Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Programação de Dispositivos Móveis
Guia para Aula Laboratorial 9
Licenciatura em Engenharia Informática
Licenciatura em Informática Web

Programming of Mobile Devices
Guide for Laboratory Class 9
Degree in Computer Science and Engineering
Degree in Web Informatics

Sumário
Desenvolvimento de uma pequena aplicação móvel que utiliza 3 componentes disponíveis na plataforma Android™ para exercitar os conceitos associados aos Serviços e Recetores de Difusão desta plataforma, bem como dos conceitos e características das *Threads*.

Summary
Development of a small mobile application that uses 3 components available in the Android™ platform with the objective of practicing concepts related with Services and Broadcast Receivers, as well as the concepts and characteristics of Threads.

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).

Tarefa 2 *Task 2*

Verifique se as variáveis `ANDROID_HOME` e `PATH` estão devidamente definidas com comandos parecidos com:

```
$ echo %ANDROID_HOME%
$ echo %JAVA_HOME%
$ echo %PATH%
```

Caso as variáveis já estejam devidamente definidas, passe para a secção seguinte. Caso contrário, precisa de as definir antes de avançar, com comandos semelhantes a:

```
$ set JAVA_HOME=RAIZ do JAVA JDK
```

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

```
$ set ANDROID_HOME=C:\installation location\sdk
$ set PATH=%PATH%; %JAVA_HOME%;
%ANDROID_HOME%\tools; %ANDROID_HOME%\emulator;
%ANDROID_HOME%\platform-tools
```

Nota: Para mais detalhes consultar o guia laboratorial 2.

2 Componente Serviço

Service Component

Este guia laboratorial tem como objetivo explorar a componente Serviço de aplicações Android™ e a segmentação de tarefas em *threads*. Também procura dar a conhecer a *User Interface* (UI) *Thread*, que é aquela que domina o fluxo de execução por defeito numa aplicação Android™.

O guia está estruturado de forma a construir, de forma faseada, e através de tentativa e erro, uma só aplicação móvel. De forma a atingir os vários objetivos propostos ou a enfatizar alguns aspetos, alguns exercícios levam, contudo, a aplicação para concretizações erradas, que depois são solucionadas. Sugere-se, assim, que vá guardando as várias versões da aplicação, ou que faça aplicações diferentes quando achar pertinente.

Tarefa 3 *Task 3*

Comece por criar um novo projeto Android™ através do Android Studio™ com as seguintes especificações:

- Nome da aplicação — `exServices1`;
- 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 `FloatingAlarms`;
- Peça para gerar o ficheiro de *layout* (o nome do ficheiro de *layout* deve ser `activity_floatingalarms.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 4 *Task 4*

Altere o *layout* da atividade principal de modo a que esta contenha apenas uma etiqueta de texto e um botão, ambos centrados no ecrã. A etiqueta de texto deve dizer `Floating Alarms Application` e o botão deve dizer `Start Service`. Quando pressionado, o botão deve despoletar o método `onButtonClick()`.

Tarefa 5 *Task 5*

Prepare, instale e execute a aplicação.

Q1.: A aplicação está a funcionar?

- ☐ Sim, funciona na perfeição, e já mostra alarmes e tudo!
- ☑ Sim, funciona, mas o botão ainda não faz nada...
- ☐ Não há nada que funcione!

Tarefa 6 *Task 6*

Crie um novo ficheiro `.java` na diretoria `app\src\main\java\pt\ubi\di\pmd\exservices1` com o nome `ServiceAlarms.java`. Dentro desse ficheiro, coloque o seguinte excerto de código:

```
package pt.ubi.di.pmd.exservices1;

import android.content.Intent;
// MISSING IMPORTS

public class ServiceAlarms extends Service {

  @Override
  public int onStartCommand
  (Intent intent, int flags, int startId) {
      Toast.makeText(
        this,
        "Service Started!",
        Toast.LENGTH_SHORT)
      .show();
    return START_NOT_STICKY;
  }

  @Override
  public IBinder onBind(Intent intent) {
    return null;
  }
}
```

Faltam alguns *imports* no excerto de código anterior. **Q2.: Assinale, nas opções seguintes, quais são esses *imports* (e complete o código de acordo com a resposta).**

- ☑ `import android.app.Service`
- ☑ `import android.os.IBinder`
- ☐ `import android.util.Log`
- ☐ `import android.view.View`

Q3.: Em que pacote é que pode ser encontrada a classe `Toast`?

- ☐ Na classe `import java.lang`.
- ☐ Na classe `import android.view`, como de resto não havia outra hipótese.
- ☑ Na classe `import android.widget`, e faz todo o sentido que assim seja.
- ☐ Na classe `import android.os`, como não podia deixar de ser!

Analise o código anterior atentamente. **Q4.: Era mesmo preciso reescrever o método `onBind()` para o Serviço em questão?**

- ☐ Neste caso, este método não está lá a fazer nada, porque até devolve `null`. Portanto: não!
- ☑ Na verdade, sim. Eu até experimentei, e não compila se o método não estiver lá.
- ☑ À campeão: sim, é preciso.
- ☐ À campeão: não, é totalmente desnecessário.

Q5.: Dada a implementação do Serviço, acha que este vai ser um `started service` ou um `bound service`?

- ☐ Nem sei por onde começar a procurar a resposta a isto.
- ☑ Um *started service*, i.e., um Serviço sem vínculo.
- ☐ Um *bound service*, i.e., um Serviço com vínculo.

Tarefa 7 *Task 7*

Implemente o método `onButtonClick()` na atividade principal de forma a que esta inicie o Serviço implementado na tarefa anterior. Sugestão:

```
import android.content.Intent;
import android.view.View;
...
public void onButtonClick(View v){
  Intent oIntent = new Intent(this, ServiceAlarms.
      class);
  startService(oIntent);
}
```

Tarefa 8 *Task 8*

Prepare, instale e execute a aplicação. **Q6.: O botão funciona?**

- ☐ Não funciona, e não consigo perceber a razão por detrás desse facto.
- ☐ Sim, funciona!
- ☑ Ahhhh... esqueci-me de declarar o Serviço no `AndroidManifest.xml`!

Tarefa 9 *Task 9*

Caso ainda não o tenha feito, declare o Serviço que criou no `AndroidManifest.xml`, colocando o seguinte excerto de XML dentro do elemento `application`:

```
<service android:name="ServiceAlarms">
 <intent-filter>
 <action android:name="pt.ubi.di.pmd.ServiceAlarms.
     SERVICE" ></action>
 </intent-filter>
</service>
```

No final, teste de novo a aplicação e certifique-se de que ficou a funcionar. **Q7.: Qual o artefato que demonstra que o serviço ficou realmente a correr?**

- ☐ Não há artefato, e eu não sou o Indiana Jones.
- ☐ A aplicação desaparece (porque o serviço começou a correr em segundo plano).
- ☑ O artefatto é a presença de uma *toast* quando carrego no botão, e que é despoletada no serviço, não na atividade...

Q8.: Já agora, acha que, para as funcionalidades que já foram enunciadas e implementadas, é necessária a definição do filtro de intentos incluído no código anterior?

- ☐ É estritamente necessária, já que sem essa definição, o Serviço não fica bem declarado no manifesto.
- ☐ É estritamente necessária, já que sem essa definição, o Intento que despoleta o Serviço na atividade principal não iria funcionar.
- ☑ Penso ser desnecessário, visto que ainda não vi qualquer funcionalidade que precisasse desta definição.

3 Serviços e *Threads*

Services and Threads

Tarefa 10 *Task 10*

Coloque as linhas de código incluídas em baixo depois da emissão da mensagem `Toast` implementada no `ServiceAlarms.java`:

```
try{
  Thread.sleep(7000);
}catch( InterruptedException oIE ){
  Log.e("SERVICEALARMS", "Interrupted Exception!");
}
```

Provavelmente vai precisa de importar as seguintes classes:

```
import java.lang.Thread;
import java.lang.InterruptedException;
import android.util.Log;
```

Q9.: Quanto tempo fica a *thread* adormecida após inclusão das linhas de código anteriores?

- ☐ 1 segundo.
- ☑ 7 segundos.
- ☐ 7 milissegundos.
- ☐ 1000 segundos.
- ☐ 7000 segundos.
- ☑ 7000 milissegundos.
- ☐ 1000 minutos.
- ☐ 7000 minutos.
- ☐ 7000 horas.

Tarefa 11 *Task 11*

Prepare, instale e teste a aplicação. **Q10.: Ao carregar no botão, aparece a mensagem `Service Started!`?**

- ☐ Não aparece... e o meu dispositivo ou aplicação parecem ter *encravado*.
- ☑ Não aparece... e aplicação parece ter *encravado*, sendo possível, contudo, evoluir para o *Home Screen* com um pouco de paciência.
- ☐ Não aparece... e no fim de um bocado, aparece uma mensagem a perguntar se quero terminar a aplicação. Má onda!

☐ Aparece sem problemas.

Tarefa 12 *Task 12*

Repita as últimas duas tarefas, desta feita, colocando a interrupção da *thread* antes da implementação da mensagem `Toast`.

Q11.: Qual o comportamento da aplicação quando a executa e carrega no botão `Start Service`?

☐ A aplicação tem exatamente o mesmo comportamento de antes.
☑ A aplicação bloqueia (e.g., o botão fica selecionado e não me deixa mexer em nada), mas no fim de 7 segundos, aparece a mensagem `Service Started!` no ecrã!
☐ Desta vez, a aplicação já não *encrava*.

Q12.: Dado o que observou, qual das seguintes afirmações lhe parece ser a mais correta?

☑ A *thread* que adormeceu é a mesma *thread* que trata da interface do utilizador.
☐ A *thread* que adormeceu é diferente da *thread* que trata da interface do utilizador.

Q13.: Qual lhe parece ser a explicação para a diferença entre comportamentos que observou?

☑ Aparentemente, se adormecer a *thread* logo depois de tentar mostrar a mensagem, esta não é mostrada no ecrã, e quando a aplicação recupera já é tarde demais para a mostrar.
☐ Aparentemente, se adormecer a *thread* logo depois de tentar mostrar a mensagem, esta não é mostrada no ecrã, por ter sido cancelada na colocação da *thread* a dormir.
☐ Aparentemente, se adormecer a *thread* antes de tentar mostrar a mensagem, esta não é mostrada no ecrã, e quando a aplicação recupera já é tarde demais para a mostrar.

Q14.: Que nome se dá à *thread* que, entre outras responsabilidades, trata da interação com o utilizador?

UIThread

---

4 *UI Thread e Handlers*

UI Threads and Handlers

Note que a ideia principal desta aplicação é implementar um Serviço que, depois de despoletado, emite um alarme em forma de `Toast`, de 5 em 5 segundos, mas que claramente não bloqueie a aplicação. Para isso, possivelmente, terá de recorrer a *threads*.

Tarefa 13 *Task 13*

Edite o ficheiro `ServiceAlarms.java` e substitua o código do método `onStartCommand()` pelo seguinte:

```
new Thread(){
  public void run(){
    for(int i = 0; i < 10; i++){
      Toast.makeText(
         ServiceAlarms.this,
         "Service Started!",
         Toast.LENGTH_SHORT)
      .show();

      try{
        Thread.sleep(5000);
      }catch( InterruptedException oIE ){
        Log.e("SERVICEALARMS", "Interrupted
            Exception!");
      }
    }
  }
}.start();
```

Q15.: Assim só de olhar para o código, acha que esta abordagem vai funcionar?

☐ Tudo parece indicar para que sim.
☑ Vê-se logo que não vai funcionar, nomeadamente porque

A Toast é um elemento gráfico, como tal deve estar no UIThread

---

Estamos a tentar adormecer a thread em que queremos mostrar a Toast

---

Tarefa 14 *Task 14*

Compile, instale e teste a aplicação. **Q16.: A aplicação funciona?**

☐ A aplicação é interrompida se carregar no botão `Start Service`, provavelmente porque apanhou a exceção e escreveu no *Log*. Tenho de verificar o *Log*.
☑ A aplicação é interrompida se carregar no botão `Start Service`, provavelmente porque estou a invocar um método para a *UI thread* onde não devia... De qualquer forma, o Android™ pede desculpa, pelo que não há problema!
☐ A aplicação é interrompida porque a *thread* principal é adormecida demasiado tempo...
☐ A aplicação funciona nos trinques.

Q17.: Consegue encontrar alguma entrada ANR no *logcat*?

☐ Nem sei o que significa ANR (e sinceramente, nem quero saber)!
☐ Consigo encontrar mas não sei o que significa e, agora tenho uma consulta, pelo que também não posso ir ver do que se trata, mas não fica esquecido.
☑ Consigo encontrar e sei o que significa. Até as galinhas dizem: imp-imp-imp; impecável.

Tarefa 15 *Task 15*

Existem várias formas de resolver o problema apontado

antes. Na verdade, o problema principal (o de estarmos a adormecer a *thread* principal) já está tratado, faltando apenas arranjar forma de, quando necessário, executarmos o método de criação da mensagem `Toast` na *UI Thread*. Para tal, sugere-se a utilização de um `Handler`, que é um objeto que **fica associado à *thread* onde é criado**, permitindo que sejam enviadas mensagens ou objetos `Runnable` para essa *thread* a partir de outras. Em baixo deixa-se uma sugestão para implementação do Serviço em questão:

```
package pt.ubi.di.pmd.exservices1;

import android.app.Service;
import android.content.Intent;
import android.widget.Toast;
import android.os.IBinder;

import java.lang.Thread;
import java.lang.Runnable;
import java.lang.InterruptedException;
import android.util.Log;
// IMPORT MISSING!

public class ServiceAlarms extends Service
{
  Handler oHandler;
  @Override
  public int onStartCommand
  (Intent intent, int flags, int startId) {
    oHandler = new Handler();
    new Thread(){
      public void run(){
        for(int i = 0; i < 10; i++){
          Runnable oRun = new Runnable(){
            @Override
            public void run(){
              Toast.makeText(
                ServiceAlarms.this,
                "Service Started!",
                Toast.LENGTH_SHORT)
              .show();
            };
          };
          oHandler.post(oRun);

          try{
            Thread.sleep(5000);
          }catch( InterruptedException oIE ){
            Log.e("SERVICEALARMS", "Interrupted
                Exception!");
          }
        }
      }
    }.start();
    return START_NOT_STICKY;
  }

  @Override
  public IBinder onBind(Intent intent) {
    return null;
  }
}
```

Q18.: Qual o *import* que falta no excerto de código anterior?

- ☐ `import android.util.Handler`
- ☐ `import android.Handler`
- ☐ `import android.lang.Handler`
- ☑ `import android.os.Handler`

Observe o código atentamente. **Q19.: A que *thread* é que o *oHandler* está associado?**

- ☑ À *UI thread*.
- ☐ À *thread* secundária.
- ☐ *Star Wars: Episode I - The Phantom Thread*

Q20.: Quanto tempo dura uma `Toast.LENGTH_SHORT`?

- ☐ Aproximadamente 1 segundo.
- ☑ Cerca de 2 segundos.
- ☐ Mais do que 1,5 segundos, mas menos de 1,8 segundos.
- ☐ Mais ou menos 3,5 segundos.
- ☐ Menos do que -2 segundos, mas mais de 3 segundos.

Tarefa 16 *Task 16*

Compile, instale e teste a aplicação. **Q21.: Funcionou?** ☑ Às mil maravilhas. ☐ Nem por isso.

Nota: caso tenha feito tudo como deve ser, a aplicação deve ter funcionado. Caso contrário, tome as providências que achar necessárias para resolver os problemas encontrados antes de avançar neste guia.

5 Recetores Difusão

Broadcast Receivers

Considere que pretendia que o seu Serviço de alarmes fosse ativado cada vez que o dispositivo móvel (do tipo *smartphone*) recebia uma SMS, independentemente de este ter sido ou não despoletado a partir da atividade principal.

Tarefa 17 *Task 17*

Modifique a aplicação que implementou antes de modo a que esta também contemple um recetor de difusão de eventos `android.provider.Telephony.SMS_RECEIVED`. Para conseguir este objetivo, considere as seguintes sugestões:

- Implemente um *BroadcastReceiver*, estendendo (e importando) a classe `android.content.BroadcastReceiver`;
- Reescreva o método `onReceive(Intent)` de forma a que este comece o Serviço pretendido;
- Declare o novo recetor de difusão no manifesto Android™, como filho do elemento `<application>` (não se esqueça de declarar também os filtros que lhe permitem receber o evento pretendido, nomeadamente `android.provider.Telephony.SMS_RECEIVED`);
- Declare as permissões que necessita para aceder àquele evento em particular, nomeadamente

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

```
</uses-permission>
```

Tarefa 18 *Task 18*

Resolva todos os problemas que eventualmente encontrar. Compile, instale e teste a aplicação num emulador Android™ ou num dispositivo real que suporte a receção de SMSs (um *tablet* não deverá servir para esta tarefa). Tome as medidas que achar necessárias para se certificar que o recetor está a funcionar. Por exemplo, num emulador, irá precisar de simular a chegada de uma SMS. Tal funcionalidade pode ser conseguida via `Android™ monitor`.