Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Programação de Dispositivos Móveis
Guia para Aula Laboratorial 6
Licenciatura em Engenharia Informática
Licenciatura em Informática Web

Programming of Mobile Devices
Guide for Laboratory Class 6
Degree in Computer Science and Engineering
Degree in Web Informatics

Sumário
Estudo do funcionamento dos filtros de intentos através de uma aplicação móvel capaz de lidar com a ação de partilha. Implementação de uma aplicação móvel simples que exemplifica a forma como se pode despoletar uma atividade com devolução de resultados. Introdução aos fornecedores de conteúdo e aos mecanismos de acesso a recursos protegidos de um sistema Android™.

Summary
Study of the functioning of intent filters via the development of an application capable of dealing with the share action. Implementation of a simple mobile application exemplifying how an activity can be started and return results to its caller. Introduction to content providers and to the mechanisms to access protected resourced in Android™.

Pré-requisitos:
Algumas das tarefas enunciadas a seguir requerem o acesso a um sistema com o Android Studio e com o SDK Android™, bem como com a Gradle™ instalados ou, alternativamente, com permissões para instalação e configuração do IDE, *kit* e ferramenta. Serão suficientes permissões para criar diretorias e ficheiros num disco local e para configurar variáveis de sistema, nomeadamente a *path*. É necessário ter acesso a uma versão e imagem da plataforma Android™ ou a um dispositivo físico com o sistema operativo e com a opção de *debug* ativa. É igualmente necessário ter um compilador Java instalado.

1 Preliminares
Preliminaries

O guia laboratorial 2 elabora nos passos necessários a criação e compilação (*build*) de projetos de aplicações para a plataforma Android™ via linha de comandos. Esta abordagem, apesar de não comportar algumas das facilidades oferecidas por ambientes de desenvolvimento integrados, nomeadamente ambientes de edição da interface de utilizador *What You See Is What You Get* (WYSIWYG), permite conhecer em maior profundidade os detalhes de implementação de uma aplicação Android™, mas requer que, após instalação do Android Studio, se atualize e instalem as várias ferramentas do *Software Development Kit* (SDK) Android™[1]. Depois do sistema estar devidamente configurado, 4 passos são suficientes para criar um projeto Android™, gerar o ficheiro `.apk` e instalar a aplicação num dispositivo (virtual ou real):

1. Inicializar o dispositivo móvel virtual ou ligar um real ao computador[2];
2. Gerar o projeto através do Android Studio;
3. Compilar o projeto com a ferramenta Gradle™, emitindo o comando `$ gradlew assembleDebug` na raiz do projeto;
4. Instalar a aplicação com um comando semelhante a `$ adb install -r path\NomeApp-debug.apk`.

[1] Ver `https://developer.android.com/studio/intro/update`.
[2] Se o dispositivo for real, tem de ter a opção de depuração ativada.

Tarefa 1 *Task 1*

Como já vem sendo habitual, a primeira tarefa consiste em iniciar um *Android Virtual Device* (AVD). Para isso, pode emitir o comando `$ emulator`, incluído na pasta `emulator` do SDK, e lançar um AVD. Caso não exista nenhum AVD configurado, crie um[3]. O ideal será um emulador de uma versão superior à 6.0 do Sistema Operativo (SO).

2 Filtros de Intentos
Intent Filters

Tarefa 2 *Task 2*

Depois de se certificar que tem um dispositivo virtual a

[3] O guia laboratorial 1 contém uma breve discussão acerca deste assunto.

correr e que tem uma plataforma alvo superior à 4.2 disponível, crie um novo projeto Android™ através do Android Studio™ com as seguintes especificações:

- Nome da aplicação — `exIntentFilters`;
- Domínio — `pmd.di.ubi.pt`;
- Sem suporte para C++ ou Kotlin;
- Deve ser um projeto para *smartphone* out *tablet*, mínimo API 21;
- Com uma `Empty Activity` chamada `ToLog`;
- Peça para gerar o ficheiro de *layout* (o nome do ficheiro de *layout* deve ser `activity_tolog.xml`);
- Retire qualquer suporte de retrocompatibilidade.

Note que os nomes sugeridos antes devem ser seguidos com rigor, já que deles depende, por vezes, o funcionamento bem sucedido da aplicação a ser desenvolvida.

Tarefa 3 *Task 3*

Procure a forma de especificar que a atividade que criou no âmbito do projeto anterior é capaz de lidar com a ação de partilha (i.e., `ACTION.SEND`). Use as seguintes questões para o guiar no processo.

Q1.: Onde (em que ficheiro) é que se criam os filtros de intentos?

☐ Em `res\layout\activity_tolog.xml`
☑ No `AndroidManifest.xml`
☐ No código Java da atividade respetiva.
☐ Em ficheiros de sistema.
☐ Em todas as outras aplicações que querem fazer partilha.

Q2.: Dentro de que elemento é que deve ser colocado o filtro de intentos no ficheiro XML?

☐ Dentro do elemento `manifest`.
☐ Dentro do elemento `linear_layout`.
☑ Dentro do elemento `activity`.
☐ Dentro do elemento `caotic_layout`.
☐ Dentro do elemento `activity`, mas fora do elemento `manifest`.
☐ Em tudo quanto é lado!

Q3.: Qual dos seguintes excertos de código XML define corretamente o filtro de intentos?

☑
```
<intent-filter>
 <action android:name="android.intent.action.SEND"/>
 <category android:name="android.intent.category.DEFAULT"/>
 <data android:mimeType="text/plain"/>
</intent-filter>
```

☐
```
<intent-filter>
 <action android:name="android.intent.action.VIEW"/>
 <category android:name="android.intent.category.DEFAULT"/>
 <data android:mimeType="text/plain"/>
</intent-filter>
```

☐
```
<intent-filter>
 <action android:name="android.intent.action.DIAL"/>
 <category android:name="android.intent.category.DEFAULT"/>
 <data android:mimeType="audio/*"/>
</intent-filter>
```

☐
```
<intent-filter>
 <action android:name="android.intent.action.EDIT"/>
 <category android:name="android.intent.category.DEFAULT"/>
 <data android:mimeType="image/png"/>
</intent-filter>
```

Depois de se certificar que respondeu corretamente às questões anteriores, faça as alterações necessárias no ficheiro correto. Compile, instale e teste a sua aplicação.

Tarefa 4 *Task 4*

Repare que, quando é despoletada uma intenção de partilha, é comum transportar dados para a atividade invocada através de `putExtra(Intent.EXTRA_TEXT,string)`. Abra o ficheiro com o código fonte Java para a única atividade da aplicação. Adicione as instruções que achar necessárias para que esta atividade seja capaz de escrever no `logcat` (`Log.i()`) o conteúdo que é enviado por outras aplicações no *Extra* `Intent.EXTRA_TEXT`. A `TAG` a ser mostrada no *log* deve ser `INTFILT`.

Q4.: Acha que vai precisar de alguns dos *imports* seguintes?

☐ `import android.util.Log;`
☑ `import android.content.Intent;`
☑ Vou, sim senhor.
☐ Não preciso de nenhum destes *imports*.

Tarefa 5 *Task 5*

Compile, instale e teste a aplicação. Neste caso, para a testar, vai precisar de:

1. Encontrar uma aplicação que permita partilhar conteúdo (e.g., o *browser*);
2. Arranjar forma de observar o `logcat` (o que, a esta altura do campeonato nacional, já não deve ser problema).

Q5.: Das opções seguintes, quais concretizam formas de ver o `logcat`?

☐ `$ adb logcat`
☐ `$ android logcat`
☐ Android™ monitor.
☐ Aplicação *logcat* no emulador.

☐ Emitir o comando `$ adb shell` seguido de `$ logcat`.

Sugestão: para testar a aplicação, abra o *browser* no emulador e navegue até um site. Depois, carregue no botão do canto superior direito, e escolha `Share Page`.
Q6.: A sua aplicação aparece na caixa de diálogo de seleção seguinte?
☑ Pois aparece, que engraçado.
☐ Ehh... não?!

Certifique-se que o *output* que definiu na atividade principal aparece, de facto, no *logcat*.

3 Intentos com o Retorno

Intents Returning Results

Anteriormente, foi explorado como se podiam usar intentos para transportar dados entre componentes de uma aplicação Android™, no sentido da que define o intento para a que o recebe. Esta parte do guia laboratorial foca-se no fluxo inverso da comunicação, e na definição, à cabeça, de que a componente (uma atividade) invocada através de um intento deve devolver um resultado.

Tarefa 6 *Task 6*

Comece pela criação, compilação, instalação e teste de um novo projeto Android™ através do Android Studio™ com as seguintes especificações:

- Nome da aplicação — `exwithresult`;
- Domínio — `pmd.di.ubi.pt`;
- Sem suporte para C++ ou Kotlin;
- Deve ser um projeto para *smartphone* out *tablet*, mínimo API 21;
- Com uma `Empty Activity` chamada `MainActivity`;
- Peça para gerar o ficheiro de *layout* (o nome do ficheiro de *layout* deve ser `activity_main_activity.xml`);
- Retire qualquer suporte de retrocompatibilidade.

Note que os nomes sugeridos antes devem ser seguidos com rigor, já que deles depende, por vezes, o funcionamento bem sucedido da aplicação a ser desenvolvida.

Tarefa 7 *Task 7*

A atividade principal deve ter duas caixas de texto, um botão e uma etiqueta de texto em baixo:

- As **caixas de texto** servem para colocar o primeiro e o último nome de um utilizador.
- O **botão** deve dizer `Concat Names!`, e redirecionar para uma nova atividade, cuja única função será concatenar os dois nomes e devolver esse resultado para a primeira atividade.
- A **etiqueta de texto** em baixo deve estar vazia ao início, mas mais tarde deve conter a concatenação dos dois nomes colocados nas caixas de texto, após esta operação ser feita pela segunda atividade.

Q7.: Qual o método que permite que a atividade principal chame a segunda de forma a que esta lhe devolva um resultado?

- ☐ `beginActivity(Intent,int);`
- ☐ `startActivity(Intent,int);`
- ☐ `beginActivityForResult(Intent,int);`
- ☐ `devolveUMresultadoPoça(Intent,int);`
- ☑ `startActivityForResult(Intent,int);`

Q8.: Qual a função do segundo parâmetro (um inteiro (`int`)) referido na questão anterior?

- ☐ Este segundo parâmetro serve para transportar erros da componente que invoca para a que é invocada.
- ☑ Este segundo parâmetro serve para transportar erros da componente que é invocada para a que invoca.
- ☐ Este segundo parâmetro não serve para nada. É ridículo.
- ☐ Este segundo parâmetro deve ser igual a 1 caso se esteja à espera de retorno, e 0 no caso contrário.
- ☑ Este segundo parâmetro serve para identificar um pedido de retorno em particular. I.e., se forem lançados vários intentos com pedido de retorno, é este inteiro que permite identificar uma resposta específica.

Considere o seguinte excerto de código para o conteúdo do seu ficheiro `activity_main.xml` (**preencha** corretamente os espaços marcados):

```
<TextView
    PREENCHER
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enter your first and last name:"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<EditText
    android:id="@+id/fName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="First Name"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@id/
        explanation" />

<EditText
    android:id="@+id/lName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Last name"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@id/fName" />

<Button
    android:id="@+id/PREENCHER"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Concat Names!"
    android:onClick="goToSecond"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@id/lName" />

<TextView
    android:id="@+id/concat"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Empty at first..."
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    PREENCHER="@id/button" />
```

Tarefa 8 *Task 8*

Combine toda a informação reunida na secção anterior, e codifique o método `onCreate(Bundle)` de modo a que o botão redirecione para a atividade `ProcessNames.class` quando o botão `Concat Names!` é clicado. Não se esqueça de enviar, no intento respetivo, as duas *strings* com o conteúdo das caixas de texto, em variáveis chamadas `name1` e `name2`.

Tarefa 9 *Task 9*

Crie uma segunda atividade com nome `ProcessNames`. Esta atividade deve ter apenas um botão que diz `Done. Go Back!`. Ao ser clicado, o botão deve forçar a que a aplicação volte para a primeira atividade, devolvendo já a concatenação dos dois nomes numa string chamada `concatNames`.

Não se esqueça de declarar a nova atividade no `AndroidManifest.xml`, para que esta seja visível para o sistema. O ficheiro de *layout* desta nova atividade terá apenas um botão, cuja definição deve ser semelhante à seguinte:

```
<Button
  android:id="@+id/button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Go Back!"
  android:onClick="goBack"
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintRight_toRightOf="parent"
  app:layout_constraintTop_toTopOf="parent"
  app:layout_constraintBottom_toBottomOf="parent"/>
```

Já agora, e também para ajudar, a função que trata o evento de clique no botão `Done. Go Back!` é o seguinte:

```
this.finish();
```

Tarefa 10 *Task 10*

Note que deve fazer o processamento dos dois nomes no método `onCreate(Bundle)`, sendo que apenas no método `finish()` é que deverá instanciar o intento que devolve o resultado para a atividade principal. O conteúdo do método `onCreate(Bundle)` será semelhante a:

```
public String sConcat;
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main2);
  Intent iCameFromMain = getIntent();
  sConcat = iCameFromMain.getStringExtra("name1") + " " +
      iCameFromMain.getStringExtra("name2");
}
```

Enquanto que o método `finish()` será parecido com:

```
@Override
public void finish(){
  Intent iResult = new Intent();
  iResult.putExtra("concatNames",sConcat);
  setResult(RESULT_OK,iResult);
  super.finish();
}
```

Q9.: Qual é a função que ajusta o resultado de novo para a primeira atividade?

☐ `super.finish();`
☑ `setResult(RESULT_OK,iResult);`
☐ `iResult.putExtra("concatNames",sConcat);`
☐ `Intent iResult = new Intent();`

Q10.: O resultado é enviado no mesmo intento que despoletou a segunda atividade?

☐ Sim, é.
☑ Não, é enviado num novo intento.
☐ Não, é enviado através de um canal de comunicação criado para o efeito.

Tarefa 11 *Task 11*

Esta parte do guia sugeriu um fluxo de implementação que começava pela atividade principal, que coleciona dois nomes, e avançava para a segunda atividade, que processa esses nomes. A ideia é retornar à atividade principal e exibir a concatenação dos nomes na etiqueta de texto ao fundo. Para isso, precisa implementar um último método na `MainActivity.java`. **Q11.: Qual é o método que é automaticamente despoletado quando uma atividade retorna um valor àquela que a invocou?**

☐ `onResume();`
☐ `onActivity(int, int, Intent);`
☐ `onResult(int, int, Intent);`
☑ `onActivityResult(int, int, Intent);`
☐ `onResume(int, int, Intent);`

O método acima mencionado deve ficar com um conteúdo semelhante ao seguinte (note que falta especificar corretamente o nome e os parâmetros do método):

```
@Override
protected void method(int, int, Intent){
 if( ( iReqCode == iRequest_code ) &&
     ( iResultCode == RESULT_OK  ) ){

   TextView oVT = (TextView) findViewById(R.id.concat);
   oVT.setText( iResult.getStringExtra("concatNames") );

  }
}
```

Q12.: Qual o objetivo da primeira condição do `if`?

☐ O objetivo é verificar se o resultado foi efetivamente devolvido ou se houve algum erro no processo.
☑ O objetivo é verificar se o intento que foi devolvido diz respeito ao que foi despoletado.
☐ O objetivo é verificar se o resultado está de acordo com o que a atividade estava à espera.
☐ O objetivo é verificar se a segunda condição verifica o que a primeira faz e vice-versa.

Tarefa 12 *Task 12*

Compile, instale e teste a aplicação tantas vezes quantas forem necessárias para conseguir os objetivos desta parte do guia.

Q13.: Ao todo, quantos ficheiros precisou de editar para conseguir a aplicação desejada?

☐ 0. ☐ 1. ☐ 2. ☐ 3. ☐ 4. ☑ 5.
☐ O rácio dourado.

4 Pedir Permissões no Manifesto

Asking for Permissions in the Manifest

O próximo desafio consiste na implementação de uma aplicação, com uma única atividade, que mostre informação da última chamada de voz feita a partir do dispositivo móvel emulado (ou real).

Tarefa 13 *Task 13*

Comece por aceder à aplicação de chamadas do emulador ou dispositivo real. Verifique o registo de chamadas e, caso este esteja vazio, faça uma chamada para o número 123456789. No final, verifique se o registo de chamadas já contém entradas. Eventualmente, pode tentar colocar mais entradas no registo, nomeadamente mediante a abertura de um segundo emulador, e do estabelecimento de chamadas entre os dois. Uma outra alternativa consistem em fazer `telnet` para o emulador alvo (ver aula 2).

Q14.: Como é que se pode fazer uma chamada de um emulador para outro[4]?

☐ Abrindo a aplicação de chamadas num dos emuladores, digitando o número da instância do segundo emulador (dada por `$ adb devices`) e executando as chamadas.
☐ Telefonando para a assistência da Google, dizendo que está a testar uma aplicação e que precisa que ponham o seu emulador a tocar.
☐ Marcando o número 127000001 (que corresponde ao endereço IP do *localhost*).

Uma terceira opção para simular chamadas concretiza-se pela utilização do programa `monitor` fornecido com o SDK. **Q15.: Como pode executar este programa?**

[4] É assumido aqui que ambos os emuladores são fornecidos com o SDK.

Q16.: Qual o nome do separador que lhe permite aceder a esta funcionalidade?

☐ *Emulator Control* ☐ *System Information*
☐ *Statistics* ☐ Separador Amarelo
☐ *Heap* ☐ DDMS.

Tarefa 14 *Task 14*

Crie um novo projeto Android™ com nome `exPermissions` e com uma só atividade chamada `LastCall`. O nome do pacote deste projeto deve ser `pt.ubi.di.pmd.exPermissions` e a respetiva raiz deve ficar em `C:\Users\aluno\workspace\exPermissions`

Tarefa 15 *Task 15*

Compile, instale e teste o projeto que criou só para se certificar de que está tudo bem até esta parte.

Tarefa 16 *Task 16*

Mude o ficheiro XML que define o *layout* da atividade principal de forma a que esta passe a exibir uma etiqueta de texto no centro do ecrã, tanto na vertical como na horizontal. Especifique um identificador (`id`) a esta etiqueta. O `id` deve ser `lc`.

Q17.: Das seguintes, qual concretiza a linha que define corretamente o id da etiqueta de texto?

☐
```
android:id="@id/lc"
```

☑
```
android:id="@+id/lc"
```

☐
```
android:id="lc"
```

Tarefa 17 *Task 17*

Altere o código da atividade principal de maneira a que a etiqueta de texto definida antes mostre todas as informações disponíveis no registo do sistema para a última chamada efetuada ou recebida.

Para esta tarefa, vai precisar de um componente Android™ conhecido como provedor de conteúdos. Estes componentes serão objeto de estudo adiante. Contudo, para já, fica a ideia de que estes componentes permitem a partilha de informação entre várias aplicações, nomeadamente as de sistema. Estes provedores de conteúdos são tipicamente acedidos através da especificação de Uniform Resource Identifiers (URIs). Procure *online* qual é o URI que deverá permitir aceder ao registo de chamadas do dispositivo móvel e escreva-o a seguir:

_____

Ordene (i.e., coloque números nos espaços respetivos) os passos/instruções seguintes de forma a que reflitam o seu raciocínio/abordagem para conseguir o efeito desejado:

___ Cristalizar o *layout* através do método

```
setContentView(R.layout.main);
```

___ Instanciar a etiqueta de texto com uma instrução parecida com

```
TextView wTV = (TextView)
    findViewById(...);
```

___ Instanciar um objeto `Uri` com o URI do provedor de conteúdo relativo ao registo de chamadas. A instrução é semelhante a

```
Uri uriCalls = Uri.parse(...)
```

___ Instanciar um objeto `Cursor` que permite navegar pela informação disponível na base de dados acedida através de uma instrução semelhante a

```
Cursor curCalls = getContentResolver
    ().query(uriCallLog,null,null,
    null,null);
```

___ Declarar uma string com o texto `"Last Call: "`

```
String sInfo = "Last Call: ";
```

___ Movendo o cursor para a última linha dos dados, e mostrar o conteúdo de alguns campos da linha para onde está a apontar, através de um conjunto de instruções parecidas com as seguintes:

```
if(curCalls.moveToLast())
  sInfo += curCalls.getString(curCalls.
      getColumnIndex(CallLog.Calls.NUMBER))
  + "\nDuration: " + curCalls.getString(
      curCalls.getColumnIndex(CallLog.Calls
      .DURATION));
oTV.setText(sInfo);
```

Q18.: Quais dos seguintes pacotes interessa importar para a atividade principal desta aplicação?

☐ `android.app.Activity;`
☐ `android.os.Bundle;`
☐ `android.widget.TextView;`
☐ `android.net.Uri;`
☐ `android.content.intent`
☐ `android.database.Cursor;`
☐ `android.provider.CallLog;`

Tarefa 18 *Task 18*

Depois de se assegurar que o código está minimamente completo, compile, instale e teste a aplicação. **Q19.: Funcionou?**

☐ O Professor já ouviu a expressão *às mil maravilhas?*
☐ Não funcionou.

Q20.: O registo de chamadas é considerado como um recurso a proteger no sistema Android™?

☐ Pelos vistos não. ☐ Pelos vistos sim.

Q21.: De que forma é que o sistema responde a pedidos a recursos protegidos?

☐ Dando acesso imediato e total.
☐ Dando acesso parcial.
☐ Negando todo o acesso, a não ser que a permissão para aceder ao recurso tenha sido declarada no `AndroidManifest.xml`.
☐ Negando todo o acesso, a não ser que a permissão para aceder ao recurso tenha sido declarada no `AndroidManifest.xml`, e o utilizador tenha especificamente dado essa permissão aquando da instalação da aplicação.

Tarefa 19 *Task 19*

Adicione o elemento incluído a seguir no `AndroidManifest.xml` como filho direto de `manifest`:

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

Volte a preparar e a instalar a aplicação. **Q22.: Desta vez já funcionou?**

☐ Tudo nos trinques. ☐ Ainda não. *Reset!*

Q23.: A aceitação ou rejeição da permissão acima indicada foi-lhe colocada aquando da instalação da aplicação?

☐ Sim, foi.
☐ Não, não foi. Algo deve estar mal.
☑ Quando instalo uma aplicação via `$ adb`, as permissões são dadas por omissão desde que declaradas no manifesto! Afinal, estou em fase de desenvolvimento e depuração, caramba!

Tarefa 20 *Task 20*

Esta é só para os(as) mais corajosos(as). Simular a instalação real de uma aplicação Android™, de forma a que esta peça as permissões, num dispositivo virtual fornecido com o SDK pode não ser simples. Talvez seja possível depois de:

1. Definir um *sdcard* no emulador;
2. Usar o `adb` para transferir o ficheiro para o emulador;
3. Instalar uma aplicação gestora de ficheiros no emulador;
4. Executar o instalador, clicando diretamente no ficheiro `.apk`.

A sua missão, caso decida aceitá-la, é simular a instalação da aplicação com o pedido explicito de permissões.