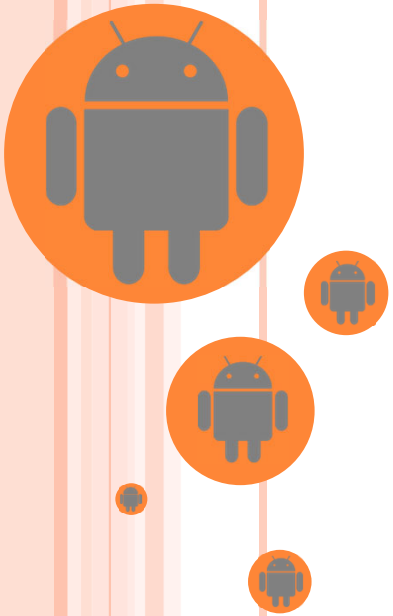
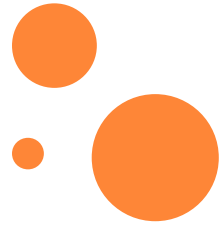
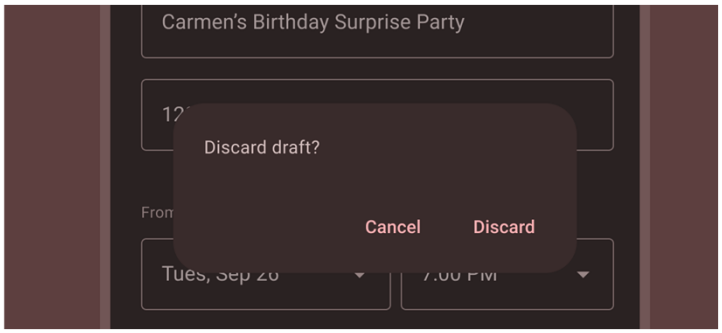


Dialogsy notificaciones



Dialogs y notificaciones

- Dialogs
  - Ventanas emergentes sobre nuestra aplicación
  - Permiten interactuar con el usuario



  - En este tema:
    - Veremos cómo crear diferentes tipos de Dialogs.
    - NO veremos heurísticos de diseño de Dialogs.
      - Consultar https://material.io/components/dialogs

Imágenes: https://developer.android.com/guide/topics/ui/dialogs

Dialogs y notificaciones

- Dialogs
  - Es RECOMENDABLE usar la clase *DialogFragment*
    - Paquete: *androidx.fragment.app.DialogFragment*
    - Gestiona el ciclo de vida automáticamente
    - Alternativa: utilizar *AlertDialog* directamente en la actividad
      - Controlar el giro, las llamadas, etc. depende del desarrollador
  - Se crea una clase nueva que extienda de *DialogFragment*

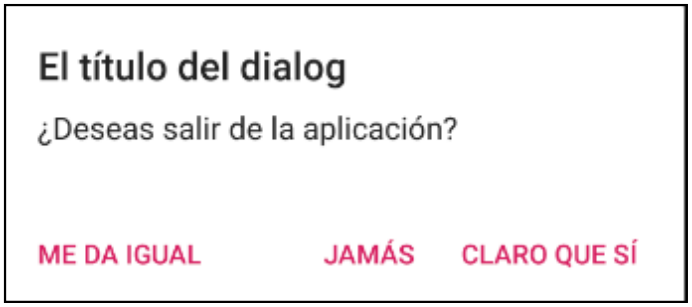
```
public class ClaseDialogo extends DialogFragment {
    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        super.onCreateDialog(savedInstanceState);
        …
        return builder.create();
    }
}
```

Configurar el diálogo usando la clase builder como veremos a continuación.

Devolver el método create del builder

Dialogs y notificaciones

- Dialogs
  - De tipo Alert
    - Mensaje al usuario




```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
builder.setTitle("El título del dialog");
builder.setMessage("¿Deseas salir de la aplicación?");
```

Contexto

    - Respuesta del tipo SI/NO/NEUTRAL
      - Se definen sólo las opciones que se deseen

Texto a mostrar en el "botón"

```
builder.setPositiveButton("Claro que sí", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        …
    }
});
```

Qué hacer si el usuario elige esta opción

Métodos *setNegativeButton* y *setNeutralButton* para el resto de opciones

Dialogs y notificaciones

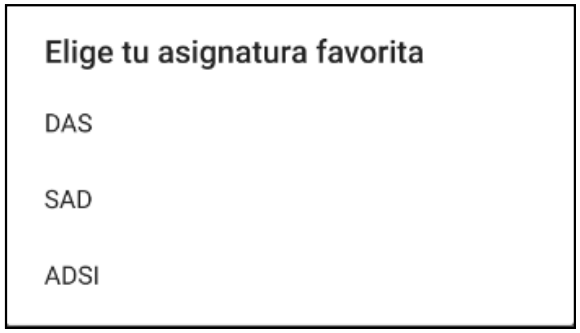
- Dialogs
  - Con listado de opciones
    - Se usa el método *setItems*

```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
builder.setTitle("Elige tu asignatura favorita");
CharSequence[] opciones = {"DAS", "SAD", "ADSI"};
builder.setItems(opciones, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        …
    }
});
```

Array con las opciones

Qué hacer al elegir una opción

La opción elegida viene marcada por el valor de i (su posición)



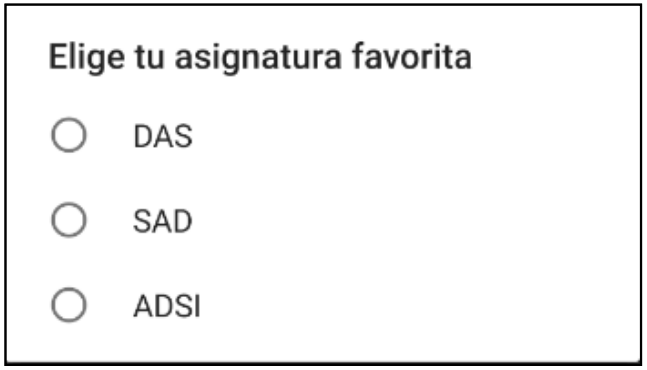
Se pueden incluir botones del tipo SI/NO/Neutral

Dialogs y notificaciones

- Dialogs
  - Con listado de opciones de tipo *radiobutton*
    - Se usa el método *setSingleChoiceItems*

```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
builder.setTitle("Elige tu asignatura favorita");
final CharSequence[] opciones = {"DAS", "SAD", "ADSI"};
builder.setSingleChoiceItems(opciones, -1, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        …
    }
});
```

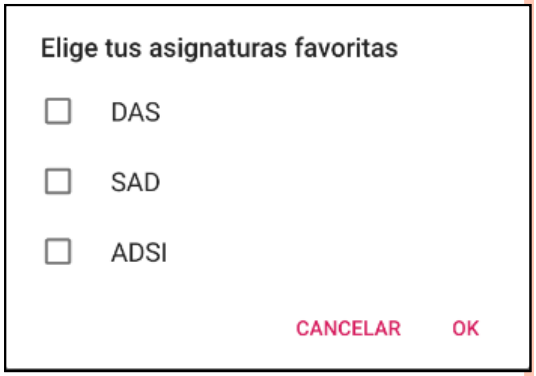
Opción marcada por defecto. Una posición que no exista indica ninguna



Se pueden incluir botones del tipo SI/NO/Neutral

DIALOGS Y NOTIFICACIONES

- Dialogs
  - Con listado de opciones de tipo *checkbox*
    - Se usa el método *setMultichoiceItems*
    - Hay que gestionar cuáles son las opciones marcadas




```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
builder.setTitle("Elige tus asignaturas favoritas");
final CharSequence[] opciones = {"DAS", "SAD", "ADSI"};
final ArrayList <Integer> loselegidos=new ArrayList<>();
builder.setMultiChoiceItems(opciones, null, new DialogInterface.OnMultiChoiceClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i, boolean b) {
        if (b == true){
          loselegidos.add(i);
        }
        else if (loselegidos.contains(i)){
            loselegidos.remove(Integer.valueOf(i));
        }
    }
});
```

Array para almacenar las opciones elegidas

Indica si está seleccionado o no

Array de booleanos con los seleccionados por defecto

Gestión de los elegidos

Se DEBERÍAN incluir botones del tipo SI/NO/Neutral

Si recibe un int elimina el contenido de esa posición. Si recibe un elemento del tipo de datos del ArrayList (Integer) elimina ese contenido del array

Dialogs y notificaciones

- Dialogs
  - Se puede incluir un diseño personalizado
    - Se define un fichero .xml con la interfaz

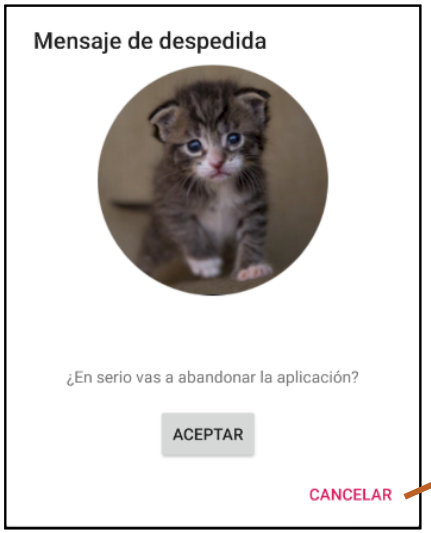
```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
builder.setTitle("Mensaje de despedida");
LayoutInflater inflater=getActivity().getLayoutInflater();
View elaspecto= inflater.inflate(R.layout.ellayoutcustomizado,null);
builder.setView(elaspecto);
TextView etiqueta= elaspecto.findViewById(R.id.textView);
etiqueta.setText("¿En serio vas a abandonar la aplicación?");
```

Contexto

Asignación del aspecto personalizado al dialog

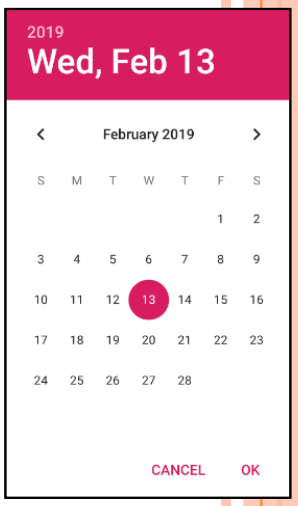
Acceso a los elementos de la interfaz personalizada

El XML con la interfaz



Se pueden incluir el resto de elementos de un dialog.

DIALOGS Y NOTIFICACIONES



- Dialogs
  - De tipo *DatePickerDialog* para seleccionar una fecha
    - Se devuelve directamente la instancia creada
    - La clase que extiende a *DialogFragment* debe implementar el método *DatePickerDialog.OnDateSetListener*

```
public class ClaseDialogoFecha extends DialogFragment implements DatePickerDialog.OnDateSetListener {
    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        super.onCreateDialog(savedInstanceState);
        Calendar calendario=Calendar.getInstance();
        int anyo=calendario.get(Calendar.YEAR);
        int mes=calendario.get(Calendar.MONTH);
        int dia=calendario.get(Calendar.DAY_OF_MONTH);
        DatePickerDialog eldialogo= new DatePickerDialog(getActivity(),this, anyo,mes,dia);

        return eldialogo;
    }

    @Override
    public void onDateSet(DatePicker datePicker, int i, int i1, int i2) {
        …
    }
}
```

Obtener la fecha actual

Contexto

Dónde está implementado el listener

Fecha que aparece marcada por defecto

Cuántos meses y cuántos días de desplazamiento

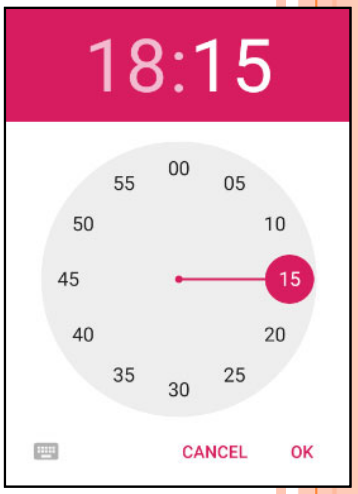
Qué hacer al elegir una fecha (Cuando el usuario pulsa OK)

Año

Mes (Valores de 0 a 11)

Día del mes

DIALOGS Y NOTIFICACIONES



- Dialogs
  - De tipo *TimePickerDialog* para seleccionar una hora
    - Se devuelve directamente la instancia creada
    - La clase que extiende a *DialogFragment* debe implementar el método *TimePickerDialog.OnTimeSetListener*

```
public class ClaseDialogoHora extends DialogFragment implements
TimePickerDialog.OnTimeSetListener {
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        super.onCreateDialog(savedInstanceState);
        Calendar calendario=Calendar.getInstance();
        int hora=calendario.get(Calendar.HOUR_OF_DAY);
        int minuto=calendario.get(Calendar.MINUTE);
        TimePickerDialog eldialogo= new TimePickerDialog(getActivity(),this, hora,minuto,
                                        DateFormat.is24HourFormat(getActivity()));

        return eldialogo;
    }

    @Override
    public void onTimeSet(TimePicker timePicker, int i, int i1) {
        …
    }
}
```

Obtener la hora actual

Contexto

Dónde está implementado el listener

Boolean para indicar si el formato es 24h. Así usa la preferencia del usuario en los ajustes de Android. Importar *android.text.format.DateFormat*

Hora seleccionada por defecto

Qué hacer al elegir una hora (Cuando el usuario pulsa OK)

Hora

Minuto

Dialogs y notificaciones

- Dialogs
  - Para usarlos, se crea una nueva instancia de la clase que define el diálogo y se llama al método *show()*

```
public class MainActivity extends AppCompatActivity {

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

        DialogFragment dialogoalerta= new ClaseDialogoAlert();
        dialogoalerta.show(getSupportFragmentManager(), "etiqueta");
    }
}
```

La clase del diálogo

Gestor de fragmentos

Etiqueta para la gestión del fragmento. Debe ser única

Dialogs y notificaciones

- Dialogs
  - En ocasiones, necesitaremos que las acciones se ejecuten en la actividad que llamó al dialogo
  - Para ello, la clase del diálogo definirá una interfaz con los métodos que nos interesen

```
public class ClaseDialogoAlert extends DialogFragment {
    ListenerdelDialogo miListener;

    public interface ListenerdelDialogo {
        void alpulsarSI();
        void alpulsarNO();
    }
    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        super.onCreateDialog(savedInstanceState);
        miListener =(ListenerdelDialogo) getActivity();
        …

}
```

Los métodos que interesa gestionar en la actividad

Se referencia a la implementación de la actividad

Dialogs y notificaciones

- Dialogs
  - La actividad que llame al diálogo deberá implementar la interfaz definida

```
public class MainActivity extends AppCompatActivity implements ClaseDialogoAlert.ListenerdelDialogo
{

    …
    @Override
    public void alpulsarSI() {
        …
    }

    @Override
    public void alpulsarNO() {
        …
    }
}
```

Implementa la interfaz definida en la clase deldíálogo

Implementación de los métodos de la interfaz

DIALOGS Y NOTIFICACIONES

- Dialogs
  - En los métodos de control de los botones (en la clase del diálogo) , usamos los métodos de la interfaz para que los métodos implementados se ejecuten en la actividad

```
builder.setPositiveButton("Claro que sí", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        miListener.alpulsarSI();
    }
});
builder.setNegativeButton("Jamás", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        miListener.alpulsarNO();
    }
});
```

Llamada al método implementado en la actividad

Dialogs y notificaciones

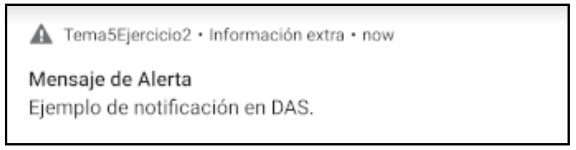
- Dialogs
  - El *dialog* se cierra automáticamente
    - Cuando el usuario "responde" (excepto en los listados con radiobuttons y checkboxes)
      - Método *onDismiss()*
    - Al pulsar el botón *Back* o fuera del *dialog*
      - Método *onCancel()*
      - Se puede evitar ese comportamiento
        - *setCancelable(boolean)*
        - *setCanceledOnTouchOutside(boolean)*
  - Se puede forzar el cierre
    - Método *dismiss()*
    - Método *cancel()*

Dialogs y notificaciones

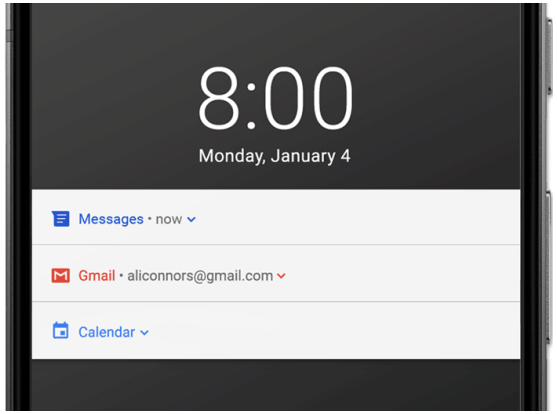
- Ejercicio 1: Cread una aplicación que muestre un diálogo para seleccionar una fecha y muestre la fecha seleccionada en un TextView de la actividad.
- Añadid un botón para cerrar la aplicación que solicite confirmación al usuario.
- Gestionad que el usuario no pueda salir de los diálogos sin seleccionar una fecha o responder a la pregunta.

Dialogs y notificaciones

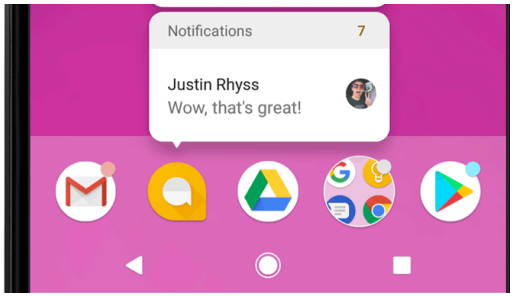
- Notificaciones locales
  - Se muestran en la barra de tareas



  - Notificaciones emergentes (heads-up)
  - Notificaciones en la pantalla de bloqueo



  - Insignias (punto de notificación) en el icono de la app



Dialogs y notificaciones

- Notificaciones locales
  - A partir de Android 8 (Oreo) se gestionan por canales
  - Cada aplicación puede definir distintos canales
    - Ejemplo (un juego):
      - Canal para notificaciones de eventos en el juego
      - Canal "lleva tiempo sin jugar"...
  - El usuario puede silenciar un canal y dejar activos el resto de canales
  - Desde Android 13, necesitan permisos en el manifiesto

```
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
```

DIALOGS Y NOTIFICACIONES

- Notificaciones locales
  - Una vez añadidos los permisos en el manifiesto, se solicitan los permisos

```
if (ContextCompat.checkSelfPermission(this, Manifest.permission.POST_NOTIFICATIONS)!=
        PackageManager.PERMISSION_GRANTED) {
          //PEDIR EL PERMISO
          ActivityCompat.requestPermissions(this, new
          String[]{Manifest.permission.POST_NOTIFICATIONS}, 11);
}
```

Es un código a definir por el desarrollador

Dialogs y notificaciones

- Notificaciones locales
  - Hay que crear un *NotificationManager* y un *Builder*
  - Si queremos que las notificaciones funcionen en Android 8 y superiores, también hay que crear un *NotificationChannel*

```
NotificationManager elManager = (NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
NotificationCompat.Builder elBuilder = new NotificationCompat.Builder(this, "IdCanal");


if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    NotificationChannel elCanal = new NotificationChannel("IdCanal", "NombreCanal",
                                  NotificationManager.IMPORTANCE_DEFAULT);
    …
    elManager.createNotificationChannel(elCanal);
}
```

El Manager y el Builder, siempre

El Canal sólo si es >=8 (Oreo)

Configurar el Canal

Unir el Canal al manager

Dialogs y notificaciones

- Notificaciones locales
  - Hay que crear un *NotificationManager* y un *Builder*
  - Si queremos que las notificaciones funcionen en Android 8 y superiores, también hay que crear un *NotificationChannel*

El Manager y el Builder, siempre

```
NotificationManager elManager = (NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
NotificationCompat.Builder elBuilder = new NotificationCompat.Builder(this, "IdCanal");


if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    NotificationChannel elCanal = new NotificationChannel("IdCanal", "NombreCanal",
                                  NotificationManager.IMPORTANCE_DEFAULT);
    …
    elManager.createNotificationChannel(elCanal);
}
```

El Canal sólo si es >=8 (Oreo)

Configurar el Canal

Unir el Canal al manager

Dialogs y notificaciones

- Notificaciones locales
  - Las características se definen a través del Builder

```
elBuilder.setSmallIcon(android.R.drawable.stat_sys_warning)
         .setContentTitle("Mensaje de Alerta")
        .setContentText("Ejemplo de notificación en DAS.")
        .setSubText("Información extra")
        .setVibrate(new long[]{0, 1000, 500, 1000})
        .setAutoCancel(true);
```

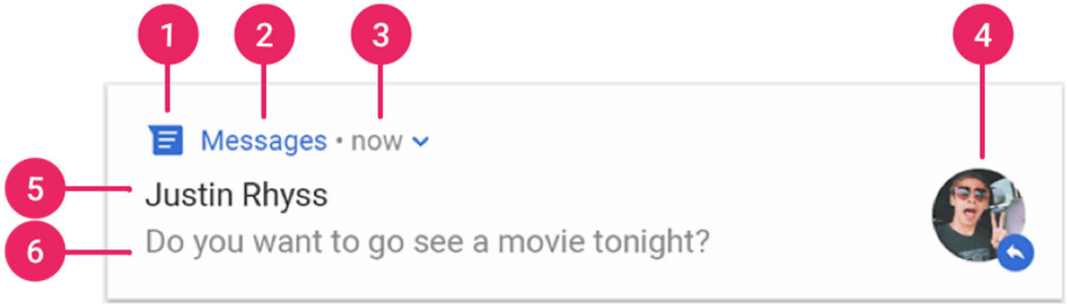
Icono en la barra

Vibración

Para que desaparezca una vez se haga clic sobre ella

Dialogs y notificaciones

- Diseño de las notificaciones locales



1) Ícono pequeño: obligatorio, se establece con `setSmallIcon()`.

2) Nombre de la app: proporcionado por el sistema.

3) Marca de tiempo: El sistema la proporciona, pero puedes anularla con `setWhen()` o bien ocultarla con `setShowWhen(false)`.

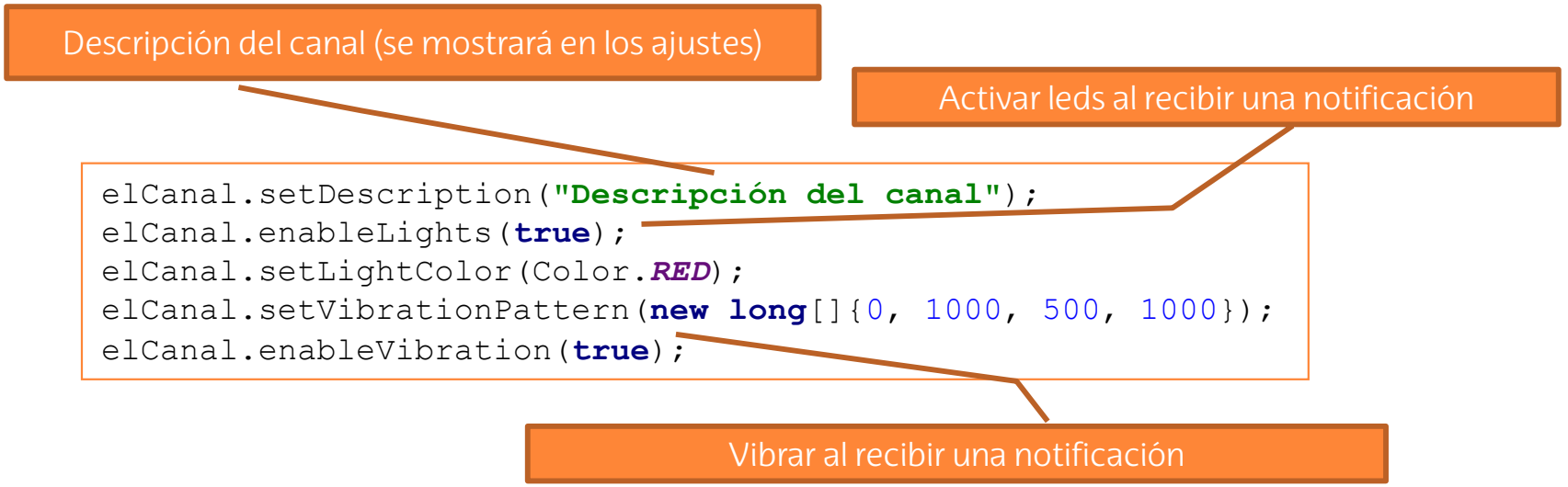
4) Ícono grande: Es opcional y, por lo general, se usa solo para fotos de contacto. No la uses para el ícono de tu app. Se establece con `setLargeIcon()`.

5) Título: Es opcional y se establece con `setContentTitle()`.

6) Texto: Es opcional y se establece con `setContentText()`.

Dialogs y notificaciones

- Notificaciones locales
  - La configuración del *NotificationChannel*



Dialogs y notificaciones

- Notificaciones locales
  - Lanzar la notificación

```
elManager.notify(1, elBuilder.build());
```

Identificador para la notificación

Dialogs y notificaciones

- Intents en notificaciones
  - Se puede dejar pendiente (para que se ejecute más tarde, aunque la aplicación no esté en ejecución)
  - Usados, p.ej., con las notificaciones en barra de tareas

```
Intent i = new Intent(this,SegundaActividad.class);
PendingIntent intentEnNot = PendingIntent.getActivity(this, 0, i, 0);
```

Código

FLAGS

Dialogs y notificaciones

- Intents en notificaciones

```
Intent i = new Intent(this,SegundaActividad.class);
PendingIntent intentEnNot = PendingIntent.getActivity(this, 0, i, 0);
```

FLAGS

  - Los FLAGS:
    - PendingIntent.FLAG_NO_CREATE : Si ya existe, no hace nada
    - PendingIntent.FLAG_UPDATE_CURRENT: Si ya existe, actualiza los datos que pueda haber en el intent
    - PendingIntent.FLAG_CANCEL_CURRENT: Si ya existe, elimina el anterior
    - PendingIntent.FLAG_ONE_SHOT: Sólo se puede lanzar el intent una vez
    - PendingIntent.IMMUTABLE: El intent no puede ser modificado por otras apps
    - PendingIntent.MUTABLE: El contenido del intent puede ser modificado por otras apps.

Dialogs y notificaciones

- Intents en notificaciones

```
Intent i = new Intent(this,SegundaActividad.class);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            intentEnNot = PendingIntent.getActivity(this,
                    0, i, PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);

        }else {
            intentEnNot = PendingIntent.getActivity(this,
                    0, i, PendingIntent.FLAG_UPDATE_CURRENT);
        }
```

FLAGS

- Los FLAGS:
  - Desde Android 12 en adelante, hay que indicar explícitamente si el PendingIntent es MUTABLE o IMMUTABLE

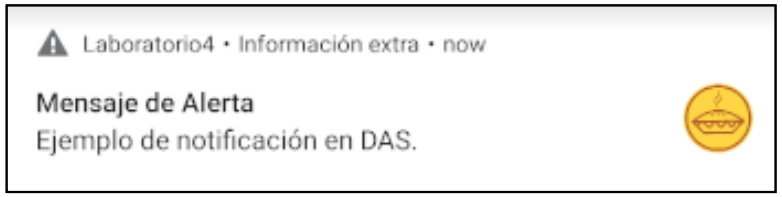
Dialogs y notificaciones

- Intents en notificaciones
  - Para asociar a una acción a una notificación

```
elBuilder.setLargeIcon(BitmapFactory.decodeResource(getResources(),R.drawable.tarta))
        .setSmallIcon(android.R.drawable.stat_sys_warning)
        .setContentTitle("Mensaje de Alerta")
        .setContentText("Ejemplo de notificación en DAS.")
        .setSubText("Información extra")
        .setVibrate(new long[]{0, 1000, 500, 1000})
        .setAutoCancel(true)
        .setContentIntent(intentEnNot);
```

Icono grande

El intent de la notificación



Dialogs y notificaciones

- Ejercicio 2: Añadid a la aplicación del ejercicio 1 que se lance una notificación cada vez que el usuario elija una fecha.

- Añadid un intent a la notificación para que al pulsarla se abra otra actividad de la aplicación.