

**MP_0489. Programación
multimedia y dispositivos móviles**

**UF2. Programación de
aplicaciones para dispositivos móviles**

2.4. Notificaciones

Índice

☰	Objetivos	3
☰	Toast	4
☰	Barras de estado	13
☰	Diálogos	19
☰	Diálogos de alerta, confirmación y selección	22
☰	Diálogos personalizados	32
☰	Resumen	36

Objetivos

Con esta unidad perseguimos los siguientes objetivos:

1

Conocer las distintas maneras de que disponemos para notificar acciones al usuario.

2

Descubrir la clase *Toast* para notificaciones rápidas.

3

Aprender a generar notificaciones en la barra de estado.

4

Conocer los diferentes diálogos que podemos mostrar al usuario.

¡Ánimo y adelante!

Toast

Un *toast* es un mensaje que se muestra en pantalla al usuario durante unos segundos para luego desaparecer automáticamente.

Aunque son personalizables, aparecen por defecto en la parte inferior de la pantalla, sobre un rectángulo gris translúcido. Estas notificaciones son **ideales para mostrar mensajes rápidos y sencillos al usuario**.

Usar la clase *Toast*

Su uso es muy sencillo, toda la funcionalidad está en la clase *Toast*.

Esta clase dispone de un método estático *makeText()* al que deberemos pasar como parámetro el contexto de la actividad, el texto a mostrar y la duración del mensaje. La duración puede tomar los valores **LENGTH_LONG** o **LENGTH_SHORT**, dependiendo del tiempo que queramos que la notificación aparezca en pantalla.

Tras obtener una referencia al objeto *Toast* a través de este método, ya solo nos quedaría mostrarlo en pantalla mediante el método *show()*.

Ejemplo 1. Cómo funciona un *toast*

Hagamos una aplicación de ejemplo para ver el funcionamiento de este tipo de notificaciones.

Tendrá un botón que muestre un *toast* básico:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout 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"
    tools:context=".MainActivity">

    <Button
        android:id="@+id(btnPorDefecto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:layout_editor_absoluteY="174dp" />

</android.support.constraint.ConstraintLayout>
```

activity_main.xml

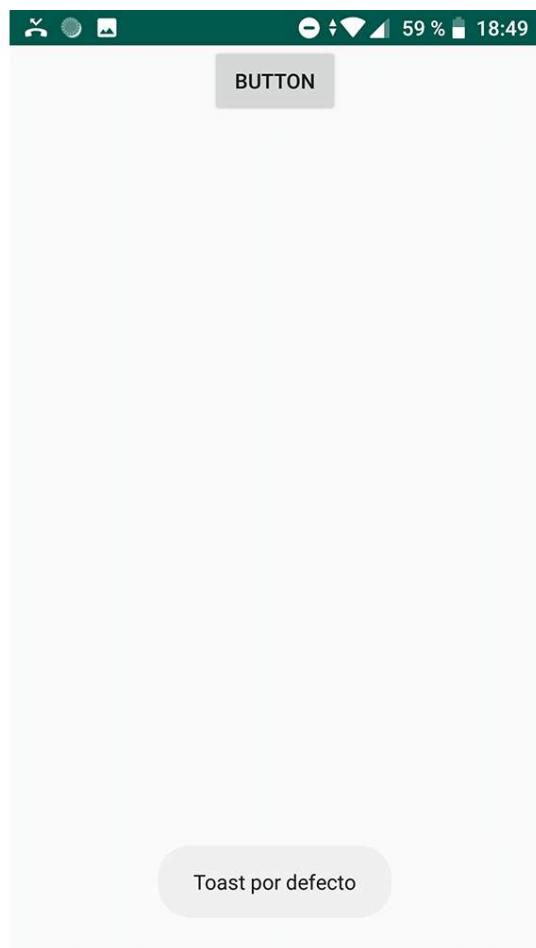
```
package com.miejemplo.notificaciones1;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
```

```
public class MainActivity extends Activity {  
    private Button btnDefecto = null;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        btnDefecto = (Button)  
            findViewById(R.id.btnPorDefecto);  
        btnDefecto.setOnClickListener(new View.OnClickListener()  
  
        {  
            @Override  
            public void onClick(View v) {  
                Toast toast1 =  
                    Toast.makeText(getApplicationContext(),  
                    "Toast por defecto", Toast.LENGTH_SHORT);  
  
                toast1.show();  
            }  
        });  
    }  
}
```

MainActivity.java

Si ejecutamos esta sencilla aplicación en el emulador y pulsamos el botón que acabamos de añadir, veremos cómo en la parte inferior de la pantalla aparece el mensaje ***Toast por defecto***, que desaparecerá automáticamente tras varios segundos.



Este es el comportamiento por defecto de las notificaciones *toast*. Sin embargo, podemos personalizarlo un poco cambiando su posición en la pantalla, como veremos a continuación.

Ejemplo 2. Cómo modificar la posición del *toast* en la pantalla

Para este nuevo ejemplo, en el que modificaremos la posición de la notificación *toast* en la pantalla, utilizaremos el método *setGravity()*.

Podremos indicar a este método en qué zona deseamos que aparezca la notificación con alguna de las constantes definidas en la clase *Gravity*:

- CENTER
- LEFT
- BOTTOM

1

Para empezar, vamos a colocar la notificación en la zona central izquierda de la pantalla. Para ello, modificamos nuestro código:

```
package com.miejemplo.notificaciones1;

import android.os.Bundle;
import android.app.Activity;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {
    private Button btnDefecto = null;

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

        btnDefecto = (Button)
            findViewById(R.id.btnPorDefecto);
        btnDefecto.setOnClickListener(new View.OnClickListener()

        {
            @Override
            public void onClick(View v) {
                Toast toast1 =
                    Toast.makeText(getApplicationContext(),
                        "Toast por defecto", Toast.LENGTH_SHORT);
                toast1.setGravity(Gravity.CENTER|Gravity.LEFT,0,0);
                toast1.show();
            }
        });
    }
}
```

MainActivity.java

2

Este es el **resultado** con el *toast* colocado en otra posición.



Ejemplo 3. Personalizar del todo el **toast**

En esta ocasión, personalizaremos por completo el aspecto de la notificación.

Si necesitamos **personalizar por completo el aspecto de la notificación**, Android nos ofrece la posibilidad de definir un *layout XML* propio para *toast*, donde podremos incluir todos los elementos necesarios para adaptar la notificación a nuestras necesidades.

1

Vamos a definir un *layout* sencillo, con una imagen y una etiqueta de texto sobre un rectángulo gris. Guardaremos este *layout* con el nombre “*toast_layout.xml*” y, como siempre, lo colocaremos en la carpeta “*res\layout*” de nuestro proyecto.

2

Además, añadiremos a la carpeta “*res\drawable*” la siguiente imagen:

Se trata de un *bullet* amarillo, que puedes ver a la izquierda en su tamaño real y descargar a continuación.



icono_toast.gif

1.1 KB



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

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/lytLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    android:background="#555555"
    android:padding="5dip" >

    <ImageView android:id="@+id/imgIcono"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:src="@drawable/icono_toast" />

    <TextView android:id="@+id/txtMensaje"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:textColor="#FFFFFF"
        android:paddingLeft="10dip" />

</LinearLayout>
```

toast_layout.xml

3

Para asignar este *layout* a nuestro *toast*, en primer lugar deberemos **inflar el layout mediante un objeto *LayoutInflater***.

4

Una vez construido el *layout*, **modificaremos los valores de los distintos controles para mostrar la información que queramos**. En nuestro caso, solo modificaremos el mensaje de la etiqueta de texto, puesto que la imagen ya la asignamos de forma estática en el *layout XML* mediante el atributo *android:src*.

5

Tras esto, solo nos quedará establecer la duración de la notificación con *setDuration()* y asignar el *layout* personalizado al *toast* mediante el método *setView()*.

Así quedaría todo el código:

```
package com.miejemplo.notificaciones1;

import android.os.Bundle;
import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {
    private Button btnDefecto = null;

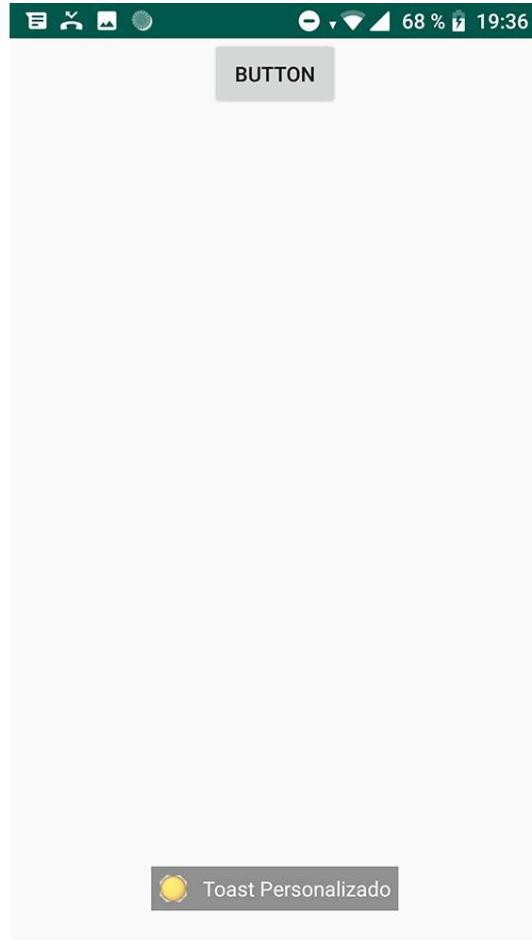
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnDefecto = (Button)
            findViewById(R.id.btnPorDefecto);

        btnDefecto.setOnClickListener(new OnClickListener() {
            @Override
```

```
public void onClick(View arg0) {  
    Toast toast2 = new Toast(getApplicationContext());  
    LayoutInflator inflater = getLayoutInflator();  
    View layout = inflater.inflate(R.layout.toast_layout,  
        (ViewGroup) findViewById(R.id.lytLayout));  
    TextView txtMsg = (TextView)layout.findViewById(R.id.txtMensa-  
je);  
    txtMsg.setText("Toast Personalizado");  
    toast2.setDuration(Toast.LENGTH_SHORT);  
    toast2.setView(layout);  
    toast2.show();  
}  
});  
  
}  
}
```

Si ejecutamos ahora la aplicación, veremos que **nuestro toast aparece con la estructura definida en el layout personalizado.**



Barras de estado

Las barras de estado se muestran en nuestro dispositivo cuando, por ejemplo, recibimos un mensaje SMS, o cuando tenemos actualizaciones disponibles.

Estas notificaciones constan de **un ícono y un texto mostrado en la barra de estado superior, y adicionalmente un mensaje más descriptivo y una marca de fecha/hora**, que podemos consultar desplegando la bandeja del sistema.

Veamos cómo utilizar este tipo de notificaciones en nuestras aplicaciones.

Vamos a hacer un único botón que genere una notificación de ejemplo en la barra de estado.

```
<LinearLayout 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"  
    android:paddingBottom="16dp"  
    android:paddingLeft="16dp"  
    android:paddingRight="16dp"  
    android:paddingTop="16dp"  
    tools:context=".MainActivity" >
```

```
<Button android:id="@+id/BtnNotif"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Notificación" />  
  
</LinearLayout>
```

Para generar notificaciones en la barra de estado del sistema utilizamos la clase ***NotificationCompat.Builder***. Lo que haremos será crear un nuevo objeto de este tipo, pasándole el contexto de la aplicación, y **asignaremos todas las propiedades que queramos mediante sus métodos *set()***.

- ***setLargeIcon()***

Con este método establecemos el icono a mostrar a la derecha del contenido.

- ***setSmallIcon()***

Establecemos el icono a mostrar a la izquierda del contenido. En versiones más antiguas, tan solo se mostrará el icono pequeño a la izquierda de la notificación. Además, este icono también se mostrará en la barra de estado superior.

- ***setContentTitle()***

Establecemos el título de la notificación.

- ***setContentText()***

Establecemos el texto de la notificación.

- ***setTicker()***

Establecemos el *ticker*, que es el texto que aparece por unos segundos en la barra de estado al generarse una nueva notificación.

- ***setContentInfo()***

Establecemos el texto auxiliar (opcional) que aparecerá a la izquierda del icono pequeño de la notificación.

- ***setWhen()***

La fecha/hora asociada a nuestra notificación se tomará automáticamente de la fecha/hora actual si no se establece nada, o bien emplearemos este método para indicar otra marca de tiempo.

Pongamos en práctica todo lo que hemos visto hasta ahora con un ejemplo.

```
package com.miejemplo.notificaciones2;

import android.os.Bundle;
import android.app.Activity;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.BitmapFactory;
import android.support.v4.app.NotificationCompat;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {

    private Button btnNotificacion;

    private static final int NOTIF_ALERTA_ID = 1;

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

        btnNotificacion = (Button)findViewById(R.id.BtnNotif);

        btnNotificacion.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                NotificationCompat.Builder mBuilder =
                new NotificationCompat.Builder(MainActivity.this)
                    .setSmallIcon(R.drawable.ic_launcher)
                    .setContentTitle("Mensaje de Alerta")
                    .setContentText("Ejemplo de notificación.");
            }
        });
    }
}
```

1

Debemos construir un objeto *PendingIntent*, que será el que contenga la información de la actividad asociada a la notificación y que será lanzado al pulsar sobre ella.

2

Para ello, definimos un *intent*, indicando la clase de la actividad concreta a lanzar, que en nuestro caso será la propia actividad principal de ejemplo (*MainActivity.class*). Este *intent* lo utilizaremos para construir el *PendingIntent* final mediante el método *PendingIntent.getActivity()*.

3

Una vez hecho, asociaremos este objeto a la notificación mediante el método *setContentIntent()* del *Builder*.

```
package com.miejemplo.notificaciones2;

import android.os.Bundle;
import android.app.Activity;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.drawable.BitmapDrawable;
import android.support.v4.app.NotificationCompat;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {

    private Button btnNotificacion;

    private static final int NOTIF_ALERTA_ID = 1;

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

        btnNotificacion = (Button)findViewById(R.id.BtnNotif);

        btnNotificacion.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                NotificationCompat.Builder mBuilder =
                new NotificationCompat.Builder(MainActivity.this)
                    .setSmallIcon(R.drawable.ic_launcher)
                    .setContentTitle("Mensaje de Alerta")
                    .setContentText("Ejemplo de notificación.");
            }
        });
    }

    Intent notIntent =
        new Intent(MainActivity.this, MainActivity.class);

    PendingIntent contIntent = PendingIntent.getActivity(
        MainActivity.this, 0, notIntent, 0);

    mBuilder.setContentIntent(contIntent);

    });

}

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

4

Para finalizar, podemos generar nuestra notificación llamando al método `notify()` del *Notification Manager*, al que se accede mediante una llamada a `getSystemService()` con la constante `Context.NOTIFICATION_SERVICE`.

5

Por su parte, le pasaremos como parámetro un identificador único al método `notify()`, definido por nosotros, que identifique nuestra notificación y el resultado del *builder* que hemos construido antes, y que obtenemos llamando a su método `build()`.

```
package com.miejemplo.notificaciones2;

import android.os.Bundle;
import android.app.Activity;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.drawable.BitmapDrawable;
import android.support.v4.app.NotificationCompat;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {

    private Button btnNotificacion;

    private static final int NOTIF_ALERTA_ID = 1;

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

        btnNotificacion = (Button)findViewById(R.id.BtnNotif);

        btnNotificacion.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                NotificationCompat.Builder mBuilder =
                new NotificationCompat.Builder(MainActivity.this)
                    .setSmallIcon(R.drawable.ic_launcher)
                    .setContentTitle("Mensaje de Alerta")
                    .setContentText("Ejemplo de notificación.");
            }
        });
    }

    Intent notIntent =
        new Intent(MainActivity.this, MainActivity.class);

    PendingIntent contIntent = PendingIntent.getActivity(
        MainActivity.this, 0, notIntent, 0);

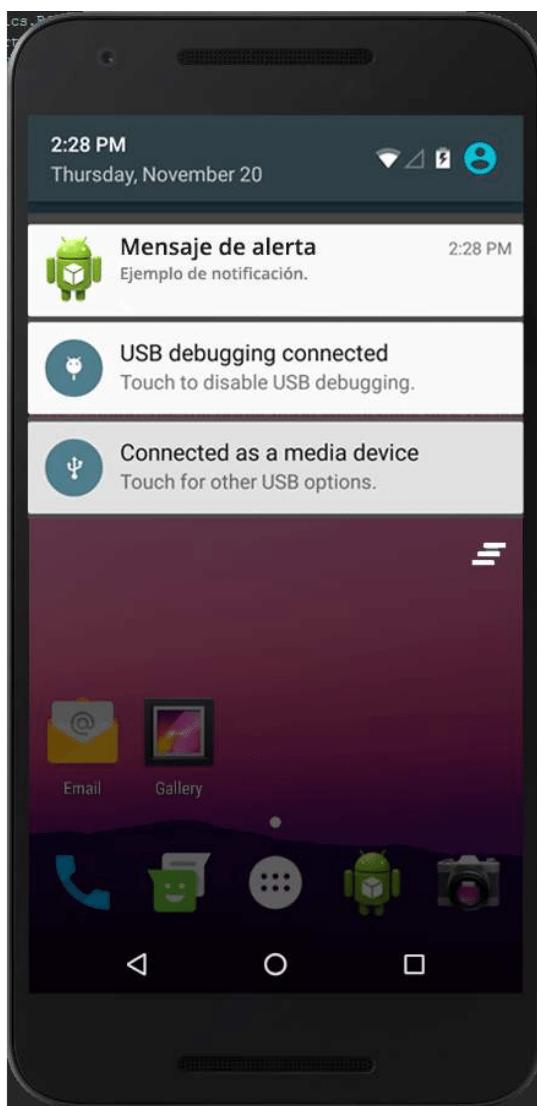
    mBuilder.setContentIntent(contIntent);

    NotificationManager mNotificationManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_
SERVICE);
```

```
        mNotificationManager.notify(NOTIF_ALERTA_ID, mBuilder.build());
    }
}

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

Este sería el resultado, visto en el emulador:



Diálogos

Podemos utilizar los diálogos de Android con distintos fines.

Por ejemplo:

- Mostrar un mensaje.
- Pedir una confirmación rápida.
- Solicitar al usuario una elección entre varias alternativas.

Para crear un diálogo tenemos que **crear una nueva clase que herede de *DialogFragment* y sobrescribir uno de sus métodos *onCreateDialog()***, que será el encargado de construir el diálogo.



¡Ojo! La forma de construir cada diálogo dependerá de la información y funcionalidad que vayamos a necesitar.

Practiquemos lo aprendido con un ejemplo.

Vamos a crear una aplicación con cuatro botones, cada uno de los cuales lanzará un diálogo distinto.

1

Para ello, creamos un nuevo proyecto, que tendrá el siguiente `activity_main.xml`:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <Button
        android:id="@+id/BtnAlerta"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cuadro de diálogo: Alerta" />

    <Button
        android:id="@+id/BtnConfirmacion"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cuadro de diálogo: Confirmación" />

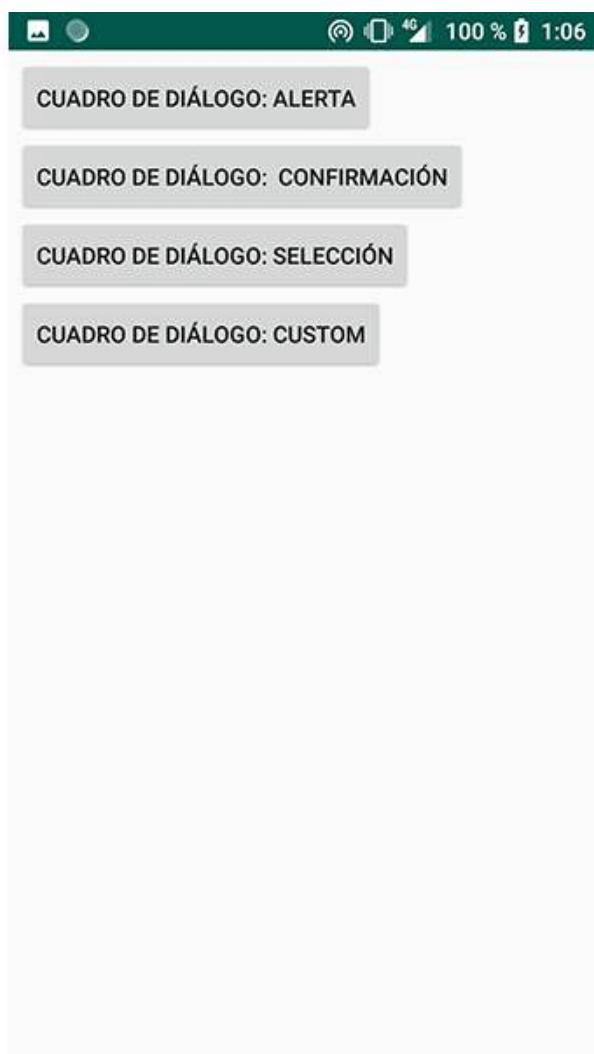
    <Button
        android:id="@+id/BtnSeleccion"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cuadro de diálogo: Selección" />

    <Button
        android:id="@+id/BtnCustom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cuadro de diálogo: Custom" />

</LinearLayout>
```

2

Como resultado, obtendremos la siguiente pantalla en nuestra aplicación:



Diálogos de alerta, confirmación y selección

Existen distintos **tipos de diálogos Android**.

- Diálogo de alerta.
- Diálogo de confirmación.
- Diálogo de selección.
- Diálogos personalizados

Diálogo de Alerta

Este tipo de diálogo muestra un mensaje sencillo y un botón de OK para confirmar su lectura. Lo construiremos mediante la clase *AlertDialog*.

Su utilización es muy sencilla; bastará con crear un objeto de tipo *AlertDialog.Builder* y establecer las propiedades del diálogo mediante sus métodos correspondientes: título [`setTitle()`], mensaje [`setMessage()`], y el texto y comportamiento del botón [`setPositiveButton()`].

1

Para ello, creamos una clase que llamaremos *DialogoAlerta* y que guardaremos como *DialogoAlerta.java*.

2

Al método *setPositiveButton()* le pasaremos como argumentos el texto a mostrar en el botón y la implementación del evento *onClick* en forma de objeto *OnClickListener*.

3

Después, nos limitaremos a cerrar el diálogo mediante su método *cancel()*.

```
package com.miejemplo.notificaciones3;

import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;

public class DialogoAlerta extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {

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

        builder.setMessage("Mensaje de tipo alerta.")
            .setTitle("Información")
            .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    dialog.cancel();
                }
            });

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

DialogoAlerta.java

4

Lanzamos el diálogo desde *MainActivity.java*, mediante una llamada a *getSupportFragmentManager()*.

5

Creamos un nuevo objeto de tipo *DialogoAlerta* y mostramos el diálogo mediante *show()*, pasándole la referencia al *fragment manager* y una etiqueta identificativa del diálogo.

```
package com.miejemplo.notificaciones3;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.view.Menu;
import android.view.View;
import android.widget.Button;

public class MainActivity extends FragmentActivity {

    private Button btnAlerta = null;

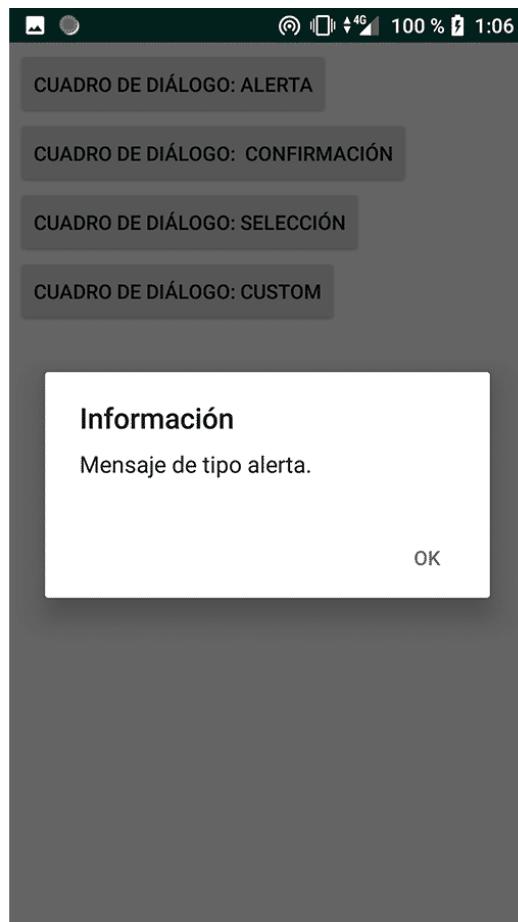
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnAlerta = (Button)findViewById(R.id.BtnAlerta);

        btnAlerta.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                FragmentManager fragmentManager = getSupportFragmentManager();
                DialogoAlerta dialogo = new DialogoAlerta();
                dialogo.show(fragmentManager, "tagAlerta");
            }
        });
    }
}
```

MainActivity.java

Obtendremos este resultado:



Diálogo de Confirmación

En un diálogo de confirmación solicitamos al usuario que nos confirme una determinada acción, con respuestas del tipo Sí/No.

Lo haremos igual que para el ejemplo de alerta, pero en esta ocasión añadiremos dos botones, uno para la respuesta afirmativa (`setPositiveButton()`), y otro para la respuesta negativa (`setNegativeButton()`).

```
package com.miejemplo.notificaciones3;

import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
import android.util.Log;

public class DialogoConfirmar extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {

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

        builder.setMessage("¿Quiere confirmar la acción?")
            .setTitle("Confirmacion")
            .setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    Log.i("Dialogos", "Aceptar.");
                    dialog.cancel();
                }
            })
            .setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    Log.i("Dialogos", "Cancelar.");
                    dialog.cancel();
                }
            });
    }

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

DialogoConfirmar.java

```
package com.miejemplo.notificaciones3;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.view.Menu;
import android.view.View;
import android.widget.Button;

public class MainActivity extends FragmentActivity {
```

```
private Button btnAlerta = null;
private Button btnConfirmacion = null;

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

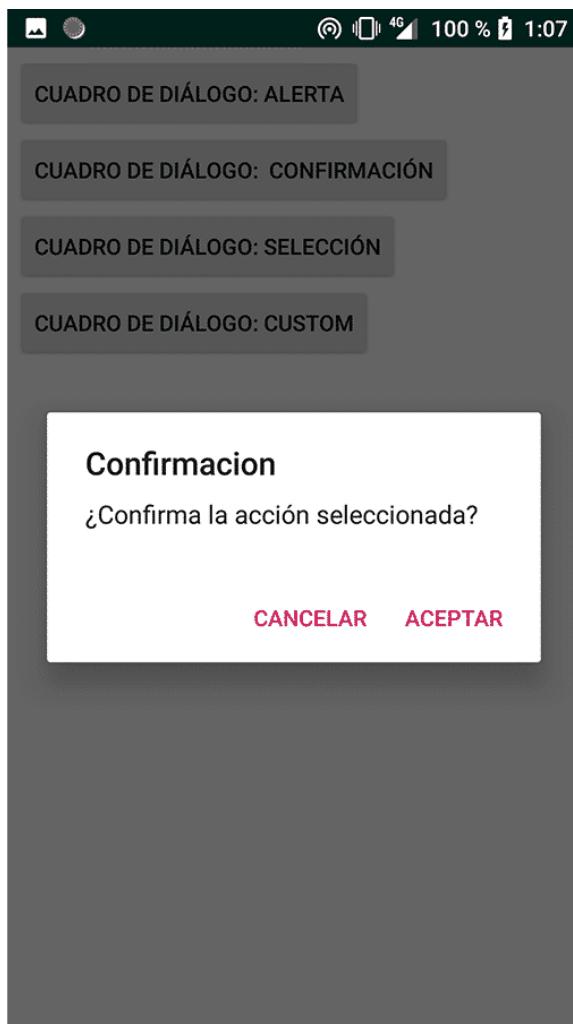
    btnAlerta = (Button)findViewById(R.id.BtnAlerta);
    btnConfirmacion = (Button)findViewById(R.id.BtnConfirmacion);

    btnAlerta.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            FragmentManager fragmentManager = getSupportFragmentManager();
            DialogoAlerta dialogo = new DialogoAlerta();
            dialogo.show(fragmentManager, "tagAlerta");
        }
    });
}

btnConfirmacion.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        FragmentManager fragmentManager = getSupportFragmentManager();
        DialogoConfirmacion dialogo = new DialogoConfirmar();
        dialogo.show(fragmentManager, "tagConfirmacion");
    }
});

}
```

Obtendremos este resultado:



Diálogo de Selección

Podemos utilizar los diálogos de selección para **mostrar una lista de opciones entre las que el usuario puede elegir**.

Para ello, utilizaremos la clase *AlertDialog*, pero esta vez indicaremos la lista de opciones a mostrar (mediante el método *setItems()*) y proporcionaremos la implementación del evento *onClick()* sobre dicha lista (mediante un *listener* de tipo *DialogInterface.OnClickListener*), evento en el que realizaremos las acciones oportunas según la opción elegida.

La lista de opciones la definiremos como un *array* tradicional.

```
package com.miejemplo.notificaciones3;

import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
import android.util.Log;

public class DialogoSeleccion extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {

        final String[] items = {"Español", "Inglés", "Francés"};

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

        builder.setTitle("Selección")
            .setItems(items, new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int item) {
                    Log.i("Dialogos", "Opción elegida: " + items[item]);
                }
            });
        return builder.create();
    }
}
```

DialogoSeleccion.java

```
package com.miejemplo.notificaciones3;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.view.Menu;
import android.view.View;
import android.widget.Button;

public class MainActivity extends FragmentActivity {

    private Button btnAlerta = null;
```

```
private Button btnConfirmacion = null;
private Button btnSeleccion = null;

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

    btnAlerta = (Button)findViewById(R.id.BtnAlerta);
    btnConfirmacion = (Button)findViewById(R.id.BtnConfirmacion);
    btnSeleccion = (Button)findViewById(R.id.BtnSeleccion);

    btnAlerta.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            FragmentManager fragmentManager = getSupportFragmentManager();
            DialogoAlerta dialogo = new DialogoAlerta();
            dialogo.show(fragmentManager, "tagAlerta");
        }
    });
}

btnConfirmacion.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        FragmentManager fragmentManager = getSupportFragmentManager();
        DialogoConfirmacion dialogo = new DialogoConfirmar();
        dialogo.show(fragmentManager, "tagConfirmacion");
    }
});
}

btnSeleccion.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        FragmentManager fragmentManager = getSupportFragmentManager();
        DialogoSeleccion dialogo = new DialogoSeleccion();
        dialogo.show(fragmentManager, "tagSeleccion");
    }
});
}
```

MainActivity.java

Obtendremos este resultado:



Diálogos personalizados

También podemos establecer completamente el aspecto de un cuadro de diálogo.

Lo hacemos a través de los **diálogos personalizados**.

Para ello, definiremos un *layout* XML con los elementos a mostrar en el diálogo.

1

En este caso, crearemos un *layout* llamado `dialog_personal.xml` en la carpeta `res/layout`. Contendrá, por ejemplo, una imagen a la izquierda y dos líneas de texto a la derecha.

Esta es la imagen que utilizaremos:



```
<?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="horizontal"
    android:padding="3dp" >

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/icon_android" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="3dp">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Frase 1" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Frase 2" />
    
```

</LinearLayout>

</LinearLayout>

dialog_personal.xml

2

Por su parte, en el método *onCreateDialog()* correspondiente utilizaremos el método *setView()* del *builder* para asociarle nuestro *layout* personalizado.

3

Además, podremos incluir **botones**, tal como vimos para los diálogos de alerta o confirmación.

```
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
import android.view.LayoutInflater;

public class DialogoCustom extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {

        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        LayoutInflater inflater = getActivity().getLayoutInflater();

        builder.setView(inflater.inflate(R.layout.dialog_personal, null))
            .setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    dialog.cancel();
                }
            });
        return builder.create();
    }
}
```

DialogoCustom.java

```
package com.miejemplo.notificaciones3;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.view.Menu;
import android.view.View;
import android.widget.Button;

public class MainActivity extends FragmentActivity {

    private Button btnAlerta = null;
    private Button btnConfirmacion = null;
    private Button btnSeleccion = null;
    private Button btnPersonalizado = null;

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

        btnAlerta = (Button) findViewById(R.id.BtnAlerta);
        btnConfirmacion = (Button) findViewById(R.id.BtnConfirmacion);
        btnSeleccion = (Button) findViewById(R.id.BtnSeleccion);
        btnPersonalizado = (Button) findViewById(R.id.BtnCustom);

        btnAlerta.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                FragmentManager fragmentManager = getSupportFragmentManager();
                DialogoAlerta dialogo = new DialogoAlerta();
                dialogo.show(fragmentManager, "tagAlerta");
            }
        });

        btnConfirmacion.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
```

```
FragmentManager fragmentManager = getSupportFragmentManager();
DialogoConfirmacion dialogo = new DialogoConfirmar();
dialogo.show(fragmentManager, "tagConfirmacion");
}

});

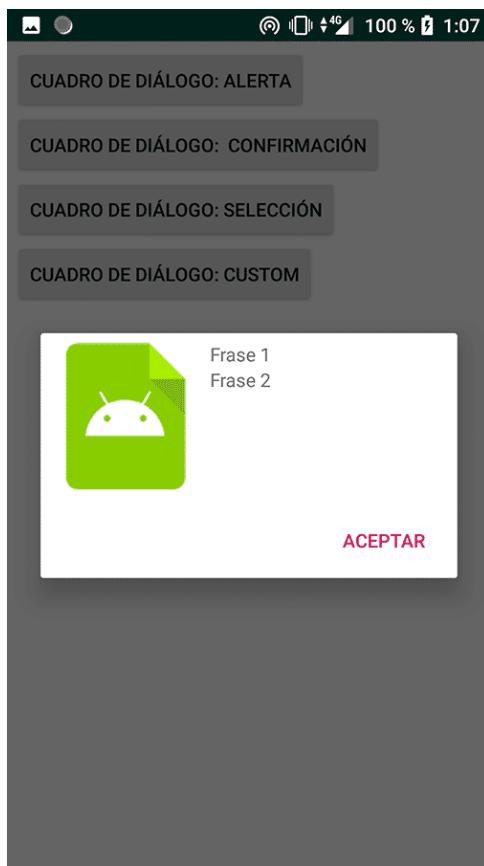
btnSeleccion.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        FragmentManager fragmentManager = getSupportFragmentManager();
        DialogoSeleccion dialogo = new DialogoSeleccion();
        dialogo.show(fragmentManager, "tagSeleccion");
    }
});

btnPersonalizado.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        FragmentManager fragmentManager = getSupportFragmentManager();
        DialogoPersonalizado dialogo = new DialogoCustom ();
        dialogo.show(fragmentManager, "tagPersonalizado");
    }
});

}
}
```

MainActivity.java

Obtendremos este resultado:



Resumen

Hemos terminado la lección, repasemos los puntos más importantes que hemos tratado.

- A lo largo de esta unidad hemos conocido la **clase Toast**, y hemos aprendido cómo utilizarla y cómo modificar su aspecto.
- También hemos aprendido cómo mostrar y personalizar notificaciones en la **barra de estado**.
- Hemos visto cómo mostrar **diálogos** al usuario, y hemos analizado los distintos tipos de diálogo que podemos mostrar: de **alerta**, de **confirmación** y de **selección**.
- Para finalizar, hemos personalizado el diálogo que mostramos al usuario mediante los **diálogos personalizados**.



PROEDUCA