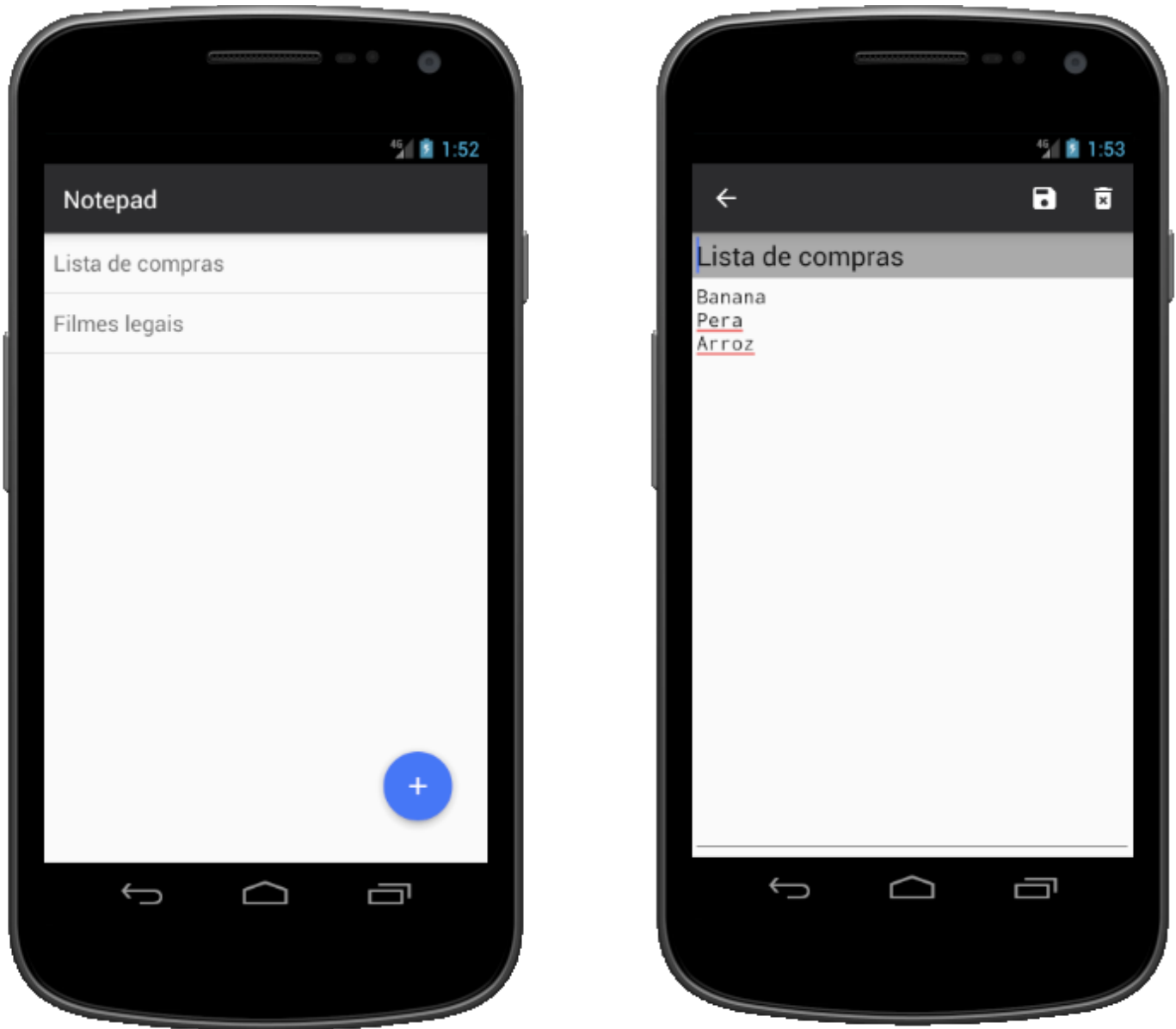


Desenvolvimento Mobile - Trabalho 2

App Notepad

O aplicativo Notepad possui duas *Activities*:



1- Listar Arquivos: tela organizada com uma *ListView* e um *Floating Action Button*. A *ListView* deve exibir os arquivos salvos na memória interna do celular. Quando um dos itens da lista for clicado, o app deve abrir a tela de *Edição do Arquivo* exibindo o título e o conteúdo do arquivo. O *FAB* deve abrir a tela de *Edição do Arquivo*, com o título e conteúdo do arquivo em branco (novo arquivo).





2- Edição do Arquivo: Esta tela contém um *TextView* (com background cinza) para o título do arquivo e um *TextView* (branco, ocupando toda a tela) para o conteúdo do arquivo. Além disso, esta Activity também possui um menu de contexto disponível na *ActionBar*, onde os botões de *Voltar*, *Salvar* e *Apagar* estão disponíveis.



- **Voltar:** retorna à Activity *Listar Arquivos* sem alterar os dados do arquivo que está sendo exibido.
- **Salvar:** salva ou altera os dados do arquivo na memória interna do aparelho. Caso o nome do arquivo esteja vazio, o app deve exibir uma mensagem "Nome do arquivo inválido!" com um componente *Toast*. Se o nome do arquivo já existir na memória interna, o sistema deve sobrescrevê-lo (sem informar nada ao usuário). Se o nome do arquivo não existir, a app deve criar um novo arquivo. Após a ação de salvar o app deve voltar para a tela *Listar Arquivos*.
- **Apagar:** apagar o arquivo cujo nome está indicado na *TextView*. Ao clicar neste botão, o app deve exibir uma *Snackbar* com uma mensagem "Deseja realmente apagar?" e uma action "APAGAR". Se o usuário clicar na action "APAGAR", o app deve excluir o arquivo da memória interna do celular e voltar para a tela *Listar Arquivos*.

Detalhes Técnicos

Deve-se trabalhar com classes específicas para as responsabilidades de **Modelo de Domínio, Controller, Apresentação e Repositório**.

Apresentação	Controller	Modelo	Repositório
			

Como manipular arquivos de texto na memória interna?

A classe repositório deve conter métodos para ler, salvar e apagar arquivos internos do celular. Para fins de construção desta funcionalidade, deve-se utilizar a classe *ArquivoHelper* disponível no Github:

<https://github.com/lgapontes/aulas-mobile/blob/master/ArquivoHelper.java>

```
public class ArquivoHelper {
    public static String[] listarArquivos(Context context) {
        File diretorio = context.getFilesDir();
        return diretorio.list();
    }
    public static void salvarArquivo(Context context, String nome, String conteudo) throws IOException {
        FileOutputStream fileOutputStream = context.openFileOutput(nome, Context.MODE_PRIVATE);
        fileOutputStream.write(conteudo.getBytes());
        fileOutputStream.close();
    }
    public static String lerArquivo(Context context, String nome) throws IOException {
        FileInputStream fileInputStream = context.openFileInput(nome);
        BufferedReader reader = new BufferedReader(new InputStreamReader(fileInputStream));
        StringBuilder conteudo = new StringBuilder();
        String linha;
        while ( (linha = reader.readLine()) != null) {
            conteudo.append(linha + "\n");
        }
        fileInputStream.close();
        return conteudo.toString();
    }
    public static void apagarArquivo(Context context, String nome) {
        String path = context.getFilesDir() + "/" + nome;
        File arquivo = new File(path);
        if (arquivo.exists()) {
            arquivo.delete();
        }
    }
}
```

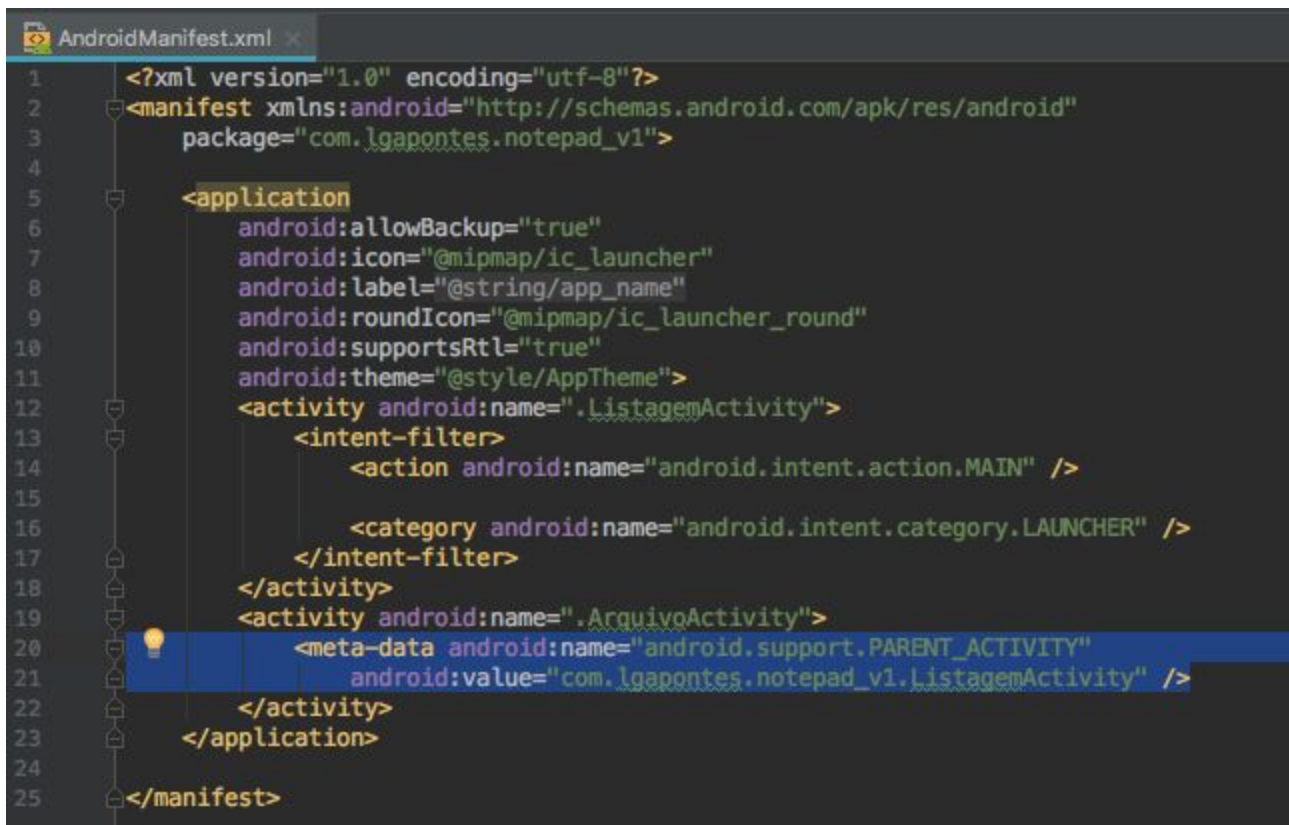
Veja abaixo detalhes desta classe:

- **listarArquivos():** método que retorna um array de strings com o nome dos arquivos disponíveis na memória interna do celular.
- **salvarArquivo():** método que salva o arquivo na memória do celular.
- **lerArquivo():** método que permite realizar a leitura do conteúdo do arquivo. Recebe como parâmetro o nome do arquivo.
- **apagarArquivo():** método que exclui um arquivo da memória interna do celular. Isso só é realizado para arquivos existentes (nomes de arquivos não encontrados são ignorados).

Mais detalhes de implementação desta classe serão discutidos no futuro - principalmente os métodos disponíveis no objeto *context* (que é a *Activity* aberta no momento).

Como criar um menu de contexto na *ActionBar*?

O menu de contexto na *ActionBar* nada mais é do que um menu com um botão de voltar que retorna a uma **Activity** classificada como pai. A classificação da Activity pai deve ser realizada no arquivo *Manifest.xml*, conforme código a seguir:



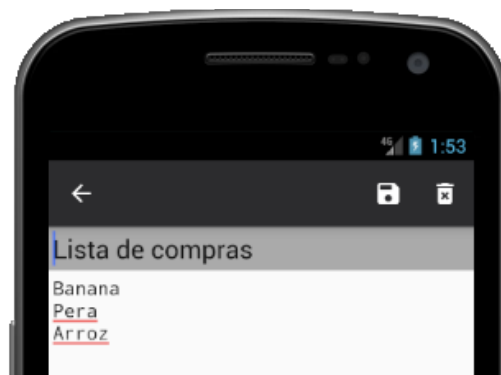
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.lgapontes.notepad_v1">
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="@string/app_name"
9         android:roundIcon="@mipmap/ic_launcher_round"
10        android:supportsRtl="true"
11        android:theme="@style/AppTheme">
12        <activity android:name=".ListagemActivity">
13            <intent-filter>
14                <action android:name="android.intent.action.MAIN" />
15
16                <category android:name="android.intent.category.LAUNCHER" />
17            </intent-filter>
18        </activity>
19        <activity android:name=".ArquivoActivity">
20            <meta-data android:name="android.support.PARENT_ACTIVITY"
21                android:value="com.lgapontes.notepad_v1.ListagemActivity" />
22        </activity>
23    </application>
24
25 </manifest>
```

Além deste ajuste, precisamos incluir as seguintes linhas no método *onCreate()* da Activity cujo menu de contexto será exibido.

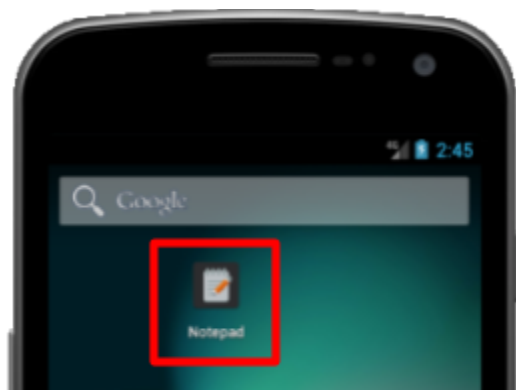
```
getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
getSupportActionBar().setTitle("");
```

Observação: como o menu de contexto também é uma menu da *ActionBar*, método *onCreateOptionsMenu()* também é necessário para inflar o menu e seus itens. Veja a seguir um exemplo de como deve ficar este menu:

As imagens utilizadas no *Floating Action Button* (+) e no menu de contexto (disquete e lixeira) estão disponíveis na pasta *imagens* do trabalho 2 no EAD. A imagem de voltar (seta para esquerda) já é automaticamente incluída com a configuração supracitada.



Inclusão do Launcher Icon



Deve-se definir o Launcher Icon da aplicação a partir das imagens de *background* e *foreground* disponíveis na pasta *imagens* do EAD.

Avaliação do Trabalho

Detalhes que serão considerados:

- Posicionamentos das *Views*
- Lógica da App
- A App deve estar rodando no Emulador ou Celular
- Explicação do código-fonte do projeto