Desenvolvimento Móvel em Android

View Binding

Menus e Settings

Sónia Luz, sonia.luz@ipleiria.pt
David Safadinho, david.safadinho@ipleiria.pt
Departamento de Engenharia Informática
Escola Superior de Tecnologia e Gestão
Instituto Politécnico de Leiria
1º Semestre - 2021/2022

Vinculação de visualização

- *View Binding*: Recurso que facilita a programação de código que interage com visualizações
  - disponível a partir da versão **3.6** do Android Studio;
  - efetua o *Binding* automático;
    - através de uma classe de vinculação gerada para cada ficheiro xml de layout;
  - substitui o ***findViewById()***;
  - permite evitar erros de acesso incorreto;

Vinculação de visualização

- Exemplo de acesso com ***findViewbyId()***

```
public class AboutActivity extends AppCompatActivity {
    private TextView textViewNome;
    private EditText editTextEmail;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about);

        textViewNome = findViewById(R.id.textViewNome);
        editTextEmail = findViewById(R.id.etEmail);

        textViewNome.setText("Sónia Luz");
        editTextEmail.setText("sonia.luz@ipleiria.pt");
    }
}
```

Vinculação de visualização

- Configurar o *Binding* automático
  - Ativar o viewBinding no Modulo

build.gradle (Module: app)

```
android {
  ...
     viewBinding{
         enabled = true
     }
}
```

  - Sincronizar o projeto: **Sync Now**

Vinculação de visualização

- Na classe ***AboutActivity***
  - Definir um atributo **`binding`** para a classe de vinculação **`ActivityAboutBinding`**
  - Efetuar o *inflate* do layout da atividade através do método **`inflate()`** da classe de vinculação
  - Atribuir a referência do layout ao **`setContentView`** da atividade através do método **`getRoot()`**
  - Aceder diretamente aos componentes *view* do *layout* através do atributo **`binding`**

Vinculação de visualização - Exemplo

```
public class AboutActivity extends AppCompatActivity {

    private ActivityAboutBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // setContentView(R.layout.activity_about);

        binding = ActivityAboutBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        binding.textViewNome.setText("Sónia Luz");
        binding.editTextEmail.setText("sonia.luz@ipleiria.pt");
    }
}
```

Menus

- São componentes comuns da UI em diversos tipos de aplicações;
- Servem para apresentar ações do utilizador e outras opções em cada atividade;
- Existem 3 tipos principais:
  - ***Option Menu*** (menu de opções)
  - ***Context Menu*** (menu flutuante)
  - ***Popup Menu***

Menus

- ***Options Menu***
  - É o tipo de menu mais comum em Android;
  - Nas versões 2.3 e anteriores do Android, o menu é acionado pressionando o botão físico do menu:
    - Aparece na parte inferior e pode conter até 6 itens;
    - Se esse número for ultrapassado a sexta opção terá o texto “Mais”, que depois apresenta as restantes opções;
  - Na versão 3.0 e posteriores surgiu o conceito de *Action Bar*
    - Na *Action Bar* os menus aparecem no canto superior direito;
      - Podem ser apresentados como um menu *drop-down*, ou serem forçados a aparecer na barra superior;

Menus

- ***Context Menu***
  - É um menu flutuante que aparece quando selecionamos e seguramos (*click* longo) um determinado componente visual do Android, como um botão;
  - Fornece ações que afetam o conteúdo selecionado ou a estrutura do contexto;
  - Assim, será apresentado no ecrã uma lista de opções
    - que correspondem ao menu de contexto;

Menus

- *Popup Menu*
  - É a implementação de menu mais atual
    - Foi incluído na versão 3.0 do Android;
  - É um menu modal ancorado a uma *View*
    - Aparece sob a *view* que serve de âncora se tiver espaço, ou sobre a *view*;

Definição de Menu em XML

- Em vez de criar um menu no código java da atividade;
- O Android fornece um formato xml para definir os itens de menu
  - Porque é mais fácil visualizar a estrutura do menu em XML;
  - Separa o conteúdo do menu do código da atividade;
  - Permite criar configurações alternativas
    - Por diferentes versões de plataforma, tamanho do ecrã, etc.;

Definição de Menu em XML

- Deve definir um menu e respetivos itens através de um recurso de menu
  - Para depois poder inflar (apresentar) o respetivo recurso de menu
  - Ou seja, carregá-lo como um objeto *Menu* na atividade ou fragmento;

Definição de Menu em XML

- Criar um ficheiro XML na pasta `res/menu/` do projeto;
- Criar o menu com os seguintes elementos:
  - `<menu>` - É o contentor para os itens de menu. Pode conter um ou mais elementos `<item>` e `<group>`;
  - `<item>` - Representa um único item de um menu. Pode conter um `<menu>` aninhando para criar um submenu;
  - `<group>` - É um contentor invisível e opcional para os elementos `<item>`. Permite categorizar itens de menu para partilharem propriedades como estado ativo e visibilidade.

Definição de Menu em XML

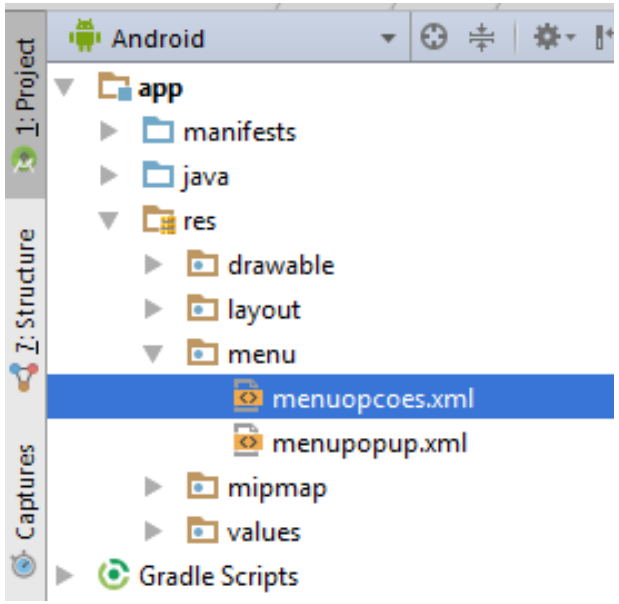
- Elemento `<item>` é compatível com vários atributos usados para definir aparência ou comportamento;
- Os principais atributos a definir são:
  - `android:id` - Id único do recurso, para que seja reconhecido quando o utilizador o seleciona;
  - `android:icon` - Referência a um *drawable* (desenhável) usado como ícone do item;
  - `android:title` - Referência a uma string usada como título do item;
  - `app:showAsAction` - Permite especificar como e quando esse item deve aparecer como item de ação na *Action Bar; Ex*: *never, ifRoom, always*, …;

Usar o Menu na Atividade

- Para usar o menu na atividade pretendida
  - É necessário inflar o recurso do menu
    - Ou seja, converter o recurso XML num objeto programável;
  - Através de `MenuInflater.inflate()`;
  - Cuja aplicação depende do tipo de menu a criar;

Criar um *Options Menu*

- Definir o ficheiro `menu_opcoes.xml` correspondente ao menu;




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

    <item
        android:id="@+id/menu_1"
        android:icon="@android:drawable/ic_menu_sort_alphabetically"
        android:title="@string/txtItemMenu1"
        app:showAsAction="always" />
    <item
        android:id="@+id/menu_2"
        android:icon="@android:drawable/ic_input_add"
        android:title="@string/txtItemMenu2"
        app:showAsAction="always" />
    <item
        android:id="@+id/menu_3"
        android:icon="@android:drawable/ic_search_category_default"
        android:title="@string/txtItemMenu3"
        app:showAsAction="always" />
</menu>
```

Criar um *Options Menu*

- Para especificar o menu de opções para uma atividade, é necessário modificar o método `onCreateOptionsMenu()`;
  - Inflar o recurso de menu definido no XML no `Menu` fornecido pelo método *callback*;

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    //inflar o menu criado
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_opcoes, menu);
    return super.onCreateOptionsMenu(menu);
}
```

Criar um *Options Menu*

- Processar os eventos de seleção
  - Quando o utilizador seleciona um item do menu de opções
    - Incluindo os itens de ação da *Action Bar;*
  - O sistema invoca o método `onOptionsItemSelected()` da atividade;
    - Este método recebe o `MenuItem` selecionado;
    - É possível identificar o item através do método `getItemId()` que devolve o Id único definido;
    - Quando se processa um item de menu com sucesso devemos devolver *true*;
    - Se nenhum item for processado devemos chamar a implementação da superclasse que devolve *false*;

Criar um *Options Menu*

- Processar os eventos de seleção

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()){
        case R.id.menu_1:
            Toast.makeText(this, item.getTitle(), Toast.LENGTH_SHORT).show();
            return true;
        case R.id.menu_2:
            Toast.makeText(this, item.getTitle(), Toast.LENGTH_SHORT).show();
            return true;
        case R.id.menu_3:
            Toast.makeText(this, item.getTitle(), Toast.LENGTH_SHORT).show();
            return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Criar um *Context Menu*

- Para especificar um menu de contexto para uma atividade:
  - Registar a *view* ao qual o menu deve estar associado através do método `registerForContextMenu()`;
  - Implementar o método `onCreateContextMenu()` na atividade, onde deve inflar o menu de contexto;
  - Para tratar o item selecionado deve implementar o método `onContextItemSelected()`;

Criar um *Popup Menu*

- Para definir um menu de *Popup*, para uma atividade:
  - Deve implementar um método para o *onClick* da *View* que servirá de ancora;
    - Onde deve instanciar um `PopupMenu`, indicando o contexto e a *view* a que está ancorado;
    - Inflar o menu através do `MenuInflater`;
    - Apresentar o menu através do `PopupMenu.show()`;
  - Para processar os eventos de seleção sobre um item:
    - A atividade deve implementar a interface `PopupMenu.OnMenuItemClickListener`;
    - Registá-la com o `PopupMenu` através do método `setOnMenuItemclickListener()`;
    - Quando o utilizador seleciona um item, o sistema chama o método `onMenuItemclick()`;

Up Action – Acção Voltar Atrás

- Cada aplicação deve ter uma forma fácil de encontrar o caminho até à atividade principal;
- Uma forma simples é adicionar um botão ***Up*** (ação voltar atrás)
  - Na *Action Bar*;
  - Para todas as atividades menos para a principal;
  - Assim, quando o utilizador selecionar o botão *Up*
    - A aplicação irá navegar para a atividade pai;

Up Action – Acção Voltar Atrás

- Para suportar esta funcionalidade:
  - É necessário cada aplicação declarar no ***manifest*** a sua atividade *Pai;*
  - Ativar o botão *Up* da *Action Bar*

    (passou a ser opcional);

Up Action – Acção Voltar Atrás

- Declarar uma atividade pai no ficheiro **AndroidManifest.xml**
  - Em versões mais antigas é necessário definir um <meta-data> com um par nome-valor
    - O nome é o "android.support.PARENT_ACTIVITY"
    - E o valor é nome da atividade pai

```
<activity android:name=".ListViewContactosActivity" >
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="pt.ipleiria.estg.dei.amsi.gestaocontactos.MainActivity"/>
</activity>
```

  - A partir da versão 4.1
    - Foi introduzido o atributo **parentActivityName** que deve ser definido na declaração da atividade

```
<activity android:name=".ListViewContactosActivity"
        android:parentActivityName=".MainActivity"/>
```

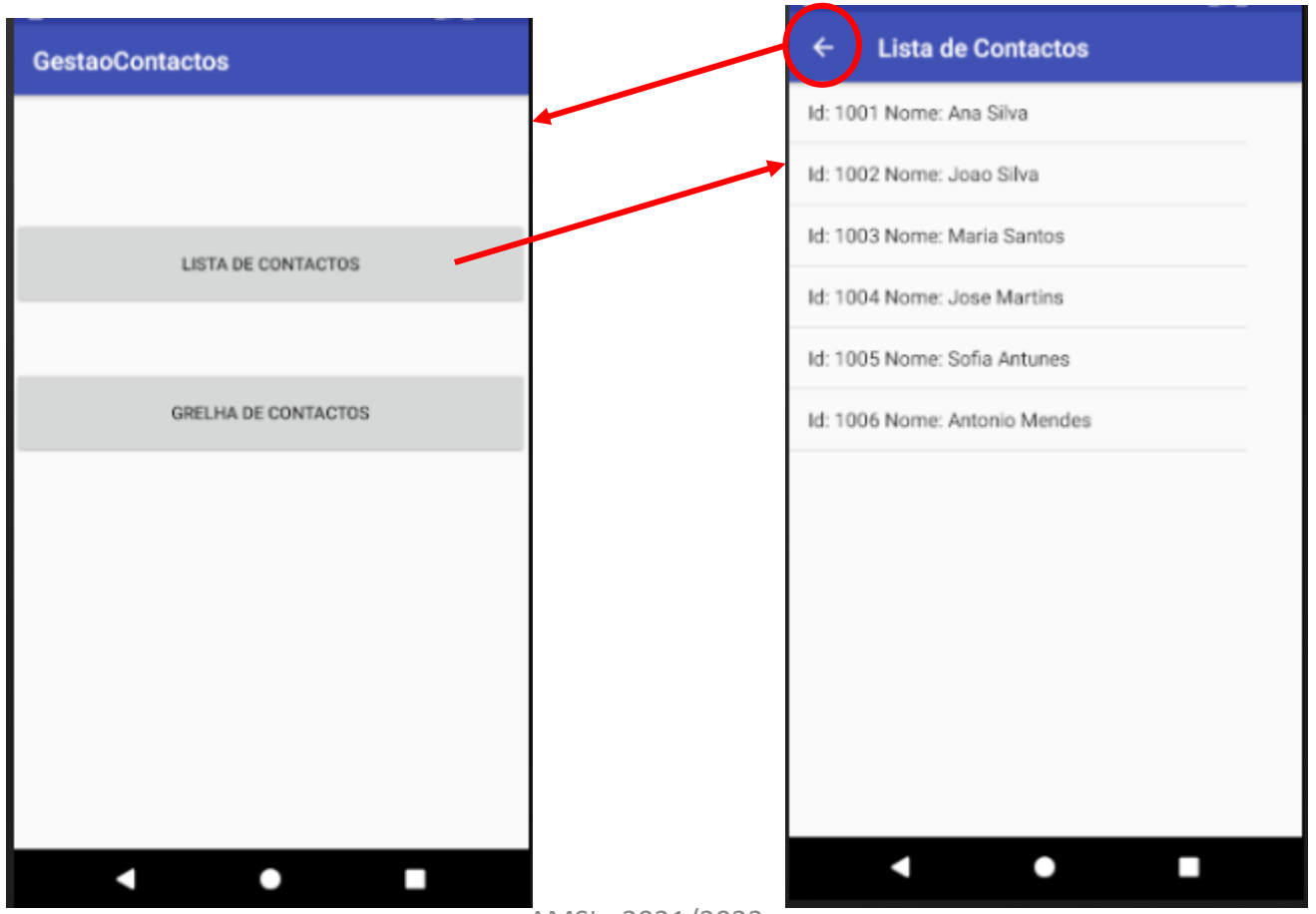
Up Action – Acção Voltar Atrás

- Ativar o botão *Up* da *Action Bar* da atividade (opcional)
  - Para aceder à *action bar* da atividade deve usar o método **`getSupportActionBar()`**
  - E usar o método **`setDisplayHomeAsUpEnabled()`** para efetivamente ativar o botão;

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
...
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
}
```

Up Action – Acção Voltar Atrás

- Após ativar o botão *Up* da *Action Bar* da atividade



SharedPreferences - Persistência de dados simples

- Necessidade:
  - Persistir dados durante o uso da aplicação:
    - Para que possam ser utilizados posteriormente;
    - Como preferências de utilizador;
  - Dados simples que não justificam o uso de bases de dados ou outro tipo de armazenamento mais complexo;
- Solução:
  - O Android disponibiliza uma forma de armazenamento;
    - Designada de ***SharedPreferences***, ou preferências de utilizador.

SharedPreferences - Persistência de dados simples

- Usadas para dados simples, como por exemplo:
  - *Flag* que indica se é a primeira vez que o utilizador entra na aplicação;
  - Dados de login;
  - Preferências do utilizador: "Exibição de temperatura em que tipo de graus";
  - Mas **NUNCA** guardar passwords ou dados pessoais como telefone, cartão crédito, etc;

SharedPreferences - Persistência de dados simples

- Consiste numa interface que permite aceder e modificar dados de preferência de utilizador;
- O valor armazenado apresenta-se sob o formato *key-value* (chave-valor);
- Ou seja, cada preferência armazenada possui uma identificação ou chave
  - E associada a esta chave está um valor;
- Permite armazenar diversos tipos de valor:
  - int, float, booleans, Strings e conjuntos de Strings;

SharedPreferences - Persistência de dados simples

- Existem duas formas de trabalhar com a SharedPreferences:
  - `getPreferences()` – Quando há poucas preferências a armazenar;
    - Guardar dados dentro da mesma chave, como preferências de configuração da aplicação;
  - `getSharedPreferences()` - Quando há muitas preferências a armazenar;
    - Criar uma *sharedPreference* para cada categoria de dados a armazenar;

- Aceder a *sharedPreference* com apenas uma chave-valor:
  - Deve criar um atributo do tipo ***SharedPreference*** o preenchê-lo através do método `getPreference()`, indicando o modo de acesso:
    - `PRIVATE` - Preferência apenas pode ser acedida pela aplicação;
    - `APPEND` - Aplicado à criação de ficheiros;
    - …;

SharedPreferences - Persistência de dados simples

- Para alterar uma preferência utiliza-se um atributo do tipo ***SharedPreference.Editor***
  - Que irá permitir guardar o valor da *sharedPreference* alterado, ou atribuído;
- Para guardar a preferência desejada é preciso utilizar o método `edit()` do ***Editor*** criado anteriormente;

SharedPreferences - Persistência de dados simples

- Após isso, atribui-se o valor consoante o tipo de dados a guardar;

- Finalmente é necessário definir o método a usar para que a alteração seja efetivada:
  - `commit()` – Disponível desde a primeira versão da API, e é executado de forma síncrona;
  - `apply()` – Disponível a partir da versão 9, e é executado de forma assíncrona;

SharedPreferences - Persistência de dados simples

```
public class MainActivity extends AppCompatActivity {
    private String nome;
    SharedPreferences sharedPreferences;
    SharedPreferences.Editor editor;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //aceder à sharedPreference e definir o modo de acesso
        sharedPreferences = getPreferences(Context.MODE_PRIVATE);
        //definir o Editor para permitir guardar / alterar o valor
        editor = sharedPreferences.edit();
        //ir buscar o seu valor a partir da chave
        nome = sharedPreferences.getString("meuNome", "");

        if( nome == ""){
            Toast.makeText(this, "Nome não estava guardado", Toast.LENGTH_SHORT).show();
            editor.putString("meuNome", "Sónia");
            editor.apply();
        }else{
            Toast.makeText(this, "Nome: " + nome, Toast.LENGTH_SHORT).show();
        }
    }
```

Desafio:

- Criar um menu de opções na MainActivity da aplicação de gestão de contactos:
  1. Definir um menu.xml com 3 itens;
  2. Associar o menu à atividade principal;
  3. Implementar o método para processar o evento de seleção;
  4. Verificar quais as principais diferenças aos mudar o valor do atributo ***showAsAction***;

Fontes e Mais Informação

- *View Binding*
  - https://developer.android.com/topic/libraries/view-binding
- Menus
  - https://developer.android.com/guide/topics/ui/menus.html
- *Action Bar*
  - https://developer.android.com/training/appbar/index.html
- *Adicionar Acção Up (Voltar Atrás)*
  - https://developer.android.com/training/appbar/up-action.html#declare-parent
- *SharedPreferences*
  - https://developer.android.com/training/data-storage/shared-preferences

Próximo Tema:

Personalização de Aplicações com Estilos e Temas

- Estilos e Temas
  - https://developer.android.com/guide/topics/ui/themes.html
- Estilos do Android
  - https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/res/values/styles.xml
- Temas do Android
  - https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/res/values/themes.xml
- *FloatingAction Button*
  - https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html