P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Programação Para Dispositivos Móveis I

DIALOGS

2023/_24 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

NORTE 2030 Programa Regional do Norte

PORTUGAL 2030

Cofinanciado pela União Europeia

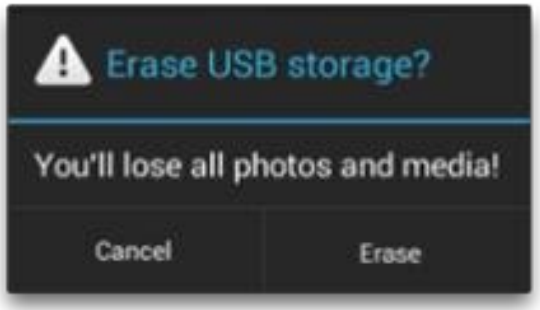
Índice

- Dialog;
- Apresentar um Dialog;
- Adicionar botões a uma Dialog;
- Dialog com listas;
- Dialog com Layout Personalizado;
- Processar Eventos do Dialog na Activity;
- Leitura Adicional.

Dialog

É uma **pequena janela flutuante** que permite ao utilizador **tomar uma decisão** ou **introduzir informação adicional.**

- Não ocupa totalmente o ecrã e é normalmente utilizado para os utilizadores tomarem ações antes de prosseguirem na aplicação.



Dialog

Tipos

A Dialog é a **classe base** para criação de um elemento deste tipo, contudo não devemos utilizar uma das suas subclasses:

- **AlertDialog :** Apresenta um título, até três botões, a lista de vários itens selecionáveis ou um layout personalizado;

- **DatePickerDialog** e **TimePickerDialog:** Uma dialog com layout próprio que permite ao utilizador selecionar uma data ou uma hora;

Estas classes definem o **estilo** e a **estrutura** da Dialog, mas na sua criação deve ser utilizada a classe DialogFragment como container.

AlertDialog

Elementos

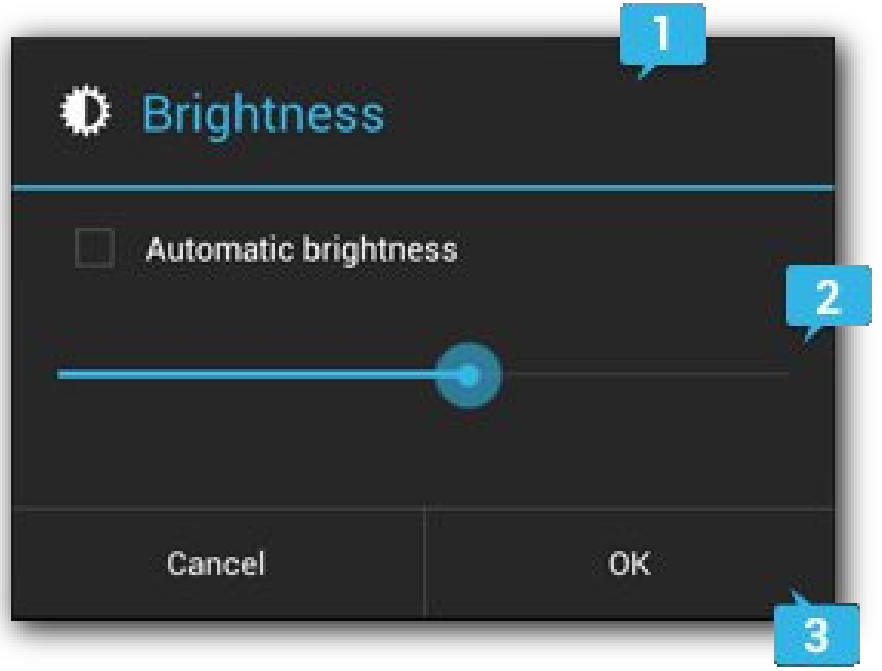
1. Título (opcional)

2. Área de conteúdo

   - Pode apresentar uma mensagem, uma lista ou um layout personalizado;

3. Botões

   - Não devem existir mais do que três;



AlertDialog

Dialog com mensagem e 3 botões

AlertDialog

FireMissilesDialogFragment.java

Extende DialogFragment

```
public class FireMissilesDialogFragment extends DialogFragment {
```

Definição do Builder de AlertDialog

```
@NonNull
@Override
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

    builder.setMessage(R.string.fire_missiles);

    return builder.create();
}
```

AlertDialog

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btn_missiles"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/missiles" />

</LinearLayout>
```

Botão que vai ser usado para mostrar o DialogFragment

(FireMissilesDialogFragment.java)

AlertDialog

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private Button btnMissiles;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnMissiles = (Button) findViewById(R.id.btn_missiles);
        btnMissiles.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_missiles: fireMissiles();
                break;
        }
    }

    private void fireMissiles() {
        new FireMissilesDialogFragment().show(getSupportFragmentManager(), "Missiles");
    }
}
```

Gestor de Fragments do Android

TAG

AlertDialog

Resultado atual

E os botões?

AlertDialog

Adicionar botões [FireMissilesDialogFragment.java]

Implementa o OnClickListener para detetar os clickes nos botões

```
public class FireMissilesDialogFragment extends DialogFragment implements DialogInterface.OnClickListener {

    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {

        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setMessage(R.string.fire_missiles)
                .setPositiveButton(R.string.fire, this)
                .setNegativeButton(android.R.string.cancel, this)
                .setNeutralButton(R.string.later, this);

        return builder.create();
    }
```

Adicionar os botões através do builder

AlertDialog

Tipos de botões

Existem três tipos de botões de ação que podemos adicionar:

- **Positivo:** Deve-se utilizar este botão para aceitar e continuar com a ação que foi apresentada (botão de "OK");
- **Negativo:** Deve-se utilizar este botão para cancelar a ação;
- **Neutro:** Deve-se utilizar este botão quando o utilizador não pretende continuar com a ação, mas também não pretende cancelar. Pode ser comparado a uma ação de "Remind me Later";

AlertDialog

Detetar clicks [FireMissilesDialogFragment.java]

```
@Override
public void onClick(DialogInterface dialog, int which) {
    String message = " ";
    switch(which){
        case DialogInterface.BUTTON_POSITIVE:
            message = "!! MISSILES LAUNCHED !!";
            break;
        case DialogInterface.BUTTON_NEGATIVE:
            message = "!! YOU GOT LUCKY TODAY !!";
            break;
        case DialogInterface.BUTTON_NEUTRAL:
            message = "Just kidding .... :D";
            break;
    }
    Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
}
```

AlertDialog com lista

Adicionar lista

Existem três tipos de listas em AlertDialogs:

1. Uma lista tradicional de escolha única;
2. Uma lista persistente de uma escolha única (radio buttons);
3. Uma lista persistente de múltipla escolha (checkboxes);

AlertDialog com lista

1. Lista tradicional de escolha única

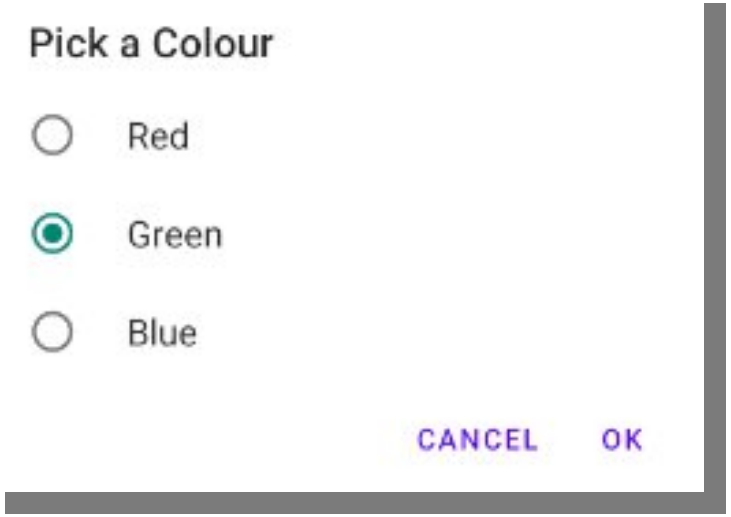
```
@NonNull
@Override
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {

    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    builder.setTitle(R.string.pick_colour)
            .setItems(R.array.colours_array, this);

    return builder.create();
}
```


```
// The 'which' argument contains the index position
// of the selected item
@Override
public void onClick(DialogInterface dialog, int which) {

}
```

AlertDialog com lista

2. Lista persistente de uma escolha única (radio buttons)

```
@NonNull
@Override
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {

    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

    builder.setTitle(R.string.pick_colour)
            .setSingleChoiceItems(R.array.colours_array, -1, this)
            .setPositiveButton(android.R.string.ok, this)
            .setNegativeButton(android.R.string.cancel,this);

    return builder.create();
}


// The 'which' argument contains the index position
// of the selected item
@Override
public void onClick(DialogInterface dialog, int which) {

}
```



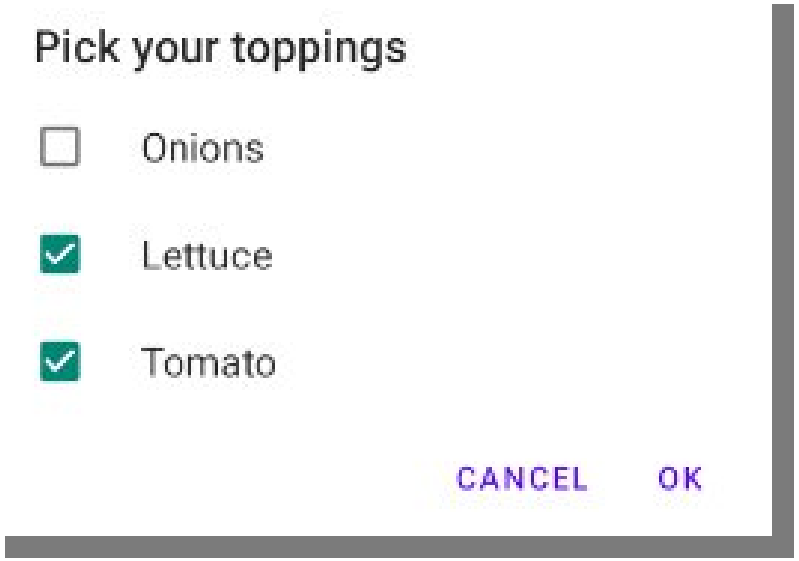
AlertDialog com lista

3. Lista persistente de múltipla escolha (checkboxes)

Implementa o OnMultiChoiceClickListener para detetar os items que são selecionados

```
public class ListDialogFragment extends DialogFragment
                            implements DialogInterface.OnClickListener, DialogInterface.OnMultiChoiceClickListener {
```

```
@NonNull
@Override
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {

    selectedItems = new ArrayList(); // Variável para guardar os items selecionados

    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

    builder.setTitle(R.string.pick_toppings)
            .setMultiChoiceItems(R.array.toppings_array, null, this)
            .setPositiveButton(android.R.string.ok, this)
            .setNegativeButton(android.R.string.cancel,this);

    return builder.create();
}
```



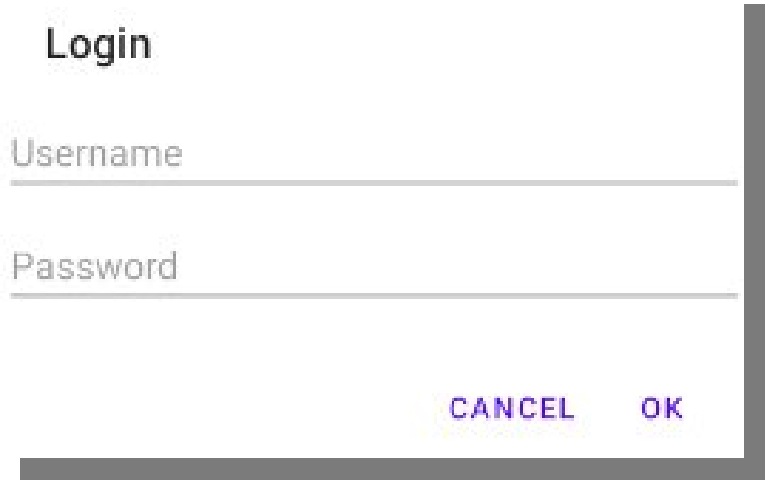
AlertDialog com lista

3. Lista persistente de múltipla escolha (checkboxes) (2)

```java
// Método onClick para os botões
@Override
public void onClick(DialogInterface dialog, int which) {
    //TODO: Implementar comportamento dos botões OK e Cancel
}

// Método onClick para os items da lista
@Override
public void onClick(DialogInterface dialog, int which, boolean isChecked) {
    if(isChecked){
        selectedItems.add(which); // Se o item foi selecionado é adicionado
    }else if(selectedItems.contains(which)){
        selectedItems.remove(Integer.valueOf(which)); //Se o item já tiver no array, é removido
    }
}
```

AlertDialog

Layout Personalisado



Dialog Personalizado

dialog_login.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_gravity="center">

    <EditText
        android:id="@+id/username"
        android:inputType="textEmailAddress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/username" />

    <EditText
        android:id="@+id/password"
        android:inputType="textPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fontFamily="sans-serif"
        android:hint="@string/password" />

</LinearLayout>
```

Dialog Personalizado

LoginDialogFragment.java

```
@NonNull
@Override
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {

    LayoutInflater inflater = requireActivity().getLayoutInflater();

    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

    return builder.setTitle(R.string.dlg_title)
            .setView(inflater.inflate(R.layout.dialog_login, null))
            .setPositiveButton(android.R.string.ok, this)
            .setNegativeButton(android.R.string.cancel, null)
            .create();
}
```

Inflate do layout personalizado do nosso Dialog

Processar eventos do Dialog na Activity

LoginDialogFragment.java

```
public interface NoticeDialogListener {
    void onDialogPositiveClick(DialogFragment dialog);
    void onDialogNegativeClick(DialogFragment dialog);
}

private NoticeDialogListener listener;


@Override
public void onAttach(@NonNull Context context) {
    super.onAttach(context);
    try {
        listener = (NoticeDialogListener) context;
    } catch (ClassCastException e) {
        throw new ClassCastException(getActivity().toString()
                + " must implement NoticeDialogListener");
    }
}
```

Processar eventos do Dialog na Activity

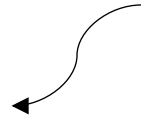
LoginDialogFragment.java

```
@Override
public void onClick(DialogInterface dialog, int which) {
    if(listener ≠ null){
        listener.onDialogPositiveClick(LoginDialogFragment.this);
        switch(which){
            case DialogInterface.BUTTON_POSITIVE:
                listener.onDialogPositiveClick(LoginDialogFragment.this);
                break;
            case DialogInterface.BUTTON_NEGATIVE:
                listener.onDialogNegativeClick(LoginDialogFragment.this);
                break;
        }
    }
}
```

Processar eventos do Dialog na Activity

MainActivity.java

Implementa a interface que definimos no DialogFragment



```
public class MainActivity extends AppCompatActivity
                  implements View.OnClickListener, LoginDialogFragment.NoticeDialogListener {



@Override
public void onDialogPositiveClick(DialogFragment dialog) {
    //TODO: Ação que acontece quando carregamos em OK
}

@Override
public void onDialogNegativeClick(DialogFragment dialog) {
    //TODO: Ação que acontece quando carregamos em Cancel
}
```

Leitura Adicional

- Dialogs
  https://developer.android.com/guide/topics/ui/dialogs

P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Programação Para Dispositivos Móveis I

DIALOGS

2023/_24 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

NORTE 2030 Programa Regional do Norte

PORTUGAL 2030

Cofinanciado pela União Europeia