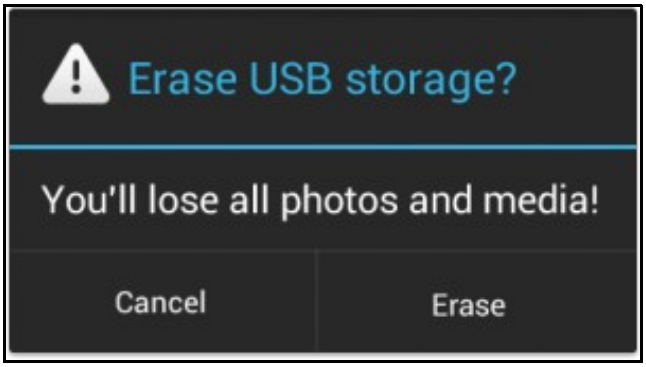
1 Ventás de Diálogos

1.1 Sumario

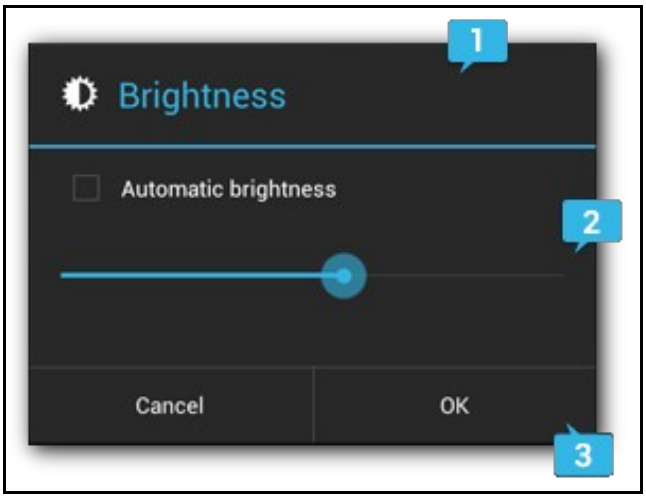
1.2 Introdución

- Un **diálogo** é unha ventá pequena que lle pregunta ao usuario para tomar unha decisión, para informalo, ou para que o usuario introduza información.
- Non cubre toda a pantalla
- É usado para que o usuario tome algún tipo de acción antes de seguir coa aplicación.



- Hai catro tipos de ventás de diálogo:
    - **AlertDialog**: pode conter de cero a tres botóns, unha lista, RadioButtons, CheckBoxes, etc.
    - **ProgressDialog**: amosa unha barra de progreso. Herda de da clase AlertDialog. Verase na Unidade 5.
    - **DatePickerDialog**: este diálogo é para seleccionar unha fecha.
    - **TimePickerDialog**: este diálogo permite seleccionar unha hora.

- Neste unidade imos centrarnos nos **AlertDialog**
    - Un diálogo baseado na clase AlertDialog ten 3 rexións:



- **1.-Título**: é opcional.
- **2.-Área de contido**:pode amosar unha mensaxe, lista, un layout personalizado, etc.
- **3.-Botóns de acción**: Non debe haber máis de tres botóns.
- Hai tres tipos de botóns:
    - **Positivo**: usarase para aceptar ou continuar unha acción.
    - **Negativo**: usarase cando se desexe cancelar unha acción.
    - **Neutral**: usarase cando non se sabe que facer coa acción.

- Para amosar un diálogo úsase o método **showDialog()** indicando un parámetro enteiro, único na actividade
- Imos construír os diálogos no método onCreateDialog(), que recibe un enteiro facendo referencia ao dialogo e devolve un obxecto de tipo diálogo.

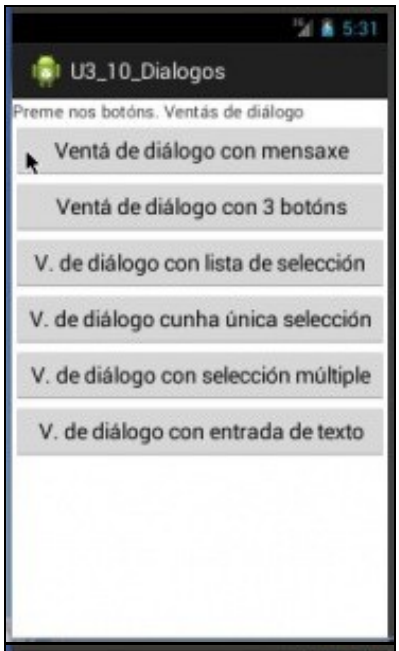
- **Referencias**:
    - http://developer.android.com/design/building-blocks/dialogs.html
    - http://developer.android.com/guide/topics/ui/dialogs.html

1.3 Caso práctico

- Creamos un novo proxecto: **U3_10_Dialogos**
- Esta aplicación está baseada no curso de Android da Aula Mentor: http://www.mentor.mec.es/

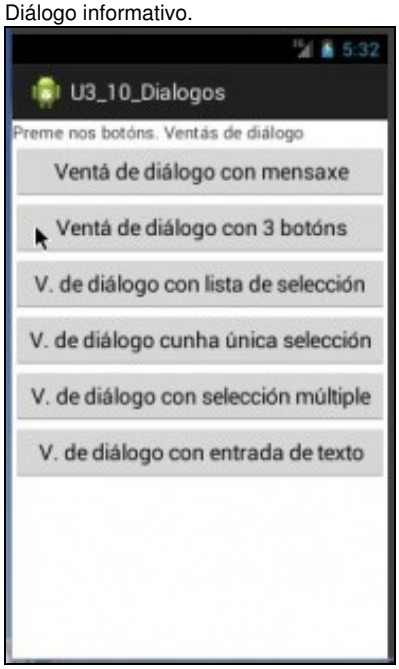
- As seguintes imaxes amosan o funcionamento da aplicación.
- Todos os elementos dos diálogos teñen asociado un Toast como acción, aínda que só se amose na última imaxe da seguinte secuencia.

- Aplicación Diálogos

- 

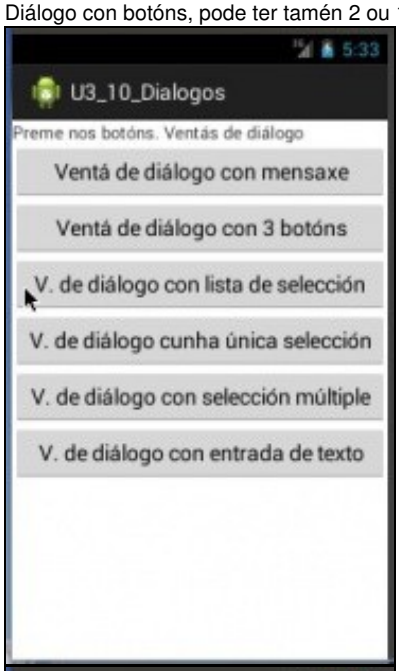
- 

Diálogo informativo.

- 

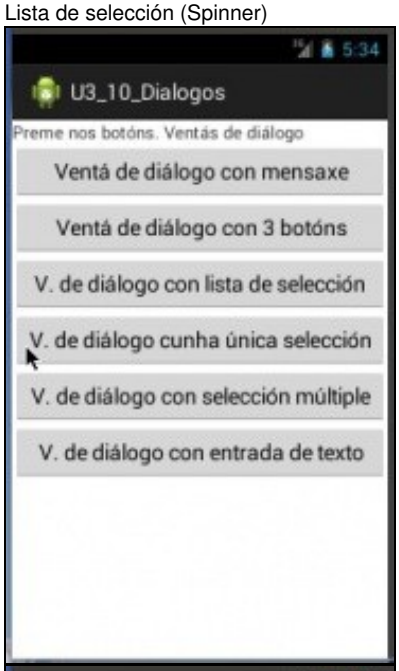
- 

Diálogo con botóns, pode ter tamén 2 ou 1 botón soamente.

- 

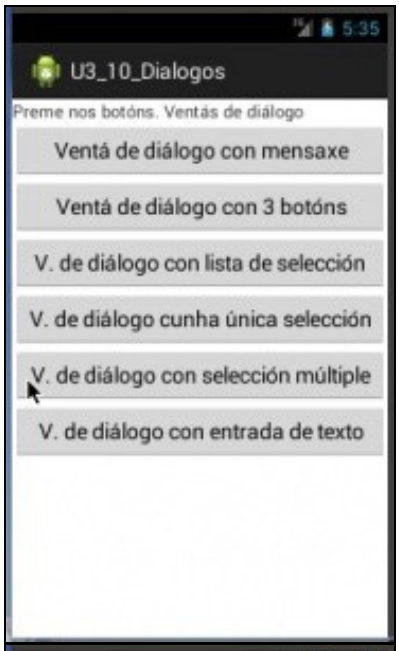
- 

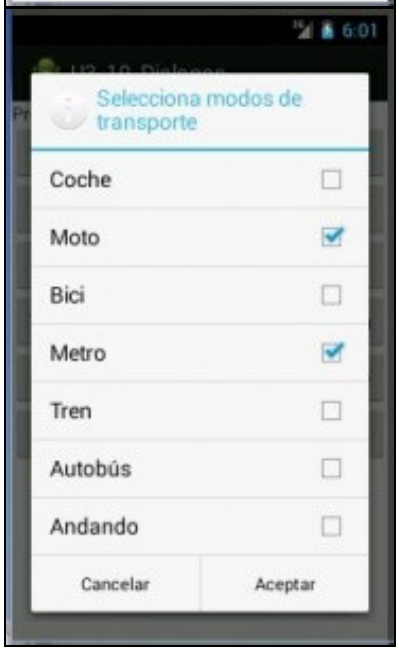
Lista de selección (Spinner)

- 

- 

RadioButtons

- 

- 

Múltiple selección con algúns ítems marcados por defecto.

- 

- 

Layout definido polo usuario para a entrada de información a través dun cadro de díálogo.

- 

1.3.1 O XML do Layout

- Todos Botóns executan o mesmo método para o evento Click.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Preme nos botóns. Ventás de diálogo" />

    <ScrollView
        android:id="@+id/screen"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
```

```
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >

            <Button
                android:id="@+id/btn_dialogo"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="onBotonClick"
                android:text="Ventá de diálogo con mensaxe" >
            </Button>

            <Button
                android:id="@+id/btn_diag_tres_botons"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="onBotonClick"
                android:text="Ventá de diálogo con 3 botóns" />

            <Button
                android:id="@+id/btn_diag_list_selecc"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="onBotonClick"
                android:text="V. de diálogo con lista de selección" />

            <Button
                android:id="@+id/btn_diag_radio_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="onBotonClick"
                android:text="V. de diálogo cunha única selección" />

            <Button
                android:id="@+id/btn_diag_checkbox"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="onBotonClick"
                android:text="V. de diálogo con selección múltiple" />

            <Button
                android:id="@+id/btn_diag_entrada_texto"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="onBotonClick"
                android:text="V. de diálogo con entrada de texto" />
        </LinearLayout>
    </ScrollView>

</LinearLayout>
```

1.3.2 Definición de recursos tipo array en XML

- Aproveitamos o ficheiro /**res**/**values**/**strings.xml** e definimos dous recursos de tipo Array, para a lista de selección e para a selección múltiple.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">U3_10_Dialogos</string>
    <string name="action_settings">Settings</string>


    <string-array name="elementos_dialog_seleccion">
        <item>Opción un</item>
        <item>Opción dous</item>
        <item>Opción tres</item>
        <item>Opción catro</item>
    </string-array>
```

```
    <string-array name="elementos_dialog_seleccion2">
        <item>Coche</item>
        <item>Moto</item>
        <item>Bici</item>
        <item>Metro</item>
        <item>Tren</item>
        <item>Autobús</item>
        <item>Andando</item>
    </string-array>

</resources>
```

1.3.3 Definición do recurso de tipo Layout usado polo diálogo con entrada de texto (o último)

Explicado posteriormente.

- Ficheiro /**res/layout/dialogo_entrada_texto.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="wrap_content"
    android:background="#000"
    android:orientation="vertical">

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:text="Nome"
        android:gravity="left"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/et_nome"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:scrollHorizontally="true"
        android:autoText="false"
        android:capitalize="none"
        android:gravity="fill_horizontal"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:text="Contrasinal"
        android:gravity="left"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/et_contrasinal"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:scrollHorizontally="true"
        android:gravity="fill_horizontal"
        android:password="true"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```

1.3.4 Código Java

```
package com.example.u3_10_dialogos;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.res.Resources;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class U3_10_Dialogos extends Activity {
private static final int DIALOGO_MENSAXE = 1;
private static final int DIALOGO_TRES_BOTONS = 2;
private static final int DIALOGO_LISTA = 3;
private static final int DIALOGO_RADIO_BUTTON = 4;
private static final int DIALOGO_CHECK_BOX = 5;
private static final int DIALOGO_ENTRADA_TEXTO = 6;

// Variable para crear as ventás de diálogo
AlertDialog.Builder venta;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_u3_10__dialogos);

}

protected Dialog onCreateDialog(int id) {
switch (id) {

case DIALOGO_MENSAXE:
venta = new AlertDialog.Builder(this);
venta.setTitle("Atención");
venta.setMessage("Nova amensaxe. Preme o botón 'Back' para volver á pantalla principal");
venta.setIcon(android.R.drawable.ic_dialog_email);
return venta.create();



case DIALOGO_TRES_BOTONS:
venta = new AlertDialog.Builder(this);
venta.setIcon(android.R.drawable.ic_dialog_info);
venta.setTitle("Enquisa");
venta.setMessage("Compras sempre en grandes superficies?");
venta.setCancelable(false);
venta.setPositiveButton("Si", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int boton) {
/* Sentencias se o usuario preme Si */
Toast.makeText(getApplicationContext(), "Premeches 'Si'", 1).show();
}
});
venta.setNegativeButton("Non", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int boton) {
/* Sentencias se o usuario preme Non */
Toast.makeText(getApplicationContext(), "Premeches'Non'", 1).show();
}
});
venta.setNeutralButton("Ás veces", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int boton) {
/* Sentencias se o usuario preme Ás veces */
Toast.makeText(getApplicationContext(), "Premeches 'Ás veces'", 1).show();
}
```

```
});
return venta.create();



case DIALOGO_LISTA:
venta = new AlertDialog.Builder(this);
venta.setIcon(android.R.drawable.ic_dialog_alert);
venta.setTitle("Escolle unha opción");
venta.setItems(R.array.elementos_dialog_seleccion, new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int opcion) {
// O usuario selecciona unha das opcións do listado
String[] opcions = getResources().getStringArray(R.array.elementos_dialog_seleccion);
Toast.makeText(getApplicationContext(), "Seleccionaches: '" + opcions[opcion] + "'", 1).show();
}
});
return venta.create();



case DIALOGO_RADIO_BUTTON:
venta = new AlertDialog.Builder(this);
venta.setIcon(android.R.drawable.ic_dialog_info);
venta.setTitle("Selecciona un smartpohone");
// Non incluír mensaxe dentro de este tipo de diálogo!!!
final CharSequence[] smartphones = { "iPhone", "Blackberry", "Android" };
venta.setSingleChoiceItems(smartphones, 0, new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int opcion) {
// Evento que ocorre cando o usuario selecciona una opción
Toast.makeText(getApplicationContext(), "Seleccionaches: " + smartphones[opcion], Toast.LENGTH_SHORT).show();
}
});
venta.setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int boton) {
Toast.makeText(getApplicationContext(), "Premeches 'Aceptar'", 1).show();
}
});
venta.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int boton) {
Toast.makeText(getApplicationContext(), "Premeches 'Cancelar'", 1).show();
}
});
return venta.create();



case DIALOGO_CHECK_BOX:
venta = new AlertDialog.Builder(this);
venta.setIcon(android.R.drawable.ic_dialog_info);
venta.setTitle("Selecciona modos de transporte");
Resources res = getResources();
final String[] matriz = res.getStringArray(R.array.elementos_dialog_seleccion2);
// Non incluír mensaxe dentro de este tipo de diálogo!!!
venta.setMultiChoiceItems(matriz, new boolean[] { false, true, false, true, false, false, false }, new DialogInterface.OnMultiChoiceC
public void onClick(DialogInterface dialog, int opcion, boolean isChecked) {
// Evento que ocorre cando o usuario selecciona unha opción
if (isChecked)
Toast.makeText(getApplicationContext(), "Seleccionaches " + matriz[opcion], Toast.LENGTH_SHORT).show();
else
Toast.makeText(getApplicationContext(), "Deseleccionaches " + matriz[opcion], Toast.LENGTH_SHORT).show();
}
});
venta.setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int boton) {
Toast.makeText(getApplicationContext(), "Premches 'Aceptar'", 1).show();
}
});
venta.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int boton) {
Toast.makeText(getApplicationContext(), "Premeches 'Cancelar'", 1).show();
}
});
return venta.create();
```

```
case DIALOGO_ENTRADA_TEXTO:
// Primeiro preparamos o interior da ventá de diálogo inflando o seu
// fichero XML
String infService = Context.LAYOUT_INFLATER_SERVICE;
LayoutInflater li = (LayoutInflater) getApplicationContext().getSystemService(infService);
// Inflamos o compoñente composto definido no XML
View inflador = li.inflate(R.layout.dialogo_entrada_texto, null);
// Buscamos os compoñentes dentro do Diálogo
final TextView etNome = (TextView) inflador.findViewById(R.id.et_nome);
final TextView etContrasinal = (TextView) inflador.findViewById(R.id.et_contrasinal);

venta = new AlertDialog.Builder(this);
venta.setTitle("Indica usuario e contrasinal");
// Asignamos o contido dentro do diálogo (o que inflamos antes)
venta.setView(inflador);
// Non se pode incluír unha mensaxe dentro deste tipo de diálogo!!!
venta.setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int boton) {
Toast.makeText(getApplicationContext(), "Escribiches nome: '" + etNome.getText().toString() + "'. Contrasinal: '" + etContrasinal.get
1).show();
}
});
venta.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int boton) {
Toast.makeText(getApplicationContext(), "Premeches en 'Cancelar'", 1).show();
}
});
return venta.create();

}
return null;
}



public void onBotonClick(View view) {

switch (view.getId()) {
case R.id.btn_dialogo:
showDialog(DIALOGO_MENSAXE);
break;

case R.id.btn_diag_tres_botons:
showDialog(DIALOGO_TRES_BOTONS);

break;

case R.id.btn_diag_list_selecc:
showDialog(DIALOGO_LISTA);

break;

case R.id.btn_diag_radio_button:
showDialog(DIALOGO_RADIO_BUTTON);

break;

case R.id.btn_diag_checkbox:
showDialog(DIALOGO_CHECK_BOX);

break;

case R.id.btn_diag_entrada_texto:
showDialog(DIALOGO_ENTRADA_TEXTO);

break;

                default:
break;
}
```

```
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.u3_10__dialogos, menu);
return true;
}

}
```

- **Liñas 20-25**: Creamos constantes enteiras para cada tipo de dialogo que queremos crear.

- Métodos máis comúns para todos os tipos de diálogos:
    - **setTitle()**: establece o título da ventá de diálogo.
    - **setMessage()**: pon a mensaxe na área de contido da ventá de diálogo.
    - **setIcon()**:establece a propiedade Icon cunha das imaxes predefinidas.

- No caso dos cadros de diálogo que teñan botóns:
    - Indicamos para cada tipo de botón o Texto que debe ver o usuario
    - Indicamos a acción a realizar se se preme. Neste caso chámase ao Listener de DialogInterface asociado ao evento Click.

- No método **onBotonClick(View view)** achamos que botón foi o que se pulsou e en función diso chamamos ao método showDialog() pasándolle a constante enteira asociada ao diálogo que queremos crear.
- No método onCreateDialog() temos asociado un diálogo para cada unha das constantes enteiras. O método recibe o enteiro e crear o diálogo asociado a ese número enteiro.
- Para cada novo diálogo crease un obxecto **venta** co construtor **AlertDialog.Builder()**

1.3.5 Ventás de diálogo personalizadas

- O último diálogo (**Liñas 148-179**) ten asociado un Layout XML. Ficheiro /**res/layout/dialogo_entrada_texto.xml** (xa posto anteriormente).
- Para visualizalo, no canto de utilizar **setMessage()** usamos o método **setView()** (liña 162 do código da activity) para amosar na área de contido do diálogo o XML inflado;
- Ese ficheiro XML hai que instancialo nas súas correspondentes Vistas.
- Ese proceso coñecese co nome de **Inflar** e é necesario facelo para poder visualizar o recurso xml.
- Podemos ver como é dito proceso no código:

```
case DIALOGO_ENTRADA_TEXTO:
        // Primeiro preparamos o interior da ventá de diálogo inflando o seu
        // fichero XML
        String infService = Context.LAYOUT_INFLATER_SERVICE;
        LayoutInflater li = (LayoutInflater) getApplicationContext().getSystemService(infService);
        // Inflamos o compoñente composto definido no XML
        View inflador = li.inflate(R.layout.dialogo_entrada_texto, null);
        .............
        venta.setView(inflador);
```

Neste caso estamos a facer un 'inflate' explícito, pero aínda que non vos dades conta, tamén se fai un 'inflate' cando chamamos o método setContentView dentro do onCreate da activity. Lembrar que o que pasamos como dato nesa chamada é o layout que vai visualizar a activity. Isto se coñece como 'inflate' implícito e o fai o S.O. automaticamente.

- Para nos (aínda que non sexa exactamente iso) o 'inflado' serve para pasar dun recurso de texto xml a un obxecto (View) onde se atopan todos os elementos gráficos que están definidos no arquivo xml. É importante sinalar que cando queiramos acceder a un compoñente do diálogo (por exemplo, no xogo das preguntas, cando engadimos unha pregunta nova aparece un diálogo deste tipo. Ao dar de alta a

pregunta leva consigo acceder as caixas de texto) o temos que facer a través do view que obtemos do 'inflado'. Se isto o facemos dende dentro dunha clase anónima, teremos que definir o View como final.
No exemplo, o que se define como final é o campo de texto (liñas 156 e 157), pero poderíamos definir como final o View da forma:

```
case DIALOGO_ENTRADA_TEXTO:
                        // Primeiro preparamos o interior da ventá de diálogo inflando o seu
                        // fichero XML
                        String infService = Context.LAYOUT_INFLATER_SERVICE;
                        LayoutInflater li = (LayoutInflater) getApplicationContext().getSystemService(infService);
                        // Inflamos o compoñente composto definido no XML
                        final View inflador = li.inflate(R.layout.dialogo_entrada_texto, null);

                        venta = new AlertDialog.Builder(this);
                        venta.setTitle("Indica usuario e contrasinal");
                        // Asignamos o contido dentro do diálogo (o que inflamos antes)
                        venta.setView(inflador);
                        // Non se pode incluír unha mensaxe dentro deste tipo de diálogo!!!
                        venta.setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int boton) {
                                        // Buscamos os compoñentes dentro do Diálogo
                                        TextView etNome = (TextView) inflador.findViewById(R.id.et_nome);
                                        TextView etContrasinal = (TextView) inflador.findViewById(R.id.et_contrasinal);

                                        Toast.makeText(getApplicationContext(), "Escribiches nome: '" + etNome.getText().toString() +
                                                        1).show();
                                }
                        });
```

- No inflate o segundo parámetro:

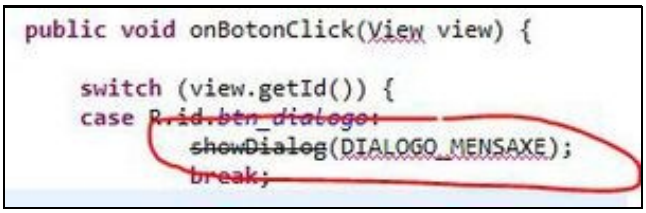
    View inflador = li.inflate(R.layout.dialogo_entrada_texto, **null**);)

normalmente ponse null, pero podemos poñer un view que queiramos que sexa 'pai' do view que se xera a partires do arquivo de recursos xml.

1.4 DialogFragment

A partires da API 13 (Android 3.2) a forma de 'construír' os diálogos modificouse.

Se se usa a forma anteriormente explicada funcionará pero marcará no Compilador a liña como **deprecated**.



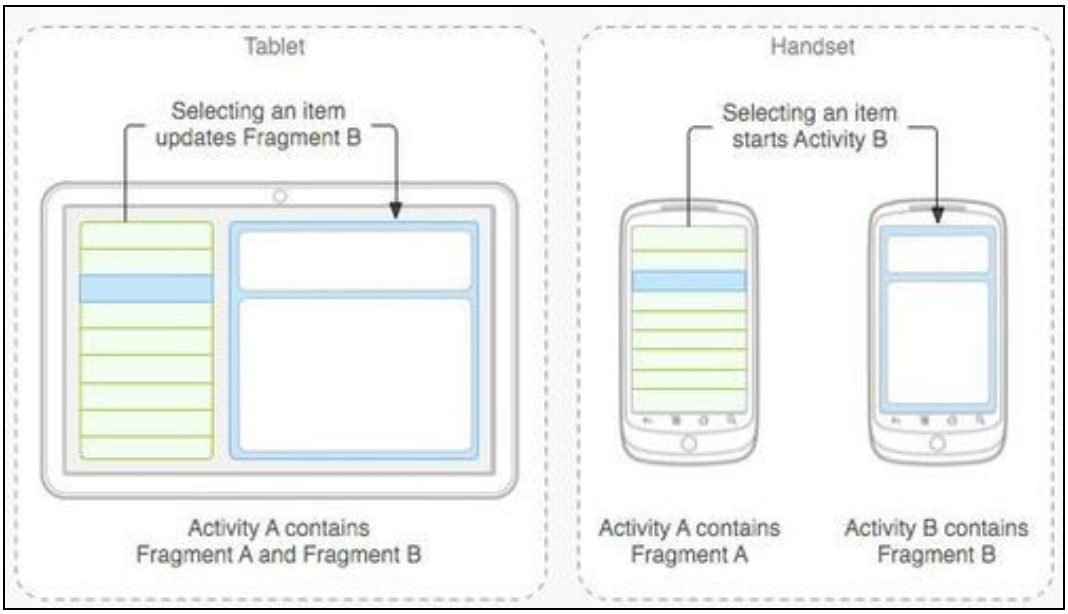
Imos ver neste punto como poderíamos facer para construír os diálogos doutra forma.

Para explicalo teremos que falar antes dos DialogFragment e por extensión dos Fragment.

Información adicional:

- DialogFragment
- Fragment
- http://www.sgoliver.net/blog/fragments-en-android/ (En español).

Un fragment representa un 'trozo' da interface dun usuario. A idea dos fragment xurdiu polas dificultades que tiñan as aplicacións a adaptarse a tamaños grandes de pantalla (como as tablet´s).



Imaxe obtida de [1]

Imaxinade que tedes unha aplicación que amosa unha lista de usuarios e que ó premer sobre un deles apareceran os seus datos completos.

Isto feito nun móbil cunha pantalla de 4 polgadas levaría consigo a creación de dúas activities, xa que na mesma non cabe toda a información (necesitaríamos usar scroll, bastante incómodo).

Esta mesma aplicación levada a unha tablet de 10 polgadas non tería problema en visualizar todo na mesma pantalla.

Os fragment vannos permitir dividir os elementos gráficos en 'anacos' que imos poder usar de forma independente nas activities. Así, no caso anterior podemos crear dous fragment, un coa lista e outro có detalle de cada usuario.

Agora dependendo do tamaño da pantalla podo facer que na pantalla pequena se amose un só fragment (a lista) é que o premer sobre un elemento da mesma apareza o outro fragment (o detalle).
Se o tamaño é grande (tablet) podo facer que aparezan os dous fragment na mesma activity.

O código que xestiona a pulsación dun elemento da lista e amosar os datos estará definido nun só sitio, o que modificamos é o sitio onde se visualiza o fragment en función do tamaño da pantalla.

Tendo claro o concepto de Fragment isto lévanos ao seguinte concepto: **DialogFragment**.

1.4.1 Caso práctico

- Un DialogFragment é un Fragment que amosa unha ventá de diálogo e situase por enriba da nosa Activity.

Os DialogFragment poden ser utilizados en versións anteriores á API 13, pero nese caso teremos que utilizar unha biblioteca de compatibilidade que nos proporciona Eclipse:

Polo tanto se temos un Min SDK no android manifiesto menor á API 13 teremos que importar a clase DialogFragment de dita librería:

```
import android.support.v4.app.DialogFragment;
```

Un DialogFragment vai ter un layout asociado que vai compoñer o seu contido.

Definimos polo tanto o contido do noso diálogo.

1.4.1.1 O XML do DialogFragment

Código da clase fragment_u3_15_layout_dialogo

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

     <EditText
           android:id="@+id/editTexto"
           android:layout_height="wrap_content"
           android:layout_width="match_parent"
           android:inputType="text"
           android:gravity="fill_horizontal"
     />
    <Button
        android:id="@+id/buttonPecharDialogo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editTexto"
        android:layout_centerInParent="true"
        android:text="Pechar" />

</RelativeLayout>
```

Agora deberemos crear unha clase que vai representar o DialogFragment e que vai cargar o layout anterior. Esta clase será a que instanciemos dende a nosa Activity.

1.4.1.2 O código Java do DialogFragment

Código da clase UD3_15_DialogoFragmento
Obxectivo: Clase que representa ó DialogFragment

```
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

public class UD3_15_DialogoFragmento extends DialogFragment{

public String valorTexto;

@Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

final View rootView = inflater.inflate(R.layout.fragment_u3_15_layout_dialogo, container, false);
        getDialog().setTitle(getTag());        // O Tag se envía dende a activiy có método show.

        Button btn = (Button) rootView.findViewById(R.id.buttonPecharDialogo);
        btn.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub
EditText edit = (EditText)rootView.findViewById(R.id.editTexto);
valorTexto = edit.getText().toString();
((U3_15_Dialogos)UD3_15_DialogoFragmento.this.getActivity()).pecharDialogo();
dismiss();
}
});

        // Do something else
        return rootView;
    }
}
```

Analicemos o código.

- Liña 2: Estamos a utilizar a librería de compatibilidade con versións anteriores de Android.
- Liña 10: A nosa clase deriva de DialogFragment.
- Liña 15: Ó derivar de DialogFragment sobreescribimos o método onCreateView que ten que devolver o View que conforma o diálogo.

  No noso caso deberemos 'inflar' o layout deseñado anteriormente.
  O obxecto da clase LayoutInflater permite pasar dun deseño gráfico (o layout) e un obxecto manexable por programación. Isto o fai chamando ó método inflate e o garda no obxecto rootView.

- Liña 21: Có paso anterior, podemos acceder ós compoñentes gráficos que se atopan no layout por programación. No noso caso accedemos ó botón pechar e xestionamos o evento de click.
- Liñas 25-31: O que facemos nestas liñas é acceder ó contido da caixa de texto do diálogo, o gardamos nunha propiedade da clase.
    - ◊ Liña 28: Esta liña vos dará un erro ata que usedes o código que ven a continuación. Esta liña chama a un método definido na activity que lanzou o DialogFragment.
    - ◊ Liña 29: Pechamos o diálogo.

1.4.1.3 O XML da Activity

Definimos o layout da nosa activity:

Código da clase activity_u3_15__dialogos.xml
Obxectivo: Layout da nosa activity.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:orientation="vertical" >

        <TextView
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:text="Ventás de diálogo" />


        <Button
        android:id="@+id/btn_dialogo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onBotonClick"
        android:text="Ventá de diálogo con fragment" >
        </Button>

    </LinearLayout>
```

1.4.1.4 O código Java da Activity

Agora implantaremos a activity que vai facer uso do DialogFragment.

Aquí teremos dúas opcións:

- Derivar a clase de Activity. Deberemos facer uso do método getFragmentManager().
- Derivar a clase de FragmentActivity. Esta clase é unha subclase de Activity e está posta por compatibilidade coas versións anteriores.

  No caso de utilizar esta clase deberemos facer uso do método getSupportFragmentManager. **No noso exemplo imos facer uso deste método.**

Código da clase U3_15_Dialogos
Obxectivo: Amosar un diálogo baseado en Fragment.

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.view.View;
import android.widget.Toast;

public class U3_15_Dialogos extends FragmentActivity {

private UD3_15_DialogoFragmento dialogoFragmento;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u3_15__dialogos);
        dialogoFragmento = new UD3_15_DialogoFragmento();
    }

    public void pecharDialogo(){
        Toast.makeText(this, dialogoFragmento.valorTexto, Toast.LENGTH_LONG).show();
    }

    public void onBotonClick(View view) {
        FragmentManager fm = getSupportFragmentManager();

        switch (view.getId()) {
        case R.id.btn_dialogo:
        dialogoFragmento.show(fm, "EXEMPLO DE DIALOGO!!!!");
                break;
```

```
        }
    }
}
```

Analicemos o código.

- Liñas 2,3,7: Por compatibilidade con versións anteriores de Android escollemos a opción de utilizar un FragmentActivity.
- Liña 9,15: Definimos e instaciamos o diálogo.
- Liñas 18-20: Este é o método que chama a clase do Diálogo cando prememos o botón pechar.
- Liña 23: Necesitamos un FragmentManager para amosar o diálogo.
- Liña 27: Amosamos o diálogo.

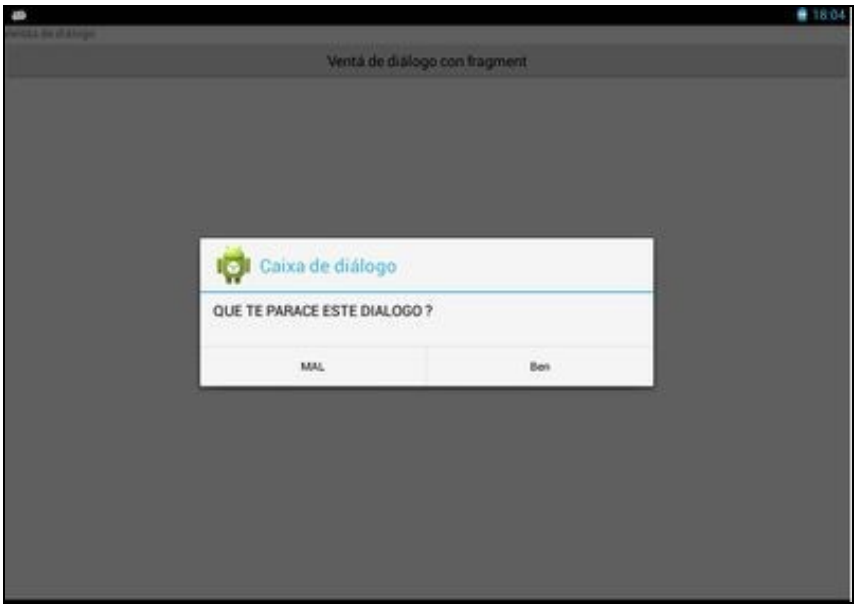
1.4.1.5 Unha variación

Esta é unha primeira aproximación os DialogFragment. Comentar que en vez de sobreescribir o método onCreateView do DialogFragment podemos sobreescribir o método onCreateDialog e devolver un diálogo construído por nos (coma un DatePickerDialog, AlertDialog.builder,....) Por exemplo:

Código da clase UD3_15_DialogoFragmento
Obxectivo: Modificamos o código para amosar outra forma de crear un diálogo. **Debemos comentar o código do método onCreateView.**

```
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
import android.widget.Toast;

public class UD3_15_DialogoFragmento extends DialogFragment{


@Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity())
.setTitle("Caixa de diálogo").setIcon(R.drawable.ic_launcher)
.setMessage("QUE TE PARACE ESTE DIALOGO ?")
.setPositiveButton("Ben", new OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
Toast.makeText(getActivity(), "PULSADA OPCION BOA",Toast.LENGTH_LONG).show();
}
}).setNegativeButton("MAL", new OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
//

Toast.makeText(getActivity(),"PULSADA OPCION MALA", Toast.LENGTH_LONG).show();
}
});
return builder.create();


}

}
```

Nota: Fixarse que estamos a importar a interface onClickListener do DialogInterface (liña 4).



-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2015).