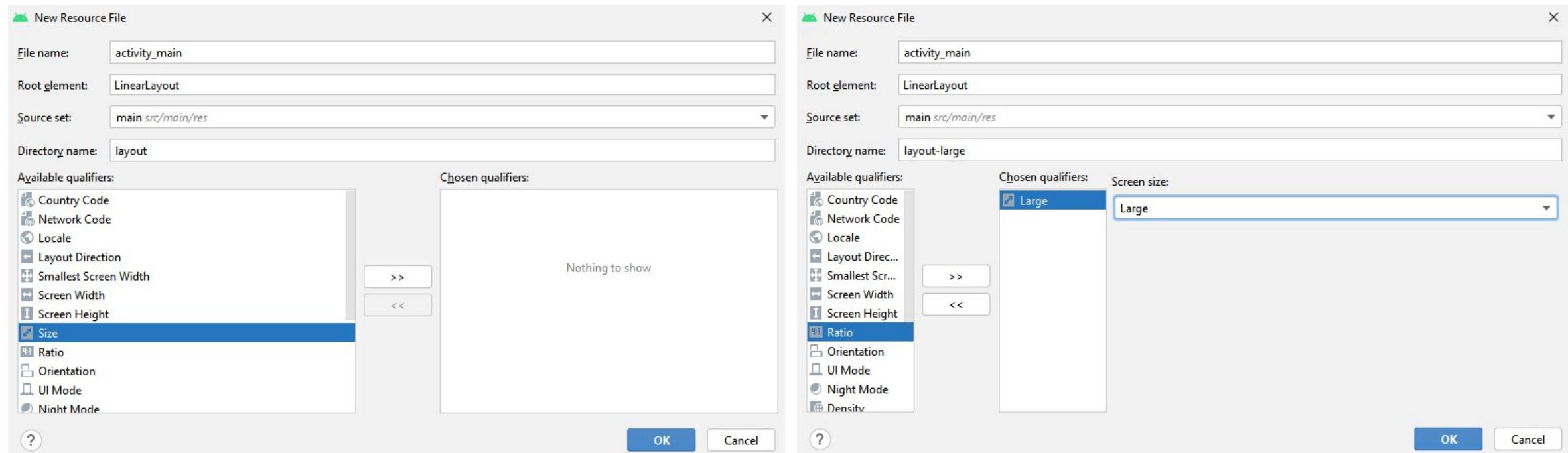


Tablet e Smartphone

Criação do Layout

Ao criar um **novo layout** podemos especificar qual o **tamanho** a que o mesmo se adequa.

Na configuração do layout escolhemos o qualifier **Size** e depois o tamanho **Large**



Tablet e Smartphone

activity_main.xml [smartphone]

```
<?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:orientation="vertical"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Tablet e Smartphone

activity_main.xml (large) [tablet]

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

    <fragment
        android:name="pt.ipp.estg.t_fragment.CountriesListFragment"
        android:id="@+id/fragment_countries"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <fragment
        android:name="pt.ipp.estg.t_fragment.DetailsFragment"
        android:id="@+id/fragment_details"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="2" />

</LinearLayout>
```

Tablet e Smartphone

MainActivity.java [verificação de layout a utilizar]

```
private FragmentManager fragmentManager;
private FragmentTransaction fragmentTransaction;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    if(findViewById(R.id.fragment_container) != null){
        if(savedInstanceState!= null){
            return;
        }
        CountriesListFragment countriesListFragment = new CountriesListFragment();
        fragmentManager = getSupportFragmentManager();
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.add(R.id.fragment_container, countriesListFragment);
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
    }
}
```

Se tivermos a utilizar um dispositivo Tablet, esta View não será inicializada (ou seja, será null)

Tablet e Smartphone

MainActivity.java [ações variam dependendo do layout]

```
@Override  
public void onCountrySelected(long id) {  
    DetailsFragment detailsFragment =  
        (DetailsFragment) getSupportFragmentManager().findFragmentById(R.id.fragment_details);  
  
    if(detailsFragment != null){  
        detailsFragment.updateCountry(id);  
    }else{  
        detailsFragment = new DetailsFragment();  
        detailsFragment.setCountryId(id)  
  
        fragmentTransaction = fragmentManager.beginTransaction();  
        fragmentTransaction.replace(R.id.fragment_container, detailsFragment);  
        fragmentTransaction.addToBackStack(null);  
        fragmentTransaction.commit();  
    }  
}
```



Verificação se estamos a utilizar o layout de tablet. Em caso afirmativo este valor de id da view será diferente de null

Tablet e Smartphone

App final [smartphone]

T_fragment
Country : Portugal
Continent : Europe
Country : United Kingdom
Continent : Europe
Country : Mexico
Continent : America
Country : Argentina
Continent : America
Country : Brazil
Continent : America
Country : Egypt
Continent : Africa
Country : Sidney
Continent : Australia
Country : Greece
Continent : Europe

T_fragment
Country : Portugal
Continent : Europe
Portugal is a southern European country on the Iberian Peninsula, bordering Spain. Its location on the Atlantic Ocean has influenced many aspects of its culture: salt cod and grilled sardines are national dishes, the Algarve's beaches are a major destination and much of the nation's architecture dates to the 1500s–1800s, when Portugal had a powerful maritime empire.

Tablet e Smartphone

App final [tablet]

T_fragment	
Country : Portugal	Country : Portugal
Continent : Europe	Continent : Europe
Country : United Kingdom	Portugal is a southern European country on the Iberian Peninsula, bordering Spain. Its location on the Atlantic Ocean has influenced many aspects of its culture: salt cod and grilled sardines are national dishes, the Algarve's beaches are a major destination and much of the nation's architecture dates to the 1500s–1800s, when Portugal had a powerful maritime empire.
Continent : Europe	
Country : Mexico	
Continent : America	
Country : Argentina	
Continent : America	
Country : Brazil	
Continent : America	
Country : Egypt	
Continent : Africa	
Country : Sidney	
Continent : Australia	
Country : Greece	
Continent : Europe	

Leitura Adicional

Fragments

<https://developer.android.com/guide/components/fragments>

Comunicação entre Fragments

<https://developer.android.com/training/basics/fragments/communicating>

UI para Tablets e Smartphones

<https://developer.android.com/training/basics/fragments/fragment-ui>

Programação Para Dispositivos Móveis I

FRAGMENTS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

DIALOGS

2023/_24 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

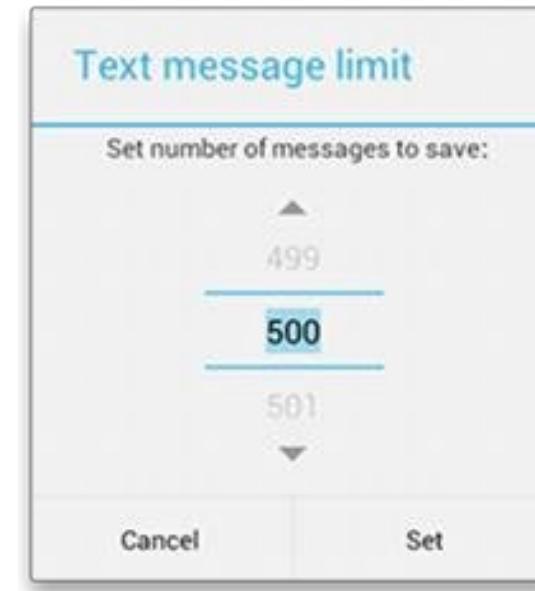
Índice

- Dialog;
- Apresentar um Dialog;
- Adicionar botões a uma Dialog;
- Dialog com listas;
- Dialog com Layout Personalizado;
- Processar Eventos do Dialog na Activity;
- Leitura Adicional.

Dialog

É uma **pequena janela flutuante** que permite ao utilizador **tomar uma decisão** ou **introduzir informação adicional**.

- Não ocupa totalmente o ecrã e é normalmente utilizado para os utilizadores tomarem ações antes de prosseguirem na aplicação.



Dialog

Tipos

A Dialog é a **classe base** para criação de um elemento deste tipo, contudo não devemos utilizar uma das suas subclasses:

- **AlertDialog** : Apresenta um título, até três botões, a lista de vários itens selecionáveis ou um layout personalizado;
- **DatePickerDialog** e **TimePickerDialog**: Uma dialog com layout próprio que permite ao utilizador selecionar uma data ou uma hora;

Estas classes definem o **estilo** e a **estrutura** da Dialog, mas na sua criação deve ser utilizada a classe DialogFragment como container.

AlertDialog

Elementos

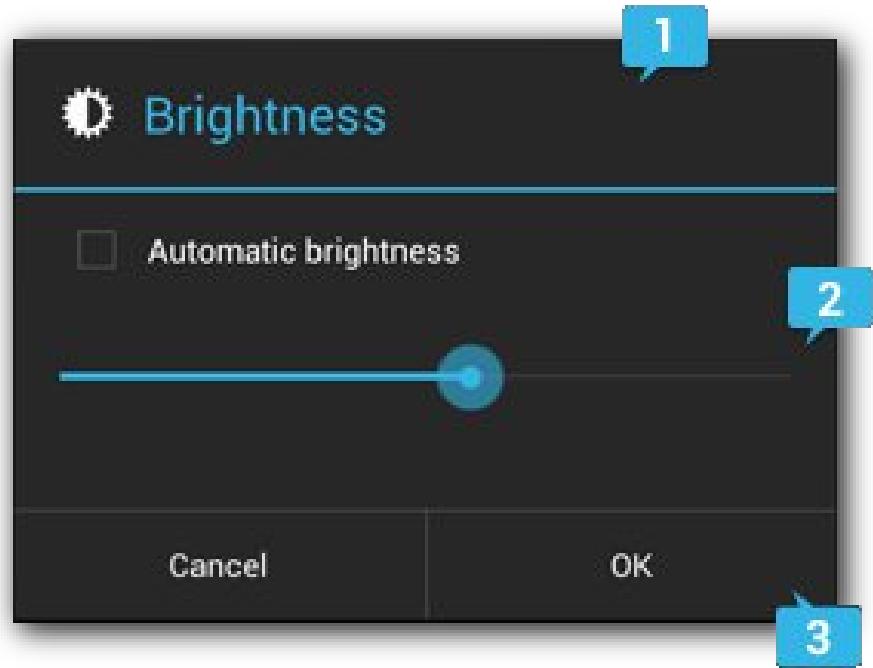
1. Título (opcional)

2. Área de conteúdo

- Pode apresentar uma mensagem, uma lista ou um layout personalizado;

3. Botões

- Não devem existir mais do que três;



AlertDialog

Dialog com mensagem e 3 botões



AlertDialog

FireMissilesDialogFragment.java

```
public class FireMissilesDialogFragment extends DialogFragment {  
  
    @NotNull  
    @Override  
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
  
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
  
        builder.setMessage(R.string.fire_missiles);  
  
        return builder.create();  
    }  
  
    Extende DialogFragment  
    Definição do Builder de  
    AlertDialog
```

AlertDialog

activity_main.xml

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

    <Button
        android:id="@+id/btn_missiles"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/missiles" />

</LinearLayout>
```

Botão que vai ser usado para
mostrar o DialogFragment
(FireMissilesDialogFragment.java)

AlertDialog

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
  
    private Button btnMissiles;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        btnMissiles = (Button) findViewById(R.id.btn_missiles);  
        btnMissiles.setOnClickListener(this);  
    }  
  
    @Override  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.btn_missiles: fireMissiles();  
                break;  
        }  
    }  
  
    private void fireMissiles() {  
        new FireMissilesDialogFragment().show(getSupportFragmentManager(), "Missiles");  
    }  
}
```

Gestor de Fragments do Android

TAG

AlertDialog

Resultado atual

Fire Missiles?

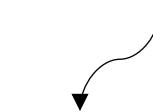
E os botões?

AlertDialog

Adicionar botões [FireMissilesDialogFragment.java]

```
public class FireMissilesDialogFragment extends DialogFragment implements DialogInterface.OnClickListener {  
  
    @NotNull  
    @Override  
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
  
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
  
        builder.setMessage(R.string.fire_missiles)  
            .setPositiveButton(R.string.fire, this)  
            .setNegativeButton(android.R.string.cancel, this)  
            .setNeutralButton(R.string.later, this);  
  
        return builder.create();  
    }  
}
```

Implementa o OnClickListener para detetar os clickes nos botões



Adicionar os botões através do builder

AlertDialog

Tipos de botões

Existem três tipos de botões de ação que podemos adicionar:

- **Positivo:** Deve-se utilizar este botão para aceitar e continuar com a ação que foi apresentada (botão de “OK”);
- **Negativo:** Deve-se utilizar este botão para cancelar a ação;
- **Neutro:** Deve-se utilizar este botão quando o utilizador não pretende continuar com a ação, mas também não pretende cancelar. Pode ser comparado a uma ação de “Remind me Later”;

AlertDialog

Detetar clicks [FireMissilesDialogFragment.java]

```
@Override  
public void onClick(DialogInterface dialog, int which) {  
    String message = " ";  
    switch(which){  
        case DialogInterface.BUTTON_POSITIVE:  
            message = " !! MISSILES LAUNCHED !! ";  
            break;  
        case DialogInterface.BUTTON_NEGATIVE:  
            message = " !! YOU GOT LUCKY TODAY !! ";  
            break;  
        case DialogInterface.BUTTON_NEUTRAL:  
            message = "Just kidding .... :D";  
            break;  
    }  
    Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();  
}
```

AlertDialog com lista

Adicionar lista

Existem três tipos de listas em AlertDialogs:

1. Uma lista tradicional de escolha única;
2. Uma lista persistente de uma escolha única (radio buttons);
3. Uma lista persistente de múltipla escolha (checkboxes);

AlertDialog com lista

1. Lista tradicional de escolha única

```
@NonNull  
@Override  
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
  
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
    builder.setTitle(R.string.pick_colour)  
        .setItems(R.array.colours_array, this);  
  
    return builder.create();  
}  
  
// The 'which' argument contains the index position  
// of the selected item
```

```
@Override  
public void onClick(DialogInterface dialog, int which) {  
}
```

Pick a Colour

Red

Green

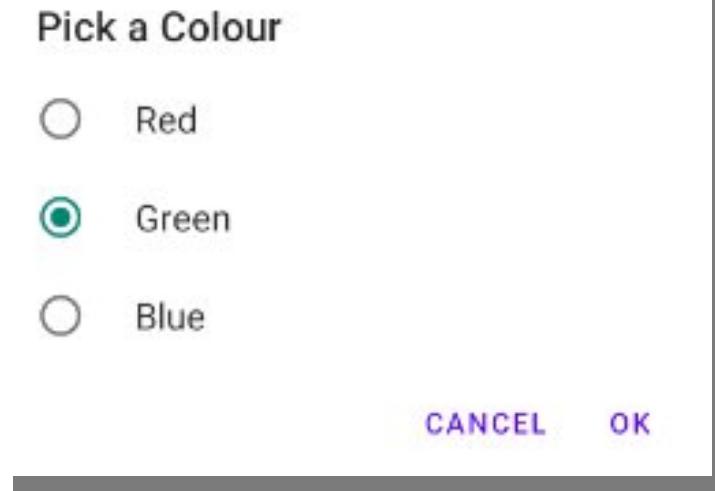
Blue

AlertDialog com lista

2. Lista persistente de uma escolha única (radio buttons)

```
@NonNull  
@Override  
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
  
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
  
    builder.setTitle(R.string.pick_colour)  
        .setSingleChoiceItems(R.array.colours_array, -1, this)  
        .setPositiveButton(android.R.string.ok, this)  
        .setNegativeButton(android.R.string.cancel, this);  
  
    return builder.create();  
}
```

```
// The 'which' argument contains the index position  
// of the selected item  
@Override  
public void onClick(DialogInterface dialog, int which) {  
}
```



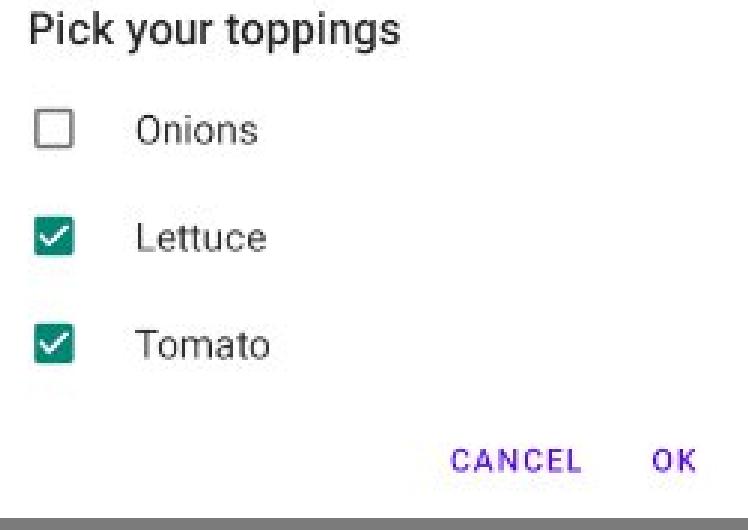
AlertDialog com lista

3. Lista persistente de múltipla escolha (checkboxes)

Implementa o
OnMultiChoiceClickListener para detetar os items
que são selecionados

```
public class ListDialogFragment extends DialogFragment  
    implements DialogInterface.OnClickListener, DialogInterface.OnMultiChoiceClickListener {
```

```
@NotNull  
@Override  
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
    selectedItems = new ArrayList(); // Variável para guardar os items selecionados  
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
    builder.setTitle(R.string.pick_toppings)  
        .setMultiChoiceItems(R.array.toppings_array, null, this)  
        .setPositiveButton(android.R.string.ok, this)  
        .setNegativeButton(android.R.string.cancel, this);  
  
    return builder.create();  
}
```



AlertDialog com lista

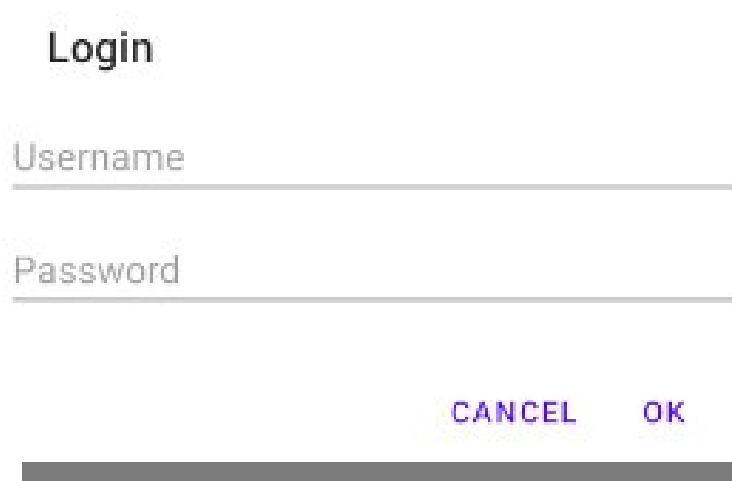
3. Lista persistente de múltipla escolha (checkboxes) (2)

```
// Método onClick para os botões
@Override
public void onClick(DialogInterface dialog, int which) {
    //TODO: Implementar comportamento dos botões OK e Cancel
}

// Método onClick para os items da lista
@Override
public void onClick(DialogInterface dialog, int which, boolean isChecked) {
    if(isChecked){
        selectedItems.add(which); // Se o item foi selecionado é adicionado
    }else if(selectedItems.contains(which)){
        selectedItems.remove(Integer.valueOf(which)); //Se o item já tiver no array, é removido
    }
}
```

AlertDialog

Layout Personalizado



Dialog Personalizado

dialog_login.xml

```
<?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:orientation="vertical"
    android:layout_gravity="center">

    <EditText
        android:id="@+id/username"
        android:inputType="textEmailAddress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/username" />

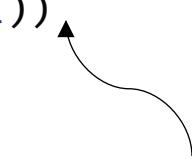
    <EditText
        android:id="@+id/password"
        android:inputType="textPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fontFamily="sans-serif"
        android:hint="@string/password" />

</LinearLayout>
```

Dialog Personalizado

LoginDialogFragment.java

```
@NonNull  
@Override  
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
  
    LayoutInflator inflater = requireActivity().getLayoutInflator();  
  
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
  
    return builder.setTitle(R.string.dlg_title)  
        .setView(inflater.inflate(R.layout.dialog_login, null))  
        .setPositiveButton(android.R.string.ok, this)  
        .setNegativeButton(android.R.string.cancel, null)  
        .create();  
}
```



Inflate do layout
personalizado do nosso
Dialog

Processar eventos do Dialog na Activity

LoginDialogFragment.java

```
public interface NoticeDialogListener {  
    void onDialogPositiveClick(DialogFragment dialog);  
    void onDialogNegativeClick(DialogFragment dialog);  
}  
  
private NoticeDialogListener listener;  
  
@Override  
public void onAttach(@NotNull Context context) {  
    super.onAttach(context);  
    try {  
        listener = (NoticeDialogListener) context;  
    } catch (ClassCastException e) {  
        throw new ClassCastException(getActivity().toString()  
            + " must implement NoticeDialogListener");  
    }  
}
```

Processar eventos do Dialog na Activity

LoginDialogFragment.java

```
@Override  
public void onClick(DialogInterface dialog, int which) {  
    if(listener != null){  
        listener.onDialogPositiveClick(LoginDialogFragment.this);  
        switch(which){  
            case DialogInterface.BUTTON_POSITIVE:  
                listener.onDialogPositiveClick(LoginDialogFragment.this);  
                break;  
            case DialogInterface.BUTTON_NEGATIVE:  
                listener.onDialogNegativeClick(LoginDialogFragment.this);  
                break;  
        }  
    }  
}
```

Processar eventos do Dialog na Activity

MainActivity.java

Implementa a interface que definimos no DialogFragment

```
public class MainActivity extends AppCompatActivity
    implements View.OnClickListener, LoginDialogFragment.NoticeDialogListener {

    @Override
    public void onDialogPositiveClick(DialogFragment dialog) {
        //TODO: Ação que acontece quando carregamos em OK
    }

    @Override
    public void onDialogNegativeClick(DialogFragment dialog) {
        //TODO: Ação que acontece quando carregamos em Cancel
    }
}
```

Leitura Adicional

- **Dialogs**

<https://developer.android.com/guide/topics/ui/dialogs>

Programação Para Dispositivos Móveis I

DIALOGS

2023/_24 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

AppBar, Toolbar e NAVIGATION DRAWERS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

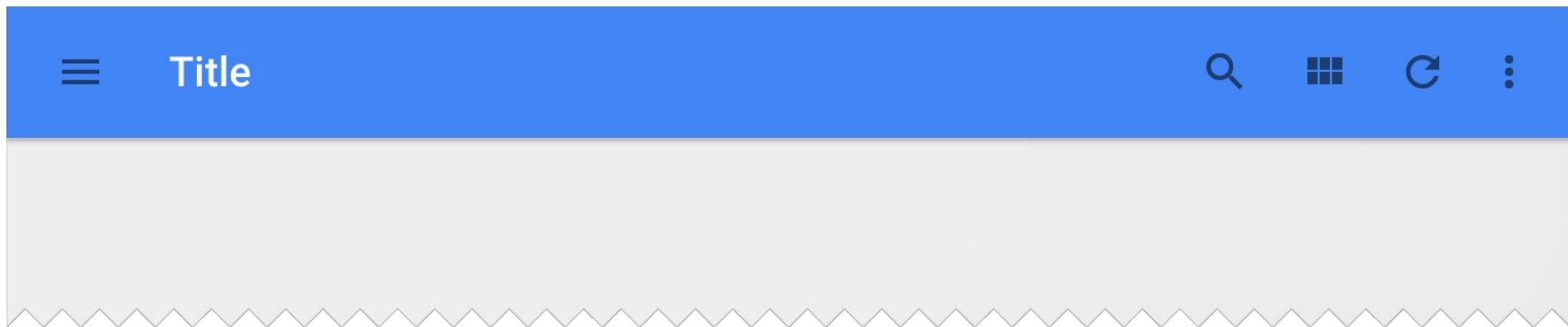
Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Índice

- AppBar;
- Criar uma AppBar;
- AppBar Menu;
- Navigation Drawer;
- Leitura Adicional.

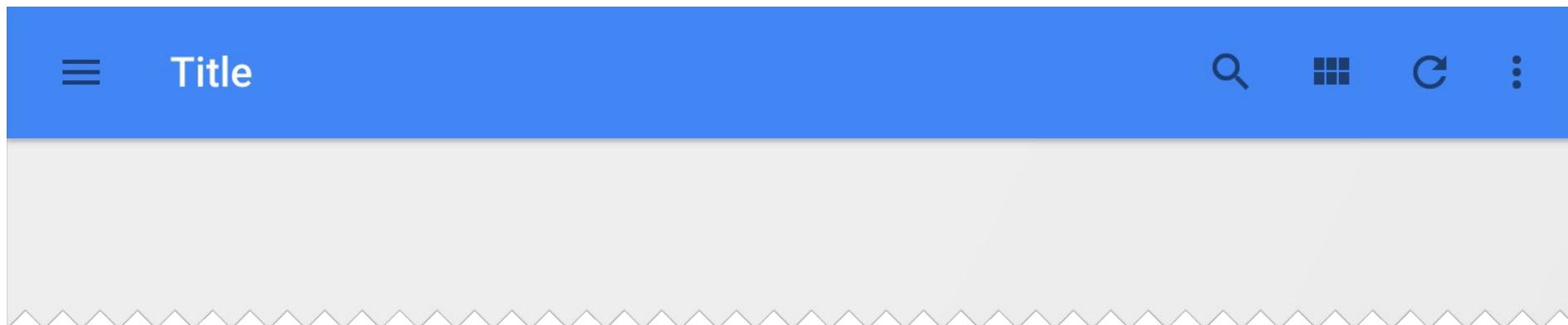
App Bar & Toolbar



App Bar

A App Bar foi introduzida no Android Lollipop (API 21) e é o sucessor da ActionBar.

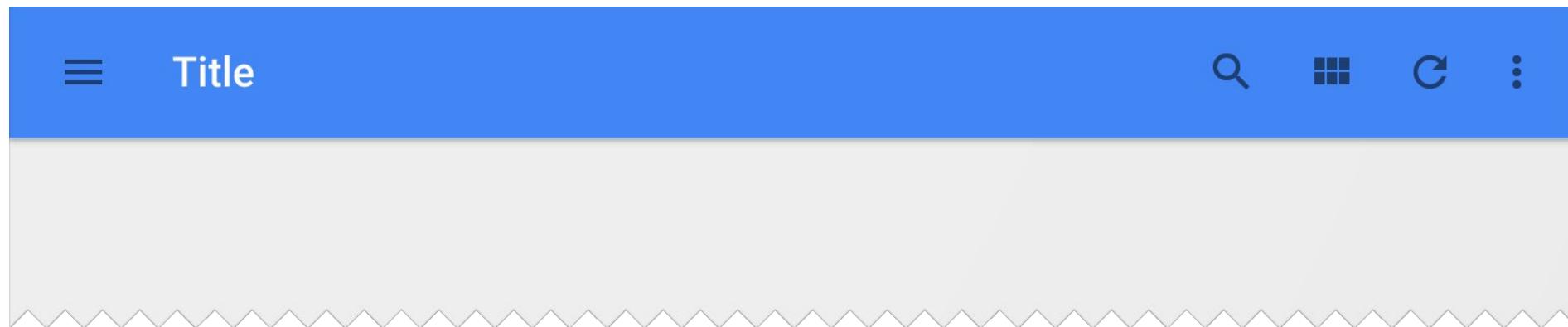
- É um dos principais elementos de design de uma activity pois define uma estrutura visual que é familiar para o utilizador;
- Ao utilizar a App Bar as aplicações funcionam todas de forma consistente e similar;



App Bar

■ Principal objetivo:

- Indicar a localização do utilizador na aplicação (navegação);
- Acesso a ações comuns a toda a aplicação (ex.: procura);
- Suportar diferentes conceitos de navegação: tabs ou dropdown lists.



App Bar

Estrutura App Bar

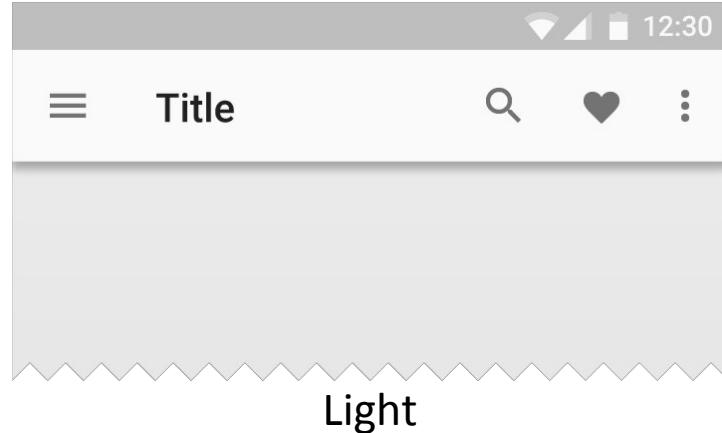


Nav icon: pode controlar a abertura/fecho do NavigationDrawer ou uma seta de navegação;

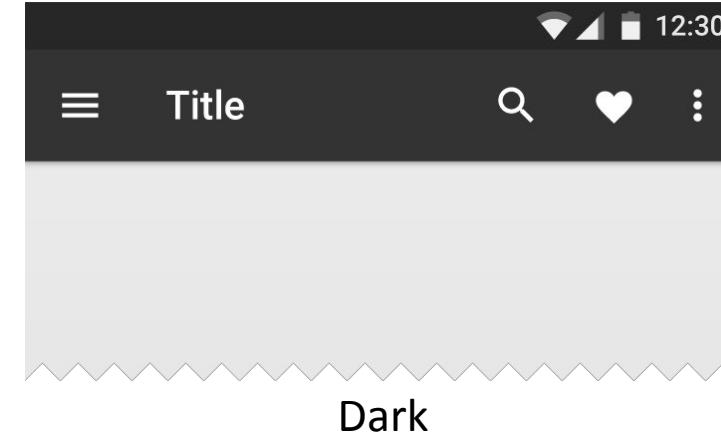
Title: normalmente reflete a pagina atual do utilizador (pode ser o titulo da app, titulo da activity ou filtro);

Action Icons: contem ações de fácil acesso ao utilizador: procura, definições, etc.

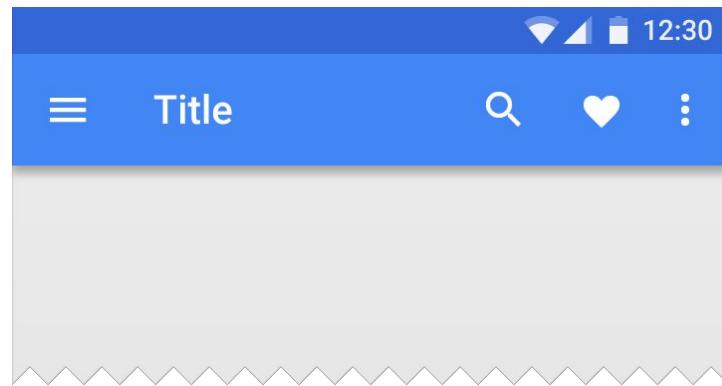
App Bar Themes



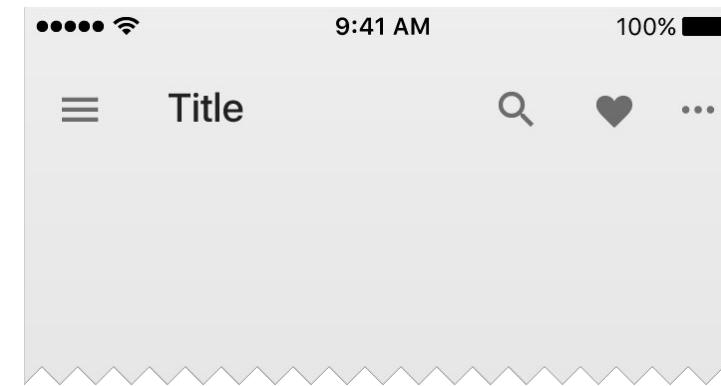
Light



Dark



Colored



Transparent

Adicionar AppBar

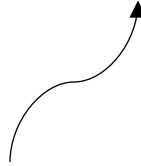
1. Verificar as dependências no ficheiro build.gradle(module)

```
dependencies {  
    ( ... )  
    implementation 'androidx.appcompat:appcompat:1.7.0'  
    ( ... )  
}
```

Adicionar AppBar

2. Modificar o tema da aplicação [themes -> themes.xml]

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.AppTheme" parent="Theme.MaterialComponents.DayNight.NoActionBar">
        ( ... )
    </style>
</resources>
```



Como vamos substituir a
ActionBar, necessitamos de
um tema que não a inclua

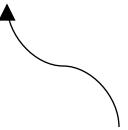
Adicionar AppBar

3. Adicionar a toolbar ao layout da activity [activity_main.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:orientation="vertical"
    tools:context=".MainActivity">

    <androidx.appcompat.widget.Toolbar
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        android:elevation="4dp" />

</LinearLayout>
```



O atributo elevation faz parte das recomendações do Material Design

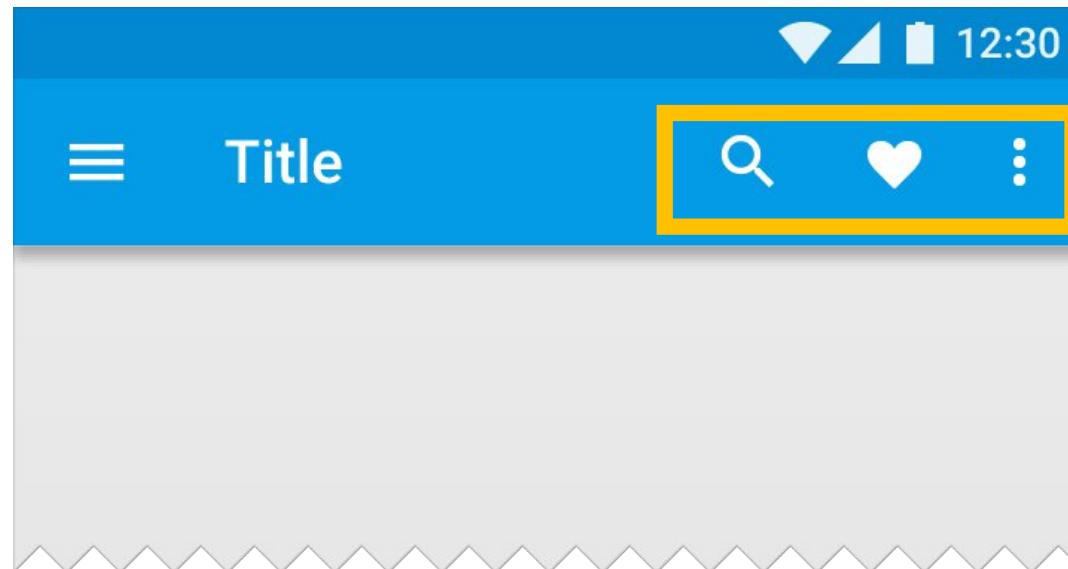
Adicionar AppBar

4. Adicionar a toolbar [>MainActivity.java]

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);  
    setSupportActionBar(toolbar);  
  
}  
}
```

AppBar Menu

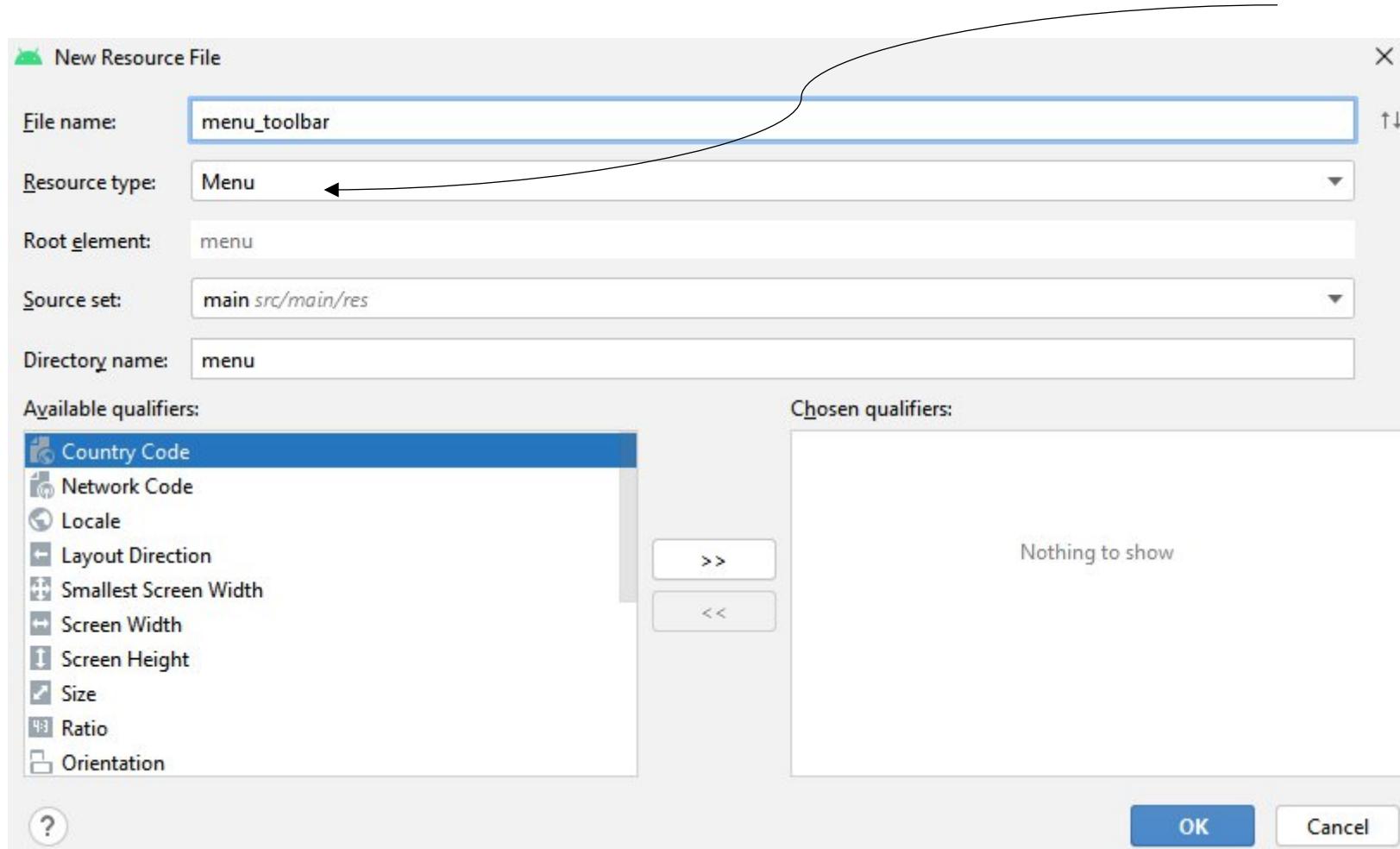
- A Toolbar permite definir um conjunto de ações através de um conceito chamado “Menu”.
 - Cada activity ou fragment é responsável por definir um menu e de que forma são mostradas essas ações (icon ou dropdown);



AppBar Menu

1. Criar ficheiro menu[res/menu/menu_toolbar.xml]

Garantir que escolhemos
Menu como valor de
Resource Type



AppBar Menu

2. Adicionar items [res/menu/menu_toolbar.xml]

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <!-- "Mark Favorite", should appear as action button if possible -->
    <item
        android:id="@+id/action_favorite"
        android:icon="@drawable/ic_favorite_black_48dp"
        android:title="@string/action_favorite"
        app:showAsAction="ifRoom" /> ←

    <item android:id="@+id/action_search"
        android:title="@string/action_search"
        android:icon="@drawable/ic_search"
        app:showAsAction="ifRoom|collapseActionView"
        app:actionViewClass="androidx.appcompat.widget.SearchView" />

    <!-- Settings, should always be in the overflow -->
    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        app:showAsAction="never" /> ←
</menu>
```

Este tipo de ação indica que o item irá aparecer na AppBar sempre que haja espaço disponível

Este tipo de ação indica que o item apenas irá sempre aparecer na zona de overflow

AppBar Menu

3. Adicionar o menu [MainActivity.java]

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.menu_toolbar, menu);  
    return super.onCreateOptionsMenu(menu);  
}
```

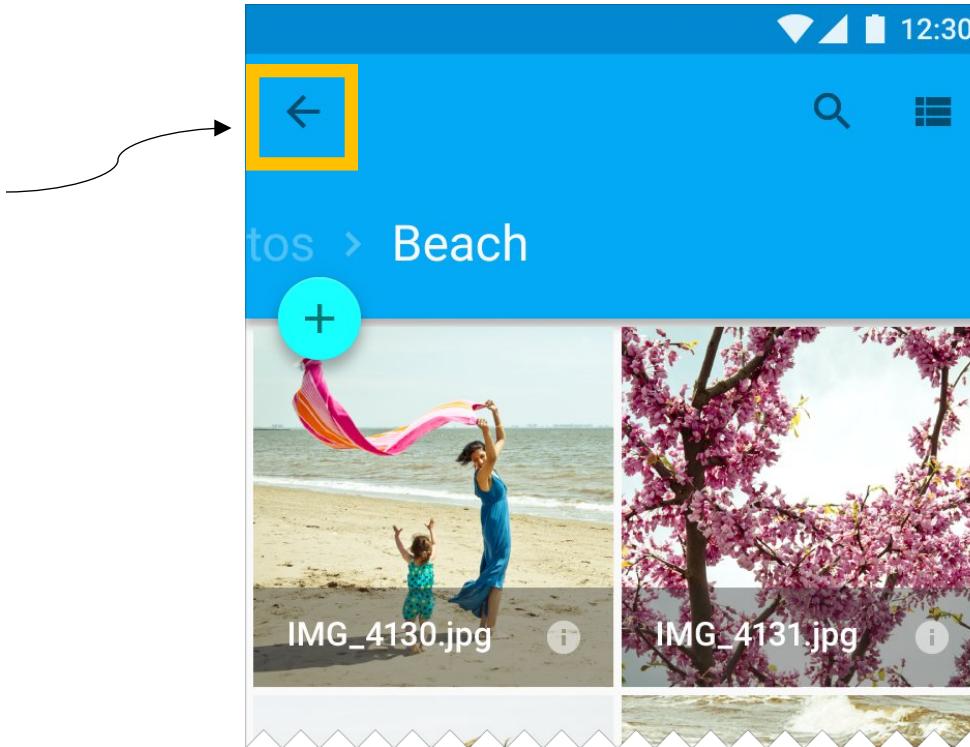
AppBar Menu

4. Adicionar comportamento aos items [MainActivity.java]

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action_settings:  
            // User chose the "Settings" action, should be prompted  
            // with settings activity  
            return true;  
  
        case R.id.action_favorite:  
            // User chose the "Favorite" action, mark the current item  
            // as a favorite...  
            return true;  
  
        default:  
            // If we got here, the user's action was not recognized.  
            // Invoke the superclass to handle it.  
            return super.onOptionsItemSelected(item);  
    }  
}
```

AppBar Menu (Up Action)

Permite ao utilizador navegar para o ecrã anterior



AppBar Menu (Up Action)

1. Reutilizar a AppBar [toolbar.xml]

```
< androidx.appcompat.widget.Toolbar  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="?attr/actionBarSize"  
    android:background="?attr/colorPrimary"  
    android:elevation="4dp" />
```

AppBar Menu (Up Action)

2. Incluir a toolbar na Activity [activity_settings.xml]

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

    <include
        android:id="@+id/my_toolbar"
        layout="@layout/toolbar" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/text_settings" />
</LinearLayout>
```

Inclusão do layout da AppBar

AppBar Menu (Up Action)

3. Adicionar toolbar na Activity [SettingsActivity.java]

Modificação do
título da toolbar

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_settings);  
  
    Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);  
    toolbar.setTitle("Settings");  
    setSupportActionBar(toolbar);  
  
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
}
```

Adição do botão
UpAction

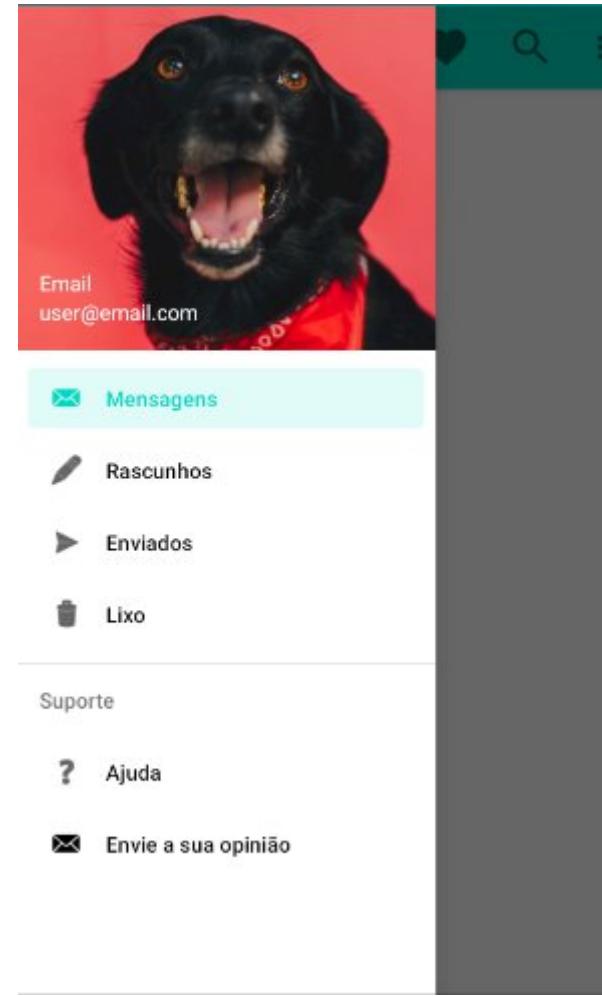
AppBar Menu (Up Action)

4. Adicionar Hierarquia [AndroidManifest.xml]

```
<activity android:name=".SettingsActivity"
    android:parentActivityName=".MainActivity">
    <!-- Parent activity meta-data to support 4.0 and lower -->
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity" />
</activity>
```

O click na UpAction irá retornar à MainActivity

Navigation Drawer



Navigation Drawer

1. Definir o header [nav_header.xml]

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="230dp">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
        android:src="@drawable/profile_pic" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="bottom|left" <-->
        android:layout_margin="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/app_name"
            android:textAppearance="@style/TextAppearance.AppCompat.Body1"
            android:textColor="@color/white" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/user_name"
            android:textColor="@color/white" />
    </LinearLayout>
</FrameLayout>
```

Este atributo coloca
o texto no canto
inferior esquerdo da
imagem

Navigation Drawer

2. Adicionar items [menu_nav.xml]

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

    <group android:checkableBehavior="single"> ←
        <item
            android:id="@+id/nav_inbox"
            android:icon="@android:drawable/sym_action_email"
            android:title="@string/nav_inbox"
            android:checked="true"/> ←
        <item
            android:id="@+id/nav_drafts"
            android:icon="@android:drawable/ic_menu_edit"
            android:title="@string/nav_drafts"/>
        <item
            android:id="@+id/nav_sent"
            android:icon="@android:drawable/ic_menu_send"
            android:title="@string/nav_sent"/>
        <item
            android:id="@+id/nav_trash"
            android:icon="@android:drawable/ic_menu_delete"
            android:title="@string/nav_trash"/>
    </group>

    (...)

</menu>
```



O item que vai estar selecionado por defeito

Navigation Drawer

2. Adicionar items [menu_nav.xml]

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

(...)

<item android:title="@string/nav_support">
    <menu>
        <item
            android:id="@+id/nav_help"
            android:icon="@android:drawable/ic_menu_help"
            android:title="@string/nav_help" />
        <item
            android:id="@+id/nav_feedback"
            android:icon="@android:drawable/sym_action_email"
            android:title="@string/nav_feedback" />
    </menu>
</item>
</menu>
```

Suporte

? Ajuda

✉ Envie a sua opinião

Navigation Drawer

3. Adicionar a Navigation Drawer ao Layout [activity_main.xml]

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <include
            android:id="@+id/my_toolbar"
            layout="@layout/toolbar" />
        <FrameLayout
            android:id="@+id/fragment_container"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </LinearLayout>

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="left"
        app:headerLayout="@layout/nav_header"
        app:menu="@menu/menu_nav" />
</androidx.drawerlayout.widget.DrawerLayout>
```

O elemento de Root é um DrawerLayout

O "antigo" layout que tínhamos

Definição de um container para os fragments

Definição da NavigationView

Definição do layout e dos items do menu da NavigationView

Navigation Drawer

4. Adicionar a ação de toggle [MainActivity.java]

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    (...)  
  
    drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);  
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,  
        drawerLayout,  
        toolbar,  
        R.string.nav_open_drawer,  
        R.string.nav_close_drawer);  
  
    drawerLayout.addDrawerListener(toggle);  
  
    toggle.syncState();  
  
}
```

O elemento
DrawerLayout

A toolbar previamente
criada

Navigation Drawer

5. Responder aos clicks [MainActivity.java]

Implementar a interface do listener

```
public class MainActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
  
    (...)  
  
    NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);  
    navigationView.setNavigationItemSelectedListener(this); ← Adicionar o listener  
}
```

Navigation Drawer

5. Responder aos clicks [MainActivity.java]

```
@Override  
public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
    Fragment fragment = null;  
  
    switch (item.getItemId()) {  
        case R.id.nav_drafts:  
            fragment = new DraftsFragment();  
            break;  
        case R.id.nav_trash:  
            fragment = new TrashFragment();  
        case R.id.nav_sent:  
            fragment = new SentFragment();  
            break;  
        default:  
            fragment = new InboxFragment();  
    }  
    if(fragment != null){  
        fragmentTransaction = getSupportFragmentManager().beginTransaction();  
        fragmentTransaction.replace(R.id.fragment_container, fragment);  
        fragmentTransaction.addToBackStack(null);  
        fragmentTransaction.commit();  
    }  
    drawerLayout.closeDrawer(GravityCompat.START);  
  
    return true;  
}
```

Navigation Drawer

6. Fechar a Navigation quando se utiliza o Back Button [MainActivity.java]

```
@Override  
public boolean onSupportNavigateUp() {  
    onBackPressed();  
    return super.onSupportNavigateUp();  
}
```

O método é chamado quando se carrega no Back Button

```
@Override  
public void onBackPressed() {  
    if(drawerLayout.isDrawerOpen(GravityCompat.START)){  
        drawerLayout.closeDrawer(GravityCompat.START);  
    }else{  
        super.onBackPressed();  
    }  
}
```

Se a Drawer estiver aberta, então dizemos a aplicação para a fechar

Leitura Adicional

- Appbar:

<https://developer.android.com/training/appbar>

- Navigation Drawer:

<https://developer.android.com/guide/navigation/navigation-ui>

Programação Para Dispositivos Móveis I

AppBar, Toolbar e NAVIGATION DRAWERS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

PREFERENCES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Índice

- Shared Preferences;
- Preference Screen;
- Preference Fragment;
- Leitura Adicional.

Utilização de Preferências

As preferências podem ser definidas para uma única atividade, ou serem partilhadas entre todas as atividades numa aplicação. Outros componentes, como serviços, também podem utilizar as preferências partilhadas (`sharedPreferences`).

Para aceder às preferências podemos escolher uma de três APIs:

- `getPreferences()`: Para aceder a preferências específicas de uma atividade;
- `getSharedPreferences()`: Para aceder às preferências partilhadas por toda a aplicação;
- `getDefaultsSharedPreference()`: utilizando um `PreferenceManager`, permite aceder a preferências que também estão relacionadas com a framework global de preferências do Android.

Utilização de Preferências

A utilização das APIs `getPreferences()` e `getSharedPreferences()` requerem um parâmetro de modo de segurança. Deve ser utilizado o `MODE_PRIVATE` para que nenhuma outra aplicação tenha acesso a este ficheiro de preferências.

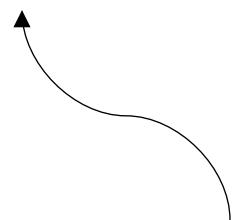
- Tipicamente é utilizada a API `getDefaultSharedPreferences()`.

Utilização de Preferências

Editor

O editor permite interagir com a lista de preferências presente no objeto SharedPreferences. Cada preferência é declarada definida por um par **chave -> valor** (chave é uma String e valor pode ser de qualquer tipo primitivo)

```
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);  
SharedPreferences.Editor editor = preferences.edit();
```



Instanciação do Editor de
SharedPreferences

Utilização de Preferências

Editor

Para além de permitir adicionar preferências, o editor também permite efetuar os seguintes métodos:

- `remove()`: para remover uma preferência;
- `clear()`: para remover todas as preferências;
- `apply()` or `commit()`: para persistir as alterações feitas pelo editor.

Utilização de Preferências

Obter preferência através da chave (KEY)

```
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);  
preferences.getString("CHAVE", "VALOR_POR_DEFEITO");
```

Para além do método `getString()`, também temos disponíveis os métodos `getFloat()`, `getInt()`, `getBoolean()`, `getLong()`.



Utilização de Preferências

PreferenceFragment

Podemos também definir um conjunto de preferências específicas da nossa aplicação e associadas ao contexto da mesma.

Para isso, temos de indicar ao Android quais as preferências que estamos a tentar recolher.

Esse processo é descrito de seguida.

Notifications

Show Notifications

Display notifications when receiving a new email



Appearance

Display custom username

Display a custom username instead of an email address



Select username

Choose a custom username to display in app

Show last X emails

Choose the amount of emails you want to be displayed

Select Folders

Choose which folders you want to be displayed

Region & Language

Select application language

PreferenceFragmentCompat

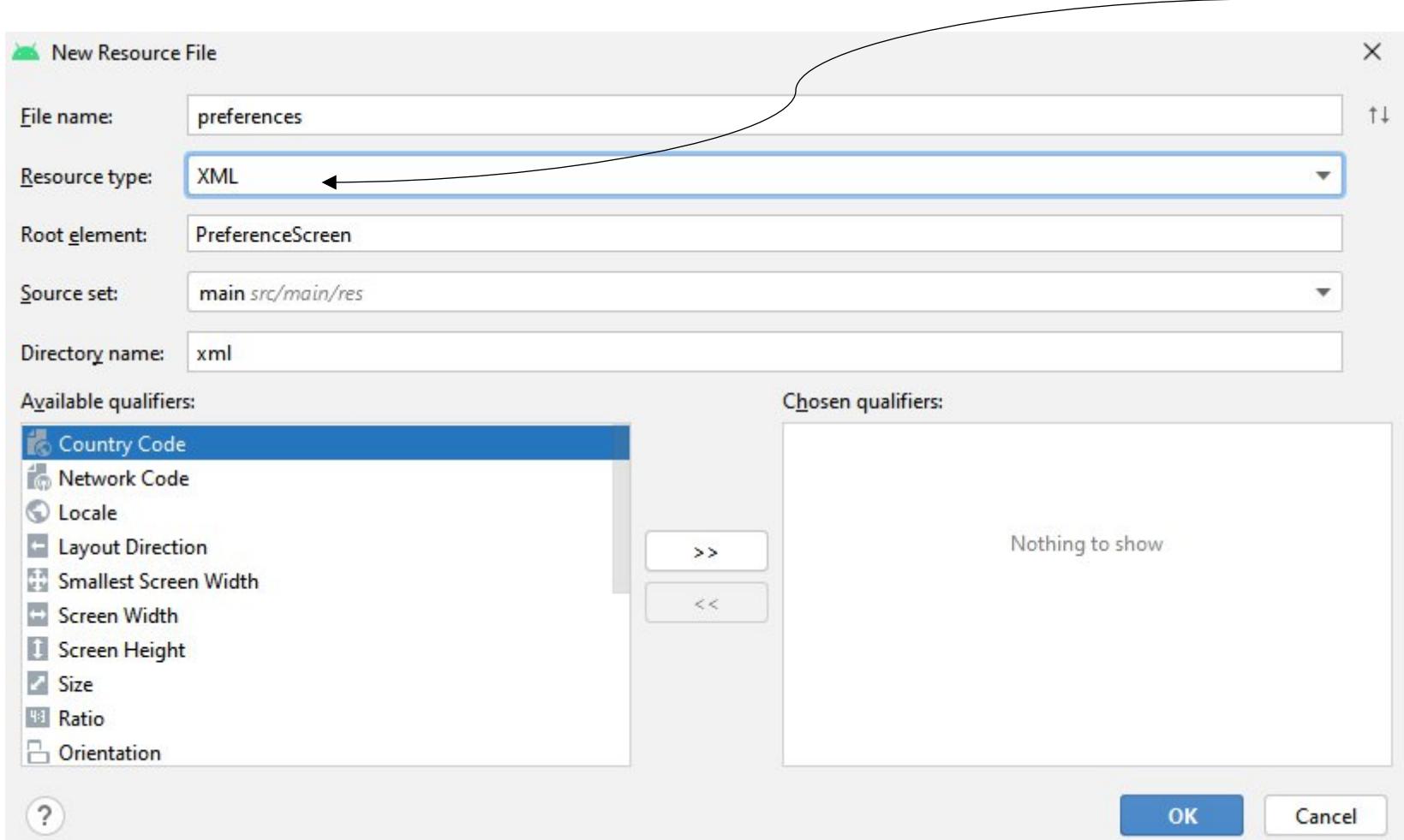
1. Adicionar as dependências ao ficheiro build.gradle(module)

```
dependencies {  
    ( ... )  
    implementation 'androidx.preference:preference:1.2.1'  
    ( ... )  
}
```

PreferenceFragmentCompat

2. Criar o ficheiro de preferências [xml/preferences.xml]

Garantir que escolhemos XML como valor de Resource Type



PreferenceFragmentCompat

2. Adicionar preferências [xml/preferences.xml]

PreferenceScreen é o
element ROOT do ficheiro
de preferências

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

    <CheckBoxPreference
        android:key="@string/pref_notification_key"
        android:title="@string/pref_notification_title"
        android:summary="@string/pref_notification_summary"
        android:defaultValue="false" />

</PreferenceScreen>
```

Exemplo de uma
preferência

PreferenceFragmentCompat

3. Adicionar as preferências ao Fragment [PreferencesFragment.java]

```
public class PreferencesFragment extends PreferenceFragmentCompat
```



O fragment extende
PreferenceFragmentCompat

```
@Override  
public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {  
    setPreferencesFromResource(R.xml.preferences, rootKey);  
}
```



Tipos de Preferência

CheckBoxPreference

Show Notifications

Display notifications when receiving a
new email



```
<CheckBoxPreference  
    android:key="@string/pref_notification_key"  
    android:title="@string/pref_notification_title"  
    android:summary="@string/pref_notification_summary"  
    android:defaultValue="false" />
```

Key e Title são os valores
obrigatórios para todas as
preferências

Tipos de Preferência

PreferenceCategory

```
<PreferenceCategory  
    android:title="@string/category_notifications">  
  
    <CheckBoxPreference  
        android:key="@string/pref_notification_key"  
        android:title="@string/pref_notification_title"  
        android:summary="@string/pref_notification_summary"  
        android:defaultValue="false" />  
</PreferenceCategory>
```

Podemos categorizar um conjunto de preferências através deste elemento. O único valor obrigatório é o android:title

Notifications

Show Notifications

Display notifications when receiving a new email



Tipos de Preferência

SwitchPreferenceCompat

Display custom username

Display a custom username instead of an email address



```
<SwitchPreferenceCompat  
    android:key="@string/pref_username_key"  
    android:title="@string/pref_username_title"  
    android:summary="@string/pref_username_summary"  
    android:defaultValue="false" />
```

Semelhante à CheckBoxPreference mas utiliza um slider para classificar como ativo ou inativo

Podemos definir um valor por defeito para cada preferência

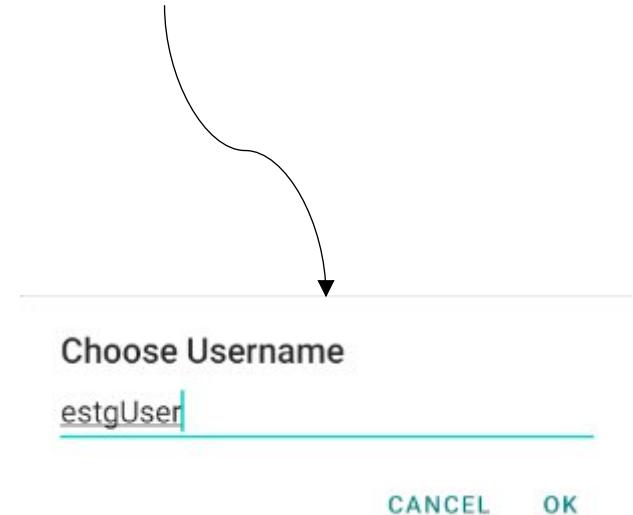
Tipos de Preferência

EditTextPreference

```
<EditTextPreference  
    android:key="@string/pref_setusername_key"  
    android:title="@string/pref_setusername_title"  
    android:summary="@string/pref_setusername_summary"  
    android:dialogTitle="@string/pref_setusername_dialog" />
```

Podemos definir o valor de alguns elementos da dialog (ex. Titulo)

Select username
Choose a custom username to display in app



Tipos de Preferência

EditTextPreference

Display custom username

Display a custom username instead of an email address

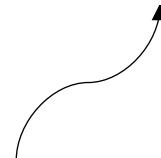


Select username

Choose a custom username to display in app

<EditTextPreference

```
    android:key="@string/pref_setusername_key"
    android:title="@string/pref_setusername_title"
    android:summary="@string/pref_setusername_summary"
    android:dialogTitle="@string/pref_setusername_dialog"
    android:dependency="@string/pref_username_key" />
```



Podemos criar dependências entre preferências.

Neste exemplo esta preferência apenas está ativa se a opção da preferência indicada for True

Tipos de Preferência

ListPreference

```
<ListPreference  
    android:key="@string/pref_language_key"  
    android:title="@string/pref_language_title"  
    android:summary="@string/pref_language_summary"  
    android:entries="@array/language_options"  
    android:entryValues="@array/language_values"  
    android:defaultValue="@string/default_language_value"  
    android:dialogTitle="@string/pref_language_dialog" />
```

Select application language
Select your preferred language

Language

- EN - English
- PT - Português

CANCEL

Lista de valores que aparece ao utilizador

Lista de valores que a aplicação vai utilizar

Tipos de Preferência

ListPreference [res/values/strings.xml]

```
<string-array name="language_options">
    <item>EN - English</item>
    <item>PT - Português</item>
</string-array>
```

Lista de valores que aparece ao utilizador

```
<string-array name="language_values">
    <item>EN</item>
    <item>PT</item>
</string-array>
```

Lista de valores que a aplicação vai utilizar

Tipos de Preferência

DropdownPreference

Semelhante à ListPreference mas utiliza uma dropdown em vez de uma dialog

```
<DropDownPreference  
    android:key="@string/pref_language_dropdown_key"  
    android:title="@string/pref_language_title"  
    android:summary="@string/pref_language_summary"  
    android:entries="@array/language_options"  
    android:entryValues="@array/language_values"  
    android.defaultValue="@string/default_language_value" />
```

Select application language
Select your preferred language



Tipos de Preferência

MultiSelectListPreference

Só está disponível em API's superiores à versão 11 (onze) (Android 3.0)

```
<MultiSelectListPreference  
    android:key="@string/pref_show_folder_key"  
    android:title="@string/pref_show_folder_title"  
    android:summary="@string/pref_show_folder_summary"  
    android:entries="@array/folder_options"  
    android:entryValues="@array/folder_values"  
    android:dialogTitle="@string/pref_show_folder_dialog" />
```

Select Folders

Choose which folders you want to be displayed

Which folder(s) you want to be displayed?

- Inbox
- Drafts
- Sent Email
- Spam
- Trash

CANCEL OK

Tipos de Preferência

SeekBarPreference

Notification Volume

Select the volume of the applications sound

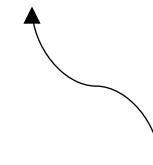


```
<SeekBarPreference  
    android:key="@string/pref_volume_key"  
    android:title="@string/pref_volume_title"  
    android:summary="@string/pref_volume_summary" />
```

PreferenceFragmentCompat

4. Registar o listener [PreferencesFragment.java]

```
public class PreferencesFragment extends PreferenceFragmentCompat implements SharedPreferences.OnSharedPreferenceChangeListener
```



Implementar o listener que verifica se existiram alterações nas preferências

PreferenceFragmentCompat

4. Registrar o listener [PreferencesFragment.java]

```
@Override  
public void onResume() {  
    super.onResume();  
    getPreferenceScreen().getSharedPreferences().registerOnSharedPreferenceChangeListener(this);  
}  
  
@Override  
public void onPause() {  
    super.onPause();  
    getPreferenceScreen().getSharedPreferences().unregisterOnSharedPreferenceChangeListener(this);  
}
```

PreferenceFragmentCompat

4. Registrar o listener [PreferencesFragment.java]

Select username
Choose a custom username to display in app



Select username
Choose a custom username to display in app.
Chosen username: estgUser

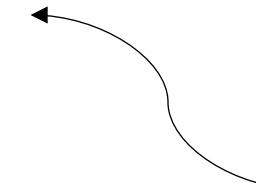
```
@Override
public void onSharedPreferenceChanged(SharedPreferences sharedpreferences, String key) {
    if(key.equals(getResources().getString(R.string.pref_setusername_key))){
        EditTextPreference preferenceCustomUsername = findPreference(key);
        preferenceCustomUsername.setSummary(
            preferenceCustomUsername.getSummary() + ". Chosen username: " +
            sharedpreferences.getString(key, ""));
    }
}
```

Comparação de
keys

PreferenceFragmentCompat

Aceder aos objetos de preferências

```
ListPreference languagesPref =  
    (ListPreference) findPreference(getResources().getString(R.string.pref_language_key));  
  
languagesPref.setEntries(R.array.language_options);  
languagesPref.setEntryValues(R.array.language_values);
```



Definição de
valores através
de código

Leitura Adicional

Settings:

<https://developer.android.com/guide/topics/ui/settings>

Programação Para Dispositivos Móveis I

PREFERENCES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

FICHEIROS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Índice

- Ficheiros;
- Armazenamento Interno;
- Armazenamento Externo;
- Notas Finais;
- Leitura Adicional.

Ficheiros

Em conjunto com SharedPreferences e Databases, Ficheiros são um de três métodos de **persistência de dados** em Android.

- A sua utilização **não é recomendada** (geralmente);
- O Android utiliza os mesmos mecanismos de manipulação de ficheiros do Java.

Os ficheiros podem ser armazenados em:

- Armazenamento interno (Ficam na mesma localização que os outros recursos como os ícones, imagens, música,...);
- Armazenamento externo (ex. cartão SD).

Ficheiros

Armazenamento Interno

Os ficheiros gravados neste tipo de armazenamento são privados à aplicação (tal como os resources), e as outras aplicações (e o utilizador) não conseguem obter acesso aos mesmos.

- Quando o utilizador desinstala a aplicação estes ficheiros são também eliminados;

ATENÇÃO: O espaço Interno é muito limitado comparativamente com o externo.

Ficheiros

Armazenamento Interno –> Criação e Escrita

1. Invocar o método `openFileOutput()`

- Recebe como parâmetros o nome do ficheiro e o modo de operação;
 - `MODE_PRIVATE` – cria o ficheiro e torna-o privado à aplicação;
 - `MODE_APPEND` – se o ficheiro já existir então adiciona os dados no final deste em vez de o apagar;
- Retorna um objeto do tipo `FileOutputStream`;

2. Escrever o ficheiro utilizando o método `write()`;

3. Fechar o stream de dados utilizando o método `close()`.

Ficheiros

Armazenamento Interno -> Criação e Escrita

```
String file_name = "test_file.txt";
String textContent = "Hello World!";

FileOutputStream fos = null;

try {
    fos = openFileOutput(file_name, this.MODE_PRIVATE);
    fos.write(textContent.getBytes());

} catch (FileNotFoundException fileNotFoundException) {
    Log.e("FILE_NOT_FOUND", fileNotFoundException.getMessage());
} catch (IOException ioException) {
    Log.e("INPUT", ioException.getMessage());
} finally {
    if (fos != null)
        fos.close();
}
```

Ficheiros

Armazenamento Interno –> Leitura

1. Invocar o método `openFileInput()`
 - Recebe como parâmetro o nome do ficheiro a ler;
 - Retorna um objeto do tipo `FileInputStream`;
2. Ler os bytes do ficheiro utilizando o método `read()`;
3. Fechar o stream de dados utilizando o método `close()`;

Ficheiros

Armazenamento Interno -> Leitura

```
String file_name = "test_file.txt";
FileInputStream fileInputStream = null;
try {
    fileInputStream = openFileInput(file_name );
    int i;
    while((i=fileInputStream.read())≠-1) {
        ...
    }

} catch (FileNotFoundException fileNotFoundException) {
    Log.e("FILE_NOT_FOUND", fileNotFoundException.getMessage());
} catch (IOException ioException) {
    Log.e("INPUT", ioException.getMessage());
} finally {
    if (fileInputStream ≠ null)
        fileInputStream.close();
}
```

Ficheiros

Armazenamento Interno –> Outros métodos

getFilesDir()

- Obtém o caminho absoluto para o diretório onde os ficheiros são gravados;

getDir()

- Cria (ou abre caso exista) o diretório especificado;

deleteFile()

- Apaga o ficheiro;

fileList()

- Retorna um array com todos os ficheiros existentes no diretório.

Ficheiros

Armazenamento Externo

Este tipo de armazenamento pode ser do tipo **removível** (ex. SD Card) ou **embebido** no dispositivo. Neste tipo de armazenamento os ficheiros podem ser adicionados, editados, removidos e copiados pelo utilizador **não havendo controlo de segurança** sobre os mesmos.

- O armazenamento externo pode ficar indisponível caso o utilizador faça *mount* num computador ao armazenamento externo (ligação por USB)

Ficheiros

Armazenamento Externo -> Disponibilidade

Antes de fazer qualquer operação sobre o armazenamento externo deve-se verificar qual o estado do mesmo, utilizando o método `Environment.getExternalStorageState()`, o qual pode retornar:

- `Environment.MEDIA_MOUNTED` – o armazenamento está mounted a um computador e não pode ser utilizado;
- `Environment.MEDIA_MOUNTED_READ_ONLY` – só temos acesso de leitura ao armazenamento;
- `Environment.MEDIA_REMOVED` – armazenamento externo não está disponível;
- Entre outros...

Ficheiros

Armazenamento Externo -> Permissões [AndroidManifest.xml]

Permissões de leitura e escrita

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Permissões só de leitura

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Ficheiros

Armazenamento Externo -> Acesso

A path para este armazenamento pode ser obtida utilizando os métodos:

`getExternalFilesDir()`

- Recebe como parâmetro o tipo de diretório:
 - `DIRECTORY_MUSIC`;
 - `DIRECTORY_RINGTONES`;
 - `DIRECTORY_PICTURES`;
 - entre outros...
- Ou `null` caso seja de outro tipo;
- O ficheiro **é apagado** quando a aplicação é desinstalada.

`getExternalStoragePublicDirectory()`

- Recebe como parâmetro o tipo de diretório:
 - `DIRECTORY_MUSIC`;
 - `DIRECTORY_RINGTONES`;
 - `DIRECTORY_PICTURES`;
 - entre outros...
- Ou `null` caso seja de outro tipo;
- O ficheiro **não é apagado** quando a aplicação é desinstalada.

Ficheiros

Armazenamento Externo -> Exemplo

```
public File getAlbumStorageDir(String albumName) {  
  
    // Get the directory for the user's public pictures directory.  
    File file = new File(Environment.getExternalStoragePublicDirectory(  
        Environment.DIRECTORY_PICTURES), albumName);  
  
    if (!file.mkdirs()) {  
        Log.e("DIRECTORY", "Directory not created");  
    }  
  
    return file;  
}
```

Notas

Armazenamento Interno

Está sempre disponível

Ficheiros apenas acessíveis pela aplicação

Quando a aplicação é removida os ficheiros também são removidos

Armazenamento Externo

Não está sempre disponível. (Ex: Ligado ao PC, retirado o cartão memória, ...)

É visível a todas as aplicações android e pode ser acedido por fontes externas

Quando a aplicação é desinstalada, o sistema android só elimina o ficheiro se a diretoria onde foi gravado foi obtida através de `getExternalFilesDir()`

Leitura Adicional

Data and File Storage:

<https://developer.android.com/training/data-storage>

Programação Para Dispositivos Móveis I

FICHEIROS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

DATABASES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Índice

- Base de Dados em Android;
- SQLiteOpenHelper;
- Ligação à base de dados;
- Execução de SQL;
- Pesquisas SQL;
- Transações;
- Notas Finais;
- Leitura Adicional.

Base de Dados em Android

O Android fornece, nativamente, suporte para base de dados SQLite. Estas bases de dados apenas podem ser acedidas pela aplicação que as cria, nunca por aplicações terceiras;

- Para criar/gerir a base de dados deve-se criar uma subclasse de SQLiteOpenHelper com override aos métodos onCreate() e onUpgrade().

Utiliza tipos de atributos simples

- text, varchar, integer, float, numeric, date, time, timestamp, blob, boolean, ...

Base de Dados em Android

Considerações

Não armazenar ficheiros (imagens, áudio ou vídeo)

- Em alternativa armazenar os caminhos relativos num atributo do tipo string;

Funcionalidades:

- Criar bases de dados e respetivas tabelas;
- Índices;
- Queries;
- Vistas;
- Triggers (parcialmente);
- Inserir, eliminar e atualizar registo;
- Transações;
- Chaves estrangeiras (primeiro tem de se ativar)
 - db.execSQL("PRAGMA foreign_keys = ON;").

SQLiteOpenHelper

Esta classe vai auxiliar os processos de criação e manutenção da base de dados. Pode ser vista como um “assistente pessoal” que trata dos processos gerais de gestão da base de dados, nomeadamente:

- Criação da base de dados: O SQLiteHelper vai garantir que o ficheiro de base de dados é criado, com o nome correto, e correta estrutura de tabelas;
- Acesso à base de dados: A aplicação não precisa de saber todos os detalhes sobre a localização da base de dados, o SQLiteHelper fornece um objeto para obtermos acesso à base de dados sempre que necessário;
- Coerência a base de dados: É provável que a estrutura da base de dados sofra alterações ao longo do tempo, com o SQLiteHelper podemos converter uma versão antiga da base de dados para a versão mais atual.

SQLiteOpenHelper

Herança

onCreate()

- Executado quando a base de dados é criada (ficheiro com o nome definido em **DATABASE_NAME**);
- Permite a inicialização da base de dados (por ex.: criação das tabelas).

onUpgrade()

- Executado quando a versão da base de dados (**DATABASE_VERSION**) é alterada para um valor superior;
- Permite efetuar operações de atualização (por ex.: apagar e criar de novo as tabelas).

SQLiteOpenHelper

Estrutura [MyDBHelper.java]

```
public class MyDBHelper extends SQLiteOpenHelper {  
  
    private static final String DATABASE_NAME = "mydatabase.db";  
    private static final int DATABASE_VERSION = 1;  
  
    public MyDBHelper (Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) { ... }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { ... }  
}
```

Extende SQLiteOpenHelper

Nome da Base de Dados

Versão atual

Este parâmetro está relacionado com cursors (a abordar futuramente)

SQLiteOpenHelper

Criação Tabelas [MyDBHelper.java]

Sabemos que este é um valor calculável, mas vamos utilizar para efeitos de exemplo

```
CREATE TABLE PERSON (_id INTEGER PRIMARY KEY AUTOINCREMENT,  
                    NAME TEXT,  
                    AGE INTEGER)
```

Criação da tabela Person em SQLite

```
@Override  
public void onCreate(SQLiteDatabase db) {
```

Criação da tabela Person com SQLiteHelper

Este método executa instruções de SQL

```
}
```

```
        db.execSQL("CREATE TABLE PERSON(  
                  + "_id INTEGER PRIMARY KEY AUTOINCREMENT,"  
                  + "NAME TEXT,"  
                  + "AGE INTEGER);");
```

Em Android é convenção identificar os atributos que são chaves primárias como `_id`

SQLiteOpenHelper

Inserção de Dados [MyDBHelper.java]

- Para inserir dados numa tabela podem ser utilizados dois métodos.
Podemos criar uma string com o método execSQL() ou utilizar a operação de insert() presente no SQLiteHelper.

```
private static void insertPerson(SQLiteDatabase db, String name, int age){  
  
    Retorna o id da  
    linha inserida, ou  
    -1 em caso de  
    erro  
  
    ContentValues personValues = new ContentValues();  
    personValues.put("NAME", name);  
    personValues.put("AGE", age);  
  
    db.insert("PERSON", null, personValues);  
}
```

Nome da tabela

Conjunto de valores
(par chave - valor)

SQLiteOpenHelper

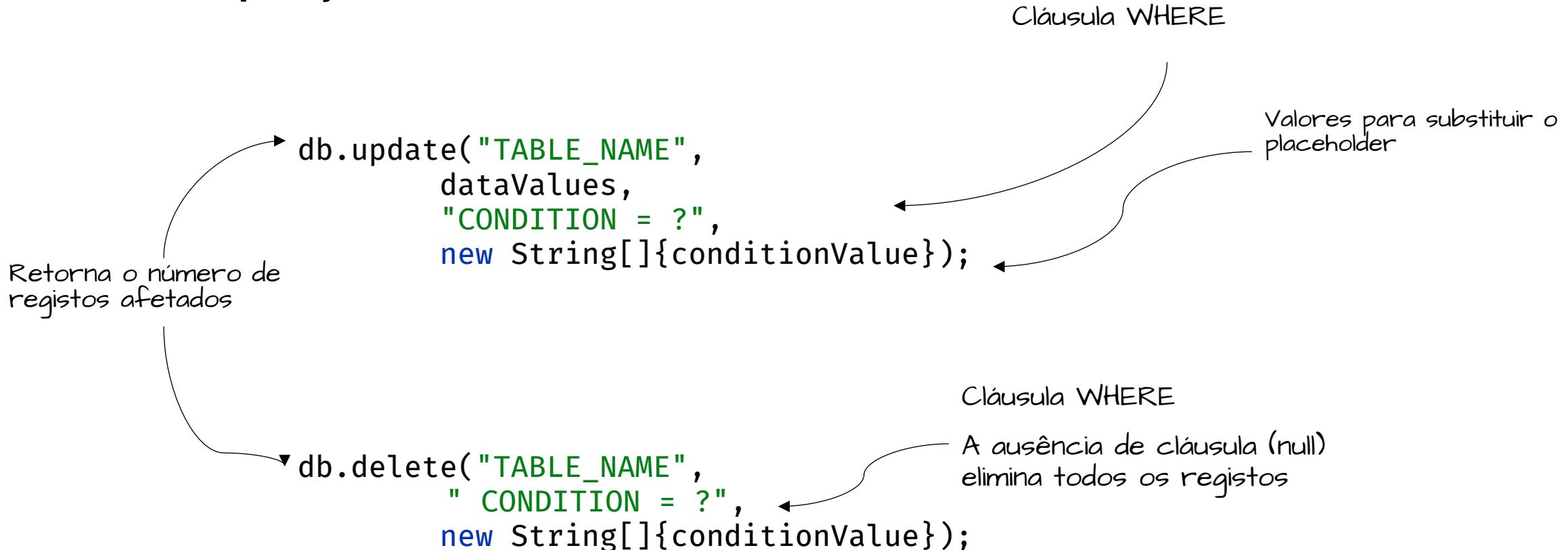
Inserção de Dados [MyDBHelper.java]

```
@Override  
public void onCreate(SQLiteDatabase db) {  
  
    db.execSQL("CREATE TABLE PERSON(  
        + "_id INTEGER PRIMARY KEY AUTOINCREMENT,"  
        + "NAME TEXT,"  
        + "AGE INTEGER);");  
  
    insertPerson(db, "John", 35);  
    insertPerson(db, "Chris", 18);  
}
```

Iniciar a BD com
dados

SQLiteOpenHelper

Outras operações de dados



Base de Dados

Execução de SQL -> execSQL()

```
db.execSQL("CREATE TABLE PESSOA (_id INTEGER PRIMARY KEY AUTOINCREMENT,  
name TEXT,  
age INTEGER);");
```

Cria a tabela PESSOA e
respetivos campos

```
db.execSQL("INSERT INTO PESSOA(name, age) VALUES ('John', 35)");
```

Insere uma linha de valores
na tabela PESSOA

```
db.execSQL("UPDATE PESSOA SET name = 'Dan' WHERE _id=1");
```

Atualiza dados na linha com
o atributo id = 1

```
db.execSQL("DELETE FROM PESSOA WHERE _id=1");
```

Remove o registo com o
atributo id = 1

NOTA - A invocação do método execSQL deve ser feito dentro de um **bloco try-catch-finally**.

É necessário ter em atenção potenciais situações onde uma exceção SQLiteException possa ser lançada

SQLiteOpenHelper

Obtenção de Dados (Pesquisa) -> rawQuery()

```
String sql = "SELECT * FROM PESSOA WHERE name = 'John'";  
Cursor cursor = db.rawQuery(sql, null);
```

Não tem argumentos de seleção
porque já estão definidos na query

```
String[] params = {"John"};  
String sql = "SELECT * FROM PESSOA WHERE name = ?";  
Cursor cursor = db.rawQuery(sql, params);
```

Os parâmetros substituem os
placeholders definidos na query

SQLiteOpenHelper

Obtenção de Dados (Pesquisa) -> query()

```
Cursor cursor = db.query(  
    Distinct false,  
    Nome da tabela "TABLE_NAME",  
    Nome das colunas selecionadas new String[]{"COLUMN 1", "COLUMN 2"},  
    Cláusula WHERE "CONDITION = ?",  
    Parametros do placeholder conditionParameters,  
    Group By null,  
    Having null,  
    Order by null,  
    Limit null);
```

SQLiteOpenHelper

Obtenção de Dados (Pesquisa) -> Cursor

O cursor permite aceder aos dados produzidos pelas queries. Incluem métodos para aceder aos dados:

- **Métodos de posicionamento**

`isFirst()`, `isLast()`, `isBeforeFirst()`, `isAfterLast()`

- **Métodos de navegação**

`moveToFirst()`, `moveToLast()`, `moveToNext()`, `moveToPrevious()`, `move(n)`

- **Métodos de extração**

`getInt()`, `getString()`, `getFloat()`, `getBlob()`, `getDate()`, etc.

- **Métodos de análise da estrutura**

`getColumnName()`, `getColumnNames()`, `getColumnIndex()`, `getColumnCount()`,
`getCount()`, `getPosition()`

SQLiteOpenHelper

Obtenção de Dados (Pesquisa)

Verifica se o cursor
não está vazio

```
public Person getPerson(SQLiteDatabase db, String personName){  
    Cursor cursor = db.query("PERSON",  
        new String[]{"_id", "name", "age"},  
        "name = ?",  
        new String[]{personName},  
        null, null, null);  
  
    if(cursor != null && cursor.moveToFirst()){  
        int id = cursor.getInt(0);  
        String name = cursor.getString(1);  
        String age = cursor.getInt(2);  
        return new Person(id, name, age);  
    }  
    cursor.close(); ← Fechar sempre o  
    return null;  
}
```

SQLiteOpenHelper

Obtenção de Dados (Pesquisa)

```
public ArrayList<Person> getAllPersons(SQLiteDatabase db){  
    ArrayList<Person> persons = new ArrayList<>();  
    Cursor cursor = db.query("PERSON",  
        new String[]{"_id", "name", "age"},  
        null, null, null, null, null); ←  
  
    if(cursor != null && cursor.moveToFirst()) {  
        do {  
            int id = cursor.getInt(0);  
            String name = cursor.getString(1);  
            String age = cursor.getInt(2);  
            persons.add(new Person(id, name, age));  
        } while (cursor.moveToNext());  
    }  
    cursor.close();  
    return persons;  
}
```

Percorre o cursor enquanto existirem resultados

Não temos condição porque queremos obter todos os registos

Ligaçāo à Base de Dados

Estrutura [ex. MainActivity.java]

```
My DBHelper dbHelper = new DBHelper(context);  
SQLiteDatabase db = dbHelper.getWritableDatabase();  
...  
db.close();
```

É importante fechar
sempre as ligações à base
de dados

utilizamos o SQLiteOpenHelper
para ligar à Base de Dados

Enviamos sempre o
context como argumento

Liga à base de dados
invocando o método
onCreate() se ela ainda não
tiver sido criada.

Base de dados em modo
de escrita.

Para apenas modo de
leitura utilizar
getReadableDatabase()

SQLiteOpenHelper

onUpgrade()

Este método têm 3 parâmetros: a base de dados, a versão atual da base de dados que o utilizador possui, a nova versão existente da base de dados.

```
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { }
```

Com o número de versões podemos controlar que alterações devem ser realizadas à base de dados.

SQLiteOpenHelper

onUpgrade()

```
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    if(oldVersion = 1){  
        //TODO: Implementar código caso a base de dados anterior esteja na versão 1  
    }  
    if(oldVersion < 3){  
        //TODO: Código a correr caso a versão da base de dados seja a 1 ou a 2  
    }  
}
```

SQLiteOpenHelper

onDowngrade()

Tal como o `onUpgrade()` este método têm 3 parâmetros: a base de dados, a versão atual da base de dados que o utilizador possui, a nova versão existente da base de dados.

```
@Override  
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    super.onDowngrade(db, oldVersion, newVersion);  
}
```

Transações

```
db.beginTransaction();
try {
    ...
    db.setTransactionSuccessful();
} catch (SQLiteException sqLiteException) {
    Log.e("SQL", sqLiteException.getMessage());
} finally {
    db.endTransaction();
}
```

Transações

A transação é definida entre os métodos `beginTransaction()` e `endTransaction()`

- É necessário invocar o método `setTransactionSuccessful()` para submeter qualquer alteração à base de dados;
- A ausência do método `setTransactionSuccessful()` provoca um rollback automático para o estado da base de dados antes do início da transação.

Notas Finais

Localização da Base de Dados (Device File Explorer)

/data/data/[myAppPackage]/databases/[myDatabaseFile]

Name	Permiss...	Date	Size
com.google.android.printse	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.project	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.provider	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.sdksetup	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.setupw	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.soundp	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.syncad	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.tag	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.tts	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.videos	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.webview	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.wifi.res	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.android.youtub	drwxrwx--x	2020-11-17 21:58	4 KB
com.google.mainline.telem	drwxrwx--x	2020-11-17 21:58	4 KB
org.chromium.webview_she	drwxrwx--x	2020-11-17 21:58	4 KB
pt.ipp.estg.t_database	drwxrwx--x	2020-11-17 21:58	4 KB
cache	drwxrws--x	2020-11-15 19:33	4 KB
code_cache	drwxrws--x	2020-11-17 08:02	4 KB
databases	drwxrwx--x	2020-11-15 19:51	4 KB
CountryDetails.db	-rw-rw----	2020-11-17 10:49	24 KB
CountryDetails.db-jor	-rw-rw----	2020-11-17 10:49	0 B
shared_prefs	drwxrwx--x	2020-11-16 23:07	4 KB

Notas Finais

Database Inspector

Database Inspector

Pixel 2 API 30 > pt.ipp.estg.tdatabase

Databases

continent

Refresh table Live updates

	_id	name
CountryDetails.db	1	Europe
continent	2	America
country_info	3	Africa
pt.ipp.estg.t_database/CountyData	4	Asia
android_metadata	5	Oceania
continent		
country_info		

Notas Finais

Acesso

- Os conteúdos da pasta `data/data/myAppPackage` são sempre acessíveis através da aplicação identificada pelo `myAppPackage`;
- Do exterior (outras aplicações, ADB, ...) não é possível aceder a esta pasta, com a exceção de:
 - Emuladores
 - Dispositivos com Acesso Root

Leitura Adicional

SQLite on Android:

<https://developer.android.com/training/data-storage/sqlite>

SQLite API:

<https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase>

AndroidX e SQLite:

<https://developer.android.com/jetpack/androidx/releases/sqlite>

Programação Para Dispositivos Móveis I

DATABASES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

ROOM

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Índice

- Android Architecture Components;
- Room;
- Leitura Adicional.

Android Architecture Components

Uma coleção de bibliotecas recomendadas que ajudam no desenvolvimento de aplicações robustas

Principais componentes:

- LiveData
- Handling Lifecycles
- ViewModel
- Room

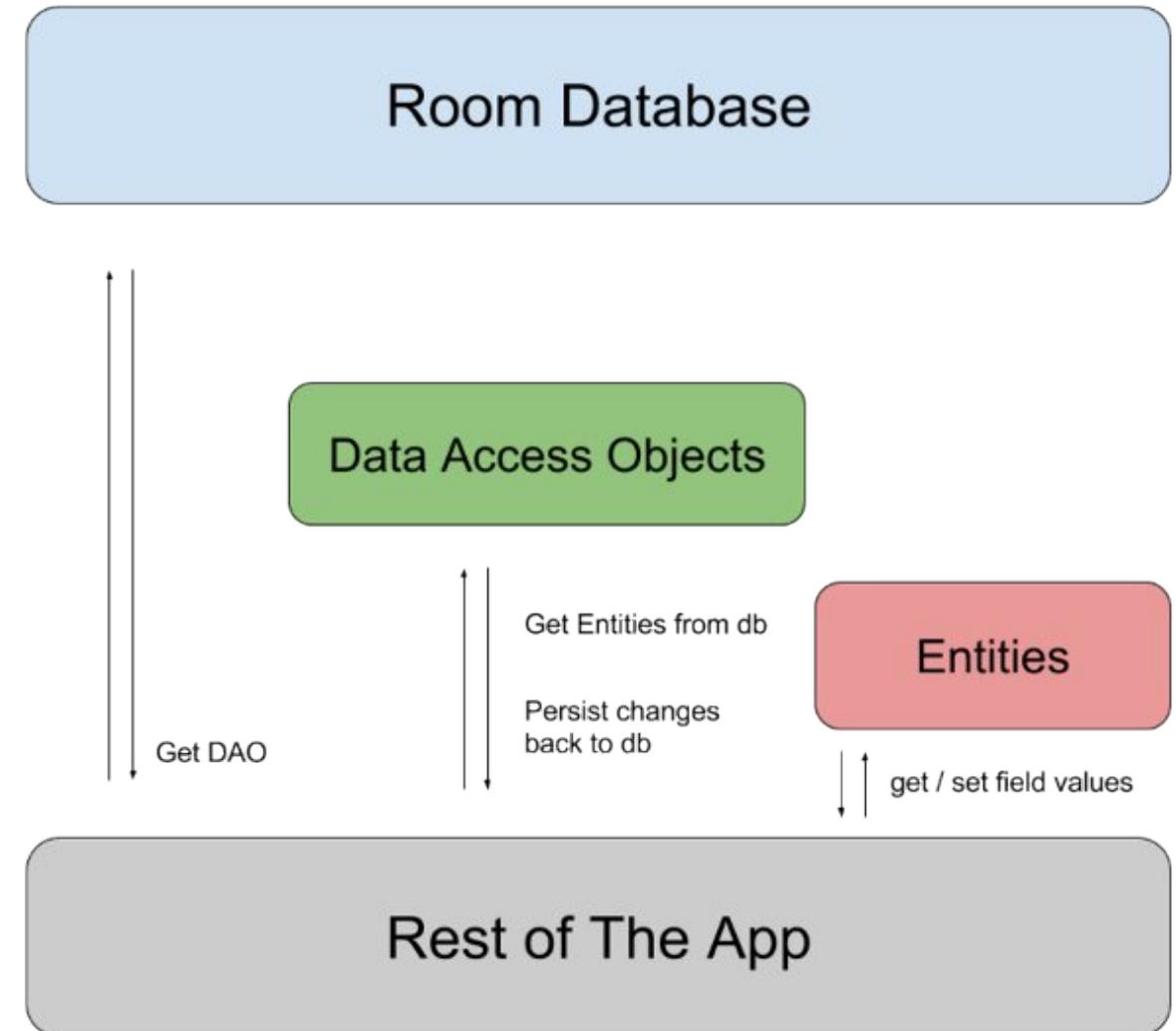
!! Vamos nos focar na biblioteca Room !!



Room

Faz parte dos novos componentes de arquitetura do Android;

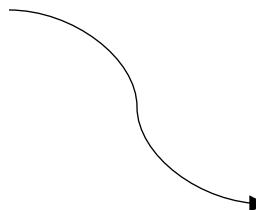
- Suporte para gerir a persistência de dados com recurso a entidades e objetos usando o componente Room;
- Evita desenvolvimentos de classes para conexão à base de dados;
- Converte tabelas em SQLite para objetos JAVA;
- Testa comandos SQLite em tempo de compilação;



Room

Instalação [build.gradle (project)]

Em principio já estão definidos estes repositórios, mas confirmar ajuda sempre



```
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}
```

Room

Instalação [build.gradle (module)]

```
dependencies {  
    ...  
    def room_version = "2.6.1"  
  
    implementation "androidx.room:room-runtime:$room_version"  
    annotationProcessor "androidx.room:room-compiler:$room_version"  
    ...  
}
```

Room

Componentes

Existem 3 componentes principais em ROOM:

- Base de dados – contém a base de dados e serve de ponto de acesso aos dados persistidos;
- Entidades – representa uma tabela na base de dados;
- Data Access Objects (DAO) – contêm os métodos para acesso à base de dados.

Room

Entidades

```
@Entity(tableName = "user")
public class Person {

    @PrimaryKey
    public int _id;

    @ColumnInfo(name = "person_name")
    public String name;

    public int age;
}
```

Se pretendermos nomes diferentes

Chave primária composta

```
@Entity(primaryKeys = {"name", "age"})
public class Person {

    public String name;

    public int age;
}
```

Room

Relação entre entidades 1:1

```
@Entity  
public class Person {  
  
    @PrimaryKey  
    public int _id;  
  
    public String name;  
  
    public int age;  
}
```

```
@Entity  
public class Playlist{  
    @PrimaryKey  
    public int _id;  
  
    public int userOwnerId;  
}
```

Room

Relação entre entidades

```
@Entity(foreignKeys = @ForeignKey(entity = Person.class,
                                    parentColumns = "_id",
                                    childColumns = "personID"))
public class Book {

    @PrimaryKey
    public int _bookID;

    public String title;

    public int personID;
}
```

```
@Entity
public class Person {

    @PrimaryKey
    public int _id;

    public String name;

    public int age;
}
```

Room

Relação entre entidades 1:N

```
@Entity
public class Person {
    @PrimaryKey
    public int _id;
    public String name;
    public int age;
}

@Entity
public class Playlist{
    @PrimaryKey
    public int _id;
    public String name;
    public int userOwnerId;
}

public class UserWithPlaylist{
    @Embedded public Person user;
    @Relation(
        parentColumn = "_id"
        entityColumn = "userOwnerId"
    )
    public List<Playlist> playlists;
}
```

Room

Relação entre entidades N:N

```
@Entity  
public class Playlist{  
    @PrimaryKey  
    public int _id;  
  
    public String name;  
}
```

```
@Entity  
public class Song{  
    @PrimaryKey  
    public int _id;  
  
    public String name;  
    public String artist;  
}
```

```
@Entity(primaryKeys = {"playlistId", "songId"})  
public class PlaylistSongCrossRef{  
    private int playlistId;  
  
    private int songId;  
}
```

Room

Objetos aninhados

```
class Address{  
    public String street;  
    public String state;  
    public String city;  
  
    @ColumnInfo(name = "postal_code")  
    public int postalCode;  
}  
  
@Entity  
public class Person {  
  
    @PrimaryKey  
    public int _id;  
  
    public String name;  
  
    public int age;  
  
    @Embedded  
    public Address address;  
}
```

Room

DAO

DAO (Data Access Objects) são usados para aceder a dados persistidos na base de dados como objetos JAVA.

- Acedendo à base de dados através de DAO permite separar diferentes componentes da arquitetura da base de dados;
- DAO pode ser uma interface ou uma classe abstrata;
- A implementação das interface é feita pela biblioteca DAO;

Room

DAO -> Possíveis interfaces

Inserção de dados

@Dao

```
public interface PersonDAO {  
  
    @Insert(onConflict = OnConflictStrategy.REPLACE)  
    void insertPersons(Person ... persons);
```

Remoção de dados

@Insert

```
void insertPerson(Person person);
```

@Update

```
void updatePerson(Person person);
```

@Delete

```
void deletePerson(Person person);
```

```
}
```

Atualização de dados

Room

DAO -> Queries

@Dao

```
public interface PersonDAO {
```

Sem argumentos

```
    @Query("SELECT * FROM Person")  
    List<Person> getAll();
```

Com argumentos

```
    @Query("SELECT * FROM Person WHERE age BETWEEN :minAge AND :maxAge")  
    List<Person> loadPersonsBetweenAges(int minAge, int maxAge);
```

```
    @Query("SELECT * FROM Person WHERE name LIKE :search")  
    List<Person> findPersonsWithName(String search);
```

```
}
```

Room

DAO -> Queries

@Dao

```
public interface PersonDAO {  
  
    @Query("SELECT * FROM Book"  
        + " INNER JOIN Loan ON loan.book_id = book._id"  
        + " INNER JOIN Person ON person._id = loan.person_id"  
        + " WHERE Person.name LIKE :name")  
    List<Book> findBooksBorrowedByPerson(String name);
```

Query com múltiplas tabelas

Query com múltiplas tabelas para objetos
personalizados

```
@Query("SELECT Person.name, Pet.name AS petName"  
        + "FROM Person, Pet"  
        + "WHERE person._id = pet.person_id")  
public LiveData<List<PersonPet>> loadPersonsAndPets();
```

```
static class PersonPet {  
    public String personName;  
    public String petName;  
}  
}
```

Room DAO

```
@Dao
public interface PersonDAO {
    @Query("SELECT * FROM person")
    List<Person> getAll();

    @Query("SELECT * FROM Person WHERE name LIKE :search")
    List<Person> findPersonsWithName(String search);

    @Query("SELECT * FROM person WHERE name LIKE :name LIMIT 1")
    Person findByName(String name);

    @Insert
    void insertAll(Person ... persons);

    @Delete
    void delete(Person person);
}
```

Room

Criação de Base de Dados

Uma classe abstrata não pode ser instanciada

```
@Database(entities = {Person.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {

    public abstract PersonDAO personDAO();
}
```

Podemos obter uma instancia da base de dados criada através
deste código (ex. MainActivity.java)

```
AppDatabase db = Room.databaseBuilder(getApplicationContext(),
    AppDatabase.class, "database-name").build();
```

Room

Utilização

O presente código dá erro de execução!!
Operações sobre a base de dados não são permitidas na thread da UI.
É necessário recorrer a threads em background para operações sobre a interface DAO.
No futuro iremos abordar possíveis soluções.

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        AppDatabase db = Room.databaseBuilder(getApplicationContext(),  
            AppDatabase.class,  
            "database-name")  
            .build();  
  
        Person person = new Person();  
        person.name = "John";  
        person.age = 35;           Aceder à interface DAO e fazer a chamada do método  
        db.personDAO().insertPerson(person);  
    }  
}
```

Room

Utilização

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        AppDatabase db = Room.databaseBuilder(getApplicationContext(),  
            AppDatabase.class,  
            "database-name")  
            .allowMainThreadQueries() ←  
            .build();  
  
        Person person = new Person();  
        person.name = "John";  
        person.age = 35;  
  
        db.personDAO().insertPerson(person);  
    }  
}
```

Para efeitos de teste podemos
adicionar esta propriedade

Room

Migração de Base de Dados

À medida que novos parâmetros são adicionadas à base de dados, existe a necessidade de implementar métodos corretivos para a aplicação não apagar os dados ao fazer o update à base de dados.

- Necessário fazer export dos Schemas atualizando o ficheiro build.gradle (module)

```
    android {  
        ...  
  
        defaultConfig {  
            ...  
  
            javaCompileOptions{  
                annotationProcessorOptions{  
                    arguments = ["room.schemaLocation":  
                                "$projectDir/schemas".toString()]  
                }  
            }  
            ...  
        }  
    }  
}
```

Room

Migração de Base de Dados

Adicionar os métodos de migração à classe da base de dados

- Cada versão da versão da base de dados deve originar um método de migração correspondente

```
@Database(entities = {Person.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {

    public abstract PersonDAO personDAO();

    private static AppDatabase INSTANCE;
    private static final Object sLock = new Object();

    static final Migration MIGRATION_1_2 = new Migration(1,2){
        @Override
        public void migrate(@NotNull SupportSQLiteDatabase database) {
            //TODO: Implementar código para alterar as tabelas da versão 1 para a 2
        }
    };
}

...
```

Room

Migração de Base de Dados (continuação)

...

```
static final AppDatabase getInstance(Context context){
    synchronized (sLock){
        if(INSTANCE == null){
            INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
                AppDatabase.class, "sample.db")
                .addMigrations(MIGRATION_1_2)
                .build();
        }
        return INSTANCE;
    }
}
```

Leitura Adicional

Room Tutorial

<https://developer.android.com/training/data-storage/room>

Room Dependencies

<https://developer.android.com/jetpack/androidx/releases/room>

AndroidX Room

<https://developer.android.com/reference/androidx/room/package-summary>

Programação Para Dispositivos Móveis I

ROOM

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

THREAD, EXECUTOR SERVICE & WORKMANAGER

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Índice

- Background Processing;
- Thread;
- ExecutorService;
- WorkManager;
- Leitura Adicional.

Prefácio

Recordamos o conteúdo apresentado nos slides de Databases. Uma aplicação que reproduzisse os passos da implementação apresentada teria o seguinte comportamento:

1. No momento da abertura da aplicação, iria procurar por um ficheiro de base de dados;
2. Caso esse ficheiro não existisse teria de criar uma base de dados nova;
3. Depois teria de correr todas as instruções de SQL para criar as tabelas dentro da base de dados, e alguns dados iniciais;
4. Por fim teria de executar algumas queries para retirar dados da base de dados e apresenta-los ao utilizador.

Prefácio

Para uma base de dados pequena, este processo não demoraria muito tempo e seria imperceptível para o utilizador. Contudo, com o constante crescimento de uma base de dados ao longo do tempo, não iria demorar muito até a aplicação tornar-se lenta, até ao ponto de ser inutilizável.

Não há muito que se possa fazer para aumentar as velocidades de escrita e leitura numa base de dados, mas **podemos impedir** que estes processos tornem a nossa aplicação mais lenta.

Threads

Desde a versão Lollipop (API 21), existem três tipos de threads que devemos considerar:

1. Main Thread (ou User Interface (UI) Thread)

Esta é a thread principal de Android. Está à escuta de intents, recebe inputs do ecrã, e procede à chamada de métodos dentro das activities.

2. Render Thread

Normalmente não existe interação com esta thread, mas é responsável por “ler” uma lista de requisitos para atualizações do ecrã, e depois instruir o hardware gráfico do dispositivo para “repintar” o ecrã e tornar a aplicação bonita.

3. Outras Threads que sejam criadas

Threads

Sem algum cuidado e controlo, a aplicação irá executar a maioria das suas tarefas na UI Thread (Main Thread).

Ao colocar todo o código associado com a base de dados (por exemplo) no método `onCreate()`, a UI Thread vai estar ocupada a comunicar com a base de dados, em vez de rapidamente procurar por eventos a partir do ecrã ou outras aplicações. Se algum conjunto de instruções relacionada com a base de dados demorar muito tempo, o utilizador irá sentir que está a ser ignorado ou irá questionar-se se a aplicação terá falhado.

O “truque” é remover estas instruções “pesadas” da UI Thread e executa-las numa Thread personalizada em background.

Background Processing

Um processo contém um ambiente de execução com **um conjunto de recursos associado** (ex. espaço de memória), necessários para a execução de instruções.

Em sistemas multiprocessamento (como é o caso do Android), a execução destes processos é feita de forma **concorrente/paralela**, dando ao utilizador a sensação de que diversas aplicações estão a ser executadas em simultâneo.

No entanto, a criação de múltiplos processos para a execução de código em paralelo facilmente se torna uma operação pesada, devido aos recursos alocados à sua criação.

Background Processing

Um processo pode ter múltiplas Threads que:

- Executam de forma concorrente/paralela;
- Partilham alguns dos recursos do processo (por ex., espaço de memória);
- São muito mais leves de criar e lançar.

Algumas operações não devem ser executadas na UI Thread, visto existir o risco de bloquearem a interação com o utilizador (**congelamento da aplicação**):

- Tarefas que sejam demoradas ou com um tempo de execução imprevisível;
- Operações de I/O (por ex., sobre ficheiros, comunicações de rede), que não são permitidas na UI Thread a partir da versão Android 2.3.3.

Mais em: <http://developer.android.com/guide/components/processes-and-threads.html> e <http://docs.oracle.com/javase/tutorial/essential/concurrency/>

Background Processing

Threads e Executor Services

Em Android, existem diversas formas de lançarmos novas linhas de execução paralelas (Threads):

- Classe Thread (tradicionalmente utilizada em Java);
- Classe ExecutorService (abstração que faz parte da API Android).
- Na Thread, o código que executa uma nova linha de instrução encontra-se no método `run()`
- Todo o resto, **a não ser que seja invocado a partir deste método**, executa na UI Thread.

Thread

Utilização

A definição de uma Thread é simples, sendo a sua complexidade adicional derivada da **sincronização com a UI Thread**.

```
public class DownloadThread extends Thread {  
    public DownloadThread(Context context, URL url) { ... }  
    public void run() { ... } // As instruções são executadas aqui  
}
```

Extende Thread

Método construtor

As instruções são executadas aqui

Lançamento da DownloadThread a partir da UI Thread (ex. MainActivity.java)

```
DownloadThread thread = new DownloadThread(url);  
thread.start();
```

Thread

Utilização

- Cancelamento da Thread

O mecanismo tem que ser implementado (não existe na classe Thread)

- Obtenção do estado da Thread

```
Thread.State sts = thread.getState();
```

Possíveis estados: NEW, RUNNABLE, BLOCKED, WAITING, TIMED_WAITING, TERMINATED;

- Esperar pela sua execução e obter o resultado

```
thread.join();
```

Thread

Utilização

- Execução de uma operação na UI Thread a partir de outra Thread

```
Handler mainHandler = new Handler(context.getMainLooper());
```

A UI Thread é obtida através do Contexto da aplicação

```
Runnable postRunnable = new Runnable() {  
    public void run() { ... }  
}
```

O método run() irá ser executado na UI Thread

```
mainHandler.post(postRunnable);
```

A instância de Runnable é enviada para a UI Thread

ExecutorServices

Os ExecutorServices permitem a criação de uma *pool* de *worker threads* (thread pool) onde diferentes tarefas podem ser executadas de forma concorrente.

Lançam uma nova Thread e possuem métodos específicos para a execução de instruções na UI Thread.

Executor Services

Implementação

```
ExecutorService executorService = Executors.newSingleThreadExecutor(AVAILABLE_CORES);
```

Cria uma única thread

```
private static final int AVAILABLE_CORES = Runtime.getRuntime().availableProcessors();
```

```
ExecutorService executorService = Executors.newFixedThreadPool(AVAILABLE_CORES);
```

Cria uma Thread Pool com n threads

Podemos utilizar o número de cores disponíveis para definir o total de threads.

1 core = 1 thread

Executor Services

Implementação

```
private void runBackgroundRunnable() {  
    executorService.execute(new Runnable() {  
        @Override  
        public void run() {  
            ( ... )  
  
            runOnUiThread(new Runnable() {  
                ( ... )  
            });  
        }  
    });  
}
```

Todo o código que deverá executado pela thread está no método run()

É utilizado o método runOnUiThread para poder comunicar com a Main Thread.
Utilizado para, por exemplo, atualizar Views.

Executor Services

Implementação

```
private void runBackgroundRunnable() {  
    executorService.execute(new Runnable() {  
        @Override  
        public void run() {  
            try {  
                URL imageURL = new URL(String.valueOf(inputExternalURL.getText()));  
                HttpURLConnection httpURLConnection = (HttpURLConnection) imageURL.openConnection();  
                httpURLConnection.connect();  
                resultPicture = BitmapFactory.decodeStream(httpURLConnection.getInputStream());  
                Log.d("ImageURL :: ", imageURL.toString());  
            } catch (IOException | RuntimeException exception) {  
                Log.e("BitmapFactory Exception :: ", exception.getMessage());  
                executorService.shutdown();  
            }  
            runOnUiThread(new Runnable() {  
                @Override  
                public void run() {  
                    externalPictureView.setImageBitmap(resultPicture);  
                }  
            });  
        }  
    });  
}
```

A thread irá realizar o download de uma imagem a partir de um URL fornecido pelo utilizador

Conversão do resultado para um objeto Bitmap (imagem)

Atualizar a ImageView com a imagem que foi obtida a partir do URL.

Executor Services

Implementação

```
@Override  
protected void onDestroy() {  
    super.onDestroy();  
  
    executorService.shutdown();  
}
```

Não esquecer de terminar a execução sempre que existirem interrupções na Thread.

Executor Services (Future)

[BitmapDownloadTask.java]

```
public class BitmapDownloadTask implements Callable<Bitmap> {  
    private static final String THREAD_TAG = "CURRENT THREAD";  
    private final String userUrl;  
    private Bitmap resultPicture;  
  
    public BitmapDownloadTask(String userUrl) {  
        this.userUrl = userUrl;  
    }  
  
    @Override  
    public Bitmap call() {  
        Log.d(THREAD_TAG, Thread.currentThread().getName());  
        try {  
            URL imageURL = new URL(userUrl);  
            HttpURLConnection httpURLConnection = (HttpURLConnection) imageURL.openConnection();  
            httpURLConnection.connect();  
            resultPicture = BitmapFactory.decodeStream(httpURLConnection.getInputStream());  
        } catch (IOException exception) {  
            Log.e("BitmapDownloadTask Exception :: ", exception.getMessage());  
            executorService.shutdown();  
        }  
  
        return resultPicture;  
    }  
}
```

Método onde deverá ser incluído o código a ser executado numa nova Thread.

Informação sobre a Thread selecionada para correr este método

Em vez de manipular a Main Thread, vamos apenas retornar o resultado

Executor Services (Future)

[MainActivity.java]

```
Future<Bitmap> resultExternalPicture = null;
```

Criação do objeto Future, que irá conter o resultado do pedido assíncrono.

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    (...)  
  
    resultExternalPicture = executorService.submit(  
        new BitmapDownloadTask(String.valueOf(inputExternalURL.getText())));  
}
```

Envio da "nova tarefa" para ser executada pela Thread Pool

O retorno da função Call() será armazenado neste objeto.

Executor Services (Future)

[MainActivity.java]

```
if(resultExternalPicture != null && resultExternalPicture.isDone()){\n    try {\n        Bitmap resultExternalPicture = this.resultExternalPicture.get();\n        externalPictureView.setImageBitmap(resultExternalPicture);\n    } catch (ExecutionException | InterruptedException exception) {\n        Log.e(BITMAP_TAG, exception.getMessage());\n    }\n}
```

Verificar se o pedido executado
pela Thread já foi concluido

Obtemos o conteúdo do Future
Object (que estaria a guardar o
resultado do pedido assíncrono).

Executor Services (CompletableFuture)

[MainActivity.java]

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    ( ... )  
  
    CompletableFuture<Void> completableFuture = CompletableFuture.supplyAsync(  
        () -> resultExternalPicture = executorService.submit(new BitmapDownloadTask(DEFAULT_URL)))  
        .thenAccept(resultPicture -> {  
            try {  
                externalPictureView.setImageBitmap(resultPicture.get());  
            } catch (ExecutionException | InterruptedException exception) {  
                throw new RuntimeException(exception);  
            }  
        });  
}  
}
```

Envio da "nova tarefa" para ser executada pela Thread Pool

Depois da tarefa estar concluída, podemos utilizar o resultado da mesma para interagir com a Main Thread, ou para encadear novos processos concorrentes.

WorkManager

Componente integrante do Android Jetpack que permite a **execução fiável de tarefas** em background. (Também permite a execução de tarefas fora da UI Thread)

- Permite configurar condições de execução:

- Apenas quando o dispositivo estiver livre;
- Apenas quando a carregar;
- (...)

- Permite encadear tarefas.

WorkManager

build.gradle (module)

```
dependencies {  
    implementation "androidx.work:work-runtime:2.10.0"  
    ...  
}
```

WorkManager

MyWorker.java

```
public class MyWorker extends Worker {

    public MyWorker(@NotNull Context context, @NotNull WorkerParameters workerParams) {
        super(context, workerParams);
    }

    @NotNull
    @Override
    public Result doWork() {

        //Obter input
        getInputData().getString("KEY");

        //TODO: realizar tarefa

        //Criar output

        Data outputData = new Data.Builder()
            .putString("KEY", "VALUE")
            .build();

        return Result.success();
    }
}
```

WorkManager

Execução [ex. MainActivity.java]

Também pode ser utilizado um PeriodicWorkRequest



```
OneTimeWorkRequest workRequest = new OneTimeWorkRequest.Builder(MyWorker.class)
                                .build();

WorkManager.getInstance(this).enqueue(workRequest);
```

WorkManager

Execução com inputs [ex. MainActivivity.java]

```
Data.Builder inputs = new Data.Builder()
    .putString("KEY", "VALUE");

OneTimeWorkRequest workRequest = new OneTimeWorkRequest.Builder(MyWorker.class)
    .setInputData(inputs.build())
    .build();

WorkManager.getInstance(this).enqueue(workRequest);
```

Adicionamos os valores
de input

WorkManager

Definição de condições de execução [ex. MainActivitiy.java]

```
Constraints constraints = new Constraints.Builder()  
    .setRequiresDeviceIdle(true)  
    .setRequiresBatteryNotLow(true)  
    .build();
```

Só é executado se o dispositivo estiver em Idle (inativo)

```
OneTimeWorkRequest compressionWorker = new OneTimeWorkRequest.Builder(MyWorker.class)  
    .setConstraints(constraints)  
    .build();
```

Só é executado se não tivermos níveis baixos de bateria

```
WorkManager.getInstance(this).enqueue(compressionWorker);
```

Adicionamos as condições de execução

WorkManager

Consultar o estado da classe Worker [ex. MainActivity.java]

```
WorkManager.getInstance(this).getWorkInfoByIdLiveData(workRequest.getId())
    .observe(this, new Observer<WorkInfo>() {
        @Override
        public void onChanged(WorkInfo workInfo) {
            if(workInfo != null && workInfo.getState() == WorkInfo.State.SUCCEEDED){
                String output = workInfo.getOutputData().getString("KEY");
            }
        }
    });
});
```

Leitura Adicional

Background Processing:

<https://developer.android.com/guide/background>

Threads:

<https://developer.android.com/guide/background/threading>

ExecutorServices:

<https://developer.android.com/reference/java/util/concurrent/ExecutorService>

WorkManager:

<https://developer.android.com/reference/androidx/work/WorkManager?hl=en>

Programação Para Dispositivos Móveis I

THREAD, EXECUTOR SERVICE & WORKMANAGER

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

CONECTIVIDADE

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Índice

- Conectividade;
- Conectividade em Android;
- Informação da conexão;
- Conexão HTTP;
- Leitura Adicional.

Que tipos de conexão podem existir em Android?

Conectividade

- Funcionalidade básica para dispositivos móveis;
- Permitem a comunicação com o “exterior”;
- Não implica apenas vias de comunicação com a Internet.

Conectividade

- Internet:

- 5G/4G/3G/2G
- Wi-Fi

- Outras:

- Bluetooth
- NFC
- IR
- ...

Conectividade em Android

As operações de comunicação em Android devem ser feitas com cuidados adicionais:

- Necessário **explicita permissão** do utilizador;
- Devem ser feitas fora da UI thread;
- Devem ter em consideração o contexto da aplicação;
- Devem estar otimizadas para conservação de recursos (ex: bateria).

Existem vários componentes na plataforma Android para operações em background que devem ser utilizadas.

Conectividade em Android

Permissões [AndroidManifest.xml]

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Permite obter o estado das redes de comunicação (Wi-Fi e 4G)

```
<uses-permission android:name="android.permission.INTERNET" />
```

Permite utilizar a comunicação à internet (seja por Wi-Fi ou por 4G)

Conectividade em Android

ConnectivityManager

Connectivity Manager é um serviço responsável por obter o estado de conectividade à Internet.

```
ConnectivityManager connectivityManager =  
    (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
```

Tem a capacidade para notificar as aplicações quando **existem alterações** no estado da conectividade através de callbacks.

<https://developer.android.com/reference/android/net/ConnectivityManager>

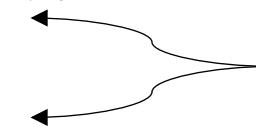
Conectividade em Android

ConnectivityManager

```
ConnectivityManager connectivityManager =
        (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
Boolean isWifiConnected = false;
Boolean isMobileConnected = false;

for(Network network : connectivityManager.getAllNetworks()){
    NetworkInfo networkInfo = connectivityManager.getNetworkInfo(network);
    if(networkInfo.getType() == ConnectivityManager.TYPE_WIFI){
        isWifiConnected = networkInfo.isConnected();
    }
    if(networkInfo.getType() == ConnectivityManager.TYPE_MOBILE){
        isMobileConnected = networkInfo.isConnected();
    }
}

Log.d(DEBBUG_TAG, "Wi-Fi Connected: " + isWifiConnected);
Log.d(DEBBUG_TAG, "Mobile Connected: " + isMobileConnected);
```



Conexão que
estamos a verificar

Conectividade em Android

ConnectivityManager -> NetworkInfo

getState() - retorna o estado da conexão para o tipo especificado

- NetworkInfo.State.CONNECTED
- NetworkInfo.State.DISCONNECTED
- NetworkInfo.State.CONNECTING
- NetworkInfo.State.DISCONNECTING
- NetworkInfo.State.UNKNOWN

isConnected() - indica se existe conectividade e se é possível haver transferência de dados;

isConnectedOrConnecting() - indica se existe conectividade ou se a mesma está a ser estabelecida;

Conectividade em Android

Conexão HTTP

Criar um objeto URL com o endereço a comunicar

```
URL url = new URL("http://www.mywebsite.com");
```

Abrir conexão HTTP

```
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
```

Definição dos tempos de timeout

```
connection.setConnectTimeout(3000);  
connection.setReadTimeout(3000);
```

Escolher o tipo de método HTTP a utilizar (GET/PUT/POST/DELETE)

```
connection.setRequestMethod("GET");
```

Conectividade em Android

Conexão HTTP

Estabelecer a conexão e obter o tamanho do ficheiro
(se o tamanho for inferior a 0 é porque ocorreu um erro)

```
connection.getContentLength();
```

Obter o código/mensagem da resposta HTTP

```
connection.getResponseCode();
connection.getResponseMessage();
```

Obter resposta HTTP

```
InputStream inputStream = connection.getInputStream();
```

Conectividade em Android

Conexão HTTP

Os pedidos de conexão HTTP devem ser feitos em background. Devemos procurar os componentes adequados para as operações:

- ExecutorService
- DownloadManager
- Volley
- Retrofit

A tentativa de conexão na UI Thread pode resultar na exceção NetworkOnMainThreadException.

Outros Recursos para Comunicação

Podemos consultar outras bibliotecas para outros tipos de comunicação em Android:

- Bluetooth <https://developer.android.com/guide/topics/connectivity/bluetooth>
- NFC <https://developer.android.com/guide/topics/connectivity/nfc>
- USB <https://developer.android.com/guide/topics/connectivity/usb>

Leitura Adicional

- Conectividade Android:

<https://developer.android.com/guide/topics/connectivity>

<https://developer.android.com/training/basics/network-ops/connecting.html>

- Volley para operações de comunicação:

<https://developer.android.com/training/volley>

<https://github.com/google/volley>

Programação Para Dispositivos Móveis I

CONECTIVIDADE

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

RETROFIT

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Índice

- Retrofit;
- Configuração;
- Exemplo Retrofit;
- Leitura Adicional.

Retrofit

É um cliente HTTP que abstrai a complexidade na comunicação de uma aplicação Android com serviços REST.

- A biblioteca expõe a API de um servidor em forma de uma **interface** em Java;
- Permite executar diferentes tipos de pedidos: GET, POST, PUT, DELETE, etc;
- Converte automaticamente a resposta do servidor em objetos Java (**Gson Converter**);
- Permite executar pedidos assíncronos ou síncronos;
- Possui boas capacidades de logging (**OkHttp 3 Http Logging Interceptor**).

Retrofit

Configuração [AndroidManifest.xml]

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">  
    <uses-permission android:name="android.permission.INTERNET" />  
    ...  
</manifest>
```

Permite utilizar a comunicação à internet (seja por Wi-Fi ou por 4G)

Retrofit

Configuração [build.gradle (Module)]

```
dependencies {  
    implementation 'com.squareup.retrofit2:retrofit:2.11.0'  
    implementation 'com.squareup.retrofit2:converter-gson:2.11.0'  
    ...  
}
```

Retrofit

Exemplo

Vamos utilizar a API da Tour-Pedia como exemplo. A mesma pode ser encontrada na ligação <http://tour-pedia.org/api/> e retorna respostas no formato JSON com pontos de interesse de várias cidades europeias.

- A documentação para a utilização da API está presente na hiperligação.

É possível testar algumas queries através do browser ou de uma aplicação externa (ex. Postman).

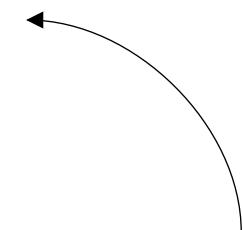
<http://tour-pedia.org/api/getPlaces?location=Amsterdam&category=attraction&name=Science>

<http://tour-pedia.org/api/getPlaces?location=London&name=Restaurant>

Retrofit

Configuração [AndroidManifest.xml]

```
<application  
    android:usesCleartextTraffic="true"  
    ...  
</application>
```



A partir das versões 8.0, como a API não é HTTPS é necessário adicionar a seguinte propriedade

Retrofit

Modelo de Dados

O próximo passo é criar o modelo de dados semelhante ao que é retornado pelo servidor:

- Devemos criar uma classe Java que representa o modelo de dados do servidor

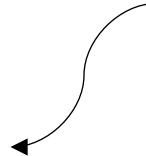
O Retrofit vai recorrer ao conversor Gson para **automaticamente** converter a resposta JSON do servidor num objeto Java. Esta classe Java deve ser uma cópia exata da resposta JSON, isto é, o **nome e tipo de dados** das variáveis da classe devem fazer match com o JSON

Mais informação de como é feita a conversão automática:
<https://github.com/google/gson/blob/master/UserGuide.md>

Retrofit

Modelo de Dados

```
{  
    "id": 216741,  
    "name": "Natural History and Science Museums",  
    "address": "London, , United Kingdom",  
    "category": "attraction",  
    "location": "London",  
    "lat": 51.496795820542,  
    "lng": -0.17435263598631,  
    "details": "http://tour-pedia.org/api/getPlaceDetails?id=216741"  
}
```



Exemplo de um resultado de um pedido GET à API (em formato JSON)

Retrofit

Modelo de Dados [Attraction.java]

```
{  
    "id": 216741,  
    "name": "Natural History and Science Museums",  
    "address": "London, , United Kingdom",  
    "category": "attraction",  
    "location": "London",  
    "lat": 51.496795820542,  
    "lng": -0.17435263598631,  
    "details": "http://tour-pedia.org/api/getPlaceDetails?id=216741"  
}
```

Podemos ter nomes de variáveis diferentes das variáveis do JSON, contudo devemos especificar o seu correspondente

Modelo de Dados para corresponder aos nomes e tipos de dados recebidos

```
public class Attraction {  
  
    private String name;  
  
    private String address;  
  
    private String details;  
  
    @SerializedName("lat")  
    private float latitude;  
  
    @SerializedName("lng")  
    private float longitude;  
  
    ...  
}
```

Retrofit

Interface API [TourDataApi.java (Interface)]

```
@GET("/api/getPlaces")
Call<List<Attraction>> getAttractionList(@Query("location") String location,
                                              @Query("category") String category,
                                              @Query("name") String name);
```

Retorno utilizado para associação
ao modelo de dados

Correspondência aos pares chave-
valor presentes na API



<http://tour-pedia.org/api/getPlaces?location=London&category=attraction&name=Science>

Retrofit

Instância da API [MainActivity.java]

```
private Retrofit getRetrofit(){
    return new Retrofit.Builder()
        .baseUrl("http://tour-pedia.org")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
}

private TourDataApi getApi(){
    return getRetrofit().create(TourDataApi.class);
}
```



Interface da API

Retrofit

Execução de um pedido assíncrono [MainActivity.java]

```
private void getRetrofitData(String city) {  
  
    getApi().getAttractionList(city, "attraction", "Science")  
        .enqueue(new Callback<List<Attraction>>() {  
  
            @Override  
            public void onResponse(Call<List<Attraction>> call, Response<List<Attraction>> response) {  
                List<Attraction> attractionList = response.body();  
                //TODO: Instruções caso o pedido seja bem sucedido  
            }  
  
            @Override  
            public void onFailure(Call<List<Attraction>> call, Throwable t) {  
                //TODO: Instruções caso o pedido falhe  
            }  
        });  
}
```

Retrofit

Questão #1

Os pedidos do Retrofit são executados na UI Thread?

Retrofit

Questão #2

Para que é utilizada a biblioteca **JSON** no exemplo?

Leitura Adicional

Retrofit:

<https://square.github.io/retrofit/>

Programação Para Dispositivos Móveis I

RETROFIT

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

VIEW MODEL
LIVE DATA

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Índice

- ViewModel;
- LiveData;
- LiveData in Android;
- Leitura Adicional.

ViewModel

Um componente da biblioteca de Architecture Components de Android responsável pela preparação de dados para a interface do utilizador:

- Interage bem com os ciclos de vida dos componentes de Android;
- Permite encapsular modelos de dados de forma a poderem ser usados em padrões de software como o Observable;
- Capaz de reagir a alterações do estado dos dados no modelos de dados e acionar callbacks.

ViewModel

Motivação

- O Android gera os ciclos de vida de componentes da UI como activities e fragments. Pode decidir destruir ou recriar cada componente em resposta a determinadas ações do utilizador ou eventos do dispositivo;
- Se o sistema destruir ou recriar um controlador de UI, qualquer tipo de dado guardado no mesmo será perdido. Por exemplo, se a nossa aplicação incluir uma lista de utilizadores numa das suas Activities, quando a Activity é recreada devido a uma alteração de configuração (ex. rodar ecrã), terá de voltar a construir a lista de utilizadores;
- Os dados subjacentes a cada componente podem ser recriados através dos respetivos ciclos de vida do componente onde foram inicializados.

0

COUNT

ViewModel

Exemplo

Recordamos a aplicação “Hello Toast” presente na Ficha Prática nº 1.

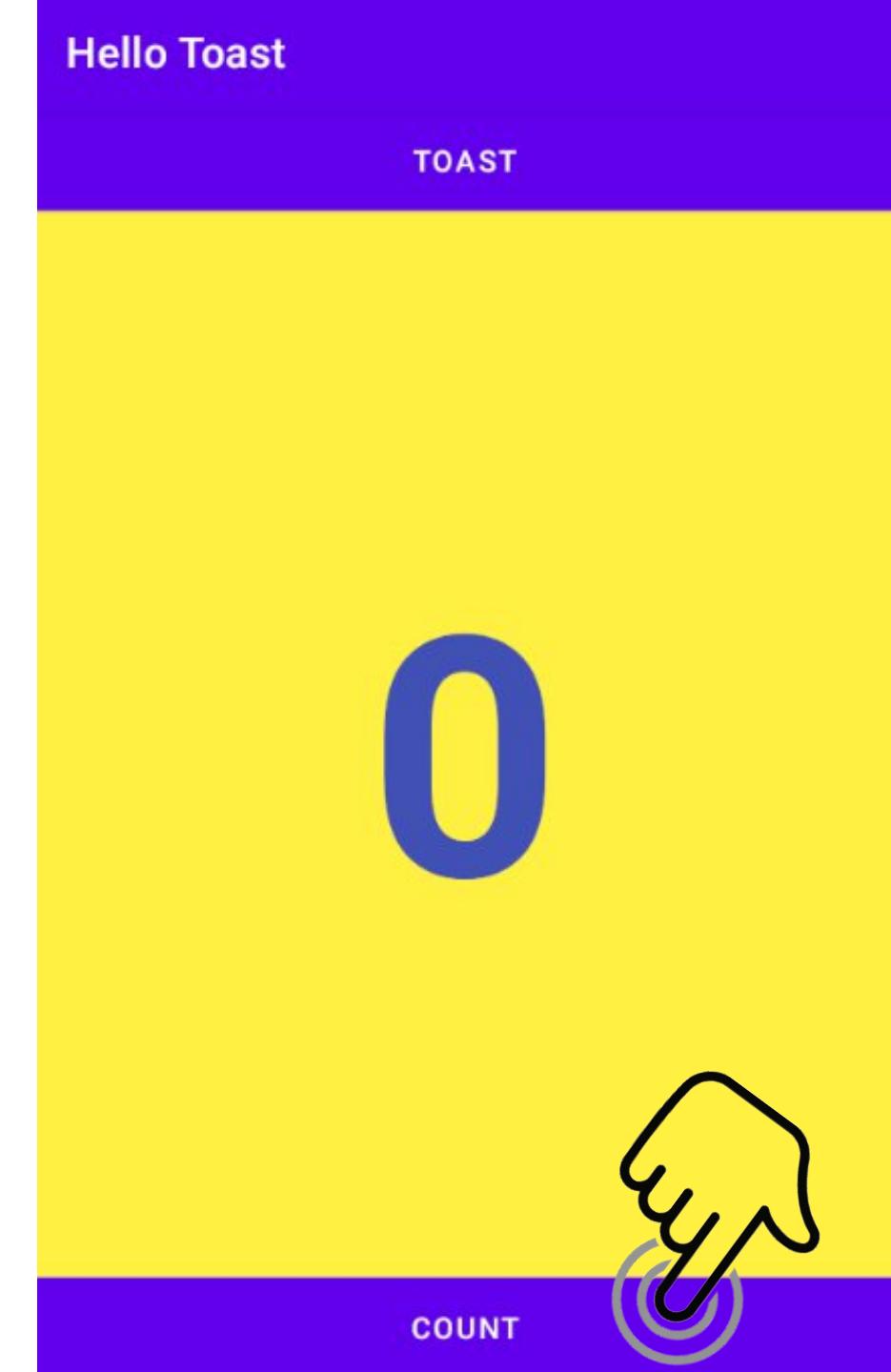
- Uma das funcionalidades da aplicação era a presença de um contador. Cada click que o utilizador der no botão COUNT, o valor do contador incrementa em uma unidade.

ViewModel

Exemplo

Recordamos a aplicação “Hello Toast” presente na Ficha Prática nº 1.

- Uma das funcionalidades da aplicação era a presença de um contador. Cada click que o utilizador der no botão COUNT, o valor do contador incrementa em uma unidade.



1

COUNT

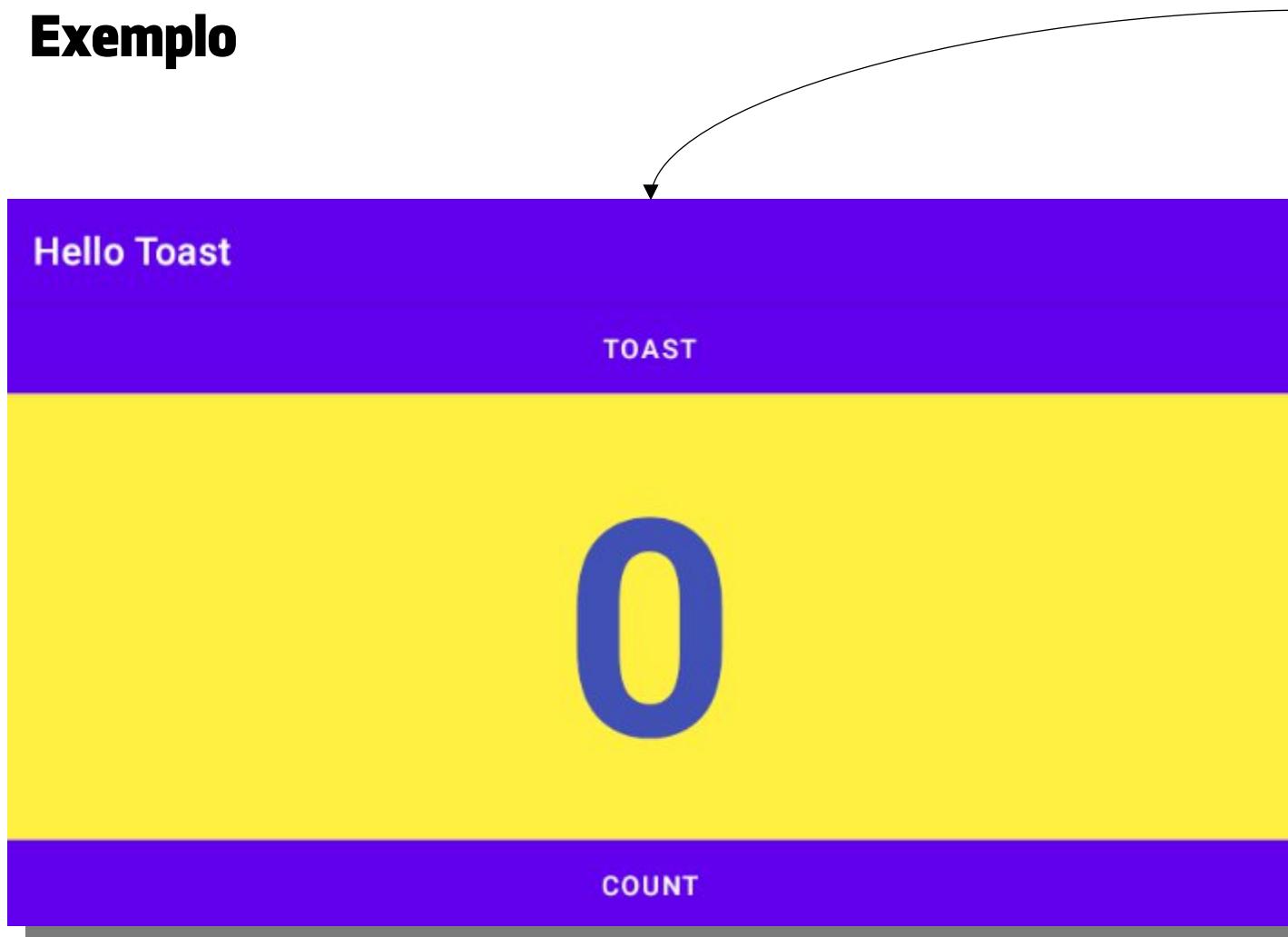
ViewModel

Exemplo

O que acontece se o utilizador decidir rodar o ecrã?

- O comportamento esperado é que a aplicação nos mostre o valor atual do contador e que nos permita incrementá-lo;

ViewModel Exemplo



Hello Toast

TOAST

1

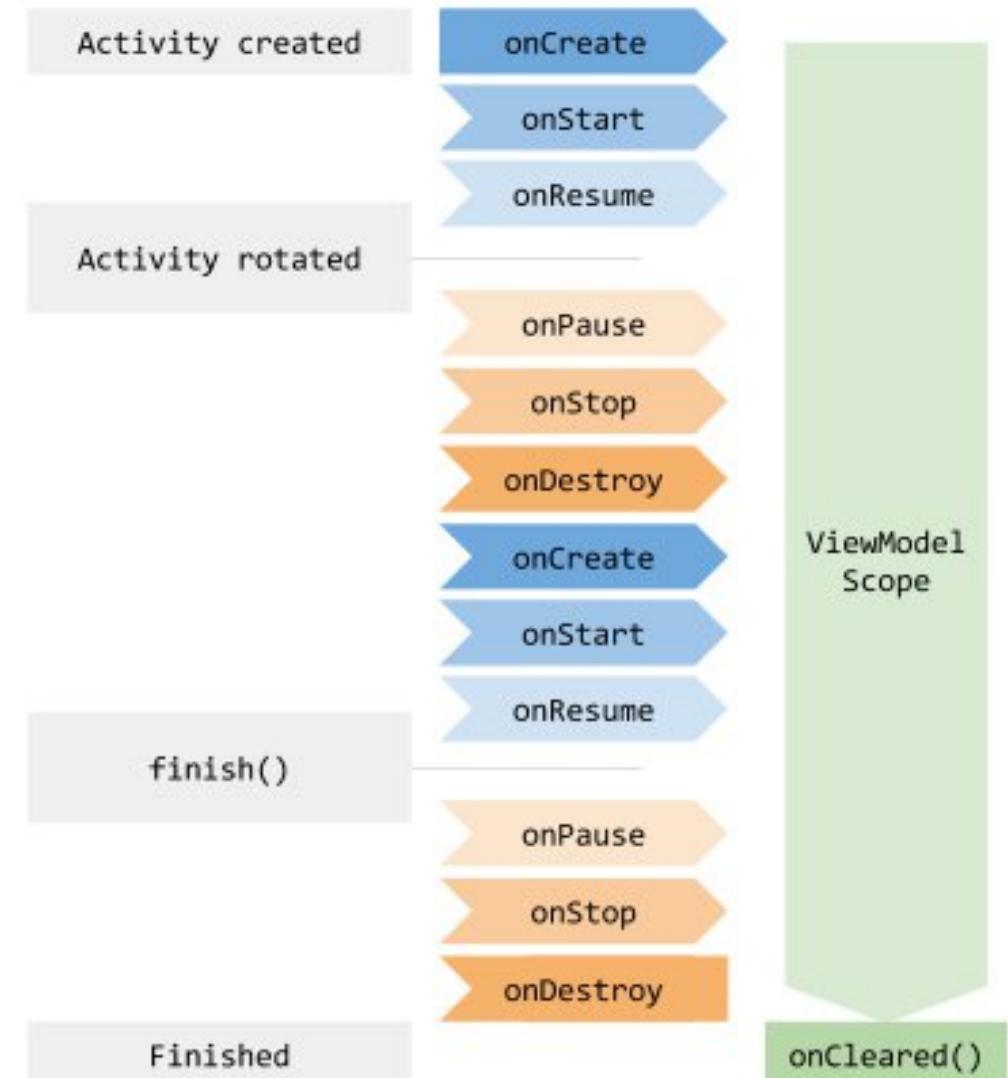
COUNT

ViewModel

Scope

O scope de um ViewModel pode estar associado a uma activity;

Implica que pode ser utilizado por outros componentes que se enquadram no ciclo de vida da activity como por exemplo fragments.



ViewModel

Configuração [build.gradle (Project)]

```
ext{  
    lifecycle_version = "2.8.7"  
}
```

ViewModel

Configuração [build.gradle (Module)]

```
dependencies {  
    // ViewModel  
    implementation "androidx.lifecycle:lifecycle-viewmodel:$lifecycle_version"  
}
```

ViewModel

Criação [CounterViewModel.java]

```
public class CounterViewModel extends ViewModel {  
  
    public int counter = 0;  
  
    public void count(){  
        counter++;  
    }  
  
    @Override  
    protected void onCleared() {  
        super.onCleared();  
    }  
}
```

Extende ViewModel

Procedimento é chamado quando a Activity termina, e é utilizado para libertar recursos

ViewModel

Adicionar o ViewModel

[MainActivity.java]

```
private CounterViewModel counterViewModel;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    (...)  
  
    counterViewModel = new ViewModelProvider(this).get(CounterViewModel.class);  
  
    updateCounter(counterViewModel.counter());  
}  
  
@Override  
public void onClick(View v) {  
    switch (v.getId()){  
        case R.id.btn_count:  
            counterViewModel.count();  
            updateCounter(counterViewModel.counter());  
            break;  
    }  
}  
  
private void updateCounter(int count){  
    txtCount.setText(String.valueOf(count));  
}
```

Definição do ViewModel

counter inicialmente tem o valor de zero

Mudar o valor da TextView para o valor da variável counter

Chamada da função responsável por incrementar o contador

ViewModel

Resultado

utilizador rodou ecrã



Hello Toast

TOAST



LiveData

Em Android LiveData refere-se a um objeto dentro de um observable data holder class (ViewModel).

- Permite a atualização de componentes em tempo real usando o padrão de software Observable, de forma parecida com publish/subscribe;
- É lifecycle-aware, ou seja, respeita o do ciclo de vida de componentes da aplicação

LiveData

Vantagens

Entre as vantagens desta abordagem encontramos:

- Garante que a UI está actualizada com o último estado do modelo de dados;
- Evita memory leaks;
- Evita erros e crashes devido a atividades paradas;
- Não necessita de cuidados manuais com o ciclo dos componentes em Android;

LiveData em Android

Desde que associada ao mesmo contexto, neste caso a Activity em Android é possível subscrever alterações ao modelo de dados através do método observable e do callback que será executado sempre que houverem alterações ao modelo de dados.

- A classe MutableLiveData e LiveData serão responsáveis por executar cada callback registado em cada componente Android;
- Quando o componente principal onde o LiveData é utilizado (ex: Activity) é removido ou destruído, também o objeto de LiveData é destruído.

LiveData

Implementação

- Para usar o conceito de LiveData em Android vamos usar ViewModels para encapsular os nossos modelos de dados;
- Os nossos modelos de dados devem representar live data, que queremos usar nos diferentes componentes de Android;

LiveData

Configuração [build.gradle (Project)]

```
ext{  
    lifecycle_version = "2.8.7"  
}
```

LiveData

Configuração [build.gradle (Module)]

```
dependencies {  
    // ViewModel  
    implementation "androidx.lifecycle:lifecycle-viewmodel:$lifecycle_version"  
    // LiveData  
    implementation "androidx.lifecycle:lifecycle-livedata:$lifecycle_version"  
}
```

LiveData

Definição [CounterViewModel.java]

```
public class CounterViewModel extends ViewModel {  
  
    private MutableLiveData<Integer> counter;  
  
    public void count(){  
        if (counter == null) {  
            counter = new MutableLiveData<Integer>();  
            counter.setValue(0);  
        } else{  
            counter.setValue(counter.getValue() + 1);  
        }  
  
        public MutableLiveData<Integer> getCount(){  
            if (counter == null) {  
                counter = new MutableLiveData<Integer>();  
                counter.setValue(0);  
            }  
            return counter;  
        }  
    }  
}
```

MutableLiveData significa que o valor irá sofrer alterações durante a execução

Estende ViewModel

Verificar se o counter foi iniciado. Em caso negativo iniciamos a zero.

Se já tivermos um counter ativo, incrementamos o seu valor

Esta função será observada e chamada sempre que o valor de counter sofrer alterações

LiveData

Adicionar o ViewModel [MainActivity.java]

```
private CounterViewModel counterViewModel;

@Override
protected void onCreate(Bundle savedInstanceState) {
    (...)

    counterViewModel = new ViewModelProvider(this).get(CounterViewModel.class);

    Observer<Integer> countObserver = new Observer<Integer>() {
        @Override
        public void onChanged(Integer count) {
            updateCounter(count);
        }
    };
    counterViewModel.getCount().observe(this, countObserver);
}

@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.btn_count:
            counterViewModel.count();
            break;
    }
}

private void updateCounter(int count){
    txtCount.setText(String.valueOf(count));
}
```

Definição do ViewModel

Criação de um Observer.
Sempre que ele for
invocado vai chamar a
função updateCounter

Adição do Observer à
função getCount. Esta
função devolve um
MutableLiveData

LiveData, ViewModel e Fragments

Com combinação de LiveData, ViewModel e Fragments é possível estabelecer comunicação entre fragments sem necessidade de “passar” essa comunicação pela Activity.

- Vamos utilizar o exemplo da aplicação “Hello Toast” para demonstrar esse comportamento.

LiveData, ViewModel e Fragments

Exemplo

Hello Toast

TOAST

DisplayFragment.java

0

CounterFragment.java

COUNT

LiveData, ViewModel, Fragments

Definição [SharedViewModel.java]

```
public class SharedViewModel extends ViewModel {  
  
    private MutableLiveData<Integer> counter;  
  
    public void count(){  
        if (counter == null) {  
            counter = new MutableLiveData<Integer>(); ←  
            counter.setValue(0);  
        } else{  
            counter.setValue(counter.getValue() + 1);  
        }  
    }  
  
    public MutableLiveData<Integer> getCount(){  
        if (counter == null) {  
            counter = new MutableLiveData<Integer>();  
            counter.setValue(0);  
        }  
        return counter;  
    }  
}
```

Extende ViewModel

Verificar se o counter foi iniciado. Em caso negativo iniciamos a zero.

Se já tivermos um counter ativo, incrementamos o seu valor

Esta função será observada e chamada sempre que o valor de counter sofrer alterações

LiveData, ViewModel, Fragments

Definição [CounterFragment.java]

```
public class CounterFragment extends Fragment implements View.OnClickListener {  
  
    private Button btnCounter;  
    private SharedViewModel sharedViewModel;  
  
    @Nullable  
    @Override  
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
        View mContentView = inflater.inflate(R.layout.counter_ui, container, false);  
        btnCounter = mContentView.findViewById(R.id.btn_add);  
        btnCounter.setOnClickListener(this);  
  
        sharedViewModel = new ViewModelProvider(requireActivity()).get(SharedViewModel.class);  
        return mContentView;  
    }  
  
    @Override  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.btn_add:  
                sharedViewModel.count();  
                break;  
        }  
    }  
}
```

Cada vez que o botão é pressionado, chamamos a função responsável por incrementar o contador

Definição do ViewModel

LiveData, ViewModel, Fragments

Definição [DisplayFragment.java]

```
public class DisplayFragment extends Fragment {  
  
    private TextView txtCounter;  
    private SharedViewModel sharedViewModel;  
  
    @Nullable  
    @Override  
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,  
                             @Nullable Bundle savedInstanceState) {  
        View mContentView = inflater.inflate(R.layout.counter_display, container, false);  
  
        txtCounter = mContentView.findViewById(R.id.txt_counter);  
  
        sharedViewModel = new ViewModelProvider(requireActivity()).get(SharedViewModel.class);  
  
        Observer<Integer> counterObserver = count -> updateCounter(count);  
        sharedViewModel.getCount().observe(getViewLifecycleOwner(), counterObserver);  
  
        return mContentView;  
    }  
  
    private void updateCounter(int count){  
        txtCounter.setText(String.valueOf(count));  
    }  
}
```

Definição do ViewModel

Criação de um Observer.
Sempre que ele for
invocado vai chamar a
função updateCounter

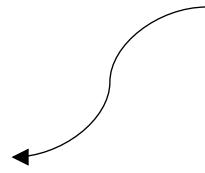
Adição do Observer à
função getCount. Esta
função devolve um
MutableLiveData

LiveData, ViewModel, Fragments

Definição [MainActivity.java]

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

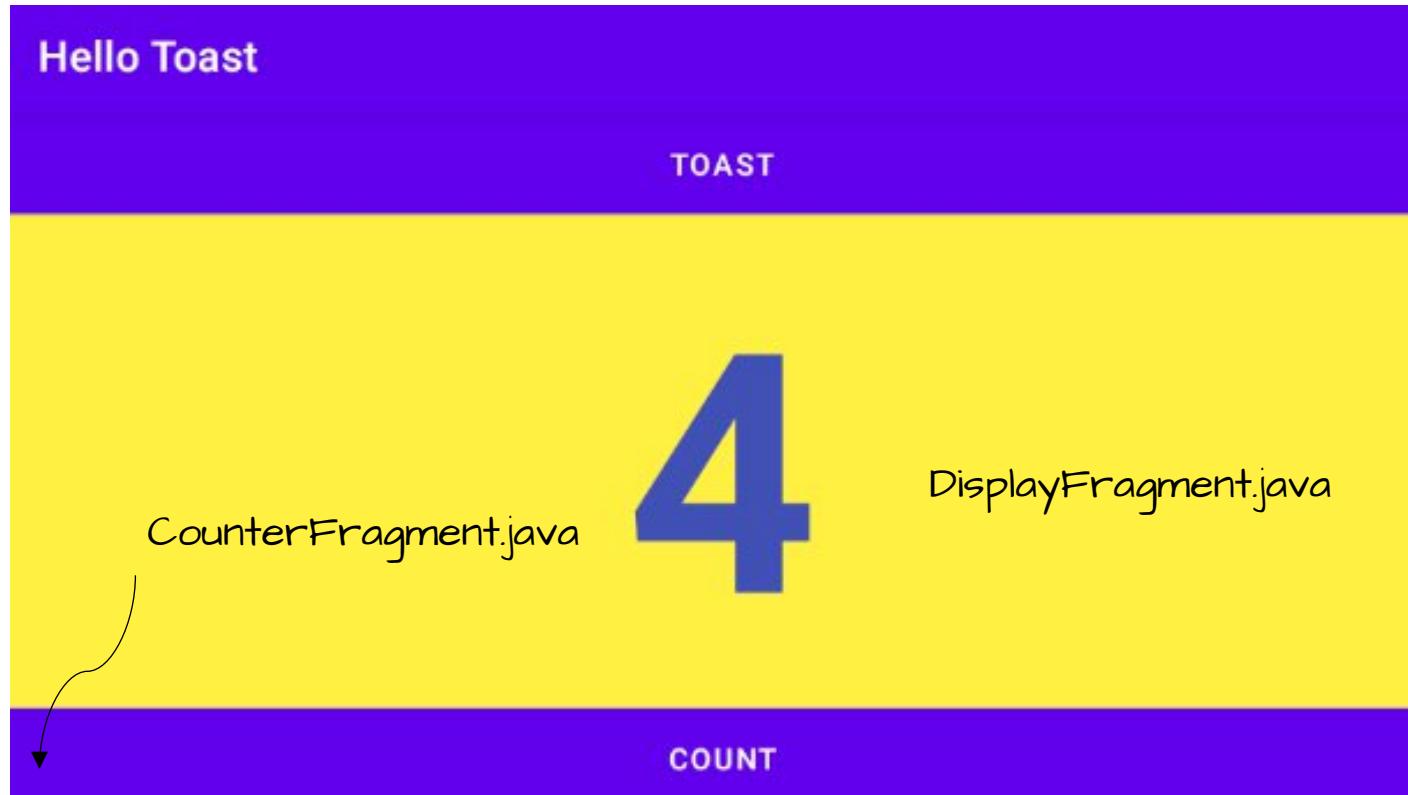
Fragments foram definidos no layout



MainActivity não tem informação sobre as comunicações feitas entre os fragments

LiveData, ViewModel, Fragments Resultado

Utilizador rodou ecrã



Hello Toast

TOAST

DisplayFragment.java

4

CounterFragment.java

Leitura Adicional

Live Data:

<https://developer.android.com/topic/libraries/architecture/livedata>

View Model:

<https://developer.android.com/topic/libraries/architecture/viewmodel.html>

Programação Para Dispositivos Móveis I

VIEW MODEL
LIVE DATA

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

GOOGLE PLAY SERVICES
LOCALIZAÇÃO, GOOGLE MAPS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

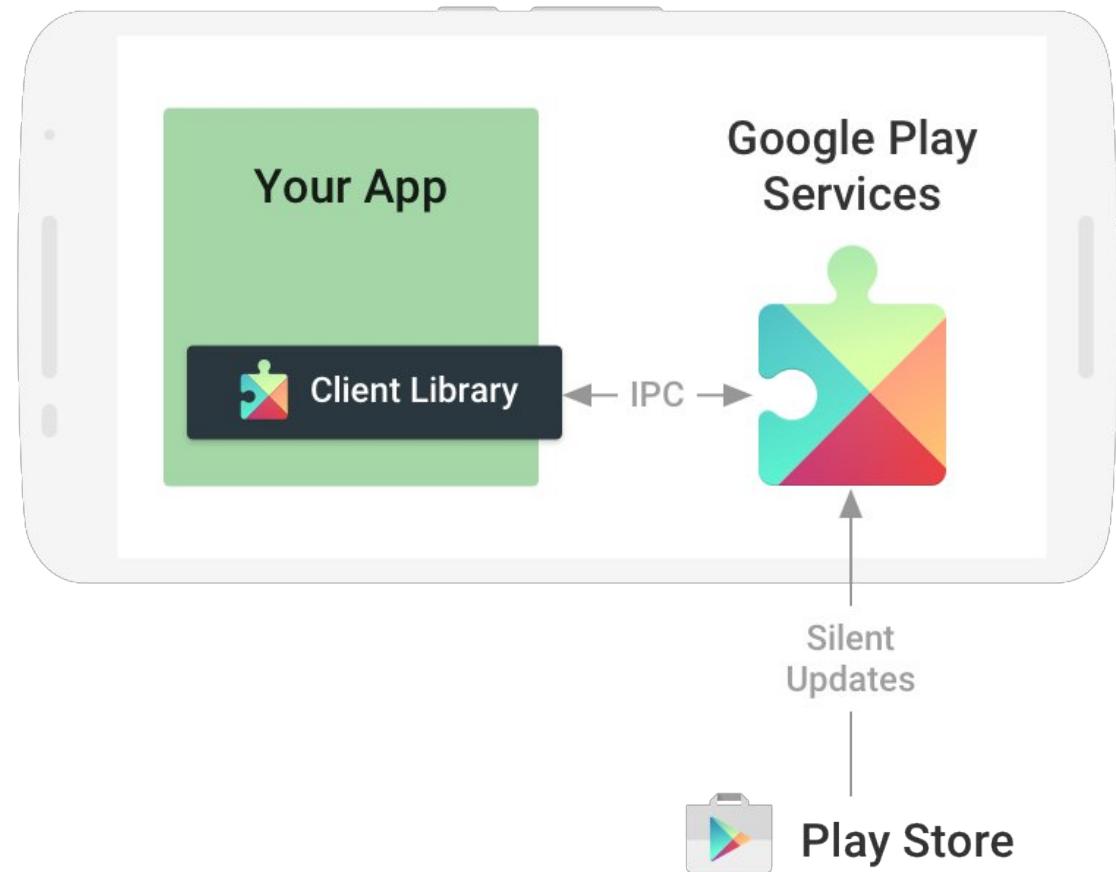
Índice

- Google Play Services;
- Localização;
- Google Maps;
- Leitura Adicional.

Google Play Services

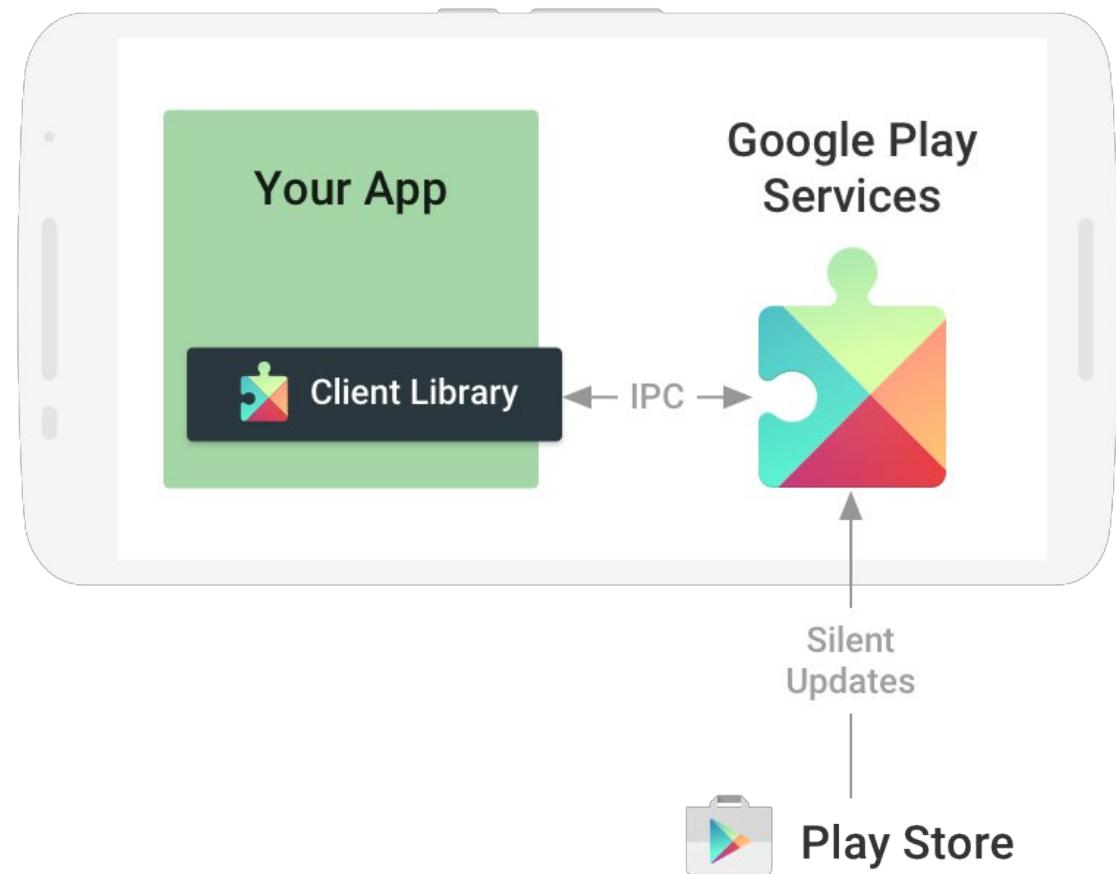
Biblioteca fornecida pela Google que permite interagir com diferentes tipos de funcionalidades, entre elas: mapas, localização, autenticação, etc.

- Os Google Play Services estão sempre a correr em background no dispositivo.



Google Play Services

- Múltiplas apps interagem ao mesmo tempo com os Google Play Services;
- Biblioteca atualizada automaticamente através da Google Play Store;
- Permite aos programadores usarem as APIs mais recentes do Google.



Google Play Services

Funcionalidades

Versões atuais a
abril de 2025



API	Description in build.gradle
Google Account Login	com.google.android.gms:play-services-auth:21.3.0
Google Actions, Base Client Library	com.google.android.gms:play-services-base:18.7.0
Google Awareness	com.google.android.gms:play-services-awareness:19.0.1
Google Cast	com.google.android.gms:play-services-cast:22.0.0
Cloud Messaging	com.google.firebaseio:firebase-messaging:24.1.1
Google Drive	com.google.android.gms:play-services-drive:17.0.0
Google Fit	com.google.android.gms:play-services-fitness:21.2.0
Google Location and Activity Recognition	com.google.android.gms:play-services-location:21.3.0
Google Mobile Ads	com.google.android.gms:play-services-ads:24.2.0
Mobile Vision	com.google.android.gms:play-services-vision:20.1.3
Google Nearby	com.google.android.gms:play-services-nearby:19.3.0
Google Play Game services	com.google.android.gms:play-services-games:23.2.0
SafetyNet	com.google.android.gms:play-services-safetynet:18.0.1
Google Pay	com.google.android.gms:play-services-wallet:19.4.0
Wear OS by Google	com.google.android.gms:play-services-wearable:19.0.0

Google Play Services

Configuração [build.gradle (Project)]

```
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}
```

Repositório Google para
aceder aos Google Play
Services

Google Play Services

Dependências [build.gradle (Module)]

```
dependencies {  
    implementation 'com.google.android.gms:play-services-maps:18.2.0'  
    implementation 'com.google.android.gms:play-services-location:21.2.0'  
}
```

Dependência para
Google Maps



Dependência para Serviços
de Localização

Localização

Existem diferentes formas de obter a localização de um dispositivo

Android:

- apenas a ultima localização conhecida;
- atualizações periódicas da localização (x em x minutos);
- atualizações com base na distancia percorrida (x km);
- geofencing (dispositivo entrou num raio pré-definido).

Localização

O sistema operativo fornece diferentes tipos de *providers* de localização, isto é, a utilização de GPS, Wifi ou outro tipo de hardware vai variar consoante os requisitos da aplicação:

- precisão da localização (accuracy);
- consumo de bateria;
- periodicidade das atualizações à localização (30 em 30 segundos, hora em hora).

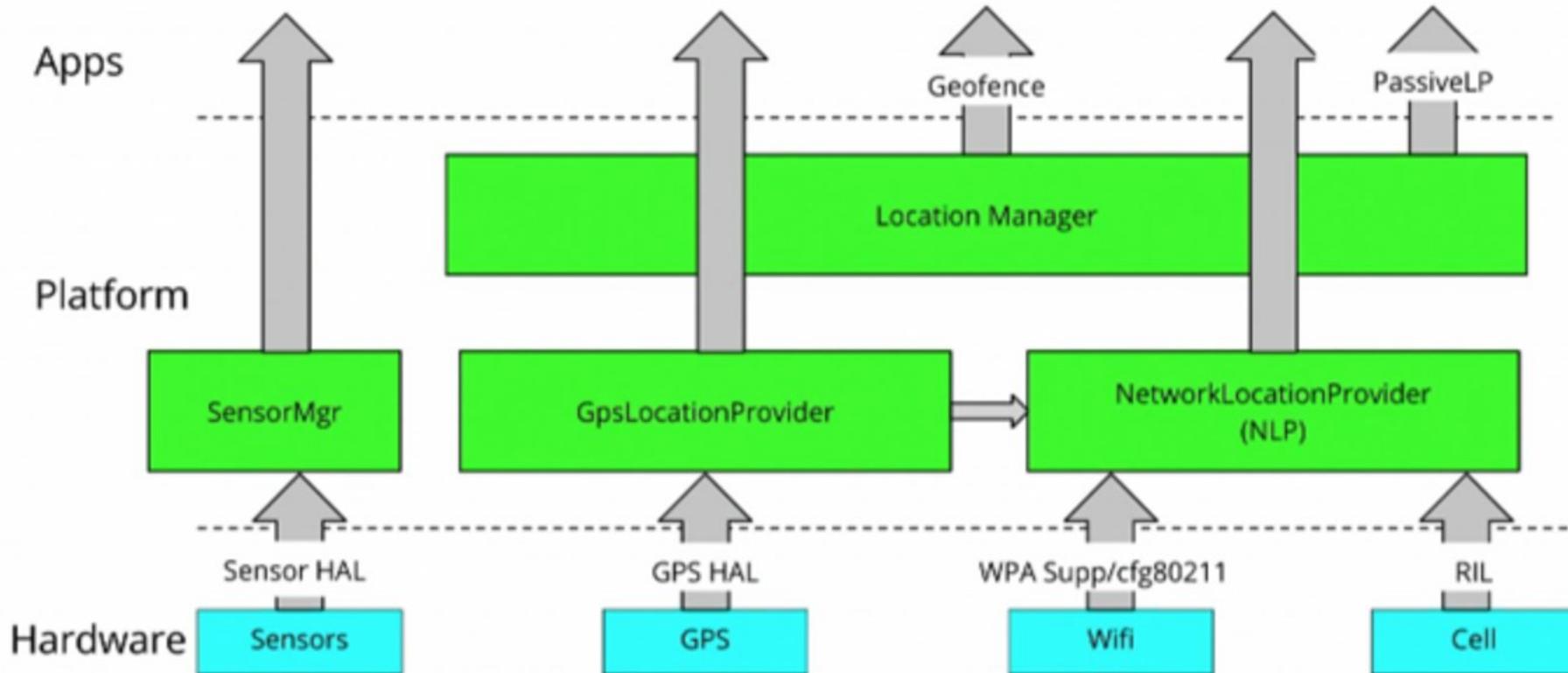
Localização

Providers de localização

	 GPS	 WiFi	 Cell	 Sensors
Power				
Accuracy				
Coverage				

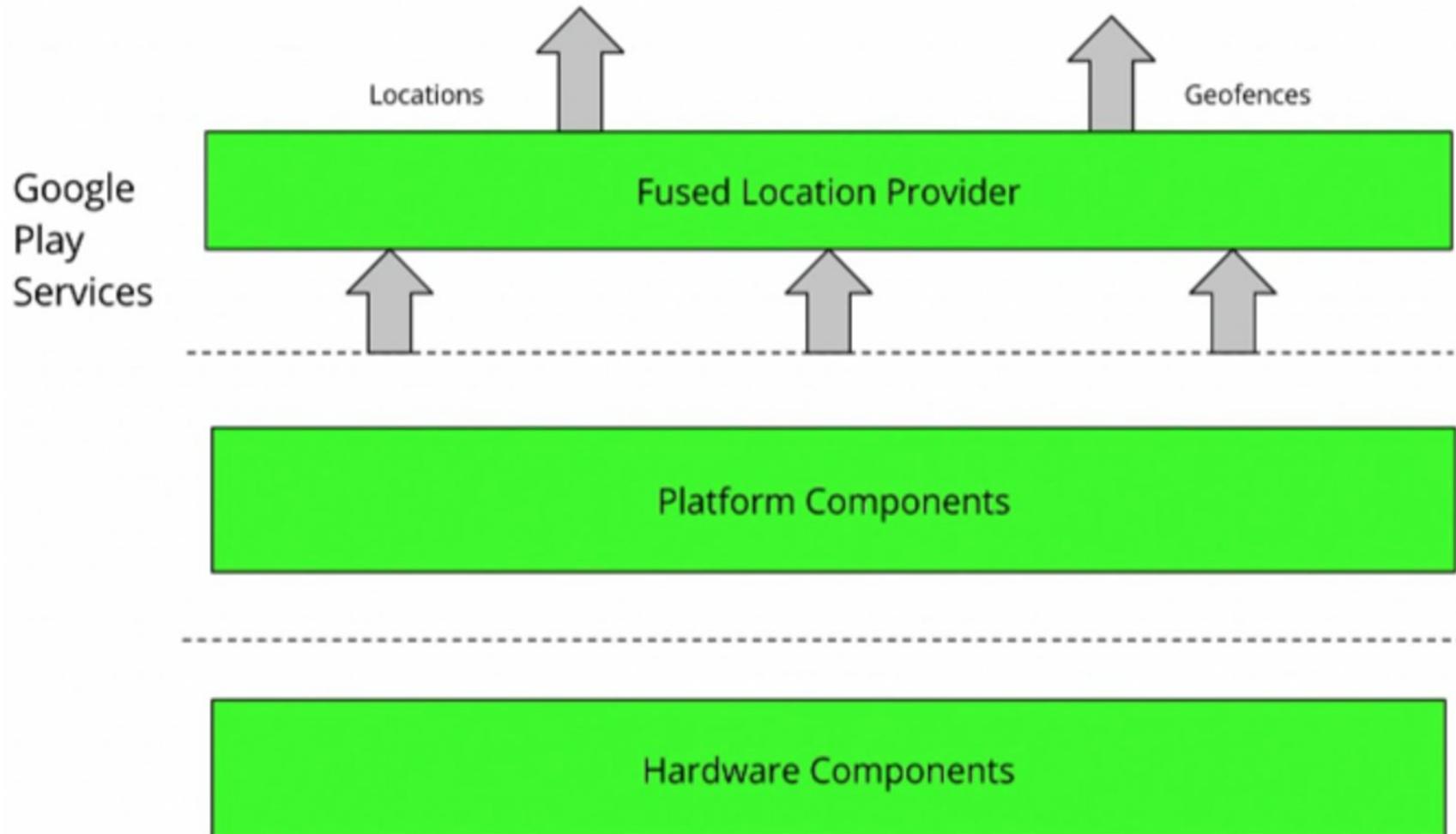
Localização

API Nativa de Localização



Localização

API Google Play Services de Localização



Localização

Permissões [AndroidManifest.xml]

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />  
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Wi-fi, Rede do
Operador

Wi-fi, Rede do
Operador, e GPS

Localização

Permissões [MainActivity.java]

```
private static final int REQUEST_FINE_LOCATION = 100;

@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    getLastLocation();
}

private void getLastLocation(){
    if(ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED){
        requestPermissions();
        return;
    }
}

private void requestPermissions(){
    ActivityCompat.requestPermissions(this,
        new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
        REQUEST_FINE_LOCATION});
}
```



Allow T_GMaps to access this device's location?

While using the app

Only this time

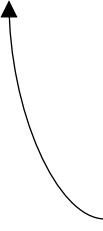
Deny

Nas versões mais recentes de Android temos de verificar se o utilizador deu permissões à aplicação para aceder aos recursos de localização

Localização

Cliente de localização [MainActivity.java]

```
private FusedLocationProviderClient fusedLocationProviderClient;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    (...)  
  
    fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(this);  
}
```



Este cliente permite interagir com as API de localização utilizando os Google Play Services

Localização

Obter última localização [MainActivity.java]

```
private void getLastLocation(){  
  
    if(ActivityCompat.checkSelfPermission(this,  
        Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED){  
        requestPermissions();  
        return;  
    }  
    fusedLocationProviderClient.getLastLocation()  
        .addOnSuccessListener(this,  
            new OnSuccessListener<Location>() {  
                @Override  
                public void onSuccess(Location location) {  
                    if(location != null){  
                        //TODO: Código em caso de sucesso  
                    }  
                }  
            })  
        .addOnFailureListener(this, new OnFailureListener() {  
            @Override  
            public void onFailure(@NonNull Exception e) {  
                //TODO: Código em caso de erro  
            }  
        });  
}
```

Para obter a última localização conhecida do dispositivo devemos invocar o cliente fused e adicionar o listener de sucesso e erro.

Localização

Atualizações periódicas

Para obter atualizações periódicas de localização é necessário criar um **LocationRequest** e um **LocationCallback**.

- **LocationRequest** define os parâmetros da localização que queremos obter (precisão, periodicidade, timeout);
- **LocationCallback** é o *listener* que vai receber periodicamente uma lista de localizações.

Localização

Atualizações periódicas (LocationRequest) [MainActivity.java]

```
private LocationRequest locationRequest;

@Override
protected void onCreate(Bundle savedInstanceState) {
    (...)

    locationRequest = new LocationRequest();
    locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    locationRequest.setInterval(10000);
    locationRequest.setFastestInterval(5000);
}
```

Localização

Atualizações periódicas (LocationCallback) [MainActivity.java]

```
private LocationCallback locationCallback;

@Override
protected void onCreate(Bundle savedInstanceState) {
    (...)

    locationCallback = new LocationCallback(){
        @Override
        public void onLocationResult(LocationResult locationResult) {
            super.onLocationResult(locationResult);
            for(Location location : locationResult.getLocations()){
                //TODO: Atualizar UI com a localização
            }
        }
    };
}
```

Localização

Iniciar Atualizações periódicas [MainActivity.java]

```
private void startLocationUpdates(){
    //TODO: Verificar permissões
    (...)

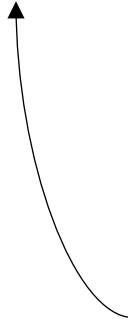
    fusedLocationProviderClient.requestLocationUpdates(
        locationRequest,
        locationCallback,
        null);
}
```

Para iniciar as atualizações periódicas de localização, devemos fazer um request ao cliente

Localização

Parar Atualizações periódicas [MainActivity.java]

```
private void stopLocationUpdates(){  
    fusedLocationProviderClient.removeLocationUpdates(locationCallback);  
}
```



Para parar as atualizações periódicas de localização, basta remover o LocationCallback do cliente

Mapas

As aplicações em Android podem usar mapas e Informação geográfica utilizando bibliotecas externas:

- Google Maps;
- Here Maps;
- Leaflet.

Para efeitos de exemplo iremos utilizar o Google Maps presente nos Play Services vistos anteriormente.

Google Maps

API Key

É necessário obter uma API key para aceder aos serviços do Google Maps.

- A chave pode ser obtida no seguinte endereço:

<https://developers.google.com/maps/documentation/android-sdk/get-api-key>

Google Maps

API Key -> 1º Passo

Efetuar login com credenciais Google
(ex. gmail) se assim solicitado

<https://console.cloud.google.com/projectselector2/google/maps-apis>

The screenshot shows the Google Cloud Platform dashboard. At the top, there is a blue header bar with the text "Google Cloud Platform" and "Select a project". To the right of the header are several icons: a magnifying glass for search, a square for projects, a question mark for help, a bell for notifications, three vertical dots for more options, and a user profile picture. Below the header, the word "Dashboard" is displayed. A light gray callout box contains the text "To view this page, select a project." followed by two buttons: "SELECT PROJECT" and "CREATE PROJECT". A hand-drawn style arrow originates from the text "Criar projeto" and points towards the "CREATE PROJECT" button.

≡ Google Cloud Platform Select a project

Dashboard

To view this page, select a project.

SELECT PROJECT CREATE PROJECT

Criar projeto

Google Maps

API Key -> 2º Passo

Project name * — pdm-2021

Project ID: pdm-2021. It cannot be changed later. [EDIT](#)

Location * — No organisation [BROWSE](#)

Parent organisation or folder

CREATE **CANCEL**

Nome do Projeto

Criar Projeto

Google Maps

API Key -> 3º Passo

The screenshot shows the Google Cloud Platform dashboard for a project named "pdm-2021". The "APIs" section is highlighted, showing a chart of requests per second over time. A callout arrow points from the text "Aceder à zona de API's" to the "Go to APIs overview" button at the bottom of the chart area.

Google Cloud Platform

pdm-2021

Search products and resources

DASHBOARD ACTIVITY RECOMMENDATIONS CUSTOMISE

Project info

Project name: pdm-2021

Project ID: pdm-2021

Project number: 82894806614

ADD PEOPLE TO THIS PROJECT

→ Go to project settings

Resources

This project has no resources

API APIs

Requests (requests/sec)

No data is available for the selected time frame.

19:30 20:00

→ Go to APIs overview

Google Cloud Platform status

All services normal

→ Go to Cloud status dashboard

Monitoring

Set up alerting policies

Create uptime checks

View all dashboards

→ Go to Monitoring

Google Maps

API Key -> 4º Passo

The screenshot shows the Google Cloud Platform API & Services dashboard. On the left sidebar, under the 'API' section, 'Dashboard' is selected. In the main area, the title 'APIs & Services' is displayed above a button labeled '+ ENABLE APIs AND SERVICES'. A handwritten note 'Ativar API's e Serviços' with an arrow points to the enable button. Below this, there is a traffic chart titled 'Traffic' showing data over a 30-day period from November 08 to November 29. The chart displays traffic rates of 1.0/s, 0.8/s, 0.6/s, 0.4/s, and 0.2/s. The time scale at the bottom includes Nov 08, Nov 15, Nov 22, and Nov 29.

Google Maps

API Key -> 5º Passo

The screenshot shows the Google Cloud Platform API Library interface. At the top, there's a blue header bar with the text "Google Cloud Platform" and "pdm-2021". A search bar says "Search products and resources". Below the header, a left navigation bar has a "API Library" link. The main area is titled "Welcome to the API Library" with the sub-instruction "The API Library has documentation, links and a smart search experience.". A search bar at the bottom says "Search for APIs & Services". On the left, a sidebar titled "Filter by" includes sections for "VISIBILITY" (Public 312, Private 4) and "CATEGORY" (Advertising 14, Analytics 3). The main content area is titled "Escolher MAPS SDK for Android" and shows two items: "Maps SDK for Android" (Google) and "Maps SDK for iOS" (Google). A handwritten note "Escolher MAPS SDK for Android" with an arrow points to the first item.

Welcome to the API Library

The API Library has documentation, links and a smart search experience.

Search for APIs & Services

Filter by

VISIBILITY

Public (312)

Private (4)

CATEGORY

Advertising (14)

Analytics (3)

Maps

VIEW ALL (17)

Escolher MAPS SDK for Android

Maps SDK for Android

Google

Maps for your native Android app.

Maps SDK for iOS

Google

Maps for your native iOS app.

Google Maps

API Key -> 6º Passo

The screenshot shows the Google Cloud Platform dashboard. At the top, there is a blue header bar with the text "Google Cloud Platform", a dropdown menu "pdm-2021", a search bar "Search products and resources", and several icons for notifications and user profile.

The main content area displays a service page for "Maps SDK for Android" by Google. The page includes a logo featuring a stylized Android figure inside a red and yellow square, the title "Maps SDK for Android", the developer name "Google", and a brief description: "Maps for your native Android app." Below this, there is a blue button labeled "ENABLE". A handwritten-style annotation with the word "Ativar" (Activate) and an arrow points to the "ENABLE" button.

Google Maps

API Key -> 7º Passo

The screenshot shows the Google Cloud Platform interface. The top navigation bar includes the 'Google Cloud Platform' logo, a project dropdown ('pdm-2021'), a search bar ('Search products and resources'), and various status icons. The main menu on the left lists 'Home', 'Pins appear here', 'Marketplace', 'Billing', 'APIs & Services' (selected), 'Support', 'IAM & Admin', 'Getting started', 'Security', and 'Compliance'. The 'APIs & Services' menu is expanded, showing sub-options: 'Dashboard', 'Library', 'Credentials' (selected), 'OAuth consent screen', 'Domain verification', and 'Page usage agreements'. A callout bubble points to the 'Credentials' option with the handwritten note 'Aceder à zona de credenciais'. The central content area is titled 'Credentials compatible with this API' and contains a message: 'Remember to configure the OAuth consent screen with information about your application.' Below this is a table with columns: 'Creation date', 'Restrictions', and 'Key'. A blue button labeled 'CONSENT SCREEN' is visible above the table.

Google Maps

API Key -> 8º Passo

Adicionar credenciais

The screenshot shows the Google Cloud Platform interface with the project 'pdm-2021'. The left sidebar is titled 'API & Services' and includes 'Dashboard', 'Library', 'Credentials' (which is selected), 'OAuth consent screen', 'Domain verification', and 'Page usage agreements'. The main content area is titled 'Credentials' and features a 'CREATE CREDENTIALS' button. A callout bubble points to the 'API key' option under the 'Create credentials to access' section. The 'API keys' section below shows a table with one row: 'Name' (checkbox) and 'No API keys to display'.

Google Maps

API Key -> 9º Passo

API key created

Use this key in your application by passing it with the `key=API_KEY` parameter.

Your API key



Copiar API key

⚠ Restrict your key to prevent unauthorised use in production.

CLOSE

RESTRICT KEY

Google Maps

Dependências [build.gradle (Module)]

```
dependencies {  
    implementation 'com.google.android.gms:play-services-maps:19.2.0'  
    implementation 'com.google.android.gms:play-services-location:21.3.0'  
}
```

Dependência para
Google Maps



Dependência para Serviços
de Localização

Google Maps

API KEY [AndroidManifest.xml]

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="pt.ipp.estg.t_gmaps">  
  
    <application  
        <meta-data android:name="com.google.android.geo.API_KEY"  
                  android:value="AAAAaaaAAAAaaaAAAAAaaaaAAA" />  
    </application>  
  
</manifest>
```



Substituir pela API Key
gerada anteriormente

Google Maps

Fragment [main_activity.xml]

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

    <fragment
        android:id="@+id/maps_google"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        xmlns:map="http://schemas.android.com/apk/res-auto"
        map:cameraZoom="16"
        map:mapType="normal"
        map:uiCompass="true"
        map:uiRotateGestures="true"
        map:uiScrollGestures="true"
        map:uiTiltGestures="true"
        map:uiZoomControls="true"
        map:uiZoomGestures="true"
    />

</LinearLayout>
```



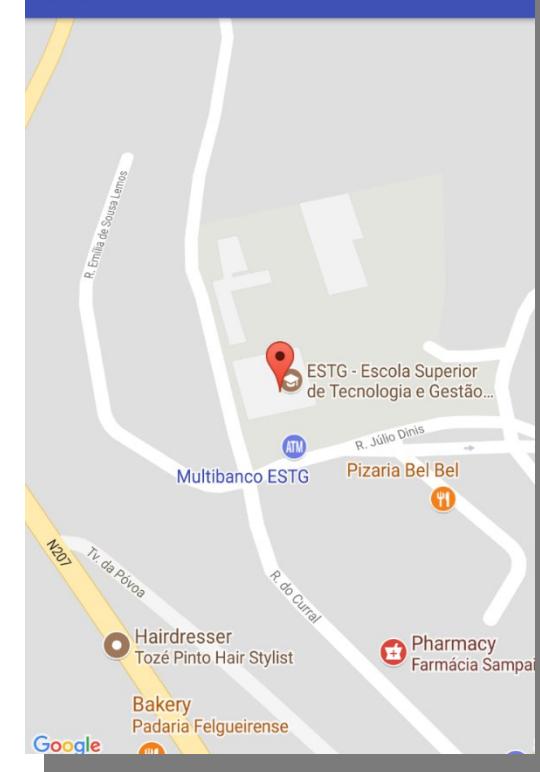
Configurações do mapa. É possível controlar níveis de zoom, tipo de mapa, localização inicial, etc.

Google Maps

Adicionar mapa [ActivityMain.java]

Implementa
OnMapReadyCallback

```
public class MainActivity extends AppCompatActivity implements OnMapReadyCallback {  
  
    private SupportMapFragment mapFragment;  
    private GoogleMap mGoogleMap;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
  
        (...)  
  
        mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.maps_google);  
        mapFragment.getMapAsync(this);  
    }  
  
    @Override  
    public void onMapReady(GoogleMap googleMap) {  
        mGoogleMap = googleMap;  
    }  
}
```



Google Maps

Adicionar Marker [ActivityMain.java]

```
private void addMarker(Location location, String title, String snippet){
    LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());

    Marker marker = mGoogleMap.addMarker(new MarkerOptions()
        .position(latLng)
        .title(title)
        .snippet(snippet));
}
```

Google Maps

Marker Click [ActivityMain.java]

```
mGoogleMap.setOnMarkerClickListener(new GoogleMap.OnMarkerClickListener() {  
    @Override  
    public boolean onMarkerClick(Marker marker) {  
        //TODO: Adicionar ação  
        return false;  
    }  
});
```

Google Maps

Zoom para localização [ActivityMain.java]

```
private void zoomToLocation(Location location, String title, String snippet){  
  
    LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());  
  
    CameraUpdate cameraUpdate = CameraUpdateFactory.newLatLng(latLng);  
    mGoogleMap.animateCamera(cameraUpdate);  
}
```

Google Maps

Procurar localização pelo nome [ActivityMain.java]

```
private LatLng getLocation(String location) {  
    Geocoder geocoder = new Geocoder(this); ←  
    List<Address> addressList = new ArrayList<>();  
    Address address;  
    LatLng latlng = null;  
  
    try {  
        addressList = geocoder.getFromLocationName(location, 1);  
    } catch (IOException ioException) {  
        ioException.printStackTrace();  
    } finally {  
        if(addressList.size() > 0){  
            address = addressList.get(0);  
            latlng = new LatLng(address.getLatitude(), address.getLongitude());  
            return latlng;  
        }  
    }  
  
    return new LatLng(0,0);  
}
```

Objeto Geocoder permite obter dados geográficos (ex. Latitude e Longitude) através de um nome de localização

Leitura Adicional

Setup Play Services:

<https://developers.google.com/android/guides/setup>

Location:

<https://developer.android.com/develop/sensors-and-location/location>

Google Maps for Android:

<https://developers.google.com/maps/documentation/android-sdk/overview>

Here Maps for Android:

<https://developer.here.com/products/here-sdk>

Programação Para Dispositivos Móveis I

GOOGLE PLAY SERVICES
LOCALIZAÇÃO, GOOGLE MAPS

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

SERVICES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Índice

- Services;
- Tipos de Services;
- Ciclo de Vida;
- Unbounded Services;
- Bounded Services;
- Leitura Adicional.

Existem algumas operações que queremos que continuem a ser executadas, independentemente da aplicação que temos ativa de momento.

Por exemplo, se iniciarmos o download de um ficheiro, não queremos que esse processo seja interrompido apenas porque trocamos de aplicação.

Services

O que são?

Componente de uma aplicação Android (tal como uma Activity) que pode controlar **operações demoradas em background, não possui interface gráfica e tem um ciclo de vida simplificado.**

- Utilizado por outros componentes da aplicação e pode ficar em execução mesmo que o utilizador saia da aplicação;
 - Ex. leitor de música ou download de um ficheiro.
- Deve ser independente dos componentes que o utilizam, de forma a poder interagir com novos componentes;
- Pode lançar e controlar uma nova Thread ou AsyncTask se assim for pretendido.

Services vs Threads

Um serviço é simplesmente um componente que pode funcionar em segundo plano, mesmo quando o utilizador não está a interagir com a aplicação, por isso só deve ser criado se for essa a intenção.

Se tiver de executar uma tarefa fora da UI Thread, mas apenas enquanto o utilizador estiver a interagir com a aplicação, deverá, em vez disso, criar uma nova thread no contexto de outro componente da aplicação.

Ex. se quisermos tocar alguma música, mas apenas enquanto a aplicação estiver em execução.

Services

Tipos de Services

- **Started Services:** pode funcionar em segundo plano indefinidamente, mesmo quando a Activity que o iniciou é destruída. Uma vez que a operação é feita, o serviço para. (Ex. download de ficheiro);
- **Bound Services:** está ligado a outra componente de aplicação, tal como uma Actividade, através do método `bindService()`. A Atividade pode interagir com ele, enviar pedidos, e obter resultados. Um Bound Service funciona desde que os componentes estejam vinculados a ele. Quando os componentes já não estão vinculados, o serviço é destruído. (Ex. Odómetro para medir a distância percorrida por um veículo);
- **Schedule Services:** é um serviço que está programado para funcionar num determinado momento temporal.

Services

Started Service (Background)

Um background service realiza uma operação que não é diretamente perceptível pelo utilizador.

- Invocado por um componente (por ex., uma Activity) através do método `startService()`;
- Depois de inicializado o serviço pode estar em **background indefinidamente**, mesmo que componente que o invocou seja destruído;
- Normalmente é utilizado para realizar **uma operação única**, que não devolve um resultado a quem o invocou;
- **É gerido pelo sistema** em caso de falta de memória ou bateria.

Services

Started Service (Foreground)

Um serviço em foreground realiza alguma operação que é perceptível para o utilizador. Estes serviços devem exibir uma notificação para que os utilizadores estejam ativamente conscientes de que o serviço está em execução (esta notificação não pode ser rejeitada a menos que o serviço seja interrompido ou removido do primeiro plano), e continuam a funcionar mesmo quando o utilizador não está a interagir com a aplicação.

- Invocado por um componente (Ex. uma Activity) através do método `startService()`;
- Depois de inicializado o serviço pode estar em background **indefinidamente**, mesmo que componente que o invocou seja destruído;
- Normalmente é utilizado para executar **ações que devem ser perceptíveis** pelo utilizador;
- **Não é gerido pelo sistema** em caso de falta de memória ou bateria.

Services

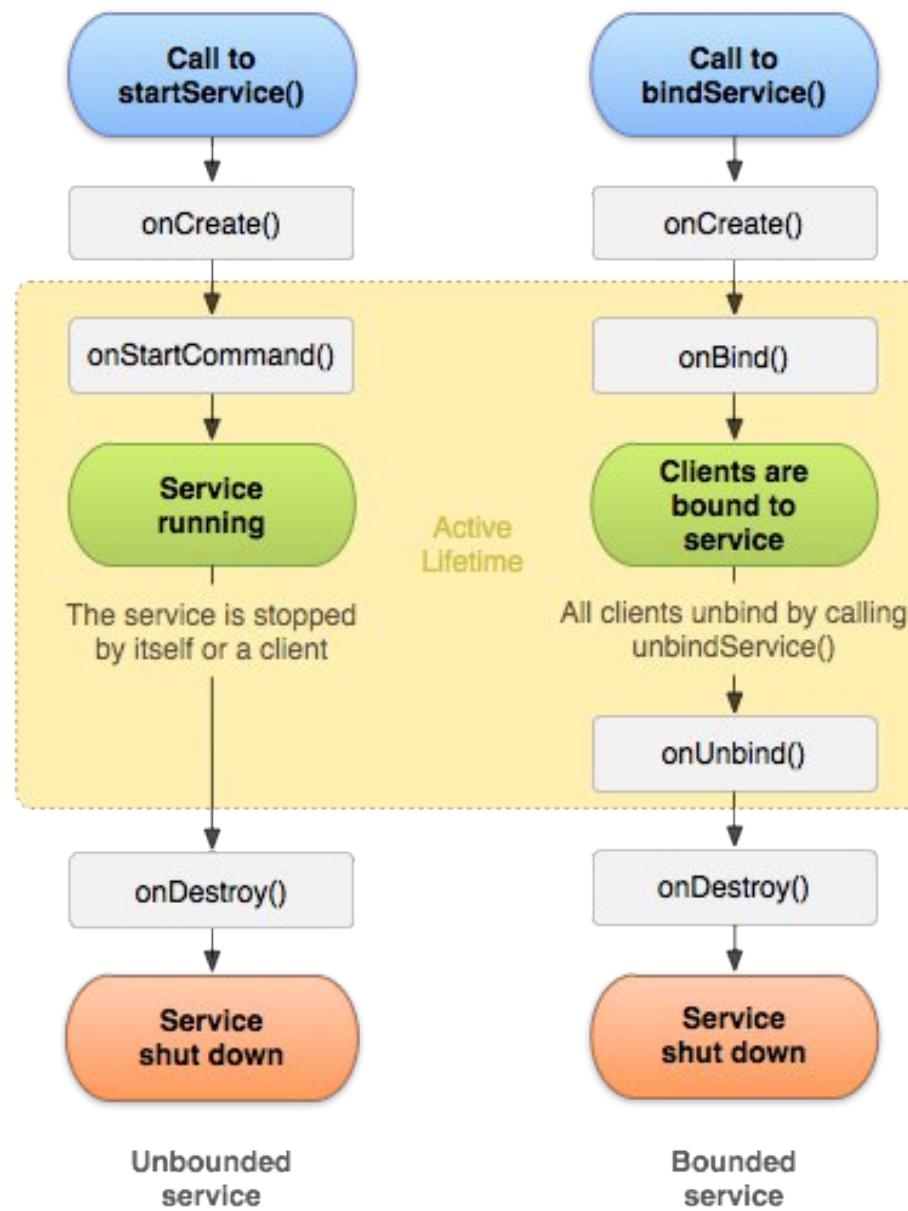
Bound Services

Oferece uma interface cliente-servidor que permite aos componentes interagir com o serviço, enviar pedidos, receber resultados, e mesmo fazê-lo através de processos com comunicação interprocessada (IPC). Um serviço vinculado funciona apenas enquanto outro componente da aplicação estiver vinculado. Múltiplos componentes podem ligar-se ao serviço de uma só vez, mas quando todos eles se desvinculam, o serviço é destruído.

- Invocado por um componente através do método `bindService()`;
- Interface tipo cliente-servidor, que permite aos componentes interagir com o service;
- Existe apenas enquanto houver pelo menos um componente conectado ao mesmo;
- Vários componentes podem estar conectados ao serviço de cada vez, mas quando todos eles fazem o `unbind()`, o service é destruído.

Services

Ciclo de Vida



Services

Ciclo de Vida (Unbounded)

`startService(Intent)`

- Método utilizado na Activity para iniciar um Service;

`onCreate()`

- Executado durante a criação do Service;

`onStartCommand()`

- Invocado quando o Service é utilizado no modo **Started** (através do método `startService()`);
 - O Intent recebido representa um comando dado ao Service.

Services

Ciclo de Vida [onStartCommand()]

Devolve uma das flags com diferentes comportamentos caso o serviço seja destruído após a execução deste método:

- START_NOT_STICKY
 - Não recria o serviço, a menos que hajam Intents pendentes de entrega. Esta é a opção mais segura para evitar correr o serviço quando não é necessário, e quando a sua aplicação pode simplesmente reiniciar quaisquer trabalhos inacabados.
- START_STICKY
 - o serviço é recriado mas o Intent recebido no onStartCommand não será recebido novamente;
- START_REDELIVER_INTENT
 - o service é recriado e o onStartCommand irá receber o último Intent recebido antes do serviço ser destruído.

Services

Ciclo de Vida (Bounded)

onBind()

- Invocado quando o Service é utilizado no modo Bound (através do método `bindService()`);
- Devolve uma interface de comunicação com o Service (`IBinder`);

onUnbind()

- Invocado quando o Service é utilizado no modo Bound (através do método `unbindService()`);
- Devolve true ou false conforme permite ou não novos bindings (rebind);

onDestroy()

- Executado durante a destruição do Service.

Services

Declaração [AndroidManifest.xml]

```
<manifest ... >
    ...
    <application ... >
        <service android:name=".ExampleService"
                android:exported="false" />
    ...
</application>
</manifest>
```

O nome tem um ponto de forma a que o Android possa combinar com o package name

Os serviços devem ser declarados/registados no Manifest da aplicação

Este atributo define se o serviço pode ser utilizado por outras Aplicações

Services

Unbounded [ExampleService.java]

```
public class ExampleService extends Service {  
  
    public static final String EXTRA_MESSAGE = "message";  
  
    private final String CHANNEL_ID = "SERVICE_NOTIFICATION";  
    private final int NOTIFICATION_ID = 123;  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        createNotificationChannel(); ← Criação de um notification channel para  
    }                                         mostrar uma notificação  
  
    @Nullable  
    @Override  
    public IBinder onBind(Intent intent) {  
        return null; ← Se não tencionarmos criar um Bound  
    }                                         Service, retornamos null no método que  
                                                faz o Bound do serviço
```

Services

Unbounded [ExampleService.java]

Neste exemplo vamos esperar 10 segundos e depois notificar o utilizador

```
@Override  
public int onStartCommand(Intent intent, int flags, int startId) {  
    synchronized (this){  
        try {  
            wait(10000);  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        } finally {  
            String text = intent.getStringExtra(this.EXTRA_MESSAGE);  
            showText(text);  
        }  
    }  
  
    return START_NOT_STICKY;  
}
```

Método que criamos para notificar o utilizador

Se o serviço for destruído, não é recriado

Services

Unbounded [ExampleService.java]

```
private void showText(String text) {  
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)  
        .setSmallIcon(android.R.drawable.sym_def_app_icon)  
        .setContentTitle(getString(R.string.question))  
        .setContentText(text)  
        .setPriority(NotificationCompat.PRIORITY_HIGH)  
        .setVibrate(new long[] {0,1000})  
        .setAutoCancel(true);  
  
    Intent actionIntent = new Intent(this, MainActivity.class);  
    PendingIntent actionPendingIntent = PendingIntent.getActivity(  
        this,  
        0,  
        actionIntent,  
        PendingIntent.FLAG_UPDATE_CURRENT);  
  
    builder.setContentIntent(actionPendingIntent);  
  
    NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);  
    notificationManager.notify(NOTIFICATION_ID, builder.build());  
}
```

Builder da notificação

Adicionar um PendingIntent para o utilizador abrir a App se carregar na notificação

Iniciar notificação

Services

Unbounded [ExampleService.java]

```
private void createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

        String channelName = "Service Notifications";
        String channelDescription = "Include all Service Notifications";
        int channelImportance = NotificationManager.IMPORTANCE_DEFAULT;

        NotificationChannel notificationChannel = new NotificationChannel(CHANNEL_ID, channelName, channelImportance);

        notificationChannel.setDescription(channelDescription);
        notificationChannel.enableVibration(true);
        notificationChannel.setSound(defaultSoundUri, null);

        NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        notificationManager.createNotificationChannel(notificationChannel);
    }
}
```

Definição do Notification Channel

Services

Unbounded [MainActivity.java]

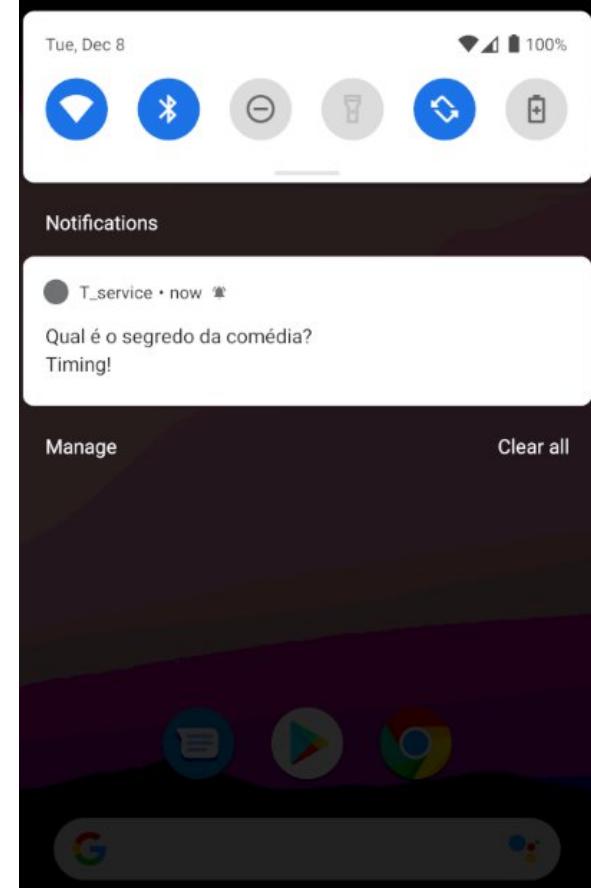
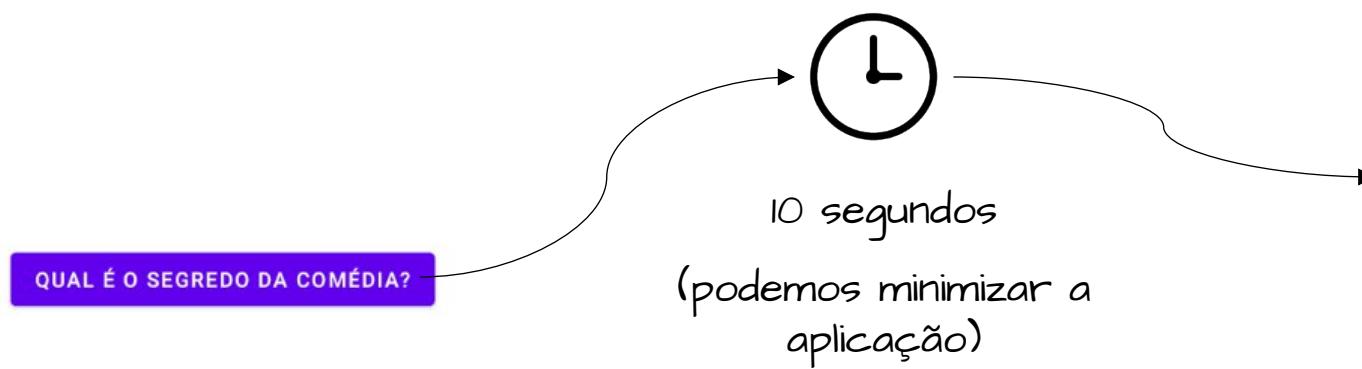
```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
    private Button btnService;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        btnService = findViewById(R.id.btn_service);  
        btnService.setOnClickListener(this::onClick);  
    }  
  
    @Override  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.btn_service:  
                Intent intent = new Intent(this, ExampleService.class);  
                intent.putExtra(ExampleService.EXTRA_MESSAGE, getResources().getString(R.string.response));  
                startService(intent);  
                break;  
        }  
    }  
}
```



Iniciar o serviço ao pressionar o botão

Services

Unbounded [Resultado]



Services

Declaração [AndroidManifest.xml]

Os serviços devem ser declarados/registados no Manifest da aplicação

```
<manifest ... >  
  ...  
  <application ... >  
    <service android:name=".OdometerService"  
            android:exported="false" />  
  ...  
 </application>  
</manifest>
```

O nome tem um ponto de forma a que o Android possa combinar com o package name

Este atributo define se o serviço pode ser utilizado por outras Aplicações

Services

Bounded [OdometerService.java]

```
public class OdometerService extends Service {  
  
    private final IBinder binder = new OdometerBinder();  
    private final Random random = new Random();  
    private double currentDistance = 0.0;  
  
    public class OdometerBinder extends Binder{  
        OdometerService getOdometer(){  
            return OdometerService.this;  
        }  
  
        @Nullable  
        @Override  
        public IBinder onBind(Intent intent) {  
            return binder;  
        }  
  
        public double getDistance(){  
            currentDistance += random.nextDouble();  
            return currentDistance;  
        }  
    }  
}
```

Quando criamos um Bound Service
temos de fornecer uma
implementação de um Binder

Retornamos o Binder que
criamos

Utilização do objeto Random para
simulação e geração de valores
aleatórios

Método para obter a distância
percorrida

Services

Bounded [MainActivity.java]

```
public class MainActivity extends AppCompatActivity{  
  
    private OdometerService odometerService;  
    private boolean bound = false;  
  
    private ServiceConnection serviceConnection = new ServiceConnection() {  
        @Override  
        public void onServiceConnected(ComponentName name, IBinder service) {  
            Variável de controlo  
            para percebermos  
            o estado de  
            vinculação do  
            serviço  
            OdometerService.OdometerBinder odometerBinder = (OdometerService.OdometerBinder) service;  
            odometerService = odometerBinder.getOdometer();  
            bound = true;  
        }  
  
        @Override  
        public void onServiceDisconnected(ComponentName name) {  
            bound = false;  
        }  
    };
```

Criação de um objeto de
ligação ao Service

Utilizamos o IBinder para obter
uma referência ao serviço

Ações a executar quando um serviço e a
Activity são desconectados

Services

Bounded [MainActivity.java]

```
@Override  
protected void onStart() {  
    super.onStart();  
    Intent intent = new Intent(this, OdometerService.class);  
    bindService(intent, serviceConnection, Context.BIND_AUTO_CREATE);  
}
```

Iniciamos a vinculação com o

serviço

```
@Override  
protected void onStop() {  
    super.onStop();  
    if(bound){  
        unbindService(serviceConnection);  
        bound = false;  
    }  
}
```

Se o serviço estiver vinculado, removemos
essa vinculação

Services

Bounded [MainActivity.java]

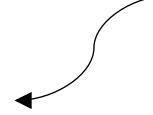
```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    displayDistance();  
}
```

Services

Bounded [MainActivity.java]

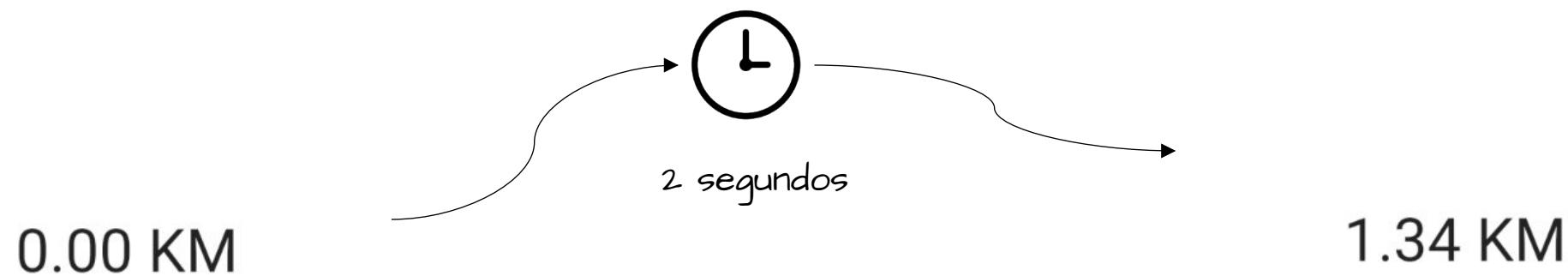
```
private void displayDistance(){
    final TextView distanceView = findViewById(R.id.txt_distance);
    final Handler handler = new Handler(Looper.myLooper());
    handler.post(new Runnable() {
        @Override
        public void run() {
            double distance = 0.0;
            if(bound && odometerService != null){
                distance = odometerService.getDistance();
            }
            String distanceStr = String.format(Locale.getDefault(), "%1$, .2f KM", distance);
            distanceView.setText(distanceStr);
            handler.postDelayed(this, 2000);
        }
    });
}
```

Se tivermos uma referência do serviço e estivermos vinculados, chamamos o método



Services

Bounded [Resultado]



Componentes baseados em Services

Existem na plataforma componentes que tentam simplificar o trabalho dos services

- Por exemplo para executar tarefas curtas num service podemos usar o JobIntentService
<https://developer.android.com/reference/androidx/core/app/JobIntentService>
- Para facilitar o processamento paralelo podemos usar Loopers e ServiceHandlers em conjunto com o service, e assegurar que o trabalho dentro do service é executado fora da main thread
<https://developer.android.com/guide/components/services#ExtendingService>

Leitura Adicional

Services:

<https://developer.android.com/guide/components/services>

Threads e Services:

<https://developer.android.com/guide/components/processes-and-threads.html#Threads>

Programação Para Dispositivos Móveis I

SERVICES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

CAMERA E MEDIA

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Índice

- Camera;
- Media Recorder;
- Leitura Adicional.

CameraX

CameraX é um novo componente e faz parte das bibliotecas do Android Jetpack, entre os principais objetivos encontramos:

- Facilidade de uso;
- Consistência através de vários dispositivos.

Serão apenas introduzidos aspectos iniciais desta API, mas a documentação pode ser consultada em <https://developer.android.com/training/camerax>

Câmara

Configuração [build.gradle (Project)]

```
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}
```

Verificar a inclusão dos repositórios da Google

Câmera

Configuração [build.gradle (Module)]

```
dependencies {  
  
    def camerax_version = '1.4.2'  
  
    implementation "androidx.camera:camera-core:$camerax_version"  
    implementation "androidx.camera:camera-camera2:$camerax_version"  
    implementation "androidx.camera:camera-lifecycle:$camerax_version"  
    implementation "androidx.camera:camera-video:$camerax_version"  
    implementation "androidx.camera:camera-view:$camerax_version"  
    implementation "androidx.camera:camera-extensions:$camerax_version"  
  
    ...  
}
```

Câmara

Permissões [AndroidManifest.xml]

```
<uses-permission android:name="android.permission.CAMERA" />  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Câmara

Permissões [MainActivity.java]

```
private static final int REQUEST_CAMERA_FEATURES = 200;  
  
private static final String[] REQUIRED_PERMISSIONS =  
    new String[]{  
        Manifest.permission.CAMERA,  
        Manifest.permission.WRITE_EXTERNAL_STORAGE  
   };
```

Lista de Permissões necessárias

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    if(!hasCameraPermission() || !hasStoragePremission()) {  
        requestPermissions();  
    }  
}
```

Se o utilizador não deu permissões de câmara ou de armazenamento externo, solicitamos essas permissões

Câmara

Permissões [MainActivity.java]

```
private boolean hasCameraPermission() {  
    return ContextCompat.checkSelfPermission(  
        this,  
        Manifest.permission.CAMERA  
    ) == PackageManager.PERMISSION_GRANTED;  
}  
  
private boolean hasStoragePremission() {  
    return ContextCompat.checkSelfPermission(  
        this,  
        Manifest.permission.WRITE_EXTERNAL_STORAGE  
    ) == PackageManager.PERMISSION_GRANTED;  
}  
  
private void requestPermissions(){  
    ActivityCompat.requestPermissions(this,  
        REQUIRED_PERMISSIONS, REQUEST_CAMERA_FEATURES);  
}
```

Verificação de autorização de utilização da câmara

Verificação de autorização de escrita em armazenamento externo

Câmara

Activity Camara [activity_camera.xml]

```
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".CameraActivity">  
  
    <FrameLayout  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:id="@+id/container">  
        <androidx.camera.view.PreviewView ←  
            android:id="@+id/preview_view"  
            android:layout_width="match_parent"  
            android:layout_height="match_parent" />  
    </FrameLayout>  
  
    <ImageButton  
        android:id="@+id/btn_picture"  
        android:layout_width="72dp"  
        android:layout_height="72dp"  
        android:layout_margin="24dp"  
        android:background="?android:attr/selectableItemBackground"  
        app:srcCompat="@android:drawable/ic_menu_camera"  
        app:layout_constraintBottom_toBottomOf="parent"  
        app:layout_constraintEnd_toEndOf="parent"  
        app:layout_constraintStart_toStartOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>
```

Elemento PreviewView que vai ser utilizado para mostrar a captura de imagem atual pela câmara

Câmera

[CameraActivity.java]

```
private PreviewView previewView;
private ImageButton btnPicture;
private ListenableFuture<ProcessCameraProvider> cameraProviderFuture;
private ImageCapture imageCapture;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_camera);

    previewView = findViewById(R.id.preview_view);
    btnPicture = findViewById(R.id.btn_picture);
    btnPicture.setOnClickListener(this::onClick);

    cameraProviderFuture = ProcessCameraProvider.getInstance(this);
    cameraProviderFuture.addListener((Runnable) this::run, ContextCompat.getMainExecutor(this));
}
```

Adição do listener, e
respetivo Executor

Câmera

[CameraActivity.java]

```
private void run() {  
    try {  
        ProcessCameraProvider cameraProvider = cameraProviderFuture.get();  
        bindPreview(cameraProvider);  
  
    } catch (ExecutionException | InterruptedException exception) {  
        exception.printStackTrace();  
    }  
}
```

Método a ser executado pelo
listener

Instanciação do
cameraProvider

Método para gerir o
preview da câmara

Câmera

[CameraActivity.java]

```
private void bindPreview(ProcessCameraProvider cameraProvider) {  
    Preview preview = new Preview.Builder()  
        .build();  
  
    CameraSelector cameraSelector = new CameraSelector.Builder()  
        .requireLensFacing(CameraSelector.LENS_FACING_BACK)  
        .build();  
  
    preview.setSurfaceProvider(previewView.getSurfaceProvider());  
    cameraProvider.bindToLifecycle((LifecycleOwner) this, cameraSelector, preview);  
}
```

Iniciar o preview da imagem

Selecionar a câmara traseira

Câmera

Captura de Imagem [CameraActivity.java]

```
private void bindPreview(ProcessCameraProvider cameraProvider) {
    ImageAnalysis imageAnalysis = new ImageAnalysis.Builder()
        .setBackpressureStrategy(ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST)
        .build();

    imageAnalysis.setAnalyzer(ContextCompat.getMainExecutor(this), image -> {
        image.close();
    });

    Preview preview = new Preview.Builder().build();

    CameraSelector cameraSelector = new CameraSelector.Builder()
        .requireLensFacing(CameraSelector.LENS_FACING_BACK)
        .build();

    ImageCapture imageCapture = new ImageCapture.Builder()
        .setTargetRotation(this.getDisplay().getRotation())
        .setFlashMode(ImageCapture.FLASH_MODE_AUTO)
        .setCaptureMode(ImageCapture.CAPTURE_MODE_MAXIMIZE_QUALITY)
        .build();

    preview.setSurfaceProvider(previewView.getSurfaceProvider());
    cameraProvider.bindToLifecycle((LifecycleOwner) this, cameraSelector, imageCapture, imageAnalysis, preview);
}
```

Câmera

Captura de Imagem [CameraActivity.java]

```
private void onClick(View view) {
    if(view == btnPicture){
        takePicture();
    }
}

private void takePicture() {
    File cameraOutputPath = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DCIM),
        "IMG_" + System.currentTimeMillis() + ".jpg");
    ↗ Obtenção do tempo atual em milissegundos para garantir nomes de ficheiros únicos
    ImageCapture.OutputFileOptions cameraOutput = new ImageCapture.OutputFileOptions.Builder(cameraOutputPath).build();

    imageCapture.takePicture(cameraOutput, ContextCompat.getMainExecutor(this), new ImageCapture.OnImageSavedCallback() {
        @Override
        public void onImageSaved(@NonNull ImageCapture.OutputFileResults outputFileResults) {
            Toast.makeText(getApplicationContext(), "Picture saved !! ", Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onError(@NonNull ImageCaptureException exception) {
        }
    });
}
```

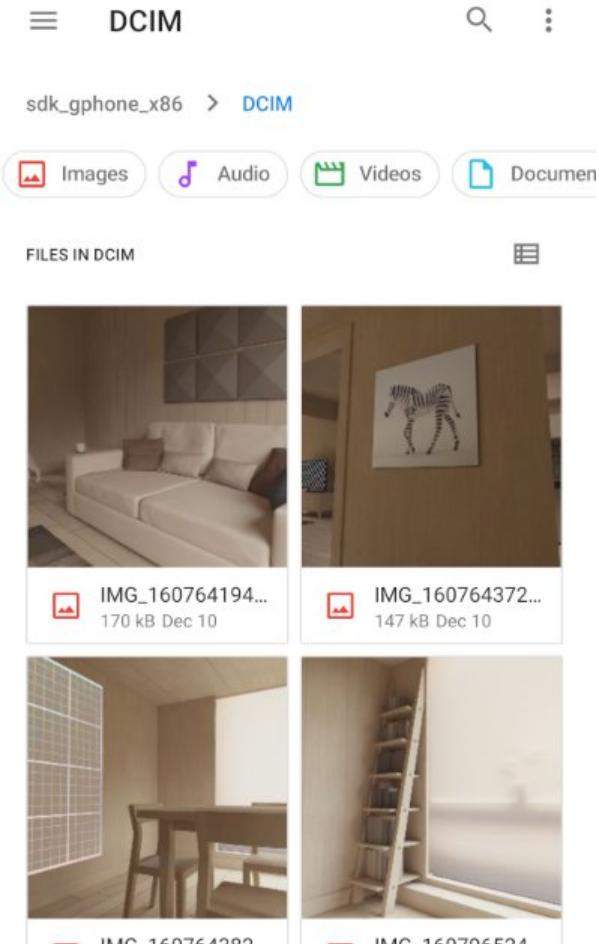
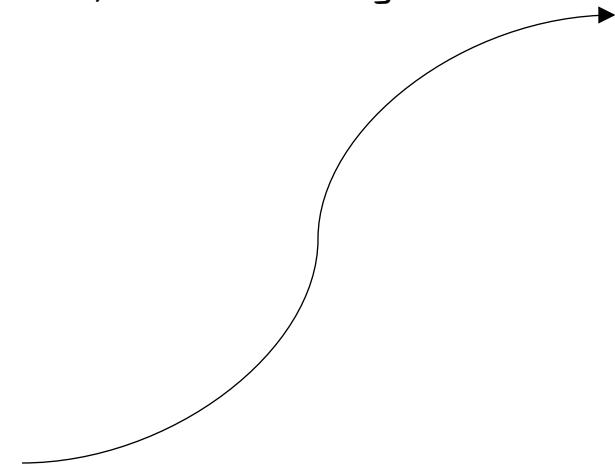
Path para guardar as fotografias tiradas

Câmara

Resultado



Captura de fotografia



Media

O Audio e Video, nomeadamente a sua captura e reprodução, pode ser feita através dos componentes Media para Android.

Entre outros, podemos encontrar nestes componentes

- MediaRecorder – gravação de vídeo e áudio;
- MediaPlayer – leitura de vídeo e áudio;

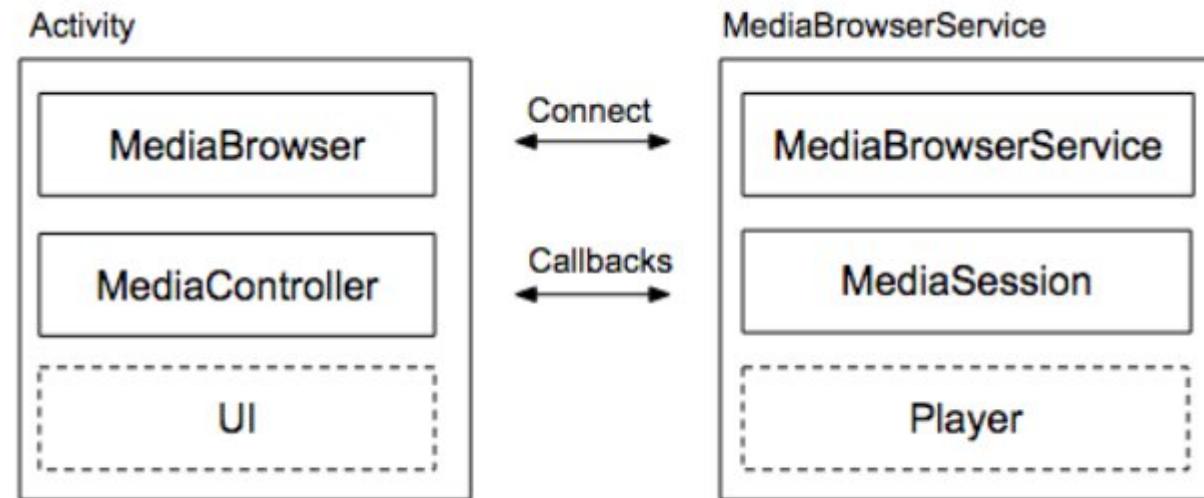
Os seguintes exemplos são focados na captura sonora, documentação e casos de uso adicionais podem ser encontrados em

<https://developer.android.com/guide/topics/media>

Media

Arquiteturas de Media Apps em Android

A gravação e leitura de Media é feita num serviço separado da Interface



Captura de Audio

Permissões [AndroidManifest.xml]

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Captura de Audio

Permissões [MainActivity.java]

```
private static final int REQUEST_AUDIO_RECORD = 300;  
  
private static final String[] REQUIRED_PERMISSIONS =  
    new String[]{  
        Manifest.permission.RECORD_AUDIO,  
        Manifest.permission.WRITE_EXTERNAL_STORAGE  
   };
```

Lista de Permissões necessárias

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    if(!hasRecordingPermission() || !hasStoragePermission()) {  
        requestPermissions();  
    }  
}
```

Se o utilizador não deu permissões de captura de áudio ou de armazenamento externo, solicitamos essas permissões

Captura de Audio

Permissões [MainActivity.java]

```
private boolean hasRecordingPermission() {
    return ContextCompat.checkSelfPermission(
        this,
        Manifest.permission.RECORD_AUDIO
    ) == PackageManager.PERMISSION_GRANTED;
}

private boolean hasStoragePermission() {
    return ContextCompat.checkSelfPermission(
        this,
        Manifest.permission.WRITE_EXTERNAL_STORAGE
    ) == PackageManager.PERMISSION_GRANTED;
}

private void requestPermissions(){
    ActivityCompat.requestPermissions(this,
        REQUIRED_PERMISSIONS, REQUEST_AUDIO_RECORD);
}
```

Verificação de autorização de utilização da captura de áudio

Verificação de autorização de escrita em armazenamento externo

Captura de Audio

[MainActivity.java]

```
private void startRecording() {  
  
    File recordingOutputPath = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DCIM),  
        "recording_" + System.currentTimeMillis() + ".3gp");  
  
    mediaRecorder = new MediaRecorder();  
    mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);  
    mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {  
        mediaRecorder.setOutputFile(recordingOutputPath);  
    }  
    mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);  
  
    try {  
        mediaRecorder.prepare();  
    } catch (IOException ioException) {  
        ioException.printStackTrace();  
    } finally {  
        mediaRecorder.start();  
    }  
}
```

Path para guardar as capturas de áudio

Obtenção do tempo atual em milissegundos para garantir nomes de ficheiros únicos

Capturar a partir do microfone

Captura de Audio

[MainActivity.java]

```
private void stopRecording(){  
  
    mediaRecorder.stop();  
    mediaRecorder.release();  
  
}
```

Tanto este método como o
startRecording estão
associados a botões

Reprodução de Audio

Permissões [AndroidManifest.xml]

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Reprodução de Audio

Permissões [MainActivity.java]

```
private static final int REQUEST_AUDIO_PLAYBACK = 400;  
  
private static final String[] REQUIRED_PERMISSIONS =  
    new String[]{  
        Manifest.permission.READ_EXTERNAL_STORAGE  
    };
```

Lista de Permissões necessárias

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    if(!hasPlaybackPermission() || ){  
        requestPermissions();  
    }  
}
```

Se o utilizador não deu permissões de leitura de armazenamento externo, solicitamos essas permissões

Reprodução de Audio

Permissões [MainActivity.java]

```
private boolean hasPlaybackPermission() {
    return ContextCompat.checkSelfPermission(
        this,
        Manifest.permission.READ_EXTERNAL_STORAGE
    ) == PackageManager.PERMISSION_GRANTED;
}

private void requestPermissions(){
    ActivityCompat.requestPermissions(this,
        REQUIRED_PERMISSIONS, REQUEST_AUDIO_RECORD);
}
```

Verificação de autorização
de leitura de
armazenamento externo

Reprodução de Audio

Serviço de Reprodução [MusicPlayService.java]

```
public class MusicPlayerService extends Service {  
  
    public static final String EXTRA_MUSIC = "song_name";  
    private MediaPlayer mediaPlayer;  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
    }  
  
    @Nullable  
    @Override  
    public IBinder onBind(Intent intent) {  
        return null;  
    }  
}
```

Reprodução de Audio

Serviço de Reprodução [MusicPlayService.java]

```
@Override  
public int onStartCommand(Intent intent, int flags, int startId) {  
    String resourcesPath = "android.resource://pt.ipp.estg.t_media/";  
    mediaPlayer = new MediaPlayer();  
  
    try {  
        mediaPlayer.setDataSource(getApplicationContext(),  
            Uri.parse(resourcesPath + intent.getIntExtra(this.EXTRA_MUSIC, 0)));  
        mediaPlayer.setLooping(false);  
        mediaPlayer.prepare();  
    } catch (IOException ioException) {  
        ioException.printStackTrace();  
    } finally {  
        mediaPlayer.start();  
    }  
    return START_NOT_STICKY;  
}
```

Acesso à pasta de resources. O Path deve ser substituído pelo path do projeto Android

Reprodução de Audio

Serviço de Reprodução [MusicPlayService.java]

```
@Override  
public void onDestroy() {  
    super.onDestroy();  
    mediaPlayer.release();  
}
```

Terminar a reprodução se o serviço for destruído.

Reprodução de Audio

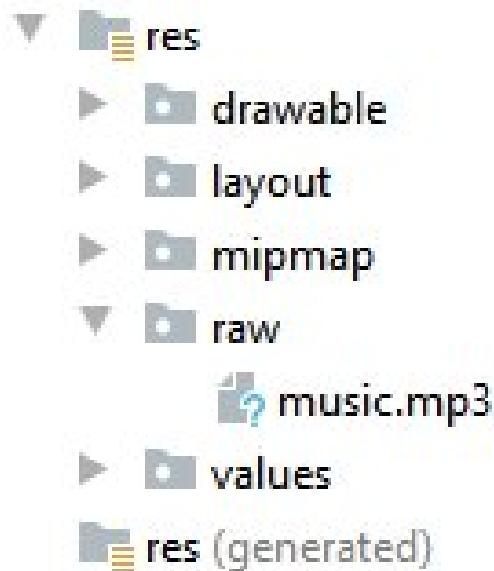
Reproduzir Música [MainActivity.java]

```
private void startPlayer() {  
    intent = new Intent(this, MusicPlayerService.class);  
  
    intent.putExtra(MusicPlayerService.EXTRA_MUSIC, R.raw.music);  
    startService(intent);  
}
```

Este procedimento está associado a um botão

Reprodução de Audio

Localização dos ficheiros de media



Para efeitos de teste,
podemos adicionar o
ficheiro que queremos
reproduzir na pasta raw
(deve ser criada)

Leitura Recomendada

CameraX:

<https://developer.android.com/training/camerax>

Media Recorder:

<https://developer.android.com/guide/topics/media/mediarecorder>

Programação Para Dispositivos Móveis I

CAMERA E MEDIA

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

SENSORES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Índice

- Sensores;
- Leitura Adicional.

Sensores

- Os sensores são (tipicamente) componentes físicos que permitem percecionar fenómenos externos a partir do dispositivo Android.
- São utilizados em aplicações que necessitam de contextos externos para funcionar, ou para adicionar recursos avançados (ex: localização ou movimento ao seu aplicativo);
- O seu uso é feito diretamente a partir de serviços de sistema sobre os quais podemos registar *callbacks* para serem notificados de eventuais alterações dos valores registados.

Sensores

A plataforma Android suporta várias categorias de sensores:

- **Movimento:** sensores que medem forças de aceleração, rotação e outros ao longo de três eixos (x,y,z). Nesta categoria incluem-se acelerómetros, sensores de gravidade, giroscópios, e sensores vetoriais de rotação.
- **Ambiente:** sensores que medem parâmetros ambientais, como temperatura, pressão do ar ambiente, iluminação e humidade. Esta categoria inclui barómetros, fotómetros e termómetros.
- **Posição:** sensores que medem a posição física de um dispositivo. Esta categoria inclui sensores de orientação e magnetómetros.

Sensores

Podemos aceder aos sensores disponíveis no dispositivo e adquirir dados em brutos usando a plataforma de sensores do Android. Fornece várias classes e interfaces que ajudam na execução de tarefas relacionadas com sensores:

- Determinar os sensores disponíveis no dispositivo;
- Determinar os recursos de um sensor individual:
 - Taxa de atualização;
 - Fabricante;
 - Requisitos de energia;
 - Resolução.
- Adquirir dados em brutos do sensor e definir a taxa de atualização mínima;
- Registrar e cancelar o registro de *listeners*;

Sensores

Sensor	Descrição	Usos comuns
<u>TYPE_ACCELEROMETER</u>	Mede a força de aceleração em m/s ² que é aplicada a um dispositivo nos três eixos físicos (x, y, e z), incluindo a força da gravidade.	Deteção de movimento (agitação, inclinação, etc.)
<u>TYPE_AMBIENT_TEMPERATURE</u>	Mede a temperatura ambiente em graus Celsius (°C).	Monitorização da temperatura do ar
<u>TYPE_GRAVITY</u>	Mede a força da gravidade em m/s ² que é aplicada a um dispositivo nos três eixos físicos (x, y, z).	Deteção de movimento (abanar, inclinar, etc.)
<u>TYPE_GYROSCOPE</u>	Mede a taxa de rotação de um dispositivo em rad/s em torno de cada um dos três eixos físicos (x, y, e z).	Deteção de rotação (girar, rodar, etc.)
<u>TYPE_LIGHT</u>	Mede o nível de luz ambiente (iluminação) em lx.	Controlar o brilho do ecrã
<u>TYPE_LINEAR_ACCELERATION</u>	Mede a força de aceleração em m/s ² que é aplicada a um dispositivo nos três eixos físicos (x, y, e z), excluindo a força da gravidade.	Monitorização da aceleração ao longo de um único eixo
<u>TYPE_MAGNETIC_FIELD</u>	Mede o campo geomagnético ambiente para os três eixos físicos (x, y, z) em µT.	Criação de uma bússola

Sensores

Sensor	Descrição	Usos comuns
<u>TYPE_ORIENTATION</u>	Mede os graus de rotação que um dispositivo faz em torno dos três eixos físicos (x, y, z). A partir do nível API 3 pode-se obter a matriz de inclinação e a matriz de rotação de um dispositivo usando o sensor de gravidade e o sensor de campo geomagnético em conjunto com o método <code>getRotationMatrix()</code> .	Determinação da posição do dispositivo.
<u>TYPE_PRESSURE</u>	Mede a pressão do ar ambiente em hPa ou mbar.	Monitorização de alterações de pressão de ar.
<u>TYPE_PROXIMITY</u>	Mede a proximidade de um objeto em cm em relação ao ecrã de visualização de um dispositivo. Este sensor é tipicamente utilizado para determinar se o dispositivo está a ser segurado junto ao ouvido de uma pessoa.	Posição do dispositivo durante uma chamada.
<u>TYPE_RELATIVE_HUMIDITY</u>	Mede a humidade ambiente relativa em percentagem (%).	Monitorização do ponto de orvalho, humidade absoluta e relativa.
<u>TYPE_ROTATION_VECTOR</u>	Mede a orientação de um dispositivo, fornecendo os três elementos do vetor de rotação do dispositivo.	Deteção de movimento e deteção de rotação.
<u>TYPE_TEMPERATURE</u>	Mede a temperatura do dispositivo em graus Celsius (°C). A implementação deste sensor varia entre dispositivos e este sensor foi substituído pelo sensor <code>TYPE_AMBIENT_TEMPERATURE</code> no nível API 14	Monitorização das temperaturas.

Sensores

▪ Sensor	▪ Android 4.0 (API Level 14)	▪ Android 2.3 (API Level 9)	▪ Android 2.2 (API Level 8)	▪ Android 1.5 (API Level 3)
▪ TYPE_ACCELEROMETER	▪ Yes	▪ Yes	▪ Yes	▪ Yes
▪ TYPE_AMBIENT_TEMPERATURE	▪ Yes	▪ n/a	▪ n/a	▪ n/a
▪ TYPE_GRAVITY	▪ Yes	▪ Yes	▪ n/a	▪ n/a
▪ TYPE_GYROSCOPE	▪ Yes	▪ Yes	▪ n/a	▪ n/a
▪ TYPE_LIGHT	▪ Yes	▪ Yes	▪ Yes	▪ Yes
▪ TYPE_LINEAR_ACCELERATION	▪ Yes	▪ Yes	▪ n/a	▪ n/a
▪ TYPE_MAGNETIC_FIELD	▪ Yes	▪ Yes	▪ Yes	▪ Yes
▪ TYPE_ORIENTATION	▪ Yes	▪ Yes	▪ Yes	▪ Yes
▪ TYPE_PRESSURE	▪ Yes	▪ Yes	▪ n/a	▪ n/a
▪ TYPE_PROXIMITY	▪ Yes	▪ Yes	▪ Yes	▪ Yes
▪ TYPE_RELATIVE_HUMIDITY	▪ Yes	▪ n/a	▪ n/a	▪ n/a
▪ TYPE_ROTATION_VECTOR	▪ Yes	▪ Yes	▪ n/a	▪ n/a
▪ TYPE_TEMPERATURE	▪ Yes	▪ Yes	▪ Yes	▪ Yes

Sensores em Android

Sensores de Ambiente

Para utilizar os sensores devemos registrar *listeners* para utilizar os dados fornecidos pela plataforma Android.

- Devemos garantir que os *listeners* são apenas utilizados durante as etapas relevantes do ciclo de vida dos componentes em Android (Activities, Fragments, ...);
- O uso criterioso da API SensorEventListener ajuda na conservação de energia dos equipamentos Android;

Sensores de Ambiente

Exemplo obtenção de um sensor

```
private SensorManager sensorManager;  
private Sensor sensor;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
    sensor = sensorManager.getDefaultSensor(Sensor.TYPE_AMBIENT_TEMPERATURE);  
}
```

Instanciação do sensor do tipo
Ambient_Temperature



Sensores de Ambiente

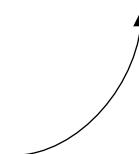
Exemplo obtenção da lista dos sensores disponíveis

```
private SensorManager sensorManager;
private List<Sensor> deviceSensors;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    deviceSensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
}
```

Lista de todos os sensores
disponíveis no dispositivo



Sensores de Ambiente

Exemplo (Sensor de Luz Ambiente) [MainActivity.java]

```
public class MainActivity extends AppCompatActivity implements SensorEventListener {  
    private SensorManager sensorManager;  
    private Sensor luxSensor;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
        luxSensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);  
    }  
}
```

Obter sensor de Luz Ambiente

Implementação do Listener de eventos

Sensores de Ambiente

Exemplo (Sensor de Luz Ambiente) [MainActivity.java]

```
@Override  
protected void onResume() {  
    super.onResume();  
    sensorManager.registerListener(this, luxSensor, SensorManager.SENSOR_DELAY_NORMAL);  
}
```

Registrar o Listener respeitando o ciclo
de vida da Activity

```
@Override  
protected void onPause() {  
    super.onPause();  
    sensorManager.unregisterListener(this);  
}
```

Remover o registo do listener

Sensores de Ambiente

Exemplo (Sensor de Luz Ambiente) [MainActivity.java]

```
@Override  
public void onSensorChanged(SensorEvent event) {  
    Log.d("DEVICE_SENSOR", String.valueOf(event.values[0]));  
}  
  
@Override  
public void onAccuracyChanged(Sensor sensor, int accuracy) {  
    //TODO: Implementar comportamento caso ocorra alguma alteração  
    //       no valor de accuracy do sensor  
}
```

Invocado cada vez que o valor registado pelo sensor sofre alteração

Obtenção do valor atual do sensor

Sensores em Android

Movimento

O sensor de movimento significativo aciona um evento cada vez que um movimento significativo é detetado e, em seguida, ele é desativado.

Uma moção significativa é uma moção que pode levar a uma alteração na localização do utilizador:

- Caminhar;
- Correr;
- Andar de bicicleta;
- Andar de carro;
- Etc.

Sensores de Movimento

Significant Motion Sensor [MainActivity.java]

```
private SensorManager sensorManager;
private Sensor sensor;
private TriggerEventListener triggerEventListener;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    sensor = sensorManager.getDefaultSensor(Sensor.TYPE_SIGNIFICANT_MOTION);

    triggerEventListener = new TriggerEventListener() {
        @Override
        public void onTrigger(TriggerEvent event) {
            //TODO: Implementar comportamento caso ocorra um evento de SIGNIFICANT_MOTION
        }
    };
    sensorManager.requestTriggerSensor(triggerEventListener, sensor);
}
```

Sensores de Movimento

Step Counter Sensor [MainActivity.java]

```
private SensorManager sensorManager;
private Sensor sensor;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    sensor = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER);
}
```

Este sensor tem um valor maior de latência (10 segundos), mas uma accuracy superior ao TYPE_STEP_DETECTOR

Nota: deve ser declarada a permissão ACTIVITY_RECOGNITION para que a aplicação use este sensor em dispositivos com Android 10 (API 29) ou superior.

Sensores em Android

Existem ainda em alguns dispositivos Android sensores especiais que requerem bibliotecas fornecidas diretamente a partir do fabricantes dos dispositivos.

- Exemplos de sensores não disponíveis na API Android:
 - Project Soli – Pixel 4 e Pixel XL;
- Sensores como o sensor de impressões digitais, requerem também permissões e usos especiais:
 - Sensibilidade dos dados previne o acesso aos dados em bruto;
 - Necessidade de proteger privacidade dos utilizadores.

Podemos no entanto usar os sensores biométricos em casos de uso previstos pelo Android como por exemplo autenticação pessoal

<https://developer.android.com/training/sign-in/biometric-auth>

Leitura Recomendada

Sensores:

https://developer.android.com/guide/topics/sensors/sensors_overview

Programação Para Dispositivos Móveis I

SENSORES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Programação Para Dispositivos Móveis I

FIREBASE

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Índice

- Firebase;
- Funcionalidades;
- Extensões;
- Adicionar Funcionalidades Firebase;
- Bibliotecas Firebase;
- Leitura Adicional.

Firebase

Conjunto de funcionalidades desenvolvidas pela Google que permite

- Criar aplicações de forma mais célere;
- Utilização de base de dados, análise de *performance*, entre outras.

Estas funcionalidades possuem compatibilidade com as plataformas:

- Android;
- iOS;
- Web;
- ...

Funcionalidades Firebase

Desenvolvimento



Criação de aplicações seguras e escalonáveis com recurso à infraestrutura de nível global.

- Cloud firestore;
- ML Kit;
- Cloud Functions;
- Authentication;
- Hosting;
- Cloud Storage;
- Realtime Database.

Funcionalidades Firebase

Qualidade



Insights sobre o desempenho e a estabilidade da aplicação.

- Crashlytics;
- Performance Monitoring;
- Test Lab;
- App Distribution.

Funcionalidades Firebase

Expansão



- In-App Messaging;
- Google Analytics;
- Predictions;
- A/B Testing;
- Cloud Messaging;
- Remote Config;
- Dynamic Links.

Funcionalidades Firebase

Extensões (BETA)

- Permite a adição de funcionalidades através de soluções pré-implementadas;
- São configuráveis e permitem integração com outras funcionalidades Firebase e produtos da Google Cloud Platform.



Edição de imagens



Tradução de texto



Sincronização de email



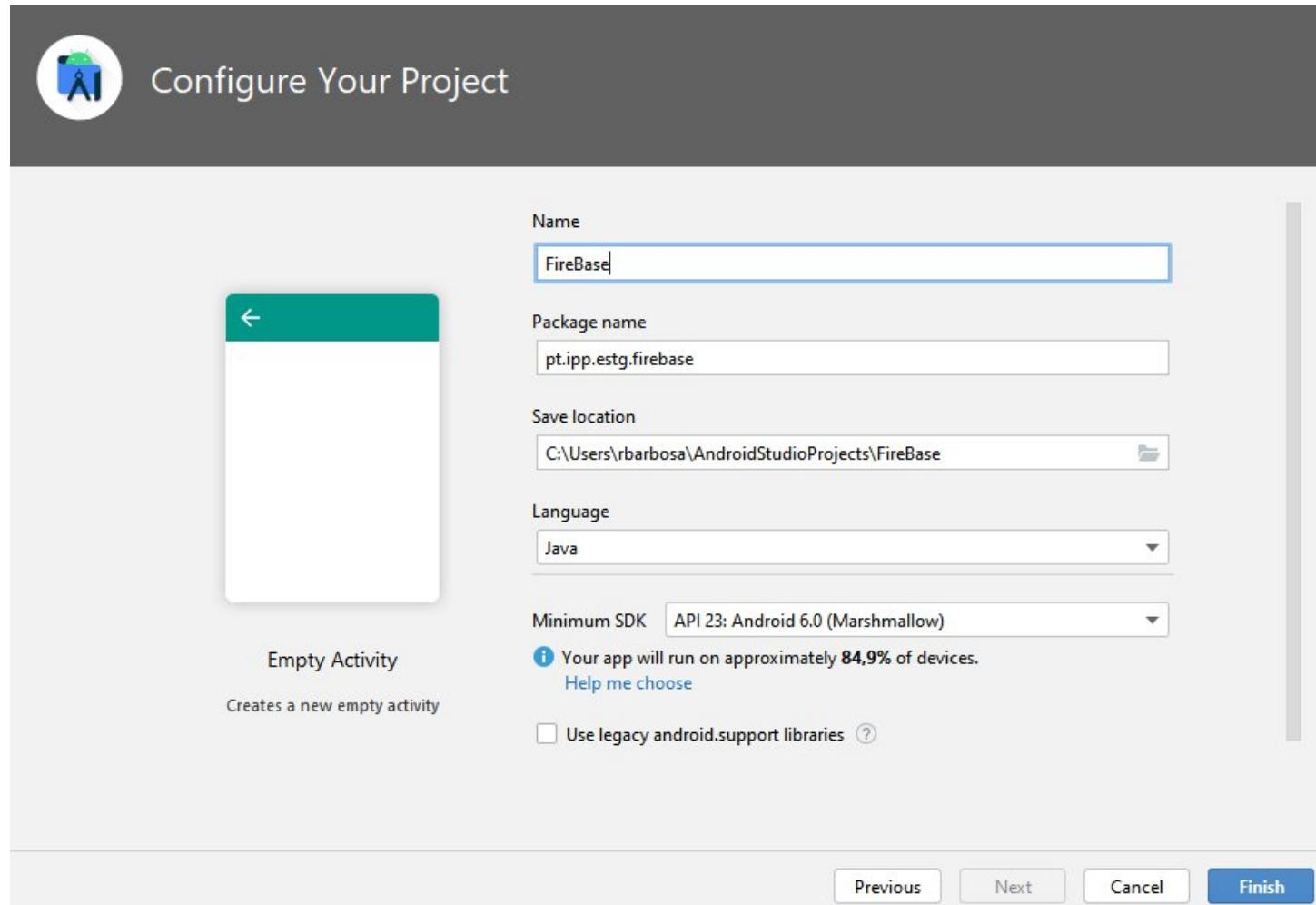
Triggers de email

Adicionar Funcionalidades Base

1. Criar um projeto (em branco) no Android Studio
2. Ir para a consola do Firebase (<https://console.firebaseio.google.com>)
 1. Criar um novo projeto Firebase;
 2. Atribuir nome ao projeto;
 3. Ativar Google Analytics (opcional, mas recomendado);
 4. Atribuir conta Google (de programador);
 5. Adicionar aplicação Android;
 1. Registar aplicação (copiar o applicationId do gradle – module app)
 2. Download do ficheiro de configurações em formato *json*
 3. Adicionar o ficheiro ao projeto Android
 4. Adicionar o SDK do Firebase

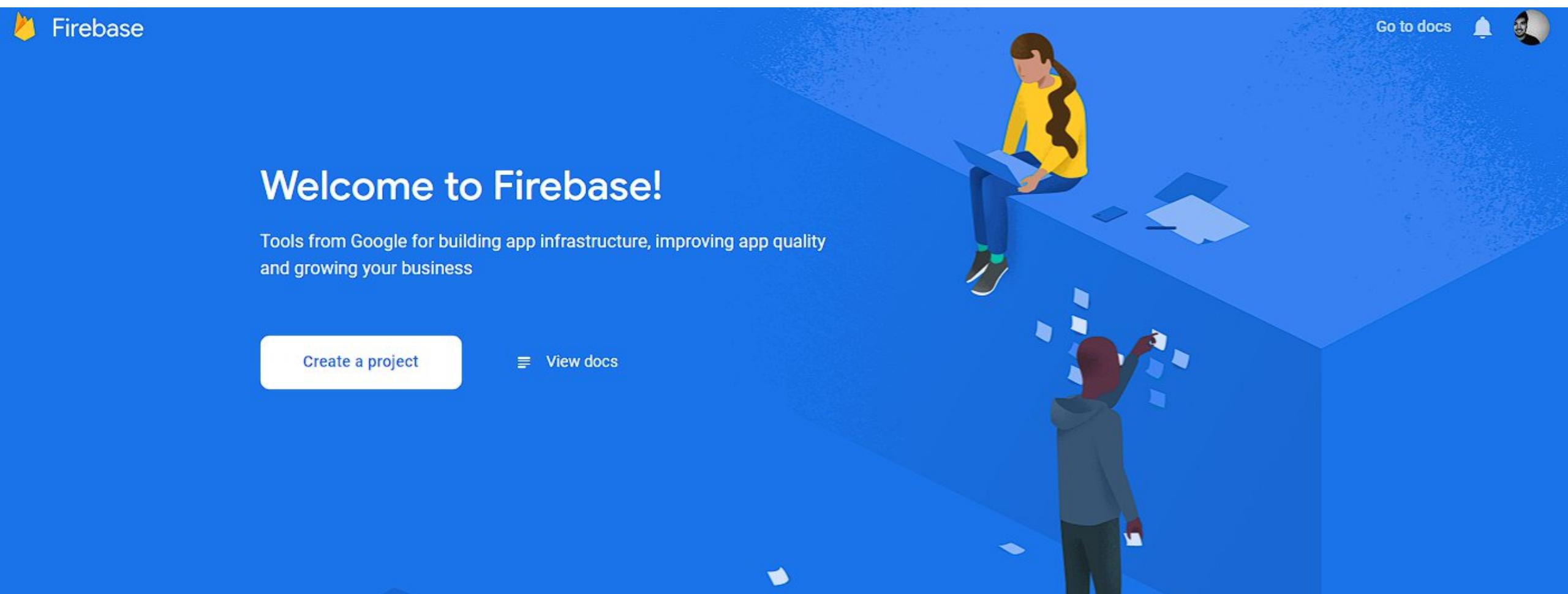
Adicionar Funcionalidades Base

1. Criar um projeto (em branco) no Android Studio



Adicionar Funcionalidades Base

2. Ir para a consola do Firebase (<https://console.firebaseio.google.com>)



Adicionar Funcionalidades Base

2.1. Criar um novo projeto Firebase

The screenshot shows the Firebase Welcome page. At the top left is the Firebase logo. On the right are links for "Go to docs", a bell icon, and a user profile. The main heading is "Welcome to Firebase!" followed by a subtext: "Tools from Google for building app infrastructure, improving app quality and growing your business". Below this are two buttons: "Create a project" and "View docs". A large blue arrow points from the "Create a project" button towards the bottom left. A callout bubble with the text "Criar novo projeto" is positioned near this arrow. To the right, there is a stylized illustration of two people: one sitting on a ledge working on a laptop, and another standing below them working on a wall covered in sticky notes.

Welcome to Firebase!

Tools from Google for building app infrastructure, improving app quality and growing your business

Create a project

View docs

Criar novo projeto

Adicionar Funcionalidades Base

2.2. Atribuir nome ao projeto

×

Create a project(Step 1 of 3)

Let's start with a name for your project®

Enter your project name

Project name is required

my-awesome-project-id

Continue

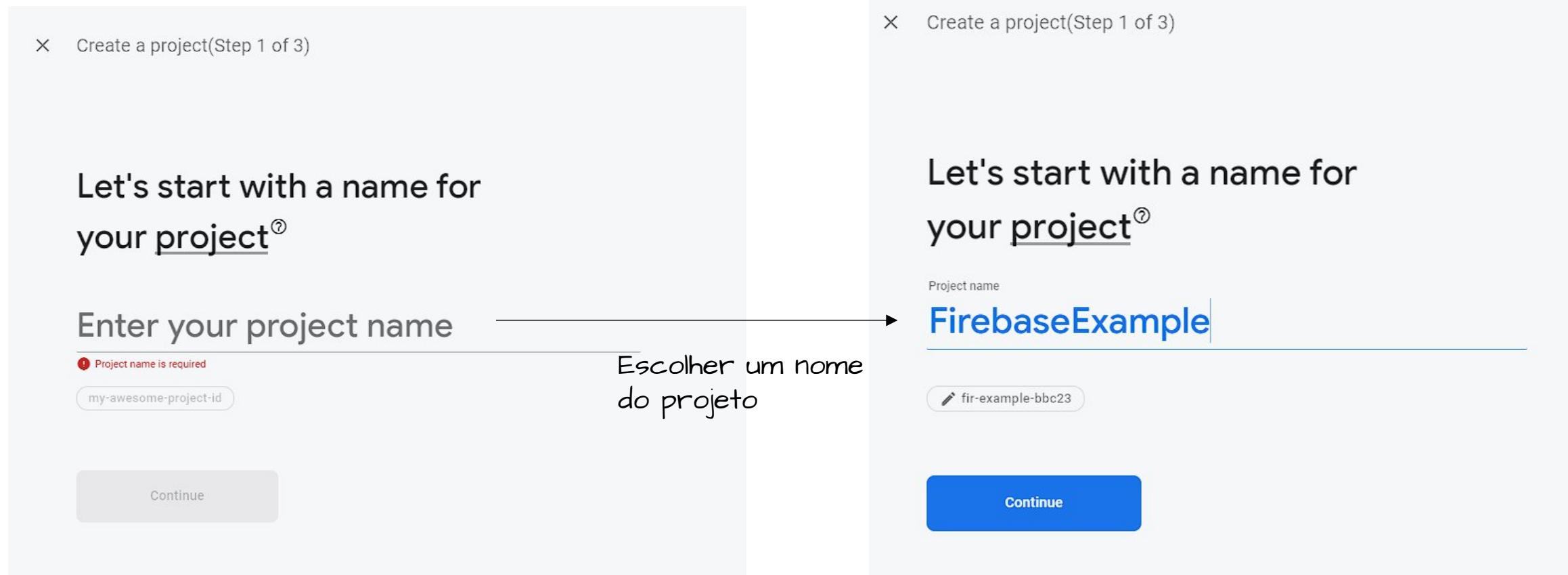
Escolher um nome do projeto

Project name

FirebaseExample

fir-example-bbc23

Continue



Adicionar Funcionalidades Base

2.3. Ativar Google Analytics (opcional, mas recomendado)

X Create a project(Step 2 of 3)

Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions and Cloud Functions.

Google Analytics enables:

- A/B testing ⓘ
- Crash-free users ⓘ
- User segmentation and targeting across Firebase products ⓘ
- Event-based Cloud Functions triggers ⓘ
- Predicting user behaviour ⓘ
- Free unlimited reporting ⓘ

Enable Google Analytics for this project
Recommended

[Previous](#) [Continue](#)



Adicionar Funcionalidades Base

2.4. Atribuir conta Google (de programador)

× Create a project(Step 3 of 3)

Configure Google Analytics

Choose or create a Google Analytics account ⓘ

Default Account for Firebase ▾

Automatically create a new property in this account ✎

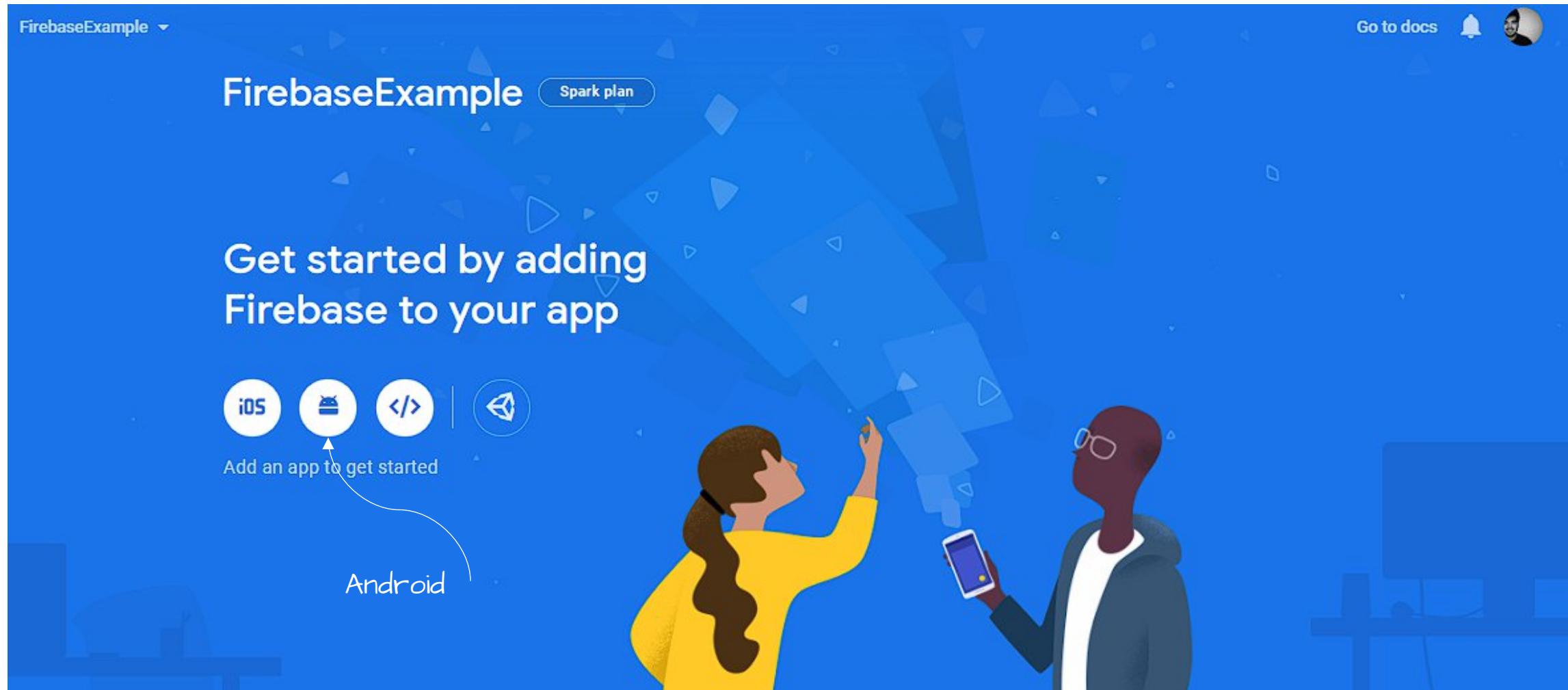
Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more](#)

[Previous](#) [Create project](#)



Adicionar Funcionalidades Base

2.5. Adicionar Aplicação Android



Adicionar Funcionalidades Base

2.5.1. Registar aplicação

× Add Firebase to your Android app

1 Register app

Android package name ?

com.android.application

App nickname (optional) ?

My Android App

Debug signing certificate SHA-1 (optional) ?

44:05:2E:14:A8:45:1A:6D:5E:2C:CC:94:2D:BC:23:49:0:

Required for Dynamic Links, Invites and Google Sign-In or phone-number support in Auth. Edit SHA-1s in Settings.

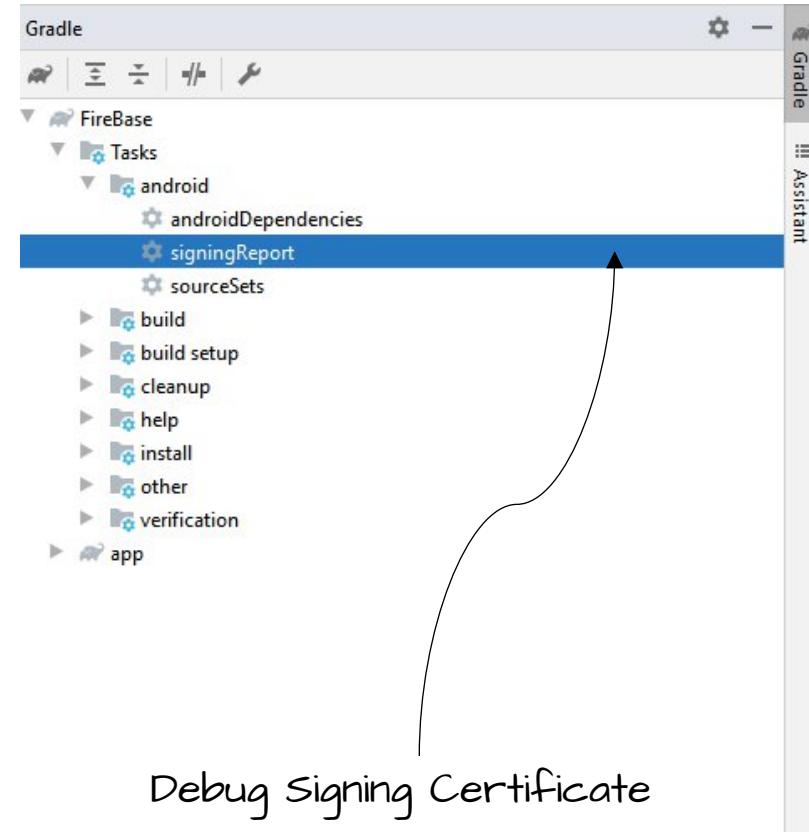
Adicionar Funcionalidades Base

2.5.1. Registar aplicação

```
plugins {  
    id 'com.android.application'  
}
```

Android Package Name

build.gradle(Module)



Debug Signing Certificate

Adicionar Funcionalidades Base

2.5.2. Download ficheiro de configurações

X Add Firebase to your Android app

1 Register app
Android package name: com.android.application

2 Download config file

Instructions for Android Studio below | [Unity](#) [C++](#)

Download do ficheiro

[Download google-services.json](#)

Switch to the Project view in Android Studio to see your project root directory.

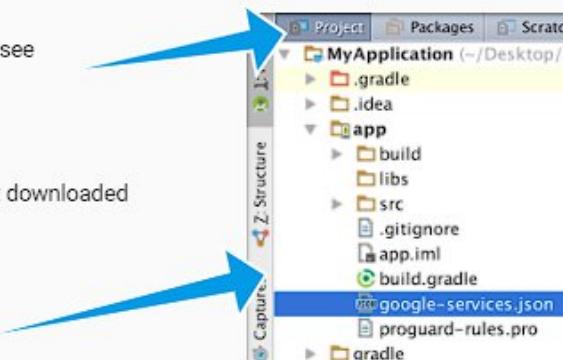
Move the google-services.json file that you just downloaded into your Android app module root directory.

google-services.json

Project Packages Scratches

MyApplication (~/Desktop/MyApplication)

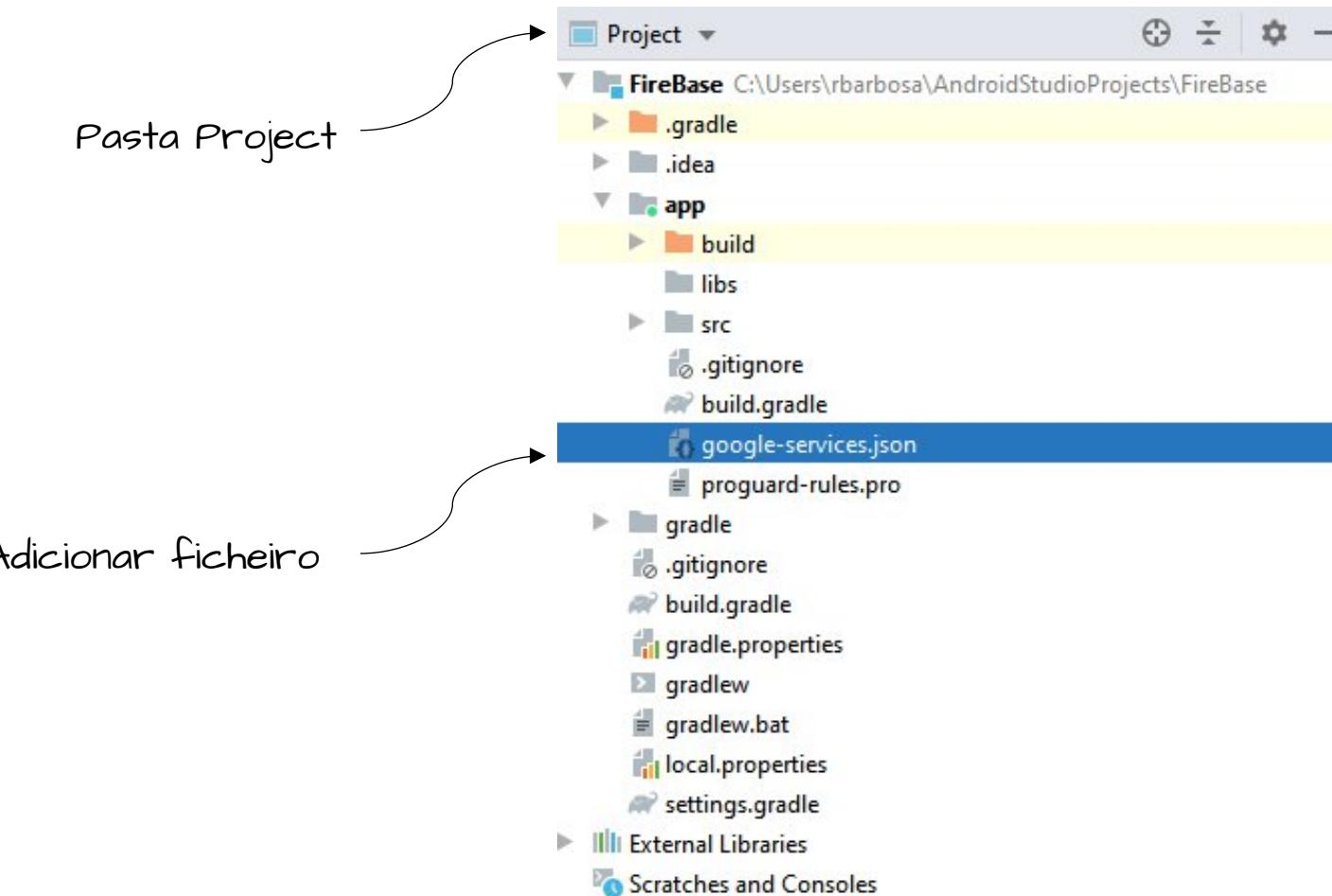
.gradle .idea app build libs src .gitignore app.iml build.gradle google-services.json proguard-rules.pro gradle



Previous [Next](#)

Adicionar Funcionalidades Base

2.5.3. Adicionar ficheiro ao projeto



Adicionar Funcionalidades Base

2.5.4. Adicionar o SDK do FireBase [build.gradle (Project)]

```
buildscript {  
    repositories {  
        // Check that you have the following line (if not, add it):  
        google() // Google's Maven repository  
    }  
    dependencies {  
        ...  
        // Add this line  
        classpath 'com.google.gms:google-services:4.4.2'  
    }  
}  
  
allprojects {  
    ...  
    repositories {  
        // Check that you have the following line (if not, add it):  
        google() // Google's Maven repository  
        ...  
    }  
}
```

Adicionar Funcionalidades Base

2.5.4. Adicionar o SDK do FireBase [build.gradle (Module)]

```
apply plugin: 'com.android.application'  
// Add this line  
apply plugin: 'com.google.gms.google-services'  
  
dependencies {  
    // Import the Firebase BoM  
    implementation platform('com.google.firebase:firebase-bom:33.13.0')  
  
    // Add the dependency for the Firebase SDK for Google Analytics  
    // When using the BoM, don't specify versions in Firebase dependencies  
    implementation 'com.google.firebase:firebase-analytics'  
  
    // Add the dependencies for any other desired Firebase products  
    // https://firebase.google.com/docs/android/setup#available-libraries  
}
```

Bibliotecas Firebase

Estão divididas em 3 categorias:

Play services required

Precisam dos serviços da Google Play ativos para funcionar

Play services recommended

Precisam dos serviços da Google Play ativos para funcionar na íntegra, mas oferecem grande parte das funcionalidades sem os serviços da Google Play

Play services not required

Não precisam dos serviços da Google Play ativos para funcionar

Google Play services not required

Product	Library	Google Play services?
App Check custom and debug providers	com.google.firebaseio:firebase-appcheck:18.0.0 com.google.firebaseio:firebase-appcheck-ktx:18.0.0 com.google.firebaseio:firebase-appcheck-debug:18.0.0	Not Required
App Distribution API	com.google.firebaseio:firebase-appdistribution-api:16.0.0-beta13 com.google.firebaseio:firebase-appdistribution-api-ktx:16.0.0-beta13	Not Required
App Distribution	com.google.firebaseio:firebase-appdistribution:16.0.0-beta13	Not Required
Authentication	com.google.firebaseio:firebase-auth:23.0.0 com.google.firebaseio:firebase-auth-ktx:23.0.0	Not Required
Cloud Firestore	com.google.firebaseio:firebase-firestore:25.0.0 com.google.firebaseio:firebase-firestore-ktx:25.0.0	Not Required
Cloud Functions for Firebase Client SDK	com.google.firebaseio:firebase-functions:21.0.0 com.google.firebaseio:firebase-functions-ktx:21.0.0	Not Required
Cloud Storage for Firebase	com.google.firebaseio:firebase-storage:21.0.0 com.google.firebaseio:firebase-storage-ktx:21.0.0	Not Required
Crashlytics	com.google.firebaseio:firebase-crashlytics:19.0.1 com.google.firebaseio:firebase-crashlytics-ktx:19.0.1	Not Required
In-App Messaging	com.google.firebaseio:firebase-inappmessaging:21.0.0 com.google.firebaseio:firebase-inappmessaging-ktx:21.0.0	Not Required
In-App Messaging Display	com.google.firebaseio:firebase-inappmessaging-display:21.0.0 com.google.firebaseio:firebase-inappmessaging-display-ktx:21.0.0	Not Required
Firebase installations	com.google.firebaseio:firebase-installations:18.0.0 com.google.firebaseio:firebase-installations-ktx:18.0.0	Not Required
Performance Monitoring	com.google.firebaseio:firebase-perf:21.0.1 com.google.firebaseio:firebase-perf-ktx:21.0.1	Not Required
Realtime Database	com.google.firebaseio:firebase-database:21.0.0 com.google.firebaseio:firebase-database-ktx:21.0.0	Not Required
Remote Config	com.google.firebaseio:firebase-config:22.0.0 com.google.firebaseio:firebase-config-ktx:22.0.0	Not Required
Vertex AI for Firebase	com.google.firebaseio:firebase-vertexai:16.0.0-beta01	Not Required

Bibliotecas Firebase

Google Play services required or recommended

Product	Library	Google Play services?
AdMob	com.google.android.gms:play-services-ads:23.1.0	Recommended*
Analytics	com.google.firebaseio:firebase-analytics:22.0.1 com.google.firebaseio:firebase-analytics-ktx:22.0.1	Recommended*
App Check Play Integrity provider	com.google.firebaseio:firebase-appcheck-playintegrity:18.0.0	Required
App Check SafetyNet provider	com.google.firebaseio:firebase-appcheck-safetynet:16.1.2	Required
App Indexing	com.google.firebaseio:firebase-appindexing:20.0.0	Required
Cloud Messaging	com.google.firebaseio:firebase-messaging:24.0.0 com.google.firebaseio:firebase-messaging-ktx:24.0.0	Required
Dynamic Links	com.google.firebaseio:firebase-dynamic-links:22.1.0 com.google.firebaseio:firebase-dynamic-links-ktx:22.1.0	Required
Firebase ML Vision	com.google.firebaseio:firebase-ml-vision:24.1.0	Required
Firebase ML Custom Model	com.google.firebaseio:firebase-ml-model-interpreter:22.0.4	Required

Leitura Recomendada

- Primeiros Passos:

<https://firebase.google.com/docs/android/setup?authuser=0>

- Firebase Console:

<https://console.firebaseio.google.com/u/0/>

Programação Para Dispositivos Móveis I

FIREBASE

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfp@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt