Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Programação de Dispositivos Móveis
Guia para Aula Laboratorial 7
Licenciatura em Engenharia Informática
Licenciatura em Informática Web

Programming of Mobile Devices
Guide for Laboratory Class 7
Degree in Computer Science and Engineering
Degree in Web Informatics

Sumário
Estudo de como se podem usar vários tipos de armazenamento para dados persistentes em dispositivos com Android™, com foco no recurso de preferências partilhadas, armazenamento interno e externo.

Summary
Study concerning the usage of the several storage options for persistent data, provided by devices with Android™, with focus on the shared preferences, internal and external storage resources.

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 Ficheiros Disponibilizados como Recurso do Projeto

Files Provided as Resources with the Project

O objetivo deste guia laboratorial é o de construir uma aplicação móvel Android™ totalmente funcional, partindo de parte do que já foi aprendido antes e adicionando mecanismos, recursos e funcionalidades relacionados com o armazenamento de dados persistentes (não estruturados). O objetivo principal é que esta aplicação permita escrever e guardar notas de uma forma muito simples.

Tarefa 3 *Task 3*

Crie um novo projeto Android™ através do Android Studio™ com as seguintes especificações:

- Nome da aplicação — `exStorage1`;
- 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 `SimpleNotes`;
- Peça para gerar o ficheiro de *layout* (o nome do ficheiro de *layout* deve ser `activity_simplenotes.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*

Crie a subdiretoria `raw`, dentro da diretoria `res`. Dentro dessa subdiretoria crie depois um ficheiro chamado `instructions.txt` com o seguinte texto:

```
To use this app, start writing your note
down, and exit or save anytime you want.
1. If you exit, any partial note will be
saved automatically but not sent anywhere.
2. When you feel the note is complete,
just save it and send it to your e-mail.
It will be made available in your external
storage also.
```

O ficheiro de texto anterior contém as instruções de utilização da aplicação que vai construir, e o seu conteúdo irá ser mostrado sempre que for relevante.

Tarefa 5 *Task 5*

Compile a aplicação e procure saber se este ficheiro (`instructions.txt`) é automaticamente mapeado no ficheiro `R.java`.

Q1.: Ainda se lembra em que diretoria é que este `R.java` fica alojado?

☐ Já me esqueci...

☐ Claro que lembro! Humm... Mas só vou verificar para ter a certeza. Fica em:

_____

☐ Sim, lembro! Fica em:

_____

Q2.: O ficheiro é mapeado no ficheiro `R.java`?

☐ Não, não é.

☑ Sim, é, ficando com o identificador `R.raw.instructions`.

☐ Sim, é, ficando com o identificador `R.id.instructions`.

Q3.: O identificador para o qual é mapeado o ficheiro tem a ver com o seu nome?

☑ Sem dúvida. ☐ Nem por isso.

Tarefa 6 *Task 6*

A aplicação a construir será muito simples. A funcionalidade que oferece é apenas a de permitir que um utilizador escreva notas e as guarde no armazenamento externo, ou as transmita via uma ação de partilha. A aplicação será constituída por uma única atividade que deve conter os seguintes objetos interativos (*widgets*):

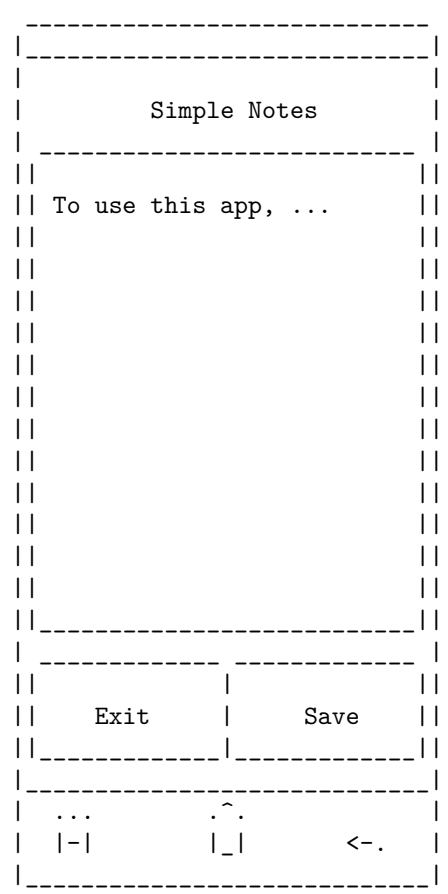
- Uma etiqueta de texto (`TextView`) com o nome da aplicação `Simple Notes` centrada ao cimo.
- Uma caixa de texto (`EditText`), que deve ocupar todo o espaço que não esteja ocupado por mais nenhum objeto;
- Um botão com o texto `Exit`, que deve poder ser usado para sair da aplicação; e
- Um botão `Save`, cuja funcionalidade é a de salvar a nota, introduzida na caixa de texto no armazenamento externo, e sair da aplicação.

Tome as providências que achar necessárias para que a atividade principal da aplicação tenha o aspeto mostrado

a seguir. Note que deve procurar a forma de **forçar a que caixa de texto ocupe todo o espaço deixado livre pelos outros objetos**, e que também os dois botões devem ocupar, horizontalmente, todo o espaço disponível.

Q4.: A definição da largura e da altura dos objetos interativos é obrigatória nas versões mais recentes do Android?
☑ Sim, é; caso contrário dá erro de compilação!
☐ Sim, é...
☐ Não, não é.

Q5.: O que é que acontece se definir a largura ou a altura de um *widget* a zero (0) (e.g., com `android:layout_width="0"`)?
☐ Esse *widget* não se vê por ter altura e largura 0.
☐ Esse *widget* passa a ocupar o ecrã todo.
☑ Esse *widget* ocupa sempre o máximo que consegue nessa dimensão tendo em conta outros elementos gráficos e as suas restrições.
☐ Esse *widget* fica com altura a mais e largura a menos ou vice-versa.



Tarefa 7 *Task 7*

Já que está a editar o ficheiro XML que define o *layout* da atividade principal, aproveite para definir os métodos `onClick` para os dois botões e um identificador para a caixa de texto. Especifique os métodos `exitNotSave` e `sendSave` para os botões `Exit` e `Save`, respetivamente, e o identificador `etext` para a caixa de texto, i.e., adicione as seguintes linhas nos locais corretos:

```
android:id="@+id/etext"
```

```
android:onClick="exitNotSave"
```

e

```
android:onClick="sendSave"
```

Tarefa 8 *Task 8*

Espera-se que a aplicação tenha o seguinte comportamento:

1. Quando é iniciada pela primeira vez, deve mostrar as instruções na caixa de texto;
2. Se um utilizador escrever uma nota na caixa de texto e sair usando o botão `Exit`, a aplicação deve guardar o conteúdo que está na caixa de texto, mas apenas para poder retomar o estado da próxima vez que o utilizador voltar à aplicação;
3. Se o utilizador carregar no botão `Save`, a nota deve ser guardada num ficheiro do armazenamento externo, a aplicação deve sair e, quando voltar a entrar, devem ser novamente mostradas as instruções (como se tivesse sido feito um *reset*).

Para já, implemente o código que lhe permita mostrar o conteúdo do ficheiro `instructions.txt` na caixa de texto. Se estiver com tempo, considere ainda a seguinte questão. Pelo texto, parece ser possível ler de ficheiros que são colocados na subdiretoria `res/raw`. **Q6.: Também é possível escrever nesses ficheiros?**
☐ Claro. Por que não?
☐ Sim, é, mas com muito jeitinho.
☐ Só a própria aplicação é que pode escrever nesses ficheiros.
☑ Não, não é possível escrever nesses ficheiros, principalmente porque estarão dentro do arquivo `apk` aquando da sua execução.

Tarefa 9 *Task 9*

Compile, instale e teste a aplicação tantas vezes quantas forem necessárias para conseguir o objetivo da tarefa anterior. Resolva os vários problemas que for encontrando, nomeadamente relacionados com exceções de leitura e escrita em ficheiros, com alguma pesquisa.

Nota: caso precise capturar e tratar exceções, considere escrevê-las no *log* do sistema.

Q7.: Quais os pacotes que necessitou incluir para concluir esta parte do guia?
☐ `import android.os.Environment;`
☐ `import android.content.SharedPreferences;`

☐ `import java.io.InputStream;`
☑ `import java.io.FileInputStream;`
☐ `import java.io.FileOutputStream;`
☑ `import java.io.IOException;`
☐ `import java.io.File;`
☐ `import android.widget.EditText;`
☐ `import android.view.View;`
☐ `import android.util.Log;`
☐ Não havia lá mais?

3 Preferências Partilhadas

Shared Preferences

Note que será necessário guardar, de alguma forma, e entre utilizações da aplicação, se determinada nota já foi guardada de forma persistente ou não (i.e., se é necessário mostrar as instruções ou a nota anteriormente começada). Para isso, vamos fazer uso do recurso chamado `SharedPreferences`.

Tarefa 10 *Task 10*

Considere analisar o seguinte excerto de código Java e incluí-lo, completando-o, no método `onCreate(Bundle)`:

```
SharedPreferences oSP = getPreferences(0);
if( !oSP.getBoolean("recover",false) ){
// Code to populate the text box with the
// instructions in the instructions.txt
// file.
}else{
// Code to inicialize the text box with the
// text: "This functionality has not been
// implemented yet."
}
```

Q8.: Para que serve o inteiro no método `getPreferences(int)`?
☐ Este inteiro define qual o ficheiro de preferências a abrir (os nomes dos ficheiros de preferências são dados por `numero.xml`).
☐ Este inteiro define a quantos ficheiros de preferências vai aceder.
☑ Este inteiro define o modo de acesso ao ficheiro.
☐ Este inteiro define quantas variáveis vão ser acedidas ou guardadas no ficheiro de preferências.
☐ Este inteiro é sempre igual a `0xff` na API 255.

Q9.: Em que diretoria é que o ficheiro das preferências partilhadas é normalmente guardado?

/data/data

A função `getBoolean(string, boolean)` aceita uma *string* e um `boolean`. **Q10.: Para que serve o `boolean`?**
☐ Este valor deve ser `false` quando queremos obter o valor que está guardado com a chave `recover`; e `true` quando queremos substituir esse valor.
☐ Este valor deve ser `true` quando queremos obter o valor que está guardado com a chave `recover`; e `false` quando queremos substituir esse valor.
☑ Este valor é devolvido de novo pela função `getBoolean()` caso a chave-valor não exista no ficheiro de preferências.
☐ Esta variável é usada para definir qual é o tipo primitivo da variável que se quer obter.

Q11.: É possível guardar tipos complexos (e.g., objetos) nas `SharedPreferences`?
☑ Não. Só tipos simples primitivos.
☐ Sim, pode-se guardar tudo tudo o que quisermos exceto, talvez, dados estruturados. Esses não! Mas de resto podemos guardar tudo.

Tarefa 11 *Task 11*

Compile, instale e teste a aplicação. Não avance antes de se certificar de que tudo está bem até esta parte do guia.

Q12.: Quais os pacotes que necessitou incluir para concluir esta parte do guia (para além dos que já tinha assinalado antes)?
☐ `import android.os.Environment;`
☑ `import android.content.SharedPreferences;`
☐ `import java.io.FileInputStream;`
☐ `import java.io.FileOutputStream;`
☐ `import java.io.File;`
☐ `import android.view.View;`

Tarefa 12 *Task 12*

Implemente os dois métodos que tratam o evento de clique nos dois botões definidos. Num dos métodos (`exitNotSave`) deve colocar código que permita colocar a variável `recover` a `true`. No outro (`sendSave`), deve colocar código que permita ajustar a variável `recover` a `false`.

Q13.: Qual é a classe do objeto que lhe permite ajustar os valores guardados nas preferências partilhadas?
☐ `SharedPreferences` ☑ `Editor`
☐ `Adjuster` ☐ `Property`
☐ `Activity` ☐ `putMethod()`

Q14.: Qual ou quais os nomes dos métodos que lhe permitem guardar, de facto, as alterações que estiver a introduzir nas preferências partilhadas?
☐ `forFact()` ☐ `Save()` ☑ `Commit()`
☑ `Apply()` ☐ `Reply()` ☐ `Undo()`

Q15.: Para que serve o método `Undo()`, enunciado na questão anterior?

☐ Para refazer uma determinada ação no objeto.
☐ Para desfazer uma determinada ação no objeto.
☑ Este método não existe. *Busted*!

Tarefa 13 ***Task 13***

Compile, instale e teste a aplicação. Note que deve testar ambos os botões e verificar se a aplicação já exibe o comportamento esperado.

4 Armazenamento Interno

Internal Storage

Anteriormente, foi dito que, caso o utilizador carregasse no botão Exit, qualquer nota que estivesse na caixa de texto deveria ser salva temporariamente, para que quando voltasse, esta ainda persistisse na caixa de texto. Para conseguir este efeito, faça uso do armazenamento interno. Esta secção foca-se, portanto, na implementação de duas funcionalidades diferentes:

1. Aquela que permite guardar o conteúdo da caixa de texto num ficheiro;
2. Aquela que permite restaurar o conteúdo desse ficheiro para a caixa de texto.

Tarefa 14 ***Task 14***

Foque-se na implementação do método `exitNotSave(View v)`. Recorde quando é que este método é executado:
☐ Nunca é executado.
☐ Quando a aplicação vai para segundo plano.
☐ Quando carregamos no botão Exit.
☐ Logo após a execução do método `onCreate()`.

Considere simplesmente copiar o código seguinte para o método mencionado em cima:

```
try{
  FileOutputStream fosFile =
      openFileOutput("savednote.txt",0);
  EditText oET = (EditText) findViewById(R
      .id.etext);
  fosFile.write( oET.getText().toString().
      getBytes() );
  fosFile.close();
}catch(IOException e){ Log.v("SIMPLENOTES"
    ,"FILE IO PROBLEM"); }
```

Q16.: O que faz o código incluído antes?
☐ Abre e fecha um ficheiro.
☐ Abre um ficheiro, escreve algo nesse ficheiro, e depois fecha o ficheiro.
☐ Abre um ficheiro, lê o seu conteúdo e escreve-o na caixa de texto.

Q17.: Qual o significado do número 0 no método `openFileOutput(string, int)`?
☐ Que o ficheiro aberto não pode ser fechado.
☐ Que o ficheiro é criado de novo com 0 bytes.
☐ Que o ficheiro é criado no modo privado, o que significa que só a aplicação é que lhe pode aceder.
☐ É um erro. Não se pode abrir um ficheiro para leitura especificando um 0 no segundo parâmetro.

Q18.: Lembra-se de ter implementado, no método `exitNotSave(View v)`, um editor para as preferências partilhadas?
☐ Sim, lembro.

Quando implementou essa parte, deve ter colocado o método `commit()` ou o `apply()` no método que agora está a completar. **Q19.: Pense bem: onde é que faz mais sentido colocar esses métodos?**
☐ Dentro do bloco `try{...}catch{...}`
☐ Antes do bloco `try{...}catch{...}`
☐ Depois do bloco `try{...}catch{...}`
☐ Como estou na dúvida, meto em tudo quanto é lado.

Tarefa 15 ***Task 15***

Note que falta implementar parte do código no método `onCreate(Bundle)`, nomeadamente aquela secção que restaura o conteúdo de uma nota inacabada na caixa de texto. Use o que já aprendeu até aqui para completar esta parte do código. **Sugestão: use a linha de código incluída a seguir.**

```
FileInputStream fisFile =
                openFileInput("savednote.txt");
```

Q20.: Em que diretoria do sistema de ficheiros Android™ é que o ficheiro `savednote.txt` é guardado?

Q21.: É possível verificar a existência do ficheiro usando o `Android Monitor`?
☐ Olha! Boa ideia!
☐ Não, não é possível.

Tarefa 16 ***Task 16***

Compile, instale e teste a aplicação. Saia e entre várias vezes da aplicação carregando no botão Exit e alterando o conteúdo da caixa de texto, para se certificar de que o que implementou nesta parte do guia está correto.

5 Armazenamento Externo

External Storage

As duas funcionalidades que estão em falta dizem respeito ao botão `Save`.

Tarefa 17 *Task 17*

No método referido antes, implemente a parte do código que permite guardar a nota escrita na caixa de texto num ficheiro do armazenamento externo. O ficheiro deve chamar-se `note.txt`, e a diretoria **pública** onde guarda o ficheiro pode ser, por exemplo, a das imagens (não faz muito sentido), que costuma estar sempre disponível em emuladores ou dispositivos com Android™. Em princípio, vai precisar das seguintes instruções:

```
File path = Environment.
    getExternalStoragePublicDirectory(Environment.
    DIRECTORY_PICTURES);
File fFnote = new File(path, "note.txt");
FileOutputStream fosFile = new FileOutputStream(
    fFnote);
...
```

No excerto de código anterior, o objeto `path` é declarado como um `File` (um ficheiro). **Q22.: Isto faz sentido?**

- ☐ Em Linux faz todo o sentido, visto que tudo, inclusive as diretorias, são ficheiros.
- ☐ Não faz sentido nenhum. O Prof. devia ter arranjado outro nome para a variável.
- ☐ O método `getExternalStoragePublicDirectory(...)` não devolve um `File`, mas sim um `Directory`.

Q23.: Lembrou-se de colocar o `commit()` no local certo da função?

- ☐ Se não fosse o Prof., não sei o que seria de mim.
- ☐ Atão não lembrei?

Q24.: Precisou de importar pacotes adicionais para esta parte do guia laboratorial?

- ☐ Sim, nomeadamente o(s) pacote(s):

_____

_____

- ☐ Não, já tinha tudo o que precisava.

Q25.: Tem fechado todos os ficheiros que tem aberto?

- ☐ Eh... estava só a guardar essas partes assim mais para o fim da implementação...
- ☐ Eh... sim sim, tenho fechado tudo.

Tarefa 18 *Task 18*

Compile, instale e teste a aplicação. Saia e entre várias vezes da aplicação carregando no botão `Save` e verificando que o ficheiro `note.txt` é gerado ou reescrito com o conteúdo correto.

Nota: pode eventualmente revelar-se útil a instalação de um gestor de ficheiros no emulador (para navegar pelo sistema de ficheiros, nomeadamente pelo armazenamento externo). Há várias formas de instalar pacotes num emulador (e.g., usado `$ adb install...`). Uma dessas formas consiste em fazer o *download* do arquivo `.apk` via *browser* do próprio emulador, instalando-o depois de o selecionar na pasta *downloads*. Neste âmbito, talvez deva considerar uma visita a `http://www.appsapk.com/es-file-explorer/`.

Tarefa 19 *Task 19*

Finalmente, note que também foi pedido que fornecesse a possibilidade do conteúdo da caixa de texto ser enviada por e-mail ou SMS. Isto deve acontecer ao mesmo tempo que a nota é guardada no ficheiro do armazenamento externo, i.e., ao ser pressionado o botão `Save`. Esta última tarefa consiste em implementar esta funcionalidade (usando um `ACTION_SEND`). No final, teste a aplicação.